ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет "Комп'ютерних наук і технологій"

Кафедра "Комп'ютерної інженерії"

Мальчева Раїса Вікторівна

ЛЕКЦІЇ

З КУРСУ: «АПАРАТНО-ПРОГРАМНІ ЗАСОБИ СИСТЕМ КОМП'ЮТЕРНОЇ ГРАФІКИ»

Донецьк - 2013

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
Кафедра «Комп'ютерної інженерії»

ЛЕКЦІЇ

з курсу «АПАРАТНО-ПРОГРАМНІ ЗАСОБИ
СИСТЕМ КОМП'ЮТЕРНОЇ ГРАФІКИ»

(для студентів напрямку підготовки 6.050102 «Комп'ютерна інженерія»)

Розглянуто на засіданні кафедри
«Комп'ютерна інженерія»
Протокол № 1 від «29» серпня 2013 р.

Затверджено на засіданні навчально –
методичної комісії кафедри
«Комп'ютерна інженерія»
Протокол № 1 від «29» серпня 2013 р.

Донецьк - 2013

УДК 681.3

Лекції з курсу «Апаратно-програмні засоби систем комп'ютерної графіки» (для студентів напрямку підготовки 6.050102 «Комп'ютерна інженерія») / уклад.: Р.В. Мальчева. - Донецьк, ДонНТУ, 2013. - 132 с.

Лекції складаються з двох частин.

Перша частина присвячена розгляду алгоритмічних основ комп'ютерної графіки. Наведений опис методів апроксимації об'єктів та типова структура системи підготовки графічних даних. Розглянуті цвітові моделі та математичні основи метода випромінюваності. Надані два алгоритму синтезу зображень трьохмірних сцен.

Друга частина містить приклади та аналіз деяких архітектурних рішень при проектуванні систем комп'ютерної графіки. Наданий приклад дослідження одного з архітектурних рішень.

Призначено для студентів вищих навчальних закладів, які навчаються за напрямком підготовки «Комп'ютерна інженерія».

Укладач: Р.В. Мальчева, к.т.н., доцент

Рецензент: Н.С. Костюкова, к.т.н., доцент

Відпов. за випуск: В.А. Святний, зав. каф., д.т.н., проф.

ЗМІСТ

		стор.
Вступ		5
Частина 1	Алгоритмічні основи комп’ютерної графіки	10
Лекція 1	Опис геометричних параметрів об'єктів синтезу: Методи розбивки сцен і моделі апроксимації об'єктів	11
Лекція 2	Реалізація бази даних. системи підтримки бази даних. Машини баз даних (МБД)	18
Лекція 3	Представлення енергетичних параметрів об'єктів синтезу. Колірні моделі. Інтерполяція	32
Лекція 4	Використання методу випромінювальності. Особливості опису сцен	43
Лекція 5	Синтез тримірного об'єкту, апроксимованого випуклим багатогранником	52
Лекція 6	Використання методу пріоритетів для синтезу тримірного об'єкту, апроксимованого випуклими багатогранниками	57
Лекція 7	Використання методу трасування промінів	61
Частина 2	Архітектура систем комп’ютерної графіки	66
Лекція А1	Аналіз підходів до організації графічних систем реального часу	67
Лекція А2	Аналіз основних архітектурних рішень побудови систем синтезу зображення реального часу	79
Лекція А3	Розробка алгоритму розподілу даних у графічній системі реального часу	83
Лекція А4	Апаратна підтримка алгоритму розподілу даних	91
Лекція А5	Дослідження алгоритму трасування промінів	101
Лекція А6	Розробка алгоритму фільтрації для композитного пристрою багатоканальної архітектури	110

стор.

Лекція А7	Розрахунок продуктивності сценарного процесора багатоканальної архітектури	117
Лекція А8	Систолічні процесора	123
Перелік рекомендованої літератури		129

Вступ

ОСНОВИ КОМП’ЮТЕРНОЇ ГРАФІКИ

Поняття про комп'ютерну графіку. Визначним досягненням людства в останні десятиріччя є швидкий розвиток електроніки, обчислювальної техніки та створення на їхній основі багатопланової автоматизованої системи комп'ютерної графіки.

На початку свого розвитку комп'ютерну графіку розглядали як частину системного програмування для ЕОМ або один з розділів систем автоматизованого проектування (САПР). Сучасна комп'ютерна графіка становить ряд напрямів і різноманітних застосувань. Для одних із них основою є автоматизація креслення технічної документації, для інших – проблеми оперативної взаємодії людини й комп'ютера, а також задачі числової обробки, розшифрування та передачі зображень.

Однією з основних підсистем САПР, що забезпечує комплексне виконання проектних робіт на основі ЕОМ, є комп'ютерна графіка (КГ).

Комп'ютерною, або машинною, графікою називають наукову дисципліну, яка розробляє сукупність засобів та прийомів автоматизації кодування, обробки й декодування графічної інформації. Іншими словами, комп'ютерна графіка розробляє сукупність технічних, програмних, інформаційних засобів і методів зв'язку користувача з ЕОМ на рівні зорових образів для розв'язання різноманітних задач при виконанні конструкторської та технічної підготовки виробництва.

Вивчення комп'ютерної графіки зумовлене:

- широким впровадженням системи комп'ютерної графіки для забезпечення систем автоматизованого проектування, автоматизованих систем конструювання (АСК) та автоматизованих систем технологічної підготовки виробництва (АСТПВ) в усіх сферах інженерної діяльності;

- значним обсягом перероблюваної геометричної інформації, що становить 60 .80 % загального обсягу інформації, необхідної для

проектування, конструювання та виробництва літаків, кораблів, автомобілів, складних архітектурних споруд тощо;

- необхідністю автоматизації виконання численних креслярсько-графічних робіт;

- необхідністю підвищення продуктивності та якості інженерної праці.

Метою комп'ютерної графіки є підвищення продуктивності інженерної праці та якість проектів, зниження вартості проектних робіт, скорочення термінів виконання їх.

Завданням комп'ютерної графіки є звільнення людини від виконання трудомістких графічних операцій, які можна формалізувати, вироблення оптимальних рішень, забезпечення природного зв'язку людини з ЕОМ на рівні графічних зображень.

Історія і перспективи розвитку комп'ютерної графіки. Розвиток комп'ютерної графіки (КГ) почався з появою пристроїв графічного виведення. Становлення КГ та широке її використання пов'язано із створенням засобів графічного введення — виведення та дисплеїв — пристроїв побудови зображень на електронно-променевій трубці.

При взаємодії користувача з комп'ютером розрізняють три режими роботи: пакетний; пасивно-інтерактивний; інтерактивний.

У 60-роках виникає ряд дослідних проектів, з'являються розробки, придатні для комерційного розповсюдження. Найбільш значними серед них були проекти фірми GENERAL MOTORS з використанням багатопультової графічної системи з розподілом часу для багатьох етапів проектування автомобілів, система була створена фірмою для проектування лінз і дисплейна система IBM 2250, заснована на прототипі фірми GENERAL MOTORS.

В Україні перша інтерактивна графічна система автоматизованого проектування електронних схем була розроблена Київським політехнічним інститутом і Науково-дослідним інститутом автоматизованих систем

управління та планування в будівництві (НДІАСБ, Київ) і демонструвалася в 1969 р.

В інтерактивному режимі роботи графічна інформація формується і виводиться в режимі оперативної графічної взаємодії користувача та комп'ютера.

Основні галузі застосування комп'ютерної графіки та її компонентів. Застосування КГ для формування різноманітної графічної інформації в різних галузях людської діяльності свідчить про те, що комп'ютерна графіка та геометрія – явища різноманітні й багатопланові. У рамках КГ розв'язуються такі проблеми: розробка нових методів математичного забезпечення; розробка програмних систем графічних мов; створення нових ефективних технічних засобів для проектувальників, конструкторів та дослідників; розвиток нових наукових дисциплін і навчальних предметів, які увібрали в себе аналітичну, прикладну та нарисну геометрії, програмування для ЕОМ, методи обчислювальної математики, приладобудування.

Математичне забезпечення ґрунтується на методі математичного моделювання, згідно з яким математична структура, відношення елементів у математичній моделі відповідає структурі й відношенням у реальному об'єкті. У КГ використовується геометрична версія математичного моделювання, при якому дво- та тривимірні зображення складаються з точок, ліній і поверхонь.

Програмне забезпечення включає програми в машинних кодах, тексти програм та експлуатаційні документи. Основу програмного забезпечення КГ становлять пакети прикладних програм (ППП), які є набором програм, що реалізують на ЕОМ інваріантні та об'єктно-орієнтовані графічні процедури.

Технічне забезпечення – це пристрої обчислювальної й організаційної техніки, засоби передачі даних, вимірювальні та інші пристрої або їх з'єднання. Технічне забезпечення включає комплекс технічних засобів, які забезпечують введення графічної інформації, формування та виведення графічної інформації, редагування. Методичне забезпечення – це документи,

в яких відображено склад, правила відбору та експлуатації засобів автоматизації проектування. До методичного забезпечення належать також кадрові питання, розробка ефективних методів та заходів щодо роботи із системою комп'ютерної графіки.

Функціональний склад комп'ютера. Комплекс засобів для обробки даних складається з компонентів апаратного й програмного забезпечення.

Технічне забезпечення включає комплекс технічних засобів, призначених для введення графічної інформації, формування та виведення результатів у вигляді графічної інформації, редагування зображень.

Технічні засоби – це всі електричні та механічні складові частини комплексу технічних засобів комп'ютера: центральний пристрій, пристрої введення, виведення та передавання даних.

Критерій вибору ЕОМ зумовлений задачами, що розв'язуються. Наприклад, якщо потрібна висока точність обчислень або якщо обробляються великі масиви даних, то доцільно використовувати суперміні-ЕОМ. Основним типом великих і суперміні-ЕОМ, які використовуються в сучасних САПР, є ЕОМ єдиної системи (ЄС ЕОМ).

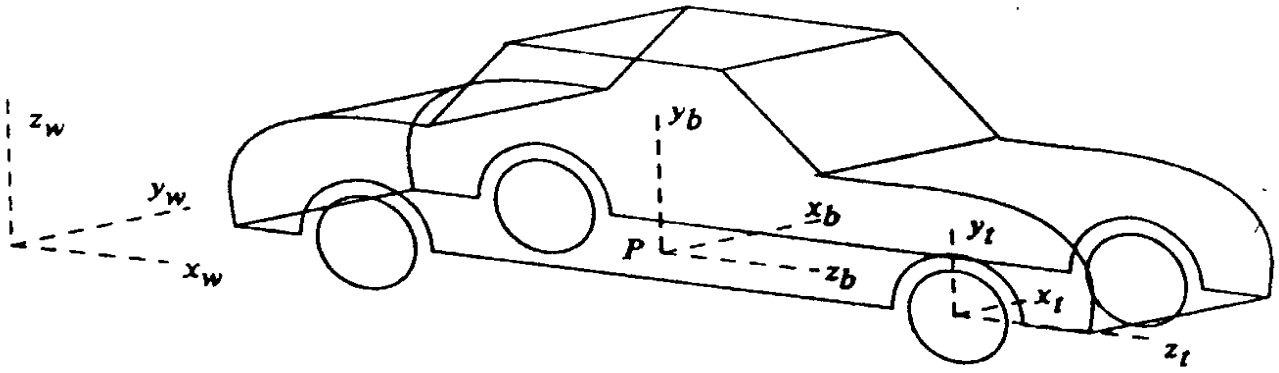
Міні-ЕОМ та мікро-ЕОМ відкривають нові можливості для конструкторів і проектувальників, оскільки сучасні ЕОМ цих класів, об'єднані в мережі, наближаються до великих ЕОМ.

Комп'ютерна система керується центральним процесорним пристроєм, який є головним елементом комп'ютерної системи. Центральний процесорний пристрій (ЦПП) – це пристрій, який розшифровує зміст команд у програмі й виконує (інтерпретує) їх. Іншими словами, ЦПП є засобом переробки графічної інформації, яка проходить етапи створення, обробки та зберігання моделей об'єктів.

Програміська модель інтерактивної графіки. Процес удосконалення програмного забезпечення комп'ютерної графіки був досить довгим і повільним. Пройдено шлях від апаратно-залежних пакетів програм низького рівня, які поставляються виготовниками разом з конкретними дисплеями, до

апаратно-незалежних пакетів високого рівня. Основна мета апаратно-незалежного пакета полягає в тому, щоб забезпечити його мобільність при переході від однієї ЕОМ до іншої. У процесі розв'язання цієї задачі група американської асоціації з використання обчислювальних машин (GGRAPH) запропонувала проект графічної системи CORE SYSTEM, а спеціалісти Німеччини розробили графічну базову систему Grafikal Kernel System (GKS), прийняту Міжнародною організацією по стандартизації (ISO) як міжнародний графічний стандарт. Ця система звільняє програміста від урахування особливостей графічних пристроїв при розробці програмного забезпечення, при цьому користувачеві доступні різноманітні графічні пристрої.

Відповідно до викладеного підходу при розробці програмних засобів комп'ютерної графіки виділяються такі задачі: моделювання, які призначені для створення, перетворення та зберігання моделей геометричних об'єктів (моделюючі системи); відображення цих моделей на графічних пристроях та організації графічного інтерфейсу користувача та ЕОМ (базова графічна система). Склад програмного забезпечення відбивається у програмістській моделі інтерактивної комп'ютерної графіки. До нього входять три компоненти: моделююча система, базова графічна система та прикладна структура даних (база даних). Моделююча система здобуває інформацію та засилає її у бази даних, а також надсилає графічні команди до графічної системи.



ЧАСТИНА 1

АЛГОРИТМІЧНІ ОСНОВИ КОМП'ЮТЕРНОЇ ГРАФІКИ

ЛЕКЦІЯ 1
ОПИС ГЕОМЕТРИЧНИХ ПАРАМЕТРІВ ОБ'ЄКТІВ СИНТЕЗУ: МЕТОДИ РОЗБИВКИ СЦЕН І МОДЕЛІ АПРОКСИМАЦІЇ ОБ'ЄКТІВ

1.1 Критерії оцінки якості графічної системи

Одним із критеріїв оцінки якості графічної системи є ступінь адекватності синтезованого зображення реальної обстановки. При оцінці ступеня відповідності використовують 3 рівні подоби :

- фізичне;
- психофізичне(фізіологічне);
- психологічне.

Фізична подоба встановлюється на рівні трьох груп характеристик: геометричних, яркісних, часових.

Фізіологічна подоба встановлюється на рівні зорових відчуттів і зв'язана з можливостями зорового апарату.

Психологічна подоба припускає, що по загальному сприйняттю синтезоване зображення й оригінал є схожими, тобто забезпечують формування у спостерігача цілком визначеного судження про реальний об'єкт чи сюжет.

У загальному виді процес синтезу можна представити в такий спосіб:

$$G_{CI} = A_{MГ} \cdot G_{мод}, \qquad (1.1)$$

де $G_{мод}$ - математична модель простору, яку відображують (3D-модель);

$A_{MГ}$ - процес перетворення (обробки);

G_{CI} - двомірне зображення, сформоване системою машинної графіки.

Висновок: Ступінь адекватності синтезованого зображення залежить як від характеристик системи обробки, так і від якості вихідної моделі.

1.2 ВИМОГИ ДО МАТЕМАТИЧНОЇ МОДЕЛІ ВИЗНАЧЕННЯ ОБ'ЄКТА ВІЗУАЛІЗАЦІЇ

Серед необхідних складових частин, які повинні бути представлені в моделі, можна назвати наступні:

- основні елементи сцени і їхні взаємини;
- геометрія сцени, тобто просторове розміщення, форма, розміри окремих складових сцени;
- топологія сцени, тобто інформація про взаємне розташування і связністі компонентів сцени;
- специфічні для розглянутого застосування дані, наприклад, електричні чи механічні характеристики; описовий текст.

У загальному випадку модель складається з прикладної структури даних і набору процедур прикладної програми для підтримки цієї структури.

1.3 МОДЕЛІ ОБ'ЄКТІВ ТА ЇХНЯ КЛАСИФІКАЦІЯ

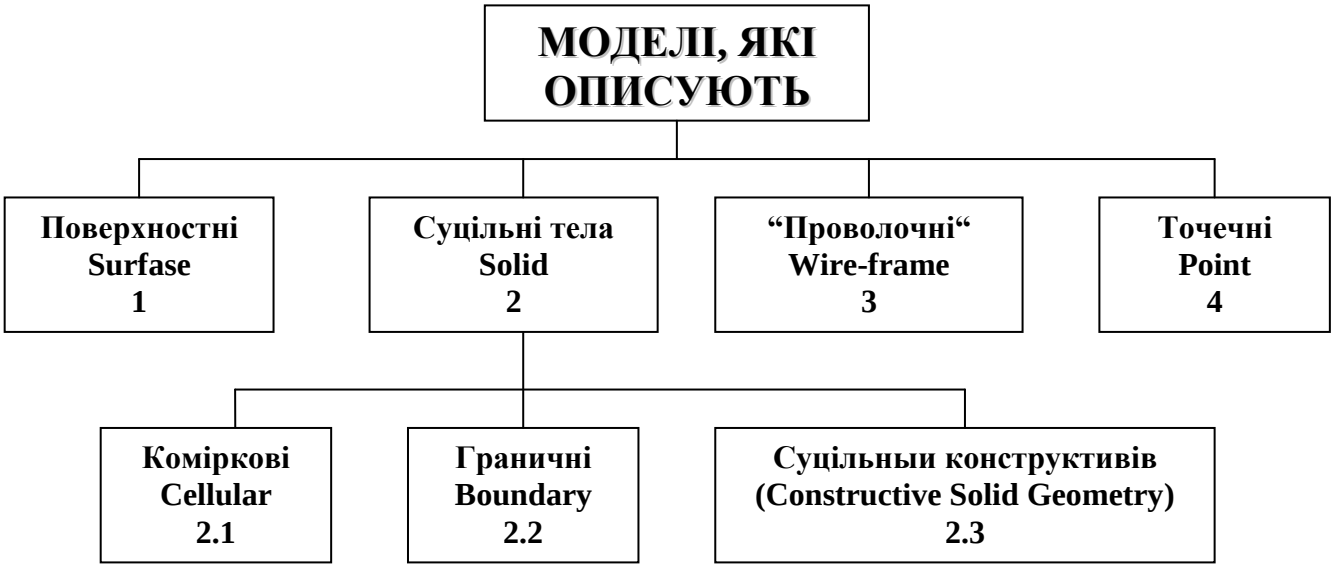


Рис.1.1. Класифікація моделей опису сцен

Підхід **1** представляє объект у вигляді тонких поверхонь , під якими знаходиться порожній простір, не заповнене матеріалом об'єкта.

Підхід **2** має на увазі, що чи об'єкту окремому примітиву преналежать усі крапки об'єкта - як зовнішні, так і внутрішні.

Опис типу **3** полягає в представленні поверхні серією пересічних ліній, що належать поверхні об'єкта.

Опис **4** представляє поверхню об'єкта безліччю окремих крапок. Основною особливістю такого опису служить відсутність інформації про поверхню між крапками. Застосовується в тих випадках, коли поверхня дуже складна, не має гладкість а детальне представлення численних геометричних особливостей важливо для практики. (Ділянки ґрунту планет, форми малих небесних тіл - із супутників / мікрооб'єкти - зняті за допомогою мікроскопів)

Поверхні

1-го порядку:

З поверхонь першого порядку можна скласти опис об'єкта типу полігонального покриття (polygon - багатокутник). Таким покриттям називають серію суміжних багатокутників, які не мають розриву між собою. Багатокутник задається матрицею координат

$$\Lambda = \begin{bmatrix} X_1 & Y_1 & Z_1 \\ . & . & . \\ . & . & . \\ . & . & . \\ X_n & Y_n & Z_n \end{bmatrix}$$

і нормальним вектором

N = iA + jB + kC

де Ax + By + Cz + D =0 - рівняння поверхоні першого порядку

n = {[ay*bz - az*by] , [az*bx - ax*bz] , [ax*by - ay*bx]}

Простота обробки площин заслужено привернула увагу багатьох дослідників і розроблювачів.

2-го порядку :

З аналітичної геометрії відомо, що функція виду

$f2(x,y,z) = Ax^2 + By^2 + Cz^2 + 2Dxy + 2Eyz + 2Fzx + 2Gx + 2Hy + 2I z + K = 0$

у залежності від набору коеф. A, B, ... , K може описувати поверхні еліпсоїда, гіперболоїда, конуса, параболоїда, циліндра і 2-х площин. У матричному виді функція

$$f2(x,y,z) = [\ X\ Y\ Z\ 1\ ]\ P\ [\ X\ Y\ Z\ 1\ ],$$

де

$$\mathbf{P} = \begin{bmatrix} A & D & F & G \\ \mathbf{D} & \mathbf{B} & \mathbf{E} & \mathbf{H} \\ F & E & C & I \\ G & H & I & K \end{bmatrix}$$

нормаль у крапці (X, Y, Z) $N = \Omega * \Psi$, где

$$\Psi = \begin{bmatrix} AX + DY + EZ + G \\ BY + DX + EZ + H \\ CZ + FX + EY + J \end{bmatrix}$$

$\Omega = [\ i\ j\ k\ ]$ ортогональних осей Ox, Oy, Oz.

Наприклад , для кулі $X^2 + Y^2 + Z^2 - 1 = 0$ нормаль йде усередину , а для $-X^2 - Y^2 - Z^2 + 1 = 0$ - назовні кулі.

Матричні операції дозволяють виконати всі етапи обробки і застосовувати поверхні 2-го порядку в методі зворотного трасування променів.

Поверхні типу екструзій (extrusion) - металеві чи керамічні профілі, видавлені з розплаву чи глини, а також поверхні обертання, вирізані з заготівлі.

Звичайно поверхні цього типу описують у виді усічених конусів.

Фрактальні поверхні - клас нерегулярних форм, що задаються можливісним образом на основі вихідного опису низького дозволу.

“Фрактальний” - що складається з часток, частин. Найбільше часто застосовують для моделювання гірського ландшафту:

- попередньо гірський масив описують дуже приблизно полігональним полем з чотирикутників;
- кожен чотирикутник розбивають за допомогою випадкової функції на 4 фігури менших розмірів і вероятнісним образом зрушують ці фігури щодо площини вихідного чотирикутника, зберігаючи для кожної фігури по одній загальній вершині з вихідним чотирикутником;
- і т.д. до досягнення бажаного рівня зрізаності поверхні.

Зображення створені на основі фрактальних поверхонь, тільки статично ідентичні реальним об'єктам, від них не можна вимагати ідеальної точності.

Суцільні тіла

Ячєєчні методи . Ідеї дуже прості.

Весь моделюємий об'єкт вважається розбитим на велике число дискретних кубічних осередків . Моделююча система повинна просто записати інформацію про приналежність чи неприналежності кожного куба тілу об'єкта.

Структура даних представляється тривимірною матрицею, у якій кожен елемент відповідає просторовому осередку .

Недолік методу: великий обсяг пам'яті, необхідний для запису об'єкта з високим дозволом .

Граничне представлення .

У пам'яті зберігаються поверхні, краї поверхонь, покажчики перетинань поверхонь. При цьому структура даних будується одночасно з процесом синтезу.

Моделі опису границями допускають використання булевих операцій і операцій над безлічами .

Переваги : великі можливості геометричного моделювання форм.

Недоліки : модель логічно хитлива , тобто можливе створення суперечливих конструкцій .

Приклад : Новосибірська система.

Метод суцільних конструктивів

Представляє складні об'єкти складеними з простих об'ємних примітивів : куби, циліндри, конуси, еліпсоїди.

Структура даних такої моделі ідентична бінарному дереву (побудованому знизу нагору, рис.1.2).

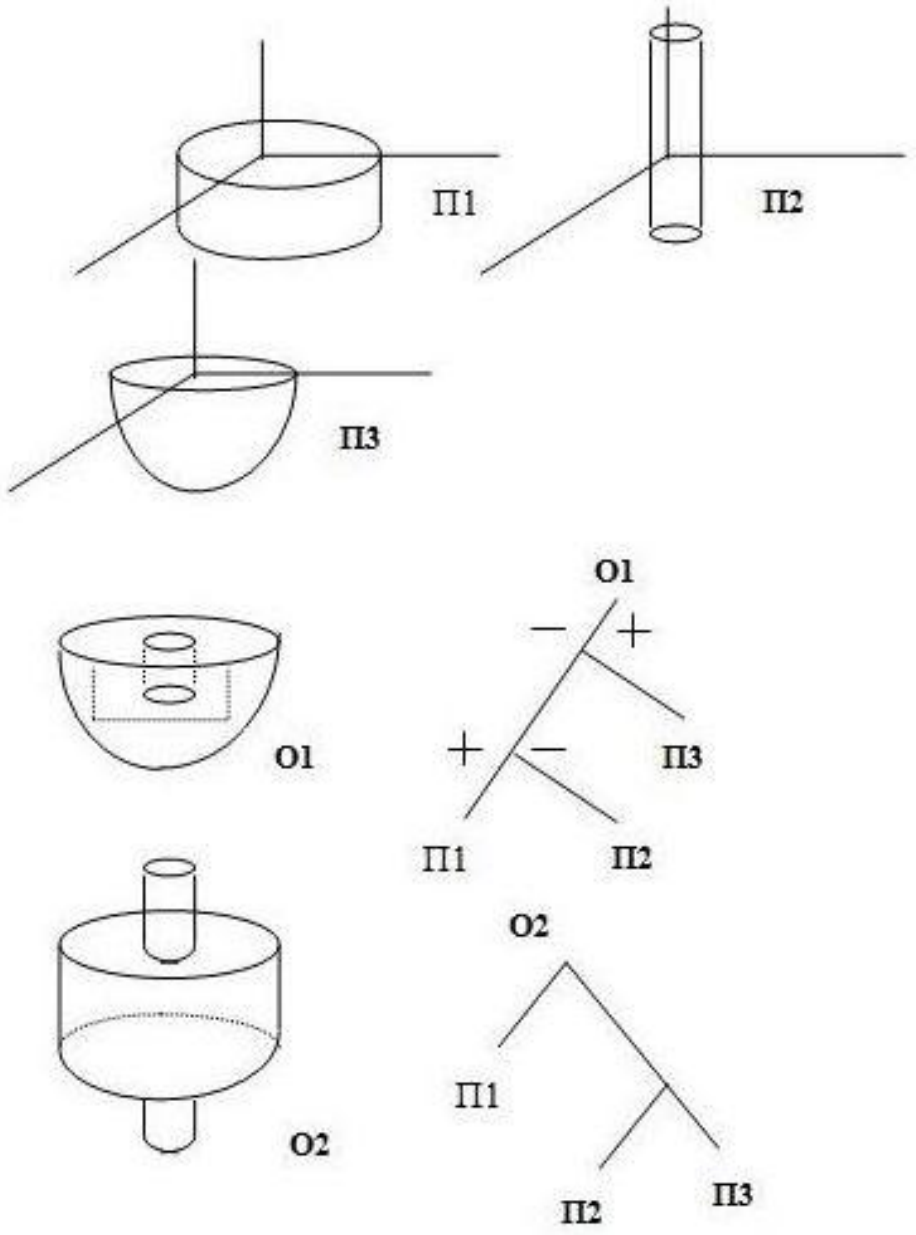


Рис.1.2. Структура даних моделі суцільних тел:
вузли - оператори над примітивами ; листи – примітиви.

Можна додавати оператори переносу, повороту, масштабного перетворення.

Переваги: простота концепції, малий обсяг вихідної пам'яті , пристосованість до ускладнення моделі ; простота представлення частин і перетинів.

Недоліки:

1) обчислювальні алгоритми обробки обмежені рамками булевих функцій;

2) більш другого порядку створити дуже складно.

Лекція 2

РЕАЛІЗАЦІЯ БАЗИ ДАНИХ. СИСТЕМИ ПІДТРИМКИ БАЗИ ДАНИХ. МАШИНИ БАЗ ДАНИХ (МБД)

2.1 Підтримка графічних баз даних

Для реалізації графічної бази даних необхідная апаратна та програмна підтимка. Конфігурація апаратних засобів підтримки графічних баз даних наведена на рис.2.1. Структура даних в графічних базах даних дана на рис.2.2. На рис.2.3. наведена конфігурація програмних засобів підтримки. Загальна класифікація графічних баз даних наведена на рис.2.5.

