ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық университеті

Ақпараттық технологиялар институты

Желілер мен жүйелердің программалық қамтамасы кафедрасы

Өскенбаева Р.Қ., Нұралықызы С.

СТУДЕНТТІҢ ПӘНДІК ОҚУ-ӘДІСТЕМЕЛІК КЕШЕНІ

«Интерактивті графикалық жүйелер» пәні бойынша
5В070400 – Есептеу техникасы және программалық қамтама мамандығы үшін

Алматы 2010

Құрастырғандар: техника ғылымдарының докторы, профессор Өскенбаева Р.Қ., оқытушы Нұралықызы С. «Интерактивті графикалық жүйелер» пәні бойынша 5В070400 – Есептеу техникасы және программалық қамтама мамандығы үшін пәннің оқу-әдістемелік кешені. – Алматы: ҚазҰТУ, 2010. – 100 б.

Аңдатпа

Осы кезге дейін интерактивті компьютерлік графика сөз тіркесі көп кездесіп келеді. Компьютерлік жүйемен графика құру және адаммен сұқбат жүргізу деп түсінуге болады. Бұрынғы кезде жүйенің бума режимінде жұмыс істеудің сұқбат тәсілі өркендемеген. Қазіргі уақытта кез-келген программаны компьютерлік интерактивті графика деп санауға болады.

60 жылдары бірінші тарихи интерактивті жүйелер автоматтандырылған жоба жүйесі деп айтылды. Оларды компьютер мен программалық қамтама эволюциясының бастапқы кезеңі болып саналады. Пайдаланушы интерактивті компьютерлі графика жүйесінде дисплейде бейнені қабылдайды және объект жазбасына өзгеріс енгізеді. Мұндай өзгеріске жеке элементтерді енгізу және редактірлеу, кез-келген параметрлер үшін сандық мән беру, сонымен қатар бейнені қабылдау негізінде мәліметті енгізу операциясы кіреді.

© Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық университеті, 2010

1 ПӘННІҢ ОҚУ БАҒДАРЛАМАСЫ

1.1 Оқытушылар туралы мәліметтер:

Пәнді жүргізетін оқытушылар: Өскенбаева Раиса Қабиқызы, техника ғылымдарының докторы, профессор;

Нұралықызы Салтанат, оқытушы.

Байланыс телефондары: тел.: 257-71-92, ішкі тел.: 71-92

Кафедрада болу уақыты: сағат 9^{00}–17^{00}

1.2 Пән туралы мағлұматтар:

Аты: «Интерактивті графикалық жүйелер»

Кредит саны: 3

Өткізетін орны: «ЖмЖПҚ» кафедрасы, Басты оқу ғимараты, 1023 ауд.

Кесте 1.1

Оқу жоспарынан көшірме

Курс	Семестр	Кредит	Бір аптадағы академиялық сағат						Бақылау түрі
			Дәріс	Зерт. сабақ	Практ. сабақ	СӨЖ	СОӨЖ	Барлық	
2	4	3	2	1		3	3	9	Емтихан

1.3 Пререквизиттер

Берілген пәнді оқып үйрену үшін керек пәндер:

– ақпараттану;

– программалау технологиясы мен тілдері.

1.4 Постреквизиттер

Берілген пән келесі пәндерді оқып үйренуде қолданылады:

– интернет-технологиялар;

– программалаудың жаңа технологиялары;

– таратылған жүйелер технологиясы;

– мультимедиа-технологиялар.

1.5 Пәннің қысқаша сипаттамасы

Пәнді оқытудың мақсаты

«Интерактивті графикалық жүйелер» пәнінің оқыту мақсаты студенттерге машиналық графиканың теориялық негіздерін, алфавитті-цифрлік режимдегі графикалық режимнің ерекшеліктері мен айырмашылықтарын, инструментальді ортасын және қазіргі заманғы графикалық қосымшаларды өңдеуге арналған технологияларды оқып үйрету болып табылады.

Пәнді оқытудың міндеттері

Пәнді оқытудың міндеттері:

– 2D және 3D графиканың негізгі алгоритмдерін білу;

– видеоадаптердің төменгі, орташа және жоғары деңгейдегі әдісі мен тәсілдерін оқып үйрену;

– графикалық программалардың интерфейсін құруды, жазықтықта және кеңістікте графикалық сценасын программалауды білу қабылетін оқып үйрену.

1.6 Тапсырмалардың түрлері мен тізімі және оларды орындау кестесі

Төменгі форма түрінде тьютор кесте құрады (кесте 1.2).

Кесте 1.2

Тапсырмалар түрлері және олардың орындалу түрі

Бақылау түрі	Жұмыс түрі	Жұмыс тақырыбы	Беті көрсетілген ұсынылған әдебиетке сілтеме	Тапсыру мерзімі
Ағымдық бақылау 1	Зертханалық жұмыс №1	CorelDRAW-да қарапайым логотиптерін құру	5 [500-508 б.] 13 [29-80 б.]	1 апта
Ағымдық бақылау 2	Өзіндік жұмыс №1	Векторлық графиканың негізгі принциптері	5 [500-508 б.] 13 [29-80 б.]	2 апта
Ағымдық бақылау 3	Зертханалық жұмыс №2	CorelDRAW-да үшөлшемді объект құру	5 [509-539 б.] 13 [81-130 б.]	3 апта
Ағымдық бақылау 4	Өзіндік жұмыс №2	CorelDRAW-да мәтінмен жұмыс. Фигуралық мәтін құру	5 [650-678 б.] 13 [407-478 б.]	4 апта
Ағымдық бақылау 5	Зертханалық жұмыс №3	Adobe Photoshop-та бөлініп шығу жабдығымен жұмыс	5 [650-678 б.] 13 [407-478 б.]	5 апта
Ағымдық бақылау 6	Өзіндік жұмыс №3	Растрлық графика негіздері. Көп қабатты бейнелермен жұмыс	5 [343-374 б.] 13 [283-302 б.]	6 апта
Ағымдық бақылау 7	Зертханалық жұмыс №4	Adobe Photoshop мәтінімен жұмыс	5 [343-374 б.] 13 [283-302 б.]	7 апта
Аралық бақылау 1	1 модуль аттест.сұрақтар жауабы	1 модуль бойынша бақылау өткізуге арналған сұрақтар	1-13	8 апта
Ағымдық бақылау 9	Зертханалық жұмыс №5	Жазықтықта берілген координаттар бойынша объектілерді түрлендіу	5 [377-382 б.] 13 [303-333 б.]	9 апта
Ағымдық бақылау 10	Өзіндік жұмыс №4	Координаттарды түрлендіру	5 [377-382 бет] 13 [303-333 б.]	10 апта
Ағымдық бақылау 11	Зертханалық жұмыс №6	Берілген координаттар бойынша объектілерді түрлендіу	5 [383-407 б.] 13 [407-438 б.]	11 апта
Ағымдық бақылау 12	Өзіндік жұмыс №5	Объектілерді түрлендіру	5 [407-424 б.] 13 [473-496 б.]	12 апта
Ағымдық бақылау 13	Зертханалық жұмыс №7	OpenGl программасын қолданып графикалық программалар құру	5 [407-424 б.] 13 [473-496 б.]	13 апта
Ағымдық бақылау 14	Өзіндік жұмыс №6	OpenGL-дің негізгі мүмкіншіліктері	1-6	14 апта
Аралық бақылау 2	2 модуль аттест.сұрақтар жауабы	2 модуль бойынша бақылау өткізуге арналған сұрақтар	1-13	15 апта
Қорытынды бақылау	Емтихан	Курс бойынша емтихан сұрақтары	1-13	

1.7 Әдебиеттер тізімі

Негізгі әдебиеттер тізімі

1 Фоли Дж., Вэн Дэм А. Основы интерактивной машинной графики. – Т. 1, 2. – М.: Мир, 1985.

2 Роджерс Д. Алгоритмические основы машинной графики. – М.: Мир, 1989.

3 Роджерс Д.Ф., Адамс Дж. Математические основы машинной графики. – М.: Машиностроение, 1980.

4 Пореев В. Компьютерная графика. – СПб.: БХВ – Санкт-Петербург, 2004.

5 Петров М.Н., Молчанов В.П. Компьютерная графика. – СПб.: Питер, 2002.

6 Краснов М. В. OpenGL. Графика в проектах Delphi. – СПб.: БХВ – Санкт-Петербург, 2000.

Қосымша әдебиеттер тізімі

7 Шикин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистичные изображения. – М.: Диалог-МИФИ, 2000.

8 Ньюмен У., Спрулл Р. Основы интерактивной машинной графики. – М.: Мир, 1976.

9 Иванов В.П., Батраков А.С. Трехмерная компьютерная графика / Под. Ред. К.М. Полищука. – М.: Радио и связь, 1995.

10 Хирн Д., Беркер М. Микрокомпьютерная графика. – М.: Мир, 1987.

11 Павлидис Т. Алгоритмы машинной графики и обработка изображений. – М.: Радио и связь, 1986.

12 Музыченко В.Л., Андреев О.Ю. Самоучитель компьютерной графики. Учебное пособие. – М.: Технолоджи – 3000, 2003.

13 Бурлаков М. CorelDRAW 12. – СПб.: БХИ-Петербург, 2004.

1.8 Білімді бағалау және бақылау

Қ.И. Сәтбаев атындағы ҚазҰТУ-дың барлық курстары мен барлық пәндері бойынша студентердің білімін тексеруге рейтингілік бақылау қолданылады. Пәннің қорытынды бақылау рейтингі 100 % тең. Сессия кезінде қорытынды бақылау емтихан (жазбаша), ол 100 % шкала бойынша бағаланады.

Семестр кезінде 8, 15-ші аптада екі аралық бақылау жазбаша түрде жүргізіледі, ол 100 % бағаланады(кесте 1.3).

Кесте 1.3

Бақылау түрлері бойынша рейтинг балдарын бөлу

Семестр	Қорытынды бақылау түрі	Бақылау түрі	%
4	Емтихан	Ағымдық бақылау	100
		Аралық бақылау	100
		Қорытынды бақылау	100

Студенттің білімін бағалаудағы ағымдық бақылауға 7 зертханалық жұмыс 6 өзіндік жұмыс орындау жатады.

Ағымдық бақылаудың нәтижелерін тапсыру мерзімі пәннің бақылаудың барлық түрін өткізу бойынша күнтізбелік кестесінде көрсетілген (кесте 1.4).

Кесте 1.4

«Интерактивті графикалық жүйелер» пәнінің
бақылаудың барлық түрін өткізу бойынша күнтізбелік кестесі

Апталар	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Бақылау түрлері	ЗЖ 1	ӨЖ 1	ЗЖ 2	ӨЖ 2	ЗЖ 3	ӨЖ 3	ЗЖ 4	АБ 1	ЗЖ 5	ӨЗ 4	ЗЖ 6	ӨЖ 5	ЗЖ 7	ӨЖ 6	АБ 2
Бақылау түрлері: ЗЖ – зертханалық жұмыс; ӨЖ – өзіндік жұмыс, АБ – аралық бақылау.															

Пәннің қорытынды бағасы шкала бойынша анықталады (кесте 1.5).

Кесте 1.5

Студенттердің білімінің бағасы

Баға	Әріптік эквивалент	Процент бойынша %	Балл бойынша
Өте жақсы	A	95-100	4
	A-	90-94	3,67
Жақсы	B+	85-89	3,33
	B	80-84	3,0
	B-	75-79	2,67
Қанағаттандырарлық	C+	70-74	2,33
	C	65-69	2,0
	C-	60-64	1,67
	D+	55-59	1,33
	D	50-54	1,0
Қанағаттандырарлықсыз	F	0-49	0

Аралық аттестация және модуль бойынша бақылау өткізуге арналған сұрақтар тізімі

1 модуль бойынша бақылау өткізуге арналған сұрақтар:

1. Интерактивті графикалық жүйелер дегеніміз не?
2. Бейнені өңдеу және компьютерлік қарау дегеніміз не?
3. Адамның көзқарасы түсті қалай қабылдайды?
4. Компьютердің видеожүйесінің өзгеруі қалай болады?
5. Видеоплаттың негізгі мінездемелері.
6. Графикалық бейнелерді баспаға шығару қалай болады?
7. Векторлық графика дегеніміз не? Ол растрлық графикадан несімен

ерекшеленеді?

8. Векторизация түсінігін жазыңыз. Векторлық графиканың кемшіліктері мен артықшылықтары.

9. Қандай векторлық графикалық форматтар бар? Олардың артықшылығы мен айырмашылығын айтыңыз.

10. Растрлық графика дегеніміз не және растеризация түсінігі туралы айтыңыз.

11. Растрлық графиканың векторлық графикадан айырмашылығы. Растрлық графиканың артықшылығы мен айырмашылығын айтыңыз.

12. Растрлық өрбіту дегеніміз не?

13. Бейненің генерациясы тәсілдері?

14. Жазық алаңның растрлық өрбітуі қалай болады?

15. Полутон әдісінің мағынасын айтыңыз?

16. Ауыстыру әдісінің мағынасын айтыңыз?

17. Координаттық әдіс. Ол не үшін қолданылады?

18. Сызықтық және сызықтық емес түрлендіруді айтыңыз?

19. Жазықтықта аффиндік түрлендіру қалай жазылады?

20. Жазықтықта параллельді жылжыту және кері түрлендіру қалай болады?

2 модуль бойынша бақылау өткізуге арналған сұрақтар:

1. Адамның көзқарасы түсті қалай қабылдайды?

2. Түс және жарық дегеніміз не?

3. Түстің спектрлі құрамы қалай іске асырылады?

4. Түс пен жарық қалай анықталынады?

5. Сәулененетін түс дегеніміз не?

6. Шағылысатын түс дегеніміз не?

7. Шағылысу мен жіберудің спектрлі сипаттамаларын жаэыңыз.

8. Адамның көзқарасы түсті қалай қабылдайды?

9. Түстің спектрлі құрамы қандай?

10. Түс пен жарықтың табиғаты қандай?

11. Түс пен жарықтың байланысы.

12. Динамикалық және түстік диапазон дегеніміз не?

13. Түстік модель дегеніміз не?

14. Қандай түстік модельдер бар?

15. Аддитивті түсткі модельдерді жазыңыз. RGB түстік моделінің артықшылықтары.

16. RGB моделінің жағымды және жағымсыз қасиеттерін айтыңыз.

17. Түстік және субстрактивті модельдерді жазыңыз. CMY CMYK-тен айырмашылығы.

18. RGB және CMY модельдерінде түс қалай құрылады және олардың айырмашылықтары.

19. Перцепциондық түстік модельдер?

20. HSB түс моделі туралы айтыңыз?

Аралық аттестацияға дайындалуға арналған сұрақтар:

1. Компьютердік графика дегеніміз не?
2. Компьютердің бағыныңқы жүйесі қалай өзгереді?
3. Бейне платтың негізгі сипаттамалары.
4. Графикалық кескіндеуді баспаға шығару болады?
5. Кескіндеуді өңдеу және компьютердік көзқарас дегеніміз не?
6. Линиатура дегеніміз не? dpi және lpi-дің айырмашылықтары?
7. Қандай растрлық форматтар бар? Олардың айырмашылығы мен өзгешелілігі.
8. Қандай векторлық форматтар бар? Олардың артықшылықтары мен өзгешелілігі.
9. Растрлық графика дегеніміз не және растеризация түсінігін айтыңыз.
10. Растрлық графиканың векторлық графикадан айырмашылығы. Растрлық графиканың артықшылығы мен кемшілігі.
11. Векторлық графика дегеніміз не? Ол растрлық грфикадан қалай ерекшеленеді?
12. Векторизация түсінігі.
13. Векторлық графиканың артықшылығы мен кемшілігі.
14. Растрлық өрбіту дегеніміз не?
15. Кескіндеу генерациясының тәсілдері?
16. Жазық алаңның растрлақ өрбітуі қалай болады?
17. Полутон әдісінің мағынасын түсіндіріңіз.
18. Тасымалдау әдісінің мағынасын түсіндіріңіз.
19. Координаттық әдіс және ол не үшін қолданылады?
20. Сызықтық және сызықтық емес түрлендіруді айтыңыз.

1.9 Курстың саясаты мен процедурасы

Оқу процесінде студенттер мынадай талаптарды орындаулары керек:

– студенттер міндетті түрде сабаққа қатысуы керек;

– студент сабақты босатқан жағдайда (себепті немесе себепсіз) сабақтан тыс уақытта жұмыстарын өткізеді.

Зертханалық жұмыс тапсырмаларын алдынғы зертханалық жұмыс тапсырмасын тапсырған уақытта ғана ала алады.

Студент қорытынды бақылауға тек қана барлық бақылау түрлерін өткізгенде ғана жіберіледі.

Сабақ кестесіне енгізілмеген сабақтарға студенттің қатысуы міндетті емес, себепті жағдайлармен қалып кеткен сабақты қайта тапсыруға, бақылау мерзімінен бұрын тапсыруға рұхсат етіледі.

2 НЕГІЗГІ ТАРАТЫЛАТЫН МАТЕРИАЛДАР МАЗМҰНЫ

2.1 Пәннің тематикалық жоспары

Тақырыптардың аты көрсетілген пәннің тематикалық жоспары және сабақтың барлық түрі бойынша академиялық сағаттары тақырыптары бойынша қарастырылған. Пәннің сабақ түрлері бойынша сағаттарды бөлу 2.1 кестеде көрсетілген.

Кесте 2.1

«Интерактивті графикалық жүйелер» пәнінің
сабақ түрлері бойынша сағаттарды бөлу

№	Тақырып атаулары	Академиялық сағаттар саны			
		Дәріс-тер	Зертха-налық жұмыс-тар	СОӨЖ	СӨЖ
1	Интерактивті графикалық жүйеге кіріспе. Компьютерлік графиканың аппараттық қамтамасыздандыруы	2		3	3
2	Векторлық графиканың негізгі принциптері	2	2	3	3
3	Векторлық графиканың математикалық негіздері	2	2	3	3
4	Растрлық графиканың алгоритмдері	2	2	3	3
5	Растрлық өрбіту	2	2	3	3
6	Қадамдықты жою әдістері	2		3	3
7	Геометриялық түрлендіру	2	2	3	3
8	Объектілерді түрлендіру	2	2	3	3
9	Компьютерлік графикадағы түс	2		3	3
10	Жазықтықтағы фигуралардың кесілуі	2		3	3
11	Көрінбейтін сызықтарды жою	2		3	3
12	Шын кескіндеулерді құру	2		3	3
13	Сәулелердің трассировкасы	2		3	3
14	OpenGL-дің негізгі мүмкіншіліктері	2	3	3	3
15	Open GL-де проекция мен координаттарды түрлендіру	2		3	3
Барлығы:		30	15	45	45

2.2. Дәріс сабақтарының конспектісі

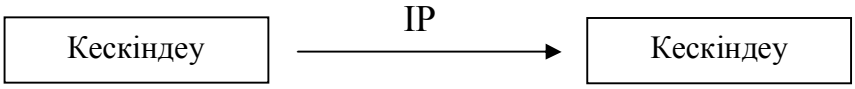
Дәріс 1. Интерактивті графикалық жүйеге кіріспе. Компьютерлік графиканың аппараттық қамтамасыздандыруы

1.1 Интерактивті графикалық жүйелердің (ИГЖ) негізгі есептері. ИГЖ-ді қолдану аймағы. ИГЖ-ді қолданудың жіктелуі. ИГЖ қысқаша тарихы

Компьютердің ең маңызды функциясы – ақпаратты өңдеу. Кескіндермен байланысты ақпаратты ерекше атап өтуге болады. Ол негізгі үш бағытқа бөлінеді: компьютерлік графика (КГ), кескіндерді өңдеу және кескіндерді тану.

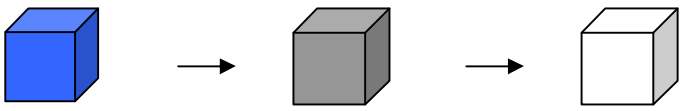
Компьютерлік графиканың мақсаты – көрсетушілік (визуализация), яғни кескінді құру. Көрсетушілік сипатталатын затты бейнелеуден тұрады. Көрсетушіліктің көптеген әдістері мен алгоритмдері бар. Олардың бір-бірінен нені бейнелеу керек және оны қалай бейнелеу керек екендігі жөнінде айырмашылықтары бар. Мысалға, адамның ойында тек қана мына бейнелер болуы мүмкін – функция графигі, диаграмма, схема, карта. Немесе керісінше – компьютерлік ойындарда, көркем фильмдерде, тренажерда, архитекторлық жобалау жүйелерінде сценаны кескіндеу. Мұнда ең маңызды және бір-бірімен байланысты факторлар кадрларды өзгерту жылдамдығы, сценаның объектілермен толтырылуы, кескіндеу сапасы, графикалық құрылымдардың ерекшеліктерінің тіркелуі болып табылады. Ақпаратты өңдеу кезінде, яғни мониторда кескіндеумен байланысты үш бағыт бар: COMPUTER VISION, IMAGE PROCESS-ING және COMPUTER GRAPHICS. Образдарды танудың негізгі есебі бар кескіндеуді символдардың түсінікті форальді тілінде түрлендіру болып табылады. ***Образдарды танып білу немесе техникалық көру жүйесі (COMPUTER VISION) деп*** – кескіндеуді сипаттауды алу әдісі, немесе берілген кескіндеуді қандай да бір класқа жатқызуды (бұл жағдайда мысалға, почтаны сараптауды) айтамыз. COMPUTER VISION-нің негізгі есебі болып объектілердің склетизациясы жатады, яғни объект негізі, «скелеті». Мысалға қандай да бір сцена бар делік (аудитория, бөлме және т.б.). Компьютер осы сценаға сипаттама беруі керек (онда заттар бар ма, қандай жарықтану және т.б.). Computer Vision (CV) кескіндеуді сипаттауға ауыстырады.

Кескіндерді өңдеу ***(IMAGE PROCESSING)*** кіру және шығу деректері кескіндеу болатын есептерді қарастырады. Мысалы, шумды болдырмау және деректерді сығымдауда кескіндерді беру, кескіндеудің бір түрден екінші түрге ауысуы (түрлі-түстіден ақ-қараға) және тағы басқа. Image Processing – кескіндеуді өңдеу.



Кескіндеуді өңдеу есептері:

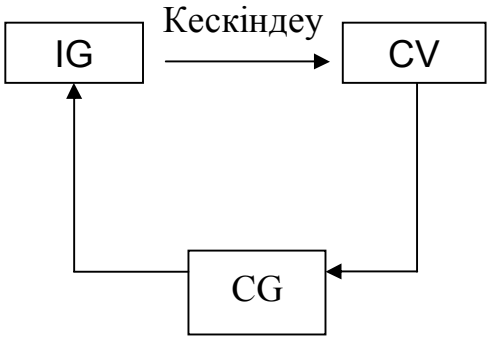
1. Кескіндеуде дефектіні алып тастау (мысалы, фото-пленкада қарды алып тастау;
2. Кескіндеуді жақсарту (мысалы, ұсталынбаған фотосуретті сәл қараңғылау);
3. Кескіндеуді жеңілдету (мысалы, түрлі-түстіні ақ-қараға, каркасқа).



Компьютерлік (машиналық) графика (COMPUTER GRAPHICS) – Бейнеленбейтін табиғат мәліметі шығатын бейнелеуді қайта түрлендіру. Мысалы эксперименталь деректерді графика, гистограмма немесе диаграмма түрінде көрсету, компьютерлік ойындарда ақпаратты экранға шығару, тренажерда сценаны синтездеу болып табылады.

Компьютерлік графика – құру, сақтау және модельдерді өңдеу және оларды ЭЕМ көмегімен кескіндеуді оқып үйренетін ғылым. Егер қолданушы объектілердің сипаттамаларын басқаратын болса онда оны интерактивті компьютерлік графика деп түсінуге болады.

Мынадай схеманы аламыз:



Объектіден таралатын және контуры мен деталын көрсететін оптикалық сәуле жүйесі нәтижесінен өтетін сурет – оптикалық кескін болады. Кескіндеуді оптикалық сыйымдылық функциясы көмегімен сипаттауға болады:

$I = g(x, y)$ – үздіксіз жарық функциясы.

Егер де біз белгілі бір нүктелерде функцияның мәнін алатын болсақ, онда матрицаны аламыз (g_{ij} матрицаның әрбір элементінде (i, j) – нүктесіндегі интенсивті функция мәні). Осы матрицаның көмегімен кескін машинаға беріледі. Бейненленетін матрицаның элементі – **picture element – pixel**.

Егер кескін ақ-қара болса, онда бізге градация да жеткілікті: 0 – жарықтың жоқтығы (қара); 255-ақ.

Сондықтан бізге әрбір пикселге 1 байт жеткілікті. Егер кескін үш түспен берілетін болса (қызыл, сары, жасыл), онда әрбір пикселге жатады.

1.2 Компьютер құрылымы және комплектісі. Монитордың негізгі сипаттамалары. Графикалық бейнелерді баспаға шығару. Графикалық жұмыс станциялары

Бұл дәрісте біз негізінен графикамен тиімді жұмыс істеудегі компьютерге қойылатын талаптарды қарастырамыз. Мұнда біз компьютерлік техникагың қарыштап дамуына байланысты бұл талап үнемі өсіп отырғанын атап өтуіміз керек. Келесі ішкі пунктер қарастырылады:

– кескінді енгізу және шығару әдістері жайындағы алдын-ала ескертулер;

– компьютер құрлымы мен кешені;

– компьютердің графикалық жүйесі;

– периферия;

– графикалық мәліметтерді енгізу құрылғысы.

Бейнені енгшізу және шығару құралдары жөнінде алдын-ала ескертулер. Компьютерде сандық кескінді басқару үшін тек қана графикалық редактор емес

сонымен бірге оларға сәйкес ДК аппараттық бөлігі қажет. Сіздің жабдығыңыз тиімді жұмыс істеу үшін мынандай сипаты болу керек, графикалық ақпаратты тез, қолайлы және ыңғайлы өңдеуді қамтамасыз ету керек. Бұл дәрісте біз компьютерлік "темірдің" құрамын және құрлымын қарастырамыз. Дәрістік матералдарды дайындаған кезде графика ауданындағы компьютерге деген минимальді талаптар жиыны келесідей болды:

– қолдау параметрі 1024x768 нүкте және одан жоғары деңгейге рұқсатталады;

– түс тереңдігі 24 бит (TrueColor);

– 64 Мбайт кем емес ЖСҚ (ОЗУ) болуы;

– CD-ROM болуы.

Дерес компьютердің базалық конфигурациясы келесі құрылғыдан тұрады:

– жүйелік блок;

– монитор;

– пернетақта;

– басқарушы «тышқан».

Компьютерге қосымша құрылғы қосылады. Оларды периферейінді деп атайды: принтер, джойстик, сканер, музыкалық қойлым, модем, плоттер, дигитайзер, сандық фотокамера, жарықтық перо, графикалық планшет және т.б. Үстелдік компьютерлерден басқа тасымалдау компьютерлер бар. Қазіргі тасымалдау компьютерлерін ағылшынша «ноутбук» (note-book, немесе блокнот түріндегі компьютер) деп жиі атайды.

Жүйелік блок. Жүйелік блок дербес компьютердің орталық компоненті болып табылады. Оның корпусының ішінде компьютердің негізгі электрондық құрлымдар бар, оларға кіреді:

– видеоадаптерлі жүйелік тақта;

– қорек блогы;

– қатты және иілгіш магниттік дискілердегі жинағыштар.

Олардан басқа, жүйелік блокта компакт-дискідегі жинағышты, дыбысты тақтаны, видеобластерді, периферлік құрылғылар бақылаушы тақтасын ішкі модемді орналастырады.

Корпус. Жүйелік блок корпусы екі тпте дайындалады:

– горизонтальді корпус – desktop (сөзбе сөз «үстел беті»);

– вертикальді корпус – tower (мұнара), бірнеше варианттар кіреді;

– Mini-Tower, Midi-Tower, BigTower, Super-Big-Tower, File Server.

Негізгі әдебиет: 1 [бет 8-15]; 5 [бет.3-90]

Қосымша әдебиет: 8 [бет 11-19]; 10 [бет.10-31]

Бақылау сұрақтары:

1. Рұқсат етілген мүмкіндік дегеніміз не?
2. Растр өлшемі дегеніміз не?
3. Бейнені өңдеу қалай жүргізіледі?
4. Компьютерлік көз дегеніміз не?
5. Компьютерлік графика дегеніміз не?
6. Түстік мониторлар қалай түс шығарады?

Дәріс 2. Векторлық графиканың негізгі принциптері

2.1 Векторлық графиканың негізгі принциптері. Объектілер және олардың атрибуттары. Векторлық графиканың артықшылықтары мен кемшіліктері

Дискретизация (бұл жалпы сызықты) кез-келген біртекті формуланың негізінде тұрғызылған векторлардың ерікті контурын жасауға мүмкіндік береді.

Бұл жерден-сызықтық контурдың көп түрін сипаттауға мүмкіндік беретін, формулаларды іздеу есебі қалыптасады.

Дискретизация сипаты сызықтық болғандықтан ортақ контур өте кіші фрагменттерге-сплайндарға бөлінеді. Ол кезде олардың параметрлік формада көрсетілуін сипаттайттын қарапайым формуланы (функцияны) таңдап алу қажет.

Безье қисықтарының векторлық графиктерін және NURBS-қисықтарың таңдаудың ең маңызды себептерінің бірі, олардың басқарылуының біртектігінде, сонымен қатар, тек жанындағы қисықтың бөлігінің ғана формасына анықтайды.

Векторлық графиканық бағдарламасында форманың (пішімнің) өзгеруінің жалғыз әдісі – басқаратын және тіреу нүктелерін интерактивті орын ауыстыруы.

Безье қисығының базасында, парақтарды сипаттау тілі PostScript, шығару құрылғыларының (түрлі-түсті баспа, түспен шрифтерді басқару жүйелері) жаңа мүмкіндіктерін жинақтау жолының дамуына негізделген.

Сіздер мүмкін ұмытып үлгерген, мектеп курсынан, анықталған сызықтар, мысалы түзуді немесе параболаны екі түрлі әдіспен көрсетуге болатыны белгілі:

– аналитикалық,математикалық, формулалар қолданылатын кезде;

– графикалық, немесе геометриялық, ол жазықтықта визуальды бейнелеген кезде.

Сызықтық графикалық барлық көптеген түрлерін, оны сипаттайтын және тілді бекітуге мүмкіндік беретін формула түрінде көрсетуге болады деп жобаланады.

Сондықтан дискретизация басқа-сызықтық сипат қабылдайды, яғни пиксельдік графикаға негізделген кеңістіктен дискретизация, сызық бойында тек қана бір *-1 бар* болғандықтан сызықтыққа ауысады.

Егер ерекші қисықтарды жеке фрагменттерге (сегменттерге), бөлетін болсақ, онда келесі берілген шарттарды қабылдаған мақұл.

– сызықтарды өте майда (қысқа) фрагменттерге бөлу;

– оларды сипаттау үшін негұрлым қарапайым формулаларды (функцияларды) тандау.

Ең қарапайым табиғи функция-сызықтық тәуелділік, оның көмегімен жазықтықта жатқан екі нүктенің арасындағы ең кіші ара қашықтық – түзу сызық сипатталады.

Сызықтық суретті дискретизацияның кішкентай элементтеріне бөлу жеткілікті және пайда болған дискретизация нүктелерін түзу сызықпен қоса

отрып кез келген сызықты объекті және кез келген күрделі қисықты бұл түзудің саналатын (сонғы) санының көмегімен көрсетуге болады. Мұндай технологияның ең басты жетістігі, оның қарапаймдылығы: әрбір нүкте үшін, осы нүктенің кординаталарын анықтайтын бар болғаны екі сан жеткілікті. Сонымен, өте үлкен қисықты бар болғаны, жүздеген қос сан арқылы сипаттауға болады.

Бірақ көрсетілген қарапайымдылық, маңызды кемшіліктердің себебі болып таблады:

– тек тік сызықтардан құрастырылған объектер, еркін масштабтану мүмкіншілігінен айрылады. Кесінділер өте майда болған кезде, олар бұрыштық сипаттама бермейді, бірақ маңызды коэффициентті үлкейткенде, бұрыштар көрнекті болады;

– сызықтық кесінділермен аппроксимацияланған объект формалары (пішімдері) өзгеруі мүмкін, мысалы айналғанда. Ал объект формасын (пішімін) нақты ақиқат аппроксимациялау үшін, шеңбер көп бұрыш емес шеңбер сияқты көрінгенде ондаған мың сызық сегменттері қажет.

Көрсеттілген кемшіліктер объект формасын сипаттаудың басқа әдістерін және өте күрделі жеке жағдайда жоғарғы дәрежелі (екінші, үшінші және т.б) қисықтарды іздеуге мәжбүр етеді.

Негізгі әдебиет: 1 [бет 8-15]; 5 [бет.3-90]
Қосымша әдебиет: 8 [бет 11-19]; 10 [бет.10-31]
Бақылау сұрақтары:
1. Векторлық суреттің құрылымы қандай?
2. Объектердін қасиеттері қандай?
3. Бейнені өңдеу қалай жүргізіледі?
4. Компьютерлік графика дегеніміз не?

Дәріс 3 Векторлық графиканың математикалық негіздері

3.1 Векторлық графиканың математикалық негіздері. Безье қисығы. Векторлық графикалық форматтар

Параметрлік теңдеулер. Жоғарыда айтылғандай қисықтарды аналитикалық және графикалық түрде көрсетуге болады, яғни функция графигі сияқты. Математиктер оны былай жазлады:

$$y = f(x),$$

мәні нені білдіреді y – функция, яғни x мәнінен тәуелді, мысалы қарапайым функция $y = 2x$ қарапайым тәуелділікті білдіреді y-шің әрбір мәні, x-тің кез келген мәнінен екі есе артық. Бұл функцияның графигі координатаның басынан өтетін, түзу сызықты көрсетеді.

Тригонометриялық функциялардың түрі қызығырақ, мысалы синусоидтың:

$$y = \sin x .$$

Бұл қисықтың графигі әркімге белгілі.

Формуланың және оның графигінің мұндай әдіспен көрсетілуі *айқын* деп

аталады. Ол графиктерді жеңіл салуға мүмкіндік береді. Бірақ бұл әдістің графиқалық,түрін қарастырғанда маңызды кемшіліктері бар:

– *x*-тің әрбір мәніне, *y*-тің тек бір ғана мәні сәйкес келеді. Бұл қисықтың жаңа фрагметін кез келген жерден бастауға мүмкіншілік бермейді;

– қисық тұйық бола алмайды.

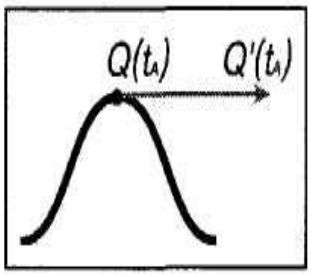
Нәтижесінде қисықтық айқын әдісі жазықтықтың кез-келген жерінде орналасқан кез-келген қисықты сипаттау керек болғанда қолданылмайды. Қисықты параметрлік фукция сияқты анықтау Альтернативті әдіс болады. Бірінші кезекте келесі ерекшеліктерді белгілеу керек: Мұндай әдістің екі кординатасы (*x, y*) тепе-тең, әдетте *t* символы арқылы белгіленетін кейбір көмекші параметерлердің функциясы сияқты есептелінеді. Жалпы жағдайда ондай тәуелділіктің түрі:

$$q(t) = \{x(t), y(t)\},$$

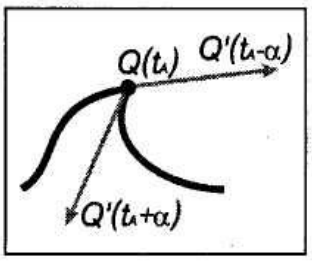
мұндағы $x(t)$ және $y(t)$ – *t параметрінің функциялары, t-ға* бірдей мән беріп, $x(t)$ функциясы *x*-тің координатасының мәнін, ал $y(t)$ – фунциясы *y* координатасының мәнін есептейді..

