O‘ZBEKISTON RESPUBLIKASI
OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI

MUHAMMAD AL-XORAZMIY NOMIDAGI
TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

Nuraliyev Faxriddin Murodillayevich

KOMPYUTER GRAFIKASI

5350200 – “Televizion texnologiyalar” (“Audiovizual texnologiyalar”, “Telesudia tizimlari va ilovalari”)”
5350200 – “Kompyuter injenering” (“Multimedia texnologiyalari”) bakalavriat ta’lim yo‘nalishlari talabalari uchun darslik

Toshkent - 2022

Muallif: t.f.d., dotsent Nuraliyev F.M.

Mazkur darslik 5350200 – “Televizion texnologiyalar”, 5330200 – “Informatika va axborot texnologiyalari”, “Kompyuter injenering” (“Multimedia texnologiyalari”) kabi bakalavr ta’lim yo‘nalishlarida o‘qitiladigan “Kompyuter grafikasi” o‘quv fani mazmuni asosida tuzilgan.

Darslikning maqsadi kompyuter grafikasi asoslari: geometrik almashtirishlar, geometrik proeksiyalash, rastr grafikasi, rang va yorug‘lik va hokazolar haqida tushuncha berish. Bir so‘z bilan aytganda realistik tasvirlarni yaratish va ularni harakatini bajarish. Shuningdek geometrik almashtirishlar, geometrik proeksiyalash, rastr grafikasi, rang va yorug‘lik bilan ishlashni OpenGL grafik kutubxonasi yordamida amalga oshirish. Bunda OpenGL grafik kutubxonasi buyruqlarini asosiy sintaksisi va ularni ishlatish yordamida real tasvirlarni yaratish keltirib o‘tilgan.

Mazkur darslik oliy o‘quv yurtlari talabalari uchun mo‘ljallangan bo‘lib, undan ilmiy tadqiqotchilar, tegishli texnikumlar o‘quvchilari va barcha qiziquvchular foydalanishlari mumkin.

Taqrizchilar:

Texnika fanlari doktori, k.i.h. dotsent N.Mirzayev

Texnika fanlari doktori, dotsent S.S.Beknazarova

KIRISH

XXI asr axborot haqiqatan ham texnoloyiyalari asri ekanligini tan olish joiz, chunki biz yashab turgan ushbu davrda axborotlar oqimi shu darajada jadal rivojlanib bormoqda-ki, bu jarayonni ko‘rmaslik mumkin emas va har birimiz usbu jarayonning ma’lim ma’noda ishtirokchisiga aylanib ulgurdik.

Zamon talablari va sanoat ehtiyojlaridan kelib chiqgan holda “Kompyuter grafikasi va dizayn” fani har bir soha bilan uzviy bog‘lanib, unga bo‘lgan ehtiyoj tobora o‘shib borayotganligini kuzatish qiyin emas.

Ma’lumki axborot almashinuvida insonning ko‘rish sezgi organi yordamida qabul qilingan axborot eng samarali qabul qilinadi va u xotirada ham chuqur iz qoldiradi. Jumladan tovush vositasida berilgan axborot ham ijobiy ta’sir etadi. Ammo axborot almashinuvi nafaqat so‘zlar va tovushlar, balki tasvirlar, ranglar va shakllar bilan ham amalgam oshiriladi. Buning yorqin dalili sifatida turli xil kitoblar, daftar va jurnallar muqovalari, ko‘chalar yoqasida va binolar peshtoqida ilingan reklamalar, ommaviy axborot vositasi bo‘lgan televideniya orqali uzatilayotgan turli xildagi kinolar, kliplar va boshqa ijtimoiy-madaniy ko‘rsatuvlar, gazeta va internet orqali berilayotgan manbalarning naqadar did bilan ishlanganligi, uyali aloqa vositalarining platformalarida ham ko‘rishimiz mumkin. Albatta ushbu ishlar zamirida yurtimiz iqtisodiyotini ichki va tashqi bozorda yanada mustahkamlash va xalq farovonligi ta’minlash uchun o‘zining intellektual qobiliyatlarini namoyon etadigan yuksak malakali mutaxassis kadrlar tayyorlash kabi vazifalarga bog‘liq ravishda oliy ta’lim muassasasining ilmiy salohiyati va moddiy-texnik ta’minlanganligi muhim ahamiyat kasb etadi. O‘zbekiston Respublikasi Prezidentining 2022 yil 28 yanvardagi “2022-2026 yillarga mo‘ljallangan Yangi O‘zbekistonning taraqqiyot strategiyasi to‘g‘risida” gi PF-60 sonli farmonida oliy talim bilan qamrov darajasini 50% ga oshirirsh va ta’lim sifatini oshirish va 2026 yilga qadar 10 ta salohiyatli oliy ta’lim muassasini QS va THE xalqaro reytinglariga kirishga madsaqli tayyorlash ko‘zda tutilgan. Mamlakatimiz oliy ta’lim tizimida raqobatbardosh kadrlarni tayyorlash va iqtisodiyotning rivojlanishida munosib hissa qo‘shishiga katta e’tibor qaratilmoqda.

O‘zbekiston Respublikasi Prezidentining 2017 yil 7 fevraldagi PF-4947-son “O‘zbekiston Respublikasini yanada rivojlantirish bo‘yicha Harakatlar strategiyasi to‘g‘risida”gi Farmoni, 2017 yil 20 apreldagi PQ-2909-son “Oliy ta’lim tizimini yanada rivojlantirish chora-tadbirlari to‘g‘risida” qarori, 2018 yil 5 iyundagi PQ-3775-son “Oliy ta’lim muassasalarida ta’lim sifatini oshirish va ularning mamlakatda amalga oshirilayotgan keng qamrovliislohotlarda faol ishtirokini ta’minlash bo‘yicha qo‘chimcha chora-tadbirlar to‘g‘risida” qarori, 2019 yil oktabrdagi PF-5847-son “O‘zbekiston Respublikasi oliy ta’lim tizimini 2030 yilgacha rivojlantirish konsepsiyasini tasdiqlash to‘g‘risida”gi farmoni hamda mazkur faoliyati tegishli boshqa me’yoriy-huquqiy hujjatlarda belgilangan vazifalarni amalga oshirishda ushbu dissertasiya tadqiqoti muayyan darajada xizmat qiladi. Jumladan o‘quv va ilmiy laboratoriyalarni lingafon kabinetlari hamda ulardagi ilmiy asbob-uskunlari, jihozlari zamon talabiga mos ravishda yangilanishini jadallashtirish, fanning eng ilg‘or yutuqlari bilan boyitilgan o‘quv adabiyotlari, zamonaviy kompyuter texnologiyalarining texnik va dasturiy vositalari bilan ta’minlash, axborot resurs markazlarining avtomatlashtirilishi va Internet tarmog‘iga chiqish imkoniyatini yaratish kabi vazifalar belgilangan. Hozirgi kunda ushbu vazifalarga bog‘liq ravishda respublikada zamonaviy axborot-kommunikasiya texnologiyalari sohasida yangi o‘quv adabiyotlarni yaratish, axborot resurs markazlariga joylashtirish va ulardan samarali foydalanishni rivojlantirishga alohida e‘tibor qaratilayotganini ko‘rish mumkin.

Ushbu vazifalarga bog‘liq ravishda mazkur o‘quv qo‘llanma yurtimiz oliy ta’lim tizimidagi bakalavr bosqichida o‘qitiladigan “Kompyuter grafikasi va dizayn” o‘quv fani mazmunini yoritishga bag‘ishlangan. Kompyuter grafikasi va dizaynining qo‘llanish ko’lami juda ham keng bo‘lib, avvalom bor ushbu sohani vizualligi va dizayni diqqatga sazovordir. Grafik dizaynda o‘lcham, shakl, rang teksturasi, kompozitsiya, ko‘chirish va shriftlar muhim ahamiyatga ega. Berilgan topshriqni mavjud grafik dasturlarda bajarish va kerakli natijaga erishish uchun shakllar, shriftlar va ularning o‘lchamlari bilan ishlash, ularga rang berishda rang modellari va tekstura haqida tasavvurga ega bo‘lish, kompozitsiyani bilish, tasvirni kompyuter

ekraniga chiqarish va u bilan bog‘liq amallarni bajarish foydalanuvchidan ma’lum darajada geometrik bilimlarni talab etadi. Jumladan, obyektlarni tekislikda va fazoda almashtirish, proeksiyalash, fazoda tasvirlash, ko‘rinmas chiziq va sirtlarni olib tashlash, bo‘yash, nurning yo‘nalishini kuzatish, rang modellari haqidagi ma’lumotlar qo‘llanmada o‘z aksini topgan.

Keltirilgan nazariy ma’lumotlar asosida real obyektlarni yaratish ikki va uch o‘lchovli grafika sohasida ilovalar yaratish uchun keng tarqalgan amaliy dasturiy interfeyslardan biri hisoblangan OpenGL muhitida qarab chiqilgan. OpenGL da axborotlar birligi uchlar hisoblanadi va ular yordamida murakkab obyektlar quriladi. Dasturchi uchlarni yaratadi va ularni qanday birlashtirish (chiziqlar yoki ko‘pburchaklar) kerakligini ko‘rsatadi, kamera va chiroq koordinatalari va parametrlarini o‘rnatadi, OpenGL esa ekranda tasvirni yaratish ishi bilan shug‘ullanadi. OpenGL dasturchilar uchun katta bo‘lmagan uch o‘lchovli sahnani qurishda juda ham qo‘l keladi va uch o‘lchovli grafika algoritmlarini amalga oshirish tafsilotlari haqida o‘ylashga hojat yo‘q. Uch o‘lchovli dasturlashtirish bilan shug‘ullanuvchi professionallar uchun ham kutubxona foydali, chunki u asosiy mexanizmlarni namoyon etadi va belgilangan avtomatlashuvni bajaradi. OpenGL apparatga bog‘liq bo‘lmagan kutubxona hisoblanadi. OpenGL dan foydalanib uch o‘lchovli sahnani osongina yaratish, unga teksturalar qo‘yish, yorug‘lik manbalari bilan yoritish, shaffoflik, tuman effektini berish, ranglarni aralashtirish, shuningdek trafaret joylashtirish, sahna obyektlarini, kameralar va chiroqlarni belgilangan trayektoriya bo‘yicha harakatlantirish, aynan animatsiya tayyorlash mumkin.

I BOB. KOMPYUTER GRAFIKASI NAZARIYASI

1.1. Kompyuter grafikasi faniga kirish.

Tasvirlarni tanlash, qayta ishlash va kompyuter grafikasi. Rastr, vektor va fraktal grafikasi. Kompyuter grafikasining rivojlanish tarixi. Kompyuter grafikasining texnik vositalari: kiritish qurilmalari, displey printerlar, plotterlar, skanerlar. Kompyuter grafikasining dasturiy vositalari: qurilma drayverlari, grafik dasturlar, maxsus grafik tizimlar va dasturiy paketlar

Tasvirlarni tanlash, qayta ishlash va kompyuter grafikasi.

Kompyuter monitoridagi tasvir (rasm) bilan bog‘liq bo‘lgan axborotni qayta ishlashda uchta asosiy yo‘nalishga ajratishadi: tasvirni o‘rnatish yoki aniqlash, tasvirni qayta ishlash va kompyuter (mashina) grafikasi [1-20].

Tasvirni o‘rganishning asosiy vazifasi bu mavjud bo‘lgan obrazni (tasvirni) formal, tushunarli (aniq) bo‘lgan belgilar tiliga o‘tkazish. Bu holda qaralayotgan tasvir abstrakt tassavurga aynaltiriladi, ya’ni sonlarga, maxsus belgilar yoki graflar to‘plamiga o‘tkaziladi. Buni quyidagicha yozish mumkin:

COMPUTER VISION:

- input- tasvir (rasm);
- output-belgi (matn) va uning tahlili.

Tasvirni qayta ishlashda kiruvchi va chiquvchi ma’lumotlar-tasvirlar. Masalan: tasvirdagi ayrim elementlarni olib tashlash (ovoz, rang, …) ёki qo‘shish, uning hajmini o‘zgartirish va hakazo. Ya’ni uni quyidagicha ёzish mumkin:

IMAGE PROCESSING

- input – tasvir (o‘zgartirilmagan);
- output - tasvir (o‘zgartirilgan).

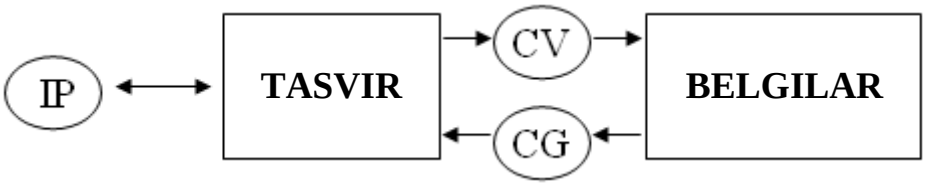
Kompyuter (mashina) grafikasi dastlabki, ya’ni kiruvchi axborotni (noma'lum tabiatga ega) tasvir ko‘rinishiga olib keladi. Masalan: ekspert ma’lumotlarni grafik, diagramma yoki boshqa shakllarga vizuallashtirish. Bundan tashqari shakllarni

almashtirish, harakatlantirish, virtual tasavvurga yaqinlashtirish. Kompyuter grafikasini quyidagicha tasvirlash mumkin:

COMPUTER GRAPHICS

- input – belgilar, belgili tavsif;
- output – tasvir.

Ularning o‘rtasida keskin chegara yo‘q va umumiy sxemada quyidagicha tasvirlash mumkin:



Tasvirni (dastlab matn, formula so‘ng oddiy rasm) shaxsiy kompyuter ekranida chiqarish kompyuter grafikasining rivojlanishida birinchi qadam bo‘ldi. Qisqa vaqt (50-yillardan boshlab) ichida kompyuter grafikasi tezkor rivojlandi va o‘zining e'tiborini ikki asosiy yo‘nalishga qaratdi: tasvirga etarlicha tasavvur (reallik) va harakat (dinamika) berish, va ularni birlashtirish.

Tasvirni kompyuter ekraniga chiqarish va u bilan bog‘liq amallarni bajarish foydalanuvchidan ma'lum darajada geometrik bilimlarni talab qiladi. Geometrik tushunchalar, formulalar, faktlar, (birinchi navbatda ikki va uch o‘lchovga tegishli) kompyuter grafikasida o‘ziga xos maxsus o‘rinni egallaydi. Geometrik yondashish, tasavvur va fikrlar hisoblash texnikasining imkoniyatlarini doimo tezkor kengayishi bilan birgalikda kompyuter grafikasining jiddiy rivojlanishi yo‘lida va ko‘p sohalarda keng ishlatilishiga manba bo‘ldi. Ayrim hollarda oddiy, elementar geometrik metodikalar katta geometrik masalalarni echish bosqichlarida rivojlanishni sezilarli darajada ta'minlaydi.

Kompyuter grafikasining rivojlanish tarixi. Kompyuter grafikasining texnik vositalari: kiritish qurilmalari, displey printerlar, plotterlar, skanerlar.

Monitorlar va LSD displeylar

Monitorlar axborotni aks ettirishning juda muhim qurilmalaridir. Katta sondagi videostandartlar bo‘lgani kabi hozirgi vaqtda mavjud monitorlar tiplari ham rang-barangligi bilan ajralib turadi.

Raqamli (TTL) monitorlar.

TTL inglizcha Transistor Transistor Manic atamasining qisqartmasi bo‘lib, o‘zbekchada tranzistor-tranzistor manikasini anglatadi. TTL atamasi elektron texnikasida qo‘llaniladigan raqamli mikrosxemalar standart talqinini anglatadi. Har doimgidek, raqamli texnikada, signallar faqat ikkita:

Mantiqiy “1”va “0” holatga ega.

Monoxromli monitorlar (MDA).

TTL-monitorlar haqida so‘z borganida, ko‘pincha monoxromli monitorlar ko‘zda tutilib, ularni boshqarish signallari MDA yoki Hercules kartalari tomonidan shakllantiriladi. Monoxromli deganidan tushunarliki, ekrandagi nuqta faqat yorqin yoki qora rangli bo‘lishi mumkin. Yaxshi holda nuqtalar yana o‘zlarining intensivliklari bilan farqlanib turishi lozim.

Raqamli RGB-monitorlari (CGA, EGA, VGA).

RGB-raqamli monitorlar (Red/Green/Blue – qizil/yashil/ko‘k) asosan EGA standartdagi kartaga ulanish uchun mo‘ljallangan. Bunday tuzilmalar 16 ta raqamlarni gradatsiyasini aks etishga imkon beradigan monoxrom rejimda ham ishlaydilar. Shaxsiy kompyuterlar paydo bo‘lganidan beri, bir necha standartlar o‘zgaradi: MDA (monoxrom), CGA (4 ta rang), EGA (16 ta rang), VGA (256 ta rang). Hozirgi kunda qator standart qiymatlardan ekran kengaytmasini to‘g‘ri tanlash imkoniyatiga ega bo‘lgan, 16,7 mln ranglarni tasvirlay oladigan SVGA monitorlari ishlatilmoqda.

Monitorlarning ishlash tamoyillari.

Rastr shakllanishi

Start Elektron – nurli trubka asosidagi monitorni ishlash tamoyili oddiy televizor ishlash tamoyilidan ozgina farq qiladi (8.1 rasm). Katod (elektron to‘plari) orqali chiqariladigan elektronlar to‘plami lyuminofor bilan qoplangan ekranga tushib nurlanish hosil qiladi. Elektronlar to‘plami yo‘lida odatda qo‘shimcha elektrodlar

joylashgan bo‘ladi: elektronlar to‘plami tezligini va u bilan bog‘liq tasvir yorug‘ligini to‘g‘rilovchi modulyator va to‘plam yo‘nalishini o‘zgartirish imkoniga ega bo‘lgan, og‘diruvchi tizim. Kompyuter monitori (hamda televizor) ekranida har qanday matnli yoki grafik tasvir piksel deb ataluvchi tasvir (rastr)ining minimal elementini ifodalovchi lyuminofor ning ko‘plab diskret nuqtalaridan iboratligiga e’tibor beramiz. Bunday monitorlar rastrli deb ataladi. Bu holatda elektron nur butun ekranni doim skanerlab turadi va yoymaning yaqin joylashgan qatorlarini hosil qiladi. Modulyatorga uzatiladigan videosignal qatorlari bo‘yicha nurni harakat qilishi yorug‘lik ravshanligini o‘zgartiradi va ba’zi bir ko‘rinadigan tasvirni tashkil etadi. Monitor qobiliyati gorizontal va vertikal bo‘yicha aks eta oladigan tasvirlarning elementlari soni bilan aniqlanadi, masalan: 640x480 yoki 1024x768 piksellar. Rastrni shakllantirish uchun monitorda maxsus signallar ishlatiladi. Skanerlash davrida nur siniq chiziq tarzidagi traektoriya bo‘yicha chap yuqori burchakdan o‘ng quyi burchakkacha harakat qiladi. Gorizontal bo‘yicha nurning to‘g‘ri yurishi qatorli yoyma signali bilan, vertikal bo‘yicha esa kadrli yoyma signali bilan amalga oshiriladi. Nur qatorni chetdagi o‘ng nuqtasidan, keyingi qator chetdagi o‘ng nuqtasidan, keyingi qator chetdagi chap nuqtasiga (nurni gorizontal bo‘yicha qayta yurishi) o‘tishi va ekranni oxirgi qatorini chetdagi o‘ng holatida birinchi qator chetdagi chap holatiga (nurni vertikal bo‘yicha qayta yurishi) o‘tishi maxsus qayta yurish signallari bilan amalga oshiriladi. Shunday qilib, monitor uchun eng muhimi quyidagi parametrlar hisoblanadi: yoymaning (kadrli) chastotasi, yoymaning gorizontal(qatorli) chastotasi va videosignalni o‘tkazish chizig‘i.

Rangli tasvir shakllanishi

Rangli monitorda rastrni shakllanish tamoyili monoxromnikiga o‘xshaydi. Biroq rangli tasvirni shakllanish usuli asosiga rangli ko‘rishni boshqa muhim xususiyatlari kiritilgan bo‘lib, masalan, bulardan biri: rangli o‘zlashtirishni uch komponentliligidir. Bu barcha ranglar uchta yorug‘lik oqimlarini, masalan, qizil, ko‘k va yashilni qo‘shish (aralashtirish) yo‘li bilan olish mumkinligini bildiradi. Bu esa rangli televizorlarda va monitorlarda ranglarini additiv aralashtirish usulini qo‘llash imkonini berdi. Ushbu usulni ekranga uchta asosiy ranglarni ekran yuzini qoplash

sharti bilan bir vaqtda uzluksiz tasvirlash yo‘li bilan ko‘rsatish mumkin. Uchta asosiy ranglarni birgalikda ishlatib, rangni qabul qilishni uch komponentli nazariyasiga muvofiq rang tuslarini kerakli gammasini olish imkoni tug‘ildi.

Rangli fazoviy o‘rtalatish. Agar rangli tasvirda rangli detallar yaqin joylashgan bo‘lsa, unda uzoq masofadan biz alohida detallarni rangini farqlay olamiz. Butun guruh ranglarni aralashtirish qonuniga muvofiq bir rangga bo‘yalgan bo‘ladi. Tasvirning bu xususiyati monitorni elektron – nurli trubkasida lyuminofor donachalari yonida joylashgan uchta rang yordamida tasvir bitta elementining rangini shakllanish imkonini beradi. Odamni ko‘rish qobililayi xususiyatlariga muvofiq rangli monitorning elektron – nurli trubkasida alohida boshqarish sxemalariga ega bo‘lgan uchta elektron to‘plari lari mavjud. Ekranni ichki tomoniga esa uchta asosiy ranglar lyuminofori: qizil - R (Red), ko‘k – B (Blue) va yashil – G (Gren) o‘rnatilgan. Shu tarzda to‘plari lyuminoforni faqat o‘zini rangiga “otishi” kerak. Buning uchun har bir rangli kineskopda soyaviy maska yoki apertuar panjara mavjud. Ular to‘plarini nurlari faqat muvaffaq rangdagi lyuminofor nuqtasiga tushishi uchun xizmat qiladi. Soyaviy maska kineskopni ichki yuziga qo‘yilgan nuqtalarga muvofiq teshiklar tizimiga ega bo‘lgan invar nomli (bu material juda kichik kengayishli koeffiseyentga ega) maxsus materialdan bo‘lgan metall plastinalardan yasalgan.

Apertuar panjara soyaviy maskadagi teshiklar bajaradigan funksiyani bajaruvchi tirqichlar tizimidan tashkil topgan. Talab etiladigan yechimni hal qilish uchun zarur bo‘lgan lyuminofor nuqtalarini miqdori ekran hajmiga bog‘liq.

Nuqtalarni qancha ko‘p joylashtirish kerak bo‘lsa va ekran qancha kichik bo‘lsa, nuqtalarni shuncha zich joylashtirishga to‘g‘ri keladi.

Ekranni ichki yuzida lyuminofor nuqtalarini kattaligi qancha kichik bo‘lsa, monitorda tasvirni tiniqligi shuncha yuqori bo‘ladi. Bu nuqtalar o‘lchovi, aniqrog‘i ular orasidagi o‘rtacha masofasi donachalar deb ataladi. Monitorlarni turli rusumlarida bu o‘lchov 0,25 dan 0,41 mm gacha bo‘lgan diapazonda joylashgan. Biroq yaxshi monitorlar uchun bu diapazon ancha qisqaradi - 0,28 mm gacha.

Monitorlar tavsifnomasi

Monitor ekranining diagonali deb ekranni chap quyi va o‘ng yo‘qori burchaklar orasidagi masofaga aytamiz. Monitor ekrani o‘lchovi 14 – dyuymli (36 sm), 15 – dyuymli (39 sm), 17 – dyuymli (44 sm), 19 – dyuymli (49 sm) va 21 – Apertuar maska

Regeneratsiya chastotasi.

Tasvirni regeneratsiyalash (yangilash) chastotasi, monitordagi tasvirni soniya davomida necha marta to‘liq o‘zgartira olishini ko‘rsatadi (shuning uchun kadrlar chastotasi deb ham atashadi). Bu o‘lchov nafaqat monitorga, balki videoadapter sozligi va xususiyatiga ham bog‘liq bo‘ladi. Ammo baribir oxirgi imkoniyatini monitor aniqlab beradi. Regeneratsiya chastotasi gersda (Gts) o‘lchanadi. U qancha yo‘qori bo‘lsa, tasvir shuncha aniq va mustahkam bo‘ladi. Ko‘z kamroq charchaydi, kompyutyerdan uzoqroq ishlash imkoni bo‘ladi. Regenerasiya chastotasi 60 Gts bo‘lganda, tasvirni pirillashi darrov ko‘rinadi. Bugungi kunda bunday narsaga yo‘l qo‘yilmaydi. Minimal 75 Gts, normativ – 85 Gts va – 100 Gts va undan ortig‘i qulay hisoblanadi.

Ekran kattaligi.

Ekran kattaligi parametrlarni eng muhimi hisoblanadi. U qancha yuqori bo‘lsa, ekranda shuncha ko‘p ma’lumotlar aks ettirsa bo‘ladi, ammo har bir alohida nuqtaning o‘lchami shuncha kam bo‘ladi. Kichik hajmdagi monitorda oshirilgan kattalikni qo‘llash, tasvir elementlari noaniq bo‘lishi sababli hujjat va dasturlar bilan ishlashda ko‘rish organlarining toliqishiga olib keladi. Kattaliklarni pasaytirish tasvirni kattalashib ketishiga olib keladi. Ammo ekranga ular sig‘maydi. Shunday qilib, har bir monitor o‘lchovi uchun, videoadapter ta’minlab beruvchi, ekranning o‘z kattaliklari mavjud

Jadval

Monitor o‘lchovi	**Ekanni optimal kattaligi**
14 dyuym	640x480
15 dyuym	800x600

17 dyuym	1024x768
19, 21 dyuym	1280x1024
26 dyuym	

Rang kattaligi (rang to‘qligi)

Rang kattaligi ekranni har bir nuqtasi qabul qila oladigan turli xil tuslarni sonini aniqlaydi. Maksimal kiritilgan rang kattaliklar videoadapter xususiyatiga va birinchi navbatda undagi o‘rnatilgan videoxotira soniga bog‘liq bo‘ladi. Undan tashqari, u ekranning belgilangan kattaligiga ham bog‘liq bo‘ladi. Ekranning yuqori kattaligida, tasvirning har bir nuqtasiga videoxotirada kamroq joy ajratishga to‘g‘ri keladi.

Berilgan ekran kattaligiga va rang to‘qligiga qarab, quyidagi formula orqali videoxotira hajmini aniqlash mumkin:

R – videoadapter xotirasini zarur hajmi;

m – ekran (nuqtalar) gorizontal kattaligi;

n – ekran (nuqtalar) vertikal kattaligi;

b – rangni kodlash razryadi;

bugungi kunda ranglar to‘qligining asosiy talabi – 256 ta rang. Ko‘p dasturlar 65 ming rangni talab qiladi (High Color tartibida).

Ekran maskasi.

Tasvir sifati soyaviy maskani ishlatilishini turiga va tavsifnomasiga bog‘liq. Bu yerda maska oralig‘idagi masofani o‘lchaydigan o‘lchov birligi sifatida millimetr ishlatiladi. 14 dyuymli 0,28 mm maskali monitor uchun taxminan 600000 teshik bo‘ladi. Teshiklar orasida masofa qancha kichik bo‘lsa, teshiklar qancha ko‘p bo‘lsa, tasvir sifati shuncha yuqori bo‘ladi.

Muhofaza sinfi. Monitorni muhofaza sinfi xavfsizlik texnikasi talablari nuqtai nazaridan monitor muvoffiqlik standarti bilan aniqlanadi. Hozirgi kunda quyidagi halqaro standartlar qabul qilingan: MPR-II, TCO-92, TCO-95, TCO-99. MPR-II standarti inson uchun xavfsiz bo‘lgan elektromagnit nurlari darajasini cheklaydi. TSO-92 standartida bu me’yorlar saqlanib qoldi, TSO-95 va TSO-99 standartlarida bu me’yoarlar qat’iylashgan. Ergonomik va ekomanik me’yorlar

birinchi bo‘lib TSO-95 da paydo bo‘lgan. TSO-99 standarti esa tasvir sifatini (yorug‘lik aks etishi, jimillashi, qobiqni dog‘larga qarshi xususiyati) aniqlovchi parametrlarni qat’iy me’yorlarini kiritdi

Dog‘ga qarshi qobiqlar.

Hamma monitorlar dog‘ga qarshi qobiqqa ega bo‘lishi kerak. Ekran yuzida bunday xususiyat bo‘lmasa, monitorda ma’lumotlarni o‘qish qiyin bo‘ladi. Ekran yuzi cheklansa, qum zarralari bo‘lgan pistolet yordamida nazoratlanadi. Bunday usul faqat arzon monitorlar uchun tavsiya etiladi. Uning kamchiligi shundaki, ekranda grafika va rasmlar aniq bo‘lmaydi, tasvir yoyilgan bo‘ladi. Kineskopni qoplashning eng yaxshi yo‘li - dog‘ga qarshi qatlam bilan qoplash. Bunday holatda elektron nurli trubkani ekran yuziga kimyoviy modda surtiladi, buning natijasida yorug‘lik uning yuzida aks etolmaydi.

Suyuq kristalli displeylar

Ba’zi bir kompaniyalar yassi indekatsion panellarni ishlab chiqaruvchilardan texnologiyani o‘rganib, suyuq kristalli displeylar ishlab chiqishdi. Bu displeylar LCD displey (Liquid-Crystal Display) deb ataladi. Ularda dog‘siz yassi ekran va kam ishlatiladigan quvvat (bunday displeylarning ba’zi rusumlari 5 Bt ishlatishadi, elektron – nurli trubkali monitorlar esa, - 100 Bt ishlatishadi) mavjud. Rang uzatish sifati bo‘yicha faol Matrisali suyuqkristalli panellar hozirgi kunda elektron nurli trubkali monitorlar rusumlaridan o‘zib ketyapti. Shuni aytib o‘tish kerakki, suyuq kristalli ekranlarning qobiliyati elektron nurli trubkalarga qaraganda sust va ular qimmatroq turadi. Suyuqkristalli displeylarni bir necha turlari mavjud, bular: passiv Matrisali rangli, aktiv Matrisali rangli (analogli) va aktiv Matrisali rangli (raqamli) eng zamonaviy displey. Suyuq kristalli ekranda poryarizatsion yorug‘lik filtri ikkita alohida yorug‘lik to‘lqin tug‘diradi va faqat qutblanish tekisligi uni o‘ziga parallel bo‘lgan to‘lqinni o‘tkazadi. Suyuq kristalli monitorda, birinchi o‘qiga perpendikulyar qilib ikkinchi yorug‘lik filtrni joylashtirsak, yorug‘lik o‘tishini to‘liq bartaraf etishimiz mumkin (ekran to‘q qora bo‘ladi). Ikkinchi filtr qutblanish o‘qini aylantirsak, ya’ni yorug‘lik filtri o‘qlar orasidagi burchakni o‘zgartirsak, yorug‘lik

energiyasini o‘tuvchanlik sonini, ya’ni ekran yorug‘ligini ham o‘zgartirishimiz mumkin bo‘ladi.

Rangli suyuq kristalli ekranida tasvirni har bir pikseliga uchta yacheykasi bo‘lgan yana bir qo‘shimcha yorug‘lik filtri bor – qizil, yashil va ko‘k nuqtalarni aks etish uchun bittadan yacheyka yorug‘lik to‘lqini suyuq kristalli yacheyka orqali o‘tadi, lekin har bir rang o‘z yacheykasiga ega.

Suyuqkristallar sterjen sifatli molekulalarni ifodalaydi, ularni xususiyati suyuqlikka o‘xshaydi.Bu suyuqlik o‘zidan yorug‘likni bemalol o‘tkazadi, uni qutblanish tekisligi optik o‘qiga parallel, ammo, molekulalar elektr zaryadi ta’siri ostida o‘z yo‘nalishini o‘zgartiradi.Bir vaqtda bundan o‘tuvchi yorug‘lik to‘lqinni qutblanish tekisligi yo‘nalishi o‘zgaradi. Biroq monoxrom suyuq kristalli monitorda rang filtri mavjud emas, unda bo‘linishning bir elementiga kul rang gradatsiyasini uzatish uchun bir nechta suyuq kristalli monitorlarda har bir yacheyka yorug‘ligi bilan tranzistor orqali o‘tuvchi elektr zaryadi (aniqrog‘i kuchlanish) boshqaradi. Tranzistorlarning nomerlari ekran Matrisasidagi mazkur yacheykaning qator va ustunining raqamiga teng bo‘ladi. Tranzistorlar sonini (ustin va qatorlar bo‘yicha) ekran kattaligi aniqlaydi. Masalan, 800x600 kattalikdagi ekran gorizontal bo‘yicha 800 va vertikal bo‘yicha 600 ta tranzistorga ega. Yacheyka keladigan kuchlanish impul’siga shunday ta’sir qiladiki, o‘tuvchi yorug‘lik to‘lqinining qutblanish tekisligi buriladi, bunda kuchlanish qancha yuqori bo‘lsa, burilish burchagi ham katta bo‘ladi. Yacheykani barcha kristallarini to‘liq o‘zgarishi, masalan, yoqilgan holatiga muvofiq bo‘ladi va tasvirning maksimal kontrastini aniqlaydi. Shunday qilib, qo‘shni yacheykalarning qutblanish tekisligini yo‘nalishlarida o‘zgarishlar qancha katta bo‘lsa, tasvirning kontrasti shuncha yuqori bo‘ladi. Passiv Matrisali suyuq kristalli monitor yacheykasiga pul’slovchi kuchlanish uzatiladi, shuning uchun ular tasvir yorug‘ligi bilan aktiv Matrisali suyuq kristalli monitordan qolib ketadi. Aktiv Matrisali suyuq kristalli monitorlarni har bir yacheykasiga doimiy kuchlanish beriladi. Tasvir yorug‘ligini ba’zi bir konstruksiyalarda yaxshilanishi uchun boshqaruv usuli ishlatiladi. Uni ikkilamchi skanerlash deyishadi va uning uskunasi – ikkilamchi skanerlashli suyuq kristalli monitorlardir (double-scan LCD). Ekran

mustaqil ishlaydigan ikkita bo‘lakka (yuqori va quyi) bo‘linadi. Bu yacheykaga tushadigan impul’slar orasidagi intervallarni qisqarishiga olib keladi.

Ikkilamchi skanerlash tasvir yorug‘ligini oshiribgina qolmay, ekran reaktsiyasi vaqtini ham tushiradi, chunki yangi tasvir yaratilishiga vaqtni qisqartiradi. Shuning uchun ikkilamchi skanerlash suyuqkristalli monitorlar tez o‘zgaradigan tasvirni yaratish uchun, masalan, televizion tasvir uchun ko‘proq to‘g‘ri keladi. Aktiv Matrisali suyuq kristalli monitorlarda har bir yacheykani alohida tranzistor xarita boshqaradi.Masalan, 1024x768 aktiv Matrisali displey 786x432 tranzistorlarga ega. Bu passiv Matrisali suyuq kristalli monitorlarga qaraganda, tasvirning yuqori yorug‘ligini ta’minlaydi, chunki har bir yacheyka impul’sli emas, balki doimiy elektr maydoni ta’siri ostida bo‘ladi. Bunda, albatta, aktiv Matrisa ko‘proq energiya sarflaydi. Bundan tashqari, har bir yacheyka uchun alohida tranzistor kaliti mavjudligi bunday asboblarni ishlab chiqarilishini murakkablashtiradi va ularni qimmatbaho qiladi.

Aktiv Matrisada bo‘lganidek, passiv Matrisali suyuqkristalli monitorlarda ham ikkinchi qutblangan yoryg‘lik filtri yacheyka orqali o‘tuvchi yorug‘lik miqdorini boshqaradi. Yacheykalar yorug‘lik filtri orqali o‘tadigan qutblash tekisligiga iloji boricha yaqin o‘tadigan qilib yorug‘lik to‘lqinini qutblash tekisligini qaytaradi. Har bir yacheykada yorug‘lik qancha ko‘p bo‘lsa, piksel shuncha yorug‘ bo‘ladi. Monoxromli (oq-qora) suyuq kristalli monitorlarda kulrang gradatsiyasi (CI gacha) yoki yacheyka yorug‘ligi o‘zgarishi hisobiga, yoki bitta pikselga muvaffaq bo‘lgan yoqilgan va o‘chirilgan yacheykalar miqdori orasidagi mutanosibligi hisobiga yaratiladi. Rangli suyuq kristalli monitorlarda bir pikselga uchta yacheyka to‘g‘ri keladi va ularning yorug‘ligini boshqargan holda ekranda tasvirni turli rangliligiga erishish mumkin. Hozirgi kunda passiv Matrisali va ikkilamchi skanerlashli suyuq kristalli monitorlarkeng tarqalgan, chunki tasvir sifati bo‘yicha aktiv Matrisali suyuq kristalli monitorlarga yaqinlashtirib olindi, ammo oddiy passiv Matrisali suyuq kristalli monitorlarga qaraganga uncha qimmat emas.

Aktiv Matrisali monitorlarni ishlab chirarishda uchraydigan jiddiy muammo chiqish nazoratida braklash foizi yo‘qori: panellarda ishlamaydigan yacheykalar

haddan tashqari ko‘p topiladi (asosan tranzistorlar buzilgani uchun). Bu ularni ancha qimmatlashtiradi, chunki braklangan mahsulot qiymati sifatli mahsulot qiymatiga kiradi.

Eng yaxshi rangli displey – aktiv Matrisali displeylardir, yoki har bir pikselni uchta tranzistor (qizil, yashil va kko‘k rang uchun) boshqaradigan ingichka plyonkali tranzistorlardir (TFT). Aktiv Matrisali monitorlar tasvir yorug‘ligi bo‘yicha passiv displeylardan oshadi, shuning uchun ulardagi tasvir burchak ostida yaxshi ko‘rinadi.

Hozirgi kunda suyuqkristalli monitorlar nafaqat portativ kompyuterlarda faol ishlayapti, balki stol usti tizimlarda ham qo‘llanila boshladi. Ularni elektron nurli trubkali monitorlardan bir qator faziolatlari ajratib turadi:

- Ma’lumot aks etishi uchun monitor ekranining butun yuzi ishlatiladi. Masalan, 17 dyuymli suyuqkristalli monitorlarda ko‘rish doirasi – 17 dyuym, elektron nurli trubkali monitorlarda esa faqat 15 dyuym.
- Ish joyini tejashga imkon beradigan kichik chuqurligi.
- Kam energiya ishlatadi, natijada issiqlik kam chiqaradi.
- Suyur kristalli monitorlarda lyuminoforni “kuiyshiga” ga moyil emas.
- Dizaynerlarni xursand qiladigani, monitorni 90^0 ga burish mumkinligi.

Suyuqkristalli monitorlarni sotib olishdan oldin barcha kamchilik va yutuqlarini hisobga olish kerak. Bu monitorlarni keng tarqalmasligiga sabab – ularning yo‘qori narxi (ammo kompyuter apparatlarini ta’minlashga taaluqli narxi doim tushib turadi)

Skanerlar haqida umumiy ma’lumotlar.

Skaner bu ma’lumotlarni qog‘ozli hujjatdan bevosita EHM ga kiritish qurilmasidir. Matnlar, sxemalar, rasmlar, grafiklar, fotografiyalar va boshqa grafik axborotni kiritish mumkin. Skaner nusxa ko‘chirish apparatiga o‘xshab qog‘ozli hujjatning tasviri nusxasini qog‘ozda emas, balki elektron ko‘rinishda yaratadi — tasvirning elektron nusxasi yaratiladi.

Skanerlar hujjatlarni qayta ishlash elektron tizimining muxim bo‘g‘ini va istalgan «elektron stol» ning kerakli elementidir. O‘z faoliyatining natijalarini

fayllarga yozib va ma’lumotni kog‘ozli hujjatlardan SHK ga obrazlarni avtomatik anglash tizimi orqali skaner yordamida kiritib, qog‘ozsiz ish yuritish tizimini yaratishga amaliy qadam qo‘yish mumkin.

Skanerlar juda xilma-xildir va ularni bir qator belgilari bo‘yicha tasniflash mumkin. Skanerlar oq-qora va rangli bo‘ladi.

Oq-qora skanerlar shtrixli va nimrangli tasvirlarni o‘qishi mumkin. SHtrixli tasvirlar nim ranglarni, yoki boshqacha aytganda, qo‘l rang darajalarini uzatmaydi. Nim rangli tasvirlar qo‘l rangning 16, 64 yoki 256 darajalarini anglash va uzatish imkonini beradi.

Rangli skanerlar oq-qora va rangli asl nusholar (originallar) bilan ishlaydi. Birinchi holatda ular ham shtrixli, ham nim rangli tasvirlarni o‘qish uchun ishlatilishi mumkin.

Rangli skanerlarda rangli RGB (Red-Green-Blue) modul ishlatiladi: skanerlanadigan tasvir aylanadigan RGB yorug‘lik filtri yoki ketma-ket yondiriladigan uchta rangli chiroqlar orqali yoritiladi; har bir asosiy rangga mos signal alohida qayta ishlanadi.

Uzatiladigan ranglar soni 256 tadan 65536 tagacha va xatto 16,8 milliontagacha tebranishi mumkin.

Skanerlarning o‘tkazish qobiliyati tasvirning bir dyuymdagi ajratiladigan nuqtalar miqdori bilan o‘lchanadi va 75 dan 1600 dpi gacha (dot per inch) bo‘ladi.

Konstruktiv jihatdan skanerlar dastaki va stolli bo‘ladi.

Stolli skanerlar, o‘z navbatida planshetli, rolikli va proeksion bo‘ladi.

Shaffof tashuvchilardan tasvirni o‘qiydigan slayd-skanerlar alohida ajralib turadi.

Skanerlarning tiplari

Dastlaki Skanerlarning tuzulishi juda oddiydir: ular qo‘l bilan tasvir bo‘ylab siljitiladi. Ular yordamida bir marta o‘tishda tasvir satrlarining ozgina miqdori kiritiladi (ularning qamrab olishi odatda 105 mm dan oshmaydi). Dastaki skanerlarda qayd qiluvchi chiroq bo‘lib, u skanerlashning ruxsat etiladigan tezligi oshganligini

operatorga bildirib turadi. Bu skanerlar kichik o‘lchamli va past narxdadir. Skanerlash tezligi 5—50 mm/s (o‘tkazish qobiliyatiga bog‘liq).

Masalan, Mustek GS-400L — oq-qora nim rangli, SG-8400T—rangli.

Planshetli skanerlar eng ko‘p tarqalgan; ularda skanerlovchi kallak asl nusxaga nisbatan avtomatik siljiydi; ular ham varaqli, ham risolalangan (broshyuralangan) hujjatlarni (kitoblarni) skanerlash imkonini beradi. Skanerlash tezligi: bir betga (A4 O‘lchamli) 2—10 sekund.

Masalan, rangli skanerlar: Muctek Paragon 1200, Epson ES 1200, HP Scan Jet 5 S va R, HP Scan Jet 11SX.

Katta formatdagi hujjatlar bilan ishlaydigan skanerlar orasida Agfa firmasining ommaviy skanerlarini, masalan, Agfa Argus II ni ko‘rsatib o‘tish kerak, u 600 x 1200 dpi fizik o‘tkazish qobiliyatiga (Ultra View 2400x2400 dpi interpolyasiyalovchi texnologiyani ishlatgandagi mantiqiy o‘tkazish) ega, 4096 rang tuslarini uzatadi, tasvirni 7—9 marta masshtablaydi.

Rolikli skanerlar eng avtomatlashtirilgandir; ularda asl nusxa skanerlovchi kallakka nisbatan avtomatik siljiydi, ko‘pincha hujjatlar avtomatik beriladi, lekin skanerlanadigan hujjatlar faqat varaqli.

Misol: Mustek CF-63 skaneri, tezligi bir betga 10 sekund.

Proeksiey skanerlar tashqi ko‘rinishdan fotokattalashtirgichni eslatadi, lekin pastda skanerlanadigan hujjat yotadi, yuqorida esa skanerlovchi kallak joylashadi. Skaner malumotli hujjatni optik yo’l bilan skanerlaydi va olingan ma’lumotni fayl ko‘rinishda kompyuter xotirasiga kiritadi.

Slayd-skanerlar ham tuzulish jihatdan turlicha bo‘ladi: planshetli, barabanli, proeksiey va b. Shaffof asl nusxa 35 mm dan 300 mm gacha chiziqli o‘lchamli to‘g‘ri to‘rtburchak tomonlari ko‘rinishiga ega. Tavsiflari bo‘yicha slayd-skanerlar eng yuqori sifatlidir: ularning o‘tkazish qobiliyati odatda 2000 dan 5000 dpi gacha oraliqda yotadi.

Masalan, barabanli skanerlar, ularda taxminan 200x300 mm li shaffof asl nusxa (slayd) aylanadigan barabanga mahkamlanadi. Howtek Scan Master skanerida o‘tkazish qobiliyati 4000 dpi, Scan View Scan Mate Magis skanerida 4096 ta tusni

uzatishda o‘tkazish qobiliyati 2000 dpi. Eng katta o‘tkazish qobiliyatiga kichik o‘lchamli slaydlar (tomoni 120 mm gacha) bilan ishlaydigan skanerlar ega. Scitex Leaf Scan 45 skanerida 64500 ta tusni uzatishda o‘tkazish qobiliyati 5080 dpi ga teng.

Grafik axborotni shaxsiy kompyuterlarda tasvirlash formatlari

Grafik axborotni kompyuter fayllarida tasvirlashning ikkita formati: rastrli va vektorli formatlar mavjud.

Rastrli formatda grafik tasvir nuqtalar to‘plamining naqshinkor termasi ko‘rinishida (nollar va birlar) faylda eslab qolinadi, bu to‘plam nuqtalari tasvirning displey ekranida aks etishining piksellariga mos keladi. Mashina xotirasida skaner bilan yaratilayotgan fayl rastrli formatga (bitli karta deb ataladigan) ega. Bu faylni standart matnli va grafik prosescorlar bilan taxrir qilish imkoniyati yo‘q, negaki bu prosescorlar axborotni naqshinkor tasvirlash bilan ishlay olmaydi.

Vektorli formatda axborot shriftlar, belgilar kodlari, xat boshi va sh.o‘. tavsiflari bilan identifikatsiyalanadi.

Vektorli formatlarning rastrli formatdan asosiy farqini bunday misolda ko‘rsatish mumkin: vektorli formatda aylana radiusi, o‘z markazining koordinatasi, chiziq qalinligi va tili bilan identifikatsiyalanadi, rastrli formatda esa aylanani geometrik shakllantiruvchi nuqtalarning oddiygina ketma-ket qatorlari saqlanadi.

Yana shuni hisobga olish kerakki, bitli karta o‘zining saqlanishi uchun katta xotira sig‘imini talab etadi. Demak, o‘tkazish qobiliyati millimetrga 10 ta nuqtali va nim ranglarni uzatmaydigan (shtrixli tasvir) A4 formatli (204x297 mm) hujjat 1 varapshing bitli kartasi xotiraning 1 Mbaytdan ortiqrog‘ini band qiladi, shuni o‘zi esa qo‘l rangning 16 ta tusini amalga oshirishda 4 Mbaytni, rangli yuqori sifatli tasvirni (High Color standarti — 65536 ta ranglar) — 16 Mbaytni band qiladi. Boshqacha aytganda, True Color standartini ishlatganda va o‘tkazish qobiliyati millimetrga 50 ta nuqta bo‘lganda, xattoki bitta bitli kartani saqlash uchun MDYning sig‘imi etmasligi mumkin.

Bitli kartani saqlash uchun kerak bo‘ladigan xotira sig‘imini qisqartirish maqsadida axborotni siqishning (zichligining) turli usullari ishlatiladi. Eng ko‘p

tarqalgan binarli rastrli siqish Group 4 formati ma'lumotlarni 40:1 gacha siqish koeffitsiyentini beradi (qiymatlarning ichida bor narsasiga bog‘liq ravishda). Boshqa ishlatiladigan siqish formatlari: Group 3, CUFF (Compressed Tagged Image Fayl Format), MPEG, CALC, BMP, GIF va b. (bitli kartalar fayllari qisqartma so‘zga mos kelgan qisqartmaga ega bo‘ladi).

Siqishtirilmagan formatlar: Uncompressed TTIF, PCX, RLC va b.

Skanerni obrazlarni anlash tizimlarini dasturlari bilan, masalan OCR (Optical Charaster Resoghition) tipidagi dasturlar bilan birgalikda ishlatish eng afzal hisoblanadi. OSR tizimi skaner bilan hujjatdan o‘qilgan belgilarning (harflar va raqamlar) bitli (naqshinkor) konturlarini anglaydi va ularni matnli taxrirchilar uchun qulay vektorli formatga aylantirib, ASC II kodlari bilan kodlaydi.

Ba'zi OSR tizimlarini oldindan anglashga o‘qitish kerak—skaner xotirasiga anglanadigan belgilarni va ularga mos kelgan kodlarning andazalarini va timsollarini kiritish kerak. Turli alfavitlarda (masalan, lotincha (inglizcha) va ruscha-kirillitsa) va shriftlarning turli garniturida (yozilish usullari) yozilishi bo‘yicha mos keladigan harflarni anglashda qiyinchiliklar paydo bo‘ladi. Lekin ko‘pchilik tizimlar o‘qitishni talab etmaydi: ularning xotirasiga oldindan anglanadigan belgilar joylashtirilgan. Masalan, eng yaxshi OSR lardan biri — TIGER 2.0 dasturli paketi 30 ta turli garniturining timsolini o‘z ichiga oladi, ingliz va rus harflarini anglash uchun esa ichiga sozlangan elektron lug‘atlar ishlatiladi.

Keyingi yillarda Omnifont tipidagi obrazlarni anglash intellektual dasturlari paydo bo‘ldi, u belgilarni nuqtalar bo‘yicha emas, balki belgilarni har biri uchun tavsifli bo‘lgan shaxsiy topologiya bo‘yicha anglaydi. Obrazlarni anglash tizimi mavjud bo‘lganda matn endi shaxsiy kompyuter xotirasiga bitli karta ko‘rinishida emas, balki kodlar ko‘rinishida yoziladi va uni oddiy matn muharrirlari bilan taxrir qilish mumkin.

Fayllarni rastrli formatda quyidagi holatlarda saqlash mumkin, agar:

• hujjatlar va ularga mos kelgan fayllar ularni ishlatish jarayonida taxrir kil inmasligi kerak;

• hujjat asl nusxaning faksimil nusxasi ko‘rinishida saqlanishi kerak (fotografiyalar, rasmlar, imzolangan hujjatlar va sh.o‘.);

• ko‘p sonli ulkan fayllarni (1—20 Mbayt) saqlash va ko‘rib chiqish uchun texnik imkoniyatlar mavjud.

Skaner shaxsiy kompyuterning ketma-ket portiga ulanadi.

Skaner bilan ishlash uchun shaxsiy kompyuter maxsus drayverga, imkoni bo‘lsa, TWAIN standartiga mos keluvchi drayverga ega bo‘lishi kerak. Bu holda ko‘p sonli TWAIN bilan mos keladigan skanerlar bilan ishlash va TWAIN standartini qo‘llaydigan fayllarni qayta ishlovchi dasturlar bilan ishlash imkoni bor, masalan, keng tarqalgan Corel Draw, Adobe Photoshop, MaxMate, Pisture Publisher, Photo Finish va b. dasturlar.

Ko‘pchilik drayverlar SCSI lokal kompyuter interfeysi bilan ishlashga mo‘ljallangan.

Skanerni tanlashda hisobga olinadigan asosiy omillar:

• skanerlanishi kerak bo‘lgan hujjatlarning o‘lchani, rangliligi va shakli (varaqdi, risolalangan va b.) skaner imkoniyatlariga mos kelishi kerak;

• skanerning o‘tkazish qobiliyati hujjatlarning yuqori sifatli qattiq nusxasini ularning elektron obrazlari bo‘yicha ta’minlashi kerak;

• skaner unumdorligi olinayotgan tasvirning yaroqli sifatini ta’minlaydigan darajasida etarlicha yuqori bo‘lishi kerak;

• agar elektron hujjat o‘lchamlari hisoblashlarni amalga oshirish uchun asos bo‘lib xizmat qilsa, asl nusxaga nisbatan olinayotgan elektron tasvirning o‘lchamlarida minimal xatolik ta’minlanishi kerak;

• fayllar kompyuter xotirasida saqlanganda rastrli fayllarni siqishning dasturli vositalari borligi;

• rastrli fayllarda tasvir sifatini yaxshilash uchun dasturli-apparatli vositalarning borligi (tasvirning yorqinligini va keskinligini oshirish, asosiy rangning «kirini» yo‘qotish);

• tashuvchi qog‘ozini sifati va tipi olinayotgan elektron tasvirning sifatiga ma’lum chegaralarda kuchli ta’sir qilmasligi kerak;

• skanerda ishlash oddiy va qulay bo‘lishi kerak va tashuvchi noto‘g‘ri quyilganda skanerlashdagi xatoliklar bo‘lmasligi kerak;

• skaner narxi.

PRINTERLAR

Bosuvchi qurilmalar (printerlar) bu qiymatlarni EHM dan chiqarish qurilmasi bo‘lib, u ma’lumotning ASC II kodlarini ularga mos kelgan grafikli belgilarga (harflar, raqamlar, ishoralarga va h.k.) o‘zgartiradi va bu belgilarni qog‘ozda qayd etadi.

Printer shaxsiy kompyuter texnik qurilmasining eng rivojlangan guruhidir, ularning 1000 tagacha turli xil modifikasiyalari bor. Printerlar o‘zaro quyidagi tavsiflar bo‘yicha farqlanadi:

- rangliligi (oq-qora va rangli);
- belgilarni shakllantirish usuli (belgilarni bosuvchi va belgilarni sintezlovchi);
- ish tamoyili (Matrisali, ternik (qizdirishga oid), purkagichli, lazerli);
- bosish (zarbli va zarbsiz) va satrlarni shakllantirish (ketma-ket va parallel) usullari;
- karetka kengligi (375 450 mm li keng va 250 mm li tor karetkali);
- bosish satri uzunligi (80 ta va 132-136 ta belgi);
- belgilarni terish (ASCII belgilarini to‘liq terishgacha);
- bosish tezligi;
- o‘tkazish qobiliyati va h.k.

Vir qator guruxlarning ichida printerlarning bir nechta turlarini ajratish mumkin: masalan, shaxsiy kompyuterda keng ishlatiladigan belgilarni sintezlovchi Matrisali printerlar ish tamoyili bo‘yicha zarbli, termografikli, elektrografikli, elektrostatik, magnitografikli va boshqa bo‘lishi mumkin.

Zarbli printerlar orasida ignali (Matrisali) lar eng ko‘p tarqalgan, lekin hali ham literli, shar ko‘rinishli, gulbargli (moychechak tipidagi) va boshqalar uchrab turadi.

Printerlarda bosish belgi bo‘yicha, satr va saxifa bo‘yicha bo‘lishi mumkin. Bosish tezligi sekundiga 10-300 ta ishoradan (zarbli printerlar) sekundiga 500-1000 tagacha va xattoki sekundiga bir necha o‘nlab (20 tagacha) saxifalargacha (zarbsiz lazerli printerlar) oraliqda; o‘tkazish qobiliyati millimetrda 3-5 nuqtadan millimetrda 30-40 nuqtagacha bo‘ladi (lazerli printerlar).

Matnli bosish uchun umumiy holda turlicha bosish sifati bilan tavsiflanuvchi quyidagi rejimlar bor:

- xonaki bosish rejimi (Draft);
- bosmaxonanikiga yaqin bosish rejimi (NLq – Near Letter quality);
- bosmaxonaniki kabi bosish rejimi (Lq - Letter quality);
- yuqori sifatli bosish rejimi (SLq - Super Letter quality).

Printerlar, odatda, ikki rejimda - matnli va grafikli rejimlarda ishlashi mumkin.

Matnli rejimda printerga bosilishi kerak bo‘lgan belgilar koda yuboriladi, shu bilan birga belgilar konturi printerning ishora generatoridan tanlab olinadi.

Grafikli rejimda printerga tasvir nuqtalarining ketma-ketligi va joylashgan joyini aniqlovchi kodlar yuboriladi.

Matnli rejimda printerlar odatda bir nechta shriftlarni va ularning turli ko‘rinishlarini qo‘llaydi, ularning ichida Roman (yozuv mashinkasining mayda shrifti), Italis (kursiv), boldfase (yarim qora), Expanded (cho‘zilgan), Elite (yarim siqilgan), Condensed (siqilgan), pisa (to‘g‘ri shrift - sisero), sourier (kurer), san serif (yorilgan shrift san serif), serif (serif), prestige elite (prestij-elita) va proporsionalli shrift (belgi uchun ajratiladigan maydon kengligi belgining kengligiga bog‘liq bo‘ladi) keng tarqalgandir.

Printerni ruslashtirilganligi (milliylashtirilishi) maqsadga muvofiqdir-o‘zining vositalari bilan rus harflarini kirillitsani bosishni ta’minlasin; aks holda SHK ga maxsus drayverlarni qo‘shish talab etiladi.

Ko‘pgina printerlar grafikli ma’lumotlarni samarali chiqarishni amalga oshirish imkonini beradi; bosishning servis rejimlari: qalin bosish, ikkilangan kenglikdagi bosish, ostita chizib bosish, yuqorigi va pastki indekslar bilan, ajratilgan bosish (har

bir belgi ikki marta bosiladi) va ikki marta o‘tib bosish (ikkinchi marta belgi ozgina surilib bosiladi); ko‘p rangli bosish (100 tagacha turli xil rang va tuslar).

Matrisali printerlar

Matrisali printerlarda tasvir nuqtalardan zarbli usul bilan shakllanadi, shuning uchun ularni zarbli-Matrisali printer deb atash to‘g‘riroqdir, shunday ham ishorani sintezlovchi printerlarni boshqa tiplari ko‘pincha belgilarni Matrisali shakllantirishni, lekin zarbsiz usul bilan, ishlatadi. SHunga qaramay, Matrisali printerlar, bu ularning umumqabul qilingan kodi, shuning uchun uni asos qilib olaniz.

Ignali (zarbli) Matrisali printerlarda nuqtalarni bosish, bo‘yovchi lenta orqali qog‘ozga zarba beruvchi ingichka ignalar bilan amalga oshiriladi. Har bir igna xususiy elektromagnit bilan boshqariladi. Bosuvchi uzel gorizontal yo‘nalishda siljiydi va qatordagi belgilar ketma-ket bosiladi. Ko‘pchilik printerlar bosishni ham to‘g‘ri, ham teskari yo‘nalishda bajaradi. Bosuvchi kallakdagi ignalarning soni bosish sifatini belgilaydi. Qimmat bo‘lmagan printerlar 9 ta ignaga ega. Bunday printerlarda belgilar Matrisasi 7x9 yoki 9x9 nuqtalar o‘lchamiga ega. Mukammallashgan Matrisali printerlar 18 va xattoki 24 ta ignaga ega bo‘ladi.

Matrisali printerlarning bosish sifati yana bosuvchi kallakning bir nechta o‘tishida qisman qoplash bilan bosish jarayonida nuqtalarni chiqarish imkoniyati orqali ham aniqlanadi.

Turli sondagi ignali printerlarda bosishning turli rejimlari turlicha amalga oshiriladi. 9 ta ignali printerlarda Draft rejimidagi bosish bosuvchi kallakning satr bo‘ylab bir marta o‘tishida bajariladi. Bu eng tez bosish rejimi, lekin u eng past sifatli hamdir. NLq rejimi ikki marta o‘tishda amalga oshiriladi: kallak bir marta o‘tgandan keyin qog‘oz nuqtaning yarim ulchamiga mos masofaga cho‘ziladi; keyin nuqtalarni qisman qoplash bilan ikkinchi o‘tish bajariladi. Bunda bosish tezligi ikki marta kamayadi.

Matrisali printerlarning ish rejimini qayta ulash va shriftlarni almashtirish dastur bilan ham, apparat bilan ham qurilmalarda bor bo‘lgan klavishlarni va yoki qayta ulagichlarni tegishlicha o‘rnatish bilan amalga oshirilishi mumkin.

Matrisali printerlarning tezkorligi matnni Draft rejimida bosishda sekundiga 100 tadan 300 tagacha belgi oralig‘ida yotadi, bu taxminan minutiga ikki saxifaga to‘g‘ri keladi (varaqlarni almashtirishni hisobga olgan holda).

Termoprinterlar

Ignali Matrisali printerlardan tashqari Matrisali termoprinterlar guruxi ham bor, ular bosuvchi ignali kallak o‘rniga termoMatrisali kallak bilan jixozlangan va bosishda termoqog‘oz yoki termonusholovchi ishlatiladi (bu, shubxdsiz, ularning jiddiy kamchiligidir).

Purkagichli printerlar

Purkagnchli printerlar bosuvchi kallakda ignalar o‘rniga ingichka naychalar - soplolarga (konus naychalarga) ega, u orqali qog‘ozga bo‘yoq rangning (siyoxning) mayda tomchilari purkaladi. Bu zarbsiz bosuvchi qurilmadir. Bosuvchi kallakning Matrisasi odatda 12 tadan 64 tagacha soploga ega.

Keyingi yillarda ularning mukammallashishida jiddiy rivojlanishga erishildi: tasvirni shakllantirishda bosuvchi kallakning juda mayda soplolari yordamida qog‘ozga siyox tomchilarining yo‘naltirilgan portlatishga o‘xshash purkash - purkagichli bosishning pufaknali texnologiyasi deb ataluvchi usuli ishlatiladi.

Purkash jarayoni texnikasi quyidagicha bo‘ladi. Soplo devoriga elektrik qizdiruvchi element o‘rnatilgan bo‘lib, uning harorati elektr impulsi beriltanda 5-10 mks ichida keskin ortadi. qizdiruvchi element bilan kontaktda joylashgan siyoxning hammasi bir zumda burlanadi, bu bosimning keskin oshishiga olib keladi, buning okibatida siyox soplodan qog‘ozga otilib chiqadi. Otilgandan keyin siyox burlari kondensasiyalanadi, soploda pasaygan bosim zonasi hosil bo‘ladi va unga siyoxning yangi porsiyasi (ulushi) so‘riladi. Bu yangi texnologiya purkagichli printerlar va plotterlar olanida burilish yasadi, bu esa ularning o‘tkazish qobiliyatini yana bir pag‘onaga (dyuymda 600-1440 ta nuqtagacha) ko‘tarish imkonini berdi.

SHunday qilib, hozirgi vaqtda purkagichli printerlar millimetrga 50 tagacha nuqtali o‘tkazish qobiliyatini va sekundiga 500 tagacha belgini bosish tezligini ta’minlaydi va bunda bosish sifati lazerli bosish sifatiga yaqin bo‘ladi.

Purkagichli printerlar yozuvchi kallakda katta miqdordagi soplolarni ishlatib, **rangli bosishni** ham bajaradi, lekin bunda o‘tkazish qobiliyati oq-qoraga nisbatan taxminan ikki marta kamayadi (lekin Epson firmasi o‘tkazish qobiliyati 400 dpi bo‘lgan, rangli bosish tezligi sekundiga A4 o‘lchamli 4 betni tashkil etgan noyob rangli purkagichli Stylus 600 printerini yaratganligi to‘g‘risida axborot mavjud).

Rangli tasvirni yaratish uchun odatda, poligrafiyada qabul qilingan SMYK rangli sxema ishlatilib, u o‘z ichiga to‘rtga bazaviy (asosiy) rangni oladi: Cyan - havo rang, Magenta - to‘k qizil rang, Yellow - sariq rang, Key - etakchi (qora rang). Murakkab ranglar bazaviy ranglarni aralashtirib hosil qilinadi. Bosish sifati juda ajoyibdir - to‘liq rangli plakat deyarli bosmaxonanikidan farq qilmaydi.

Purkagichli printerlarning asosiy afzalliklari:

- yuqori bosish sifati, katta miqdorli soploli printerlar uchun lazerli printer bosish sifati bilan taqqoslasa bo‘ladi;

- xomaki bosish rejimida yuqori tezlik;

- oddiy, albatta, yaxshi zichlikdagi qog‘ozni ishlatish (60 dan 135 g/m^2 gacha), siyox yoyilib ketmasligi uchun;

- shovqinsiz ishlashi.

Purkagichli printerlarning asosiy kamchiliklari:

- soplo ichida siyoxning ko‘rib qolish xavfi, bu ba’zida bosuvchi kallakni almashtirish zarurligiga olib keladi;

- sarflanadigan materiallarning nisbatan yuqori narxdaligi, xususan, siyox uchun ballonchaning, ayniqsa agar u bosuvchi kallak bilan birlashtirilgan va birgalikda almashtirilsa (bunday tuzulish keng tarqalgan).

Lazerli printerlar

Lazerli printerlarda tasvirni shakllantirishning elektrografik usuli ishlatilib, bu usul shu nomdagi nusxa ko‘chiruvchi apparatlarda ishlatiladi. Lazer o‘ta ingichka yorug‘lik nurini yaratish uchun xizmat qiladi, bu nur oldindan tayyorlab quyilgan yorug‘likka sezgir baraban sirtida ko‘rinmaydigan nuqtali elektron tasvir konturini chizadi - elektr zaryad lazer nuri bilan yoritilgan nuqtalardan baraban sirtiga oqib tushadi. Elektron tasvir tushgandan keyin razryadlangan uchastkalarga yopishib

qolgan bo‘yoq, (toner) kukuni bilan bosish bajariladi - tonerni barabandan qog‘ozga olib o‘tiladi va tasvirni qog‘ozda tonerni qizdirib, u erib ketguncha kotiriladi.

Lazerli printerlar millimetrda 50 tagacha nuqtalarni va sekundiga 1000 tagacha belgilarni bosuvchi tezlikni ta’minlaydigan o‘tkazish kobililiyatli eng yuqori sifatli bosishni ta’minlaydi. Rangli lazerli printerlar keng ishlatiladi. Masalan, Tektonik (AQSH) firmasining Phaser 550 lazerli printeri gorizontal bo‘yicha ham, vertikal bo‘yicha ham millimetrda 48 nuqtali o‘tkazish qobiliyatiga ega; rangli bosish tezligi - minutiga A4 o‘lchamli 5 bet, monoxromli bosish tezligi - minutiga 14 bet.

SHK larga printerlar ham parallel, ham ketma-ket portlar orqali ulanishi mumkin.

Parallel portlar Sentronisc tipidagi adapterlar orqali parallel ishlovchi (malumotni birdaniga baytlab qabul qiladigan) printerlarni ulash uchun (odatda bir vaqtning o‘zida 3 tagacha printerni ulash mumkin) ishlatiladi.

Ketma-ket portlar (2 dona) RS 232S (S2 birikish joyi) tipidagi adapterlar orqali ketma-ket ishlaydigan (ma’lumotni ketma-ket 1 bitdan qabul qiladigan) printerlarni ulash uchun xizmat qiladi. Ko‘pchilik tez ishlovchi printerlar parallel portlarni ishlatadi.

Tezkor printerlar shaxsiy buferli xotiraga ega bo‘ladi, ular SHK bilan ma’lumotlarni almashishda ham, yuklanadigan shrifglarni saqlash uchun ham ishlatiladi. Matrisali printerlarning xotirasi katta emas - bir necha yuzlab kilobaytlargacha, purkagichli printerlarda bir necha megabaytlargacha va lazerli printerlarda bir necha o‘nlab megabaytlargacha.

Xulosa qilib shuni ta’kidlash kerakki, SHK larning eng ommaviy printerlarini Seiko Epson (YAponiya) firmasi (ularning ulushi kamida 30% ni tashkil etadi) ishlab chiqaradi. Xattoki IBM PC printerlarining standarti - Epson standarti mavjud. Star, Mannesmann, Sitizen, Panasonic va boshqa firmalarning printerlari ham keng ishlatiladi.

