Лекція №7

Тема: Апаратне прискорення растеризації тривимірного зображення OpenGL

Мета: Реалізувати каркасне тривимірних об’єктів за допомогою апаратного прискорювача OpenGL.

ПЛАН

1. Поняття Mesh

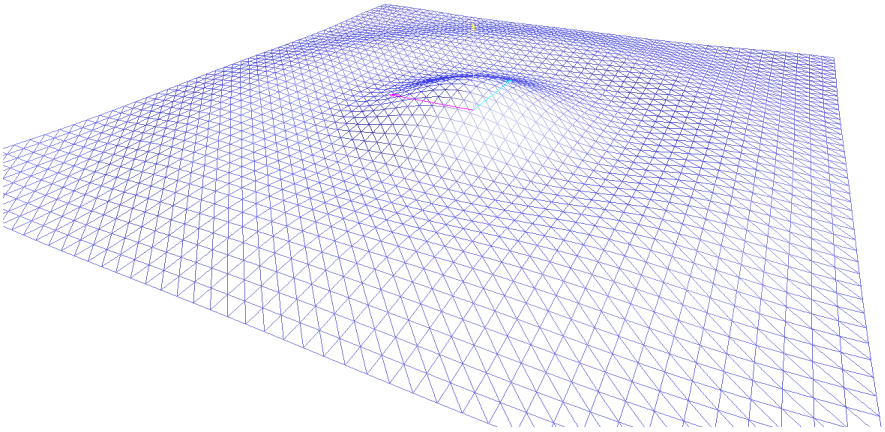

Рис. 1 — Полігональна сітка з пласких многокутників Mesh

В попередніх лекціях використовувалося представлення векторних об’єктів як множина вершин та множина полігонів, кути яких збігаються з переліченими вершинами. Це і є поняття мешу.

Полігональні сітки - набір полігонів (граней), які в сукупності формують оболонку об'єкта. Це стандартний спосіб візуального представлення широкого класу об’ємних фігур. Багато систем візуалізації засновані на зображенні об’єктів за допомогою малювання послідовності полігонів.

Переваги полігональних сіток засновані на простоті використання полігонів: легко задати і перетворювати. Мають прості властивості. В меші чітко визначені внутрішня і зовнішня області. Відносно спрощуються алгоритми малювання та зафарбовування полігонів або накладення текстури на плоску грань. Полігональні сітки дозволяють представляти тривимірні об’єкти практично будь-якого ступеня складності.

Мешами можна задати монолітні об’єкти і тонкі оболонки. Полігональні сітки дозволяють задавати об'єкти двох типів: монолітні (solid) об’єкти полігональні межі щільно прилягають один до одного і обмежують деякий простір. Прикладами тут є куб, сфера та інше. Тонкі оболонки виникають коли полігональні межі примикають один до одного без обмеження простору, представляючи собою поверхню нескінченно малої товщини. Приклад: графік функції z = f (x, y).

Кожен полігон визначається шляхом перерахування його вершин. Вершина задається за допомогою перерахування її координат у просторі. Вектор нормалі задає напрямок перпендикуляра полігону. При малюванні об’єкта ця інформація використовується для визначення того, скільки світла розсіюється на даній точці полігону.

Використання нормалей погано підходить для візуалізації гладких поверхонь, наприклад, сфери. Зручніше виявляється пов’язувати вектор нормалі з кожною вершиною поверхні. Такий спосіб спрощує процес відсікання і процес зафарбовування гладких криволінійних форм.

У OpenGL нормаль є атрибутом вершини. З точки зору. швидкодії вигідніше зберігати окрему копію вектора нормалі для кожної вершини. Одна і та ж вершина може входити до складу кількох суміжних граней. Тому краще зберігати все вершини сітки (з їх атрибутами) в окремому масиві. При завданні граней вказувати лише індекси використовуваних вершин.

2. TGLScene

під об'єктами GLScene я розумію об'єкти, не представлені у вкладках Delphi. Об'єкти GLScene можуть бути створені, відредаговані або видалені за допомогою Scene Editor.

Об'єкти розбиті за категоріями. Деякі об'єкти не віднесені до жодної з категорій і стоять окремо.