Тегіс қисықтар. Векторлық графиканың құралы ретінде Безье және NURBS-қисықтарының тандалуының ең маңызды себебі тегіс басқарылатындығында. Тегістік, модельденген кезде тұзақ құрмайды және кенеттен қисаю (әсіресе үзіліс) болмайтындығын білдіреді, сондықтан тегістік жанасуының және сол сияқты шешімдердің спиральдың үшбұрыш бұрыштарды құруы мүмкін. Тегіс қисықтардың және үшкір иілулер спиральдық бірігуінің керемет мысалы авиақанаттардың профилі бола алады. Қисықтардың тегістігін талқылайық.

Бөлшектің қозғалу бағытын ойда елестету үшін оған параметрлік қисықтың бойындағы қозғалысты үзіліссіз көрсететін стрелканы бағыттауышты «бекітсе» болғаны. Математикалық тілде бөлшектегі бағыттауыш *жанама* деп аталады. Егер көрші нүктедегі жанама өзінің бағытын кенеттен өзгертпесе, онда ондай қисықты тегіс біртектес деп есептейді. (3.1 сурет). Егер қисықта сыну болса, *Q* нүктесінде жанаманың бағыты кенеттен өзгереді (3.2 сурет).



Сурет 3.1 – Біртекті қисықтарға жүргізілген жанама



Сурет 3.2 – Біртекті қисықтарға жүргізілген жанама

Енді біздер векторлық компьютерлік графикте қолданылатын біртекті тегіс қисықты салудың негізімен танысуымыз қажет. Мұндай қисықтардың жалпы жағдайы (яғни оған сәйкес, және күрделірекк) болатын NURBS - қисығынан бастайық.

NURBS - қисығы

Түсіндіруді NURBS терминінен Non-Uniform Rational B-spline-нің, аббревиатурасы.

Мұндағы:

– "Non-Uniform" (бір текті емес) – қисықтың формасына (пішіміне) бақылау нүктесінің әсер ету аймағы әртүрлі болуы мүмкін. *Иррегулерлық* қисықтарды модельдеу үшін бұл өте маңызды қасиет;

– "Rational" (рациональды) – модельденеттін қисықтың формасын (пішімін) сипаттайтын математикалық өрнек, екі полиномның қаынасын білдіреді. Бұл ерекшелік әртүрлі қисықтарды дәлірек модельдеуге мүмкіншілік береді, мысалы конустық қима;

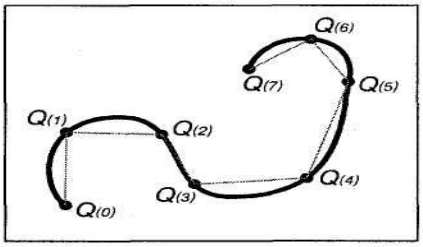
– "B-spline" (basis spline, базалық сплайн) – үш немесе одан да көп бақылау нүктелерінің арасындағы қисықтық интерполяциясын математикалық сипаттау әдісі.

Бақылау нүктелері

Алдымен, бұрын айтылған параметрлік қисықтың анықтамасын еске түсіру керек. Бұл анықтамада q функциясы сипаттайтын өрнектің сол бөлігі, осылай көрінеді:

$$q(t) = ...,$$

мұндағы t – анықталған диапазонда берілген мәндердің жиынын көрсететін параметр, әдетте, 0-ден 1-ге дейін. Бұл мәндерді пайдалана отырып модельденетін қисық, салынатын $\{x, y\}$ жұп тізбегін алады (3.3 сурет).



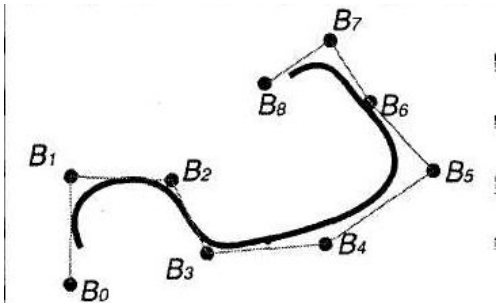
Сурет 3.3 – Параметрлік қисықты салу мысалы

Жоғарыда көрсетілген өрнектің оң жағы, яғни өзіндік параметрлік тендеулер, ал дәлірек параметрлік тендеулер анықталмаған NURBS – қисығының негізгі болжамалы ерекшеліктерің бірі, оның формасы бақылау нүктелері (control points) жиынының орналасуымен анықталады. 3.4-ші суретте бұл нүктелер B_t-де белгілетен. NURBS-қисығынаң бұл ерекшелігі маңызды, бүкіл қисықтың формасы өзгермей, жеке бақылау нүктелерінің орын ауыстыруымен қисық формасының өзгеруін жинақтауға мүмкіндік береді.

Әрбір бақылау нүкте, оның аймағындағы қысықтық бөлігінің формасын анықтайды және оған аз әсер етеді немесе қисықтың қалған бөлігінің формасына мүлдем әсер етпейді.

Базалық функциялар

Қисық формасының мықты екенін анықтайтын B_j нақты бақылау нүктесіне тәуелді функция, осы бақылау нүктесінің базалық функциясы (basis function) деп аталады.



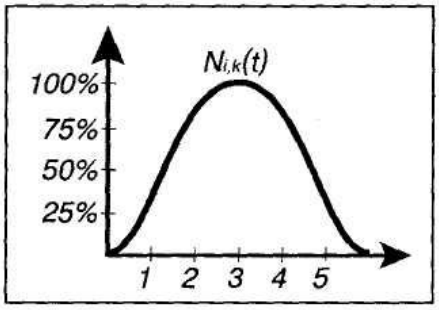
Сурет 3.4 – Бақылау нүктесінің орнын ауыстыруымен
шақырылған қисық фрагментінің формасының өзгеруі

Базалық функцияның мәндері нақты сандар NURBS-қысығының сипаттамасы әр бақылау нүкте үшін базалық функцияның берілуін талап етеді.

Бақылау нүктесінің бұл сияқты «әсері» тек сандық мәнмен ғана емес графикпен(бұл функция болғандықтан) де өрнектелуі мүмкін екеніне көңіл аудару керек.

Сонымен, қозғалған бөлікке әсердің тәуелділігі сияқты, базалық функцияның графигін салуға болады, мысалы, мәндердің проценті (3.5 сурет).

Нақты анықталған нүктеде максимальды тиімділікке (максимальды әсерге) жетеді және ол жойылуға байланысты біртіндеп азаяды. Бұл тәуелділікті сипаттайтын қисықтың формасы, қоңырауды еске түсіреді.



Сурет 3.5 – Жеке бақылау нүктесінің базалық функциясының
біртекті графигі

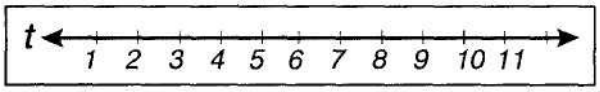
Әрбір бақылау нүктесінің «міндетті» түрде өзнің базалық функциясы болатындықтан, мысалы бес бақылау нүктесінен құрылған NURBS-қисығы кейбір аймақты жабатын қисықтың нәтижесінде бес сондай функция болу міндетті (3.5 сурет).

Тараптар

3.5-ші суретте барлық базалық функциялардың формалары бірдей және олар бір-біріне бірдей қашықтықта орналасқан. Бұл өте симметриялы және элегантты, бірақ, анықталған бақылау нүктелері қисықтың біршама үлкен, ал анықталғаны-кішірек сегметтке әсер ететіндей интервалдар ұзындықтарының әртүрлі өзгеруі керек. Бұл қисықты сипаттағанда оның біртекті (Non-Uniform) еместігіне жағдай жасайды.

Бірақ, параметр өсі бөлінетін нүктелер тізбегін анықтау оңай тапсырма емес. Мұндай нүктелер арасында салыстырмалы интервалдар өзгергенде, сіздер қисық бойымен қозғалып келе жатқан бөлшекке, бақылау нүктелері әсерінің уақытын өзгерту мүмкіндігіне не боласыз. Интервалдарға шек қоятын нүкте *тораn* (knots), ал олардың реттелген тізімі – *тораптың векторы* (knot vector) деп аталады.

{0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0} 3.6-шы суретте көрсейтілген базалық функцияның тораптық векторының түрі. Бұл барлық функциялары бірдей уақыт интервалында анықталған біртекті тораптық вектордың (uniform knot vector), мысалы.



Сурет 3.6 – Біртекті тораптық вектордың мысалы

Безье қисығы

Жалпы жағдайда Безье кисығын, бақылау нүктелерінің *n+1* салмақталған қосындысы, *B* – сиплайндардың (NURBS-1 қисықтарының) жеке жағдайы деп анықтауға болады.

Мұндағы Бернштейн полиномы салмақты коффициент. Безье қисығының алғашқы үш дәрежесінің анықтамасын қарастырайық.

Сызықтық қисық, бірінші дәрежелі қисық (түзу), келесі параметрлік формуламен анықталады:

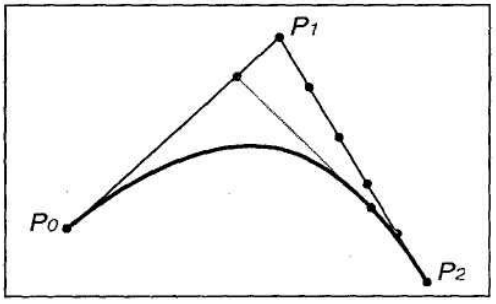
$$P(t) = (1-t)P_0 + tP_1$$

Бұл өрнек екі нүкте арасындағы сызықтық интерполяцияны көрсетеді (3.7 сурет).

Сурет 3.7 – Бірінші дәрежелі қисық (түзу)

Бұл өрнек нүктелер арасындағы сызықтық дәл интерполяцияны корсетеді (3.8 сурет).

$$P(t) = (1-t)[(1-t)P_0 + tP_1] + t[(1-t)P_1 + tP]$$

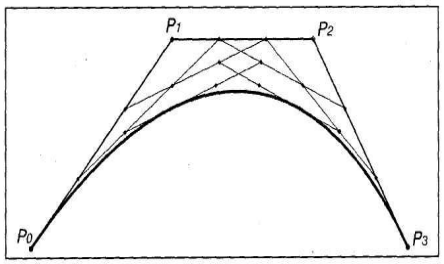


Сурет 3.8 – Екінші дәрежелі қисық (квадраттық қисық)

Кубтық қисықтық, үшінші дәрежелі қисық, төмендегі формуламен анықталады:

$$P(t) = (1-t)^3 P_0 + 3(1-t)^2 tP_1 + 3(1-t)t^2 P_2 + t^3 P_3$$

Бұл өрнек сызықтық интерполяциялар арасындағы сызықтық интерполяцияны, нүктелер арасындағы сызықтық интерполяцияны көрсетеді (3.9 сурет).



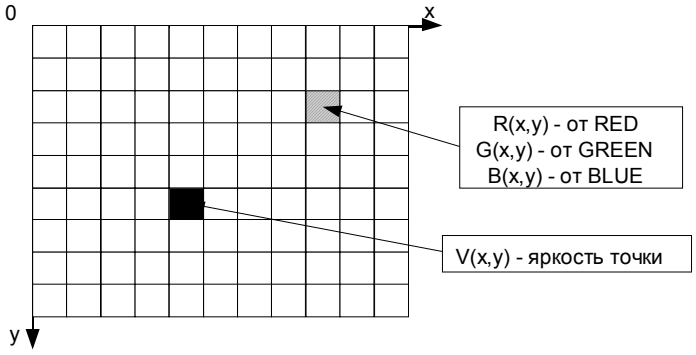
Сурет 3.9 – Үшінші дәрежедегі қисық (кубтық қисық)

Негізгі әдебиет: 1 [бет 8-15]; 5 [бет.3-90]
Қосымша әдебиет: 8 [бет 11-19]; 10 [бет.10-31]
Бақылау сұрақтары:
1. Безье қисығы дегеніміз не?
2. Сіздер қандай векторлық программаларды білесіздер?
3. Векторлық программалардың қандай кемшіліктері мен артықшылықтары бар?
4. Базалық функциялар;
5. Бақылау нүктелері.

Дәріс 4. Растрлық графиканың алгоритмдері

4.1 Квадратты растрда сызықтарды құру. Сызықтарды салудың параметрлік алгоритмі

Цифрлық кескін – кескіннің нүкте пиксель жиыны, кескіннің әрбір нүктесі координаталар мен x және y және $V(x, y)$ жарықтықпен сипатталады, бұл дискреттік мән, көбіне бүтін. Түсті кескін жағдайында әрбір пиксель координаттармен және үш жарықтықпен: қызыл жарықтық, көк жарықтық және жасыл жарықтықпен (V_R, V_B, V_G) сипатталады. Осы берілген үш түсті қосып әр түрлі түс алуға болады (4.1 сурет).

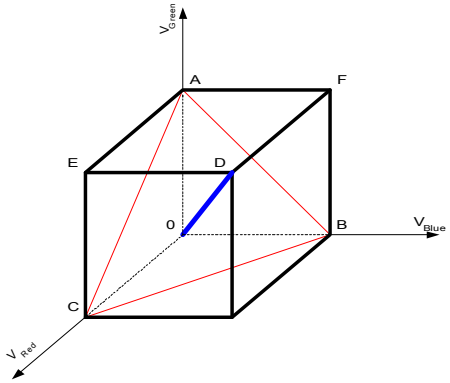


Сурет 4.1 – Кескіннің нүкте пиксель жиыны

Көбіне жарық градациясы кезінде 1 байт орын беріледі, сонымен бірге, 0 – қара, ал 255 – ақ түс (максимальді интенсивтік).

Градация кезінде жарықтықты көбіне (байтқа бөлінеді, мұнда 0 – қара түс, ал 25 – ақ (максималді интенсивтілігі). Түсті кескінді байт бойынша барлық үш түстің жарықтық градациясына бөлінеді. Биттің басқа мөлшерімен жарықтық градациясын кодтау мүмкін бірақ адамзат көзі әрбір түсте 8 бит градацияны ғана айыру қабілеті бар, бірақ арнайы аппаратура бұданда дәлірек түстер жіберуді талап етуі мүмкін. Қызыл, жасыл және көк интенсивтігінен құрылған түс кеңістігі түсті куб түрінде ұсынылады.

Кубтың *A*, *B*, *C* төбелері жасыл, көк және жылдық максималды интенсивтілігі болып табылады, сәйкесінше, олар құрған үшбұрыш. Паскал үшбұрышы деп аталады, осы бұрыш периметрлі максимальді қанықкан түскен сәйкеседі. *ОД* қосындысында сұр түстің түрлі түсі, сонымен бірге *О* нүктесі қараға сәйкеседі, ал *Д* нүктесі ақ түске сәйкеседі (4.2 сурет).



Сурет 4.2 – Түсті куб

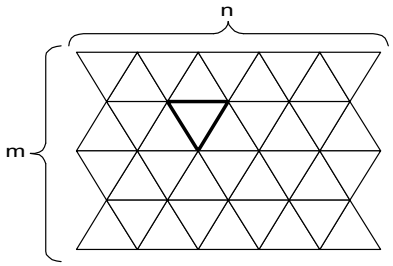
Растр – бұл нүктенің орналасу реті. (растрлық элементтер) 4.2 суретте квадрат болып табылатын растр элементтермен кескінделген, мұндай растр квадраттық деп аталады, дәл осындай расторлар өте жиі қолданылады. Фигураның растрлық элементі ретінде, басқа форманы қолдануға болады, ол келесі талапқа сәйкес келуі керек:

– барлық фигуралары бірдей болу керек;

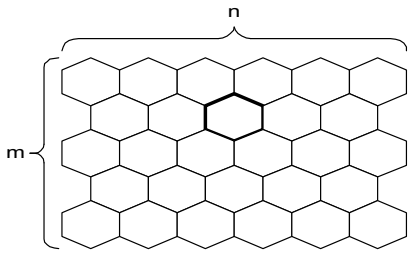
– ешқандай саңылаусыз жазықтықты толық жабуы керек.

Растрлық элементт ретінде тең қабырғалы Δ, дұрыс алтыбұрышты (гексаэдрді) қолдануға болады (4.3, 4.4 сурет).

Растрларды дұрыс емес көпбұрышты қолданып құруға болады, бірақ, мұндай растрларда оның практикалық мәні жоқ.



Сурет 4.3 – Үшбұрышты растр



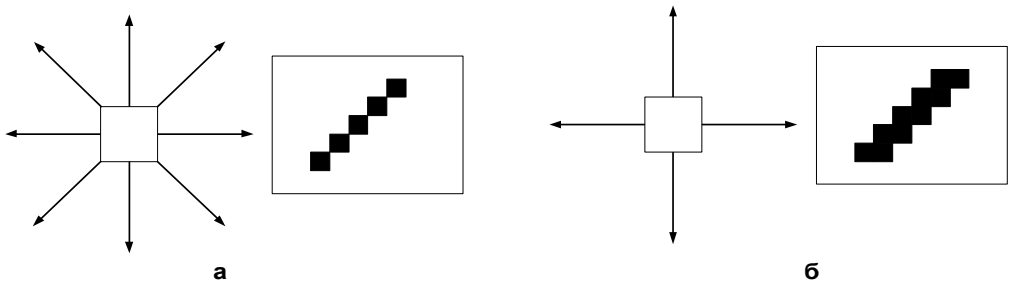
Сурет 4.4 – Гексагоналді растр

Тікбұрышты және гексагональді растрда сызықтарды құру әдісін қарастырамыз.

Квадраттың расторда сызықтарды құру екі әдіспен жүзеге асады.

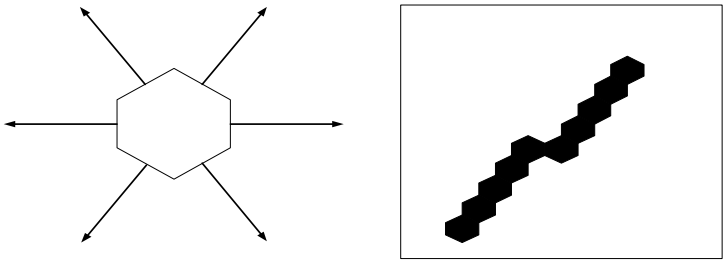
1) Нәтиже – сегізбайланысты сызық. Сызықтың көршілес пиксельдері мүмкін сегіз (4.5, а сурет) күйдің біреуінде болуы мүмкін. Кемшілігі 45^0 кезінде өте жіңішке сызық.

2) Нәтиже – төртбайланысты сызық. Сызықтық көршілес пиксельдері мүмкін болатын төрт күйдің бірінде болуы мүмкін (4.5, б сурет). Кемшілігі – 45^0 бұрыш кезіндегі жуан сызық.



Сурет 4.5 – Үшбұрышты растрда сызық құру

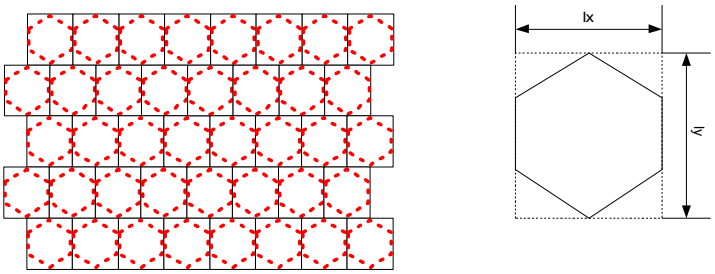
Гексагональді растрда сызықтар алты байланыста, мұндай сызық ені бойынша өте тұрақты, квадраттық растрға қарағанда сызықтық енінің дисперсиясы кіші (4.6 сурет).



Сурет 4.6 – Гексагональді растрда сызықты құру

Қай растр артық екенін қалай бағалауға болады. Бағалаудың бір әдісі болып жеткен сананың визуальды талдауы және келесі қалпына келтіргені кескін қолданылатын растр есебінде кодталған байланыс каналы бойынша жіберу табылады. Гексагональді растр оригиналдан ең аз ауытқуды қамтамасыз ететін эксперттік және математикалық дайындалған. Бірақ айырмашылығы көп емес.

Гексагональді растрды – модельдеу. Гексагональді растрды квадраттың негізінде құруға болады. Бұл үшін гексабұрыш тікбұрыш түрінде ұсынылады. Гексагональді кескінде қандай пропорция болу керек екенін анықтаймыз (4.7 сурет).



Сурет 4.7 – Гексагональді растрды квадраттықты құру

$$\frac{l_X}{l_Y} = \frac{\frac{a}{2} \cdot \tan(30^\circ) \cdot 2}{\frac{3}{2} \cdot a} = \frac{2}{\sqrt{3}};$$

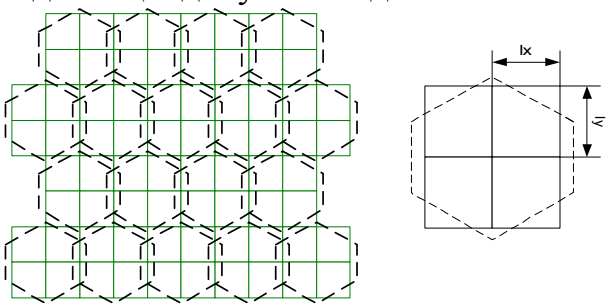
Кескіннің әрбір тақ емес жағынан 1 пиксель ұстап, және кескінді экранда $\frac{l_x}{l_y} = \frac{2}{\sqrt{3}}$ түрінде созып тікбұрыштыдан гексагональді растр моделін алуға болады.

Бұл таза аппараттық әдіс.

Гексагональді растр қолданылмайтындығы, келесі себептермен түсіндіріледі:

1. Алгоритмдердің кейбір күрделілігі;
2. Гексагональді растрдың артықшылығы онша кең емес;
3. Тікбұрышты растрды тарихи бағдарлау.

Гексагональді растрды квадраттың программалық тұрғызу үшін 4.8 суретте көрсетілген моделін қолдануға болады.



Сурет 4.8 – Гексагональді растрды квадраттыққа тұрғызу

Квадраттық растрда сызық тұрғызу

Электронды сәулелі түтікті (ЭСТ) расторлық дисплейдің экранды бір жүйеден басқа нүктеге жанама қиынды өткізуге болмайтын, жарықтанатын дискреттік элементтер матрицасы сияқты қарастыруға болады. Пиксельдерді анықтау процесі, берілген қиындықтың өте жақсы түрде аппароксимациялануы растрға жүктеу деп аталады.

Кескінді әр жол сайын визуализациялау процесі растрлық жаймалауды түрлендіру сияқты белгілі. Горизонтальді вертикальді және 45^0 бұрышқа еңкейген қиындыларға растрлық элементтерді таңдау белгілі. Басқа кез келген бағдарламауда керек пиксельді таңдау қиын, ол сурет 4.8 көрсетілген.

Қиынды кескініне жалпы талап:

– қиынды соңы берілген нүктеде жату керек;

– қиындылар түзу сияқты көріну керек;

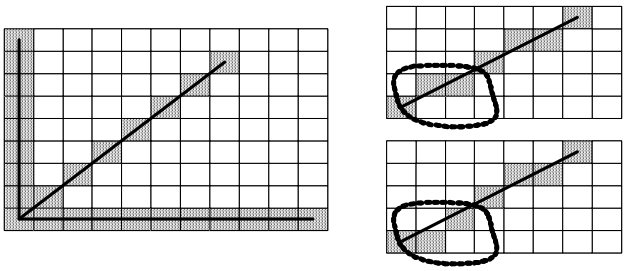
– жарықтық қиындының ұзындығы мен еңкеюіне тәуелсіз тұрақты болмауы (басынан соңына дейін) керек.

Осы шарттардың бірі растрлық дисплейде нақты орындалуы мүмкін емес, өйткені кескін соңғы өлшем пиксельдерінен тұрғызылады, негізінде:

1. Қиынды соңы жалпы жағдайды пиксельдің талап еткен позицияға ең жақыны және кейбір жеке жағдайда ғана қиындының соңғы координаттары пиксельдер координатының сәйкес болады.

2. Қиынды пиксельдер жиынымен аппроксимацияланады және кейбір жағдайда вертикальді горизонтальді жәнпе 45^0 астындағы қиындылар ғана түзу болып көрінеді, сонымен қатар вертикальді және горизонтальді қиындылар ғана баспалдақсыз жатық түзу боп көрінеді.

3. Жарықтық әртүрлі қиындыларда, тіпті қиынды бойында әртүрлі, мысалы, 45^0 қиынды және вертикальді қиындылар үшін пиксельдер центрі аралығында арақашықтық әртүрлі (4.9 сурет).



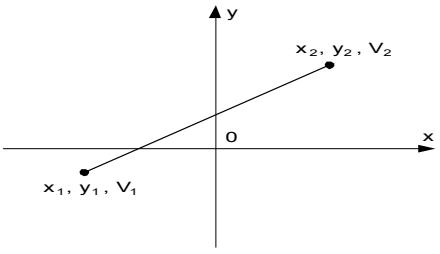
Сурет 4.9 – Растрды қиындыларға жіктеу

Аппроксимацияның объективтік жақсаруы дисплейдің рұқсаттығының ұлғаюына жетеді, бірақ растрлық жүйеде техникалық проблеманы шешу қабілетінің қолданылатын жылдамдығы 1280×1024 ретті құрайды.

Аппроксимацияның субъективтік жақсаруы көздің психофизиологиялық ерекшелігіне, жекелікке негізделген, оған экранда размерлік кішірейту кезінде жетеді. Аппроксимацияның субъективті сапасын жақсартудың басқа әдісі кескіннің керек шекарасын «бұзу» бойынша әртүрлі программалық айлаға негізделген.

Сызық сызудың параметрлік алгоритмі

Жарықтық бойынша интерполяциямен бір нүктеден (x_1, y_1) келесі (x_2, y_2) нүктеге сызық жүргізу қажет (4.10 сурет).



Сурет 4.10

Осы сызықтағы кез келген нүктені келесі түрде көрсетеміз.

$$x_{i+1} =]x_i + t \cdot (x_2 - x_1)[$$
$$y_{i+1} =]y_i + t \cdot (y_2 - y_1)[$$
$$V_{i+1} =]V_i + t \cdot (V_2 - V_1)[,$$

мұндағы $t \in [0;1]$,][– . бүтінге дейін жуықтау таңбасы.

$$t = \frac{1}{\max\{|x_2 - x_1|, |y_2 - y_1|\}} = \frac{1}{N-1};$$

N – сызық ұзындығы, пиксельмен.

Есептеуді координат өсімшесі арқылды есептеуге болады

$$\Delta x = \frac{x_2 - x_1}{N-1}; \Delta y = \frac{y_2 - y_1}{N-1}; \Delta V = \frac{V_2 - V_1}{N-1}$$

Өсімше мәні функция басында есептеледі және экранда сызық тұрғызу циклі кірмейді, оның есебіндегісіз жоғарылайды.

Алгортим кемшілігі:

– нақты сандармен жұмыс істеу қажеттілігі;

– алгоритмде бөлу операциясының болуы, аппараттың ұйымдастыруды күрделендіреді және алгоритммен жұмыс уақытын ұзартады.

Алгортим артықшылығы:

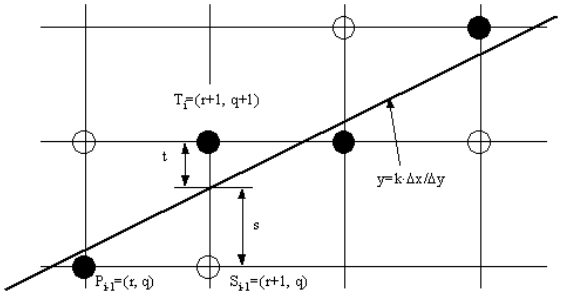
– программаны жүзеге асырудың қарапайымдылығы;

– жарық бойынша сызықтық интерполяцияның жүзеге асырылуының қарапайымдылығы.

4.2 Сызықтарды салудың Брезенхем алгоритмі. Айналдыруды құруға арналған алгоритм. Айналдыру генерациясына арналған Брезенхем алгоритмі

1965 жылы Брезенхем қиындыны растрлық тұрғызуға арналған жай бүтін сандық алгоритм ұсынды. Алгоритмде басқаратын айнымалы d_i қолданылады, ол әрбір қадамдағы S және t арасындағы пропорционалды айырмашылық сурет 20 i-қадам көрсетілген, нақты кескінделген қиындының ең жақын пикселі ретінде P_i табылған енді пиксельдердің қайсысы келесі екенін анықтау керек: T_i немсеес S_i.

Егер $S < t$ онда S_i қиындыға жақыны деп таңдап алу керек; кері жағдайда T_i жақыны болады. Басқаша айтсақ егер $S - t < 0$, онда S_i таңдалады; қарсы жағдайда T_i таңдалады (4.11 сурет).



Сурет 4.11

Кескінделген кесінді (x_1, y_1) нүктесінен (x_2, y_2) нүктесіне дейін жүргізіледі. Айталық алғашқы нүкте координаттар басына жақын жатыр дейік, онда екі нүктені де (T (x_1, y_1) қиындының бастапқы нүктесі координаттың басына келетіндей етіп көшіреміз, онда соңы (Δx, Δy) болады, мұнда

$$s = \frac{\Delta y}{\Delta x}(r+1) - q; t = q + 1 - \frac{\Delta y}{\Delta x}(r+1)$$

Шеңбер тұрғызу алгоритмі

Центрі координат басы болаттың шеңберді қараңыз, $x^2+y^2=R^2$ және параметрлік түрі (4.12 сурет):

1. $x=R\cos(a)$;
2. $y=R\sin(a)$.

Онда шеңбер сызу программасын жазу оңай:

```
void Circle (int x, int y, int R, int color)
{
    int a;
int x1;
int x2;
int y1;
int y2;
x2=x+R;
y2=y;
for ( int a=1; a<=360; a++)
{
    x1=x2; y1=y2;
x2=round(R*cos(a))+x;
y2=round(R*sin(a))+x;
Line (x1, y1, x2, y2, color);
    }
     }
```

Негізгі әдебиет: 5 [бет 252-297]

Қосымша әдебиет: 13 [бет 283-406]

Бақылау сұрақтары:

1. Растрлық графика неге нүктелік деп атайды?
2. Растрлық программада маскамен жұмысістеуді түсіндіріңіз?
3. Гистограмма не үшін керек?
4. Сіз «Бейнелеу қабығы» терминнің қалай түсінесіз?
5. Растрлық графиканың артықшылығы мен кемшілгіні атаңыз?

Дәріс 5. Растрлық өрбіту

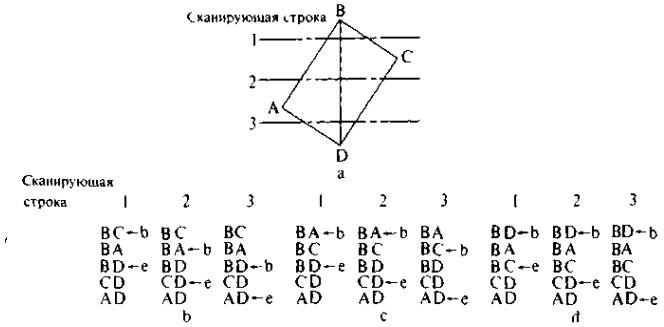
5.1 Растрлық өрбіту – кескіндеуді генерациялау әдісі

Растрға жайылған образды бейнемониторға шығару үшін, оны дисплей талап ететін шаблонда көрсету қажет. Бұл түрлендіру растрлық жаймалау деп аталады. Тізімдік дисплейден айырмашылығы тек кесінділер немесе литерлер

туралы ақпараты бар векторлық дисплей үшін, қазіргі жағдайда тізімдік дисплей экранындағы әрбір пиксель туралы ақпарат болуы қажет. Одан басқа, бұл ақпарат жолды сканирлеу ретімен бейне генерация жылдамдығымен ұйымдастырылуы және шығарылуы қажет, яғни, жоғарыдан төмен және солдан оңға. Мұндай нәтижеге жетудің төрт әдісі бар-нақты уақытта растрлық жаймалау, топтық кодтау, торлық ұйымдастыру және кадр буферінің жадысы.

Нақты уақыттағы растрлық жаймалау. Нақты уақытта жаймаланғанда немесе «ұшқанда» сцена еркін визуальды атрибуттар терминдерімен және геометриялық сипаттамалармен көрсетіледі. Түс, оттенок және интисивность визуалды абрибуттар, *x*, *y*, координаталары, иілу бұрыштары және мәтіндер геометриялық сипаттамаға жатады. Соңғылары *y* координаты бойынша реттелген. Әрбір кадрды көрсеткен уақытта бұл ақпаратты процессор сканирлейді және экрандағы әрбір пиксельдің интексивтігінін есептейді. Мұндай жаймалауда жады санының үлкендігінің қажеті жоқ. Әдетте, жадыға талап, дисплейлік тізімді плюс сканирленетін бір жолды сақтаудын қажеттілігімен шектеледі. Сцена туралы ақпарат ерікті ұйымдастырыған дисплейлік тізімде сақталатындықтан, тізімнен ақпаратты алып тастау немесе қосу өте жеңіл іске асады, ол бұл динамикалық колданбалар қосымшалар үшін өте ыңғайлы. Бірақ, шығарылатын бейненің күрделігі дисплейлік процессордың жылдамдығымен шектеледі. Әдетте, бұл картинада көпбұрыштардың немесе кесіндінің санының шектеулі екенін, сканирленетін жолмен қиылысу санын және түстердің немесе сұр п*олутондардың* санын білдіреді. Дисплейлік тізімнің әрбір кесіндісінің сканирленетін жолмен қилысуын (егер олар болса) алу үшін, қарапайым тәсілді іске асырғанда, жолды бейнелеген сайын барлық дисплейлік тізім өңделеді. Әрбір сканирленейтін жолға видеобейне регинерациянғанда, яғни, барлық тізімдер секунтарда өңделінеді. Мұндай аз уақыт бұл тәсілді тек күрделі емес чертеждерді сызғанда ғана пайдалануға мүмкіндік береді. Жалпы жағдайда сценадағы әрбір кесінді, әрбір сканирленейтін жолымен қиылыспайды, онда есептеулер саны активті, қабырға тізімін (САР) енгізу жолымен қысқарту мүмкін. Бұл тізімде сканирленген жолдармен қиылысатын кесінділердің бейнелері бар. САР ұйымдастыру және басқару үшін түрлі тәсілдерді қолдануға болады.

САР ұйымдастыру және басқару үшін түрдлі тәсілдерді қолдануға болады. Алдымен кесінді бейнелері *y*-тің ең үлкен координатасымен сұрыпталады,. Қарапайым тәсілдердің бірінде мұндай сұрыптаулар, сұрыпталған тізімде нұсқағыштардың орын ауыстыруы үшін пайдаланылады (5.1 сурет).