Printerni tanlashda quyidagi omillarni hisobga olish kerak:

- funksional imkoniyatlar to‘plami, ular bo‘yicha printerni konkret masalani echish uchun qo‘llanishligini baholash mumkin (bosilgan hujjatlar o‘lchamlari, bajariladigan ishlar hajni, ruslashtirilganligi, kerakli shriftlarning borligi va b.);

- rangli tasvirni shakllantirish imkoniyati;

- tasvir sifati (O‘tkazish qobiliyati);

- ishlash ishonchliligi va qulayligi, servis;

- tashuvchi, sarflanadigan materiallar, qurilmaga xizmat ko‘rsatish, elektroenergiyani iste’mol qilish narxlarini o‘z ichiga olgan ekspluatasiya harajatlari;

- printer narxi.

Planshet - bu oddiy tekis to'g’ri burchakli yuza bo'lib, unga tayyor chizma qo'yiladi va chizmaning ustidan ko'rsatkich bilan yurgiziladi.

Nurli qalam - nurli qalam bilan chizma ekraning yuzasiga chiziladi. Qalam korpusi ichida teshik bor, bu teshikka linza o'rnatilgan. Nur ekrandan fotoelementga tushadi. Fotoelementdan kuchlanish kuchayturuvchi qurilmada kuchaytirilgandan keyin, maxsus qayta ulovchi moslamaga uzatiladi.

Sichqon - bu grafik xabar kiritish qurilmasi tashqi ko'rinishdan sichqonga o'xshaydi. Shuning uchun ixtirochilar aniq texnik nom o'ylab topmasdan uni sichqon deb atadilar. Sichqon qo'lga sig’adigan quticha bo'lib tepasida uchta yoki ikkita tugmasi bor.

Klaviatura - foydalanuvchi uchun muhim qurilma bo’lib, uning yordamida SHKga qiymatlar, buyruqlar va boshqarish tasirlari kiritiladi. Klavishalarga lotin va rus alfavita harflari: o’nli raqamlar; matematik, grafik va maxsus xizmat belgilari; tinish belgilari; ba’zi buyruqlar va b. belgilangan.

SHK tipiga bog’liq ravishda klavishalarning vazifalari, ularning belgilanishi va joylashishi o’zgarib turishi mumkin.

Klaviatura ko’pincha 101 klavishaga ega, lekin 84 klavishali eski klaviatura va yangi Windows sistemasida ishlatish uchun qulay 104 klavishali klaviaturalar ham mavjud.

Modemlar (MOdulyator-DEModulyator) - aniq bir aloqa kanalida ishlatish uchun qabul qilingan signallarni to'g'ri (modulyator) va teskari (demodulyator) o'zgartirish qurilmasidir.

Eng avvalo modem quyidagi vazifalarni bajarish uchun mo'ljallangan:

- uzatishda: keng polosali impulslarni (raqamli kodni) tor polosaliga (analog signallarga) o'zgartirish;
- qabul qilishda: qabul qilingan signalni filtrlash va detektorlash uchun, ya'ni tor polosali analogli signalni raqamli kodga teskari o'zgartirish.

Kompyuter grafikasining dasturiy vositalari: qurilma drayverlari, grafik dasturlar, maxsus grafik tizimlar va dasturiy paketlar.

Grafika bilan ishlovchi dasturlarning turli-tumanligini ko‘rganda yosh dizayner o‘zini yoqotib qo‘yishi mumkin. Kerakli masalalarni echish uchun ideal mos keluvchi, hamda maksimal qulay va foydali ishni sizga ta'minlovchi dastur qanday qilib tanlanadi? Bu bo‘limda grafik dasturlarning sharhiga to‘xtalib o‘tamiz, siz ko‘rasizki, dasturni to‘g‘ri tanlash dizaynerning oldida turgan masala atrofida aniqlanadi: chop etish uchun tasvir tayyorlash kerakmi yoki siz shriftlarni chizasizmi yoki web-sahifa yaratmoqchimisiz yoki hammasi birgalikda qilmoqchimisiz. qolgani endi o‘zingizni xohshingiz bo‘yicha bo‘ladi.

Microsoft Paint. Minimum imkoniyatga ega sodda grafik redaktor bo‘lib, Windows operatsion tizimi bilan o‘rnatiladi, jiddiy masalalarni echishda qo‘llanilmasa ham har holda eslab o‘tishimizga loyiqdir. Unda web-sahifa uchun rasm tayyorlanadi va GIF ga o‘tkazilganda ajoyib rasm hosil bo‘ladi. SHuning uchun sodda grafik redaktorlaridan o‘z o‘rnida foydalanish kerak.

Microsoft Photo Editor – bu redaktor asosan fotografiyalar bilan ishlash uchun mo‘ljallangan. Ko‘pincha Microsoft Office paketi bilan joylashtiriladi, shuning uchun ham keng tarqalgan.

Microsoft Image Composer – Microsoft firmasining grafikani qayta ishlovchi bir muncha rivojlangan dasturlaridan biri. Ishlab chiqaruvchilar fikriga ko‘ra Internet uchun grafika tayyorlashda Adobe Photoshop redaktoriga raqobatdosh deb hisoblanadi. Microsoft Mahsulotining asosiy xislati shundaki, uning interfaysi sodda

va qulay. Bu redaktor diskda ozgina joy egallaydi va juda tez o‘rnatiladi. Lekin poligrafiyada foydalanadigan tasvirlar uchun asqotmaydi. Microsoft Gif Animator animatsiya grafika redaktori va web-sahifa yaratish uchun mo‘ljallangan dastur Microsoft Front Page bilan birgalikda qo‘yiladi. Bu dasturlarning fazilatlari: agar oldin web-saytning sahifalarida soatlab o‘tirilgan bo‘lsa, xozir esa ular sanoqli daqiqalarda qilinadi.

Adobe Photoshop – eng mashhur va rastr grafikada eng zamonaviy profeScional redaktordir. Uning sohasi – bu skaner qilingan fotosuratlarning tayyor tasvirlarni qayta ishlashdir. So‘nggi versiyalarida Web-grafika bilan ishlovchi komponentlar qo‘shilgan. U Adobe firmasining boshqa dasturlari bilan birgalikda eng talabchan so‘rovlarni qoniqtira oladigan dizeynerlik dasturlarining integrallangan paketini tashkil qila oladi.

Macromedia Flash – vektor animatsion grafikaning redaktori. Internetga joylashtiriladigan kichik o‘lchamli animatsion filmlarni tayyorlash imkonini beradi.

CorelDRAW – profeScional grafik redaktorlari ichida e’tirof etilgan etakchidir. Agar siz vektor redaktorlari bilan jiddiy shug‘ullanmoqchi bo‘lsangiz SorelDRAWni tanlang. Mutaxasislik mavqeingizni ko‘taradi. Redaktor rastr va vektor grafikani yaratish va qayta ishlash, Web-dizayn, verstka, rang taqsimlash, yangi shriftlarni ishlab chiqish, shtrix-kodlarni kiritish masalalarini a’lo darajada uddalaydi. Poligrafiya uchun eng yaxshi dastur xisoblanadi. Qo‘shimcha dasturlar, u bilan birga tarkatiladigan, animatsion vektor grafikasi bilan ishlashni ta’minlaydi. Bu redaktor ko‘p sahifali xujjatlarni qo‘llab quvvatlaydi. Lekin yuqori quvvatlilik natijasi kompyuterning resurslariga talabchanlik hisoblanadidi.

SHunday qilib, hozirgi vaqtda eng yaxshi grafik redaktorlar quyidagilar hisoblanadi:

- Vektor grafikaning redaktori – SorelDRAW.
- Animatsion vektor grafikaning redaktori – Macromedia Flash.
- Web-sahifa redaktori – Microsoft FrontPage.
- Rastr grafika redaktori – Adobe Photoshop.

Yuqorida sanab o‘tilganlarning har biri kompyuter grafikasining biror bir sohasida etakchi o‘rinlarni egallaydi. Lekin bu firmalarning har biri raqobatchilar bilan kurashish quroli sifatida diversifitsirovanno‘e redaktorlarni ham chikardi. Bu erda priverjens(tarafdor)lar qandaydir Mahsulotning ukalaridan ham foydalanishlari hisobga olingan. Barcha firmalar rastr, vektor, animatsion grafika dizaynerlik redaktorlarining va web-sahifalarni ishlab chiquvchi vositalarining to‘liq jamlanmasini taklif qilishi mumkin. Microsoft interfeysning komfortlilik darajasi bo‘yicha etakchi hisoblanadi. SHuni ishonch bilan etish mumkinki, web-sayt yaratish uchun FrontPage dasturi eng yaxshi vosita bo‘ladi. Uning soddaligi, qulayligi, kirishimliligi, ofis ilovalari bilan moslashuvchanligi shu bilan qatiylashmoqdaki, ayni vaqtda Microsoft Internet Explorer eng keng tarqalgan brauzer mavqeini egallab bormoqda.

CorelDRAW redaktorining interfeysi Bat uchun sodda va tushunarli bo‘ladi, ingliz tilida podskazkalar(yo‘l yo‘riqlar) va to‘lik ma’lumot tizimi berib o‘tilgan. SorelDRAW redaktorining murakkabligi uning imkoniyatlari ko‘pligidandir.

Adobe mahsuloti tushunarli podskaz(yo‘l yo‘riqlar)larning deyarli yo‘qligi va to‘liq bo‘lmagan ma’lumotnomasi bilan ajralib turadi, lokal versiyasi esa, afsuski, odatda kechikib paydo bo‘ladi (rasmiy lokalizatsiya haqida gap ketmoqda, bundan tashqari Internetda siz rusifikatsiya(ruslashtirish) uchun patchey to‘plamini topasiz, ularning yaratuvchilarini qonuniy harakatlarini chetga sursak ham, ularning sifatini yaxshiligi qoldiriladi, ular dastur ishlarini stabilnost(mustahkam)ini qoida bo‘yicha yomonlashtirishadi). Bu firmaning Mahsulotlarida masalan SorelDRAW foydalanuvchilariga mumkin bo‘lgan ba’zi bir zo‘r effektlar yo‘q, lekin hisoblovchi resurslar kichik hajmni talab qiladi. SHuni ta’kidlash kerakki, Page Maker mashhur tizim verstkasi xuddi shu firmada ishlab chiqariladi.

Macromedia firmasi oddiy deb nomlash mumkin bo‘lgan redaktorlarni ishlab chiqaradi, ingliz tilida podskazka(yo‘l yo‘riqlar)lar va tushunarli ma’lumotnoma berib o‘tilgan. Macromedia Flash animatsion grafikasi redaktori raqobatchilaridan kuchliroqdir. Uning murakkabligi imkoniyatlari ko‘pligidan kelib chiqqan.

Ikki va uch o‘lchovli geometrik almashtirishlarni mashina grafikasida qo‘llanilishini [1, 5-10, 20] adabiyotlari asosida ko‘rib chiqamiz.

Fraktal grafika dasturlari

1997 yili kompyuter bozorida ajoyib voqe’lar bo‘lib shtdi. Grafika uchun taniqli professional dastur ta’minoti ishlab chiqaruvchilar (Adobe, Macromedia, Autodeck, Corel, Microsoft) orsida grafik dasturlar ta’minoti bozorining bir qismini egallash imkoniyatiga ega bo‘lgan iste’dodli Meta Creations Inc kompaniyasi paydo bo‘ldi. Bu kompaniya ikki pog‘anali (2D)va uch pog‘anili (3D)grafika doirasida mutahassislanayotgan bir necha jamoa loyhachilari qo‘shilishi natijasida tashkil topgan edi. E’tiborga sazovarlarida to‘xtalib o‘tamiz.

Bu birlashmaga kirga firmalardan biri- Meta Tools o‘zining asoschisi Kaem Krauz hamda ularning mahsulotlari-rastrli (Ki’s Power Tools)va vektor (Vector Effects Effects)grafika raqamli videoga ishlov berish uchun dastur modullari (Studio) va uch pog‘anali KPT Brece landshaft generatori paketlari uchun filtrlar to‘plami (plug-ins)bilan mashhur.

Ikkinchi firma-Fractal Design kompyuter taomiliga birinchi marta musavvirlarning “tabiiy” instrumentlarini taqlid etish imkoniyati bo‘lgan Natural Media tushunchasini kiritdi. Faraz qiling, Siz vertual matoda (ya’ni, kompyuter sichqonni gilam ustida yuritib natijasini monitor ekranida ko‘ryapsiz)Painter dastur asbobi bilan Oil rejimida (Mo‘yqalam moyli bo‘yoq bilan)yashil chiziq chizyapsiz. So‘ngra uning yonida shu asbob bilan qizil rangdagi chiziq chizdingiz. Natijada ranglar birlashgan erda elektron ranglarning xuddi musavvir palitrasida yoki matoda ranglar aralashayotgandek ko‘rasiz. Ajablanarli emas Fractal DesignPainter, Expression va Ray Dream Studio dasturlari kompyuter grafikasiga sa’at sifatida katta turtki berdilar.

Aynan shuning uchun fraktal grafikasi bo‘yicha so‘nggi paytlaracha Meta Creations karvon boshi edi. Uning mahsulotlar spektri kompyuter grafikasining ko‘pgina sohalarini egalladi.

-**Fractal Design Painter**-yuqori sifatli badiiy rastrli illustrasiyalarni yaratish va ishlov berish dasturi. U AdobePhotoshop dasturi filtrlaridan foydalanishni va

tasvirlarning ko‘p qatlamligini qo‘llaydi. Ushbu dastur katta sonli badiiy asboblardan: qalamlar, mo‘yqalamlar, turli rangdagi bo‘yoqlardan foydalanish imkonini beradi. Bugungi kunda Fractal DesignPainter-o‘z ijodlarida kompyuterdan foydalanuvchi musavvirlar uchun “birinchi raqamli” dastur. Ishda maksimal qulaylik uchun grafikli planshetdanfoydalanish tasviya etiladi chunki, u sichqonga nisbatan mo‘yqalam harakati yo‘lini aniqroq ko‘rsatadi.

-**Fractal DesignExpression-** o‘zida kompyuter grafikasining rastr va vektor texnikasini kombinatsiyalaydi. Ya’ni Siz vektor obektlarini Copel DRAW Adobe Illustratordagi kabi chizaasiz, tayanch uzellar bo‘yicha tahrirlaysiz va barcha vektoroperatsiyalarini bajarasiz. Ammo har bir chiziq va shakilga Siz istalgan turdagi mo‘yqalamni belgpilashingiz mumkin. Bunda barcha real rastrli badiiy asboblar va bo‘yoqlar imitatsiyalanadi (taqlid), ishningt natijasi esa vektorli tasvir.

-**Fractal DesignDetailer 3D-**modellari yuzasini bo‘yash imkonini beradi.

-**Fractal DesignPoser 2D** tasvirini ,3D-sahnani, web grafikani va animatsiyani integratsiyalash imkonini beradi.

-**Art Dabbler**rasm chizishni o‘rgatish uchun a’lo darajadagi vosita hisoblanadi.

-**Add Depth** 3D –bezaklar, matnlar va boshqa 3Deffektlarni yaratishda foydalaniladi.

-**KPT-**AdobePhotoshop, Illustrator, Macromedia Free Hand grafik dasturlarning standart imkoniyatlarini kengaytirish uchun dastur modullri (filtrlar) to‘plami. Filtrlarning ishlashi tasvirlarni fraktal yaratishning matematik mexanizmi asos qilib olingan.

-**Painter3D-** illustrasiya va teksturalarni 3D modelliga qo‘yish uchun foydalaniladi. Illustrasiya va teksturalar dasturning o‘zida tayyorlanishi mumkin yoki AdobePhotoshop va Fractal DesignPainter dasturlaridan import qilinishi mumkin.

-**Bryce dasturida** kompyuter grafikasi uchun yangi yo‘nalish-uch pog‘analitabiiy landshaftlarni yaratish amalga oshirilgan. Uning yordamida tasvirlarni tabiatning shunday xodisalari tuman, oftob va oy yorug‘ligi, nurlarning ko‘p qaytarilishi va sinilishlarni yaratishda foydalanish mumkin.

1999-2000 yillarda Copel korporatsiyasi Meta Creations kompaniyasidan uch seriya: Painter, KPT va Bryce grafik paketlarni sotib oldi.Kelishuvlarga asosan Copel o‘z ixtiyoriga Painter, PainterClassic, Painter 3D, Art Dabbler,KPT, KPT-X, Vector Effects vaBrycedastur ta’minotlarini oldi. Sotib olingan dastur ta’minotlari Windows va Mac OS platformalarida ishlaydi. Copel kompaniyasining prezdenti Maykl Kouplend ta’kidlaganidek yaqin yillarda Copel loyihachilari uchun Meta Creations grafikli paketlarni multi tilli qo‘llash bo‘ladi hozirgi paytgacha ingliz, fransuz, nemis va yapon tillarini biladiganlar foydalanishi mumkin edi. Undan tashqari, Meta Creations kompaniyasidan sotib olingan Linux-versiyasi paydo bo‘lishi rejalashtirmoqda.

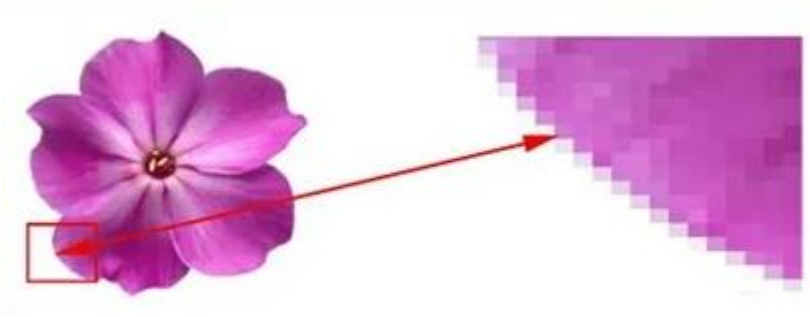
1.2. Rastr grafikasi.

Rastr grafikasi asoslari, rastr grafikasi turlari, amaliy dasturiy ta'minoti, rastr grafikasi formatlari.

Rastr tasvirlar to‘g‘ri burchakli matritsa shaklida namoyon bo‘lib, har bir yacheykasi rangli nuqtadan iborat.

Rastr grafikasining asosi piksel (nuqta) hisoblanib, u rang bilan ifodalanadi.

Tasvir turli rangdagi va yorug‘likdagi to‘rtburchak qism (piksel)lardan tashkil topgan bo‘lishi mumkin. Bunday turdagi tasvirlar rastrli tasvir deyiladi (1-rasm)



1-rasm. Tastir tasvir

Rastrli tasvirlar har bir katagi ranglangan katakli qo‘ozni eslatadi. Piksel (inglizcha pixel – picture element) – rastrli tavirlarning asosiy elementi hisoblanib, tasvir aynan shu elementlardan tashkil topadi. Rasmdan ko‘rinib turibdiki rastrli tasvir uzunligiga qandaydir sondagi piksellar joylashgan bo‘ladi. Bu kattalik razresheniye (разрешение) deb ataladi va odatda bir duymdagi piksellar soni bilan o‘lchanadi (pixels per inch, ppi). Ayrim hollarda ppi o‘lchovi o‘rniga unga yaqin bo‘lgan dpi (dots per inch - bir dyumdagi nugtalar soni) o‘lchovidan ham keng foydalaniladi. Tasvir nuqtalar to‘plami sifatida akslanib ular qanchalik ko‘p bo‘lsa ko‘rinish shunchalik tiniq va sifatli, fayl esa ko‘p joy egallaydi. Ya‘ni, aynan bitta tasvirning o‘zi yuqori yoki past sifatli bo‘lishi, o‘lchov birligiga qarab nuqtalar ko‘p yoki kam (odatda bir dyuymga nisbatan nuqtalar soni – dpi yoki piksellar soni – ppi bilan belgilanadi) bo‘lishi mumkin.

Tabiiyki, nuqtalar soni qancha ko‘p bo‘lsa (ular zich qilib joylashtirilsa), unga asoslangan rasm, shakl, grafik va hokazolar shuncha aniq ko‘rinib turadi. Shu

munosabat bilan ekranning ruxsat etish qobiliyati tushunchasi kiritilgan bo‘lib, unda gorizontal va vertikal yo'nalishlardagi nuqtalar soni muhim ahamiyatga ega va u ekranning ruxsat etish imkoniyati deyiladi.

Odatda, bunday ko‘rsatkich 640 x 480, 800 x 600, 1024 x 768 yoki bulardan yuqori piksellarda beriladi. Tasvir o’lchovi ruxsat etish qobiliyati bilan bogliqdir. Bu parametr dpi (dots per inch - nuqtalar soni zichligi) bilan o‘lchanadi. Ekran diagonal! 15 duymli (1 duym =2,54 sm) monitorda tasvir o‘lchovi 28x21 sm ni tashkil qiladi. Buni hisobga olsak, 800 x 600 pikselli monitor ekranining tasvirlash qobiliyati 72 dpi ga teng bo‘ladi. Demak, kompyuter xotirasidagi rangli tasvir ko‘p joy olishini tushunish qiyin emas. Misol uchun 10x15 sm li rasm taxminan 1000 x 1500 piksellardan iborat bo‘l adi.

Agar har bir rangli nuqtani tasvirlash uchun 3 bayt xotira ishlatilishini hisobga olsak, bitta o‘rtacha kattalikdagi rasmning o ‘zi xotirada taxminan 4 mln bayt joyni egallaydi. Bunday ma’lumot, xususan, Internet sahifalarini yaratishda e’tiborga olinishi zarur. Shuning uchun ham hozirgi kunda yaxshi multimedia dasturlarini, videoroliklarni yaratish uchun operativ xotirasi 128 Mbaytdan kam bo‘lmagan va, mos ravishda, tezligi katta bolgan kompyuterlardan foydalanish maqsadga muvofiq. Demak, rastrli grafika bilan ishlash uchun yuqori unumli kompyuter talab qilinadi.

Rastrli grafikaning kamchiligi sifatida shuni aytish mumkinki, tasvirni mashtablashtirish (kattalashtirish, kichiklashtirish) jarayoni natijasida nuqtalar o‘lchovi kattalashishi bilan tasvir aniqligi yomonlashishi mumkin va hatto, tasvir tanib bo‘lmaydigan darajada o'zgarishi mumkin. Rastrli grafika elektron (multimedia) va matbaa nashrlarida keng qo‘llaniladi. Nashrlarda turli illustratsiyalarni yaratishda, odatda, skaner orqali olingan raqamli foto yoki videokamera (hozirda bunday fotoapparat va videokameralar keng tarqalgan) yoki rassom, loyihachi tomonidan tayyorlangan tasvirlardan foydalaniladi. Shuning uchun ham rastrli grafikada tahrir qiluvchi dastur vositalaridan keng foydalaniladi. Bu dasturlar, odatda, tasvirlarning aniqroq ko‘rinishda bo‘lishini ta ’minlaydi.

Rastrli (nuqtali) grafikada tasvirlar nuqtalardan hosil qilinadi. Shuning uchun uning asosiy tushunchasi - "ruxsat" (bir birlik uzunlikka to‘g‘ri keladigan nuqtalar soni) bo‘lib, uning quyidagi shakllari mavjud:

- ✓ Originalga ruxsat;
- ✓ Ekranda tasvirga ruxsat;
- ✓ Qog‘ozga chiqarishga ruxsat.

Original (asl)ga ruxsat 1 duymdagi nuqtalar bilan o‘lchanib, kompyuterga kiritilayotgan tasvir sifati fayl kattaligiga, kodlashtirish usuli (tayanch shaklni hosil qilish usuli) va boshqa parametrlarga bog‘liq bo‘ladi. Tasvir sifatiga qo‘yilgan talab qancha katta bo‘lsa original (asl)ga ruxsat shuncha katta bo‘ladi.

Ekranda tasvirga ruxsat deganda biz monitor (ekran)da hosil bo‘ladigan tasvirning parmetrlarini tushunamiz. Tasvir nusxasini ekranda hosil qiluvchi oddiy nuqta piksel deyiladi. Piksel kattaligi ekranda tasvirga joizlik bilan original (asl)ga joizlik orasidagi masshtabga bog‘liq. Diagonali 20-21 duymli ekranlarning rastrlari 640*480, 800*600, 1024*768, 1280*1024, 1600*1200, 1600*1280, 1920*1200, 1920*1600 standart nuqtali bo‘lgandagina tasvir chiqarishga ruxsat berilgan.

Ekrandan nusxa olish uchun (ekran tasvirini printerda oddiy qog‘ozga chiqarish uchun) 72 dpi, rangli yoki lazerli printerda tasvir hosil qilish uchun 150-200 dpi, fotoeksponentlovchi qurilma uchun 200-300 dpi joizlik kifoya.

Tasvir parametrlari va fayl hajmi orasidagi bog‘lanish

Rastrli grafika bo‘yicha hosil qilinadigan tasvirlar yuqori aniqlik va ranglar jilosidan foydalanishni talab qiladi. Ammo bunday tasvirlarning aniqligini oshiruvchi "ruxsat”ning kattalashuvi fayl hajmining keskin oshishiga olib keladi. Masalan, 10*15 santimetrli fotosurat 200-300 dpi ijozatli tasvir TIFF formatda 4 Mbayt hajmga ega. A4 formatli rangli surat 120- 150 Mbayt bo‘ladi.

Rastrli tasvirlarni masshtablashtirish

Agar maxsus choralar ko‘rilmasa, rastrli tasvirlarni kattalashtirish tasvirning asl holati buzilishiga olib kelishi mumkin. Chunki rastrli grafikada tasvirni aniq sondagi nuqtalar (piksellar) hosil qiladi. Shaklni kattalashtirganda nuqtalar sonini oshirish m

um kin b o ‘lmaganligi sababli ba’zi bir buzilishlar hosil bo‘lishi mumkin. Bu hol ro‘y bermasligi uchun orginal (asl) tasvirni oldindan raqamlashtirish kerak bo‘ladi. Bu usulga o‘xshash boshqa usul, ya’ni ma’lum diapazonda piksellashtirish effektini kamaytiruvchi stoxastik rastrlashni qo‘llash kerak bo‘ladi va nihoyat masshtablashtirishda interpolyatsiya usuli oraliq qo‘shimcha nuqtalarni qo‘shish orqali tasvir kattaligini oshirish keng qoilaniladi.

Rastrli grafikani tahrirlaganda biz piksellarni tahrirlaymiz, ya‘ni ularning rangini o‘zgartiramiz. Rastrli grafika asosan sifat sig‘imi (разрешение)ga bog‘liq, chunki tasvirni ifodalovchi ma‘lumot ma‘lum bir o‘lchamdagi to‘rga bog‘langan bo‘ladi. Rastrli tasvir tahrirlanganda uning sifati o‘zgarishi (odatda pasayishi), ya‘ni tasvirni o‘zini emas, balki u joylashgan to‘rning o‘lchamlari o‘zgartirilib tasvirning aniqlik, tiniqlik darajasi o‘zgartiriladi va fayl egallagan hajmi kamaytiriladi. Rastrli grafikani tasvirning sifat sig‘imiga nisbatan past bo‘lgan qurilma (printer)da chop ettirish rasm sifatini pasaytishiga olib keladi. Demak rastrli grafikani chop etish qurilmasining sifat sig‘imi tasvir sifat sig‘imiga nisbatan yuqori bo‘lishi tavsiya etiladi.

Rastrli grafikaning keng tarqalgan formatlari: *.tif, *.gif, *.jpg, *.png, *.bmp, *.pcx va boshqalar.

Afzalliklari:

- ✓ Rastr grafikasi real obrazlarni effektiv namoish eta oladi. Sifatli rastr tasvirlari foto surat darajasidagi yuqori aniqlikda real va haqqoniy aks ettirilishi mumkin.
- ✓ Rastr tasvirlar chiqarish qurilmalari – asosan lazer printerlarida juda yaxshi chop etiladi. Ya‘ni rastr grafikasining sifati chop etishda o‘zgarmaydi.

Kamchiliklari:

- ✓ Rastr tasvirlari saqlash qurilmalarida (qattiq disk, CD-DVD, fleshka va h.) ko‘p joy egallaydi;
- ✓ Rastr tasvirlarni tahrirlashda kompyuterning xotira resurslari – xususan tezkor xotirada ko‘proq joy talab etiladi;
- ✓ Rastr tasvirlarni tahrirlash mehnat talab qiladi va mashaqqatli;
- ✓ Rastr tasvir masshtablashtirilganda tasvir sifati o‘zgaradi.

Qo‘llanish sohasi:

- Elektron va poligrafik nashriyotlar, Web dizaynda;
- Foto tasvirlarni tahrirlash va restavratsiya qilishda;
- Badiiy grafika va skaner bilan ishlashda.

Rastrli grafika muharrirlari

Rastrli grafika muharrirlariga misol qilib Adobe Photoshop va Paint dasturlarini aytish mumkin. Ushbu dastrularda rasmlar mayda kvadrat-piksellardan iborat bo‘lib mozaika holatida rasmni hosil qiladi. Rastrli grafikadan raqamli fotosuratlar va skanerdan olingan rasmlar bilan ishlash uchun foydalaniladi. Kompyuter grafikasida duymdagi piksellar soni (dpi) asosiy shart bofiadi. Piksellar soni qancha ko'proq bo‘lsa, tasvir shuncha sifatliroq bo‘ladi. Masalan, agar dpi =72 bo'lsa, u holda 1 kvadrat duymga 5184 piskel joylashadi va uning hajmi 6 Kb bo‘ladi, agar dpi = 144 bo‘lsa u holda 1 kvadrat duymga 20736 piskel joylashadi va endi uning hajmi 21 Kb ga teng bo‘ladi.

Shu bilan birga monitorning ko‘rsatish va printerning chiqarish sifati - duymga piskellar soni (dpi) (72 yoki 96 dpi) va duymga chiziqlar soni (Ipi) (300-2400 dpi lazerli, sepuvchi printerlar uchun va 75-200 Ipi matrisali printerlar uchun) hamda kompyuter ranglar sifati (2, 16, 256, 32 000, 16 000 000 ranglar soni) ham katta ahamiyatga ega bo‘ladi.

Rang holatlari - ranglarni chiqarish va ko‘rsatish yo‘li. Rang holatlari 2 xil bo‘ladi:

- ✓ RGB (qizil, yashil, ko‘k) monitorlarda tasvir ko‘rsatishda foydalaniladi;
- ✓ CMYK (havorang, purpur, sariq, qora) bosmada foydalaniladi.

RGB holatidagi ranglar soni CMYK holatiga qaraganda ko‘proq.

Paint pikselli tasvirlar muharriri

Paint pikselli tasvir muharriri bo‘lib, mazkur dastur turli xil rasm va shakllarini hosil qilish va qayta ishlashda foydalaniladi. Unda hosil qilingan tasvir boshqa amaliy dasturlarda qo‘llanilishi mumkin. Aytib o‘tilganidek, dastur ko‘magida oddiy matnli jadval va diagrammalar hamda yuksak saviyali san’at asarlarini yaratish mumkin. Dastur yordamida Windowsning boshqa dasturlarida yaratilgan ixtiyoriy

matn yoki grafik ma'lumotlar nusxasini olish yoki skaner qurilmasi yordamida o‘ta qiyin talqindagi san’at asarlaridan nusxa olish, tahrir qilish va chop qilish ishlari majmuini bajarish mumkin.

Adobe Photoshop dasturi

Adobe Photoshop - Adobe Inc. firmasi tomonidan ishlab chiqilgan va tarqatilyotgan ko‘pfunksiyali grafik redaktor. Asosan rastrli tasvirlar bilan ishlashga mo‘ljallangan, biroq bir nechta vektorli vositalariga ega.

Dastur Adobe firmasi mahsuloti sifatida mashhur va rastrli tasvirlarni tahrirlashda dunyoda eng oldi brendi hisoblanadi. Hozirda Photoshop macOS, Windows platformalariga, iOS, Windows Phone va Android mobil tizimlariga moslashtirilgan. Yana Windows 8 va Windows 8.1 uchun Photoshop Express versiyasi ham mavjud. Adobe Photoshop tasvirlarni tahrirlashdagi proffesional redaktor hisoblanadi.

Dastur afzalliklari

Adobe Photoshop dasturi 20 dan ortiq formatdagi fayllar bilan ishlash imkoniga ega. Ular: PSD, PSB, PostScript, EPS, DCS, EPS TIFF, BMP, GIF, JPEG, TIFF, PICT, PNG, PDF, ICO, IFF, PCX, RAW, TGA, Scitex CT, Filmstrip, FlashPix, JPEG2000 va boshqalar. Eng ko‘p qo‘llaniladigan formatlar:

BMP (Windows Bit map - Windows ning vit xaritasi) - Windows muhitida ishlovchi kompyuterlarda ekran osti tasvirlarini qo‘llovchi dastur Microsoft Paintda keng qo‘llaniladi.

JPEG, JPG (Joint Phonographic Experts Group) - hozirgi kunda eng ko‘p qo‘llaniladigan formatlardan biri bo‘lib, uning asosiy afzaliklaridan biri maxsus dastur yordamida fayl hajmini yetarlicha siqish imkonining mavjudligidir. Ammo faylni siqib, hajmini kichraytirish jarayonida tasvir sifatida o‘zgarish bo‘ladi. Fayl kuchli siqilganda tasvir sifati yomonlashishi mumkin. Ushbu formatdagi fayllar kompyuter xotirasida ko‘p joy egallamaydi va hajm jihatidan kichikligi bois mazkur formatdagi tasvirlar bilan ishlash ancha oson.

TIFF (Tagger Image File Format) - bu formatdagi fayllar ham keng qo‘llanadi. Lekin Tiff formatidagi fayllar kompyuter xotirasida ko‘p joyni egallaydi. Adobe

Photoshop dasturida ushbu formatdagi tasvirlar bilan ishlashda dasturining ishlash tezligi sezilari ravishda kamayishi mumkin.

RGB (red, Green, Blue - qizil, ko‘k, yashil) moduli tasvirni ekranda tahrir qilish nuqtai nazaridan kelib chiqqan holda juda qulay va u 24 razryadli ranglar platasi yordamida deyarli barcha 16 million ranglarni monitorlarda aks ettiriladi.

RGB ranglar majmuasi bilan ishlagan barcha tasvirlarni xohlagan formatda diskka yozish mumakin. RGB ranglar majmuasidagi ayrim ranglar umuman tabiatda uchramaydi.

CMYK - tabiatda mavjud bo‘lgan ranglar majmuasi, quyosh nurlari inson ko‘zlari ajrata oladigan barcha ranglarni o‘ziga mujassamlashtirgan. Quyosh nurlari biror - bir jismga tushganda uning ta’siri ostida inson ko‘zlari jism shakli va rangini idrok etadi. Misol uchun, binolarning o‘t o‘chirish burchaklariga osib qo‘yilgan o‘t o‘chirgichlar to‘q ko‘k va zangori ranglar bilan bo‘yalishiga qaramay, bizning ko‘zimizga to‘q qizil rangda ko‘rinadi. Ranglarni bir biriga qo‘shilishi natijasida boshqa ranglar hosil qilinadi: S - xavorang, M - binafsharang, Y - sariq rang, K - qora rang. RGB ranglar majmui keng ko‘lamdagi ranglarni taklif etadi. Lekin ularning ko‘p qismi (ayniqsa, yorkinlari) tasvirni chop etganda monitordagi bilan keskin farq qiladi. Shu bois xam ko‘plab mutaxassislar tasvirni chop etishdan avval uni CMYKsistemasiga o‘tkazadilar.

Photoshop mediafayllar, animatsiya va boshqa turdagi ijod namunalarini qayta ishlovchi dasturlar bilan aloqadorlikka ega. Adobe ImageReady, Adobe Illustrator, Adobe Premiere, Adobe After Effects, Adobe Encore DVD dasturlari bilan hamkorlikda Photoshop proffesional DVD yaratilishida, turli darajadagi maxsus effektlarni yaratishda, televideniya, kinomotografiyada va o‘rgimchak turida ishlatilishi mumkin.

Hozirda kompyuter o‘yinlarini yaratishda ham Photoshop keng ishlatilyapti. Photoshopning asosiy formati PSD yuqorida nomi keltirilgan barcha dasturlarga import va eksport qilinishi mumkin. PS CS DVDlarda menyu hosil qilish funksiyalariga ega.

Dasturning juda mashxurligi sabab PSD formati Adobe Fireworks, Photo-Paint, WinImages, GIMP, SAI, PaintShop Pro va boshqa grafik dasturlarda tan olinadi.

Photoshop quyidagi rang modellarini tan oladi va ishlaydi: RGB, LAB, CMYK, Kulrang tusi, Oq-qora, Duotone, 256 rangli palitra (Indexed), Ko‘p kanalli (Multichannel)

...

1.3. Vektor grafikasi.

Vektor grafikasi asoslari, Vektor grafikasi turlari, qo‘llanilish sohalari, amaliy dasturiy ta’minoti, vektor grafisi formatlari.

Vektorli tasvirlar deb – tuzilishi jihatidan murakkabroq va har xil ko‘rinishga ega bo‘lgan geometrik ob’yektlar to‘plamiga aytiladi.

Vektorli grafikada tasvirning asosiy elementi sifatida chiziq qaraladi.

Vektor grafikasida tasvir vektor deb nomlanuvchi chiziqlar asosida qurilib, ularga turli parametrlar – rang, chiziq qalinligi va joylashuvi (vaziyati) xususiyatlari beriladi.

Vektor grafikasida primitivlar deb nomlanuvchi ob‘ektlar bilan ishlanadi. Primitivlarga ikki va uch o‘lchamli oddiy geometrik figuralar kiradi. Ikki o‘lchamli geometrik figuralarga – nuqta, to‘g‘ri chiziq, egri chiziq, aylana, ko‘pburchak kabi tekis shakllar kirsa, uch o‘lchamli geometrik figuralarga – kub, prizma, piramida, sfera, konus, silindr kabi jismlar kiradi. Ushbu oddiy geometrik figuralar asosida murakkab bo‘lgan geometrik figuralar – ob’yektlar yaratiladi.

Vektor grafikasida asosiy mantiqiy element primitivlar bo‘lganligi uchun, asosiy e’tibor primitivlarni qurishda ularning parametrlariga qaratiladi. Misol uchun yopiq ko‘pburchak tomonlari teng yonli yoki ixtiyoriy bo‘lishi, yopiq hududlar aylana, ellips yoki ixtiyoriy egri chiziq asosida qurilishi mumkin bo‘lib ushbu yopiq ko‘pburchak yoki hudud ichini ranglash, gradiyentlash yoki shtrixlash mumkin.

Rastrli grafikada bunday chiziqlar nuqtalar (piksellar) yordamida yaratilsa, vektorli grafikada esa tasvirlarni yaratishda nuqtaga nisbatan umumiyroq bo‘lgan chiziqlardan foydalaniladi va shuning hisobiga tasvirlar aniqroq ko‘rinishga ega bo‘ladi.

Vektorli grafikaning afzallik tomoni tasvirning xotirada kamroq joy olishidir, chunki bu holda xotirada joy chiziq o‘lchoviga bog‘liq bo‘lmagan ravishda bo‘ladi. Buning sababi xotirada chiziqning o‘zi emas balki uni ifodalovchi formula yoki parametrlar saqlanishidadir. Vektorli grafikaning ixtiyoriy tasviri chiziqlardan tashkil topadi va oddiy chiziqlardan murakkablari hosil qilinadi.

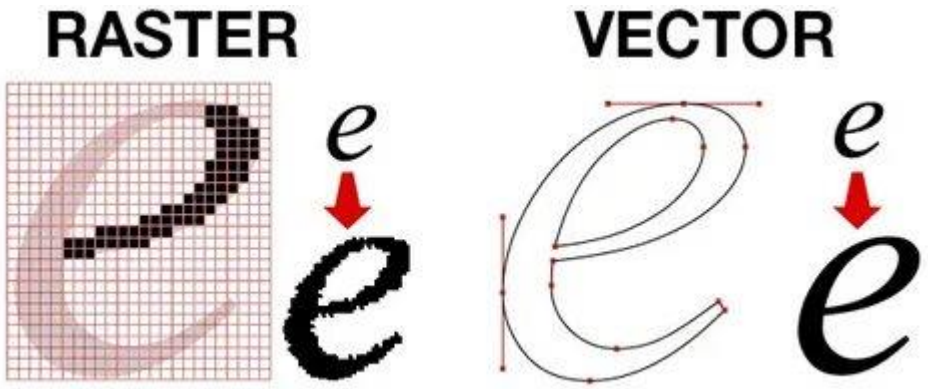
Ko‘pincha vektorli grafikani obyektga mo‘ljallangan grafika deyish mumkin. Chunki bunda, masalan, uchburchak hosil qilish uchun 3 ta chiziq (kesma)dan foydalanilsa, piramida hosil qilish uchun uni uchburchakdan foydalanibgina hosil qilish mumkin.

Vektorli grafikani hisoblanadigan grafika deb ham atash mumkin, chunki tasvirni (obyektni) ekranga chiqarishdan avval uning koordinatalari hisoblanadi va mos nuqtalar hosil qilinadi. Vektorli grafikaning matematik asosini geometrik figuralarning xossalarini o‘rganish tashkil etadi. Ma’lumki, nuqta tekislikda 2 ta (x,y) koordinatasi bilan, To‘g‘ri chiziq kanonik ko‘rinishida y = kx + b (bunda k\ab ixtiyoriy sonlar)da, kesma esa mos ravishda boshlang‘ich va oxirgi nuqtasini berish bilan tasvirlanadi. Egri chiziqlar ham mos ravishda o‘z tenglamalariga ega.

Vektorli grafika asosan illustratsiyalar yaratish uchun yo‘naltirilgandir. Reklama agentliklarida, loyihalash byurolarida, nashriyotlarda va boshqa joylarda keng qo‘llaniladi. Vektorli grafika bilan ishlaydigan dasturlarga misol sifatida Adobe Illustrator, Corel Draw va boshqa bir qancha dasturlarni keltirish mumkin.

Vektorli grafikalar piksellar soniga bog‘liq emas, ya’ni. sifatni yo‘qotmasdan turli xil piksellar soniga ega turli xil chiqish moslamalarida namoyish etilishi mumkin. Ammo, afsuski, ko‘p sonli soyalar yoki mayda tafsilotlar (masalan, fotosuratlar) bilan tasvirlarni uzatishda vektor formati noqulay bo‘ladi. Axir, bu holda har qanday eng kichik alanga monoxrom nuqtalar to‘plami bilan emas, balki har biri formuladan iborat bo‘lgan murakkab matematik formula yoki grafiklar to‘plamidan iborat bo‘ladi. Bu og‘irroq faylga olib keladi.

Raster va vektorli rasm o‘rtasidagi farq



Расм ….

Rasterli tasvirlar piksellar sonini koʻpaytirish orqali ishlab chiqariladi, ularning har biri tasvirni yanada toʻgʻri uzatish uchun oʻz rangiga ega.

Vektorli rasm uning qanday yaratilganligini va buning uchun qanday ranglardan foydalanilishini koʻrsatadigan matematik formulalar (vektorlar) oʻrnatish orqali ishlab chiqariladi.

Asosiy farq shundaki, bitmap rasmlari kattalashganda tashqi koʻrinishini saqlab qolmaydi - agar siz rasmni kattalashtirsangiz, u xiralashadi. Vektorli tasvirlar tashqi koʻrinishini saqlab qoladi ...

Grafik tasvirni yaratish usuliga qarab rastr, vektor va fraktal grafikalar ajralib turadi.

Raster grafikasi afzalliklari

- ranglarning silliq oʻtishi bilan murakkab tasvirlarni yaratish qobiliyati
- tarqalganlik
- murakkab rasmlarni qayta ishlash tezligi
- koʻpchilik kirish / chiqish qurilmalarining oʻqilishi

Raster grafikasi kamchiliklari

- kam oʻlchovlilik
- vektorli grafikaga muammoli tarjima qilish
- murakkabligidan qat'i nazar, katta hajmdagi rasm
- plotterda chop etolmaslik

Vektor grafikasining afzalliklari

- rasm kattalashganda, fayl hajmi oʻzgarmaydi
- a'lo miqyosi
- oddiy tasvirlar rastrli rasmlarga qaraganda ancha kam joy egallaydi
- vektorli grafikadan rastrga tarjima qilish qulayligi

Vektor grafikasining kamchiliklari

- vaqtni talab qiladi
- saqlash uchun xotira

- ba'zan esa murakkab tasvirlarni yaratish imkoniyati yo‘q
- ko‘plab qurilmalar bilan ishlashda qiyinchilik

Vektor grafikaning dasturiy ta’minoti

COREL DRAW- vektor grafikasining WINDOWS operatsion sistemasida ishlaydigan yangi grafiklar yaratish va tahrir qiluvchi dasturidir. Uning yordamida turli grafik ko`rinishlarni loyihalash, fotomatn, tasvirlar ustida ishlash, ayniqsa badiiy ko`rininshdagi kompozisiyalarni tahrir qilish bilan bog`liq amallarni bajarish mumkin.

Расм …

Adobe Illustrator - bu vektorli grafika bilan ishlashga mo‘ljallangan dastur. Ko‘pincha rasmlar, komikslar, logotiplar uchun ishlatiladi. Nuqtalar qatorida chizma haqidagi ma'lumotlarni saqlaydigan bit xaritalari bilan taqqoslaganda, Illustrator shakllarni chizishda matematik hisoblardan foydalanadi. Bu ravshanligi oshgani sayin grafikani sifatini yo‘qotmasdan kengaytira oladi.

Расм …

Adobe Flash - Adobe kompaniyasi tomonidan shunday dasturlaridan biri Flash dasturi yaratilgan bo‘lib, bu dastur texnik WEB - dizayn vositalarining to‘liq imkoniyatidan foydalanish imkoniyatini beradi. Dasturning imkoniyatlari juda keng bo‘lib, bunda harakatlar va tovushlar 100 kb fayl hajmnigina egallaydi. Flash quyidagi imkoniyatlarga ega: Yaratilayotgan faylni hajmi kichikligi va Flash

dasturining tarmoqdan tez yuklanuvchanligi. Flashda vektorli format qo‘llanganligi sababli, unda fayllar siqiladi va shuning uchun fayl hajmi kamayadi.

Расм. …

...

1.4. Fraktal grafika.

Fraktal grafikasi asoslari, Fraktal grafikasi tarixi, Fraktal grafikasi turlari, qo‘llanilish sohalari, dasturiy ta’minoti.

Fraktallar noyob ob’yektlar bo‘lib, xaotik dunyoning aytib bo‘lmaydigan darajadagi harakatlaridan paydo bo‘ladi. Ularni juda kichik bo‘lgan membrana hujayralaridan tortib, juda katta hisoblangan Quyosh tizimidan ham topish mumkin. Daraxtlarning barglari, qo‘lning vena qon tomirlari, to‘lqinli daryolar, qimmatbaho qog‘ozlarning bozori - bularning hammasi fraktallardir. Qadimgi zamon taraqqiyoti vakillaridan to hozirgi kunning taraqqiyoti vakillari, olimlar, matematiklar va artistlar, shuningdek yer yuzida yashovchi odamlar fraktallar bilan hayratlanganlar hamda ulardan o‘zlarining ishlarida foydalanganlar. Shuningdek, dasturchilar va kompyuter texnikasi sohasidagi mutaxassislar fraktallardan cheksiz murakkablikdagi go‘zallikni oddiy formulalar orqali kompyuterlarda qurishlari mumkin.

Fraktallarning ixtiro etilishi fan va matematikada, san’atdagi yangi estetikaning ochilishidir, shuningdek insonning olamni idrok qilishdagi kashfiyotdir.

“Fraktal” so‘zi bu biz yashab turgan kunda ko‘pchilik insonlar, fiziklardan tortib maktab o‘quvchisigacha gap yuritadigan tushunchadir. U ko‘plab darsliklar va ilmiy jurnallar muqovalarida hamda kompyuterlarning dasturiy ta’minoti qutilarida paydo bo‘ldi. Bugun fraktallarning rangli suratini hamma joyda uchratish mumkin: tabriknomalardan tortib futbolkalargacha. Oxirgi uch o‘n yillikda oyda chiqarilayotgan mahsulotlarning soni bir necha o‘nliklardan ko‘plab minglargacha o‘sdi. Shunday qilib, biz atrofimizda ko‘rib turgan barcha rangli shakllar bu nima? Oddiygina tilda aytish mumkin: *Fraktal - bu geometrik shakl bo‘lib, aniq bir qismi o‘lchamlari o‘zgargan holda qayta-qayta takrorlanishidir.*

Bu yerdan o‘ziga-o‘zi o‘xshashlik xususiyati kelib chiqadi. Barcha fraktallar o‘ziga-o‘zi o‘xshashdir, ular barcha darajalarda o‘xshashdir. Biroq fraktallar-murakkab shakllar bo‘lib qolmay, balki kompyuterlarda bo‘g‘imlarga bo‘lingan. Xulosa shuki, tasodifiy va tartibsiz harakatlar fraktallardir. Nazariy tomondan mavjud

olamdagi barcha narsalar ular bulutlarmi yoki kichkina kislorod molekulasi ularning hammasi fraktallardir.

Fraktallar xaos so‘zi bilan doimo bog‘langandir. Fraktallarni xaosning qismi sifatida aniqlash maqsadga muvofiqdir. Fraktallar tartibsiz va tasodifiy bo‘lishi bilan xaotik hatti-harakatlarni namoyon etadi. Agar juda yaqindan qaralsa fraktalning ichida juda ko‘p o‘ziga-o‘zi o‘xshashlik tomonlarni ko‘rish mumkin. Masalan, daraxtga qarang, bitta shoxni ajratib olib uni yaqindan o‘rganing. Endi bir necha barglarni bog‘lamlarini ajratib oling. Fraktallar bilan shug‘ullanuvchi olimlar (xaologlar) uchun bu uchta ob’yekt aynan o‘xshash ifodalanadi.

Xaos so‘zi ko‘pchilik odamlarning xayoliga tartibsiz va so‘z bilan ifodalab bo‘lmaydigan narsalarni olib keladi. Aslida bunday emas. Demak, xaos qanchalik xaotik? Javob shunday, haqiqatda xaos nima yetarlicha tartiblangan va aniq qonuniyatga amal qiladi. Muammo shundaki, bu qonunlarni qidirib topmoq juda murakkab. Xaos va fraktallarni o‘rganishdan maqsad - aytib bo‘lmaydigan va xaotik tizimdagi qonuniyatlarni bashorat qilishdir.

Xaologlar bulutli tasvirlarni, ob-havoni, suv oqimini, hayvonlarning ko‘chishini, shuningdek ona tabiat hayotidan ko‘plab aspektlarni o‘rganishni yoqtirishadi. Tizim - bu buyumlar to‘plami, yoki o‘rganish sohasi. Shunday qilib, bizni o‘rab turgan dunyo fraktallardan iboratdir.

Ko‘plab xaologlar uchun xaos va fraktallarni o‘rganish dunyoni bilishni yangi sohasi bo‘lib qolmay, matematiklar, nazariy fiziklar, san’at va kompyuter texnologiya sohasidagi mutaxassislarni birlashtiruvchi inqilobdir. Bu kashfiyot nafaqat darsliklarda ko‘radigan, balki tabiatda hamda cheksiz olamda bizni o‘rab turgan olamni ifodalab beruvchi geometriyaning yangi turidir.

Bu yangi sohani o‘rganuvchilar fraktallar nazariyasining otasi deb Fronka-Amerika matematigi professor Benua Mandelbrot (Fransiyada tavallud topgan) deb hisoblaydilar. 1960 - yillarning oxirgi o‘n yilligida Mandelbrot ishlagan ilmiy ijodini *fraktal geometriya* yoki *tabiat geometriyasi* deb ataydi (bu haqida u o‘zining Tabiatning fraktal geometriyasi - “The fractal geometry of nature” nomli asarida yozadi). Fraktal geometriyaning maqsadi - sindirilgan, burishgan va noravshan

shakllarni tahlil qilishdan iborat. Mandelbrot parchalangan va qismlardan tashkil topgan bu shakllar uchun fraktal so‘zidan foydalangan.

Mandelbrot boshqa olimlar Klifford A.Pikkover (Clifford A.Pickover), Djeyms Gleyk (James Gleick) yoki G.O.Peytgen (H.O.Peitgen) fraktal geometriyaning sohasini kengaytirishga harakat qiladilar, ya’ni butun dunyoda ularni amaliy qo‘llashga, bozordagi qimmatli qog‘ozlarni narxlarini bashorat qilishdan tortib nazariy fizikaning yangi kashfiyotlarini bajarishgacha.

Fraktallar fanda ko‘pdan-ko‘p qo‘llanilmoqda. Buning asosiy sababi u mavjud borliqni an’anaviy fizika yoki matematikaga nisbatan juda aniq bayon etadi. Bir necha misol keltiramiz: kompyuter grafikasi, kompyuter tizimlari, telekommunikatsiya, radiotexnika, kino, suyuqlik mexanikasi, sirtlar fizikasi, astronomiya, tibbiyot, biologiya, musiqa, san’at sohasi va boshqalarda.

Fraktal geometriyaning asosiy g‘oyalaridan biri borliqda o‘lchovlar miqdori uchun butun bo‘lmagan qiymatlar g‘oyasidir. Mandelbrot butun bo‘lmagan o‘lchov 2.76 ni fraktal o‘lchov deb nomladi. Oddiy Yevklid geometriyasi mavjud borliq tekis va silliq ekanligini ta’kidlaydi. Bunday borliqning xususiyati nuqtalar, chiziqlar, burchaklar, uchburchaklar, kublar, sferalar, tetraedrlar va boshqalarni beradi.

Tabiatdagi ko‘plab ob’yektlar (masalan, inson tanasi) biri ikkinchisi bilan qo‘shilgan fraktallar to‘plamidan tashkil topgan bo‘lib, har bir fraktal boshqalarining o‘lchamidan farq qiladigan o‘zining o‘lchamiga ega. Masalan, insonning ikki o‘lchamli sirtidagi tomirli tizimlari egiladi, tarmoqlanadi, buraladi va qisiladi, uning fraktal o‘lchami 3.0 ga teng. Ammo agar u alohida bo‘laklarga bo‘lingan bo‘lsa, arteriya qon tomirining fraktal o‘lchami faqatgina 2.7 ga teng bo‘ladi, unda o‘pka branxial yo‘lidagi fraktal o‘lcham 1.07 ga teng.

Ma’lumki hozirgi vaqtda fraktallar kompyuter grafikasi, fizika, va boshqa turli tabiiy fanlarda keng qo‘llanilmoqda, shuningdek radiotexnikada antennalarni loyihalashda, telekommunikatsiyada signallarni qayta ishlashda, kino hamda televideniyada maxsus effektlar va vizualizatsiya elementlari sifatida, yengil sanoatda gazlama va gilamlarga zamonaviy dizaynlar uchun naqshlar chizishda va h.k.

Fraktallarning tarixi. Fraktal so‘zi lotincha “fractus” so‘zidan olingan bo‘lib, “bo‘laklangan”, “qismlardan tashkil topgan” degan ma’noni anglatadi va u “fraction, fractional” (bo‘luv, bo‘linma) terminlaridan kelib chiqqan. Hozirgi kunga qadar fraktal tushunchasi aynan ta’rifga ega emas, biroq matematik nuqtai nazardan fraktal-bu kasrli o‘lchamlar to‘plamidan.

Fraktal va fraktal geometriya tushunchasi 20-asrning 70-80 yillari o‘rtalarida matematiklar hamda dasturchilarning ilmiy izlanishlariga qat’iy ravishda kirib keldi.

Fraktallarning ta’riflari. Quyida fraktallarga berilgan ta’riflarni keltiramiz.

Yuqorida aytib o‘tganimizdek, fraktal aniq bir ta’rifga ega emas, biroq adabiyotlarda unga berilgan turlicha ta’riflarni uchratamiz.

Fraktal-bu geometrik fraktal bo‘lib, qismlardan tashkil topgan hamda ularning har biri butun fraktalning nusxasini kichiklashtirgan holatini ifodalaydi.

Fraktal-aniq bir qism o‘lchamini o‘zgartirgan holda qayta va qayta takrorlovchi geometrik shakldir.

Fraktal-qismlardan tashkil topgan, qaysidir ma’noda to‘laligicha o‘ziga-o‘zi o‘xshash tuzilishdir.

Fraktal-bu singan fazoviy shakl, tekis yoki notekis, xaotik yoki botartib va o‘ziga-o‘zini turli mashstabda takrorlaydigan murakkab tuzilish hisoblanadi.

Fraktal masshtabiga bog‘liq bo‘lmagan tasvirlarning o‘ziga-o‘zi o‘xshash tuzilishlaridir.

Fraktal-Xausdorf o‘lchami topologik o‘lchamidan qat’iy katta bo‘lgan to‘plam.

Fraktal-nobutun o‘lchamli o‘ziga-o‘zi o‘xshash to‘plamlar va cheksiz o‘ziga-o‘zi o‘xshash shakllardir, o‘lchami kasriy to‘plamdir. Bunday ta’riflardan yana bir nechtasini keltirish mumkin.

Yuqoridagi ta’riflardan kelib chiqib ularni quyidagi ikkita guruhga ajratish mumkin:

Fraktallarning matematik ta’rifi. Fraktallar cheksiz rekursiv jarayonlar natijasida ifodalangan funktsional yoki hosil bo‘luvchi to‘plam va quyidagi xususiyatlarga ega:

- o‘ziga-o‘zi o‘xshash yoki masshtabning invariantligi (cheksiz skeyling), ya’ni kichik masshtabda va o‘rta masshtabda xuddi katta masshtabdagi kabi ko‘rinadi;

- kasrli o‘lchami (Xausdorf o‘lchami) topologik o‘lchamidan qat’iy katta;

- differensiallanmaydi va kasrli ko‘paytmalar hamda integrallarda aniqlashtiriladi.

Fraktallarning fizik ta’rifi. Fraktallar - kuchli qirqilgan tuzilishni ifodalovchi hamda chegaralangan masshtabda o‘ziga-o‘zi o‘xshash xususiyatini egallovchi geometrik ob’yektlar (chiziq, sirt, jism)dir.

Fraktal bu avvalo abstrakt, nazariy model, reallikda mavjud bo‘lmagan chegaraviy o‘tish natijalaridir.

Biroq fraktallarning qat’iy aniq ta’rifi mavjud emas. Ammo fraktal geometriya tabiat geometriyasidir.

Fraktallarning xususiyatlari.

1. *O‘ziga-o‘zi o‘xshashlik*. Eng oddiy holatda fraktallarning katta bo‘lmagan qismi ular haqidagi barcha axborotlarni o‘zida saqlaydi.

2. *Kasriylik*. Fraktallarning kasriyligi fraktallar noto‘g‘riligining o‘lchamini matematik ifodalash deyiladi.

3. *Nomuntazamlik*. Agar fraktal funksiya ta’riflangan bo‘lsa matematika terminlarida nomuntazam funksiya hech bir nuqtada tekis emas va differensiallanmaydi.

4. Masshtablashtirish.

5. Rivojlanish qobiliyati (uzluksiz shakllantirish tamoyili).

6. Kasrli metrik o‘lchov (o‘lchamlarning o‘ziga xoslik tamoyili).

7. Chegaralarning noaniqligi tamoyili (chetlarning noaniqlgi).

8. Dinamik xoas tamoyili.

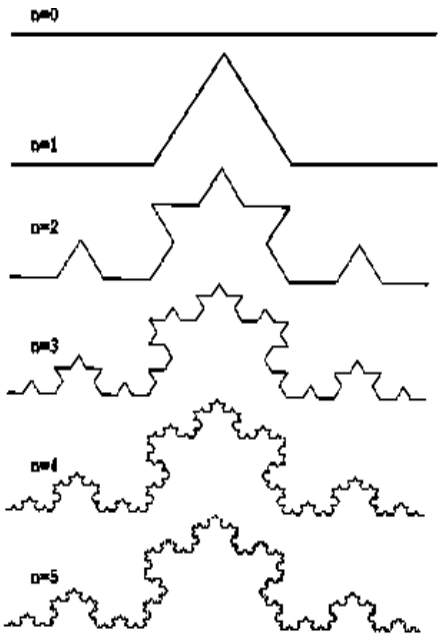
4. **Fraktal o‘lchov** - B.Mandelbrot butun bo’lmagan songa teng o‘lchovni fraktal o’lchov deb kiritdi.

Fraktallarning turlari

Fraktallarni o‘rganish uchun ularni aniq turlarga ajratish kerak bo‘ladi.

Tabiatda fraktallarning bir necha ko‘rinishini (turini) uchratish mumkin: geometrik fraktallar, algebraik fraktallar, stoxastik fraktallar, qo‘l-ijodiy fraktallar, tabiiy fraktallar va boshqalar.

Geometrik fraktallar-bu turdagi Kox triad egri chizig‘i, Levi egri chizig‘i, Gilbert egri chizig‘i, Xartera-Xeytueya ajdari nomli siniq chiziqlar, Kontor to‘plami, Serpin uchburchagi, Serpin gilami, Pifagor daraxti va hokazo kabi fraktallar guruhi eng ko‘rgazmali hisoblanadi.



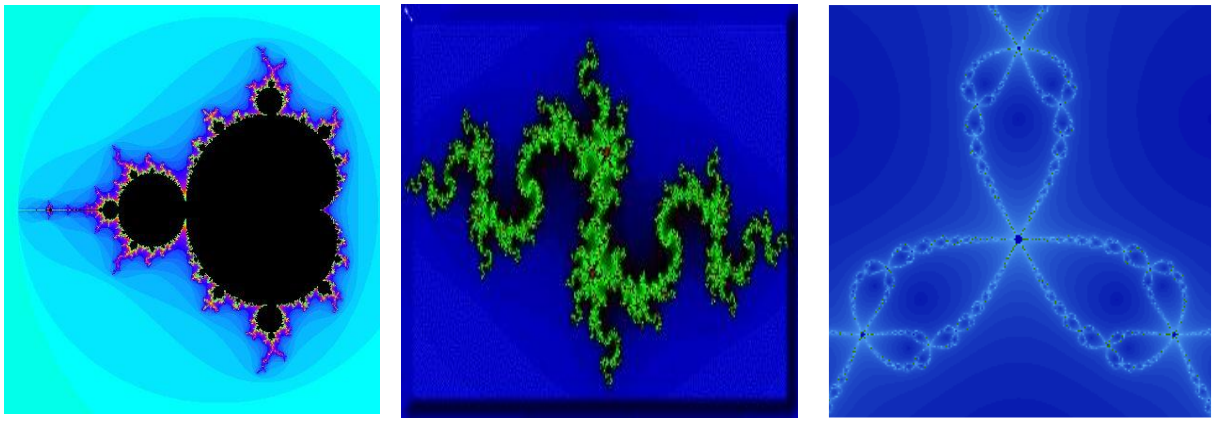
1-rasm. n=1,…,5 larda Kox triad egri chizig‘ini qurish

Fraktallar tarixi aynan shu fraktallardan boshlanadi. Geometrik fraktallarni loyihaviy fraktallar ham yuritiladi.

Bu turdagi fraktallar oddiy geometrik qurish yo‘li bilan shuningdek, iteratsion funksiyalar tizimi, L-tizimlar usuli, R-funksiya (Rvachev funksiyasi) usuli, binomial ko‘phadlarning arifmetik xususiyatlari nazariyasi asoslangan usulda va to‘plamlar nazariyasiga asosan quriladi.

Odatda geometrik turdagi fraktallarni qurish uchun ma’lum ***“kesma–aksioma–bo‘laklar yig‘indisi”*** kabi qoida o‘rinlidir. Masalan, Kox egri chiziqlari, Serpin uchburchaklari va boshqalar.

Algebraik fraktallar-fraktallarning yana bir katta guruhidir. Ular o‘z nomlariga oddiy algebraik formulalarga asosan qurilgani uchun ega bo‘lgan. Ularni nochiziq jarayonlar yordami bilan *n*-o‘lchovli fazolarda hosil qilinadi. Ma’lumki, nochiziq dinamik tizimlar bir necha barqaror holatlarni o‘zida mujassamlashtiradi. Bulardan bittasi, bir necha takrorlashlar sonidan keyin boshlang‘ich shartga bog‘liq bo‘lib qoladi.

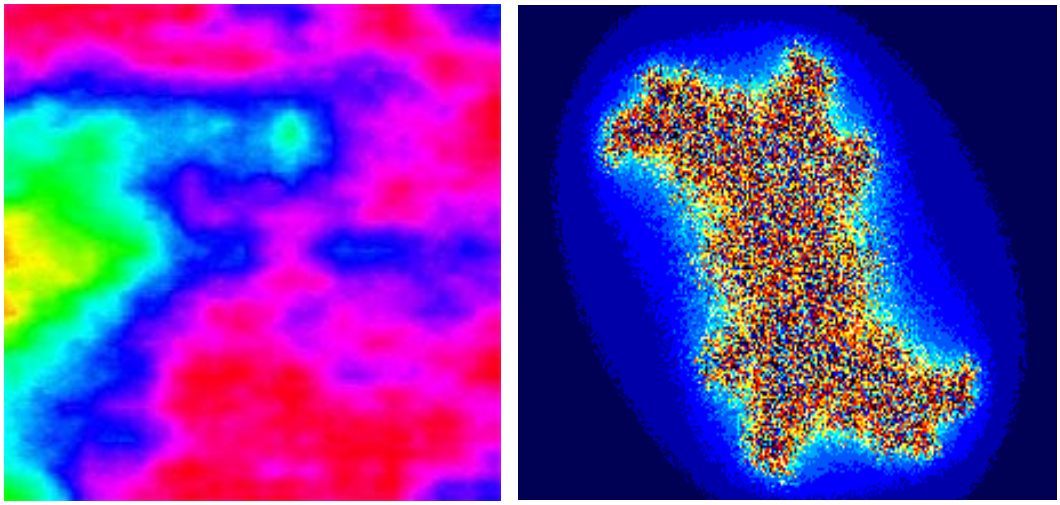


Mandelbrot fraktali Jyulia Fraktali Nyuton fraktali

2-rasm. Algebraik fraktallar

Shuning uchun har bir barqaror holat, bir necha boshlang‘ich holat sohalarini o‘zida namoyon etadi. Ulardan eng taniqlilari Mandelbrot, Julia, Nyuton to‘plamlari (1.3-rasm) va boshqalar.

Stoxastik fraktallar - eng taniqli fraktallar guruhi hisoblanadi. Ular iteratsion jarayonda to‘satdan birorta parametrni o‘zgartirishi holatidan paydo bo‘ladi (1.4-rasm). “Stoxostik” termini grek so‘zidan kelib chiqqan bo‘lib, “faraz” (tasavvur)ni anglatadi.



3-rasm. Stoxastik fraktallar

Fraktallarni yana bir qiziq sinflarga ajratish mavjud. Bunda fraktallar ikkita sinfga ajaratiladi: qo‘l - ijodiy (ideal) fraktallar va tabiiy (fizik) fraktallar.

Qo‘l-ijodiy fraktallar - olimlar tomonidan o‘ylab topilgan va ixtiyoriy masshtabda fraktallar xususiyatlarini o‘zida namoyon etadigan fraktallar.

Tabiiy fraktallar - mavjudlik sohasida chegaraga ega fraktallar.

...

1.5. Tekislikda almashtirishlar. Bir jinsli koordinatalar.

Koordinatalar sistemasi va tekislikda (ikki o‘lchovli) geometrik almashtirishlar (ko‘chish, masshtablash, burish va akslantirish). Koordinatalar sistemasi: borliq, ob’ekt, kuzatuvchi va tasvir tekisligi (ekran). Tekislikda bir jinsli koordinatalar. Matritsalar yordamida bir jinsli koordinatalarda tekislikdagi almashtirishlar. Fazoviy geometrik almashtirishlar. Fazoda bir jinsli koordinatalar. Matritsa yordamida bir jinsli koordinatalarda fazodagi almatirishlar

Koordinatalar sistemasi,

…

Tekislikdagi (ikki o‘lchovli) almashtirishlar.

Ikki o‘lchovli barcha narsalarni kompyuter grafikasida 2D (2-dimension) belgisi bilan ifodalash (kiritilgan) qabul qilingan [9, 20].

Faraz qilamizki tekislikda to‘g‘ri chiziqli koordinatalar sistemasi kiritilgan (berilgan) bo‘lsin. Unda har qanday *M* nuqtaning koordinatasini aniqlash uchun ikki juft (*x*, *y*) sonlari olinadi.