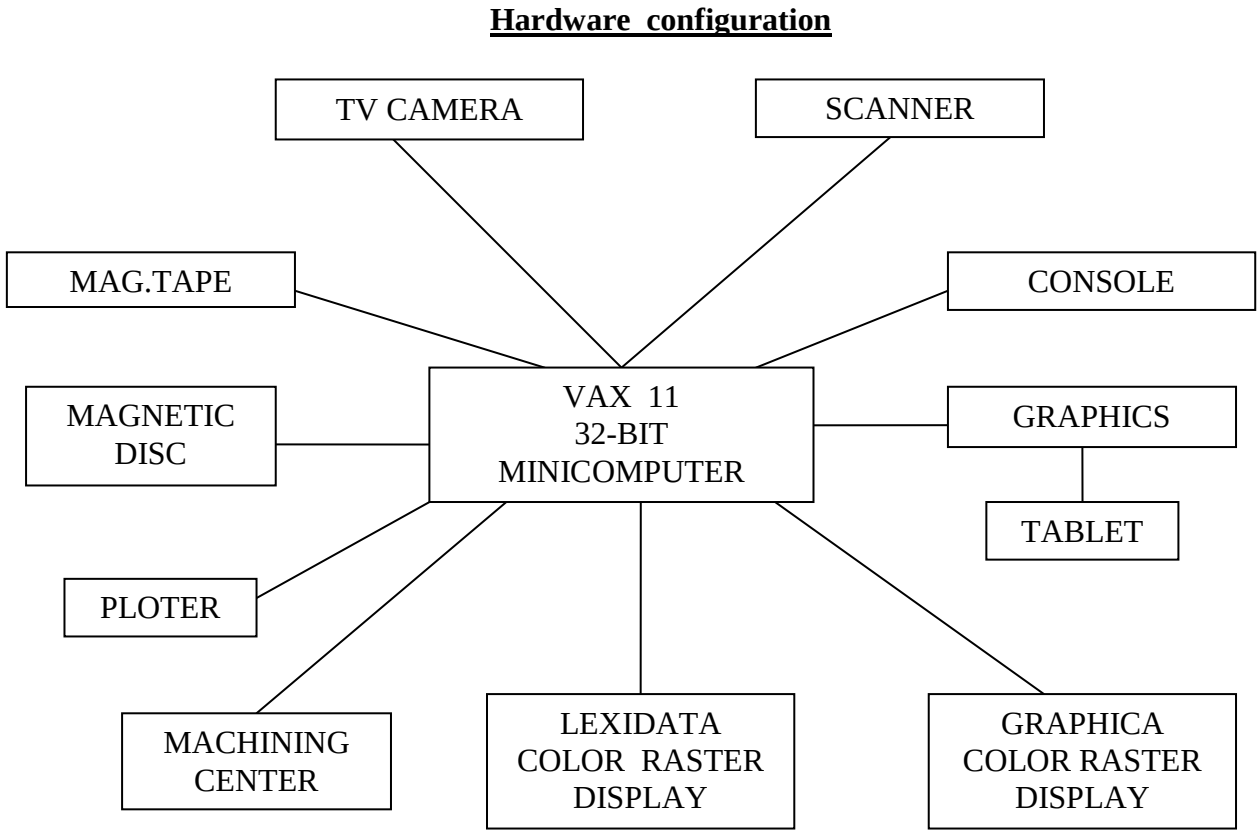


Рис. 2.1. Конфігурація апаратних засобів підтримки графічних баз даних

Data structure.

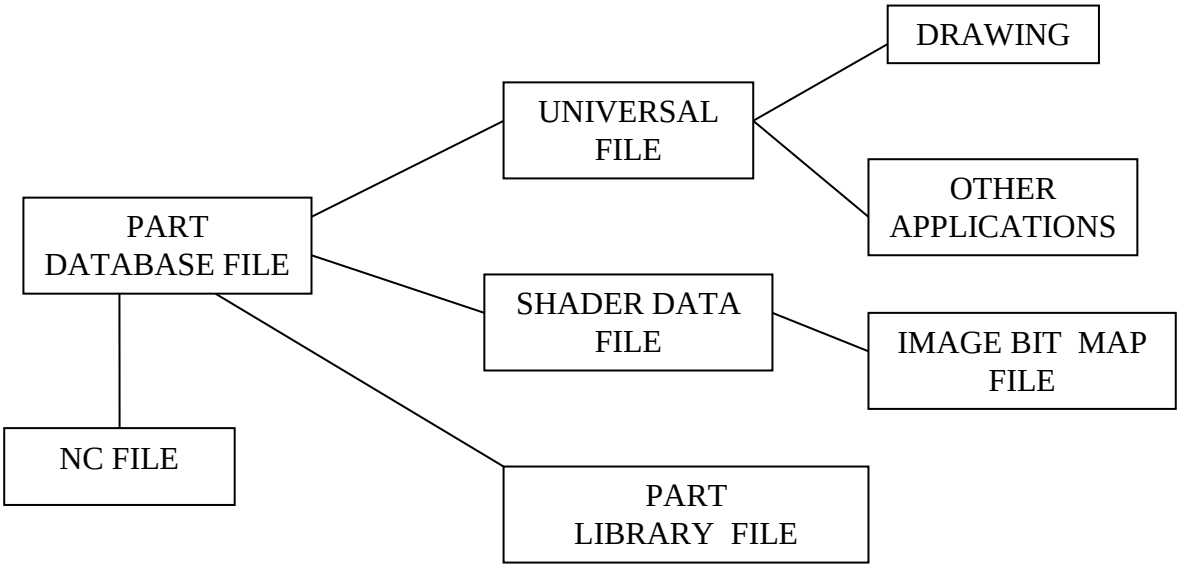


Рис. 2.2. Структура даних в графічних базах даних

Software configuration.

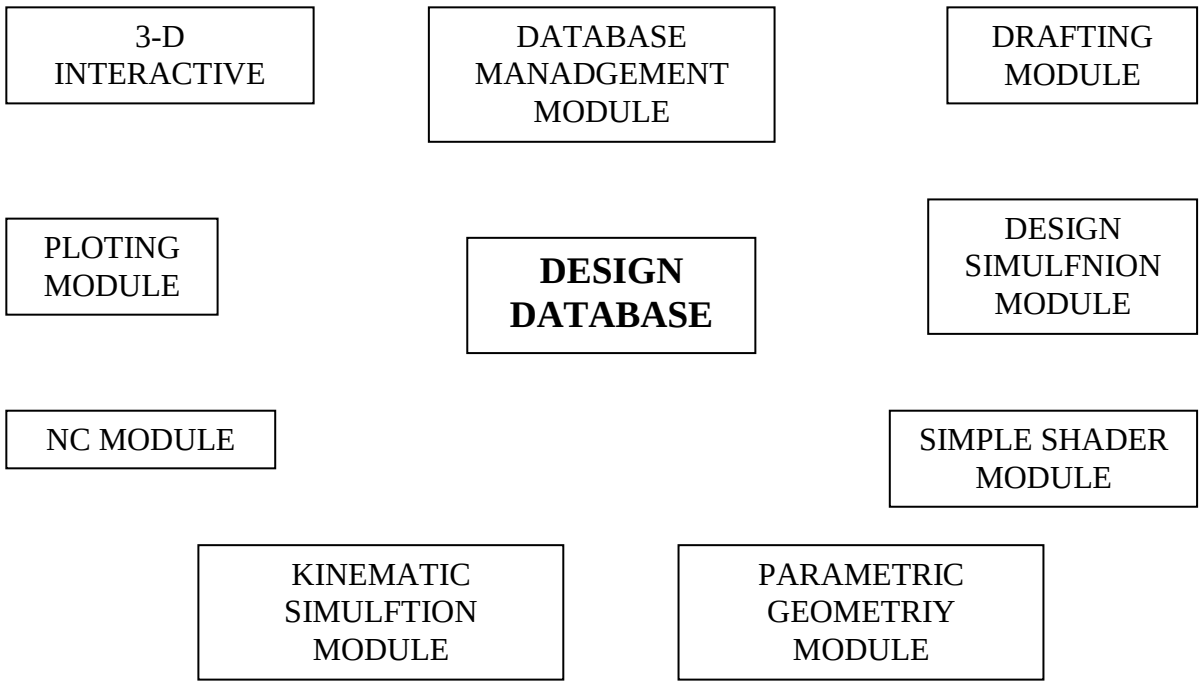


Рис. 2.3. Конфігурація програмних засобів для підтримки графічних баз даних

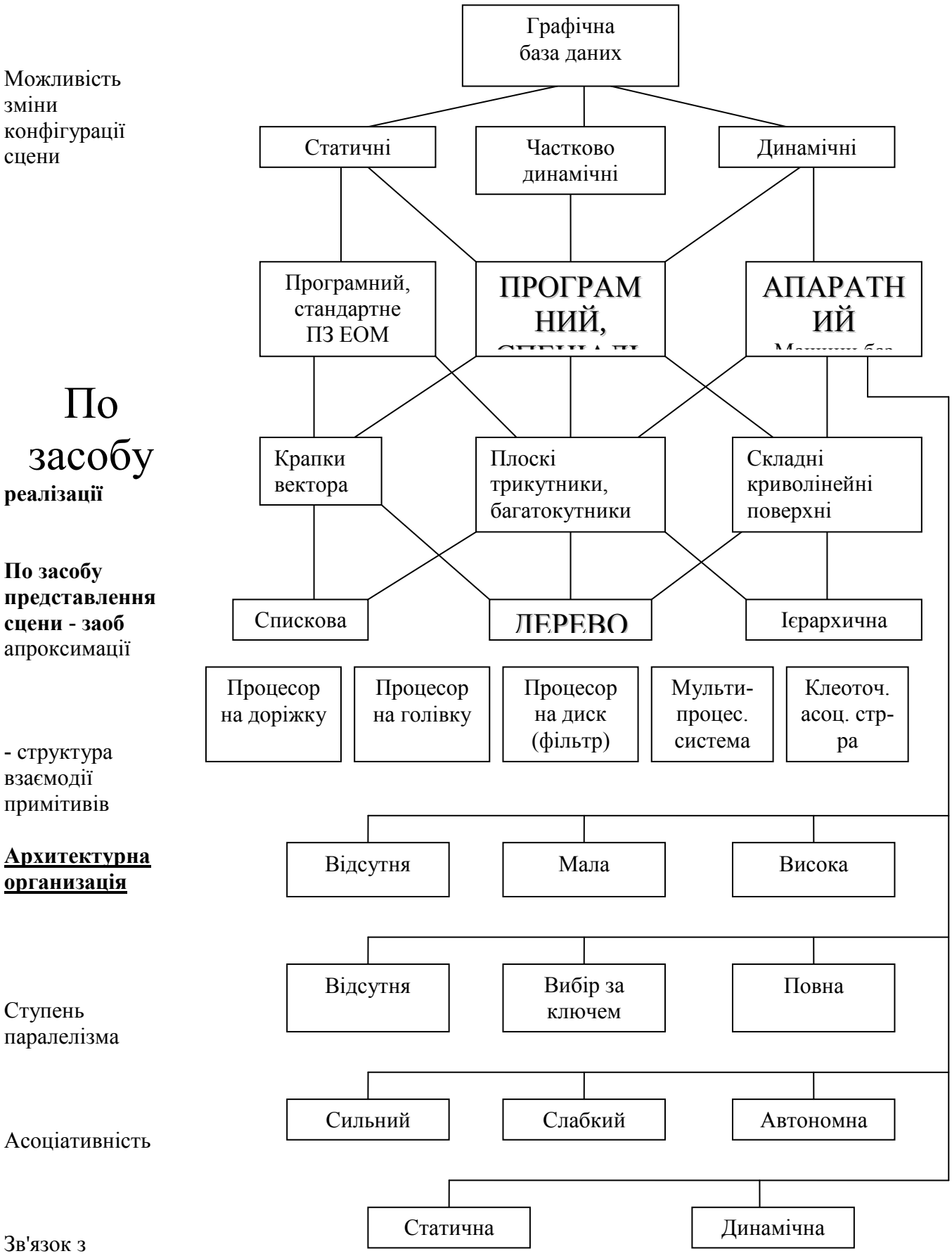


Рис. 2.5. Класифікація графічних баз даних

2.2 Приклад система автоматизованого введення бази даних відображуваної сцени

Система автоматизованого введення бази даних відображуваної сцени передбачає напівавтоматичне створення бази дані описи синтезованої сцени. Сцена, у загальному випадку, являє собою кілька об'єктів візуалізації, що рухаються незалежно друг від друга в деякому просторі.

Під об'єктом візуалізації розуміється тіло в тривимірному просторі, заданий у своїй системі координат і єдине функціональне призначення, що має. Об'єкт візуалізації (складений об'єкт) підрозділяється на примітивні об'єкти, що представляють собою опуклі багатогранники і мають єдине геометричне представлення і функціональне призначення. Топологія об'єкта визначає взаємне розташування складових його примітивних об'єктів.

Для опису примітивного об'єкта застосовується граничне представлення: сукупність обмежуючих його плоских поверхонь. Т.ч. границя примітивного об'єкта розділяється на кінцеве число граней. Грань представляється обмежуючими її ребрами і вершинами. Геометрія примітивного об'єкта задається координатами його вершин і орієнтацією граней у просторі, а його топологія визначає зв'язки між вершинами і гранями.

Побудова бази даних одного об'єкта візуалізації провадиться в наступному порядку:

- виділення примітивних об'єктів;
- формування характеристик примітивних об'єктів;
- розбивка навколооб'єктного простору на субпростіри поділяючими площинами, обираними на підставі топології об'єкта;
- формування пріоритетних списків граней для кожного з субпростірів;
- побудова оптимального дерева пошуку;
- компонування складеного об'єкта з примітивних об'єктів.

2.2.1 Виділення примітивних об'єктів

У [1] Робертс запропонував рішення, відповідно до якого будь-який замкнутий об'єкт із плоскими гранями можна представити у виді набору опуклих багатогранників. Перевагою завдання об'єктів у такій формі є однозначність визначення опуклого багатогранника набором площин, що утворять його грані.

Виділення примітивних об'єктів доцільно виконувати також по наступним причинах. По-перше, у складеному об'єкті можуть бути присутнім ідентичні об'єкти, розміщені в різних крапках (наприклад, декілька ЛА на рухливій платформі), у цьому випадку формується тільки один опис характеристик примітивного об'єкта в зв'язаній системі координат, а потім виконується розмноження (з відповідними перетвореннями) цих характеристик у процесі компонування складеного об'єкта. По-друге, при зміні топології складеного об'єкта в процесі компонування досить скорегувати тільки місце розташування окремих примітивних об'єктів. По-третє, зміна чи форми інших характеристик якого-небудь примітивного об'єкта приводить до необхідності коректування опису тільки цього об'єкта без зміни інших, що знижує вероятність внесення помилок у процесі коректування.

2.2.2 Формування характеристик примітивного об'єкта

Структура опису примітивного об'єкта включає:

Кгр - кількість граней, опис характеристик граней і список номерів граней.

Формування опису примітивного об'єкта містить наступні етапи:

- апроксимацію примітивного об'єкта плоскими гранями;
- формування тривимірних координат вершин граней у зв'язаної системі координат (центр зв'язаної системи координат, звичайно збігається з конструктивним центром об'єкта);
- завдання орієнтації кожної грані в просторі;
- формування пріоритетного списку граней.

Тому що реальні об'єкти, у загальному випадку, обмежені досить складними геометричними поверхнями, як перший етап опису примітивного об'єкта виконується апроксимація його плоскими гранями. Усі вершини і грані отриманого опуклого багатогранника нумеруються (для кожного примітивного об'єкта використовується своя нумерація), при цьому кількість граней і кількість вершин у кожне грані обмежено. Апроксимація примітивного об'єкта виконується проектувальником вручну на підставі наявних креслень.

Підготовка тривимірних координат вершин граней виконується або по кресленнях проекцій примітивного об'єкта з використанням графічного пристрою введення, або спеціальними перетвореннями описів раніше сформованих об'єктів. Пристрій графічного введення дозволяє вводити тривимірні координати вершин граней, уведення списку номерів вершин грані, ознак граней і показників інтенсивності для кожної вершини виконується з клавіатури. Тому що пристрою введення мають визначену погрішність, виконується усереднення координат кожної вершини по

$$Xi = \frac{\overset{Ki}{S}(X_i)j \;\; i \;\; SP_j}{K_i}, \qquad (2.1)$$

наступній формулі:

де *i* - номер вершини,

Ki - кількість граней, яким вершини належить,

j - номер грані,

SPj - список вершин грані,

(*Xi)j* - уведене значення координати *X* вершини *i* при введенні описів для грані *j*,

Xi - усереднене значення координати *X* вершини *i*.

По введених координатах вершини автоматично виконується розрахунок нормалей вектора орієнтації **N** = *Nx,Ny,Nz* кожної грані в просторі у відповідності з наступними вираженнями:

$$Nx = \sum_{i=0}^{m-1} (Y_j - Y_i)*(Z_j - Z_i)$$

$$Ny = \sum_{i=0}^{m-1} (Z_j - Z_i)*(Y_j + Y_i) \qquad (2.2)$$

$$Nx = \sum_{i=0}^{m-1} (X_j - X_i)*(Y_j + Y_i)$$

де *Xi, Yi, Zi* - координати вершин грані,

i = 0,1,...,m-1,

j = 0 для *i=m-1*, інакше *j=i+1*.

Для формування характеристик симетричних примітивних об'єктів застосовується дзеркальне відображення, виконуване для кожної вершини симетричним її відображенням щодо заданої площини.

Симетрія щодо площини, рівнобіжної *Xo* і знаходиться на відстані *a* від її*:*

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 2a \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (2.3)$$

Симетрія щодо площини, рівнобіжної *Yo* і знаходиться на відстані *b* від її*:*

$$\begin{bmatrix} -1 & 0 & 0 & 2b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad (2.4)$$

Симетрія щодо площини, рівнобіжної *Yo* і знаходиться на відстані *c* від її*:*

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 2c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.5}$$

Для виконання симетричного відображення щодо довільної площини додаються перетворення обертання.

Після дзеркального відображення примітивного об'єкта виконується перевірка правильності обходу вершин при завданні списку вершин грані і коректування, при необхідності, цих списків. Перевірка правильності обходу вершин для кожної грані виробляється в такий спосіб:

- по отриманих координатах вершин з використанням співвідношень (2.1) розраховується нормаль грані;
- обчислюється скалярний добуток розрахованої нормалі і нормалі, отриманої в процесі дзеркального відображення;
- якщо нормалі мають різний напрямок, порядок обходу вершин грані міняється на протилежний (з одночасним обміном порядку розташування характеристик граней у загальному описі примітивного об’єкта.

Пріоритетний список граней примітивного об’єкта будується по наступному принципу. Усієї грані об’єкта поділяються на два типи. Перший тип – грань, що задає границю об’єкта. Другий тип – грань-розмітка, що представляє собою малюнок на граничній грані. Грань-розмітка цілком належить граничної грані, що породжує, і не може бути увігнутої. У пріоритетному списку номера граней-розміток повинні випливати безпосередньо за номером відповідної граничної грані. Послідовність граничних граней у пріоритетному списку примітивного об’єкта довільна.

Над сформованим примітивним об’єктом виконується два види операцій.

1. Коректування топології об’єкта: додавання грані, додавання вершини в грань, видалення грані, видалення вершини з опису грані.
2. Коректування геометрії об’єкта: коректування координат вершини грані, коректування інтенсивності вершини, масштабування координат вершин.

2.2.3 Формування бази даних тривимірного об'єкта по його ортогональних проекціях

Побудова бази даних тривимірного опуклого об'єкта по його ортогональних проекціях (видам зверху і попереду) на персональному комп'ютері IBM PC в операційній системі MS DOS реалізовано апаратно-програмним комплексом. При цьому введення координат виробляється напівавтоматично з креслення, зафіксованого на графічному планшеті СМ6404. Перед викликом програми формування бази даних тривимірного об'єкта необхідно завантажити драйвер планшета dg6404.exe. Для цього досить у файл AUTOEXEC.BAT уключити рядок dg6404

На рис.2.6. показано координатну систему графічного планшета. Координати крапки, що повертаються драйвером планшета, лежать у межах від 0 до 20480.

Для зчитування координат крапки, на яку установлений візир графічного планшета, необхідно ініціювати переривання 79h при AX=3. Якщо після виходу з переривання в регістрі AX повертається 0, то графічний планшет очікує натискання якої-небудь клавіші на своєму пульті і йому чогось передавати. Якщо після повернення з переривання в регістрі AX повертається 2, то драйвер передає координату X візира в регістрі BX і координату Y у регістрі CX, причому ці координати передаються при ненатиснутій пік-клавіші планшета. Якщо після повернення з переривання в регістрі AX повертається 3, то драйвер передає координату X візира в регістрі BX і координату Y у регістрі CX, причому ці координати передаються при натиснутій пік-клавіші планшета. Якщо після повернення з

переривання в регістрі AX повертається 4, то це вказує на те, що натиснуто клавішу на пульті планшета. При цьому номер натиснутої клавіші повертається в регістрі BX (клавіші на пульті планшета пронумеровані з 0 ліворуч праворуч і зверху вниз). Якщо після повернення з переривання в регістрі AX повертається 5, то драйвер передає координати крапки, на яку установлений візир планшета (координата X у регістрі CX, координата Y у регістрі DX), і номер натиснутої клавіші (у регістрі BX).

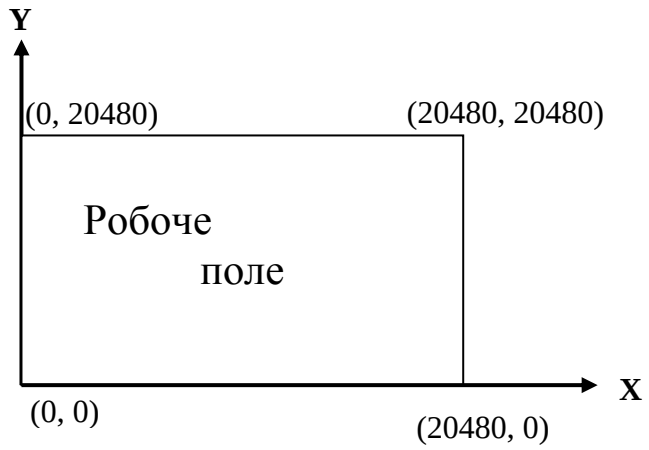


Рис. 2.6. Система координат графического планшета CM6404

Процес формування бази даних тривимірного об'єкта включає наступні етапи:

1. Завдання границь для кожного виду на кресленні.
2. Завдання координат екземплярних крапок, уведення масштабів і розрахунок матриць перетворень.
3. Уведення характеристик граней об'єкта.
4. Перегляд сформованої бази даних.

Причому користувач має можливість реалізувати дану послідовність дій шляхом вибору необхідних пунктів з меню. Якщо користувач спробує виконувати перераховані чотири операції не в зазначеному порядку, то програма видасть повідомлення про помилку. Програма написана мовою ТурбоСі і відкомпільована в інтегрованому середовищі Turbo 2.0.

2.2.4 Ініціалізація роботи

У головній функції main програми після формування зображення меню виконується чекання натискання клавіші на клавіатурі комп'ютера. При натисканні функціональної клавіші F1 (розширений код 0:59) викликається функція help, за допомогою якої реалізована убудована допомога для користувача. При натисканні функціональної клавіші F2 (розширений код 0:60) відбувається виклик функції view_limits, що реалізує інтерактивне завдання границь для кожного виду креслення. При натисканні функціональної клавіші F3 (розширений код 0:61) викликається функція exampl_points, завдання координат екземплярних крапок, розрахунок матриць перетворень і введення масштабів. При натисканні функціональної клавіші F4 (розширений код 0:62) викликається функція input, призначена для інтерактивного введення граней об'єкта. Нарешті при натисканні функціональної клавіші F5 (розширений код 0:61) викликається функція output, що дозволяє переглянути на дисплеї сформовану базу даних тривимірного об'єкта. Для виходу з програми в MS DOS необхідно натиснути клавішу ESC (код ASCII 27).

2.2.5 ЗАВДАННЯ ГРАНИЦЬ ВИДІВ

При натисканні функціональної клавіші F2 (розширений код 0:60) викликається функція view_limits, призначена для інтерактивного введення граничних крапок для кожного виду креслення. Як для виду попереду, так і для виду зверху користувач повинний задати нижній лівий і правий верхній кути (рис.2).

2.2.6 Завдання координат екземплярних крапок, уведення масштабів і розрахунок матриць перетворень

При натисканні функціональної клавіші F3 (розширений код 0:61) викликається функція exampl_points, призначена для інтерактивного введення екземплярних крапок. Користувач, випливаючи підказкам програми, повинний задати початок координат на кожнім виді: (x0_t,y0_t)

для виду зверху і (x0_f,y0_f) для виду попереду, а також увести по три екземплярні крапки для кожного виду: (x1X_t,y1X_t), (x1Y_t,y1Y_t), (x1Z_t,y1Z_t) для виду зверху і (x1X_f,y1X_f), (x1Y_f,y1Y_f), (x1Z_f,y1Z_f) для виду попереду. Екземплярні крапки розташовуються на осях координат на одиничній відстані від початку координат (при цьому одна з екземплярних крапок на осі, перпендикулярній площині креслення, буде сполучена з початком координат).

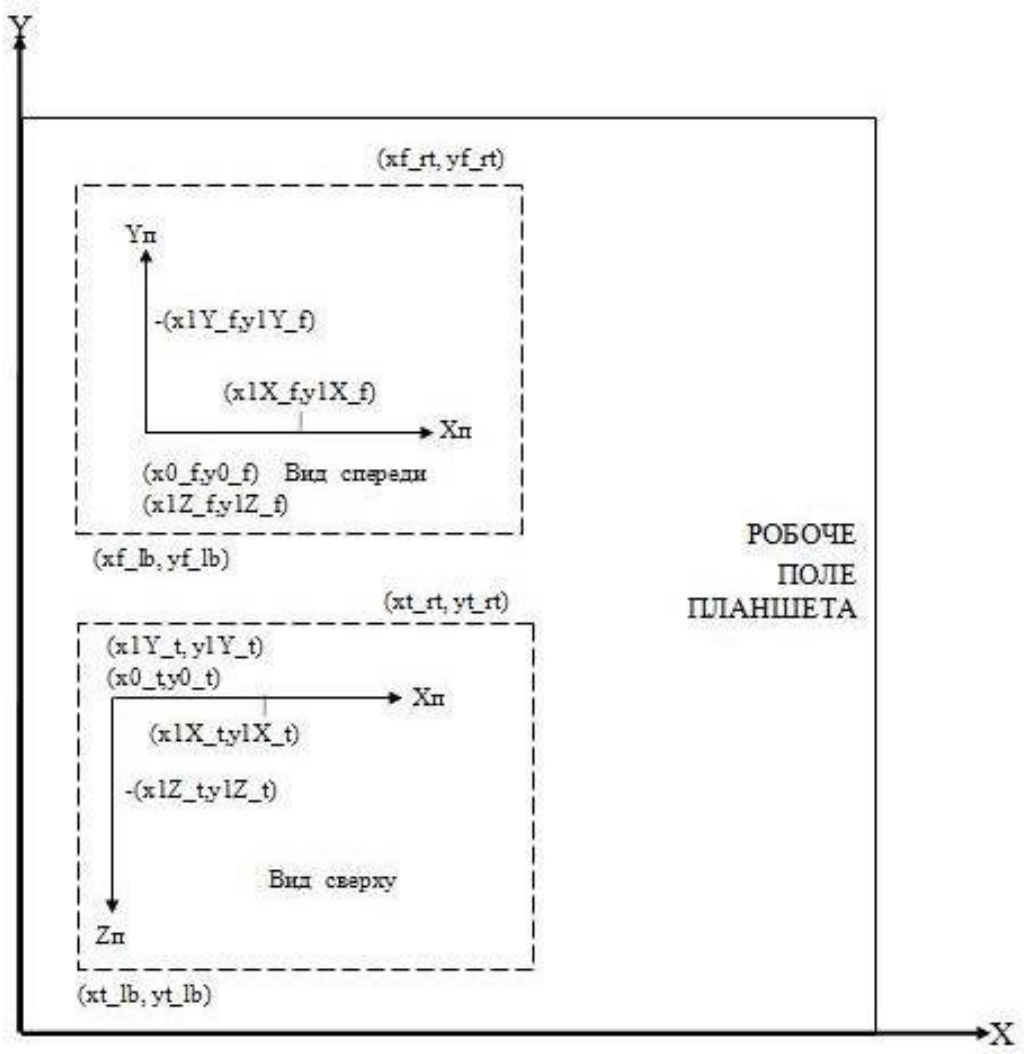


Рис. 2.7. Координатные системы чертежа в ортогональных проекциях

Потім необхідно ввести з клавіатури комп'ютера чисельні значення масштабів по кожної їхніх трьох осей Хп, Уп, Zп: відповідно scale[0], scale[1], scale[2]. Уведені значення використовуються для розрахунку матриці obr_matrix, що буде використана надалі на етапі введення координат вершин граней об'єкта і дозволить розрахувати тривимірні координати

вершини по двовимірних координатах її проекцій з компенсацією помилок за рахунок повороту креслення щодо координатних осей планшета. Для формування матриці obr_matrix використовується функція invmatrix.

2.2.7 Введення характеристик граней об'єкта

Формування опису тривимірного об'єкта складається з послідовності введення граней. Грань уводиться шляхом обходу її вершин по годинній стрілці, причому кожна вершина задається по підказці програми спочатку на виді зверху (x_t, y_t), а потім на виді попереду (x_f, y_f). Отримані чотири координати перетворяться в результуючу тривимірну крапку x[i], y[i], z[i] шляхом множення їх на матрицю obr_matrix у функції multiply. Крім того, для вершини необхідно ввести з клавіатури її номер і код кольору (у межах від 0 до 255). Після уведення вершин грані програма розраховує координати вектора-нормалі до площини грануй і додає відповідну йому крапку в список вершин грані. Потім програма попросить користувача ввести з клавіатури чисельне значення номера грані й ознаки грані. Для закінчення введення граней необхідно натиснути клавішу П0 на пульті планшета. При цьому у функції input у початок файлу з розширенням .num заноситься сумарне число граней об'єкта.