Сурет 5.1 – Активті қабырғаның қарапайым тізімі

Нұсқағыштың басы активті қабырға тізімінің басын белгілеу үшін пайдаланылады, ал нұсқағыштың соңы-ондағы тізімнің сонын көрсету үшін. 5.1, а суретте бірнеше кесінділердің сценасы, үш сипатталған сканирленетін жолмен көрсетілген. 5.1, в суретте сұрыпталған кесінді фигураларының тізімі көрсетілген. Берілген жағдайда нұсқағыштың басы берілген тізімнің басына орнатылады, яғни *BC* кесіндісіне. Нұсқағыштың соңы қарастырылатын сканирленетін жолдан жоғары басталатын тізімдегі сонғы кесіндіге орнатылған, яғни *BD* кесіндесіне. Бейнелерді сканирлегенде САР-ты, түзету қажет, бұл кезде нұсқағыштың соңын тізімге, ағымдағы сканирленетін жолдан немесе одан жоғары басталатын жаңа кесінділер кіргізу үшін төмен жылжытады. Тура сол уақытта ағымдағы сканирленетін жолдан жоғары бітетін кесінділерді алып тастау үшін нұсқағыш басын төмен жылжытады. Бұл сканирленетін жолдар үшін 5.1 суретте көрсетілген, 5.1, а суретте 2 және 3 цифрларымен белгіленген. 5.1, с және d суреттері бұл қарапайым алгоритмде пайда болатын проблемаларды иллюстрациялайды. Бір *y* координатасынан басталатын кесінділерді сұрыптау реті, активті қабырға тізімінің мөлшеріне әсер етеді. Мысалы, 5.1, d суреттегі *BC* кесіндісі ешқашан бул тізімді тастамайды. Нәтижесінде қажеттіліктен көп ақпарат өңделуі мүмкін. Мұны және бұған ұқсас проблемаларды деректердің құрылымын қосымша енгізу жолымен жоюға болады. Бұл кезде сканирленетін жолмен әрбір кесінді бейнесінің қиылысуын есептеуді жеңілдетуге болады.

Барлық бейнелер әрбір бейне кадр үшін өңделетіндіктен, жаймалау жоғарғы интерактивті графика үшін қолданылады. *У* бойынша топтық сұрыптаманы пайдаланғанда дисплейлік тізімнен, оларды оларға сәйкес *y* тобынан және олармен байланыстағы деректер құрылымынан қарапайым қосумен және жоюмен кесінділердің алып тасталынуы немесе қосылуы мүмкін.

Топтап кодтау. Топтап кодтау тәсілінде, үлкен аймақтағы бейнелер бірдей түс немесе интенсивтік бар екенін пайдалануға әрекет етілген. Қарапайым топтық кодтауда тек интенсивтік және берілген сканерленетін жолдағы пиксельдер тізбегінің саны осы интенсивтікпен анықталады (сурет 5.2).

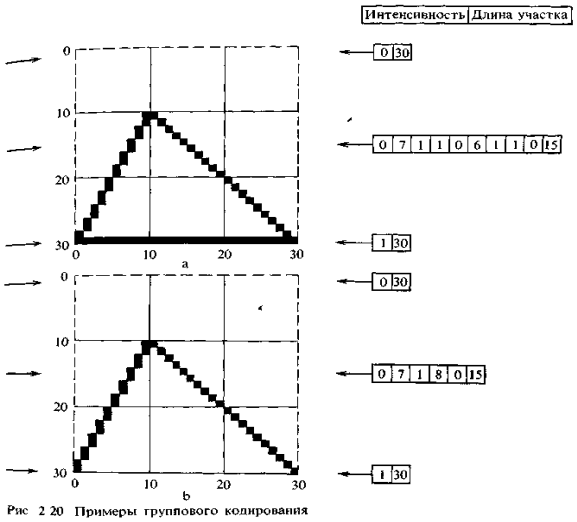


Рис 2 20 Примеры группового кодирования

Сурет 5.2 – Топтық кодтау мысалы

Түсті қосу үшін, топтық кодтаудың қарапайым схемасы жеңіл кеңейтілуі мүмкін. Берілген сканерленетін жолда, түс үшін интенсивтіктің қызыл, жасыл және көк, ал олардан кейін – осы түстермен пиксельдер тізбегінің саны келтіріледі, мысалы,

Интенсивтік қызыл	Интенсивтік жасыл	Интенсивтік көк	Ұзындық

Қарапайым түрлі-түсті дисплей үшін түстілік өшірілген (0),өшірілген (1), 26,b-суретте 15 жолдың көк фонда сары үшбұрышпен кодталуы былай көрінеді.

0	0	1	7	1	1	0	8	0	0	1	15

Топталып кодталған бейнелер үшін деректерді сығу 10:1-ге жақындауы мүмкін. Бұл бар топтап кодтау жадыны жәй үнемдегендіктен ғана емес, сонымен қатар машиносинтезделген кадрлар және фильмдер тізбектері үшін жадты үнемдейді. Ол сол сияқты топтық кодтау кең қоданылатын телеграфпен бергенде фотографиялар және факсимильдер үшін уақытты үнемдейді. Мысалы, 30-секундтық фильмде тығыздығы 512x512x8 бейне кадрлары бейне генерацияның жиілігімен, яғни 30 кадр/с болатын үшін қанша жады қажет екенін қарастырайық. Қажет жады:

(512 x 512 x 8 x 30 x 30)/(8 бит/байт) = 236 Мбайт

Ақпараттың мұндай саны тек үлкен дискалық құрылғыларға сыйады. Бірақ, топтап кодтауда 4:1-ге жайлап сығу оны кіші немесе орташа өлшемдегі бір дискіде сақтауға рұқсат етеді.

Топтап кодтаудың кемшіліктері де бар. Бейнеден кесінділерді қосу немесе жою мәтінді өте көп еібекті қажет ететін операция және учаскелер ұзындықтары тізбектелетін сақталынатындықтан өте көп уақыт алады. Бейнені кодтау және декодтау артық шығынды қажет етеді. Ақырында, бірдей интенсивті қысқа учаскелер үшін, пиксельдік сақталуға қарағанда екі есе көп жад талап етілуі мүмкін (5.3 сурет).

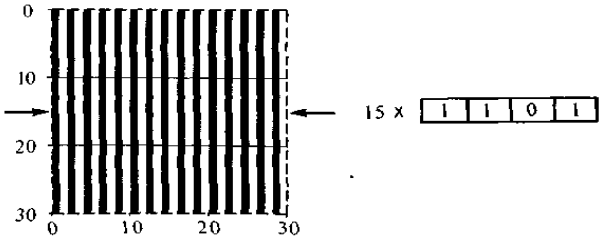


Рис. 2.21. Ограничения группового кодирования для коротких участков.

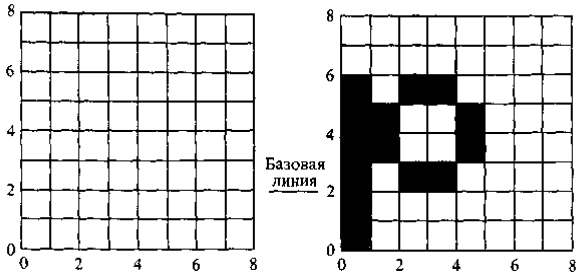
82 Гл. 2. Растровая графика

Сурет 5.3 – Топтық кодтау мысалы

Торлық кодтау. Топтап кодтау тәсілінде бейне сызықтық немесе пиксельдердің бір өлшемдік жиыны деп қарастырылады. Торлық кодтау тәсілінде минимум ақпараттың көмегімен бейненің бүкіл аймағын көрсетуге әрекет істелген, яғни торлар. ЭЛТ-мен қарапайым алфавитті – цифрлық терминалда, нақты уақытта операцияларды орындау мүмкін болу үшін торлық кодтау пайдаланылады. Мұндай терминалда экран аймағы торларға бөлінеді немесе аймақтар бір литераны ұстауға жеткілікті үлкен. Мысалы, экранды

өлшемі 8х8 пиксель аймаққа бөлуге болады. *Разрешениясы* 512х512 дисплей үшін 64х64 тор, ал стандартты бейне қатынасы 4:3 телевизиялық 480х640 дисплей үшін 60х80 тор. Әдетте 8х8 пиксельдік тор өлшемі 5х7нүктелі матрица литерін шығару үшін пайдаланылады. Қосымша пиксельдер литерлерді бөлу үшін, сонымен қатар, элементтері төмен шығарылған жолдық литер үшін пайдаланылады. 5.2 суретте литер маскаларының мысалы келтіріледі. Тордың әрбір екінші қатары оқылу үшін бос қалдырылады., онда соңғы схема үшін көптеген алфавитті-цифрлық дисплейлерге лайық 80 литерден 30 жол жасалынады. Торлардың басқа да өлшемдері пайдаланылады. Мысалы, 7х9 матрицалық литер үшін, әдетте, 8х10 пиксельдік тор пайдаланылады, нәтижесінде дисплейде әрқайсысы 80 литерден 30 жол бар. Әр литер үшін, пиксельден құрастырылған шаблондар тұрақты есте сақтау құрылғысында (ТЕСҚ-ПЗУ) сақталады.

Торлап кодтау тәсілін сызықтарды сызғанда пайдалануға болады, тек ТЕСҚ-да кесінділер сегменттерінің шаблондарын да сақтау керек. Онда, қажетті сызықтарды тұрғызу үшін, көрші торларда орналасқан сондай сегменттердің комбинациясына пайдалану мүмкіни (сурет 5.4).



Сурет 5.4 – Торлық кодтау мысалы

Өлшемі *n*х*n* ерікті тор үшін пиксельдерден тұратын 2^{n2} шаблондар бар. *N*-нің кез келген мәні үшін өте көп шаблондарды сақтауға тура келер еді. Мысалы, *n*=8 тең болғанда 2^{n2}-нің мәні $2^{n2}=1.8x10^{19}$. Бірақ, барлық шблондар нақты мүмкін сегменттерді көрсетпейді. Мысалы, жоғарыда талданған Брезенхем алгоритмі үшін 0 мен 1 арасындағы тангенс бұрышымен иілген кесінділер үшін 2^{n}-1-ден көп емес кесінділер сегментін көрсететін шаблондар бар.Ақыры, Жордан және Баррет 8х8тасымалдауды пайдаланғанда шағылысу және қалқалауға кесінді сегменттерінің тек 108 шаблоны талап етілетінін көрсетті.

Торлық кодтау тәсілі түрлі-түсті дисплейлерге және біртегіс бейнелердің көрсетілуіне бөлінген болатын. Бұл кезде, бірақ, деректерді сығу коэффициенттері, ақ-қара (екі деңгейлі) бейнелеудегідей үлкен емес.

Кадрлар буфері. Қайта өндіретін растрлық графикалық құрылымдармен танысқанда растрлық дисплейерікті қатынастағы жартылай өткізгішті жадтан тұратын кадр буфері түрінде есептелінген. Бұл жиірек кездесетін іске асыру тәсілі болғанымен, бірақ кадр буфері үшін екінші жад – диск немесе барабан пайдалануы мүмкін.

Кадрлар буферін ығысу регистрларының көмегімен іске асыруға болады. Схема түрінде ығысу регистрін FIFO (бірінші келді – бірінші қызмет көрсетілді) типті стек деп есептеуге болады. Егер стек толтырылған болса, онда стектің төбесіне деректердің жаңа биттерін қосқанда, түбінен деректердің алғашқы биттері сканерленетін жолдардың пикселдерінің интенсивтілігі сияқты интерпретациялауға болады. Ығысу регистрларындағы кадрлар буферін әр регистрдің ұзындығы, жол санына тең болғанда сканерленетін жолдың пикселіне бір регистрден пайдалана отырып, іске асыруға болады. Басқа вариант – ұзындығы, жол санына көбейтілген сканерленетін жолдағы пикселдер санына тең регистрді пайдалану.

5.2 Жазық алаңның растрлық өрбітуі

Осыған дейңн түзу сызықтың кесінділерін растрлық графикалық құрылымда көрсету туралы сөз болды. Бірақ, мұндай құрылымның бірегей сипаттамасының бірі *сплошной* аймақты көрсету мүмкіндігі сплошной аймақтың генерациясы қабырғаның қарапайым сипаттамасын немесе төбесіне *сплоштың* аймақтың растрлық жаймасы көпбұрыштарды толтыру немесе контурларды толтыру деп атаймыз. Бұл үшін бірнеше әдетте, екі кең категорияға бөлінетін: растрлық жаймалау және *затравочное* толтыруды пайдалануға болады. Растрлық жаймалау тәсілінде жолдарды сканерлеу ретінде нүкте көпбұрыштың немесе контурдың ішінде жатқанын анықтауға тырысады. Бұл алгоритмдер, әдетте көпбұрыштың немесе контурдың «төбесінен» «төмен» жүреді. Жаймалау тәсілдері векторлық дисплейлерге штрихтау немесе контурды бояу үшін пайдалануға болады. Толтыру тәсілдерінде тұйық контурдың ішінде кейбір нүктелер белгілі деп алынады. Алгоритмдерде затравочныйлармен көрші және контурдың ішінде орналасқан нүктелерді іздейді. Егер көршілес нүкте ішінде орналаспаған болса, контурдың шекарасы табылған, егер нүкте контурдың ішінде болса, онда ол жаңа затравочный нүкте болады және іздеу рекурсивті жалғасады. Мұндай алгоритмдерді тек растрлық құрылымдарда ғана пайдалануға болады (5.5 сурет).

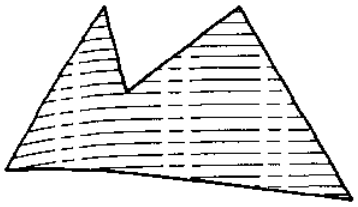


Рис. 2.32. Штриховка или закраска контура.

Сурет 5.5 – Контурды штрихтау

Көпбұрыштарды толтыру. Көптеген тұйық контурлар қарапайым көпбұрыш болады. Егер контур қисық сызықтардан тұратын болса, онда оны дәл келетін көпбұрышпен немесе көпбұрыштармен аппроксимациялауға болады.

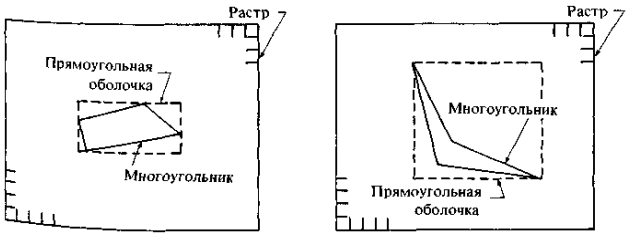


Рис 2.33. Прямоугольная оболочка многоугольника.

Сурет 5.6 – Көпбұрыштың тік бұрышты қабықшалары

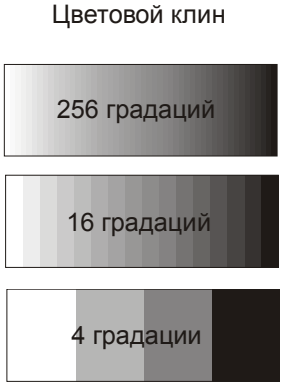
Көпбұрышты толтырудың қарапайым тәсіліне көпбұрыштың растрына әрбір пикселінің ішкі екенін тексеру. Әдетте көптеген пиксельдер көпбұрыштан тыс жататындықтан, бұл берілген тәсіл өте қымбат. Шығынды, тікбұрышты қабықшаның көпбұрыштары үшін өз ішінде көпбұрышы бар, кішірек тікбұрышты есептеу жолымен кішірейтуге болады. 5.5 суретте көрсетілгендей бұл қабықшаның тек ішкі нүктелерін ғана тексеріледі. 5.6, *а* суретте бейнеленген көпбұрыштар үшін бұрыштың қабықшаларын пайдалану тексерілетін пиксельдер санын біраз қысқартады. 5.6, *в* суретте көрсетілген көпбұрышқа сол уақытта қысқарту аз.

Негізгі әдебиет: 2 [стр.73-96]
Қосымша әдебиет: 11 [стр.66-90]
Бақылау сұрақтары:
1. Растрлық жаймалау дегенімізне?
2. Сіздер нақты уақыттағы растрлық жаймалауды қалай түсінесіздер?
3. Топтық кодтау дегеніміз не?
4. Топтық кодтау не үшін қолданылады?
5. Торлық кодтау дегеніміз не?

Дәріс 6. Қадамдықты жою әдістері

Айталық бізде тек екі түрде ақ және қара бар дейін және бізге түстер сапасын құру керек ол үшін сұр түстің түрлі түсін алуымыз керек (6.1 сурет).

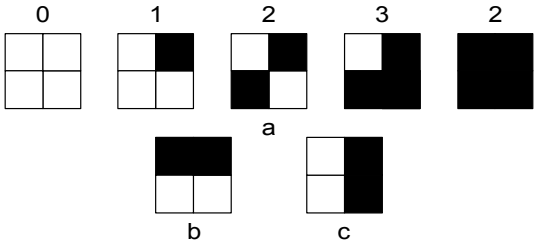
Бұл тапсырманы шешу үшін баспалдақты жою методы жүзеге асырылады.



Сурет 6.1 – Түсті клин

6.1 Полутон әдісі

Маңыздылық бастапқы кескіннің әрбір пикселі пиксельдер тобымен айырбасталынады (6.2 сурет).



Сурет 6.2

Осындай ұйымдастыру кезінде бес мүмкін деңгей немесе сұр тондар (0-4) алуы мүмкін Жалпы екіденгейлік дисплей жағдайында мүмкін интенсивтік саны клеткадағы пиксельдер санымен бірге үлкен конфигурацияны таңдау кезінде байқау керек, әйтпесе кішімасштабты құрылымдар пайда болуы мүмкін. Мысалы, 6.2 суретте кескінделген конфигураның біреуінде қолданбау керек, немесе ол кескінде қажет емес горизонтальді немесе вертикальді сызықтар пайда болуына әкеп соғады.

Интенсивтіліктің жеткен деңгей санын клетканың өлшемін ұлғайту көмегімен арттырамыз.

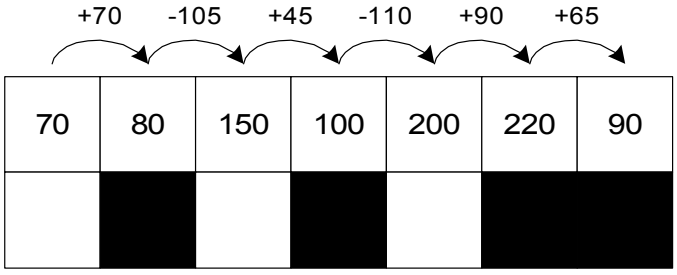
Конфигурацияны қолдану кеңістіктің рұқсатын жоғалтуға әкеледі.

6.2 Тасымалдау әдісі

Берілген әдісте кескін рұқсаты өзгермейді.

Маңыздылық: сұрының градиациясының жуық саны беріледі, осы диапазон ортасы есептеледі және кескін оңнан солға және төменнен жоғарыға өтеді, кескіннің нүктелері қарастырылады: егер нүкте жарықтылығы аққа жақын болса, онда ақ нүкте қойылады, ал егер қараға жақын болса, онда қара нүкте қойылады, алынған қателіктің жартысы. Сол жақтағы нүкте жарықтығына қосылады, ал екінші жартысы төмендегі нүктеге (пропорцияны өзгертуге болады) қосылады.

Берілген алгоритмді түзу үшін жүзеге асыру мысалы, 6.3 сурет.



Сурет 6.3

Тасымалдау әдісі градацияның үлкен саны жағдайында қолдануға болады. Жақсы нәтижелер мына жағдайда алынады әрбір жол үшін кездейсоқ жағдаймен алынса қателіктің алғашқы мәні. Осылайа суреттің жүйелілігі жойылады.

Негізгі әдебиет: 2 [бет119-139]
Қосымша әдебиет: 11 [бет 91-130]
Бақылау сұрақтары:
1. Баспалдақты жою әдістері не үшін қолданылады?
2. Баспалдақты жою әдістерінің қандай түрлері бар?
3. Полутон әдісінің мақсаты не?
4. Тасымалдау әдісінің мақсаты не?
5. Жоғары көрсетілнген әдістердің қайсы жақсы және жаман?

Дәріс 7. Геометриялық түрлендіру

7.1 Координаттар және түрлендіру. Жазықтықта аффиндік түрлендіру

Геометриялық объектіні конструкциялау, сипаттау және ұсыну геометриялық жүйенің орталық жұмысы болып табылады. Олардың математикалық әдістері, алгоритмдері және программалары сәйкестігі ретінде талап ететін көлемінде қолдануы графикалық жүйенің тиімділігі мен мүмкіндігіне маңызды әсер етеді.

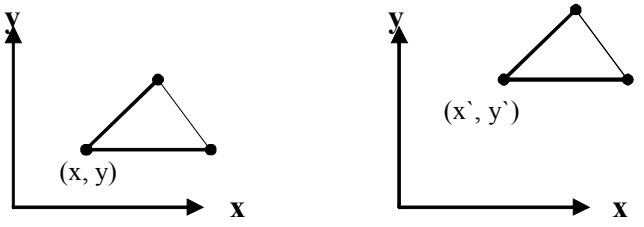
Берілген бөлімде екі немесе үш өлшемді координаттарды геометриялық түрлендіруді сипаттауға арналған математикалық әдістері қарастырылады, растрлық картинаны геометриялық түрлендіруде қарастырылған кейбір тиімділік сұрақтары талқыланған.

Әрі қарай *X*, *Y*, *Z* үлкен әріптермен кәдімгі департтық координаттар белгіленеді, ал *x*, *y*, *z* кіші әріптер біртекті координаттарды белгілеу үшін қолданылады.

7.2 Координаттардың бір жүйеге түрлендірілуі

Компьютерлік графикада екі өлшемдіге қатыстының бәрі 2D (2 dimension) символымен белгілеу қабылданған. Мұнда түрлендірудің үш түрін қарастырамыз: параллельдік көшіру,масштабтау, бұру.

Параллельді көшіру (7.1 сурет).



Сурет 7.1 – Параллельді көшіру

Айталық жазықтықтағы түзу сызықты координаттық жүйе жүргізілген. Онда әрбір нүктеге сәйкес салдар жұбы (x, y) олардың координаты қойылады. Жазықтыққа тағы да бір түзу сызықты координат жүйесін жүргіземіз, содан кейін біз сәйкес нүктеге келесі сандар жұбын – (x', y') қоямыз.

$$x' = x + Dx$$
$$y' = y + Dy$$
$$[\underbrace{x', y'}] = [\underbrace{x, y}] + [\underbrace{Dx, Dy}]$$

немесе векторлық формада

$$P' = P + T,$$

мұндағы,

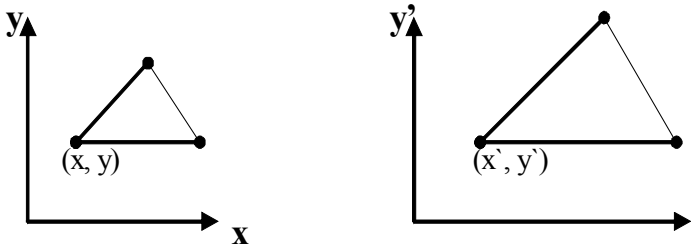
$P' = [x', y']$ – координатқа түрлендірілген жол векторлық;

$P = [x, y]$ – бастапқы координатаның жол векторы;

$T = [Tx\ Ty]$ – жылжу жол векторы;

x', y' – нүктенің бастапқы координат;

Tx, Ty – ось бойынша жылжу шамасы; масштабтау.



Сурет 7.2

Масштабтауды түрлендіру бастапқы координат көбейтіндісі сияқты көрсетіледі және түрлендіру матрицалары және координатаның салыстырмалы басы мына түрде болады.

$$\begin{cases} x' = x \cdot Sx \\ y' = y \cdot Sy \end{cases}$$

немесе

$$\underbrace{\begin{pmatrix} Sx & 0 \\ 0 & Sy \end{pmatrix}}_{S}$$
$$[\underbrace{x', y'}_{P'}] = [\underbrace{x, y}_{P}]$$

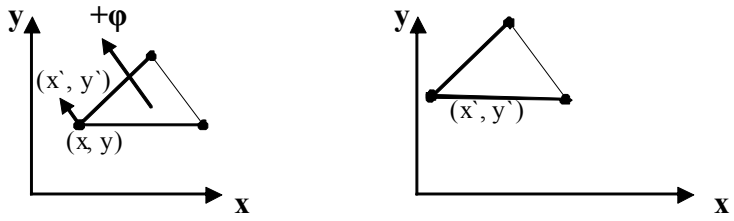
немесе матрицалық формада

$$P' = P \cdot S,$$

мұндағы, Sx, Sy – осьтер бойынша масштабтау коэффициенттері, ал

S – түрлендіру матрицасы

Бұрылыс



Сурет 7.3

Координат басына салыстырмалы түрлендірудің түрі мынадай:

$$\begin{cases} x' = x \cdot \cos(\varphi) - y \cdot \sin(\varphi) \\ y' = x \cdot \sin(\varphi) + y \cdot \cos(\varphi) \end{cases}$$

немесе

$$[x`, y`] = [x, y] \bullet \underbrace{\begin{pmatrix} \cos(\varphi) & \sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{pmatrix}}_{R}$$

мұндағы:

R – бұрылыс матрицасы;

Ф – бұрылыстың оң бұрышы (сағатқа қарсы бағытта).

Біртекті координат жүйесіне түрлендіру.

Жоғарыда көргендей екі өлшемді түрлендірудің әртүрлі түрі бар. Жылжыту қосумен, ал масштабтау мен бұрылыс – көбейтумен жүзеге асырылады. Осы айырмашылық суммарлық түрлендірудің қалыптасуын қиындатады.

Жазықтағы ерікті нүктелерінің реттелмеген сандар жұбы сипаттамасына көшеміз, декарттық жүйеде де сол сияқты, онда реттелген үш сан.

$$[X, Y, W].$$

x және *y* координаттарымен жазықтықтың ерікті нүктелерінің біртекті дкоординаттары берілген, бір уақытта нөлге тең емес берілген координаттармен байланысты *X*, *У*, *W* кез келген үш сан келесі қатынаста болады:

$$x = \frac{X}{Y}; y = \frac{Y}{W}; W \neq 0$$

Біртекті координаттарды *Z=W* жазықтықтарда орнатылған екіөлшемді координатты *W* коэффициенті мәнімен масштабтау сияқты көрсетуге болады. Біртекті координаттарда *W* ерікті мәнінде декараттық координата берілген нүктенің бір ғана көрсетілуі болмайды. Координат центріне салыстырмалы біртекті координаттарда параллельді көшіру, масштабтау және бұруды түрлендіруде бастапқы координат векторының түрлендіру матрицасына көбейтіндісі барлығы бірдей формада болады.

Үш біртекті координат және үшінші реттік матрица көмегімен жазықтықтағы кез келген аффиндік түрлендіруді сипаттауға болады. Шындығындпа *W*=1екенін ескеріп жоғарыдағы қарастырылған операцияларды сәйкес матрицалардың көбейтіндісі түрінде жазуға болады.

Параллельді көшіру,

$$\begin{pmatrix} 1 & 0 & 0 \\ [X', Y', 1]=[X, Y, 1] & & \\ 0 & 1 & 0 \\ Dx & Dy & 1 \end{pmatrix};$$

Көбейтіп, мынаны аламыз: [*X*+*Dx*, *Y*+*Dy*, 1]. декарттық координаттар жүйесіне көшу [*x*+*Dx*, *y*+*Dy*]. береді.

Масштабтау:

$$P' = P \cdot S;$$

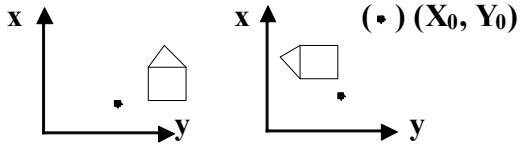
$$S = \begin{bmatrix} S_X & 0 & 0 \\ 0 & S_Y & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

Бұрылыс:

$$P' = P \cdot R;$$

$$R = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

Жазықтықты аффиндік түрлендіруге мысал қарастырамыз. Бекітілген нүкте айналасында бұру есебін оны қарапайымдырақ үш ішкі есепке бөлу арқылы шешу оңайырақ: ығыстыруды түрлендіру, бұрылысты түрлендіру ығыстыруды түрлендіру.



Сурет 7.4

$$P' = P \cdot M,$$

Ығыстыруды түрлендіру

Бұрылыс центрін координаттар басымен қосу үшін T(-X_0, - Y_0) векторына көшу көрсетіледі.

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -X_0 & -Y_0 & 1 \end{pmatrix}$$

Негізгі әдебиет: 4 [бет63-94]

Қосымша әдебиет: 11 [бет 119-139]

Бақылау сұрақтары:

1. Координаттарды түрлендіру дегеніміз не?
2. Координаттарды сызықты және сызықты емес түрлендіру дегеніміз не?

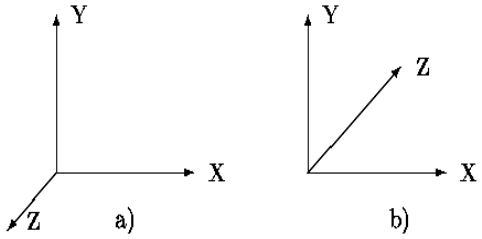
3. Координаттардың аффинді жүйесі дегеніміз не?
4. Жазықтықта параллель жылжыту дегеніміз не?
5. Жазықтықта сығу мен созу қалай іске асрылады?

Дәріс 8. Объектілерді түрлендіру

8.1 Жазықтықта обьектілердің аффиндік түрлендірілуі. Обьектілердің үшөлшемді аффиндік түрлендіріліуі

Үшөлшемді сұлбаның синтез алгоритмі

Енді үшөлшемді жағдайда (3D) – (3-dimension) қарастырамыз. Әрі қарай үшөлшемді түрлдендіру кезінде негізінен векторлық алгебрада жалпы қабылданған оң координат жүйесі қолданылады (8.1, а сурет). Сонымен қатар, егер координат центрінің оң жартылай ось жағынан қарастырсақ, онда бұрылыс $+90^0$ (сағат тіліне қарсы) бір оң осьті келесіге (осьтен қашық орналасқан қозғалыс бағыты және оң винттің сағат тіліне қарсы бұрылатын бағыты және оң жартылай ось сәйкес) ажыратылады. Кейбір арнайы ескертілген жағдайларда координаттың жағы қолданылады (8.1, б сурет). Үшөлшемді машиналық графикада ең ыңғайлы болып сол координат жүйесі табылады. Мысалы, *X*, *Y* жазықтығымен экран бетін қосарлау, онда бақылаушыдан үлкен аумақтану нүктесі *Z* үлкен мәнімен сәйкеседі (8.2, б сурет).



Сурет 8.1

Біртекті үшөлшемді координаттар мен түрлендіру матрицаларымен жұмыс екіөлшемді жағдай үшін ұқсас, сондықтан ығысуды масштабтауды және бұрылыс түрлендірудің матрицалары ғана қарастырылады және оған белгілі нәтиже бойынша түрлендіру матрицасын конструкциялау.

Сол тәрізді нүкте екіөлшемді жағдайда біртекті координаттарда үшөлшемді вектормен беріледі [*X*, *Y*, *W*], ал түрлендіру матрицалары 3х3 өлшемді болады, үшөлшемді жағдайда нүктеге төртөлшемді жағдайда көрсетіледі [*X*, *У*, *Z*, *W*], мұнда *W* нөлге тең емес, ал түрлендіру матрицаларының өлшемі 4х4 болады.

Түрлендіруге арналған формулалар: $x = \frac{X}{W}$; $\text{у} = \frac{Y}{W}$; $\text{z} = \frac{Z}{W}$; $W \neq 0$;

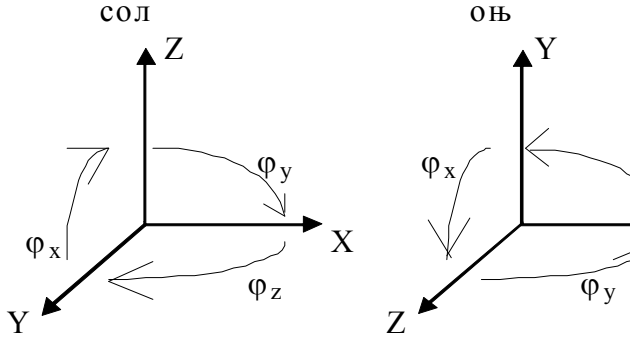
Параллельді көшіру

$$P' = P \cdot T$$

Бұрылыс. Үшөлшемді кеңістікте бұрылыстың бірінші ерекшелігі, бұл кеңістіктегі бір бұрышы 3 бұрышқа таратуға болады: абсцисса осі айналасында, ордината осі айналасында, апплита осі айналасында. Екінші ерекшелігі – бұл

бұрылыс бұрышы бағыты. Екі координаттар жүйесі үшін бұрылыс бұрышының оң бағыты қысқарақ өту жүзеге асатын бағыт деп саналады.

Сол координаттар жүйесінде сағат бағыты бойынша бұрылыс оң, егер жартылай осьтің оң ақырынан қарасақ: оң үшін – сағат тіліне қарсы (8.2 сурет).



Сурет 8.2

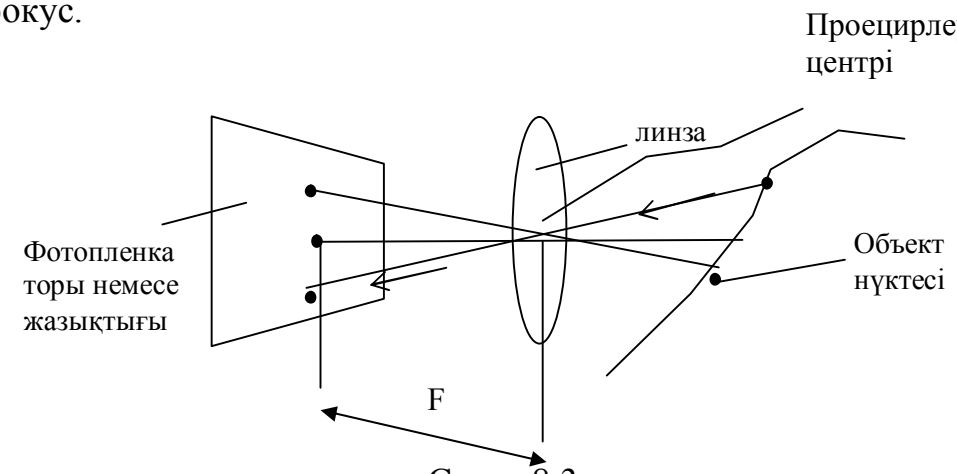
Жоғарыда қаралған матрицаның екіөлшемді жағдайында бұрылыс сол уақытта *Z* осінен қашық өлшем өзгермейді, онда үшінші жол мен бағанның барлық элементтері 0-ге тең 1-ге тең диагональдіктен басқасы:*X* осі айналысында (*Y*, *Z* жазықтығында) бұрылыс кезіндже *X* осінен қашықтық өлшемі өзгермейді, сондықтан бірінші бағанның бірінші жолының барлық элементтері 0-ге тең, 1-ге тең диагональдіктен басқасы

$$Rz(\varphi_z) = \begin{pmatrix} \cos(\varphi_z) & \sin(\varphi_z) & 0 & 0 \\ -\sin(\varphi_z) & \cos(\varphi_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Y осі айналысында (*X*, *Z* жазықтарда) бұрылыс кезіндже *Y* осінен қашықтық өлшемі өзгермейді, сондықтан бірінші бағанның бірінші жолының барлық элементтері 0-ге тең, 1-ге тең диагональдіктен басқасы.

Проекциялау. Жазықтықтағы объектілер кескіні тағы да бір операциямен байланысты – түзулер будасы көмегімен проекциялау. Картинкалық жазықтықта объектінің проекицясын алу үшін берілген проекцияланатын ауаның әрбір нүктесінен түзу жүргізу және содан кейін кескін жазықтығымен осы түзудің қиылыс нүктесінің координаттарын табу (8.3 сурет).

