лекция №7

Тема: Аппаратное ускорение растеризации трехмерного изображения OpenGL

цель: Реализовать каркасное трехмерных объектов с помощью аппаратного ускорителя OpenGL.

ПЛАН

1. Понятие Mesh

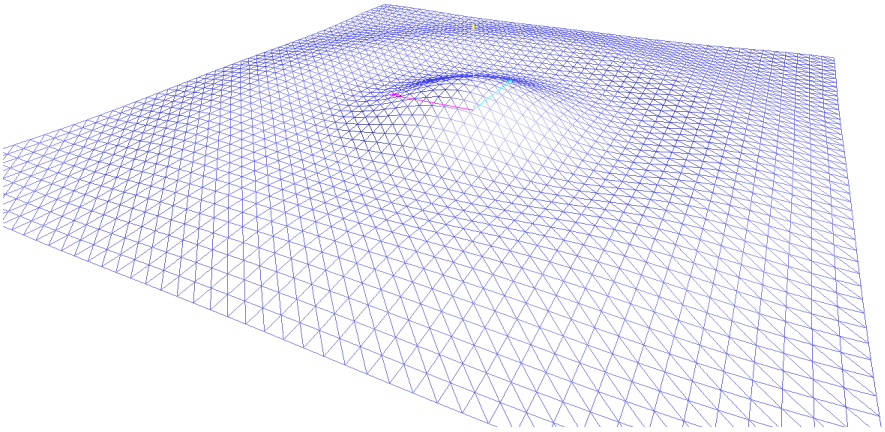

Рис. 1 - Полигональная сетка из плоских многоугольников Mesh

В предыдущих лекциях использовалось представление векторных объектов как множество вершин и множество плагинов, углы которых совпадают с перечисленными вершинами. Это и есть понятие Мешу.

Полигональные сетки - набор плагинов (граней), которые в совокупности формируют оболочку объекта. Это стандартный способ визуального представления широкого класса объемных фигур. Многие системы визуализации основаны на картинке объектов с помощью рисования последовательности полигонов.

Преимущества полигональных сеток основаны на простоте использования полигонов: легко задать и преобразовывать. Имеют простые свойства. В Меши четко определены внутренняя и внешняя области. Относительно упрощаются алгоритмы рисования и закрашивания полигонов или наложение текстуры на плоскую грань. Полигональные сетки позволяют представлять трехмерные объекты практически любой степени сложности.

Мешамы можно задать монолитные объекты и тонкие оболочки. Полигональные сетки позволяют задавать объекты двух типов: монолитные (solid) объекты полигональные грани плотно прилегают друг к другу и ограничивают некоторое пространство. Примерами здесь есть куб, сфера и прочее. Тонкие оболочки возникают когда полигональные границы примыкают друг к другу без ограничения пространства, представляя собой поверхность бесконечно малой толщины. Пример: график функции z = f (x, y).

Каждый полигон определяется путем перечисления его вершин. Вершина задается с помощью перечисления ее координат в пространстве. Вектор нормали задает направление перпендикуляра полигона. При рисовании объекта эта информация используется для определения того, сколько света рассеивается на данной точке полигона.

Использование нормалей плохо подходит для визуализации гладких поверхностей, например, сферы. Удобнее оказывается связывать вектор нормали с каждой вершиной поверхности. Таким образом упрощает процесс отсечения и процесс окрашивания гладких криволинейных форм.

В OpenGL нормаль является атрибутом вершины. С точки зрения. быстродействия выгоднее хранить отдельную копию вектора нормали для каждой вершины. Одна и та же вершина может входить в состав нескольких смежных граней. Поэтому лучше хранить все вершины сетки (с их атрибутами) в отдельном массиве. При задании граней указывать только индексы используемых вершин.

2. TGLScene

под объектами GLScene я понимаю объекты, не представленные во вкладках Delphi. Объекты GLScene могут быть созданы, отредактированы или удалены с помощью Scene Editor.