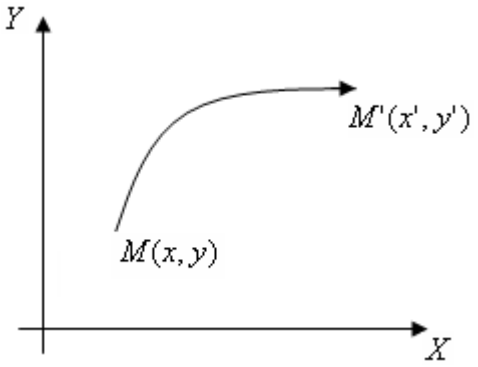
Ushbu tekislikda yana bitta to‘g‘ri chiziqli koordinatalar sistemasini kiritgan holda *M* nuqta uchun yangi mos juft (x', y') koordinatalarni hosil qilamiz.

Tekislikda bitta to‘g‘ri chiziqli koordinatalar sistemasidan boshqasiga o‘tish quyidagi tenglamalar orqali amalga oshiriladi:

$$\begin{cases} x' = \alpha x + \beta y + \lambda, \\ y' = \gamma x + \sigma y + \mu, \end{cases} \quad \begin{vmatrix} \alpha & \beta \\ \gamma & \sigma \end{vmatrix} \neq 0. \tag{1}$$

Bu yerda $\alpha, \beta, \gamma, \sigma, \lambda, \mu$ - ixtiyoriy sonlar.

Boshqa tomondan qaraganda, agar biz nuqta o‘zgarib koordinatalar sistemasi o‘zgarmas deb qabul qilsak, u holda (1) formulalar $M(x, y)$ nuqtani $M'(x', y')$ nuqtaga almashtirishini ifodalaydi (2.1-rasm).



1.1-rasm. Tekislikdagi (ikki o‘lchovli) almashtirish.

(1) formulalarni nuqtani almashtirishni ifodalaydi deb qabul qilamiz.

Almashtirish formulalaridagi koeffisientlarning geometrik ma’nosini o‘rganish uchun berilgan koordinatalar sistemasini to‘g‘ri burchakli dekart koordinatalar sistemasi deb hisoblash qulay.

Ikki o‘lchovli almashtirishlarning xususiy hollarini ko‘ramiz.

1. Ko‘chish.

$M(x, y)$ nuqtani $M'(x', y')$ nuqtaga ko‘chish berilgan λ va μ ko‘chish konstantalari vektorining koordinatalariga qo‘shish orqali amalga oshiriladi

$$x' = x + \lambda,$$
$$y' = y + \mu.$$

2. Masshtablash. Cho‘zish (siqish).

Koordinatalar o‘qlari bo‘yicha cho‘zish (yoki siqish) ko‘paytirish orqali ifodalanadi:

$$x' = \alpha x,$$
$$y' = \delta y,$$

$\alpha > 0, \delta > 0$ - masshtablash koeffisientlari bo‘lib mos ravishda X va Y o‘qlari bo‘yicha cho‘zish va siqishni bildiradi.

Agar $\alpha > 1, \delta > 1$ bo‘lsa koordinata o‘qlari bo‘yicha cho‘zish va $\alpha < 1, \delta < 1$ bo‘lsa, siqish ta’minlanadi.

Cho‘zish (siqish) almashtirishlarini vektor-matritsa shaklida kuyidagicha yozish mumkin:

$$(x',\ y') = (x,\ y)\begin{pmatrix} \alpha & 0 \\ 0 & \delta \end{pmatrix}$$

3. Burish.

Burish quyidagi formula orqali beriladi:

$$\mathrm{x}' = \mathrm{x}\ \cos\ \varphi - y \sin\ \varphi,$$
$$\mathrm{y}' = \mathrm{x}\ \sin\ \varphi + y\cos\ \varphi,$$

Bu yerda koordinatalar sistemasining boshlang‘ich nuqtasi bo‘ylab soat yo‘nalishiga nisbatan teskari φ burchakka burish bajariladi.

Vektor-matritsa shaklida burishni quyidagicha yozish mumkin:

$$(x',\ y') = (x,\ y)\begin{pmatrix} \cos\ \varphi & \sin\ \varphi \\ -\sin\ \varphi & \cos\ \varphi \end{pmatrix}.$$

4. Akslantirish.

Tekislikda akslantirish koordinatalar sistemasining o‘qlariga nisbatan bajariladi. Abtsissa o‘qiga nisbatan akslantirish quyidagicha ifodalanadi:

$$x' = x;$$
$$y' = -y.$$

Vektor-matritsa shaklida esa:

$$(x',\ y') = (x,\ y)\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Ordinata o‘qiga nisbatan akslantirish quyidagicha ifodalanadi:

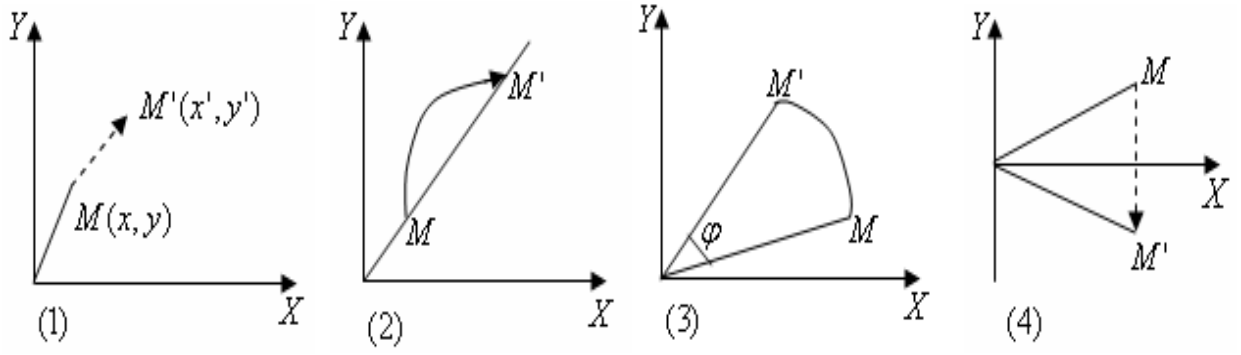
$$x' = -x;$$
$$y' = y.$$

Vektor-matritsa shaklida:

$$(x',\ y') = (x,\ y)\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Almashtirishlarni yuqoridagi berilgan to‘rtta xususiy holini berilishidan maqsad:

- har qaysi almashtirish oddiy va tushunarli geometrik ma’noga ega;
- ixtiyoriy almashtirishni ularni ketma-ket bajarish (superpozisiyasi) orqali ifodalash mumkin.



1.2-rasm. Tekislikdagi (ikki o‘lchovli) almashtirishlar.

Ammo keyingi masalalarni ko‘rish uchun to‘rtta oddiy almashtirishlarni ham (ko‘chirishni hisobga olgan holda) matritsa shaklida ifodalash kerak.

Nuqtaning bir jinsli koordinatalari. Tekislikdagi almashtirishlarni vektor-matritsa shaklida ifodalash.

Faraz qilamizki tekislikda $M(x, y)$ nuqta berilgan bo‘lsin. Ixtiyoriy x_1, x_2, x_3 (bir vaqtda noldan farqli sonlar M nuqtaning bir jinsli koordinatalari deb ataladi, agarda:

$$\frac{x_1}{x_3} = x, \quad \frac{x_2}{x_3} = y.$$

Ya’ni ixtiyoriy $h \neq 0$ kupaytiruvchi uchun $-M(hx, hy, h)$.

Kompyuter grafikasi masalasini ishlash jarayonida ixtiyoriy $M(x, y)$ nuqtaning bir jinsli koordinatalari quyidagicha kiritiladi:

$M(x, y, 1)$, ya’ni $h=1$.

Ko‘rish mumkinki (1) almashtirish formulalarni bir jinsli koordinatalarda quyidagicha ifodalash mumkin:

$$(x',\ y',\ 1) = (x,\ y,\ 1)\begin{pmatrix} \alpha & \gamma & 0 \\ \beta & \sigma & 0 \\ \lambda & \mu & 1 \end{pmatrix}. \qquad (2)$$

Ikki o‘lchovli almashtirishlarning xususiy hollari, ya’ni 1, 2, 3, 4 uchun mos matritsalarni yozib chiqamiz:

1. *Ko‘chish matritsasi (translation):*

$$K = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \lambda & \mu & 1 \end{pmatrix}.$$

2. *Masshtablash (cho‘zish, siqish) matritsasi (dilatation):*

$$M = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \delta & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

3. *Burish matritsasi (rotation):*

$$Б = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

4. *Akslantirish matritsasi (reflection)*:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Ixtiyoriy almashtirishlarning matritsasini yuqorida keltirilgan *K*, *M*, *B*, *A* matritsalarni ketma-ket (superpozisiya qoidasi asosida) ko‘paytirish orqali hosil qilish mumkin. Ular oddiy almashtirishlarning bajarilishiga qarab mos ravishda ko‘paytiriladi.

Misol: *ABC* uchburchakni $A(x, y)$ uchiga nisbatan φ burchakka burish almashtirishining matritsasini quring.

1-qadam. $A(x, y)$ nuqtani koordinatalar sistemasi boshiga *O(0,0)* nuqtaga, ya’ni $(-x,-y)$ vektoriga ko‘chish:

$$K_{-A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x & -y & 1 \end{pmatrix}.$$

2-qadam. Koordinatalar sistemasi boshiga nisbatan φ burchakka burish:

$$B_{\varphi} = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3-qadam. *A* nuqtani dastlabki holatiga qaytarish uchun (x, y) vektorga ko‘chish:

$$K_A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x & y & 1 \end{pmatrix}.$$

Keltirilgan tartibda almashtirish matritsalarini ketma-ket ko‘paytiramiz:

$$B_A = (K_{-A} \times B_\varphi) \times K_A .$$

Natijada matritsa ko‘rinishidagi almashtirishni quyidagi ko‘rinishda olamiz:

$$(x', y', 1) = (x, y, 1) \times \begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ -x\cos\varphi + y\sin\varphi + x & -x\sin\varphi - y\cos\varphi + y & 1 \end{pmatrix}.$$

E'tibor berilsa barcha almashtirishlarning matritsalari determinantlari noldan farqli.

1.6. Fazodagi (uch o‘lchovli) almashtirishlar. Platon jismlar.

Fazoviy geometrik almashtirishlar. Fazoda bir jinsli koordinatalar. Matritsa yordamida bir jinsli koordinatalarda fazodagi almatirishlar. Muntazam ko‘pyoqliklar. Platon jismlari va ularning xususiyatlari

Fazodagi, ya’ni uch o‘lchovli (3D, 3-dimension) almashtirishlarni ko‘ramiz va ularni bir jinsli koordinatalarni kiritgan holda qaraymiz.

Ikki o‘lchovli holdagidek nuqtani fazoda aniqlovchi uchta koordinatasini (x, y, z) to‘rtta bir jinsli koordinatalarga almashtiramiz $(x, y, z,1)$ yoki umumiy hol uchun $(hx, hy, hz, h), h \neq 0$. Bu yerda ham h - ko‘paytiruvchi va *h=1* deb olamiz .

Keltirilgan bir jinsli koordinatalar uch o‘lchovli almashtirishlarni matritsalar orqali yozish imkonini beradi.

Ixtiyoriy almashtirish uch o‘lchovli fazoda ko‘chish, masshtablash (cho‘zish, siqish), burish va akslantirishlarni superpozisiyasi orqali aniqlanishi mumkin. Shuning uchun birinchi navbatda ushbu akslantirishlarning matritsalarini ko‘ramiz.

Ma'lumki qaralayotgan holatda matritsalarning o‘lchovi to‘rtga teng.

1. Ko‘chish. Fazoda ko‘chish ham ko‘chish vektori bilan beriladi, ya’ni ko‘chish vektorining λ, μ, ν koeffisientlari bilan ifodalanadi:

$$K = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \lambda & \mu & \nu & 1 \end{pmatrix}$$

bu yerda (λ, μ, ν) – ko‘chirish vektori .

2. Cho‘zish (siqish). Fazoda ham masshtablash (cho‘zish, siqish) masshtablash koeffisientlari bilan ifodalanadi:

$$Ch = \begin{pmatrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

bu yerda $\alpha > 1$ $(1 > \alpha > 0)$ - abtsissa o‘qi bo‘ylab cho‘zish (siqish) koeffisienti,

$\beta > 1$ $(1 > \beta > 0)$ - ordinata o‘qi bo‘ylab cho‘zish (siqish) koeffisienti,

$\gamma > 1$ $(1 > \gamma > 0)$ - applikata o‘qi bo‘ylab cho‘zish (siqish) koeffisienti.

3. Burish. Fazoda burish koordinata sistemasining koordinatalariga nisbatan amalga oshiriladi va burish burchagi bilan beriladi.

Absissa o‘qi buylab φ burchakka burish matritsasi:

$$B_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Ordinata o‘qi buylab ψ burchakka burish matritsasi:

$$B_y = \begin{pmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Applikata o‘qi buylab θ burchakka burish matritsasi:

$$B_z = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4. Akslantirish. Fazoda akslantirish koordinata sistemasining tekisliklariga nisbatan amalga oshiriladi.

XY tekisligiga nisbatan akslantirish matritsasi:

$$A_{XY} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

YZ tekisligiga nisbatan akslantirish matritsasi:

$$A_{YZ} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

zx tekisligiga nisbatan akslantirish matritsasi:

$$A_{ZX} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Bu yerda shuni aytish joizki barcha matritsalarning determinantlari noldan farqli, geometrik almashtirishdan keyin geometrik obyekt o‘lchamini yo‘qotmaydi.

Fazodagi barcha almashtirishlarni keltirilgan oddiy to‘rtta almashtirishlar ketma-ket bajarilishi (superpozisiya) orqali amalga oshirilish mumkin.

Ixtiyoriy fazodagi almashtirishning matritsasi quyidagi ko‘rinishga ega:

$$M = \begin{pmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 0 \\ \beta_1 & \beta_2 & \beta_3 & 0 \\ \gamma_1 & \gamma_2 & \gamma_3 & 0 \\ \lambda & \mu & \nu & 1 \end{pmatrix}.$$

Agar biror bir geometrik obyekt *n*-ta nuqtalardan iborat bo‘lsa (ya’ni *n*-ta nuqta orqali berilgan bo‘lsa), u holda almashtirish matritsasi M aniqlangandan so‘ng, berilgan nuqtalarni $V_i(x_i, y_i, z_i),\ (i = 1, n)$ matritsasini hosil qilamiz va so‘ng ko‘paytirish amalini bajaramiz:

$$V' = V \times M = \begin{pmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & .z_2 & 1 \\ \dots & \dots & \dots & \dots \\ x_n & y_n & z_n & 1 \end{pmatrix} \times M.$$

Natijada geometrish almashtirishdan so‘ng gosil bulgan obyektning nuqtalari matritsasini olamiz.

Platon jismlari (ko‘pyoqliklar).

Barcha yoqlari to‘g‘ri ko‘pburchaklardan va barcha uchlariga tegishli burchaklar o‘zaro teng bo‘lgan qavariq ko‘pyoqliklar muntazam ko‘pyoqliklar (Platon jismlari) deb ataladi [4, 13].

Beshta muntazam ko‘pyoqliklar mavjud (buni Yevklid isbotlagan): to‘g‘ri tetraedr, geksaedr (kub), oktaedr, dodekaedr, ikosaedr. Ularning o‘ziga xos asosiy xususiyatlari:

Nomi	Yoqlari soni (Yo)	Qirralari soni (Q)	Uchlari soni (U)
Tetraedr	4	6	4
Geksaedr	6	12	8
Oktaedr	8	12	6
Dodekaedr	12	30	20
Ikosoedr	20	30	12

Yoqlar, Qirralar va Uchlar soni o‘zaro quyidagi Eyler formulasi bilan bog‘liq:

$$Yo+U=Q+2.$$

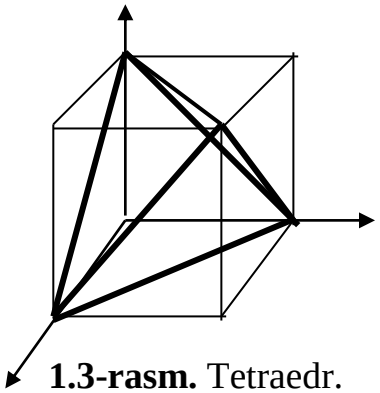
Ko‘pyoqliklarni qurish algoritmlarini ko‘ramiz. Buning uchun ularning uchlarini topish etarli.

Geksaedrni (kub) qurish qiyinchilik tug‘dirmaydi. Masalan farz qilamizki kubning bitta uchi koordinata sistemasining boshida yotsin. Unga qo‘shni bo‘gan uchlari esa koordinata sitemasining o‘qlarida bo‘lsin. Unda kubning yana uchta uchlari mos ravishda koordinata sistemisining tekisliklarida yotadi va so‘nggi uchi fazoda joylashadi. Agarda kubning qirrasi d ga teng deb olsak, u holda uning koordinatalari quyidagicha topiladi:

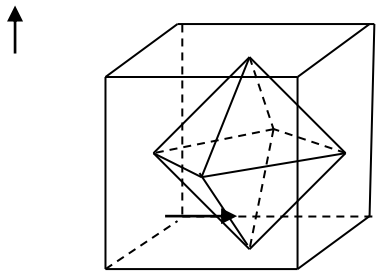
$$V_1 = (0,0,0),$$
$$V_2 = (d,0,0),$$
$$V_3 = (0,d,0),$$
$$V_4 = (0,d,0),$$
$$V_5 = (d,d,0),$$
$$V_6 = (0,d,d),$$
$$V_7 = (d,0,d),$$
$$V_8 = (d,d,d),$$

Tetraedrni qurish algoritmi: Tetraedrni qirralari kubning qarama-qarshi yoqlaridagi ayqashgan diagonallari bo‘ladi.

Oktaedrni qurishda quyidagi xossadan foydalanamiz: oktaedrning uchlari kub yoqlarining markazlariga (og‘irlik) mos keladi, ya’ni mos yoq uchlarining o‘rta arifmetik qiymatlari.

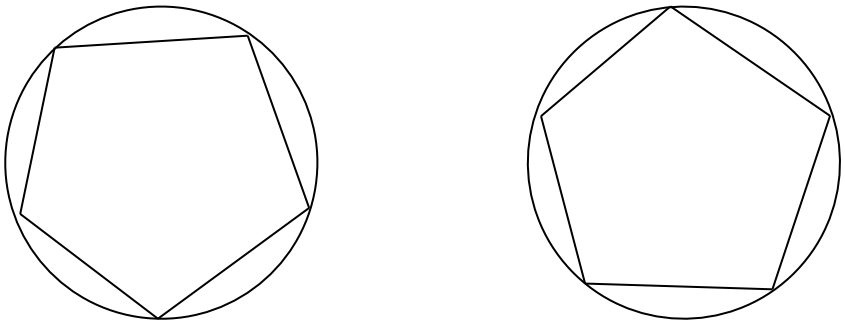


1.3-rasm. Tetraedr.



1.4-rasm. Oktaedr.

Ikosaedrni qurishni ko‘ramiz. Z o‘qida $z = \pm 0.5$ markazli, $r = 1$ radiusli va XY tekisligiga parallel ikkita aylana o‘tkazamiz. Har bir aylanani beshta teng bo‘lakka bo‘lib, ularni rasmda ko‘rsatilgan tartibga mos birlashtiramiz. Ikkita aylanalarning nuqtalarini ketma-ket birlashtiramiz va ikosaedrning yoqlarini tashkil qiluvchi o‘nta muntazam uchburchakni hosil qilamiz. Qolgan yoqlari uchun $z = \pm \sqrt{5}/2$ nuqtalarini olamiz va mos aylanalarning nuqtalari bilan tutashtiramiz. Bunda yana unta yoq hosil bo‘ladi.



Расм …

Dodekaedrning uchlari ikosaedr yoqlarining og‘irlik markazlari bo‘ladi.

Nazorat savollari:

1. Axborotlarni qayta ishlashning asosiy yo‘nalishlari hisoblangan tasvirni tanlash, qayta ishlash va kompyuter grafikasiga izoh bering?

2. Rastr, vektor va fraktal grafikaga izoh bering?
3. Rastr va vektor grafika daturlari haqida ma’lumot bering?
4. Tekislikdagi almashtirishlar deganda nima tushuniladi?
5. Ikki o‘lchovli almashtirishlarning xususiy hollarini ifodalang?
6. Nuqtaning bir jinsli koordinatalari nima va ular qanday vazifalarda qo‘llaniladi?
7. Fazodagi (uch o‘lchovli) almashtirishlar deganda nima tushuniladi?
8. Platon jismlari haqida umumiy ma’lumot bering?

Tayanch iboralar: kompyuter grafikasi, tasvirni tanlash, tasvirni qayta ishlash, rastr, vektor va fraktal grafika, kuchish, burish, akslantirish, masshtablash.

1.7. Poligonal to‘rlar va ularning xususiyatlari.

Poligonal to‘rlar va ularni berish usullari. Poligonal to‘rlarni ifodalash algoritmlari. Muntazam ko‘pyoqliklar.

Kompyuter grafikasida fazodagi uch o‘lchovli obyektlarni sirtlarini tasvirlashni ikkita usuli keng tarqalgan: Poligonal setkalar va bikubik parametrik bo‘laklar.

Poligonal setka bu fazoviy obyektni tasvirlovchi o‘zaro bog‘liq balandliklar, qirralar va yoqlar (ko‘pburchaklar) to‘plami. Nuqtalar (uchlar) qirralar bilan tutashtiriladi, ko‘pburchaklar esa uchlar va qirralar bilan ifodalanadi [4, 13].

Poligonal setkalarni qurishni 3 xil usuli mavjud:

1. Ko‘pburchaklarni oshkora berish.

Har bir ko‘pburchak uning uchlari koordinatalari bilan beriladi, ya’ni

$$P=((X_{1,}Y_1,Z_1),(X_2,Y_{2,}Z_2),\ldots(X_{n,}Y_n,Z_n)).$$

Uchburchakni ifodalovchi (aniqlovchi) uchlar ketma-ket saqlanadi va qirralar bilan tutashtiriladi, shu jumladan oxirgi va birinchi uchlar ham.

Har bir alohida ko‘pburchak uchun bu usul albatta qulay, hamma umumiy uchlarni koordinatalarini takroran saqlash evaziga poliganal setka xotirada ko‘p joyni egallaydi.

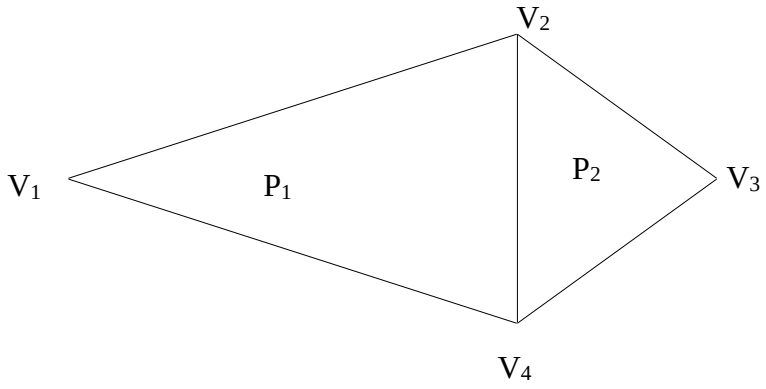
2. Uchlarni oshkora berish. Ko‘pburchaklarni uchlar ro‘yxatidagi ko‘rsatkichlari orqali ifadalash.

Bu holda poligonal setkaning har bir tuguni uchlar ro‘yxatida bir marta saqlanadi:

$$V=((X_1,Y_1, Z_1), (X_2, Y_2, Z_2),...., (X_n,Y_n, Z_n)).$$

Ko‘pburchak uchlar ro‘yxatidagi (indeks) ko‘rsatkichlari orqali beriladi. Ko‘pburchakning har bir uchi bir marta saqlanadi va bu xotira hajimini tejashga olib keladi. Ammo umumiy qirralar ikki martada chiziladi. Misol:

$$V = (V_1, V_2, V_3, V_4) = ((X_1, Y_1, Z_1), \ldots,(X_4, Y_4, Z_4)).$$

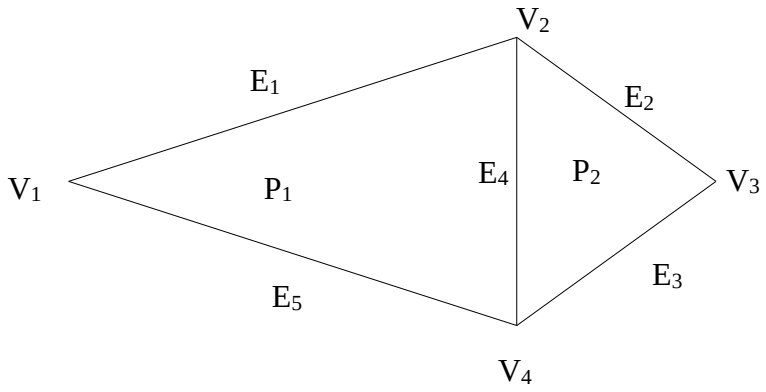


1.9- rasm. Poligonlarni tasvirlash.

3. Qirralarni oshkora berishi.

Bu holda ko‘pburchak qirralar ro‘yxatidagi ko‘rsatkichlari to‘plami orqali beriladi. Qirralar ro‘yxatida har bir qirra bir marta uchraydi va har bir qirra ro‘yxatda uchlari (ikkita) va mos ko‘pburchaklar (1 yoki 2 ta) orqali ifodalanadi. Ya’ni har bir ko‘pburchak quyidagicha, $R=(E_{1,\ \dots\ ,}E_n)$, va har bir qirra quyidagicha $E=(V_1, V_2, R_1, R_2)$ ifodalanadi. Agar qirra bitta ko‘pburchakka tegishli bo‘lsa u holda R_1 yoki R_2 bo‘sh to‘plam.

Qirralarni oshkora berishda poniganal setka hamma qirralarni chizish orqali beriladi va umumiy qirralar qayta chizilmaydi. Misol:



1.10-rasm. Poligonlarni tasvirlash.

$V= (V_1, V_2, V_3, V_4) = ((X_1, Y_1, Z_1), \dots,(X_4, Y_4, Z_4))$.

$R_1=(E_1,E_4,E_5)$ $R_2=(E_2,E_3,E_4)$

$E_1=(V_1, V_2, R_1, 0)$, $E_2=(V_2, V_3, R_2, 0)$, $E_3=(V_3, V_4, R_2, 0)$,

$E_4=(V_4, V_2, R_1, R_2)$, $E_5=(V_4, V_1, R_1, 0)$.

1.8. Geometrik splaynlar. Splayn egri chiziqlari.

Splayn egri chiziqlari. Ermit, Beze, B-splayn egri chiziqlari. Egri chiziqlarning geometrik xususiyatlari

Splayn egri chiziqlari.

Kompyuter grafikasida parametrik kubik (3 chi darajali) egri chiziqlar ishlatiladi.

Parametrik ko‘rinishda berilgan γ egri chizig‘i deb *x,u,z* koordinatalari

$$x = x(t),\ y = y(t),\ z = z(t),\quad a \le t \le b, \tag{3}$$

munosabatlar bilan aniqlanuvchi $M(x, y, z)$ nuqtalar to‘plamiga aytiladi, bu yerda $x(t),\ y(t),\ z(t)\ -\ [a,b]$ kesmada uzluksiz formulalar:

u=t-a/b-a almashtirish orqali [*a,b*] kesmani [0;1] kesmaga olib kelishi mumkin. Vektor ko‘rinishda (1) chi tenglamani quyidagicha yozish mumkin.

Vektor forma (3) $r = r(t) = (x(t),\ y(t),\ z(t)),\quad 0 \le t \le 1.$

Parametrik kub (3) darajali egri chiziqning tenglamasini quyidagicha ko‘rinishda yozamiz.

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y, 0 \le t \le 1$$

$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$

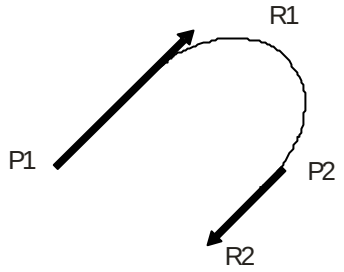
Ermit egri chizig‘i.

Ermit egri chizig‘i boshlang‘ich va oxirigi nuqtalari R_1 va R_2 va ushbu nuqtalardagi egri chiziqqa urunma vektorining yo‘nalishlari bilan R_1 va R_2 beriladi (2.11-rasm).

Egri chiziqni (3) noma’lum koeffisientlarini aniqlash uchun (3) tenglamaning birinchi tenglamasini ko‘ramiz va uni quyidagi ko‘rinishda yozib olamiz.

$x(t) = at^3 + bt^2 + ct + d$ yoki $x(t) = (t^3,\ t^2,\ t,\ 1)(a,b,c,d)$ yoki

$$x(t) = T \cdot X.\ \ T = (t^3,\ t^2\ t,\ 1),\ X = (a,b,c,d). \tag{4}$$



1.11-rasm. Ermit egri chizig‘i.

(4) chi ifoda differensiallangandan so‘ng:

$$X(t) = (3t^2, 2t, 1, 0) \cdot X. \tag{5}$$

Berilgan shartlarni va (4),(5) ni hisobga olgan holda:

$X(0)=R_{1x}=(0,0,0,1) \cdot X$

$X(1)= R_{2x}=(1,1,1,1) \cdot X$ yoki $\begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} = \begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{pmatrix}$

$X(0)= R_{1x}=(0,0,1,0) \cdot X$

$X(1)= R_{2x}=(3,2,1,0) \cdot X.$

Qidiralayotgan X ni topish uchun teskari matritsani hisoblash kerak:

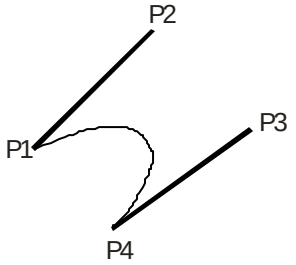
$$X = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} = M_э \cdot P_x \tag{6}$$

Hosil bulgan M_e matritsasi va R_x ermit geometrik vektori deb ataladi.

Beze egri chizig‘i.

Agar Ermit egri chizig‘i $R_e=(R_1,R_2,R_1,R_2)$ bilan berilsa Beze egri chizig‘i R_1,R_2,R_3,R_4, nuqtalar yoki $R_b=(R_1,R_2,R_3,R_4)$, orqali beriladi. R_e Ermit geometrik matritsalari va R_b Beze geometrik matritsalari o‘zaro quyidagi munosabatlar bilan bog‘liq:

$$R_e = \begin{pmatrix} P_1 \\ P_2 \\ R_1 \\ R_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{pmatrix} \cdot \begin{pmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{pmatrix} = M \cdot R_b .$$



1.12-rasm. Beze egri chizig‘i.

Ermit matritsasini M_e M matritsaga ko‘paytirib Beze matritsasini olamiz:

$$M_b = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

R_1,R_2,R_1,R_2 nuqtalari bilan beriluvchi Beze egri chizig‘i vektor parametrik tenglamasi: $r(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t)P_3 + t^3 P_4$.

Yoki matritsa kurinishda: $r(t)=T\cdot M_e\cdot R_e= T\cdot(M_e\cdot M)\cdot R_b= T\cdot M_b\cdot R_b,\ 0\le t\le 1$.

$R_0, R_i, \ldots R_m$ nuqtalar bilan aniqlanuvchi Beze egri chizig‘i:

1. C – uzluksiz bo‘lishi uchun uning har bir uchta R_{3i-1}, R_{3i+1} – nuqtalari bitta to‘g‘ri chiziqda yotishi kerak:

2. C – uzluksiz va berk bo‘lishi uchun birinchi va oxirigi nuqtasi ustma-ust tushib va $R_{m-1},\ R_m = R_0,\ R_1$ – nuqtalari bitta to‘g‘ri chiziqda yotishi kerak.

3. Umumiy holda Beze egri chizig‘ini quyidagi ko‘rinishda yozish mumkin.

4. $r(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} P_i, \quad 0 \le t \le 1.$

5. R_i, $i=0$ egri chiziqni aniqlovchi nuqtalar: $r(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} = (t + (1-t))^m$.

6. $C_m^i t^i (1-t)^{m-i}$ funksional koeffisienlar, ya’ni universal Bershteyn ko‘p hollari ular har doim manfiy emas va ularning yig‘indisi doim 1 ga teng.

B-splayn egri chizig‘i.

P_1,P_2,P_3, va P_4 nuqtalari bilan aniqlanuvchi B-splayn kubik egri chizig‘i matritsa ko‘rinishdagi tenglamasi quyidagi ko‘rinishda:

$$r(t) = T \cdot M_{б} \cdot P$$

$$r(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}, T = (t^3, t^2, t, 1)$$

$$R=(P_1,P_2,P_3,P_4), \quad M_{б} = \frac{1}{6} \cdot \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

B-splayn egri chizig‘i bazis matritsasi yoki vektori parametrik ko‘rinishda:

$$r(t) = \frac{(1-t)^3}{6} P_1 + \frac{3t^3 - 6t^2 + 4}{6} P_2 + \frac{-3t^3 + 3t^2 + 3t + 1}{6} P_3 + \frac{t^3}{6} P_4 .$$

Kubik B-splayn egri chizig‘i uzluksiz va bundan tashqari birinchi va ikkichi hosilalari uzluksiz. P_1,P_2,P_3, va P_4 nuqtalari (4) qavariq ko‘pburchakni uchlarini (5) qavariq kupyoqlikni uchlarini tashkil qiladi va egri chiziq uning ichida yotadi.

B-splayn egri chizigi berk bulishi uchun uchta nuqta qo‘shishi etarli:

$$P_{m+1} = P_0, \quad P_{m+2} = P_1, \quad P_{m+3} = P_2.$$

Agar B-splayn egri chizig‘ining uchta ko‘shni nuqtasi P_i,P_{i+1}, P_{i+2} bitta to‘g‘ri chiziqda yotsa egri chiziq to‘g‘ri chiziqqa urinib o‘tadi.

P_1, P_2, R_1, R_2 nuqtalari bilan beriluvchi Beze egri chizig‘i vektor parametrik tenglamasi: $r(t) = (1-t)^3 P_1 + 3t(t-1)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4$

yoki matritsa ko‘rinishda $r(t) = T * M_{Б} * P_{Б} \quad 0 \le t \le 1$.

$R_0, R_i, \ldots R_m$ nuqtalar bilan aniqlanuvchi Beze egri chizig‘i:

1) C^1 – uzluksiz bo‘lishi uchun uning har bir uchta R_{3i-1}, R_{3i+1} – nuqtalari bitta to‘g‘ri chiziqda yotishi kerak yoki $\overline{P_{3i-1}P_{3i}} = k\overline{P_{3i}P_{3i+1}}$.

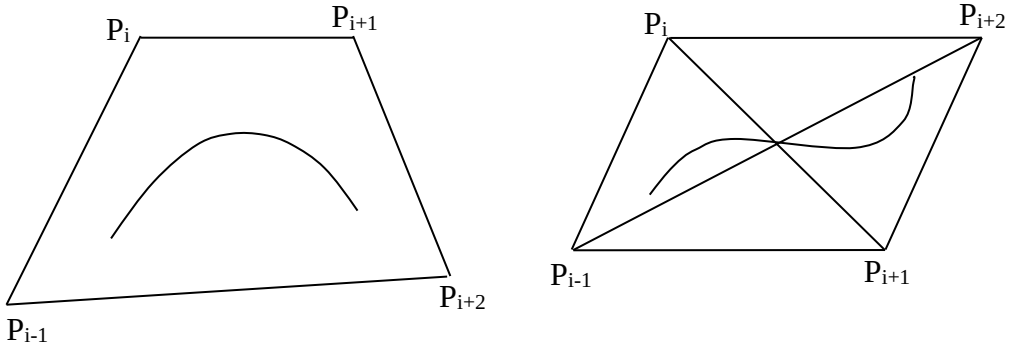
2) C^1 – uzluksiz va berk (yopiq) bo‘lishi uchun birinchi va oxirigi nuqtasi ustma-ust tushib va R_{m-1} , R_m=R_0, R_1 nuqtalari bitta to‘g‘ri chiziqda yotishi kerak.

3) C^2 – uzluksiz bo‘lishi uchun $P_{3i-2}, P_{3i-1}, P_{3i}, P_{3i+1}, P_{3i+2}$, $(i \geq 1)$ nuqtalari bitta tekislikda yotishi kerak.

Umumiy holda Beze egri chizig‘ini quyidagi ko‘rinishda yozish mumkin:

$$r(t) = \sum_{i=0}^{m} C_m^{\ i} t^i (1-t)^{m-i} Pi\ , \quad 0 \leq t \leq 1$$

bu yerda: R_i, $i = \overline{0, m}$ egri chiziqni aniqlovchi nuqtalari; $r(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} = (t + (1-t))^m = 1;$ $C_m^i t^i (1-t)^{m-i}$ funksional koeffisientlar, ya’ni universal Bershteyn ko‘phadlari ular har doim manfiy emas va ularning yigindisi doim 1 ga teng.



1.13-rasm. B-splayn egri chizig‘i.

1.9. Splayn sirtlari.

Splayn yuzalar. Ermit, Bez'e, B- splayn yuzalari.

Kompyuter grafikasida bikubik splayn sirtlari keng ishlatiladi. Xususan Beze va B-splayn sirtlari.

Beze kubik sirtlari fazoda 16 ta nuqta bilan aniqlanadi:

$$P_{ij},\ i=1,2,3,4,\ \ j=1,2,3,4$$

Parametrik tenglamasi quydagi ko'rinishga ega:

$$r(s,t)=\sum_{i=1}^{4}\sum_{j=1}^{4} C_3^{\,i-1} C_3^{\,j-1} S^{\,i-1}(1-S)^{4-i}\, t^{\,j-1}(1-t)^{4-j} P_{i.j}$$

bu yerda $0 \le s \le 1,\ \ 0 \le t \le 1,\ \ r(s,t)=(x(s,t),\ y(s,t),\ z(s,t))$

yoki quyidagi qo'rinishda:

$$X(s,t) = SM_b P_x M_b^t T^t$$
$$Y(s,t) = SM_b P_y M_b^t T^t$$
$$Z(s,t) = SM_z P_x M_b^t T^t$$

bu yerda: $S=(S^3,S^2,S,1),\ T=(T^3,T^2,T,1)$.

M_b- Beze matritsasi.

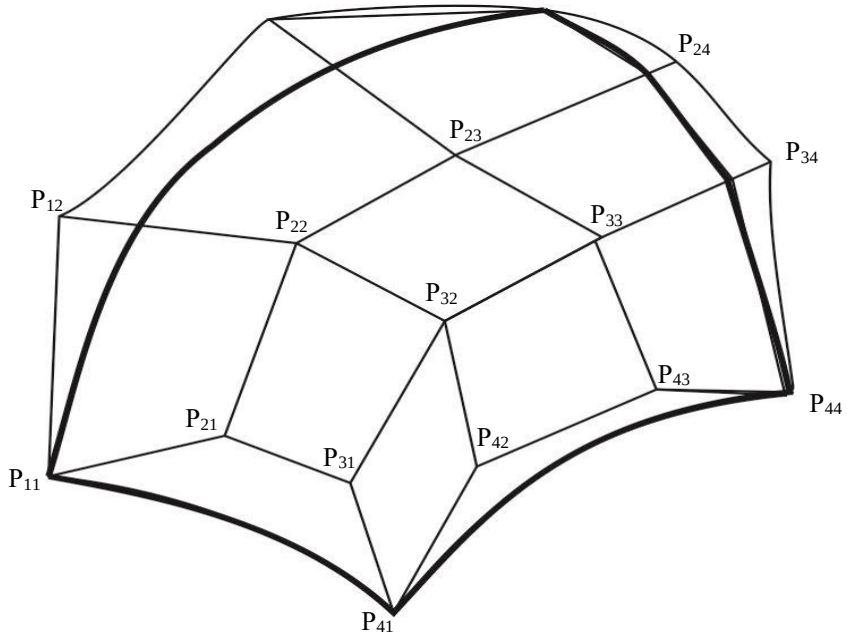
$$P_x = \begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{pmatrix}$$

P_Y, P_Z mos sirtni aniqlovchi u, z koordinatalari matritsalari.

Beze sirtining xossalari:

1. Sirt qavariq kubikda yotadi;
2. Sirt silliq (uzluksiz);
3. P_{11}, P_{14}, P_{41}, P_{44} nuqtalarga tayanadi.

P_{13} P_{14}



1.14-rasm. Beze splayn sirti.

B-splayn sirti tenglamasini quydagicha bo‘sh bo‘yash usullari:

$$X(s,t) = SM_B P_x M_B^t T^t$$

$$Y(s,t) = SM_B P_y M_B^t T^t$$

$$Z(s,t) = SM_B P_z M_B^t T^t$$

Nazorat savollari:

1. Poligonal setka nima?
2. Poligonal setkani berish usullarini tavsiflang?
3. Ixtiyoriy obyektni poligonal setka orqali qurilishini tushuntiring?
4. Geometrik splaynlar. Splayn egri chiziqlariga izoh bering?
5. Ermit splayn egri chizig‘ining tavsifini bering?
6. Beze splayn egri chizig‘ining tavsifini bering?
7. B-splayn egri chizig‘ining tavsifini bering?
8. Splayn sirtlari izoh bering?

Tayanch iboralar: Poligonal setka, ko’pburchak, uchlar, yoqlar, qirralar, Splayn egri chiziqlari, Ermit egri chizig‘i, Beze egri chizig‘i, B-splayn egri chizig‘i, Splayn sirtlari.

1.10. Proyeksiyalash. Parallel proeksiyalash.

Proyeksiyalashning asosiy turlari. Parallel proeksiyalash: Ortografik, aksonometrik va qiya burchakli proeksiyalash.

Umuman olganda, *proeksiya* (*proeksiyalash*) deb *n* o‘lchovli koordinatalar sistemasida berilgan nuqta(lar)ni *n* dan kam bo‘lgan o‘lchovli koordinatalar sistemasidagi nuqta(lar)ga geometrik almashtirishlarga aytiladi. Xususiy holda, kompyuter grafikasida 3 o‘lchovlidan 2 o‘lchovliga proeksiyalashni ko‘ramiz.

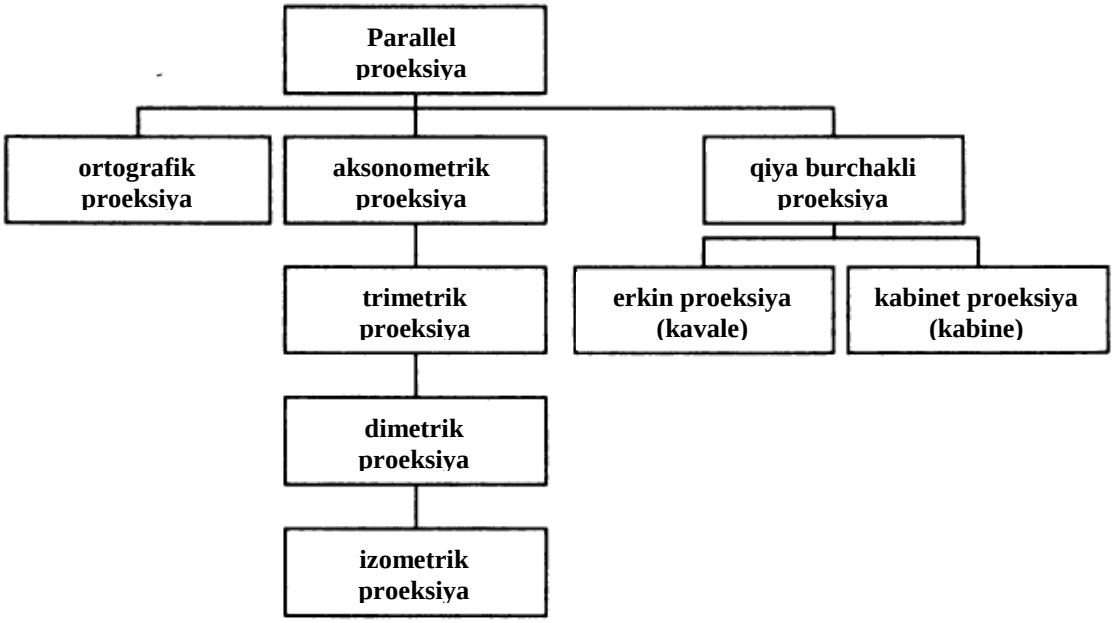
Uch o‘lchovli *obyektni* proeksiyasi proeksiya nurlari orqali quriladi va ular *proektorlar* deyiladi, ular *proeksiya markazi* (nuqta)dan chiqib obyektning har bir nuqtasidan o‘tadi va *proeksiya (tasvir) tekisligidan* kesib o‘tib unda obyektning *proeksiyasini* yasaydilar.

Kompyuter grafikasida proeksiyalarni bir necha turlari ishlatiladi. Ulardan amaliyotda ko‘p ishlatiladigani va asosiylari bu *parallel* va *markaziy* (*perspektiv*) proeksiyalar. Ular proeksiya markazi va proeksiya tekisligi orasidagi masofa orqali farqlanadi, ya’ni parallel proeksiyada ushbu masofa cheksiz.

Aytish joizki, markaziy proeksiyalar vizual taassurotni vujudga keltiradi, ya’ni fotografiya sistemalari yoki odamning ko‘rish sistemasi kabi bo‘ladi va ma’lum darajada reallik darajasiga erishiladi. Bu effekt (taassurot) perspektiv qisqartirish deyiladi. Parallel proeksiyalarda tasvirning realligi kamroq, ammo obyektning ayrim haqiqiy o‘lchovlari va paralelligi saqlanadi. Ular asosan muhandislik grafikasida ishlatiladi.

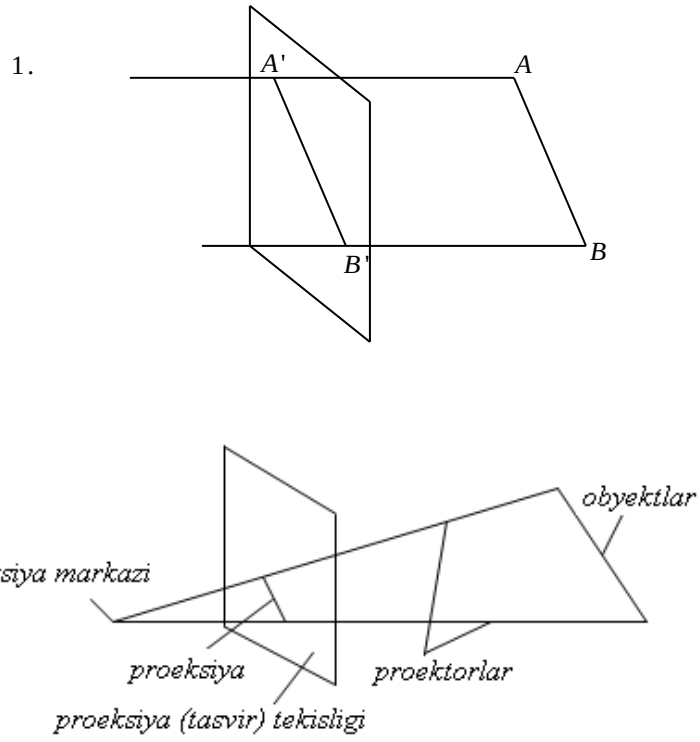
Har turdagi proeksiyalar o‘z navbatida proeksiya tekisligi, proektorlar va koordinatalar sistemasining joylashishiga qarab bir necha turlarga bo‘linadi.

1. <u>Parallel proeksiyalar</u>:



2. Markaziy (perspektiv) proeksiyalar:

Bir nuqtali, ikki nuqtali va uch nuqtali.



1.5-rasm. Parallel va markaziy proeksiyalar.

Proeksiyalashda bir jinsli koordinatalardan va to‘rtinchi tartibdagi geometrik almashtirish matritsalaridan foydalanish geometrik masalalarni echishda ko‘pgina qulayliklarni beradi.

Yuqorida aytib o‘tilgan proeksiyalarni qarab chiqamiz.

Ortografik proeksiyalarda proeksiya tekisligi biror-bir koordinatalar tekisligi bilan ustma-ust yoki parallel bo‘ladi deb olinadi, proektorlar esa unga perpendikulyar bo‘ladi, ya’ni koordinata o‘qlariga nisbatan parallel bo‘ladi.

Abtsissa *X* o‘qi bo‘ylab *YZ* tekisligiga proeksiyalash matritsasi quyidagicha:

$$M_X = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Agar proeksiya tekisligi koordinatalar tekisligiga parallel bo‘lsa, M_X matritsasini ko‘chish matritsasiga ko‘paytirish kerak:

$$M_x \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & 0 & 0 & 1 \end{pmatrix}.$$

Shu kabi ordinata va applikata o‘qlari bo‘ylab proeksiyalarni quyidagicha yozish mumkin:

$$M_Y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & q & 0 & 1 \end{pmatrix},$$

$$M_Z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & r & 1 \end{pmatrix}.$$

Bu yerda *p*, *q*, *r* – mos ravishda proeksiya tekisligidan koordinatalar sistemasining tekisliklarigacha bo‘lgan masofa.

Boshqa so‘z bilan aytganda: obyektning oldi, yon va ustidan ko‘rinishi.

Eslatma. Barcha matritsalarning determinantlari nolga teng.

Aksonometrik proeksiyalarda proeksiya tekisligi koordinata sistemasining tekisliklari bilan usta-ust tushmaydi va proektorlar proeksiya tekisligiga perpendikulyar bo‘ladi.

Proeksiya tekisligi va koordinatalar o‘qlari yo‘nalishiga qarab aksonometrik proeksiya uchta sinfga bo‘linadi :

- *Trimetriya*, ya’ni proeksiya tekisligining normal vektori koordinatalar o‘qlari bilan o‘zaro har xil burchaklarni tashkil qiladi,

- *Dimetriya*, ikkita burchaklari o‘zaro teng,

- *Izometriya*, barcha burchaklar o‘zaro teng.

Proeksiyalash matritsasi ordinat o‘qi bo‘ylab ψ burchakka, abtsissa o‘qi bo‘ylab φ burchakka burish va so‘ng applikata o‘qi bo‘ylab ortografik proeksiyalash orqali hosil qilinadi.

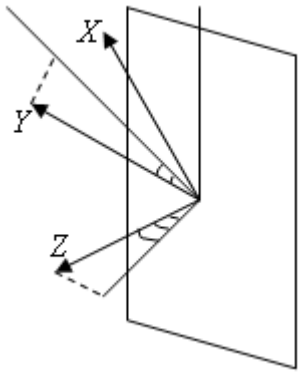
$$M = \begin{pmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos\psi & \sin\varphi\sin\psi & 0 & 0 \\ 0 & \cos\varphi & 0 & 0 \\ \sin\psi & -\sin\varphi\cos\varphi & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Bu yerda X, Y, Z o‘qlari bo‘ylab bazis vektori quyidagicha o‘zgaradi.

$(1,0,0,1)\cdot M = (\cos\psi, \sin\varphi\sin\psi, 0, 1),$

$(0,1,0,1)\cdot M = (0, \cos\varphi, 0, 1),$

$(0,0,1,1)\cdot M = (\sin\psi, -\sin\varphi, \cos\psi, 0, 1).$



1.6-rasm. Oksonometrik proeksiya.

Aytish joizki dimetriya holida:

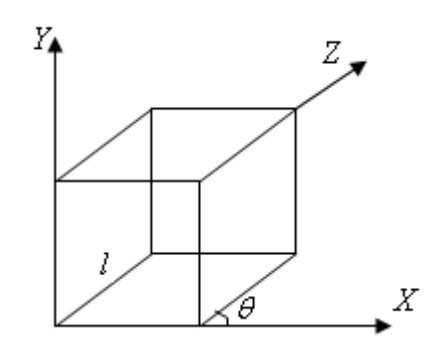
$$\sin^2\psi = tg^2\varphi.$$

Izometriyada esa:

$$\sin^2\varphi = \frac{1}{3}, \sin^2\psi = \frac{1}{2}.$$

Qiya burchakli proeksiyalashda proektorlar proeksiya tekisligiga perpendikulyar emas.

Proeksiyalash matritsasi bu holda quyidagi ko‘rinishga ega:



1.7-rasm. Kubning qiya burchakli proeksiyasi.

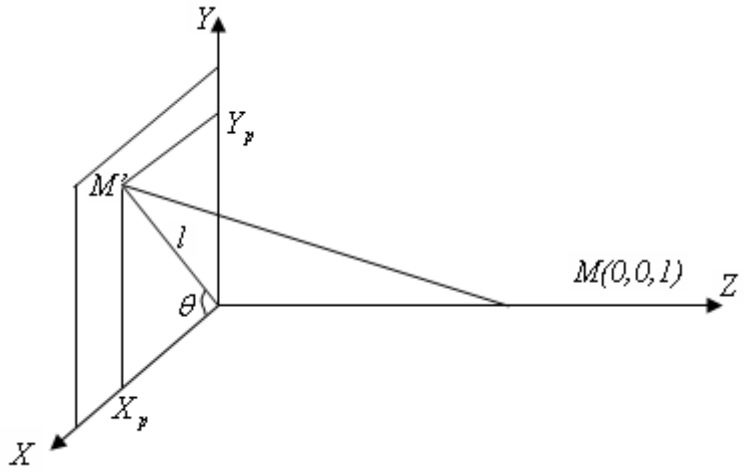
$$K = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \alpha & \beta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Qiya burchakli proeksiyalarni ikkita sinfga ajratiladi: Kavale (erkin) proeksiya va Kabinet proeksiyasi.

Kavale proeksiyasida: $\alpha = \beta = \cos\frac{\pi}{4}.$

Kabine proeksiyasida: $\alpha = \beta = \frac{1}{2}\cos\frac{\pi}{4}.$

z o‘qi bo‘ylab yo‘nalgan ortning (birlik vektor) qiya burchakli proeksiyasini ko‘ramiz.



1.8-rasm. Qiya burchakli proeksiya.

Bu biz ko‘rayotgan $M(0,0,1) \to M'(l\cos\theta, l\sin\theta, 0).$

Umumiy holda $M(x,y,z)$ nuqtani qiya burchakli akslantirishni quyidagicha ifodalash mumkin:

$$X_p = X + Z(l\cos\theta), \quad Y_p = Y + Z(l\sin\theta).$$

Proeksiyalash matritsa-K ni quyidagicha yozish mumkin:

$$K = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ l\cos\theta & l\sin\theta & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Kavale : $l = 1, \theta = 45^0,$

Kabine: $l = \frac{1}{2}, \theta = 45^0$

1.11. Markaziy proeksiyalash.

Markaziy (perspektiv) proeksiyalash: bir nuqtali, ikki nuqtali, uch nuqtali proeksiyalash

Faraz qilamizki proeksiya markazi Z o‘qida yotib $C(0,0,c)$ nuqtada joylashgan bulsin. Proeksiya tekisligi XY tekisligi bilan ustma-ust joylashgan bo‘lsin. Proeksiya markazidan proeksiya tekislikigacha masofa $d = c$ ga teng. Fazoda $M(x,y,z)$ nuqta olamiz va uni markaz bilan $C(0,0,c)$ tutashtiruvchi to‘g‘ri chiziq o‘tkazamiz. Ushbu to‘g‘ri chiziqni parametrik tenglamasini tuzamiz:

$x' = xt,\ y' = yt,\ z' = c + (z \cdot c)t.\quad z' = 0$ shartiga ko‘ra $t = \dfrac{1}{1 - \dfrac{z}{c}}$ topamiz va bundan foydalangan holda $M(x,y,z)$ proeksiyasining koordinatalarini topamiz :

$$x' = \frac{1}{1 - \frac{z}{c}} x, \qquad y' = \frac{1}{1 - \frac{z}{c}} y.$$

Ushbu natijani proeksiya markazi $C(0,0,c)$ va $M(x,y,z)$ nuqtani tutashtiruvchi to‘g‘ri chiziq tenglamasidan ham olish mumkin, ya’ni

$$\frac{x' - x_1}{x_2 - x_1} = \frac{y' - y_1}{y_2 - y_1} = \frac{z' - z_1}{z_2 - z_1}, \qquad (x_1, y_1, z_1) = (0,0,c), \qquad (x_2, y_2, z_2) = (x, y, z).$$

Bir nuqtali markaziy proeksiyalashning matritsasi quyidagicha:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{c} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

tekshirib ko‘ramiz: $(x, y, z, 1) \cdot P = \left(x, y, 0, 1 - \dfrac{z}{c} \right).$

Bir jinsli koordinatalarning xossalaridan foydalangan holda (ya'ni $\frac{1}{1-\frac{z}{c}}$ ga ko'paytiramiz)

$$M'\left(\frac{x}{1-\frac{z}{c}}, \frac{y}{1-\frac{z}{c}}, 0,1\right).$$

Ushbu akslantirishga mos almashtirish matritsasi quydagicha:

$$P_A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -\frac{1}{c} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad P_A \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = P.$$

Umuman olganda markaz nuqtasi uchta bulishi mumkin va bu holda almashtirish matritsasi:

$$P_Y = \begin{pmatrix} 1 & 0 & 0 & -\frac{1}{a} \\ 0 & 1 & 0 & -\frac{1}{b} \\ 0 & 0 & 1 & -\frac{1}{c} \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Mos ravishda x, y, z o'qlariga parallel to'g'ri chiziqlar dastalari $(1,0,0,0)$, $(0,1,0,0)$, $(0,0,1,0)$ quyidagi markazli $\left(1,0,0,-\frac{1}{a}\right)$, $\left(0,1,0,-\frac{1}{b}\right)$ va $\left(0,0,1,-\frac{1}{c}\right)$ to'g'ri chiziqlar dastasiga o'tadi.

Ularni quyidagicha ham tasvirlash mumkin, mos ravishda: $(-a,0,0,1)$, $(0,-b,0,1)$ va $(0,0,-c,1)$ bular bosh tutashish nuqtalarini aniqlaydi.

Nazorat savollari:

1. Proeksiyadan nima va qanday maqsadda foydalaniladi, uning ta'rifini bering?
2. Parallel va markaziy proeksiyalarni bir-biridan qanday farqlash mumkin?
3. Muhandislik grafikasida proeksiyaning qanday turidan foydalaniladi?

4. Parallel proeksiya qanday turlarga bo‘linadi?
5. Ortografik proeksiyalarda proeksiya koordinata o‘qlariga nisbatan qanday joylashadi?
6. Aksonometrik proeksiya qanday turlarga bo‘linadi va uning koordinata o‘qlariga nisbatan joylashishi?
7. Qiyabukchakli proeksiyalashda proektorlar proeksiya tekisligiga nisbatan qanday joylashadi?
8. Markaziy proeksiyalash haqida umumiy ma’lumot bering?

Tayanch iboralar: Proeksiya, proeksiya markazi, proeksiya tekisligi, parallel va markaziy proeksiya, ortografik, aksonometrik, qiya burchakli proeksiyalash.

1.12. Rastr algoritmlari asoslari. Sohani bo‘yash.

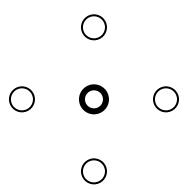
Rastr tushunchasi. Bog‘lanishlik. Tekislikda sohalarni bo‘yash algoritmlari.

Ko‘pgina grafik qurilmalar rastrli, ya’ni tasvirni piksellar (rastr) to‘g‘ri burchakli matritsasi (butun sonlardan tuzilgan setka) ko‘rinishda ifodalaydi. Shu sababli rastr algoritmlariga zaruriyat tug‘iladi. Ammo aytish joizki ko‘pgina grafik kutubxonalarda (modul) yetarlicha oddiy rast algoritmlari mavjud [4, 8, 13].

Rastr (grafikasida) setkasida asosiy tushunchalardan biri bu bog‘lanishlik, ya’ni rastr chizig‘ining ikki qo‘shni (yonma-yon joylashgan) piksellarning bog‘lanish imkonyati. Savol: qachon (x_1, y_1)va (x_2, y_2) piksellar qo‘shni deb hisoblanadi?

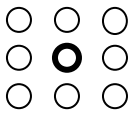
To‘rt bog‘lanishlik. Piksellar qo‘shni deyiladi agar ularning yoki *x* - koordinatalari yoki *y* - kordinatalari, birga (1) farq qilsa, ya’ni:

$$|x_1 - x_2| + |y_1 - y_2| \le 1.$$



Sakkiz bog‘lanishlik. Piksellar qo‘shni deyiladi agar ularning x - va y - koordinatalari birdan ko‘pga farq qilmasa, ya’ni:

$$|x_1 - x_2| + |y_1 - y_2| \le 1.$$



to‘rt bog‘lanishlik tushunchasi sakkiz bog‘lanishdan kuchliroq, ya’ni ikkita to‘rt bog‘lanishlik piksellar har doim sakkiz bog‘lanishlik, teskarisi har doim o‘rinli emas.

Rastr setkasida ixtiyoriy egri chiziq $P_1, P_2, \dots, P_n$ piksellar guruhi orqali ifodalanadi, bu yerda ixtiyoriy ikkita P_i *ва* P_{i+1} qo‘shni piksellar.

Yuqorida keltirilgan ta’riflarga ko‘ra egri chiziq to‘rt bog‘lanishlik va sakkiz bog‘lanishlik bo‘lishi mumkin.

Sohani bo‘yash (rang berish).

Komp’yuter grafikasida soha 2 ta usul bilan berilishi mumkin:

1. Sohani tashkil etuvchi tashqi nuqtalari bilan, ya’ni sohani ichida yotuvchi har bir piksel biror bir rang (oldcolor) bilan beriladi (chegaradagi piksellar bu qiymatga ega emas).

2. Soha chegarasi bilan berilishi mumkin, ya’ni chegaradagi piksellar biror bir rang bilan (bcolor) beriladi (chegara ichidagi piksellar bu qiymatga ega emas). Va shu sababli sohani bo‘yash, algoritmlari ikki turga bo‘linadi.

Bundan tashqari 4 va 8 bog‘lanishlik sohalar uchun algoritmlar mavjud. Ichki oldcolor rang bilan berilgan yangi newcolor rang bilan 4 bog‘lanishlik sohani bo‘yash oddiy rekursiya algoritmini keltiramiz:

```
function fill4 (int x, int y,int newcolor,int oldcolor)
{
    if (getpixel(x,y)==oldcolor)
    {
        putpixel (x,y,newcolor);
        fill4 (x,y-1,newcolor,oldcolor);
        fill4 (x,y+1,newcolor,oldcolor);
```

```
            fill4 (x-1,y,newcolor,oldcolor);
            fill4 (x+1,y,newcolor,oldcolor);
      } }
function fill4 (int x, int y,int newcolor,int oldcolor)
{
      if (getpixel(x,y)==oldcolor)
      {
            putpixel (x,y,newcolor);
            fill4 (x,y-1,newcolor,oldcolor);
            fill4 (x,y+1,newcolor,oldcolor);
            fill4 (x-1,y,newcolor,oldcolor);
            fill4 (x+1,y,newcolor,oldcolor);
      }
}
```

Bu yerda, (*x*,*y*) ixtiyoriy sohani ichida yotuvchi nuqta, oldcolor qiymatiga ega piksel.

Chegaradagi rangi bilan berilgan (bcolor) sohani bo‘yash algoritmi quyidagicha:

```
#include <iostream.h>
#include <graphics.h>
#include <conio.h>
#include <dos.h>
void floodfill (int x,int y,char BorderColor,char Newcolor)
{
  if (getpixel(x,y)!=BorderColor)
  {
 if(getpixel(x,y)!=Newcolor)
 {
    putpixel (x,y,Newcolor);
    floodfill (x-1,y,BorderColor,Newcolor);
```

```
        delay (10);
        floodfill (x+1,y,BorderColor,Newcolor);
        delay (10);
        floodfill (x,y-1,BorderColor,Newcolor);
        delay (10);
        floodfill (x,y+1,BorderColor,Newcolor);
    }
     }
}
void main()
{int gd=0,gm,errorcode;
initgraph(&gd,&gm,"c:\\borlandc\\bgi");
   circle (300,300,20);
   floodfill (300,300,15,10);
   getch();
   closegraph();
}
```

Bu yerda, (*x,y*) - sohani ichida yotuvchi biror bir nuqta (piksel), newcolor-bo‘yash rangi.

Keltirilgan algoritmlarni 8 bog‘lanishlik sohalarga 4 ta yo‘nalishni 8 ta yo‘nalishga almashtirish orqali osongina o‘tkazish mumkin.

1.13. Rastr algoritmlari asoslari. Brezenxeym va Sazerland-Koxen algoritmlari.

Rastr tushunchasi. Kesma, aylana, ellips uchun Brezenxeym algoritmlari. Sazerlend-Koxen algoritmlari

Brezenxeym algoritmi. Kesmaning rastr tasviri.

(X_1, Y_1) va (X_2, Y_2) nuqtalarini tutashtiruvchi kesmaning rastr tasvirini ko‘rish masalasini ko‘ramiz.

Faraz qilamizki $0 \le y_1 \le y_2 \le x_1 \le x_2$.

Berilgan ikki nuqtadan o‘tuvchi to‘g‘ri chiziq tenglamasini tuzamiz:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1},$$

Unda kesma quyidagi tenglama bilan beriladi:

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1), \quad x \in [x_1, x_2].$$

yoki: $y = ky + b,$ bu yerda $k = \frac{y_2 - y_1}{x_2 - x_1}, \quad b = y_1 - kx_1.$

$$d_i = 2dyx_{i-1} - 2dxy_{i-1} + 2dy - dx,$$

keyingi qadamga, ya’ni *i+1:*

$$d_{i+1} = 2dyx_i - 2dxy_i + 2dy - dx$$

d_{i+1} dan d_i ayiramiz va $x_i - x_{i-1} = 1$ ni hisobga olgan holda:

$$d_{i+1} = d_i + 2dy - 2dx(y_i - y_{i-1})$$

So‘ng, agar $d_i < 0$ bo‘lsa s_i tanlanadi, u holda $y_i = y_{i-1}$ va $d_{i+1} = d_i + 2dy$.

Aks holda, ya’ni $d_i \ge 0$ bo‘lsa s_i tanlanadi, va u holda $y_i - y_{i-1} = 1$.

$$d_{i+1} = d_i + 2(dy - dx).$$

Shunday qilib biz d_{i+1} ni d_i ning qiymati orqali hisoblash va s_i, T_i nuqtalarni tanlash uchun iterativ usulni hosil qildik. Boshlang‘ich holatda $d_i = 2dy - dx$ $(x_0, y_0) = (0,0)$ ni hisobga olgan holda $i = 1$ da topiladi.

Brezenxeym algoritmi uchun dastur quyidagicha ifodalanadi:

```
#include <iostream.h>
#include <graphics.h>
#include <conio.h>

void brezline(int x1,int y1,int x2,int y2,int c)
{
   int dx,dy,d,d1,d2,x,y;
   dx=x2-x1;
   dy=y2-y1;
   d=2*dy-dx;
   d1=2*dy;
   d2=2*(dy-dx);
   x=x1;
   y=y1;
   putpixel(x,y,c);
   while (x<x2)
   {
  x=x++;
  if (d<0) d=d+d1; else
  {
     y=y++;
     d=d+d2;
  }
  putpixel(x,y,c);
   }
}
void main()
{int gd=0,gm;
initgraph(&gd,&gm,"c:\\borlandc\\bgi");
```

```
    brezline(100,100,200,200,10);
    getch();
    closegraph();
}
```

Aylana uchun Brezenxeym algoritmi.

...

Ellips uchun Brezenxeym algoritmi.

...

Sazerland-Koxen algoritmi. Kesmaning kesilishi.

Kompyuter ekraniga chiqarish kerak bo‘lgan tasvirni biror berilgan chegara bo‘yicha kesilishi keng qo‘llaniladi. Ko‘p hollarda chegara sifatida to‘g‘ri to‘rtburchakli soha ishlatiladi, xususan kompyuter ekrani.

Kesmani biror bir to‘rtburchakli soha bilan kesilish oddiy va qulay algoritmini ko‘rib chiqamiz.

Faraz qilamizki bizga (x_1,y_1) va (x_2,y_2) nuqtalari bilan kesma berilgan bo‘lsin. To‘g‘ri burchakli to‘rtburchak esa quyidagi qiymatlar bilan berilgan bo‘lsin:

$$x_{min}, y_{min}, x_{max}, y_{max}.$$

Xususiy holni ko‘ramiz, ya’ni kesmaning bir uchi to‘g‘ri to‘rtburchakli sohani ichida, ikkinchisi esa tashqarida joylashgan bo‘lsin. Aynan shu holat bizni qiziqtiradi. Bu yerda kesmani soha chegarasi bilan kesilish nuqtasi $M(x,u)$ ni topish kerak.