F- фокус.



Сурет 8.3

Геометриялық проекцияны проектор түзу болған кезде аламыз. Осыдан кез келген түзу сызықтың қиындысы кеңістікте жазықтықтағы түзусызық қиындысына түрленеді.

Кеңістік пен беттік координаттар қатынасының есебін қарастырамыз.

8.2 Проекциялар. Орталықтандырылған проекцияландыру. Параллельді проекцияландыру

Центрден проекциялағанда барлық түзулер бір нүктеге келеді. Бұл бума центрі матрицалық проекцияны қарастыру үшін әлі бірнеше түсініп енгіземіз:

1. Пиксель, нүкте – проекцияланатын бума центрі;
2. Суреттік жазықтық – объект проекцияланатын жазықтық;
3. Проекциялау қажет объект;
4. Проектор – объект нүктесі және центрі арқылы өтетін сызық (жалпы жағдайда проекторлар бұл түзу емес сызық).

Егер бет картинасы – жазық болса, онда жазық болса, онда жазық проекциясы туралы айтылады. Егер проектор түзу сызық болса, онда геометриялық проеция туралды айтылады. Осы екі фактор да болған жағдайда жазық геометриялық проекция (ПГП-ЖГП) жайында айтылады. ЖГП фотоға түсіру және көру кезінде көрінеді.

Дүниежүзілік координат жүйесін енгіземіз және басқарушыны қарастырамыз. Проекциялау центр бақылаушымен байланысты.

Кеңістікте жазықтықты бағыттау үшін беріледі: негізгі жағдайы – крен жоқ негізгі жағдай. Мұнда x осі хоу әлемдік координат жүйесіне параллель және негізгі өске бұрылыс бұрыш салыстырмалы.

Тәуелділікті табу керек:

$$\begin{cases} x = x(X, Y, Z, параметрлер) \\ y = y(X, Y, Z, параметрлер) \end{cases}$$

Проекциялау параметрлері (·) P(x_P, y_P, z_P), F(L, N_1, N_2). жазықтығының бағыттаушы.

Екі көмекші координаттар енгіземіз:

бір штрихпен

+X′ Y′ Z′: PX′||ox

PY′||oy

PZ′ – жалғасы LP

екі штрихпен:

X″ Y″ Z″: PX″||OX

PY″||OY

PZ″||OZ

Жүйеде кеңістіктің нүктелердің координаттарын PX′Y′Z′ сонымен қатар көрнекті (видовой) (.)t(x, y, -F). координат жүйесінде T(X′,Y′,Z′), проекицясын қарастырамыз.

РТ түзу теңдеуін құрамыз.

$$\frac{X^{|}}{x}=\frac{Y^{|}}{y}=\frac{Z^{|}}{-F};\qquad \frac{P'_{xyz}}{T,t:T:\mathrm{X'Y'Z'}}\qquad \begin{cases} x=-F\cdot\dfrac{X^{|}}{Z^{|}} \\ y=-F\cdot\dfrac{Y^{|}}{Z^{|}} \end{cases}$$

$$t:x,y,-F$$

Берілген теңдеуді матрицалық түрде жазамыз.

$$\left[-\frac{x}{F}\quad -\frac{y}{F}\quad \mathrm{A}\quad 1\right]=[X^{|}\ Y^{|}\ Z^{|}\ 1]\cdot M;$$

$$M=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&\alpha&0\end{bmatrix};$$

(2) мен (1) эквиваленті екенін көз жеткіземіз.

$$[X^{|}\ Y^{|}\ Z^{|}\ 1]\cdot M=[X^{|}\ Y^{|}\ \alpha\ Z^{|}];$$

Декарттық координаттар жүйесіне көцшеміз.

$$\left[\frac{X^{|}}{Z^{|}}\frac{Y^{|}}{Z^{|}}\frac{\alpha}{Z^{|}}1\right];$$

Алынған (2) көрнекті салыстыру керек болады.4

$$-\frac{x}{F}=\frac{X^{|}}{Z^{|}};\quad -\frac{y}{F}=\frac{Y^{|}}{Z^{|}};\quad A=\frac{\alpha}{Z};\ \alpha\neq 0.\qquad \equiv \mathrm{F}$$

α=1. [*x y … 1*] = [x' y' z']×M' қабылдаймыз

$$\mathrm{M}=\begin{vmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{vmatrix};\ \mathrm{M^{-1}=M^{T}=M}$$

S= –Z′ болса «тереңдік» /–S^{-1} = 1/Z', енгіземіз S шамасын кеңістіктегі нүкте тереңдігі бақылаушы алдында (таңба +) немесе артында (таңба -) орналасқандығы айтуға болады.

Негізгі әдебиет: 2 [бет.119-139]
Қосымша әдебиет: 2 [бет.119-139]
Бақылау сұрақтары:
1. Сіз үшөлшемді аффиндік түрлендіруді қалай түсінесіз?
2. Объектілерді түрлендіру координат түрлендіруінен қалай ерекшелінеді?
3. Объектіні сығу және созу қалай іске асырылады?

4. Объектіні бұру қалай жүргізіледі?
5. Объектіні жылжыту қалай жүргізіледі?

Дәріс 9. Компьютерлік графикадағы түс

9.1 Түстер теориясының негізгі түсініктері. Түстік модельдер. RGB аддитивті түстік модельдері. CMY(K) түстік моделі. HSB және HSL түстік моделі

Түспен біз барлық жерде кездесеміз. Ол біздің демалып отыған ауа сияқты. Біздің өміріміздің негізгі компонентті. Бізді таң қалдырған қарапайым түстер бірлестігіне көңіл аударсақ өз-өзімізге түс дегеніміз не? деген сұрақ туындайды. Тоқтап ағаш басындағы жасыл түстің түрлі түсіне көңіл аударсақ , біз олардың өзара айырмашылығымен басқа ағаш айналасындағы жасыл шөптерден де айырмашылығы бар. Күн ашық күндегі жарық пен көлеңкені байқаңыз.

Ғылымда адам көзі жануарлардан айырмашылығы түсті айыра алатындығын дәлелдеген. Көз сырттан ақпараты қабылдайтын каналдар бірінің функциясын атқаратындықтан, түс оны интерпретациялау процесінде маңызды роль атқарады. Түстің адамға әсері көп. Ол күнделікті өмірде біздің көңіл күйімізді анықтайды. Біздің жұмыс қабілетімізге және психологиялық жағдайымызға әсер етеді.

Түстік модельдер

Түстік модельде құру негізінде әмбебап тілдерді қолдану жатыр, олар математикалық стандартты өрнектер көмегімен нақты түсті сипаттау тәсілін іске асыруға көмектеседі. Олардың көмегінсіз сандық кескіндердің баспаның редактрлеу,сканерлеудің өңдеудің ешбір кезеңін орындай алмаймыз. Қазіргі компьютердегі программада түспен басқару режимдерге түстік модель көмегімен іске асырылады. Түстік модельдер (немесе түстік кескіндік) түсті сипаттаудың концептуальды және мөлшері үшін амалдарын ұсынады. Негізгі түсті концептуальды ұсынылын танысқаннан кейін сіз жұмыс барысында түстер арасындағы қатынасты жақсы түсінесіз. Режим бұл анықталған түстік модельдердің нақты графикалық программа көлемінде іске асырылу тәсілі.

Түстік модельдің түсінігі

Түстік модельдер (color model) спектрдің анықталған түстік ауданын математикалық сипаттау үшін қолданылады. Көптеген компьютерлі түсті модельдер адамдар көзімен қабылданатын үш негізгі түстер қолдануға негізделген. Әр негізгі түске анықталған сандық код мәні меншіктеледі, содан кейін қалған түстердің барлығы негізгі түстер комбинациясы ретінде анықталады. Осындай тәсілді сүретшілер шектеулі түстер палитрасы базасында сүрет салу үшін қолданады. Түстік модельдер түсті математикалық ұсынылуына қарамастан әлсіз. Бірақ олар компьютерлік программада шығарылатын түстің бірмәнді анықталуы үшін қолданылу ыңғайлы. Егер мониторға R.255 G000 Б255 түсті сигналын жіберсек онда жақсы мониторда жақсы бір түс пайда болу керек. (берілген жағдайда) өзінің негізінде не жатқанына тәуелсіз кез-келген модель үш талапты қанағаттандыруы керек:

– қандай болмасын нақты құрылғы мүмкіндіктеріне тәуелсіз, түстерді кейбір стандартты әдіспен анықтауды іске асыру;

– қайта өңдейтін түстер диапазонын нақты беру;

– түстерді қабылдау механизімін ескері шағылысу немесе сәулелену.

Қазіргі графикалық дестелер керек түсті модельді таңдау үшін дамыған интерфейстер орналасады. Ары қарай бұл бөлімде қазіргі графикалық дестелерде қолданылатын көптеген түстік модельдер толық қарастырылады.

Түстік модельдер типі

Графикалық дестелердің көпшілігі түстік модельдердің кең көлемімен операцияларды олардың бір бөлігі арнайы мақсаттарға құрылған, ал келесі бөлігі-бояудың маңызды типтері үшін атап өтелік CMY, CMYK, RGB, HSB, HLS, Lab, YIQ және YCC.

Әрекеттер принциптері үшін аталған түстік модельдерді үш класқа бөлуге болады:

– аддитивті (RCB), түстерді қоюға негізделген;

– субтрактивті (СМУ, СМУК) есептеу операциясының негізін құрайды (субрактивті синтез);

– перцепционды (HSB, HLS, Lab, YCC) қабылдауға қор жинайды.

Нақты түстік модельге көшпестен бұрын түс табиғатына тән физикалық заңдылықтарға назар аударайық.

Аддитивті түстік моделдер

Аддитивті түсті әртүрлі түстер жарық сәулелерін біріктіру жолымен Грассман заңы негізінде алынады. Көптеген түстері үш негізгі түстік код көрінетін спектрдің компоненттерін араластырылғанда алынады. Мұндай түстер теорияда бастапқы түстер деп аталады, қызыл (Red), жасыл (Green) және көк (Blue) түстер. Бастапқы түстерді жұпты араластырғанда екінші түстер пайда болады: көгілдір (Cyan), сары (Yellow) бастапқы және екінші түстер қорлық үстерге жатады. Көрінетін түстердің барлық спектрын алуға болатын түстерді қорлық түстер деп атайды.

Аддитивті синтез көмегімен жаңа түстерді алу үшін әртүрлі комбинациялауға 2 негізгі түстерді қолдануға болады. Қызыл және жасыл түстерді қолданып екі бастапқы қорда жаңа түстерді алу сұлбасы келтіріледі. Қызыл және жасыл әртүрлі пайызды екі бастапқы шешім базасында түстерді жаңа түстер аддитивті синтез.

Аддитивті түстер жарақаттану жүйелерінде видеожүйелерінде, фотоүлгілерінде, жазба құрылғыларында, мониторда, сканерде, сандық комераларда, кең қолданыс тапты. RGP модельдерді тұрғызу үшін қолданылатынбастапқы немесе Аддитивті түстердің тағы бір атауы. Кейде интенсивті жарықты түске қосқан кезде түстер көбейеді мұндай модельдерді қосатын немесе терминдер деп аталады, олар RGB –модельді сипаттау үшін қолданылады.

Субрактивті түстік модельдер

Монитор экранына қарағанда жарық шағылысуына негізделген түстерді қайта өндіру, баслатын бет тек түсті ғана бейнелейді. Сондықтан бұл жағдайда RGB модель қолайсыз. Оның орнына басылатын түстрді сипаттау үшін

субрактивті түстерге бағаланатын сму моделі қолданылады.

Аддитивті түстердің субрактивті түстерден айырмашылығы жарықтың жалпы сәулесінен екінші түстерді есептеунде.

Бұл жүйеде ақ түс барлық түстер болмағанда пайда болады, олар болғанда қара түс береді.

Соңғы кезде «субрактивті» терминнің синонимі деп «шығарылатын» термині қолданылады. Бұл атау бояғыш жабылған жарықтың шағылысуына байланысты пайда болады.

Үш базалық түстің көгілдір, қара қошқыл, сары қағазға түсуі көптеген субрактивті түстерді құруға мүмкіндік береді.

Түстің субрактивті синтез негізінде жатқан физикалық процестер «шағылысқан және бейнеленген түс» бөлімінде қаралған. Біз осы модельді қолданудың кейбір бөлшектерін атап өтеміз. Біз аддитивті және субрактивті түстерді байланыстратын қатынас жазуымыз керек:

Жасыл+көк□көгілдір

Жасыл+қызыл□сары

Қызыл+көк **=** қара қошқыл

Жасыл+көк+қызыл **=** ақ

Көгілдір+сары+қара қошқыл **=** қара

Сонымен бояғыш жағылған қағазға ақ жарық түскенде не болады. Егер бояғыш көгілдір болса онда, спектірден қызыл түсті алып, көгілдір түсті бейнелейді.

Қарақошқыл жасыл түсті, сары – көк түсті, егер басу кезінде қара қошқыл мен сары түсті қойса, онда қызыл түс алынады. Субрактивті түсті үшеуінде қойса, онда нәтижесі қара болады.

Полиграфияда бояйтын затты баспа бояуы д.а. бояу сүйық немесе қатты пигмент бөліктерінен тұрады. Қатты пигмент бөлігінің орнына бояғыштар қолданатын бояулар бар, лоарды әдетте қара сия деп атайды.

Бояулар, сиялар тлерлер триадты немесе аралас деп бөлінеді. Триядты бояулар спектральды сипаттамалармен сәйкестелген: сары көкте, қарақошқыл жасылда, көгілдір қызылда.

Аралас бояулар жеке түстерді алу үшін триадты синтез бояуларына қосымша ретінде қолданылады.

СМУ және СМУК

Субрактивті модельдің кең таралған екі версиясы бар: СМУ және СМУК. 1-сі сурет немесе кескін ақ-қара принтерде шығарылатын болғанда қолданылады. Оның негізінде үш субрактивті түстер қолданылады. Көгілдір, қарақошқыл, сары.

Бұл үш түсті араластрғанда ақ қағазда қара түс пайда болады.

Бірақ қазіргі техникалық процестерде үш түсті араластрып қара түс алу үш себепке байланысты тиімсіз:

– ашық таза қара қошқыл көк, сары болуды қайта өндіру мүмкін емес. Сондықтан түс ашық қара емес, лай қоңыр болып шығады;

– СМУ моделінің көмегімен қара түсті құруға үш есе көп бояу кетеді;

– кез-келген түсті бояулар қара бояудан қымбат.

Айтылған факторлар күйінде таза қара түсті басқанда қара түс компонентасы қосымша қолданылады. Бұл технология көлеңкелер сапасын және сұр реңді жақсартады.

Әр төрт түстің компенанта интенсивтігі 0 ден 100% диапозонға дейін өзгере алады.

СМУК модельінің қысқартылған “К” әрпі.

RGB және СМУ – моделіндегі қалыптастру механизімінің айырмашылығы.

4 бояуды қосып қағазда қағазда милион түс синтездеуге болатындығына кейбіреулер таң қалатын шығар. Ал кейбіреулер таң қалмайды. Түстердің субрактивті синтезін іске асыруды талдамас бұрын алдымен түсті таңба құрылымы мен танысуымыз керек.

Перцепционды түстік модельдер

Дизайнерлер, суретшілер, фотографтар үшін негізгі құрал болып көп қызмет етеді. Бұл «құрал» – техникалық құрылғыдан ол сканер, принтер болсын, асып түседі.

Жоғарыда айтылғандай RGB және СМУК түстік жүйенің техникалық құрылғыларын сипаттау үшін аппаратты тәуелділік қолданылады. Бұл дегеніміз өндірілетін немес құрылатын түс шығару құрылғысы сипаттамасына да тәуелді болады;

Аппаратты тәуелділікті шеттету үшін перцепционды түстік модел жасалған келесідей артықшылықтары бар:

– түстермен түсінікті деңгейде қатынасуға мүмкіндік береді;

– түстерді келістіру проблемасын жеңілдетеді.

Жарықтық және түстілік концепциясын қолданатын түстік мод-р прототилі-HSV-модель. Басқа да жүйелер бар: HSV, HSB, HSL,YUV. Олар үшін жалпы болып түстер негізгі үш түсі арқылы көк, қызыл, жасылмен берілгенінде емес, екі компакетті көрсету жолымен анықталады: түстік және жарықтық.

HSB түстік моделі

HSB моделі (НИЕ-түстік тон, saturation-қанықтылық Brightness-жарықтың)немесе оның жақын аналогын HSL қазіргі графикалық дестелерде ұсынылған. Көптеген қолданған модельге қарағанда осы модель адам көзімен тез қабылдайды.

HSB модельде барлық түстер үш базалық параметр комбинациясы көмегімен анықталады:

– түстік рең (H);

– қанықтылық (S);

– жарықтық (B).

Түстік рең түстік ортаның түстік жағдайын анықтап жатқан уақытта қанықтылық және жарықтық «таза» түс мөлшерін және берілген реңде жарықты береді.

HSB моделінің табиғатымен жақсы танысу үшін оның негізгі компоненттері мен танысалық.

9.2 Түстік рең. Түстің қоюлығы. Түстің ашықтығы

Әр жарық қоры ұзындықтағы толқындардан тұрады. Түстік рең (пие) ұзын толқын ретінде түсіндіріледі. Әдетте түстік реңді сипаттау үшін түс аты қолданылады, мыс, қызыл, жасыл, сарғыш. Осы моделінің дәстүрлі интерпретациясында әр түстік рең түстің ортада бір анқталған орында алады және 0 ден 360^0 аралықтан деопозонда бұрыш шамасымен сипатталады. Әдетте қызыл үшіе 0^0 бұрыш алынса, жасыл үшін-120^0, көк үшін-240^0 алынады. Түстік ортада бастапқы түстер бір-бірімен бірдей арақашықтықта орналасқан. Екінші түстер бастапқы түстер арасында орналасады. Өз кезегінде әр түс өзінің қосымша түсіне қарсы орналасқан. Мысалы, сары мен көгілдір жасыл түсті береді. Осылайша түстік ортада жасыл түс сары түспен көгілдір түстің арасында орналасады, бірақ оранжевый мен фиолетовый түстер бастапқы, екінші түстерге жатпайды, бірақ олар түстер диограмма ортасында көрсетілген.

Түстік рең түсінігі толық түс жайлы ақпараттан тұрмайды. Мысалы 450 нанометр толқын ұзындықты жарық дамдармен рең ретінде қабылданады.

Қанықтылық

Түстік рең адамдар мен ажыратылатын жалғыз атрибут емес. Басқа компонент-қанықтылық-түс тазалығын сипаттайды. Ол негізгі түс компонентасы және түсті қалыптастыруда қатысатын қалған толқын ұзындықтары арасындағы қатынасты анықтайды. Бұл параметрдің мөлшерлі мәні процентпен белгіленеді, 0%-тен (сұр) 100%-ке (толық қанықтылық) дейін.

Жарықтық (ашықтық)

Жарықтық біздің көзіміздің рецепторларына әсер етеді. Жарық энергиясы интенсивтілікті сипаттайды, оны жарықтыққа қатынастығын интерпретациялауымызға болады. Күн қояны-жоғары жарықтану интенсивтігінің мысалы. Түстер мен түрлердің олардың түстік реңіне тәуелсіз жарықтығымен салыстыруымызға болады, қайсысы ашық, қайсысы қоюырақ екенін анықтаймыз.

Жарықтық компонентасының әмбебаптығы

Жарықтық – хромотикалық және ахроматикалық түстерге тән қасиет. Сондықтан жарықтық арқылы кез келген түстер мен реңдерді салыстыра аламыз: ашық жасылды қою жасылмен, ашық қызылды көкпен, қызылды көгілдірмен және т.б.

Бұл қасиет түсті кескіндердің ақ қараға конвертирлегенде қолданылады.

Суретшілер жарықты қатнасты тональді д/а. сондықтан жарықты және түсті тонды ажыратады. Картинаны жарық тонды жазылған десе алдымен жарықты қатынасын айтады. Ал түсі бойынша ол әртүрлі болуы мүмкін. HSB және HSL - модельдері арасындағы айырмашылық сызықты емес. Brightntss компонентасының сызықты lightness компонентіне ауыстырылуы.

Негізгі әдебиет: 5 [бет 121-199]

Қосымша әдебиет: 13 [бет 119-139]

Бақылау сұрақтары:

1. Түс режимінен түс модельдерінің айырмашылығы?
2. Сіз компьютерлік графика, объектілер және қабықтарды қалай түсінесіз?

3. 24 битті түс 16,7 млн. Түс шығаратыны есептеңіз.

4. Субтрактивті және аддитивті түстік моделдер дегеніміз не?

5. RGB-моделінің барлық негізгі түстерінің қосымшасын қандай түс береді?

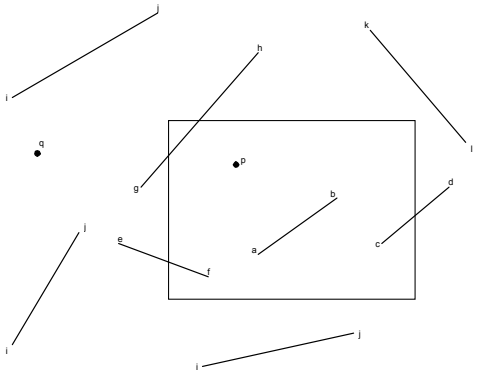
Дәріс 10. Жазықтықтағы фигуралардың кесілуі

10.1 суретте жазық сцена және реттелген форманың қиылысатын терезесі көрсетілген. Терезе сол (с), оң (о), жоғарғы (ж), төменгі (т) екі өлшемді қабырғамен беріледі. Реттелген қиылысқан терезелер тіктөртбұрыштар болады. Олар объектіні кеңістіктің координат объектілеріне параллель немесе экран координат осьтеріне параллель. Немесе экран координат осіне қиылысу алгоритмінің мақсаты қиылысатын терезенің ішінде жатқан нүктелер, қиындылар визуализация үшін қалдырылады.

Қарапайым сценалар немесе картинкада нүктенің көп санын қию керек болса, онда көпжағдайда қиылысу алгоритмін тиімді нүктелер толық ішінде жатады. Сондықтан қиындыларды тез алу керек.

Қиылысатын терезенің ішінде жататын нүктелер келесі шартты қанағаттандырады.

Теңдік белгісі терезе шекарасында жатқан барлық нүктелер оның ішінде $x_л <= x <= x_п$ и $y_н <= y <= y_в$. деп саналады.



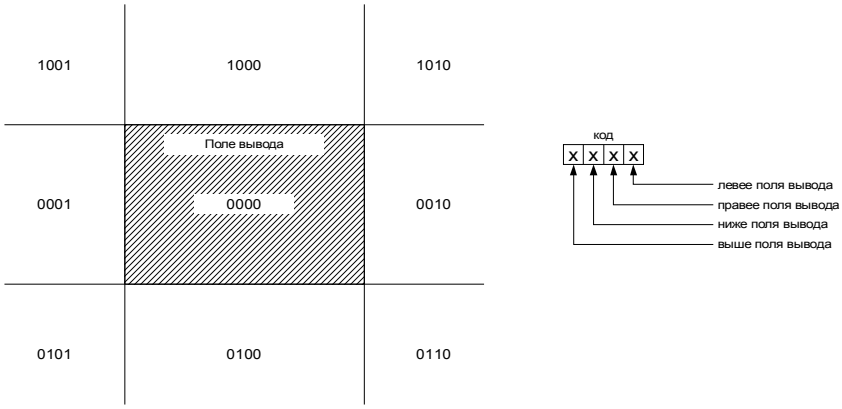
Сурет 10.1

Теңдік белгісі көрсетеді. Мұнда терезе шегінде нүктелер оның ішінде орналасқан деп саналады. Қиынды терезе ішінде жатыр.

Сондықтан ол көрінеді. Егер оның соңғы нүктелері терезе ішінде жатса, мысалы 10.1 суретте ab қиындысы. Бірақ қиындының екі соңында терезеге жатпаса, онда бұл қиынды терезеден тысжатуы міндетті емес, мысалы 10.1 суретте gh қиындысы.

10.1 Коэн-Сазерленд кесу алгоритмі

Шығару өрісіне қиындыны жатқызуды анықтау есебін шешу үшін келесі әдіс қолданылады. Кеңістік 9 ауданға бөлінеді, әр аудан бинарлы 4 битті (10.2 сурет).



Сурет 10.2 – Кеңістікті кодтау

Әр қиынды үшін код соңы есептеледі. Содан кейін талдау экспрет жүргізіледі.

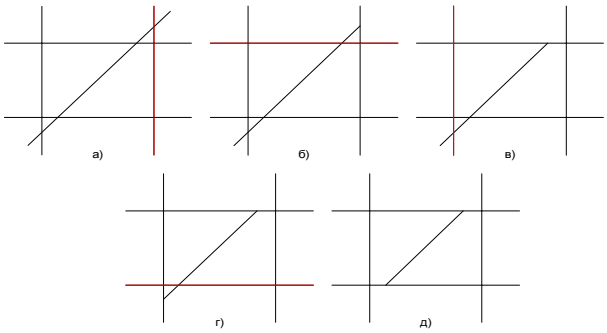
Егер $К_1$ Λ $К_2$ #0, онда қиынды шығару өрісінде жатқан жоқ – қиынды тасталады.

Егер $К_1$=$К_2$=0, онда қиынды толығымен шығару өрісінде жатыр – қиылысу керек емес, қиынды толық суреттеледі.

Егер $К_1$ Λ $К_2$ =0 қиынды бөліктеп шығару өрісінде жатуы мүмкін – шығару өрісі не қиылысу қажет.

Тік бұрышты аудан бойынша қиылысу алгоритмі

Егер $К_1$ Λ $К_2$ =0 қиындыны шығару өрісі шекарасымен қию керек, қиылысу суреттің барлық жағынан жүреді.



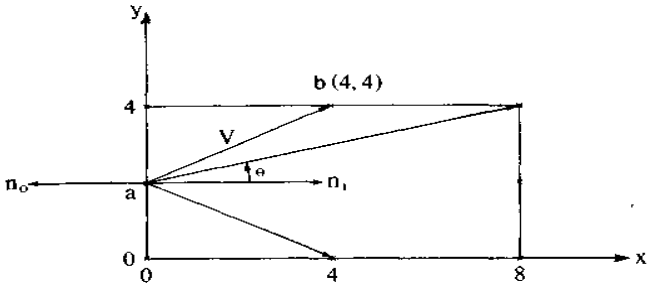
Сурет 10.3 – Тік бұрыш ауданы бойынша қиындыны қию

10.3 суретте қиылысу жүретін қабырға бөлінген. Шығару өрісі шекарасында жатқан нүктелер шығару өрісіне жатады.

10.2 Тікбұрышты аудан бойынша кесу алгоритмі. Кируса-Бек алгоритмі

Қиюдың сенімді алгоритмін құру үшін кесіндіге жататын нүктенің терезеге (ішінде, шекарасында немесе одан тыс) қатынасты орналасуын анықтайтын жақсы тәсіл болуы керек. Бұл мақсат үшін Кируса-Бек алгоритмінде вектор нормалі пайдаланылады.

R қиылатын *выпукл* аймағын алайық. *R*-дің екі өлшемді болуы міндетті емес, берілген бөлімде келтірілетін мысалдарда,ол екі өлшемді. Яғни, *R* кез келген выпукл жазық көпбұрыш болуы мүмкін. Ол вогнут көпбұрыш болмауы керек. Ерікті а нүктесіндегі *R*-дің шекарасында жатқан *n* ішкі нормалі *n*-(Ь- а)≥ 0 шартын қанағаттандырады, мұндағы $b - R$ шекарасындағы кез келген басқа нүкте. Бұғн *убедиться*, еске алайық, V_1 және V_2 екі сколярлық вектордың көбейтіндісі V_1V_2*[V_1][V_2] cos-кетең, мұндағы $Q - V_1$ және V_2-ден жасалған екі бұрыштың кішісі. Байқайық, егер $\theta = \pi / 2,$болса, онда cos $\theta = 0$ және $V_1*V_2 = 0$, яғни жұп векторлардың сколярлық көбейтінділері нөлге тең болғанда олар перпендикуляр. 10.4 суретте *R* выпукл аймағы көрсетілген, яғни қиятын терезе. Сол жерде ішкі (сыртқы) $n_н$ және а нүктесінен шығатын сол шекарада жатқан терезенің шекарасына нормальдың ішкі $n_в$ көрсетілген.



Сурет 10.4

Одан басқа 10.4 суретте а нүктесінен, терезе шекарасының басқа нүктелеріне жүргізілген бірнеше векторлар көрсетілген. $n_в$-мен кез келген осындай векторлар арасындағы бұрыш $-\pi /2 \le \theta \le \pi / 2$ интервалына жатады. Бұрыштың мұндай мәндерінде оның косинусы әр уақытта оң. Сондықтан, жоғарыда тағайындалғандай бұл векторлардың сколярлық көбейтіндісі оң. Ал сыртқы нормальдар арасындағы бұрышпен және сол сияқты кез келген векторлар тең $\pi - \theta$, *a cos* $(\pi - \theta) = -$ cos θ бұл жағдайда теріс.

Анықтамаға қайта келсек, терезенің жағымен кесіндінің қиылысуы қайтадан P_1 P_2 кесінділерінің параметрлік көрсетілуін аламыз:

$$P(t) = P_1 + (P_2 - P_1)t,\ 0 \le t \le 1.$$

Егер $f - R$ выпуклый аймағының шекаралық нүктесі, ол n – бұл аймақты шектейтін жазықтықтың біреуіне ішкі нормаль, онда кез келген нақты шама *t*-ға, яғни P_1 P_2,кесінділерінің кез-келген нүктелеріне $n_*[P(t) - f] < 0$ шартынан $P(t) - f$ векторы *R* аймағынан тыс бағытталғаны туындайды. $n*[P(t)-f] = 0$ шартынан P(t) $- f$ векторы Суретте көрсетілгендей *R*-дің ішіне бағытталған. Барлық осы шарттарды бірге алған кезде, шексіз түзу екі өлшемді жағдайда теңдей екі нүктедегі тұйық выпукл көпбұрышқа әкелетін выпукл аймақпен қиылысады. Ары қарай, бұл екі нүкте бір шектелген жазықтыққа немесе қабырғаға жатпасын. Онда $n*[P(t)-f]>0$ теңдеуі тек бір ғана шешімге ие. Егер f нүктесі n – ішкі нормаль болатын сол шектелген жазықтықта немесе сол қабырғада жататын болса, онда $P(t)$ кесіндісіндегі нүкте, соңғы теңдеуді қанағаттандыратын, көрсетілген шектелген жазықтықпен осы кесіндінің

қиылысу нүктесі болады.

Негізгі әдебиет: 2 [стр.170-180]
Қосымша әдебиет: 11 [стр.165 -199]
Бақылау сұрақтары:
1. Кируса-Бекка тәсілінің мағынасы неде?
2. Нормаль дегеніміз не?
3. Ішкі және сыртқы нормальдар қандай роль ақарады?
4. Көрінбейтін кесінділер дегеніміз не?

11 Дәріс. Көрінбейтін сызықтарды жою

Көрінбейтін сызықтарды және беттерді жою компьютерлік графикада күрделі тапсырмалардың бірі. Көрінбейтін сызықтарды және беттерді жою алгоритмі кеңістіктің берілген нүктесіндегі бақылаушыларға көрінетін немесе көрінбейтін қабырға сызықтарын, беттерді немесе көлемдерді анықтауға қызмет етеді.

Көрінбейтін сызықтарды, қабырғаларды, беттерді немесе көлемдерді жоюдың қажеттілігі иллюстриленген. 11.1, а суретте куб сызбасының қаңқасы көрсетілген. Қаңқалы сызба, оның қабырғаларының штрихтік бейнесін үш өлшемді объект түрінде көрсетеді. 11.1, а суретті екі ұшты кубтың түрін жоғарыдан, солдан немесе төменнен, оңнан интерпретирлеуге болады. Ол үшін көзді қысып, фокусын өзгерту керек. Сәйкес көз қарастан көрінбейтін сызықтарды және беттерді жою бір мәнділік еместен және арылуға мүмкіншілік береді. Нәтижелері 11.1, b және ссуреттерде көрсетілген.

Көрінбейтін сызықтарды және беттерді жоюдың күрделілігі оны шешуде көптеген әртүрлі әдістердің пайда болуына әкелді. Оның көбі арнайы қолданбаға (қосышаға) бағытталған. Жалпы көрінбейтін сызықтар мен беттерді жоюдың өте жақсы шешімі жоқ. Нақты уақытта процестерді модификациялау үшін, мысалы, авиатренажерлар үшін нәтижелерді бейнегенерацияның (30 кадр/секунд) жиілігімен тудыратын жылдам алгоритмдер талап етіледі. Машиналық мультипликациялар үшін, мысалы, нақты күрделі бейнелерді генерациялайтын түстердің шағылысу және сыну эффектері кішкентай оттеноктарына дейін ескерілген көлеңкелер, фактуралар және айқындық көрсетілген.

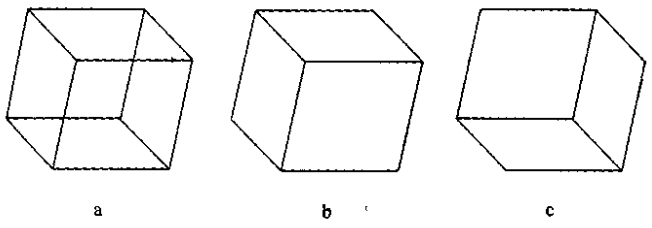


Рис. 4.1. Необходимость удаления невидимых линий.

Сурет 11.1 – Көрінбейтін сызықтарды жою мүмкіндігі

Ұқсас алгоритмдер жәй жұмыс жұмыс істейді және көбінесе есептеулерге бірнеше минут немесе сағаттарда талап етіледі. Қатаң айтқанда, эффектерді есепке алу айқындық, фактуралары, шағылысу және т.б. көрінбейтін сызықтарды, беттерді жою есебіне кірмейді. Әрине, оларды бейнелерді визуализациялау процессінің бөлігі деп санаған дұрыс. Визуализация процессі интерпретация немесе бейнені көрсету немесе реалистік манердегі сценалар. Бірақ, бұл эффектердің көбі көрінбейтін беттерді жою алгоритмінде енгізілген және сондықтан бұл тарауда айтылады.

Алгоритмнің жұмыс істеу жылдамдығы мен оның нәтижесінің дәлдігі арасында тығыз қарым-қатынас бар. Бұл екі көрсеткіш үшін бір уақытта алгоритмдердің бірде біреуі жақсы бағаға жете алмайды. Одан ад шапшаң алгоритмдерді құрғанда анығырақ бейнелерді тұрғызуға болады. Бірақ, нақты есептер барлық уақытта детальдардың одан да көп санын есепке алуды талап етеді.

Көрінбейтін сызықтарды (беттерді) жоюдың барлық алгоритмдерін сұрыптау бар. Объект координаталарын сұрыптау реті, былайша айтқанда, бұл алгоритмдердің эффективтілігінеәсер етпейді. Басты сұрыптау денеден, беттен, қабырғадан немесе бақылау нүктесіне дейінгі нүктеден геометриялық қашықтықта жүргізіледі, қашықтықтан сұрыптаудың негізіне салынған негізгі идея бақылау нүктесінен неғұрлым қашықтықта орналасқан объект бақылау нүктесіне жақынырақ орналасқан объектілердің біреуімен түгел немесе бөліктері көлеңкеленуң ықтималдығы өте үлкен.