При натисканні функціональної клавіші F4 (розширений код 0:62) викликається функція input, призначена для інтерактивного введення граней тривимірного опуклого об'єкта. Результуюче опис тривимірного опуклого об'єкта включає два двоїчних файли. Ім'я першого файлу вводиться користувачем із клавіатури і не має розширення (його можна вважати ім'ям об'єкта). Цей файл складається з послідовності записів, кожна з який описує одну грань об'єкта. Структура запису показана нижче.

```
struct
        {
          unsigned    int num_edge, /* Номер грані          */
                      priznak_edge, /* Ознака грані        */
                      num_points,   /* Кількість вершин     */
                      points[16];   /* Список номерів вершин */
             struct
```

```
        {
          long int x, y, z;   /* Тривимірні
                                 координати вершини*/
          unsigned char color;/* Колір вершини      */
         }pnt[16];       /* Масив структур вершин */
 }edge;                  /* Опис однієї грані */
```

Ім'я другого файлу виходить шляхом автоматичного додавання до імені першого файлу розширення .num. Даний файл складається з двобайтних записей, кожна з який містить ціле число, причому найперший запис зберігає загальне число граней об'єкта, а наступні записи містять номера граней об'єкта. Послідовність номерів граней у другому файлі відповідає послідовності описів граней у першому файлі. Є наступні обмеження: кількість граней об'єкта не повинний перевищувати 31, а кількість вершин грані повинне бути не менш 3 і не більш 16.

При виконанні функції input користувачу спочатку пропонується ввести з клавіатури ім'я об'єкта (тобто ім'я файлу). Програма перевіряє, чи є на диску файл із таким ім'ям. Якщо такого файлу ні, то він створюється, що відповідає формуванню нового об'єкта. Якщо ж такий файл існує, то вводиться в наступному інформація дописується в кінець файлу, що відповідає додаванню граней до наявного об'єкта.

2.2.8 Перегляд сформованої бази даних

При натисканні функціональної клавіші F5 (розширений код 0:63) викликається функція output, призначена для перегляду сформованих раніше баз даних тривимірних опуклих об'єктів. Користувач повинний увести з клавіатури ім'я об'єкта. Програма відкриває два файли(ім'я першого файлу збігається з ім'ям об'єкта, а ім'я другого файлу виходить шляхом додавання до імені об'єкта розширення .num), а потім виводить на екран інформацію про одну грань у відповідності зі структурою, що описує одну грань об'єкта. Користувач може переглянути наступні грані, натискаючи на клавіатурі клавішу PgDn, чи попередні грані, натискаючи клавішу PgUp.

Лекція 3

ПРЕДСТАВЛЕННЯ ЕНЕРГЕТИЧНИХ ПАРАМЕТРІВ ОБ'ЄКТІВ СИНТЕЗУ. КОЛІРНІ МОДЕЛІ. ІНТЕРПОЛЯЦІЯ

3.1 Колір

Колір - надзвичайно складна проблема як для фізики, так і для фізіології. Колір предмета залежить не тільки від самого предмета, але також від джерела кольору, що висвітлює предмет, і від системи людського бачення. Більш того, одні предмети відбивають світло, а інші його пропускають.

Коли 2 кольори розрізняються тільки відтінком, око може розрізнити порядку 128 відтінків [Новаковський - 120...200]. Коли два кольори розрізняються насиченням, око може розрізнити до 20 ступенів насичення, у залежності від відтінку. Кожному кольору відповідає додатковий колір – додавання основного з додатковим дає білий.

Стандартна відносна видимість [Новаковьский.88]

МКО 1924 році установило функцію відносної видимості v, що є стандартною чутливістю зору стандартного спостерігача МКО.

Приклад (повна в літ. Новаковський Табл.1.2)

λ, нм	v	
400	0.0004	
410	0.0012	
....		
530	0.862	максимум приходиться на 555 нм = 1
540	0.954	
550	0.995	
560	0.955	
....		
760	0.00006	

Власне це залежність відчуваються яркості монохроматичного джерела світла при постійній потужності випромінювання від довжини хвилі.

Потужність монохроматичного випромінювання, помножена на v, дає світлову (фотометричну) потужність цього випромінювання.

[Новаковський.88]

Зоровий апарат має інтегральну чутливість - реагує на суму всіх складових видимого спектра, то відчуття однакового кольору можуть створюватися при різних спектральних складах світла, що надходить в око. Два різних по складу спектрів світлових потоків, що створюють у спостерігача відчуття однакового кольору, називають метамеричною парою.

Психофізичні характеристики кольору

- яскравість кольору (світлового потоку або кольорової поверхні) розуміється значення світлового потоку (тобто потужності випромінювання, оцінена оком стандартного спостерігача МКО), випромінюваного в даному напрямку одиницею площі поверхні в межах одиничного тілесного кута. Яскравість виміряється в канделах на метр у квадраті.

- переважна (домінуюча) довжина хвилі кольору – розуміють довжину хвилі монохроматичного (спектрального) кольору 7l 4d, колірний тон якого такий же як у даного кольору.

- колорометрична чистота кольору (світлового потоку F) - розуміється відносний зміст монохроматичного світлового потоку F(7l 0), тобто

[Роджерс.89]

Психофізичні характеристики		Психофізіологічні характеристики
основна довжина хвилі	<----->	колірний тон
чистота	<----->	насиченість

яскравість <------> світлота

[Новаковський.88]

Явище дихроматизму: при нормальному зорі при кутовому розмірі предмета порядку 10...20 хвилин слабка чутливість до синьої частини спектра - два основних спектральних кольори 475 і 650 нм.

Наприклад, висота 1 метр - з відстані 172 метра тільки двоцвітний зір. - не відрізняє синього від зеленого, червоного від пурпурного.

Ще більш дрібні предмети людина сприймає без кольору – як сірі!!!

[Роджерс.89]

Білими виглядають об'єкти, що відбивають більш 80 відсотків світла ахроматичного джерела (ахроматичне джерело містить усі довжини хвиль у рівних кількостях - сприймається як джерело білого). Чорними виглядають об'єкти, що відбивають менш 3 відсотків світла. Все інше – сіре.

[Новаковський.88]

Суб'єктивні ефекти:

- послідовний і одночасний контрасти кольорів
- адаптація зору до яскравості = настроювання на діапазон яскравості
- час настроювання порядку 0,2 с. У темряві тривалість адаптації може збільшуватися до 10...15 хв.

([Роджерс.89] дає діапазон по яркості 10 510 0. Чутливість до відносного яркості 100...150)

- адаптація до кольору
- контрастність кольору

<u>Фізичні аспекти кольору</u>

[Hill.90]

Колір може бути заданий як комбінація

- основна довжина хвилі чи відтінок (hue - колір, відтінок)
- насичення (saturation) чи чистота (purity)

- яскравість (luminance)

[Роджер.88] Довжини хвиль

	Максвелл	МКО 31
R = 700 нм	630	700
G = 546 нм	528	546.1
B = 436 нм	457	435.8

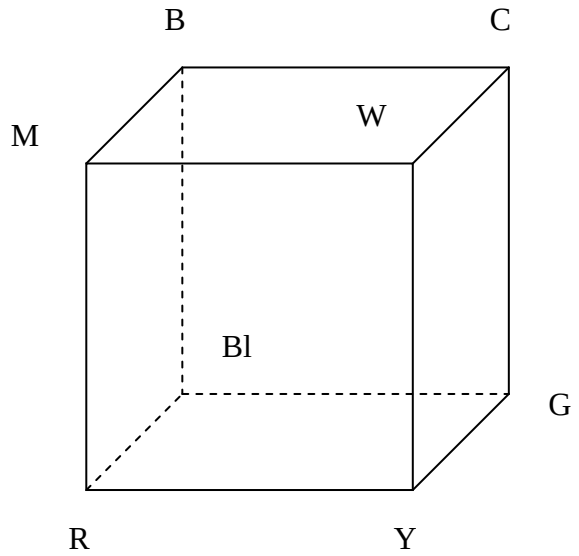
Роджер/ Новаковський приводять три закони Грассмана

- три стимули
- лінійна залежність кольору - колір є лінійна комбінація трьох стимулів
- безперервність колірного простору

3.2 КОЛІРНІ МОДЕЛІ ДЛЯ РАСТРОВОЇ ГРАФІКИ

При виборі колірної моделі визначається 3-D кольоровий координатний простір і в ньому 3-D простір, усередині якого кожен виведений на екран колір представляється крапкою.

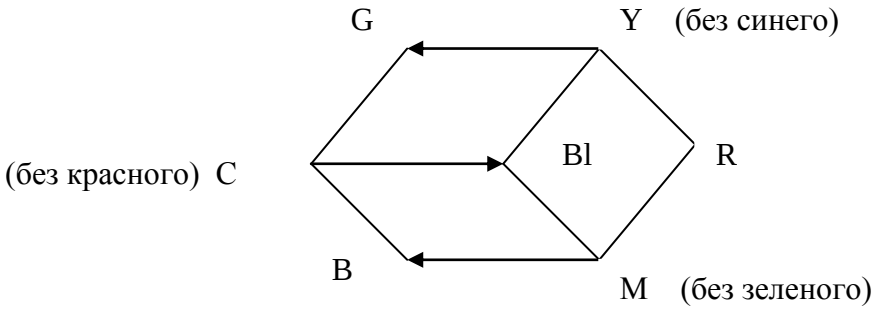
1. Колірна модель RGB



R - (red) красный
G - (green) зеленый
B - (blue) синий
M - (magenta) пурпурный
C - (cyan) голубой
Y - (yellow) желтый
W - (white) белый
Bl- (black) черный

У колірній моделі використовуються декартові координати (одиничний куб). На головній діагоналі лежать сірі кольори. Всі інші кольори виходять змішанням основних.

2. Колірна модель CMY



C, M, Y - кольори є додатковими до R, G, B (результат вирахування з білого), вони називаються основними субтрактивними кольорами.. Якщо RGB - пряма, то CMY - зворотна модель.

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

Стовпець одиниць у RGB відповідає білому кольору, у CMY - чорному. Якщо поверхня покрита блакитною фарбою, то червоне світло від її не відбивається.

Моделі CMY використовуються в пристроях одержання твердої копії, наприклад, копіювальних пристроях фірми Xerox і струминних плоттерах фірми Applicon.

3. Колірна модель YIQ

Використовується в комерційному кольоровому телевізійному віщанні і тісно зв'язана з кольоровою растровою графікою. Це деякий варіант кодування кольорів RGB, здійснюваний з метою підвищення ефективності їхньої передачі в ефір, а також для забезпечення сумісності з чорно-білим телебаченням.

Перетворення моделі RGB у модель YIQ задається в такий спосіб:

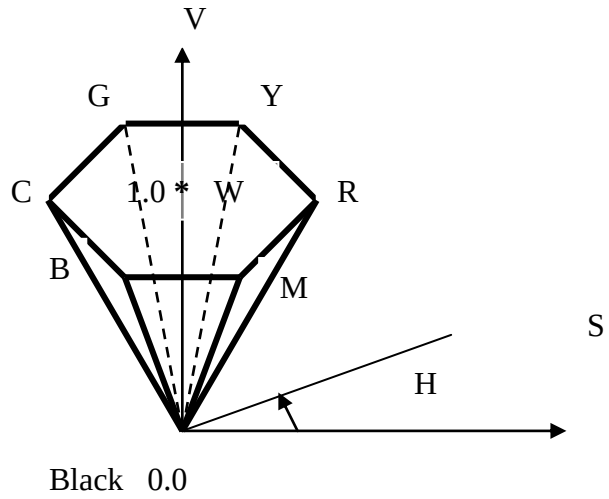
$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.30 & 0.59 & 0.11 \\ 0.60 & -0.28 & -0.32 \\ 0.21 & -0.52 & 0.31 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Такий спосіб перетворення прийнятий Національним комітетом з TV-стандартів (NTSC) США.

Координатата Y - це основний колір, спектральний розподіл енергії якого відповідає кривої спектральної чутливості ока. Y задає значення яркості, для її уявлення виділяється більша кількість розрядів (або більш широка смуга частот), отже, більш високий дозвіл, ніж для I і Q. У такий спосіб враховується основна властивість людського бачення: велика чутливість до змін яркості, ніж до змін колірного тону (I) або насиченості (Q).

4. Колірна модель HSV

HSV- модель запропонована Смитом і орієнтований на користувача. Модель заснована на інтуїтивно прийнятих художниками поняттях розбела, відтінку і тону.



H (hue) - тон
S (saturation) - насыщенность
V (value) - количество света

S=0, V<1 - серый цвет
V=1, S=1 - чистые цвета

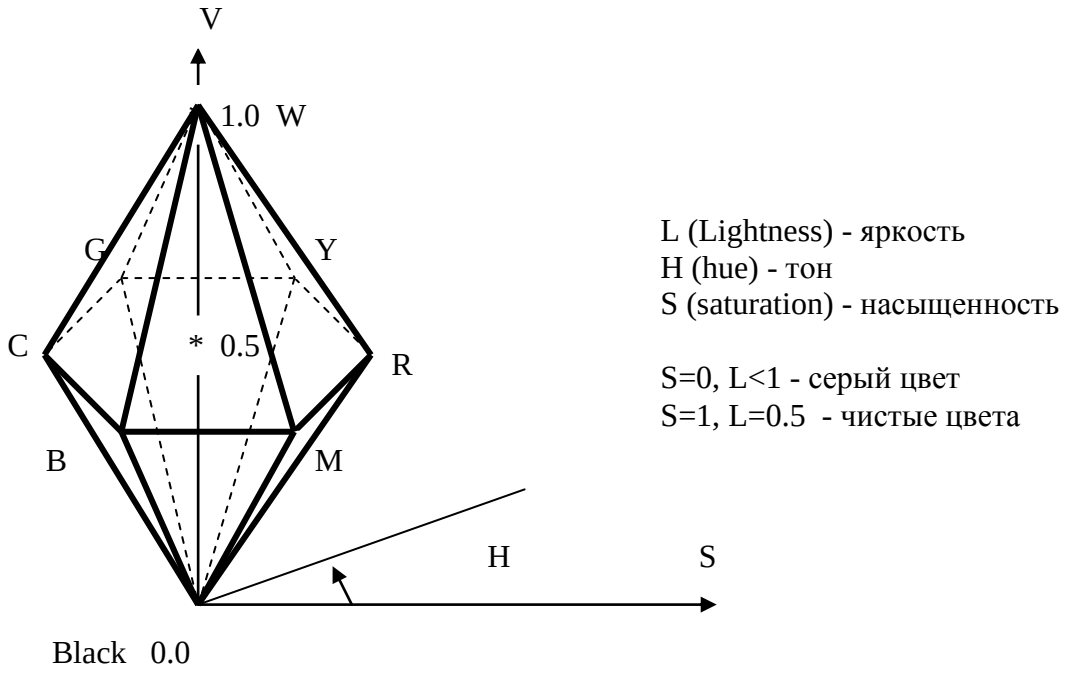
Простір - 6-гранний конус.

Верх - V=1 - кольори виражені з максимальною інтенсивністю.

Додаткові кольори розташовані друг проти друга (H=180 градусів).

Додавання білого пігменту відповідає зменшенню V без зміни S. Тони виходять зменшенням S і зменшенням V.

5. Колірна модель HLS



Застосована фірмою Tektronix. Є модифікацією моделі HSW.

Тон задається кутом повороту (H) навколо осі (для R - H=0).

3.3 МЕТОДИ ЗАФАРБУВАННЯ І МОДЕЛІ ОСВІТЛЕНОСТІ

Методи зафарбування вирішують задачу визначення інтенсивності світла або колірних компонентів у деякій довільній крапці поверхні об'єкта по відомим інтенсивностям у заданих крапках поверхні. У системах синтезу ИВО з полігональною апроксимацією поверхонь використовуються наступні методи зафарбування. Постійне зафарбування, при якій кожна крапка грані мають постійну інтенсивність, розраховану заздалегідь. Вважається, що таке зафарбування припустиме тільки для найпростіших систем. У зафарбуванні Гуро [1] інтенсивність у вершинах полігональної апроксимації розраховується по нормалі, отриманої усередненням нормалей до граней,

що містять цю вершину, з урахуванням деякої моделі освітленості. Інтенсивність внутрішніх крапок грані обчислюється за допомогою лінійної інтерполяції уздовж кожного ребра й уздовж кожного скануючого рядка. Зафарбування Фонга [2] припускає усереднення направляючих векторів вершин грані для кожної крапки і розрахунок освітленості в крапці по усередненому направляючому вектору і деякій моделі освітленості. Вважається, що кращі результати дає модель Фонга, хоча вона більш трудомістка. Але визначальними факторами, є, по-перше, прийнята модель освітленості і, у других, розміри зафарбовуваних поверхонь.

Перша модель освітленості, запропонована Фонгом [2], враховувала відображення розсіяного світла, а також дифузійного і дзеркального відображення крапкових джерел світла. Найбільш загальної з використовуваних моделей є модель Холу [1], [227], що вирішує задачу розрахунку рівня освітленості. Модель розділяє світло, що падає на тіло, на два класи: прямої світло - безпосередньо падаючий від джерел світла, і непряме висвітлення - світло, що надходить від інших тел. Світло "переробляється" такими способами: дзеркальне переломлення, дзеркальне відображення, дифузійне переломлення, дифузійне відображення.

У загальному випадку розрахунок інтенсивності EI вимагає інтегрування по довжині хвилі (по всьому діапазоні для формування монохромного зображення, або по трьох поддиапазонам для формування кольорового зображення), по поверхні об'єкта (для обліку дифузійних складових) і по шляху променя (для обліку розсіювання). Максимально повно модель освітленості Холу може бути врахована в методі трасування променів, де для кожного EI, по-перше, прослідкується траєкторія променя, що проходить через заданий EI з обліком "розмноження" при відображенні від об'єктів сцени і переломленні на границях розподіла середовищ, і, по-друге, розраховується інтенсивність EI відповідно до характеристик поверхонь і середовищ, через які пройшов промінь. Метод трасування поєднує етапи G, R, L процесу синтза, усуваючи етапи S і H. Однак висока

обчислювальна складність трасування унеможливлює її використання в реальному часі.

Крім того, трасування променів погано відтворює ефект дифузійного розсіювання світла, тому що не враховуються інтегральні по поверхні характеристики.

3.4 ІНТЕРПОЛЯЦІЯ В КОЛІРНОМУ ПРОСТОРІ

Необхідність в інтерполяції кольорів виникає принаймні в трьох випадках:

- при зафарбуванні методом Гуро;
- при побудові послідовності зображень, що передають ефект поступового "посилення" і "угасання";
- у випадку змішання кольору частково прозорої поверхні з кольором іншої поверхні.

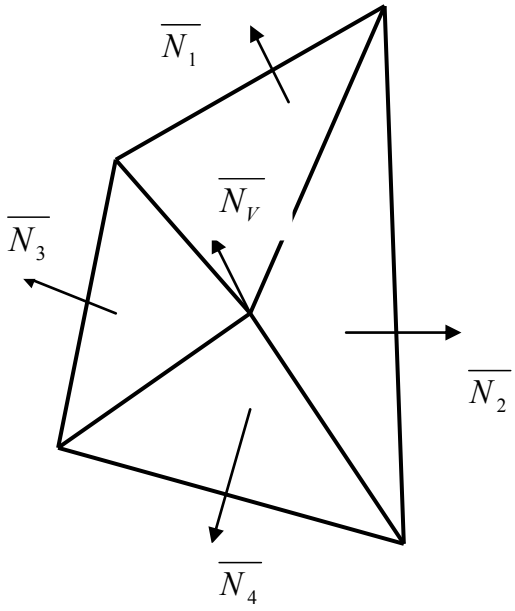
Результати інтерполяції багато в чому залежать від колірної моделі, у рамках якої виробляється інтерполяція кольорів. Якщо при перетворенні з однієї колірної моделі в іншу пряма лінія (що являє собою шлях інтерполяції), задана в одному колірному просторі, переходить у пряму лінію в іншому колірному просторі (моделі RGB, CMY, YIQ), результати інтерполяції в моделях будуть ідентичні. Однак, пряма лінія, задана в RGB, не переходить у пряму лінію, задану в HSV або HLS.

Для інтерполяції між кольорами, заданими тим самим колірним тоном, переважніше використовувати моделі HSV і HLS.

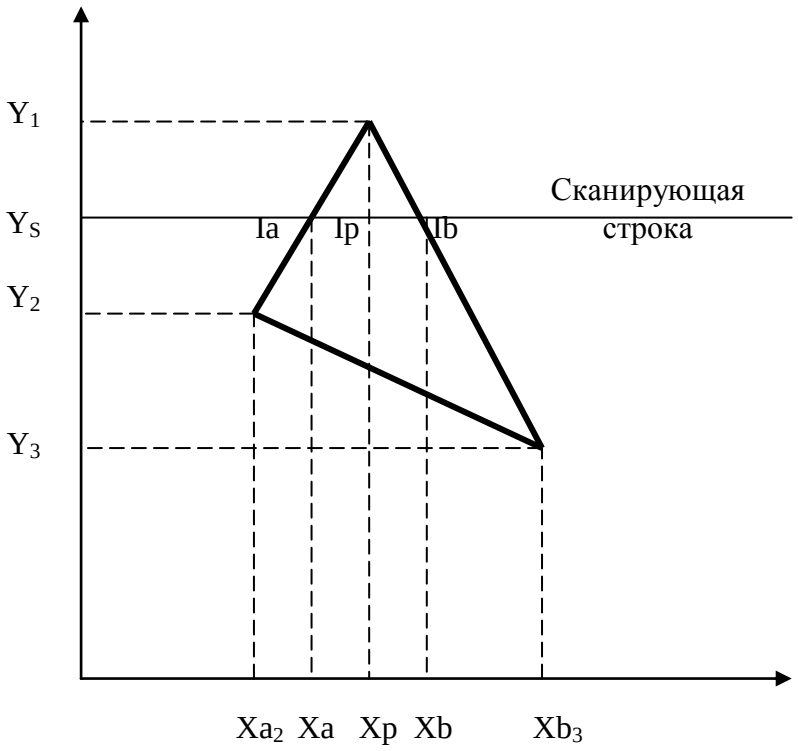
Якщо інтерполюються різні тони, то кращий ефект дає модель RGB.

Інтерполяція при зафарбуванні по методу Гуро

Метод зафарбування, заснований на інтерполяції інтенсивності (метод Гуро), дозволяє усунути дискретність зміни інтенсивності.



$$\overline{Nv} = \frac{\overline{N_1} + \overline{N_2} + \overline{N_3} + \overline{N_4}}{4}$$



$$I_a = I_1 \frac{Y_S - Y_2}{Y_1 - Y_2} + I_2 \frac{Y_1 - Y_S}{Y_1 - Y_2}$$

$$I_b = I_1 \frac{Y_S - Y_3}{Y_1 - Y_3} + I_3 \frac{Y_1 - Y_S}{Y_1 - Y_3}$$

$$I_P = I_a \frac{X_b - X_P}{X_b - X_a} + I_b \frac{X_P - X_a}{X_b - X_a}$$

Інтерполяція кольорів при зафарбуванні по методу Гуро виконується за чотири етапи:

1) Обчислюються нормалі для всіх поверхонь.

2) Визначаються нормалі у вершинах шляхом усереднення нормалей по всіх полігональних гранях, яким належить вершина.

$$\overline{N_v} = \frac{\sum \overline{N_i}}{n},$$ n – кількість граній.

3) Використовуючи нормалі у вершинах і застосовуючи довільний метод зафарбування, обчислюються значення інтенсивності у вершинах.

4) Кожен багатокутник зафарбовується шляхом лінійної інтерполяції значень інтенсивностей у вершинах спочатку уздовж кожного ребра, а потім і між ребрами уздовж кожного скануючого рядка.

Лекція 4
ВИКОРИСТАННЯ МЕТОДУ ВИПРОМІНЮВАЛЬНОСТІ. ОСОБЛИВОСТІ ОПИСУ СЦЕН

4.1 Метод випромінювальності

Метод випромінювальності ґрунтується на припущенні дифузійного відображення світла, що дозволяє попередньо, на етапі препроцесирування, розрахувати енергетичні параметри поверхонь відображуваних об'єктів і тим самим виключити етап розрахунку параметрів освітленості з робочої стадії й об'єднати етапи геометричних перетворень, розшарування/растрирування і видалення невидимих елементів. У зв'язку з цим метод випромінювальності є перспективним для побудови синтезуючих ИВО з високим ступенем реалістичності генеруємих зображень (рис.4.1).

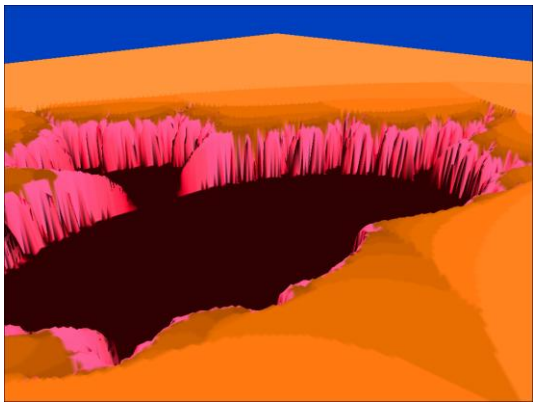


Рис.4.1. Приклад зображення

У результаті рішення системи формуються значення щільності потоку енергії, для кожної ділянки перераховуються в інтенсивності *Iko* ділянок, значення яких не залежать від положення спостерігача. Для побудови зображення значення інтенсивностей перераховуються у вузли апроксимації граней ділянками, а на етапі відрисовки виконується лінійна чи білинейна інтерполяція інтенсивностей найближчих до крапки перетинання лучачи візування з гранню вузлів.

Таким чином, метод випромінювальності складається з чотирьох обчислювальних кроків:

- розбивка поверхонь об'єктів синтезованої сцени на плоскі ділянки;
- розрахунок формфакторів для кожної пари ділянок; - рішення системи енергетичного балансу і розподіл інтенсивностей по вузлах апроксимації;
- відрисовка - визначення для кожного пікселя екрана крапки перетинання лучачи з конкретною ділянкою і розрахунок інтенсивності для даної крапки (етапи **T**, **G**/**S**/**W**).

При цьому перші три кроки можуть бути виконані на стадії препроцесирування і тільки останній - на робочій стадії.

Метод природно поширюється на кольорові зображення шляхом триразового рішення системи баланса для щільності випромінювання кожного колірного компонента. Як відзначається в [I5] метод випромінювальності дозволяє синтезувати зображення, що наближаються по якості до фотографічних, при цьому правильно відтворюються розподілені джерела світла, розподілені тіні, перенос кольору між близькими поверхнями.

Основна перевага методу випромінювальності в порівнянні з іншими підходами полягає в тім, що розраховані на перших трьох етапах значення інтенсивностей не залежать від крапки і напрямку візування і, отже, етап відрисовки може виконуватися незалежно і багаторазово для кожного необхідного положення спостерігача. До недоліків фотометричного характеру методу випромінювальності відносять неможливість обліку дзеркального відображення, так він заснований на дифузійному поширенні випромінювання, хоча в [I10] почата спроба включити недифузійне поширення енергії.

4.2 Облік поширення світла

1). Чисте дзеркальне відображення

Падаючий промінь, нормаль до поверхні і відбитий промінь лежать в одній площині. Кут падіння дорівнює куту відображення

r = l - 2<n, l>n

Is = Ilrs

l - одиничний вектор падаючого променя

n - одиничний вектор нормалі до поверхні

r - одиничний вектор відбитого променя.

Is- дзеркальний компонент освітленості

Il- інтенсивність джерела

rs- коефіцієнт дзеркального відображення *(*залежить від матеріалу поверхні)

2). Чисте дзеркальне переломлення

Проходження світла через границю двох середовищ. Падаючий промінь, нормаль до поверхні і переломлений промінь лежать в одній площині. Для кутів падіння і переломлення рівність

$$\frac{\sin \varphi}{\sin \psi} = \frac{h_2}{h_1} = h_{21}$$

де h_2 - коефіцієнт переломлення середовища 2 щодо вакууму;

h_1 - коефіцієнт переломлення середовища 1 щодо вакууму;

h_{21} - коефіцієнт переломлення середовища 2 щодо середовища 1;

Коефіцієнт переломлення середовища !- функція довжини хвилі світла.

Є ефект повного внутрішнього відображення. - При переході із середовища з великим h у середовище з меншим h при визначених кутах падіння відбувається повне внутрішнє відображення - для пари стекло - повітря цей кут 41.8 градуса.

t = hiti + (hitCi - ?(1+h2it(Ci - 1)))n

IT = ILRT

де t - одиничний вектор переломленого променя

CI - КОСИНУС КУТА ПАДІННЯ

hit - коефіцієнт переломлення середовища i (падаючий промінь) до середовища t (переломлений промінь)

3). Чисте дифузійне відображення

Падаючий промінь відбивається у всіх напрямках рівномірно. Модель Ламберта - рівень відбитого променя пропорційна косинусу кута падіння.

$$I_D = I_{LRDCOSA}$$

Id - дифузійний компонент освітленості

Il - інтенсивність крапкового!!! джерела світла вилученого на нескінченно велику відстань (усі промені від нього паралельно падають на поверхню тіла по напрямку вектора l)

rd - коефіцієнт дифузійного відображення, визначається характеристикою матеріалу, лежить у межах від 0 до 1

a - кут між нормаллю n і вектором l.

Якщо n і l одиничні, то

Id = Ilrd<n,l>

4). Чисто дифузійне переломлення

(приклад - кольоровий, напівпрозорий пластик)

Падаючий промінь попадає в заломлюючу середовище й у ній поширюється дифузно - в усі сторони. Рівень - залежить від косинуса кута падіння.