Объекты разбиты по категориям. Некоторые объекты не отнесены ни к одной из категорий и стоят особняком.

Рассмотрим GLScene Editor, прежде чем перейдем к описанию объектов по порядку. GLScene Editor может быть открыт по двойному щелчку на компоненте GLScene в инспекторе объектов Delphi. Здесь с помощью древовидной структуры представлены все используемые вами объекты. Сначала будет только одна запись Scene, разделен на два подразделения: Cameras и Scene Objects. Корневой запись Scene относится к GLScene.Objects и является базовым (самым верхним) элементом в иерархии GLScene. Правым кликом по Cameras и выбором Add camera на сцену, добавляется камера. Позже камера может быть перемещена по иерархии дерева и может стать потомком любого объекта GLScene. Все остальные объекты будут размещены под записью Scene Objects.

Для создания объекта щелкните правой кнопкой мыши по Scene Objects и выберите нужный вам объект. Если вы хотите создать новый объект как потомок уже имеющегося в сцене, то кликните правой кнопкой мыши по этому объекту. Новый объект всегда помещается на последнюю позицию в пределах текущего уровня иерархии.

Порядок прохождения в иерархическом дереве GLScene Editor важен. Самый верхний объект растеризуеться первым. Если у него есть потомки, то растеризуеться объект следующий за ним. Визуализация продолжается от ветви к ветви, пока не будет достигнуто остова дерева.

Порядок прохождения также важен для прозрачных объектов. Прозрачные объекты всегда должны растеризуватися последними, иначе можно получить визуальные несоответствия в сцене. Некоторые объекты (как объекты HUD или частицы) могут рендерится самостоятельно и, поэтому, неважно, где они будут расположены в иерархическом дереве. Вы можете изменять положение объекта сцены в дереве просто перетаскивая его на другой объект (и соответственно делая объект потомком того, на какой перетащили) или правым кликом по объекту и выбирая Move object down или Move object up.

Общие свойства объектов. Все объекты GLScene имеют некоторые общие свойства, которые используются особенно часто. Здесь перечислены свойства, общие для всех объектов:

• Bechaviors - (поведение) если вы добавите к объекту обращения, то объект будет действовать в соответствии с этой поведения. Обычно поведение объекта отвечает за движение, звук или проверку на столкновения. Вы можете добавить более одной поведения. Поведение объектов используется совместно с компонентами вкладки GLSceneUtils.

Список поведений:

- 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 - (ребенок) не изображается в инспекторе объектов. Список всех потомков данного объекта. К потомку можно обратиться по индексу, который начинается с нуля.

• Count - (количество) не отображается в инспекторе объектов. Количество потомков объекта плюс единица, так как массив потомков начинается с нулевого индекса.

• Direction - (направление) вектор, указывающий лицевую сторону объекта. Этот вектор является нормаль (его длина равна 1). Значение по умолчанию - [0,0,1].

• Effects - (эффекты) каждый объект может выпускать эффект частиц. Если использовано один из менеджеров PFX, то добавляйте эффекты здесь. Список доступных эффектов:

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

• Objects Sorting - (сортировка объектов) изменение порядка рендера объектов сцены.

• Material - (материал) отвечает за окраску и расчет отражение света.

• Parent - (отец) не показаны в инспекторе объектов. Объект уровнем выше в иерархическом дереве.

• Pitch Angle - (угол наклона) угол, который описывает вращение объекта вокруг оси Y, в градусах.

• Position - (позиция) вектор, который описывает позицию объекта в 3D пространстве. Значение по умолчанию [0, 0, 0].

• Roll Angle - (угол ротации) угол, который описывает вращение объекта вокруг оси X, в градусах. Изменение этого параметра не всегда производит одинаковый эффект. Рекомендуется использовать векторы Direction и Up, вместо этого параметра.

• Scale - (масштаб) этим вектором устанавливается размер объекта. Можно масштабировать объект неоднородно, например, только по оси X. Потомки не наследуют значение масштаба.

Значение по умолчанию [1, 1, 1].