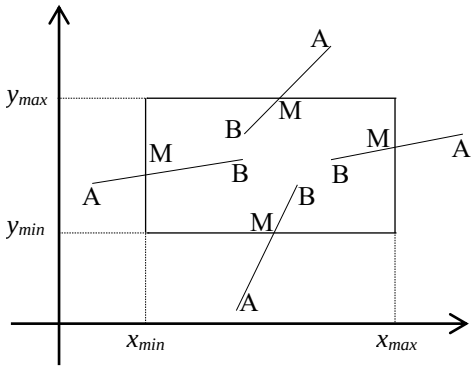
Faraz qilamizki (x_1, y_1) nuqta to‘g‘ri burchakli to‘rtburchak tashqarisida, (x_2, y_2) nuqta esa soha ichida yotsin.

Ushbu masalani echishda (x_1, y_1) va (x_2, y_2) nuqtalaridan o‘tuvchi

$$\frac{x - x_2}{x_1 - x_2} = \frac{y - y_2}{y_1 - y_2}$$

to‘g‘ri chiziq tenglamasidan foydalanamiz.

Qaralayotgan masalada, ya’ni kesilish nuqtasi $M(x,u)$ ni aniqlash jarayonida quyidagi hollar bo‘lishi mumkin:



1.15-rasm. Kesmani to‘g‘ri burchakli to‘rtburchak bilan kesilish xollari.

1. Agar $x_1 < x_{\min}$ u holda $x = x_{\min}$, $\frac{x_{\min} - x_2}{x_1 - x_2} = \frac{y - y_2}{y_1 - y_2} \Rightarrow y = (y_1 - y_2)\frac{x_{\min} - x_2}{x_1 - x_2} + y_2.$

2. Agar $y_1 < y_{\min}$ u holda $y = y_{\min}$, $\frac{x - x_2}{x_1 - x_2} = \frac{y_{\min} - y_2}{y_1 - y_2} \Rightarrow x = (x_1 - x_2)\frac{y_{\min} - y_2}{y_1 - y_2} + x_2.$

3. Agar $x_1 > x_{\max}$ u holda $x = x_{\max}$, $\frac{x_{\max} - x_2}{x_1 - x_2} = \frac{y - y_2}{y_1 - y_2} \Rightarrow y = (y_1 - y_2)\frac{x_{\max} - x_2}{x_1 - x_2} + y_2.$

4. Agar $y_1 > y_{\max}$ u holda $y = y_{\max}$, $\frac{x - x_2}{x_1 - x_2} = \frac{y_{\max} - y_2}{y_1 - y_2} \Rightarrow x = (x_1 - x_2)\frac{y_{\max} - y_2}{y_1 - y_2} + x_2.$

Bu yerda $M(x,y)$ biz qidirayotgan nuqtaning koordinatalari, ya’ni soha bilan kesilgandan so‘ng kesma (x_1,y_1) va (x_2, y_2) nuqtalari orqali ifodalanadi.

Nazorat savollari:

1. Rastr grafikasida bog‘lanishlik tushunchasini izohlang?
2. Bog‘lanishlik qanday turlarga bo‘linadi va ularning farqi?
3. Brezenxeym algoritmini misollar orqali tushunturing?
4. Kesmaning rastr tasvirini misol keltiring?
5. Sohani bo‘yash usullarini misollar orqali berilishini tavsiflang?
6. Komp’yuter grafikasida soha necha hil usulda beriladi?
7. Kesmani kesilishi. Sazerland-Koxen algoritmi tavsiflang?
8. Kesmani kesilishiga oid misollar keltiring?

Tayanch iboralar: To‘rt bog‘lanishlik, sakkiz bog‘lanishlik, kesmaning rastr tasviri, sohani bo‘yash, kesmaning kesilishi.

1.14. Ko‘rinmas chiziqlar olib tashlash.

Ko‘rinmas sirtlarni ajratish va olib tashlash algoritmi. Ko‘rinmas chiziqlarni olib tashlash Roberts algoritmi.

Biror bir uch o‘lchovli obyektni ikki o‘lchovli tekislikda (kompyuter ekranida) qurish uchun avvalo uni qaysi qismlari ko‘rinarli, qaysi qismlari ko‘rinmas, ya’ni obyektning boshqa yoqlari bilan yopiqligini aniqlash kerak. Proeksiyalashda markaziy yoki paralel proeksiyalash ishlatiladi [4, 13].

Proeksiyalashda proektorlar obyektning har bir nuqtasidan o‘tadi. Proeksiyalash yo‘nalishi bo‘yicha tasvir tekisligiga yaqinroq masofadagi nuqtalar ko‘rinadigan hisoblanadi.

Sodda ko‘ringanligiga qaramay ushbu masalani echish ancha qiyinchiliklarga va ayrim hollarda biroz hisob kitoblarga olib keladi. Ushbu masalani echishda kompyuter grafikasida ikkita asosiy yondashuv mavjud:

1. Proeksiyalash yo‘nalishi bo‘yicha tasvir tekisligiga yaqinroq masofada joylashgan obyektning nuqtalarini aniqlash. Bunda displeyning rastr xossalaridan foydalaniladi.
2. Obyektlarni yoki obyekt qismlarini o‘zaro taqqoslab obyektlarni yoki obyekt qismlarini ko‘rinarliligini aniqlash.

   Bu ikki yondashuvni o‘zaro ichiga oluvchi algoritmlar ham mavjud.

Ko‘rinmas yoqlarni ajratish.

Har bir yoqlari uchun tashqi birlik normal vektori *n* berilgan ko‘p yoqlikni ko‘ramiz.

Agar yoqning normal vektori *n* va proeksiyalash yo‘nalishini beruvchi vektor *l* o‘rtasidagi burchak o‘tmas bo‘lsa, u holda qaralayotgan yoq ko‘rinmaydi va ko‘rinmas yoq deb ataladi. Agar mos bo‘lgan burchak o‘tkir bo‘lsa, u holda qaralayotgan yoq ko‘rinadigan yoq deyiladi. Parallel proeksiyalashda burchakka qo‘yiladigan shartni quyidagicha yozish mumkin:

$$(n, l) = (n_1 l_1 + n_2 l_2 + n_3 l_3) \le 0 .$$

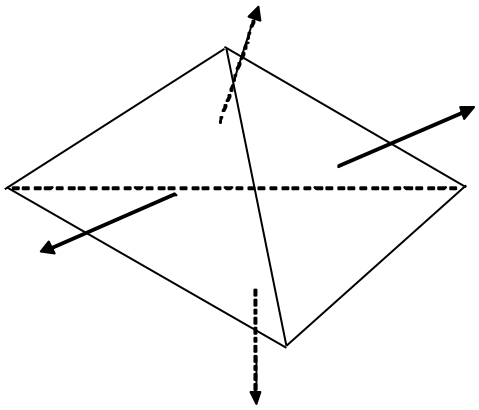
Ushbu shart bajarilsa yoq ko‘rinmas.

Yoqning ixtiyoriy *R* nuqtasini markazi C nuqtada joylashgan markaziy proeksiyalashning yo‘nalish vektori quyidagicha topiladi:

$$L=C-P.$$

Va so‘ng yoqning ixtiyoriy *R* nuqtasi uchun shart tekshiriladi.

$$(n,l)\le 0.$$



1.16-rasm. Normallarning ko‘rinishi.

Ko‘rinmas chiziqlarni (qirralarni) olib tashlash. Robert algoritmi.

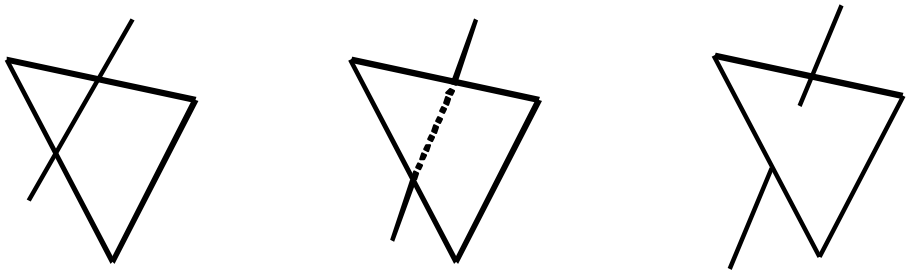
Qavariq ko‘pburchaklardan tuzilgan obyektning ko‘rinmas qirralarinini olib tashlash Robert algoritmi hisoblanadi. Ushbu algoritmni keltirib o‘tamiz.

Dastlab ikkita aniqlovchi yoqlarni ko‘rinmas bo‘lgan qirralari olib tashlanadi. Keyingi qadamlarda qolgan qirralar har bir yoqlar bilan yopiqlikka tekshiriladi. Bunda uchta holat mavjud va ular alohida tekshiriladi:

1. Yoq qirrani yopmaydi, bu holda qirra olib tashlanmaydi.

2. Yoq qirrani to‘liq yopadi, bu holda qirra olib tashlanadi.

3. Yoq qirrani qisman yopadi, bu holda qirra bir necha bo‘laklarga bo‘linadi. Qirra ko‘rilgan qirralar ro‘yxatiga qirraning yoq bilan yopilmaydigan qismlari qo‘yiladi.



1.17-rasm. Kesmani ko‘pburchak bilan opilish xolari.

1.15. Ko‘rinmas sirtlarni olib tashlash.

Appel algoritmi. Tartiblash algoritmlari. Ko‘rinmas qismlarni olib tashlash Z-bufer usuli. Varnok algoritmi

Ko‘rinmas yoqlarni chiqarib yuborish. Z bufer usuli.

Ko‘rinmas chiziq va sirtlarni olib tashlash algoritmlaridan biri bu Z bufer usuli hisoblanadi.

Bu usul bir xil yondashishga to‘g‘ri keladi va har bir nuqta bilan alohida ishlanadi. Tasvir tekisligidagi har bir nuqtaga (pikselga) (*x*,*y*) rangdan tashqari u xotirada saqlanadi. Dastlab uni (chuqurlik) +∞ teng deb hisoblaymiz. Ixtiyoriy yoqni tasvir tekistligiga tasvirlash uchun uning har bir pikseli uchun Z chuqurligi hisoblanadi. Agar u dastlabki chuqurlikdan kichik bo‘lsa, bu qiymat Z buferiga kiritiladi va eski qiymati chiqarilib yuboriladi. Va Z buferidagi piksellar ekranda chiqariladi. Qo‘shni piksellarning Z chuqurligini hisoblashda Brezenxeym algoritimidan foydalanish tavsiya etiladi. Aytish joizki Z koordinasiya qiymati obyektlarning yorug‘ligini berishda yoki ularni umuman olib tashlashda keng qo‘llaniladi.

Tartiblash algoritmlari. Chuqurligi bo‘yicha tartiblash usuli.

Yoqlarni tartiblashning eng oddiy algoritmi bu ularning proeksiyalash yo‘nalishi bo‘yicha tasvir tekisligigacha bo‘lgan minimal masofa bo‘yicha tartiblash hisoblanadi, qaysiki ularni yakinlashish tartibida chiqarish maqsadida.

OZ o‘qi bo‘yicha parallel proeksiyalashni ko‘ramiz. Faraz qilamizki bizga *R* va *Q* yoqlar berilgan bo‘lsin. Ularni tasvir tekisligida (kompyuter ekranida) tartiblangan holda chiqarish uchun 5 ta shartni tekshirish tavsiya etiladi. Ularni tekshirish murakkabligi oshishi tartibida keltiramiz:

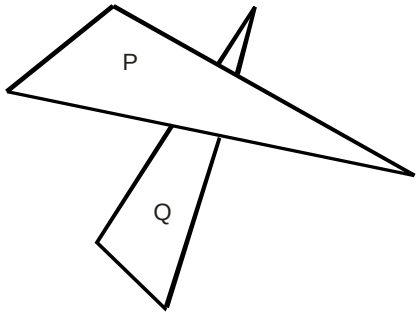
1. *OX* o‘qidagi yoqlarni proeksiyalari kesishadimi?
2. *OY* o‘qidagi ularning proeksiyalari kesishadimi?

3. *R* yoki *Q* yoqidan o‘tuvchi tekislikdan nisbatan koordinatalar boshi yotadigan tomonida yotmaydi.

4. *Q* yoki *P* yoqidan o‘tuvchi tekislikga nisbatan koordinatalar boshi yotadigan tomonida yotadi.

5. Yoqlarning tasvir tekisligidagi proeksiyalari o‘zaro kesishadi.

Agar keltirilgan shartlardan birortasi inkor bo‘lsa *R* yoki *Q* yoqiga nisbatan tasvir tekisligida yaqinroq joylashadi va quyidagicha tasvirlanadi:



1.18-rasm. Poligonlarning tartiblanishi.

Varnok algoritmi.

Varnok algoritmi tasvir tekisligini to‘rt qismga bo‘lishga asoslangan va har bir qismi uchun algoritm oson echiladi.

Ekran to‘rt qismga bo‘linadi. Agar qism eng yaqin yoq proeksiyasi bilan to‘liq yopilsa yoki birorta ham yoqning proeksiyasi bilan yopilmasa unda masala yopiladi, ya’ni to‘liq bo‘yaladi yoki chetlashtiriladi. Agar ikkala shart ham bajarilmasa u holda qism yana to‘rt qismga bo‘linadi va shartlar tekshiriladi. Ushbu jarayon qismning o‘lchovi bir pikseldan kichik bo‘lgunga qadar bajariladi.

Nazorat savollari:

1. Ko‘rinmas chiziq va sirtlarni olib tashlashga bo‘lgan asosiy yondashuvlar.
2. Ko‘rinmas yoqlarni ajratish metodi.
3. Ko‘rinmas qirralarni olib tashlashda Robert algoritmining qo‘llanilishi.
4. Ko‘rinmas yoqlarni chiqarib yuborishda Z bufer usulining qo‘llanilishi.
5. Tartiblash algoritmlarining tavsifi.

6. Varnok algoritmi nimaga asoslanadi?

Tayanch iboralar: Ko‘rinmas yoqlar, ko‘rinmas chiziqlar, Z bufer, tartiblash algoritmi, Varnok algoritmi.

1.16. Geometrik ob’yektlarni nur bilan kesisish algoritmlari.

Sferani, tekislikni, qavariq ko‘pburchakni, nur bilan kesishish algoritmlari

Nurni oddiy geometrik obyektlar bilan kesilish nuqtalarini topishning qulay algoritmlari kompyuter grafikasida juda ko‘p qo‘llaniladi.

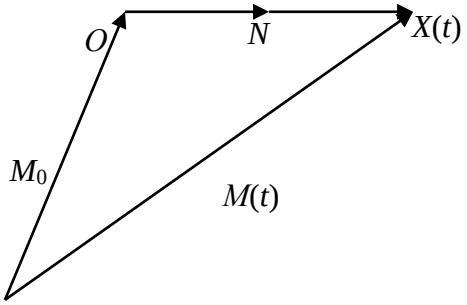
Nurni sfera, tekislik va to‘g‘ri to‘rtburchakli parallelepiped bilan kesilish nuqtalarini topishning qulay algoritmlarini ko‘rib chiqamiz.

M_0= (x_0,y_0,z_0) nuqtadan chiquvchi N=(l,m,n) yo‘nalishli vektor bilan ifodalanuvchi nurning vektor-parametrik tenglamasi quyidagi ko‘rinishda

$$M(t) = M_0 + N \cdot t, \quad t > 0,$$

Yoki koordinatali parametrik tenglama orqali ifodalanadi:

$$\begin{cases} x = x_0 + l \cdot t, \\ y = y_0 + m \cdot t, \quad (t > 0). \\ z = z_0 + n \cdot t, \end{cases} \tag{7}$$



1.19-rasm. Nur.

Agar N-birlik vektor bo‘lsa

$$l^2 + m^2 + n^2 = 1$$

bu holda t parametri oddiy geometrik manoga ega bo‘ladi: t ning qiymati nurning boshlang‘ich nuqtasi O dan $X(t)$ nuqtasigacha bo‘lgan masofaga teng.

Ixtiyoriy vektorni birlik vektorga olib kelish uchun uning har bir koordinatasini uning uzunligiga bo‘lish kerak.

Sferaning nur bilan kesilishi.

$C(x_c,y_c,z_c)$ markazi va r-radiusi bilan beriluvchi sferaning tenglamasi quyidagicha ifodalanadi:

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = r^2. \quad (8)$$

Sferani (7) ifoda bilan berilgan nur orqali kesilishi nuqtalarini aniqlash uchun x,y,z qiymatlarini (8) tenglamaga qo‘yamiz:

$$(x_0 + lt - x_c)^2 + (y_0 + mt - y_c)^2 + (z_0 + nt - z_c)^2 = r^2$$

soddalashtirib quyidagi kvadrat tenglamaga kelamiz:

$$at^2 + 2bt + c = 0, \quad (9)$$

bu yerda:

$$a = l^2 + m^2 + n^2.$$

(N-birlik vektori uchun);

$$b = l(x_0 - x_c) + m(y_0 - y_c) + n(z_0 - z_c),$$
$$c = (x_0 - x_c)^2 + (y_0 - y_c)^2 + (z_0 - z_c)^2 - r^2.$$

Kvadrat tenglamaning echimlari

$$t_{\pm} = -b \pm \sqrt{b^2 - c}, \ (a = 1).$$

Agar $D = b^2 - c < 0$ bo‘lsa , u holda nur sferani kesmaydi.

Aks holda, ya’ni $D \geq 0$ bo‘lsa

$$t^*_- = -b \pm \sqrt{b^2 - c}, \ \ t^*_+ = -b \pm \sqrt{b^2 - c}$$

topiladi, va $t^*_{\pm} > 0$ tekshiriladi.

Nurni boshiga yaqinroq kesilish nuqtasini topish uchun ulardan kichikroq qiymati aniqlanadi, ya’ni

$$t^* = \min(t^*_-, t^*_+)$$

Nuqtaning koordinatalarini topish uchun nurning tenglamasidan foydalanamiz

$$\begin{cases} x^* = x_0 - lt^*, \\ y^* = y_0 + mt^*, \\ z^* = z_0 + nt^*, \end{cases}$$

va $M^*(x^*, y^*, z^*)$ nuqtasi topiladi sferaning ushbu nuqtadagi birlik normal vektori quydagicha topiladi:

$$N = \frac{1}{r}\left(x^* - x_C,\ y^* - y_C,\ z^* - z_C\right).$$

Tekislikni nur bilan kesilishi.

Faraz qilamizki tekislik umumiy tenglamasi bilan berilgan bo‘lsin

$$ax+by+cz+d = 0. \qquad (10)$$

Bu yerda $N(a,b,c)$ - tekislikning normal vektori. Agar N vektor birlik, $a^2+b^2+c^2=1$ bo‘lsa, d tekislikdan (0,0,0) markazgacha bulgan masofa.

(10) tenglamada x,y,z qiymatlarini nurning tenglamasi orqali ifodalasak , u holda t ga nisbatan chiziqli tenglamani olamiz, ya’ni

$$a(x_0 + lt) + b(y_0 + mt) + c(z_0 + nt) + d = 0.$$

Bu yerda

$$t^* = -((ax_0 + by_0 + cz_0 + d)) /(al + bm + cn),$$

yoki vektor ko‘rinishida

$$t^* = (-((A_0, M) + d)) /(A, N).$$

Agar skalyar ko‘paytma $(A, N) = al + bm + cn = 0$ bo‘lsa, nur tekislikka paralel va uni kesmaydi.

Aks holda $(A, N) \neq 0$ va $t^* > 0$ bo‘lsa, kesilish nuqtasini quyidagicha topamiz.

$$\begin{cases} x^* = x_0 + lt^*, \\ y^* = y_0 + mt^*, \\ z^* = z_0 + nt^*, \end{cases}$$

va $M^*(x^*, y^*, z^*)$ nuqtasi topiladi.

Ushbu nuqtada normal vektor quyidagicha olinadi:

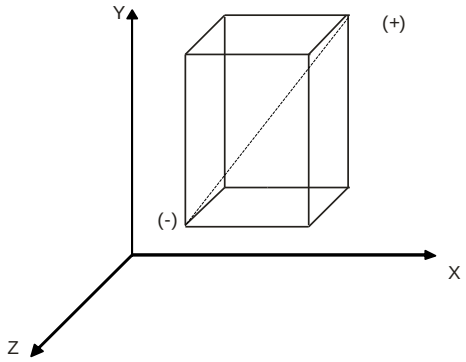
A (agar $(A, N) < 0$), yoki -A (agar $(A, N) > 0$).

1.17. Geometrik obektlarni nur bilan kesisish algoritmlari.

Paralellopipedni, silindrni, konusni nur bilan kesishish algoritmlari.

Parallelepipedni nur bilan kesilishi.

To‘g‘ri burchakli va tamonlari koordinata tekisliklariga parallel bo‘lgan parallelepipedni ko‘ramiz. U holda uning ixtiyoriy birorta diagonaliga tegishli uchlari orqali ifodalash mumkin, ya’ni $(x_-, y_-, z_-), (x_+, y_+, z_+)$, bu yerda $x_- < x_+$, $y_- < y_+$, $z_- < z_+$.



1.20-rasm. Parallelepiped.

$M_0 = (x_0, y_0, z_0)$ nuqtadan chiquvchi va $N=(l,m,n)$ yo‘nalishli vektori, bu yerda $(l^2+m^2+n^2=1)$, orqali beriluvchi nurni ko‘rib chiqamiz.

Parallelepipedning qarama-qarshi tomonlari koordinata tekisliklarga paralel. Dastlab, *YZ* tekisligiga paralel tomonlarini ko‘ramiz, ya’ni $x=x_-$ va $x=x_+$ tekisliklarini.

Agar $l=0$ bo‘lsa, nur tekisliklarga paralel va $x_0<x_-$ yoki $x_0>x_+$ holda parallelepipedni kesmaydi. Agar $l \neq 0$ bo‘lsa, nur tekisliklarga paralel emas va

$$t_{1x} = (x_- - x_0)/l, \quad t_{2x} = (x_+ - x_0)/l.$$

Hisoblanadi va ularda eng kichik yoki kattasi belgilanadi:

$$t_{near} = \min(t_{1x} t_{2x}), t_{far} = \max(t_{1x} t_{2x})$$

Shu kabi *XY* va *XZ* tekisliklarga parallel tomonlari tekshiriladi va $0< t_{near} <t_{far}$ yoki $0<t_{far}$ bo‘lsa nur paralellepipedni kesadi.

Silindrni nur bilan kesishish algoritmi.

…

Konusni nur bilan kesishish algoritmi.

…

Nazorat savollari:

1. Kompyuter grafikasida nurni geometrik obyektlar bilan kesilishi.
2. Sferaning nur bilan kesilishi tavsifi.
3. Sferaning nur bilan kesilishiga oid misol keltiring?
4. Tekislikning nur bilan kesilishi tavsifi.
5. Tekislikning nur bilan kesilishiga oid misol keltiring?
6. Parallelepipedni nur bilan kesilishi tavsifi.
7. Parallelepipedni nur bilan kesilishiga oid misol keltiring?

Tayanch iboralar: Nurning kesilishi, sferaning nur bilan kesilishi, tekislikni nur bilan kesilishi, parallelepipedni nur bilan kesilishi.

1.18. Yorug‘lik.

Yorug‘lik. Simmetrik qaytish, diffuzion qaytish,ideal sinish, diffuzion sinish, energiya taqsimoti, nur yo‘nalishning asosiy modellarini o‘rganish

...

1.19. Bo‘yash usullari.

Fazoda bo‘yash usullari. Diffuzion qaytish, Simetrik aks,

Real tasvirlarni yaratishning keyingi qadami bu qurilgan obyektlarni chegaralovchi sirtlarni bo‘yash masalasini echish. Bo‘yash ko‘rinmas chiziq va sirtlarni olib tashlashdan so‘ngi tartibda bajariladi. Bo‘yashning bir necha oddiy usullarini (modellarini) ko‘ramiz.

Yorug‘lik nuqtasidan sirtga tushuvchi yorug‘lik energiyasi singishi, qaytishi (aks etish) yoki o‘tkazib yuborishi mumkin. Singigan, qaytgan yoki o‘tkazib yuborilgan energiya miqdori yorug‘lik to‘lqinining uzunligiga bog‘liq.

Diffuzion qaytish. Yorug‘likning barcha yunalishlar bo‘yicha tekis tarqalishi. Qaytgan yorug‘likning xossalari yorug‘lik manbasining shakli, yo‘nalishiga va yana yoritalayotgan sirtning joylashishiga va uning xossalariga bog‘liq.

Ideal tarqatuvchidan nuqtaviy manbaning yorug‘ligi Lamberning kosinuslar qonuniga asosan qaytariladi.

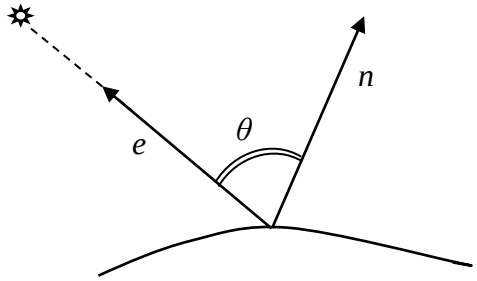
$$I = I_e * k_d * \cos\theta, \qquad 0 \le \theta \le \frac{\pi}{2} \tag{11}$$

Bu yerda I-qaytgan yorug‘likning intensivligi;

I_e – nuqtaviy manbaning intensivligi;

k_d – diffuzion qaytishning koeffisienti (const, $0 \le K_d \le 1$);

θ – yorug‘lik manbasiga e va sirtning (tashqi) normali n yo‘nalishlari o‘rtasidagi burchak.



1.21-rasm. Nurning qaytishi.

Yorug‘lik manbasiga *e* va sirtning (tashqi) normali *n* yo‘nalishlarining birlik vektorlaridan foydalangan holda (11) munosabatni quyidagi ko‘rinishda yozish mumkin:

$$I = I_e \cdot K_d \cdot (n \cdot e). \tag{12}$$

Real sahna obyektlariga bundan tashqari obyektlarning qaytishi egriligiga mos sochilgan (tekis tarqalgan) yorug‘lik tushadi. Bu holda intensivlik quyidagicha hisoblanadi:

$$I = I_a \cdot k_a + I_e K_d \cos\theta, \tag{13}$$

bu yerda I_a - sochilgan yorug‘likning intensivligi; k_a – sochilgan yorug‘likning diffuzion koeffisienti (*const*, $0 \le k_a \le 1,$).

Albatta yorug‘likning intensivligi obyektdan yorug‘lik manbasigacha bulgan masofa *d* ga bog‘liq. Buni hisobga olgan holda quyidagi yoritish modelini olamiz:

$$I = I_a \cdot k_a + \frac{I_e k_d \cos\theta}{d + K} \cdot \cos\theta, \tag{14}$$

bu yerda *K*- ixtiyoriy konstanta.

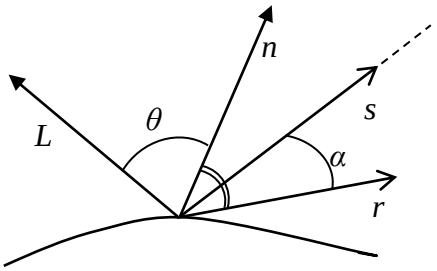
Simmetrik aks obyektning (tashqi) sirti yaltiroq bo‘lganda hosil bo‘ladi. Simmetrik akslangan yorug‘likning intensivligi tushish burchagi to‘lqinning uzunligi va moddaning xossasiga bog‘liq. Simmetrik aksning fizik xossalari juda murakkab, shu sababli kompyuter grafikasida oddiy modellardan foydalaniladi.

Fong modeli.

$$I_f = I_e k_s \cos^{P_d}, \tag{15}$$

bu yerda k_s – eksperimental doimiy (yorug‘likning simmetrik akslanuvchi qismi); α-aks nuri *r* va *S* orasidagi burchak; *R*-yorug‘likning fazoviy taqsimotining aproksimasiyalovchi darajasi.

(14) va (15) larni birlashtirgan holda obyekt sirtining nuqtalari intensivligini hisoblovchi yoritish modelini olamiz.



1.22-rasm. Nurning qaytishi.

$$I = I_a k_a + \frac{I_e}{d+K}(k_d \cos\theta + k_S \cos^P \alpha). \quad (16)$$

Tashqi normal vektori *n*, yorug‘lik manbasiga *L*, aks nuri *r*, kuzatish *s* yo‘nalishlari vektorlarining birlik vektorlaridan foydalangan holda (16) ni quyidagi ko‘rinishda yozish mumkin:

$$I = I_a k_a + \frac{I_e}{d+K}(k_d (n * l = L) + k_S (r * s)^P). \quad (17)$$

Rangli tasvirni hosil qilish uchun, har bir asosiy ranglar uchun (red-qizil, green-yashil, blue-ko‘k) bo‘yash funksiyasini topish zarur.

Simmetrik aksda doimiy R_s har bir keltirilgan rang uchun bir xil hisoblanadi. Agar nuqtaviy manbalar soni bir nechta bo‘lsa (*m*) unda yoritish modeli quyidagicha aniqlanadi:

$$I = I_a k_a + \sum_{j=1}^{m} \frac{I_{ei}}{d+K}(k_d \cos\theta_i + k_s \cos^{P_i} \alpha_i). \quad (18)$$

1.20. Bo‘yash. Fong, Guro usullari.

Guro bo‘yash usuli, Fong bo‘yash usuli.

Guro usuli.

Bu usul uchlarning yorug‘liklari aniqligiga asoslangan holda ularning qiymatlarini bir chiziqli interpolyasiyalash orqali butun yoqning yorug‘lik qiymatlarini topishga asoslangan. Qavariq to‘rtburchakli yoqni quramiz. Faraz qilamizki V_1,V_2,V_3,V_4 uchlarida mos I_{V1}, I_{V2}, I_{V3}, I_{V4} intensivliklar berilgan. Yoqda

ixtiyoriy *W* nuqtasini olamiz. Ushbu nuqtalardan o‘tuvchi gorizontal to‘g‘ri chiziqni o‘tkazib yoqning chegarasi bilan kesishish nuqtalarini belgilaymiz *U* va *V*.

Faraz qilamizki intensivlik kesmada chiziqli o‘zgaradi, ya’ni:

$$I_w = (1-t)I_u + tI_v$$

bu yerda: $t = \dfrac{|UW|}{|UV|}, \quad 0 \le t \le 1.$

Shu kabi *U* va *V* nuqtalardagi intensivliklarni yozamiz, ya’ni ular yoqni uchlarining intensivliklari orqali ifodalanadilar.

$$I_u = (1-U)I_{V4} + UI_{V1}$$
$$I_V = (1-V)I_{V1} + UI_{V2}$$

Bu yerda $U = \dfrac{|V_4U|}{|V_1V_4|}, \ 0 \le U \le 1$

$V = \dfrac{|V_1V|}{|V_1V_2|}, \ 0 \le V \le 1.$

Fong usuli.

Fong usuli har bir nuqtada normal vektorni hisoblashdan iborat, so‘ng qaralayotgan nuqtadagi yorug‘lik intensivligi (16) chi formulaga asosan hisoblanadi. Bu yerda interpolyasiya sxemasi Guro bo‘yash interpolyasiyasiga o‘xshaydi.

W nuqtaning normal vektorini n_w topish uchun ushbu nuqtadan gorizoantal to‘g‘ri chiziqni o‘tkazamiz va yoqning qirrasini kesuvchi nuqtalarning *U* va *V* normal vektorlaridan foydalanilgan holda topamiz:

$$n_w = \frac{(1-t)n_u + tn_v}{(1-t)n_u + tn_v} \quad \text{bu yerda } t = \frac{|uw|}{|uv|}, \ 0 \le t \le 1.$$

U va *V* nuqtalarda normal vektorlarni topish uchun mos qirralarning uchlarini normal vektorlardan foydalanamiz.

$$N_u = (1-u)nv_4 + unv_1$$
$$N_v = (1-v)nv_1 + vnv_2$$

Bu yerda:

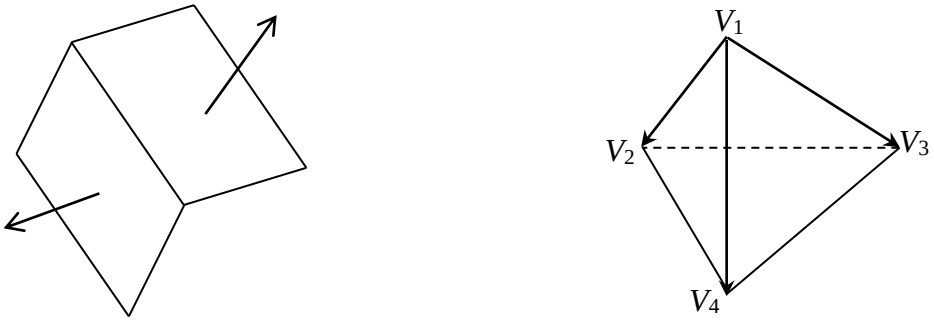
$$u = \frac{|v_4u|}{|v_4v_1|}, \ v = \frac{|v_1v|}{|v_1v_2|}$$

Fonga usuli orqali bo‘yashda tasvir Guro usuliga nisbatan realroq bo‘ladi, ammo hisob-kitoblar sezilarli darajada ko‘p hajmni talab qiladi.

Qirralarda va uchlarda normallarni aniqlash.

Faraz qilamizki bitta nuqtada tutashuvchi qirralar yotadigan tekisliklar tenglamalari quyidagicha:

$$A_i x + B_i y + C_i z + D_i = 0\ ,\ \ i = 1,..., m\ .$$



a) b)

1.23-rasm. Qirra va uchlar yo‘nalishini aniqlash.

Ushbu tekisliklarning normal vektorlari mos ravishda:

$$(A_i, B_i, C_i)\ ,\ \ i = 1,..., m\ .$$

Agar ular tashqi normal vektorni bermasa ularning koordinatalarini ishorasini o‘zgartirishi kifoya. Taqribiy normal vektorning yo‘nalishini aniqlovchi vektor quyidagicha topiladi:

$$(A, B, C) = \sum_{i=1}^{m} (A_i, B_i, C_i)$$

2.23-rasmda ko‘rsatilgan V_1 uning tashqi normal vektorini topish uchun mos vektor ko‘paytmalar yig‘indisi hisoblanadi:

$$V_1 V_2 * V_1 V_3 \qquad V_1 V_3 * V_1 V_4 \qquad V_1 V_4 * V_1 V_2$$

ya’ni

$$n_{V1} = V_1V_2 * V_1V_3 + V_1V_3 * V_1V_4 + V_1V_4 * V_1V_2\ .$$

Nazorat savollari:

1. Yorug‘likning diffuzion qaytishini tavsiflang?
2. Diffuzion qaytishning koeffisienti deganda nima tushuniladi?
3. Yorug‘likning intensivligi nima?
4. Yorug‘lik manbasi deganda nimani tushunasiz?
5. Simmetrik aks qanday vaziyatda hosil bo’ladi?
6. Fong modelini tushuntirish?
7. Guro va Fong usullarini tavsiflang?
8. Qirralarda va uchlarda normallarni aniqlash usullari.

Tayanch iboralar: Bo‘yash, diffuzion qaytish, yorug‘lik manbasi, manbaning intensivligi, diffuzion koeffisient, simmetrik aks.

Nazorat savollari:

1. Nurni yo‘nalishini kuzatish usullari qanday vaziyatlarda qo’llaniladi?
2. Nurlarni to‘g‘ri yo‘nalishini kuzatish deb nimaga aytiladi?
3. Simmetrik qaytishni tushuntiring?
4. Diffuzion qaytishni tushuntiring?
5. Ideal sinish deganda nima tushuniladi?
6. Diffuzion sinishga misol keltiring?
7. Energiya taqsimoti nima, unga izoh bering?
8. Nur yo‘nalishini kuzatishning asosiy modeli.

Tayanch iboralar: Simmetrik qaytish, ideal sinish, diffuzion sinish, energiya taqsimoti, nurning tarqalishi.

1.21. Kompyuter grafikasida rang.

Rang tushunchasi, rang ta'sirchanligi, energiya taqsimoti, yorug'likka bo'lgan ta'sirchanlik.

Rang tushunchasi odam (odamning ko'zi) yorug'likni qanday qabul qilishi bilan bog'liq.

Yorug'likni o'z navbatida ikki xil tushunish mumkin – har-xil energiyali zarrachalarning oqimi (u holda rangni zarrachalarning energiyasi aniqlaydi), yoki elektromagnit to'lqinlarning oqimi (bu holda rang to'lqin uzunligi λ orqali aniqlanadi). Ko'rinadigan yorug'lik bu 400-700 nm (nanometr) gacha bo'lgan to'lqin uzunligiga ega elektromagnit to'lqinlar.

Rang odamning ko'zida tug'iladi. Odamning ko'zi yorug'likni qanday qabul qilishini ko'ramiz.

Ko'zning "setchatkasi" fotoreseptorga ega – "kolbochki". Ular tor (ensiz) spektral egri chiziqlar bilan tavsiflanadi va rang ta'sirchanligiga ega. Ular ("kolbochki") uch xil bo'ladi – uzun, o'rta va qisqa to'lqinlar ta'sirchanligiga javob beruvchi. Ular ("kolbochki") tomonidan beriladigan qiymat spektral funksiya $I(\lambda)$ bilan ta'sirchanlik funksiyasini integrallash natijasini beradi.

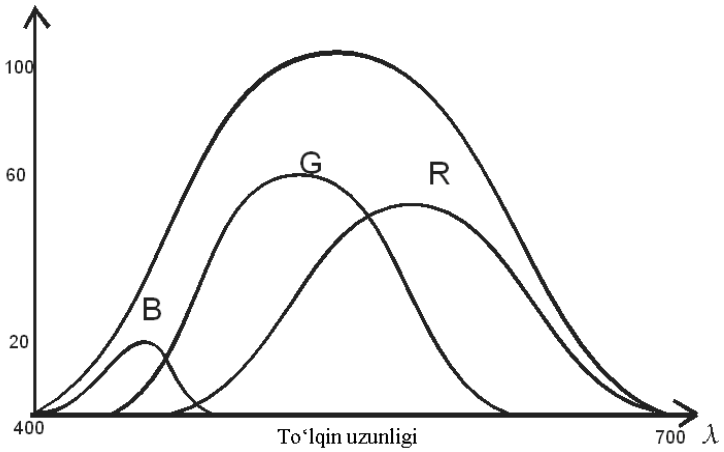
2.26-rasmda uch xil tipdagi "kolbochyok"lar uchun ta'sirchanlik funksiyalarining grafiklari keltrilgan.

Shunday qilib, odam ko'zi spektral funksiya uchun $I(\lambda)$, mos ravishda uchta sonni qabul qilandi (R, G, B) va ular quyidagi formula yordamida hisoblanadi:

$$R = \int I(\lambda) P_R(\lambda) d\lambda,$$
$$G = \int I(\lambda) P_G(\lambda) d\lambda,$$
$$B = \int I(\lambda) P_B(\lambda) d\lambda.$$

Bu yerda: $P_R(\lambda)$, $P_G(\lambda)$, $P_B(\lambda)$ – mos ravishda har-xil tipdagi "kolbochyok" larning vazniy ta'sirchanlik funksiyalari.

Odam ko‘zining umumiy ta’sirchanligi uchun javob beruvchi egri chiziqlar grafigini olishda uchta egri chiziqlar grafiklari o‘zaro yig‘iladi.



1.26-rasm. Ko‘zning nisbiy ta’sirchanligi.

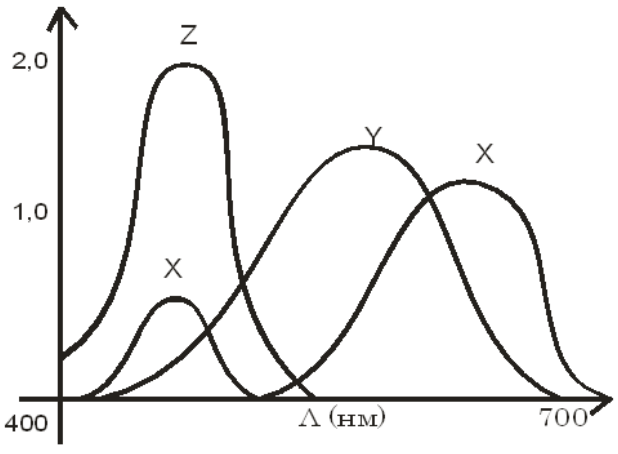
Aslida ayrim grafiklar manfiy qiymatlarni ham qabul qilishlari mumkin. 1931 yilda Yoritish (Yorug‘lik) bo‘yicha Xalqaro Komissiya (YoXK)si (CIE-Comission Internationale de L’Eclairage) gipotetik ideal kuzatuvchi standart egri chiziqlarni qabul qildi. Ular yordamida *XYZ* rang modeli quriladi, bunda *x,y,z* asosiy ranglar. *X,Y,Z* ning qiymatlari quyidagi munosabatlar orqali ifodalanadi:

$$X = \int I(\lambda)x(\lambda)d\lambda ,$$

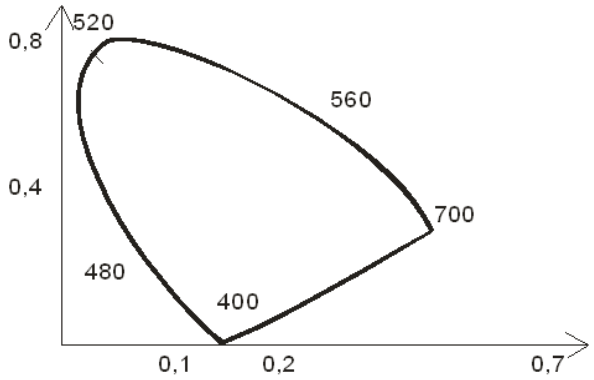
$$Y = \int I(\lambda)y(\lambda)d\lambda ,$$

$$Z = \int I(\lambda)z(\lambda)d\lambda .$$

Ushbu uchta sonlar orqali odam ko‘zi qabul qiladigan ixtiyoriy rangni bir qiymatli ifodalash mumkin.



1.27-rasm. Rang bo‘yicha tenglashtirish koeffisienti.



1.28-rasm. Yoritish bo‘yicha halqaro komissiyaning rang grafigi.

Aytish joizki *Y* rangi uchun javob beruvchi energiya taqsimoti egri chizig‘i odam ko‘zining yorug‘likka bo‘lgan ta’sirchanlik spektral egri chizig‘i bilan ustma-ust tushadi.

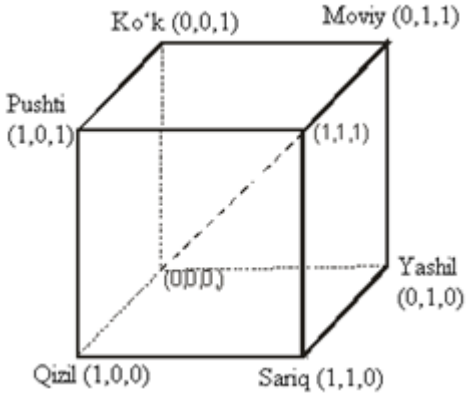
1.22. Rang modellari.

Rang modellari: RGB, CMY, HSV va boshqalar

RGB rang modeli.

RGB (red-qizil, green-yashil, blue-ko‘k) rang modeli eng oddiy deb xisoblanadi.

Bu rang modeli additiv, ya’ni biror bir kerakli rangni hosil qilish uchun uning asosiy ranglari yig‘iladi. Bu tizim orqali ifodalanuvchi ranglar birligi kubni tashkil qiladi (ya’ni uning ichida yotadi).



1.29-rasm. RGB rangli kub.

Kubning bosh diagonali, ya’ni barcha asosiy ranglar miqdori barobar, kul ranglarni beradi, ya’ni qoradan (0,0,0) oq (1,1,1) ranggacha.

CIE XYZ sistemasidan RGB sistemasiga o‘tish uchun quyidagi munosabatlardan foydalaniladi:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 3{,}240479 & -1{,}537156 & -0{,}498535 \\ -0{,}969256 & 1{,}875992 & 0{,}041556 \\ 0{,}055648 & -0{,}204043 & 1{,}057311 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}.$$

Agar biror bir rangni RGB tizimi orqali ifodalab bo‘lmasa, u holda uning biror bir asosiy ranggi yoki manfiy (<0) yoki birdan katta (>1).

Teskari almashtrishni topish uchun teskari matritsadan foydalaniladi.

RGB rang modeli bir qator video qurilmalarda ishlatiladi, xususan rangli televizion monitorlarda.

CMY rang modeli.

Rangli bosmaga chiqaruvchi qurilmalarda CMY (cyan-moviy; magenta-pushti; yellow-sariq) rang modeli tez-tez ishlatiladi. Moviy, pushti va sariq asosiy ranglar qizil, yashil va ko‘k ranglarni to‘ldiruvchi bo‘ladilar.

CMY rang modeli - subtraktiv, ya’ni biror kerakli bo‘lgan rangni hosil qilish uchun asosiy ranglar oq rangdan ajraladi.

Moviy rang qog‘ozga tushirilayotgan qizil rang to‘liq yutiladi, ya’ni moviy rang tushayotgan oq rangdan (qizil, yashil va ko‘k ranglarning yig‘indisi) qizil rangni ayirib tashlaydi. Pushti rang yashil rangni yutadi, sariq rang esa – ko‘k rangni. Moviy

va sariq ranglar bilan bo‘yalgan sirt qizil va ko‘k ranglarni yutib faqat yashil rangni qoldiradi.

Moviy, sariq va pushti ranglar qizil, yashil va ko‘k ranglarni yutib, natijada qora rangni qoldiradi.

Yuqorida keltirilgan munosabatlarni quyidagi formula orqali ifodalaymiz:

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix}.$$

RGB rang modelidan CMY modeliga o‘tish quyidagi munosabatlar orqali bajariladi:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} G \\ M \\ Y \end{pmatrix}.$$

Ayrim sabablarga ko‘ra qora rangni hosil qilish uchun uchta asosiy ranglardan foydalanish noqulay. Shu sababli CMY modelining asosiy ranglariga qora (black) qo‘shiladi va natijada CMYK rang modeli hosil qilinadi.

CMY rang modelidan CMYK modeliga o‘tish uchun quyidagi munosabatlardan foydalanamiz:

$$K = \min(C, M, Y),$$
$$C = C - K,$$
$$M = M - K,$$
$$Y = Y - K.$$

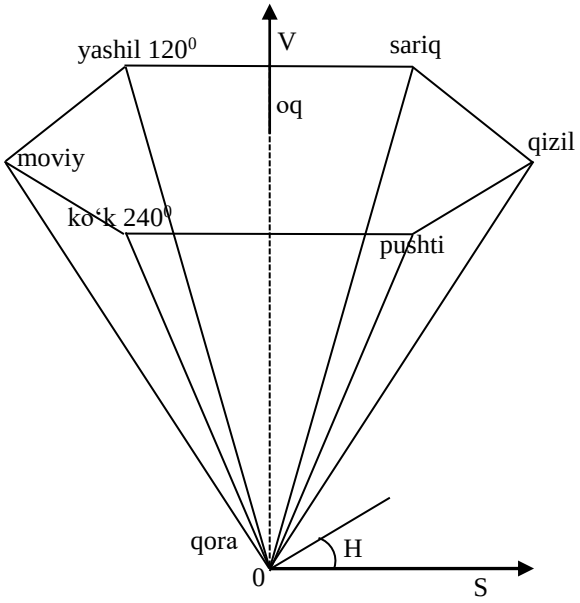
YIQ rang modeli.

Televideniyada YIQ rang modeli keng foydalaniladi. Bu model RGB rang modelining bir varianti hisoblanadi, efirga uzatish samaradorligini oshirish maqsadida ishlatiladi va oq-qora televideniya bilan ishlashni ta’minlaydi.

RGB rang modelidan YIQ modeliga o‘tish uchun quyidagi munosabatdan foydalaniladi:

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0{,}30 & 0{,}59 & 0{,}11 \\ 0{,}60 & -2{,}28 & -0{,}32 \\ 0{,}21 & -0{,}52 & 0{,}31 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Teskari almashtrishlar uchun teskari matritsadan foydalaniladi.



2.30-rasm. HSV olti qirrali konussimon rang modeli.

HSV rang modeli.

Yuqorida keltirilgan RGB, CMY(K) va YIQ rang modellari qurilmalar uchun mo‘ljallangan va odam tomonidan rang berish uchun noqulay.

HSV (Hue-ton, saturation-to‘yinganlik, value-yorug‘lik qiymati) rang modeli foydalanuvchiga mo‘ljallangan bo‘lib rangni ton, to‘yinganlik, yorug‘lik qiymati kabi tushunchalar orqali berish imkonini beradi.

HSV modelida silindrik koordinatalar sistemasi ishlatiladi, barcha ifodalanuvchi ranglar esa olti qirrali konusni tashkil qiladi (2.31-rasm).

Konusning asosi yorug‘ ranglarga mos keladi (V=1). OV o‘qi kulranglarga mos keladi (S=0), bu holda ya’ni S=0 bo‘lganda H ning qiymati aniqlanmagan bo‘ladi (HUE_UNDEFGIUNED).

Ton N burchak gradusi bilan o‘lchanadi, 0^0 ga qizil rang mos keladi, 120^0 ga esa yashil rang va x.k.

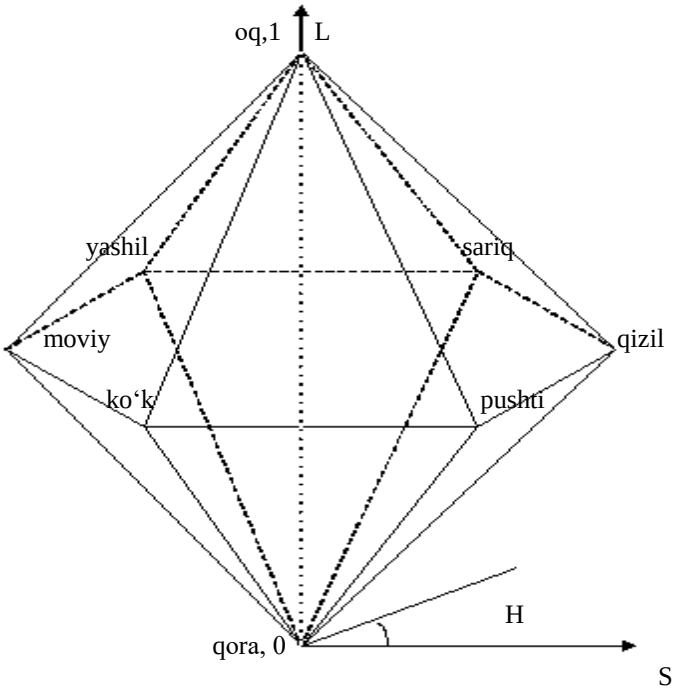
HSV rang modelini ifodalovchi asosiy qiymatlar mos ravishda quyidagicha o‘zgaradi:

$$0^0 \le H \le 360^0, \qquad 0 \le S \le 1, \qquad 0 \le V \le 1.$$

oq ranga S=0, V=1 mos keladi, qora ranga esa V=0.

HLS rang modeli.

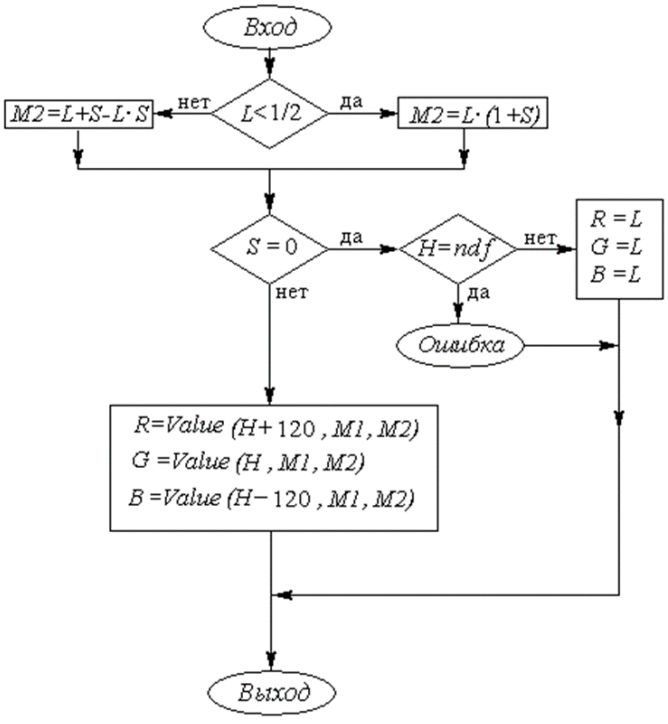
HLS (Hue–ton, Lightness-yorug‘lik, Saturation-to‘yinganlik) HSV modelining modifikasiyasi bo‘lib, u orqali ifodalanuvchi ranglar ikkita olti qirrali va asoslari birlashtirilgan konusni tashkil qiladi. Oq rang yuqoriga siljitilgan (2.32-rasm).



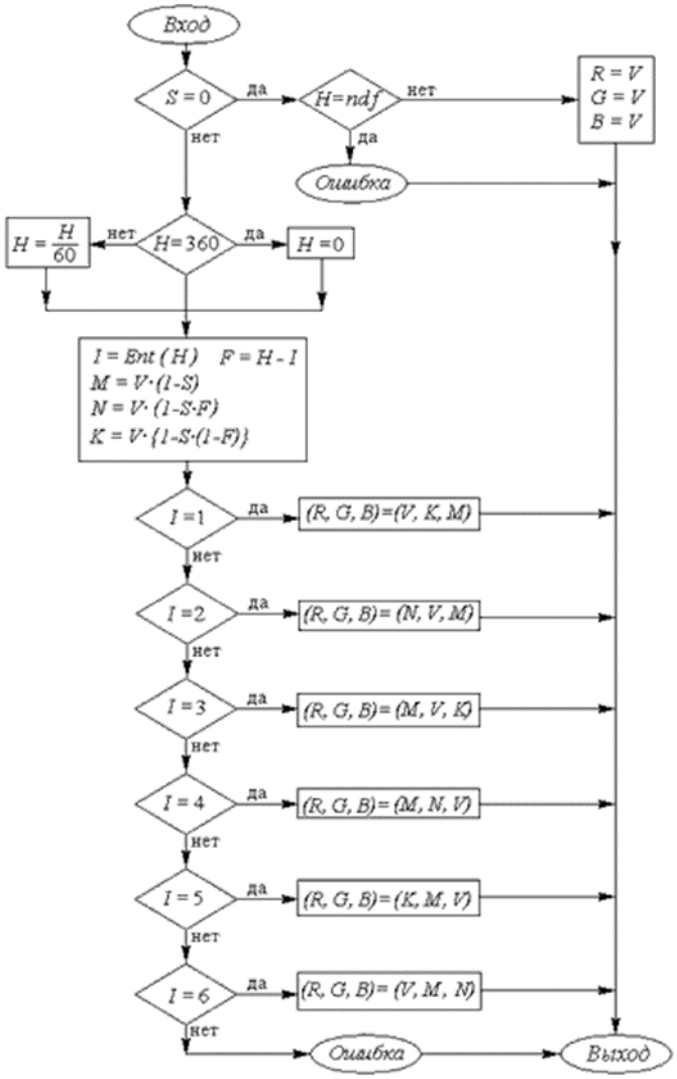
1.31-rasm. HLS olti qirrali konussimon rang modeli.

Алгоритмы перевода из одной модели в другую

Преобразование модели HLS в RGB



Преобразование модели HSV в RGB



Nazorat savollari:

1. Rang tushunchasiga ta’rif bering?
2. Odam ko‘zi spektral funksiya uchun qanday ranglarni qabul qiladi?
3. RGB rang modelining o‘ziga xos jihatlari.
4. CMYK rang modelini izohlang.
5. RGB rang modelidan CMYK rang modeliga o‘tish.
6. YIQ rang modelini izohlang.
7. RGB rang modelidan YIQ rang modeliga o‘tish.
8. HSV va HLS rang modellari.

Tayanch iboralar: Rang, RGB modeli, CMY modeli, YIQ modeli, HSV modeli, HLS modeli.

1.23. Grafik formatlar.

Fayllarning grafik shakllari

Grafik fayl formati – ma’lumotni faylda saqlash usuli (raster yoki vektor), shuningdek ma’lumotni saqlash shakli (ishlatilgan siqish algoritmi).

Raster formatlari Bitmap fayllari quyidagilarni saqlaydi:

- Tasvir hajmi - tasvirdagi gorizontal va vertikal joylashgan piksellar soni;
- bit chuqurligi - bitta piksel rangini saqlash uchun ishlatiladigan bitlar soni;
- tasvirni tavsiflovchi ma’lumotlar (tasvirdagi har bir pikselning rangi), shuningdek ba’zi bir qo‘shimcha ma’lumotlar.

Grafik fayl formati - bu tashqi ma’lumotlarga grafik ma’lumotlarni taqdim etish va tartibga solish usuli.

Piksel - bu monitordagi eng kichik element (nuqta) bitta rastr tasvir elementi (piksel) rastr tasvirning juda kichik elementlardan iborat mozaikadir.

Rastr tasvir chizmasi katakli qog‘ozga o‘xshaydi, uning ustiga har bir katak ma’lum bir rang bilan to‘ldiriladi. Har qanday grafik tasvir faylda saqlanadi. Grafik ma’lumotlarini faylga saqlab qo‘yish uslubi faylning grafik formatini aniqlaydi.

Rastr tasvir va vektor tasvir fayl formatlari mavjud.

Rastr tasvirlari to‘rtburchaklar jadval shaklidagi faylda saqlanadi, har bir kamerada mos keladigan pikselning ikkilik rang kodi saqlanadi. Ushbu fayl grafik tasvirning ma’lumotlarni va boshqa xususiyatlarini, shuningdek, siqishni algoritmini saqlaydi.

Vektorli tasvirlar faylga ob’yektlar va ularning xususiyatlarining qiymatlari - koordinatalar, o‘lchamlar, ranglar va shunga o‘xshashlar ro‘yxati sifatida saqlanadi.

Ikkala raster va vektor grafik fayl formatlari ham juda ko‘p qo‘laniladi. Ushbu turli xil formatlar orasida barcha mumkin bo‘lgan talablarga javob beradiga fayl formati yo‘q.

Tasvirni saqlash uchun bir yoki bir nechta formatni tanlash tasvir bilan ishlash maqsadlariga bog‘liq. Agar rangni qayta tiklash va tasvir aniqligini oshirish talab etilsa, raster formatlardan biri ustunlikka ega hisoblanadi. Chizmalar, sxemalar,

dizayn elementlari vektor formatida saqlanishi kerak. Fayl formati ushbu faylni egallagan xotira miqdoriga ta’sir qiladi.

Grafik dasturlar foydalanuvchilarga tasvirni saqlash formatini tanlash imkonini beradi. Agar faqatgina bitta dasturda grafik tasvir bilan ishlashni rejalashtirilgan bo‘lsa, grafik dastur taqdim etadigan formatni tanlash tavsiya etiladi. Ma'lumotlar boshqa dasturlar tomonidan qayta ishlanadigan bo‘lsa, universal formatlardan birini qo‘llash kerak.

Vektorli va rastrli tasvirlarni qo‘llab-quvvatlaydigan universal grafik formatlar mavjud.

PDF (Portable Document Format) - (portativ hujjat formati) Acrobat dasturiy ta'minot to‘plami bilan ishlash uchun mo‘ljallangan. Ushbu formatda ham vektor, ham rastr formatdagi tasvirlar, ko‘p sonli shriftlar bilan matn, gipermatnli matnlar va hatto printerni sozlamalari saqlanishi mumkin. Fayl hajmi juda kichik. Bu faqat fayllarni ko‘rish imkonini beradi, bu formatdagi rasmlarni tahrirlash mumkin emas

EPS (Encapsulated PostScript) - turli xil operatsion sistemalar uchun dasturlar tomonidan qo‘llab-quvvatlanadigan formatdir. Ushbu format tasvirlarni chop etish va yaratish uchun tavsiya etiladi. Rasmlar, fotosuratlar, ko‘rsatuvlar va boshqalarni yaratishda ishlatiladigan eng keng tarqalgan grafik formatlar.

CGM (Computer Graphics Metafile) - ochiq format va grafik ma'lumotlarni (2D vektor va rastrli grafik va matn) saqlash va almashish uchun xalqaro standartdir. ISO/IEC 8632 tomonidan qabul qilingan standart (inglizcha kompyuter grafikasi metafaylidan) ochiq format va grafik ma'lumotlarni (2D vektor va rastrli grafik va matn) saqlash va almashish uchun xalqaro standartdir. ISO/IEC 8632 tomonidan qabul qilingan standart.

DjVu (fransuzcha déjà vu – “allaqachon ko‘rgan”) - bu tasvirni yo‘qotishli siqish yordamida hujjatlarni (kitoblar, jurnallar, qo‘lyozmalar va shunga o‘xshashlar, birinchi navbatda skanerlangan) taqdim etish va saqlash texnologiyasi.Texnologiya dastlab AT&T laboratoriyasida 1996 yildan 2001 yilgacha Yan LeKun, Leon Botu va

Patrik Xeffner tomonidan ishlab chiqilgan. DjVu bir nechta ilmiy kitob kutubxonalari uchun asos bo‘ldi. Endi bu format juda mashhur va keng tarqalgan.

Faylni yuklab olish tugagunga qadar sahifani ko‘rish uchun format tarmoq uzatish uchun optimallashtirilgan. DjVu fayli matn (OCR) qatlamini o‘z ichiga olishi mumkin, bu fayl ichida to‘liq matnli qidirish imkonini beradi. Bundan tashqari, DjVu faylida DjVu kitoblarida qulay navigatsiyani amalga oshirish imkonini beruvchi o‘rnatilgan interaktiv tarkib va kirish nuqtalari - havolalar bo‘lishi mumkin.

Rasmlar, fotosuratlar, ko‘rsatuvlar va boshqalarni yaratishda ishlatiladigan eng keng tarqalgan grafik formatlar

Расм …

BMP (BitMap Picture) - formati Windows ning asosiy formati bo‘lib, uning nazorati ostida ishlaydigan barcha grafik tahrirlovchilar tomonidan qo‘llab-quvvatlanadi. BMP formati juda ko‘p dasturlar bilan ishlaydi, chunki uning qo‘llab-quvvatlanishi operatsion tizimga integratsiyalashgan. BMP formatidagi fayllar .bmp, .dib va .rle kengaytmalariga ega bo‘lishi mumkin. Bundan tashqari, ushbu formatdagi ma'lumotlar RES Resurs ikkilik fayllari va PE fayllarida mavjud. bmp formatida 2, 16, 256 va 16777216 raqamlariga mos keladigan 1, 4, 8 va 24 bitlik rang chuqurligi (bittadan pikselni tasvirlaydigan bit sonlari) yordamida tasvirlarni saqlashmumkin.

Tiff (Tagged Image File Format) - tasvirlarni saqlash uchun mo‘ljallangan. Bu ranglarning chuqurligi yuqori bo‘lgan tasvirlarni saqlash uchun ishlatiladigan yuqori sifatli format. TIFF fayllari ham siqilgan, ham paketlanmagan holda saqlanishi mumkin. Formatning katta afzalligi deyarli har qanday siqish algoritmini qo‘llab-quvvatlashdir. TIFF-dagi rasm har bir fayl saqlangandan keyin sifatini yo‘qotmaydi. Ammo, afsuski, aynan shu sababli TIFF fayllari og‘irligi JPG va GIF-dan bir necha baravar ko‘proq. TIFF raster grafikalarini vektor dasturlari va nashriyot tizimlariga import qilishda eng yaxshi tanlovdir.

PSD (Photoshop Document) - Adobe Photoshop dasturi uchun maxsus yaratilgan va uning barcha xususiyatlarini qo‘llab-quvvatlaydigan yo‘qotishsız siqishda ishlatadigan grafik ma'lumotlarni saqlashni raster formati. PSD kengaytmasi bilan birgalikda turli xil grafik qatlamlar, matnli ma'lumotlar, muayyan eslatmalar, qatlam maskalari, kalit so‘zlar va Photoshopdan boshqa ma’lumotlar taqdim etiladi. 48-bit rang kodlash, ranglarni ajratish va turli xil rangli modellar qo‘llab-quvvatlanadi. Asosiy kamchilik, samarali axborotni siqish algoritmining etishmasligi fayllarning katta hajmiga olib kelishi bilan izohlanadi.

PCX (PCExchange) - ZSoft Corporation (Marietta, Gruziya, AQSh) tomonidan ishlab chiqilgan grafik ma'lumotlarni taqdim etish standarti. MS-DOS uchun zsoft PC Paintbrush (birinchi mashhur grafik dasturlardan biri) grafik dasturi, so‘z protsessorlari va Microsoft Word va Ventura Publisher kabi ish stoli nashriyot tizimlari tomonidan ishlatilgan. Adobe Photoshop, Corel Draw, GIMP va boshqalar kabi o‘ziga xos grafik muharrirlar tomonidan qo‘llab-quvvatlanadigan bo‘lsa-da, BMP analoglari hozirgi vaqtda eng yaxshi siqishni qo‘llab-quvvatlaydigan formatlar bilan almashtiriladi: GIF, JPEG va PNG.

JPEG (Joint Photographic Experts Group) - (Qo‘shma Fotografik Ekspertlar Guruhi). Format rastr tasvirlarni saqlash uchun mo‘ljallangan (fayl nomi kengaytmasi .JPG). Faylni siqish darajasi va tasvir sifati o‘rtasidagi nisbatni moslashtirish imkonini beradi. Amaldagi siqishni usullari “keraksiz” ma’lumotlarni olib tashlashga asoslanadi, shuning uchun format faqat elektron nashrlarda foydalanish uchun tavsiya etiladi.

Gif (Graphics Interchange Format) - (Grafika O‘zaro almashinuv Formati). 1987-yilda (GIF87a) CompuServe tomonidan rastr tasvirlarni tarmoqlar orqali uzatish uchun apparat-mustaqil GIF formati ishlab chiqildi. 1989 yilda format o‘zgartirildi (GIF89a), shaffoflik va animatsiyani qo‘llab-quvvatlash qo‘shildi. GIF LZW-siqishni ishlatadi, bu juda ko‘p bo‘lmagan logotiplar, yozuvlar, diagrammalarda bo‘lgan fayllarni yaxshi siqilishini ta'minlaydi.

PNG (portable network graphics) - Deflate algoritmidan foydalangan holda, grafik ma'lumotni yo‘qotishsiz siqishni yordamida saqlash uchun rastr formati. PNG gifni almashtirish uchun bepul format sifatida yaratilgan. Internet uchun maxsuis ishlab chiqilgan. Muhim xususiyati unda tasvirning tiniqligini saqlash mumkin. Barcha brauzer foydalanuvchilari bunday formatdagi fayllarni ko‘rishi mumkin.

WMF (WindowsMetafile) - Windows operatsion tizimining (fayl kengaytmasi .WMF) vektor tasvirini saqlash formati. Ta’rif bo‘yicha, ushbu tizimning barcha ilovalari qo‘llab-quvvatlanadi. Biroq, poligrafiyada qabul qilingan standart rangli sxemalar bilan ishlaydigan asboblarning etishmasligi va boshqa kamchiliklar uni ishlatishni cheklaydi (WMF rangi buzadi, turli xil vektor muharririda ob’yektlarga berilishi mumkin bo‘lgan bir qator parametrlarni saqlab bo‘lmaydi).

Truevision TGA (TGA) - rastr grafik formatidir. Dastlab Truevision Inc tomonidan yaratilgan. 1984 yilda o‘z ishlab chiqarishining grafik adapterlari uchun, lekin keyinchalik turli platformalarda, ayniqsa videoni qayta ishlash va animatsiya sohasida mashhur bo‘ldi. Odatda, bu formatdagi fayllar eski DOS tizimlarida .tga yoki Macintosh kompyuterlarida .tpic kengaytmasiga ega. Format har bir piksel uchun 1-32 bit rang chuqurligini qo‘llab-quvvatlaydi. Alfa-kanallarni qo‘llab-quvvatlash, RLE siqish mavjud.

WebP - bu Google Inc tomonidan ishlab chiqilgan yo‘qotishli va yo‘qotishsiz tasvirni siqish formatidir. 2010 yilda VP8 video kodekidan harakatsiz tasvirni siqish algoritmi (asosiy kadrlar) asosida RIFF konteyneridan foydalanadi. Ushbu format bilan ishlash uchun ochiq kodli dasturiy ta'minot, xususan, libvpx kutubxonasi va webpconv konvertori mavjud.