Тереңдігі бойынша ара қашықтықты немесе проритетті анықтағаннан кейін горизонталь және вертикаль бойынша сұрыптауды өткізу. Қарастырылатын объект шынында бақылау нүктесіне жақын орналасқан объектпен көлеңкеленетіндігін анықтау қалады. Көрінбейтін сызықтарды және беттерді жоюдың кез келген алгоритмінің эффективтілігі көбінесе сұрыптау процессінің эффективтігіне тәуелді. Сұрыптаудың эффективтігін өсіру үшін, сценаның когерентностығы пайдаланылады, яғни сцена сипаттамасының азға өзгермейтін тенденциясында растрлық графикада когерентносты сұрыптау нәтижесін жақсарту үшін көрінбейтін беттерді жою алгоритмінде пайдалану, растрлық жаймалау алгоритмін еске салатын алгоритмге әкеледі. Көрінбейтін сызықты немесе беттерді жою алгоритмін координат жүйелерін таңдау әдісі немесе олар жұмыс істейтін кеңістікке жіктеуге болады. Объектік кеңістікте жұмыс істейтін алгоритмдердің осы объекттер сипатталған физикалық координат жүйелерімен ісі бар. ол кезде нақты нәтижелер тек есептеу дәлдігімен шектелгені алынады. Алынған бейнелерді көп рет еркін үлкейтуге болады. Объектілік кеңістікте жұмыс істейтін алгоритмдер, әсіресе, жоғарғы дәлдік қажет (қосымшаларда) қолданбаларда пайдалы. Бейнелеу кеңістіктігінде жұмыс істейтін алгоритмдер, объектілер көрсетілген экранның координаттар жүйесімен жұмыс істейді. Сондықтан, есептеу дәлдігі экранның шешу анықтау қабілетінен шектелген. Әдетте, экран анықталуы өте төмен болады, типтік мысал – 512 х 512 нүкте. Бейнелеу кеңістігінде алынған нәтиже, одан кейін бірнеше рет үлкейтілген, берілген сценаға сәйкес келмейді. Мысалы, кесінділердің ұштары кездеспеуі мүмкін. Приоритеттер тізімін құрастыратын

алгоритмдер, кезек-кезек ескертілген екі координаты жүйесінде жұмыс істейді.

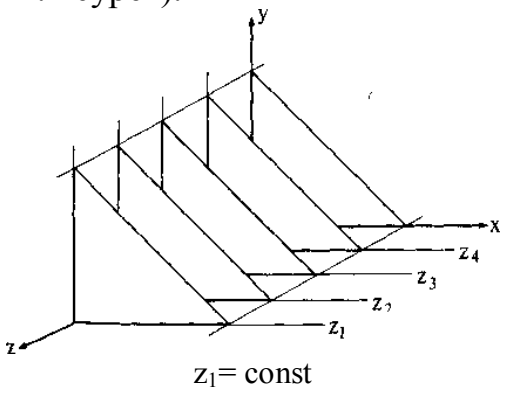
11.1 Қалқымалы көкжиек алгоритмі

Жүзетін горизонттың алгоритмі көбінесе функцияның үш өлшемді бетті мынадай түрде сипаттайтынсызықтарды жою үшін

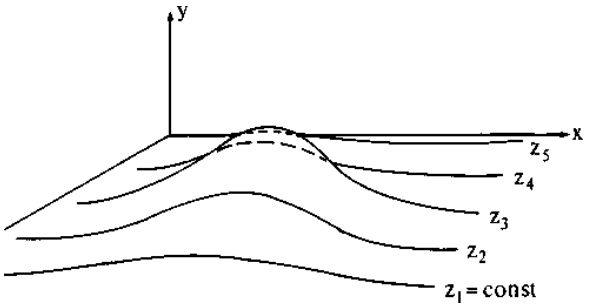
$$F(x, y, z) = 0$$

Ұқсас функциялар көптеген қолданбаларда математикада, техникада, табиғи ғылымда және басқа пәндерде пайдаланылады.

Бұл подходты пайдаланатын көптеген алгоритмдер ұсынылған. Қолданбалыларда негізінен беттің сипаттамасы қызықтырады, әдетте бұл алгоритм бейнелеу кеңістігінде жұмыс істейді. Берілген тәсілдің басшы идеясы үш өлшемді есепті тұрақты *x*, *y* немесе *z* координата мәндері бар жуылықты параллельді қиятын тізбектің берілген бетпен қиылысу жолымен екі өлшемді есепке айналдыруында (11.2 сурет).



Сурет 11.2 – Тұрақты координаталы қиылысатын жазықтықтар



Сурет 11.3 – Тұрақты координаталы қиылысатын жазықтықтардағы қисықтар

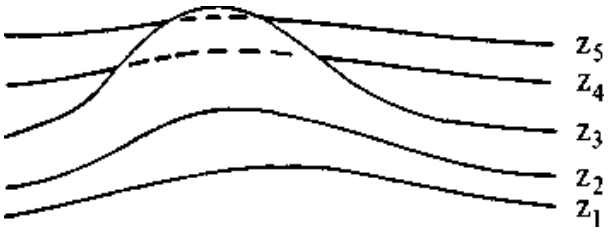
11.2 суретте мысал келтірілген параллельдік жазықтықтар онда тұрақты *z* мәнімен анықталады. $F(x, y, z) = 0$ функциясы, бұл параллельдік жазықтықтың әрқайсысында жататын қисықтар тізбегіне әкелінеді, мысалы,

$$y = f(x, z) \text{èëè} x = g(y, z)$$

тізбегіне, мұндағы *z* әрбір берілген параллель жазықтыққа тұрақты.

Сонымен, енді бет 11.3 суретте көрсетілгендей осы жазықтықтың әрқайсысында жататын қисықтар тізбегінен құрастырылады. Бұл жерде алынған қисықтар бір мәнді функциялардың тәуелсіз айнымалылары деп есептелінеді. Егер алынған қисықтарды 11.4 суретте көрсетілгендей $z=0$

жазықтығына жобаласа, онда берілген беттің көрінбейтін бөліктерін жою алгоритмінің идеясы бірден анық болады. Алгоритм алдымен *z*=const жазықтығын бақылау нүктесінен оларға дейінгі қашықтықты ұлғаюы бойынша реттейді. Содан кейін бақылау нүктесіне жақын әрбір жазықтық үшін онда жататын қисық салынады, яғни бейнелеу кеңістігінде *x* координатасының әрбір мәні үшін сәйкес *y* мәні анықталады. Көрінбейтін сызықты жоятын алгоритм:

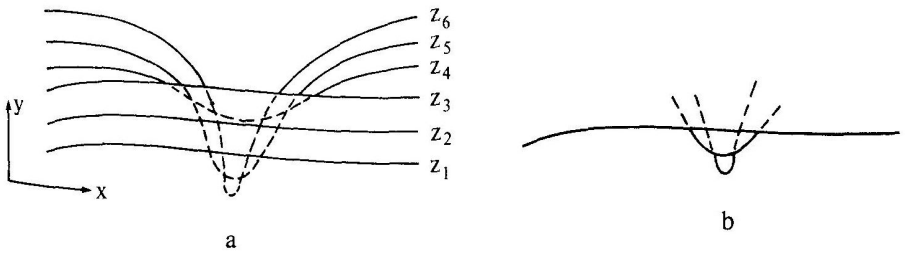


Сурет 11.4 – z=0 жазықтығындағы қисықтар проекциясы

Егер ағымдағы жазықтықта *y*-ке сәйкес келетін *x*-тің мәнінде, *x*-тің осы берілген мәнінде алдыңғы барлық берілген қисықтарда, қисықта *y*-тің мәні көп, онда ағымдағы қисық бұл нүктеде көрінеді; кері жағдайда ол көрінбейді.

11.4 суретте көрінбейтін учаскелер пунктирмен көрсетілген. Берілген алгоритмді іске асыру өте қарапайым. *x*-тің әрбір мәнінде *y*-тің максимальды мәнін сақтау үшін, ұзындығы бейнелеу кеңістігінде *x* осі бойынша анықталатын нүктелер санына тең массив пайдаланылады. Бұл массивте сақталынатын мән «горизонтта» ағымдағы мәнді көрсетеді. Сондықтан, әрбір кезекті қисқты салғанда, бұл горизонт «всплывает» бетке шығады.Бұл көрінбейтін сызықтарды жою алгоритмі әр кез бір сызықпен жұмыс істейді.

Алгоритм кейбір кезектегі қисық, 11.5, а суретте көрсетілгендей ең бірінші қисықтан болғанша, өте жақсы жұмыс істейді. Ұқсас қисықтар,, әрине көрінеді және өзі берілген беттің төменгі жағын көрсетеді, бірақ алгоритм көрінбейді деп есептейді. Беттің төменгі жағы егер ол алгоритмге, алгоритмнің жұмысы кезінде төмен түсетін, төменгі горизонтты енгізіп модификацияласа көрінетін болады. Бұл екінші бейне кеңістігінде x осі бойынша ұзындығы айқындалатын нүктелердің санына тең массивтің көмегімен іске асады. Бұл массивте *x*-тің әр мәні үшін y-тің ең кіші мәні бар. енді алгоритм мынандай болады:

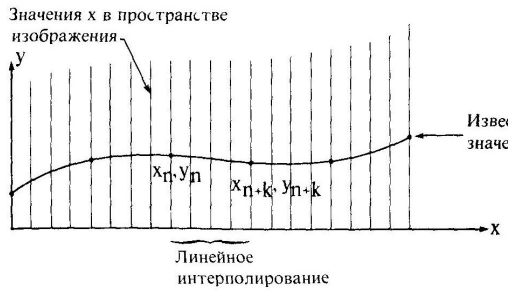


Сурет 11.5 – Жазықтықтың төменгі жағын өңдеу

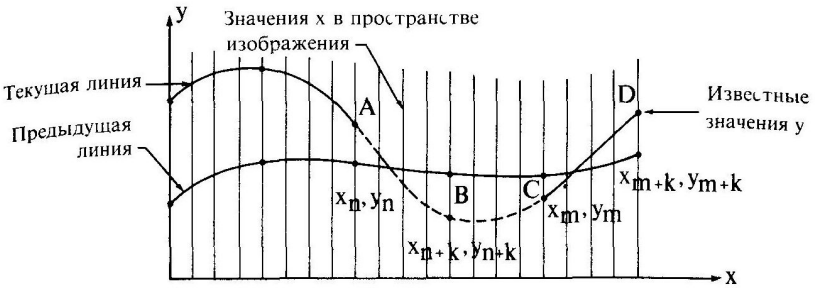
Егер ағымдағы жазықтықта *x*-тің кейбір берілген мәніне *y* бойынша барлық алдыңғы қисықтар үшін ол кезде *x* максимумнан үлкен немесе минимумнан

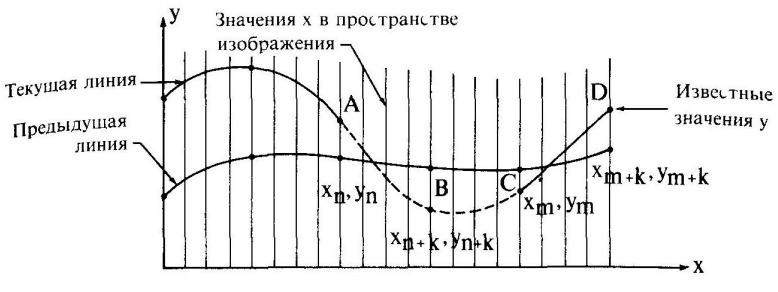
кіші қисықтың y мәні сәйкес, онда ағымдағы қисық көрінеді. Керісінше, ол көрінбейді. Алынған нәтиже 11.5, в суретте көрсетілген.

Жоғарыда көрсетілген алгоритмде функцияның мәні, яғни бейнелеу кеңістігіндегі x-тің әрбір мәні үшін y белгілі. Бірақ, егер әрбір мән үшін оған сәйкес y-тің мәнін көрсетуге (есептеуге) болмаса, онда жүзуші горизонттардың төменгі және жоғарғы массивтерін қолдау мүмкін емес. Мұндай жағдайда белгілі мәндердің арасында 11.6 суретте көрсетілгендей жүзуші горизонттың төменгі және жоғарғы массивтерін толтыру үшін y мәнінің сызықтық интерполяциясы пайдаланылады. Егер қисықтың көрінуі өзгеретін болса, онда мұндай қарапайым интерполяциямен тәсіл қатесіз нәтиже бермейді. Бұл тиімділік 11.7, а-суретте көрсетілген. Массивтерді толтыру операциясы көрінуді тексергеннен кейін өткізіледі деп отырып ағымдағы қисықтың көрінетінінен көрінбейтін жағдайда (11.4 суретте ИВ сегменті) өткенде кезде (x_{n+k}, y_{n+k}) нүктесі көрінбейді деп хабарланатынын аламыз. Онда (x_n, y_n) және (x_{n+k}, y_{n+k}) нүктелерінің арасындағы қисықтың бөлігі бейнеленбейді және массивтерді толтыру операциясы жүргізілмейді.



Сурет 11.3 – Берілген нүктелер арасындағы сызықты интерполяция





Сурет 11.4 – Қиылысатын қисықтар эффектісі

Ағымдағы және алдыңғы қисықтар арасында бос орын пайда болады. Егер ағымдағы қисықтың бөлігінде көрінбейтін жағдайдан көрінетін жағдайға (11.4, а суреттегі *CD* сегменті) өткенде (x_{m+k}, y_{m+k}) нүктесі көрінеді деп хабарланады және массивтерді толтыру операциясы жүргізіледі. Сондықтан *CD* сегментінің көрінбейтін бөлігі бейнеленеді. Одан басқа, жүзетін горизонттар массивтері *y*-тің нақты мәндерін ұстамайды. Бұл кейінгі қисықтар үшін қосымша керек емес эффектер болуы мүмкін.

Сонымен, ағымдағы және алдыңғы қисықтардың қиылысу нүктелерін іздеу есептерін шығару қажет. Қисықтардың қиылысу нүктесін алудың бірнеше тәсілдері бар. растрлық дисплейлерде *x* координатасының мәнін cx_n немесе x_m (11.4,а сурет) бастап 1-ге үлкейтуге болады. Бейнелеу кеңістігінде *x* координатасының ағымдағы мәніне сәйкес, *y* мәні алдыңғы *x* координатасының мәніне сәйкес *y*-тің мәніне берілген қисықты бойлап Δy вертикальды өсуді қосу арқылы пайда болады. Содан кейін, (y+1, $y+\Delta y$) координаталарымен жаңа нүктенің көрінетіндігі анықталады. Егер нүкте көрінетін болса, онда онымен байланыстағы пиксель активтеленеді. Егер көрінбесе, пиксель активтенбейді, ал *x* 1-ге көбейеді. Бұл процесс x_{n+k} немесе x_{m+k} кездескенше созылады. Растрлық дисплейлер үшін, қиылысу жеткілікті дәлдікпен айтылған тәсілмен анықталады. Қиылысуды анықтаудың жақын және өте элеганттық тәсілі екілік іздеуге негізделген.

Екі түзу сызықты кесіндінің (x_n, y_n) және (x_{n+k}, y_{n+k}) нүктелерінің арасындағы алдыңғы және ағымдағы қисықты интерполяциялайтын қиылысу нүктелерінің дәл мәні мынандай формулалармен беріледі:

$$x = x_n - \frac{\Delta x(y_{np} - y_{nc})}{(\Delta y_p - \Delta y_c)}$$

$$y = m(x - x_n) + y_n$$

мұндағы:

$$\Delta x = x_{n+k} - x_n$$

$$\Delta y_p = (y_{n+k})_p - (y_n)_p$$

$$\Delta y_c = (y_{n+k})_c - (y_n)_c$$

$$m = [(y_{n+k}) - (y_n)] / \Delta x$$

11.2 Робертс алгоритмі

Робертс алгоритмі көрінбейтін сызықтарды өшіру (жою) есебінің алғашқы атақты шешімі болып табылады. Бұл объектілі кеңістікте жұмыс істейтін математикалық әдіс. Ең алдымен алгоритм әрбір денеден қажетті қабырғаны немесе шекті жояды, ал дене болса өзі экрандалады. Содан соң әрбір дененің көрінетін қабырғалары әрбір қалған денелермен салыстырылып, қай бөлшек немесе бөлшектер оған сәйкес екенін, егер ондайлар болса онда осы денелермен (экрандалатынын) анықтау үшін қолданылады. Сол үшін Робертс алгоритмінің есептеу қиындығы теория жүзінде объектінің квадрат саны болып өседі. Бұл кеңістіктегі суреттермен жұмыс істейтін және растрлық дисплейлерге қызығушылықтың төмендеуіне әкеліп соқты. Дегенмен бұл

алгоритмде қолданылатын математикалық әдістер ыңғайлы, қуатты және нақты. Сондай-ақ бұл алгоритмді кейбір маңызды концепциялардың көрнекіліктерінде қолдануға болады. Объектілер санынан сызықты тәуелділікті *z* осі бойындағы алдын-ала приоритетті реттеу және қарапайым габариттік немесе минимаксті тесттер, яғни алгоритмнің кейінгі реализациялары қолданылады.

Робертс алгоритмінде барлық суреттелетін денелер немесе объектілер дөңес болуын талап етеді. Дөңес емес денелер дөңес бөлшектерге бөлінуі керек. Бұл алгоритмде жайпақ шектері бар дөңес көпжақты дене қиылысушы жазықтар жиынымен көрсетілуі қажет. Үшөлшемді кеңістіктегі еркін жазықтық теңдеуі мынадай түрде көрсетіледі:

$$ax+by+cz+d=0 \qquad (1)$$

Ал матрицалық түрде бұл нәтиже былай көрсетіледі:

$$[xyz1]\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = 0$$

немесе

$$[xyz1][P]^T = 0$$

Мұндағы $[P]^T$ = [a b c d] жазықтықты көрсетеді. Сондықтан дөңес қатты денені жазықтықтардың теңдеулерінің коэффициенттерінен тұратын матрица дөңесімен көрсетуге болады. Яғни:

$$[V] = \begin{bmatrix} a_1 a_2 \ldots a_n \\ b_1 b_2 \ldots b_n \\ c_1 c_2 \ldots c_n \\ d_1 d_2 \ldots d_n \end{bmatrix}$$

Мұндағы әрбір тізбек бір жазықтықтың коэфициентін құрайды. Кеңістіктегі кез-келген нүкте бірыңғай координаттарда вектормен көрсетіледі.

$$[S]= [x\ y\ z\ 1]$$

Сондай-ақ егер [S] нүктесі жазықтықта жатса, онда $[S] - [P]^T = 0$ ([3-5] қараңыз). Ал егер [S] нүктесі жазықтықта жатпаса, онда осы скаляр туындысы жазықтықтың қай жағында нүктенің орналасқанын көрсетеді. Робертс алгоритмінде дененің ішінде жатқан нүктелер жағымды скалярлы туындыны береді деп болжайды.

Келтірілген мысалда жазықтықтар теңдеулерінің келістірілуі тәжірибе жүзінде тексерілген. Әрине, бұл үнемі мүмкін болмайды. Жалпы жағдай үшін бірнеше пайдалы әдіс бар. (4.1) жазықтың теңдеуі 4 белгісіз коэффициент құраса да, оны d = 1 етіп жөндеуге болады. Артынша, бұл коэффициентті анықтау үшін 3 коллинеар емес нүкте жеткілікті. 3 коллинеар емес нүктелердің координаттарын тұрғызғанда (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3) нормаль теңдеуі (1) мынаны береді:

$$ax_1 + by_1 + cz_1 = -1$$
$$ax_2 + by_2 + cz_2 = -1$$
$$ax_3 + by_3 + cz_3 = -1$$

Ал матрицалық түрде:

$$\begin{bmatrix} x_1 y_1 z_1 \\ x_2 y_2 z_2 \\ x_3 y_3 z_3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

немесе

$$[X][C] = [D] \quad (2)$$

Бұл теңдеудің шешімі жазықтық теңдеудің коэффициенттерінің мәніне тең болады:

$$[C] = [X]^{-1}[D]$$

Егер жазықтықтағы нормаль вектор белгілі болса, басқа әдіс қолданылады. Яғни:

$$N=ai+bj+ck$$

Мұндағы i, j, k — сәйкесінше x, y, z осьтерінің бірлік векторлары. Сонда жазықтық теңдеуі мынадай түрге ие болады:

$$ax+by+cz+d=0 \quad (3)$$

d өлшемі жазықтықтағы нүктенің туындысымен есептеледі. Егер жазықтықтағы нүктенің компоненттері (x_1, y_1, z_1) болса, онда:

$$d = -(ax_1 + by_1 + cz_1) \quad (4)$$

Көрінбейтін сызықтар немесе беттерді жою алгоритмдерінің есептеу көлемі көпбұрыштардың санының өсуіне байланысты, сол үшін олардың бетін суреттеу үшін үшжақты көпбұрыштардың түрлерін пайдаланған жөн. Бұл көпбұрыштар дөңес емес немесе жазық емес болуы мүмкін. Мартин Ньюэл [4-1] ұсынған әдіс жазық көпбұрыштардың жазықтық теңдеулерін және жазық емес көпбұрыштардың “өте жақсы” жазықтық теңдеулерін табуға көмек береді. Бұл әдіс көпбұрыштардың әр төбесінің, оған қатысты қабырғалардың векторлық туындылары арқылы нормалін анықтаумен және нәтижелердің ортақ мәнін табумен эквивалентті. Егер a, Ь, c, d – жазықтық теңдеуінің кэффициенттері болса, онда

$$a = \sum_{i=1}^{n} (y_i - y_j)(z_i + z_j)$$
$$b = \sum_{i=1}^{n} (z_i - z_j)(x_i + x_j) \quad (5)$$
$$c = \sum_{i=1}^{n} (x_i - x_j)(y_i + y_j)$$

мұндағы

$$\text{if i = n then j = 1 else j = i + 1}$$

ал d жазықтықтағы кез-келген нүкте арқылы анықталады.

Көрінбейтін сызықтар немесе беттер алгоритмін құру жұмысы басында керекті түрде алу үшін үшөлшемді түрлендіру қолданылады. Түрлендірілген

бұрынғы дене матрицаларын түрлендірілген төбелер немесе нүктелерді қолданып жаңа дене матрицаларын есептеумен алуға болады.

Егер [В] – бірыңғай координаттар матрицасы болса, онда ол бұрынғы дене төбесі болады, ал [Т] – 4х4 өлшемді түрлендірілген матрицасы, сонда түрлендірілген төбелер мынадай [1-1]:

$$[ВТ] = [В] [Т] \quad (6)$$

мұндағы [ВТ] — төбелердің түрлендірілген матрицасы. (2) теңдеуді пайдалана отырып, денені қоршаған жазықтықтардың бұрынғы теңдеуін алуға болады:

$$[B][V] = [D] \quad (7)$$

мұндағы [V] — дене матрицасы, ал [D] оң жағындағы — нөлдік матрица. Түрлендірілген жазықтықтар теңдеулері келесі түрде беріледі:

$$[ВТ][VT] = [D] \quad (8)$$

мұндағы [VT] — түрлендірілген дене матрицасы. Теңдеудің сол жағын теңестіре отырып (7) және (8) аламыз,

$$[ВТ][VT] = [В][V]$$

(6) теңдеуді қойып, [В]-ға қысқартып және сол жағын $[Т]^{-1}$ көбейтіп, мынаны аламыз:

$$[VT]=[T]^{-1}[V]$$

Сонымен, түрлендірілген дене матрицасын бұрынғы дене матрицасын сол жағын кері матрицаға көбейткеннен аламыз. Келесі мысал осыған көрнекілік болады.

Жазықтықтар шексіз ара қашықтыққа ие болады және нүктенің скалярлы туындысы дене матрицасына теріс, егер нүкте бұл денеден тыс жатса, онда бұл әдіс ұсынуға мүмкіндік береді, яғни дене матрицасы осы денемен экрандалады шектерді анықтау үшін қолданылады.

Дене матрицасындағы жазықтық (тізбек) теріс скалярлы туындыны береді, егер нүкте мысалда көрсетілгендей тысқары жатса. Мысалда мұндай болып түрлендірілген дене матрицасында [VT] сол жақ жазықтық (екінші тізбек) және түрленбеген нүкте [S]. Бұл ойлар көрнекіленген.

Егер көрермен z жарты осінің оң жақ шексіздігінде болса және координат басына қараса, онда оның көзі z жарты осінің теріс жағына бағытталған. Бірыңғай мұндай бағыттағы вектор [1-1] тең:

$$[E] = [0\ 0\ -\ 1\ 0]$$

Бұл сондай-ақ z жарты осінің сол жағындағы шексіздікте жататын нүкте болады. $z = -\infty$ жазықтығында жатқан кез-келген нүкте [Е]. Кез-келген нүкте, яғни $(x, y, -\infty)$. Егер скалярлы туынды [Е] тізбектей және кез-келген дене матрицасының жазықтығына тиісті болып теріс болса, онда [Е] осы жазықтықтың сол жағында жатады. $z = \infty$ жазықтығында жататын кез-келген қарау нүктесінің бұл жазықтықтар көрінбейді, ал нүкте $z = -\infty$ дененің өзімен экрандалады. Мұндай жазықтықтар теріс бетті деп аталады, ал оларға сәйкес қырлары-артқы (задними). Сәйкесінше,

$$[E]\cdot[V]< O$$

бұл әдіс бірлік дөңес көпбұрыштарды көрсететін денелер үшін көрінбейтін беттерді жою үшін қолданылатын ең қарапайым алгоритмі болып табылады. Ол сондай-ақ көрінбейтін сызықтарды жою алгоритмінің бірін қолданбас бұрын

теріс бетті немесе артқы (задний) қырларды жою үшін қолданылады. Бұл әдісті артқы жазықты алып тастау (отбрасыванием задних плоскостей) деп атайды. Дөңес көпбұрыштар үшін қырлар саны жуықтап алғанда жарты есе қысқарады. Бұл әдіс әрбір жеке көпбұрыштың бетінің нормалін есептеу үшін эквивалентті. Нормальдің бетке терістігі оның қараушы жағына бағытталғанын көрсетеді, берілген көпбұрыш көрінбейді.

Бақылау нүктесі белгіленген болып қалуы үшін көріністі түрлендіруді денеге қолдануға болады. Басқа әдісте дене қозғалыссыз болады. Сол жақтағы матрицаны көріністі түрлендірудегі кері матрицаға көбейткенде, сәйкес бақылау нүктесі және бақылау бағыты болады.

Беттік емес теріс бетті жазықтықтарды тапқан соң, беттік емес теріс кесінділерді табу керек. Теріс екі жазықтықтардың қиылысуы нәтижесінде теріс беттік емес кесінді түзіледі. 6 мысалда алтыншы жазықтық терi, дегенмен теріс кесінділер жоқ, өйткені тек бір жазықтық өзін-өзі тартады. Сондай-ақ, 7 мысалда бірінші және алтыншы жазықтықтардың қиылысуы нәтижесінде түзілген қабырға теріс болып саналады.

Теріс бетті кесінділерді жоюдың 1-ші кезеңінен кейін басқа денелермен суретте немесе көрініс бейнеленетін кесінділердің бар жоғын білу керек. Ол үшін әрбір қалған кесіндіні немесе қабырғаны басқа көрініс немесе сурет денелерімен салыстыру керек.

Приоритетті реттеу және қарапайым минимаксті немесе тік бұрышты көлемді қабықты тесттерді қолдану группаларды немесе кесінді және дене кластерлерін жоюға мүмкіндік береді.

Мысалы, бақылаушыға дейінгі қашықтықты көрсету үшін қолданылатын ең жақын төбенің z мәнін кейбір басты тізімдерде барлық денелер көріністерде дұрысталады, онда ең алыстағы қабырғаның нүктесінен гөрі, бақылаушыдан ары орналасқан ең жақын болады, бұл тізімнің ешқандай денесі бұл қабырғаны жаба алмайды.Қалған денелердің бірде-бірі тік бұрышты қабығы түгелдей қабырғаның оң жағында, сол жағында, төбесінде немесе астында орналасса, онда бұл қабырғаны бейнелей алмайды. Бұл әдістерді қолдануда әрбір кесінді немесе қабырғаны денелер санымен салыстыруды қажет етуін барынша қысқартады.

P_1P_2 кесіндісін денемен салыстыру үшін, осы кесіндінің параметрлік көрінісін қолданған ыңғайлы:

$$P(t)=P_1+(P_2-P_1)t \quad 0\le t\le 1$$

$$v = s + dt$$

мұндағы v – кесіндідегі нүктенің векторы, s – бастапқы нүкте, ал d – кесінді бағыты. Кесіндінің көрінбейтіндігін анықтау керек. Егер ол көрінбесе, онда t – ның көрінбейтін мәндерін табу керек. Ол үшін P(t) нүктесінен g қарау нүктесіне дейінгі басқа параметрлік кесінді түзіледі:

$$Q(a,t) = u = v + ga = s + dt + ga \quad 0\le t\le 1,\ a\ge 0$$

Мұндағы a және t тура сондай функцияны орындайды. Берілген t мәні P(t) кесіндісіндегі нүктені көрсетеді, ал a кесіндідегі нүктені, яғни жүргізілген P(t) нүктесінен бақылау нүктесіне дейінгіні көрсетеді. Q(a,t) үшөлшемді кеңістіктегі жазықтықты көрсетеді. (a,t) жұбы осы жазықтықтағы нүктені

анықтайды. а мәні оң, себебі дененің экрандалатын P(t) мен бақылау нүктісінің арасында орналасады. Дененің ішінде жатқан кез-келген нүктенің скалярлы туындысы ден матрицасына оң болатынын еске алсақ егер нүкте дененің ішінде жатса, онда ол көрінбейді.

Негізгі әдебиеттер: 2 [бет.250-290]
Қосымша әдебиеттер: 11 [бет.186-199]
Бақылау сұрақтары:
1. Робертс әдісінің мәні қандай?
2. Робертс әдісінің орындалуында қарапайымдылық пен туралылығы неде?
3. Дене матрицасы деген не?
4. Жазық емес көпбұрыштар түсінігі нені білдіреді?
5. Жазық емес көпбұрыштар түсінігі нені білдіреді?

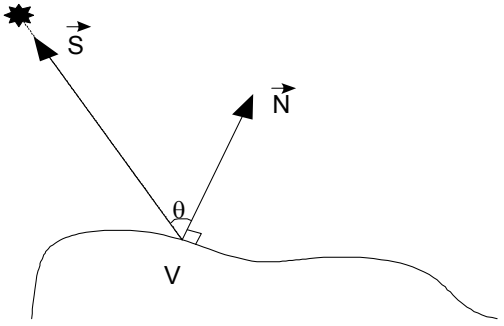
Дәріс 12. Шын кескіндеулерді құру

12.1 Жарық беруді модельдеу. Жарықтың көрінісінің негізгі заңдылықтары. Гура әдісімен бояу

Жарықтың негізгі сыну заңы

Ламберт заңы (диффузиялық сыну)

Егер бір бет болып және сол беттегі бір нүктенің $\vec{N}$ нормаліне жарық бастауынан сәуле бағытталса. Кез-келген нүктеде тұрған бақылаушы үшін оның көріп тұрған нүктенің келесі түрде болады. $V = E \cdot I \cdot \cos\theta + E \cdot I_0$, мұндағы V – жарық (яркость); E – беттің диффузиялық сыну коэффициенті (Альбедо). $E \in [0;1]$, I – нүктенің жарықтандыруы, I_0 – фондық жарықтандыру, **θ** - жарық бастауына бағытталған вектор және нормаль ($\vec{N}$) арасындағы бұрыш, ($\vec{S}$), $\cos\theta = \dfrac{\vec{S} \cdot \vec{N}}{\left|\vec{S}\right| \cdot \left|\vec{N}\right|}$.



Сурет 12.1

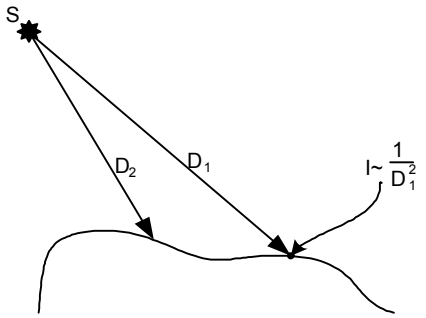
Берілген әдіс жарықтың сынуын ескермейді, сол үшін қараушының тұрған орны мән атқармайды. Осы әдістің көмегімен ең жақсы буланған беттер модельденеді.

Бұрын қаралған Ламберт заңын ыңғайлы түрде көрсетсек.

$V = V_{MAX} \cdot E \cdot \left((1 - \varepsilon) \cdot \cos\theta + \varepsilon \right)$, I=const болғанда мұндағы ε – шашыранды жарықтың бөлігі ($\varepsilon \in [0.1; 0.2]$).

Жарық бастауларының 2 түрі қарастырылады:

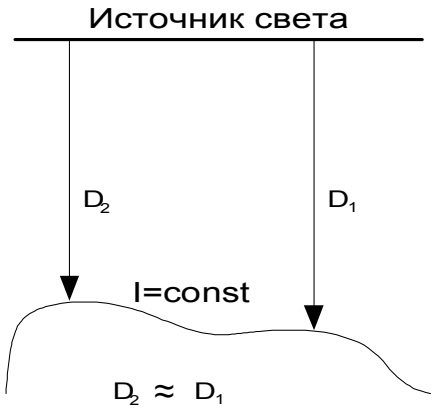
а) нақты жарық бастауы:



Сурет 12.2

Егер нақты жарық бастауы болса, онда жарықтандыру ара қашықтықтың квадратына кері пропорционал болады.

б) параллель жарық: шексіз жойылған бастаудан (мыс: прожектор, яғни сәулелері параллель)



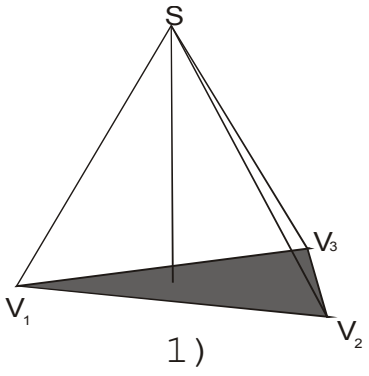
Сурет 12.3

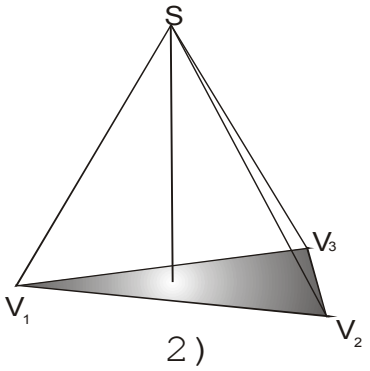
I=const идеалды түрде, яғни жарық интенсивтілігі арақашықтыққа байланысты емес.

Шындығында, жарық ауадағы шаң бөлшектерінде жоғалады, бірақ ара қашықтықтың квадратының турапропорционалдығына емес.

Гуро әдісімен бояу

Жарықтандырудың үздіксіздігін қамтамасыз етеді. Негізгі идеясы: ішкі аймақтарын бояу тек сызықты интерполяциясы арқылы іске асады және тек көпбұрыш төбесінде ғана есептеледі.





