SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 912
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology

Course Name –Computer Graphics

III Year / V Semester

Unit 1– INTRODUCTION TO COMPUTER GRAPHICS

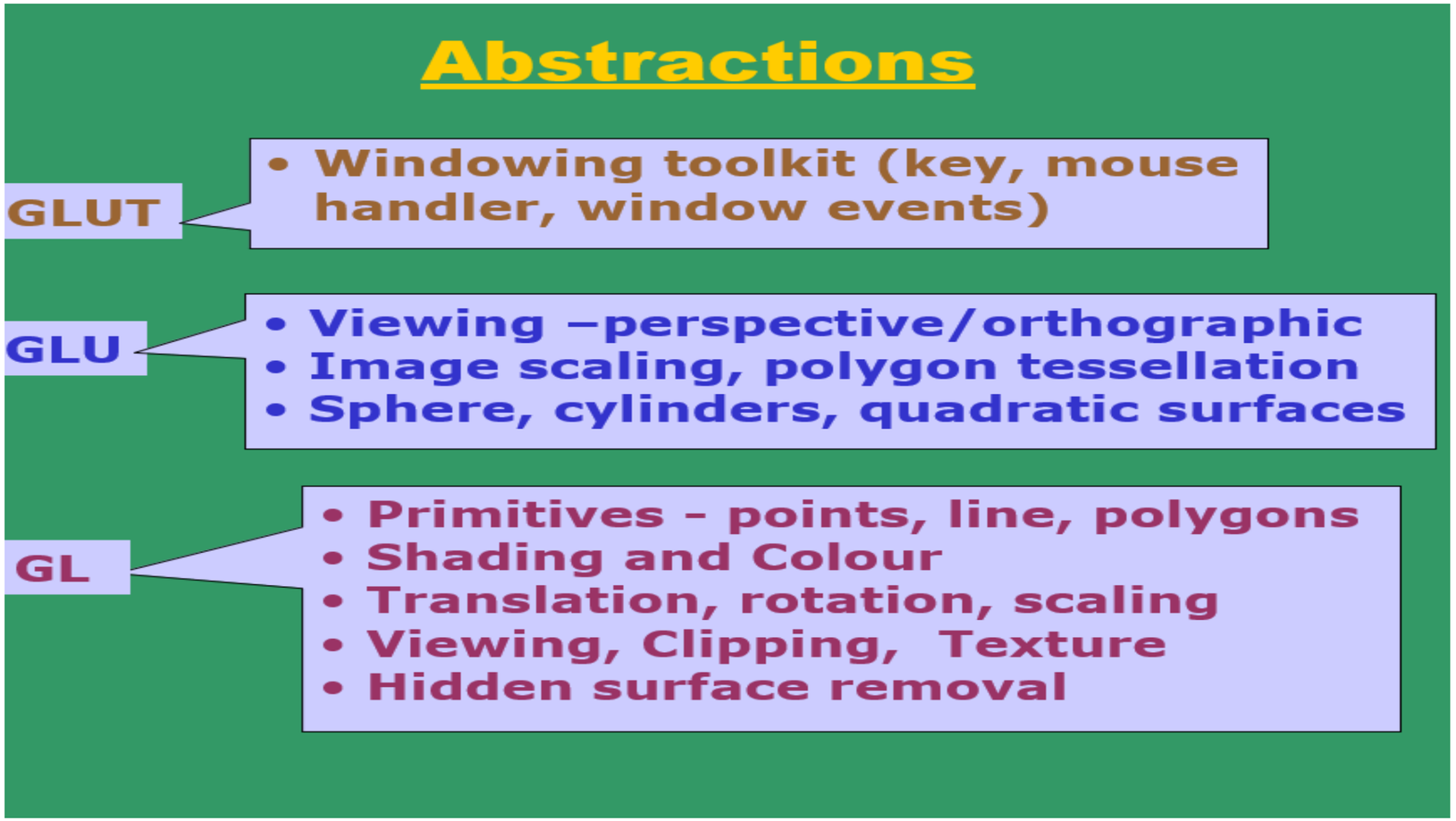
Topic :OPENGL Basics

Basic Primitives

- Geometric data (vertices, lines, and polygons) follows the path through the row of boxes that includes evaluators and per-vertex operations, while pixel data (pixels, images, and bitmaps) is treated differently for part of the process.

- Both types of data undergo the rasterization and per-fragment operations before the final pixel data is written into the frame buffer.

