Computer Science Department

Sam Houston State University

COSC 4332 Computer Graphics

OpenGL Labs
Lab 1

Dr. Khaled Rabieh

Outline

1. **Simple clock (Basic primitives +Transformations)**

OpenGL

Setting up the environment

- Install Visual studio 2015
- Check C++
  - C++ is not installed by default in visual studio 2015
- Go through the following link to add OpenGL Libraries the easy way using nuGet packages

http://in2gpu.com/2014/11/29/setting-opengl-visual-studio-using-nuget/

What Is OpenGL?

Graphics rendering API

- high-quality color images composed of geometric and image primitives
- window system independent
- operating system independent

Related APIs

- Additional libraries are used to modify a native window into an OpenGL capable window
  - AGL → Apple Mac
  - GLX → Unix platforms
  - WGL → Microsoft Windows
- GLU (OpenGL Utility Library)
  - Simplify common tasks such as quadric surfaces (i.e. spheres, cones, cylinders, etc.),
- GLUT (OpenGL Utility Toolkit)
  - making simple OpenGL applications
  - portable windowing API

Application Structure

1. Configure and open window
   - Choose the type of window that you need for your application and initialize it
2. Initialize OpenGL state
   - Background color, light positions and texture maps.
3. Register input callback functions
   - Render, resize, input: keyboard, mouse, etc.
4. Enter event processing loop
   - This is where your application receives events, and schedules when callback functions are called

Sample Program

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

initialize OpenGL state
->Set the background

Register callback routines

compose the window configuration step.

OpenGL Initialization

Set up whatever state you're going to use

```
void init( void )
{
//  Set the frame buffer clear color
    glClearColor( 0.0, 0.0, 0.0, 1.0 );
    glClearDepth( 1.0 );

    glEnable( GL_LIGHT0 );
    glEnable( GL_LIGHTING );
    glEnable( GL_DEPTH_TEST );
}
```

OpenGL to Draw Polygon

Type

Set of vertices

```
glBegin (GL.GL_POLYGON);
    glVertex2f (-0.5f, -0.5f);
    glVertex2f (-0.5f,  0.5f);
    glVertex2f ( 0.5f,  0.5f);
    glVertex2f ( 0.5f, -0.5f);
glEnd ();
glFlush ();
}
```

Set of vertices Between glBegin(type) and glEnd

GLUT Callback Functions

- A callback is a routine to call when something happens
  - window resize or redraw
  - user input
  - Animation

glutDisplayFunc(*display*);

called when pixels in the window need to be refreshed

glutIdleFunc(*idle*);

Called when nothing else is going on. Very useful for animations.

glutKeyboardFunc(*keyboard*);

called when a key is struck on the keyboard

Rendering Callback

```
glutDisplayFunc( display );
```

Add your drawing here

```
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();
}
```

