Simple Keytime Animation

Mike Bailey
mjb@cs.oregonstate.edu

This work is licensed under a
Creative Commons
Attribution-NonCommercial-NoDerivatives 4.0
International License

Keyframing

Keyframing involves creating certain key positions for the objects in the scene, and then the program later interpolating the animation frames in between the key frames.

In hand-drawn animation, the key frames are developed by the senior animators, and the in-between frames are developed by the junior animators.

In our case, you are going to be the senior animator, and the computer will do the in-betweening.

class Keytimes:

```cpp
    void AddTimeValue( float time, float value );
    float GetFirstTime( );
    float GetLastTime( );
    int GetNumKeytimes( );
    float GetValue( float time );
    void PrintTimeValues( );
```

Instead of Key Frames, I Like Specifying Key Times Better

And, so, we created a C++ class to do it all for you

```cpp
4 time-value pairs have been given:
   (  0.00,   0.000)   (  0.50,   2.718)   (  1.00,   3.142)   (  2.00,   0.333)
Time runs from    0.000 to    2.000
```

Setting Up the Time-Value Pairs

```cpp
#define MAXSECONDS              30.f
.
.
Keytimes ThetaX, ThetaY, ThetaZ; // global
Keytimes ScaleXYZ; // global
.
// in main( ) or in InitGraphics( ):
ScaleXYZ.AddTimeValue( 0.f,  1.f);
ScaleXYZ.AddTimeValue( 7.5f, 0.25f);
ScaleXYZ.AddTimeValue(15.f,  1.f);
ScaleXYZ.AddTimeValue(22.5f, 2.f);
ScaleXYZ.AddTimeValue(30.f,  1.f);
ThetaX.AddTimeValue(0.0f, 0.0f);
ThetaX.AddTimeValue(5.f, glm::radians(720.f));
ThetaX.AddTimeValue(10.f, glm::radians(0.f));
ThetaX.AddTimeValue(20.f, glm::radians(-720.f));
ThetaX.AddTimeValue(30.f, glm::radians(0.f));
ThetaY.AddTimeValue(0.0f, 0.0f);
ThetaY.AddTimeValue(30.f, glm::radians(10.f*360.f+180.f));
.
.
Number of seconds in the animation cycle
```
Using the System Clock for Timing

// in the GLFW polling loop:
Time = glfwGetTime(); // elapsed time, in double-precision seconds
// do this for cyclic animation:
Time = fmod(Time, MAXSECONDS); // fmod gives the remainder of dividing Time by MAXSECONDS
// so Time stays between 0. and MAXSECONDS

// change the object matrix:
float time = (float)Time;
Object.uModel = glm::mat4(1.); // identity
Object.uModel = glm::rotate(Object.uModel, ThetaX.GetValue(time), glm::vec3(1.f, 0.f, 0.f));
Object.uModel = glm::rotate(Object.uModel, ThetaY.GetValue(time), glm::vec3(0.f, 1.f, 0.f));
Object.uModel = glm::scale(Object.uModel, glm::vec3(ScaleXYZ.GetValue(time)));
Object.uNormal = glm::mat4(glm::inverseTranspose(glm::mat3(Scene.uSceneOrient*Object.uModel)));
Object.uColor = glm::vec4( 1.f, 0.2f, 0.2f, 1.f);
Fill05DataBuffer( MyObjectUniformBuffer, IN (void *) &Object );