Vektor formatlar

2D formatlari:

- Masshtabli vektor grafikasi (SVG va SVGZ)
- Encapsulated PostScript (EPS)
- Windows metafayllari: WMF, EMF
- CorelDraw fayllari: →CDR, CMX
- Adobe Illustrator (AI)
- XAR

3D formatlari

- COLLADA - bu 3D ilovalari o‘rtasida almashish uchun mo‘ljallangan format.
- SKP
- STL - stereolitografiya uchun format
- U3D - universal 3D
- VRML - Virtual haqiqatni modellashtirish tili
- X3D
- .3ds
- 3DXM
- .blend

...

1.24. Nurning yo‘nalishini kuzatish usullari (trace - kuzatish).

Nur yo‘nalishini kuzatishning asosiy modeli.

Real tasvirlarni ko‘rishda nurni yo‘nalishini kuzatish usullari, uning qaytish (aks etish) va sinish effektlarini hisobga olgan holda keng qo‘llaniladi.

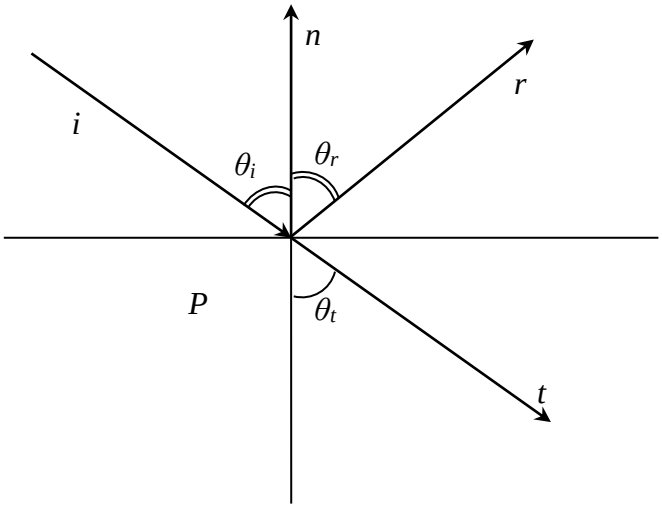
Bunda yorug‘lik manbasidan to‘g‘ri chiziqli trayektoriya bo‘yicha biror bir obyektga tushadi. Obyektga tushgan yorug‘lik nuri sinib obyektning ichiga o’tishi yoki qaytishi mumkin. Obyektdan qaytgan nur yana to‘g‘ri chiziqli tarqalib keyingi obyektga tegishi mumkin va hokazo. Oxirida nurlarning bir qismi nazoratchi ko‘ziga tushib unda tasvirni hosil qiladi. Agar nazoratchi ko‘zi oldida tasvir tekisligini joylashtirsak uni kesib o‘tgan nurlar unda tasvirini paydo qiladi. Yuqorida aytib o‘tilgan jarayon nurlarni to‘g‘ri yo‘nalishini kuzatish deyiladi. Bu holda yorug‘lik manbasidan tarqalgan ko‘pgina nurlar obyektlarga tushmasligi yoki nazoratchi ko‘ziga etib kelmasligi mumkin.

Shuning uchun kompyuter grafikasida faqat foydali nurlar, ya’ni nazoratchi ko‘ziga tushuvchi nurlarni hisobga olish tavsiya qilinadi. Va shu sababli nurlarni yo‘nalishini kuzatish teskari bajariladi, ya’ni yorug‘lik manbasidan emas nazoratchi ko‘zidan boshlab biror bir obyektning nur bilan kesilish nuqtasiga qadar.

Yuqorida aytib o‘tilgan jarayon nurning yo‘nalishini teskari kuzatish yoki shunchaki nurning yo‘nalishini kuzatish deyladi. Aynan shu usulni quyida ko‘ramiz:

Nurni yo‘nalishini kuzatish usulida obyektning ixtiyoriy nuqtasining yorug‘ligini va berilgan yo‘nalishda undan qaytuvchi yorug‘lik energiyasi qismini aniqlash masalasi ko‘riladi. Bu energiya ikki qismdan tashkil topadi – bevosita (dastlabki) yorug‘lik, ya’ni yorug‘lik manbasidan bevosita olinuvchi energiya va ikkinchi yorug‘lik, ya’ni boshqa obyektlardan keluvchi energiya. Bu bo‘lish nisbiy.

Ma’lumki, bevosita yorug‘lik tasvirga katta hissa qo‘shadi.



2.24-rasm. Nurning qaytishi va sinishi.

Simmetrik qaytish.

Berilgan *i* yo‘nalish bo‘yicha *P* nuqtaga tushgan nur *r* yo‘nalish bo‘yicha qaytadi va quyidagi qonun bilan aniqlanadi: nurning yo‘nalishini aniqlovchi *i* vektor, qaytgan nurning yo‘nalishini aniqlovchi *r* vektori va *P* nuqtaning normal vektori *n* bitta tekislikda yotadi. Normal vektorga nisbatan nurning tushish burchagini Q_i va qaytish burchagi Q_r deb belgilaymiz.

Faraz qilamizki *i,n,r* vektorlar birlik (normal) vektorlar bo‘lsin. Yuqorida keltirilgan shartlarga ko‘ra *r* vektori *i* va *n* vektorlari chiziqli kombinasiyasiga teng:

$$r = \alpha\, i + \beta\, n, \tag{19}$$

tushish va qaytish burchagi o‘zaro teng, ya’ni $Q_i = Q_r$.

Bunga ko‘ra:

$$(-i, n) = Cos\ \theta_i = Cos\ \theta_r = (r, n), \tag{20}$$

bu yerdan quyidagi ifodani olamiz:

$$r = i - 2(i, n) \cdot n.$$

Isbot:

$(r,n)=(-i,n) \Rightarrow (r,n)= - (i,n) \Rightarrow (r,n)=(i,n)-2(i,n)-|(n,n)=1| \Rightarrow$

$\Rightarrow (r,n)=(i,n)-2(i,n)(n,n) \Rightarrow (r,n)=(i-2(i,n)*n,n) \Rightarrow$

$\Rightarrow r=i-2(i,n)* n.$

(1) ifoda bilan berilgan *r* vektor – birlik vektor.

Isbot:

$$||r||^2 = (r,r) = (i - 2(i,n)n, i - 2(n,i)n) = (i,i) - 2(n,i)(i,n) - 2(n,i)(n,i) + +4(n,i)\,(n,i)\,(n,n) =$$
$$|=(i,i)=1, (n,n)=1| = 1 - 4(n,i)^2 + 4(n,i)^2 = 1.$$

Diffuzion qaytish.

Diffuzion qaytish Lambert qonuni orqali ifodalanadi va unga asosan tushayotgan yorug‘lik hamma yo‘nalishlarga bir xil intensivlik bilan tarqaladi. Tushayotgan nurning qaytish yo‘nalishini aniqlab bo‘lmaydi, barcha yo‘nalishlar bir xil va nuqtaning yorug‘ligi (i,n) ga proporsional.

Ideal sinish.

i vektori yo‘nalishi bo‘yicha R nuqtaga tushuvchi nur ikkinchi sohaga t yo‘nalish bo‘yicha sinadi. Sinish Sneliusning qonuniga bo‘ysunadi va burchaklari uchun quyidagi ifoda bilan beriladi.

$$\eta_i \cdot Cos\ \theta_i = \eta_t \cdot Cos\ \theta_t , \tag{21}$$

bu yerda: η_i - (i) nur tushuvchi muhitning sinish koeffisienti; η_i – (t) nur sinuvchi muhitning (sohaning) sinish koeffisienti; θ_i – tushuvchi nur (i) va R nuqtaning normal vektori (n) orasidagi burchak; θ_t - qaytgan nur (t) va R nuqtaning normal vektori (n) yo‘nalish orasidagi burchak.

Sneliusning qonuniga asosan i,n va t vektorlari bitta tekislikda yotadi, ya’ni:

$$t = \alpha_i + \beta_n , \tag{22}$$

bu yerda α va β qiymatlarini topish uchun (1) ifodani quyidagi ko‘rinishda yozamiz:

$$\eta \cdot Sin\ \theta_i = Sin\ \theta_t$$

bu yerda: $\eta = \frac{\eta_i}{\eta_t}$.

Kvadratga ko‘taramiz:

$$\eta^2 Sin^2 \theta_i = \eta^2 Sin^2 \theta_t$$

yoki

$$\eta^2 (1 - Cos^2 \theta_i) = 1 - Cos^2 \theta_t$$
$$Cos\ \theta_i = (-i, n)\ ,\quad Cos\ \theta_t = (-t, n)$$

larni hisobga olgan holda

$$\eta^2(1-(i,n)^2) = I-(-t,n)^2$$

(2) tenglamani hisobga olgan holda

$$\eta^2(1-(i,n)^2) = 1-(\alpha i+\beta n,n)^2$$

bundan

$$\alpha^2(i,n)^2 + 2\alpha\beta(i,n) + \beta^2 = 1+\eta^2((i,n)^2-1), \quad (3)$$

t vektorning normalligini shartiga ko‘ra:

$$\| t \|^2 = (t,t) = \alpha^2 + 2\alpha\beta(i,n) + \beta^2 = 1, \quad (4)$$

(4) ifodani (3) ifodadan ayiramiz.

$$\alpha^2((i,n)^2-1) = \eta^2((i,n)^2-1)$$

bundan $\alpha = \pm\eta$.

Fizik nuqtai nazarga ko‘ra

$$\alpha = \eta.$$

β - ni qiymatini topish uchun (4) ifodadan foydalanamiz;

$\alpha=\eta$ hisobga olgan holda

$$\beta^2 + 2\beta\eta(i,n) + \eta^2 - 1 = 0,$$

diskriminant $D = 4(1+\eta^2((i,n)^2-1))$.

Tenglamaning yechimi

$$\beta = -\eta(i,n) \pm \sqrt{(1+\eta^2((i,n)^2-1)}.$$

Buni hisobga olgan holda *t* vektori quyidagicha ifodalanadi:

$$t = \eta i + (\eta c_i - \sqrt{(1+\eta^2(c_i^2-1))*n}),$$

bu yerda $c_i = \cos\theta_i = (-i,n) = -(i,n)$.

Agar $1+\eta^2(c_i^2-1) < 0$ bo‘lsa hamma yorug‘lik energiyasi chegarada aks etadi va sinish bo‘lmaydi.

Diffuzion sinish.

Diffuzion sinish, diffuzion qaytish kabi, ya’ni singan yorug‘lik barcha yo‘nalishlar t, $(t,n) < 0$ bo‘yicha bir xil intensivliklar bilan tarqaladi.

Energiya taqsimoti.

Yorug‘likning qaytishi va sinishida energiya taqsimotini ko‘ramiz. Fizika kursidan ma’lumki qaytgan energiya hissasi (ulushi) Frenel koeffisientlari orqali beriladi.

$$F_r(\lambda,\theta)=\frac{1}{2}\left\{\left(\frac{Cos\ \theta_i-\eta Cos\ \theta_t}{Cos\ \theta_i+\eta Cos\ \theta_t}\right)^2+\left(\frac{\eta Cos\ \theta_i-Cos\ \theta_t}{\eta Cos\ \theta_i+Cos\ \theta_t}\right)^2\right\},$$

bu yerda λ - to‘lqinning uzunligi.

Ushbu formula yarim o‘tkazgichlar uchun o‘rinli va uni boshqa ko‘rinishda ham yozish mumkin:

$$F_r(\lambda,\theta)=\frac{1}{2}(\frac{c-g}{c+g})(1+(\frac{c(c+g)-1}{c(c-g)-1})^2),$$

bu yerda $c=Cos\ \theta_i;\ g=\sqrt{(\eta^2+c^2-1)}=\eta Cos\ \theta_t$.

O‘tkazuvchilar uchun odatda quydagi formulalardan foydalaniladi:

$$F_2(\lambda,\theta)=\frac{1}{2}((\frac{(\eta_t^2+k_t^2)\cos^2\theta_i-2\eta_t\cos\theta_i+1}{(\eta_t^2+k_t^2)\cos^2\theta_i+2\eta_t\cos\theta_i+1})^2+(\frac{(\eta_t^2+k_t^2)-2\eta_t\cos\theta_i-\cos^2\theta_i}{(\eta_t^2+k_t^2)+2\eta_t\cos\theta_i+\cos^2\theta_i})^2),$$

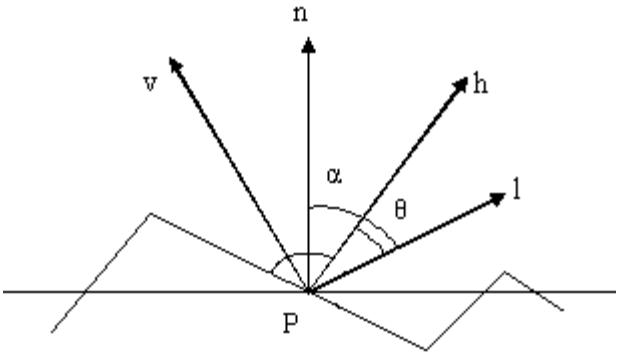
bu yerda k_t - singish koeffisienti.

Yuqorida keltirilgan barcha hollar ideallashtirilgan. Aslida ideal qaytuvchi va ideal silliq tekisliklar yo‘q.

Nur yo‘nalishini kuzatishning asosiy modeli

Odatda amaliyotda obyektning tashqi taqsimot qonuni berilgan tasodifiy yo‘naltirilgan tekis ideal mikroyoqlardan tashkil topadi deb hisoblanadi.

Faraz qilamizki: n - sirtning R nuqtadagi normal vektori, ya’ni o‘rta tekislik normal vektori; h - mikroyoqning normal vektori; α - ular orasidagi burchak: $\alpha=arccos(n,h)$.



1.25-rasm. Nurning tarqalishi.

Sirt α - ning tasodifiy qiymati taqsimot zichligini beruvchi funksiya $D(\alpha)$ orqali ifodalanadi. $D(\alpha)$ uchun bir nechta model mavjud:

Gauss taqsimoti

$$D(\alpha) = ce^{-(\frac{\alpha}{m})^2}.$$

Bekmen taqsimoti

$$D(\alpha) = \frac{1}{4\pi m^2 \cos^4 \alpha} e^{-(tg\frac{\alpha}{m})^2}.$$

Bu yerda m - sirtning tekis emasligini harakterlovchi daraja, m ning qiymati kichik bo‘lgan sari sirt shuncha tekisrok bo‘ladi.

R nuqtaga l yo‘nalishi bo‘ycha tushayotgan yorug‘lik nurining aks etishini ko‘ramiz.

V yo‘nalishi bo‘yicha aks etuvchi yorug‘lik energiyasi qismatini topish uchun quyidagi ifoda kerak bo‘ladi:

$$h = \frac{l + V}{\left\| l + V \right\|}.$$

Mikroyoqdan aks etuvchi energiya ulushi Frenel $F_r(\lambda,\theta)$ koeffisientlari bilan aniqlanadi, bu yerda

$$\theta = \arccos(h, v) = \arccos(h, l),$$

h,v,l - vektorlari birlik vektorlar.

Qo‘shni yoqlarning soya qiluvchi ta’siri odatda quyidagi funksiya orqali ifodalanadi:

$$G = mm(l, \frac{2(n,h)(n,v)}{(v,h)}, \frac{2(n,h)(n,l)}{(v,h)}).$$

Mikroyoqlar to‘plamidan tashkil topuvchi sirt uchun bizni qiziqtiruvchi energiya ulushi juft yo‘nalgan akslanish funksiyasi orqali ifodalanadi, ya’ni (Bidirectional Reflection Distribution Function):

$$BRDF\ (l,v,\lambda) = \frac{F_2(\lambda,\theta)D(\alpha)G(n,v,l)}{(n,l)(n,v)}.$$

Bu formula hisoblashlar uchun murakkab, shuning uchun oddiyroq formuladan foydalaniladi:

$$BRDF\ (l,v,\lambda) = (n,h)^k F_r(\lambda,\theta). \qquad (23)$$

Umuman qaraganda *BRDF* funksiyasi simmetrik shartga bo‘ysunadi:

$$BRDF\ (l,v) = BRDF\ (v,l).$$

Frenel koeffisientlari tasvirning realligiga sezilarli ta’sir ko‘rsatishiga qaramay amaliyotda ular juda kam ishlatiladi va (23) formula o‘rnida soddaroq formuladan foydalaniladi:

$$BRDF\ (l,v,\lambda) = (n,h)^k$$

Yana bir sabablardan biri formulada ishlatiladigan to‘lqin uzunligi λ haqida aniq ma’lumot yo‘qligi.

Berilgan yo‘nalish bo‘yicha *R* nuqtadan qaytuvchi energiya quyidagicha topiladi:

$$I(\lambda) = K_a I_a(\lambda)C(\lambda) + K_d C(\lambda)\sum_i I_{ei}(\lambda)(n,l_i) + K_r \sum_i I_{ei}(\lambda)\frac{F_r(\lambda,\theta)D(\lambda)G}{(n,l_i)(n,v)} +$$

$$+ K_r I_r(\lambda)F_r(\lambda,\theta_r)e^{-\beta_r d_r} + K_t I_t(\lambda,\theta_t)(1 - F_r(\lambda,\theta_t)e^{-\beta_t d_t})$$

$$I(\lambda) = K_a I_a(\lambda)C(\lambda) + K_d C(\lambda)\sum_i I_{ei}(\lambda)(n,l_i) + K_r \sum_i I_{ei}(\lambda)\frac{F_r(\lambda,\theta)D(\lambda)G}{(n,l_i)(n,v)} + K_3 I_r(\lambda)F_r(\lambda,\theta_r)e^{-\beta d_r} + K_t I_t(\lambda,\theta_t)(1 - F_r(\lambda,\theta_t)e^{-\beta_t d_t})$$

bu yerda: $I_a(\lambda)$, $I_{ei}(\lambda)$, $I_r(\lambda)$, $I_t(\lambda)$ - mos ravishda *i* – chi yorug‘lik manbasi, fon yorug‘ligi, aks etuvchi nur bo‘yicha yo‘nalgan yorug‘lik, singan nur bo‘yicha yo‘nalgan yorug‘lik intensivligi;

$S(\lambda)$ - *P* nuqtaning rangi;

K_a, K_d, K_r - fon, diffuzion, aks yorug‘lik koeffisientlari;

K_t – signal turining ulushi;

l_i – P nuqtaning i-chi yorug‘lik manbasiga bo‘lgan yo‘nalishning birlik vektori;

θ_r, θ_t - aks va sinish burchaklari;

d_r, d_t - aks va sinish nurlarining o‘tgan masofalari;

β_r ,β_t - aks va sinish nurlarining bo‘shashish koeffisientlari.

Keltirilgan model fizik nuqtai nazardan qaralganda korrekt bo‘lgani bilan, hisoblashlar uchun juda murakkab. Yorug‘likning oddiy modeli:

$$I(\lambda)=K_aI_a(\lambda)C(\lambda) + K_dC(\lambda)\sum I_{li}(\lambda)(n,l_i) + K_s\sum I_{li}(\lambda)(n,h_i)^P$$

$$I(\lambda) = K_a I_a(\lambda) C(\lambda) + K_d C(\lambda) \sum I_{li}(\lambda)(n, l_i) + K_S \sum I_{li}(\lambda)(n, h_i)^P \ .$$

Metallar uchun oxirgi hadda rang $C(\lambda)$ qo‘shiladi.

II BOB. OPENGL GRAFIK KUTUBXONASI

2.1. Open Graphics Library (OpenGL).

OpenGL grafik kutubxonasi va uning tavsifi. Arxitekturasi va o‘ziga xos xususiyatlari. Dasturiy ilovalarda OpenGL grafik kutubxonasini qo‘llay olish.

OpenGL - ikki va uch o‘lchovli grafika sohasida ilovalar yaratish uchun ancha keng tarqalgan amaliy dasturiy interfeys (API – Application Programming Interface) lardan biri hisoblanadi.

OpenGL (Open Graphics Library – ochiq grafik kutubxona) standarti turli xil platformalarda amalga oshirish uchun kerakli, apparatga bog‘liq bo‘lmagan interfeys sifatida dasturiy ta’minot ishlab chiqarish sohasidagi yetuk firma tomonidan 1992 yilda ishlab chiqilgan va tasdiqlangan. Standartning asosi bo‘lib Silicon Graphics Inc firmasi tomonidan ishlab chiqilgan IRIS GL kutubxonasi hisoblanadi [2-4, 15, 16].

Kutubxona 120 ga yaqin turli buyruqlardan iborat bo‘lib, dasturchi operatsiyalar va obyektlarni tayinlashda hamda interaktiv grafik ilovalarni yozishda foydalanadi. Bugungi kunda OpenGL grafik tizimini ko‘pgina apparatli va dasturiy platformalarni quvvatlaydi. Ushbu tizim Windows va Linux operatsion tizimlarida ishlovchilar uchun ochiq.

OpenGL kutubxonasining o‘ziga xos xususiyatlari, ya’ni ushbu grafik standartning rivojlanishi va keng tarqalishini ta’minlovchi jihatlar, quyidagilar hisoblanadi:

- *Barqarorlik.* Standartga o‘zgartirish va qo‘shimchalar kiritish, oldingi dasturiy ta’minotlarda ishlab chiqilganlarning mosligini saqlab qolish ko‘rinishida amalga oshiriladi.
- *Ishonchlilik va uzatuvchanlik.* OpenGL dan foydalanayotgan ilova, axborotlar tasvirlanishini tashkil qilish va foydalanilayotgan operatsion tizim turiga bog‘liq bo‘lmagan holda bir hil vizual natijani kafolatlaydi. Bundan tashqari, ushbu

ilovalar ShK larda qanday bajarilsa, xuddi shunday ishchi stansiya va superkompyuterlarda ham bajariladi.

- *Qo‘llashning osonligi.* OpenGL standarti mukammal o‘ylangan tuzilmaga va tushunarli interfeysga ega bo‘lib, boshqa grafik kutubxonalardan foydalanishga nisbatan dastur kodi kamroq bo‘lgan samarali ilovalarni yaratish imkonini beradi. Turli qurilmalar bilan moslikni ta’minlash uchun zaruriy funksiyalar kutubxona darajasida tashkillashtirilgan bo‘lib, ilovalar ishlab chiqishni ancha osonlashtiradi.

Asosiy imkoniyatlar

OpenGL imkoniyatlarini biz uning kutubxonasidagi funksiyalar orqali ta’riflaymiz. Barcha funksiyalarni beshta toifaga ajratish mumkin:

- *Primitivlarni tavsiflash funksiyasi* grafik nimtizimni aks ettirishga qodir bo‘lgan, ierarxiyaning quyi darajasidagi obyektlarni belgilaydi. OpenGLda primitivlar sifatida nuqta, chiziq, ko‘pburchak va boshqalar nazarga tutiladi.
- *Ranglar manbasini tavsiflash funksiyasi* uch o‘lchovli sahnada joylashgan, ranglar manbasi parametri va holatini tavsiflash uchun xizmat qiladi.
- *Atributlarni tayinlash funksiyasi.* Atributlarni tayinlash yordamida dasturchi aks etadigan obyekt ekranda qanday ko‘rinishga kelishini belgilaydi. Boshqacha so‘z bilan aytganda, agar ekranda *nima* hosil bo‘lishi primitivlar yordamida belgilansa, unda atributlar ekranga chiqarish *usulini* belgilaydi. Atributlar sifatida OpenGL rang, material xususiyati, tekstura, yorug‘lik parametrlarini berish imkonini beradi.
- *Vizuallashtirish funksiyasi* virtual fazoda kuzatuvchi (kamera obyektivi parametri) holatini belgilash imkonini beradi. Ushbu parametrlarni bilish, tizim nafaqat tasvirni to‘g‘ri qurishi, balki kuzatuv maydonidan tashqari bo‘lgan obyektlarni ajratib qo‘yishi ham mumkin.
- *Geometrik o‘zgartirish funksiyalari* to‘plami dasturchiga obyektlarni turli xil o‘zgartirish – burish, ko‘chirish, masshtablashtirishni bajarish imkonini beradi.

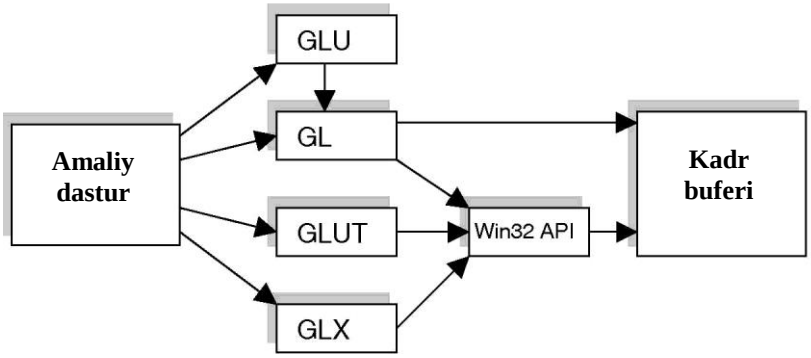
Shunday ekan OpenGL qo‘shimcha operatsiyalarni ham bajara olishi mumkin, masalan, chiziqlar va sirtlarni qurishda splaynlardan foydalanish, tasvirning ko‘rinmas qismlarini olib tashlash, piksellar darajasidagi tasvirlar bilan ishlash va b.

OpenGL interfeysi

OpenGL kutubxonalar to‘plamidan tarkib topgan. Barcha bazaviy funksiyalar asosiy kutubxonada saqlanadi, belgilab qo‘yish uchun keyinchalik biz *GL* qisqartmasidan foydalanamiz. Bundan tashqari OpenGL o‘zida bir qancha qo‘shimcha kutubxonalarni qamrab oladi [3, 20].

Ulardan birinchisi – *GL* (*GLU – GL Utility*) *utilit kutubxonasi*. Ushbu kutubxonaning barcha funksiyalari GL ning bazaviy fuknsiyalari orqali belgilanadi, masalan, keng tarqalgan geometrik primitivlar to‘plami (kub, shar, silindr, disk), splaynlar qurish funksiyalari, matritsalar ustida qo‘shimcha operatsiyalarni amalga oshirish (amallar bajarish) va x.k.

OpenGL da oynalar bilan ishlash yoki foydalanuvchi ma’lumot kiritishi uchun zarur hech qanday maxsus buyruqlar mavjud emas. Shu sabab foydalanuvchi bilan ishlash uchun va oynalar tizimi yordamida ma’lumotlarni ko‘rsatish uchun maxsus uzatuvchi kutubxonalar yaratilgan, ya’ni tez-tez ishlatiladigan funksiyalarning foydalanuvchilar bilan o‘zaro aloqasini ta’minlash uchun va oynali nimtizimlar yordamida axborotlarni aks ettirish uchun. *GLUT (GL Utility Toolkit)* kutubxonasi bugungi kunda keng tarqalgan. Umuman olganda GLUT kutubxonasi OpenGL tarkibiga kirmaydi, ammo de facto barcha distributlari tarkibiga kiradi va turli platformalar uchun ishlay oladi. GLUT OpenGL ilovalarini yaratish uchun minimal darajadagi kerakli funksiyalar to‘plamidan tashkil topgan. Funksional jihatdan GLX kutubxonasi nisbatan kamroq ommaviylashgan. Ushbu qo‘llanmada asosan GLUT kutubxonasi ko‘riladi.



2.1-rasm. OpenGL kutubxonalarining tashkil etilishi.

Bundan tashqari, odatda oynalar tizimi bilan ishlovchi funksiyalar uning amaliy dastur interfeysiga kiradi. Demak, OpenGLni ishlashini ta’minlovchi funksiyalar Win32 API va X Window lar tarkibida mavjud. Yuqorida keltirilgan rasmda, Windows tizimida ishlaydigan kutubxonalar tizimlarini tashkil qilish sxemasi keltirilgan. OpenGLning boshqa versiyalarida ham tashkil qilish shunga o‘xshash amalga oshiriladi.

OpenGL tuzilishi

OpenGL funksiyalari mijoz-server texnologiyasi asosida yaratilgan. Ilovalar mijoz vazifasini bajaradi – ular buyruqlarni ishlab chiqadilar, server OpenGL esa ularni qayta ishlaydi va bajaradi. Server o‘zi mijoz joylashgan kompyutyerda joylashgan bo‘lishi ham mumkin (masalan, dinamik yuklanuvchi kutubxona ko‘rinishida – DLL), yoki boshqa kompyutyerda (ushbu holda mashinalar o‘rtasida ma’lumot almashinish uchun maxsus protokol qo‘llaniladi).

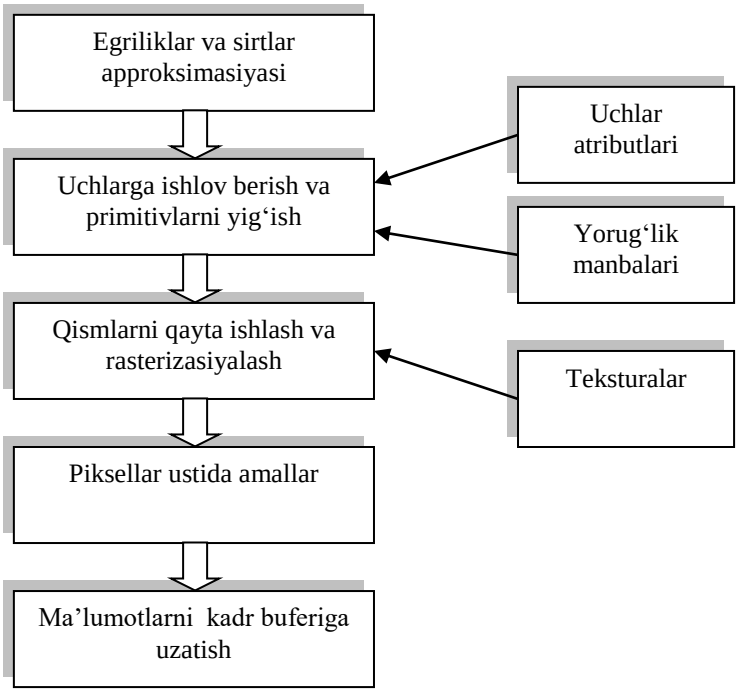
GL bir qancha tanlangan rejimlar bo‘yicha grafik *primitivlarni* kadr buferida chizadi va qayta ishlaydi. Har bir primitiv – bir nuqta, kesma, ko‘pburchak va boshqalar bo‘lishi mumkin. Har bir rejim boshqasiga bog‘liq bo‘lmagan holda o‘zgartirilishi mumkin. Primitivlarni aniqlash, rejimlarni tanlash va boshqa amallar kutubxona funksiyalarini chaqirish ko‘rinishidagi *buyruqlar* orqali tavsiflanadi.

Primitivlar bir yoki bir nechta *uchlar* (vertex) to‘plami yordamida aniqlanadi. Uch nuqta orqali ifodalanadi, kesma oxiri yoki ko‘pburchak uchi. Har bir uch atributlar deb ataluvchi ma’lum bir ma’lumotlarga ega (koordinatalar, rang, normal,

tekstura koordinatalari va x.k.). Ko‘p hollarda uchlar bir biriga bog‘liqsiz holda qayta ishlanadi.

OpenGL grafik tizimi tuzilishi jihatidan grafik obyektlarni ketma-ket qayta ishlashning bir necha bosqichlaridan iborat konveyer hisoblanadi.

OpenGL buyruqlari har doim ularning kelishi tartibida amalga oshiriladi, biroq ularning hosil qilinish effektidan oldin to‘htalishlar bo‘lib qolishi mumkin. Ko‘p hollarda OpenGL aniq interfeysni taqdim etadi, ya’ni obyektning aniqlanishi uning kadr buferida vizualizasiya bo‘lishiga olib keladi.



2.2-rasm. OpenGL konveyerining funksionallashuvi.

Dasturchilar nuqtai nazarida, OpenGL – grafik qurilmalardan foydalanishni boshqarib turadigan buyruqlar to‘plamidir. Agarda qurilma faqatgina adreslangan kadr buferidan tarkib topgan bo‘lsa, u holda OpenGL markaziy prosessor resurslaridan foydalanish yordamida to‘liq amalga oshirilishi kerak. Odatda grafik qurilma turli darajadagi tezlikni ta’minlaydi: chiziq va ko‘pburchaklarni chiqarishdan geometrik malumotlar ustida turli amallarni qo‘llab murakkab grafik usullarni uskunaviy amalga oshirilishigacha.

OpenGL qurilma va foydalanuvchi darajalari o‘rtasidagi qatlam hisoblanib, qurilmalar imkoniyatlaridan foydalangan holda turli platformalarda yagona interfeysni taqdim etish imkonini beradi.

Bundan tashqari, OpenGLni chekli avtomat sifatida qarash mumkin, uning holati joriy normal, rang, tekstura koordinatalari va boshqa atributlar va alomatlar qiymatlari va maxsus o‘zgaruvchilarning ko‘pgina qiymatlari bilan belgilanadi. Ushbu axborotlarning barchasi ekranga chiqariladigan figuralarni qurish uchun uchlar koordinatasi grafik tizimiga kirishda ishlatiladi. Holatlarni almashtirish funksiyalarni chaqirish uchun ishlatiladigan buyruqlar orqali amalga oshiriladi.

Buyruqlar sintaksisi

GL buyruqlarining belgilanishi gl.h faylida joylashgan, uni qo‘shish uchun quyidagini yozish zarur bo‘ladi:

```
#include <gl/gl.h>
```

GLU kutubxonasi bilan ishlash uchun glu.h faylini qo‘shish zarur. Ushbu kutubxona versiyalari qoida sifatida dasturlash tizimlari distributivlariga avtomatik o‘rnatiladi, masalan, Microsoft Visual C++, DevC++ yoki Borland C++ .

Standart kutubxonalardan farqli ravishda, GLUT paketini alohida o‘rnatish va qo‘shish zarur. OpenGL bilan ishlash uchun dasturlash muhitini sozlash to‘g‘risidagi batafsil ma’lumot ilova C da berilgan.

GL kutubxonasining barcha buyruqlari (prosedura va funksiyalar) gl old qo‘shimchasi bilan boshlanadi, barcha o‘zgarmaslar - GL_ old qo‘shimchasi bilan boshlanadi. GLU va GLUT kutubxonalarining tegishli buyruqlari va o‘zgarmaslari glu (GLU_) va glut (GLUT_) old qo‘shimchalariga ega bo‘ladi.

Bundan tashqari, buyruqlar nomiga parametrlar soni va turi to‘g‘risidagi ma’lumotlarni o‘zida saqlaydigan suffikslar ham kiradi. OpenGL da buyruqlarning to‘liq nomi quyidagi ko‘rinishga ega:

type **glCommand_name[1 2 3 4][b s i f d ub us ui][v]** (type1 *arg1*,…,typeN *argN*)

Nomlar bir qancha qismlardan tashkil topadi:

gl bu funksiya ko‘rsatilgan kutubxona nomi: OpenGL ning bazaviy funksiyalari uchun, GL, GLU, GLUT, GLAUX kutubxona funksiyalari, bular mos ravishda gl, glu, glut, aux.

Command_name buyruqlar nomi (proseduralar yoki funksiyalar)

[1 2 3 4] buyruqlar argumentlari soni

[b s i f d ub us ui] argument turi: b – GLbyte (C\C++ da char singari), i – GLint (butun), f – GLfloat (kasrli), s – GLshort (qisqa butun). d – GLdouble (ikkili aniqlikdagi kasrli), ub – GLubyte (belgisiz bayt), us – GLushort (belgisiz qisqa butun), ui – GLuint (belgisiz butun).

[v] ushbu belgining mavjudligi funksiya parametrlari sifatida belgilar massiviga yo‘nalish ishlatilishini ko‘rsatadi.

Kvadrat qavs ichidagi belgilar ba’zi nomlarda ishlatilmaydi. Masalan, glVertex2i() buyrug‘i GL kutubxonasida ko‘rsatilgan va parametrlari sifatida ikkita butun sonni ishlatadi, glColor3fv() buyrug‘i esa uchta haqiqiy sondan iborat massivga ko‘rsatkich parametr sifatida ishlatiladi.

Ilovaga misol

OpenGL dan foydalanib dastur tuzishda, dastlab tasvirni ko‘rsatuvchi oyna aniqlab olinadi. Shundan so‘ng OpenGL konteksi (mijoz) yaratiladi va shu oyna bilan bog‘lanadi. Keyigi qadamlarda dasturchi erkin holda OpenGL API buyruqlari va operatsiyalaridan foydalanishi mumkin.

Quyida GLUT kutubxonasidan foydalanib yozilgan kichik bir dastur matni keltirilgan.

Ushbu dastur oyna markazida qizil rangli kvadrat chizadi. Shu kichik dastur orqali ham OpenGL yordamida dastur tuzishning mohiyatini tushinib olish mumkin.

```
#include <stdlib.h>
/* GLUT kutubxonasini bog‘lash*/
/* oynaning dastlabki eni va balandligi (o‘lchamlari)*/
GLint Width = 512, Height = 512;
/* kubning hajmi */
const int CubeSize = 200;
/* ushbu funksiya ekranga chiqarilayotgan barcha ma’lumotlarni boshqaradi */
void Display(void)
{
   int left, right, top, bottom;
   left  = (Width - CubeSize) / 2;
   right = left + CubeSize;
   bottom = (Height - CubeSize) / 2;
   top = bottom + CubeSize;
   glClearColor(0, 0, 0, 1);
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3ub(255,0,0);
      glBegin(GL_QUADS);
         glVertex2f(left,bottom);
         glVertex2f(left,top);
         glVertex2f(right,top);
         glVertex2f(right,bottom);
     glEnd();
  glFinish();
}

/* ushbu funksiya oyna o‘lchami o‘zgargan holatini chaqirish uchun chaqiriladi */
void Reshape(GLint w, GLint h)
{
   Width = w;                    Height = h;
```

```
   /* ko‘rsatish sohasi o‘lchamini o‘rnatamiz  */
   glViewport(0, 0, w, h);     // x, y, balandlik, kenglik

   /* ortografik proeksiya */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0, w, 0, h, -1.0, 1.0);  // chap, o‘ng, quyi, yuqori, yaqin, uzoq
}

/* klaviaturadan kelgan ma’lumotlarni qayta ishlovchi funksiya */
void Keyboard( unsigned char key, int x, int y )
{
#define ESCAPE '\033'
   if( key == ESCAPE )
      exit(0);          }

/* ilovaning asosiy sikli */
main(int argc, char *argv[])
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGB);
   glutInitWindowSize(Width, Height);
   glutInitWindowPosition(10,10);
   glutCreateWindow("Qizil maydonga misol");
        glutDisplayFunc(Display);
        glutReshapeFunc(Reshape);
        glutKeyboardFunc(Keyboard);
   glutMainLoop();
}
```

Ushbu dastur hajmi kichik bo‘lishiga qaramay OpenGL va GLUT kutubxonalarini qo‘llaydigan, istalgan tizimda ishlaydigan va kompilyasiya qilinadigan to‘liq tugallangan dastur.

GLUT kutubxonasi foydalanuvchi bilan teskari aloqa (*callback function*) deb nomlanuvchi funksiya yordamida o‘zaro aloqani quvvatlaydi. Agarda foydalanuvchi sichqonchani siljitsa, klaviatura tugmasini bossa yoki oyna o‘lchamini o‘zgartirsa, hodisa yuz beradi va foydalanuvchining tegishli (hodisalarga ishlov berish (teskari aloqaga ega funksiya)) funksiyasi chaqiriladi.

Yuqoridagi misolda keltirilgan main funksiyasini batafsil ko‘rib chiqamiz. U uchta qismdan tarkib topgan – OpenGL chizishi kerak bo‘lgan oynani inisiallash (faollashtirish), teskari aloqali funksiyalarni va hodisalarga ishlov berishning asosiy davrini sozlash.

Oynani inisiallash (faollashtirish) kadrning tegishli buferlari, boshlang‘ich holati va oyna o‘lchami, shuningdek oyna sarlavhasini sozlashdan tarkib topadi.

glutInit (&argc, argv) funksiyasi GLUT kutubxonasining o‘zini dastlabki inisiallashuvga tayyorlaydi.

glutInitDisplayMode (GLUT_RGB) buyrug‘i kadr buferini inisiallashtiradi va to‘liq rangli (palitrasiz) RGB rejimiga to‘g‘rilaydi.

glutInitWindowSize (Width, Height) oynaning boshlang‘ich o‘lchamini tayinlash uchun ishlatiladi.

glutCreateWindow ("Qizil maydonga misol") oyna sarlavhasini kiritadi va oynani ekranda vizuallashtiradi.

Shunda quyidagi buyruqlar

glutDisplayFunc (Display);

glutReshapeFunc (Reshape);

glutKeyboardFunc (Keyboard);

Display(), Reshape() va Keyboard() funksiyalari oynani qayta chizish, oyna o‘lchamlarini o‘zgartirish, klaviatura klavishini bosganda chaqiriladigan funksiyalar sifatida qayd qiladi.

Barcha hodisalarni nazorat qilish va kerakli funksiyalarni chaqirish glutMainLoop () funksiyasining cheksiz davri ichida bajariladi.

Ko‘rishimiz mumkinki, GLUT kutubxonasi OpenGL tarkibiga kirmaydi, ya’ni OpenGL va oynali nimtizimning orasidagi keltirilgan eng kichik interfeysdir. OpenGL ilovasi aniq bir platformalar uchun maxsus API (Win32, X Window va b.) yordamida yozilishi mumkin, ular esa o‘z navbatida yanada kengroq imkoniyatlarni taqdim etadi. GLUT kutubxonasi bilan ishlash *A* ilovada batafsil berilgan.

OpenGL buyruqlarini chaqirish hodisalarni qayta ishlovchida sodir bo‘ladi. Ular keyingi bo‘limlarda kengroq qarab chiqiladi. Hozir esa, ekranda chizishga javob beradigan kodga ega bo‘lgan Display funksiyasiga e’tibor qaratamiz.

Display funksiyasiga tegishli quyidagi buyruqlar ketma-ketligi

```
glClearColor(0, 0, 0, 1);
glClear(GL_COLOR_BUFFER_BIT);

glColor3ub(255,0,0);
glBegin(GL_QUADS);
  glVertex2f(left,bottom);
  glVertex2f(left,top);
  glVertex2f(right,top);
  glVertex2f(right,bottom);
glEnd();
```

oynani tozalaydi va rang hamda to‘rtta burchak uchlari koordinatalarini berib ekranga kvadrat chiqaradi.

D.1 ilovasida sichqoncha tugmasini bosganda ekranda har xil rangdagi tasodifiy to‘g‘ri to‘rtburchakni chizadigan yana bir murakkab bo‘lmagan dastur keltirilgan.

Nazorat savollari:

1. OpenGL kutubxonasini yaratishdan ko‘zlangan maqsad nima?
2. OpenGL kutubxonasi asoschisi?

3. Kutubxonada mavjud buyruqlar soni va ular qanday vazifalarni bajaradi?
4. OpenGL kutubxonasining o‘ziga xos jihatlarini keltiring?
5. OpenGL kutubxonasidan qanday operatsion tizimlarda foydalanish mumkin?
6. Turli qurilmalar bilan ishlash qanday tashkillashtirilgan?

Tayanch iboralar: OpenGL, kutubxona, barqarorlik, ishonchlilik va uzatuvchanlik, qo‘llashning osonligi, moslashish.

Nazorat savollari:

1. Standart grafik kutubxonalar yaratish zaruriyati nimadan iborat?
2. OpenGL kutubxonasi funksiyalarini tavsiflang?
3. Konveyerni tashkil qilish va OpenGL kutubxonasi tuzilishini qisqacha ifodalang?
4. Kutubxona buyruqlari (funksiyalari) toifalarini keltiring?
5. OpenGL kutubxonasiga qanday kutubxonalar qo’shimcha o’rnatiladi?
6. Faqatgina parametrlar turi bilan farq qiluvchi OpenGL buyruqlarining turlicha variantlari nima uchun kerak?
7. Nima sababdan OpenGL ni tashkil etish ko‘pincha chekli avtomat bilan qiyoslanadi?

Tayanch iboralar: Primitivlar, ranglar manbasi, atributlarni tayinlash, vizuallashtirish, geometrik o‘zgartirish, OpenGL ilovalari, GL, GLU, GLUT, GLX, konveyer.

2.2. OpenGL yordamida geometrik obektlarni chizish

Tasvirlarni yangilash jarayoni. Uchlar va primitivlar. Uchlarning fazodagi holati. Uchlar rangi. Uchlar massivlari. Ob’ektlarni o‘zgartirish. Matritsalar bilan ishlash. Modelli tasvir o‘zgartishlar. Chiqarish sohasi.

Tasvirlarni yangilash jarayoni

Qoida sifatida, OpenGLni ishlatadigan dasturning vazifasi uch o‘lchamli tasvirni qayta ishlash va kadr buferida interaktiv tasvirlash hisoblanadi. Tasvir kuzatuvchining joriy holatini aniqlaydigan uch o‘lchamli obyektlar to‘plami, yorug‘lik manbai va virtual kameradan tashkil topadi.

Odatda OpenGL ilovasi uzluksiz siklda oynada tasvirni yangilash funksiyasini chaqiradi. Ushbu funksiyada OpenGL ning asosiy buyruqlari chaqirish joriy qilingan. Agar GLUT kutubxonasi ishlatilsa, unda bu glutDisplayFunc() chaqiruvi bilan qayd qilingan teskari aloqali funksiya bo‘ladi. GLUT bu funksiyani oyna tarkibini boshqatdan chizish kerakligi to‘g‘risida operatsion tizim ilovaga ma’lumot berganda bu funksiyani chaqiradi. Hosil qilinadigan tasvir ham statik, ham animatsiya ko‘rinishida bo‘lishi mumkin, ya’ni vaqt bo‘yicha o‘zgaradigan parametrlarga bog‘liq bo‘lganda. Bu holatda yangilash funksiyasini mustaqil ravishda chaqirish maqsadga muvofiq. Misol uchun, glutPostRedisplay () buyrug‘i yordamida. Yanada batafsil ma’lumot *A* ilovada keltirilgan.

Tasvirni yangilashning tipik funksiyasi qanday vazifalarni bajarishini ko‘rib chiqamiz. Qoida sifatida, u uch bosqichdan tarkib topadi:

1. OpenGL buferlarini tozalash;
2. Kuzatuvchini holatini o‘rnatish;
3. Geometrik obyektlarni chizish va o‘zgartirish.

Buferni tozalash quyidagi buyruqlar yordamida amalga oshiriladi:

void glClearColor (clampf *r*, clampf *g*, clampf *b*, clampf *a*) // *(clamp-fiksator)*

void glClear (bitfield *buf*) *//(bitfield - bitovoe pole)*

glClearColor buyrug‘i rangni o‘rnatadi, ya’ni kadr buferi to‘ldiriladi. Buyruqning dastlabki uchta parametri ranglarning R, G va B komponentlarini tayinlaydi va ular [0,1] kesma oralig‘ida bo‘lishi kerak. To‘rtinchi parametr alfa kompetentani beradi. Qoidaga ko‘ra, u 1 ga teng. Shart berilmaganda rang – qora (0,0,0,1).

glClear buyrug‘i buferni tozalaydi, *buf* parametri tegishli bufer o‘zgarmaslari kombinasiyasini belgilaydi. Odatdagi dastur bufer rangi va chiqurligini tozalash uchun

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

buyrug‘ini chaqiradi.

Kuzatuvchi holatini o‘rnatish va uch o‘lchovli obyektlarni o‘zgartirish (burish, kuchirish va b.) almashtirish matritsalarini tayinlash yordamida nazorat qilinadi. Obyektlarni o‘zgartirish va virtual kamera holatini sozlash keyingi bo‘limlarda keltirilgan.

Diqqatimizni sahnada joylashgan obyektlar tavsifini OpenGL ga uzatishga qaratamiz. OpenGL ning har bir obyekti primitivlar to‘plamidan tashkil topgan.

Uchlar va primitivlar

Uchlar (nuqta) OpenGL ning grafik primitivlari hisoblanadi va nuqtani, kesmaning oxiri, ko‘pburchakning burchagi va boshqalarni tavsiflaydi. Qolgan barcha primitivlar uchlarni tayinlash orqali shakllantiriladi. Masalan, kesma ikkita uchlar orqali ifodalanadi.

Har bir uch bilan birga uning *atributlari* ham mavjud. Asosiy atribut sifatida uchning fazodagi holati, uchning rangi va vektor normallarini olish mumkin.

Uchlarning fazodagi holati

Uchlarning holati ikki, uch yoki to‘rt o‘lchovli (bir jinsli koordinatalar) fazoda uning koordinatalarini tayinlash orqali belgilanadi. Bu glVertex* buyrug‘ini bir qancha variantlari yordamida amalga oshiriladi:

void **glVertex [2 3 4][s i f d]** (type *coords*)

void **glVertex[2 3 4][s i f d]v** (type **coords*)

Har bir buyruq uchlarning to‘rtta koordinatasini tayinlaydi: x, y, z, w. glVertex2* buyrug‘i x va y qiymatlarini qabul qiladi. Bunday holatda z koordinatasi 0 ga teng, w koordinatasi – 1 ga teng.

Vertex3* buyrug‘i x, y, z koordinatalarini qabul qiladi va w koordinatasiga 1 qiymatini kiritadi.

Vertex4* barcha to‘rtta koordinatalarni tayinlash imkonini beradi.

Uchlar rangi, normallar va teksturlar koordinatalarini birlashtirish uchun mavjud ma’lumotlarning joriy qiymatlari ishlatiladi. Ushbu qiymatlar ixtiyoriy vaqtda tegishli buyruqni chaqirish yordamida o‘zgartirilishi mumkin.

Uchlar rangi

Uchlarning joriy rangini tayinlash uchun quyidagi buyruqlar ishlatiladi:

void **glColor[3 4][b s i f]** (GLtype *components*)

void **glColor[3 4][b s i f]v** (GLtype *components*)

Dastlabki uchta parametr rangning R, G, B komponentlarini tayinlaydi, oxirgi parametr notiniqlik koeffisienti (alfa-qism deb ataluvchi)ni belgilaydi. Agarda buyruqning nomlanishida ‘f’ (float) tipi ko‘rsatilgan bo‘lsa, unda barcha parametrlar qiymatlari [0,1] kesma oralig‘ida joylashishi kerak, bunday holda alfa-qism qiymati 1.0 ga tenglashtirilishi o‘rnatiladi. ‘ub’ (unsigned byte) tipi qiymat [0,255] kesmada joylashishi kerakligi ko‘zda tutadi.

Uchlarga turli xil ranglarni tayinlash mumkin, agarda mos rejim yoqilgan bo‘lsa. Unda primitiv sirti bo‘ylab ranglarning chiziqli interpolyasiyasi amalga oshiriladi.

Interpolyasiya jarayonini boshqarish uchun quyidagi buyruqdan foydalaniladi

void **glShadeModel** (GLenum *mode*)

chaqiruv **GL_SMOOTH** parametr bilan berilsa, interpolyasiya qo‘shiladi (tanlangan parametr asosida), **GL_FLAT** parametr berilsa esa ajratadi, o‘chiradi.

Normal

Normallarni ifodalash uchun quyidagi buyruqlardan foydalaniladi

void **glNormal3[b s i f d]** (type *coords*)

void **glNormal3[b s i f d]v** (type *coords*)

Yorug‘lik hisobini to‘g‘ri olish uchun normal vektori yagona uzunlikka ega bo‘lishi zarur. glEnable(GL_NORMALIZE) buyrug‘i maxsus buyruqlarni qo‘shishi mumkin, bunda tayinlanayotgan normal avtomatik normallashadi.

Avtomatik normallash rejimi modelli kengayish-torayish almashtirilishi ishlatilganda yoqilgan bo‘lishi kerak, chunki bu holda normal uzunligi model-matritsasiga ko‘paytirilganda o‘zgaradi. Ammo ushbu rejimni joriy qilish OpenGL ning vizuallashtirish mexanizmi ishini sekinlashtiradi, xuddi shunday vektorlarni normallash hisoblashning sezilarli qiyinchiliklariga olib keladi (kvadrat ildiz olish va b.). Shuning uchun birdaniga yagona normalni berish ma’qul.

Eslatib o‘tish kerakki,

void **glEnable** (GLenum *mode*)

void **glDisable** (GLenum *mode*)

buyruqlari OpenGL konveyerining u yoki bu ish rejimini o‘chirilishi yoki yoqilishini ta’minlaydi. Ushbu buyruqlar ko‘p hollarda qo‘llaniladi va ularning parametrlari har bir aniq holatlar uchun qaraladi.

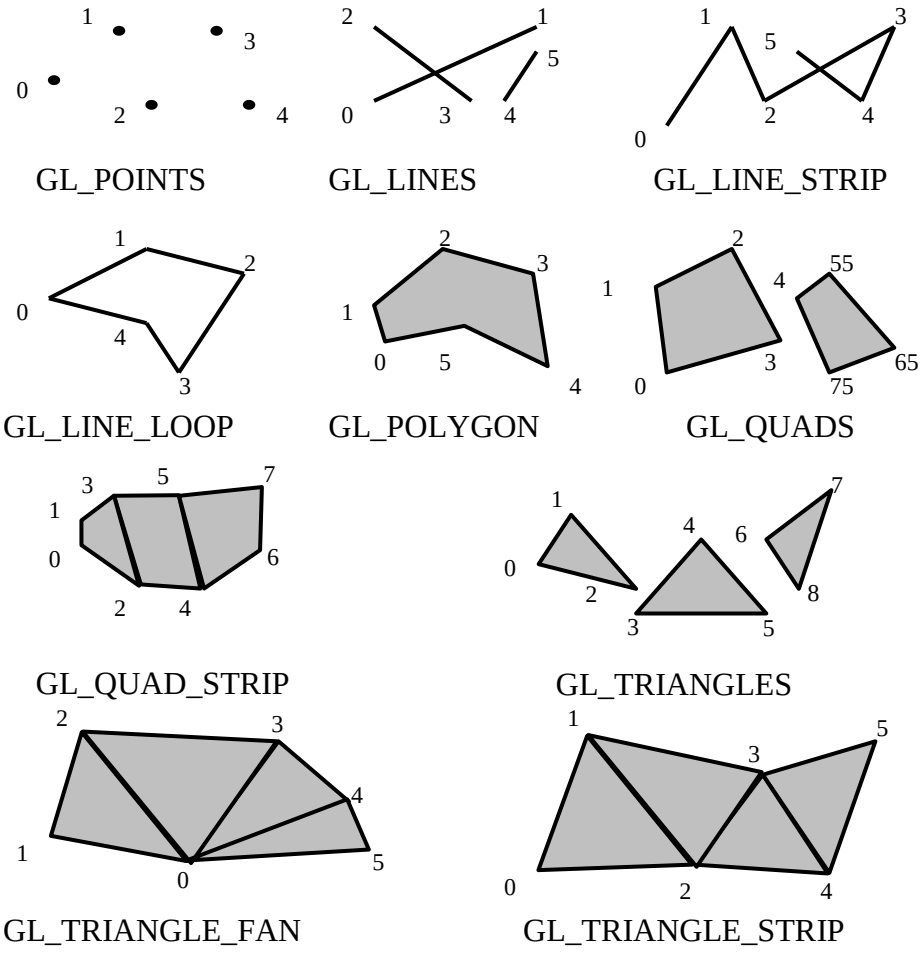
Operatorli qavslar glBegin / glEnd

Biz yuqorida bitta uchning atributlarini tayinlashni ko‘rib chiqdik. Ammo, grafik primitiv atributlarini tayinlash uchun, uchning bitta koordinatasi etarli emas. Ushbu uchlarni zaruriy xususiyatni belgilovchi bir butunlikka birlashtirish kerak. Buning uchun OpenGL da operatorli qavs deb ataluvchi maxsus buyruqlarni chaqirishda xizmat qiluvchilardan foydalaniladi. Primitivlarni belgilash yoki primitivlar ketma-ketligi

void **glBegin** (GLenum *mode*);

void **glEnd** (void);

buyruqlarini chaqirish oralig‘ida olib boriladi.



2.3-rasm. OpenGL primitivlari.

Mode parametri primitivning tipini belgilaydi, qaysiki ichkarida tayinlanadi va quyidagi qiymatlarni qabul qilishi mumkin:

GL_POINTS har bir uchlar bir qancha nuqtalar koordinatalarini belgilaydi.

GL_LINES har bir alohida uchlar juftligi kesmani belgilaydi; agarda toq sonli uchlar ko‘rsatilsa, u holda oxirgi uch e’tiborga olinmaydi.

GL_LINE_STRIP har bir keyingi uch oldingisi bilan birgalikda kesmani belgilaydi

GL_LINE_LOOP oldingi primitivdan farqli jihati shuki, oxirgi kesma so‘ngi va birinchi uchni belgilaydi, berk to‘g‘ri chiziq timsolida.

GL_TRIANGLES har bir alohida uchta uch uchburchakni belgilaydi; agarda uchta sonli uchlar karrali berilmasa, unda oxirgi uchlar e’tiborga olinmaydi.

GL_TRIANGLE_STRIP har bir keyingi uch oldingi ikkitasi bilan birgalikda uchburchakni belgilaydi.

GL_TRIANGLE_FAN uchburchaklar birinchi uch va har bir keyingi juft uchlar bilan beriladi (juftliklar kesishmaydi).

GL_QUADS har bir alohida to‘rtta uch to‘rtburchakni belgilaydi; agarda to‘rtta sonli uchlar karrali berilmasa, unda oxirgi uchlar e’tiborga olinmaydi.

GL_QUAD_STRIP *n* nomerli to‘rtburchak 2n-1, 2n, 2n+2, 2n+1 nomerli uchlar bilan belgilanadi.

GL_POLYGON qavariq ko‘pburchakning uchlarini ketma-ket berish.

Masalan, uchlari turli rangda bo‘lgan uchburchakni chizish uchun, quyidagilarni yozish etarli:

```
GLfloat BlueCol[3] = {0,0,1};
glBegin(GL_TRIANGLES);
    glColor3f(1.0, 0.0, 0.0);        /* qizil */
    glVertex3f(10.0, 10.0, 0.0);
    glColor3ub(0,255,0);             /* yashil */
    glVertex2i(100, 100);
    glColor3fv(BlueCol);             /* ko‘k */
    glVertex3f(100.0, 100.0, 0.0);
glEnd();
```

Qoida sifatida, primitivlarning har xil turlari turlicha platformalarda turlicha vizuallashish tezligiga ega. Unumdorlikni oshirishda, serverga uzatish uchun kam sonli axborotlarni talab etuvchi, GL_TRIANGLE_STRIP, GL_QUAD_STRIP, GL_TRIAGLE_FAN singari primitivlardan foydalanish afzalroqdir.

Mazkur ko‘pburchaklarni tayinlashdan tashqari, ularni ekranda akslantirish usullarini ham berish mumkin.

Ammo dastlab old va orqa yoq tushunchalarini belgilab olish lozim.

Yoq deganda ko‘pburchakning tomonlaridan biri tushuniladi, va odatda soat strelkasiga teskari aylanuvchi uchlar joylashgan tomon old hisoblanadi. Old yoqning uchlari aylanish yo‘nalishini quyidagi buyruqni chaqirish orqali o‘zgartirish mumkin

void **glFrontFace** (GLenum *mode*)

mode parametrining qiymati GL_CW (clockwise) ga teng, qiymatni odatdagi holatiga qaytarish uchun GL_CCW (counter-clockwise) ni ko‘rsatish kifoya.

Ko‘pburchakni tasvirlash usulini o‘zgartirish uchun quyidagi buyruq ishlatiladi

void **glPolygonMode** (GLenum *face*, Glenum *mode*)

mode parametri ko‘pburchak qanday akslanishishini belgilaydi, *face* parametri ko‘pburchakning tipini o‘rnatishda qo‘llaniladi va quyidagi qiymatlar ham qabul qilinishi mumkin:

GL_FRONT	ko‘rinadigan tomon (old yoq) uchun
GL_BACK	ko‘rinmas tomon (orqa yoq) uchun
GL_FRONT_AND_BACK	barcha tomon (yoq)lar uchun

mode parametri quyidagilarga teng bo‘lishi mumkin:

GL_POINT	ko‘pburchakning faqat uchlari tasvirlanadi.
GL_LINE	ko‘pburchaklar qismlar to‘plamida ifodalanadi.
GL_FILL	yorug‘likni hisobga olib ko‘pburchaklar joriy rang bilan buyab chiqiladi, va bu odatdagi rejim sifatida qaraladi.

Shu bilan birga, ekranda qanday chegara tasvirlanishini ko‘rsatish ham mumkin. Buning uchun dastlab **glEnable** (GL_CULL_FACE) buyrug‘ini chaqirish rejimi o‘rnatilishi kerak, shundan so‘ng quyidagi buyruqlar yordamida tasvirlanuvchi tomon tipi tanlanadi

void **glCullFace** (GLenum *mode*)

Funksiya **GL_FRONT** parametri bilan chaqirilsa, tasvirdan barcha ko‘rinadigan tomonlar olib tashlanadi, **GL_BACK** – parametr bilan esa teskarisi (tanlov asosida o‘rnatiladi).

Ko‘rilgan standart primitivlardan tashqari GLU va GLUT kutubxonalarida yanada murakkabroq figuralar mavjud, jumladan sfera, silindr, disk (GLU da) va

sfera, kub, konus, tor, tetraedr, dodekaedr, ikosaedr, oktaedr va chaynik (GLUT da). Teksturalarni avtomatik joylashtirish faqat GLU kutubxonasi figuralari uchun ko‘rilgan (OpenGL da tekstura yaratish 3.6 paragrafda ko‘riladi).

Misalan, sfera yoki silindr chizish uchun, avvalo quyidagi

GLUquadricObj* **gluNewQuadric** (void);

buyruq yordamida maxsus tipdagi GLUquadricObj obyekti yaratiladi.

Undan so‘ng kerakli buyruq chaqiriladi:

void **gluSphere** (GLUquadricObj * *qobj*, GLdouble *radius*,
GLint *slices*, GLint *stacks*)

void **gluCylinder** (GLUquadricObj * *qobj***,**
GLdouble *baseRadius***,**
GLdouble *topRadius***,**
GLdouble *height***,** GLint *slices***,**
GLint *stacks*)

bu yerda *slices* parametri z o‘qi atrofida siniq egri chiziqlar sonini beradi, *stacks* – esa z o‘qi atrofi bo‘ylab bo‘linishlar sonini beradi.

Primitivlarni qurishning ushbu va boshqa buyruqlari haqida yanada aniqroq ma’lumotlar *B* ilovada keltirilgan.

Displeyli ro‘yxatlar

Agar biz bitta buyruqlar guruhiga bir necha marotaba murojaat qilsak, unda ularni displey ro‘yxati (display list) ga birlashtirish va zarur bo‘lganda uni chaqirish mumkin. Yangi displey ro‘yxatini hosil qilish uchun unga kirishi kerak bo‘lgan barcha buyruqlarni quyidagi operator qavslari orasiga joylashtirish kerak:

void **glNewList** (GLuint *list*, GLenum *mode*)

void **glEndList** ( )

Ro‘yxatlarni ajratish uchun butun musbat sonlardan foydalaniladi, berilgan ro‘yxat *list* parametri bilan beriladi, *mode* parametri esa ro‘yxatga kirgan buyruqni qayta ishlash rejimini tavsiflaydi:

GL_COMPILE	buyruq bajarilmasdan oldin ro‘yxatga yoziladi
GL_COMPILE_AND_EXECUTE	buyruqlar oldin bajarilib, keyin ro‘yxatga yoziladi

Ro‘yxat tuzilgandan keyin, *list* parametrda kerakli ro‘yxatning identifikatorini ko‘rsatib, uni quyidagi buyruq bilan chaqirib olish mumkin:

void **glCallList** (GLuint *list*)

Birdaniga bir necha ro‘yxatni chaqirib olish uchun, quydagi buyruqdan foydalanish mumkin:

void **glCallLists** (GLsizei *n*, GLenum *type*, const GLvoid **lists*)

Bu buyruq identifikatorlari *lists* massivida bo’lgan *n* ro‘yxatlarni chaqirib oladi. *lists* massivi elementlarining turi *type* parametrda ko‘rsatiladi. Bular quyidagi turlar bo‘lishi mumkin: **GL_BYTE**, **GL_UNSIGNED_BYTE**, **GL_ShORT**, **GL_INT**, **GL_UNSIGNED_INT.**

Ro‘yxatlarni o‘chirish uchun

void **glDeleteLists** (GLint *list*, GLsizei *range*)

buyrug‘i ishlatiladi. Bu buyruq ID identifikatorlarga ega bo‘lgan *list* ≤ ID ≤ *list*+*range*-1 diapazonida ro‘yxatlarni o‘chiradi.

Masalan:

```
glNewList(1, GL_COMPILE);
  glBegin(GL_TRIANGLES);
    glVertex3f(1.0f, 1.0f, 1.0f);
    glVertex3f(10.0f, 1.0f, 1.0f);
    glVertex3f(10.0f, 10.0f, 1.0f);
  glEnd();
```

```
glEndList()
…
glCallList(1);
```

Displey ro‘yxatlari optimal, kompilyasiya qilingan holatda server xotirasida saqlanadi. Bu shakl primitivlarni maksimal tezlikda chizishga imkon beradi. Shu bilan birga, katta hajimli ma’lumotlar xotirani ko‘p qismini egallaydilar, bu o‘z navbatida, unumdorlikning pasayishiga olib keladi. Bunday katta hajmlarni (bir necha o‘n ming primitivlar) uchlar massivlari bilan chizish qulay.

Uchlar massivlari

Agar uchlar soni ko‘p bo‘lsa va ularning har birini glVertex*() buyrug‘i bilan chaqirmaslik uchun

void **glVertexPointer** (GLint *size*, GLenum *type*, GLsizei *stride*, void* *ptr*)

buyrug‘idan foydalanib, massivlarga birlashtirish qulayroq. Bu buyruq uchlarning koordinatalari va saqlash uslubini belgilaydi. Bu yerda *size* uchlarning koordinatalar sonini ko‘rsatadi (2,3,4 bo‘lishi mumkin), type ma’lumotlarning turini belgilaydi (**GL_ShORT**, **GL_INT**, **GL_FLOAT**, **GL_DOUBLE**). Ba’zan bitta massivda boshqa uchlarning atributlarini saqlash qulayroq, shunda *stride* parametri bitta uchning koordinatasidan ikkinchisining koordinatasigacha kuchishini belgilaydi. Agar *stride* nolga teng bulsa, bu degani koordinatalar ketma ket joylashgan. *Ptr* parametrida ma’lumotlar joylashgan manzil ko‘rsatiladi.

Shunga o‘xshab quyidagi buyruqlardan foydalanib normal, ranglar va uchlarni boshqa atributlarining massivlarini aniqlash mumkin:

void **glNormalPointer** (GLenum *type*, GLsizei *stride*, void **pointer*)

void **glColorPointer** (GLint *size*, GLenum *type*, GLsizei *stride*, void **pointer*)

Bu massivlarni keyinchak ham ishlatish uchun quyidagi buyruqni chaqirish kerak:

void **glEnableClientState** (GLenum *array*)

shu buyruq bilan keyingi parametrlar qo‘llaniladi:

GL_VERTEX_ARRAY, **GL_NORMAL_ARRAY**, **GL_COLOR_ARRAY.**

Massiv bilan ishlashni tugatgandan sung void **glDisableClientState** (GLenum *array*) buyrug‘ini qo‘llaymiz.

Massivning ichini qurish uchun

void **glArrayElement** (GLint *index*)

buyrug‘i ishlatiladi.

Bu buyruq *index* nomerga ega massiv elementlaridan foydalanib uchlarning atributlarini OpenGL ga yuboradi. Uning o‘rniga odatta

void **glDrawArrays** (GLenum *mode*, GLint *first*, GLsizei *count*)

buyrug‘i ishlatiladi. U *mode* parametri bilan aniqlanadigan *count* primitivlarni chizadi. Bunda u *first* dan *first*+*count*-1 gacha indeksli massiv elementlaridan foydalanadi. Bu glArrayElement() buyrug‘ini chaqirish bilan teng.

Agarda uch bir necha massivga kirsa uning koordinatalarini qaytarish o‘rniga massivda indeksini qo‘llash qulayroq.

