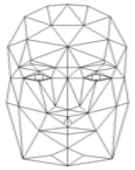


# Lecture 2

## 2D transformations

## Introduction to OpenGL





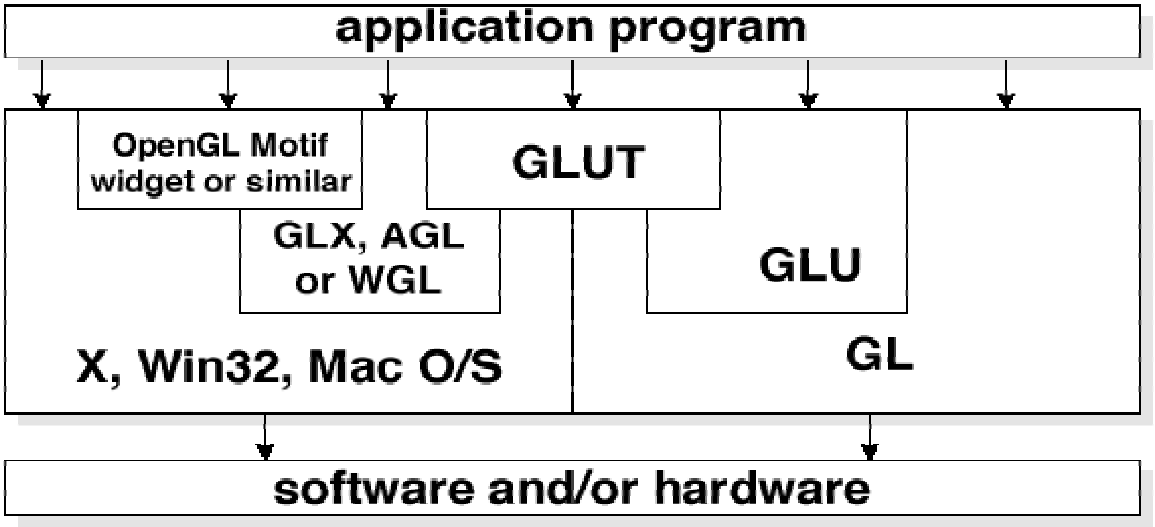
# OpenGL

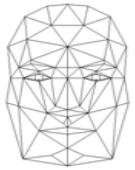
where it fits

what it contains

how you work with it

# OpenGL and Related APIs







# OpenGL parts:

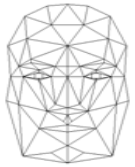
**GL = Graphics Library (core lib)**

**GLU = GL Utilities (always present)**

**GLX, AGL, WGL = System dependent libraries**

**GLUT = GL Utility Toolkit (optional)**

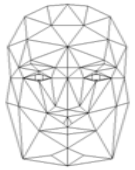
**Also note: SDL (Simple Directmedia Layer)**





# Sample main program (in C)

```
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();
}
```





# Name conventions

**All calls prefixed by gl, glu, glut**

**All constants prefixed by GL, GLU, GLUT**

**All calls suffixed by number, f,d,v**

# Example

## glVertex2fv( v )

**umber of omponents**

- (x,y)
- (x,y,z)
- (x,y,z,w)

**ata type**

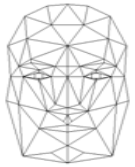
- byte
b - unsigned byte
- short integer
s - unsigned short
- int (integer)
i - unsigned int
- float
- double

**ector**

f present: arguments re array

f omitted: arguments as eparate scalars:

lVertex2f( x, y )



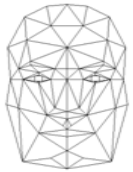


# OpenGL type names

**Standard types redefined prefixed GL**
**(for compatibility reasons)**

**From gl.h:**

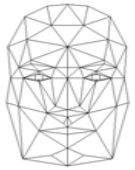
```
typedef unsigned long GLenum;
typedef unsigned char GLboolean;
typedef unsigned long GLbitfield;
typedef signed char GLbyte;
typedef short GLshort;
typedef long GLint;
typedef long GLsizei;
typedef unsigned char GLubyte;
typedef unsigned short GLushort;
typedef unsigned long GLuint;
typedef float GLfloat;
typedef float GLclampf;
typedef double GLdouble;
typedef double GLclampd;
typedef void GLvoid;
```





