Отчет по лабораторной работе №6 по курсу «компьютерная графика»
Студент группы 08-306: Баскаков О.А.
Преподаватель: Измайлов А.А.
Работа выполнена: "22" декабря 2009 г.

Тема: Создание шейдерных эффектов.

Цель работы

Программно сгенерировать заданную поверхность , наложить на нее подходящую по размеру текстуру и применить указанный эффект. Обеспечить просмотр объекта с разных точек, динамическую загрузку текстуры из файла, включение и отключение шейдерного эффекта, регулирование скорости анимации эффекта, если необходимо

Задача

Вариант 1. Поверхность: x=u cosv; y=u sinv; z=v;

Эффект: Анимация. Координата X изменяется по закону X=Xcost;

Оборудование

Ноутбук с предустновленным QT Creator, видеокарта ATI.

Теоретическая часть

Ше́йдер (англ. *Shader*, изначально исользовались для просчета теней) — программа для одной из ступеней графического конвейера, используемая в трёхмерной графике для определения окончательных параметров объекта или изображения. Она может включать в себя произвольной сложности описание поглощения и рассеяния света, наложения текстуры, отражение и преломление, затенение, смещение поверхности и эффекты пост-обработки.

Программируемые шейдеры гибки и эффективны. Сложные с виду поверхности могут быть визуализированы при помощи простых геометрических форм. Технологии, подобные Occlusion Mapping позволяют задасть рехмерный объект(кирпичная кладка) одной текстурой.

Ход решения

Программа использует вершинный и фрагментный шейдеры.
Из основной программы вершины передаются и обрабатываются в вершинном шейдере. Затем они отправляются в фрагментный, где происходит наложение текстур.
Также существует возможность включения и выключения анимации с помощью нажатия колеса мыши. Также предусмотрена возможность изменения скорости анимации и загрузки текстуры из файла.

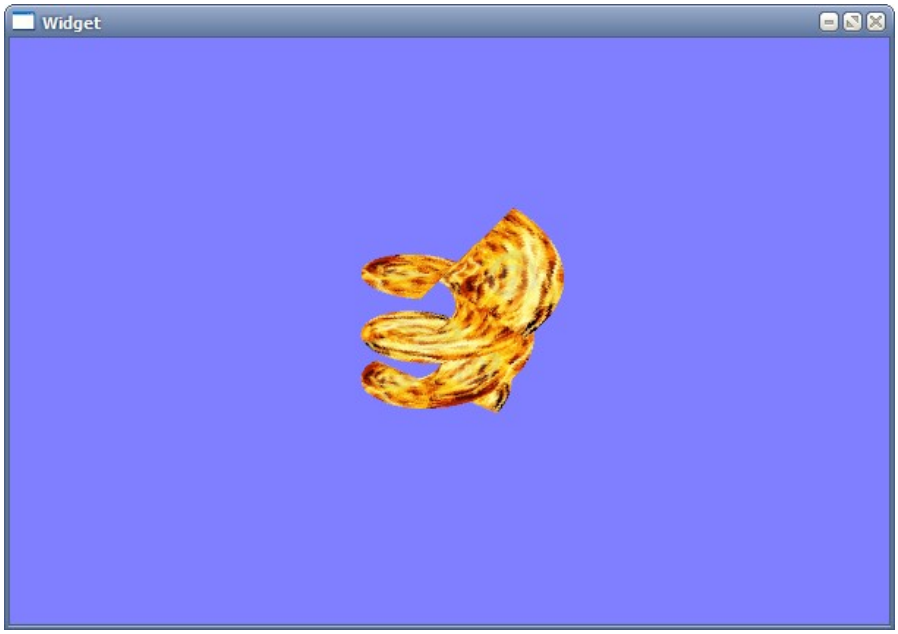
Вершинный шейдер:

```
uniform float t;
void main(){
    gl_TexCoord[0] = gl_MultiTexCoord0;
    vec4 pos;
    pos = gl_Vertex;
    pos.x = pos.x*cos(t);
    gl_Position = gl_ModelViewProjectionMatrix * pos;
}
```