Basic Primitives



Basic Primitives

OpenGL supports several basic primitive types, including points, lines, quadrilaterals, and general polygons. All of these primitives are specified using a sequence of vertices.

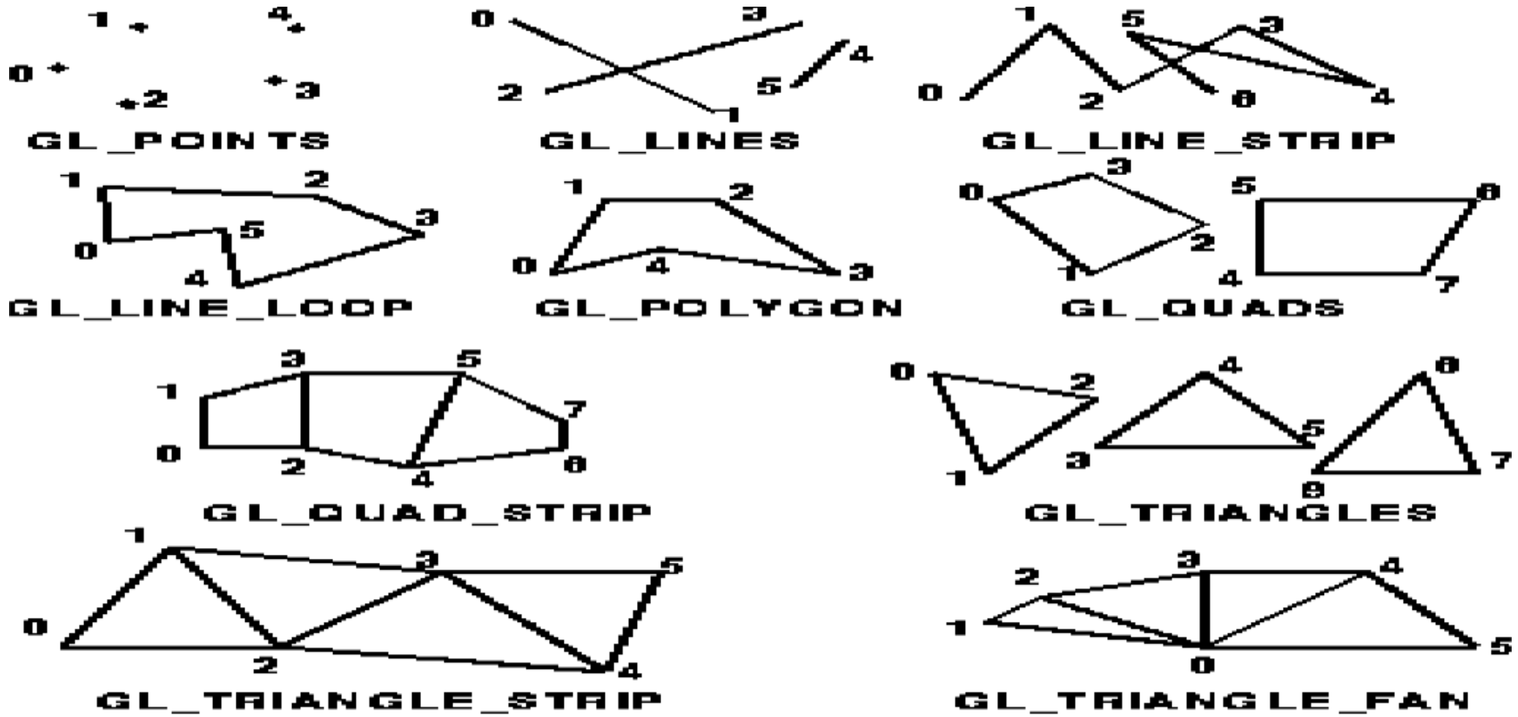
```
glVertex2i(Glint xi, Glint yi);
glVertex3f(Glfloat x, Glfloat y, Glfloat z);
 Glfloat vertex[3];

glBegin(GL_LINES);
glVertex2f(x1, y1);  glVertex2f(x2, y2);
glEND();
```

Define a pair of points as: glBegin(GL_POINTS);

```
glVertex2f(x1, y1);  glVertex2f(x2, y2);
glEND();
```

Basic Shapes



Draw several isolated points

```
GLfloat pt[2] = {3.0, 4.0};
 glBegin(GL_POINTS);
glVertex2f(1.0, 2.0);
glVertex2f(2.0, 3.0);
 glVertex2fv(pt);
glVertex2i(4,5);
glEnd();
```