# OpenGL Initialization
## Set up whatever state you're going to use

```
void init( void )
{
    glClearColor( 0.0, 0.0, 0.0, 1.0 );
    glClearDepth( 1.0 );
    glEnable( GL_LIGHT0 );
    glEnable( GL_LIGHTING );
    glEnable( GL_DEPTH_TEST );
}
```





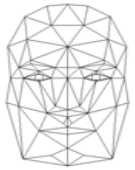
# GLUT Callback Functions

## Routines for GLUT to call when something happens

- window resize or redraw
- user input
- animation

“Register” callbacks with GLU

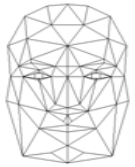
```
glutDisplayFunc( display );
glutIdleFunc( idle );
glutKeyboardFunc( keyboard );
```





# Rendering Callback
# Do all of our drawing here

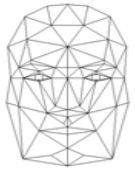
```
glutDisplayFunc( display );
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT );
    glBegin( GL_TRIANGLE_STRIP );
    glVertex3fv( v[0] );
    glVertex3fv( v[1] );
    glVertex3fv( v[2] );
    glVertex3fv( v[3] );
    glEnd();
    glutSwapBuffers();
}
```





# Timer Callbacks
# Use for animation and continuous update

```
glutTimerFunc( time, idle );

void timerFunc( ignoredValue )
{
    t +=dt;
    glutTimerFunc(20, timerFunc, 0);
    glutPostRedisplay();
}
```

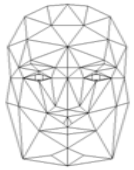




# Simple Example

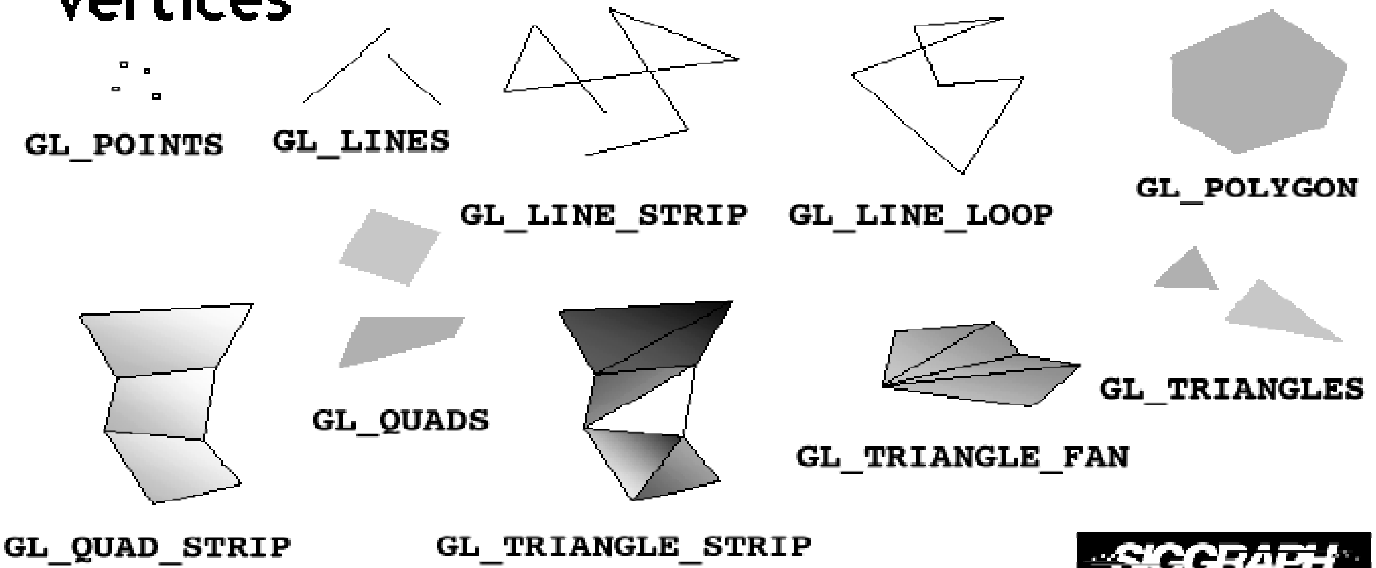
## A shape can be hard-coded as a function

