OpenGL

Jimmy Johansson
Norrköping Visualization and Interaction Studio
Linköping University

Background

- Software interface to graphics hardware
- 250+ commands
- Objects (models) are built from geometric primitives
  - Points, lines, triangles and polygons

- Current Versions
  - 2.x
  - 3.x

Background

- Portability
- Fast rendering
- Only the core
  - Other features are available elsewhere
  - GLU, GLUT, GLEW, GLUI

The State Machine

- No object oriented design
- State machine
  - Set states and parameters
  - Execute commands
- States
  - Transform mode
  - Rendering mode
  - Error state

The State Machine

- glEnable and glDisable
  - e.g.: GL_LIGHTING, GL_DEPTH_TEST
- Error control and feedback
  - glGetError
- Parameters
  - Colour, normal, texture, lighting

API Conventions

- Function names begin with gl and use mixed case:
  `glBegin, glEnd, glFrontFace`
- Constants begin with GL_ and use only upper case:
  `GL_TRIANGLES, GL_BACK, GL_LIGHT0`
- Types begin with GL and use only lower case:
  `GLbyte, GLubyte, GLint, GLfloat, ...`

API conventions: Function names show arguments

- **`GlVertex{234}{sifd}[v]()`**
  - **`glVertex3i(Glint x,Glint y,Glint z)`**
  - **`glVertex3fv(Glfloat *v)`**
- argument types:
  - GLbyte, GLshort, GLint, GLfloat, GLdouble
  - unsigned byte (ub), short (us), int (ui)
- v indicates vector

Example function names

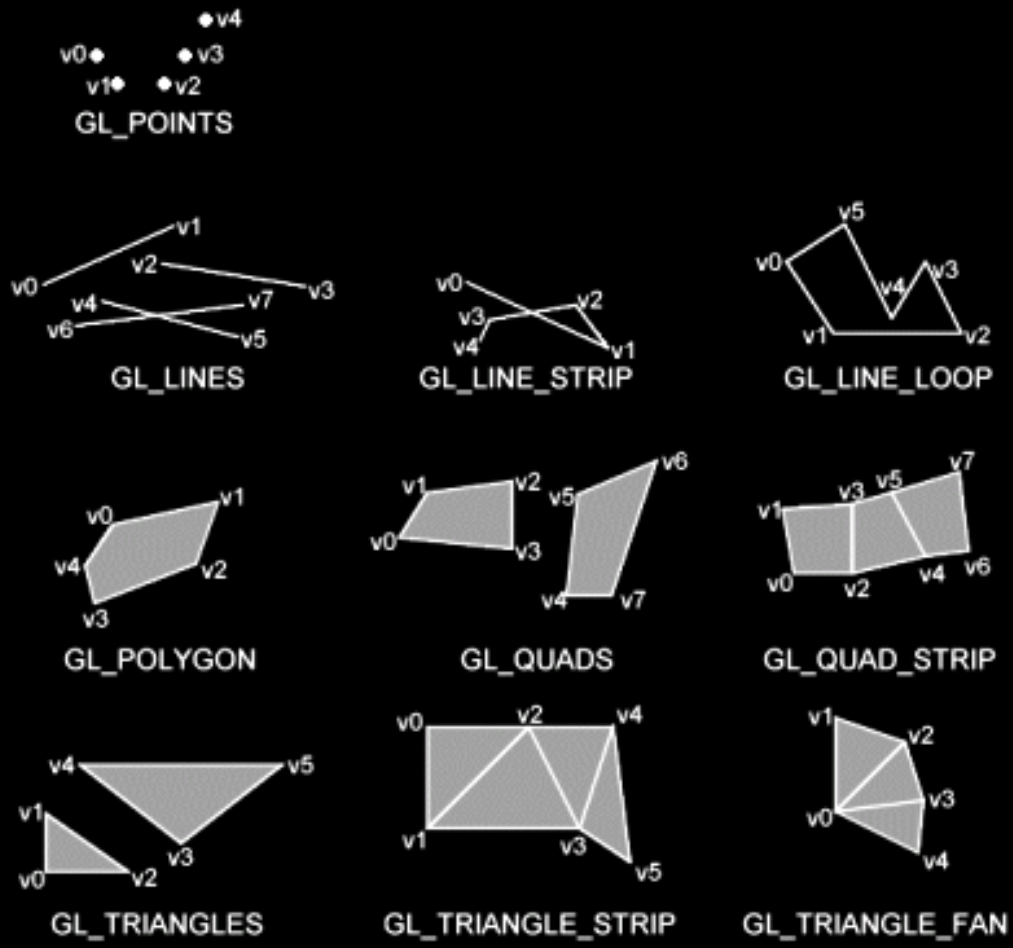
- glVertex3f(1.0, 2.0, 3.142);

