Animation Effects using a Timer

Mike Bailey
mjb@cs.oregonstate.edu

Computer Graphics

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

Using Timers with Shaders

glm has a built-in Timer variable. You just need to declare it:

```cpp
uniform float Timer;
```

Then, just use it in your code.

It goes from 0. to 1. in 10 seconds, and then instantly back to 0.

Or, you can program a Timer yourself in your .cpp program:

```cpp
float Timer; // global variable
class Timer { const int MS_PER_CYCLE = 10*1000; // 10,000 ms = 10 seconds . . . void Animate() { int ms = glutGet( GLUT_ELAPSED_TIME ); ms %= MS_PER_CYCLE; Timer = (float)ms / (float)MS_PER_CYCLE; // 0. to 1. in 10 seconds glutPostRedisplay(); } }
void InitGraphics() { . . . glutIdleFunc( Animate ); }
```

Fun With Zero-to-One:
There are many ways to map 0.→1. to a different function

**Single ramp 0.→1.**
- float t = Timer;
- float t = Timer*Timer;
- float t = Timer*Timer*Timer;
- float t = 10.*Timer^3 – 15.*Timer^4 + 6.*Timer^5

**Double ramp 0.→1. →0.**
- float t; if(Timer <= .5 ) t = 2.*Timer; else t = 2. *( 1. – Timer );

**Smooth oscillation -1. → 1. → -1.**
- float t = sin( 2.*π*Timer );

**Smooth oscillation 0. → 1. → 0.**
- float t = .5 + .5*sin(2.*π*Timer );

**Faster oscillation**
- float t = sin( 2.*π*S*Timer );

**Bigger oscillation**
- float t = Mag * sin( 2.*π*S*Timer );

**float t = sin( 2.*π*S*Timer );**

```
float t;
if( Timer <= .5 )
  t = 2.*Timer;
else
  t = 2. *( 1. – Timer );
float t = sin( 2.*π*Timer );
```
Fun-With-Zero-To-One

Sidebar: Why Do These Two Curves Match So Closely?

The Taylor Series expansion of \( y = \sin \left( \frac{x}{2} \right) \) around \( x = 0.5 \) is:

\[
y = \frac{1}{2} \cdot x + \frac{\pi^2}{2} \cdot x^3 + \frac{\pi^4}{36} \cdot x^5 + \frac{\pi^6}{1280} \cdot x^7 + \cdots
\]

which is somewhat close to: \( y = 3x^2 - 2x^3 \)