МОДЕЛЬ ФОНГА [Struss]

Ir = Il (*ra* + *rd*cosa + *rs*coshb)

I - інтенсивність джерела світла

Ir - інтенсивність відбитого від поверхні світла

ra - коефіцієнт відображення розсіяного світла

rd - коефіцієнт дифузійного відображення

rs - коефіцієнт дзеркального відображення

a - кут падіння

b - кут між відбитим компонентом і напрямком на спостерігача

h - показник відбитого компонента.

Компонента coshb дозволяє врахувати ефект швидкого загасання дзеркального компонента при відхиленні кута зору від кута відображення. Показник h (1 - 200) визначається видом поверхні. Чим ближче до ідеального дзеркала, тим більше. Для ідеального дзеркального відображення h = $.

Пропонується також враховувати відстань до поверхні від крапки спостереження. Якщо *d* - відстань від спостерігача до поверхні, а *k* - деяка константа, то [Фоли]:

$$Iy = Iya + \frac{I\,l}{d+k}\,(d < n,1 > + ys < r, y > h$$

4.3 Математичні основи методу випромінювальності

Метод випромінювальності [I1], [I2], [I3], [I4], заснований на припущенні про рівноважний розподіл енергії в замкнутому просторі. При цьому передбачається, що процес емісії світлової енергії і її відображень ідеально дифузні. Тобто, після відображення спрямованість променів губиться й освітленість тіл не залежить від положення спостерігача.

Нехай у сцені беруть участь два тіла S і D. Потік випромінювання тіла D визначається як:

Pdo = Pde + Pdr,

де:

Pdo [ут=дж сек$^{-1}$] - повний потік променистої енергії, що виходить від поверхні тіла *D*;

Pde - емісія тіла *D* - потік променистої енергії, випромінюваної тілом;

Pdr - потік відбитої від поверхні тіла *D* променистої енергії.

Відбитий потік енергії у свою чергу можна визначити як

$Pdr = \chi_d\, Pdi = \propto d\, \Psi_{s\text{-}d}\, Pso$,

де

Pdi - потік променистої енергії, що падає на тіло *D*;

χ_d - коефіцієнт відображення тіла *D*;

Pso - повний потік променистої енергії, що виходить від тіла *S*;

$\Psi_{s\text{-}d}$ - коефіцієнт опромінення тіла *D* щодо тіла *S*(коефіцієнт форми тіла), що показує, яка частка потоку енергії, випромінювана тілом S, попадає на D.

Тоді, для системи тіл *D, S* можна записати:

$Pdo = Pde + \chi_d\, \Psi_{s\text{-}d}\, Pso$,

$Pso = Pse + \chi_s \Psi_{s\text{-}d}\, Pdo$;

Для тіла довільної форми вираження для *Po* є інтеграл по поверхні тіла від щільності повного потоку випромінювання, а $\Psi_{s\text{-}d}$ - подвійний інтеграл по поверхнях взаємодіючих тел.

Вирішуючи отриману систему рівнянь можна визначити Pdo, Pso і, потім, перейти до інтенсивності в напрямку візування. Якщо припускати чисто дифузійне поширення променистої енергії, сила випромінювання не залежить від положення спостерігача і можна досить просто по Po розрахувати енергетичну яркість будь-якої крапки об'єкта.

Нехай моделюєма сцена містить S об'єктів Os, s=1,2,...,S (не обов'язково опуклих). Кожен об'єкт складається з Rs простих (опуклих) підоб'єктів Wsr, r=1,2,...,Rs. У свою чергу поверхня підоб'єкта апроксимована Qsr плоскими гранями Gsrq, q=1,2,...,Qsr, кожна з який покривається сукупністю плоских ділянок (шматків, patch) Pij, i=1,2,...,nsrq, j=1,2,...,msrq. Т.ч. моделюєма сцена представлена N плоскими ділянками Uk, k=1,2,....,N;

$N = S*Rs*Qsr*n_{srq}*m._{srq}$

Вважаючи, що щільність потоку променистої енергії вихідної від кожної ділянки постійна (як за часом, так і по поверхні), енергетичний баланс для ділянки k можна переписати як

$$Eko\,Sk = Eke\,Sk + \chi_K \sum_{j=1}^{N} Pj-k$$

$$Bk\ Sk = Ek\ Sk + \chi_K \sum_{j=1}^{N} Pj-k \qquad (1)$$

чи

де: *Bk = Eko* - щільність потоку енергії вихідної від поверхні k-го ділянки;

Ek = Eke - щільність потоку енергії випромінюваної (імітуємої) k-ым ділянкою;

Pj-k- потік енергії, що падає від j-го ділянки, на k-й;

Sk - площа k-го ділянки;

χ_d - коефіцієнт відображення k-го ділянки.

Позначення *Bk, Ek* звичайно використовуються в літературі по випромінювальності.

За законом Ламберта потік випромінювання від елементарної площадки dSj до елементарної площадки dSk дорівнює

$$d^2 P_{j-k} = B_j \frac{\cos\varphi_1 \cos\varphi_2}{\pi\rho^2} dS_j\, dS_k \qquad (2)$$

де *Bj* - щільність потоку енергії, що виходить від j-го ділянки;

ρ - відстань між елементарними площадками;

φ_1, φ_2 - кути між нормалями до dSk, dSj, відповідно, і напрямком випромінювання.

З обліком того, що щільність потоку випромінювання площадки постійна:

$$P_{j-k} = B_j \int_{Sk} \int_{Sj} \frac{\cos\varphi_1 \cos\varphi_2}{\pi\rho^2} dS_j\, dS_k \qquad (3)$$

Підставивши (3) у (1) і позначивши:

одержуємо для k-го ділянки:

$$F_{j-k} = \frac{1}{S_k} \int_{Sk} \int_{Sj} \frac{\cos\varphi_1 \cos\varphi_2}{\pi\rho^2} dS_j\, dS_k \qquad (4)$$

$$Bk = Ek + \chi_K \sum_{j=1}^{N} Fj-k\ Bj \qquad (5)$$

де: Fj-k - формфактор (коефіцієнт форми поверхні) k-го ділянки відносно і-го ділянки - частка потоку енергії, випромінюваної і-м ділянкою, що попадає на k-й.

Таким чином, енергетичний баланс у замкнутій системі з N поверхонь з постійної випромінювальностю і дифузійним поширенням енергії представляється системою рівнянь

$$Bk = Ek + \chi_K \sum_{j=1}^{N} Fj-k\ Bj \qquad k = 1,2...,N$$

або

$$\begin{bmatrix} -\chi_1 F_{1-1} & -\chi_1 F_{2-1} & -\chi_1 F_{3-1} \ldots\ldots & -\chi_1 F_{N-1} \\ -\chi_2 F_{1-2} & -\chi_2 F_{2-2} & -\chi_2 F_{3-2} \ldots\ldots & -\chi_2 F_{N-2} \\ -\chi_3 F_{1-3} & -\chi_3 F_{2-3} & -\chi_2 F_{3-3} \ldots\ldots & -\chi_3 F_{N-3} \\ \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots & \ldots \\ -\chi_N F_{1-N} & -\chi_N F_{2-N} & -\chi_N F_{3-N} \ldots\ldots & -\chi_N F_{N-N} \end{bmatrix} * \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ \ldots\ldots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ \ldots\ldots \\ E_N \end{bmatrix} \qquad (6)$$

Систему (6) можна спростити, якщо врахувати ряд додаткових факторів. По-перше, для ділянок, відбивна здатність яких $\chi_k = 0$, залишається тільки діагональний член, тобто *Bk=Ek*.

Як правило, це джерела світла, для яких частка відбитої енергії $\chi_k \Sigma Bj$Fj-

$$F_{j-k} = \frac{1}{S_k} \int_{Sk} \int_{Sj} H_{j-k} \frac{\cos\varphi_1 \cos\varphi_2}{\pi\rho^2} dS_j\, dS_k \qquad (7)$$

k багато менше частки емісії *Ek*.

По-друге, ряд ділянок не обмінюються енергією, тобто Fj-k=0:

- той самий ділянка (j = k);

- ділянки лежачі в одній площині (ділянки однієї грані);

- ділянки "невидимі" друг для друга (закриті іншими ділянками, чи розгорнуті в різні сторони). Для обліку "видимості" ділянок у вираження для формфактора звичайно вводиться множник Hj-k:

який приймає значення 1, якщо "лицьові" сторони ділянок звернений друг до друга і не перекриваються іншими об'єктами, і значення 0 - у противному випадку.

Тоді (6) приймає вид:

$$\begin{bmatrix} 1 & -\chi_1 F_{2-1} & -\chi_1 F_{3-1} & \ldots\ldots & -\chi_1 F_{N-1} \\ -\chi_2 F_{1-2} & 1 & -\chi_2 F_{3-2} & \ldots\ldots & -\chi_2 F_{N-2} \\ -\chi_3 F_{1-3} & -\chi_3 F_{2-3} & 1 & \ldots\ldots & -\chi_3 F_{N-3} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ -\chi_N F_{1-N} & -\chi_N F_{2-N} & -\chi_N F_{3-N} & \ldots\ldots & 1 \end{bmatrix} * \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ \ldots\ldots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ \ldots\ldots \\ E_N \end{bmatrix} \qquad (8)$$

причому, для деяких рядків (ділянки джерел світла) всі елементи, крім діагонального, будуть нульовими, а деякі рядки будуть містити значну частину нульових елементів і/чи права частина *Eke* буде дорівнює 0 (не випромінюючі поверхні).

Тому що

$$\chi_K < 1 \qquad u \qquad \sum_{j=1}^{N} Fj - k \le 1$$

для кожного k, сума недіагональних елементів кожного рядка строго менше 1 і, отже, система (8) добре обумовлена. У результаті рішення системи (8) формуються значення щільності потоку енергії, для кожної ділянки перераховуються інтенсивності I_{ko} ділянок, значення яких не залежать від положення спостерігача. Для побудови зображення значення інтенсивності перераховуються у вузли апроксимації граней ділянок, а на етапі отрисовки виконується лінійна і білинейна інтерполяція інтенсивностей найближчих до крапки перетинання лучачи візування з гранню вузлів.

Лекція 5
СИНТЕЗ ТРИМІРНОГО ОБ'ЄКТУ, АПРОКСИМОВАНОГО ВИПУКЛИМ БАГАТОГРАННИКОМ

Метою синтезу є розробка програмної системи для формування та відображення динамічного тримірного об'єкту. Послідовність виконання роботи наступна:

- апроксимація об'єкту 8 - 10 випуклими гранями;
- підготовка файлів вихідного опису тримірного об'єкту, апроксимованого випуклими гранями;
- розробка програми синтезу об'єкту;
- розробка інтерактивної системи управління переміщенням об'єкту (або точки спостереження) для тестування об'єкту.

5.1. Апроксимація об'єкту

Під об'єктом визуализації розуміється тіло у тримірному просторі, задане у своїй системі координат та єдине функціональне призначення. Об'єкт визуализації (складений об'єкт) поділяється на примітивні об'єкти, являють собою випуклі багатогранники, що мають єдине геометричне представлення та функціональне призначення. Топологія об'єкту визначає взаємне розташування примітивних об'єктів ,що складають його.

Для опису випуклого багатогранника застосовується полигональная апроксимація, т.е. поверхня об’єкта поділяється на кінцеве кількість граней. Грань представляється її ребрами,що обмежують та вершинами. Геометрія об'єкту задається координатами його вершин та орієнтацією граней у просторі, а його топологія визначає зв'язки між багатогранниками просторі.

5.2 Підготовка файлів вихідного опису тримірного об'єкту

Файл опису об'єкту включає кількість багатогранників, опис кожного багатогранника та, при необхідності, дерево топології об'єкту.

Опис багатогранника

Структура опису багатогранника включає:

- кількість граней;
- координати вершин граней;
- нормалі граней.

Грань може бути описана, наприклад, наступною структурою:

```
struct      {
   unsigned int num_edge,/*Номер грані*/
   priznak_edg,/*Ознака грані*/
   num_points,/*Кількість вершин*/
   points [16];/*Список номерів вершин*/
      struct {
            long int x, y, z;/*Тримірні координати вершини*/
            unsigned char color;/*Колір вершини*/
           }pnt [16];/*Масив структур вершин*/
}   edge;   /*Опис однієї грані*/
```

Через те, що реальні об'єкти, у загальному випадку, обмежені досить складними геометричними поверхнями, у якості першого етапу опису примітивного об'єкту виконується апроксимація його плоскими гранями.

Всі вершини та грані отриманого випуклого багатогранника нумеруються (для кожного примітивного об'єкту використовується своя нумерація), при цьому кількість граней та кількість вершин у кожній грані обмежена. Апроксимація примітивного об'єкту виконується проєктувальником вручну на підставі наявних креслень.

5.3 Синтез багатогранника

Процес синтезу включає наступні етапи:

1) Завантаження опису багатогранника.

2) Введення векторів положення обїекту та спостерігача.

3) Розрахунок матриць перетворення координат.

4) Обробка граней:

- аналіз орієнтації відносно постерігача;

- перетворення координат;

- 3d відсічення;

- проецирування;

- 2d відсічення;

- масштабування;

- вивід в дисплейний файл (відображення).

Приклад функції формування матриць перетворення координат:

```
void matr_fun(double psr, double ter, double gar,
                              double *matr, int pr)
{
 /*опис даних */

 double ps_s,ps_c;
 double te_s,te_c;
 double ga_s,ga_c;
 double psga_ss,psga_sc,psga_cs,psga_cc;

 /*обчислення синусів та косинусів */

 ps_c = cos(psr); ps_s = sin(psr);
 te_c = cos(ter); te_s = sin(ter);
 ga_c = cos(gar); ga_s = sin(gar);

 psga_ss=ps_s*ga_s;
 psga_cc=ps_c*ga_c;
 psga_sc=ps_s*ga_c;
 psga_cs=ps_c*ga_s;

 /*Обчислення матриці А об'єкт => мирова*/
 if (pr==1)
   {
   *(matr+0)=  ps_c*te_c;
   *(matr+1)=  psga_ss-psga_cc*te_s;
   *(matr+2)=  psga_cs*te_s+psga_sc;
   *(matr+3)=  te_s;
   *(matr+4)=  te_c*ga_c;
   *(matr+5)= -te_c*ga_s;
```

```
   *(matr+6)= -ps_s*te_c;
*(matr+7)=  te_s*psga_sc+psga_cs;
   *(matr+8)=  psga_cc-te_s*psga_ss;
   }

 /*Обчислення матриці В мирова => спостерігач */
 if (pr==2)
   {
   *(matr+0)=  te_c*ps_c;
   *(matr+1)=  te_s;
   *(matr+2)= -te_c*ps_s;
   *(matr+3)=  psga_ss - psga_cc*te_s;
   *(matr+4)=  ga_c*te_c;
   *(matr+5)=  psga_sc*te_s+psga_cs;
   *(matr+6)=  psga_cs*te_s+psga_sc;
   *(matr+7)= -ga_s*te_c;
   *(matr+8)=  psga_cc - psga_ss*te_s;
   }
}  // Кінець прикладуу
```

Лекція 6

ВИКОРИСТАННЯ МЕТОДУ ПРІОРИТЕТІВ ДЛЯ СИНТЕЗУ ТРИМІРНОГО ОБ'ЄКТУ, АПРОКСИМОВАНОГО ВИПУКЛИМИ БАГАТОГРАННИКАМИ

Для виконання синтезу необхідні:

- апроксимація об'єкту 8 - 10 випуклими багатогранниками;
- підготовка файлів вихідного опису тримірного об'єкту, апроксимованого випуклими багатогранниками;
- розробка програми синтезу об'єкту з використанням методу пріорітетів;
- розробка інтерактивної системи управління переміщенням об'єкту (або точки спостереження) для тестування об'єкту.

6.1 Побудова дерева топології об'єкту та формування пріоритетних списків

Побудова дерева топології об'єкту необхідна для синтезу об’єкта пріоритетним методом. Ідея методу висловлена Шумейкером, який запропонував заздалегідь формувати пріоритетні списки непорушних випуклих об'єктів сцени, розбиваючи простір сцени на субпростіри - кластери, що лінійно незалежні.

Ідея тут полягає у тому, що для кожного такого субпростіра заздалегідь можливо вірно визначити пріоритети об'єктів, які визначають порядок їхнього висновку на екран дісплею для правильного вилучення невидимих з точки зору спостерігача поверхонь у процесі синтезу зображення. Алгоритми статичного просторового сортування дозволяють за рахунок односторонньої видимості граней статически переставити їх так, щоб при будь-якому положенні спостерігача порядок обробки граней відповідав порядку їх потенційного закриття один одним. Порядок проходження від ближніх до

дальніх відносно положення спостерігача називають прямим пріоритетним порядком. У цьому випадку грані, що потрапили на обробку першими, не можуть перекриватися наступними гранями. При зворотному пріоритетному порядку (від дальніх до ближніх) кожна наступна грань потенційно закриває попередню. Пріоритетні системи відображення вилучають невидимі поверхні або шляхом послідовного накладення (при зворотному пріоритетному порядку), або за рахунок помітки вже зайнятих пикселов (при прямому пріоритетному порядку). Грані, що утворюють випуклий об'єкт, однопріоритетні. Наявність такої інформації значно зменшує обсяг обчислень на етапі синтезу зображення.

Дерево топології будується на основі розкладення простору розділяючими плоскістями на кластеры (рис.6.1a), у кожному з яких порядок відображення багатогранників визначається однозначно. У вузлах дерева задаються нормалі розділяючих плоскістей, листя дерева містять пріоритетні списки відображення багатогранників (рис.6.1b).

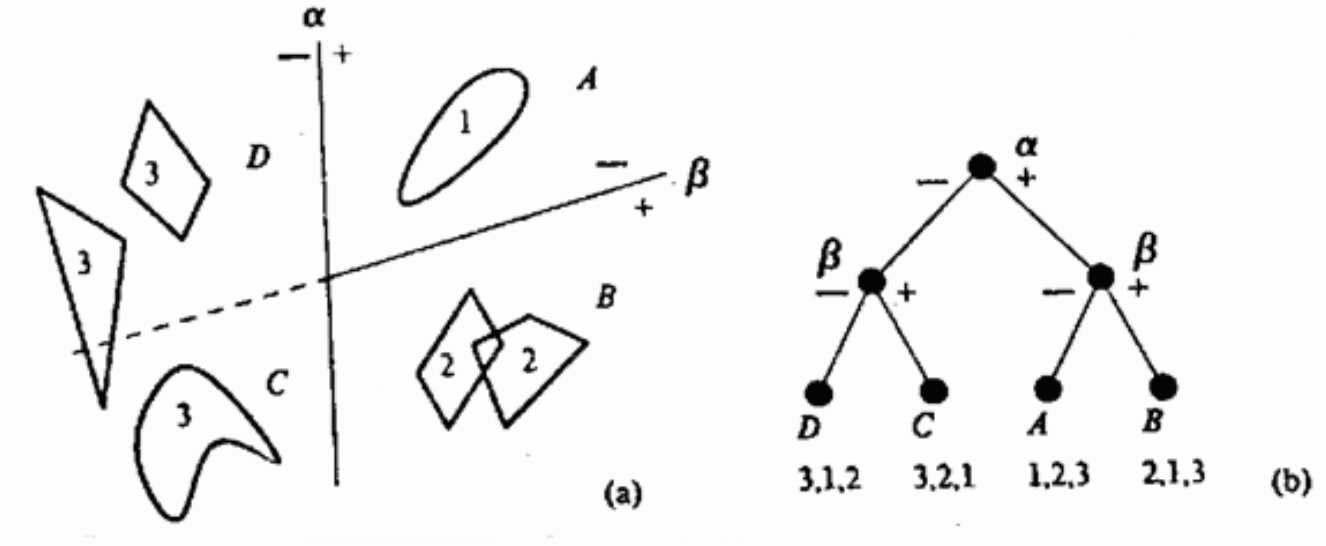


Рис.6.1. Приклад розділу простору на кластери (a) та симетричне дерево пріоритетів (b)

Для несиметричного дерева можна використати наступний опис (нормаль задана початковою та кінцевою точкою):

struct

```
    {
        int N1,ST1,N2,ST2;
        signed long X1,Y1,Z1;
        signed long X2,Y2,Z2;
    } VERT[NVMAX];
```

6.2 Метод пріоритетів

Програма синтезу випуклого багатогранника розглянута у лабораторній роботі N9 з курсу 'Комп'ютерна графіка'. Для синтезу методом пріоритетів необхідно додати зовнішній цикл по багатогранникам, а, отже, виконати вибірку пріоритетного списку багатогранників для кожної точки спостереження. Аналіз кластера, у якому знаходиться спостерігач виконується шляхом перевірки знаку скалярного добутку вектору спостереження на нормаль що поділяє грані. Аналіз виконується у системі координат об'єкту, тому точка спостереження, задана у мировій системі координат, перетворюється у систему координат об'єкту.

Фрагмент програми для вибору списку багатогранників:

```
struct
    {
    int N1,ST1,N2,ST2;
    signed long X1,Y1,Z1;
    signed long X2,Y2,Z2;
    } VERT[NVMAX];
struct
    {
    char namef[20];
    } FLIST[NSMAX];
int main()
{
  ...

  / *читання файлу даних*/
cikl='Y';
while(cikl=='Y')
{
// Введення векторів положення точки спостереження та об'єкту
```

```
      matr_a=&a[0][0]; matr_b=&b[0][0];
      matr_d=&d[0][0]; matr_c=&c[0][0];
  matr_fun(pso,teo,gao,pss,tes,gas,        //обчислення
         matr_a,matr_b,matr_d,matr_c);   //матриць
  fun_smn(x_o,y_o,z_o,x_s,y_s,z_s,matr_b,matr_c);//зміщення
// переведення точки спостереження в систему об'єкту
  x_s1=mul_x_y(x_s,d[0][0])+mul_x_y(y_s,d[0][1])
              +mul_x_y(z_s,d[0][2])-x_o;
  y_s1=mul_x_y(x_s,d[1][0])+mul_x_y(y_s,d[1][1])
              +mul_x_y(z_s,d[1][2])-y_o;
  z_s1=mul_x_y(x_s,d[2][0])+mul_x_y(y_s,d[2][1])
              +mul_x_y(z_s,d[2][2])-z_o;
// спуск по дереву - вибірка номера списку
ivs=1;
    sts=1;
  while(sts!=0)
       {
        iv=ivs-1;
        XX1=VERT[iv].X1;
        YY1=VERT[iv].Y1;
        ZZ1=VERT[iv].Z1;
        XX2=VERT[iv].X2;
        YY2=VERT[iv].Y2;
        ZZ2=VERT[iv].Z2;
        S=(XX1-x_s1)*(XX2-XX1)+(YY1-y_s1)*(YY2-YY1)+(ZZ1-z_s1)*(ZZ2-ZZ1);
        if(S<0)
             { sts=VERT[iv].ST1; ivs=VERT[iv].N1;              }
             else
             { sts=VERT[iv].ST2; ivs=VERT[iv].N2;               }
        }
         ils=ivs;
         il=ils-1;
// выбірка списка номер il довжиною DLS
  for(k=0;k<DLS+1;k++) //основний цикл
  {
  ...
  // синтез багатогранника
  ...
  } //кінець основного циклу
  ...
} // кінець програми
```

Лекція 7

ВИКОРИСТАННЯ МЕТОДУ ТРАСУВАННЯ ПРОМЕНІВ

При зворотньому трасуванні прменів відстежуються тільки промені, які попадають в око спостерігача. Це виконується слідуючим чином. З ока спостерігача черезкожний пиксел екрану площини в сцену, що синтезується,пропускається промінь, і після цього він відстежується у зворотному напрямку. Коли промінь наштовхується на поверхню, інтенсивність відповідного пиксела визначається освітленістю найближчої точки пересічення променя з поверхнею. Якщо на шляху променя не виникає жодного об'єкту, слідує брати освітленість навколишнього простору (неба, землі чи]спеціально спроектованой однорідної моделі поверхні).

аким чином, метод трасировки променей полягає в моделюванні проходження променей в рамках геометричної оптики.

Краще всього він підходить для сцен, що не містять дифузних поверхонь, бо обмін енергії між ними моделюється складно.

Приклад обчислення параметру ***t*** та аналізу видимості об'єкту

```
void main()
{

      Point   nabl;                 //Координати спостерігача
      Angl    nabl_r;       //Кутові координати спостерігача
      Point   obj;                  //Координати об’єкту
      Angl    obj_r;        //Кутові координати об’єкту
      Point   s_ray;                      //Поч. значення променя
      Point   k_ray;        //Коеф. променя
      Point   obj_b;        //Межі об’єкту
      Point   obj_h;        //Крок розтрощування
      int     rez;          // Ознака видимості об'єкту
      int           i,j;
      double  xp,yp,zp;       //Координата пиксела в системі спостерігача
double  xp_m,yp_m,zp_m;  //                  в мировій системі
      double  kx,ky,kz;       // коефіцієнти напрямку променя
      double  dl;             // довжина вектору
      double  x_e,y_e;        //координата пикселу на екрані
      double  hx_e,hy_e;      //крок по екрану

      double  matr_e[9];      //з спостерігача в мирову
```

```
        double  matr_d[9];      //з мирової в об'єкт
        double  matr_a[9];      //з об'єкту в мирову
        clrscr();

// введення положень та габаритів
  in_nabl_tr( &nabl, &nabl_r,&obj,&obj_r,&obj_b,&obj_h);
// РОЗРАХУНОК МАТРИЦЬ ПЕРЕТВОРЕНЬ

  matr_fun(nabl_r.ps,nabl_r.te,nabl_r.ga,matr_e,1);
  matr_fun(obj_r.ps,obj_r.te,obj_r.ga,matr_d,2);

// цикл перебора пикселей екрану
 for (j=0; j<Ny; j++)
   {
   for (i=0; i<Nx; i++)
     {
     hx_e= (double)Xe*2./(double)Nx;
     hy_e= (double)Ye*2./(double)Ny;
     x_e = -(double)Xe + hx_e*(double)i;  // у плошині екрану
     y_e = -(double)Ye + hy_e*(double)j;

     xp = De;   // у системі спостерігача
     yp = y_e;
     zp = x_e;

     matr_fun(nabl_r.ps,nabl_r.te,nabl_r.ga,matr_e,1);
     matr_fun(obj_r.ps,obj_r.te,obj_r.ga,matr_a,1);

     // в мирової системі
     xp_m = xp*matr_e[0] + yp*matr_e[1] + zp*matr_e[2] + nabl.x;
     yp_m = xp*matr_e[3] + yp*matr_e[4] + zp*matr_e[5] + nabl.y;
     zp_m = xp*matr_e[6] + yp*matr_e[7] + zp*matr_e[8] + nabl.z;
// розрахунок коефіцієнтів
     dl = sqrt( (xp_m-nabl.x) * (xp_m-nabl.x) +
                          (yp_m-nabl.y) * (yp_m-nabl.y) +
                          (zp_m-nabl.z) * (zp_m-nabl.z) );
     kx = (xp_m-nabl.x)/dl;
     ky = (yp_m-nabl.y)/dl;
     kz = (zp_m-nabl.z)/dl;

     s_ray.x=xp_m;
     s_ray.y=yp_m;
     s_ray.z=zp_m;
     k_ray.x=kx;
     k_ray.y=ky;
     k_ray.z=kz;

     rez = vid_t(&s_ray, &k_ray, &obj_b , &obj_h, &obj, matr_d);

     }
   }
        while(getch()!='q');
```

```
}
/*****************************************************************
 * Name:              vid_t
 * Title:      Розрахунок параметрів пересічення
 *
 * Descr:      Виконується розрахунок параметрів t для всіх площин
 *
 * Proto:      int  vid_t(Point *ps, Point *ks, Point obj_b,
 *                    Point obj_h, Point obj, double matr_d)
 * Param:
 *            Point  *ps    -  початок променя
 *            Point  *ks    -  коеф. наклону променя
 *            Point  obj_b  -  межі об'єкту
 *            Point  obj_h  -  крок розтрощування об'єкту
 *            Point  obj    -  лінійна координата об'єкту
 *            double matr_d -  матриця: мирова->об'єкт
 *
 * Return:             0 - об'єкт не видний
 *            1 - об'єкт видний
/*****************************************************************/
int  vid_t(Point *ps, Point *ks, Point *obj_b , Point *obj_h,
                        Point *obj, double matr_d[9])
{
int k,j,i,n_i,n_j,n_k;
int ig,ng;

double m_nx[6]= {-1., 0., 0., 1., 0., 0.};
double m_ny[6]= {0., -1., 0., 0., 1., 0.};
double m_nz[6]= {0., 0., -1., 0., 0., 1.};

double m_x1[6],m_y1[6],m_z1[6];  // початкова точка грані
// double nx,ny,nz;            // напрямок вектору грані
double Kx,Ky,Kz;            // напрямок вектору визирування
double xv,yv,zv;            // координата точки визирування
double xv_o,yv_o,zv_o;      // координата точки визирування (СК об'єкту)
double kx_o,ky_o,kz_o;      // вектор визирування (СК об'єкту)
double x_t,y_t,z_t;         // координата точки пересічення
double d,dev;
double div,t,t_in,t_out,t_o;
double t_in_k,t_out_k;
int rez;
int one_hit;
int g_in,g_out;             // номер площини входу та виходу променя
int g_in_k,g_out_k;         // номер площини входу та виходу променя

// перезапис точки визирування
xv = ps->x;
yv = ps->y;
zv = ps->z;
Kx = ks->x;
Ky = ks->y;
Kz = ks->z;
```

```
printf("\n Прийняли точку %lf %lf %lf коеф. %lf %lf %lf",
                        xv,yv,zv,Kx,Ky,Kz);
// координати точки визирування у  системі об’єкту
  xv_o = xv*matr_d[0] + yv*matr_d[1] + zv*matr_d[2] + obj->x;
  yv_o = xv*matr_d[3] + yv*matr_d[4] + zv*matr_d[5] + obj->y;
zv_o = xv*matr_d[6] + yv*matr_d[7] + zv*matr_d[8] + obj->z;
  kx_o = Kx*matr_d[0] + Ky*matr_d[1] + Kz*matr_d[2];
  ky_o = Kx*matr_d[3] + Ky*matr_d[4] + Kz*matr_d[5];
  kz_o = Kx*matr_d[6] + Ky*matr_d[7] + Kz*matr_d[8];
  printf("\n У системі об’єкту %lf %lf %lf коеф. %lf %lf %lf",
                              xv,yv,zv,Kx,Ky,Kz);

  // Визначення видимості об'єкту
  t_in  = -Fmax;
  t_out =  Fmax;
  t_o = Fmax;
  ng = 6;
  one_hit = 0;
  rez = 0;

  m_x1[0]=m_x1[1]=m_x1[2] = 0;
  m_y1[0]=m_y1[1]=m_y1[2] = 0;
  m_z1[0]=m_z1[1]=m_z1[2] = 0;
  m_x1[3]=m_x1[4]=m_x1[5] = obj_b->x;
  m_y1[3]=m_y1[4]=m_y1[5] = obj_b->y;
  m_z1[3]=m_z1[4]=m_z1[5] = obj_b->z;

  g_in = -1;
// g_out = -1;
  for (ig=0; ig<ng; ig++)
    {
    // знаменник t
    div =  m_nx[ig]*kx_o + m_ny[ig]*ky_o + m_nz[ig]*kz_o;
    d  = -(m_nx[ig]*m_x1[ig] + m_ny[ig]*m_y1[ig] + m_nz[ig]*m_z1[ig]);
    dev = -( m_nx[ig]*xv_o + m_ny[ig]*yv_o  + m_nz[ig]*zv_o + d );

    if (f_abs(div)>Fmin)
      {
      t = dev / div;

      printf("\n ig=%d t=%lf div=%lf",ig,t,div);

      if (div<0)
        {
if (t>t_in)
          {
          t_in=t;
          g_in = ig;
           }
         }
       else
```

```
          {
           if (t<t_out)
             t_out=t;
           }
          }
        else
          { //Промінь суворо паралелен грані
          if(dev < 0.0)
           {
            t_in  = 10.0;
            t_out =  0.0;
            break;
           }
          }

      } // кінець циклу по граням
    if ((t_in<t_out))
      {
      rez = 1;
      one_hit = 1;
      }

if (one_hit == 0)  //Немає пересічення променя з об'єктом
  {
   rez = 0;
   }
if (t_in <= 0.0)     //Промінь пересікається, але ззаду
  {
   rez = 0;
   }
printf("\n Результат видимості %d t_in=%lf t_out=%lf",rez,t_in,t_out);
if ( rez != 0 )
  {
  // цикл по граням
         for (ig=0; ig<ng; ig++)
           {
           // знаменник t
           div =  m_nx[ig]*Kx + m_ny[ig]*Ky + m_nz[ig]*Kz;
           d = -(m_nx[ig]*m_x1[ig] + m_ny[ig]*m_y1[ig]
                              + m_nz[ig]*m_z1[ig]);

            if (f_abs(div)>Fmin)
              {
              t = -( m_nx[ig]*xv_o + m_ny[ig]*yv_o
                  + m_nz[ig]*zv_o + d) / div;

              if (div<0)
                {
                 if (t>t_in_k)
                   {
                    t_in_k=t;
                    g_in_k=ig;
```

```
                    }
                  }
                else
                  {
                   if (t<t_out_k)
                     {
                     t_out_k=t;
                     g_out_k=ig;
                     }
                   }
                 }
              } // кінець циклу по граням
             if ((t_in_k<t_out_k))
               {
                printf("\n видний кубик i= %d j=%d k=%d t=%lf",
                      i,j,k,t_in_k);
                one_hit = 1;
                }
}
           t_in_k = t_out_k;
           }
         // Аналіз завершення об'єкту
         while(f_abs(t_out_k - t_out) < Fmin );
  }
return(rez);
} //кінець процедури
```

