Advanced Games Engine Creation

Textures

In this tutorial you will

- Learn how to load, enable and render a texture
- Experiment with texture parameters and coordinates

Getting Started

Make a new copy of the OpenGL_SDL Base Project Framework and set up the paths.

Above the Render() method of GameScreenLevel1, add a method to draw a textured square as follows:

```
void DrawTextured2DSquare()
{
    glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 0.0f);
        glVertex3f(-0.5f, -0.5f, 0.0f);
        glTexCoord2f(1.0f, 0.0f);
        glVertex3f(0.5f, -0.5f, 0.0f);
        glTexCoord2f(1.0f, 1.0f);
        glVertex3f(0.5f, 0.5f, 0.0f);
        glTexCoord2f(0.0f, 1.0f);
        glVertex3f(-0.5f, 0.5f, 0.0f);
    glEnd();
}
```

Replace the triangle-drawing code in the Render() method with the following

```
glColor3f(1.0f, 1.0f, 1.0f);
glPushMatrix();
    glScalef(5.0f, 5.0f, 0.0f);
    DrawTextured2DSquare();
glPopMatrix();
```

Run your program. You should see a white square in the middle of the window.

Defining a Texture Class

We will now define a Texture class to load and hold the texture.

First create a file Texture.h with the following content:

```
#ifndef _TEXTURE2D_H
#define _TEXTURE2D_H
#include <windows.h>
#include <GL\glu.h>

class Texture2D
{
private:
    GLuint _ID; //Texture ID
    int _width, _height;

public:
    Texture2D();
    ~Texture2D();

    bool Load(char* path, int width, int height);

    GLuint GetID() const { return _ID; }
    int GetWidth() const { return _width; }
    int GetHeight() const {return _height; }
};

#endif //_TEXTURE2D_H
```

The file Texture.cpp should start with the following include statements:

```
#include "Texture2D.h"
#include <iostream>
#include <fstream>

using namespace::std;
```

Define the constructor and destructor as follows:

```
Texture2D::Texture2D()
{
}

Texture2D::~Texture2D()
{
    glDeleteTextures(1, &_ID);
}
```

Add the Load() function defined below. Go through it line by line and make sure you understand what is happening.

```
bool Texture2D::Load(char* path, int width, int height)
{
    char* tempTextureData; int fileSize; ifstream inFile;
    _width = width; _height = height;
    inFile.open(path, ios::binary);

    if (!inFile.good())
    {
        cerr  << "Can't open texture file " << path << endl;
        return false;
    }

    inFile.seekg (0, ios::end); //Seek to end of file
    fileSize = (int)inFile.tellg(); //Get current position in file - The End, this gives us total file size
    tempTextureData = new char [fileSize]; //Create an new aray to store data
    inFile.seekg (0, ios::beg); //Seek back to beginning of file
    inFile.read (tempTextureData, fileSize); //Read in all the data in one go
    inFile.close(); //Close the file

    cout << path << " loaded." << endl;

    glGenTextures(1, &_ID); //Get next Texture ID
    glBindTexture(GL_TEXTURE_2D, _ID); //Bind the texture to the ID
    glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, tempTextureData);

    delete [] tempTextureData; //Clear up the data - We don't need this any more
    return true;
}
```

Test your program – it should still compile and run, but you won't see anything new.

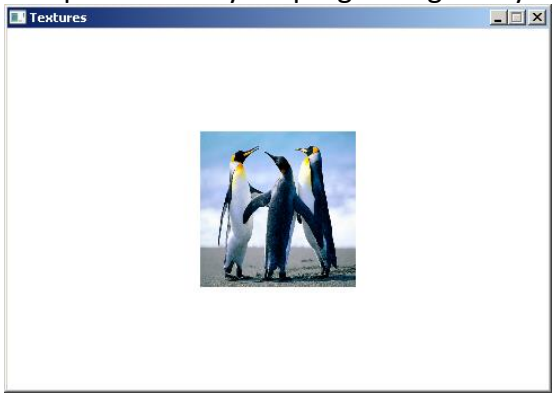
Enabling the texture

You will now set up and enable the OpenGL texturing functionality. Add the following to the end of the GameScreenLevel1 constructor:

```
glEnable(GL_TEXTURE_2D);

Texture2D* texture = new Texture2D();
texture->Load("Penguins.raw", 512, 512);

glBindTexture(GL_TEXTURE_2D, texture->GetID());

//set some parameters so it renders correctly
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

Download the "Penguins.raw" image file from the module website into the same folder as your project .cpp files.

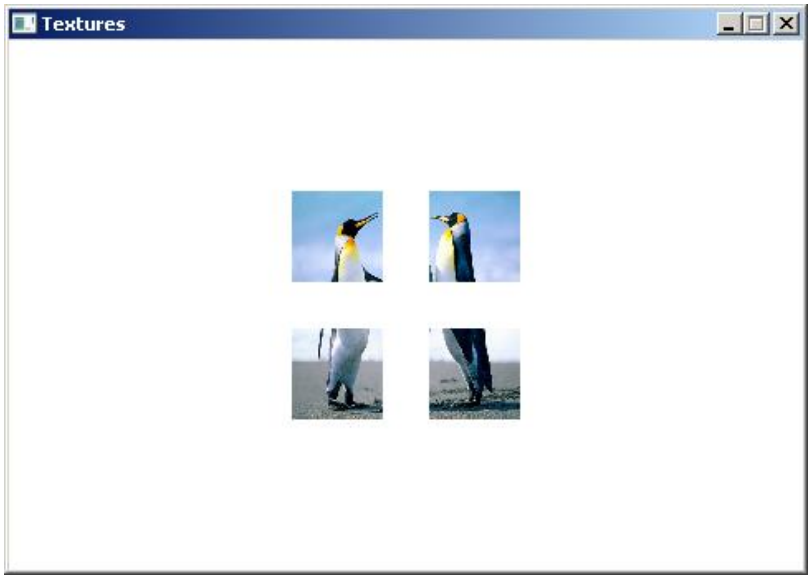
Compile and run your program again – you should see some penguins!



Experiment

Try the following:
- Using glTranslatef and glScalef, position and draw multiple textured squares of different sizes.
- change the drawing colour of each square, so you draw tinted penguins
- download and apply the stars.raw texture from Blackboard
- experiment with using the GL_NEAREST rather than GL_LINEAR filter – what effect does it have on large and small images?

So far we have mapped the entire texture on to the square. Try mapping a different portion of it to each square. Can you reproduce the image below? Experiment with texturing other shapes, such as triangles or triangle strips.



Read the section "Repeating and Clamping a Texture" at the end of chapter 9 of the Red Book. http://glprogramming.com/red/chapter09.html Experiment with using texture coordinates outside the range [0,1] and different arguments for glTexParameter*()

As discussed in the lecture, experiment with creating, loading and rendering your own .raw images. You can use the bmpReader.c or tgaLoader.cpp, available on Blackboard, to convert .bmp or .tga files to the .raw format.