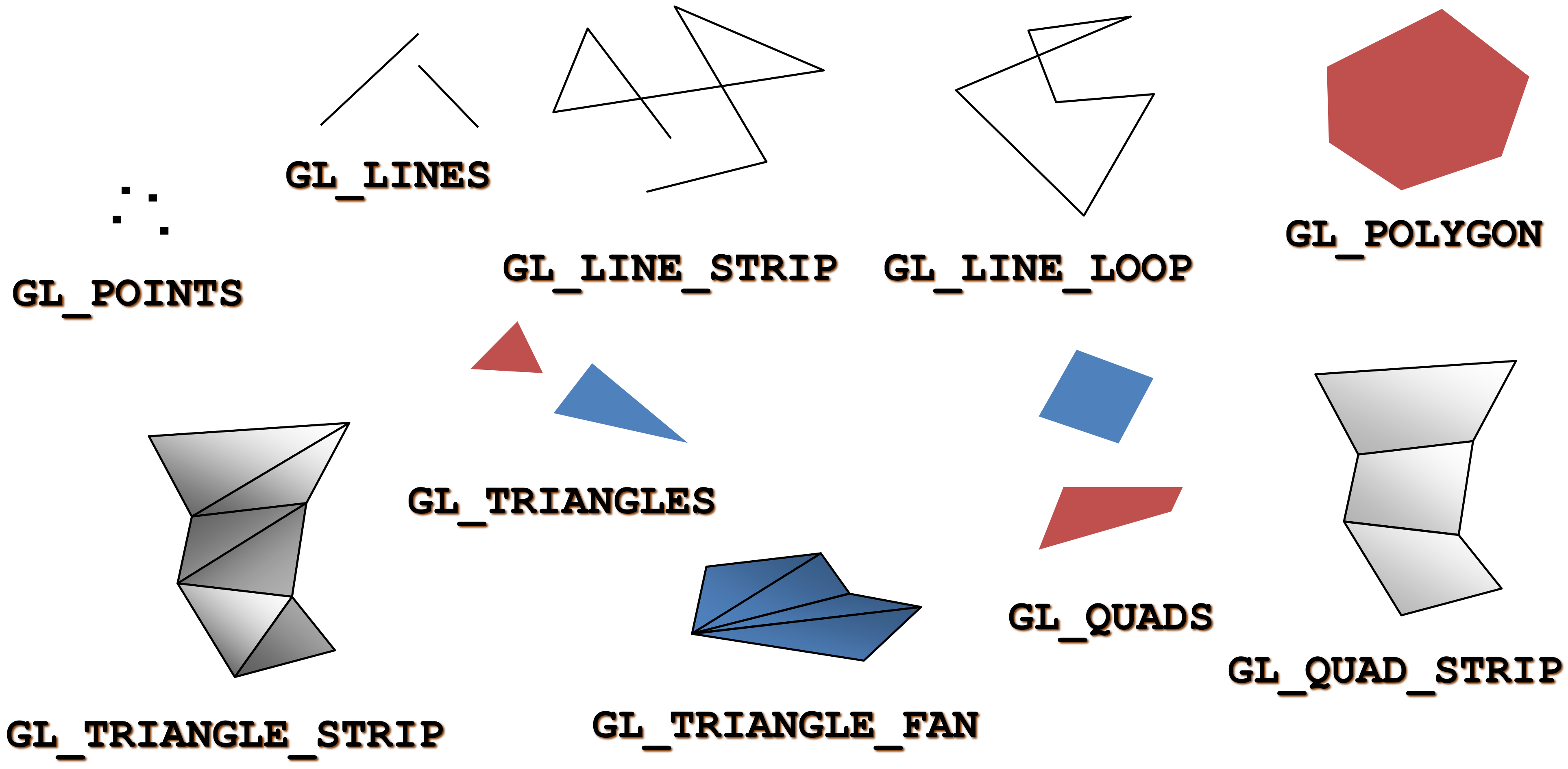
User Input Callbacks

- 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;
  }
}
```

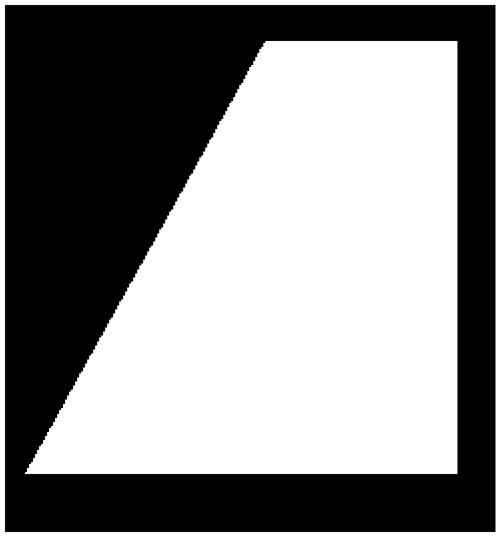
OpenGL Geometric Primitives

All geometric primitives are specified by vertices



Simple Example

```
void drawRhombus( GLfloat color[] )
{
      glBegin(GL_QUADS);
      glVertex2f(0.0f, 0.0f);
      glVertex2f(0.9f, 0.0f);
      glVertex2f(0.9f, 0.9f);
      glVertex2f(0.5f, 0.9f);
      glEnd();
   }
```



OpenGL Command Formats