Buning uchun

void **glDrawElements** (GLenum *mode*, GLsizei *count*, GLenum *type*, void **indices*)

buyrug‘i ishlatiladi, bu yerda *indices*-uchlarning raqamlar massivi. *Type* bu massivning elementlarini turini belgilaydi **GL_UNSIGNED_BYTE**, **GL_UNSIGNED_ShORT**, **GL_UNSIGNED_INT**, *count* esa ularning sonini belgilaydi.

Massivlarning qo‘llanishi OpenGL serveriga ma’lumotlarning jo‘natilishini optimal holatga keltiradi va shu bilan birga uch o‘lchamli sahnani chizishni tezlashtiradi. Bunday uslub primitivlarni aniqlash uchun juda tez va katta hajmli ma’lumotlarni vizualizasiyalash uchun juda qulay.

Nazorat savollari:

1. Teskari chaqiruv funksiyasi nima va OpenGL bilan ishlash uchun teskari chaqiruv funksiyasidan qanday foydalanish mumkin?

2. Tasvirlarni yangilash funksiyasi nima uchun kerak va qanday ish bajaradi?
3. OpenGL da primitiv degani nima?
4. Atribut nima? OpenGL da sizga ma’lum bo‘lgan uchlar atributlarini keltiring?
5. OpenGL dagi atomar primitiv deb nima aytiladi? Qanday turdagi primitivlarni bilasiz?
6. glEnable/glDisable buyruqlari OpenGL da nima uchun ishlatiladi?
7. Operatorli qavslar nima va ular OpenGL da qanday maqsadda ishlatiladi?
8. Displey ro‘yxatlari nima? Ro‘yxatni qaydan belgilash va uni tasvirlashga qanday chaqirish mumkin?
9. Massivlar bilan ishlashni tushuntirib bering va displey ro‘yxatlaridan farqlanishini ayting?
10. glDrawElements () buyrug‘ining ishlashini tushuntiring?

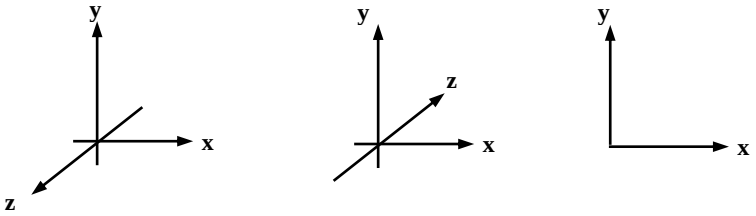
Tayanch iboralar: Tasvirni yangilash, virtual kamera, buferni tozalash, kuzatuvchini holati, geometrik obyektlar chizish, uchlarning fazodagi holati, uchlar rangi, uchlar massivlari, primitivlar.

2.3. Obyektlarni o‘zgartirish

OpenGl da geometric proektsiyalar. Matritsalar bilan ishlash. Parallel proektsiya. Markaziy proektsiya.

OpenGL da asosan uchta koordinalar sistemasidan foydalaniladi: chap tomonlama, o‘ng tomonlama va oynali. Birinchi ikkita sistema uch o‘lchovli hisoblanadi va bir-biridan *z* o‘qining yo‘nalishi bilan farqlanadi: o‘ng tomonlamada u kuzatuvchiga yo‘naltiriladi, chap tomonlamada esa ekran orqasiga yo‘naltiriladi. *x* o‘qi kuzatuvchiga nisbatan o‘ng tomonga, *y* o‘qi yuqoriga yo‘naladi.

Chap tomonli sistema gluPerspective(), glOrtho() buyruqlari parametrlari qiymatlarini berish uchun ishlatiladi. O‘ng tomonli koordinatalar sistemasi qolgan barcha holatlar uchun ishlatiladi. Uch o‘lchovli axborotlarni tasvirlash ikki o‘lchovli oynali koordinatalar sistemasida olib boriladi.



a) o'ng tomonlama sistema b) chap tomonlama sistema c) oynali sistema

3.4-rasm. OpenGL da koordinatalar sistemasi.

Shuni alohida ta'kidlash lozimki, OpenGL matritsalar bilan murakkab hisoblashlarni olib borish yo'lida o'ng va chap koordinatalar sistemasini modellashtirish imkonini beradi. OpenGL ning asosiy koordinatalar sistemasi o'ng tomonlama sistema hisoblanadi.

Matritsalar bilan ishlash.

OpenGLda sahna obyektlarini turlicha o'zgartirishni ko'rsatish uchun matritsalar ustida operatsiyalardan foydalaniladi, buning uchun matritsalar uch turga ajratiladi: modelli-tasvir, proeksiyalash matritsasi va teksturalash matritsasi. Ularning barchasi 4x4 o'lchamga ega. Tasvirli matritsa obyektni parallel ko'chirish, masshtabni o'zgartirish va burish singari eng yuqori koordinatalarda o'zgarishni belgilaydi. Proeksiyalash matritsasi, uch o'lchovli obyektning ekran tekisligiga (oynali koordinatalarda) proeksiyalanishini belgilaydi, teksturlash matritsasi obyekt teksturalar bilan qoplanishini belgilaydi.

Koordinatani matritsaga ko'paytirish, koordinatani belgilovchi (qoida sifatida, bu buyruq glVertex*) OpenGL ning tegishli buyrug'ini chaqirish vaqtida sodir bo'ladi.

Matritsani qanday o'zgartirish kerakligiga qarab kerakli buyruqlar ishlatiladi:

void **glMatrixMode** (GLenum *mode*)

mode parametrli qiymatga ega bo'lgan chaqiruv **GL_MODELVIEW, GL_PROJECTION,** yoki **GL_TEXTURE** larga teng bo'lib, mos holda modelli tasvir matritsasi, proeksiyalash matritsasi yoki teksturalash matritsalari bilan ishlash rejimini qamrab oladi. U yoki bu tipdagi matritsani tayinlash buyrug'ini chaqirish uchun, avvalo tegishli rejimni o'rnatib olish zarur.

Joriy tipdagi matritsa elementlarini aniqlash uchun quyidagi buyruq chaqiriladi:

void **glLoadMatrix [f d]** (GLtype **m*)

bu yerda *m* buyruqning nomlanishiga mos xolda float yoki double tipidagi 16 elementli massivni bildiradi. Bunday xolda dastlab unda matritsaning birinchi ustuni, so‘ngra ikkinchi, uchinchi va to‘rtinchi ustuni yozilishi kerak.

void **glLoadIdentity** (void) buyrug‘i joriy matritsani birlik matritsaga o‘zgartiradi.

Ko‘p hollarda joriy matritsa elementlarini saqlash talab qilinadi, buning uchun quyidagi buyruqlar ishlatiladi.

void **glPushMatrix** (void)

void **glPopMatrix** (void)

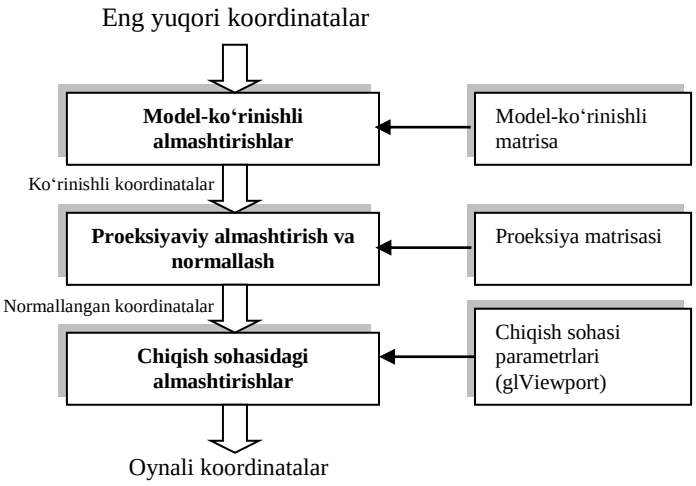
Ular joriy matritsani birlashmasidan yozib oladi va qayta tiklaydi, bunda har bir matritsa o‘zining birlashmasiga ega. Model-ko‘rinishli matritsalar uchun uning chuqurligi minimum 32 ga teng, boshqalar uchun minimum 2 ga teng.

Joriy matritsani boshqa matritsaga ko‘paytirish uchun quyidagi buyruq ishlatiladi:

void **glMultMatrix [f d]** (GLtype **m*) .

bunda *m* parametr 4x4 o‘lchamli matritsani berishi kerak. Agar joriy matritsani *M* deb, yuboriladigan matritsani *T* deb belgilasak, unda glMultMatrix buyrug‘i bajarilganda *M * T* joriy matritsa bo‘lib qoladi. Biroq u yoki bu turdagi matritsani o‘zgartirish uchun, qiymati jihatidan kerakli matritsani hosil qilib joriy matritsaga ko‘paytiradigan maxsus buyruqlarni qo‘llash qulay.

Umuman olganda, sahnaning uch o‘lchamli obyektlarini ilova oynasida tasvirlash uchun rasmda ko‘rsatilgan ketma-ketlik bajariladi.



2.5-rasm. OpenGL da koordinatalrni almashtirish.

Esda saqlang: OpenGL da obyektlar va kamerani almashtirishlar koordinata vektorlarini matritsaga ko‘paytirish orqali amalga oshiriladi. Bunda ko‘paytirish joriy matritsaga nisbatan glVertex* buyrug‘i yordamida koordinatalar aniqlangan vaqtda amalga oshiriladi.

Modelli-tasvir o‘zgartirishlar.

Modelli-tasvir o‘zgartirishlarini koordinata o‘qlari buylab ko‘chirish, burish va masshtabni o‘zgartirish deb hisoblaymiz. Ushbu operatsiyalarni bajarish uchun obyektning har bir uchi bilan tegishli matritsani ko‘paytirish va ushbu uchlarning o‘zgartirilgan koordinatalarini olish etarli:

$$(x', y', z', 1)^T = M \cdot (x, y, z)^T.$$

Bu yerda M – modelli-tasvir o‘zgartirish matritsasi. Matritsaning o‘zi quyidagi buyruqlar yordamida yaratilishi mumkin:

void **glTranslate [f d]** (GLtype *x*, GLtype *y*, GLtype *z*)

void **glRotate [f d]** (GLtype *angle*, GLtype *x*, GLtype *y*, GLtype z)

void **glScale [f d]** (GLtype *x*, GLtype *y*, GLtype *z*)

glTranlsate*() obyektni ko‘chirishga tayyorlaydi, o‘zining parametri qiymatlarini uning uchlari koordinatalariga qo‘shadi.

glRotate*() buyrug‘i obyektni (*x*,*y*,*z*) vektori atrofida *angle* (graduslarda o‘lchanadi) burchagi ostida soat strelkasiga teskari burilishini bajaradi.

glScale*() buyrug‘i (*x,y,z*) vektorlar o‘qi buyicha obyektni masshtablanishini bajaradi.

Obyekt holatini o‘zgartirishdan tashqari, kuzatuvchi holatini o‘zgartirish zarurati ham tug‘iladi. Buni quyidagi buyruq yordamida bajarish mumkin:

void **gluLookAt** (GLdouble *eyex*, GLdouble *eyey*,
GLdouble *eyez*, GLdouble *centerx*,
GLdouble *centery*, GLdouble *centerz*,
GLdouble *upx*, GLdouble *upy*,
GLdouble *upz*)

bu yerda (*eyex,eyey,eyez*) nuqtasi kuzatuvchi nuqtasini belgilaydi, (*centerx*, *centery*, *centerz*) chiqarish sohasi markazida proeksiyalanovchi sahna markazini beradi, (*upx,upy,upz*) vektori esa kamera burilishini aniqlab, *y* o‘qining musbat yo‘nalishini belgilaydi. Misol uchun agarda kamerani burish talab qilinmasa, (0,1,0) qiymati beriladi va sahnani (0,-1,0) qiymati bilan burish amalga oshiriladi.

Qisqacha aytganda, bu buyruq sahna obyektlarini ko‘chirish va burishni amalga oshiradi, lekin parametrlarni bunday ko‘rinishda berish qulayroq. Shuni ta’kidlash joizki, gluLookAt() buyrug‘ini chaqirish obyektlarni almashtirishdan oldin, ya’ni model-ko‘rinishli matritsa birlik matritsasiga teng bo‘lganda ma’noga ega.

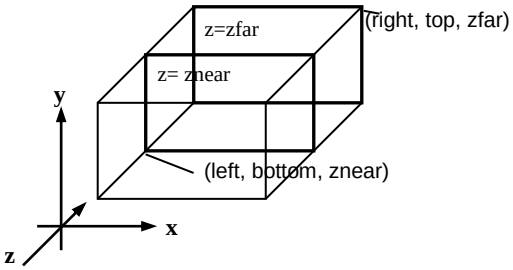
Eslab qoling: umumiy holda OpenGL da matritsaviy almashtirishlarni teskari tartibda yozib borish kerak. Misol uchun, agar siz avval obyektni burib, keyin ko‘chirmoqchi bo‘lsangiz, oldin glTranslate() buyrug‘ini, keyin esa glRotate() buyrug‘ini chaqirasiz. Bundan keyin esa obyektning o‘zini belgilaysiz.

Proeksiyalar.

OpenGL da ortografik (parallel) va perspektiv (markaziy) proeksiyalarni tayinlash uchun standart buyruqlar mavjud. Proeksiyalashning birinchi turi quyidagi buyruq orqali ifodalanadi:

void **glOrtho** (GLdouble *left*, GLdouble *right*,
GLdouble *bottom*, GLdouble *top*,
GLdouble *near*, GLdouble *far*)

void **gluOrtho2D** (GLdouble *left*, GLdouble *right*, GLdouble *bottom*, GLdouble *top*)



2.6-rasm. Ortografik proeksiya.

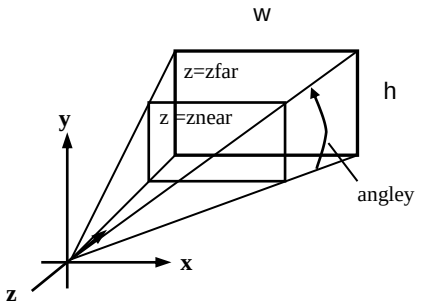
Birinchi buyruq chap tomonlama koordinatalar tizimida proeksiya matritsasini hosil qiladi. Buyruq parametrlari chiqarish oynasining chapki past va o‘ng yuqori burchaklariga javob beruvchi (*left, bottom, znear*) va (*right, top, zfar*) nuqtalarni beradi. *near* va *far* parametrlari yaqin va uzoq kesishish tekislilari orasidagi masofani yo‘qolish tartibida (0,0,0) nuqtasidan boshlab belgilaydi va ular manfiy bo‘lishi mumkin.

Ikkinchi buyruqda, birinchisidan farqli ravishda *near* va *far* qiymatlari mos ravishda -1 va 1 ga teng qilib belgilanadi. Bu hol OpenGL ikki o‘lchamli obyektlarni chizish uchun qo‘llanilganda qulay. Bunda cho‘qqilarning holatini glVertex2*() buyrug‘i yordamida berish mumkin.

Perspektiv proeksiyalash quyidagi buyruq bilan beriladi:

void **gluPerspective** (GLdouble *angley*, GLdouble *aspect*,
GLdouble *znear*, GLdouble *zfar*)

qaysiki chap tomonli koordinatalar sistemasida kesik konus ko‘rinishini beradi. *angley* parametri *y* o‘qi bo‘yicha graduslarda ko‘rish burchagini belgilaydi va 0 dan 180 gacha oraliqda graduslarni aniqlaydi. *x* o‘qi bo‘ylab ko‘rish burchagi *aspect* parametri bilan beriladi. *x* o‘qi bo‘ylab ko‘rish burchagi *aspect* parametri bilan, odatda chiqarish sohasi tomonlari orasidagi munosabat sifatida beriladi.



2.7-rasm. Perspektiv proeksiya.

zfar va *znear* parametrlari kuzatuvchidan kesishish chuqurligi bo‘ylab kesilgan tekisliklarigacha masofani belgilaydi, ular musbat bo‘lishi kerak. *zfar*/*znear* orasidagi munosabat qanchalik katta bo‘lsa, bufer chuqurligida unga yaqin joylashgan tekislik shunchalik yomon farqlanadi, chunki shart berilmaganda unga 0 dan 1 gacha oraliqda chuqurlikning “siqilgan” sonlari yoziladi.

Proeksiyalarning matritsasini berishdan oldin, glMatrixMode(GL_PROJECTION) buyrug‘i yordamida kerakli matritsa bilan ishlash rejimini yoqishni hamda glLoadIdentity() ni chaqirib joriy sozlashlarni bekor qilishni unutmaslik lozim.

Misol uchun:

```
/* ortografik proeksiya */
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0, w, 0, h, -1.0, 1.0);
```

Chiqarish sohasi.

Proeksiya matritsasi qo‘llanilgandan so‘ng, keyingi almashtirish kirishiga kesilgan (clipped) koordinatalar deb ataluvchilar beriladi. Keyin esa uchlarning normallangan koordinatalari quyidagi formula yordamida topiladi:

$$(x_n, y_n, z_n)^T = \left(\frac{x_c}{w_c}, \frac{y_c}{w_c}, \frac{z_c}{w_c} \right)^T .$$

Chiqarish sohasi koordinatalar oynaviy tizimida to‘g‘ri to‘rtburchakni aks ettiradi, uning o‘lchamlari esa quyidagi buyruq yordamida beriladi:

void **glViewPort** (GLint *x*, GLint *y*, GLint *width*, GLint *height*).

Barcha parametrlarning qiymatlari piksellarda beriladi va koordinatalar tizimining oynaviy tizimida chap pastki burchak koordinatalari (*x*,*y*) bilan chiqarish sohasining kengligi va balandligini aniqlaydi. Koordinatalar oynaviy tizimining o‘lchamlari ilova oynasining joriy o‘lchamlari bilan aniqlanadi, (0,0) nuqtasi oynaning chap pastki burchagida joylashgan.

glViewPort() buyrug‘i parametrlarini qo‘llab OpenGL chiqarish sohasi markazining (o_x,o_y) oynali koordinatalarini quyidagi $o_x=x+width/2$, $o_y=y+height/2$ formulalar bilan aniqlaydi.

Aytaylik, $p_x=width$, $p_y=height$ bo‘lsin, unda har bir uchning oynaviy koordinatalarini topish mumkin:

$$(x_w, y_w, z_w)^T = \left(\left(\frac{p_x}{2} \right) \cdot x_n + o_x, \left(\frac{p_y}{2} \right) \cdot y_n + o_y, \left(\frac{f-n}{2} \right) \cdot z_n + \frac{(n+f)}{2} \right)^T.$$

Bunda *n* va *f* musbat butun qiymatlari oynada nuqtaning minimal maksimal chuqurligini beradi va shart berilmaganda, mos ravishda 0 va 1 ga teng. Har bir nuqtaning chuqurligi maxsus chuqurlik buferiga (z-bufer) yozib boriladi va u ko‘ririnmas chiziq va sirtlarni o‘chirish uchun ishlatiladi. *n* va *f* qiymatlarini quyidagi funksiyalar yordamida o‘rnatish mumkin.

void **glDepthRange** (GLclampd *n*, GLclampd *f*)

glViewPort() buyrug‘i odatda, foydalanuvchi oyna o‘lchamlarini o‘zgartirganda chaqiriladigan glutReshapeFunc () buyrug‘i bilan qayd qilingan funksiyada ishlatiladi.

Nazorat savollari:

1. OpenGL da qanday koordinatalar sistemasi ishlatiladi?
2. OpenGL da matritsali o‘zgartirish ko‘rinishlarini keltiring?
3. OpenGL da obyektlar qanday qilib o‘zgartiriladi?

4. OpenGL da proeksiyalar qanday tayinlanadi?
5. Matritsali stek deganda nimani tushunasiz?
6. OpenGL da kuzatuvchi holatini o‘zgartirish usullarini keltiring?
7. glTranslate(), glRotate() va glScale() buyruqlarini chaqirishning qanday ketma-ketligi gluLookAt(0, 0, -10, 10, 0, 0, 0, -1, 0) buyrug‘iga mos keladi?
8. Proeksiyalarni tayinlash uchun qanday standart buyruqlarni bilasiz?

Tayanch iboralar: Koordinalar sistemasi, matritsalar ustida amallar, modelli-tasvir, proeksiyalash matritsasi, teksturalash matritsasi, proeksiyalash, chiqarish sohasi.

2.4. OpenGL yordamida materiallar.

Materiallar tasnifi. Tuman effektini hosil qilish. Teksturalash. Teksturani tayyorlash. Ob'ektlarga tekstura qo'yish. Teksturali koordinata

...

2.5. OpenGL yordamida yorug‘lik.

Materiallar tasnifi. Yorug‘lik modeli. Yorug‘lik manbalarining tavsifi. Tuman effektini hosil qilish.

Haqiqiy tasvirlarni hosil qilish uchun obyektning xossalarini hamda u joylashgan muhit xossalarini ham aniqlash zarur. Xossalarning birinchi guruhi obyekt yasalgan materialning parametrlari, uning yuzasiga teksturani qo‘yish usullari, obyektning shaffoflik darajasini o‘z ichiga oladi. Ikkinchi guruhga yorug‘lik manbalarining soni va xususiyatlarini, muhitning shaffoflik darajasini hamda yoritish modelini kiritish mumkin. Bu xossalarning barchasini OpenGL ning mos buyruqlarini chaqirgan holda tayinlash mumkin.

Yorug‘lik modeli.

OpenGL da yoritish modeli ishlatiladi va unga muvofiq nuqta rangi bir necha faktorlar bilan belgilanadi: material va tekstura xossalari, shu nuqtadagi normal kattaligi hamda yorug‘lik manbai va kuzatuvchi holati bilan. Nuqtadagi yorug‘likni aniq hisoblash uchun birlik normallardan foydalanish kerak, lekin glScale*() turidagi buyruqlar normallar uzunligini o‘zgartirishi mumkin.

Yorug‘likning global parametrlarini berish uchun

void **glLightModel [i f]** (GLenum *pname*, GLenum *param*)

void **glLightModel[i f]v** (GLenum *pname,* const GLtype **params*)

buyruqlaridan foydalaniladi.

pname argumenti yorug‘lik modelining qaysi parametri sozlanishini va quyidagi qiymatlarni qabul qilishi mumkinligini belgilaydi:

GL_LIGHT_MODEL_LOCAL_VIEWER *param* parametri bul toifasiga tegishli bo‘lishi kerak va kuzatuvchining holatini belgilaydi. Agar u GL_FALSE ga teng bo‘lsa, unda tasvir yo‘nalishi ko‘rinishli koordinatalardagi holatiga bog‘liq bo‘lmagan holda, – z o‘qiga

parallel bo‘lib hisoblanadi. Agar u GL_TRUE ga teng bo‘lsa, unda kuzatuvchi ko‘rinishli koordinatalar tizimining boshida joylashgan. Bu yorug‘lik sifatini oshirishi mumkin, lekin uni hisoblashni qiyinlashtiradi. Shart qo‘yilmaganda qiymat GL_FALSE bo‘ladi.

GL_LIGHT_MODEL_TWO_SIDE *param* parametri bul toifasiga tegishli bo‘lishi kerak hamda yuza va teskari yoqlar uchun yorug‘lik hisoblash rejimini boshqaradi. Agar u GL_FALSE ga teng bo‘lsa, unda yoritilganlik faqat yuza yoqlari uchun hisoblanadi. Agarda u GL_TRUE ga teng bo‘lsa, hisoblash teskari yoqlar uchun ham amalga oshiriladi. Shart qo‘yilmaganda qiymat GL_FALSE ga teng.

GL_LIGHT_MODEL_AMBIENT *params* parametri to‘rtta butun yoki haqiqiy sonlarga ega bo‘lishi kerak, bu sonlar hattoki ma’lum bir yorug‘lik manbalari bo‘lmaganda ham fonning yorug‘ligi rangini belgilaydi. Shart qo‘yilmaganda qiymatlar: (0.2, 0.2, 0.2,1.0) ga teng.

Materiallar tasnifi

Joriy materialning parametrlarini berish uchun quyidagi buyruqlardan foydalaniladi:

void **glMaterial [i f]** (GLenum *face*, GLenum *pname*, GLtype *param*)

void **glMaterial[i f]v** (GLenum *face*, GLenum *pname*, GLtype **params*)

Ular yordamida materialning ko‘zguli, diffuziyali, tarqoq ranglarini, hamda agar obyekt nurlansa, nurning tarqalish intensivligini va ko‘zguli akslantirishni aniqlash mumkin. param qiymati bilan aynan qaysi parametr tavsiflanishi *pname* qiymatiga bog‘liq:

GL_AMBIENT *params* parametri RGBA ranglarining to‘rtta butun yoki haqiqiy qiymatlariga ega bo‘lishi kerak, ular esa o‘z navbatida materialning tarqoq rangini belgilaydi (materialning soyadagi

rangi). Shart qo‘yilmaganda qiymatlar (0.2, 0.2, 0.2, 1.0) ga teng bo‘ladi.

GL_DIFFUSE *params* parametri RGBA ranglarining to‘rtta butun yoki haqiqiy qiymatlariga ega bo‘lishi kerak, ular esa o‘z navbatida materialning diffuziyali rangini aniqlaydi. Shart qo‘yilmaganda qiymatlar (0.8, 0.8, 0.8, 1.0) ga teng.

GL_SPECULAR *params* parametri RGBA ranglarining to‘rtta butun yoki haqiqiy qiymatlariga ega bo‘lishi kerak, ular esa o‘z navbatida materialning ko‘zguli rangini aniqlaydi. Shart qo‘yilmaganda qiymatlar (0.0, 0.0, 0.0, 1.0) ga teng.

GL_SHININESS *params* parametri 0 dan 128 gacha oraliqda bitta butun yoki haqiqiy qiymatga ega bo‘lishi kerak va u materialning ko‘zguli akslanish darajasini aniqlaydi. Shart qo‘yilmaganda qiymat 0 ga teng.

GL_EMISSION *params* parametri RGBA ranglarining to‘rtta butun yoki haqiqiy qiymatlariga ega bo‘lishi kerak, ular esa o‘z navbatida materialdan tarqalayotgan yorug‘lik intensivligini aniqlaydi. Shart qo‘yilmaganda qiymatlar (0.0, 0.0, 0.0, 1.0) ga teng.

GL_AMBIENT_AND_DIFFUSE glMaterial*() buyrug‘ini pname **GL_AMBIENT** va **GL_DIFFUSE** qiymati hamda *params* bir xil qiymatlari bilan ikki marta chaqirishga ekvivalent.

Bundan kelib chiqadiki, glMaterial[i f]() buyrug‘ini chaqirish faqat materialning ko‘zguli akslanish darajasi o‘rnatilishi uchungina mumkin bo‘ladi. glMaterial[i f]v() buyrug‘i qolgan parametrlarni berish uchun ishlatiladi.

face parametri bu material beradigan yoqlar turini aniqlaydi va **GL_FRONT**, **GL_BACK** yoki **GL_FRONT_AND_BACK** qiymatlarini qabul qilishi mumkin.

Agar sahnada obyektlarning materiallari faqat bitta parametr bilan farqlansa, glEnable() ni **GL_COLOR_MATERIAL** parametri bilan chaqirib kerakli rejimni o‘rnatish tavsiya etiladi, keyin esa quyidagi buyruqdan foydalaniladi:

void **glColorMaterial** (GLenum *face*, GLenum *pname*)

bu yerda, *face* parametri yuqorida ko‘rsatilgan ma’noga ega, *pname* parametri esa barcha sanab o‘tilgan qiymatlarni qabul qilishi mumkin. Bundan keyin *pname* aniq obyektning materiali xossalari yordamida olingan qiymatlar glColor*() buyrug‘ini chaqirish orqali o‘rnatiladi va bu ko‘p resurslarni talab qiladigan glMaterial*() buyrug‘i chaqirilishini oldini oladi hamda dastur samaradorligini oshiradi.

Material xossalarini aniqlashga misol:

```
float mat_dif[]={0.8,0.8,0.8};
float mat_amb[] = {0.2, 0.2, 0.2};
float mat_spec[] = {0.6, 0.6, 0.6};
float shininess = 0.7 * 128;
…
glMaterialfv (GL_FRONT_AND_BACK,GL_AMBIENT, mat_amb);
glMaterialfv (GL_FRONT_AND_BACK,GL_DIFFUSE, mat_dif);
glMaterialfv (GL_FRONT_AND_BACK,GL_SPECULAR, mat_spec);
glMaterialf  (GL_FRONT,GL_ShININESS, shininess);
```

Yorug‘lik manbalarining tavsifi.

Obyekt materialining xossalarini aniqlash faqat sahnada yorug‘lik manbalari bo‘lgandagina o‘rinli. Aks holda barcha obyektlar qora rangda (materialning tarqoq rangida) bo‘ladi. Sahnaga yorug‘lik manbaini quyidagi buyruq yordamida qo‘shish mumkin:

void **glLight [i f]** (GLenum *light*, GLenum *pname*,
GLfloat *param*)
void **glLight[i f]** (GLenum *light*, GLenum *pname*,
GLfloat **params*)

light parametri o‘z-o‘zidan yorug‘lik manbaini belgilaydi. U **GL_LIGHTi** ko‘rinishdagi maxsus belgili nomlar to‘plami bilan tanlanadi, *i* bu yerda 0 dan odatda sakkizdan katta bo‘lmaydigan **GL_MAX_LIGHT** o‘zgarmaslari orasida yotadi.

pname va *params* parametrlari glMaterial*() buyrug‘iga o‘xshash ma’noga ega. *pname* parametri qiymatlarini qarab chiqamiz:

GL_SPOT_EXPONENT *param* parametri yorug‘lik intensivligi tarqalishini beradigan 0 dan 128 gacha butun yoki haqiqiy qiymatga ega bo‘lishi kerak. Bu parametr yorug‘lik manbaining fokuslanish darajasini tavsiflaydi. Shart qo‘yilmaganda qiymat 0 ga teng (tarqoq yorug‘lik).

GL_SPOT_CUTOFF *param* parametri yorug‘lik tarqalishining maksimal burchagini aniqlaydi. Bu parametrning qiymati manba tomonidan yaratiladigan konus ko‘rinishdagi yorug‘lik oqimi uchidagi burchakning yarmiga teng. Shart qo‘yilmaganda qiymat 180 (tarqoq yorug‘lik) ga teng.

GL_AMBIENT *params* parametri, fondagi yoritilganlikni aniqlaydigan, RGBA ranglarining to‘rtta butun yoki haqiqiy qiymatlariga ega bo‘lishi kerak. Shart qo‘yilmaganda qiymatlar (0.0, 0.0, 0.0, 1.0) ga teng.

GL_DIFFUSE *params* parametri, diffuziyali yoritilishni aniqlaydigan, RGBA ranglarining to‘rtta butun yoki haqiqiy qiymatlariga ega bo‘lishi kerak. Shart qo‘yilmaganda qiymatlar **GL_LIGHT0** uchun (1.0, 1.0, 1.0, 1.0) ga, qolganlari uchun (0.0, 0.0, 0.0, 1.0) ga teng.

GL_SPECULAR *params* parametri, ko‘zguli akslanishni aniqlaydigan, RGBA ranglarining to‘rtta butun yoki haqiqiy qiymatlariga ega bo‘lishi kerak. Shart qo‘yilmaganda qiymatlar **GL_LIGHT0** uchun (1.0, 1.0, 1.0, 1.0) ga va qolganlari uchun (0.0, 0.0, 0.0, 1.0) ga teng.

GL_POSITION *params* parametri, yorug‘lik manbaining joyini aniqlaydigan to‘rtta butun yoki haqiqiy qiymatlarga ega bo‘lishi kerak. Agar w komponentining qiymati 0.0 ga teng bo‘lsa, unda manba cheksiz uzoqda deb hisoblanadi va yoritilganlikni hisoblashda (x,y,z)

nuqtaga nisbatan yo‘nalish hisobga olinadi, aks holda manba (x,y,z,w) nuqtada joylashgan deb hisoblanadi. Birinchi holatda manbadan uzoqlashganda yorug‘lik kamayishi kuzatilmaydi, ya’ni manba cheksiz uzoqda deb olinadi. Shart qo‘yilmaganda qiymatlar (0.0, 0.0, 1.0, 0.0) ga teng.

GL_SPOT_DIRECTION *params* parametri yorug‘lik yo‘nalishini aniqlaydigan to‘rtta butun yoki haqiqiy qiymatlarga ega bo‘lishi kerak. Shart qo‘yilmaganda qiymatlar (0.0, 0.0, -1.0, 1.0) ga teng. Manbaning bu tavsifi **GL_SPOT_CUTOFF** qiymati 180 dan farqli bo‘lganda ma’noga ega.

GL_CONSTANT_ATTENUATION,
GL_LINEAR_ATTENUATION,
GL_QUADRATIC_ATTENUATION

params parametri manbadan uzoqlashtirilganda yorug‘lik intensivligini kamayishini aniqlaydigan uchta koeffisientdan bittasini beradi. Faqat manfiy bo‘lmagan qiymatlar o‘rinli. Agar manba yo‘naltirilmagan bo‘lsa, unda kamayish quyidagi summaga teskari proporsional:

$$\mathbf{att}_{constant} + att_{linear} * d + att_{quadratic} * d^2,$$

bu yerda d – yorug‘lik manbai va u yoritadigan uch orasidagi masofa, $\mathbf{att}_{constant}$, att_{linear} i $att_{quadratic}$ mos ravishda **GL_CONSTANT_ATTENUATION**, **GL_LINEAR_ATTENUATION** va **GL_QUADRATIC_ATTENUATION** konstantalari bilan berilgan parametrlarga teng. Shart qo‘yilmaganda parametrlar uchlik bilan beriladi (1, 0, 0) va yorug‘lik kamayishi sodir bo‘lmaydi.

Manba holati o‘zgarganda quyidagi omilni hisobga olish zarur: OpenGL da yorug‘lik manbalari ko‘p hollarda ko‘pburchaklar va nuqtalarga o‘xshagan obyektlar hisoblanadi. Ularga nisbatan OpenGL da koordinatalar qayta ishlanishining asosiy qoidalari o‘rinli, ya’ni fazodagi holatni tavsiflaydigan parametrlar joriy model-ko‘rinishli matritsa bilan obyekt o‘zgartirilayotganda, ya’ni OpenGL ning mos buyruqlari chaqirilganda almashtiriladi. Shunday qilib, yorug‘lik manbaini sahna obyekti yoki kamera bilan bir vaqtda o‘zgartirib, uni shu obyektga bog‘lash mumkin. Yoki aksincha boshqa obyektlar ko‘chayotganda, joyida qoladigan stasionar yorug‘lik manbaini tuzish mumkin.

Umumiy qoida quyida keltirilgan:

Agar yorug‘lik manbai holati virtual kamera holatini belgilashdan oldin glLight*() buyrug‘i bilan berilgan bo‘lsa, unda manba koordinatalari (0,0,0) kuzatish nuqtasida joylashgan deb hisoblanadi va shunga muvofiq yorug‘lik manbai holati kuzatuvchi holatiga nisbatan aniqlanadi.

Agar holat kamera holati va obyektning model-ko‘rinishli matritsasi almashtirishlari orasidagi belgilanish orqali o‘rnatilsa, unda u qo‘zg‘almas holatga keltiriladi, ya’ni bu holda yorug‘lik manbai holati eng yuqori koordinatalarda beriladi.

Yorug‘likdan foydalanish uchun dastlab mos rejim glEnable(GL_LIGHTNING) buyrug‘ini chaqirish orqali o‘rnatiladi, keyin esa kerakli manba glEnable(GL_LIGHTi) buyrug‘i bilan yoqiladi.

Yana bir bor e’tiborimizni shunga qaratishimiz kerakki, yorug‘lik o‘chirilganda uchning rangi glColor*() buyrug‘i bilan beriladigan joriy rangga teng. Yorug‘lik o‘chirilganda uchning rangi yorug‘lik manbalari, normallar va material to‘g‘risidagi ma’lumotlardan kelib chiqib hisoblanadi.

Yorug‘lik o‘chirilganda vizualizasiya tezroq amalga oshadi, lekin bu holda ilovaning o‘zi uchlarning ranglarini hisoblashi kerak bo‘ladi.

Tuman effektini hosil qilish.

Yakunda OpenGL ning qiziqarli va ko‘p ishlatiladigan imkoniyati – tuman effektini hosil qilishni ko‘rib chiqamiz. Sahnaning yengil tumanlanishi haqqoniy effektni hosil qiladi hamda sahnada uzoqlashgan obyektlar bo‘lganda hosil bo‘ladigan ba’zi artefaktlarni qisman yashirishi mumkin.

OpenGL da tuman sahnadagi obyektlarning rangini ularning chuqurligiga, ya’ni kuzatish nuqtasigacha bo‘lgan masofaga bog‘liq ravishda o‘zgartirish orqali amalga oshiradi. Rangning o‘zgarishi OpenGL ni amalga oshirishga bog‘liq ravishda rasterizasiya pog‘onasida har bir piksel uchun yoki primitivlarining uchlari uchun sodir bo‘ladi. Bu jarayonni qisman boshqarish mumkin.

Tumanlash effektini yoqish uchun **glEnable(GL_FOG)** buyrug‘ini chaqirish zarur.

Uchdagi tuman intensivligini hisoblash usulini quyidagi buyruqlar yordamida aniqlash mumkin:

void **glFog [if]** (enum *pname*, T *param*);

void **glFog[if]v** (enum *pname*, T *params*);

pname argumenti quyidagi qiymatlarni qabul qilishi mumkin:

GL_FOG_MODE *param* argumenti nuqtadagi tuman intensivligi hisoblanadigan formulani aniqlaydi.

Bu holda *param* quyidagi qiymatlarni qabul qilishi mumkin:

GL_EXP Intensivlik $f=\exp(-d*z)$ formula bilan topiladi

GL_EXP2 Intensivlik $f=\exp(-(d*z)^2)$ formula bilan topiladi

GL_LINEAR Intensivlik $f=e-z/e-s$ formula bilan topiladi

Bu yerda *z* – tuman intensivligi hisoblanadigan uchdan kuzatish nuqtasigacha bo‘lgan masofa.

d,*e*,*s* koeffisientlari *pname* argumentining quyidagi qiymatlari bilan beriladi:

GL_FOG_DENSITY *param d* koeffisientini aniqlaydi

GL_FOG_START *param s* koeffisientini aniqlaydi

GL_FOG_END *param e* koeffisientini aniqlaydi

Tuman rangi **GL_FOG_COLOR** ga teng bo‘lgan *pname* argumenti yordamida beriladi. Bu holda *params* – 4 ta rang komponentiga ega massivga ko‘rsatkich.

Bu effektni qo‘llashga misol keltiramiz:

```
GLfloat FogColor[4]={0.5,0.5,0.5,1};
glEnable(GL_FOG);
glFogi(GL_FOG_MODE,GL_LINEAR);
glFogf(GL_FOG_START,20.0);
glFogf(GL_FOG_END,100.0);
glFogfv(GL_FOG_COLOR,FogColor);
```

Nazorat savollari:

1. Yorug‘likning lokal va cheksiz uzoqlashgan manbalari o‘rtasidagi farqni tushuntiring?
2. glColorMaterial buyrug‘i nima uchun xizmat qiladi?
3. Yorug‘lik manbasi har doim kuzatuvchi holati nuqtasida bo‘lishi uchun uning holati qanday o‘rnatiladi?
4. Yorug‘lik manbasining fiksirlangan holati qanday o‘rnatiladi?
5. Obyektning lokal koordinatalariga nisbatan manba holatini o‘rnatish mumkinmi?
6. Yorug‘likning konusli manbasi qanday beriladi?
7. Yorug‘lik manbai holati kuzatuvchi holatiga nisbatan qanday aniqlanadi?
8. Agar sahna yoritilgan bo‘lib, ammo yorug‘lik manbai yo‘q bo‘lsa, unda obyektlar qanday rangga ega bo‘ladi?

Tayanch iboralar: Yorug‘lik modeli, material, yorug‘lik manbalari, yorug‘lik intensivligi, diffuziyali yoritilish, ko‘zguli akslanish, tuman effekti.

2.6. Teksturalash

Teksturalash. Teksturani tayyorlash. Ob'ektlarga tekstura qo'yish. Teksturali koordinata.

Tekstura so'zi deganda obyektga ma'lum bir usulda qo'yish kerak bo'lgan qandaydir tasvir tushuniladi.

Sahna obyektlari yuzasiga tekstura qo'yilganda uning haqqoniyligi oshadi, lekin shuni ham hisobga olish lozimki, bu jarayon ortiqcha hisob-kitobni talab qiladi.

Tekstura bilan ishlash uchun quyidagi harakatlar ketma-ketligini amalga oshirish kerak:

1. Tasvirni tanlash va uni kerakli formatga keltirish;
2. Tasvirni OpenGL ga uzatish;
3. Tekstura obyektga qanday qo'yilishini va u bilan qanday munosabatda bo'lishini aniqlab olish;
4. Teksturani obyekt bilan bog'lash.

Teksturani tayorlash.

Teksturadan foydalanish uchun, oldin xotiraga kerakli tasvirni joylashtirish va uni OpenGL ga uzatish kerak.

Grafikaviy ma'lumotlarni fayldan o'qishni va ularni almashtirishini qo'lda bajarish mumkin. Shu bilan birga GLAUX (undan foydalanish uchun qo'shimcha ravishda glaux.lib ni bog'lash kerak bo'ladi) kutubxonasiga kiruvchi muhim amallarni o'zi bajaradigan funksiyadan foydalanish mumkin. Bu funksiya

AUX_RGBImageRec* **auxDIBImageLoad** (const char **file*)

bu yerda *file* – *.bmp yoki *.dib kengaytmali fayl nomi. Funksiya ko'rsatkichni qayta ishlangan ma'lumotlar saqlanadigan xotira sohasiga qaytardi.

Xotirada tekstura obrazini hosil qilishda quyidagi talablarni hisobga olish kerak.

Birinchidan, teksturaning o'lchamlari, ham gorizontal bo'yicha, ham vertikal bo'yicha o'zida ikkining darajalarini aks ettirishi lozim. Bu talab teksturani tekstura xotirasida kompakt holda qo'yilishini ko'zda tutadi va uning samarali qo'llanilishini

ta’minlaydi. Faqat shunday teksturalar bilan ishlash noqulay, shuning uchun joylashtirilgandan keyin ularni almashtirish kerak. Tekstura o‘lchamlarini quyidagi buyruq yordamida o‘zgartirish mumkin:

void **gluScaleImage** (GLenum *format*, GLint *widthin*,
GL *heightin*, GLenum *typein*,
const void **datain*,
GLint *widthout*,
GLint *heightout*, GLenum *typeout*,
void **dataout*)

Format parametrining qiymati sifatida odatda ma’lumotlarni saqlash formatini belgilaydigan **GL_RGB** yoki **GL_RGBA** qiymati ishlatiladi. *widthin*, *heightin*, *widhtout*, *heightout* parametrlari kiruvchi va chiquvchi tasvirlarining o‘lchamlarini aniqlaydi, *typein* va *typeout* yordamida esa *datain* va *dataout* manzillarida joylashgan massiv elementlarining turi beriladi. Odatiy holga o‘xshash bu **GL_UNSIGNED_BYTE, GL_ShORT, GL_INT** va boshqa turlar bo‘lishi mumkin. Funksiya o‘z ishining natijasini *dataout* parametrini ko‘rsatadigan xotira sohasiga kiritadi.

Ikkinchidan, obyekt rasterizasiyadan keyin o‘lchamlari jihatidan unga qo‘yiladigan teksturadan kichkina bo‘lib qolishi holini ko‘zda tutish kerak. Obyekt qanchalik kichkina bo‘lsa, unga qo‘yiladigan tekstura ham shunchalik kichkina bo‘lishi kerak, shuning uchun ham *teksturani detalizasiyalash pog‘onalari* degan tushuncha kiritiladi (mipmap). Detalizasiyaning har bir pog‘onasi originalning ikki baravar kichiklashtirilgan nusxasi sifatidagi qandaydir tasvirni beradi. Bunday yondashuv obyektga tekstura qo‘yishni osonlashtiradi. Masalan, $2m$x$2n$ o‘lchamdagi tasvir uchun detalizasiyaning turli pog‘onalariga mos keluvchi $\max(m,n)+1$ kichiklashtirilgan tasvirlarni qurish mumkin.

OpenGL ning ichki xotirasida tekstura obrazini yaratishning bu ikki pog‘onasini quyidagi buyruqlar yordamida olib borish mumkin:

void **gluBuild2DMipmaps** (GLenum *target*, GLint *components*,
GLint *width*, GLint *height*,
GLenum *format*, GLenum *type*,
const void **data*)

bu yerda *target* parametri **GL_TEXTURE_2D** ga teng bo‘lishi kerak. *components* parametri teksturaning rang komponentlarini aniqlaydi va quyidagi asosiy qiymatlarni qabul qilishi mumkin:

GL_LUMINANCE bitta komponent – yorqinlik (tekstura monoxrom bo‘ladi)

GL_RGB qizil, ko‘k, yashil

GL_RGBA barcha komponentlar.

width, *height*, *data* parametrlari teksturaning joylashuvi va o‘lchamlarini aniqlaydi, *format* va *type* esa gluScaleImage() buyrug‘iga o‘xshash ma’noga ega.

Bu buyruq amalga oshrilgandan keyin, tekstura OpenGL ning ichki xotirasiga nusxalanadi va shuning uchun joriy tasvir band qilib turgan xotirani bo‘shatish mumkin.

OpenGL da bir o‘lchamli, ya’ni 1xN teksturalardan foydalanishga ruxsat berilgan, lekin buni har doim *target* sifatida **GL_TEXTURE_1D** o‘zgarmaslarini berib ko‘rsatib o‘tish kerak. Bir o‘lchamli teksturalarning kerakliligi shubha ostida bo‘lganligi sababli ularga batafsil to‘htalib o‘tmaymiz.

Sahnada bir necha teksturalar qo‘llanilganda OpenGL da tasvirlar ro‘yxatini (teksturali obyektlar) yaratishni eslatuvchi amal bajariladi. Avval

void **glGenTextures** (GLsizei *n*, GLuint* *textures*)

buyrug‘i yordamida *textures* massiviga yoziladigan teksturalar *n* identifikatorlarini hosil qilish kerak. Keyingi teksturaning xossalarini aniqlashni boshlashdan oldin uni quyidagi buyruqni chaqirib joriy qilish kerak:

void **glBindTexture** (GLenum *target*, GLuint *texture*)

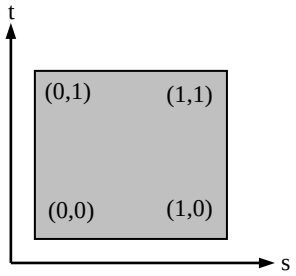
bu yerda *target* **GL_TEXTURE_1D** yoki **GL_TEXTURE_2D** qiymatlarini qabul qilishi mumkin, *texture* parametri esa keyingi buyruqlar taaluqli bo‘lgan tekstura

identifikatoriga teng bo‘lishi kerak. Chizish jarayonida joriy teksturani bir qancha identifikatorlar bilan bajarish uchun *target* va *texture* qiymatlariga mos glBindTexture() buyrug‘ini takroran chaqirish yetarli. Shu tarzda, glBindTexture() buyrug‘i *texture* identifikatori bilan teksturalar yaratish rejimini kiritadi, agarda bunday tekstura yaratilmagan bo‘lsa, yoki uning bajarilish rejimi, unda ushbu teksturani joriy etib tayinlaydi.

Chunki barcha apparaturalar ham katta hajmdagi teksturalardan foydalana olmaydi, teksturalar o‘lchamini 256x256 yoki 512x512 pikselgacha cheklash maqsadga muvofiq. Eslatib o‘tamiz, katta bo‘lmagan teksturalardan foydalanish dastur samaradorligini oshiradi.

Obyektlarga teksturalar qo‘yish.

Teksturalar qo‘yishda, yuqorida ta’kidlanganidek, teksturaning o‘lchami u qo‘yilayotgan obyektning oynasi o‘lchamidan farq qilishi holatlarini hisobga olish kerak. Bunda tasvirni cho‘zish, siqish, ushbu o‘zgartirishlar qanday olib borilishi, qurilayotgan tasvir sifatiga jiddiy ta’sir qilishi mumkin. Teksturada nuqta holatini aniqlash uchun koordinalarning parametrik tizimi (s,t) dan foydalaniladi, *s* va *t* qiymatlari [0,1] oraliqda yotadi (3.8. rasm).



3.8-rasm. Tekstura koordinatalari.

Teksturaning turli parametrlarini o‘zgartirish uchun quyidagi buyruqlardan foydalaniladi:

void **glTexParameter [i f]** (GLenum *target*, GLenum *pname*,GLenum *param*)

void **glTexParameter[i f]v** (GLenum *target*, GLenum *pname*, Glenum* *params*)

Shunda *target* **GL_TEXTURE_1D** yoki **GL_TEXTURE_2D** qiymatlarini qabul qilishi mumkin, *prame* qaysi xususiyatni almashtirishimizni belgilaydi, *param* yoki *params* yordamida yangi qiymat o‘rnatiladi. *pname* ning mumkin bo‘lgan qiymatlari:

GL_TEXTURE_MIN_FILTER *param* parametri teksturalarni siqish uchun ishlatiladigan funksiyani belgilaydi. **GL_NEAREST** qiymatida bitta (yaqin) va **GL_LINEAR** qiymatida to‘rtta yaqin tekstura elementlari ishlatiladi. Shart kiritilmagandagi qiymati: **GL_LINEAR**.

GL_TEXTURE_MAG_FILTER *param* parametri teksturani kattalashtirish (cho‘zish) uchun ishlatiladigan funksiyani belgilaydi. **GL_NEAREST** qiymatida bitta (yaqin) teskura elementi ishlatiladi. Shart kiritilmagandagi qiymati: **GL_LINEAR.**

GL_TEXTURE_WRAP_S *param* parametri *s* koordinata qiymatini o‘rnatadi, agar u [0,1] kesma oralig‘ida bo‘lmasa**. GL_ REPEAT** qiymatida *s* ning butun qismi olib tashlanadi va natijada tasvir yuza bo‘yicha ko‘payadi. **GL_CLAMP** qiymatida chetki qiymatlar ishlatiladi: 0 yoki 1, agar obyekt ustiga bitta tasvir tushirilsa qo‘llash qulay bo‘ladi. Shart kiritilmagandagi qiymati: **GL_REPEAT**.

GL_TEXTURE_WRAP_T faqat *t* koordinatasi uchun oldingi qiymatga o‘xshash.

GL_NEAREST rejimidan foydalanish teksturani to‘ldirish (ustiga qo‘yish) tezligini oshiradi, ammo bunda sifat pasayadi. Nega deganda **GL_LINEAR** ga qaraganda unda interpolyasiya ishlatilmaydi.

Teksturaning obyekt bajarilgan material bilan o‘zaro ta’sirini aniqlash uchun quyidagi buyruqlar ishlatiladi.

void **glTexEnv [i f]** (GLenum *target*, GLenum *pname*, GLtype *param*)

void **glTexEnv[i f]v** (GLenum *target*, GLenum *pname*, GLtype **params*)

target parametri **GL_TEXTURE_ENV** ga teng bo‘lishi kerak, *pname* sifatida ko‘p ishlatiladigan **GL_TEXTURE_ENV_MODE** faqat bitta qiymatini ko‘rib chiqamiz.

Tez-tez ishlatiladigan *param* parametri qiymatlari:

GL_MODULATE natijaviy rang ustida joylashgan rang nuqtasi va tekstura unga mos nuqtasini ko‘paytirish natijasida topiladi.

GL_REPLACE natijaviy rang sifatida teksturadagi rang nuqtasi ishlatiladi.

Quyidagi dastur, teksturani yaratishga bo‘lgan umumiy yondashuvni namoyon etadi:

```
/* bizga zarur bo‘lgan teksturalar soni */
#define NUM_TEXTURES 10
/* tekstura identifikatorlari */
int TextureIDs[NUM_TEXTURES];
/* teksturalar ko‘rinishi */
AUX_RGBImageRec *pImage;
…
/* 1) tekstura identifikatorlari olish */
glGenTextures(NUM_TEXTURES,TextureIDs);
/* 2) parametrlarni modifikasiyalash uchun teksturani tanlash */
glBindTexture(TextureIDs[i]); /* 0<=i<NUM_TEXTURES*/
/* 3) teksturani yuklaymiz. Tekstura o‘lchami – 2 bosqich*/
pImage=dibImageLoad("texture.bmp");
if (Texture!=NULL)
{
 /* 4) teksturani OpenGL ga uzatamiz va parametrlarini beramiz */
 /* bayt bo‘yicha to‘g‘rilaymiz */
 glPixelStorei(GL_UNPACK_ALIGNMENT,1);
```

```
gluBuildMipmaps(GL_TEXTURE_2D,GL_RGB, pImage->sizeX,
pImage->sizeY, GL_RGB, GL_UNSIGNED_BYTE,
pImage->data);
glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,
(float)GL_LINEAR);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
(float)GL_LINEAR);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
(float)GL_REPEAT);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
(float)GL_REPEAT);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
(float)GL_REPLACE);
/* 5) dastlabki tasvirni o‘chiramiz.*/
free(Texture);
}else Error();
```

Teksturali koordinata.

Teksturani obyekt ustiga qo‘shish uchun obyekt yuzasidagi nuqta bilan teksturaning nuqtasi o‘rtasidagi moslikni aniqlash kerak. Bu moslikni ikki usul bilan berish mumkin: maxsus funksiyaning tasviriy parametrlarni berib, har bir uchlar uchun alohida yoki barcha uchlar uchun barobar.

Birinchi usul buyruqlar yordamida qo‘llaniladi:

void **glTexCoord [1 2 3 4][s i f d]** (type *coord*)

void **glTexCoord[1 2 3 4][s i f d]v** (type **coord*)

Hozirgi vaqtda ko‘pincha teksturalar koordinatalarini beruvchi **glTexCoord2*(type s, type t)** ko‘rinishdagi buyruqlari ishlatiladi. Teksturaning hozirgi paytdagi koordinatalar tushunchasi hozirdagi norma tushunchasiga o‘xshash bo‘lib, uchlar atributi hisoblanadi. Ammo hatto kub uchun teksturaning mos

koordinatalarini topish yetarlicha murakkab mashg‘ulot hisoblanadi, shuning uchun GLU kutubxonasida doira, silindr va disk singari primitivlarni qurish buyruqlari, shuningdek ularga teksturani qo‘yish ko‘zda tutilgan. Buning uchun quyidagi

void **gluQuadricTexture** (GLUquadricObj* *quadObject*, GLboolean *textureCoords*)

buyruqni **GL_TRUE** ga teng bo‘lgan *textureCoords* parametri bilan chaqirish yetarli, va shunda joriy tekstura primitivlar ustiga avtomatik qo‘yiladi.

Ikkinchi usul buyruqlar yordamida qo‘llaniladi.

void **glTexGen [i f d]** (GLenum *coord*, GLenum *pname*, GLtype *param*)

void **glTexGen[i f d]v** (GLenum *coord*, GLenum *pname*,
const GLtype **params*)

Coord parametri qaysi koordinataga formula berilishini belgilaydi va **GL_S, GL_T** qiymatini o‘zlashtirishi mumkin; *pname* keyingi qiymatlardan biriga teng bo‘lishi mumkin:

GL_TEXTURE_GEN_MODE teksturani olish uchun funksiyani belgilaydi.

Bu holatda *param* argumenti qiymatlarni qabul qiladi:

GL_OBJECT_LINEAR tekstura koordinatalarining mos qiymati tekislikkacha bo‘lgan masofani, *pname* qiymati yordamida beriluvchi **GL_OBJECT_PLANE** belgilaydi. Formula quyidagi ko‘rinishda bo‘ladi: $g=x*xp+y*yp+z*zp+w*wp$, *g*–tekstura koordinatasi (*s* yoki *p*), *x*, *y*, *z*, *w* – nuqtalar koordinatasi. *xp*, *yp*, *zp*, *wp* – tekislik tenglamasi koeffisientlari. Formulada obyekt kordinatalari ishlatiladi.

GL_EYE_LINEAR oldingi qiymatga o‘xshash, faqat formulada koordinatalar ishlatiladi, ya’ni obyektning tekstura koordinatalari ushbu holatda mazkur obyektning joylashishiga bog‘liq bo‘ladi.

GL_SPHERE_MAP obyektning sirtidan aks etish imkonini beradi. Tekstura obyekt atrofida xuddi “aylanayotganga” o‘xshaydi. Ushbu usul uchun koordinatalar turi ishlatiladi va kerakli normalar beriladi.

GL_OBJECT_PLANE tekislik berishga imkon beradi, ungacha bo‘lgan masofani inersiya yordamida ishlatiladi, agar **GL_OBJECT_LINEAR** tartibi o‘rnatilgan bo‘lsa. Bu vaziyatda *params* parametri tekislik tenglamasining to‘rtta koeffisienti massivga ko‘rsatgich hisoblanadi.

GL_EYE_PLANE Oldingi qiymatga o‘xshash. **GL_EYE_LINEAR** rejimi uchun tekislikni berish imkoniyatini beradi.

Tekstura koordinatalarini tayinlashning avtomatik rejimini o‘rnatish uchun glEnable buyrug‘ini **GL_TEXTURE_GEN_S** yoki **GL_TEXTURE_GEN_P** parametrlari bilan chaqirish kerak.

Animatsiya va tekstura olinishidan foydalanilgan dastur D.3 ilovada keltirilgan.

Nazorat savollari:

1. Tekstura o‘zi nima va qanday maqsadda ishlatiladi?
2. Tekstura bilan ishlash ketma-ketligini tushuntiring?
3. Tekstura koordinatalari nima va ularni obyekt uchun qanday berish mumkin?
4. Agar tekstura o‘zida devorda osib qo‘yiladigan suratni aks ettiradigan vaziyatda material (GL_MODULATE, GL_REPLACE) bilan o‘zaro ta’sirlashishning qanday usulidan foydalanish zarur?
5. Obyektga tekstura berish buyrug’i qanday ifodalanadi?
6. OpenGL da tekstura koordinatalarini generasiyalash usullaridan sizga ma’lum bo‘lganlarini keltiring?
7. Teksturani batafsil tekshirish bosqichi (mip-mapping) nima uchun ishlatiladi?
8. Teksturani filtrasiyalash rejimi nima va OpenGL da ular qanday beriladi?

Tayanch iboralar: Tekstura, tekstura obrazi, tekstura parametrlari, detallashtirish, teksturalar qo‘yish, teksturali koordinata.

2.7. OpenGL yordamida ishlash usullari.

Piksellar ustida amallar. Tasvirlarni aralashtirish. Shaffoflik. To‘plovchi bufer. Niqob buferi. Rasterizatsiyalarni boshqarish. Pog‘onalilikni bartaraf qilish. Soyalarni qurish. Ob’ektlarni ko‘zgudagi aksi.

Piksellar ustida amallar. Tasvirlarni aralashtirish. Shaffoflik. To‘plovchi bufer. Niqob buferi. Rasterizatsiyalarni boshqarish.

Uchlar koordinatalarini o‘zgartirish bo‘yicha barcha amallarni olib borilgandan so‘ng rang va boshqalarni hisoblashda OpenGL *rasterizasiyalash* bosqichiga o‘tadi. Bu bosqichda teksturani qo‘yish, tuman effektlarini qo‘yish, barcha primitivlarni rasterizasiyalash ishlari amalga oshiriladi. Ushbu jarayonning natijasi har bir primitiv uchun kadr buferi sohasida egallangan, ushbu sohaning har bir pikseliga rang va chuqurlik qiymati yozilishi hisoblanadi.

OpenGL ushbu axborotni kadr buferida yangilangan malumotlarni yozish uchun ishlatadi. Buning uchun OpenGL nafaqat alohida qayta ishlov berish piksellar konveyeriga ega, shuningdek unda qo‘shimcha turli vazifalarni bajaradigan bufer ham mavjud. Bu dasturlovchiga eng quyi darajada vizual jarayonni nazorat qilish imkonini beradi.

OpenGL grafik kutubxonasi quyidagi buferlar ishini ta’minlaydi:

- rangning bir qancha buferlari
- chuqurlik buferi
- to‘plovchi bufer (akkumulyator)
- niqob buferi

Rang buferlari guruhida kadr buferi ham mavjud, lekin bunday bufer bir qancha bo‘lishi mumkin. Ikkitali buferizasiyalashdan foydalanganda ishchi (front) va fonli (back) bufer haqida gap boradi. Qoida sifatida fonli buferda dastur tasvirni hosil

qiladi, keyin uni birato‘la ishchi buferiga qo‘chiradi. Ekranda faqatgina rang buferlari haqidagi axborot namoyon bo‘lishi mumkin.

Chuqurlik buferi ko‘rinmas sirtlarni o‘chirish uchun ishlatiladi va ular bilan to‘g‘ridan to‘g‘ri ishlash kamdan-kam talab qilinadi.

To‘plovchi bufer har xil operatsiyalar uchun ishlatiladi. U xaqda to‘liq ma’lumot 3.3 bo‘limda keltirilgan.

Niqob buferi piksellar niqobini (trafaretlarni) shakllantirish uchun ishlatiladi. Umumiy massivlardan kerakli ekranga uzatish deb lozim topilgan piksellarni qirqib beradi.

Tasvirlarni aralashtirish. Shaffoflik

Har xil shaffof obyektlar - oyna, shaffof idishlar va boshqalar ko‘pincha amaliyotda (hayotda) tez-tez uchraydi, shuning uchun interfaol grafikada ham shunday obyektlarni tashkil qilish kerak. OpenGL dasturchiga yarim shaffof obyektlar bilan ishlash mexanizmini taqdim etadi va buning qisqacha ta’rifi ushbu bo‘limda beriladi.

Shaffoflik ranglarni aralashtirishning maxsus rejimi yordami bilan amalga oshiriladi (blending). Aralashtirish algoritmi allaqachon buferda saqlanayotgan mos piksellar rangi bilan kirish piksellari (ya’ni bufer kadrida joylashgan “nomzodlar”) deb ataluvchi ranglarni kombinatsiya qiladi. Aralashtirish uchun rangning to‘rtinchi qismi - alfa qism ishlatiladi, shuning uchun bu rejim alfa-aralashtirish deb ham ataladi. Dastur asosiy ranglar bilan qancha tez ishlasa, xuddi shunday alfa qism bilan katta tezlikda ishlaydi, ya’ni primitivning har bir uchi yoki har bir piksel uchun tezkorlik qiymatini beradi.

Rejim glEnable(GL_BLEND) buyrug‘i yordamida o‘rnatiladi.

Aralashtirish parametrlarini quyidagi buyruq yordamida aniqlash mumkin:

void **glBlendFunc** (enum *src*,enum *dst*)

src parametri pikselning dastlabki rangi k_1 koeffisientini qanday olish mumkinligini belgilasa, *dst* parametri bufer kadridagi rang uchun k_2 koeffisientni olish usulini beradi.

Natijaviy rangni olish uchun quyidagi formula ishlatiladi: res=s_{src}*k_1+c_{dst}*k_2, bu yerda s_{src} – dastlabki piksel rangi, c_{dst} – bufer kadri ichidagi piksel rangi (res, k_1, k_1, $s_{src,}$ c_{dst} – to‘rt qismli RGBA-vektori).

Tez-tez ishlatiladigan src va dst argumentlarining qiymatlarini keltiramiz.

GL_SRC_ALPHA	k=(A_s,A_s,A_s,A_s)
GL_SRC_ONE_MINUS_ALPHA	k=(1,1,1,1)-(A_s,A_s,A_s,A_s)
GL_DST_COLOR	k=(R_d,G_d,B_d)
GL_ONE_MINUS_DST_COLOR	k=(1,1,1,1)- (R_d,G_d,B_d,A_d)
GL_DST_ALPHA	k=(A_d,A_d,A_d,A_d)
GL_DST_ONE_MINUS_ALPHA	k=(1,1,1,1)-(A_d,A_d,A_d,A_d)
GL_SRC_COLOR	k=(R_s,G_s,B_s)
GL_ONE_MINUS_SRC_COLOR	k=(1,1,1,1)- (R_s,G_s,B_s,A_s)

Misol:

Shaffof obyektlarnining natijasini amalga oshirishni taqdim etamiz. Shaffoflik koeffisienti rangning alfa-qismi bilan beriladi. 1 – shaffof bo‘lmagan obyekt deylik, 0 mutlaqo shaffof, ya’ni ko‘rinmaydigan deb hisoblaymiz. Amalga oshirish uchun quyidagi kod xizmat qiladi:

```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA,GL_SRC_ONE_MINUS_ALPHA);
```

Masalan, yarim shaffof uchburchakni quyidagi ko‘rinishda berish mumkin:

```
glColor3f(1.0, 0.0, 0.0, 0.5);
glBegin(GL_TRIANGLES);
  glVertex3f(0.0, 0.0, 0.0);
  glVertex3f(1.0, 0.0, 0.0);
  glVertex3f(1.0, 1.0, 0.0);
glEnd();
```

Agar sahnada bir-birini yopib turadigan bir qancha shaffof obyektlar bo‘lsa, aniq natijani faqatgina quyidagi shartlar bajarilgan hollarda kafolatlash mumkin:

- Barcha shaffof obyektlar shaffof bo‘lmagan obyektlardan keyin chiqadi.
- Obyektlar shaffof bo‘lib chiqishida chuqurlik qisqarishi bo‘yicha tartiblangan bo‘lishi kerak. Bu degani, kuzatuvdan uzoqlashganlar birinchi bo‘lib chiqa boshlaydi.

OpenGL da buyruqlarga kelish tartibiga qarab ishlov beriladi, shu sababli keltirilgan talablarni amalga oshirish uchun tegishli tartibda glVertex*() buyrug‘ini chaqirish va kerakli joyga qo‘yish yetarli, lekin bu ham umumiy holda noto‘g‘ri.

To‘plovchi bufer.

To‘plovchi bufer (*accumulation buffer*) – bu OpenGL kushimcha buferlaridan biri. Unda vizuallashtirilgan tasvirni aniqlash mumkin, maxsus piksel operatsiyalarini ishlatgan xolda yig‘in buferi har xil maxsus samarali ishlatganda ishlatilishida keng qo‘llaniladi.

Tasvir quyidagi buyruqni tanlash orqali buferdan olinadi:

void **glReadBuffer** (enum *buf*)

buf argumenti o‘qish uchun buferni aniqlaydi. *buf* qiymati **GL_BACK**, **GL_FRONT** larga teng bo‘lib, o‘qish uchun mos buferlarni aniqlaydi. Ekranlashtirilmagan buferni **GL_BACK** bosh piksellar sifatida beradi. **GL_FRONT** – natijaning joriy tarkibiy oynasi. Buyruq qiymatga ega bo‘ladi, agarda dublikatlangan (ikkinchi nusxa) buferizasiyasi bo‘lsa. Aks holda faqat bitta bufer ishlatiladi, chiqarish oynaga mos (jiddiy aytganda, OpenGL ga qo‘shimcha bir to‘plam buferi bor. Ular sterio tasvirda ishlatiladi, lekin biz ularni xozir ko‘rib chiqmaymiz).

To‘plovchi bufer qo‘shimcha rang buferi hisoblanadi. U ko‘rinish natijalari uchun to‘g‘ridan to‘g‘ri ishlatilmaydi, lekin ular rang buferlaridan biriga natijadan so‘ng qo‘shiladi. Quyida keltirilgan, har xil amallarni qo‘llab, buferda tasvirni yig‘ish mumkin.

Keyin hosil bo‘lgan tasvirni to‘plovchi buferdan bitta rang buferiga quyidagi buyruq yordamida yozuv bo‘yicha o‘tkazish mumkin.

void **glDrawBuffer (**enum *buf)*

buf qiymati **glReadBuffer** buyrug‘idagi mos argument qiymatiga o‘xshash.

To‘plovchi bufer bilan barcha amallar quyidagi buyruq bilan nazorat qilinadi

void **glAccum** (enum *op*,GLfloat *value*)

op argumenti piksellar ustida amallarni belgilaydi va quyidagi qiymatlarni qabul qiladi:

GL_LOAD Piksel o‘qish uchun tanlangan buferdan olinadi. Uning qiymati *value* da ko‘paytiriladi va to‘plovchi buferga kiritiladi.

