2. *Головня А. И.* Омонимия как системная категория языка: Автореферат…канд.фил.наук: 10.02.19//БГУ. – Мн., 1996.
3. *Грязнухина Т. А*. Синтаксический анализ научного текста на ЭВМ. – Киев: Научная мысль, 1991.
4. *Малаховский Л. В*. Теория лексической и грамматической омонимии. – Л.: Наука, 1990.
5. *Мельчук Ч. А.* Автоматический синтаксический анализ. – Новосибирск, 1964.

О НЕКОТОРЫХ ОСОБЕННОСТЯХ ПРОГРАММИРУЕМОГО ГРАФИЧЕСКОГО КОНВЕЙЕРА DIRECTX 10 ДЛЯ GPU ЧЕТВЕРТОГО ПОКОЛЕНИЯ

А. А. Козик

ВВЕДЕНИЕ

Компьютерные игры в качестве коммерческого продукта индустрии развлечений, появившись относительно недавно, очень существенно эволюционировали во многих отношениях, особенно в способах и возможностях обрабоки графической информации и real-time визуализации.

Кроме роста вычислительных мощностей и соврешествования периферийных устройств особого внимания заслуживает развитие графического конвейера. Наилучшей иллюстрацией этого являются компьютерные игры, в которых требуется обрабатывать не менее 30 раз в секунду сотни тысяч вершин, дополнять это различным набором эффектов, выполняемых как на уровне вершин, так на и уровне пикселей. Примером может служить имитация сложных поверхностей с помощью normal mapping и parallax mapping, динамический расчет освещения и затенения объектов на основе трассировки лучей, передача свойств материала с помощью specular mapping и lightmap. Кроме того, производится расчет некоторых физических параметров, проверка на пересечение между объектами, проигрывание аудио и взаимодействие с пользователем и др.

Трехмерные изображения, представленные совокупностью вершин с тремя координатами, проецируются на двумерный экран. Кроме того каждая вершина может иметь доолнительные характеристики, такие как цвет, текстурные координаты, и любые другие, необходимые программисту данные. Для пререхода к 2D-пиксельной матрице требуется выполнение сложных расчетов. Ранее расчетом и преобразованием трехмерных сцен занимался CPU, однако, учитывая загруженность другими задачами, для повышения производительности произошло отделение задач по расчетам проекции на программном и

аппаратном уровне. Тогда появилось понятие GPU – графического процессора, на который с развитием технологий перекладывалось все больше задач по расчету графических преобразований и обработке изображений. В настоящее время архитектуру GPU делят на два конвейера программируемый и непрограммируемый. Программируемый конвейер позволяет разработчику непосредственно влиять на процесс преобразования и обработки набора вершин (вершинного буфера) и полученных результатов – пиксельного изображения. Предметом нашего рассмотрения являются особенности программируемого конвейера GPU 4-го поколения.

Для разработки графики компьютерных игр наиболее прогрессивным API на сегодняшний день является Microsoft Direct3D 10.

ГРАФИЧЕСКИЙ КОНВЕЙЕР

Графический конвейер – разделенный на этапы процесс, выполняемый на GPU, необходимый для представления трехмерного объекта на экране.

Этапы графического конвейера GPU 4G:

1. Input-Assembler Stage (стадия Ввода-Сборки).
2. Vertex Shader Stage (стадия вершинного шейдера).
3. Geometry Shader Stage (стадия геометрического шейдера).
4. Stream-Output Stage (стадия поточного вывода).
5. Rasterizer Stage (стадия растеризации).
6. Pixel Shader Stage (стадия пиксельных шейдеров).
7. Output-Merger Stage (стадия вывода).

INPUT-ASSEMBLER STAGE

Целью стадии Ввода-Сборки является получение данных о примитивах из входного вершинного буфера и преобразования их к формату, используемому остальными этапами конвейера.

Вершины группируются в списки линий (line lists) или треугольники (triangle strips). Для работы с геометрическим шейдером, введено два дополнительных типа примитивов: треугольники со смежными вершинами (до 3-х) (triangle list with adjacency) и линии со смежными вершинами (до 2-х) (line list with adjacency).Эта стадия, кроме сборки, отвечает за привязку семантических меток к еще необработанным данным, что ускоряет последующую обработку.После обработки входных примитивов данные отправляются на этап вершинных шейдеров.

VERTEX SHADER STAGE

Этот этап является обязательным. Шейдером называется программа, выполняемая непосрдственно на графическом процессоре. Вершинный шейдер обрабтывает только одну вершину за проход. Через вершинный шейдер проходят все вершины, в том числе и «смежные». Обрабатывать вершины требуется для того, чтобы выполнять операции трансформации, скининга, морфинга, и по-вершинного освещения.

Пример вершинного шейдера:

```
PS_INPUT VS(VS_INPUT Data : POSITION){
   //Данный шейдер выполняет преобразование координат
   PS_INPUT Out;
   Out = (PS_INPUT)0;
   //Умножая их на матрицы:
   Out.Pos = mul(Data.Pos, World);  //Мировая матрица
   Out.Pos = mul(Data.Pos, View);  //Матрица вида
   Out.Pos = mul(Out.Pos, Projection);//Матрица проекции
   return Out;
}
```

GEOMETRY SHADER STAGE

Геометрический шейдер введен в 10-й версии DirectX. В отличие от вершинного шейдера, геометрический, обрабатывает примитив целиком.