- glVertex3dv(dvec);
  - GLdouble dvec[3] = {1.0, 2.0, 3.142};

GL Primitives

- Points
  GL_POINTS
- Lines
  GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP
- Triangles
  GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN
- Quadrilaterals and all other polygons
  GL_QUADS, GL_QUAD_STRIP, GL_POLYGON

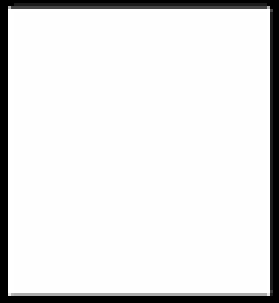
GL Primitives



Example

- Chunk of OpenGL Code (OpenGL “redbook”)

```
#include <whateverYouNeed.h>

main() {
InitializeAWindowPlease();
glClearColor (0.0, 0.0, 0.0, 0.0);
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 1.0);
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
glBegin(GL_POLYGON);
     glVertex3f (0.25, 0.25, 0.0);
     glVertex3f (0.75, 0.25, 0.0);
     glVertex3f (0.75, 0.75, 0.0);
     glVertex3f (0.25, 0.75, 0.0);
glEnd();
glFlush();
UpdateTheWindowAndCheckForEvents();
}
```



Example

- Chunk of OpenGL Code (OpenGL “redbook”)

```
#include <whateverYouNeed.h>

main() {

InitializeAWindowPlease();
glClearColor (0.0, 0.0, 0.0, 0.0);
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 1.0);
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
glBegin(GL_POLYGON);
     glVertex3f (0.25, 0.25, 0.0);
     glVertex3f (0.75, 0.25, 0.0);
     glVertex3f (0.75, 0.75, 0.0);
     glVertex3f (0.25, 0.75, 0.0);
glEnd();
glFlush();
UpdateTheWindowAndCheckForEvents();
}
```

Example

- Chunk of OpenGL Code (OpenGL “redbook”)

```
#include <whateverYouNeed.h>

main() {
InitializeAWindowPlease();
glClearColor (0.0, 0.0, 0.0, 0.0);
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 1.0);
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
glBegin(GL_POLYGON);
     glVertex3f (0.25, 0.25, 0.0);
     glVertex3f (0.75, 0.25, 0.0);
     glVertex3f (0.75, 0.75, 0.0);
     glVertex3f (0.25, 0.75, 0.0);
glEnd();
glFlush();
UpdateTheWindowAndCheckForEvents();
}
```

Example

- Chunk of OpenGL Code (OpenGL “redbook”)

```
#include <whateverYouNeed.h>

main() {
InitializeAWindowPlease();
glClearColor (0.0, 0.0, 0.0, 0.0);
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 1.0);
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
glBegin(GL_POLYGON);
     glVertex3f (0.25, 0.25, 0.0);
     glVertex3f (0.75, 0.25, 0.0);
     glVertex3f (0.75, 0.75, 0.0);
     glVertex3f (0.25, 0.75, 0.0);
glEnd();
glFlush();
UpdateTheWindowAndCheckForEvents();
}
```

Example

- Chunk of OpenGL Code (OpenGL “redbook”)

```
#include <whateverYouNeed.h>

main() {
InitializeAWindowPlease();
glClearColor (0.0, 0.0, 0.0, 0.0);
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 1.0);
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
glBegin(GL_POLYGON);
     glVertex3f (0.25, 0.25, 0.0);
     glVertex3f (0.75, 0.25, 0.0);
     glVertex3f (0.75, 0.75, 0.0);
     glVertex3f (0.25, 0.75, 0.0);
glEnd();
glFlush();
UpdateTheWindowAndCheckForEvents();
}
```