Розглянемо GLScene Editor, перш ніж перейдемо до опису об'єктів по порядку. GLScene Editor може бути відкритий за подвійним клацанням на компоненті GLScene в інспектора об'єктів Delphi. Тут за допомогою деревовидної структури представлені всі використовувані вами об'єкти. Спочатку буде тільки один запис Scene, розділений на два підрозділи: Cameras і Scene Objects. Кореневий запис Scene відноситься до GLScene.Objects і є базовим (самим верхнім) елементом в ієрархії GLScene. Правим кліком по Cameras і вибором Add camera на сцену, додається камера. Пізніше камера може бути переміщена за ієрархією дерева і може стати нащадком будь-якого об'єкта GLScene. Всі інші об'єкти будуть розміщені під записом Scene Objects.

Для створення об'єкта клацніть правою кнопкою миші по Scene Objects і виберіть потрібний вам об'єкт. Якщо ви хочете створити новий об'єкт як нащадок вже наявного в сцені, то клікніть правою кнопкою миші по цьому об'єкту. Новий об'єкт завжди поміщається на останню позицію в межах поточного рівня ієрархії.

Порядок проходження в ієрархічному дереві GLScene Editor важливий. Самий верхній об'єкт растеризується першим. Якщо у нього є нащадки, то растеризується об’єкт наступний за ним. Візуалізація триває від гілки до гілці, поки не буде досягнуто остову дерева.

Порядок проходження також важливий для прозорих об'єктів. Прозорі об'єкти завжди повинні растеризуватися останніми, інакше можна отримати візуальні невідповідності в сцені. Деякі об'єкти (як об'єкти HUD, або частки) можуть рендерится самостійно і, тому, неважливо, де вони будуть розташовані в ієрархічному дереві. Ви можете змінювати положення об'єкта сцени в дереві просто перетягуючи його на інший об'єкт (і відповідно роблячи об'єкт нащадком того, на який перетягнули), або правим кліком по об'єкту і вибираючи Move object down або Move object up.

Загальні властивості об'єктів. Всі об'єкти GLScene мають деякі загальні властивості, які використовуються особливо часто. Тут перераховані властивості, загальні для всіх об'єктів:

• Bechaviors - (поведінка) якщо ви додасте до об'єкта поводження, то об'єкт буде діяти відповідно до цієї поведінки. Зазвичай поведінка об'єкта відповідає за рух, звук або перевірку на зіткнення. Ви можете додати більше одної поведінки. Поведінка об'єктів використовується спільно з компонентами вкладки GLSceneUtils.

Список поведінок:

- Collision;

- Simple Inertia;
- Simple Acceleration;
- Sound Emitter;
- Movement controls;
- FPS Movement;
- DCE Static Collider;
- DCE Dynamic Collider;
- ODE Dynamic;
- ODE Static;
- ODE HeightField Collider;

• Children - (дитина) не зображається в інспекторі об'єктів. Список всіх нащадків даного об'єкта. До нащадка можна звернутися за індексом, який починається з нуля.

• Count - (кількість) не відображається в інспекторі об'єктів. Кількість нащадків об'єкта плюс одиниця, так як масив нащадків починається з нульового індексу.

• Direction - (напрямок) вектор, який вказує лицьову сторону об'єкта. Цей вектор є нормаль (його довжина дорівнює 1). Значення за замовчуванням - [0,0,1].

• Effects - (ефекти) кожен об’єкт може випускати ефект частинок. Якщо використано один з менеджерів PFX, то додавайте ефекти тут. Список доступних ефектів:

- PFXSource;
- FireFX;
- ThorFX;
- ExplosionFX;

• Objects Sorting - (сортування об'єктів) зміна порядку рендеру об’єктів сцени.

• Material - (матеріал) відповідає за забарвлення та розрахунок відбиття світла.

• Parent - (батько) не зображено в інспекторі об'єктів. Об’єкт рівнем вище в ієрархічному дереві.

• Pitch Angle - (кут нахилу) кут, який описує обертання об'єкта навколо осі Y, в градусах.

• Position - (позиція) вектор, який описує позицію об'єкта в 3D просторі. Значення за замовчуванням [0, 0, 0].

• Roll Angle - (кут ротації) кут, який описує обертання об'єкта навколо осі X, в градусах. Зміна цього параметра не завжди виробляє однаковий ефект. Рекомендується використовувати вектори Direction і Up, замість цього параметра.