• Show Axes - (показывать оси) булево параметр включает видимость цветных линий, которые являются осями X, Y и Z выбранного объекта. Полезно при формировании сцены.

• Tag Float - (маркер точки) подобный известному по Delphi параметру Tag, только здесь имеет тип single.

• Turn Angle - (угол поворота) угол, описывающий вращение объекта вокруг оси Z, в градусах. Изменение этого параметра не всегда производит одинаковый эффект. Рекомендуется использовать векторы Direction и Up, вместо этого параметра.

• Up - (вверх) этот вектор, вместе с вектором Direction, используется для того, чтобы описать вращения объекта в 3D пространстве. Вектор Up всегда перпендикулярен вектору Direction и нормальный (имеет длину 1). Значение по умолчанию [0, 1, 0].

• Visibility Culling - (отбор видимости) Visibility culling используется, чтобы ускорить рендеринг сцены. Объекты, которые не находятся в поле видимости камеры, быстро отбрасываются. Но это работает только в определенных случаях. Visibility culling работает только на базовых объектах (треугольниках), которые не основываются на полигонах.

• Visible - (видимость) можно показать или скрыть нужный объект. Но любой код, представленный в событии onProgress, данного объекта выполняется даже с выключенной видимостью.

3. TGLCadencer

Большинство приложений, использующих GLScene, изображаются (Рендер) в режиме реального времени. При этом время имеет большое значение. Поэтому появляется необходимость в некотором менеджере времени. И это не самый компонент для счета времени.

Сначала нужно знать сколько времени рендериться сцена. Камера может быть направлена на сложные геометрические объекты с большим количеством полигонов, и, при вращении камеры все они должны перерисовываться. Причем ваша программа может выполняться как на старой и медленной системе, так и на новой системе с высокой производительностью. Этот момент невозможно предсказать заранее.

Если предполагается использовать сцену в течение долгого времени - используйте GLCadencer. Этот компонент позаботится о необходимой синхронизации обновления объектов в сцене от кадра к кадру. Но сначала нужно настроить свойства этого компонента.

Процесс перерисовки (рендеринга) кадра приводит к возникновению события Progress компонента GLScene. Каждый объект GLScene имеет событие onProgress, где можно запрограммировать некоторые действия программы, выполняются каждый раз при перерисовке сцены. Двойным кликом на объекте в инспекторе объектов к основному кода добавляется шаблон кода реакции на событие onProgress.

Процедура Progress передает через параметры одну важную переменную - deltaTime. Это период времени в секундах, прошедшее после рендеринга последнего кадра. Если этот параметр очень большой, то значит, что сцена медленно рендерится и «тормозит».

Идеальное количество рендеренгу кадров - 30 в секунду. При этом deltaTime равна 0,033333. Если необходимо провести какие-либо вычисления связанные со временем - включайте переменную deltaTime в ваши уравнения. Например, если перемещается куб вдоль оси X со скоростью 10 пунктов в секунду, то код будет выглядеть примерно так:

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

Cadencer имеет свойство enabled. С его помощью можно просто включить или выключить компонент в нужный момент. Когда он выключен сцена будет заморожена. Cadencer может работать в нескольких различных режимах (GLCadencer.Mode). CmASAP - значение по умолчанию, сцена будет обрабатываться каждый раз с максимальным приоритетом, по сравнению с другими процессами. cmIdle - сцена будет обрабатываться только если завершены другие процессы и с cmManual вы сможете управлять запуском обработки сцены вручную. Другая интересная особенность - Cadencer.minDeltaTime. С помощью этого свойства можно установить время, только после окончания которого, начнется обработка сцены, даже если сцена уже построена. С этим можно разгрузить систему. Cadencer.maxDeltaTime - наоборот не позволит cadencer исполниться быстрее установленного времени.