```
glVertex3fv( v )
```

Number of components

2	(x,y)
3	(x,y,z)
4	(x,y,z,w)

Data Type

b	byte
ub	unsigned byte
s	short
us	unsigned short
i	int
ui	unsigned int
f	float
d	double

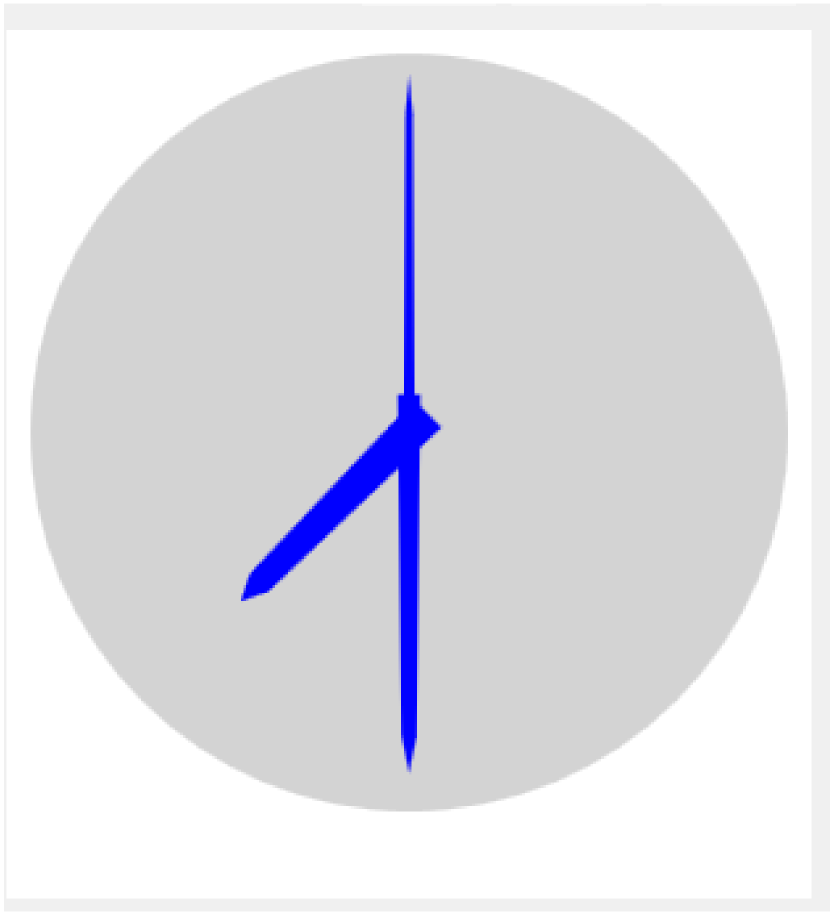
Vector

omit "v" for scalar form

```
glVertex2f( x, y )
```

The Clock Project Requirements

- Project break down
  - A sphere
  - 3 polygons (hands)
  - Animations (Rotate/translate)
  - No user interaction



The Analog Clock using OpenGL

init function

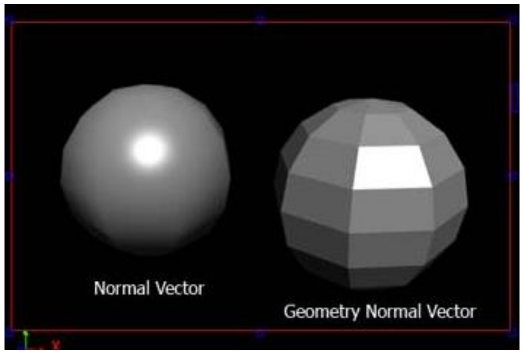
- Put your one time setup here

```
void init()
{
glClearColor(0.0, 0.0, 0.0, 0.0);
glColor3f(0.0f, 0.0f, 0.0f);
glPointSize(5.0f);
glEnable(GL_POINT_SMOOTH);
glEnable(GL_POLYGON_SMOOTH);
glEnable(GL_LINE_SMOOTH);
}
```

Back ground Color

Default paint color

Default point size

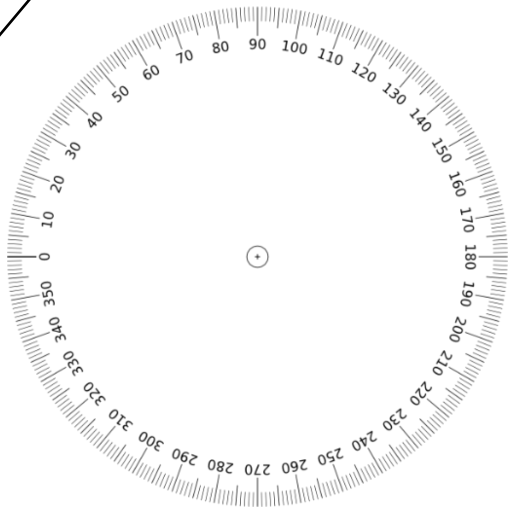


Drawing a circle

We draw a circle as a set of points