• Scale - (масштаб) цим вектором встановлюється розмір об'єкта. Можна масштабувати об'єкт неоднорідно, наприклад, лише по осі X. Нащадки не успадкують значення масштабу.

Значення за замовчуванням [1, 1, 1].

• Show Axes - (показувати осі) булевий параметр включає видимість кольорових ліній, які є осями X, Y і Z обраного об'єкта. Корисно при формуванні сцени.

• Tag Float - (маркер точки) подібний відомому по Delphi параметру Tag, тільки тут має тип single.

• Turn Angle - (кут повороту) кут, що описує обертання об'єкта навколо осі Z, в градусах. Зміна цього параметра не завжди виробляє однаковий ефект. Рекомендується використовувати вектори Direction і Up, замість цього параметра.

• Up - (вгору) цей вектор, разом з вектором Direction, використовується для того, щоб описати обертання об'єкта в 3D просторі. Вектор Up завжди перпендикулярний вектору Direction і нормальний (має довжину 1). Значення за замовчуванням [0, 1, 0].

• Visibility Culling - (відбір видимості) Visibility culling використовується, щоб прискорити рендеринг сцени. Об'єкти, які не перебувають в полі видимості камери, швидко відкидаються. Але це працює лише в певних випадках. Visibility culling працює лише на базових об'єктах (трикутниках), які не ґрунтуються на полігонах.

• Visible - (видимість) можна показати або приховати потрібний об'єкт. Але будь-який код, представлений в подію onProgress, даного об'єкта виконується навіть з виключеною видимістю.

3. TGLCadencer

Більшість додатків, що використовують GLScene, зображуються (Рендер) в режимі реального часу. При цьому час має велике значення. Тому з’являється необхідність в деякому менеджері часу. І це не найпростіший компонент для рахування часу.

Спочатку потрібно знати скільки часу буде рендеритьcя сцена. Камера може бути спрямована на складні геометричні об’єкти з великою кількістю полігонів, і, при обертанні камери всі вони повинні перемальовуватись. Причому ваша програма може виконуватися як на старій і повільної системі, так і на новітній системі з високою продуктивністю. Цей момент неможливо передбачити заздалегідь.

Якщо передбачається використовувати сцену протягом довгого часу — використовуйте GLCadencer. Цей компонент подбає про необхідну синхронізацію поновлення об’єктів в сцені від кадру до кадру. Але спочатку потрібно налаштувати властивості цього компоненту.

Процес перемальовування (рендеринга) кадру призводить до виникнення події Progress компонента GLScene. Кожен об’єкт GLScene має подію onProgress, де можна запрограмувати деякі дії програми, що виконуються кожен раз при перемальовуванні сцени. Подвійним кліком на об’єкті в інспекторі об’єктів до основному коду додається шаблон коду реакції на подію onProgress.

Процедура Progress передає через параметри одну важливу змінну — deltaTime. Це період часу в секундах, який пройшов після рендерінгу останнього кадру. Якщо цей параметр дуже великий, то значить, що сцена повільно рендерится і “гальмує”.

Ідеальна кількість рендеренгу кадрів — 30 в секунду. При цьому deltaTime дорівнює 0,033333. Якщо необхідно провести які-небудь обчислення пов'язані з часом — включайте змінну deltaTime в ваші рівняння. Наприклад, якщо переміщується куб уздовж осі X зі швидкістю 10 пунктів в секунду, то код буде виглядати приблизно так:

GLCube.Position.X: = GLCube.Position.X + 10 * deltaTime.

Cadencer має властивість enabled. З його допомогою можна просто включити або вимкнути компонент в потрібний момент. Коли він вимкнений сцена буде заморожена. Cadencer може працювати в декількох різних режимах (GLCadencer.Mode). CmASAP — значення за замовчуванням, сцена буде оброблятися щоразу з максимальним пріоритетом, в порівнянні з іншими процесами. cmIdle — сцена буде оброблятися тільки якщо завершені інші процеси і з cmManual ви зможете управляти запуском обробки сцени вручну. Інша цікава особливість — Cadencer.minDeltaTime. З допомогою цієї властивості ви можете встановити час, тільки після закінчення якого, почнеться обробка сцени, навіть якщо сцена вже побудована. З цим можна розвантажити систему. Cadencer.maxDeltaTime — навпаки не дозволить cadencer виконатися швидше встановленого часу.