Важнейшим преимуществом этого шейдера является возможность непрерывно генерировать новую геометрию, отправляя результат обратно, через поточный вывод. Это открывает возможности для real-time обработки и генерации поверхностей, шерсти, волос; динамики частиц и флюидов, улучшенния работа с материалами и др.

```
//Геометрический шейдер рассчитывает нормаль к треугольнику
void GSQuadmain(triangle PSQuadIn input[3], inout
                    TriangleStream<PSQuadIn> QuadStream) {
     PSQuadIn output;
     float4 fNormalA = input[1].pos.xyzw -input[0].pos.xyzw;
     float4 fNormalB = input[2].pos.xyzw – input[0].pos.xyzw;
     float3 fNormal = cross(faceNormalA, faceNormalB);
     fNormal = normalize(faceNormal);
     for(int i=0; i<3; i++){
          output.pos = input[i].pos;
          output.color = color1;
          output.tex = input[i].tex;
          QuadStream.Append(output);
     }
QuadStream.RestartStrip();
}
```

STREAM-OUTPUT STAGE

На этом этапе реализуется поточный вывод результата в память, который в дальнейшем может быть отправлен на стадию Ввода-Сборки, либо возвращен на обработку CPU. Этот этап является логическим дополнением геометрического шейдера. Результаты обработки могут возвращаться одновременно в несколько буферов. Если результат геометрического шейдера проходит по конвейеру дальше, то он попадает на стадию растеризаци.

RASTERIZER STAGE

На этом этапе векторная графика преобразуется в растровое изображение. Производится проверка глубины и альфа-канала. Необходимые данные запоминаются для последующей обработки на отставшихся стадиях конвейера. Данная стадия не практически не отличается от аналогов в предыдущих версиях. Однако в следующей версии, разработчики обещают сделать возможным использование программируемого растеризатора, что откроет еще больше возможностей для визуализации графики.

PIXEL SHADER STAGE

Обработка пикселей на этом этапе не является обязательной. Стадия пиксельной обработки позволяет закрашивать объекты, применять текстуру, осуществлять по-пиксельное освещение, а также использовать различные эффекты.

При программной эмуляции обработка осуществляется непосредственно по пикселю, при аппаратном ускорении, как правило сэмплами размером 2x2 пикселя. На входе пиксельный шейдер получает и пиксельную и векторную информацию, необходиую для просчета семлирования, отсечений, программного расчета глубин и пр. На выходе получаются обработанные RGBA пиксели, направляемые на последнюю стадию.

```
//Шейдер закрашиват объект в белый цвет RGBA(1, 1, 1, 1)
float4 PS(PS_INPUT Pos) : SV_Target {
   return float4(1.0f, 1.0f, 1.0f, 1.0f);
}
```

OUTPUT-MERGER STAGE

На этой стадии определяется необходимость отрисовки (ближайший к камере) и окончательный цвет каждого пикселя. При необходимости происходит смешивание диффузного цвета (и прозрачности) каждого пикселя. В итоге изображение помещается в бэк-буфер, который по

окончанию отрисовки копирует изображение в первичный буфер, согласно методике двойной буферизации.

Большинство описанных операций программируются вручную пока только на языке HLSL 4.

ЗАКЛЮЧЕНИЕ

Благодаря введенному геометрическому шейдеру появилась возможность генерирования новых примитивов, позволяющих быстро и качественно создавать реалистичные эффекты погодных эффектов, дыма огня. С помощью геометричеких шейдеров гораздо удобнее манипулировать генерируемыми экземлярами объектов.

Существует множество примеров использования аппаратного ускорения графики на базе Direct3D 10, в таких графических движках, как CryEngine v1.0 и v2.0, Unreal Engine 3, Torque Game Engine Advanced v1.8.0, Jupiter EX и других. Данные технологии были использованы при разработке таких игровых тайтлов как: Crysis, FarCry2, Gears of War 1, 2, Assassin's Creed, Unreal Tournament 3, F.E.A.R 1, 2 и многих других.

Корпорация Microsoft анонсировала Direct3D 11, который будет включен в финальную версию грядущей Windows 7. В 11-ой версии API еще больше этапов графического конвейера станет программируемым, что позволит существенно улучшить итоговый продукт.

Литература

1. Интернет-адрес: http://www.developer.nvidia.com/.
2. Интернет-адрес: http://www.gamedev.com/.
3. Интернет-адрес: http://www.gamedev.ru/.
4. Интернет-адрес: http://www.msdn.microsoft.com/.

ФРАЗЕОЛОГИЧЕСКИЕ ЕДИНИЦЫ В ПЕРЕВОДЕ (НА ПРИМЕРЕ РОМАНА Т.ДРАЙЗЕРА «ТИТАН»)

А. А. Косовец

Английский язык имеет тысячелетнюю историю. За это время в нем накопилось большое количество выражений, которые люди нашли удачными, меткими и красивыми. Так и возник особый слой языка – фразеология. Мир фразеологии современного английского языка велик и многообразен, и каждый аспект его исследования, безусловно, заслуживает должного внимания.

Устойчивые сочетания употребляются в разговорной речи, художественной литературе, публицистике, выступают выразительным стилистическим средством. Фразеологизмы – изюминки народной мудрости, национальное богатство любой страны, они оживляют речь, делают ее об-