4. TGLSceneViewer

Компонент представляет собой прямоугольную панель, в которой изображается сцена. Ее размеры не ограничивают саму сцену. Чем больше растянута эта панель, тем медленнее прорисовывается сцена. Для изображения сцены в панели необходимо указать в свойствах камеру (TGLCamera), изображение с которой будет вырисовываться в вашем GLSceneViewer и собственно саму сцену (TGLScene). Вы можете установить здесь важное свойство - контекст рендеринга через свойство Buffer. Однако значение этого свойства по умолчанию в большинстве случаев вполне достаточно.

5. TGLCamera

Камеру можно представить в виде точки в трехмерном пространстве, откуда рассматривается сцена. Камера, как и другие объекты, имеет векторы position, direction и up. С их помощью можно двигать и вращать камеру по сцене. Простой и эффективный метод ориентации камеры в 3D пространстве - это направить ее на определенный объект. Для этого выберите нужный объект в свойства Target, и камера будет направлена на этот объект, как бы не перемещалась и вращалась по сцене.

FieldOfView - параметр, который меняет фокусное расстояние "линзы" камеры. Более низкие значения делают угол обзора сцены шире, а более высокие приближают видимость (zoom). Убедитесь, что используются слишком большие или маленькие значения, иначе камера искривит пространство сцены.

Также нужно знать кое-что о ограничительной плоскости (culling planes). Полигоны, расположенные слишком далеко или, наоборот, слишком близко по отношению к камере не выводятся. Предел, которая делит сцену на видимые и невидимые части, называются ближней (near) и дальней (far) ограничительной плоскостью соответственно. Можно установить эти значения через параметры NearFrustrumRange и FarFrustrumRange.

Обычно меняют только дальнюю ограничительную плоскость. Но это создает небольшую проблему. Поскольку камера приближается к объекту слишком быстро, то камера проскакивает дальнюю ограничительную плоскость. Для уменьшения этого нежелательного эффекта часто используют туман. Туман можно включить в GLSceneViewer.Buffer туман (fog). Туман имеет зону начала и конца. Любой полигон, находящийся между этими зонами изменяет свою прозрачность от полной непрозрачности в начальной зоне до полной прозрачности в конечной. Сделайте значение зоны конца тумана таким же, как и далека ограничительная плоскость, а цвет тумана таким же как цвет фона GLSceneViewer и нежелательный эффект пропадет.

6. TGLLightSource

Без света сцена воспроизводится темная и не цветная. Свет делает ее светлой и яркой. Максимум можно иметь восемь источников света. Любое свет, кроме параллельного, имеет предел дальности свечения. От источника света до предела его свечение, свет от этого источника постепенно уменьшается. Это называется ослаблением света.

Свет от LightSource не создает теней. Если, например источник света направлен на сферу, а за ней находится плоскость, то на плоскости тени не увидим. Свет проходит через сферу ее не замечая. Для получения тени нужно использовать другие методики. Например, Lightmaps, Z-Shadows или Shadow Volumes.

Существует три типа света:

1. Omni Light - источник света находится в определенной точке. Лучи от него расходятся радиально во всех направлениях. Как аналог можно взять электрическую лампочку, которая висит где-то на проволоке.

2. Spot Light - единичный луч света или конус направленного света. Есть возможность изменить ширину и угол света. Если изменить угол на 360 °, то свет станет типа Omni. Примером Spot Light свет фонарика.

3. Parallel Light - однородная масса параллельных лучей с одинаковым направлением, которые светят от некоторой плоскости в бесконечности. Изменение позиции параллельного источника света не дает никакого эффекта. Параллельный мир обычно используют для симуляции равномерного освещения.

7. TGLFreeForm

Этот объект используется довольно часто. Он способен загружать геометрию из различных форматов сетки (mesh). Чтобы формат смог использоваться, нужно добавить в раздел uses одноименный модуль формата файла. Например, чтобы использовать формат * .3ds, необходимо добавить модуль GLFile3DS. Для загрузки геометрии используется функция LoadFromFile конкретного GLFreeForm. Координаты текстур обычно включены в файл. Некоторые форматы поддерживают мультитекстурованнння (использование нескольких текстур одновременно для одного объекта) объектов. Вы должны установить используемые текстуры в список Mesh list. Для успешной загрузки текстуры геометрия должна загрузиться перед ней.

