Simple Keytime Animation

Oregon State University

Mike Bailey

mjb@cs.oregonstate.edu

Oregon State University
Computer Graphics

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.

Instead of Key *Frames*, I Like Specifying Key *Times* Better

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

```
class Keytimes:

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

Oregon State University
Computer Graphics

Instead of Key *Frames*, I Like Specifying Key *Times* Better

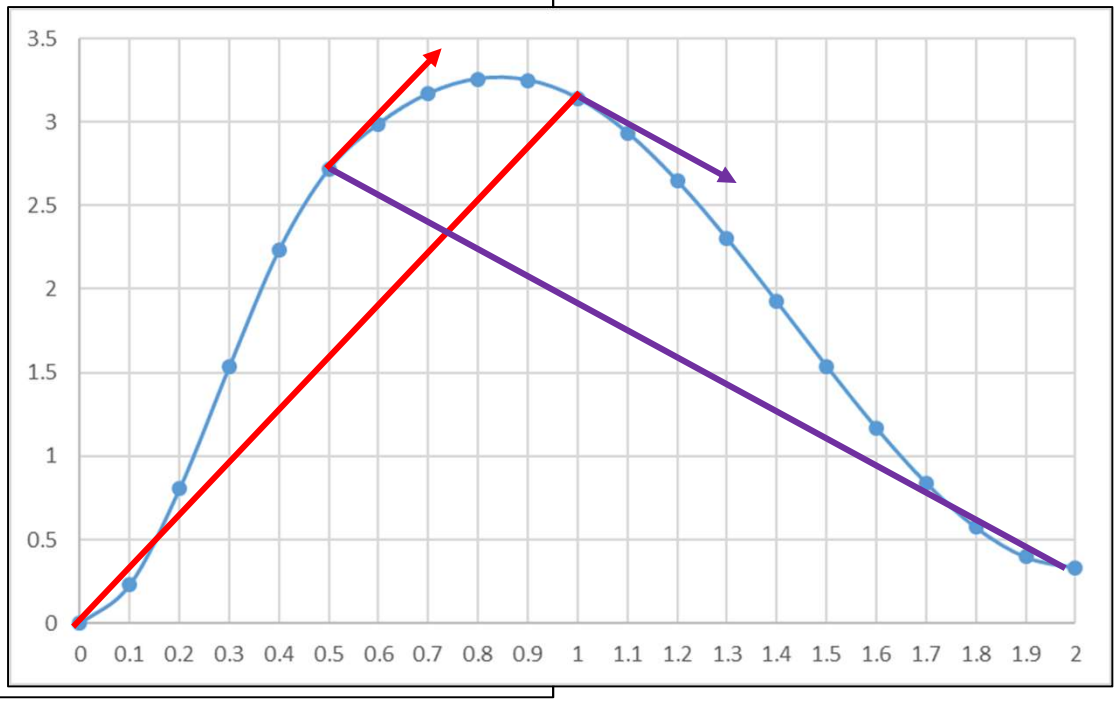
The **Time** and the **Value** form a Time-Value Pair

```
Keytimes   Xpos;

int
main( int argc, char *argv[ ] )
{
	Xpos.AddTimeValue(  0.0f,  0.000f );
	Xpos.AddTimeValue(  0.5f,  2.718f );
	Xpos.AddTimeValue(  1.0f,  3.142f );
	Xpos.AddTimeValue(  2.0f,  0.333f );
	fprintf( stderr, "%d time-value pairs have been given:\n", Xpos.GetNumKeytimes( ) );
	Xpos.PrintTimeValues( );

	fprintf( stderr, "Time runs from %8.3f to %8.3f\n", Xpos.GetFirstTime( ), Xpos.GetLastTime( ) );

	for( float t = 0.; t <= 2.01; t += 0.1 )
	{
		float v = Xpos.GetValue( t );
		fprintf( stderr, "%8.3f\t%8.3f\n", t, v );
	}
}
```

Instead of Key *Frames*, I Like Specifying Key *Times* Better

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

0.000	0.000
0.100	0.232
0.200	0.806
0.300	1.535
0.400	2.234
0.500	2.718
0.600	2.989
0.700	3.170
0.800	3.258
0.900	3.250
1.000	3.142
1.100	2.935
1.200	2.646
1.300	2.302
1.400	1.924
1.500	1.539
1.600	1.169
1.700	0.840
1.800	0.574
1.900	0.397
2.000	0.333



Setting Up the Time-Value Pairs

```
#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 );
        Object.uShininess = 32.f;
        Fill05DataBuffer( MyObjectUniformBuffer, IN (void *) &Object );
. . .
```