Сурет 12.4 Сурет 12.5

Жазық шек берілсін V_1 V_2 V_3 (12.4 сурет). Әрбір төбедегі жарықтандыру мәнін табамыз. Алынған мәндерді I_1, I_2, I_3. арқылы белгілейміз. V_1 V_2 V_3 шектерін жолақтап сала отырып, қабырғалар бойы сызықты интерполяция мәндері арқылы әрбір горизонталь кесіндінің аяғында жарықтандыру мәндерін анықтаймыз. Келесі АВ кесіндісін салу барысында ескереміз, интенсивтілік I(A) - дан I(B)-ға дейін сызықты өзгереді.

Әдістің жетіспеушілігі, ол егер жарық бастауы көпбұрыш жазықтығына көлеңкесі түссе, онда тек ішкі аймақтарды бояу әдісінен кейін қолданылады.

12.2 Фонго әдісімен бояу

N=const

const, егер прожектор
S=Var, егер нақты жарық бастауы
N, S – Ламберт үшін

N, S, P – Фонг үшін

Әдістің негізінде векторлардың интерполяциясы жатыр. Негізгі идеясы: әрбір суреттің нүктесі үшін кеңістік координаттары орнатылады, осыдан γ санап нүкте үшін анықтаймыз.

Әдістің жетіспеушілігі – есептелуінің үлкен қиындығы.

Жарықтандыруды модельдеу әдісімен текстура салу

Тек Ламберт және нақты жарық бастауы үшін қолданылады.

N=const

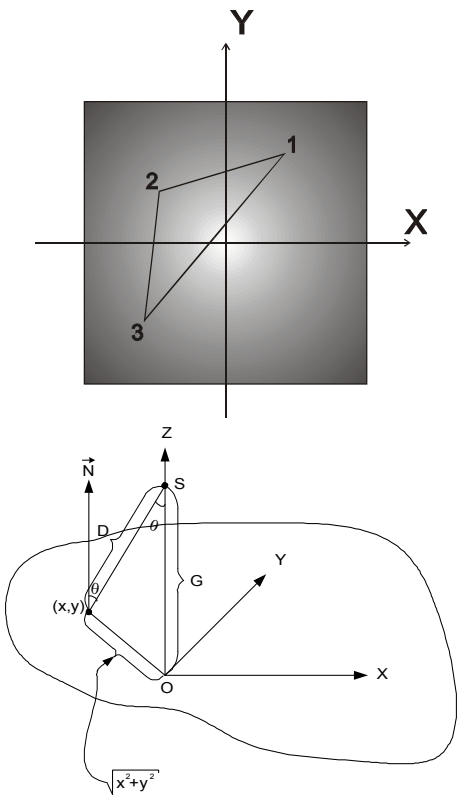
S=Var

P – есептелінбейді

Жеке шектерден құралған объектілерді бояу есебін текстуралау көмегімен есептеуге болады Фонг әдісін текстура салуға әкеліп, есептеулерді қысқартуға болады (12.6 сурет).

S – нақты жарық бастауы, ортасында максималды жарық болады.

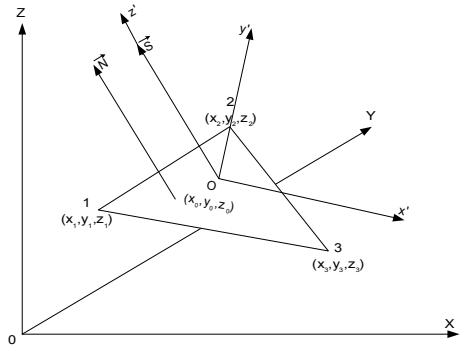
Негізгі идеясы: жадыда текстура есептеледі (12.6 сурет), содан соң объектілердің тек ішкі аймақтарын бояу алынған текстураны қолданумен іске асады.



Сурет 12.6

Егер η сәйкес образбен қарастырсақ, онда қосалқы текстурадағы сәйкес нүктенің жарық болады.

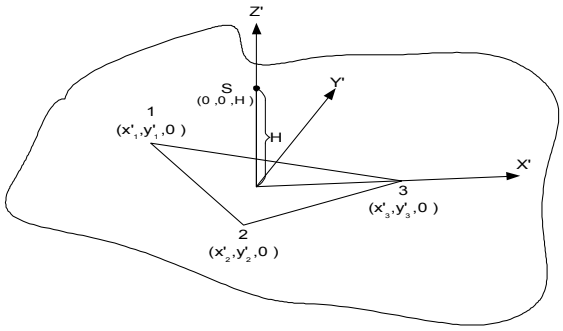
Еркін үшбұрыш үшін координат нүктелерінің есебі.



Сурет 12.7

Бүкіләлемдік координаттар жүйесінде үшбұрыш берілген (12.7 сурет) жарықтандыруды оның ішкі аймақтарды бояуын жүргізу қажет.

Ол үшін координаттар жүйесі (x’,y’z’) салынады, басы $O(x_0,y_0,z_0)$ нүктесінде, OZ осі жарық бастауы S арқылы және нормальға $\vec{N}$ параллель болуы керек, ал OX және OY үшбұрыш жазықтығында жатыр.



Сурет 12.8

$[x';y';z']=[x-x_0;y-y_0;z-z_0]\cdot M$, мұнда M – түрлендіру матрицасы.

Бір М матрицасын алайық, және мұндағы 1,2,3,S нүктелерінің проекциясы 3.2.6 суреттерде координаталарымен көрсетілген нүктелерге түсуі керек.

$N_V=(V_2-V_1)\times(V_3-V_1)$ – жүйелі емес нормальдың векторы.

Негізгі әдеюиеттер: 2 [380-401бет]
Қосымша әдебиеттер: 11 [201-256 бет]
Бақылау сұрақтары:
1. Жарықтанудың жәй моделі дегеніміз не?
2. Жазықтыққа түсірілген нормальдің анықталуы?
3. Гуро әдісімен бояу қалай жүзеге асады?
4. Фонго әдісімен бояу қалай жүзеге асады?
5. Жарықтанудың қандай модельдері болады?

Дәріс 13. Сәулелердің трассировкасы

13.1 Сәуле трассировкасы жолымен көрінетін жоғарғы бет алгроитмін анықтау

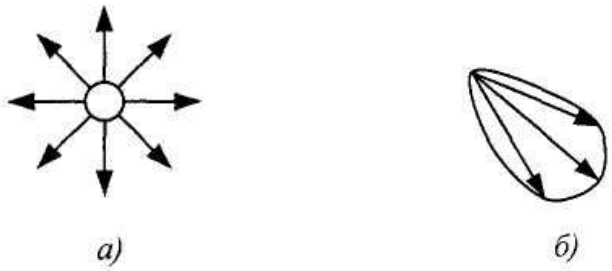
Бұл күні сәулелердің трассировка әдісі ең қуатты және шағылысуды құрастыратын әмбебап әдістері болып табылады. Ең қиын үшөлшемді көріністі сапалы шағылдыратын сәулелердің ізсалу алгоритмдері жөнінде көптеген мысалдар бар. Сәулелердің трассировка әдістерінің негізінде жәй және анық түсініктер жатыр, олар біздің қоршаған ортаны қабылдауымызды оңайлатады. Біз қоршаған ортаны қалай көреміз? Ең алдымен біз нені көре аламыз? Бұл арнайы пәндерде оқытылады, ал кейбір жағдайларда бұл философия сұрағы. Бірақ бұл жерде біз, бізді қоршаған нәрселер жарыққа қатысты мынадай қасиеттерден тұрады:

– жарық шығарады;
– шағылыстырады және жұтады;
– өзінен өткізеді.

Осы қасиеттердің әрқайсысын кейбір сипаттамалармен сипаттауға болады. Мысалы, жарық шығаруды өсімталдық пен спектрдің бағыты арқылы сипаттауға болады.

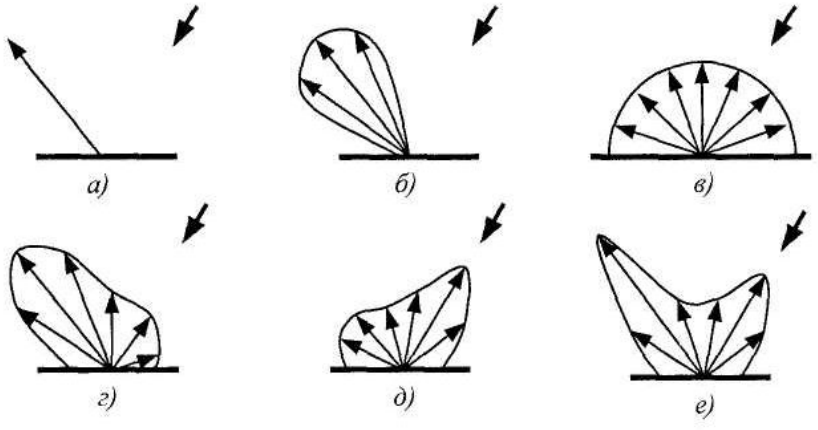
Жарық шығару бір нүктеден шығуына (алыс жұлдыз) немесе бойлық бойынша (вулканнан шашырап шыққан ыстық лава) шығуына болады. Жарықтың шашырауы бір жіңішке сәуле арқылы (бір нәрсеге бағыттандырылған лазер сәулесі) конус арқылы (прожектор), тегіс жан-жаққа шашырау (күн) арқылы немесе тағы басқаша. Шағылдыру қасиетін (жұту) диффузиялық шашырау немесе айналық шағылдыру арқылы сипаттаймыз. Түссіздікті өсімталдықтың және сынудың әлсіреуі арқылы сипаттауға болады.

Жарық энергиясын мүмкін болатын жарық сәулелеріне бөлу векторлық диаграмма арқылы жүзеге асады, мұнда вектор ұзындығы вектордың өсімталдығына (интенсивтілігіне) сәйкес келеді (13.1 сурет).



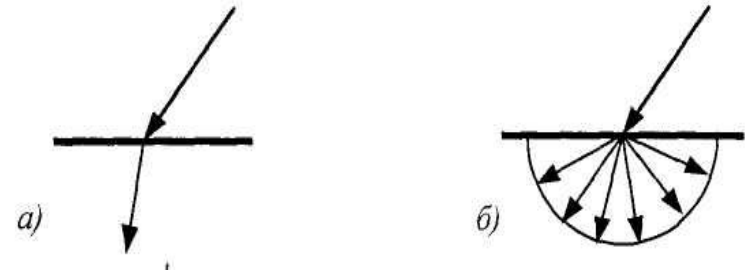
Сурет 13.1

Алдыңғы бөлімдерде біз сіздермен ең көп кездесетін айналық және диффуздық шағылу түрлерімен таныстық. Кейбір әдебиеттерде мұны теріс, яғни айналылық емес шағылдыру, онда шағылдырудың интенсивтік максимумы өз көзіне бағытталуына сәйкес келеді деген пікір жиі кездеседі. Кері айналық шағылдырумен кейбір жердің өсімдік алаңы, мысалы жоғарыдан қарағандағы күріш алқабы ие бола алады.



Сурет 13.2 – Шағылған: а – шынайы айна, б – шынайы емес айна, в – диффуздық, г – айналықтың және диффуздықтың саны, *д* – кері, е – диффуздықтың, айналықтың және керілік сан жинағы

Екі шеткісі, жарықтың шың сыну көрінісі 13.3 суретте көрсетілген.



Сурет 13.3 – Сыну: а – шынайы (идеальное), б – диффуздық.

Кейбір объектілер сәулелерді өте көп түрлермен сындырады, мысалы, мұз басқан шыны (әйнек).

Бір объект бір жағынан жарық көзі, ал келесі жағынан тек жарық өткізуші немесе шағылдырушы да болып табылады. Мысалы, аспан күмбезі, кейбір үшөлшемді көріністе байлық жарық көзін сейілдендіруі мүмкін, ал кейбір жағдайда бұл аспан күн жағынан жарықталынып, түссіз орта бола алады.

Жалпы жағдайда осы объектілердің әрқайсысы жоғарыда көрсетілген үш қасиетпен суреттеледі. Тапсырма ретінде жоғарыда айтылған үш қасиеттерімен ие болатын, яғни жарық өткізетін объектілерді мысал ретінде алыңыз. Бәлкім сіздің ойыңыз қып-қызыл болып балқыған шыныдан гөрі басқа объектілерді мысалға алар. Енді өзіңе бірнеше кеңістік объектілерін қосатын көріністердің қалай құрастырылатынын қарастырайық. Жарық шығарушы объектінің нүктелерінен сәуле шығып жатыр деп ойлайық. Мұндай сәулелерді алғашқы деп атайық, себебі олар қалғандарын жарықтандырып жатыр. Ең маңыздысы болжау болып табылады, яғни жарық сәулесі еркін кеңістікте тік жолақ сызығымен таралады (бірақ физиканың кейбір бөлімінде бәлкім болатын қисық шағылу себебі де оқылады). Бірақ геометриялық оптикада жарық сәулесі шағылатын жазықтық немесе сыну шекарасын кездестірмегенше тарала береді дейді.

Жарық шағылу көзінен әр түрлі бағытта саны жоқ алғашқы сәулелер таралады (лазер сәулесін де тіп-тіке етіп түсіруге болмайды, бәрібір де сәуле конус болады немесе бірнеше сәулелер жиыны болады).

Кеәбір сәулелер еркін кеңістікке кетеді, ал кейбіреулері (олар өте көп) басқа объектілерге түседі. Егер сәуле мөлдір денеге түссе, ал сынғанда әрі қарай кетеді, бірақ та кеәбір жарық энергиясының бөлігі жұтылады. Осы сияқты, егер сәуле жолында айна секілді шағылдырушы зат кездессе ол өзінің бағытын көрсетеді, ал жарық энергиясының бір бөлігі жұтылады. Егер объект айна секілді мөлдір болса (мысалы, жәй шыны) онда 2 сәуле болады. Бұл жағдайда сәуле бөлінеді.

Алғашқылық сәулелердің объектілеріне әсер етудің нәтижесінде екіншілік сәулелер пайда болады деп айтуға болады. Сансыз көп екіншілік сәулелер еркін кеңістікке таралады, бірақ кейбіреулері басқа объектілерге де түседі. Осылай, сансыз рет шағылып және сынып, бөлек жарық сәулелері бақылау нүктесіне келіп түседі – адам көзіне немесе камераның оптикалық жүйесіне. Бақылау

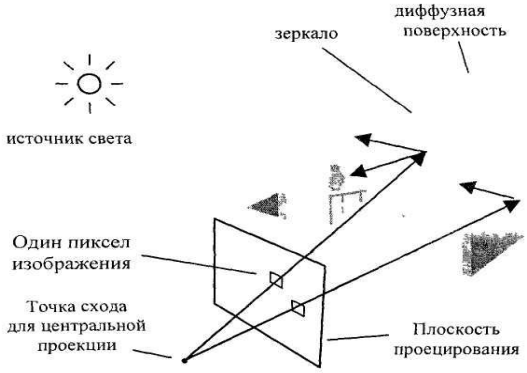
нүктесіне жарық шығарушыдан шыққан алғашқы сәулелердің жартысы түсуі мүмкін. Осылайша, көріністің түсі спектрдің жарық шығару көзінен шыққан алғашқы сәулелердің интенсивтілігімен анықталады, және де объектілердегі жарық сәулелерінің өзінің жолында сәйкес келетін сәулелермен кездескендегі пайда болатын жарық энергиясының жұтылуымен анықталады. Бұл сәулелерден көрініс құрастыру өздігінен қиын. Алгоритм құру арқылы көрініс құрастыруға тырысуға болады. Мұндай алгоритмде объектіге немесе камераға түсетін алғашқы сәулелердің қайсысы шектен шыққанын анықтау керек. Сонан соң шектен шыққан екіншілік сәулелердің объектіге немесе камераға түскендерін қарастырамыз. Осылай, әрі қарай жалғастырамыз. Бұл әдісті сәулелердің тікелей трассировка әдісі деп аталады, бірақ бұл әдістің бағасы күмән келтіреді. Шынында, әр жаққа таралған сәулелерді қалай бақылауға болады? Сәулелердің толық шектен шығуы мүмкін емес. Бір амалымен мұны соңғы нәтижелі санға әкелсек те (мысалы, бүкіл сфераның бағытын бұрыштық секторға бөліп, шексіз жіңішке түзулермен белгілемей, секторлармен белгілесек), бәрі-бір бұл әдістің басты кемшілігі қалады-нәтижесінде керек емес болатын сәулелердің есебімен байланысты көптеген операциялық амалдары бар. Қазіргі кезде ол солай шешіледі. Сәулелердің кері трассировка әдісі жарық сәулелердің шектен шығуын қысқартады. Бұл әдіс 80 жылдары *Уиттед пен Кэя* жұмыстарының нәтижесінде табылған. Бұл әдіске сәйкес сәулелерді бақылау жарық көзінен емес, ал кері бағытта-бақылау нүктесінен. Осылай тек қана көрініске өз үлесін қосатын сәулелер ғана есептеледі.

Кері трассировка әдісімен кейбір үшөлшемді көрінісінің бейнесін қалай алуға болатынын қарастырайық. Проекция жазықтығы көптеген төртбұрышты пиксельдерге бөлшектеніп кетті делік. Түсу центрінен әр түрлі қашықтықта жатқан проекция ценртін таңдап алайық. Осы түсу центрінен квадрат центрінен (пиксельден) проекция жазықтығына түзу жүргіземіз (13.4 сурет). Бұл кері трассировканың алғашқы сәулесі болады. Егер тіке түскен түзу көріністің бір немесе бірнеше объектілерге түссе, онда ең жақын қиылысқан түзуді таңдаймыз. Көрініс пикселінің түсін анықтау үшін объектінің қасиетін қарастыру керек және де бағытталған объекті нүктесіне қандай жарық сәулесі сәйкес келетіндігін анықтаған жөн. Егер объект айна секілді (кейбір жерлерінде болса да) болса, екіншілік құлау сәулесін құрастырамыз, шағылу сәулесін біріншілік трассировка сәулесі деп есептейміз.

Жоғарыда біз айналық шағылдыруды қарастырдық және белгіленген вектордың нормалі мен қыздырылған сәулелердің шағылған сәулелерінің векторы табылатын формула алдық. Бірақ бұл жерде бізге шағылған сәуленің векторы белгілі, ал құлаған векторды қалай табамыз? Бұл үшін айналық шағылу формуласын қажетті сәуленің құлау векторын шағылу векторы ретінде алып, қайтадан қолдануға болады. Яғни керісінше шағылған.

Идеал айна үшін екіншілік сәуленің басқа объектілермен қиылысқан келесі нүктесін анықтап бақылаған жеткілікті. “Идеал айна” термині нені білдіреді? Мұндай айнада теп-тегіс беті бар деп ойлайық, сондықтан бір шағылған сәулеге тек бір ғана түсу сәулесі сәкес келеді. Айна қараңғыланған болуы мүмкін, яғни жарық сәулесінің бір бөлігін жұтуы мүмкін, бірақ сонда да ереже қалады: бір

сәуле түседі-бір сәуле шағылады. “Идеал емес айнаны” да қарастыруға болады. Бұл-беттің тегіс емес екенін білдіреді. Шағылған сәуленің бағытына бірнеше түсу сәулелері сәйкес келеді (немесе керісінше, бір түсуші сәуле бірнеше шағылушы сәуле тудырады) кейде түскен сәуленің осіне симметриялы болмайтын конус тудыруы мүкін. Конус интенсивтіліктің кейбір таралу заңына сәйкестенеді, ең қарапайымын Фонг моделі суреттейді – кейбір деңгейлерге шығарылған бұрыштың косинусы. Идеал емес айна ізсалуды қиындатады – бір емес, бірнеше түсу сәулелерін бақылау керек және де берілген объекті нүктесінен шығатын көрінетін сәулелердің қосылысын ескеру керек.

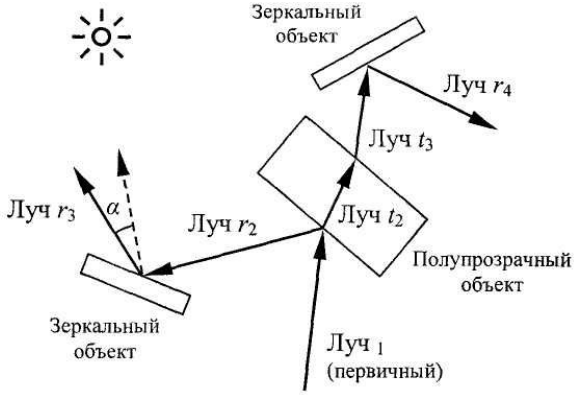


Сурет 13.4 – Сәулелердің кері трассировка схемасы

Егер объект түссіз болса, онда сынғанда алдыңғы трассировкалаушы сәуле шығатын жаңа сәуле құрастыруымыз керек. Бұл жерде сынуға жарамды қайтымдылықты қолдануға болады.

Ізделуші сәулесінің векторын есептеу үшін жоғарыда сыну сәулесінің векторының формуласын қолдануға болады, бірақ сыну кері бағытта жүзеге асады деп қарастырамыз (13.5 сурет).

Егер объект диффуздық шағылуға шағылған сәуленің интенсивтілігі, бәрімізге белгілі, жарық көзінің және нормальдің сәулесінің векторының бұрышының косинусына пропорционал болады. Бұл жерде жарық көзі ретінде жарық энергиясын бере алатын кез-келген нүктеден көрінетін объект алынуға болады.



Сурет 13.5 – Айналық шағылу мен сыну қасиеті бар объектіге кері трассировка құрылысы көрсетілген

Егер өтіп бара жатқан кері трассировкасында сәуле кез-келген объектіні қиып өтпегендігі анықталса, тек еркін кеңістікке кетсе, онда бұл сәулеге трассировка тоқтатылады.

Суретте көрсетілгендей, сәулелердің кері трассировкасының шектен шығуы қысқарғанмен, сансыз көп қарастырылатын сәулелер санынан құтқара алмайды. Шынында да, бұл әдіс көріністің әр нүктесіне кері трассировкасымен алынған жалғыз сәулені алуға мүмкіндік береді. Бірақ та екіншілік шағылған сәулелер сансыз көп кете беруі мүмкін. Мысалы, егер бір объект екінші бір объектіден жарық шағылдыра алатын болса, және де бұл объектілердің көлемдері біршама үлкен болса, онда осындай объектілерді шағылдыратын нүктелерді келісетін сәулелер құрастыру үшін қарастырған жөн, мысалы, диффуздық шағылуда? Барлық нүктелер екені анықталған.

Тәжірибеде кері трассировка әдісіне кейбір тыйымдар салынады. Олардың кейбіреулері кескіннің синтезіне арналған жаттығуларды есептеп шығару үшін қажет, ал кейбіреулері:

Барлық объектілердің түрлерінен кейбіреулерін бөліп қараймыз, оларды *жарық көзі* деп атаймыз. Жарық көздері жарықты тек шығара алады, бірақ оны шағылдыра алмайды немесе сындыра алмайды. Тек қана *нүктелі жарық* көздерін қарастырайық.

Шағылдыру беттерінің қасиеттері екі құрауыш қосындысымен бейнеленеді – диффузиялық және айналық.

Өз кезегінде айналылық (зеркальность) екі құрауыштармен бейнеленеді. Біріншісі (*reflection*) жарық көзі болмайтын объектілерден шыққан шағылдыруды есептейді. Әрі қарай трассировка үшін тек бір ғана айналы шағылған *r* құрастырылады. Екінші құрауыш *{specular)* жарық көздерінен түскен түсті дақтарды білдіреді. Бұл үшін барлық жарық көздеріне сәулелер түсіріледі және бұл сәулелердің айналы шағылдырылған кері трассировка (*r*) сәулелерімен құралған бұрыштары анықталады. Айналы шағылдырылуда бет нүктесінің түсі шағылған нүктенің түсі арқылы анықталады. Жәй жағдайда айна бетінің өз түсі болмайды.

4. Диффузиялық шағылдыруда жарық көздерінен шыққан тек сәулелер ғана берілген жарық көзіне түсірілген сәуле басқа объектімен жабылса, онда объектінің бұл нүктесі көлеңкеде орналасқан. Диффузиялық шағылдырылуда жарықтандырылған бет нүктесінің түсі өз бетінің түсі және жарық көзінің түсі арқылы анықталады.

Түссіз (мөлдір) объектілер *{transparent)* үшін негізінен толқын ұзындығының сындыру коэфициентінің тәуелділігі ескерілмейді. Кейде түссіздік жалпы алғанда сындырусыз модельдендіреді, яғни сындырылған *t* сәуленің бағыты түсірілген сәуленің бағытымен сәйкес келеді.

Объектілердің жарықпен жарықтандырылғанын есептеу үшін басқа объектілермен шашыратамыз, фондық құрастырушы енгізіледі *{ambient).*

Ізсалуды аяқтау үшін нәтижелі түске өз үлесін қоспайтын қайсыбір өтпелі жарықтандыру мәнін енгізеді немесе итерация санын шектейді. Уиттед моделіне сәйкес қайсыбір нүктесінің түсі сан жиыны (сумма) интенсивтілігімен анықталады.

$$I(\lambda) = K_a I_a(\lambda) C(\lambda) + K_d I_d(\lambda) C(\lambda) + K_s I_s(\lambda) + K_r I_r(\lambda) + K_t I_t(\lambda),$$

мұндағы, λ – толқын ұзындығы, $C(\lambda)$ – объект нүктесінің бастапқы берілген түсі, K_a, K_d, K_s, K_r және K_t – берілген объектінің фондық көмескі жарық параметрлерінің қасиеттерін ескеретін, диффузиялық шашыратудың, айналылықтың, шағылдырудың және түссіздіктің коэффициенттері:

I_a – фондық көмескі жарықтың өспенділігі (интенсивность);

I_d – диффузиялық шашыратуды есептейтін өспенділік;

I_s – айналылықты есептейтін өспенділік;

I_r – шағылған сәуледен өтетін шағылу өспенділігі;

I_t – сынған сәуледен өтетін шағылу өспенділігі.

Өайсыбір объект үшін көмескі жарықтың фондық өспенділігі жалпы жағдайда константа. Қалған өспенділік үшін формулалар жазайық. Диффузиялық шағылдыру үшін:

$$I_d = \sum_i I_i(\lambda) \cos\theta_i,$$

$I_i(\lambda)$ – *i-жарық* көзінің шағылу өспенділігі, θ_i – объект бетінің нормалі мен i –жарық көзінің бағытының арасындағы бұрыш.

Айналылық үшін:

$$I_s = \sum_i I_i(\lambda) \cos^p \alpha_i,$$

p – дәреже көрсеткіші бірлікпен бірнеше жүздікке дейін (Фонг моделіне сәйкес); α_i – шағылған сәуле мен (кері із салу) *i*-жарық көзіне түсірілген сәулесі арасындағы бұрыш.

Шағылған сәуле арқылы өтетін (I_r) және де сынған (I_t) сәуле арқылы өтетін шағылу интенсивтілігін сәуленің жүріп өткен ұзындығына байланысты әлсіздену интенсивтілігін есептейтін коэффициентке көбейтеді. Мұндай коэффициент $e^{-\beta d}$ түрінде жазылады, бұл жердегі d – өткен ұзындық. β – сәуленің таралу аймағының қасиеттерін есептейтін әлсіздену параметрі.

Сәуленің кері трассировкасының базалық операциясын функция түрінде жазып, СӘУЛЕ деп атаймыз. Ол осы трассировка сәулесінің интенсивтілік мәнін есептейді.

Мұнда СӘУЛЕ функциясы схемалық және жалпылама түрінде көрсетілген, (бірақ онша жалпылама емес-мысалы, мұнда диффузиялық сыну көрсетілмеген). Кері трассировка процесін дұрыс функцияландыру үшін функция мәтінінде көрсетілмеген тағы бірнеше қимылдарды ескеру қажет. Мұны өзіңіз тапсырма ретінде орындаңыз. Мысалға алдын-ала болжау керек, бастапқы сәуленің трассировкасы кезінде басқа объектілермен кедергіге ұшырамайтын CR $^{\circ 6}$ – көзінің көрінісін алуға болады. Одан басқа, шекарадан қайсыбір объектінің ішіне сынған сәуле шығарылғанда және де сыртқы шекараға жеткенде, бұл мезетте объектінің ішіне бағытталған айналы сәуленің пайда болуын бұғаттауға (заблокировать) болады. (мысалы, қайсыбір бетпен шектелген, көлем түрінде берілген әйнек линзаны модельдендіргенде) және де айналы сәуле сыну сәулесінің трассировкасы кезінде сыну басталған көлемнің

шекарасынан, бірақ сәуле басқа объектіні қиса ғана шығарылуын алдын-ала болжау керек. (мыс, аквариумдегі балықтардың кескініндей). Бұл үшін “Сәуле түрі” және “объект нөмірі” деген аргументтер көрсетілген. “Тістесуді болдырмау үшін (мыс, айналы қабырғалармен шектелген кеңістікке сәуленің түсуі) “Интеграция нөмірі” аргументі қарастырылған. Бұл аргументті бастапқы сәулені белгілегенде де қолдануға болады.

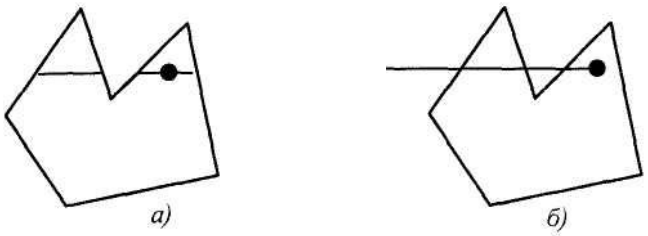
Функция денесінде осы функцияның шарттары бар екендігін, яғни функцияның анықтамасы *рекурсивті* екендігін байқаған боларсыз. Осылай барлық сәулелерді модельдеуге болады: бірінші бастапқы сәулені, содан кейін екіншілерін, үшіншілерін және т.б.-ларын. Кескіннің 1 нүктесінің кері ізсалуының бүкіл процесі 1 ғана жол түрінде көрсетілген.

I= СӘУЛЕ (1, *бастапқы, проекциялаудың бағыты,* 0).

Берілген проекцияға сәйкес келетін, бастапқы сәуле үшін беру қажет. Егер проекция центрлік болса, онда бастапқы сәулелер ортақ нүктеден тарайды, параллель үшін бастапқы сәулелер параллель болады. Сәуленің берілгенін беруге болады, мысалы, кесіндінің бастапқы және соңғы нүктелерінің координаталары арқылы, бағытпен және бастапқы нүктенің координатасымен, немесе тағы басқа жолдармен. Кескінделіп отырған сценаның проекциясын түгелдей бастапқы сәуленің берілуі анықтайды. Біз проекциялардың негізгі түрлерін және солармен байланысты координаттардың түрленуін қарастырдық. Ал сәуленің кері трассировкасы кезінде координаттардың түрленуіі мүлде қажет емес. Проекция автоматты түрде шығарады – соның ішінде тек қана жазық емес, және де мысал үшін цилиндрлік немесе сфералық. Бұл трассировкасының әмбебап әдісі.

Трассировка кезінде сәуленің түзу сызығының объектімен қиылысу нүктесін анықтау қажет. Қиылысу нүктесін анықтау әдісі объектіге және оның графикалық жүйеде берілуіне байланысты болады. Мысалы, көпқабырғалы және полигональдық тор түрінде берілген объектілер үшін аналитикалық геометрияда қарастырылған бізге белгілі түзумен жазықтықтың қиылысу нүктесін анықтайтын әдісін қолдануға болады. Ал егер сәуле мен шектің қиылысу нүктесі шек контурының ішінде жатуы тиіс.

Кез-келген нүктенің полигонында жататын-жатпайтындығын анықтау үшін бірнеше әдіс белгілі. Бір әдістің екі түрлі шығару жолдарын қарастырайық (13.6 сурет.).



Сурет 13.6 – Егер: а – нүкте қиюшы кесіндіде жатады,
б – қиылысу саны шексіз болса, нүкте ішінде орналасады.)

Бірінші әдіс. Берілген *Y* нүктесінің координаталарына сәйкес келетін контур мен горизонтальдың барлық қиылысу нүктелері табылады. Қиылысу нүктелері *X.* Координаталары мәндерінің өсуіне байланысты бөлінеді. Қиылысу нүктелерінің жұптары кесінділер түзеді. Егер қарастырылған нүкте кесінділердің біреуінде орналасса (бұл үшін берілген нүктенің *X* координаталары мен кесінді шегі салыстырылады) онда ол ішінде орналасады.

Екінші әдіс. Қарастырылып отырған нүктемен бір жазықтықта жататын және полигон контурында орналаспайтын нүкте анықталады. Табылған сыртқы нүкте және қарастырылған нүкте горизонталь кесіндінің шектері болып табылады. Берілген кесінді мен полигон контурының қиылысу нүктелері анықталады. Егер қиылысулар шексіз болса, онда қарастырылған нүкте орналасады дегенді білдіреді.

Егер сәуле мен объектілердің бірнеше қиылысу нүктелері болса, онда түсірілген сәуленің бағыты бойынша ең жақын нүкте таңдалады.

13.2 Сәуле трассировкасының глобальды модель жарықтануы

Қабық ретінде шар, параллепипед, цилиндр және басқа да қарапайым денелерді қолдануға болады.

Егер объектілер өте көп болса, онда объектілерді топтарға біріктіруге болады – бірнеше объект үшін 1 қабық. Осылай қабықтар иерархиясы құрылады: жалғыз объект үшін қабықтың төменгі деңгейі; келесі деңгейде – қабықтардың қабықтары тағы сол сияқты. Мұндай ағаш тәрізді құрылым бірнеше деңгейден тұруы мүмкін. Бұл шексіздік процесін едәуір тездетуге мүмкіндік береді, жұмыс уақытын логарифм санына пропорционал (теориялық) істейді.

Қабықтар әдісін тек сәуле трассировасына ғана емес қолдануға болатынын айтып өту қажет.

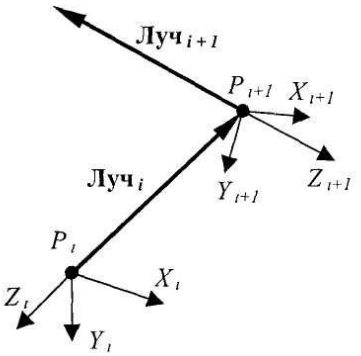
Кері трассировка кезінде орындалатын кейбір жұмыстарды қарапайым жасау үшін келесі әдісті қолдануға болады. Ол әдісті кітаптың авторы өндірген және мәні мынада. Сәулелердің трассировкасы кезінде әр сәулемен локальдық координаталар жүйесі байланысады. Бұл үшөлшемді декарттық жүйенің орталығы трассировканың түсірілген сәулесінің шыққан нүктесі болады. Z осі сәулеге қарсы бағытталған, *X және* Г осьтерінің орналасуы бәрібір. Осылайша, сәуленің объектімен қиылысу нүктесінің *X және Y* координаталары әрқашан нольге тең. Бұл сәуленің қиылысу нүктесінің координаталарын анықтауды жеңілдетеді, өйткені сәуленің бағыты әрқашан бірдей, сондықтан тек Z координатасын ғана анықтаймыз. Жазық полигонды шектер үшін мұны Z координаталарының сәйкес келетін төбелерінің сызықты интерполяциясының көмегімен орындауға болады. Одан басқа, басқа жұмыстар да жеңілденеді. Жеке жағдайда ең жақын қиылысу нүктесін табу үшін Z координатасын қарастыру жеткілікті. Операцияларды қарапайым жасау басқа операциялардың қиындатылуынан болатынын ескерту қажет,-мысалы, әр локальдық жүйе үшін координаттарды түрлендіру коэффициенттерін есептеу керек, және де объектілер координаталарының түрленуін орындау керек.