Example

- Chunk of OpenGL Code (OpenGL “redbook”)

```
#include <whateverYouNeed.h>

main() {
InitializeAWindowPlease();
glClearColor (0.0, 0.0, 0.0, 0.0);
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 1.0);
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
glBegin(GL_POLYGON);
     glVertex3f (0.25, 0.25, 0.0);
     glVertex3f (0.75, 0.25, 0.0);
     glVertex3f (0.75, 0.75, 0.0);
     glVertex3f (0.25, 0.75, 0.0);
glEnd();
glFlush();
UpdateTheWindowAndCheckForEvents();
}
```

Example

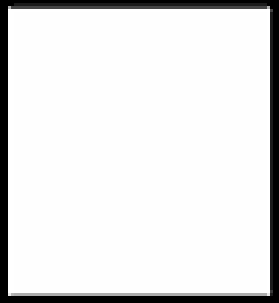
- Chunk of OpenGL Code (OpenGL “redbook”)

```
#include <whateverYouNeed.h>

main() {
InitializeAWindowPlease();
glClearColor (0.0, 0.0, 0.0, 0.0);
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 1.0);
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
glBegin(GL_POLYGON);
     glVertex3f (0.25, 0.25, 0.0);
     glVertex3f (0.75, 0.25, 0.0);
     glVertex3f (0.75, 0.75, 0.0);
     glVertex3f (0.25, 0.75, 0.0);
glEnd();
glFlush();
UpdateTheWindowAndCheckForEvents();
}
```

Example

- Chunk of OpenGL Code (OpenGL “redbook”)

```
#include <whateverYouNeed.h>

main() {
InitializeAWindowPlease();
glClearColor (0.0, 0.0, 0.0, 0.0);
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 1.0, 1.0);
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
glBegin(GL_POLYGON);
     glVertex3f (0.25, 0.25, 0.0);
     glVertex3f (0.75, 0.25, 0.0);
     glVertex3f (0.75, 0.75, 0.0);
     glVertex3f (0.25, 0.75, 0.0);
glEnd();
glFlush();
UpdateTheWindowAndCheckForEvents();
}
```



Rendering

- Vertex-by-vertex
  - Begin geometry state
  - Put vertices
  - End geometry state
- Array of data
  - Vertex pointer
  - Draw arrays
  - Draw elements

Vertex by vertex

```
glBegin(GL_TRIANGLES);
   /* First Triangle */
   glVertex3f( 1.0, 0.0, 0.0);
   glVertex3f(-0.5, 1.0, 0.5);
   glVertex3f( 0.0,-0.5,-0.2);
  /* Second Triangle */
   glVertex3f( 1.0,-0.4, 0.0);
   glVertex3f( 0.0, 0.6, 0.0);
   glVertex3f(-0.6,-0.2, 0.4);
glEnd();
```

Vertex by vertex

```
glBegin(GL_QUAD_STRIP);
  /* First quad */
  glVertex3f(-0.5,-0.5,-0.5);
  glVertex3f(-0.5, 0.5,-0.5);
  glVertex3f( 0.5,-0.5,-0.5);
  glVertex3f( 0.5, 0.5,-0.5);
  /* Second */
  glVertex3f( 0.5,-0.5, 0.5);
  glVertex3f( 0.5, 0.5, 0.5);
  /* Third */
  glVertex3f(-0.5,-0.5, 0.5);
  glVertex3f(-0.5, 0.5, 0.5);
  /* Fourth */
  glVertex3f(-0.5,-0.5,-0.5);
  glVertex3f(-0.5, 0.5,-0.5);
glEnd();
```

