Animation Effects using a Timer

Mike Bailey
mjb@cs.oregonstate.edu

Computer Graphics

Using Timers with Shaders

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

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.

Dr. you can program a Timer yourself:

float 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
  glutSetWindow( MainWindow );
  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^2 – 15.*Timer^4 + 6.*Timer^6

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 );
float t = .5 + .5*sin(2.*π*Timer );
float t = sin( 2.*π*S*Timer );
float t = Mag * sin( 2.*π*S*Timer );

Bigger oscillation

float t = sin( 2.*π*S*Timer );

Fun-With-Zero-To-One

Sidebar: Why Do These Two Curves Match So Closely?

The Taylor Series expansion of 2sin 2πx around x=0.5 is:

y = 2πx - 2π²x² + 2π³x³ - 2π⁴x⁴ + 2π⁵x⁵ +...

which is somewhat close to: y = 3x² - 2x³

which is somewhat close to: y = 3x² - 2x³