13.7 суретте локальдық координаталар көрсетілген.

Ең жақын объектіні P_{i+1} нүктесінде қиып өткен, P_i, нүктесінен шыққан *i*-ші сәуле болсын. Координаталар жүйесінде $(X_i\ Y_i,\ Z_i,)$ P_{i+1}. Нүктесінің координаталары (0, 0, z_p)-*ге* тең. Жаңа (*i*+l)-ші сәуленің бағыты өзіне параллель радиус – вектор түрінде осы координаталар жүйесінде анықталады делік, яғни $(X_i,\ Y_i,\ Z_i)$. Бұл радиус-вектордың координаттарын (x_r, y_r ,z_r) арқылы белгілейік. (*i*+ 1)-ші сәулемен байланысатын жаңа координаталар жүйесін өалай анықтаймыз? Бұл координаталар жүйесін былайша алуға болады: Басында Z_i осі бойынша ($X_i,\ Y_i$, Z_i) жүйесін z_p өлшеміне ығыстырамыз, сосын Z_{i+1} осі жаңа сәулеге қарсы бағытталатындай бұрылу орындаймыз. Түрлендіру формулаларын мына түрде жазамыз:

$$X_{i+1} = X_i \cos\alpha - Y_i \sin\alpha,$$
$$Y_{i+1} = X_i \sin\alpha \cos\beta + Y_i \cos\alpha \cos\beta - (Z_i - z_p)\sin\beta,$$
$$Z_{i+1} = X_i \sin\alpha \sin\beta + Y_i \cos\alpha \sin\beta + (Z_i - z_p)\cos\beta,$$

мұндағы α және β – бұрылу бұрыштары.



Сурет 13.7 – Түсірілген сәулелер үшін локальдық координаталар

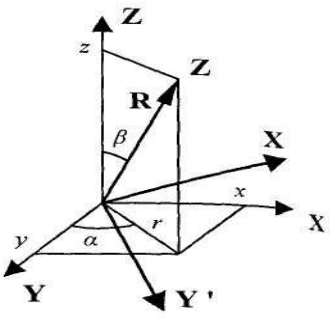
Бұрылуды бұрыштар арқылы емес, радиус-вектор арқылы белгілейміз, *Z* осінің жаңа бағытын көрсетеді. 13.8 суретті қарастырайық. 1 вектордың координаталары (*x, y, z*) болсын. Радиус-вектордың ұзындығы *R-ға тең болсын:*

$R = \sqrt{x^2 + y^2 + z^2},$ ал жазықтықтағы проекцияның ұзындығы *(XOY):*

$r = \sqrt{x^2 + y^2}.$

Сонда:

$\cos\alpha = y / r,$
$\sin\alpha = x / r,$
$\cos\beta = z / R,$
$\sin\beta = r / R.$



Сурет 13.8 – Ось бұрылысы

Бұл мәндерді координаттар түрлендіру формуласына қояйық, және жаңа жарқыраған сәуленің радиус-векторы *Z,+\ осіне* қарсы бағытталғанын, яғни (*x, y, z*) = (~x_r, -y_r, -z_r) екендігін ескере отырып, формуланы мынадай түрде жазамыз:

$$X_{i+1} = -X_i(y_r/r) + Y_i(x_r/r),$$
$$Y_{i+1} = X_i(x_r/r)(z_r/R) + Y_i(y_r/r)(z/R) - Z_i(r/R) + z_p(r/R),$$
$$Z_{i+1} = -X_i(x_r/R) - Y_i(y_r/R) - Z_i(z_r/R) + z_p(z_r/R).$$

Матрицалық формада

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \\ Z_{i+1} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{i+1} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}.$$

A,+i матрицасы аффиндік қозғалыс және бұрылыс түрлендіруіне сәйкес келеді. Сәуленің жарқырау барысында біз әлемдік жүйеден ағып жатқан сәуле координаттар жүйесіне түрлендіруін орындауымыз керек. ${}^{n}P^{e}$ матрицасын *M,+1* өрнегімен өрнектейміз:

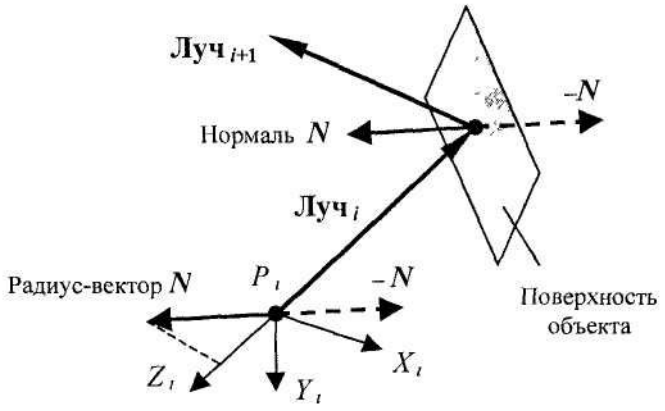
$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \\ Z_{i+1} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{M}_{i+1} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

Поскольку $\mathbf{M}_1 = \mathbf{A}_1$, то

$$\begin{bmatrix} \mathbf{M}_{i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{i+1} \end{bmatrix} \begin{bmatrix} \mathbf{M}_i \end{bmatrix}.$$

Осылайша жаңа сәуле координаттар жүйесі матрицасы алдыңғы сәуле матрицасы мен қозғалыс және бұрылыс матрицасының көбейтіндісінен пайда болады.

Шағылысу және сыну есебіндегі маңызды кезең бұлнормальдің көрінетін жарқырайтын сәуленің аймақ бетінің жағы үшін анықтау (13.9 сурет).



Сурет 13.9 – Нормальді объектінің көрінетін жағын таңдауы

Егер түбір астындағы мән теріс болса, онда сыну сәулесі тұрғызылмайды. Координаттардың локальдық жүйесін пайдалану ағып жатқан жарқырайтын сәуле сызығымен сәйкес келетін *Z* осі объект қабығымен қиылысын табуға жеңілдік береді. Шар мен қабық қиылысын табу ең оңай болып келеді. Себебі кез-келген координаттар локальдық жүйесінде сызықты түрлендірумен берілген шар проекциясы шеңбер болғандықтан оның центрінің локальдық координаттарын тауып $(Xc)^2 + (Yc)^2 <$ *(қабық радиусы)*^{2}. теңсіздігі орындалатындығын тексеру жеткілікті болады. Егер теңсіздік орындалатын болса, онда ағып жатқан сәуле қабықты қияды.

Қабық ретінде басқа да пішінді қолдануға болады, мысалы цилиндр немесе параллепипед. Еркімен бұрылған цилиндрмен немесе параллелипипедпен сәуленің қиылысуын анықтау оңай есеп емес. Бұл үшін келесіні қолдануға болады. Бізге *X, Y* осьтерінің координаттары локальдық жүйеде орналасқаны парықсыз болса (бастысы, *Z* осі сәуле бойында жатты), бірақ бізге берілген локальдық жүйе үшін координаттарды түрлендіру формуласы тек вертикаль сызықтардың бірдей бағыт сақтауды қамтамасыздандырады. Бұрын айтылғандай, бұл қасиет таңдалған әдістің ось бұрылысы аксонометриялық проекцияда ғана қолданылады. Сондықтан қабық проекциясын координаттар локальдық жүйесінде барлық бұрылыс үшін тек тік төртбұрыш түрінде (*X0Y*) жазықтығында кейбір өлшемдерімен анықтауға болады. Қабық пен сәуленің қиылысуы тексеріледі, яғни локальдық координаттар нүктесі қабық сыртындағы $(X_i, Y_i) = (0, 0)$ тік төртбұрыштың ішінде жатқандығы тексеріледі.

Ал енді сәуленің кері жарқырау әдісі бойынша жалпылама қорытынды жасадық: Оң қорытынды:

1. Әдістің әмбебаптығы, күрделі кеңістік бейне сұлбалары синтезіне қолданылуы.

2. Бұл әдістің кесілген нұсқалары да реалистік бейне алуға мүмкіндік береді. Мысалы, егер тек бастапқы сәулелермен ғана шектелсек (пропорциялау нүктесінен), көрінбейтін нүктелерін жоюға мүмкіндік береді. Кейінгі бір-екі сәулелердің жарқырауы көлеңкені, айналықты (зеркальность), мөлдірлікті (прозрачность) береді.

3. Координаттардың барлық түрленулері (егер ондай болса) сызықты, сандық текстурамен жұмыс істеуге оңай болып табылады.

Растрлық бейнесінің бір пикселіне бірнеше жақын орналасқан сәулелерді жарқыратуға болады, содан кейін баспалық эффектті жою үшін олардың түстерін орталауға болады.

Бейненің бөлек бір нүктесінің есебі қалғандарынан тәуелсіз орындалатындықтан бұл әдіс параллель есептеу жүйесінде бір уақытта сәулелердің жарқырату үшін қолдануға болады.

Кемшіліктері:

Диффуздық сыну және шағылу модельдеуінің проблемасы.

Бейненің әр нүктесіне көптеген операциялардың орындалуы, сәулелерді жарқырату бейне синтезі алгоритміне жатады.

Негізгі әдебиет: 4 [284-295]
Қосымша әдебиеттер: 8 [381-395]
Бақылау сұрақтары:
1. Сәулелерді жарықтандыру алгоритмінің мәні неде?
2. Сәулені жарықтандыруды қалай түсінесіз?
3. Бұл алгоритм қандай жағдайда қолданылады?
4. Бұл алгоритм көрінбейтін сызықтарды жоюға қолданылады ма?
5. Алгоритмнің жағымды жағын айтыңыз?

Дәріс 14. OpenGL негізгі мүмкіншіліктері

OpenGL негізгі мүмкіншіліктері:
– базалық жиындар: нүктелер, сызықтар, көпбұрыштар және т.б.;
– көріністі және координаттық түрлендірулер;
– көрінбейтін беттер мен сызықтарды жою (*z* - буфер);
– сплайнды сызықтар мен беттерді тұрғызуға қолдану;
– текстураны салу және жарықтандыруды қолдану;
– арнаулы эффектерді қосу: тұман, мөлдірліктің өзгерісі, түстердің түйіндесуі (blending), сатылықта жою (anti-aliasing).

Айтылып кеткендей, графикалық жүйенің базалық функциясы мен графикалық ақпараттарды көрсету және қолданушымен қатынасын ыңғайлы болу үшін әртүрлі платформаларға арналған OpenGL іске асуы бар. Операциялық Windows және Unix (WGL және GLX сәйкес) жүйелері үшін терезелік жүйе көмегімен ақпараты көрсету үшін, сонымен қатар консольдық қосымшаларды құруға қолданылатын GLAUX және GLUT кітапханалары құрылған GLAUX пен кеәінірек құрылған GLUT кітапханасы қарастырылатын болады. GLUT кітапханасы құрамына күрделірек функцияларды іске асыру, шар, куб, цилиндр, диск сияқты геометриялық примитивтер жинағы, сплайндар құрылысының функциялары, матрицаларды іске асырудың қосымша операциялары және т.б. кіреді. Осылардың барлығы OpenGL базалық функциясы арқылы іске асырылады.

14.1 Синтаксистің ерекшеліктері мен архитектурасы. Консольді қосымшаның құрылымы

Архитектура тұрғысынан қарасаң OpenGL графикалық жүйесі мағлұматтарды өңдеудің бірнеше кезеңдерінен тұратын ковейері болып табылады:
– қисықтар мен жазық беттер аппроксимациясы;
– төбелерді өңдеу және примитивтерді жинақтау;
– бөлшектерді өңдеу және растеризациялау;
– пиксельдер операциясы;
– текстура дайындау;
– мағлұматтарды кадр буферіне өткізу.

Жалпы алғанда, OpenGL арнаулы айнымалылардың көптеген мәндерімен жағдайы анықталатын (атаулары әдетте GL_ символдарымен басталады) және ағып жатқан нормальдің мәні текстура координаттары мен түсінің жағдайын

анықтайтын соңғы автоматпен салыстыруға болады. Осы барлық ақпараттар фигура құруға. OpenGL командасының толық аты мынадай түрде болмақ:

type glCommand_name[1 2 3 4][b s i f d ub us ui][v] (type1 arg1,…,typeN argN)

Осылайша атау бірнеше бөліктен тұрады:

gl бұл функция жазылған библиотека атауы:

OpenGL базалық функциясы үшін, GLU, GLUT, GLAUX это gl, glu, glut, aux сәйкесінше кітапханадағы функциялар.

Command_name команда аты

[1 2 3 4] команда аргументтерінің саны

[b s i f d ub us ui] аргумент түрі:

b символы GLbyte (char C\C++-ға аналог) типін білдіреді,

f символы – GLfloat (float аналогы),

i символы– GLint (int аналогы) және т.с.с.

Типтердің толық тізімі мен сипатталуын gl.h файлында көруімізге болады.

[v] наличие этого символа показывает, что в качестве

параметров функции используется указатель на массив значений

Квадратты жақшадағы кейбір символдар пайдаланылмайды. Мысалы, glVertex2i() командасы OpenGL кітапханасының базасы ретінде жазылған, екі бүтін сан параметрлері ретінде қолданылады. Ал glColor3fv() командасы үш саннан тұратын массивтерге сілтеме параметрлері ретінде қолданылады.

Консольды қосымшалар құрылымы

Соңғы кездері кең таралғын GLUT немесе GL Utility Toolkit библиотекаларының көмегімен консольды қосымшаларды тұрғызу жолын қарастырамыз. Терезелермен жұмыста бұл библиотека платформалар қатысынсыз бірегей интерфейсті құрайды. Сондықтан да Windows, Linux және тағы басқа операциялық жүйелерге төмендегі жазылып көрсетілген құрылымдар өзгеріссіз болмақ.

GLUT функциясы өзінің тағайындалуына байланысты бірнеше топтарға жіктеледі:

– инициализациялау;

– оқиғаны өңдеу бастамасы;

– терезелерді басқару;

– меню қатарын басқару;

– шақырылатын (callback) функцияларды тіркеу;

– индекстелген түстер палитрасын басқару;

– қаріптер бейнесі;

– қосымша геометриялық фигуралар бейнесі (тор, конус және т.б.);

– инициализациялау функциялар көмегімен жүргізіледі.

glutInit (int *argcp, char **argv)

argcp айнымалысы стандартты argc айнымалысына main() функциясын сипаттайтын көрсеткіш, ал argv – программаны жібергенде берілетін параметрге көрсеткіш. Бұл функция қосымша терезесін тұрғызу үшін қажетті бастапқы әрекеттерді жүргізеді, бірақ бірнеше GLUT функцияларын шақыруы мүмкін. Оларға қатысты мыналар:

– glutInitWindowPosition (int x, int y);

– glutInitWindowSize (int width, int height);

– glutInitDisplayMode (unsigned int mode).

Бастапқы екі функция қатары қосымшалар және терезе өлшемдеріне сәйкес беріледі, ал функцияның соңғысы “немесе” (“ | “) биттік операцияны пайдалану, ақпаратты әртүрлі режимде бейнелеу жолын анықтайды.

GLUT_RGBA RGBA режимі. GLUT_RGBA немесе GLUT_INDEX режимдері көрсетілмеген болса, онда үнсіз пайдаланылады.

GLUT_RGB бұл да GLUT_RGBA сияқты.

GLUT_INDEX Индекстелген түстер (палитраларды қолдану). GLUT_RGBA қолданылмайды.

GLUT_SINGLE Жалғыз буферлі терезе. Үнсіз пайдаланылады.

GLUT_DOUBLE Екі еселенген буферлі терезе. GLUT_SINGLE қолданылмайды.

GLUT_DEPTH буферлік тереңдіктегі терезе.

Бұл берілген функция параметрлерінің толық емес тізімі, бірақ көптеген жағдайларға бұл жеткілікті. Қосарлы буферді көбінесе анимация үшін қолданады, біріншіден, бір буферде сурет салып, содан соң олардың орындарын алмастырыпу арқылы жыпылықтаудың пайда болуынан құтқарады.

Тереңделген буфер немесе z-буфері көрінбейтін түзулер мен төбелерді жою үшін қолданады. GLUT кітапханалық функциясы оқиғаны басқару механизмін жүзеге асырады. Бұл дегеніміз, белгілі бір ішкі циклдің бар болу шарты, яғни бұл лайықты инициялизацияны жіберу арқылы өңдеу, бірінің артынан бірін, инициялизация уақытында барлық жарияланған оқиғаларды өңдейді. Оқиғаларға жататындар: тышқанды шерту, терезені жабу, терезе қасиеттерін өзгерту, курсорды ілгері басу, пернені басу және “бос” (idle) ештеңе болмау. Периодтық тексеруді жүргізу үшін, функцияны тіркеуден өткізу керек, ол дегеніміз өңдеуден өткізу. Бұл үшін келесі функциялар орындалады:

– 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)).

Бұларға арналған параметрлер берілген түрдің лайықты функциясы болып табылады. glutDisplayFunc() көмегімен сурет салу функциясы терезені салып қою үшін беріледі, бұл көріністі құрастыруға немесе қалпына келтіруге шақыртылады. Оны айқын нұсқау үшін, терезені жаңартып, көбінесе мына функцияны орындау жеңілге түсіреді:

– void glutPostRedisplay (void);

– glutReshapeFunc() функциясы арқылы терезе өлшемдерін өзгертуді орнатады, яғни ол үшін жаңа өлшемдерді береміз;

– glutMouseFunc() тышқанның командаларды өңдеуін анықтайды, ал glutIdleFunc() функцияны тапсыру, яғни әрбір шақырту сайын келіп тұрады, егерде оқиғалар болмаса.

Бақылау барлық ішкі циклдік функциялары жүріп жатқанда болып жатады.

– void glutMainLoop (void);

– бұл әшейінде программа соңында шақырылады, GLUT-ты атқарушылар үшін.

Анимацияны қолданатын қосымша құрылымдары төмендегідей болады:

```
#include <GL/glut.h>
void MyIdle(void){
//--Код, үзілісті ауыстырады, ол келесі кадрды анықтайды --//
....
};
void MyDisplay(void){
//--  OpenGL коды кадрды бейнелейді --//
....
//-- Сурет салудан кейін буферлерді алмастырамыз --//
glutSwapBuffers();
};
 void main(int argcp, char **argv){
….
//-- GLUT инициализациясы --//
glutInit(&argcp, argv);
glutInitWindowSize(640, 480);
glutInitWindowPosition(0, 0);
//--Терезенің ашылуы--//
glutCreateWindow("My OpenGL Application");
//-- Режим таңдау: Қосарлы буфер мен RGBA түстері --//
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
//-- Шақырылған функцияларды тіркеу --//
glutDisplayFunc(MyDisplay);
glutIdleFunc(MyIdle);
//-- Өңдеу оқиғасының жіберу механизмі --//
glutMainLoop();
};
```

Егер қолданбалар статистикалық бейнелеуді құру керек болса, онда GLUT_DOUBLE-ді GLUT_SINGLE-мен ауыстыруға болады, өйткені бір буфердің болуы жеткілікті және бұдан glutIdleFunc() функциясының шақыруын алып тастауға болады.

14.2 Open GL төбелері мен примитивтері. Төбе массивтері

Төбе астындағы нүкте кеңістікте түсіндіріледі, оның координаттарын төмендегі жолмен беруге болады:

– void glVertex[2 3 4][s i f d] (type coords);

– void glVertex[2 3 4][s i f d]v (type *coords).

Нүктенің координаталары максимум 4 мәнмен беріледі: x, y, z, w, мұнда 2 мәнмен, яғни (x,y) немесе 3 мәнмен (x,y,z) нұсқауға болады. Басқа айнымалыларды бұл жағдайда: z=0, w=1 деп қолдануға болады. Жоғарыда

айтылғандай, команда атауларының сандары берілген мәндердің санына сай болу керек, ал келесі символ-олардың типтеріне байланысты. Ось координаттары былай орналасады, нүкте (0, 0) экранның төменгі сол жақ бұрышында тұрады, ал *x* осі солға қарай бағытталады, *y* осі болса – жоғары, *z* осі – экраннан. Бұл осьтер әлемдік координата жүйесінде орналасқан, бұнда объектінің төбе координаттары беріледі, басқа координата жүйелері төменде қарастырылады. Белгілі бір фигураның қайсы бір координатасын енгізу жеткіліксіз, сондықтан барлық төбелерді біріктіру арқылы оны бір бүтін бөлікке келтіреміз.

Ол үшін OpenGL-да қарапайым түсініктер қолданылады, бұған нүкте, түзу, байланысқан немесе бекітулі түзулер, үшбұрыштар және т.б. жатады. Қарапайым жұмыстар ішкі жақша командаларында іске асады:

– void glBegin (GLenum mode);

– void glEnd (void);

– mode параметрі қарапайым түрде анықтайды, яғни ішкі жақта беріледі және келесі мәндерді қабылдайды:

GL_POINTS әрбір төбе координатасы белгілі бір нүктені береді.

GL_LINES әрбір жеке жұп төбелері кесіндіні анықтайды, егер төбенің тақ саны берілсе, онда соңғы төбе ескерілмейді.

GL_LINE_STRIP әрбір келесі төбе кесіндіні алдыңғы кесіндімен береді.

GL_LINE_LOOP алдыңғыдан өзгеше, егер соңғы кесінді соңғы немесе бірінші төбемен анықталса, онда бекітілген сынықты құрайды.

GL_TRIANGLES әрбір жекеленген үшөлшемді үшбұрыш төбелерін анықтайды, егер берілген үш төбенің саны еселі болса, онда соңғы төбе ескерілмейді.

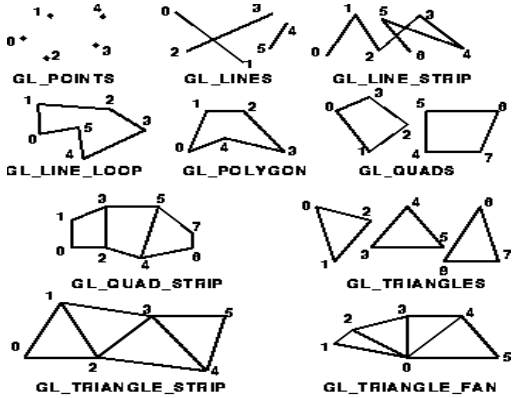
GL_TRIANGLE_STRIP әрбір келесі төбе үшбұрышты бірге алынған екі соңғымен беріледі.

GL_TRIANGLE_FAN үшбұрыштар бірінші және әрбір келесі жұп төбелерімен беріледі (жұптар қиылыспайды).

GL_QUADS әрбір жеке төртөлшемді төртбұрышты анықтайды, егер берілгені 4 төбе санынеселі болмаса, онда соңғы төбелер ескерілмейді.

GL_QUAD_STRIP төртбұрыш n – номерімен анықталады; төбелері 2n-1, 2n, 2n+2, 2n+1.

GL_POLYGON дөңес көпбұрыш төбелері жүйелі түрде беріледі.



Ағымдағы төбелер түстеріне мынадай командалар қолданады.

void glColor[3 4][b s i f] (GLtype components)

void glColor[3 4][b s i f]v (GLtype components)

Бірінші 3 параметр R, G, B құраушы түстер арқылы беріледі, ал соңғы параметр alpha-компонентін анықтап, объектінің мөлдір деңгейін береді. Егер команда атауына ‘f’ (float) типті нұсқа болса, онда барлық параметрлер [0,1] кесіндіге тиісті болу керек, бұл жағдайда alpha-компоненттері 1.0-ге болып алынады.

Егер, ‘ub’ (unsigned byte) түрі берілсе, онда параметрлер [0,255] кесіндісінде жату керек. Әр төбеге әр түрлі түстерді беруге болады. Сонда ғана төбе примитивтерінің түстер интерполяциясы жүреді. Түс интерполяциясы үшін мына команда орындалады:

void glShadeModel (GLenum mode)

GL_SMOOTH интерполяцияны қосады, ал с GL_FLAT – өшіреді.

Мысалы: әр түстес төбелі үшбұрыш салу үшін:

```
GLfloat BlueCol[3] = {0,0,1};
glBegin(GL_TRIANGLE);
glColor3f(1.0, 0.0, 0.0); // қызыл
glVertex3f(0.0, 0.0, 0.0);
glColor3ub(0,255,0); // жасыл
glVertex3f(1.0, 0.0, 0.0);
glColor3fv( BlueCol ); // көк
glVertex3f(1.0, 1.0, 0.0);
glEnd();
```

Фонның түсін беру үшін келесі команда орындалады:

void glClearColor (GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha)

[0,1] кесіндісіндіде жату тиіс және ол 0-ге тең. Бұдан кейін мына команданы шақырамыз.

void glClear (GLbitfield mask)

GL_COLOR_BUFFER_BIT параметрмен барлық буферлерінің фондық түстері орналастырылады, түстерді жазу қызметі (әшейінде буферлерге бірнеше түс қолдану қолайлы).

Тап осындай түрде алынған түстер басқа, нормальдің төбелерін анықтауға болады, ол үшін мына команданы қолданамыз:

void glNormal3[b s i f d] (type coords)

void glNormal3[b s i f d]v (type coords)

void glNormal3[b s i f d] (type coords)

void glNormal3[b s i f d]v (type coords)

Берілген бағыт (вектор) бірлік ұзындықты иемденбеуі де мүмкін, бірақ ол glEnable(GL_NORMALIZE) командасының шақырылуымен қосылатын нормалдану реттілігінде автоматты түрде нормалдана береді.

void glEnable (GLenum mode)

void glDisable (GLenum mode)

командалары сол және басқа жұмыс реттілігінің қосу және сөндіруін

тудырады. Бұл командалар көп қолданылады және олардың әсерлері нақты оқиғаларда қарастырылады.

Негізінде, glBegin() және glEnd() ішкі командалық тақтасында glVertex..(), glColor..() glNormal..(), glRect..(), glMaterial..() и glTexCoord..() кіретін бірнеше командалардың ғана шақыруын туғызуға болады.

Соңғы екі команда соңында қарастырылады, ал

void glRect[s i f d] (GLtype x1, GLtype y1, GLtype x2, GLtype y2)

void glRect[s i f d]v (GLtype *v1, GLtype *v2)

-командаларының көмегі арқылы (x1, y1) және (x2, y2)-ге қарсы бұрыштың координатасымен z=0 жазықтығында тікбұрышты немесе v1 және v2 массивтерінде координат бұрыштарымен тікбұрыштар жинағын салуға болады. Бірақта алдыменен беттік және кері жақ мағыналарын анықтап алу керек, жақтың астарында көпбұрыштың бір жағы жатады, және жасырынды түрде беттік жақ болып үстіңгі жағы сағат тіліне қарсы болатын жақ есептеледі. Беттік жақтың жоғарғысындағы тексеру бағытын

void glFrontFace (GLenum mode)

командасы арқылы өзгертуге болады, mode тең GL_CW өзгерту GL_CCW параметрлерінің мәніменен.

Көпбұрыштың бейнелеуінің нұсқасын өзгерту үшін

void glPolygonMode (GLenum face, Glenum mode) командасы қолданылады.

Mode параметрі көпбұрыштардың қалай бейнеленетінін анықтайды, ал face параметрі болса осы команда қолданатын көпбұрыштың түрін орнықтырады және келесі мәндерді қабылдай алады:

Беттік жақ үшін GL_FRONT

Кері (жақ) бет үшін GL_BACK

Барлық жақтар үшін GL_FRONT_AND_BACK

mode параметрі тең бола алады:

GL_POINT мұндай режимде көпбұрыш кесінді жинағы түрінде бейнеленеді.

GL_LINE мұндай режимде көпбұрыш кесінді жинағы түрінде бейнеленеді.

GL_FILL мұндай режимде көпбұрыштар ағымдағы жарықтандыру түсімен боялады және бұл режим дыбыссыз қондырылған.

Сонымен қатар, экраннан жақтың қандай түрін бейнеленетінін көрсетуге болады. Бұл үшін алдыменен glEnable (GL_CULL_FACE) командасының шақыруына сәйкес тәртіпті қондыру қажет, ал содан кейін

void glCullFace (GLenum mode) командасының көмегімен көрініп тұрған жақтың түрін таңдау керек.

GL_FRONT параметрімен шақырудың барлық беттік жақтың бейнелеулерден алынуына әкеліп соғады, ал GL_BACK параметрімен болса-керіліктерді (дыбыссыз қондырылған) қарастырылған стандартты примитивтерден басқа GLU және GLUT кітапханаларында сфера, цилиндр, диск және сфера, куб, конус, тор, тетраэдр, додекаэдр, икосаэдр, октаэдр және шәйнек (в GLUT) сияқты қиынырақ фигуралар бейнеленеді. Текстураның автоматты қойылымы тек GLU кітапханаларының фигурасына қарастырылған. Мысалы, сфераны немесе цилиндрді салу үшін алдыменен GLUquadricObj*

gluNewQuadric(void)

Командасының көмегімен GLUquadricObj арнайы типті объект салып алуы қажет Ал содан кейін:

void gluSphere (GLUquadricObj * qobj, GLdouble radius, GLint slices, GLint stacks)

void gluCylinder (GLUquadricObj * qobj,GLdouble baseRadius,GLdouble topRadius, GLdouble height, GLint slices, GLint stacks)

командаларын шақыру керек, мұндағы slices параметрі z осінің айналасында соғу санын, ал z осі бойымен – stacks береді.

Осы және басқа командалардағы примитивті салу жайындағы көбірек мәліметтерді қосымшадан табуға болады. Аталған корректрлік примитивтерді салуда көрінбейтін сызықтарды және беттерді алып тастау керек екенін, glEnable(GL_DEPTH_TEST) командасының шақырылуымен болған режимді қосу керек екенін айтып өткен жөн.

Төбелердің жиыны

Егер де төбелер көп болса әр команда үшін glVertex..()-ті шақырып жатпау үшін төбелерді

void glVertexPointer (GLint size, GLenum type, GLsizei stride, void *ptr)

командасын қолданып массивке біріктірген аңғайлы (сақталу әдісін және төбелердің координаталарын анықтайтын). Бұл кезде size төбелердің координат санын анықтайды, type берілген типтерді анықтайды. Кей кезде бір массивте төбелердің басқа атрибуттарын сақтаған қолайлы, ол кезде stride параметрі бір төбенің координатасынан келесі төбенің координатасына шейін ығысуды береді. Егер де stride нольге тең болса, ендеше координаттар тізбектей орнатылған ptr параметрінде берілгендер болатын адрес көрсетіледі. Нормаль массивін, түсті және басқа төбелердің атрибуттарын

void NormalPointer (GLenum type, GLsizei stride, void *pointer)

void ColorPointer (GLint size, GLenum type, GLsizei stride, void *pointer)

командасын қолдану арқылы аналогты түрде анықтауға болады.

Бұл массивтерді келесіде де қолдана беру үшін келесі командаларды шақырамыз.

void glEnableClientState (GLenum array)

GL_VERTEX_ARRAY, GL_NORMAL_ARRAY, GL_COLOR_ARRAY параметрлерімен сәйкес. Массивпен жұмыстың соңында void glDisableClientState (GLenum array) командасын шақырған жөн.

array параметрі мәніне сәйкес.

Массивтер мазмұнын көрсету үшін мынадай команда қолданылады

void glArrayElement (GLint index)

Массивтің көрінісі үшін index массив элементін қолданатын OpenGL төбе атрибуттарын беретін void glArrayElement (GLint index) командасы қолданылады. Бұл сәйкес параметрлермен glColor..(…), glNormal..(…), glVertex..(…) түрлерінің тізбектелген командаларында қолдану аналогты түрде.

Бірақта оның орнына көбінесе void glDrawArrays (GLenum mode, GLint first, GLsizei count) командасы шақыртылады. Бұл сәйкес индекстерімен glArrayElement() командасының шақырылуына эквивалентті. Егер де бір төбе

бірнеше примитивтерге кірсе, массивте оның координаттарын аударудың орнына оның индексін қолданған ыңғайлы

Ол үшін void glDrawArrays (GLenum mode, GLsizei count, GLenum type, void *indices) командасын шақыру керек

мұндағы indices – примитивтерді салу үшін қолданылатын төбе нөмірінің массиві. Type - GL_UNSIGNED_BYTE, GL_UNSIGNED_SHORT, GL_UNSIGNED_INT массив элементінің түрін анықтайды, ал count олардың сандық мәнін береді.

Кескіндердің тізімі

Егер де бір команда топтарына бірнеше рет бағытталу керек болса, бұл командалары кескіндердің тізіміне (display list) жинастыруға болады және оны қажет уақытында шақырып тұрасың. Кескіндердің тізімін жаңадан құру үшін барлық командаларды ауыстыру керек, бұған:

void glNewList (GLuint list, GLenum mode)

void glEndList() команда аралық жақшалар арасымен кіру керек. Тізімдерді ажырату үшін list параметрінің мәнімен құрылған тізімді құруда берілген оң сандардың мәндері қолданылады, ал mode параметрі команданың өңдеу режимін анықтайды, бұл командалар мына тізімге кіруі қажет:

GL_COMPILE командасы тізімге орындалмастан жазылады

GL_COMPILE_AND_EXECUTE командалары бірінші орындалады, содан кейін барып тізімге жазылады. Тізім құрылған соң, параметрде қажетті тізімнің идентификаторын көрсетіп void glCallList (GLuint list) командасы арқылы шақыртуға болады

Бірнеше тізімді бірден шақырту үшін void glCallLists (GLsizei n, GLenum type, const GLvoid *lists) командасын қолдануға болады, элементтер типі type параметрінде көрсетіледі. Бұлар GL_BYTE, GL_UNSIGNED_BYTE, GL_SHORT, GL_INT, GL_UNSIGNED_INT типі және басқалары болуы мүмкін. Тізімді алып тастау үшін list <= ID <= list+range-1. диапозонынан тізімді ID идентификаторымен алып тастайтын

void glDeleteLists (GLint list, GLsizei range) командасы қолданылады.

Негізгі әдебиет: 6 [11-47]
Қосымша әдебиеттер: 10[284-295]
Бақылау сұрақтары:
1. OpenGL қандай функцияларды орындайды?
2. Жұмыс істеу үшін қандай қосымша кітапханалар бар?
3. GLAUX және GLUT не үшін қажет?
4. GLUT қандай функцияларды орындайды?
5. OpenGL синтаксисі мен құрылымы қандай?

Дәріс 15. Open GL-де проекция мен координаттарды түрлендіру

OpenGL-да негізгі 3 координат жүйесі болып сол жақтық, оң жақтық және терезелік сияқтылар есептеледі. Бірінші екі жүйе үшөлшемді болып табылады және бір-біріне z осінің бағыты бойынша ажыратылады: оң жақтыққа ол бақылаушыға, ал сол жақтыққа экран түбіне бағытталған. x және y осьтерінің

орналасуы аналогты түрде жоғарыда сипатталған. Тапсырыс үшін сол жақтық жүйе gluPerspective(), glOrtho() командасының параметрінің мәнімен қолданылады, бұл төменде қарастырылады, ал оң жақтық немесе әлемдік координат жүйесі басқа жағдайда қарастырылады. Үш өлшемді ақпараттың көрінісі екі өлшемді терезелік координат жүйесінде жүреді. OpenGL-дағы сахна объектісінің әртүрлі түрленуі тапсырыс үшін матрицаның үстіндегі операцияны қолданады, осыдан матрицаның 3 түрін ажыратып алуға болады: көріністік, проекциялық және текстуралық. Олардың барлығы 4x4 өлшемін иемденеді. Көріністік матрица дүниежүзілік координатадағы параллельдік тасымалдау, масштабтың өзгеруі және бұрылыс сияқты объектінің түрленуін анықтайды. Проекциялық матрица экран жазықтығында үш өлшемді объектінің қалай проекцияланатынын береді, ал текстуралық матрица текстураның объектіге қабаттасуын анықтайды, Қай матрицаны өзгерту керек екенін таңдау үшін

void glMatrixMode (GLenum mode) командасын қолданамыз.