8. TGLMaterial

Material Library - это компонент для хранения материалов. Позволяет получить доступ к материалам через их индекс или имя сохраненного материала. Каждый объект, на который может быть

наложен материал, имеет одноименное свойство - material. Вы можете редактировать материал непосредственно в инспекторе объектов. По двойным кликом на значке точки напротив свойства material - открывается редактор материалов (Material Editor). Редактор материалов имеет три вкладки и окно с примером в виде куба с наложенным вами материалом. Три вкладки - это:

1. Front properties - редактирует качество материала лицевых граней объекта. диффузный цвет (Diffuse) является наиболее важным. Он определяет цвет освещенных частей объекта. Окружающий цвет (Ambient) определяет цвет затененных частей объекта. Зеркальный цвет (Specular) «отвечает» за цвет отражений и бликов. Цвет эмиссии (Emission) определяет цвет свечения GLScene руководство новичка, Jat-Studio, 2009 объекта. Но для изменения основного цвета изменять нужно именно диффузный цвет (Diffuse). Другие свойства материала, как изображение или блики, могут быть рассчитаны более реалистично с помощью использования шейдеров.

2. Back properties - отличие от front properties в том, что эти свойства "Отвечают" за материал невидимых граней объекта. Эти границы объекта становятся видимыми, только если материал прозрачный или отключен отбор (Culling) задних граней.

3. Texture - используется для наложения на объект текстуры. для включения видимости текстуры необходимо отключить свойство disabled. Это свойство по умолчанию включено, что означает, что объект не использует никакой текстуры. Объект в этом случае будет окрашен в соответствии с настройками двух предыдущих вкладок (Back и Front properties). Если нужно применить текстуру, то включают флажок disabled и загружают изображения. GLScene поддерживает форматы jpg, tga, bmp. При использовании двух первых форматов необходимо сначала в разделе кода uses добавить модули jpeg или tga соответственно. Для реалистичного освещения нужно установить для свойства Texture Mode текстуры значение tmModulate. Свет должен осветить объект, чтобы материал стал виден. Другая важная вещь - размеры текстуры должны быть кратны двум:

2,4,8,16,32,64,128,256,512,1024 и т. Д. Причем длина и ширина текстуры могут и не совпадать. Например, можно использовать текстуру размером 32x512. Если же использовать текстуру нестандартного размера, то она будет изображена медленнее, так как GLScene придется привести ее размеры к стандарту.

Внизу редактора материалов является выпадающий список для выбора режима смешивания - blending mode. Здесь можно определить, как материал будет смешиваться или перекрываться другими материалами. Непрозрачный режим смешивания (BmOpaque): непрозрачный объект. Прозрачный режим (bmTransparent) позволит видеть сквозь объект. Объект может быть однородно

прозрачным, или прозрачность может быть задана текстурой. Совокупное смешивания (bmAdditive) комбинирует цвет объекта с цветом объектов позади него. Хотя для каждого объекта можно настроить свой материал, но рекомендуется хранить все материалы в библиотеке материалов (GLMaterialLibrary). Особенно если объектов много и некоторые используют одну и ту же текстуру. Например, используется 100 кубиков и для каждого загружается одна и та же текстура. В памяти разместятся 100 одинаковых текстур. Но можно скачать эту текстуру один раз в MaterialLibrary и потом ссылаться на нее каждый раз, когда она необходима. Делается это с помощью кода GLCube.Material.MaterialLibrary для обращения в библиотеку материалов или GLCube.Material.LibMaterialName для обращения к имени материала, который нужен.

Библиотека материалов имеет еще одну удобную функцию: AddTextureMaterial. В этой функции определяется имя нового материала и загружать в него изображения (Текстура). Новый материал добавляется в библиотеку так:

Texture.Disabled = False; и Texture.Modulation = tmModulate;