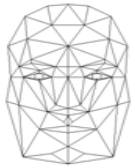
```
void drawRhombus( GLfloat color[] )
{
    glBegin( GL_QUADS );
    glColor3fv( color );
    glVertex2f( 0.0, 0.0 );
    glVertex2f( 1.0, 0.0 );
    glVertex2f( 1.5, 1.118 );
    glVertex2f( 0.5, 1.118 );
    glEnd();
}
```



# OpenGL Geometric Primitives

## All geometric primitives are specified by vertices







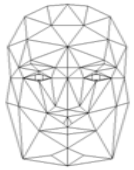
# Specifying geometric primitives

**primitive is described within a begin/end pair:**

```
glBegin( primtype );
glEnd();
```

**etween them, put glVertex for specifying vertices,**
**nd other call for changing the state (color, texture...)**

**“Immediate mode”**





# OpenGL color

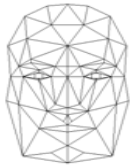
**GBA or indexed color**

**We primarily use RGBA**

**R = red**
**= green**
**= blue**
**= alpha**

**alues are specified from 0.0 to 1.0**

**Alpha = transparency!**





# OpenGL state

he state is a set of internal values in OpenGL.

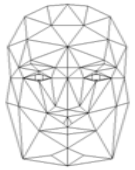
Vertex attributes (color, texture...)

Activation of features with glEnable

- Transformation matrix

ny change you make lasts until you change it again!

ou don’t automatically get the default back, you get hat you used the last time.





# Common OpenGL state manipulating calls

**Vertex attributes:**

**lColor*(), glNormal*(), glTexCoord*()**

**Setting state variables:**

**glPointSize(size)**
**glLineStipple( repeat, pattern)**
**glShadeModel( GL_SMOOTH )**

**Enabling and disabling features:**

**glEnable( GL_LIGHTING )**
**glDisable( GL_TEXTURE_2D )**



# Affine transformations in OpenGL

**Rotation, translation, scaling and more**

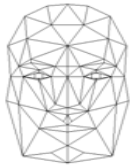
**Transformations are concatenated (multiplied) on the current atrix**

**POST-multiplication: current := current * new**

**gl LoadIdentity() sets the matrix to the identity matrix!**

**glTranslate*()**
**lRotate*()**
**glScale*()**

**glLoadMatrix*( m ) sets the current matrix to m.**





# Manipulating the current matrix

**Very important: Saving and restoring**

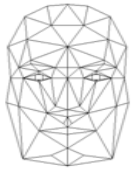
**glPushMatrix() saves a copy of the current matrix on a stack**

**glPopMatrix() restores the matrix from the top of the stack.**

**For completeness: You can use any matrix**

**glLoadMatrix*( m ) sets the current matrix to m.**

**glMultMatrix*( m ) applies the matrix m**

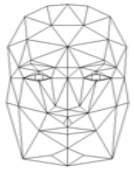




# Transformation example

```
oid display( void )

    glClear( GL_COLOR_BUFFER_BIT );
    glColor3fv((const GLfloat *)&color);

    glLoadIdentity();

    glRotatef(45, 0,0,1);
    glTranslatef(0, -1, 0);

    glBegin( GL_POLYGON);
    glVertex2f(0, 0);
    glVertex2f(1, 0);
    glVertex2f(0, 0.5);
    glEnd();

    glFlush();
```

eware that the order of transformations matter!



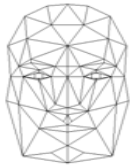


# Specifying geometric primitives II

lternative to “Immediate mode”: Specify geometry by passing arrays of vertices and onnectivity.

glDrawElements();

ore efficient. Good for complex geometry.





# Next lecture:

**Adding the third dimension!**

**All this and more. Transformations.
Camera placement and projection.**