Mode параметрінің мәніне тең GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE шақыру көріністік жұмыс режимінде қосады. Матрица беретін сол немесе басқа типті команданы шақыру үшін алдыменен сәйкес режимді орналастырамыз. Қазіргі матрицаның элементін анықтау үшін

void glLoadMatrix[f d] (GLtype *m) командасын шақырамыз.

Мұндағы m float немесе double 16 элемент түрінен тұратын массивті команданың атауына сәйкес нұсқайды, осы кезде алдыменен онда матрицаның бірінші бағаны, содан кейін 2-ші, 3-ші, 4-ші бағаны жазылу керек.

void glLoadIdentity (void) командасы

қазіргі матрицаны бірлікпен ауыстырады. Болашақта қолдану үшін сақтап отыру матрицаның құрамын жиі сақтап отыру керек. Ол үшін

void glPushMatrix (void)

void glPopMatrix (void) командасын қолданамыз.

Олар қазіргі матрицаны стекпен жазады және қалпына келтіреді. Көріністік матрица үшін оның тереңдігі минимум 32, ал қалған екі түрі үшін ол минимум 2-ге тең. Осы матрицаны сол жақтағы басқа матрицаға көбейту үшін:

void glMultMatrix[f d] (GLtype *m) командасы орындалады.

m – тұрған жерде матрицаның өлшемі берілген мағлұматтардың орналасуы бойынша массив түрі 4x4 болуы қажет. Бірақ, матрицаның сол және де басқа түрлерін өзгерту үшін арнайы командалары пайдаланған ыңғайлы. Ол командалар өз параметрлерінің мәндері бойынша қажетті матрицаны туғызады және оны қолданыстағы матрицамен көбейтеді. Қолданыстағы туғызылған матрицаны жасау үшін шақырар алдында glLoadIdentity() командасын шақыру керек.

15.1 Түр түрлендірулері

Көріністік түрлендірулерге тасымал, бұрылыс және координат осьтеріндегі орындау үшін сәйкес матрицаның әрбір объектісінің өзгерген координаттарын алғанның өзі жеткілікті:

$$(x', y', z', 1)^T = M * (x, y, z, 1)^T$$

M – көріністік түрлендіру матрицасы. Болашақ түрлендіру мен жобалау ұқсас болады. Матрицаның өзі келесі командалар арқылы пайда болады:

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..() өзінің параметрлік мәнін биіктігінің координаттарына қоса отырып, объектінің тасымалын тудырады.

glRotate..() (x,y,z) векторының айналасында angle бұрышына (градуспен өлшенеді) сағат тіліне қарама-қарсы бақытта объектінің бұрылысын тудырады.

glScale..() өз параметрлерінің мәні бойынша биіктіктің сәйкес координаттарын көбейте отырып, объектінің масштабталуын тудырады (сығылу және созылу). Осы түрлендірулердің барлығы төмендегі программаларда келтірілген примитивтарда қолданылады. Мысалы, сахнаның бір объектісін айналдыру үшін, ал басқасын қозғамай орнында қалдыру үшін, әуелі қолданыстағы көріністік матрицаны сақтаған ыңғайлы. Содан соң керек параметрлерімен glPushMatrix() командасының стегіне қолданыстағы түрлік матрицаны сақтаған ыңғайлы. Содан соң керек параметрлерімен glPopMatrix() командасын оырндап, осы объект тұратын примитивтарды келтіру керек. Осыдан кейін қолданыстағы матрицаны. glPopMatrix() командасы арқылы қалпына келтіру керек. Бірақ кейбір кезде объектінің өзінің жағдайын өзгерту үшін бақылау нүктесінің жағдайын өзгерту керек болады, сонымен қатар түрлік матрицаны да өзгертуге тура келеді. Оны мына командалардың көмегімен іске асыруға болады: void gluLookAt (GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz) (eyex,eyey,eyez) нүктелері бақылау нүктесін анықтайды, (centerx, centery, centerz) қорытындылау аймағының орталығында жобаланатын сахна ортасын көрсетеді, ал (upx,upy,upz) векторлары камера бұрылысын анықтай отырып, у осінің нақты бағытын көрсетеді. Егер, мысалы, камераны бұрмалауға болса, онда (0,1,0) мәндері көрсетіледі, ал (0,-1,0) мәндерімен сахна бұрылып тұрады. Нақты түрде бұл команда сахна объектілерінің тасымалы мен бұрылысын жүзеге асырады, бірақ бұл түрде параметрлерді белгілеу ыңғайлы.

15.2 Проекциялар. Шығыс аймағы

OpenGL командасында ортографиялық (параллельді) және перспективалық жобалар бар. Жобалаудың бірінші түрі мына командалармен беріледі:

void glOrtho (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far)

void gluOrtho2D (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top)

Бірінші команда көріністің қиық көлемінде (параллелограмм көрінісі) сол жақ координаталар жүйесінде проекция матрицасын түзеді. Команда параметрлерін нүктемен береді. (left, bottom, -near) және (right, top, -near) олар шығу терезесінің сол төмен және оң жоғары бұрыштарына байланысты near және far параметрлері жазықтық қиығының жақын және алыс арақашықтығын (0,0,0) нүктелерінен бастап береді және теріс болуы мүмкін. 2-ші командада 1-

шіден өзгеше near және far мәндері –1 және 1 сәйкес тең деп алынады.

Перспективті жобалау командасымен анықталады:

void gluPerspective (GLdouble angley, GLdouble aspect, GLdouble znear, GLdouble zfar)

бұл координаттар жүйесінің сол жағында қиық конус көрінісін береді. Angley параметрі көріністің y осі бойынша бұрышты градуспен анықтайды және 0-ден 180-ге дейінгі диапозонда жатуы қажет. *x* осі бойынша көрініс бұрышы шығару аймақтар жағының қатнасы түрінде беріледі. zfar және znear параметрлері бақылаушыдан жазықтықтар қиығы түбіне дейінгі арақашықтықты береді және қатынасы көп болса, соғұрлым тереңдік буферіне жақын орналасқан жазықтықтар ерекшеленеді, нашар жүреді. Бұл "сығылған" 0-1 диапозонында тереңдік жазылуына байланысты (келесі пункті қара).

Шығару аймағы

Жобалау матрицасын қолданғаннан кейін келесі түрлендіру кірісіне қиық (clip) координаттар беріледі, олар үшін барлық компоненттер $(x_c, y_c, z_c, w_c)^T$ мәндері [-1, 1] кесіндісінде жатады:

$(x_n, y_n, z_n)^T = (x_c/w_c , y_c/w_c, z_c/w_c)^T$

Шығару аймағы терезені координаттар жүйесінде тікбұрыш түрінде болады, оның өлшемдері командасымен беріледі:

void glViewPort (GLint x, GLint y, GLint width, GLint height)

Барлық параметрлер мәндері пиксельде беріледі және терезелер координаттар жүйесінде сол төменгі бұрыш (*x*, *y*) координаттарымен бірге шығару аймағында кеңдік пен биіктікті анықтайды. Терезелер координаттар жүйелерінің өлшемдері алдыңғы қосымша терезе өлшемдерімен анықталады, (0,0) нүктесі терезенің сол төменгі бұрышында жатады. glViewPort() командасының параметрлерін қолдана отырып, шығару аймағының орталық координаттары мына формулалар бойынша есептеледі: (o_x, o_y) формула $o_x=x$+width/2, $o_y=y$+height/2.

p_x=width, p_y=height болсын, онда әр төбенің терезелі координаттарын табуға болады: $(x_w, y_w, z_w)^T = ((p_x/2) x_n + o_x , (p_y/2) y_n + o_y , [(f-n)/2] z_n + (n+f)/2)^T$

Бұл жағдайда бүтін оң *n* және *f* шамалары минимал және максимал нүктенің терезедегі тереңдігін береді және солайша 0 және 1-ге тең. Әр нүкте тереңдігі арнайы тереңдік буферіне (z-буфер) жазылады, ол көрінбейтін сызықтарды және жазықтықтарды жоюға қолданылады. *n* және *f* мәндерін функция шақыру арқылы орнықтыруға болады.

void glDepthRange (GLclampd *n*, GLclampd *f*)

Негізгі әдебиеттер: 6 [50-149]

Қосымша әдебиеттер: 10 [284-295]

Бақылау сұрақтары:

1. Екіөлшемді құрылым қалай жасалады?
2. glScissor командасы қандай функцияны орындайды?
3. Қателер қалай өңделеді?
4. Масштабтау қалай жүреді?
5. OpenGL бұру мен айналуы қалай болады?

2.3 Зертханалық сабақтардың жоспары

Зертханалық жұмыс 1. CorelDRAW-да қарапайым логотиптерін құру

Мақсаты: Векторлық бейнелермен жұмыс ерекшеліктерін үйрену

Тапсырма: CorelDraw редакторында векторлық логотиптерді құру

Негізгі әдебиет: 5 [бет.592-610]

Қосымша әдебиет: 13 [бет.39-118]

Бақылау сұрақтары:

1. Тікбұрышты дөңгелектеу жолын көрсетіңіз?
2. Эллипс нүктелерін редактілеу қалай жүзеге асады?
3. Объектіні бөлу жолы қалай жүзеге асады?

Зертханалық жұмыс 2. CorelDRAW-да үшөлшемді объект құру

Мақсаты: Мәтінді пайдалана векторлық бейнелермен жұмыс ерекшеліктерін үйрену

Тапсырма: Фигуралық мәтін құру. Фигуралық мәтіндердің жеке символдарын трансформациялау

Негізгі әдебиет: 5 [бет.611-630]

Қосымша әдебиет: 13 [бет.409-430]

Бақылау сұрақтары:

1. Векторлық графикада мәтіндер қалай құралады?
2. Мәтіндердің қандай түрлері бар?
3. Қарапайым мәтін деп нені айтамыз?

Зертханалық жұмыс 3. Adobe Photoshop-та бөлініп шығу жабдығымен жұмыс

Мақсаты: Adobe Photoshop графикалық редакторын оқып үйрену

Тапсырма: Ерекшелеу құралдарымен жұмыс жасау

Негізгі әдебиет: 5 [бет.344-382]

Қосымша әдебиет: 12 [бет.4-49]

Бақылау сұрақтары:

1. Қандай ерекшелейтін құралдар бар?
2. Маска деген не?
3. Маска не үшін қолданылады?

Зертханалық жұмыс 4. Adobe Photoshop мәтінімен жұмыс

Мақсаты: Сүзгіні (фильтр) құруды және пайдалануды үйрену

Тапсырма: Маскамен жұмыс. Қабаттармен жұмыс. Көп қабатты бейнелермен жұмыс

Негізгі әдебиет: 5 [бет.383-391]

Қосымша әдебиет: 12 [бет.51-58]

Бақылау сұрақтары:

1. Сүзгі (фильтр) деген не?
2. Фильтр не үшін қолданылады?
3. Сүзгінің қандай түрлерін бар?

Зертханалық жұмыс 5. Жазықтықта берілген координаттар бойынша объектілерді түрлендіу

Мақсаты: Координаталар түрлендірулерін, жазықтықта аффинді координалар түрлендірулерін үйрену

Тапсырма: Координаттарды түрлендіру. Жазықтықта координаттар жүйесін түрлендіру

Негізгі әдебиеттер: 4 [бет.61-68]

Қосымша әдебиеттер: 9 [бет.68-75]

Бақылау сұрақтары:

1. Полярлы жүйеден тікбұрышты жүйеге түрлендіруді қалай түсінесіз?
2. Кері түрлендіру функциясы деген не?
3. Сызықтық түрлендіру деген не?

Зертханалық жұмыс 6. Берілген координаттар бойынша объектілерді түрлендіу

Мақсаты: Координат түрлендірулері және жазықтықта аффинді объектілер түрлендірулерін оқып үйрену

Тапсырма: Берілген координаттар бойынша жүйелерді түрлендіру

Негізгі әдебиет: 4 [бет.69-71]

Қосымша әдебиеттер: 9 [бет.77-85]

Бақылау сұрақтары:

1. Жазықтық бетінде объектіні жылжыту қалай жүзеге асады?
2. Объектілердің аффиндік түрлендіруі деген не?
3. Аффинді түрлендірулердің қандай қасиеттері бар?

Зертханалық жұмыс 7. OpenGl программасын қолданып графикалық программалар құру

Мақсаты: OpenGl программасын қолданып графикалық программалар құру

Тапсырма: OpenGl-де қарапайым объектілерді құру

Негізгі әдебиет: 6 [бет.11-85]

Қосымша әдебиет: 8 [бет.85-90]

Бақылау сұрақтары:

1. Сызық құру үшін қандай командаларды пайдаланамыз?
2. Квадрат құру командаларын талдаңыз?
3. Көпбұрыштар қалай құрылады?

2.4 Оқытушының жетекшілігімен орындалатын студенттердің өзіндік жұмыстары бойынша өткізілетін сабақтардың жоспары (СОӨЖ)

№	Тапсырма	Өткізу түрлері	Методикалық ұсынулар	Ұсынылатын әдебиеттер
1	Интерактивті графикалық жүйе классификациясы	Презентация	Интерактивті графикалық жүйенің жіктелуі мен қолдану аймағы	4 [бет.7-9]
2	Компьютерлік графиканың аппарат-тық қамтамасы	Презентация	Компьютерлік графика аппараттық қамтамасының жалпылама талаптарын жазыңыз	5 [бет.31-90]
3	Векторлық графика	Презентация	CorelDRAW көмегімен қарапайым объектілерді құрыңыз. Фигурный және қарапайым мәтіндерді пайдаланып жарнамалық логотип құрыңыз	5 [бет.501-537]
4	Растрлық графика	Презентация	Adobe Photoshop көмегімен фотомонтаж жасаңыз. Әр түрлі суреттерден бір ғана сурет құраңыз. Сурет түстерін түзету	5 [бет.253-292]
5	Қадамдықты жою әдістері	Презентация	Қадамдықты жою әдістері алгоритмін оқып, DELPHI және C++ программалау тілдерінде программа жазыңыз	2 [бет.119-127]
6	Геометриялық түрлендірулер	Презентация	Берілген нүктелер бойынша координаттар геометриялық түрлендірулеріне DELPHI немесе C++ программалау тілінде үлестіру програм-масын жазыңыз	4 [бет.63-69]
7	Объектілерді түрлендіру	Презентация	DELPHI немесе C++ программалау тілінде объектілерді берілген коор-динаталар бойынша түрлен-діру программасын жазу	4 [бет.71-94]
8	Компьютердің графикадағы түсі	Презентация	DELPHI және C++ программалау тілінде түстік гамма құру программасын жазыңыз	2 [бет.121-168]
9	Жазықтықтағы фигуралардың кесілуі Коэн-Сазерленд кесу алгоритмі	Презентация	DELPHI немесе C++ программалау тілінде фигу-ралардың кесілу алгоритмі программасын жазыңыз	2 [бет.153-158]
10	Кирус-Бек алгоритмі	Презентация	DELPHI немесе C++ программалау тілінде Кирус-Бек алгоритмін үлестіру программасын жазыңыз	2 [бет.166-170]

11	Көрінбейтін сызықтарды жою	Презентация	DELPHI немесе C++ программалау тілінде жазық алаңда көрінбейтін сызық-тарды жою программасын жазыңыз	2 [бет.230-233]
12	Көрінбейтін сызықтарды жою	Презентация	DELPHI немесе C++ программалау тілінде Робертс алгоритмін үлестіру программасын жазыңыз	2 [бет.233-250]
13	Шын кескіндеулерді құру	Презентация	DELPHI немесе C++ программалау тілінде шын кескіндерді құрастыру программасын құрыңыз	2 [бет.377-380]
14	Гуро әдісімен бояу, Фонго әдісімен бояу	Презентация	DELPHI немесе C++ программалау тілінде Гуро және Фонго әдісімен бояу алгоритмін үлестіру программасын жазыңыз	2 [бет.391-394]
15	Сәулелердің трассировкасы жолымен көрінетін жазықтық алгоритмін анықтау	Презентация	DELPHI немесе C++ программалау тілінде сәулелердің трассировкасы жолымен көрінетін жазық-тық алгоритмін анықтау алгоритмін үлестіру прог-раммасын жазыңыз	2 [бет.355-360]

2.5 Студенттердің өзіндік жұмыстары бойынша өткізілетін сабақтардың жоспары (СӨЖ)

№	Тапсырма	Әдістемелік ұсынулар	Әдебиеттер
1	Векторлық графика	CorelDRAW мүмкіндіктерін қолданып қатпарларды өңдеңіз. Мәтіндік және графикалық стиль құрыңыз	5 [бет.501-537]
2	CorelDRAW-да үш өлшемді объекті құру	CorelDRAW мүмкіндіктерін қолданып объектілер арасында ағым жүргізу. Интерактивті искажение қолданып көлемді объект құру	5 [бет.253-292]
3	CorelDRAW-да мәтіндермен жұмыс істеу	CorelDRAW мәтіндік мүмкіндіктерін қолданып мәтінді қиысық бұрышқа орналастыру. Мәтін символдарын орналасуын өзгерту. Жай мәтіннің рамкасы мен формасын өзгерту	2 [бет.119-127]
4	Adobe Photoshop танысу	Adobe Photoshop мүмкіндігін қолданып қарапайым, күрделі және автоматты белгілеулерді құру. Маскіні қолданып белгілеуді редакциялау	5 [бет.344-424]
5	Қабаттармен жұмыс жасау	Adobe Photoshop мүмкіндігін қолданып бірнеше қабық және фотомонтаж жасау. Әрбір сурет жеке қабықта сақталу керек	5 [бет.409-424]
6	Координат түрлендіруі	DELPHI немесе C++ тілінде берілген нүктеде координатаның параллельді	4 [бет.63-69]

		жылжыту, созу-қысуын	
7	Жазық алаңда берілген координаттар бойынша объектілердің түрлендірілуі	Берілген деректерді пайдалана DELPHI немесе C++ тілінде X, У, Z осьтері бойынша объектілер айналымын жүргізу	4 [бет.71-94]
8	OpenGL-мен танысу.	OpenGL мүмкіндіктерін пайдалана қарапайым объектілер құрыңыз. (Квадрат, шеңбер, эллипс және т.б.)	6 [бет.11-47]
9	OpenGl көмегімен графикалық програм-маны өңдеу (Сатылы әдісті алып тастау)	OpenGL мүмкіндігін пайдалана DELPHI-де сатылы әдісті алып тастау программасын жазыңыз	2 [бет.153-158] 6 [бет.11-85]
10	OpenGl көмегімен графикалық програм-маны өңдеу (Полутон аппроксимациясы)	OpenGL мүмкіндігін пайдалана DELPHI-де полутон аппроксимациясы алгоритмінің программасын жазыңыз	2 [бет.166-170] 6 [бет.11-149]
11	OpenGl көмегімен графикалық програм-маны өңдеу (Коэн-Сазерленд қию (қиып тастау) алгоритмі)	OpenGL мүмкіндіктерін пайдалана отырып DELPHI-де Коэн-Сазерленд қиып тастау алгоритмінің үлестіру программасын жазыңыз	2 [бет.230-233] 6 [бет.11-250]
12	OpenGl-ді қолданып графикалық програм-маларды өңдеу (Кирус-Бек алгоритмі)	OpenGL мүмкіндіктерін пайдаланып DELPHI-де Кирус-Бек үлестіру программасын жазыңыз	2 [бет.233-250] 6 [бет.11-293]
13	OpenGl көмегімен графикалық програм-маны өңдеу (Қалқу (плавающий) көкжиегінің алгоритмі)	OpenGL мүмкіндігін пайдалана отырып DELPHI-де қалқу көкжиегінің алгоритмінің үлестірілу программасын жазыңыз	2 [бет.377-380] 6 [бет.11-293]
14	OpenGl қолданылуымен графикалық програм-маларды өңдеу (Гуро әдісі)	OpenGL мүмкіншіліктерін пайдалана отырып DELPHI-де Гуро әдісінің үлестірілу программасын жазу	2 [бет.391-394] 6 [бет.11-293]
15	OpenGl қолданып графикалық програм-маларды өңдеу (Фонго әдісі)	OpenGL мүмкіншіліктерін пайдалана отырып DELPHI-де Фонго әдісінің үлестірілу программасын жазу	2 [бет.355-360] 6 [бет.11-293]

2.7 Өзін-өзі тексеру үшін дұрыс жауап кілттері көрсетілген тестік тапсырмалар

1. Түстерді көзбен көру үшін үш нәрсе қажет:
A) жарық көзі, объект, күн;
B) жарық көзі, объект, көз;
C) жарық, күн, шағылысуды қабылдау.

2. Шағылысқан түс дегеніміз не?
A) жарық, бояу көзі;
B) объектінің табиғи фоны;

С) жарық, активті жіберілетін жарық.

3. Түстердің сандық сипаттамасы деп не айтылады?
А) түс қоюлығы;
В) түстер ашықтығы;
С) жарықтылық.

4. Қызыл түспен жарықтанған (освещенная) жасыл қағаздың түсі қалай көрінеді?
А) көк;
В) сары;
С) қызыл.

5. Аддитивті түстік модельдерді анықтаңыз?
А) CMYK;
В) HSB;
С) RGB.

6. Субтрактивті түстік модельдерді көрсетіңіз?
А) CMYK;
В) HLS;
С) RGB.

7. Перцепционды түстік модельдерді көрсетіңіз?
А) CMY, CMYK;
В) HSB, HLS;
С) RGB.

8. Әмбебап өлшем бірлік болып есептеледі?
А) dpi;
В) spi;
С) ppi.

9. Қандай өлшем бірлік баспаға жіберуде қолданылады?
А) lpi;
В) dpi;
С) ppi.

10. Гистограмманың қызметі не?
А) кескінді баптау мен сапасын бағалау;
В) түсін анықтау;
С) түсінің деңгейін анықтау.

11. Қандай топтағы форматтар кескінді растрлық түрде сақтайды ?
А) BMP, TIFF, PCX, PSD, JPEG, PNG, GIF;

B) BMP, TIFF, PCX, PSD, WMF, PNG, GIF;
C) BMP, TIFF, PCX, PSD, JPEG, PNG, FH9, WMF.

12. Қандай топтағы форматтар кескінді векторлық түрде сақтайды ?
A) BMP, PCX;
B) WMF;
C) FH9, WMF.

13. Қандай топтағы форматтар әмбебап, векторлық және растрлық түрде сақтай алады?
A) BMP, TIFF, PCX, PSD, JPEG, PNG, GIF;
B) EPS, PICT,CDR, AI, FH9, FLA;
C) BMP, TIFF, PCX, PSD, JPEG, PNG, FH9, WMF.

14. Сызықтық түрлендірулер $n=N$ болғанда аталады:
A) координаталық;
B) матрицалық;
C) аффиндік.

15. Түрлендіру функцияларының түрлері бөлінеді:
A) сызықтық түрлендіру;
B) сызықтық емес түрлендіру;
C) сызықтық және сызықтық емес түрлендіру.

16. Келтірілген теңдеу көрсетеді $\begin{cases} X = Ax + By + C \\ Y = Dx + Ey + F \end{cases}$
A) жазықтықта аффиндік координат түрлендіруін;
B) аффиндік түрлендіруді;
C) үшөлшемді координат түрлендіруін.

17. Координаттың кері түрлендіру теңдеуін көрсетіңіз
A) $\begin{cases} X = A'x + B'y + C' \\ Y = Dx + Ey + F \end{cases}$;
B) $\begin{cases} X = Ax + By + C \\ Y = Dx + Ey + F \end{cases}$;
C) $\begin{cases} X = A'x + B'y + C' \\ Y = D'x + E'y + F' \end{cases}$.

18. Мына теңдеу координаттың қандай қимылын көрсетеді $\begin{cases} X = x - dx \\ Y = y - dy \end{cases}$
A) созылу;

B) параллельді жылжу;
C) қысу (сығу).

19. Мына теңдеу координаттың қандай қимылын көрсетеді $\begin{cases} X = \dfrac{x}{k_x} \\ Y = \dfrac{y}{k_y} \end{cases}$

A) созылу;
B) параллельді жылжу;
C) қысу (сығу).

20. Мына теңдеу координаттың қандай қимылын көрсетеді $\begin{cases} x = Xk_x \\ y = Yk_y \end{cases}$

A) созылу;
B) параллельді жылжу;
C) қысу (сығу).

21. Мына теңдеу координаттың қандай қимылын көрсетеді $\begin{cases} X = x + dx \\ Y = y + dy \\ Z = z + dz \end{cases}$

A) созу-қысу;
B) параллельді жылжу;
C) dx, dy, dz -на жылжу.

22. Мына теңдеу нені сипаттайды?
A) Безье қисығының параметрлік пішінін;
B) Безье сызықтық теңдеуін;
C) эллипс теңдеуін.

23. Текстура деген не?
A) бояу стилі;
B) бояудың дайын форматы;
C) бояу материалы.

24. Көрінбейтін сызықты жою әдісі қолданылады:
A) көрінбейтін сызықтарды жою;
B) көлеңкеленген сызық жасау;
C) көлеңкеленген сызықты бояу.

25. Гуро әдісі деген не?
A) жылтыратуды суреттеу тәсілі;
B) үшөлшемді объектілердің жиектерін бояу тәсілі;
C) үшөлшемді объектілерді жылтырату тәсілі.

26. Муар нені сипаттайды?
A) өрнек;
B) эффект, өрнек;
C) бейнеэффект, өрнек.

27. Линиатура деген не?
A) ұзындық бірлігіне берілетін сызықтар саны;
B) ұзындық бірлігіне берілетін нүктелер саны;
C) ұзындық бірлігіне берілетін объектілер саны.

28. Воксел деген не?
A) көлемді растр элементі;
B) растр;
C) қапас (клетка).

29. Мына теңдеудің қайсысы жазықтықта координаттың қысылуын (сығу) көрсетеді ?

A) $\begin{cases} X = x - dx \\ Y = y - dy \end{cases}$;

B) $\begin{cases} x = Xk_x \\ y = Yk_y \end{cases}$;

C) $\begin{cases} X = x - dx \\ Z = y - dy \end{cases}$.

30. Қайсы теңдеу координаттың жазықтықта параллельді жылжуын көрсетеді?

A) $\begin{cases} X = x - dx \\ Y = y - dy \end{cases}$;

B) $\begin{cases} x = Xk_x \\ y = Yk_y \end{cases}$;

C) $\begin{cases} X = x + dx \\ Y = y + dy \\ Z = z + dz \end{cases}$.

«Интерактивті графикалық жүйелер» пәнінен
тест сұрақтарының дұрыс жауаптары

Сұрақ нөмері	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Дұрыс жауабы (A, B, C, D)	C	C	B	A	C	A	B	C	A	A	A	B	B	C	C
Сұрақ нөмері	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Дұрыс жауабы (A, B, C, D)	A	C	B	A	C	C	A	A	A	B	C	A	A	B	A

2.8 Курс бойынша емтихан сұрақтары

1. ИГЖ негізгі мақсаты
2. Интерактивті графикалық жүйені қолдану классификациясы
3. ИГЖ-ң қысқаша даму тарихы
4. Компьютердің құрылымы мен тағайындалуы
5. Монитордың негізгі сипаттамасы
6. Графикалық кескіндерді басып шығару
7. Графикалық жұмыс станциялары
8. Графикалық файлдардың форматтары
9. Векторлық графикаға жалпы көзқарас
10. Объектілер және олардың атрибуттары
11. Векторлық графика артықшылықтары мен кемшіліктері
12. Векторлық графиканың математикалық негіздері
13. Безье қисығы
14. Векторлық графикалық форматтар
15. Растрлық графика алгоритмдері
16. Квадратты растрда сызықтар тұрғызу
17. Сызық салу параметрлік алгоритмі
18. Сызық салу Брезенхем алгоритмі
19. Шеңбер құру алгоритмі
20. Шеңберді генерациялауға Брезенхем алгоритмі
21. Растрлық өрбіту – кескінді генерациялау тәсілі
22. Жазық алаңның растрлық өрбітуі
23. Нақты уақыттағы растрлық жазба
24. Полутон әдісі
25. Топтық кодтау
26. Тасымалдау әдісі
27. Координаттар мен түрлендірулер
28. Кадр буферлері
29. Клеткалы кодтау
30. Координат түрлендірулері
31. Растр бағыттары
32. Жазықтықтағы аффиндік түрлендірулер
33. Біртекті жүйе координаттар түрлендіруі

34 Көпбұрыштардың толтырылуы

35. Жазықтықта объектілердің аффиндік түрлендіруі
36. Екіөлшемді аффиндік объектілердің түрлендірілуі
37. Қарапайым алгоритмді толтыру
38. Объектілердің үшөлшемді аффиндік түрлендірілуі
39. Баспалдықты жоюдың әдістерінің негіздері
40. Қатар бойынша алгоритмді толтыру
41. Проекциялар
42. Орталық проекциялау
43. Параллельді проекциялау
44. Түстер теориясының негізгі түсініктері

45. Түстік модельдер
46. RGB аддитивті түстік модельдер
47. CMY(K) түстік модель
48. HSB және HSL түстік модельдері
49. Түстер фоны
50. Түстің қанықтылығы
51. Екіөлшемді аффиндік объектілердің түрлендірілуі
52. Түстің ашықтығы
53. Коэн-Сазерленд кесу (қиып алып тастау) алгоритмі
54. Тікбұрышты аймақты кесу (қиып алып тастау) алгоритмі
55. Кирус-Бек алгоритмі
56. Тіктөртбұрышты проекцияның математикалық сипаттамасы
57. Қиықбұрышты проекцияның математикалық сипаттамасы
58. Перспективті проекцияның математикалық сипаттамасы
59. Қалқымалы (плавающий) көкжиек алгоритмі
60. Робертс алгоритмі
61. Жарықтандыру модельдеуі
62. Жарықтың негізгі шағылысу заңдары
63. Гуро әдісімен бояу
64. Фонга әдісімен бояу
65. Реалистік суреттердің құрастырылуы
66. Біркелкі бояу
67. Сәуле трассировкасы жолымен көрінетін жоғарғы бет алгоритмі
68. Сәуле трассировкасының глобальды модель жарықтануы
69. Архитектура және синтаксис ерекшеліктері
70. Консольді қосымшалар құрылымы
71. Open GL төбелері мен примитивтері
72. Төбелер массиві
73. Көріністі түрлендіруі
74. Проекциялар
75. Шығару аймағы
76. Материал қасиеті
77. Жарықтандыру моделі
78. Жолдық сканерлеу алгоритмі
79. Сәулелердің жолдық алгоритмі
80. Z буфер иерархиясы

Глоссарий

Альфа-канал – мөлдірлік сипаттамасы.

Анимация (animation) – кино болып қабылданатын кадр тізбегі.

Антиалиасинг (antialiasing) – растрлық бейнелеудің сатылық эффектісін жою, оларды жеке бұрыштық пиксельмен түстерді беру жолын тегістеу.

Аффиндік түрлендіру – сызықтық түрлендіру, мысалы, координат түрлендіруі.

Биттік массив (bitmap) – дискіде немесе жадыда сақталатын растр.

Векторизация (vectorization) – Растрлық немесе басқа пішіннен векторлық пішінге түрленуі.

Векторлық графика – Растрлық немесе басқа пшіннен векторлық кескіндерді құру.

Көрініcті түрлендіру (view transform) – ракурстікке сәйкес кеңістік объектілерін көрсетуде координаттарды түрлендіру.

Воксел (voxel – volume picture element) – көлемді растр элементі.

Пайдаланушының графикалық интерфейсі (Graphical User Interface, GUI) –компьютерлік жүйеде пайдаланушының кейбір операцияларды орындауына арналған графикалық элементтер жиыны.

Гуро әдісі – қырлар төбесінде интенсивті шағылысатын түстің интерполяциясында қолданылатын үшөлшемді объектілер қырларын бояу әдісі

Дезеринг (dithering) – әртүрлі түстердің жақын орналасқан нүктелерінен пайда болған түстің көлеңке иллюзиясы.

Интерактивті компьютерлік графика – графикалық компьютерлік жүйеде адаммен сұқбат жүргізу кезіндегі аппараттық және программалық тәсілде қолданылатын түсінік.

Компьютерлік графика – компьютер көмегімен кескіндер құру.

Линиатура – ұзындық бірлігіне берілген нүктелер (сызықтар) саны Дизеринг әдісін сипаттау.

Муар – бейнелеудің растрлық элемент құрылғысы мен бейнелеудің растрлық құрылымымен әрекет ету кезінде пайда болған бейнеэффект және өрнек.

Морфинг (morphing) – объект пішіндерін түрлендіру әдісі.

Палитра (pallete) – арнай бейнелерге қажетті түстер жиыны.

Пиксел (pixel – picture element) – растр элементі.

Полигон (polygon) – түзулер қиындысымен байланысқан контурмен шектелетін көпбұрыштар, фигуралар.

Полилиния (polyline) – түзулер қиындысымен байланысқан сынған сызық

Растеризация (rasterizetion) – бейнелеу элементтерін сипаттайтын вектор негізінде растрлық бейнелерді құру.

Растрлық кескін – әртүрлі түсті жақын орналасқан нүктелердің бейнеленуі.

Рұқсат етілген растрлық қабілет (resolution) – растр және растрлық құрылғылардың сипаттамасы. Ұзындық бірлігі пиксель санымен өлшенеді, мысалы, дюймде (dpi).

Спрайт (sprite) – биттік массивте сақталынатын және керек жеріне көшірілетін жеке суреттің растрлық бейнесі.

Тексел (texel – texture element) – растрлық текстура элементі.

Текстура (texture) – бояу стилі объект бетін рельефті етіп көрсетеді. Көбінесе растрлық үлгіде пайдаланылады.

Сәуле трассировкасы (raytracing) – әртүрлі сәуленің таратылу жолы реалистикалық бейнені құрудың әдісіне негізделген.

Триангуляция – байланысқан үшбұрыштар жиыны түріндегі беттік модельдерді қалыптастыру.

Тестураны фильтрлеу – объект бетіне текстураны бейнелейтін интерполяция, түзету әдісі.

Фонга әдісі – төбелердегі вектор нормальдарын үшөлшемді объектілер қырларын бояу әдісіне негізделген.

Фракталдар (fractals) – қарапайым итерация циклдарымен сипатталатын күрделі объектілер пішіндері.

Өскенбаева Р.Қ., Нұралықызы С.

ИНТЕРАКТИВТІ ГРАФИКАЛЫҚ ЖҮЙЕЛЕР

Студенттің пәндік оқу-әдістемелік кешені
5В070400 – Есептеу техникасы және программалық
қамтама мамандығы үшін

Редактор

Техн. редактор

«ЖмЖПҚ» кафедрасының
мәжілісінің хаттамасы «11» қазан 2010 ж. №4 хаттама

«АТ» институтының оқу-әдістемелік
кеңес мәжілісінің хаттамасы «28» қазан 2010 ж. №2 хаттама

Баспаға қол қойылды «____» _____ 2010 ж.
Тираж ____ дана. Пішімі 60х84 1/16. Типографиялық қағаз №1.
Көлемі 6,2 б.ж. Тапсырым № ____. Баға келісімді

Қ.И. Сәтбаев атындағы Қазақ ұлттық техникалық университеті
ҚазҰТУ ғылыми-техникалық баспа орталығы
Алматы қаласы, Ладыгин көшесі, 32