Array of Data

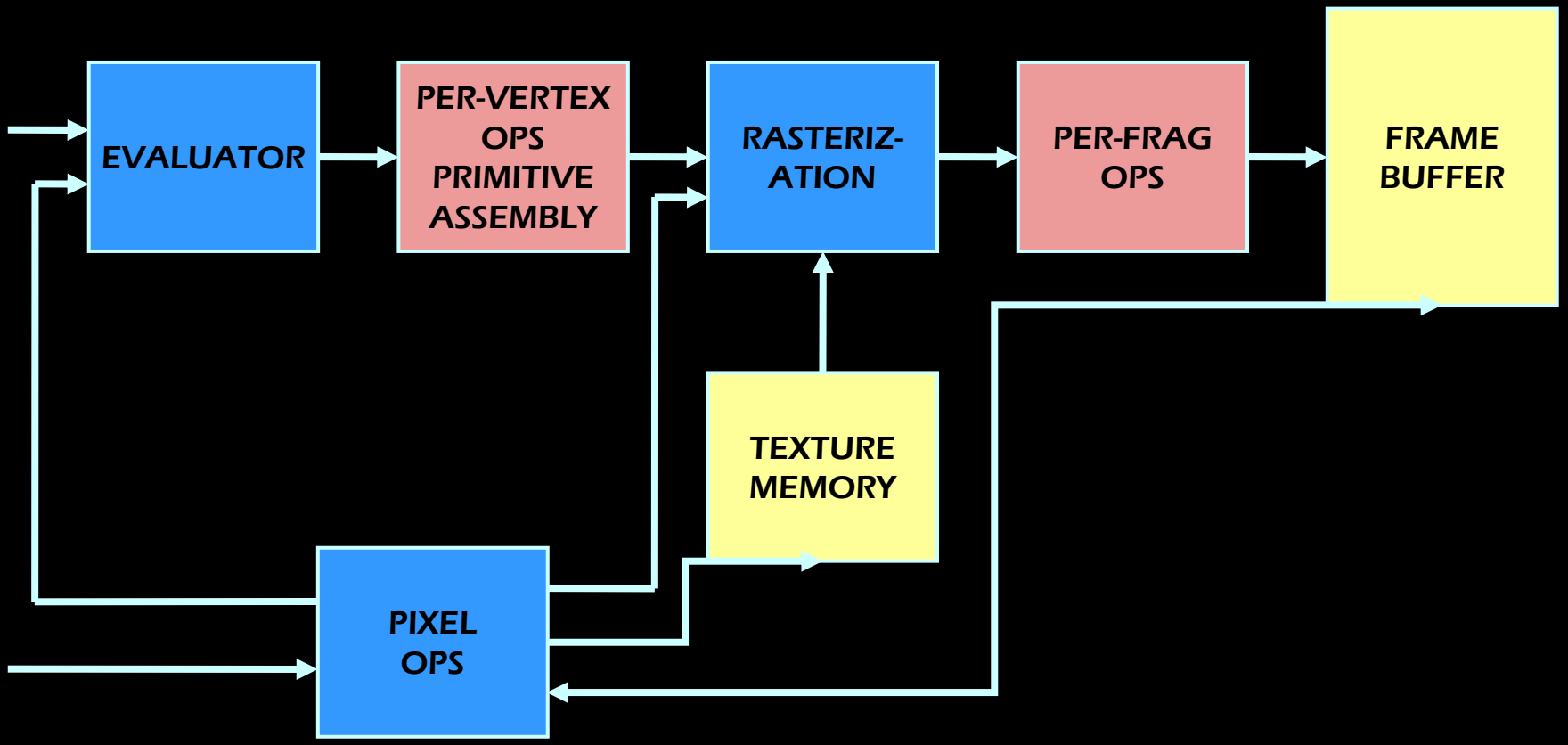
```
static GLfloat vertices[] = {  1.0, 0.0, 0.0,
                               -0.5, 1.0, 0.5,
                               0.0,-0.5,-0.2,
                               1.0,-0.4, 0.0,
                               0.0, 0.6, 0.0,
                               -0.6,-0.2, 0.4 };
```

```
glVertexPointer( 3, GL_FLOAT, 0, vertices );
glEnableClientState(GL_VERTEX_ARRAY);
glDrawArrays( GL_TRIANGLES, 0, 6 );
glDisableClientState(GL_VERTEX_ARRAY);
```