```
int ticks=0;
for (int i = 0; i < 360; i++)
{
   glPushMatrix();
   glRotated(ticks += 1, 0, 0, 1);
   glTranslated(1, 0, 0);
   glColor3f(1.0, 0.0, 0.0);
   glBegin(GL_POINTS);
   glVertex2d(0, 0);
   glEnd();
   glPopMatrix();
}
```

Rotate 1 degree in the z axis



Move 1 unit

Define the color of the point (Red)

Draw the point

glPushMatrix() & glPopMatrix()

- Imoprtant to use them when we have any transformations in our scene
  - Scale, rotate, translate
- All transformation functions (glScaled, etc.) function on the top matrix, and the top matrix is what all rendering commands use to transform their data.

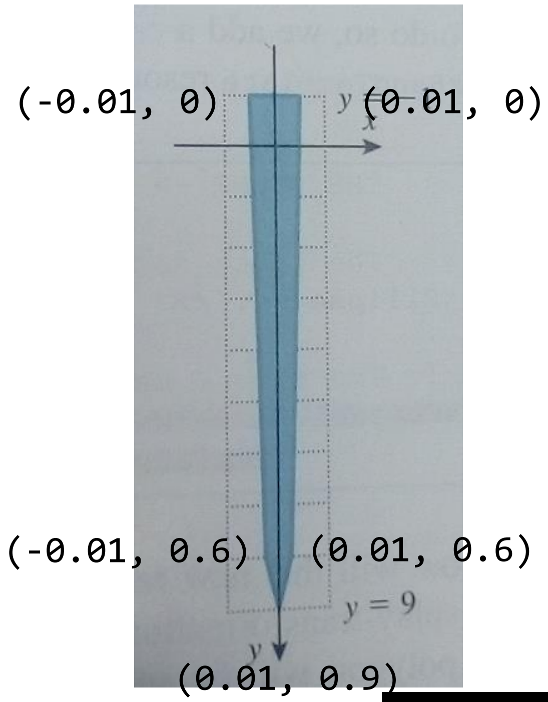
```
glPushMatrix();  //Tells OpenGL to store the current
state that we are in.
glPopMatrix();   //Then when we want to go back to
our previous state, we call glPopMatrix().
```

https://www.youtube.com/watch?v=OaBSHuP4xcE

Drawing the seconds hand

- The same idea as before
- One difference that you should consider the identity co-ordinates

```
glPushMatrix();
glRotated(-seconds, 0, 0, 1);
glBegin(GL_POLYGON);
glVertex2d(-0.01, 0);
glVertex2d(0.01, 0);
glVertex2d(0.01, 0.6);
glVertex2d(0, 0.9);
glVertex2d(-0.01, 0.6);
glEnd();
glPopMatrix();
```



Transformation using Timers

In the main

```
int main(int argc, char **argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
   glutInitWindowPosition(200, 200);
   glutInitWindowSize(640, 480);
   glutCreateWindow(“Analog Clock");
   glutDisplayFunc(display);
   init();
   Timer(0);
   glutMainLoop();
   return 0;
}
```

Timer Configuration

Timer is called every 1000 milliseconds = 1 second

```
void Timer(int value)
{
   glutTimerFunc(1000, Timer, 0);
   glutPostRedisplay();
}
```

Timer Configuration

```
int secondsAngle = 0;

void Timer(int value)
{
   glutTimerFunc(1000, Timer, 0);
   glutPostRedisplay();
   secondsAngle+=6;
}
```

Call Timer function every 1000 ms = 1 seconds

The scene has changed and makes sure that that GLUT redraws it.

Rotating the seconds hand

```
// Seconds hand
glPushMatrix
glRotated(-seconds, 0, 0, 1);
glBegin(GL_POLYGON);
glVertex2d(-0.01, 0);
glVertex2d(0.01, 0);
glVertex2d(0.01, 0.6);
glVertex2d(0, 0.9);
glVertex2d(-0.01, 0.6);
glEnd();
glPopMatrix
```

Rotate around Z axis

Thank You

Questions

Khaled Rabieh