GL_ACCUM Oldingiga o‘xshash, ammo qiymatlar ko‘paytirilgandan keyin hosil bo‘lganlar buferda mavjud bo‘lganlar bilan qo‘shiladi.

GL_MULT Bu amal buferdagi har bir pikselni *value* ga ko‘paytiradi.

GL_ADD Oldingiga o‘xshash, faqat ko‘paytirish o‘rniga qo‘shish ishlatiladi.

GL_RETURN Tasvir to‘plovchi buferdan yozish uchun buferga ko‘chiriladi. Undan oldin har bir piksel *value* ga ko‘paytiriladi.

Aytish joizki to‘plovchi buferni ishlatish uchun glEnable buyrug‘ini chaqirish shart emas. Faqatgina buferning o‘zini initsializatsiyalash faollashtirish etarli.

Rasterizasiyalashdagi (bo‘laklash) xatoliklarni bartaraf qilish uchun to‘plovchi buferdan foydalanishga oid misol 3.8 bo‘limda keltirilgan.

Niqob buferi.

Kadr buferiga piksellarni chiqarayotganda ayrim hollarda ba’zi piksellarni chiqarish zaruriyati tug‘iladi, ya’ni tasvirga trafaret (niqob) qo‘yiladi. Buning uchun OpenGL niqob buferini taqdim etadi (stencil buffer). Niqob qo‘yishdan tashqari, bu bufer ancha qiziqarli imkoniyatlar namoyon etadi.

Pikselni kadr buferiga joylashtirishdan oldin, OpenGL ning vizuallashtirish mexanizmi berilgan qiymatlar va niqob buferdagi qiymatlar o‘rtasida solishishtirish amalini (testlash) bajarish imkonini beradi. Agar test o‘tkazilsa, piksel kadr buferida chiziladi.

Solishtirish mexanizmi juda ham moslashuvchan va u quyidagi buyruqlar tomonidan nazorat qilinadi:

void **glStencilFunc** (enum *func*, int *ref*, uint *mask*)

void **glStencilOp** (enum *sfail*, enum *dpfail*, enum *dppass*)

glStencilFunc buyrug‘ining *ref* argumenti solishtirish uchun qiymat beradi. U 0 dan $2^s - 1$ gacha bo‘lgan qiymatlarni qabul qiladi. *s* – niqob buferida nuqtadagi bitlar soni.

func argumenti yordamida solishtirish funksiyasi beriladi. U quyidagi qiymatlarni qabul qilishi mumkin:

GL_NEVER test hech qachon o‘tkazilmaydi, ya’ni xar doim *false* qaytadi.

GL_ALWAYS test har doim o‘tkaziladi.

GL_LESS, GL_LEQUAL, GL_EQUAL,
GL_GEQUAL, GL_GREATE, GL_NOTEQUAL test o‘tkaziladi, agar *ref* mos holda niqob buferdagi qiymatdan kam bo‘lsa, kam yoki teng, teng, ko‘p, ko‘p yoki teng, yoki teng emas.

Mask argumenti qiymatlar uchun niqob tayyorlaydi, ya’ni bu test uchun natijada quyidagi formulaga ega bo‘lamiz: ((*ref* AND *mask*) *op* (svalue AND *mask*)).

glStencilOp buyrug‘i niqob buferidagi piksellar ustidan harakat uchun berilgan testning ijobiy yoki salbiy natijasi ko‘rsatish uchun ishlatiladi.

sfail argumenti testning natijasi salbiy bo‘lgan holatda harakatni belgilaydi va quyidagi qiymatlarni qabul qilishi mumkin:

GL_KEEP, GL_ZERO, GL_REPLACE,
GL_INCR, GL_DECR, GL_INVERT niqob buferidagi qiymatlarni mos holda saqlaydi, uni 0 ga tushiradi, berilgan qiymat (*ref*) ga o‘zgartiradi, ko‘paytiradi, kamaytiradi yoki bitlarga bo‘ladi.

dpfail argumenti z-buferi chuqurligida testning natijasi salbiy bo‘lgan hollarda harakatni belgilaydi, *dppass* argumenti esa ushbu testning natijasi ijobiy bo‘lgan hollarda harakatni belgilaydi. Argumentlar *sfail* argumentidagi qiymatlarni qabul qiladi.

Niqoblashni yoqish uchun glEnable(GL_STENCIL_TEST) buyrug‘ini bajarish ishlatish kerak. Niqob buferi soya tushishi, akslanish, bir rasmdan boshqasiga rasmga tekis o‘tish va boshqalar singari maxsus effektlarni yaratishda foydalaniladi.

Rasterizasiyalarni boshqarish

Primitivlarni rasterizasiyalashni bajarish usullarini glHint (target, mode) buyrug‘i bilan qisman tartibga solish mumkin, bu yerda *target* – nazorat qilinayotgan harakat ko‘rinishi bo‘lib, quyidagi qiymatlardan birini qabul qiladi.

GL_FOG_HINT – tuman qo‘yishda hisoblashning aniqligi. Hisoblash piksellar (yuqori aniqlikda) bo‘yicha bajarilishi mumkin yoki faqat uchlarda. Agar OpenGL piksellar bo‘yicha hisoblashni amalga oshirilishini ta’minlamasa, unda faqat uchlar bo‘yicha hisoblash bajarilishi amalga oshiriladi.

GL_LINE_SMOOTH_HINT – sifatlarni to‘g‘ri boshqarish. *mode* qiymatida **GL_NICEST** teng bo‘lib, to‘g‘rilikda ko‘p sonli piksellar hisobiga to‘g‘rilik pog‘onasi kamayadi.

GL_PERSPECTIVE_CORRECTION_HINT – teksturalar qo‘yish va ranglarni hisoblashda koordinatalar interpolyasiyasining aniqligi. Agar OpenGL **GL_NICEST** rejimini quvvatlamasa, unda koordinatalarning chiziqli interpolyasiyasi amalga oshadi.

GL_POINT_SMOOTH_HINT – nuqtalar sifatini boshqarish**. Mode** parametr qiymatida **GL_NICEST** ga teng bo‘lganda, nuqtalar doirada sifatida chiziladi.

GL_POLYGON_SMOOTH_HINT – ko‘pburchak tomonlarini chiqarish sifati boshqarish.

mode parametri quyidagi shaklda interpolyasiyalanadi:

GL_FASTEST – ancha tez algoritm ishlatiladi.

GL_NICEST – eng yaxshi sifatni ta’minlaydigan algoritm ishlatiladi.

GL_DONT_CARE - algoritmni tanlash amalga oshirishga bog‘liq.

Shuni ta’kidlash lozimki, dasturchi glHint() buyrug‘i bilan primitivlarni rasterizasiyalashning u yoki bu jihatlariga nisbatan faqatgina o‘zining hoxishidagini belgilashi mumkin. OpenGL ni aniq amalga oshirilishi ushbu o‘rnatishni e’tiborsiz qoldirishga haqli.

Ahamiyat bering, glHint() buyrug‘ini glBegin()/glEnd() operatorli qavslari o‘rtasida chaqirish mumkin emas.

Nazorat savollari:

1. OpenGL da qanday tasvir buferlaridan foydalaniladi?
2. glBlendFunc buyrug‘i nima uchun foydlaniladi?
3. OpenGL da tasvir buferlaridan nima uchun foydalaniladi?
4. Shaffof obyektlarga nimalar kiradi?
5. To‘plovchi bufer nima uchun ishlatiladi? U bilan bog‘liq bo‘lgan misol keltiring.
6. OpenGL da natijaviy tasvirga qanday qilib niqobni qo‘yish mumkin?
7. glHint() buyrug‘i nima uchun qo‘llaniladi, tushuntiring.
8. glHint(GL_FOG_HINT, GL_DONT_CARE) buyrug‘ini bajarishning qanday effektlari bor?

Tayanch iboralar: Rasterizasiyalash, kadr buferi, piksellar konveyeri, rang buferi, chuqurlik buferi, to‘plovchi bufer, niqob buferi, shaffoflik.

OpenGL da ishlash usullari

Pog‘onalilikni bartaraf qilish. Soyalarni qurish. Ob’ektlarni ko‘zgudagi aksi.

Ushbu paragrafda biz OpenGL yordamida, to‘g‘ridan-to‘g‘ri quvvatlanishi kutubxona standartida mavjud bo‘lmagan qiziqarli vizual effektlarni yaratishni ko‘rib chiqamiz.

Pog‘onalilikni bartaraf qilish.

Pog‘onalilikni (*antialiasing*) bartaraf qilish masalasidan boshlaymiz. Pog‘onalilik effekti (*aliasing*) buferning chekli (qoida sifatida, katta bo‘lmagan) ruxsati hisobiga kadr buferida primitivlarni rasterizasiyalashdagi xatoliklar natijasida yuzaga keladi. Ushbu muammoga hal qilish bo‘yicha bir necha yondashushlar mavjud. Masalan, olingan tasvirga filtrlashni qo‘llash mumkin. Shuningdek ushbu effektni, har bir primitiv ko‘rinishini tekislash, rasterizasiyalash bosqichida bartaraf

qilish mumkin. Bu yerda biz barcha sahnani butun holi uchun o‘xshash umumiy dalillar bilan bartaraf qilish imkonini beruvchi usulni ko‘rib chiqamiz.

Har bir kadr uchun sahnani bir necha marta chizish zarur, har bir o‘tishda kamera boshlang‘ich holatiga nisbatan ozroq suriladi. Kameralarning joylashishi, masalan, doira hosil qilish mumkin. Agarda kamerani surilishi nisbatan kam bo‘lsa, unda diskretizasiya xatolari har xil bo‘lishi mumkin, va, olingan tasvirni o‘rtacha ko‘rinishga olib kelguncha biz tekis tasvirni olamiz.

Kuzatuvchi joyini ko‘chirish osonroq, lekin undan oldin ko‘chirish joyini shunday hisoblash kerakki, ekranga uzatilgan koordinatalar qiymati, piksellar yarmidan oshmasligi lozim.

Barcha hosil bo‘lgan tasvirlarni to‘plovchi buferda 1/*n* koeffisienti bilan saqlaymiz, bu yerda *n*-har bir kadr uchun o‘tish yo‘laklari soni. Bunday yo‘laklar qancha ko‘p bo‘lsa, unumdorlik shuncha pasayadi, lekin natija yaxshilanadi.

```
for(i=0; i<samples_count;++i)
/* odatda samples_count  5 dan 10 gacha bo‘lgan chegarada yotadi */
{
  ShiftCamera(i); /* kamerani ko‘chiramiz */
  RenderScene();
  if (i==0)
   /* birinchi iteratsiyada tasvirni yuklaymiz */
   glAccum(GL_LOAD,1/(float)samples_count);
  else
   /* avval mavjud bo‘lganlarga qo‘shamiz */
   glAccum(GL_ACCUM,1/(float)samples_count);
}
/*Olingan natijani orqaga boshlang‘ich buferga yozamiz */
glAccum(GL_RETURN,1.0);
```

Shuni aytish joizki, barcha sahna uchun pog‘onalilikni bartaraf qilish, qoida sifatida vizuallashtirish unumdorligining pasayishiga jiddiy ta’siri bilan bog‘liq.

Negaki, har bir sahna bir necha marotaba chiziladi. Zamonaviy tezlashtiruvchilar odatda tasvir resamplingi deb ataluvchiga asoslangan boshqa usullarni apparatli amalga oshiradi.

Soyalarni qurish.

OpenGL da bazaviy bo‘yruqlar darajasida soyalarni ko‘rishning ichki ta’minlanishi mavjud emas. Buning ahamiyatli bosqichi shundaki, ularni qurishda ko‘pgina algoritmlarning mavjudligi OpenGL funksiyalari yordamida amalga oshiriladi. Soyalarning borligi uch o‘lchovli tasvirning realligiga kuchli ta’sir o‘tkazadi, shuning uchun ularni qurishga bo‘lgan yondashuvlardan birini ko‘rib chiqamiz.

Soyalarni qurish uchun mo‘ljallangan ko‘pgina algoritmlar perspektiv proeksiyalashning turlangan tamoyillarini ishlatadi. Bu yerda eng oddiy usullardan biri ko‘rib chiqiladi. Uning yordamida uch o‘lchovli obyektni tekislikka tushirib soya hosil qilish mumkin.

Umumiy yondashuv shunday: obyektning barcha nuqtalari uchun ularning proeksiyasi, ba’zi berilgan tekislikdagi yorug‘lik manbasida joylashgan ushbu nuqta va nuqtani bog‘lovchi vektorga parallel deb topiladi. Shundan so‘ng berilgan tekislikda butunligicha yotuvchi yangi obyektga ega bo‘lamiz. Ushbu obyekt dastlabki soya hisoblanadi.

Ushbu usulning matematik asosini ko‘rib chiqamiz:

Berilgan bo‘lsin:

P – uch o‘lchovli fazoda soyani tushiruvchi nuqta.

L – ushbu nuqtaga yorituvchi yorug‘lik manbasi holati.

S=a(**L**-**P**)-**P** - nuqta, unga P nuqtasi o‘z soyasini tashlaydi, bu yerda a – parametr.

Taxmin qilish mumkinki, soya $z = 0$ tekislikka tushadi. Bunday vaziyatda $a=z_p/(z_l-z_p)$. Demak,

$x_s = (x_p z_l - z_l z_p) / (z_l - z_p)$,

$y_s = (y_p z_l - y_l z_p) / (z_l - z_p)$,

$z_s = 0$.

Bir jinsli koordinatalarni kiritamiz:

$x_s=x_s'/w_s'$

$y_s=y_s'/w_s'$

$z_s=0$

$w_s'=z_l-z_p$

Bu yerda S koordinatalari matritsani quyidagi ko‘rinishda ko‘paytirish orqali hosil qilinishi mumkin:

$$[x_s' \quad y_s' \quad 0 \quad w_s'] = [x_s \quad y_s \quad z_s \quad 1] \begin{bmatrix} z_l & 0 & 0 & 0 \\ 0 & z_l & 0 & 0 \\ -x_l & -y_l & 0 & -1 \\ 0 & 0 & 0 & z_l \end{bmatrix}$$

Algoritm ixtiyoriy tekislikka tushadigan soyani hisoblashi uchun, bir jinsli koordinatalarda berilgan **S** va **P** o‘rtasidagi chiziqda ixtiyoriy nuqtani ko‘rib chiqamiz:

$$G = \begin{bmatrix} x_n \\ y_n \\ z_n \\ d \end{bmatrix}$$

$a\mathbf{P}+b\mathbf{L}$, a va b – skalyar parametrlar.

Quyidagi matritsa normal koordinatalari orqali tekislikni beradi:

Ushbu R nuqta orqali yorug‘lik manbasidan o‘tkaziluvchi nur (nuqta), G tekislikni kesib o‘tadi, quyidagi tenglamani qanoatlantiruvchi a va b parametrlar bilan belgilanadi:

$(a\mathbf{P}+b\mathbf{L})\mathbf{G} = 0$

Bu yerdan quyidagiga ega bo‘lamiz: $a(\mathbf{PG}) + b(\mathbf{LG}) = 0$.

Ushbu tenglamani $a = (\mathbf{LG})$, $b = -(\mathbf{PG})$ qanoatlantiradi.

Demak topilgan nuqtaning koordinatalari **S=(LG)P-(PG)L.** Matritsa ko‘paytmasining assosiativligidan foydalanib, quyidagiga ega bo‘lamiz:

S = **P**[(**LG**)**I** - **GL**], bu yerda **I –** birlik matritsa.

(**LG**)**I** - **GL** matritsasi ixtiyoriy tekislikda soyaga ega bo‘lish uchun foydalaniladi.

OpenGL yordamida ushbu usulni amalga oshirishning ba’zi bir jihatlarini ko‘rib chiqamiz.

Taxmin qilish mumkinki, floorShadow matritsasi oldin biz tomonimizdan (**LG)I - GL** formulasidan olingan edi. Quyidagi kod yordamida berilgan tekislik uchun soyani qurish mumkin:

```
/* ranglarni aralashtirishdan  (blending) foydalanib soyani yarim shaffof ko‘rinishga olib kelamiz*/
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
glDisable(GL_LIGHTING);
glColor4f(0.0, 0.0, 0.0, 0.5);
glPushMatrix();
  /* soyani proeksiyalaymiz  */
  glMultMatrixf((GLfloat *) floorShadow);
  /* proeksiyada sahnani vizuallashtiramiz */
  RenderGeometry();
glPopMatrix();
glEnable(GL_LIGHTING);
glDisable(GL_BLEND);
/* Odatdagi rejimda sahnani vizuallashtiramiz */
RenderGeometry();
```

floorShadow matritsasi quyidagi funksiya yordamida (*) tenglamasidan olinishi mumkin:

```
/* parametrlar:     plane – tekislik tenglamasi koeffisientlari
                    lightpos – yorug‘lik manbasi koordinatlari
  qaytaradi:              matrix – natijaviy matritsa    */
void shadowmatrix(GLfloat matrix[4][4], GLfloat plane[4],
            GLfloat lightpos[4])
{
```

```
GLfloat dot;

 dot = plane[0] * lightpos[0] +plane[1] * lightpos[1] +plane[2] * lightpos[2]
+plane[3] * lightpos[3];

matrix[0][0] = dot - lightpos[0] * plane[0];
matrix[1][0] = 0.f - lightpos[0] * plane[1];
matrix[2][0] = 0.f - lightpos[0] * plane[2];
matrix[3][0] = 0.f - lightpos[0] * plane[3];

matrix[0][1] = 0.f - lightpos[1] * plane[0];
matrix[1][1] = dot - lightpos[1] * plane[1];
matrix[2][1] = 0.f - lightpos[1] * plane[2];
matrix[3][1] = 0.f - lightpos[1] * plane[3];

matrix[0][2] = 0.f - lightpos[2] * plane[0];
matrix[1][2] = 0.f - lightpos[2] * plane[1];
matrix[2][2] = dot - lightpos[2] * plane[2];
matrix[3][2] = 0.f - lightpos[2] * plane[3];

matrix[0][3] = 0.f - lightpos[3] * plane[0];
matrix[1][3] = 0.f - lightpos[3] * plane[1];
matrix[2][3] = 0.f - lightpos[3] * plane[2];
matrix[3][3] = dot - lightpos[3] * plane[3];
}
```

Ko‘rishimiz mumkinki, shunday ko‘rinishda qurilgan soyalar ham bir qancha kamchiliklarga ega.

- Ta’riflangan algoritm tekislik cheksizligini ko‘zda tutadi, va soyani chegara bo‘yicha kesmaydi. Masalan, agar qandaydir obyekt o‘z soyasini stolga

tushursa, u chegara bo‘yicha kesilmaydi, va buning ustiga, stolning yon sirtiga “buriladi”.

- Ayrim joylarda soya “Ikkilik” aralashmasi” (reblending) effekti bo‘lib kuzatilishi mumkin, ya’ni uchburchaklar proeksiyasi bir birini qoplaydigan joyda qora dog‘lar kuzatilishi mumkin.
- Tekisliklar soni ortgani sari algoritmlar ham murakkablashib boradi, nega deganda, har bir tekislik uchun alohida sahna quriladi, hattoki soyani chegarada kesilish muammosi yechilsa ham.
- Ko‘pincha soyalar noaniq chegaralarga ega, berilgan algoritmda esa u aniq chegaraga ega. Undan qisman qochish esa, bir necha yorug‘lik soyasini hisoblash, bir joyda va keyinchalik natijasini aralashtirganda kelib chiqadi.

Birinchi va ikkinchi muammonining echimi mavjud. Uning uchun niqob buferi ishlatiladi (3.7. paragrafga qarang).

Demak, kesish vazifasining geometrik natijasi (ushbu holatda, soya) ba’zi ixtiyoriy soha chegarasi bo‘yicha va “Ikkilik aralashmasi” dan qochish hisoblanadi. Niqob buferi yordamida echimning umumiy algoritmi quyidagicha:

1. Niqob buferini 0 qiymati bilan tozalaymiz.

2. Kesilgan doirani tasvirlaymiz, niqob buferiga 1 qiymatini o‘rnatamiz.

3. Niqob buferida 1 joylashgan bo‘lsa o‘sha atrofda soyani yasaymiz. Agar test muvaffaqiyatli o‘tsa, bu sohada 2 qiymatini o‘rnatamiz.

Endi bu bosqichlarni batafsil ko‘rib chiqamiz.

1.

```
/* niqob buferini tozalaymiz */
glClearStencil(0x0);
/* test qo‘shamiz*/
glEnable(GL_STENCIL_TEST);
```

2.

```
/* shart doim bajariladi va buferdagi qiymat 1 ga teng bo‘ladi */
```

```
glStencilFunc (GL_ALWAYS, 0x1, 0xffffffff);
/* har qanday holatda niqob buferidagi qiymatni o‘zgartiramiz */
glStencilOp (GL_REPLACE, GL_REPLACE, GL_REPLACE);
/* keyinchalik soya kesilishi bo‘yicha geometriyani chiqaramiz */
RenderPlane();
```

3.

```
/* agar niqob buferidagi qiymat 1 ga teng bo‘lsa test rostlikni beradi va shart
bajarildi */
glStencilFunc (GL_EQUAL, 0x1, 0xffffffff);
/* agarda soya allaqachon aniqlangan bo‘lsa, buferdagi qiymat 2 ga teng */
glStencilOp (GL_KEEP, GL_KEEP, GL_INCR);
/* soyani chiqaramiz  */
RenderShadow();
```

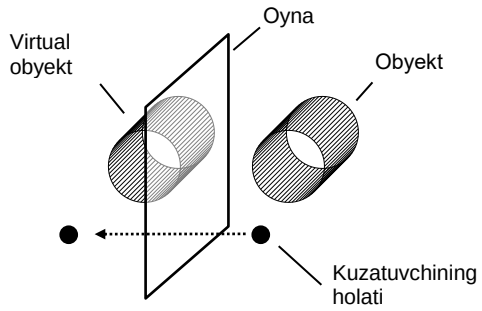
Ochiq aytganda, niqoblashtirishni ishlatganda ham z-bufer bilan bog‘liq bo‘lgan ayrim muammolar qoladi. Qisman, ayrim soyalar doirasi ko‘rinmaydigan bo‘lib qolishi mumkin. Bu muammoning echimini topish uchun tekislik ta’riflanadigan tenglamani o‘zgartirish yordamida soyani tekislik ustida ko‘tarish mumkin. Boshqa usullarni ta’riflash ushbu qo‘llanma doirasidan tashqaridir.

Obyektning ko‘zgudagi aksi.

Bu bo‘limda biz tekis obyektlarni akslantirish algoritmlarini ko‘rib chiqamiz. Bunday aks qaytarishlar tasvir ko‘rinishida katta axamiyatga ega va ularni oson amalga oshirish mumkin.

Algoritm kuzgu bilan 2 ta to‘liq sahnani jamlaydi: “haqiqiy” va “virtual” ko‘zgu orqasida joylashgan. Demak, aksni chizish jarayoni ikki qismdan iborat: 1) Har daqiqa sahnaning vizualligi va 2) Virtuallarni qurilish va vizuallashi har bir “haqiqiy” obyekt uchun uning ikkinchisi – aksi ko‘riladi, ya’ni kuzatuvchi ko‘zguda ko‘riladi.

Misol uchun devorda osilib turgan ko‘zguli xonani tasavvur qilamiz. Xona va obyektlar shunaqa ko‘zgu ko‘rinishida, huddi, agar ko‘zgu shisha bo‘lsa ko‘rinadigan, uning orqasida ham huddi shunaqa ko‘zgu bor. Ular tekislikka nisbatan simmetrik – aksi berilgan, ko‘zgu tepasidan o‘tkazilgan.



2.9-rasm. Ko‘zgudagi tasvir.

Tekislik aksini yaratish algoritmning soddalashtirilgan varianti quyidagi qadamlardan iborat:

1. Sahnani har doimgidek chizamiz, ammo ko‘zgu obyektlarisiz.
2. Niqob buferi ishlatgan xolda, ekranga ko‘zgu proeksiyasining keyingi natijalarini cheklaymiz.
3. Ko‘zgu tekisligiga nisbatan akslangan sahnani vizuallashtiramiz. Shunda niqob buferi ko‘zgu obyekti proeksiyasi shaklining natijasini cheklash imkonini beradi.

Bu harakat ketma-ketligi akslantirishning ishonchli samarasini olish imkonini beradi.

Bosqichlarni batafsil kurib chiqamiz.

Dastlab sahnani odatdagidek chizish kerak. Bu bosqichga batafsil to‘htalmaymiz. Chizishdan oldin bevosita OpenGL buferi tozalashda, niqob buferini tozalashni unutmaslik lozim:

```
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT|
        GL_STENCIL_BUFFER_BIT);
```

Sahnani vizuallashtirish vaqtida obyektlarni chizmagan ma’qul, keyinchalik ular ko‘zguda tasvirdek bo‘lib qoladi.

Ikkinchi bosqichda ekranga ko‘zgudagi obyekt proeksiyasining keyingi natijalarini cheklash zarur.

Buning uchun niqob buferini to‘g‘rilaymiz va ko‘zgu yasaymiz.

```
glEnable(GL_STENCIL_TEST);
/* shart doim bajariladi va buferdagi qiymat 1 ga teng bo‘ladi */
glStencilFunc(GL_ALWAYS, 1, 0);
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);

RenderMirrorObject();
```

Natijada biz quyidagiga ega bo‘ldik:

- Kadr buferida ko‘zgu sohasidan tashqarida bo‘lgan aniq chizilgan sahna;
- Ko‘zgu sohasida (biz akslantirishni ko‘rmoqchi bo‘lgan joyda) niqob buferi qiymati 1 ga teng.

Uchinchi bosqichda ko‘zgudagi obyekt tekisligiga nisbatan aks etuvchi sahnani chizish zarur.

Dastlab akslantirish matritsasini to‘g‘rilab olamiz. Akslantirish matritsasi tekislikka nisbatan butun geometriyani ko‘zguda akslantirish lozim, ko‘zgu – aksi yotgan obyektni ko‘rsatishi kerak. Uni topish mumkin, masalan, quyidagi funksiyalar orqali (mashq sifatida mustaqil ushbu matritsani olishga harakat qiling).

```
void
reflectionmatrix(GLfloat reflection_matrix[4][4],
                 GLfloat plane_point[3],
                 Glfloat plane_normal[3])
{
  GLfloat* p;
```

```
    GLfloat* v;
float pv;

    GLfloat* p = (Glfloat*)plane_point;
    Glfloat* v = (Glfloat*)plane_normal;
    float pv = p[0]*v[0]+p[1]*v[1]+p[2]*v[2];

    reflection_matrix[0][0] = 1 - 2 * v[0] * v[0];
    reflection_matrix[1][0] = - 2 * v[0] * v[1];
    reflection_matrix[2][0] = - 2 * v[0] * v[2];
    reflection_matrix[3][0] = 2 * pv * v[0];

    reflection_matrix[0][1] = - 2 * v[0] * v[1];
    reflection_matrix[1][1] = 1- 2 * v[1] * v[1];
    reflection_matrix[2][1] = - 2 * v[1] * v[2];
    reflection_matrix[3][1] = 2 * pv * v[1];

    reflection_matrix[0][2] = - 2 * v[0] * v[2];
    reflection_matrix[1][2] = - 2 * v[1] * v[2];
    reflection_matrix[2][2] = 1 - 2 * v[2] * v[2];
    reflection_matrix[3][2] = 2 * pv * v[2];

    reflection_matrix[0][3] = 0;
    reflection_matrix[1][3] = 0;
    reflection_matrix[2][3] = 0;
    reflection_matrix[3][3] = 1;
    }
```

va sahnani yana bir marta chizamiz (ko‘zgu obyektlarisiz)

```
    glPushMatrix();
```

```
glMultMatrixf((float *)reflection_matrix);
RenderScene();
GlPopMatrix();
```

Nihoyat, niqoblashni o‘chiramiz

```
glDisable(GL_STENCIL_TEST);
```

Shundan so‘ng yana bir marta ko‘zgudagi obyektni chiqarish mumkin, masalan alfa-aralash bilan ko‘zguni xiralashtirish effektini yaratish uchun va boshqalar.

E’tibor bering, keltirilgan metod aniq ishlaydi, agar sahnada ko‘zgudagi obyektdan boshqa obyekt bo‘lmasa. Shuning uchun ushbu algoritmning harakat ketma-ketligi bilan farqlanuvchi va geometriyada turlicha cheklovga ega bo‘lgan har xil turlari mavjud.

Nazorat savollari:

1. Tasvirning pog‘onalilik effekti natijasida nima yuz beradi?
2. Pog‘onalilikni bartaraf etish algoritmini misol orqali tushuntiring?
3. Nima uchun OpenGL da soyalarni qurish ichki madadlanmaydi?
4. Soyalarni qurishda uchraydigan kamchiliklarni keltiring?
5. Ko‘zgudagi obyektlarni vizuallashtirishning taklif etilgan metodini qisqacha yozing.
6. Ko‘zguda sahnadagi boshqa obyektlar kuzatilsa u nima sababdan ishlamaydi?
7. Bunday holda nima tasvirlanadi?
8. Ushbu cheklanishdan qanday chiqish haqida o‘ylab ko‘ring?

Tayanch iboralar: Vizual effekt, pog‘onalilik, pog‘onalilik effekti, soyalar, ko‘zgudagi aks.

2.8. OpenGL yordamida dasturlarni optimallashtirish.

Ilovani tashkil etish. Yuqori darajali optimallash. Past darajali optimallash. OpenGL da chaqirishlarni optimallashtirish. OpenGL da ma'lumotlarni uzatish. O'zgartirish. Yorug'lik. Teksturalash. Buferlarni tozalash

Ilovani tashkil etish.

Bir qarashda OpenGL ga asoslangan ilovaning unumdorligi, birinchi navbatda OpenGL ning o'z kutubxonasini amalga oshirish unumdorligini belgilanishi bo'lib ko'rinishi mumkin. Bu to'g'ri, ammo umumiy ilovani tashkil etish juda ham muhim.

Yuqori darajali optimallash.

Odatda OpenGL ostida ishlovchi dasturlardan interaktiv tezlikdagi yuqori sifatni vizuallashtirish talab etiladi. Lekin, qoida sifatida, u yoki boshqasiga birdan erishishning imkoni yo'q. Shunday ekan, sifat va unumdorlik o'tasida o'zaro kelishuvni qidirish lozim. Ko'pgina turlicha yondashuvlar mavjud bo'lsada, ammo ularning batafsil tavsifi ushbu qo'llanma doirasidan tashqarida hisoblanadi. Faqatgina bir nechta misollarni keltiramiz:

- Animatsiya vaqtida past sifatli sahna geometriyasini tasvirlash mumkin, to'xtash vaqtida esa uni eng yuqori sifatda ko'rsatish mumkin. Interaktiv burish vaqtida (masalan, sichqoncha tugmasini bosishda) primitivlar sonini qichqartirish bilan modelni vizuallashtirish. Statistik tasvirlarni chizishda modelni to'liq tasvirlash.
- Obyektlar kuzatuvchidan uzoqda joylashadi va quyi murakkablikdagi model ko'rinishida taqdim etilishi mumkin. Bu esa OpenGL konveyerining barcha pog'onalari vazifalarini sezilarli darajada pasaytiradi. Kuzatuv maydonidan to'liqligicha tashqarida yotgan obyektlar, ko'rish piramidasida ularning oddiy hajmlari (sfera yoki kub) chegaralanishining tushishini nazorat qilish yordamida OpenGL konvereyiga uzatishsiz samarali kesilishi mumkin.
- Animatsiya vaqtida soxta ishlov berish, suzuvchi zalivka, teksturalarni o'chirib qo'yish mumkin. Xuddi shunday bularning barchasini statistik tasvirlarni

ko‘rsatishda yoqish mumkin. Ushbu yondashuv OpenGL ning apparatli quvvatlanmagan tizimi uchun ayniqsa samaralidir.

Past darajali optimallash.

OpenGL yordamida tasvirlanadigan obyektlar, ayrim ma’lumotlar tuzilmasida saqlanadi. Bunday tuzilmalardan bir turi boshqasiga nisbatan foydalanish ancha samarali va vizuallashtirish tezligini belgilaydi.

OpenGL konveyeriga tez va samarali uzatish mumkin bo‘lgan, ma’lumotlar tuzilmasidan foydalanilishi maqsadga muvofiq. Masalan, agarda biz uchburchak massivini tasvirlamoqchi bo‘lsak, ushbu massivga ko‘rsatkichlardan foydalanish uni qismlar bo‘yicha OpenGL ga uzatishga qaraganda ancha samaraliroq.

Misol:

Tasavvur qilamiz, biz yer xaritasini chizishni amalga oshiruvchi ilovani yozayapmiz. Ma’lumotlar bazasidagi qismlardan biri – shaharlarning nomi, uzunligi va kengligi bo‘yicha ro‘yxati. Ma’lumotlarning mos tuzilmasi quyidagicha bo‘lishi mumkin:

```
struct city
{
  float latitute, longitude; /* shaharning holati */
  char *name;              /* nomi */
  int large_flag;      /* 0 = kichik, 1 = katta */
};
```

Shaharlar ro‘yxati shunday massiv tuzilmasida saqlanishi mumkin. Faraz qilamiz, biz yozuvlar bilan turli o‘lchamdagi nuqta ko‘rinishida xaritada shahar chizish funksiyasini yozayapmiz:

```
void draw_cities( int n, struct city citylist[] )
{
  int i;
  for (i=0; i < n; i++)
    {
```

```
    if (citylist[i].large_flag)
       glPointSize( 4.0 );
    else
       glPointSize( 2.0 );

    glBegin( GL_POINTS );
    glVertex2f( citylist[i].longitude,
              citylist[i].latitude );
    glEnd();
    /* shahar nomini chizamiz */
    DrawText(citylist[i].longitude,
           citylist[i].latitude,
           citylist[i].name);
   }
}
```

Ushbu amalga oshirish quyidagi sabablarga ko‘ra muvaffaqiyatsiz:

- glPointSize davrning har bir interasiyasi uchun chaqiriladi.
- glBegin va glEnd o‘rtasida faqat bitta nuqta chiziladi.
- uchlar qulay bo‘lmagan formatda belgilanadi.

Quyida ancha ratsional yechim keltirilgan:

```
 void draw_cities( int n, struct city citylist[] )
 {
    int i;
    /* dastlab kichkina nuqtalar chizamiz */
    glPointSize( 2.0 );
    glBegin( GL_POINTS );
    for (i=0; i < n ;i++)
    {
      if (citylist[i].large_flag==0) {
        glVertex2f( citylist[i].longitude,
```

```
                    citylist[i].latitude );
      }
     }
     glEnd();
     /* ikkinchi navbatda katta nuqtalar chizamiz */
     glPointSize( 4.0 );
     glBegin( GL_POINTS );
     for (i=0; i < n ;i++)
      {
      if (citylist[i].large_flag==1)
      {
        glVertex2f( citylist[i].longitude,
                citylist[i].latitude );
      }
     }
     glEnd();
    /* so‘ngra shaharlar nomini chizamiz */
   for (i=0; i < n ;i++)
   {
    DrawText(citylist[i].longitude,
          citylist[i].latitude,
          citylist[i].name);
   }
}
```

Bunday amalga oshirishda biz glPointSize buyrug‘ini ikki marta chaqiramiz va uchlar sonini glBegin va glEnd o‘rtasida oshiramiz.

Biroq optimallashtirish uchun yana yo‘llar qoladi. Agar biz ma’lumotlar tuzilmasini almashtirsak, unda nuqtalarni chizish samaradorligini yanada oshirishimiz mumkin. Masalan:

struct city_list

```
{
  int num_cities; /* ro‘yxatdagi shaharlar soni */
  float *position;/* shahar koordinatalari */
  char **name;    /* shaharlar nomiga ko‘rsatkich */
  float size;     /* shaharni bildiruvchi nuqta o‘lchami */
};
```

Endi turli o‘lchamdagi shahar turlicha ro‘yxatda saqlanadi. Nuqtaning holati dinamik massivda alohida saqlanadi. Qayta tashkil etishdan so‘ng biz glBegin/glEnd ichida shartli operator zaruriyatini chiqarib tashlaymiz va optimallashtirish uchun uchlar massividan foydalanish imkoniga ega bo‘lamiz. Natijada biz qurgan funksiya quyidagi ko‘rinishga keladi:

```
void draw_cities( struct city_list *list )
  {
  int i;

  /* nuqtani chizamiz */
  glPointSize( list->size );

  glVertexPointer( 2, GL_FLOAT, 0,
                   list->num_cities,
                   list->position );
  glDrawArrays( GL_POINTS, 0, list->num_cities );
  /* shahar nomini chizamiz */
  for (i=0; i < list->num_cities ;i++)
  {
    DrawText(citylist[i].longitude,
             citylist[i].latitude
             citylist[i].name);
```

```
  }
}
```

OpenGL da chaqirishlarni optimallashtirish.

OpenGL unumdorligini oshirishning ko‘pgina imkoniyatlari mavjud. Bundan tashqari optimallashtirishga bo‘lgan turlicha yondashuvlar apparatli va dasturiy vizuallashtirishda turlicha effektlar bilan olib boriladi. Masalan, ranglar interpolyasiyasi apparat ta’minotisiz juda qimmatli operatsiya bo‘lishi mumkin, apparatli vizuallashtirishda ushlanish deyarli bo‘lmaydi.

Quyidagi uslublarning har biridan keyin, simvollardan biri ko‘rsatilgan va aniq tizim uchun ushbu uslubning ahamiyatini anglatuvchi kvadrat qavslar kuzatiladi:

- [A] – OpenGL ning apparatli ta’minot tizimi uchun afzal
- [D] – dasturiy amalga oshirish uchun afzal
- [barchasi] – ehtimol barcha amalga oshirishlar uchun afzal

OpenGL da ma’lumotlarni uzatish

Biz quyida OpenGL da primitivlar haqidagi ma’lumotlarni uzatishda vaqtni minimallashtirish usullarini ko‘rib chiqamiz.

Bog‘langan primitivlardan foydalaning.

GL_LINES, **GL_LINE_LOOP**, **GL_TRIANGLE_STRIP**, **GL_TRIANGLE_FAN**, va **GL_QUAD_STRIP** singari bog‘langan primitivlar, alohida chiziq yoki ko‘pburchakka qaraganda kam uchlarni belgilash uchun talab etiladi. Bu OpenGL ga uzatilayotgan ma’lumotlar sonini kamaytiradi. [barchasi]

Uchlar massivlaridan foydalaning.

Ko‘pgina arxitekturalarda ko‘p sonli chaqirishlarni (glVertex/glColor/glNormal) almashtirish uchlar massivi mexanizmida juda ham yutuqli bo‘lishi mumkin. [barchasi]

Indekslangan primitivlardan foydalaning.

Ayrim hollarda hattoki bog‘langan primitivlardan **GL_TRIANGLE_STRIP** (**GL_QUAD_STRIP**) foydalanganda ham uchlar nusxalanadi.

OpenGL da nusxalarni uzatmaslik uchun, shinaga vazifalarni ko‘paytiruvchi, glDrawElements() buyrug‘idan foydalaning. (3.3. paragrafga qarang) [barchasi]

Kerakli massivlarni bitta buyruq bilan bering.

Buyruqni ishlatish o‘rniga

glVertexPointer/glColorPointer/glNormalPointer bitta buyruqdan foydalanish ma’qul

```
void glInterleavedArrays ( Glint format, Glsizei stride, void * ptr);
```

xuddi shunday, tuzilmaga ega bo‘lsa

```
typedef struct tag_VERTEX_DATA
    {
    float color[4];
    float normal[3];
    float vertex[3];
    }VERTEX_DATA;
VERTEX_DATA * pData;
```

unda parametrlarni quyidagi buyruq yordamida uzatish mumkin:

```
glInterleavedArrays (GL_C4F_N3F_V3F, 0, pData);
```

bunda, birinchi to‘rtta float rangga tegishli, keyingi uchta float normalga, va oxirgi uchta float uchlar koordinatalarini beradi. Buyruqning batafsil tavsifini OpenGL tasnifidan ko‘ring. [barchasi]

Uchlar haqida ma’lumotlarni xotirada ketma-ket saqlang.

Ma’lumotlarni xotirada ketma-ket saqlanishi, asosiy xotira va grafik nimtizim o‘rtasida almashinuv tezligini oshiradi. [A]

glVertex, glColor, glNormal i glTexCoord **vektorli versiyalaridan foydalaning.**

glVertex*(), glColor*() va boshqa funksiyalar, argument sifatida ko‘rsatkichlarni (masalan, glVertex3fv(v)) qabul qilib, mos versiyalariga glVertex3f(x,y,z) nisbatan tez ishlashi mumkin. [barchasi]

Primitivlar murakkabligini kamaytiring.

Ko‘pgina hollarda katta tekisliklarni keragidan ortiqcha qismlarga bo‘lib tashlamaslik uchun ogoh bo‘ling. Masalan sifat va unumdorlikning eng yaxshi o‘zaro nisbatini belgilash uchun GLU primitivlari bilan tajriba o‘tkazing. Teksturalangan obyektlar, masalan, geometriyaning o‘rtacha murakkabligi bilan sifatli tasvirlanishi mumkin.

Displey ro‘yxatidan foydalaning.

Tez-tez chaqirilgan obyektlar uchun displey ro‘yxatidan foydalaning. Displey ro‘yxati grafik nimtizimida saqlanishi mumkin va demak, asosiy xotiradan ma’lumotlarni tez-tez olib o‘tilishini olib tashlash mumkin. [A]

Uchlarning kerak bo‘lmagan atributlarini ko‘rsatmang.

Agar yorug‘lik o‘chirilgan bo‘lsa, glNormal buyrug‘ini chaqirmang. Agar tekstura ishlatilmayotgan bo‘lsa, glTexCoord buyrug‘ini chaqirmang, va b. [barchasi]

glBegin/glEnd orasidagi ortiqcha kodlar sonini kamaytiring.

high-end tizimlarida maksimal unumdorlik, uchlar haqidagi axborot grafik nimtizimga maksimal darajada tez uzatilishi uchun muhim. glBegin/glEnd o‘rtasida ortiqcha kodlardan yiroqlashing.

Muvaffaqiyatsiz echimga misol:

```
glBegin(GL_TRIANGLE_STRIP);
for (i=0; i < n; i++)
{
  if (lighting)
  {
    glNormal3fv(norm[i]);
```

```
    }
    glVertex3fv(vert[i]);
  }
  glEnd();
```

Bu konstruksiya shunisi bilan yomonki, biz lighting o‘zgaruvchisini har bir uch oldidan tekshiramiz. Bunday holatdan qochish mumkin, kodning qismlangan nusxasi hisobiga:

```
if (lighting)
{
  glBegin(GL_TRIANGLE_STRIP);
  for (i=0; i < n ;i++)
  {
    glNormal3fv(norm[i]);
    glVertex3fv(vert[i]);
  }
  glEnd();
 }
 else
 {
  glBegin(GL_TRIANGLE_STRIP);
  for (i=0; i < n ;i++)
  {
    glVertex3fv(vert[i]);
  }
  glEnd();
}
```

O‘zgartirish

O‘zgartirish o‘zida, oyna koordinatalari, kesish, yorug‘lik va boshqalarni berishda ishlatiladigan glVertex*() bilan ko‘rsatiluvchi koordinatalar orqali uchlarning o‘zgarishini namoyon etadi.

Yorug‘lik

- Yorug‘likning lokal manbalaridan foydalanishdan qochish, ya’ni manba koordinatalari (*x*,*y*,*z*,0) shaklida bo‘lishi kerak. [D]
- Yorug‘likning nuqtasimon manbasidan foydalanishdan qochish. [barchasi]
- Ikki tomonlama yoritishdan (two-sided lighting) foydalanishdan qochish. [barchasi]
- Yorug‘lik va material parametrlaridagi manfiy koeffisientlardan foydalanishdan qochish. [D]
- Yorug‘likning lokal modelidan foydalanishdan qochish. [barchasi]
- **GL_ShININESS** materiali parametrining tez-tez almashishidan qochish. [D]
- OpenGL ning ba’zi amalga oshirishlari yorug‘likning bitta manbasi holi uchun optimallashtirilgan. [A, D]
- Yorug‘likni oldindan hisoblab chiqish imkonini ko‘rib chiqish. Normal o‘rniga uchga rang berish orqali yorug‘lik effektiga ega bo‘lish mumkin. [barchasi]

Zarur bo‘lmagan hollarda, normal vektorlarini normallashtirishni o‘chirish.

glEnable/Disable(GL_NORMALIZE) buyrug‘i normallardan foydalanishdan oldin normal vektorlarini normallashtirishni boshqaradi. Agarda siz glScale buyrug‘idan foydalanmasangiz, unda normallashtirishni boshqa effektlarsiz o‘chirish mumkin. Odatda bu band o‘chirilgan bo‘ladi. [barchasi]

Bog‘langan primitivlardan foydalaning.

GL_LINES, **GL_LINE_LOOP**, **GL_TRIANGLE_STRIP**, **GL_TRIANGLE_FAN** va **GL_QUAD_STRIP** kabi bog‘langan primitivlar

OpenGL konveyeri ishini kamaytiradi, shuningdek grafik nimtizimga uzatiluvchi ma’lumotlar sonini qisqartiradi.

Rasterizasiya

Rasterizasiya ko‘pincha OpenGL ni dasturiy amalga oshirishning cheklangan joyi hisoblanadi.

Zaruriyat bo‘lmaganda ranglar interpolyasiyasini o‘chiring.

Ranglar interpolyasiyasi odatda yoqilgan bo‘ladi. Tekis soya tashlash rangning to‘rtta qismi interpolyasiyasini talab etmaydi va qoida sifatida, OpenGL da dasturiy amalga oshirishda tezroq. Apparatli amalga oshirish odatda soya tashlashning ikkala ko‘rinishini ham bir xil tezlikda bajaradi. O‘chirish uchun glShadeModel(GL_FLAT) buyrug‘idan foydalaning. [D]

Zaruriyat bo‘lmaganda chuqurlikka bo‘lgan testni o‘chiring.

Fon berilgan obyektlar, masalan, chuqurlashtirish testsiz chizilishi ham mumkin, agarda ular birinchi bo‘lib vizuallashtirilganda. [barchasi]

Poligonlarning teskari yoqlarini kesishdan foydalaning.

Yopiq obyektlar teskari yoqlarni glEnable(GL_CULL_FACE) kesish rejimini o‘rnatish bilan chizilishi mumkin. Ba’zida bu ko‘pburchaklarning yarmigacha olib tashlash imkonini beradi, ularni rasterizasiya qilmasdan. [barchasi]

Piksellar bilan ortiqcha amallardan voz keching.

Rasterizasiya bosqichida niqoblashtirish, alfa-aralashtirish va boshqa piksel bo‘yicha operatsiyalar ko‘p vaqt oladi. Foydalanilmayotgan barcha operatsiyalarni o‘chiring. [barchasi]

Oyna o‘lchami yoki ekran kengligini qisqartiring.

Rasterizasiyalash vaqtini qisqartirishning oddiy usuli – chiziladigan piksellar sonini qisqartirish. Agar oynaning kichik o‘lchami yoki ekranning kichik kengaytmasi qabul qilingan bo‘lsa, unda bu rasterizasiyalash tezligini oshirish uchun yaxshi yo‘l hisoblanadi. [barchasi]

Teksturalash

Tekstura qo‘yish dasturiy va apparatli amalga oshirishlarda muhim operatsiya hisoblanadi.

Tasvirlarni saqlashning samaradorli formatlaridan foydalaning.

GL_UNSIGNED_BYTE formati odatda OpenGL ga teksturalarni uzatish uchun ancha mos keladi.

Displey ro‘yxati yoki tekstura obyektlarida teksturalarni birlashtiring.

Agarda siz bir qancha teksturadan foydalanayotgan bo‘lsangiz bu juda muhim va grafik nimtizimga videoxotirada teksturalarni joylashtirishni samarali boshqarish imkonini beradi. [barchasi]

Katta o‘lchamdagi teksturalarni ishlatmang.

Katta bo‘lmagan teksturalarga tez ishlov beriladi va xotirada kam joy egallaydi, shu jumladan, grafik nimtizimiga barcha teksturani saqlash imkonini beradi. [barchasi]

Katta bo‘lmagan teksturalarni bittaga birlashtiring.

Agar siz bir qancha kichik teksturalarni ishlatadigan bo‘lsangiz, ularni bitta katta o‘lchamdagiga birlashtirish va kerakli tekstura ustida ishlash uchun tekstura koordinatalarini o‘zgartirish mumkin. Bu teksturaning o‘zgarishi sonini qisqartirish imkonini beradi.

Animirlashtirilgan teksturalar.

Agar siz animirlashtirilgan teksturadan foydalanmoqchi bo‘lsangiz, tekstura ko‘rinishini yangilamaslik uchun glTexImage2D buyrug‘idan foydalanmang. Buning o‘rniga glTexSubImage2D ili glTexCopyTexSubImage2D dan foydalaning.

Buferlarni tozalash

Rang, chuqurlik, niqob buferlari va to‘plovchi buferni tozalash ko‘p vaqt talab qilish mumkin, ayniqsa OpenGL ni dasturiy amalga oshirishda. Bu bo‘limda ushbu operatsiyani optimallashtirishga yordam berishi mumkin bo‘lgan ba’zi usullar berilgan.

glClear **buyrug‘idan ehtiyotkorlik bilan foydalaning. [barchasi]**

Kerakli barcha buferlarni glClear buyrug‘i yordamida tozalang.

Noto‘g‘ri:

```
glClear(GL_COLOR_BUFFER_BIT);
if (stenciling)  /* niqob buferini tozalash kerakmi? */
{
   glClear(GL_STENCIL_BUFFER_BIT);
}
```

To‘g‘ri:

```
if (stenciling) /* niqob buferini tozalash kerakmi? */
{
                        glClear(GL_COLOR_BUFFER_BIT |
        STENCIL_BUFFER_BIT);
}
else
{
  glClear(GL_COLOR_BUFFER_BIT);
}
```

Ochranglar uzatilishini (dithering) o‘chiring.

Bufer tozalashdan oldin och ranglarni uzatilishini o‘chiring. Odatda och ranglar o‘rnatilgan va u siz tozalash o‘rtasidagi farq sezilmaydi. [D]

Kichkina sohalarni tozalashda qaychilardan (scissors) foydalaning.

Agar siz butun buferni tozalamoqchi bo‘lsangiz, berilgan soha bo‘yicha tozalash chegarasi uchun glScissor() buyrug‘idan foydalaning. [barchasi]

Rang buferini to‘liqligicha tozalamang.

Agar sizning sahnangiz oynani bir qismini egallasa, rangning butun buferini tozalashning hojati yo‘q.

glClearDepth (d) buyrug‘idan foydalanmang, bu yerda d!=1.0

Ayrim dasturiy amalga oshirishlar buferni 1.0. chuqurlikda tozalash uchun optimallashtirishgan. [D]

Har xil

Dastur yozish vaqtida GL xatolarini tekshiring. [barchasi]

OpenGL funksiyalaridan birini chaqirish vaqtida xatolikka yo‘l qo‘yilmaganligini tekshirish uchun glGetError() buyrug‘ini chaqiring. Qoida sifatida, xatolik OpenGL buyruqlarining noto‘g‘ri parametrlari yoki buyruqlarning noto‘g‘ri ketma-ketligi hisobiga yuzaga keladi. Kodning yakuniy versiyasi uchun ushbu tekshirishni o‘chiring, bu ishni sezilarli darajada sekinlashtiradi. Tekshirish uchun, masalan, shunday makrosdan foydalanish mumkin:

```
#include <assert.h>
#define ChECK_GL \
  assert(glGetError() != GL_NO_ERROR);
```

Uni shunday ishlatish mumkin:

```
glBegin(GL_TRIANGLES);
glVertex3f(1,1,1);
glEnd();
ChECK_GL
```

glMaterial o‘rniga glColorMaterial buyrug‘idan foydalaning.

Agar sahnada obyektlar materiali hech bo‘lmaganda bitta parametr bo‘yicha farq qilsa, glColorMaterial buyrug‘i glMaterial buyrug‘iga nisbatan tez ishlaydi. [barchasi]

OpenGL holati o‘zgarishlar sonini minimallashtirish.

OpenGL holatini o‘zgartiruvchi buyruqlar (glEnable/glDisable/glBindTexture va boshq.), yaxlitlikni takroriy ichki tekshiradi, qo‘shimcha ma’lumotlar

tuzilmasini yaratadi va boshqalarni chaqiradi, bu esa to‘xtalishlarga olib keladi. [barchasi]

glPolygonMode buyrug‘idan foydalanmaslikka harakat qiling.

Agar sizga ko‘pgina rangsiz ko‘pburchaklilarni chizish kerak bo‘lsa, primitivlar chizish rejimini o‘zgartirish o‘rniga glBegin buyrug‘ini **GL_POINTS**, **GL_LINES**, **GL_LINE_LOOP** yoki **GL_LINE_STRIP** bilan ishlating, bu esa tezroq bo‘lishi mumkin. [barchasi]

Albatta, bu tavsiyalar OpenGL ilovasini optimallashtirish bo‘yicha faqat imkoniyatlarning kichik qismini o‘zida qamraydi. Shunga qaramasdan, ulardan to‘g‘ri foydalanish sizning dasturingiz ishini jiddiy ravishda tezlashtirishi mumkin.

Nazorat savollari:

1. OpenGL ilovalarini yuqori darajali optimallashtirish usullaridan sizga ma’lum bo‘lganlarini keltiring?
2. OpenGL ni past darajali optimallashga deganda nima tushuniladi?
3. OpenGL unumdorligini oshirish imkoniyatlari.
4. Nima uchun bog‘langan primitivlardan foydalanish afzal ko‘riladi?
5. Quyidagi ikki buyruqdan qaysi biri OpenGL ni tez ishlashini ta’minlaydi?
   glVertex3f(1,1,1) yoki float vct[3] = {1,1,1}; glVertex3fv(vct)

Tayanch iboralar: Yuqori darajali optimallash, past darajali optimallash, chaqirishlarni optimallashtirish, teksturalash.

Ilova A. GLUT - ilovasining tuzilishi

GLUT kutubxonasi yordamida konsol ilovalar tuzilishini hosil qilishni qarab chiqamiz. Bu kutubxona plotformadan qat'iy nazar oynalar bilan ishlash uchun yagona interfeysni taminlaydi, shuning uchun quyidagi ilova tuzilmalari Windows, Linux va boshqa OS lar uchun o‘zgarmas bo‘lib qoladi.

GLUT funksiyasi o‘zining vazifalari bo‘yicha bir qancha guruhlarda tavsiflanishi mumkin:

- initsializatsiya;
- hodisalarni qayta ishlashni boshlanishi (Nachalo obrabotki sobitiy);
- oynalarni boshqarish (Upravlenie oknami);
- menyularni boshqarish (Upravlenie menyu);
- teskari chaqiruvlar funksiyasini ro‘yxatdan o‘tkazish;
- indekslangan ranglar palitrasini boshqarish;
- shriftlarni aks ettirish;
- qo‘shimcha geometrik shakllar (tor, konus va hokazolar)ni tasvirlash.

Initsializatsiya quyidagi funksiyalar yordamida amalga oshiriladi:

glutInit (int *argcp, char **argv)

argcp o‘zgaruvchi bu main() funksiyasida tasvirlanadigan argc standart o‘zgaruvchisidagi ko‘rsatgichidir, argv esa ushbu funsiyadagi tasvirlanayotgan dastur boshlanishida uzatiladigan parametrlarga ko‘rsatgich bo‘ladi. Bu funksiya ilova oynasi strukturasi uchun zarur bo‘lgan boshlang‘ich harakatlarni amalga oshiradi va bir nechtagina GLUT funksiyalari uchungina chaqirilishi mumkin. Bularga:

glutInitWindowPosition (int x, int y)

glutInitWindowSize (int width, int height)

glutInitDisplayMode (unsigned int mode)

Birinchi ikkala funksiya mos ravishda oyna holati va o‘lchamini ifodalaydi, oxirgi funksiya esa ma'lumotlarni har xil aks ettirishni ifodalaydi, bular “yoki” (“ | ”) bitlik operatsiyasi bilan birgalikda berilishi mumkin;

GLUT_RGBA RGBA rejimi. **GLUT_RGBA** yoki **GLUT_INDEX** rejimlari aniq ko‘rsatilmagan holda ishlatiladi.

GLUT_RGB Bu ham **GLUT_RGBA** kabi.

GLUT_INDEX indeksirlangan ranglar rejimi (palitrlardan foydalanish). **GLUT_RGBA** ni bekor qiladi.

GLUT_SINGLE Alohida buferli oyna. Odatdagi bo‘yicha foydalaniladi.

GLUT_DOUBLE Ikkitali buferli oyna. **GLUT_SINGLE** ni bekor qiladi.

GLUT_STENCIL Niqob buferli oyna.

GLUT_ACCUM To‘plagich buferli oyna.

GLUT_DEPTH Chuqurlik buferli oyna**.**

Bu berilgan funksiyaning parametrlari ro‘yxati to‘liq emas, biroq ko‘p holatlar uchun bular yetarli bo‘ladi.

GLUT kutubxonasi funksiyasi hodisa-boshqaruv deb nomlangan mehanizmni amalga oshiradi. Bu initsializatsiyasi davomida aniqlangan barcha hodisalarni birma-bir qayta ishlaydi, tegishli initsializatsiyadan keyin ishga tushadigan qandaydir ichki siklning borligini anglatadi. Bularga: sichqoncha chertkisi, oynaning yopilishi, oynalar hossalarining o‘zgarishi, kursor holatining o‘zgarishi, klavish bosilishi va hech narsa sodir bo‘lmayotgan paytdagi hodisalar. Biror voqeaning bajarilishini davriy tekshirish o‘tkazish uchun uni qayta ishlaydigan funksiyasini registrasiyadan o‘tkazish kerak. Buning uchun funksiya ko‘rinishi quyidagicha:

void **glutDisplayFunc** (void (*func) (void))

void **glutReshapeFunc** (void (*func) (int width, int height))

void **glutMouseFunc** (void (*func) (int button, int state, int x, int y))

void **glutIdleFunc** (void (*func) (void))

void **glutMotionFunc** (void (*func)(int x, int y));

void **glutPassiveMotionFunc** (void (*func)(int x, int y));

Shu parametr tipiga mos nomi beriladi. **glutDisplayFunc()** yordamida oyna ilovasi uchun kerak bo‘ladigan tasvirlarni yaratish yoki qayta tiklash mumkin. Aniq

ko‘rsatilgan oyna uchun void **glutPostRedisplay** (void) funksiyasidan foydalanib yangilash qulay sanaladi.

glutReshapeFunc() orqali yangi o‘lchamlar berishda foydalanuvchi oyna o‘lchami o‘zgarishi uchun qayta ishlash funksiyasi sozlanadi.

glutMouseFunc() funksiyasi aniqlaydi – sichqoncha orqali beriladigan buyruqlarni qayta ishlash uchun, **glutIdleFunc()** funksiyasi esa, foydalanuvchi hodisada qatnashmagan paytda o‘zi har gal avtomatik chaqiriladi.

glutMotionFuns funksiyasi foydalanuvchi tamonidan sichqoncha ko‘rsatkichi harakatlantirilganda sichqoncha tugmasini ushlab turilganda bajariladi.

glutPassiveMotionFunc funksiyasi foydalanuvchi tomonidan sichqoncha ko‘rsatkichi harakatlantirilganda sichqoncha tugmasiga hech qanday ta’sir ko‘rsatilmaganda bajariladi.

void **glutMainLoop** (void) funksiyasi cheksiz ichki siklda barcha hodisalarni nazorat qilib, odatda GLUT kutubxonasi yordamida ixtiyoriy dasturning oxirida chaqiriladi. Quyidagi dastur ilova strukturasida animatsiyadan foydalanib ishlash ko‘rsatilgan:

```
#include <GL/glut.h>
 void MyCut(void)
 {
/*Quyidagi kadrlarni belgilovchi, o‘zgaruvchilarni almashtiradigan kod */
…
};
void MyDisplay(void)
{
 /* Kadrni tasvirlash OpenGL kodi */
  …
 /* Chizishdan keyin buferlarni joylashtiramiz */
  glutSwapBuffers();
 };
```

```
void main(int argcp, char **argv)
{
 /* GLUT Initsializatsiyasi */
 glutInit(&argcp, argv);
 glutInitWindowSize(640, 480);
 glutInitWindowPosition(200, 200);
 /* Oynani ochish */
 glutCreateWindow("My OpenGL Application cutting");
 /* tartibni tanlash:ikkilangan bufer va RGBA ranglari */
 glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
 /* funksiyani chaqirishni registrasiyalash */
 glutDisplayFunc(MyDisplay);
 glutIdleFunc(MyCut);
 /* voqealarni qayta ishlash mexanizmini yoqish*/
 glutMainLoop();
};
```

GLUT_DOUBLE ni **GLUT_SINGLE** ga almashtirish uchun ilovada tasvirning statik ko‘rinishini tuzish kerak bo‘ladi, chunki bu holatda bitta bufer yetarli bo‘ladi. *glutIdleFunc()* chaqiruv funksiyasini olib tashlaydi.

Ilova B. GLU va GLUT kutubxonalarining primitivlari

GLU va GLUT kutubxonasida amalga oshirilgan primitivlarni qurishning standart buyruqlarini ko‘rib chiqamiz.

GLU kutubxonasida primitiv qurish uchun dastlab gluNewQuadric() buyrug‘i yordamida quadric-obyekt ko‘rsatkichini yaratish kerak, so‘ngra, gluSphere(), gluCylinder(), gluDisk(), gluPartialDisk() buyrug‘idan bittasini chaqirish kerak. Bu buyruqlarni alohida ko‘rib chiqamiz:

void **gluSphere** (GLUquadricObj **qobj*, GLdouble *radius*,
GLint *slices*, GLint *stacks*)

Bu funksiya koordinata boshi sfera markazida va radiusi *radius* bo‘lgan sfera yaratadi. Bunda sferaning *z* o‘qi atrofidagi kesimlari soni *slices* parametrini beradi, *z* o‘qi bo‘ylab esa *stacks* parametrini beradi.

void **gluCylinder** (GLUquadricObj **qobj*,
GLdouble *baseRadius*,
GLdouble *topRadius*,
GLdouble *height*, GLint *slices*,
GLint *stacks*)

Berilgan funksiya *z* o‘qiga parallel o‘q bo‘ylab, orqa asos radiusi *baseRadius* bo‘lgan va *z*=0 tekislikda joylashgan, oldingi asosi radiusi *topRadius* va *z=height* tekislikda joylashgan asossiz silindr (ya’ni xalqa) tuzadi. Agar radiuslardan biri nolga teng bo‘lsa konus hosil bo‘ladi, *slices* va *stacks* parametrlari analogik ma’noga ega bo‘lib, ular haqida yuqorida buyrug‘larni ko‘rdik.

void **gluDisk** (GLUquadricObj **qobj*,
GLdouble *innerRadius*,
GLdouble *outerRadius*, GLint *slices*,
GLint *loops*)

Funksiya koordinata boshi markazi bo‘lib va radiusi *outerRadius* yassi disk (ya’ni doira) quradi. Agar bunda *innerRadius* qiymati noldan farqli bo‘lsa, shuningdek disk markazida radiusi *innerRadius* teshik joylashgan bo‘ladi. *Slices*

parametri *z* o‘qi atrofidagi desk kesimlar sonini, *loops* parametri esa – *ox* va *z* o‘qiga perpendikulyar konsentrik xalqalar soni.

void **gluPartialDisk** (GLUquadricObj **qobj*,
GLdouble *innerRadius*,
GLdouble *outerRadius*, GLint *slices*,
GLint *loops*, GLdouble *startAngle*,
GLdouble *sweepAngle*);

Bu buyruqning oldingilardan farqi shundaki, u soat strelkasiga qarama-qarshi *y* o‘qi musbat yo‘nalishidan hisoblanganda va *startAngle* va *sweepAngle* parametrlar berilganda boshlang‘ich va oxirgi burchaklaridan doira sektorini hosil qiladi. Burchaklar graduslarda beriladi.

GLUT kutubxonasidan primitivlar tuzishni olib boradigan buyruqlar, OpenGL va GLU standart primitivlari orqali amalga oshiriladi. Kerakli primitivlarni hosil qilish uchun mos keluvchi buyruqlarni chaqirish kerak bo‘ladi.

void **glutSolidSphere** (GLdouble *radius*, GLint *slices*, GLint *stacks*)

void **glutWireSphere** (GLdouble *radius*, GLint *slices*, GLint *stacks*)

glutSolidSphere() buyrug‘i sfera yasash uchun, **glutWireSphere()** – buyrug‘i esa *radius* radiusli sfera karkasini yasash uchun kerak bo‘ladi. Qolgan parametrlar oldingi buyruqlar kabi bo‘ladi.

void **glutSolidCube** (GLdouble *size*)

void **glutWireCube** (GLdouble *size*)

buyruqlari qirra uzunligi *size* va koordinata boshida markaz bilan kub yoki kubning karkasini chizadi.

void **glutSolidCone** (GLdouble *base*, GLdouble *height*, GLint *slices*, GLint *stacks*)

void **glutWireCone** (GLdouble base, GLdouble height, GLint *slices*, GLint *stacks*)

Bu buyruq orqali *z* o‘qi bo‘ylab joylashgan balandligi *height*, asos radiusi *base* bo‘lgan konus yoki uning karkas chiziladi. Asoslari z=0 tekislikda joylashgan bo‘ladi.

void **glutSolidTorus** (GLdouble *innerRadius*, GLdouble *outerRadius*, GLint *nsides*, GLint *rings*)

void **glutWireTorus** (GLdouble *innerRadius*, GLdouble *outerRadius*, GLint *nsides*, GLint *rings*)

Bu buyruqlar *z*=0 tekislikda tor yoki uning karkasini quradi. Ichki va tashqi radiuslar *innerRadius*, *outerRadius* parametrlari bilan beriladi. *nsides* parametri torning ortogonal kesimini tashkil qiluvchi tomonlari soni, *rings esa* – tor radial kesimlari sonini amalga oshiradi.

void **glutSolidTetrahedron** (void)

void **glutWireTetrahedron** (void)

Bu buyruqlar radiusi 1 ga teng bo‘lgan sferaga tashqi chizilgan tetraedr (muntazam uchburchakli) yoki uning karkasini hosil qiladi.

void **glutSolidOctahedron** (void)

void **glutWireOctahedron** (void)

Bu buyruqlar radiusi 1 ga teng bo‘lgan sferaga tashqi chizilgan oktaedr yoki uning karkasini hosil qiladi.

void **glutSolidDodecahedron** (void)

void **glutWireDodecahedron** (void)

Bu buyruqlar sferaga tashqi chizilgan ikosaedr yoki uning karkasini hosil qiladi.

void **glutSolidIcosahedron** (void)

void **glutWireIcosahedron** (void)

Bu buyruqlar radiusi 1 ga teng bo‘lgan sferaga tashqi chizilgan ikosaedr yoki uning karkasi hosil qiladi.

Quyidagi keltirilgan primitivlarni to‘g‘ri tashkil qilish uchun ko‘rinmas chiziq va sirtlarni olib tashlash kerak bo‘ladi, buning uchun **glEnable**(GL_DEPTH_TEST) buyrug‘iga mos keluvchi rejimni chaqirish kerak.

Ilova C. OpenGL ilovalarini sozlash

Borland C++ 5.02 muhitida ilova yaratish.