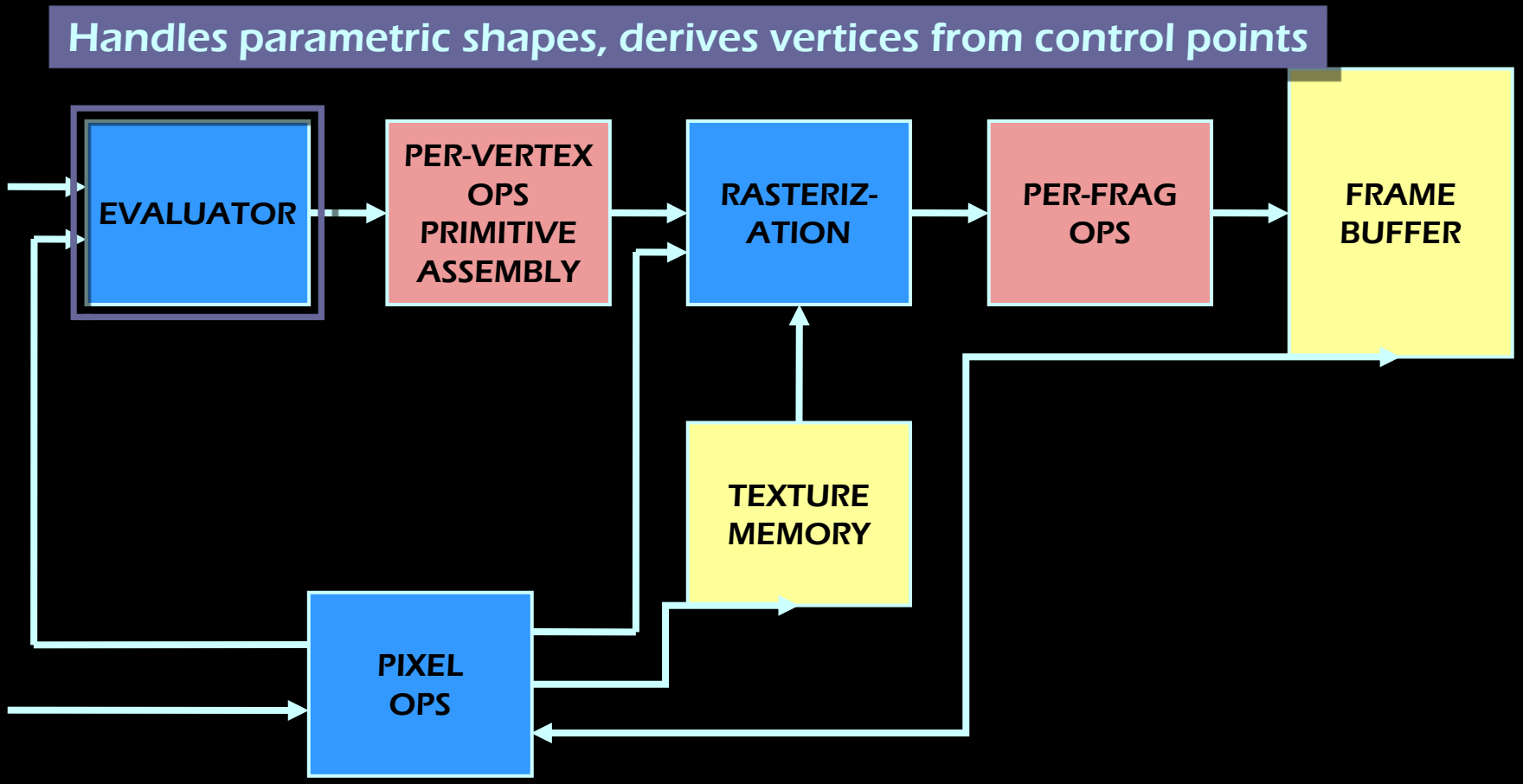
Array of Data

```
static GLfloat vertices[] = {  1.0, 0.0, 0.0,
                               -0.5, 1.0, 0.5,
                               0.0,-0.5,-0.2,
                               1.0,-0.4, 0.0,
                               0.0, 0.6, 0.0,
                               -0.6,-0.2, 0.4 };

static GLubyte indices[] = { 0, 1, 2, 3, 4, 5 };


glVertexPointer( 3, GL_FLOAT, 0, vertices );
glEnableClientState(GL_VERTEX_ARRAY);
glDrawElements( GL_TRIANGLES, 6, GL_UNSIGNED_BYTE, indices );

glDisableClientState(GL_VERTEX_ARRAY);
```