ЧАСТИНА 2

АРХІТЕКТУРА СИСТЕМ КОМП’ЮТЕРНОЇ ГРАФІКИ

Лекція А1

АНАЛІЗ ПІДХОДІВ ДО ОРГАНІЗАЦІЇ ГРАФІЧНИХ СИСТЕМ РЕАЛЬНОГО ЧАСУ

Аналіз основних підходів до побудови синтезуючих ГСРЧ виконується з погляду оцінки системи по ступені реалізму представлення сцени в базі даних і режиму реального часу процесу синтезу кадру зображення сцени. Синтез зображень у реальному часі припускає зміну кадрів з частотою 30 разів у секунду, що дозволяє здійснювати плавну зміну зображення відповідно до зовнішнього впливу.

На цей час у галузі інформатики найбільш динамічно розвиваються комп'ютерні графічно-інформаційні технології. Вони невпинно розширюють свою методологічну основу, інструментальну базу й сферу застосування, охоплюючи все більш широке коло найрізноманітніших галузей життєдіяльності людини. При цьому основним функціональним реалізатором таких технологій виступає комп'ютерна графіка, що є найвидовищнішою багатофункціональною складовою цих технологій, найлегше сприймається та найшвидше обробляється (в інформаційному плані) й засвоюється людиною, а, головне, - повною мірою відповідає природним психологічним особливостям сприйняття людиною навколишнього середовища.

Підвищений інтерес з боку спеціалістів різних фахових галузей до синтезу комп'ютерних зображень як окремого самостійного напрямку інформаційних технологій, який найбільш інтенсивно розвивається в наш час, пояснюється найвищою їх інформативністю порівняно з іншими носіями інформації. Інформація, що міститься у зображеннях, подається у найбільш концентрованій формі. Одночасно ж ця інформація є і найдоступнішою для сприйняття й аналізу за обмежений проміжок часу. Окрім того, для її сприйняття отримувачеві інформації, поданої у графічній формі (тобто у вигляді певного зображення), достатньо мати відносно невеликий обсяг спеціальних знань й загалом мати здібності звичайної нормальної людини.

Намагання візуалізувати оброблювану інформацію спостерігається практично у всіх без винятку сферах діяльності людини. Тому одночасно з появою й початком практичного використання ЕОМ виникла також й проблема подання сукупності даних, що є результатом протікання певних процесів, у вигляді зображень встановленої структури. Спочатку це виконувалось виключно програмним чином, що вимагало від фахового спеціаліста ще й певної кваліфікації у галузі програмування. Тому комп'ютерна графіка розвивалась в основному у технічних галузях, де таких спеціалістів було чимало.

З появою та поширенням персональних комп'ютерів та відповідного програмного забезпечення комп'ютерна графіка стала доступним інструментальним засобом широкого кола спеціалістів багатьох галузей, які нерідко абсолютно не пов'язані ані з технікою, ані безпосередньо з програмуванням. Збільшення обсягів пам'яті й швидкості обробки інформації у персональних ЕОМ створення складних багатокомпонентних й багатофункціональних відеокомплексів з широким набором різноманітних програм комп'ютерної графіки, можливість роботи з ними у діалоговому режимі з обов'язковою реалізацією умов "дружнього" інтерфейсу відповідно із розв'язуваними задачами конкретної предметної галузі сприяють подальшому розширенню практичного використання методів і засобів комп'ютерної графіки, її вдосконаленню і розвитку.

Швидке розширення функціональних можливостей сучасної обчислюваної техніки створило базу для подальшого розвитку й вдосконалення систем комп'ютерної графіки, що забезпечують відображення динамічних сюжетів, де зображення послідовно змінюють одне одного за визначеним сценарієм. Серед таких систем прийнято виділяти три групи:

- системи графічного (імітаційного) моделювання, основною задачею яких є подання (візуалізація) процесів у фізиці, хімії, астрономії, медицині тощо;

- системи імітації динамічних ситуацій (наприклад, динамічні тренажери);
- системи отримання двовимірних та тривимірних наочних зображень різноманітних об'єктів для телебачення й кіно з наступною їх компіляцією за певним сценарієм у вигляді, наприклад, рекламних комп'ютерних фільмів.

Будь-які системи комп'ютерної графіки відтворюють відібрану й певним чином оброблену інформацію про процес або об'єкт у вигляді синтезованих зображень на екрані дисплея. На відміну від фотографічних, телевізійних, оптико-електронних та будь-яких інших систем візуалізації зображень у системах комп'ютерної графіки джерелом вхідної інформації є не самі фізичні процеси або відтворювані об'єкти, а їх відповідні математичні (вірніше, геометричні) моделі. Ці моделі у загальному випадку являють собою упорядковану сукупність даних, числових характеристик, вербальної інформації, параметрів, математичних і логічних залежностей, що визначають структуру, властивості, взаємозв'язки й відношення між окремими елементами й складовими частинами об'єкта, а також між самим об'єктом і його оточенням. Після введення конкретних значень параметрів система комп'ютерної графіки на основі загальної моделі об'єкта й заданих умов візуалізації синтезує конкретне зображення й відтворює його на екрані дисплея.

Отож, центральним компонентом будь-якої системи комп'ютерної графіки є геометрична модель об'єкта (процесу або явища). Тобто комп'ютерна графіка має безпосередній взаємозв'язок з геометричним моделюванням. Проте на цей час література з шкільної інформатики, що розглядає комп'ютерну графіку, містить дуже обмежене за своєю суттю означення комп'ютерної графіки як окремої самостійної науково-практичної дисципліни (напрямку) та розділу інформатики як навчального предмету, навіть не згадуючи про геометричну модель. Так, найпоширенішим з таких означень є таке: "Комп'ютерна графіка - це створення й обробка зображень

(малюнків, креслень тощо) за допомогою комп'ютера". Тобто, до цих пір у школі комп'ютерна графіка розглядається лише з позицій створення комп'ютерних малюнків (зображень) на екрані дисплея за допомогою певного набору інструментальних засобів, найпоширенішими з яких є графічний редактор й маніпулятор "миша". Але ж це є найнижчий, можна сказати, досить примітивний рівень практичного застосування комп'ютерної графіки. І ним далеко не обмежуються можливі напрямки використання й вивчення можливостей комп'ютерної графіки та її апарату - методологічного, математичного, алгоритмічного, програмного, інструментального.

Враховуючи той факт, що означення будь-якої галузі зумовлює її інструментальну базу, область застосування й напрямки подальшого розвитку, та визнаючи обмеженість наведеного вище означення комп'ютерної графіки, необхідно сформулювати найпридатніше з методологічних позицій означення й запровадити його у шкільну інформатику. На цей час відомі кілька таких означень. Серед них таким, що найбільше відповідає задачам шкільної інформатики, є, на наш погляд, таке: "Комп'ютерна (машинна) графіка - це створення й маніпуляція на екрані дисплея графічними зображеннями об'єктів, процесів або явищ, що представлені у вигляді певних комп'ютерних геометричних моделей".

Тобто, будь-яке зображення на екрані дисплея - це результат комп'ютерної обробки тієї або іншої геометричної моделі об'єкта. Отже, геометрична модель є первинною відносно будь-якого комп'ютерного зображення й створюється заздалегідь (програмно) або ж синхронно із побудовою певного зображення на екрані дисплея в інтерактивному режимі.

1.1 Аналіз вимог до систем синтезу високореалістичних зображень реального часу

В даний час з'явилася можливість говорити про широке застосування високореалістичних систем синтезу зображень, що працюють у режимі

реального часу. Джерелами розробки і використання таких систем, як відомо, з'явилися авіаційне і космічне тренажеробудування. Поступово до комп'ютерної графіки починали виявляти усе більший інтерес фахівці в області проектування, архітектури, медицини, навчання, інформатики. Значну роль у цьому процесі зіграло бурхливе впровадження персональних комп'ютерів, графічних станцій і графічних співпроцесорів, що привело до визначеної стандартизації обчислювальних засобів і дозволило комп'ютерній графіці стати невід'ємною частиною будь-якого інформаційного середовища. Область застосування багато в чому визначає вимоги до характеристик систем синтезу, особливо, до тимчасового. Для систем реального часу, наприклад, тренажерних систем, найбільш критичним є реакція системи на зміну зовнішнього впливу. При цьому обчислювальна система повинна мати високу продуктивність, що не може бути забезпечена стандартними засобами. Ця обставина спонукає до розробки нових алгоритмів і спеціалізованих пристроїв, що у сукупності зі стандартними засобами дозволяють досягти необхідного ступеня адекватності зображення реальній обстановці.

Одним з критеріїв оцінки якості графічної системи є ступінь адекватності синтезованого зображення до реальної обстановки. При оцінці ступеня відповідності використовують 3 рівні подоби:

1. Фізична подоба.

Установлюється на рівні трьох груп характеристик: геометричних, яркостних, часових.

2. Фізіологічна подоба.

Установлюється на рівні зорових відчуттів і зв'язана з можливостями зорового апарата.

3. Психологічна подоба.

Припускає, що по загальному сприйняттю синтезоване зображення й оригінал є схожими, тобто забезпечують формування в спостерігача цілком визначеного судження про реальний чи створений сюжеті.

Умови функціонування ГСРЧ визначають розбивка процесу рішення задачі синтезу зображення візуальної обстановки на дві стадії:

- попередня стадія, на якій виконуються операції опису даних про сцену, що повинна бути синтезованою;

робоча стадія, на якій власне і здійснюється задача синтезу зображення сцени у відповідності з вимогами умов подоби.

Найважливіше розходження цих стадій полягає в тім, що попередня стадія не має істотних часових обмежень, тому що база даних відображуваної сцени розробляється до виконання польотних завдань на тренажері і готується тільки один раз. З іншого боку, час синтезу одного кадру зображення на робочій стадії істотно обмежений умовами психофізіологічної подоби, що визначають адекватність сприйняття динаміки переміщення об'єктів сцени в модельному просторі й у реальних умовах.

1.2 Вимоги до математичної моделі апроксимації сцени

Підвищення ступеня реалізму висуває підвищені вимоги до моделей, яки застосовуються для опису сцени. Серед необхідних складових частин, що повинні бути представлені в моделі, можна назвати наступні:

- основні елементи сцени і їхні взаємини;

- геометрія сцени, тобто просторове розміщення, форма, розміри окремих складових сцени;

- топологія сцени, тобто інформація про взаємне розташування і зв'язок між компонентами сцени;

- специфічні для розглянутого застосування дані, наприклад, електричні чи механічні характеристики;

- описовий текст.

У загальному випадку модель складається з прикладної структури даних і набору процедур прикладної програми для підтримки цієї структури.

Підготовка опису синтезованої сцени.

Побудова і реалізація математичної моделі візуальної обстановки в комп'ютерних генераторах зображень можуть бути виконані багатьма способами, заснованими на різних методах і алгоритмах представлення графічних даних і генерації зображень.

Основний зміст етапу попередньої підготовки полягає в наступному. Створення і редагування графічної бази даних заданої сцени. Відповідно до обраних графічних примітивів, способами опису й обробки об'єктів сцени на даному етапі визначаються способи їхнього представлення в пам'яті і набір операцій над ними. Створюється і наповняється даними про відображувану сцену графічна база даних. Редагування даних графічної бази необхідно тільки при зміні геометричних параметрів об'єктів і/чи енергетичних властивостей їхніх поверхонь.

Основні моделі опису геометрії синтезованої сцени:

1) Поверхневі.

2) Моделі суцільних тіл: воксельні, граничні, суцільних конструктивів.

3) Векторні.

4) Крапкові.

1.3 Аналіз основних алгоритмів синтезу зображення

Аналіз основних підходів до побудови синтезуючих систем, яки створені з використанням класичного алгоритму синтезу, який підтримується апаратними прискорювачами, дозволяє виділити ряд типових обчислювальних етапів процесу синтезу зображення сцени.

Етап 1. Створення локальної бази даних кадру. На основі введених координат точки спостерігання та центрів об'єктів сцени виконується пошук і витяг складових компонентів сцени, яки потенційно попадають в зону зору спостерігача. Отримана частина даних список передається на обробку. У залежності від обраного способу обробки до складових локальної бази

відносять: сукупність об'єктів, об'єкт, грань (елемент поверхні), поверхні і/чи їхні елементи, вектор (ребро), крапка, спеціальні елементи.

Етап 2. Геометричні перетворення компонентів сцени. На цьому етапі компоненти піддаються необхідним геометричним перетворенням: обчислення нормалей, нормалізація нормалей, поворот/зсув векторів, зміна мірила, відсікання, проектування, відсікання по площині екрана, видові перетворення.

Етап 3. Розрахунок освітленості. Виконується розрахунок енергетичних параметрів освітленість/затененность у залежності від типу і кількості джерел і прийнятої моделі освітленості.

Етап А4. Перетворення об'єктів сцени в растровий вид. Етап може включати як ряд проміжних перетворень: розшарування примітивів одного рівня на примітиви більш низького рівня .При кожній такій розбивці розраховуються геометричні і колірні параметри примітивів -координати, нахили, збільшення і т.п., так і власне растрирование.

Етап 5. Видалення невидимих елементів. На даному етапі виконується визначення видимих елементів сцени і видалення з подальшої обробки невидимих елементів.

Етап 6. Зафарбування елементів. У залежності від прийнятої моделі освітленості виробляється розрахунок колірних параметрів кожного елемента зображення.

Етап 7. Формування і запис кодів квітів елемента зображення в деякий буфер або безпосередньо в відеопам’ять для виводу на дисплей. Отримані колірні параметри на цьому етапі можуть піддаватися додатковій обробці: згладжуванню для усунення ефекту ступінчастості, гамма-корекцію і зміні мірила для узгодження з характеристиками використовуваного пристрою відображення.

Визначальним, з погляду потенційно досяжних характеристик ГСРЧ, є порядок виконання й обчислювальна складність кожного етапу.

Існує ряд методів комп'ютерного синтезу зображень, на базі яких можлива побудова синтезуючих систем. Основна відмінність цих методів полягає в підході до виконання етапів розшарування/растрирования і видалення невидимих елементів. Причому, видалення невидимих елементів може виконуватися або в об'єктному просторі (до проектування на картинну площину) або в екранному просторі (після проектування елементів на картинну площину).

Метод сканування

У цьому методі видалення невидимих елементів (звичайно граней) сполучено з розшаруванням граней в екранному просторі по рядках екрана. Відзначається, що обчислювальна складність алгоритмів такого роду лінійно залежить від кількості оброблюваних граней і для сцен великої і сверх великої складності час виконання ставати неприйнятним.

Пріоритетні методи

Пріоритетні методи припускають попередній аналіз розташування об'єктів у сцені і розбивку простору сцени на підпростори, у яких елементи сцени однозначно сортуються по глибині стосовно будь-якої крапки підпростору.

Пріоритетні алгоритми дозволяють винести етап видалення невидимих елементів на попередню підготовчу стадію для сцен, у яких взаємне розташування об'єктів або зовсім не змінюється в процесі відпрацьовування польотного завдання, або змінюється, у змісті не порушення пріоритетних списків, незначно.

Оскільки реальні об'єкти, у загальному випадку, обмежені досить складними геометричними поверхнями, як перший етап опису примітивного об'єкта виконується апроксимація його плоскими гранями –геометрія сцени. Для завдання топології сцени в пріоритетних методах виконується побудова дерева топології об'єкта і формування пріоритетних списків. Ідея попередньої розбивки простору і топологічного сортування примітивів у середині підпростору висловлена Шумейкером, що запропонував заздалегідь

формувати пріоритетні списки нерухомих опуклих об'єктів сцени, розбиваючи простір сцени на субпростори -кластери, що лінійно незалежні. Ідея тут полягає в тім, що для кожного такого субпростору заздалегідь вірно визначні пріоритети об'єктів, що визначають порядок їхнього виводу на екран дисплея для правильного видалення невидимих з погляду спостерігача поверхонь у процесі синтезу зображення. Алгоритми статичного просторового сортування дозволяють за рахунок однобічної видимості граней статично переставити їх так, щоб при будь-якому положенні спостерігача порядок обробки граней відповідав порядку їхнього потенційного закривання один одним. Порядок проходження від ближніх до далеких щодо положення спостерігача називають прямим пріоритетним порядком. У цьому випадку грані, що потрапили на обробку першими, не можуть перекритися наступними гранями. При зворотному пріоритетному порядку (від далеких до ближнього) кожна наступна грань потенційно закриває попередню. Пріоритетні системи відображення видаляють невидимі поверхні або шляхом послідовного накладення (при зворотному пріоритетному порядку), або за рахунок позначки вже зайнятих пикселов (при прямому пріоритетному порядку). Грані, що утворять опуклий об'єкт, одноприоритетні. Наявність такої інформації значно зменшує обсяг обчислень на етапі синтезу зображення.

Маючи «стандартний апарат синтезу багатокутника», для використання метода пріоритетів необхідно додати зовнішній цикл по багатокутникам, а, отже, виконати вибірку пріоритетного списку багатокутників для кожної точки спостереження. Аналіз кластера, у якому знаходиться спостерігач, виконується шляхом перевірки знаку скалярного добутку вектору спостереження на нормаль, що поділяє грані. Аналіз виконується у системі координат об'єкту, тому точка спостереження, задана у мировій системі координат, перетворюється у систему координат об'єкту.

Метод Z-буфера

Група алгоритмів Z-буфера припускає апаратну реалізацію порівняння відстані від спостерігача до потенційно відображуваного в даному елементі екрана. Часові характеристики методу не залежать від складності генеруємої сцени. Недоліком методу є необхідність виконання на робочій стадії етапу розшарування/растрирования, що для сцен великої складності приводить до порушення вимог за часом генерації кадру.

Метод трасування променів

Вище описані методи не дозволяють генерувати високоякісні зображення, що наближаються до фотографічного. Метод трасування променів забезпечує генерацію зображень фотографічної якості. Вважається, що метод трасування променів дає найбільший можливий ступінь реалізму. При побудові зображення промінь посилається в заданому напрямку для оцінки прихожої відтіля світлової енергії. Ця енергія визначається освітленістю першої поверхні, що зустрілася на шляху. Механізм виникнення освітленості наступний.

Кожне джерело світла випускає промені у всіх напрямках. Потрапляючи на поверхню, промінь частково переломлюється, частково відбивається і частково розсіюється. Проходячи через прозорий матеріал, промінь перетерплює природне ослаблення.

Розрізняють пряме і зворотне трасування променів.

Пряме трасування променів. З кожного джерела випускаються промені в усі сторони. При перетинанні променя з поверхнею виходять відбиті, переломлені пучки променів, що рівномірно поширюютьсяв усі сторони. Просліджуючи всі промені, ми знайдемо, що частина з них, перетинаючи екранну площину, попадає в око спостерігача. Такий повний прорахунок променів через величезну кількість обчислень, які необхідно зробити, як правило, не застосовується. Крім того, велика частина роботи виявиться проведеної задарма, тому що значна кількість зусиль піде на визначення освітленості поверхонь, не видимих спостерігачу.

Зворотне трасування променів. При зворотному трасуванні променів відслідковуються тільки промені, що попадають в око спостерігача. Це виконується в такий спосіб. З ока спостерігача через кожен пиксел екранної площини в синтезовану сцену пропускається промінь, і потім він відслідковується в зворотному напрямку. Коли промінь наштовхується на поверхню, інтенсивність відповідного пик села визначається освітленістю найближчої крапки перетинання променя з поверхнею. Як що на шляху променя не виникає ніякого об'єкта, варто брати освітленість навколишнього простору (неба чи землі спеціально спроектованої однорідної моделі поверхні).

В алгоритмі використовуються нескладні математичні співвідношення. При цьому обчислення на етапах геометричних перетворень, розрахунку параметрів освітленості і растрірования виконуються одночасно, однак, це вимагає значних витрат часу, що до недавнього часу не дозволяло використовувати його як метод синтезуючих ГСРЧ.

Лекція А2

АНАЛІЗ ОСНОВНИХ АРХІТЕКТУРНИХ РІШЕНЬ ПОБУДОВИ СИСТЕМ СИНТЕЗУ ЗОБРАЖЕННЯ РЕАЛЬНОГО ЧАСУ

Практично всі сучасні алгоритмічні й архітектурні рішення, зв'язані з проектуванням систем синтезу зображень реального часу, спрямовані на подальше підвищення ступеня реалістичності зображень. Реалістичність зображення зв'язана, насамперед, з підвищенням точності опису як геометричних, так і енергетичних параметрів реальної сцени, що, природно, приводить до значного збільшення обсягу бази даних опису сцени і, отже, до збільшення кількості переданої й оброблюваної інформації в кожнім кадрі синтезованого зображення.

Значні зусилля розроблювачів графічних систем привели до появи нових (і модифікації раніше розроблених) алгоритмів, що істотно прискорили складні й віднімаючи багато часу задачі візуалізації. З окрема нові алгоритми для обчислення віпромінності [2] і відображення тривимірного обсягу продемонстрували продуктивність, істотно більш високу, чим при застосуванні ранніх методів. Але, незважаючи на ці досягнення, візуалізація складних сцен чи наборів даних залишається в обчислювальному відношенні дорогою. Так, наприклад, обробка 256x256x256-вокселного набору даних вимагає приблизно 5 секунд на кадр на 100-Мгц робочої станції SiliconGraphicsIndigo при використанні алгоритму відкидання променів [3], що значне перевищує вимоги реального часу.

Просте нарощування ступіней горизонтального конвеєра (рис.А2.1) при великій кількості даних, переданих з однієї ступіні конвеєра в іншу, призводить до збільшення процентного співвідношення операцій передачі даних у порівнянні з більш швидкими операціями їхньої обробки. І, починаючи з деякого обсягу переданої в кадрі інформації, така архітектурна організація стає неприйнятною.

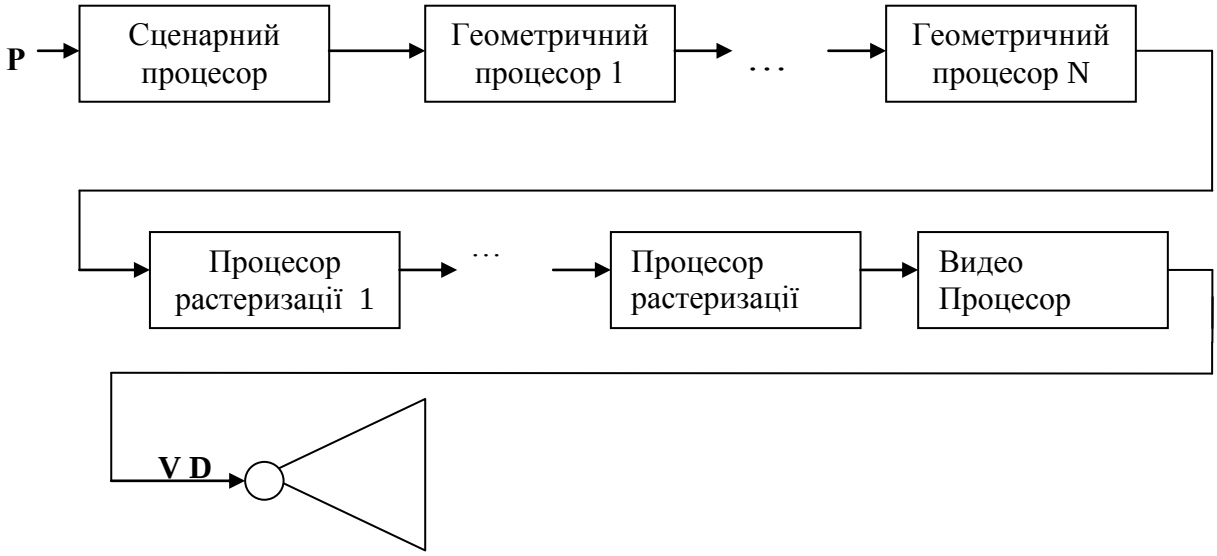


Рис. А2.1. Горизонтально-конвеєрна архітектура

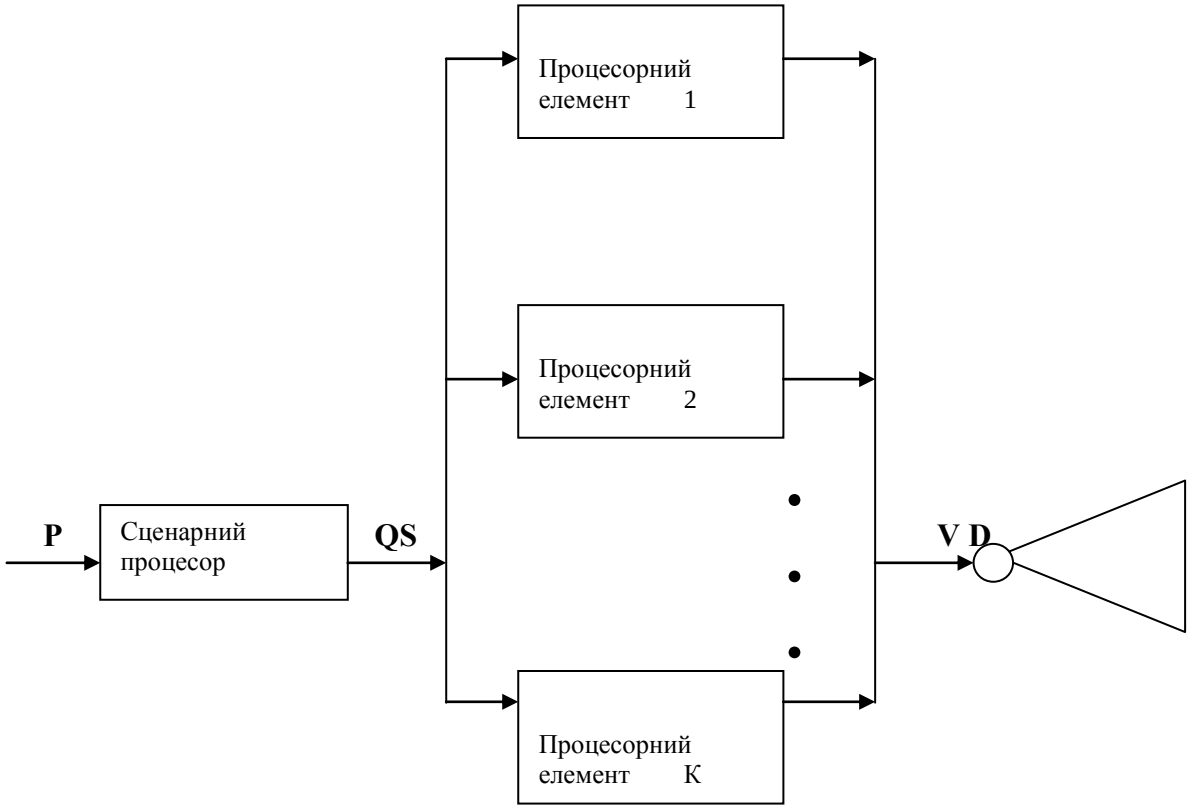


Рис. А2.2. Вертикально - конвеєрна архітектура

При використанні вертикально-конвеєрної архітектури (рис.А2.2) необхідно виконувати розподілення елементів сцени на вході та їхню композицію на виході, що є достатньо складним питанням.