4. TGLSceneViewer

Компонент являє собою прямокутну панель, в якій зображається сцена. Її розміри не обмежують саму сцену. Чим більше розтягнута ця панель, тим повільніше промальовується сцена. Для зображення сцени в панелі необхідно вказати у властивостях камеру (TGLCamera), зображення з якої буде вимальовуватися в вашому GLSceneViewer і власне саму сцену (TGLScene). Ви можете встановити тут важливу властивість — контекст рендерингу через властивість Buffer. Однак значення цієї властивості по замовчуванням в більшості випадків цілком достатньо.

5. TGLCamera

Камеру можна представити у вигляді точки в тривимірному просторі, звідки розглядається сцена. Камера, як і інші об'єкти, має вектори position, direction і up. З їх допомогою можна рухати і обертати камеру по сцені. Простіший і ефективний метод орієнтації камери в 3D просторі - це направити її на певний об'єкт. Для цього виберіть потрібний об'єкт у властивості Target, і камера завжди буде спрямована на цей об'єкт, як б не переміщалася і оберталася по сцені.

FieldOfView - параметр, який змінює фокусну відстань “лінзи” камери. Більш низькі значення роблять кут огляду сцени ширше, а більш високі наближають видимість (zoom). Переконайтеся, що не використовуються занадто великі або маленькі значення, інакше камера викривить простір сцени.

Також потрібно знати дещо про обмежувальної площині (culling planes). Полігони, розташовані занадто далеко або, навпаки, занадто близько по відношенню до камери не виводяться. Межа, яка ділить сцену на видимі і невидимі частини, називаються ближньою (near) і далекою (far) обмежувальною площиною відповідно. Можна встановити ці значення через параметри NearFrustrumRange і FarFrustrumRange.

Зазвичай змінюють лише далеку обмежувальну площину. Але це створює невелику проблему. Оскільки камера наближається до об’єкту занадто швидко, то камера проскакує дальню обмежувальну площину. Для зменшення цього небажаного ефекту часто використовують туман. Туман можна включити в GLSceneViewer.Buffer туман (fog). Туман має зону початку і кінця. Будь-який полігон, що знаходиться між цими зонами змінює свою прозорість від повної непрозорості в початковій зоні до повної прозорості в кінцевій. Зробіть значення зони кінця туману таким же, як і далека обмежувальна площина, а колір туману таким же як колір фону GLSceneViewer і небажаний ефект пропаде.

6. TGLLightSource

Без світла сцена відтворюється темна і не кольорова. Світло робить її світлою і яскравою. Максимум можна мати вісім джерел світла. Будь-яке світло, окрім паралельного, має межу дальності світіння. Від джерела світла до межі його світіння, світло від цього джерела поступово зменшується. Це називається ослабленням світла.

Світло від LightSource не створює тіней. Якщо, наприклад джерело світла направлений на сферу, а за нею знаходиться площина, то на площині тіні не побачимо. Світло проходить через сферу її не помічаючи. Для отримання тіні потрібно використовувати інші методики. Наприклад, Lightmaps, Z-Shadows або Shadow Volumes.

Існує три типи світла:

1. Omni Light - джерело світла знаходиться в певній точці. Промені від нього розходяться радіально в усіх напрямках. Як аналог можна взяти електричну лампочку, яка висить десь на дроті.

2. Spot Light - одиничний промінь світла або конус спрямованого світла. Є можливість змінити ширину і кут світла. Якщо змінити кут на 360 °, то світло стане типу Omni. Прикладом Spot Light є світло ліхтарика.

3. Parallel Light - однорідна маса паралельних променів з однаковим напрямком, які світять від деякої площини в нескінченності. Зміна позиції паралельного джерела світла не дає ніякого ефекту. Паралельний світ зазвичай використовують для симуляції рівномірного освітлення.

7. TGLFreeForm

Цей об'єкт використовується досить часто. Він здатний завантажувати геометрію з різних форматів сітки (mesh). Щоб формат зміг використовуватися, потрібно додати в розділ uses однойменний модуль формату файлу. Наприклад, щоб використовувати формат *.3ds, необхідно додати модуль GLFile3DS. Для завантаження геометрії використовується функція LoadFromFile конкретного GLFreeForm. Координати текстур зазвичай включені в файл. Деякі формати підтримують мультітекстурування (використання декількох текстур одночасно для одного об’єкта) об’єктів. Ви повинні встановити використовувані текстури в список Mesh list. Для успішного завантаження текстури геометрія повинна завантажитися перед нею.