Dastlab BorlandC\Include\Gl, BorlandC\Lib, Windows\System kataloglarida glut.h, glut32.lib, glut32.dll fayllarning mavjudligini ta'minlash zarur. Shuningdek ushbu kataloglarda gl.h, glu.h, opengl32.lib, glu32.lib, opengl32.dll, glu32.dll, fayllarni mavjudligini tekshirish kerak. Ular odatda BorlandC++ va Windows tarkibiga kiradi. Shu bilan birgalikda Microsoft ning versiyalaridagi opengl32.lib, glu32.lib, glut32.lib fayllar Borland C++ muhiti uchun to‘g‘ri kelmaydi va faqat mos tushadigan versiyalardan foydalanish kerak. Bunday versiyalar yaratish BorlandC\Bin katalogida ‘implib’ standart dasturini ishlatish kerak. Buning uchun *.lib faylini mos keluvchi *.dll faylida yaratish quyidagi tartibda bajarilishi kerak:

implib BorlandC\Lib\filename.lib filename.dll

Yana ta'kidlab o‘tish kerakki Borland noma'lum sabablarga ko‘ra BorlandC++5.02 tarkibiga kiruvchi GLAUX kutubxonasiga kiruvchi ilova kompilyasiyasi glaux.lib faylini ishlata olmaydi, shuning uchun bu kutubxonadan voz kechishga to‘g‘ri keladi. Ilova yaratish uchun quyidagi amallarni bajarish kerak:

- Loyiha yaratish: buning uchun *Project->New Project* tanlanadi va *Target Expert* oynasidagi maydonni quyidagi tartibda to‘ldirish kerak: *Platform* maydonida Win32 tanlab, *Taget Model* maydonida *Sonsole* ni tanlab, *Advanced* ni bosib va ‘*.rc ‘ va ‘ *.def ‘ tanlash punklarini bekor qilish kerak.
- Loyihaga OpenGL kutubxonalarini qo‘shish. Buning uchun loyiha oynasida ijro qilinayotgan (*.exe) faylni tanlab va sichqonchaning o‘ng tugmasini bosib kontekst menyudan *Add node* punktini tanlash kerak keyin bu opengl32.lib, glu32.lib, glut32.lib fayllarni joylashish holatini aniqlash kerak.
- Kompilyasiya uchun *Project->Build All* ni tanlash, bajarish uchun – *Debug->Run* ni tanlash kerak bo‘ladi.

MS Visual C++ 6.0 muhitida ilova yaratish.

Ishni boshlashdan oldin MSVC\Include\Gl, MSVC\Lib, Windows\System kataloglariga glut.h, glut32.lib, glut32.dll fayllarni mos ravishda ko‘chirib olish

kerak. Shuningdek bu kataloglarda Visual C++ va Windows tarkibiga kiradigan gl.h, glu.h, opengl32.lib, glu32.lib, opengl32.dll, glu32.dll fayllar borligini tekshirish kerak. GLAUX kutubxonasida joylashgan buyruqlardan foydalanish uchun sanab o‘tilgan fayllar ro‘yxatiga glaux.h, glaux.lib fayllarni ham qo‘shish kerak.

Ilova yaratish uchun quyidagi amallarni bajarish kerak:

- Loyiha yaratish: buning uchun *File->New->Projects->Win32 Console Application* ni tanlab, loyihaga nom berib OK ni bosish kerak.
- Paydo bo‘lgan oynadan ‘*An empty project*’ ni tanlab, Finish, OK ni bosish kerak.
- Dastur matni hosil qilingan tekstli faylga (*File->New->Files->Text File* ni tanlash mumkin), kengaytmasi *.c yoki *.cpp bo‘lgan faylni loyihaga qo‘shish kerak (*Project->Add To Project->Files* ni tanlash orqali).
- Loyihaga OpenGL kutubxonalarini qo‘shish. Buning uchun *Project->Settings->Link* ni tanlab va *Object/library modules* maydonida kerakli kutubxona nomini terish kerak: opengl32.lib, glu32.lib, glut32.lib va zarur hollarda glaux.lib.
- kompilyasiya uchun *Build->Build program.exe* ni, ishga tushirish uchun – *Build->Execute program.exe* ni tanlash zarur.
- Ishga tushgan paytda matnli maydon hosil bo‘lmasligi uchun *Project->Settings->Link* buyrug‘ini tanlab *Project Options* maydonida ‘subsystem:console’ o‘rniga ‘subsystem: windows’ ni terish kerak va shu qatorning o‘zida ‘/entry:mainCRTStartup’ ni terish kerak.
- Dastur tayyor bo‘lgandan keyin ‘Release’ rejimini yana bir bor kompilyasiyadan o‘tkazish maqsadga muvofiqdir, bu dasturni tez ishlashi va optimallashtirish uchun foydali hisoblanadi. Buning uchun avval *Build->Set Active Configuration…* ni tanlab ‘…-Win32 Release’ni belgilab keyin yana bir bor kerakli kutubxonani qo‘shish kerak.

Borland C++ Builder 6. muhitida ilova yaratish.

Ishni boshlashdan oldin CBuilder6Mnclude\Gl, CBuilder6\Lib, Windows\System kataloglariga glut.h, glut32.lib glut32.dll fayllarni mos ravishda

ko‘chirib olish kerak. Shuningdek bu kataloglarda Borland C++ va Windows tarkibiga kiradigan gl.h, glu.h, opengl32.lib, glu32.lib, opengl32.dll, glu32.dll fayllar borligini tekshirish kerak.

Shu bilan birgalikda Microsoft ning versiyalaridagi opengl32.lib, glu32.lib, glut32.lib fayllar Borland C++ Builder 6 muhiti uchun to‘g‘ri kelmaydi va faqat mos tushadigan versiyalardan foydalanish kerak. Bunday versiyalarni yaratish uchun SVuilder6\Bin katalogida ‘implib’ standart dasturini ishlatish kerak. Buning uchun *.lib faylini mos keluvchi *.dll faylida yaratish quyidagi tartibda bajarilishi kerak:

implib glut32.1ib glut32.dll

Ilova yaratish uchun quyidagi amallarni bajarish kerak:

- Loyiha yaratish: buning uchun *File->New->Other->Console Wizard* ni tanlab OK ni bosish kerak.
- Paydo bo‘lgan oynadan Source Type - S++, Console Application ni tanlab, ‘Use VCL’, ‘Use CLX’, ‘Multi Threaded’ opsiyalarini olib tashlab OK ni bosish kerak.
- Dastur matnini yaratilgan matnli faylga joylashtirish mumkin, yoki uni loyihadan o‘chirib tashlash mumkin (*Project->Remove From Project*) va loyihaga *.c yoki *.cpp kengaytmali fayllarni qo‘shish mumkin (*Project ->Add To Project* ni tanlash orqali).
- Yaratilgan loyihani kerakli katalogda saqlang (*File ->Save All* ni tanlab).
- Loyihaga GLUT kutubxonasini qo‘shish. Buning uchun *Project->Add To Project* ni tanlab glut32.1ib faylini qo‘shish kerak.
- Kompilyasiya uchun *Project->Build* ni, ishga tushirish uchun – *Run->Run* ni tanlash zarur.
- Dastur tayyor bo‘lganda, tezkorligi va xajmi bo‘yicha optimallashtirish uchun uni ‘Release’ rejimida qaytadan kompilyasiyaga berish tavsif etiladi. Buning uchun avvalo *Project->Options ->Compiler* ni tanlab ‘Release’ tugmasini bosish kifoya.

Ilova D. Amaliy topshiriqlarga misollar

Masala 1: Sodda GLUT-ilovasi

Ushbu oddiy masala GLUT ilovasi strukturasini va OpenGL ning oddiy asoslarini ko‘rsatadi. Dastur natijasi uchburchaklar rangini sichqonchaning chap tugmasini bosganda o‘zgaradigan tasodifiy to‘plamidir.

Sichqonchaning o‘ng tugmasi bilan uchburchaklar rangini o‘zgartirish tartibiga yordam beradi.

```
#include <stdlib.h>
#include <gl/glut.h>

#ifdef random
#undef random
#endif

#define random(m) (float)rand()*m/RAND_MAX

/* oynaning kengligi va balandligi*/
GLint Width = 512, Height = 512;
/* oynadagi to‘g‘ri to‘rtburchaklar soni*/
int   Times = 100;
/* to‘ldiruvchilar bilan */
int   FillFlag  = 1;
long  Seed = 0;
/* uchburchaklarni tasvirlash  funksiyasi*/
void
DrawTriang( float x1, float y1, float x2, float y2, float x3, float y3,
        int FillFlag )
{
```

```
  glBegin(FillFlag ? GL_TRIANGLES : GL_LINE_LOOP);
  glVertex2f(x1, y1);
  glVertex2f(x2, y2);
  glVertex2f(x3, y3);
  glEnd();
}

/* ekranga chiquvchi barcha axborotlarni boshqarish */
void
Display(void)
{
  int i;
  float x1, y1, x2, y2, x3, y3;
  float r, g, b;
  srand(Seed);
  glClearColor(0, 0, 0, 1);
  glClear(GL_COLOR_BUFFER_BIT);

  for( i = 0; i < Times; i++ ) {
    r = random(1);
    g = random(1);
    b = random(1);
    glColor3f( r, g, b );

    x1 = random(1) * Width;
    y1 = random(1) * Height;
    x2 = random(1) * Width;
    y2 = random(1) * Height;
    x3 = random(1) * Width;
    y3 = random(1) * Height;
```

```
    DrawRect(x1, y1, x2, y2, x3, y3, FillFlag);
  }
  glFinish();
}
/* oyna o‘lchamlari o‘zgaradigan funksiya */
void
Reshape(GLint w, GLint h)
{
  Width = w;
  Height = h;
  glViewport(0, 0, w, h);

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0, w, 0, h, -1.0, 1.0);

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}
/*sichqonchadan keladigan ma’lumotni qayta ishlash */
void
Mouse(int button, int state, int x, int y)
{
  if( state == GLUT_DOWN ) {
    switch( button ) {
      case GLUT_LEFT_BUTTON:
        Seed = random(RAND_MAX);
        break;
      case GLUT_RIGHT_BUTTON:
```

```
        FillFlag = !FillFlag;
        break;
    }
    glutPostRedisplay();
  }
}

/* klaviaturadan keladigan ma’lumotni qayta ishlash */
void
Keyboard ( unsigned char key, int x, int y )
{
#define ESCAPE '\033'

  if( key == ESCAPE )
      exit(0);
}

main(int argc, char *argv[])
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_RGB);
  glutInitWindowSize(Width, Height);
  glutCreateWindow("Rect draw example (RGB)");

  glutDisplayFunc(Display);
  glutReshapeFunc(Reshape);
  glutKeyboardFunc(Keyboard);
  glutMouseFunc(Mouse);
  glutMainLoop();
}
```

Masala 2: OpenGL yoritish modeli

1. Ushbu dastur OpenGL modelini ishlatishda oddiy sahna orqali tor, kub, konus va shardan tuzilgan bo‘ladi. Obyekt qilib har xil materiallarni olamiz. Albatta sahnada yorug‘lik manbai bo‘ladi.

```
#include <GL/glut.h>
#include <stdlib.h>

/*tor materialining parametrlari*/
float mat1_dif[]={0.8f,0.8f,0.0f};
float mat1_amb[]= {0.2f,0.2f,0.2f};
float mat1_spec[]={0.6f,0.6f,0.6f};
float mat1_shininess=0.5f*128;

/* kub materialining parametrlari */
float mat2_dif[]={0.2f,0.6f,0.0f,0.0f};
float mat2_amb[]= {0.4f,0.4f,0.2f,0.0f};
float mat2_spec[]={0.4f,0.6f,0.4f,0.0f};
float mat2_shininess=0.3f*128;

/* shar materialining parametrlari */
float mat3_dif[]={0.9f,0.2f,0.0f};
float mat3_amb[]= {0.2f,0.2f,0.2f};
float mat3_spec[]={0.6f,0.6f,0.6f};
float mat3_shininess=0.1f*128;

/* konus materialining parametrlari */
float mat4_dif[]={0.0f,0.0f,0.8f};
float mat4_amb[]= {0.2f,0.2f,0.2f};
float mat4_spec[]={0.6f,0.6f,0.6f};
float mat4_shininess=0.7f*128;
```

```
/* yorug‘lik manbai va materiallar parametrlarini inisiallashtirish */
void init (void)
{
  GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
  GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
  GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
  GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

  /* yorug‘lik manbaining parametrlarini o‘rnatish */
  glLightfv (GL_LIGHT0, GL_AMBIENT, light_ambient);
  glLightfv (GL_LIGHT0, GL_DIFFUSE, light_diffuse);
  glLightfv (GL_LIGHT0, GL_SPECULAR, light_specular);
  glLightfv (GL_LIGHT0, GL_POSITION, light_position);

  /* yorug‘lik manbai va yoritgichni qo‘shish va yoqish */
  glEnable (GL_LIGHTING);
  glEnable (GL_LIGHT0);

  /*  z-buferni yoqamiz*/
  glEnable(GL_DEPTH_TEST);
}

/* Tasvirlarni chizish uchun kerak bo‘ladigan funksiyani chaqirish. Unda barcha geometrik shakllarni chiqarish amalga oshiriladi.  */
void display (void)
{
  /* kadr buferi va chuqurlik buferini tozalash*/
  glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

```
  glPushMatrix ();
  glRotatef (45.0, 1.0, 0.0, 0.0);

/* tor tasviri*/
  glMaterialfv (GL_FRONT,GL_AMBIENT,mat1_amb);
  glMaterialfv (GL_FRONT,GL_DIFFUSE,mat1_dif);
  glMaterialfv (GL_FRONT,GL_SPECULAR,mat1_spec);
  glMaterialf  (GL_FRONT,GL_ShININESS,mat1_shininess);

  glPushMatrix ();
  glTranslatef (-0.75, 0.5, 0.0);
  glRotatef (90.0, 1.0, 0.0, 0.0);
  glutSolidTorus (0.275, 0.85, 15, 15);
  glPopMatrix ();

  /*kub tasviri */
  glMaterialfv (GL_FRONT,GL_AMBIENT,mat2_amb);
  glMaterialfv (GL_FRONT,GL_DIFFUSE,mat2_dif);
  glMaterialfv (GL_FRONT,GL_SPECULAR,mat2_spec);
  glMaterialf  (GL_FRONT,GL_ShININESS,mat2_shininess);

  glPushMatrix();
      glTranslatef(1,-0.5,1);
      glRotatef(901, 1.0, 0.0, 0.0);
      glutSolidCube(1);
   glPopMatrix();

  /* shar tasviri */
  glMaterialfv (GL_FRONT,GL_AMBIENT,mat3_amb);
  glMaterialfv (GL_FRONT,GL_DIFFUSE,mat3_dif);
```

```
  glMaterialfv (GL_FRONT,GL_SPECULAR,mat3_spec);
  glMaterialf  (GL_FRONT,GL_ShININESS,mat3_shininess);

  glPushMatrix ();
  glTranslatef (0.75, 0.0, -1.0);
  glutSolidSphere (1.0, 25, 25);
  glPopMatrix ();

  /*konus tasviri */
  glMaterialfv (GL_FRONT,GL_AMBIENT,mat4_amb);
  glMaterialfv (GL_FRONT,GL_DIFFUSE,mat4_dif);
  glMaterialfv (GL_FRONT,GL_SPECULAR,mat4_spec);
  glMaterialf  (GL_FRONT,GL_ShININESS,mat4_shininess);

  glPushMatrix ();
  glTranslatef (-0.75, -0.5, 0.0);
  glRotatef (270.0, 1.0, 0.0, 0.0);
  glutSolidCone (1.0, 2.0, 15, 15);
  glPopMatrix ();

  glPopMatrix ();
  /*ekranga sa?nani chi?arish */
  glFlush ();
}
/* foydalanuvchi tomonidan chaqiriladigan oyna o‘lchamlarini o‘zgartiruvchi
funksiya */
void reshape(int w, int h)
{
 /* oyna o‘lchamiga teng bo‘lgan chiqarish sohasi o‘lchamini o‘rnatish */
 glViewport (0, 0, (GLsizei) w, (GLsizei) h);
```

```
  /* oyna o‘lchamini hisobga olib proeksiyalar matritsasini beramiz */
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();

  gluPerspective(
    40.0, /* gradusda berilgan ko‘rish burchagi */
    (GLfloat)w/h, /*oynaning siqilish koeffisienti */
    1,100.0);  /*tekislik kesimigacha masofa*/
  glMatrixMode (GL_MODELVIEW);

  glLoadIdentity ();
  gluLookAt(
   0.0f,0.0f,8.0f, /* kamera holati */
   0.0f,0.0f,0.0f, /* sahnalar markazi */
   0.0f,1.0f,0.0f); /* y o‘qining musbat yo‘nalishi*/
}

/* klaviaturadagi klavishlarni bosganda chaqiriladigan funksiya */
void keyboard(unsigned char key, int x, int y)
{
   switch (key) {
      case 27: /* escape */
         exit(0);
         break;
   }
}

/* Ilovaning asosiy tsikli.
* Chuqurlik buferi bilan ekran rejimi o‘rnatiladi, oyna yaratiladi. */
```

```
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB
                       | GLUT_DEPTH);
  glutInitWindowSize (500, 500);
  glutCreateWindow (argv[0]);
  init ();
  glutReshapeFunc (reshape);
  glutDisplayFunc (display);
  glutKeyboardFunc (keyboard);
  glutMainLoop();
  return 0;
}
```

2. Ushbu dastur OpenGL muhitida beshta kubdan tashkil topgan va markaziy kubdan bir xil masofada yotuvchi kublardan tuzilgan bo‘ladi. Albatta sahnada yorug‘lik manbai bo‘ladi.

```
#include <GL/glut.h>

void init()
{
   /*  z-buferni yoqamiz*/
   glEnable(GL_DEPTH_TEST);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_COLOR_MATERIAL);
}
```

/* Tasvirlarni chizish uchun kerak bo‘ladigan funksiyani chaqirish. Unda barcha kublarni chiqarish amalga oshiriladi. */

```
void display()
{
  /* kadr buferi va chuqurlik buferini tozalash*/
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  /* oyna o‘lchamini hisobga olib proeksiyalar matritsasini beramiz */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   GLint viewport[4];
   glGetIntegerv(GL_VIEWPORT, viewport);
   double aspect = (double)viewport[2] / (double)viewport[3];
   gluPerspective(60, aspect, 1, 100);

   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();

   // move back a bit
   glTranslatef( 0, 0, -35 );

   static float angle = 0;
   angle += 1.0f;

   /* markaziy kub parametrlarini o‘rnatish */
   glPushMatrix();
      glTranslatef(0,0,0);
      glRotatef(angle, 0.1, 0.2, 0.5);
      glColor3ub(255,0,255);
      glutSolidCube(5);
```

```
glPopMatrix();

/* qo‘yidan ikkinchi kub parametrlarini o‘rnatish */
glPushMatrix();
   glTranslatef(10,-10,0);
   glRotatef(angle, 0.1, 0.2, 0.5);
   glColor3ub(255,0,0);
   glutSolidCube(5);
glPopMatrix();

/* yuqoridan ikkinchi kub parametrlarini o‘rnatish */
glPushMatrix();
   glTranslatef(10,10,0);
   glRotatef(angle, 0.1, 0.2, 0.5);
   glColor3ub(0,255,0);
   glutSolidCube(5);
glPopMatrix();

/* yuqoridan birinchi kub parametrlarini o‘rnatish */
glPushMatrix();
   glTranslatef(-10,10,0);
   glRotatef(angle, 0.1, 0.2, 0.5);
   glColor3ub(0,0,255);
   glutSolidCube(5);
glPopMatrix();

/* qo‘yidan birinchi kub parametrlarini o‘rnatish */
glPushMatrix();
   glTranslatef(-10,-10,0);
   glRotatef(angle, 0.1, 0.2, 0.5);
```

```
        glColor3ub(255,255,0);
        glutSolidCube(5);
    glPopMatrix();

    glutSwapBuffers();
}
/* foydalanuvchi tomonidan chaqiriladigan oyna o‘lchamlarini o‘zgartiruvchi
funksiya */
void reshape(int w, int h)
{
    glViewport(0, 0, w, h);
}

void timer(int extra)
{
    glutPostRedisplay();
    glutTimerFunc(16, timer, 0);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitWindowSize(640,480);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_DOUBLE);
    glutCreateWindow("CUBES");

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutTimerFunc(0, timer, 0);
```

```
    init();
    glutMainLoop();
    return 0;
}
```

3. Ushbu dastur OpenGL muhitida oddiy sahna orqali sfera, konus va tordan tuzilgan materiallarni ifodalaydi. Ushbu obyektlarning tur shaklida berilgan tasvirini va uning ranglar bilan to‘ldirilgan variantlari turli burchak ostidagi harakati keltirilgan va sahnada yorug‘lik manbai tushurilgan.

```
#include <windows.h>
#include <gl/glut.h>
#include <stdlib.h>

static int slices = 26;
static int stacks = 16;
/* GLUT qayta aloqalarga ishlov berish */
static void
resize(int width, int height)
{
/* oyna o‘lchamini hisobga olib proeksiyalar matritsasini beramiz */
const float ar = (float) width / (float) height;
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-ar, ar, -1.0, 1.0, 2.0, 180.0);

        glMatrixMode(GL_MODELVIEW);
       glLoadIdentity() ;
    }
```

```
static void
display(void)
{
   const double t = glutGet(GLUT_ELAPSED_TIME) / 1000.0;
   const double a = t*90.0;
   /* kadr buferi va chuqurlik buferini tozalash*/
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glColor3d(0,1,1);

   /* sfera materiali tasviri va uning parametrlari */
   glPushMatrix();
      glTranslated(-2.4,1.2,-6);
      glRotated(60,1,0,0);
      glRotated(a,0,0,1);
      glutSolidSphere(1,slices,stacks);
   glPopMatrix();

   /* konus materiali tasviri va uning parametrlari */
   glPushMatrix();
      glTranslated(0,1.2,-6);
      glRotated(45,1,0,0);
      glRotated(a,0,0,1);
      glutSolidCone(1,1,slices,stacks);
   glPopMatrix();

   /* tor materiali tasviri va uning parametrlari */
   glPushMatrix();
   glTranslated(2.4,1.2,-6);
glRotated(60,1,0,0);
glRotated(a,0,0,1);
```

```
        glutSolidTorus(0.2,0.8,slices,stacks);
    glPopMatrix();

   /* tur shaklidagi sfera materiali tasviri va uning parametrlari */
   glPushMatrix();
        glTranslated(-2.4,-1.2,-6);
        glRotated(60,1,0,0);
        glRotated(a,0,0,1);
        glutWireSphere(1,slices,stacks);
    glPopMatrix();

    /* tur shaklidagi konus materiali tasviri va uning parametrlari */
    glPushMatrix();
        glTranslated(0,-1.2,-6);
        glRotated(60,1,0,0.8);
        glRotated(a,0,0,1);
        glutWireCone(1,1,slices,stacks);
    glPopMatrix();

    /* tur shaklidagi tor materiali tasviri va uning parametrlari */
    glPushMatrix();
        glTranslated(2.4,-1.2,-6);
        glRotated(120,1,0,0);
        glRotated(a,0,0,1);
        glutWireTorus(0.2,0.8,slices,stacks);
    glPopMatrix();

    glutSwapBuffers();
}
```

```
static void
key(unsigned char key, int x, int y)
{
   switch (key)
   {
      case 27 :
      case 'q':
         exit(0);
         break;

      case '+':
         slices++;
         stacks++;
         break;

      case '-':
         if (slices>3 && stacks>3)
         {
            slices--;
            stacks--;
         }
         break;
   }

   glutPostRedisplay();
}

static void
idle(void)
{
```

```
   glutPostRedisplay();
}

/* yorug‘lik manbai va materiallar parametrlarini inisiallashtirish */
const GLfloat light_ambient[]  = { 0.0f, 0.0f, 0.0f, 1.0f };
const GLfloat light_diffuse[]  = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_position[] = { 2.0f, 5.0f, 5.0f, 0.0f };

const GLfloat mat_ambient[]    = { 0.7f, 0.7f, 0.7f, 1.0f };
const GLfloat mat_diffuse[]    = { 0.8f, 0.8f, 0.8f, 1.0f };
const GLfloat mat_specular[]   = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat high_shininess[] = { 100.0f };

/* Program entry point */

int
main(int argc, char *argv[])
{
   glutInit(&argc, argv);
   glutInitWindowSize(640,480);
   glutInitWindowPosition(20,20);
   glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);

   glutCreateWindow("GLUT ilovasi shakllari");

   glutReshapeFunc(resize);
   glutDisplayFunc(display);
   glutKeyboardFunc(key);
   glutIdleFunc(idle);
```

```
    glClearColor(5,0.9,0.9,0);
    glEnable(GL_CULL_FACE);
    glCullFace(GL_BACK);

    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LESS);

    /* yorug‘lik manbai va yoritgichni qo‘shish va yoqish */
    glEnable(GL_LIGHT0);
    glEnable(GL_NORMALIZE);
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHTING);

    /* yorug‘lik manbai va materialning parametrlarini o‘rnatish */
    glLightfv(GL_LIGHT0, GL_AMBIENT,  light_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE,  light_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);

    glMaterialfv(GL_FRONT, GL_AMBIENT,   mat_ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE,   mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR,  mat_specular);
    glMaterialfv(GL_FRONT, GL_ShININESS, high_shininess);

    glutMainLoop();

    return 0;
}
```

Masala 3: Teksturlash

Ushbu dasturni bajarishdan olingan natija halqalar falagi atrofida aylanib turuvchi teksturalar kiritilgan tetraedrni qurish hisoblanadi. MS Visual C++ muhitida dastur nomsiz ham kompilyasiyalanishi mumkin, Borland C++ da kompilyasiyalashda esa TextureInit() funksiyasini tasvirlash va chaqirishni tavsiflashga to‘g‘ri keladi, shundagina teksturalarni kiritish amalga oshirilmaydi. Yuqorida aytilganidek, GLAUX kutubxonasidagi funksiyalardan foydalanishga urinib ko‘rish dasturni kompilyasiyalashda hatoliklar haqidagi ma’lumotlarga olib keladi.

MS Visual C++ da dasturni kompilyasiyalashda ‘texture.bmp’ faylini loyiha katalogiga ko‘chirish yoki unga to‘liq yo‘lni ‘/’ beligsidan foydalanib ko‘rsatish kerak. Agar yo‘l ko‘rsatilmagan bo‘lsa, u holda operatsion tizimdan bajariluvchi faylni yuklashda teksturali fayl aynan bajarilayotgan fayl joylashgan katalogda bo‘lishi kerak.

```
#include <GL\glut.h>
#include <gl\glaux.h>
#include <math.h>
#define TETR_LIST 1

GLfloat light_col[] = {1,1,1};
float mat_diff1[]={0.8,0.8,0.8};
float mat_diff2[]={0.0,0.0,0.9};
float mat_amb[]= {0.2,0.2,0.2};
float mat_spec[]={0.6,0.6,0.6};
float shininess=0.7*128,
float CurAng=0, RingRad=1, RingHeight=0.1;
GLUquadricObj* QuadrObj;
GLuint TexId;
GLfloat TetrVertex[4][3], TetrNormal[4][3];
```

```
/* berilgan a,b,c nuqtaning tekislikdagi normalini hisoblash */
void getnorm (float a[3],float b[3],float c[3],float *n)
{
 float mult=0,sqr;
 int i,j;
 n[0]=(b[1]-a[1])*(c[2]-a[2])-(b[2]-a[2])*(c[1]-a[1]);
 n[1]=(c[0]-a[0])*(b[2]-a[2])-(b[0]-a[0])*(c[2]-a[2]);
 n[2]=(b[0]-a[0])*(c[1]-a[1])-(c[0]-a[0])*(b[1]-a[1]);
 /* normalning kerakli yunalishini belgilash: (0,0,0) nuqtadan*/
 for (i=0;i<3;i++)  mult+=a[i]*n[i];
 if (mult<0) for (j=0;j<3;j++) n[j]=-n[j];
}

/* tetraedr uchlari koordinatalarini hisoblash */
void InitVertexTetr()
{
 float alpha=0;
 int i;
 TetrVertex[0][0]=0;
 TetrVertex[0][1]=1.3;
 TetrVertex[0][2]=0;
 /* tetraedr asosi koordinatalarini hisoblash */
 for (i=1;i<4;i++)
 {
  TetrVertex[i][0]=0.94*cos(alpha);
  TetrVertex[i][1]=0;
  TetrVertex[i][2]=0.94*sin(alpha);
  alpha+=120.0*3.14/180.0;
 }
```

```
}

/* tetraedr tomonlari normalini hisoblash*/
void InitNormsTetr()
{
 getnorm(TetrVertex[0], TetrVertex[1],
        TetrVertex[2], TetrNormal[0]);
 getnorm(TetrVertex[0], TetrVertex[2],
        TetrVertex[3], TetrNormal[1]);
 getnorm(TetrVertex[0], TetrVertex[3],
        TetrVertex[1], TetrNormal[2]);
 getnorm(TetrVertex[1], TetrVertex[2],
        TetrVertex[3], TetrNormal[3]);
}

/* tetraedr qurish ro‘yxatini yaratish */
void MakeTetrList()
{
         int i;
  glNewList(TETR_LIST,GL_COMPILE);
  /*tetraedr tamonlarini berish */
  glBegin(GL_TRIANGLES);
  for (i=1;i<4;i++)
  {
   glNormal3fv(TetrNormal[i-1]);
   glVertex3fv(TetrVertex[0]);
   glVertex3fv(TetrVertex[i]);
   if (i!=3) glVertex3fv(TetrVertex[i+1]);
     else glVertex3fv(TetrVertex[1]);
  }
```

```
  glNormal3fv(TetrNormal[3]);
  glVertex3fv(TetrVertex[1]);
  glVertex3fv(TetrVertex[2]);
  glVertex3fv(TetrVertex[3]);

  glEnd();
  glEndList();
}

void DrawRing()
{
 /* z o‘qiga parallel joylashgan silindr (halqa) qurish.
* Ikkinchi va uchinchi parametrlar asos radiusini belgilaydi,
* to‘rtinchisi balandligini, oxirgi ikkita son
* z o‘qi bo‘ylab va atrofni bo‘lishni belgilaydi.
* Shunday ekan keyinchalik silindrning asosi
* z=0 tekislikda joylashadi   */
  gluCylinder (QuadrObj,RingRad,RingRad,RingHeight,30,2);
}
void TextureInit()
{
 char strFile[]="texture.bmp";
 AUX_RGBImageRec *pImage;

 /* bayt bo‘yicha *.bmp ga to‘g‘rilash*/
 glPixelStorei(GL_UNPACK_ALIGNMENT,1);
 /* tekstura uchun identifikator yaratish*/
 glGenTextures(1,&TexId);
 /* xotiraga tasvirni yuklash*/
 pImage = auxDIBImageLoad(strFile);
```

```
 /* tekstura xossalarini tavsiflashni boshlash */
 glBindTexture (GL_TEXTURE_2D,TexId);
 /* teksturalarni detallashtirish va initsiallash darajalarini yaratish  */
 gluBuild2DMipmaps(GL_TEXTURE_2D,3,pImage->sizeX,
                   pImage->sizeY,GL_RGB,GL_UNSIGNED_BYTE,
                   pImage->data);
 /* quadric-obyektida bu teksturani sozlashga ruxsat etish   */
 gluQuadricTexture (QuadrObj, GL_TRUE);
 /* tekstura parametrlarini berish*/
 /* s va t parametrik o‘qlar bo‘yicha tasvirni takrorlash */
 glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
 glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
 /* teksturada nuqta tanlashda qo‘shimcha kiritishlardan foydalanilmaydi   */
 glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_NEAREST);
 glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_NEAREST);
 /* tekstura va material obyektlarini birga qo‘shish*/
 glTexEnvi (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
GL_MODULATE);
}
void Init(void)
{
 InitVertexTetr();
 InitNormsTetr();
 MakeTetrList();
 /* material xossalarini aniqlash */
 glMaterialfv (GL_FRONT_AND_BACK, GL_AMBIENT, mat_amb);
 glMaterialfv (GL_FRONT_AND_BACK, GL_SPECULAR, mat_spec);
```

```
 glMaterialf  (GL_FRONT, GL_ShININESS, shininess);
 /* yoritgich xossalarini aniqlash */
 glLightfv(GL_LIGHT0, GL_DIFFUSE, light_col);
 glEnable(GL_LIGHTING);
 glEnable(GL_LIGHT0);
 /* qo‘rinmas chiziqlar va sirtlarni olib tashlash */
 glEnable(GL_DEPTH_TEST);
 /* normallarni normalashtirishni olib borish*/
 glEnable(GL_NORMALIZE);
 /* obyektlar materiallari ranglarning diffuzion aks etishi bilan farqlanadi  */
 glEnable(GL_COLOR_MATERIAL);
 glColorMaterial(GL_FRONT_AND_BACK,GL_DIFFUSE);
 /* halqa qurish uchun quadric-obyektiga ko‘rsatgich yaratish  */
 QuadrObj=gluNewQuadric();
 /* tekstura xossalarini aniqlash  */
 TextureInit();
 /* markaziy proeksiyani tayinlash  */
 glMatrixMode(GL_PROJECTION);
 gluPerspective(89.0,1.0,0.5,100.0);
/* keyinchalik faqatgina sahnadagi obyektlarni almashtirish olib boriladi */
 glMatrixMode(GL_MODELVIEW);
}
void DrawFigures(void)
{
 /* teksturalar kiritish rejimini ko‘shish (yoki yoqish) */
 glEnable(GL_TEXTURE_2D);
 /* halqa uchun rangning diffuzion tasvirini tayinlash */
 glColor3fv(mat_diff1);
 /*birlik matritsani yuklashda oldingi matritsa bilan birgalikda ko‘rsatmaslik */
 glLoadIdentity();
```

```
/* tekshirish nuqtalarini aniqlash */
gluLookAt(0.0, 0.0, 2.5,0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
/* matritsa ko‘rinishini saqlash, chunki keyingi
* halqa burilishini ko‘rsatish uchun  kerak */
glPushMatrix();
 /* yangi burilish burchaklarda bir nechta burilishlarni
 * amalga oshiradi (bu yozilgan burilish burchagi matritsasi
 * oldingi matritsa ko‘rinishidagiga nisbatan tezkor)  */
 glRotatef (-CurAng,1,1,0);
 glRotatef (CurAng,1,0,0);
 /* har bir halqani chizish uchun ularni alohida yangilash
   * kerak. Shuning uchun matritsa ko‘rinishini saqlashni
   * boshlash, keyin asl holiga qaytarish kerak */
 glPushMatrix();
  glTranslatef (0,0,-RingHeight/2);
  DrawRing();
 glPopMatrix();
 glPushMatrix();
  glTranslatef (0,RingHeight/2,0);
  glRotatef (90,1,0,0);
  DrawRing();
 glPopMatrix();
 glPushMatrix();
  glTranslatef (-RingHeight/2,0,0);
  glRotatef (90,0,1,0);
  DrawRing();
 glPopMatrix();
/* tetraedrni bo‘rish uchun matritsani asl holiga keltirish */
glPopMatrix();
/* teksturalar rejimida to‘xtatish funksiyasini o‘rnatish */
```

```
 glDisable(GL_TEXTURE_2D);
 /* bo‘rilishlarga o‘tkazish*/
 glRotatef (CurAng,1,0,0);
 glRotatef (CurAng/2,1,0,1);
 /* tetraedrni o‘z o‘qi atrofida aylantirish  uni OZ o‘qi bo‘ylab pastga siljitish */
 glTranslatef (0,-0.33,0);
 /* tetraedr uchun diffuzion tasvirlash  ranglarini berish  */
 glColor3fv(mat_diff2);
 /* tetraedr qurishni boshlash */
 glCallList(TETR_LIST);
}

void Display(void)
{
 /* joriy chuqurlik va kadr buferini  initsializatsiya qilish */
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 /*obyektlarni tuzish*/
 DrawFigures();
 /* kadr buferlari joyini o‘zgartirish */
 glutSwapBuffers();
}
void Redraw(void)
{
 /* joriy burilish burchagini kattalashtirish */
 CurAng+=1;
    /* tasvirni yaratish proseduralarini chaqirish signali (yangilash uchun) */
 glutPostRedisplay();
}
int main(int argc, char **argv)
{
```

```
  /*GLUT kutubhonasi funksiyalarini initsializatsiyalash */
  glutInit(&argc, argv);
  /* RGB formatda ranglarni aks ettiruvchi ikkilangan buferli rejimni tayinlash,
  * chuqurlik buferidan foydalanish   */
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  /* ilova oynasini hosil qilish*/
  glutCreateWindow("Example of using OpenGL");
  /* tasvirlarni yaratish funksiyasini ro‘yxatdan o‘tkazish*/
  glutDisplayFunc(Display);
  /*tasvirlarni yangilash funksiyasini ro‘yxatdan o‘tkazish */
  glutIdleFunc(Redraw);
  /* OpenGL funksiya initsializatsiyasi */
  Init();
  /* hodisalarni qayta ishlash sikli */
  glutMainLoop();
  return 0;
}
```

Ilova E. Mustaqil bajarish uchun masalalar

1. Paravoz. Ekranda harakatlanayotgan poezdni boshqaruvchi mashinist ko‘rishi mumkin bo‘lgan manzara: relslar, ustunlar, yo‘l qurilishi va boshqalar tasvirlansin.

Masalaga oydinlik. Burilishlar, strelkalar, poezd tezligini o‘zgarishi, qarama-qarshi yo‘nalishdagi poezlar va xokazolar hisobga olinsin.

2. Parda. Ekranda qandaydir teatr tomoshasini final sahnasi yakunida chapdan va ungdan yopilayotgan parda tasvirlansin. Pardada “Tamom” degan yozuv yozilgan.

3. Atom. Kimyoviy elementning atom modeli, ya’ni o‘z orbitasida aylanuvchi yadro va elektronlar tasvirlansin. Elektronlarning orbitalar bo‘yicha taqsimlanishi beriladi.

Masalaga oydinlik. Oldindan tayyorlangan faylda Mendeleyev davriy sistemasidagi barcha elementlar elektronlarning orbitalar buyicha taqsimlanishi saqlanadi. Foydalanuvchi faqat kimyoviy element tartib raqamini yoki belgilanishini kiritadi.

4. Mayatnik. Ekranda harakatlanayotgan matematik mayatnik tasvirlansin. Matematik mayatnikning uzunligi boshlang‘ich holati beriladi.

Masalaga oydinlik. Xavoning qarshiligi e'tiborga olinsin. Mayatnikning o‘rniga arg‘imchoq tasvirlansin.

5. Bayroq. Shamolda xilpirayotgan bayroqni tasvirlang (masalan yurtimizniki).

6. Reklama. Ixtiyoriy mahsulot yoki xizmat turining dinamik reklamasi ishlab chiqilsin.

7. Xisob-kitoblar. Qo‘shish va ayirish operatsiyalarini namoyish qiluvchi buxgalterlik hisob-kitobi ishlab chiqilsin. O‘n va arifmetik amallar belgisi klaviaturadan kiritiladi. Razryad bo‘yicha olib boriladigan amallar ko‘rinib turishi uchun kutish vaqti hisobga olinsin.

8. Ekranda bayram salyuti, ya’ni (bir nechta stvoldan otilgan) raketalarning uchishi, portlashi va oskolkalarning yerga tushishi tasvirlansin. Rangli effektlarni har kim o‘zi xoxishi bo‘yicha tovush yordamida berishi mumkin.

9. Oyning orbita bo‘yicha aylanib chiqish sikli 28 sutkani tashkil qiladi. Harakatlanish orbitasini doimiy deb hisoblash mumkin. Oyning harakatlanish jarayonida tug‘ilishi, o‘sishi va “eskirishi” yulduzli osmonda tasvirlansin.

10. Ekranda 10 ta aylanadan va har biriga mos ravishda qo‘yilgan aylanadan iborat (sport yoki armiya) nishonlar tasvirlangan. Berilgan vaqtda yoki kompyuter tugmasi bosilganda o‘q otiladi. Natijada nishonda o‘q tekkan joy bo‘lib qoladi. Bu jarayon zahiradagi o‘q tamom bo‘lguncha yoki tugma bosilguncha (to‘plangan ballarni hisoblash davom etadi). Foydalanuvchi o‘qlarni nishonga tekkizish nuqtalarini o‘zi berishi mumkin.

Masalaga oydinlik. Nishonga urish nuqtasini sichqoncha yoki klaviatura yordamida belgilanadi. Oldin urilgan nuqtalarga bir nechta o‘q tegishi mumkin.

11. Qimor. Kompyuter-olib boruvchi, odam-o‘yin ishtirokchisi rolida. Ekranda uchta idish, ularning bittasini ostida sharik (o‘yin boshlangan paytda u ko‘rinadi) tasvirlansin. Olib boruvchi berilgan tezlikda sharni idishlar ostida almashtiradi. Almashtirish tugaganidan so‘ng foydalanuvchi sharning qaerdaligini topish kerak.

TEST SAVOLLARI

1. “Kompyuter grafikasi”ni qanday tasvirli ko‘rinishda ifodalash mumkin?

A. belgilar->tasvir
B. tasvir->belgilar
C. belgilar ->belgilash
D. tasvir ->tasvir
E. to‘g‘ri javob yo‘q

2. Qaysi almashtirishlar matritsalari simmetrik?

A. kuchish, masshtablash
B. masshtablash, akslantirish
C. masshtablash, burish
D. burish, akslantirish
E. barcha almashtirishlar

3. Qaysi almashtirishlarda matritsalar simmetrik emas?

A. masshtablash, akslantirish
B. akslantirish, burish
C. masshtablash, burish
D. kuchish, burish
E. to‘g‘ri javob yo‘q

4. Qaysi almashtirishning matritsasi determinanti 1 ga teng?

A. burish
B. masshtablash
C. kuchish
D. akslantirish
E. to‘g‘ri javob yo‘q

5. Qaysi almashtirishning matritsasi determinanti -1 ga teng?

A. burish
B. masshtablash
C. kuchirish

D. akslantirish
E. to‘g‘ri javob yo‘q

6. Qaysi Platon jismining 20 ta yoqi, 30 ta qirrasi, 12 ta uchi mavjud?

A. geksaedr
B. tetraedr
C. oktaedr
D. dodekaedr
E. ikosaedr

7. Qaysi Platon jismining 12 ta yoqi, 30 ta qirrasi, 20 ta uchi mavjud?

A. tetraedr
B. geksaedr
C. oktaedr
D. dodekaedr
E. ikosaedr

8. Qaysi Platon jismining 8 ta yoqi, 12 ta qirrasi, 6 ta uchi mavjud?

A. geksaedr
B. oktaedr
C. tetraedr
D. dodekaedr
E. ikosaedr

9. Platon jismlarini yoqi (Yo), qirrasi (Q) va uchi (U) orasida qanday bog‘liqlik (Eyler tenglik) mavjud?

A. Yo+U=Q+1
B. Yo+U=Q+3
C. Yo+U=Q
D. Yo+U=Q+4
E. Yo+U=Q+2

10. Keltirilgan proeksiyalardan qaysilari parallel hisoblanadi?

A. ortografik proeksiyalash
B. aksonometrik proeksiyalash

C. kavale proeksiyalash
D. kabine proeksiyalash
E. barcha javoblar to‘g‘ri

11.Keltirilgan proeksiyalardan qaysilari markaziy proeksiyalash hisoblanadi?
A. ortografik proeksiyalash
B. aksonometrik proeksiyalash
C. kavale proeksiyalash
D. bir nuqtali proeksiyalash
E. kabine proeksiyalash

12.Proeksiyalash matritsasi qanday bo‘ladi?
A. determinanti 0 ga teng emas
B. simmetrik
C. nosimmetrik
D. determinanti 0 ga teng
E. birlik matritsa

13.Ikki qo‘shni piksellar 8 bog‘lamli hisoblanadi, agarda P1(x1,y1) va P2(x2,y2) nuqtalar orasida quyidagi bog‘liqlik mavjud bo‘lsa.
A. |x1-x2|<=1, |y1-y2|<=1
B. |x1-x2|+|y1-y2|<=1
C. |x1-x2|+|y1-y2|<=2
D. |x1-y1|<=1, |x2-y2|<=1
E. |x1-y1|+|x2-y2|<=1

14.Ikki qo‘shni piksellar 4 bog‘lamli hisoblanadi, agarda P1(x1,y1) va P2(x2,y2) nuqtalar orasida quyidagi bog‘liqlik mavjud bo‘lsa.
A. |x1-x2|<=1, |y1-y2|<=1
B. |x1-x2|+|y1-y2|<=1
C. |x1-x2|+|y1-y2|<=2
D. |x1-y1|<=1, |x2-y2|<=1
E. |x1-y1|+|x2-y2|<=1

15. Keltirilgan algoritmlardan qaysi biri kesmani to‘g‘ri to‘rtburchak bilan kesishga kiradi?

A. Brezenxeym algoritmi
B. Sazerland-Xogdman algoritmi
C. Sazerland-Koxen algoritmi
D. Roberts algoritmi
E. Varnok algoritmi

16. Kesmani rastr algoritmi orqali chizuvchi algoritmni aniqlang.

A. Sazerland-Koxen algoritmi
B. Roberts algoritmi
C. Appel algoritmi
D. Brezenxeym algoritmi
E. Sazerland-Xogdman algoritmi

17. Qaysi splayn egri chizig‘i ikki tayanuvchi nuqta va shu nuqtalardagi urunuvchi vektorlar bilan beriladi?

A. Beze egri chizig‘i
B. Ermit egri chizig‘i
C. B-splayn egri chizig‘i
D. Beta-splayn egri chizig‘i
E. barcha splayn egri chiziqlari

18. Qaysi splayn egri chiziqlari 4 ta tayanuvchi nuqtalar bilan beriladi?

A. Ermit, Beze, B-splayn
B. B-splayn , Ermit, Beta-splayn
C. Beze, B-splayn , Veta-splayn
D. Beze, Ermit, Beta-splayn
E. to‘g‘ri javob yo‘q

19. Keltirilgan olib tashlash algoritmlardan qaysi birida obyekt nuqtasidan chizma tekisligigacha bo‘lgan masofa haqida ma’lumot har bir piksel uchun hisobga olinadi?

A. Roberts algoritmi
B. Appelya algoritmi

C. Varnok algoritmi
D. z-bufer algoritmi
E. tartiblash algoritmi

20. Keltirilgan qaysi olib tashlash algoritmida sonli-ko‘rinmaslik tushunchasi hisobga olinadi?

A. Appel algoritmi
B. z-bufer algoritmi
C. Roberts algoritmi
D. Varnok algoritmi
E. satrma-satr tekshirish algoritmi

21. Keltirilgan qaysi uzoqlashtirish algoritmida maydonni to‘g‘ri burchakli 4 qismga bo‘lish ishlatiladi?

A. z-bufer algoritmi
B. Roberts algoritmi
C. Appel algoritmi
D. Varnok algoritmi
E. chuqurligi bo‘yicha tartiblash algoritmi

22. Poligonal setkani qaysi buyash metodida tasvir realroq bo‘ladi?

A. Guro metodi
B. Fong metodi
C. diffuzion akslantirish
D. aks
E. barcha javoblar to‘g‘ri

23. Keltirilgan rang modellaridan qaysi oddiy foydalanuvchilar uchun mo‘ljallangan?

A. RGB
B. CMY
C. CMYK
D. HSV
E. barcha javoblar to‘g‘ri

24. Keltirilgan rang modellaridan qaysilari apparatli ta’minotlar uchun mo‘ljallangan?

A. RGB, HSV
B. CMY, HLS
C. HSV, HLS
D. RGB, CMY
E. CMYK, HSV

25. Beze va B-splayn tekisligining elementar bo‘lagi nechta tayanuvchi nuqtalar bilan beriladi?

A. 16
B. 4
C. 10
D. 20
E. 8

26. OpenGL ga berilgan to‘g‘ri ta’rifni belgilang?

A. ikki va uch ulchovli grafika sohasida ilovalar yaratish uchun mo‘ljallangan amaliy dasturiy interfeys
B. ochiq grafik kutubxonaga ega bo‘lgan amaliy dasturiy interfeys
C. kompyuter grafikasi sohasida ilovalar yaratish uchun mo‘ljallangan amaliy dasturiy interfeys
D. kompyuter grafikasi sohasida tasvirlarni vizuallashtirish uchun mo‘ljallangan amaliy dasturiy interfeys
E. barcha javoblar to‘g‘ri

27. OpenGL kutubxonasining o‘ziga xos xususiyatlari belgilang?

A. Barqarorlik, ishonchlilik, qo‘llashning osonligi
B. Barqarorlik, ishonchlilik va uzatuvchanlik, qo‘llashning osonligi, ma’lumotlardan birgalikda foydalanish
C. Barqarorlik, ishonchlilik va uzatuvchanlik, qo‘llashning osonligi
D. Barqarorlik, ishonchlilik, qo‘llashning osonligi, ma’lumotlardan birgalikda foydalanish

E. Barqarorlik, uzatuvchanlik, qo‘llashning osonligi, ma’lumotlardan birgalikda foydalanish

28. OpenGL ning asosiy imkoniyatlarini belgilang?

A. Primitivlarni tavsiflash, ranglar manbasini tavsiflash, atributlarni tayinlash, vizuallashtirish va geometrik o‘zgartirish funksiyalari
B. Primitivlarni tavsiflash, ranglar manbasini tavsiflash, atributlarni tayinlash, vizuallashtirish, geometrik o‘zgartirish va tasvirlash funksiyalari
C. Primitivlarni tavsiflash, ranglar manbasini tavsiflash, atributlarni tayinlash, geometrik o‘zgartirish va tasvirlash funksiyalari
D. Primitivlarni tavsiflash, atributlarni tayinlash, vizuallashtirish, geometrik o‘zgartirish va tasvirlash funksiyalari
E. Primitivlarni tavsiflash, atributlarni tayinlash, vizuallashtirish, geometrik o‘zgartirish funksiyalari

29. OpenGL funksiyalari qanday texnologiyasi asosida qo‘rilgan?

A. Mijoz texnologiyasi
B. Server texnologiyasi
C. Mijoz-server texnologiyasi
D. Avtomatlashtirilgan loyihalash texnologiyasi
E. Dasturlash texnologiyasi

30. OpenGL kutubxonasi qaysi dasturlash tizimlarida avtomatik o‘rnatiladi?

A. Microsoft Visual C++, S##, DevC++
B. Microsoft Visual C++, S##, Borland C++
C. Microsoft Visual C##, Delphi, DevC++, Borland C++
D. Microsoft Visual C##, DevC++, Borland C++
E. Microsoft Visual C++, DevC++, Borland C++

31. Nuqta, chiziq, ko‘pburchak va boshqa geometrik obyektlar OpenGL da qanday funksiya sifatida qaraladi?

A. Primitivlarni tavsiflash
B. Atributlarni tayinlash
C. Geometrik obyektlarni berish

D. Obyektni vizuallashtirish
E. Protsedurali funksiya

32. OpenGL da virtual fazoda kuzatuvchi holatini belgilash qaysi funksiyaga mansub?

A. Primitivlarni tavsiflash
B. Atributlarni tayinlash
C. Vizuallashtirish
D. Geometrik o‘zgartirish
E. Prosedurali funksiya

33. OpenGL dan foydalanganda ekranda nima hosil bo‘lishini qaysi funksiya belgilaydi?

A. Primitiv
B. Atribut
C. Ranglar
D. Geometrik obyektlar
E. Funksiyalar

34. OpenGLda tasvirlarni yangilash funksiyasi qanday vazifalarni bajaradi?

A. OpenGL buferlarini tozalash
B. Kuzatuvchini holatini o‘rnatish
C. Geometrik obyektlarni chizish
D. Geometrik obyektlar o‘zgartirish
E. Barcha javoblar to‘g‘ri

35. OpenGLda tasvirlanyotgan obyektni ekranda chiqarish usulini nima belgilaydi?

A. Primitiv
B. Atribut
C. Ranglar
D. Funksiyalar
E. Geometrik obyektlar

36. GL kutubxonasining barcha o‘zgarmaslari qanday qo‘shimcha bilan boshlanadi?

A. GL_
B. GL
C. gl/gl.h
D. void GL
E. #include <gl/gl.h>

37. GLUT kutubxonasining barcha o‘zgarmaslari qanday qo‘shimcha bilan boshlanadi?

A. GLUT _
B. GLUT
C. gl/glut.h
D. void GLUT
E. #include <gl/glut.h>

38. GL_DEPTH_BUFFER_BIT buyrug‘i qanday vazifani bajarad?

A. Bufer rangini tozalash
B. Bufer rangini to‘ldirish
C. Bufer chuqurligini tozalash
D. Bufer chuqurligini to‘ldirish
E. To‘g‘ri javob berilmagan

39. GLU kutubxonasining barcha buyruqlari qanday qo‘shimcha bilan boshlanadi?

A. GLU_
B. GLU
C. gl/glu.h
D. void GLU
E. #include <gl/glu.h>

40. OpenGL da kadr buferini tozalash uchun qanday buyruq ishlatiladi?

A. glClearColor
B. glClear

C. glVertex
D. glTranlsate
E. GL_COLOR_BUFFER_BIT

41. OpenGL da kadr buferini to‘ldirish uchun qanday buyruq ishlatiladi?

A. glClearColor
B. glClear
C. glVertex
D. glTranlsate
E. GL_DEPTH_BUFFER_BIT

42. OpenGL da uchlarning joriy rangini tayinlash uchun qanday buyruq ishlatiladi?

A. glColor
B. glClear
C. glVertex
D. glTranlsate
E. glColor3fv

43. OpenGL da uchlarning koordinatasini tayinlash uchun qanday buyruq ishlatiladi?

A. glClearColor
B. glClear
C. glVertex*
D. glTranlsate*
E. glPushMatrix()

44. OpenGL da kadr buferini qora rang bilan to‘ldirish qaysi javobda to‘g‘ri berilgan?

A. glClearColor(1, 0, 0, 0)
B. glClearColor(0, 0, 0, 1)
C. glClearColor(0, 1,1, 1)
D. glClearColor(0, 1, 1, 0)
E. glClearColor(1, 1, 1, 0)

45.OpenGL da obyektni burish uchun qanday buyruq ishlatiladi?

A. glRotate*
B. glScale*
C. glVertex*
D. glTranlsate*
E. glDisable*

46.OpenGL da obyektni ko‘chirish uchun qanday buyruq ishlatiladi?

A. glRotate*
B. glScale*
C. glDisable*
D. glVertex*
E. glTranlsate*

47.OpenGL da obyektni masshtablashtirish uchun qanday buyruq ishlatiladi?

A. glRotate*
B. glScale*
C. glVertex*
D. glDisable*
E. glTranlsate*

48.glVertex2* buyrug‘i qanday qiymatlarni qabul qiladi?

A. x va y qiymatlarini qabul qiladi, z koordinatasiga 0, w koordinatasiga 1 qiymatini kiritadi
B. x va y qiymatlarini qabul qiladi, z va w koordinatalariga 0 qiymatini kiritadi
C. x va y qiymatlarini qabul qiladi, z va w koordinatalariga 1 qiymatini kiritadi
D. x va y qiymatlarini qabul qiladi, z koordinatasiga 1, w koordinatasiga 0 qiymatini kiritadi
E. A va B javoblari to‘g‘ri

49.OpenGL da uchburchakli primitivlarning qayday turlari mavjud?

A. GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN
B. GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_ LOOP
C. GL_TRIANGLES, GL_TRIANGLE_ FAN, GL_TRIANGLE_ LOOP
D. GL_TRIANGLES, GL_TRIANGLE_ FAN, GL_TRIANGLE_ LOAD
E. GL_TRIANGLES, GL_TRIANGLE_ LOOP, GL_TRIANGLE_ LOAD

50. void glPolygonMode (GLenum *face*, Glenum *mode***) buyrug‘ida** *mode* **parametrining vazifasi?**

A. Ko‘pburchakni qanday akslanishini belgilaydi
B. Ko‘pburchak tipini o‘rnatishda ishlatiladi
C. Ko‘pburchakni ekranda qanday tasvirlanishini belgilaydi
D. Ko‘pburchakni faqat uchlarini tasvirlaydi
E. yorug‘likni hisobga olib ko‘pburchaklar joriy rang bilan buyab chiqiladi

51. Bir qancha uchlarni glVertex*() buyrug‘ini chaqirmasdan foydalanish imkonini beruvchi buyruq to‘g‘ri berilgan javobni belgilang?

A. void glVertexPointer(GLint *size*, GLenum *type*, GLsizei *stride*, void* *ptr*)
B. void glNormalPointer (GLenum type, GLsizei stride, void *pointer)
C. void glColorPointer (GLint size, GLenum type, GLsizei stride, void *pointer)
D. void glEnableClientState (GLenum array)
E. void glDrawArrays (GLenum mode, GLint first, GLsizei count)

52. Joriy tipdagi matritsa elementlarini aniqlash uchun qaysi buyruq chaqiriladi?

A. void glLoadMatrix [f d] (GLtype *m)
B. void glLoadIdentity (void)
C. void glPushMatrix (void)
D. void glPopMatrix (void)
E. void glMultMatrix [f d] (GLtype **m*)

53. Joriy matritsani boshqa matritsaga ko‘paytirish uchun qaysi buyruq chaqiriladi?

A. void glLoadMatrix [f d] (GLtype *m)
B. void glLoadIdentity (void)
C. void glPushMatrix (void)
D. void glPopMatrix (void)
E. void glMultMatrix [f d] (GLtype **m*)

54. OpenGL da tuman effektini yoqish uchun qanday buyruqni chaqirish zarur?

A. glEnable(GL_FOG)
B. glEnable (GL_LIGHTING)
C. glLightModel

D. GL_SPOT_DIRECTION
E. GL_MAX_LIGHT

55. OpenGL da tekstura bilan ishlash uchun qanday harakatlar ketma-ketligini amalga oshirish zarur?

A. Tasvirni tanlash va uni kerakli formatga keltirish
B. Tasvirni OpenGL ga uzatish
C. Tekstura obyektga qanday qo‘yilishini va u bilan qanday munosabatda bo‘lishini aniqlab olish
D. Teksturani obyekt bilan bog‘lash
E. Barcha javoblar to‘g‘ri

ADABIYOTLAR

1. Боресков А.Е., Шикин Е.В. Компютерная графика : учебник и практикум для прикладного бакалавриата. – М.: Издательсво Юрайт, 2016. – 219 с.
2. Боресков А. Расширения OpenGL. – СПб.: БХВ-Петербург. 2006. – 288 с.
3. Васильев С.А. OpenGL. Компьютерная графика. Учебное пособие. - Тамбов: Изд-во Тамб. гос. техн. ун-та, 2005. – 80 с.
4. Гайдуков С.А. OpenGL. Профессиональное программирование трехмерной графики на C++. – СПб.: БХВ-Петербург. 2004. – 736 с.
5. Компьютерная графика: Полигональные модели / А.В. Боресков, Е.В. Шикин. – М.: Изд-во «Диалог-МИФИ», 2005. – 461 с.
6. Миронов Д. Компьютерная графика в дизайне: учебник. – СПб.: БХВ-Петербург, 2008. – 560 с.
7. Петров М., Молочков К. Компьютерная графика. Учебник. – Питер, 2002. – 736 с.
8. Петров М.Н. Компьютерная графика. – СПб.: Питер. 2011. – 544 с.
9. Порев В.Н. Компьютерная графика. – СПб.: БХВ-Петербург, 2005. – 432 с.
10. Рейнбоу В. Компьютерная графика. Энциклопедия. Питер, 2003. – 876 с.
11. Рихсибоев Т. Компьютер графикаси. – Тошкент: Ўзбекистон ёзувчилар уюшмаси «Адабиёт» жамғармаси нашриёти, 2006. – 154 б.
12. Херн, Дональд, Бейкер, М.Паулин. Компьютерная графика и стандарт OpenGL, 3-е издание.: Пер. с анг. –М.: Издательский дом “Вильямс”, 2005. – 1168 с.
13. Хилл Ф. OpenGL: Программирование компьютерной графики. – СПб.: Питер, 2002. – 1089 с.
14. Шикин А.В., Боресков А.В. Компьютерная графика. Полигональные модели. Москва, ДИАЛОГ-МИФИ, 2001. – 464 с.
15. Шрайнер Дэйв, Ву М., Нейдер Дж., Девис Том. OpenGL. Руководство по программированию. Библиотека программиста. 4-е изд. Питер, 2006. – 624 с.
16. Эйнджел Эдвард. Интерактивная компьютерная графика. Вводный курс на базе OpenGL, 2 изд.: Пер. с анг. – М.: Издательский дом “Вильямс”, 2001. – 592 с.: ил.
17. Яцюк О.Г., Романычева Э.Т. Компьютерные технологии в дизайне. Эффективная реклама. – СПб.: БХВ-Петербург, 2002. – 432 с.
18. David Salomon. The Computer Graphics Manual. Volume 1. – Springer, 2012. – 1564 p.

19. Donald Hearn, M. Pauline Baker. Computer graphics. C version. – Prentice Hall, 1997. – 662 p. – 2nd edition. – ISBN: 0135309247.
20. Nazirov Sh.A., Nuraliyev F.M., To'rayev B.Z. Komputer grafikasi va dizayn. O'quv qo'llanma. – T.: Fan va texnologiya, 2015. – 256 b.
21. Mamarajabov M., Ashurov M., Umarova U. CorelDRAW dasturi va uning imkoniyatlari. Metodik qo'llanma. – Toshkent: TDPU, 2011. – 82 b.
22. Nazirov Sh.A., Nuraliyev F.M., Aytmuratov B.Sh. Rastr va vector grafika. – T.: G'.G'ulom, 2007. – 192 b.
23. Nazirov Sh.A., Nuraliyev F.M., Tillayeva M.A. Uch o'lchovli modellashtirish. – T.: "Ilm ziyo", 2012. – 144 b.
24. Гребенников К.А. Компьютерная графика как средство профессиональной подготовки специалистов-дизайнеров. (на материалах среднего профессионального образования): Автореф. дис. ... канд. пед. наук. – Воронеж: РУДН, 2002. – 28 с.
25. Крайнова О.А. Проектирование методической системы обучения студентов дисциплине «Компьютерная графика» на примере специальности 030100 «Информатика»: Автореф. дис. ... канд. пед. наук. – Москва: МГУ, 2004. – 24 с.
26. Нодельман Л.Я. Технология обучения студентов художественно-графического факультета компьютерной графике: Автореф. дис. ... канд. пед. наук. – Москва: МГУ, 2000. – 275 с.
27. Петрова Н.П. Компьютерная графика и анимация как средство медиа-образования: Автореф. дис. ... канд. пед. наук. – Москва: МПГУ, 1997. – 156 с.
28. Чернякова Т.В. Методика обучения компьютерной графике студентов вуза: Дисс. ... канд. пед. наук. – Екатеринбург: РГППУ, 2010. – 201 с.
29. Эминов А.Ғ. Бўлажак ўқитувчиларнинг компьютер графикаси бўйича компетентлигини ривожлантириш методикаси ("Информатика ва ахборот технологиялари" ўқув фани мисолида): Дис. ... пед. фан. номз. – Тошкент, 2012. – 139 б.

GLOSSARIY

Affin almashtirishlari – chiziqli almashtirish, masalan, koordinatani almashtirish.

Alfa-kanal – shaffoflik xarakteristikasi.

Amaliy dasturlar interfeysi (Application Program Interface – API) - o‘zlarining dasturlarini tegishli operatsion tizimlar bilan uyg‘unlashuvi uchun dasturiy ta’minot ishlab chiquvchilar amal qilishlari kerak bo‘lgan vazifalar yig‘masining spetsifikatsiyasi.

Animatsiya (animation) – bir necha tasvir yoki kadrlarni ko‘rsatish orqali kino sifatida qabul qilinadigan kadrlar ketma-ketligi.

Antialiasing (antialiasing) – alohida burchakli piksellar rangini belgilanishini silliqlash yo'li orqali rastrli tasvirlarning pog'onali effektlarini bartaraf etish.

Avtomatlashgan loyihalash tizimi, SAPR (Computer Aided Design, CAD) – kompyuter yordamida murakkab ob'ektlarni loyihalash uchun mo'ljallangan tizim.

Bitli massiv – xotira yoki diskda saqlanadigan rastr.

BMP – fayllarning rastrli grafik formati, Windows operasion tizimi muhitidagi dasturlarda keng ishlatiladi. Tasvir bitli massiv shaklida saqlanadi.

CMY (Cyan, Magenta, Yellow) – bu rang modeli subtraktiv, ya’ni biror kerakli bo‘lgan rangni hosil qilish uchun asosiy ranglar oq rangdan ajraladi.

CMYK (Cyan, Magenta, Yellow, blacK) – to’rt hil rangga asoslangan subtraktiv rang modeli.

Diffuzion qaytish – yorug‘likning barcha yunalishlar bo‘yicha tekis tarqalishi.

GIF (Graphics Interchange Format) - pikselli grafik tasvirni global kompyuter tarmog'iga uzatish uchun maxsus ishlab chiqilgan format.

Grafik qurilmalar interfeysi (Graphic Device Interface, GDI) – Windows operasion tizimining quyi tizimi.

Kompyuter grafikasi – kompyuter yordamida yaratiladigan tasvir.

JPEG (Joint Photographic Experts Group) – rastli tasvirlar uchun ma'lumotlarni samarali zichlovchi standart fayl formati.

OpenGL (Open Graphics Library) – ikki va uch o‘lchovli grafika sohasida ilovalar yaratish uchun ancha keng tarqalgan, apparatga bog‘liq bo‘lmagan amaliy dasturiy interfeys.

Oyna (window) – grafik natijalar tekisligi qismi. Windows operasion tizimida bu fundamental tushuncha hisoblanadi.

Palitra – aniq bir grafik tasvir uchun foydalanadigan biror bir rang modeli asosida tashkil qilingan ranglar to‘plami.

Piksel (pixel) – rastr elementi.

Platon jismlari – barcha yoqlari to‘g‘ri ko‘pburchaklardan va barcha uchlariga tegishli burchaklar o‘zaro teng bo‘lgan qavariq ko‘pyoqliklar.

Poligonal setka – bu fazoviy obyektni tasvirlovchi o‘zaro bog‘liq balandliklar, qirralar va yoqlar (ko‘pburchaklar) to‘plami.

Proeksiya – n o‘lchovli koordinatalar sistemasida berilgan nuqta(lar)ni n dan kam bo‘lgan o‘lchovli koordinatalar sistemasidagi nuqta(lar)ga geometrik almashtirish.

Rang chuqurligi – bitta pikselning rangi haqida ma’lumot saqlash uchun ajratilgan bitlar soni bilan aniqlanadi.

Rang modeli – poligrafiyada yoki monitorning rangli kanallarida foydalanish mumkin bo‘lgan bo‘yoqlarning chegaralangan soni yordamida ranglar namoyish qilinadigan tizim.

Rasterizasiyalash (rasterization) – tasvir elementlarini vektorli tavsiflash asosida rastr tasvir yaratish.

Rendering (rendering) – axborotlarni grafik ko'rinishda tasvirlash jarayoni.

RGB (Red, Green, Blue) – bu rang modeli additiv, ya’ni biror bir kerakli rangni hosil qilish uchun uning asosiy ranglari yig‘iladi.

Shrift (front) – kompyuter tizimlari, poligrafiyada matnni aks ettirish uchun mo'ljallangan simvolli belgilar to'plami.

Splayn (spline) – murakkab shakllar sirtini yoki chiziqlar qismini aproksimasiyalash uchun ishlatiladigan maxsus tipdagi egri chiziq yoki sirt. Bir qancha bog'liq splaynlar yagona bir butun sifatida shaklni ifodalaydi.

Tekstura (texture) – ob'ektni buyash usuli, ko'proq rastli obrazlar ko'rinishida ishlatiladi. Ko'proq rastrli obrazlar ko'rinishida ishlatiladi.

TIFF (Target Image File Format) – rangli tasvirlarni skanerlashdan olingan natijalarni saqlash uchun universal format.

Vektor grafika – alohida ob'ektlarni vektorli tavsiflanishi asosida tasvirning yaratilishi.

Videoadapter – bu grafik qurilma bo'lib, bevosita uning yordamida kompyuter monitori ekranida tasvir shakllanadi.

Virtual reallik (virtual reality) – haqiqatan mavjud bo‘lmaydi, biroq kompyuter tizimlari insonning ko‘rish, eshitish va boshqa hissiyot organlariga ta’sir qilib, ushbu dunyodan erkin foydalanish illuziyasini keltirib chiqaradi.

MUNDARIJA

TATU ilmiy kengashida
ko‘rib chiqilgan va
nashrga tavsiya etilgan
« 28 » aprel 2022
Bayonnoma № 8(…)

TATU ilmiy-uslubiy
kengashida ko‘rib chiqilgan
va nashrga tavsiya etilgan
« 27 » aprel 2022
Bayonnoma № 8(…)

«Televizion texnologiyalar»
fakulteti ilmiy-uslubiy
kengashida muhokama qilingan
va ma’qullangan
« 26 » aprel 2022
Bayonnoma № 8

«Audiovizual texnoligoyalar»
kafedrasi majlisida muhokama
qilingan va tavsiya etilgan
« 19 » aprel 2022
Bayonnoma № 7