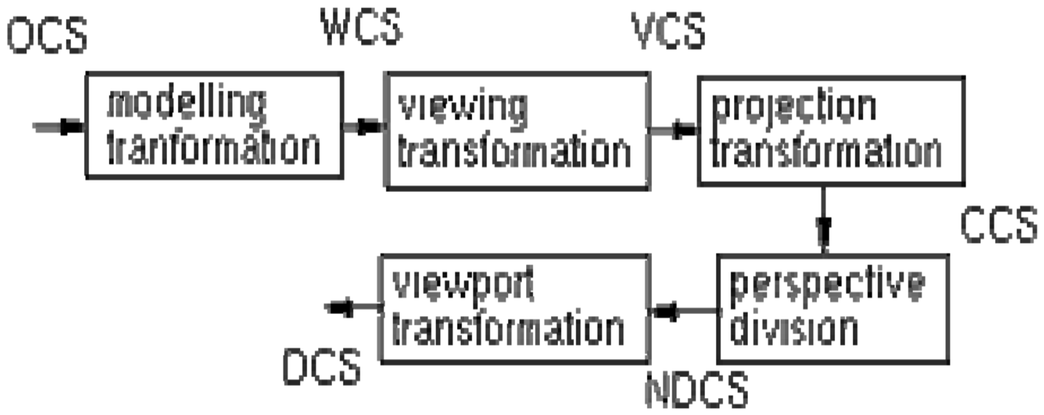
```
GLfloat p1[3] = {0,0,1};
GLfloat p2[3] = {1,0,1};
GLfloat p3[3] = {1,1,1};
GLfloat p4[3] = {0,1,1};

glBegin(GL_POLYGON);  glVertex3fv(p1);  glVertex3fv(p2);
glVertex3fv(p3);  glVertex3fv(p4);
glEnd();
```

Basic Primitives

Coordinate Systems in the Graphics Pipeline

- OCS - object coordinate system
- WCS - world coordinate system
- VCS - viewing coordinate system
- CCS - clipping coordinate system
- NDCS - normalized device coordinate system
- DCS - device coordinate system



OpenGL functions for setting up transformations

modelling transformation (modelview matrix)
glTranslatef()
glRotatef()
glScalef()

viewing transformation (modelview matrix)
gluLookAt()

projection transformation (projection matrix)
glFrustum() gluPerspective() glOrtho() gluOrtho2D()

viewing transformation
glViewport()

Structure of a GLUT Program

```
int main(int argc, char **argv) {  glutInit(&argc, argv);

glutInitDisplayMode(GLUT_DOUBLE |  GLUT_RGB | GLUT_DEPTH);

glutCreateWindow("Interactive rotating  cube"); // with size & position

glutDisplayFunc(display);
// display callback, routines for drawing

glutKeyboardFunc(myKeyHandler);
// keyboard callback

glutMouseFunc(myMouseClickHandler);
// mouse callback
glutMotionFunc(myMouseMotionHandler);
// mouse move callback
init();  glutMainLoop();

}

void display() {...}

void myKeyHandler( unsigned char key, int x,  int y) {...}

void myMouseClickHandler( int button, int  state, int x, int y ) {...}

void myMouseMotionHandler( int x, int y) {...}
```

Drawing a square in OpenGL

```
#include <stdio.h>
#include <GL/glut.h>
void display(void)
{
glClear( GL_COLOR_BUFFER_BIT);
glColor3f(0.0, 1.0, 0.0);  glBegin(GL_POLYGON);  glVertex3f(2.0, 4.0, 0.0);
glVertex3f(8.0, 4.0, 0.0);
glVertex3f(8.0, 6.0, 0.0);
glVertex3f(2.0, 6.0, 0.0);  glEnd();
glFlush();
}
int main(int argc, char **argv)
{
printf("hello world\n");
glutInit(&argc, argv);  glutInitDisplayMode
( GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
```

Drawing a square in OpenGL

```
glutInitWindowPosition(100,100);
glutInitWindowSize(300,300);
glutCreateWindow ("square");
glClearColor(0.0, 0.0, 0.0, 0.0);
// black background
glMatrixMode(GL_PROJECTION);
// setup viewing projection
glLoadIdentity();
// start with identity matrix
glOrtho(0.0, 10.0, 0.0, 10.0, -1.0, 1.0);
// setup a 10x10x2 viewing world
glutDisplayFunc(display);
glutMainLoop();
return 0;
}
```

THANK YOU