Фрагментный шейдер:

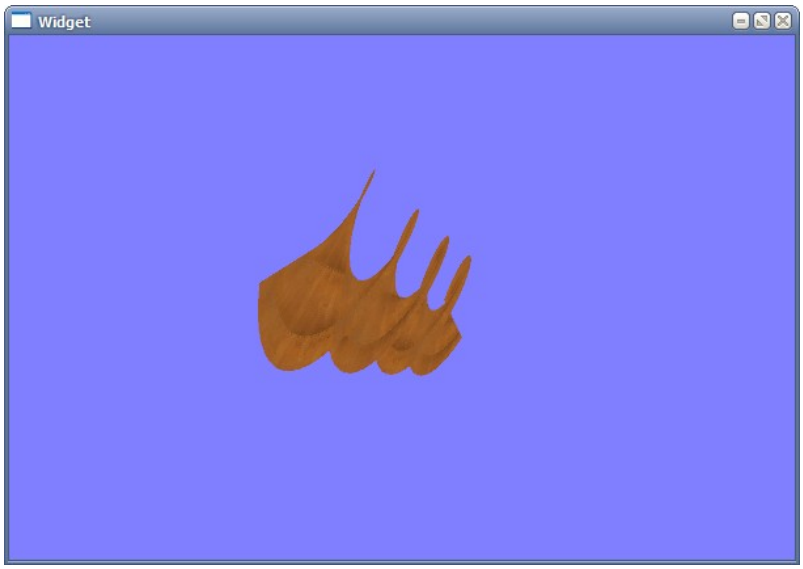
```
uniform sampler2D tex;

void main(){
    vec3 texColor = vec3(texture2D(tex, gl_TexCoord[0].st));
    gl_FragColor = vec4 ( texColor, 1.0);
}
```

Пример работы программы:



Смена текстуры

Анимация



Исходный код

В программе используются следующие исходники:

```
void Widget::setupShaders(){
    programm.addShaderFromSourceFile(QGLShader::Vertex,"../sample.vert");
    programm.addShaderFromSourceFile(QGLShader::Fragment,"../sample.frag");
    programm.link();
    programm.bind();
}

void Widget::initializeGL(){
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glViewport(0,0,this->width(),this->height());
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(100,100,100,0,0,0,0,0,1);
    glClearColor(0.5,0.5,1,1);
    glEnable(GL_DEPTH_TEST);
    tex=bindTexture(QString("tex1.jpg"),GL_TEXTURE_2D,GL_RGBA);
    setupShaders();
}

void Widget::on_pushButton_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this, "Open Texture", "",
"Image Files (*.png *.jpg *.bmp)");
        if (fileName != "")
        {
            tex = bindTexture(fileName, GL_TEXTURE_2D, GL_RGB);
            time->start(ui->horizontalSlider->value());
            glEnable(GL_TEXTURE_2D);
        }
}
...
```

Замечания

Благодаря интеграции шейдеров и OpenGL упрощается работа с графикой. Но при компиляции требуется последняя версия кроссплатформенной среды разработки QT.

Выводы

В ходе работы была написана программа, отображающая поверхность. Анимация поверхности осуществляется с помощью шейдеров. Почти все современные графические процессоры являются программируемыми и поддерживают два типа программ: вершинные программы (или вершинные шейдеры), определяющие последовательность операций для преобразования координат и атрибутов вершин объектов, и фрагментные программы (пиксельные шейдеры), вычисляющие цвет каждого пикселя. Применение шейдеров позволяет в реальном времени создавать сложные визуальные эффекты, включающие нетривиальный расчет освещения для каждого пикселя, имитацию неровных поверхностей при помощи карт нормалей, моделирование отражающих и преломляющих объектов, расчет динамических теней и объемного тумана и многие другие эффекты, доступные ранее только для алгоритмов трассировки лучей.