8. TGLMaterial

Material Library - це компонент для зберігання матеріалів. Дозволяє отримати доступ до матеріалів через їх індекс або ім'я збереженого матеріалу. Кожен об’єкт, на який може бути

накладено матеріал, має однойменну властивість — material. Ви можете редагувати матеріал безпосередньо в інспекторі об’єктів. За подвійним кліком на значку крапки навпроти властивості material - відкривається редактор матеріалів (Material Editor). Редактор матеріалів має три вкладки і вікно з прикладом у вигляді куба з накладеним вами матеріалом. Три вкладки - це:

1. Front properties - редагує якість матеріалу лицьових граней об’єкта. Дифузний колір (Diffuse) є найбільш важливим. Він визначає колір освітлених частин об'єкта. Навколишній колір (Ambient) визначає колір затінених частин об'єкта. Дзеркальний колір (Specular) «відповідає» за колір віддзеркалень і відблисків. Колір емісії (Emission) визначає колір світіння GLScene керівництво новачка, Jat-Studio, 2009 об’єкта. Але для зміни основного кольору змінювати потрібно саме дифузний колір (Diffuse). Інші властивості матеріалу, як зображення або відблиски, можуть бути розраховані більш реалістично за допомогою використання шейдерів.

2. Back properties - відміну від front properties в тому, що ці властивості “Відповідають” за матеріал невидимих граней об'єкта. Ці межі об’єкта стають видимими, тільки якщо матеріал прозорий або відключений відбір (Culling) задніх граней.

3. Texture - використовується для накладення на об'єкт текстури. Для включення видимості текстури необхідно відключити властивість disabled. Цю властивість по замовчуванням включено, що означає, що об'єкт не використовує ніякої текстури. Об’єкт в цьому випадку буде пофарбований у відповідності з налаштуваннями двох попередніх вкладок (Back і Front properties). Якщо потрібно застосувати текстуру, то вмикають прапорець disabled і завантажують зображення. GLScene підтримує формати jpg, tga, bmp. При використанні двох перших форматів необхідно спочатку в розділі коду uses додати модулі jpeg або tga відповідно. Для реалістичного освітлення потрібно встановити для властивості Texture Mode текстури значення tmModulate. Світло повинне освітити об'єкт, щоб матеріал стало видно. Інша важлива річ — розміри текстури повинні бути кратні двом: 2,4,8,16,32,64,128,256,512,1024 і т. д. Причому довжина і ширина текстури можуть і не збігатися. Наприклад, можна використовувати текстуру розміром 32x512. Якщо ж використовувати текстуру нестандартного розміру, то вона буде зображена повільніше, так як GLScene доведеться привести її розміри до стандарту.

Внизу редактора матеріалів є список, що випадає для вибору режиму змішування — blending mode. Тут можна визначити, як матеріал буде змішуватися або перекриватися іншими матеріалами. Непрозорий режим змішування (BmOpaque): непрозорий об'єкт. Прозорий режим (bmTransparent) дозволить бачити крізь об’єкт. Об’єкт може бути однорідно

прозорим, або прозорість може бути задана текстурою. Сукупне змішування (bmAdditive) комбінує колір об’єкту з кольором об’єктів позаду нього. Хоча для кожного об’єкта можна налаштувати свій матеріал, але рекомендується зберігати всі матеріали в бібліотеці матеріалів (GLMaterialLibrary). Особливо якщо об’єктів багато і деякі використовують одну і ту ж текстуру. Наприклад, використовується 100 кубиків і для кожного завантажується одна і та ж текстура. В пам’яті розмістяться 100 однакових текстур. Але можна завантажити цю текстуру один раз в MaterialLibrary і потім посилатися на неї кожен раз, коли вона необхідна. Робиться це з допомогою коду GLCube.Material.MaterialLibrary для звернення до бібліотеки матеріалів або ж GLCube.Material.LibMaterialName для звернення до імені матеріалу, який потрібен.

Бібліотека матеріалів має ще одну зручну функцію: AddTextureMaterial. В цій функції визначається ім’я нового матеріалу і завантажувати в нього зображення (Текстура). Новий матеріал додається до бібліотеки так:

```
Texture.Disabled: = False; i Texture.Modulation: = tmModulate;
```