Мультіобробка забезпечує привабливе рішення цього обчислювального вузького місця. Відомо, що алгоритми трасування променів дозволяють використовувати істотний паралелізм, те ж саме справедливо для методів відпромінності.

Відомо, що метод трасування променів припускає виконання тих самих дій при трасуванні промінів через кожен пиксел екрана. При цьому трасування через один пиксел не вимагає використання інформації, отриманої при трасуванні через інший пиксел екрана. Ця незалежність призводить до появи найпростішої, з алгоритмічної точки зору, і потребуючої великих апаратурних витрат архітектурної організації складних обчислювальних систем, що полягає у використанні окремого процесора для трасування промінів через кожен пиксел - "процесор на пиксел". При цьому для кожного променя, у загальному випадку, необхідно виконати перевірку на перетинання/не перетинання з всіма об'єктами сцени. Для рішення цієї задачі необхідна наявність "копії" опису сцени в кожнім пикселном процесорі, що при істотному розмірі синтезованих сцен вимагає використання пристроїв пам’яті великої ємності в кожнім процесорі, чи використання однієї пам'яті для всіх процесорів. Другий підхід дозволяє використовувати єдину пам'ять, але призводить до нерегулярного і непередбаченого доступу до тих самих даних з боку різних процесорів. Це вимагає розподілу даних і керування зв'язком між даними, дуже важке в парадигмі програмування з явною посилкою повідомлень, підтриманої більшістю графічних мультипроцесорів, тому що ці задачі повинні бути вирішені безпосередньо програмістом. Потреба в явному керуванні зв'язком приводить, по-перше, до ускладнення рівнобіжних алгоритмів, що виглядають трохи подібними їхнім послідовним аналогам і, по-друге, до істотної неефективності їхнього виконання.

З погляду архітектурної організації систем синтезу високо реалістичних зображень реального часу є побудова багатоканальної системи, у якій кожний з паралельно працюючих каналів синтезує свій власний об'єкт чи групу об'єктів.

Наприклад, один канал зайнятий синтезом небозводу і розрахунком параметрів повітряного середовища, другий-поверхні Землі, що встеляє, третій – наземних об'єктів, четвертий -об'єктів, що знаходяться в повітрі і т.д., при цьому для формування результуючого зображення використовується композиція канальних зрізів з обліком їхнього взаємного розташування і z-координати об'єктів чи по якому-небудь іншому принципу. Аналіз основних підходів до побудови синтезуючих МГС виконується з погляду оцінки системи по ступені реалізму - представлення сцени в базі даних - і режиму реального часу - процес синтезу кадру зображення сцени**.**

Система працює в режимі покадрової синхронізації, тобто в періоди, кратні такту, відбувається примусовий запуск кожної ступені конвеєра на обробку чергового кадру. "Вузьким" місцем багатоканальних систем є розподіл даних між вертикальними ступенями конвеєра.

Лекція А3
РОЗРОБКА АЛГОРИТМУ РОЗПОДІЛУ ДАНИХ У ГРАФІЧНІЙ СИСТЕМІ РЕАЛЬНОГО ЧАСУ

Практично всі сучасні алгоритмічні й архітектурні рішення, зв'язані з проектуванням систем синтезу зображень реального часу, спрямовані на подальше підвищення ступеня реалістичності зображень. Реалістичність зображення пов'язана, насамперед, з підвищенням точності опису як геометричних, так і енергетичних параметрів реальної сцени, що, природно, приводить до значного збільшення обсягу бази даних опису сцени і, отже, до збільшення кількості переданої й оброблюваної інформації в кожнім кадрі синтезу зображення.

А3.1 Алгоритм розподілу потоків даних у системі

Представлення й опис даних про синтезовану сцену в першу чергу залежить від особливостей розв'язуваної задачі, включаючи склад компонентів сцени і насиченість їх елементами різної важливості (у змісті вироблення рішень), а також особливості сприйняття спостерігачем (чи оператором системи) тих чи інших компонентів сцени. Ці особливості визначають необхідний і достатній ступінь реалістичності.

Площина вікна спостереження і відповідна їй площина екрана розбивається на деяку кількість фрагментів, кількість яких залежить від бажаного ступеня реалістичності зображення, а також, бажано, повинне відповідати кількості паралельно працюючих процесорних елементів системи, що виконують безпосередньо синтез своєї частини зображення.

У свою чергу, для ефективного використання розпаралелювання даних вихідна сцена повинна бути і ієрархічно розбита на квадрати (чи велику кількість частин), пропорційних розбивці площини екрана. З огляду на те, що в залежності від дальності розташування спостерігача від сцени діапазон

охоплення простору сцени змінюється, і, природно, міняється можливість розрізняти ті чи інші деталі сцени, розбивка на фрагменти варто виконати для кожного рівня деталізації сцени.

З метою з'ясування необхідного числа рівнів деталізації опису сцени, насамперед, необхідно погодити з користувачем системи діапазони просторових взаємних переміщень спостерігача і відображуваної сцени, швидкість взаємних переміщень і інші специфічні зведення з метою досягнення зразкового однакового обсягу інформації для обробки на кожнім рівні деталізації.

А3.2 Організація процесу синтезу

Перед початком циклу формування кадрів зображення необхідно виконати первісне завантаження локальної пам'яті кожного процесорного елемента частиною бази даних (опису сцени). Таке завантаження визначається умовами застосування системи синтезу як частини моделюючих комплексів, тобто для деяких з них початкове положення спостерігача і сцени заздалегідь відомо, отже, початкові фрагменти бази даних теж визначаються однозначно.

Якщо така інформація відсутня чи не є постійної, то процес синтезу починається з "грубого" трасування променів через центр кожної ділянки екрана з метою визначення початкового рівня деталізації відображення сцени для вибірки відповідного фрагмента опису сцени. При спробному "грубому" трасуванні променів виконується аналіз перетинання тієї чи іншої частини сцени тільки по покажчиках. При досягненні ступеня деталізації, що відповідає дальності від крапки спостереження до ділянок сцени, дані про опис обраної частини сцени передаються в локальний запам'ятовуючий пристрій процесорного елемента, що відповідає фрагменту екрана (рис.А3.1).

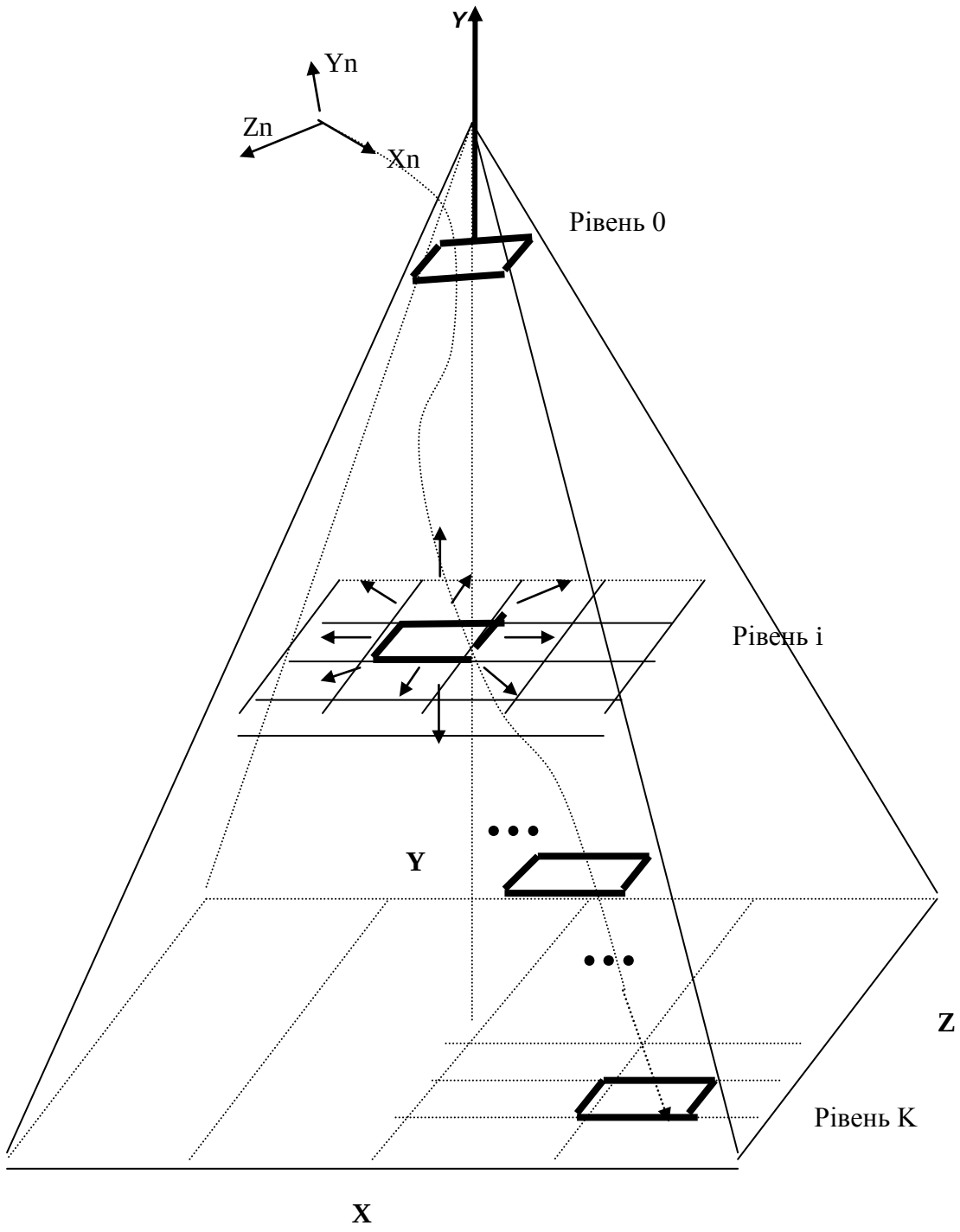


Рис. А3.1. Ієрархічне розділення сцени

Далі для наступних кадрів зображення дані вибираються з локальної пам'яті процесорного елемента і, у разі потреби, виконується до загрузка наступних сторінок локальної пам'яті. Для пояснення логіки до загрузки і

перезавантаження локальної пам'яті процесорного елемента розглянемо більш детально рис.А3.1.

Нехай вікно спостереження в даний момент часу знаходиться на і-тім рівні деталізації сцени. При цьому можливі 10 варіантів подальшого пересування вікна:

1-зміна<-x,-z>;

2-зміна<-x>;

3-зміна<-x,+z>;

4-зміна<+z>;

5-зміна<+x,+z>;

6-зміна<+x>;

7-зміна<+x,-z>;

8-зміна<-z>;

9-зміна<+y>;

10-зміна<-y>.

Варіанти 1-8 передбачають до загрузку сторінок локальної пам'яті, що відповідають зміні однієї чи двох координат.

Варіанти 9 і 10 зв'язані зі зміною ступеня деталізації і вимагають повного перезавантаження в місту локальної пам'яті процесорного елемента. Процесом до загрузки і перезавантаження локальної пам'яті повинен управляти спеціальний процесор – сценарний процесор. Дозагрузка наступних сторінок локальної пам'яті і процес синтезу в кожнім процесорному елементі виконуються паралельно за рахунок використання в процесорних елементах двох буферних запам'ятовуючих пристроїв. Процес повного перезавантаження локальної пам'яті може зажадати більше, ніж час одного кадру (3-4 кадрів), тому, щоб не порушити плавність зміни зображення, сценарний процесор дає вказівку композитному пристрою на виконання екстраполяції зображення на необхідний період.

В алгоритмі використовуються наступні математичні співвідношення. Промінь з початком в точці 0, яка визначається початковим вектором

$R_0=(x_0,y_0,z_0)$ і вектором напрямку $L=(l,m,n)\neq 0$, задається за допомогою параметричного рівняння у векторній формі $R(t)=R_0+Lt$, $t>0$, або координатними параметричними рівняннями:

$$\begin{cases} x = x_0 + lt; \\ y = y_0 + mt; \\ z = z_0 + nt. \end{cases} \qquad (t>0) \qquad (A3.1)$$

де l, m і n знаходяться з наступних формул

$$\begin{cases} l = R, \\ m = A \bullet x / X_{max} - A / 2, \\ n = B / 2 - B \bullet y / Y_{max} \end{cases} \qquad (A3.2)$$

Плоскість задана загальним рівнянням

$$ax+by+cz+d=0, \qquad (A3.3)$$

де N=(a, b, c) - нормальний вектор плоскості.

Зміною у рівнянні плоскості величин *x, y, z* їхніми виразами з (3.1), отримаємо лінійне рівняння відносно параметру *t*, вирішуя котре знаходимо, що він дорівнює:

$$t = \frac{-d}{al + bm + cn}. \qquad (3.4)$$

Як що знаменник дорівнює 0, то луч проходить паралельно плоскості й не пересякає її. Як що знаменник не дорівнює 0, то виконуємо розрахунок значення t. Якщо $t<0$, луч не перетинає плоскість. Якщо $t>0$, то координати точки перетину розраховуються згідно виразу:

$$\begin{cases} x = lt, \\ y = mt, \\ z = nt. \end{cases} \quad \text{(A3.5)}$$

Потім виконуємо аналіз: чи належить точка, що знайдена, багатокутнику. З урахуванням того, що нормальний вектор N=(a,b,c) плоскості, в якої знаходиться багатокутник, не дорівнює нулю, цей n-кутник можливо однозначно спроектувати на n-кутник, який лежіть в однієї з координатних плоскостей. Таким чином отримаємо 2D координати багатокутника й точки, а потім виритим: знаходиться дана точка всередині багатокутника, чи ні.

А3.3 Розбивка синтезованого простору і підготовка структури даних для підтримки алгоритму

Як було зазначено вище, вихідна сцена повинна бути ієрархічно розбита на квадрати пропорційно розбивці площини екрана для кожного рівня деталізації сцени.

Розбивка простору може бути виконана з використанням методу ієрархічних іррегулярних тріангуляційних мереж (HTiNs), що ще у 1990 році запропонував Scarlatosi Pavlidis [1]. Метод має високу точність і використовує стиск даних. При цьому можлива генерація різних рівнів деталізації зображень. HTiNs складаються з регулярних трикутників і , тому, можуть апроксимувати будь-які поверхні на будь-яких рівнях деталізації, зв'язаних між собою ієрархічно. Метод використовує матрицю реальних цифрових висот точок місцевості; кожен рівень ієрархії відповідає різним рівням деталізації, що апроксимує поверхню з заданою чи імовірністю заданою максимальною помилкою. Кожному трикутнику і-того рівня відповідає набір трикутників в і+1 рівні і результуюче дерево. Подібна структура забезпечує ліквідацію необхідності проміжних тріангуляцій

підлеглих рівнів. Тріангуляція виконується для 3D поверхонь з урахуванням характерних точок топології реальної місцевості- вершин, ям, сідел, гребенів, зламів і каналів. Автори вказують, що тріангуляція виконується для сцени один раз, тому виконується довго, забезпечуючи найкращу апроксимацію і мінімізацію трикутників з гострими кутами, що приводять до перекручувань.

За допомогою цього методу отримані реалістичні зображення складних ландшафтів і рельєфів.

А3.4 Особливість функціональної організації сценарного процесора для підтримки алгоритму

Сучасні високо продуктивні системи генерації зображень будуються як багатопроцесорні матричні системи. Для управління розподілом даних по процесорних елементах у таких системах використовується сценарний процесор, що формує локальну базу даних для синтезованого кадру по отриманим від зовнішньої системи моделювання динаміки транспортних засобів векторам положення спостерігача й об'єктів сцени і даним вихідної бази даних опису сцени.

У реальному часі при виконанні алгоритму синтезу сцени сценарний процесор вирішує наступні задачі:

1) Аналіз видимості сцени в даному кадрі.

2) Визначення рівня деталізації для поточного кадру (у тому числі, можливі рівні деталізації для різних фрагментів екрана, якщо вони різні).

3) Виконання прогнозування ймовірного переміщення спостерігача що до сцени на найближчі 3-4 кадри.

4) Виробку сигналів управління другими пристроями.

А3.5 Використання рівнів деталізації

Для визначення рівня деталізації об'єктів, що потрапили в піраміду видимості, досить виконати спрощену перевірку: порівняти значення координати X центра об'єкта в системі координат спостерігача зі значенням Dmin від повідного рівня: X<Dmin.

Для впровадження рівнів деталізації з метою трасування променів у реальному часі, а також для опису 3D-геометрії і текстур доцільно виконувати опис сцени у виді розрідженого октарного дерева. Це є один з ключових методів акселерації трасування променів, у рамках якого вони рекурсивно розбиваються січними площинами на сегменти, що дозволяє істотно прискорити обчислення точок їхнього перетинання з поверхнями, які будуть відображені.

Така форма організації даних легко дозволяє виконувати розподіл даних між паралельно працюючими процесорами.

А3.6 Висновки

Запропонований алгоритм розподілу даних дозволяє використовувати різні рівні деталізації та змінювати кількість паралельно працюючих процесорів.

Найбільш складним питання остається прогнозування подальшого переміщення вікна спостереження – найбільш "інтелектуальною функцією" сценарного процесора. Для прогнозування подальшого пересування сценарний процесор повинен виконувати і аналіз попередніх положень центра системи координат спостерігача, збереження яких необхідно передбачити. Ступінь вірогідності прогнозування визначається, з одного боку, кількістю аналізованих положень і, з іншого боку, обраним алгоритмом прогнозування.

Лекція А4

АПАРАТНА ПІДТРИМКА АЛГОРИТМУ РОЗПОДІЛУ ДАНИХ

А4.1 Архітектура системи

Для реалізації алгоритму розподілу даних системи повинна бути організована наступним чином (рис.А4.1).

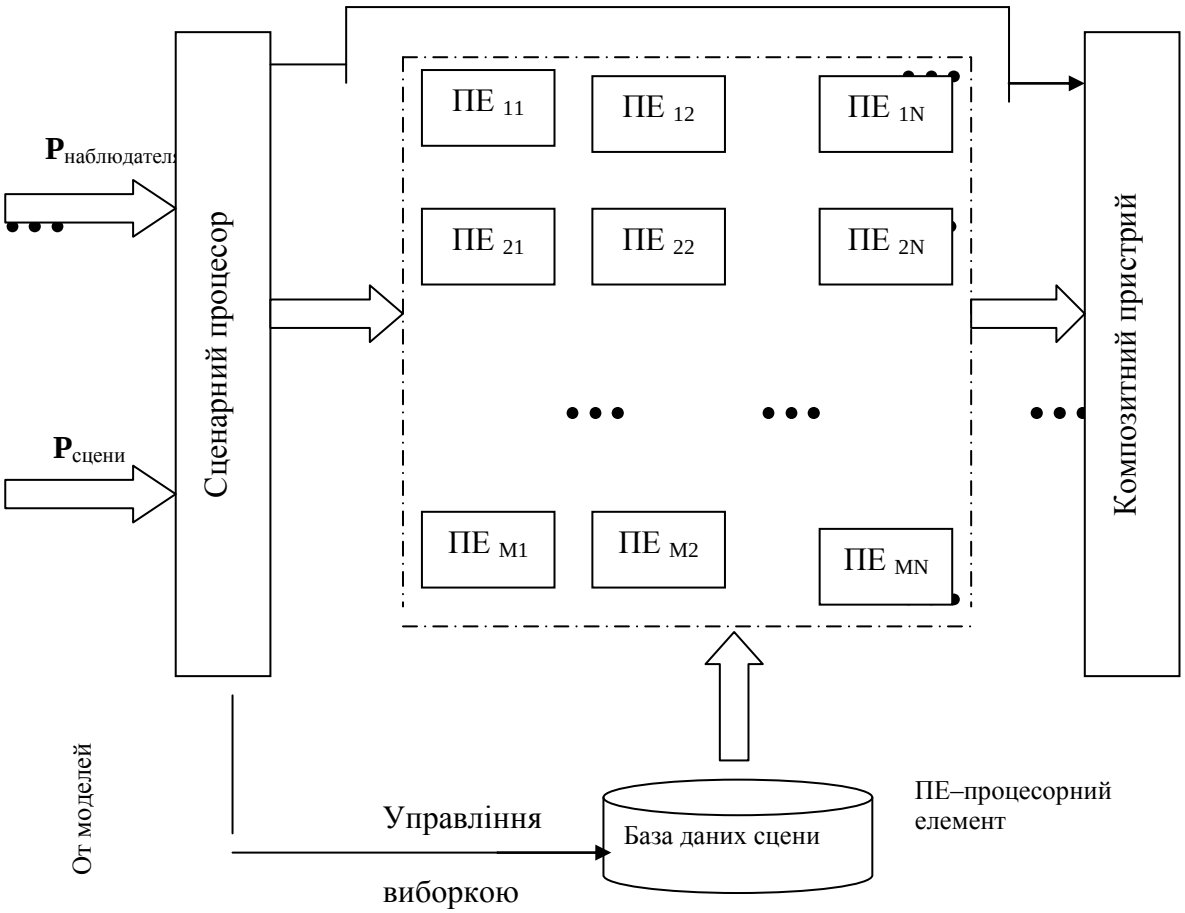


Рис.А4.1. Архітектура системи

Розглянемо варіанти реалізації процесорних елементів.

А4.2 Використання спеціалізованих процесорів

Одним з недоліків традиційного підходу при формуванні тривимірних зображень у сучасних комп'ютерних іграх і навіть у ряді професійних програм тривимірного моделювання і візуалізації є дуже наближена модель висвітлення.

Сучасні візуальні процесори (VPU) при відображенні кожного пікселя вміють також виконувати невеликі програмки, що оперують інформацією, специфічної для даної точки растеризуемого трикутника. Ці програмки призначені, зокрема, для кращого відображення матеріалу поверхні шляхом комбінування текстур кольору і рельєфу. А при розрахунку висвітлення заданої точки трикутника, що малюється, вони просто виходять з відстані до джерела світла і кута падіння світлових променів. І те лише, як що динамічне висвітлення в загалі передбачене.

Сучасна комп'ютерна графіка вже немислима без спеціалізованих прискорювачів, що в очах користувачив навіть важливіше процесорів.

Компанія Advanced Rendering Technology розробила "прискорювач трасування променів" для професійного застосування, а саме, для прискорення рендеринга в пакетах тривимірного моделювання. Називається він Pure і являє собою плату для шини PCI, на якій установлені сім спеціалізованих процесорів для рейтрейсинга - AR350. Процесор, виготовлений по 0,13-мікронному технологічному процесу, уміє за один такт визначати перетинання проміня і трикутника і працює на частоті в 350 Мгц.

Як вже вказувалося раніше, що метод як прямої, так і зворотного трасування променів прекрасно розширюється по кількості процесорів, оскільки промені і фотони можуть трасуваться практично незалежно друг від друга, і кожному процесору можна виділити свою частину променів для обробки.

Тобто на великій кількості процесорів можна одержати обчислювальну систему, цілком еквівалентну, з погляду алгоритму, одному високо

частотному процесору (а може і більш ефективну – за рахунок паралельності обчислень).

А4.3 Використання програмувальної логіки

Прототип апаратного трасувальника променів SaarCOR - і плати FPGA-співпроцесора, розрахованого на використання двох FPGA і цифро-аналогового перетворювача, що формує VGA-сигнал

Але насправді принципова реалізація у реальному часі добре відомого способу візуалізації сцен не є головним достоїнством трасування променів. Від більш традиційної растеризації -і процедури, підтримуваної більшістю графічних акселераторів сучасних відеокарт, трасування променів відрізняється фундаментально. Якщо растеризація ґрунтується на ітеративної алгоритмике, у якій в один момент часу обробляється один трикутник сцени, то трасування променів -і на доступі до всієї сцени відразу. На практиці це означає, що для програміста, що створює ПЗ візуалізації в системі, що реалізує трасування променів, такі традиційно вважаються дуже складними задачі, як, наприклад, відображення тіней і відображень від множинних джерел світла, стають тривіальними. Творці тривимірних сцен реального часу одержують також можливість уникнути "трюкових" прийомів для "апроксимації" ідеального трасувальника променів засобами растеризатора.

Але перш ніж сказати кілька слів про сучасну реалізацію апаратного трасувальника променів, давайте вислухаємо думку тих, хто поки зацікавлений в класичних растеризационных графічних підсистемах. Курт Экели з NVidia на питання " чи приведе спрощення і підвищення швидкодії апаратних засобів трасування променів до витиснення ними традиційних растеризаторів?" відповідає коротко й однозначно: "Трасування променів стане значимою технологією і вийде за межі ринку тільки тоді, коли в неї з'явиться могутня підтримуюча апаратна інфраструктура". Ларри Сейлер з ATI вважає, що "трасування променів ніколи не замінить растеризації. Обидві ці технології поступляться місцем чомусь новому по трьох причинах.

По-перше, два алгоритми (маються на увазі растеризація і трасування променів) не вирішують ту саму проблему: якщо трасування променів чудово підходить для моделювання реального світу, те растеризація як сукупність різних алгоритмів дає кращі результати при моделюванні об'єктів, що не мають аналогів у фізичній реальності. По-друге, смуга пропущення пам'яті росте набагато повільніше, ніж продуктивність обчислювачів. По-третє, алгоритмика растеризаторів менш чуттєва до затримок, що неминуче виникають у процесі обробки великих сцен.

Отже, після протверезного холодного скепсису представників двох гігантів сучасної індустрії графічних акселераторів повідомлення про те, що на виставці CeBi 2005 командою розроблювачів з університету Саарленд продемонстрований робочий прототип апаратного відеоконтролера, що реалізує трасування променів у реальному часі. Утім, прототип, названий SaarCOR, на відміну від навічно застиглих у стадії опису достоїнств і так і не упредметнилися відеокарт, дійсно працює, причому показує дуже значні для своїх скромних параметрів значення продуктивності. Судите самі: реалізований SaarCOR на мікросхемі програмувальної логіки FPGA (короткому огляду програмувальної логіки присвячена друга частина статті) сімейства Xilinx Virtex-ii з деяким більш ніж 76 тис. логічних осередків, неповними півтори сотнями апаратних 18 бітних добутковичів і двома з половиною мегабітами пам'яті. Ці можливості FPGA (далеко не найшвидшої і ємний на сьогоднішній день) дозволили реалізувати в одному кристалі рівнобіжну 64 потокову підсистему трасування променів, що підтримує до 256 джерел світла, а також інтерфейс шини PCI, VGA-інтерфейс і апаратні засоби оцінки продуктивності в реальному часі. При цьому витрати ресурсів FPGA виявилися дуже скромними -і 56% логічних осередків і 78% пам'яті. Тактова частота машини трасування, що вийшло, променів склала 90 MHz (більш розвиті сучасні FPGA дозволяють її істотно збільшити). Для порівняння: чипсет NVidia GeForce 5900FX містить 125 млн. транзисторів, а продуктивність його чотирьох сотень обчислювачів із крапкою, що плаває,

приблизно в 50 разів вище, ніж сумарна продуктивність SaarCOR. Ці дані приведені не для протиставлення апаратної підтримки алгоритмів растеризації і трасування променів GeForce і SaarCOR відповідно, а для демонстрації можливостей росту технології -і все-таки університетська команда дослідників не має у своєму розпорядженні можливості R&D-центра промислового гіганта. А от є чи зміст продовжувати розвивати технологію SaarCOR, можна сказати, порівнявши її продуктивність із сугубо програмної SSE-оптимизованої реалізацією трасувальника променів (система OpenRT), що виконується на ПК із процесором Intel Pentium 4 2,66 MHz і обсягом оперативної пам'яті 1 GB. У 30 разів більша, ніж у SaarCOR, тактова частота універсального процесора і SSE-код більш ніж непоганого трасувальника променів ніяких особливих переваг програмної реалізації не дають: навіть сцену, що містить 187 млн трикутників, SaarCOR визуализирует у чотири з зайвим разу швидше -і з частотою 32 кадру в секунду (для особливо педантичних -і при розмірі кадру 512x384 пиксела, з використанням тільки головних променів, з урахуванням затінення з повним текстурированием). Мабуть, найбільш цікавою властивістю SaarCOR стала виявлена його творцями невимогливість апаратної реалізації до ширини смуги пропущення: при цьому показнику, меншій у 4 рази, чим у NVidia GeForce3, SaarCOR досягає такий же, як у GeForce, продуктивності, обмірюваної в частоті кадрів у секунду. Якщо ми згадаємо, що говорив фахівець ATI Ларри Сейлер про вимогливість трасувальників променів до смуги пропущення підсистеми пам'яті, і врахуємо, що ці слова -і не омана, а думка фахівця, SaarCOR можна вважати винятково важливою розробкою. Хоча б у тім змісті, що, руйнуючи стереотипи, вона дає гарну відправну крапку для пошуку того нового, що неминуче прийде на зміну сьогоднішнім графічним підсистемам.

