Interactive Computer Graphics

CS 418 – Fall 2012

MP1: Dancing I

TA: Gong Chen

Email: gchen10 at illinois dot edu

Slides Taken from:

"An Interactive Introduction to OpenGL Programming"

Dave Shreiner, Ed Angel, Vicki Shreiner

Agenda

- MP1 Submission Announcement
- A little more OpenGL
  - Applications Structure
  - Callback functions
  - Double Buffering (required)
  - Animation Techniques
- MP Q&A

GLUT Basics

- Application Structure
  - Configure and open window
  - Initialize OpenGL state
  - Register input callback functions
    - render/display
    - resize
    - input: keyboard, mouse, etc.
  - Enter event processing loop

Sample Program

```
void main( int argc, char** argv )
{
   int mode = GLUT_RGB|GLUT_DOUBLE;
   glutInitDisplayMode( mode );
   glutCreateWindow( argv[0] );
   init();
   glutDisplayFunc( display );
   glutReshapeFunc( resize );
   glutKeyboardFunc( key );
   glutIdleFunc( idle );
   glutMainLoop();
}
```

OpenGL Initialization

- Set up whatever state you're going to use.
  - In our MP1 Case only this:

```
void init( void )
{
  glClearColor( 0.0, 0.0, 0.0, 1.0 );
}
```

GLUT Callback Functions

- Routine to call when something happens
  - window resize or redraw
  - user input
  - animation
- "Register" callbacks with GLUT

```
glutDisplayFunc( display );
glutIdleFunc( idle );
glutKeyboardFunc( keyboard );
```

Rendering Callback

- Do all of your drawing here

```
glutDisplayFunc( display );
```

```
void display( void )
{
  glClear( GL_COLOR_BUFFER_BIT );
  glBegin( GL_TRIANGLE_STRIP );
    glVertex3fv( v[0] );
    glVertex3fv( v[1] );
    glVertex3fv( v[2] );
    glVertex3fv( v[3] );
  glEnd();
  glutSwapBuffers();
}
```

Idle Callbacks

- Use for animation and continuous update

```
glutIdleFunc( idle );
```

```
void idle( void )
{
  glutPostRedisplay();
}
```

User Input Callbacks

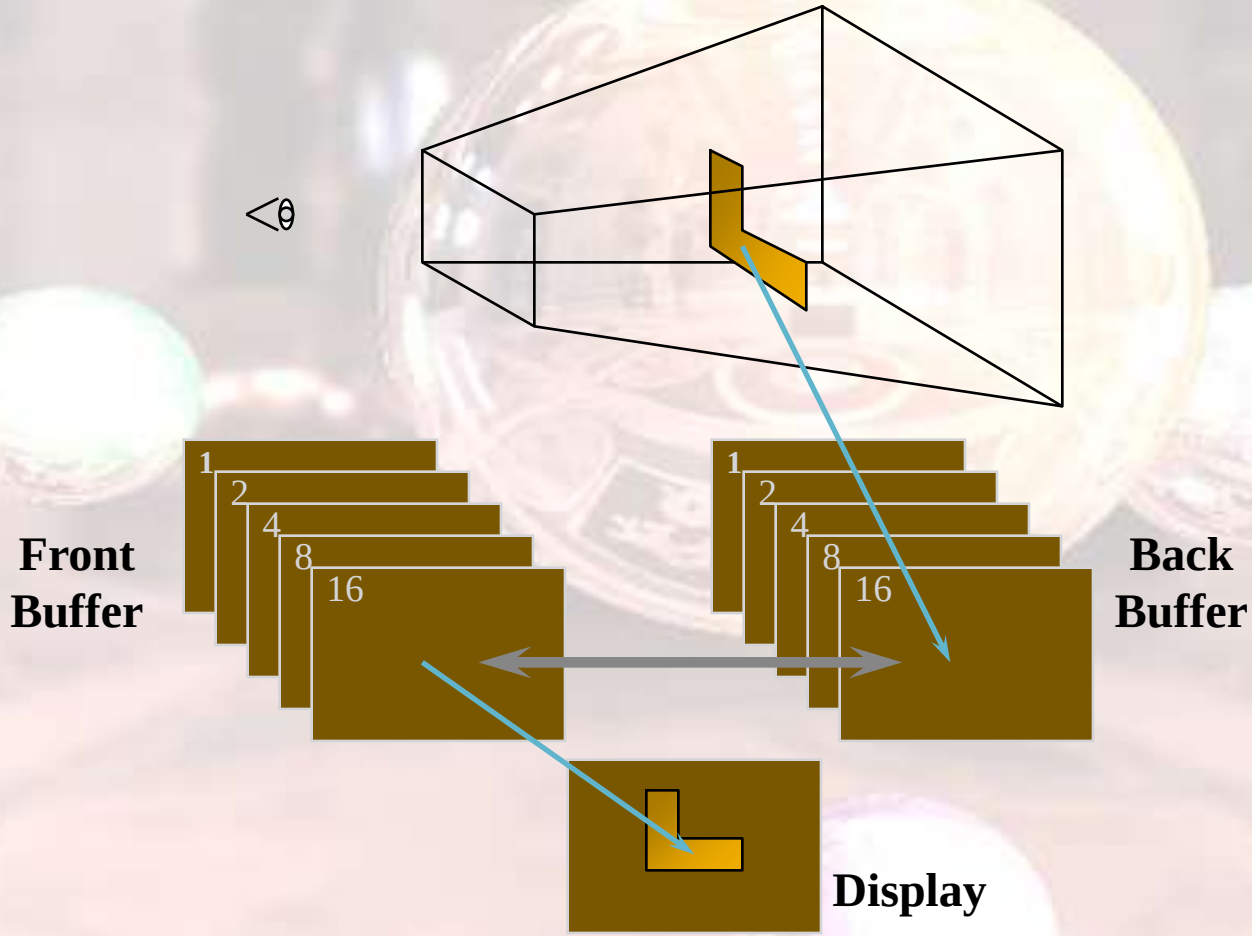
- Process user input

```
glutKeyboardFunc( keyboard );
```

```
void keyboard( char key, int x, int y )
{
  switch( key ) {
    case 'q' : case 'Q' :
      exit( EXIT_SUCCESS );
      break;

    case 'r' : case 'R' :
      rotate = GL_TRUE;
      break;
  }
}
```

Double Buffering



Animation Using Double Buffering

- Request a double buffered color buffer

```
glutInitDisplayMode(GLUT_RGB |GLUT_DOUBLE);
```

- Clear color buffer

```
glClear( GL_COLOR_BUFFER_BIT );
```

- Render scene
- Request swap of front and back buffers

```
glutSwapBuffers(); Interesting Example
```

- Repeat steps 2 - 4 for animation

Sine Waves-Simple Animation

- 3 ways to generate sine waves in software:
  - **y = sin(x)**
    - Easiest – But the slowest way
  - **2. Look Up Table**
  - **3. Look Up Table Plus First Order (Linear) Interpolation**

Lookup Table

- generate an array of the sine values and store them in memory.
  - use the symmetry properties of the sine wave to minimize the memory storage

Example:

Obtain y=sin(x) in one degree steps:

For x Є (0,90) , we can create the array:

```
float sine[91] , pi=3.141592653;
for (int i=0;i<=90;i++)
    sine[i] = sin(pi/180 * i);
```

Then, if we wanted the sine of 45 degrees, we simply write

```
y = sine[45];
```

Lookup Table (cont.)

Example (cont.):

Obtain the other 3/4's of the circle:

we can use the symmetry of sine wave.
Each quadrant is obtained as follows:

```
y = sine[ 180 - x];          /* 90 <= x <= 180 */

y = -sine[x - 180];          /* 180 <= x <= 270 */

y = -sine[360 - x];          /* 270 <= x <= 360 */
```