З маси цифрових інтегральних мікросхем, доступних конструкторам, можна виділити кілька великих і найбільш загальних класів. Базовий (самий низкорівневий) клас - це мікросхеми низького ступеня інтеграції, що реалізують порівняно прості логічні функції і характеризуються незначним

числом входів/виходів (що еквівалентно і порівняно невеликому числу "ніг" у корпуса мікросхеми).

Другий рівень - пам'ять. Вона буває декількох типів. Енергонезалежна (цей термін підкреслює незалежність умісту від енергії джерела харчування) по інерції називається постійної, хоча більшість сучасних її варіантів допускають зміна вмісту. Пам'ять, одночасно більш ємна і дешева, але вона характеризується залежністю змісту не тільки від енергії джерела харчування, але і від часу, називається динамічної оперативний (для цієї залежності значимий час виміряється мілісекундами, якщо ж говорити про тисячоріччя, те можна змело затверджувати, що уміст усіх типів пам'яті залежить від часу). Сама скромна по ємності пам'ять, що яскраво виділяється високою швидкодією (і вартістю за одиниця об'єму) -і статична оперативна. Її вміст залежить від енергії джерела харчування, але не залежить від часу.

Третій клас мікросхем - мікропроцесори і мікроконтролери, що є, по суті, "упакованої" на одному кристалі комбінацією мікросхем трьох класів. І нарешті, четвертий клас -і мікросхеми програмувальної логіки. Незважаючи на те що вони з'явилися небагато раніш, ніж перший мікропроцесор, їх можна вважати більш високорівневим "будівельним вузлом" через те, що програмувальна логіка допускає таку ж гнучкість у побудові апаратних засобів, яку мікропроцесори допускають у побудові програмних (з іншого боку, деякі сучасні інтегральні мікроконтролери розглядаються і їх виробниками, і розроблювачами кінцевих виробів, яки їх використовують, як ефективні замінники програмувальної логіки).

Затребуваність програмувальної логіки порозуміється постійним ростом потреби в "нестандартних" чи "не зовсім стандартних" логічних вузлах у виробників кінцевої продукції. Мікросхеми низького рівня як основа для їхнього виробництва непридатна - і занадто дорогими і неефективними виходять величезні друковані плати, напхані низкорівневими мікросхемами. Звичайно, існує так називана "замовлена логіка", але вона донині дуже дорога - загальна вартість циклу розробки і підготовки до серійного випуску

замовленої мікросхеми може вимірятися семизначними сумами. Отже, для того щоб домогтися рентабельності, тираж виробу, що використовує таку замовлену мікросхему, повинний бути просто астрономічним.

Перший етап розвитку програмувальної логіки можна описати наступною ланцюжком абревіатур: PROM, PLA, PAL, GAL (для цих чотирьох типів програмувальних чипів є збірна назва SPLD), CPLD. У неспеціальних виданнях найчастіше зустрічається PROM, що позначає... звичайні мікросхеми енергонезалежної пам'яті. І тим часом, ці мікросхеми дозволяють створювати "програмувальне залізо" -і раз вони здатні ставити дані у відповідність адресі (тобто деякому двоїчному слову), виходить, на їхній основі можна реалізувати різні логічні функції. Спочатку технологічною основою PROM були масиви перепалюва електрично мікроскопічних нихромових перемичок (fuse, що працювали по тім же принципі, що і звичайний запобіжник). Наступне покоління PROM, навпроти, створювалося на основі реверсивного принципу (що підкреслюється навіть назвою технології - antifuse) - в пристроях, що поставляються виробником, перемички були розімкнуті. В розімкнутому стані вони являють собою мікроскопічні стовпчики аморфного кремнію, що утворять "бутерброд" з металевими обкладками. При програмуванні опір такого стовпчика змінюється з гигаом до тисяч чи сотень ом - тобто металеві обкладки "бутерброда" фактично з'єднуються. Пристрої, виконані і по fuse, і по antifuse- технологіям, поєднуються збірним терміном "однократно програмувальні" - OTP (One-Time Programmable).

Друга технологічна фаза в історії - перехід від "простих і грубих" фізичних принципів формування "перемичок" до більш складним, що дозволяє усунути головний недолік OTP - неможливість зміни записаної в мікросхему інформації. EPROM, чи «доступна тільки по записі пам'ять» (Erasable PROM), ґрунтувалася на незначній модифікації інтегрального польового MOS (метал-окисел-напівпровідник) транзистора. Перший пристрій на основі даної технології з'явилося на ринку в 1971 р. (це була

мікросхема пам'яті Intel 1702). EPROM дозволяла стирати вміст мікросхеми, але вимагала для процедури стирання опромінення напівпровідникового кристала ультрафіолетовим випромінюванням. Відповідно, виконані по EPROM-технології чипи мають характерний конструктив корпуса - з віконцем із кварцового скла (варто помітити, що такий корпус більш ніж недешевий). Доля EPROM була визначена саме потребою в ультрафіолетовому опроміненні кристала: при росту ступеня інтеграції для забезпечення "доставки" ультрафіолетовим випромінюванням до кожного елемента мікросхеми необхідної кількості енергії "стирання" умісту приходилося збільшувати і без того чималий час цієї процедури до десятків хвилин.

На зміну однотранзисторному осередку EPROM прийшла двухтранзисторна EEPROM - її варіація, що стирається електричним шляхом. У ній власне транзистор-"перемичка" не перетерпів фундаментальних змін у порівнянні з EPROM, а за рахунок ускладнення тут додана можливість стирати осередки електричним сигналом, необхідну енергію якого легко підвести до кожного осередку. Більш розповсюджена зараз флеш-технологія, по суті, є збірною назвою для подальших (і вже зовсім незначних) модифікацій EEPROM. Принаймні , що запам'ятовує осередок у флеш- пам’яті й програмувальній логіці в більшості випадків залишилася майже незмінної двухтранзисторною комбінацією транзистора, що стирає, і транзистора-"перемички".

Технологічні досягнення PROM дозволили реалізувати перші "логічні напівфабрикати" -і програмувальні логічні масиви, PLA (Programmable Logic Array). У них "перемички" використовувалися для формування з'єднань у двох різних наборах інтегрованих елементів, що реалізують логічні функції "І" і "АБО". Найважливіша відмінність PLA від PROM полягає в тім, що ступінь складності і характер реалізованих PLA логічних функцій не визначається кількістю висновків мікросхеми. Згодом PLA були витиснуті більш швидкодіючими PAL - "програмувальними логічними матрицями", у

яких програмувалися з'єднання тільки в одному масиві елементів ("I"), з'єднання ж у другому раз і назавжди "зашивалися" виробником.

На наступному витку технологічної спіралі численні елементи вищезгаданих простих типів почали з'єднуватися на одному кристалі за допомогою багатобітних шин. Подібні складні схеми, щоб підкреслити їхній композитний характер, називають CPLD - складними програмувальними логічними пристроями.

Паралельно з програмувальною логікою розвивався і напрямок, родоначальником якого стала унікальна у своєму роді "макросхема" Micromatrix, випущена компанією Fairchild Semiconductor у середині 60 х років минулого століття. "Мікроматриця" являла собою "бутерброд" з металізованих пластикових пластин, між якими були укладені порядку сотні звичайних дискретних транзисторів. Конструктор, що використовує Micromatrix, вручну прорисовував доріжки-сполуки-доріжки на металізованих поверхнях цього "бутерброда". У 1967 р. та ж Fairchild випустила Micromosaic - кілька сотень транзисторів в одній зборці. При цьому для розроблювачів цифрових схем на основі Micromosaic поставлялася комп'ютерна програма, що генерує топологію з'єднань на основі опису майбутнього пристрою на рівні булевих функцій. Micromosaic стала дійсною революцією і відкрила нову епоху - вентильних матриць (gate arrays) і проектування електронних схем на їхній базі за допомогою комп'ютера. Згодом вентильні матриці з мікроосередками, не настільки вуж і сильно відрізняються від єдиного транзистора Micromosaic, стали основою замовлених мікросхем, що дозволяють реалізувати дуже складний пристрій високого ступеня інтеграції, але за занадто великі для мало- і середньоесерийного виробника гроші й у занадто тривалі для вимог масового ринку терміни. Між CPLD, що обмежують функціональність програмувальних з них пристроїв, і замовленими чипами на базі вентильних матриць утворилася незаповнена, але надзвичайно приваблива ніша. Її зайняла компанія Xilinx, у 1984 р., що виставила на ринок новий продукт -

програмувальну замовником в умовах експлуатації вентильну матрицю, FPGA (Field Programmed Gate Array, причому Field Programmed тут указує на те, що FPGA може програмуватися "у польоті" - наприклад, безпосередньо в закінченому пристрої). На відміну від усіх попередніх пристроїв примітивний елемент FPGA -і аналог "логічного осередку" – суттєво складніше. Це програмувальний логічний блок, здатний виконувати ряд функцій, що задаються розроблювачем.

А4.4 Висновки

Таким чином, досягнення технології останніх років дозволяє реалізовувати метод трасування промінів на апаратному рівне, що дає можливість впровадити запропонований метод розподілу даних.

Лекція А5

ДОСЛІДЖЕННЯ АЛГОРИТМУ ТРАСУВАННЯ ПРОМІНІВ

А5.1 Дослідження використання різних рівнів деталізації даних

Для дослідження різних рівнів деталізації використані дві сцени, підготовку яких виконано на базі фрагментів рисунків формату моделей Quake2 (md2). Підготовка опису сцен виконувалась з використання окремих об’єктів, отриманих з декілька рисунків (рис.А5.1.) та моделей поверхонь для фона (рис.А5.2).

Рис. А5.1. Використані рисунки для формування сцен

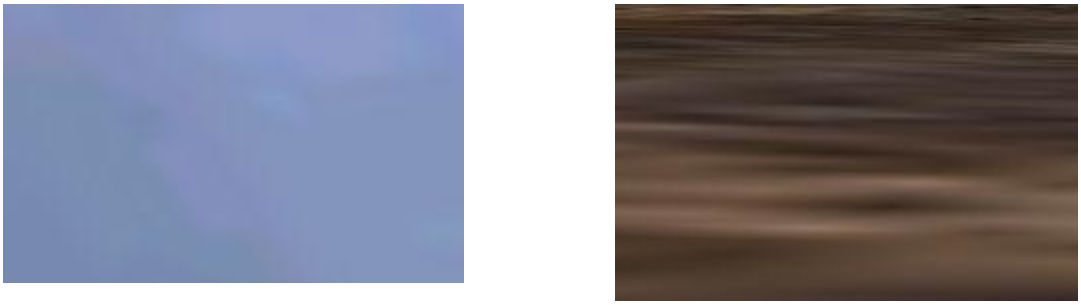


Рис. А5.2. Зразки поверхонь для формування фона

Зміст файлів «exp_scene.dat» (перша сцена) та «tree_scene.dat» (друга сцена) наведений в додатку Б.

На рис.А5.3. показний результат трасування першої сцени з першим рівнем деталізації. На рис.А5.5. показний результат трасування першої сцени з другим рівнем деталізації але без використання поверхні фона.

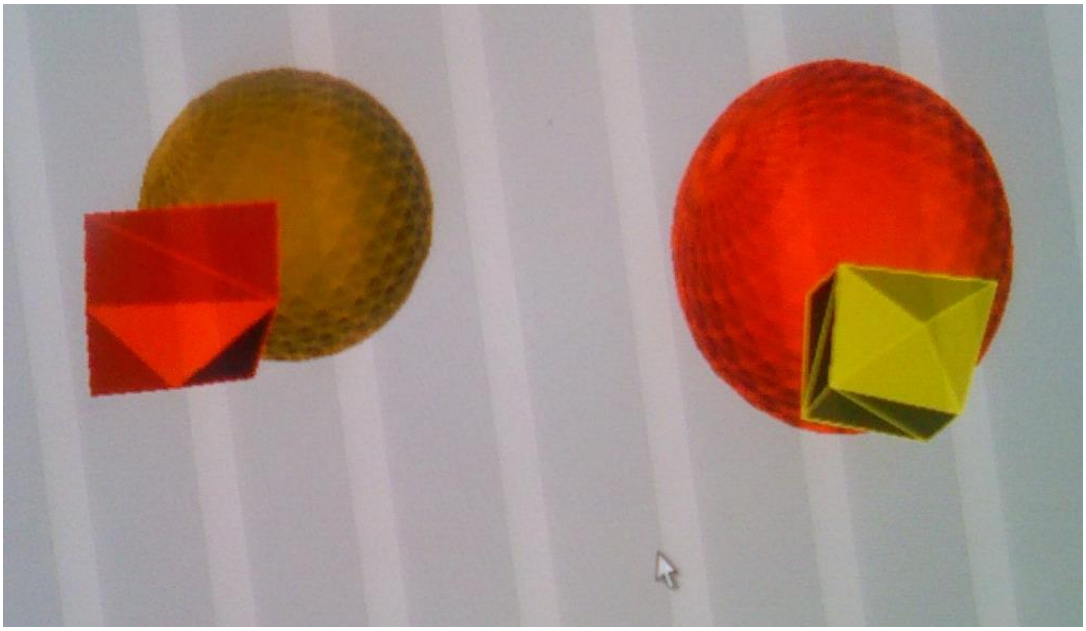


Рис. А5.3.Результат трасування першої сцени з першим рівнем деталізації

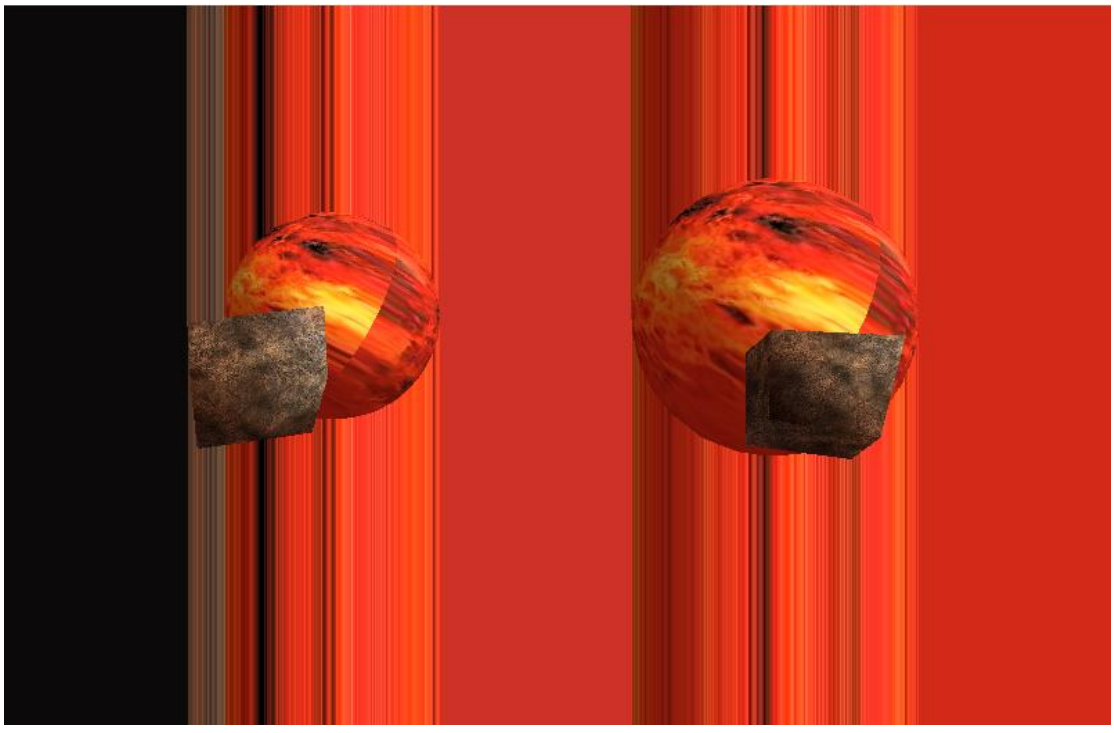


Рис. А5.4. Результат трасування першої сцени з другим рівнем деталізації без використання поверхні фона

На рис.А5.5 показні результати трасування другої сцени з першим рівнем деталізації.

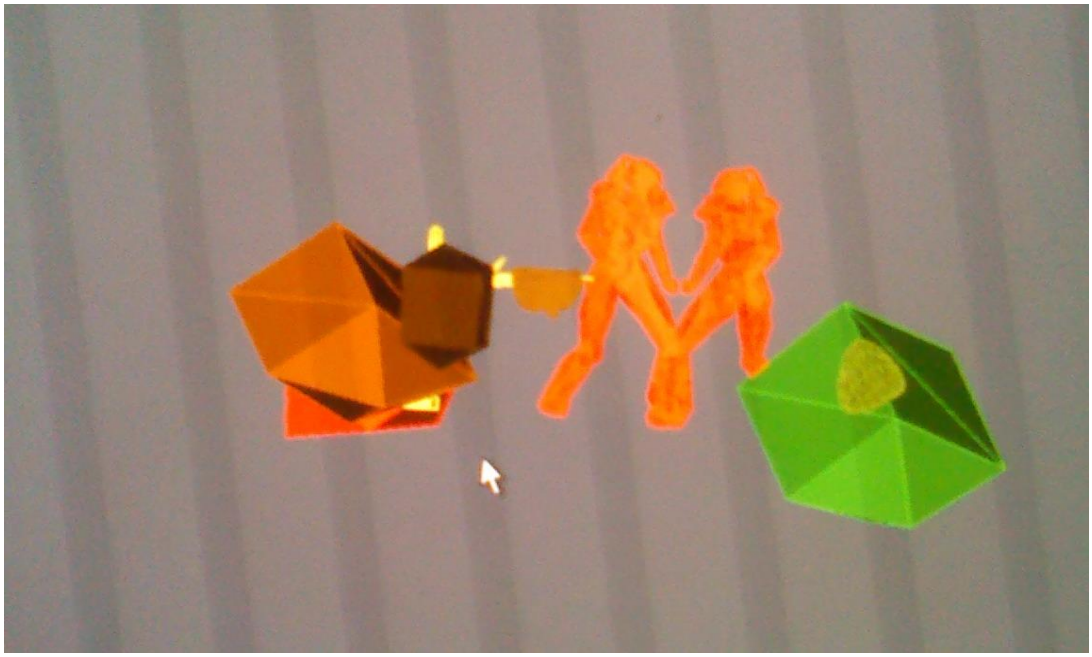


Рис. А5.5. Результат трасування другої сцени з першим рівнем деталізації

Слід відмітити схематичність виводу елементів зображення. Це пояснюється відсутністю маленьких деталей на даному рівні деталізації. Також не відображуються поверхні фона.

На рис.А5.6, А5.7. показні результати трасування другої сцени з найвищим рівнем деталізації, з використання поверхні фона, а також для різних відстань від спостерігача до сцени та різних кутів зору .

Рис. А5.6. Результат трасування другої сцени з другим рівнем деталізації з використанням поверхонь фона

Порівняння процесів створення зображень показує:

1. Використання низького рівня деталізації дозволяє обрати ракурс зображення за короткий час.

2. Другій рівень деталізації дає хорошу якість, але потрібує значно більше часу для трасування. Особливо помітне затримання процесу в центрі картинки, де проміні трасують декілька об’єктів.

3. На тривалість процесу трасування також значний вплив оказує кількість об’єктів сцени, яки передані на обробку після попереднього аналізу з використанням їхніх габаритних сфер.

Рис. А5.7. Результат трасування другої сцени з другим рівнем деталізації з використанням поверхонь фона

А5.2 Дослідження паралельної реалізації

Нажаль, при виконанні роботи не було доступу до багатопроцесорних систем. Тому дослідження паралельної реалізації проведені з використанням одно-, дво-, три- та 4-х ядерних комп'ютерів. Для досліджень обрані бази даних:

Сцена В1: 3 прожектора і 92 примітива (на діаграмах – голубий колір);

Сцена В2: 5 прожекторів і 147 примітивів (жовтий);

Сцена В3: 3 прожектора і 820 (бордо).

Дані дослідження залежності коефіцієнту прискорення від кількості процесорів наведені в табл.А5.1.

Таблиця А5.1 - Дані дослідження залежності коефіцієнту прискорення від кількості процесорів

Кількість процесорів	Сцена В1	Сцена В2	Сцена В3
1	1.00	1.00	1.00
2	1.88	1.93	1.93
3	2.78	2.85	2.88
4	3.57	3.73	3.75

Результати досліджень відображені на рис.А5.8.

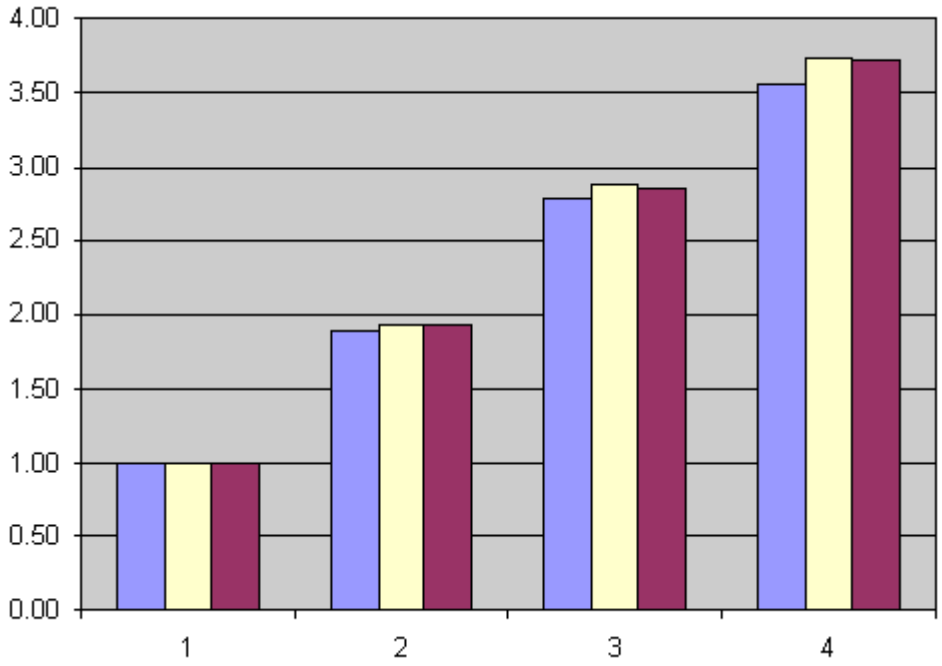


Рис. А5.8. Залежність коефіцієнту прискорення від кількості процесорів

Дані дослідження залежності часу обробки(в %) від кількості процесорів наведені в табл.А5.2.

Таблиця А5.2 - Дані дослідження залежності часу обробки в % від кількості процесорів

Кількість процесорів	Сцена В1	Сцена В2	Сцена В3
1	100.00%	100.00%	100.00%
2	53.24%	51.81%	51.88%
3	35.99%	35.06%	35.78%
4	28.02%	26.84%	26.69%

Результати досліджень відображені на рис.А5.9.

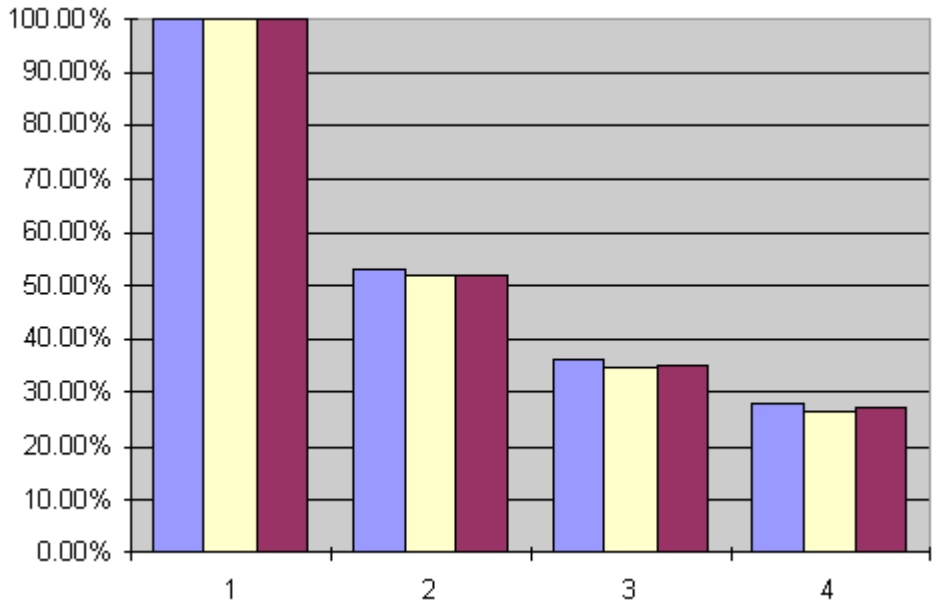


Рис.А5.9. Залежність часу обробки в % від кількості процесорів

Результати дослідження залежності часу обробки (в сек) від кількості процесорів наведені в табл.А5.3.

Таблиця А5.3 - Дані дослідження залежності часу обробки від кількості процесорів

Кількість процесорів	Сцена В1	Сцена В2	Сцена В3
1	278.11	2,785.99	1,295.25
2	148.06	1,443.52	671.96
3	100.08	976.85	450.43
4	77.94	747.64	345.68

Результати досліджень відображені на рис.А5.10.

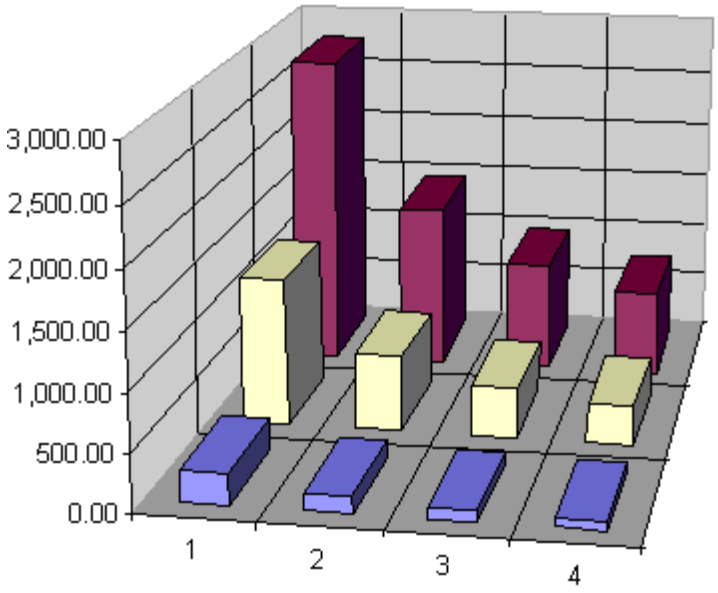


Рис. А5.10. Залежність часу обробки від кількості процесорів

А5.3 Висновки

Таким чином, можливо зробити наступні висновки.

1. Другій рівень деталізації дає хорошу якість, але потрібує значно більше часу для трасування. Особливо помітне затримання процесу в центрі картинки, де проміні трасують декілька об’єктів.

2. Результати досліджень показують можливість попереднього підбору ракурсу з використанням першого рівня деталізації за короткий час.

3. Використання паралельних обчислювань дає значне скорочення часу обробки.

Лекція А6
РОЗРОБКА АЛГОРИТМУ ФІЛЬТРАЦІЇ ДЛЯ КОМПОЗИТНОГО ПРИСТРОЮ БАГАТОКАНАЛЬНОЇ АРХІТЕКТУРИ

Складності побудови багатопроцесорних систем, що використовують паралельно працюючі синтезатори окремих фрагментів екранного зображення, полягають у наступному. По-перше, передача даних у буфери різних рівнобіжних процесорів вимагає високопродуктивної мережі даних між процесорами, а також твердої синхронізації. По-друге, у деяких сценах більшість примітивів з бази даних у процесі синтезу можуть попадати в єдиний фрагмент. У той же час, для інших сцен примітиви в переповнених фрагментах вимушено розміщаються в більш ніж одному блоці растрирования, а потім численні часткові зображення повинні бути з'єднані в єдиний образ. Хоча це ускладнює процес растризации, таке змішання зображень може бути виконано композицією різних зображень в один буфер наприкінці растризации.

Центральною ідеєю тут є правильний розподіл примітивів. Численні синтезатори синхронізовані, тому що вони використовують ідентичні матриці перетворень і обчислюють однакові кадри в однаковий час. Кожен синтезатор обчислює своє часткове зображення незалежне і записує це часткове зображення у власний буфер кадру. Чи дерево конвеєр композитних пристроїв зливає RGB і z – потоки з кожного синтезатора в один буфер кадру.

А6.1 Злиття зображень

Fussell [5] запропонував проект, що використовує бінарне дерево розподілу примітивів по процесорах. Додатково до запропонованої схеми

генерації зображень Shaw, Green і Schalffer [SHAW91] додали алгоритм усунення ступінчастості (антиелайсинг). В останньому підході VLSI-композитні пристрої змішують зображення в кожнім вузлі композитного бінарного дерева. Ці композитні пристрої виконують спрощену версію алгоритму Duff, передбачаючи придатну обробку багатьох випадків перекриття часткових зображень. Недоліком їхнього підходу є недосконаліть алгоритму антиелайсинга.

А6.2 Фільтрація на стику фрагментів зображень

У даному пункті будуть розглянуті алгоритми, об'єднані загальною ідеєю фільтрації ковзним вікном, чи апертурою. Термін "локальна" підкреслює той факт, що розміри вікна по обох осях менше відповідних розмірів фильтруемого зображення.

Метою локальної фільтрації як правило є підвищення якості зображення; найчастіше це усунення чи перешкод підвищення різкості, підкреслення контурів.

Розглянемо найбільше часто употребимие алгоритми фільтрації: лінійні і нелінійні.

Принципи локальної фільтрації

Для фільтрації зображення використовують двовимірні фільтри, що відповідають двовимірній природі самого зображення (картинка на екрані монітора).

Такий двовимірний фільтр улаштований у такий спосіб. Береться невеликий, звичайно прямокутний, ділянка площини і на ньому визначається деяка функція. Ця ділянка називається апертурою, чи вікном, а задана на ньому функція - ваговою функцією, чи функцією вікна. Таким чином, кожному елементу апертури відповідає визначене число, називане ваговим множником. Сукупність усіх вагових множників і складає вагову функцію.

Апертура разом із заданої на ній ваговою функцією часто називається маскою.

Апертура звичайно має невеликий розмір: 3x3 чи 5x5 елементів. Лінійні розміри апертури звичайно беруться непарними, щоб можна було однозначно вказати її центральний елемент.

Фільтрація здійснюється переміщенням апертури фільтра по зображенню. У кожнім положенні апертури виконуються однотипні дії, що і визначають відгук фільтра. Вагова функція в процесі переміщення вікна також залишається незмінної.

Характерним прикладом служить алгоритм лінійної фільтрації, що складає загалом у наступному. При кожнім положенні апертури вагова функція поелементно збільшується на значення відповідних пикселей вихідного зображення; добутку складуються. Сума поділяється на коефіцієнт, що нормалізує, і отримана величина, що є відгуком фільтра привласнюється тому пикселю нового (профільтрованого) зображення, що відповідає положенню центра апертури. Коефіцієнт, що нормує, звичайно береться рівним сумі всіх елементів вагової чи функції 1.

Поряд з лінійними фільтрами існують і нелінійні. Характер фільтра залежить від операцій, виконуваних у кожнім положенні вікна.

У лінійних фільтрах відгук є лінійною функцією багатьох перемінних, роль яких грають попавшие у вікно пиксели. Фільтри, у яких відгук не може бути виражений лінійною функцією від значень елементів зображення, є нелінійними.

Лінійні фільтри

Як випливає з назви відгук лінійного фільтра лінійним образом залежить від оброблюваного зображення. Розглянемо фільтри , у яких для кожного положення апертури здійснюється заелементне перемножування вагової функції на значення відповідних елементів зображення, підсумовування добутків і нормировка отриманої суми.

Уведемо необхідні позначення. Нехай апертура має розмір Mp x Mq елементів; поточний елемент апертури позначимо через (p,q), де

p=1,2,...,Mp – поточна рядок;

q=1,2,...,Mq – поточний стовпець.

Положення апертури на зображенні найчастіше задається центральною крапкою (тому в основному апертура береться непарного розміру). У цій роботі я надійшов також.

Поточне положення умовного центра на вихідному зображенні F позначимо через (i,j). Відгук фільтра привласнюється тій же крапці (i,j) нового, профильтрованого полючи Q.

Позначимо через H(p,q) функцію вікна. Масив Q вихідного зображення формується шляхом дискретної згортки вхідного полючи F і функції вікна H(p,q):

$$Q(i,j)=\sum_{p=1}^{Mp}\sum_{q=1}^{Mq} F(i-p_m+p,j-q_m+q)H(p,q)$$

де p_m, q_m - умовний центр апертури.

Різні види лінійних фільтрів відрізняються своїми ваговими функціями і коефіцієнтами, що нормують.

Розглянемо тепер ряд конкретних прикладів і приведемо найбільше употребимие маски.

Одне з найбільш розповсюджених застосувань лінійних фільтрів – згладжування шуму. Для цього застосовуються вагові функції наступного виду:

	1	1	1		1	1	1
H=1/9	1	1	1	i H=1/10	1	2	1
	1	1	1		1	1	1

За результатами досвідів можна сказати, що застосування цих фільтрів дає непоганий результат з погляду придушення шуму, однак зображення

виходить розмитим і для одержання гарного результату потрібно застосування контрастування зображення.

Для підкреслення ліній визначеного напрямку використовується маска виду:

а) $H=1/16\begin{vmatrix}1&2&1\\2&4&2\\1&2&1\end{vmatrix}$ і б) $H=1/16\begin{vmatrix}2&1&2\\1&4&1\\2&1&2\end{vmatrix}$

Вагова функція (а) підкреслює великими вагами четирехсвязние елементи вихідного зображення (горизонтальні і вертикальні лінії), (б) – діагональні лінії.

Фільтри, що підкреслюють границі (у приватній інтерпретації), використовують наступні типові вагові функції:

а) $H=\begin{vmatrix}0&-1&0\\-1&5&-1\\0&-1&0\end{vmatrix}$ і б) $H=\begin{vmatrix}-1&-1&-1\\-1&9&-1\\-1&-1&-1\end{vmatrix}$ і в) $H=\begin{vmatrix}1&-2&1\\-2&5&-2\\1&-2&1\end{vmatrix}$

Відмітною рисою даних масок є те, що сума вагових множників тут дорівнює 1.

Для виділення перепадів визначеної орієнтації використовуються в залежності від необхідного напрямку вагові функції називані курсовими градиентними масками:

а) «північ» $H=\begin{vmatrix}1&1&1\\1&-2&1\\-1&-1&-1\end{vmatrix}$ і б) «північний схід» $H=\begin{vmatrix}1&1&1\\-1&-2&1\\-1&-1&1\end{vmatrix}$

в) «схід» $H=\begin{vmatrix}-1&1&1\\-1&-2&1\\-1&1&1\end{vmatrix}$ і г) «південний схід» $H=\begin{vmatrix}-1&-1&1\\-1&-2&1\\1&1&1\end{vmatrix}$

д) «південь» $H=\begin{vmatrix}-1&-1&-1\\1&-2&1\\1&1&1\end{vmatrix}$ і е) «південний захід» $H=\begin{vmatrix}1&-1&-1\\1&-2&-1\\1&1&1\end{vmatrix}$

ж) «захід» $H=\begin{vmatrix}1&1&-1\\1&-2&-1\\1&1&-1\end{vmatrix}$ і з) «північний захід» $H=\begin{vmatrix}1&1&1\\1&-2&-1\\1&1&-1\end{vmatrix}$

$$H=\begin{vmatrix}1 & -2 & -1\\ 1 & 1 & -1\end{vmatrix} \quad \text{і} \quad H=\begin{vmatrix}1 & -2 & -1\\ 1 & -1 & -1\end{vmatrix}$$

Назва курсу говорить про напрямок перепаду яскравості, що викликає максимальний відгук фільтра.

Для виділення перепадів без указівки їхньої орієнтації використовуються наступні три види вагових функцій (оператори Лапласа):

а) $H=\begin{vmatrix}0 & -1 & 0\\ -1 & 4 & -1\\ 0 & -1 & 0\end{vmatrix}$ і б) $H=\begin{vmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{vmatrix}$ і в) $H=\begin{vmatrix}1 & -2 & 1\\ -2 & 4 & -2\\ 1 & -2 & 1\end{vmatrix}$

Дані вагарні функції дозволяють здійснювати операцію двовимірного диференціювання. Такі вагові функції мають властивість: сума всіх їхніх елементів дорівнює 0.

Нелінійні фільтри

Розглянемо два види нелінійних фільтрів:

1. Медіанні застосовувані для згладжування зображення

2. Фільтри перепади яскравості, що підкреслюють - фільтри Робертса і Собела.

<u>Медіанний фільтр.</u>

Суть медіанного фільтра полягає в тому, що відгук такого фільтра дорівнює медіані даних находящихся в апертурі. Медіана являє собою центральний елемент у варіаційному ряді, отриманому з даних, що знаходяться в межах апертури.

Медіанні фільтри застосовуються для згладжування зображень і придушення шуму. Такі фільтри дуже ефективні для згладжування імпульсного шуму і на відміну від подібних лінійних фільтрів медіанні фільтри залишають різкі перепади.

<u>Алгоритм Робертса</u>

Використовується для підкреслення перепадів яскравості і виділення контурів.

Метод Робертса складається у використанні операції двовимірного дискретного диференціювання

$$G(i,j)=\sqrt{u^2+v^2};$$

Де $u=F(i,j)-F(i+1,j+1)$

$$v=F(i,j+1)-F(i+1,j)$$

іноді застосовують наступну формулу

$$G(i,j)=|u|+|v|$$

Як видно Робертс застосовує квадратну апертуру розміром 2x2.

Алгоритм Собела

Використовується для той же цілей що й алгоритм Робертса, але використовує апертуру 3x3

$$G(i,j)=\sqrt{u^2+v^2};$$

де

$$u=(F(i-1,j+1)+2F(i,j+1)+F(i+1,j+1))-(F(i-1,j-1)+2F(i,j-1)+F(i+1,j-1))$$

$$v=(F(i-1,j-1)+2F(i-1,j)+F(i-1,j+1))-(F(i+1,j-1)+2F(i+1,j)+F(i+1,j+1))$$

застосовують також наступну формулу

$$G(i,j)=|u|+|v|$$

При підготовці роботи виконане моделювання різних видів фільтрів.

Лекція А7

РОЗРАХУНОК ПРОДУКТИВНОСТІ СЦЕНАРНОГО ПРОЦЕСОРА БАГАТОКАНОЛЬНОЇ АРХІТЕКТУРИ

А7.1 Аналіз характеристик сценарного процесора

Для оцінки продуктивності сценарного процесора необхідно задатися якісним складом баз даних об'єктів і сцени в цілому, набором операторів, виконуваних при формуванні локальної бази кадру, і кількістю елементарних операцій, необхідних для виконання кожного оператора.

Нехай сцена включає K об'єктів. Кожен об'єкт складається з Кпо примітивних об'єктів (з них Kd переміщуваних), топологія об'єкта задана деревом, що містить Kv вузлів і Kl пріоритетних списків. Примітивний об'єкт обмежений Кгр гранями, кожна з який містить Кв вершин.

Як основну одиницю виміру пам'яті приймаємо байт. Оцінимо розмірність основних характеристик. Нехай лінійні і кутові координати займають по 4 байти, кількісні характеристики й ознаки - 1 байт.

Для обчислення sin() і cos() застосуємо табличний метод. Основне слово для роботи з пам'яттю вважаємо рівним 1 байт.

Уведемо позначення елементарних операцій (табл.А7.1). Структура сценарного процесора наведена на рис.А7.1.

Розглянемо послідовність операторів, виконуваних при формуванні локальної бази кадру, для кожного блоку сценарного процесора.

Блок попередньої обробки:

- "ПРИЙОМ 1": прийом вхідних значень: K, Pj (j=1..K), Рн;
- "МАТРИЦЯ В": формування матриці В;
- "ЦЕНТР": переклад лінійних координат центра об'єкта в систему координат спостерігача;

- "ПІРАМІДА": аналіз влучення об'єкта в піраміду видимості;
- "ДЕТАЛЬ": аналіз ступеня деталізації об'єкта.

Таблиця А7.1 - Позначення елементарних операцій

Операція	Позначення
Обмін з ЕОМ або попереднім блоком	R
Порівняння+умовний перехід	C
Логічна	O
Читання/запис пам'яті	L
Додавання/вирахування	A
Множення	M
Ділення	D

Відповідно до розглянутого раніше математичного вираженнями для виконання кожного оператора можна одержати кількісні оцінки, наведені в табл.А7.2.

Таблиця А7.2 - Кількості елементарних операцій для основних операторів блоку попередньої обробки

НАЗВА	Кількість елементарних операцій оператора						
	R	C	O	L	A	M	D
Прийом 1	25+34K	-	-	25+34K	-	-	-
Матриця В	-	-	4	216	4	16	-
Центр	-	-	-	120ДО	3ДО	9ДО	-
Піраміда	-	5ДО	6ДО	60ДО	5ДО	-	-
Деталь	-	1ДО	-	10ДО	-	-	-
Разом:	25+34K	6ДО	4+6ДО	241+224ДО	4+8ДО	16+9ДО	-

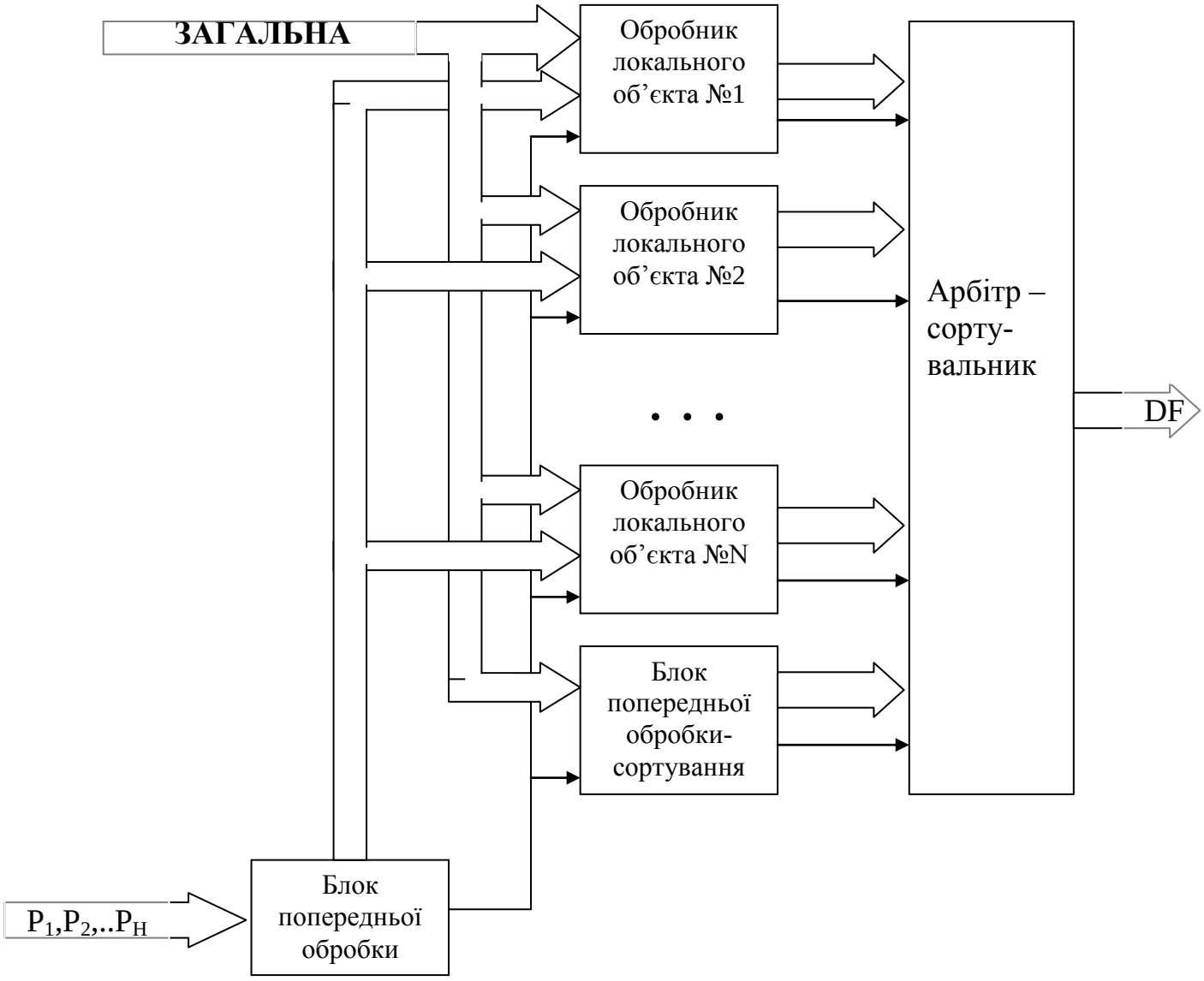


РИС.А7.1. СТРУКТУРА СЦЕНАРНОГО

У блоці попереднього сортування виконуються наступні операції:

- "ПРИЙОМ 2": прийом вхідних значень: K, Pj (j=1..K), Pн, V;
- "СПИСОК": формування списку пріоритетів об'єктів (сортування координат X центрів об'єктів);
- "ПЕРЕКРИТТЯ": аналіз взаємного перекриття об'єктів із сусідніми пріоритетами.

З огляду на математичні співвідношення для кожної операції, одержимо кількісні оцінки, наведені в табл.А7.3.

Таблиця А7.3 - Кількості операцій для блоку попереднього сортування

НАЗВА оператора	Кількість елементарних операцій						
	R	C	O	L	A	M	D
Прийом 2	25+36	-	-	25+36ДО	-	-	-
Список	-	ДО	-	9ДО	-	-	-
Перекриття	-	К-1	-	45ДО-45	7ДО-7	4ДО-4	-
Разом:	25+36	2ДО-1	-	90ДО-20	7ДО-7	4ДО-4	-

Розглянемо роботу арбітра/сортувальника. У випадку відсутності об'єктів, що перекривають один одного, арбітр/сортувальник виконує функції мультіплексора, тобто буде виконуватися тільки одна операція, трудомісткість якої наведена в табл. А7.4.

Таблиця А7.4 - Кількості елементарних операцій для арбітра/сортувальника в режимі мультіплексирування:

НАЗВА оператора	Кількість елементарних операцій	
	R	L
Мультіплексор	К[Кгр(3+7Кв)+1]	К[Кгр(3+7Кв)+1]

Відповідно до математичних співвідношень сортування двох перекриваючих друг друга об'єктів у гіршому випадку являє собою сортування 2Кгр граней, що виробляється над тривимірними координатами вершин граней об'єктів, необхідна кількість операцій, виконувана при цьому, наведена в табл. А7.5 .

Сумарна кількість операцій для сортування граней двох об'єктів наведена в табл.А7.6.

Таблиця А7.5 - Кількості елементарних операцій для основних етапів сортування арбітром/сортувальником

НАЗВА етапу	Кількість елементарних операцій						
	Коефіцієнт	R	C	L	A	M	D
Прийом координат	2Кгр	3+10Кв	-	3+10Кв	-	-	-
Рівняння площини	2Кгр-1	-	-	60	20	24	-
Проекція	Кв(2Кгр-1)	-	-	96	9	16	3
Аналіз перекриття	2Кв(2Кгр-1)	-	1	96	9	16	3

Таблиця А7.6 - Сумарна кількість операцій для сортування двох об'єктів

Операція	Позна-чення	Кількість
Обмін з ЕОМ	R	2Кгр (3+10Кв)
Порівняння/перехід	C	(2Кгр-1) 2Кв
Читання/запис пам'яті	L	404КгрКв+126Кгр-288Кв
Додавання/вирахування	A	27 Кв (2Кгр-1)
Множення	M	48 Кв (2Кгр-1)
Розподіл	D	9 Кв (2Кгр-1)

Аналіз залежностей табл. А7.7. показує, що найбільша кількість операцій виконується арбітром/сортувальником. Оцінимо необхідну

тривалість такту апаратних засобів для його побудови, виходячи з вимоги режиму реального часу (тривалість кадру 40 мс.).

Таблиця А7.7 - Сумарна кількість операцій по блоках

Операція	Ін.обр.	Ін. сорт.	Арбітр/сортувальник
Обмін	25+34К	25+36К	КвзКгр (3+10Кв) + К[Кгр(3+7Кв)+1]
Порівняння	6ДО	2К-1	(КвзКгр-1) 2Кв
Логіч.	4+6К	------	------
Читання	241+224К	90К-20	Квз(202КгрКв+63Кгр-144Кв) + К[Кгр(3+7Кв)+1]
Додавання	4+8К	7К-7	27 Кв (КвзКгр-1)
Множення	16+9К	4К-4	48 Кв (КвзКгр-1)
Ділення	------	------	9 Кв (КвзКгр-1)

Приймемо, що сценарний процесор обробляє сцену, що містить об'єкти з наступними параметрами:

кількість примітивних об'єктів Кпо = 128;
кількість граней Кгр = 1000;
кількість вершин грані Кв = 6;
кількість списків граней Кl = 32.

Аналіз завантаження блоків інструментальної системи синтезу показує, що кількість граней на виході блоку обробки видимої частини зображення знижується в середньому в 3 рази. Отже, на вхід арбітра/сортувальника з кожного локального обробника надходить порядку 350 граней.

У табл.А7.8. наведені тактові характеристики основних операцій МП 80486 і спеціалізованого процесора (дані отримані на підставі досвіду розробок спеціалізованих процесорів на базі секційних мікропроцесорів).

Таблиця А7.8 - Кількості тактів виконання операцій з плаваючої коми для МП 80486 і спецпроцесора

Операція	Позначення	МП	СП
Обмін з ЕОМ	R	¾	2
Порівняння+умовний перехід	C	¾	1
Логічна	O	3	1
Читання/запис пам'яті	L	5	1
Додавання/вирахування	A	10	1
Множення	M	11	4
Ділення	D	73	32

З урахуванням даних табл.А7.7., табл.А7.8. одержимо оцінку необхідної кількості операцій і тактів для обробки До об'єктів (табл.А7.9).

Аналіз табл. 9 дозволяє зробити наступні висновки. За час одного кадру арбітр/сортувальник, побудований на МП 80486 з тактовою частотою 66 Мгц , не зможе виконати сортування граней навіть одного об'єкта. Спеціалізований обчислювач з такою же тактовою частотою дозволить відсортувати грані двох об'єктів, що перекривають один одного, тобто значення Квз може бути не більш двох (для зазначеної тактової частоти). При цьому, з огляду на необхідність передбачати не менш чим 5% запас часу для надійної організації синхронізації, за частину кадру, що залишилася, можна виконати передачу в дисплейний файл екранних координат не більш 10 об'єктів.

Таким чином, запропонований сценарний процесор дозволяє обробляти сцену, що містить до десяти об'єктів, два з який можуть взаємно перекриватися.

Таблиця А7.9. Сумарна кількість операцій і тактів для обробки сцени

Блок попередньої обробки	
КІЛ операцій КІЛ тактів МП КІЛ тактів СП	(25+34K)R+6KC+(4+6K)O+(241+224K)L+ (4+8K)A+(16+9K)M 1500 K + 1500 380 K + 360
Блок попереднього сортування	
КІЛ операцій КІЛ тактів МП КІЛ тактів СП	(25+34K)R+(2K-1)C+(90K-20)L+(7K-7)A +(4K-4)M 700 K – 116 180 K + 18
Арбітр/сортувальник	
КІЛ операцій КІЛ тактів МП КІЛ тактів СП	(22050Квз+15750ДО)R+4200C+(56700Квз)A+(445386Квз+1575ДО)L+100800КвзM+18900КвзA 5 400 000 Квз + 70 000 K + 12 600 1 100 000 Квз + 34 000 K + 400

Затримка на виконання аналізу видимості 10 об'єктів у блоці попередньої обробки складає порядку 2500 тактів, що при тактовій частоті 66 Мгц не перевершує тисячної частки часу кадру. Т.ч. робота блоку попередньої обробки не внесе істотної затримки в початок роботи оброблювачів локальних об'єктів.

Лекція А8
СИСТОЛІЧНІ ПРОЦЕСОРА

А8.1 Визначення систолічного масиву

Систолічний масив - це обчислювальна мережа, що володіє наступними властивостями:

- синхронність: дані обчислюються ритмічно по єдиної синхросерії і пропускаються по мережі;
- модульність і регулярність: масив містить модульні процесорні елементи з однорідними зв'язками;
- конвеєрируемість: досягається прискорення виконання обчислень з коефіцієнтом прискорення к=Тп/Тк, де Тп - час обробки на одному процесорі; Тк - час обробки на конвеєрі.

Властивості систолічних архітектур:

1) простота і регулярність схеми - це важливо при інтегральному виконанні, крім того прості і регулярні схеми визначають модульність системи, що сприяє підвищенню продуктивності;

2) паралелізм і зв'язок: паралелізм збільшує продуктивність обчислювальних систем за рахунок розподілу обчислювального навантаження між декількома процесорами й одночасним виконанням задачі на цих процесорах; зв'язок здобуває значення при великій кількості працюючих одночасно процесорів, тому що вартість трасування в СВІС-технології перевищує всі інші витрати, тому регулярні і локальні зв'язки представляються вкрай корисними;

3) збалансованість обчислень із вводом/виводом - варто завжди співвідносити швидкість обчислень зі швидкістю в/в - найбільш вузького місця таких систем.

Переваги систолічної архітектури:

- модульність;
- регулярність;
- локальні міжз’єднання;
- високий ступінь мультіобробки;
- безупинний потік даних між процесорними елементами.

Недолік - глобальне керування за допомогою потактової синхронізації. Це призводить до:

- розфазування синхронізуючих імпульсів унаслідок різних довжин з'єднань генератора тактових імпульсів з кожним процесорним елементом, а також зміні параметрів сигналу в різних ланцюгах поширення;
- відказостійкості - найменший збій одного з процесорів приводить до повної зупинки системі;
- пікової потужності - навантаженню на момент обміну між процесорними елементами, коли можливі наведення в шинах живлення через великі струми перевантаження, викликаних одночасним поданням живлення чи зміною стану компонентів процесорів (наприклад, буферів).

Зазначені недоліки можна усунути застосуванням принципу організації функціонування процесорів з керуванням потоком даних.

Фоннеймонівська архітектура припускає наявність програмного правління, тобто виконання команд у відповідність із заданою програмою. У пристроях з керуванням потоком даних команда ставиться на виконання тільки в тому випадку, якщо активізовані її дані. Результат її виконання у свою чергу використовується іншими командами при активізації їхніх операндів і т.д. Тобто. виключається необхідність глобального керування і глобальної синхронізації.

А8.2 Приклад систолічної матриці для множення квадратних матриць

При множенні квадратних матриць

$$A = \begin{matrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{matrix} \quad \text{і} \quad B = \begin{matrix} в11 & в12 & в13 \\ в21 & в22 & в23 \\ в31 & в32 & в33 \end{matrix}$$

виходить матриця С, компоненти якої визначаються як:

$c11 = a11в11 + a12в21 + a13в31$

$c12 = a11в12 + a12в22 + a13в32$

$c13 = a11в13 + a12в23 + a13в33$

$c21 = a21в11 + a22в21 + a23в31$

$c22 = a21в12 + a22в22 + a23в32$

$c23 = a21в13 + a22в23 + a23в33$

$c31 = a31в11 + a32в21 + a33в31$

$c32 = a31в12 + a32в22 + a33в32$

$c33 = a31в13 + a32в23 + a33в33$

Організація систолічної матриці:

ПЕРЕЛІК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Raisa Malcheva. Application of Multilevel Design on the Base of UML for Digital System Developing // in book "Design of Digital Systems and Devices. Series: Lecture Notes in Electrical Engineering, Vol. 79. – Springer-Verlag Berlin Heildelberg, 2011. – PP. 93-117. (Зареєстрований Навч.мет. Радою ДонНТУ як навч. посібник, регистр. Протокол №2, від. 21.01.12).

2. Веселовська Г.В. Основи комп'ютерної графіки. Навчальний посібник (с грифом МОН України). – К.: Центр учбової літератури, 2004. – 392 с.

3. Вельтмандер П.В. Машинная графика. Учебное пособие в 3-х книгах. - Новосибирск: НГУ, 1997.

Кн.1. Вводный курс. - 123 с., ил.

Кн.2. Основные алгоритмы. - 193 с., ил.

Кн.3. Архитектуры графических систем. – 90 с., ил.

4. Foley J., Dam A. Computer Graphics. Principles and practice. - 2nd ed. In C. - AWPC, 1997. – 1175 p.

5. Головашин, В.Л. Основы компьютерной графики : учебное пособие / В.Л. Головашин, С.А. Вязовов, С.И. Лазарев. – Тамбов: Изд-во Тамб. гос. техн. ун-та, 2008. – 80 с. [pdf - версия]

6. S.J. Mellor and M.J. Balcer. Executable UML. A Foundation for Model - Driven Architecture. Indianapolis: Addison-Wesley, 2002.

Навчальне видання

Лекції з курсу «Апаратно-програмні засоби систем комп’ютерної графіки» для студентів заочної, заочної прискореної з наданням денних освітніх послуг і денної форм навчання напрямку підготовки 6.050102 «Комп’ютерна інженерія»

Укладач: Мальчева Раїса Вікторівна