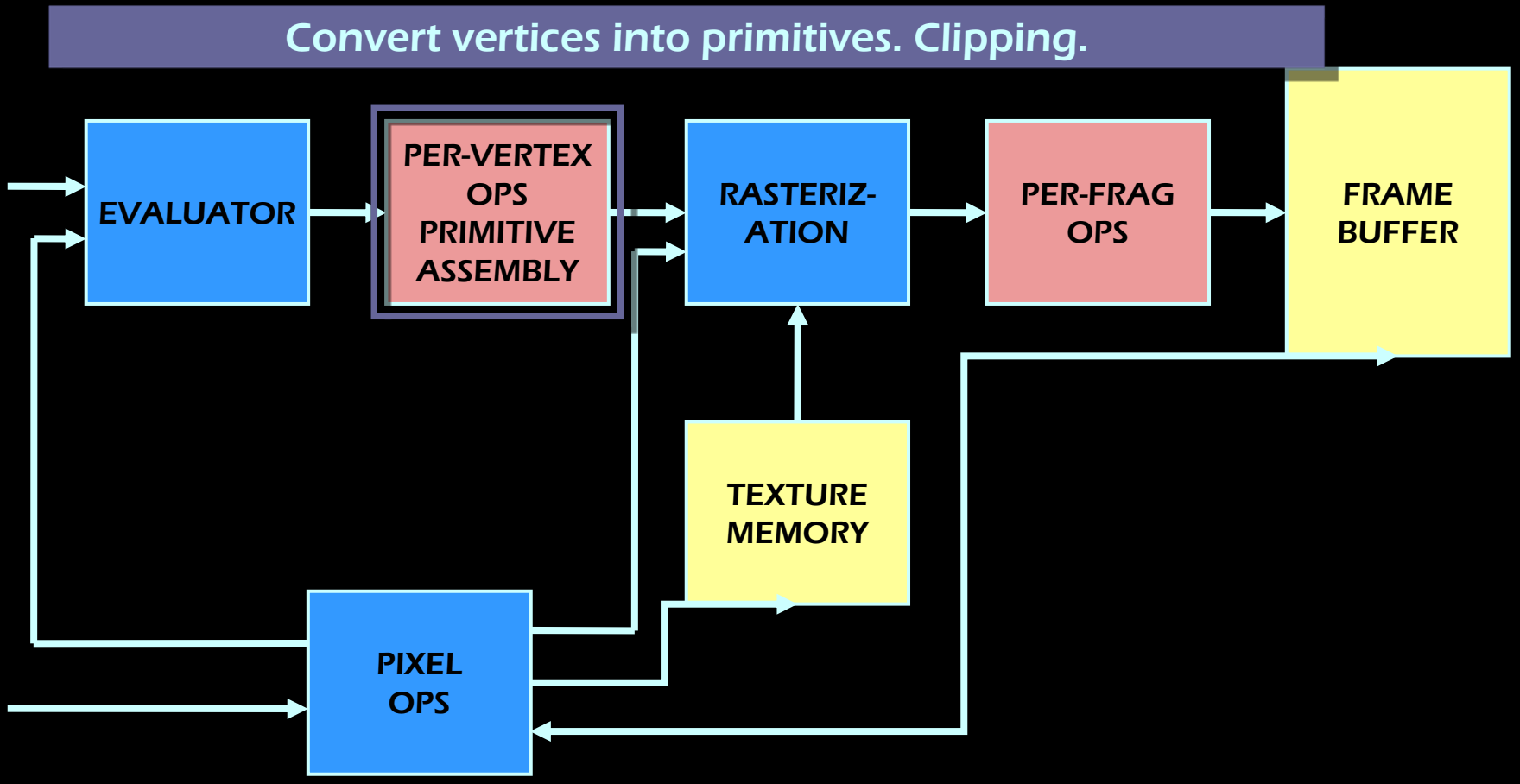
OpenGL Block Diagram



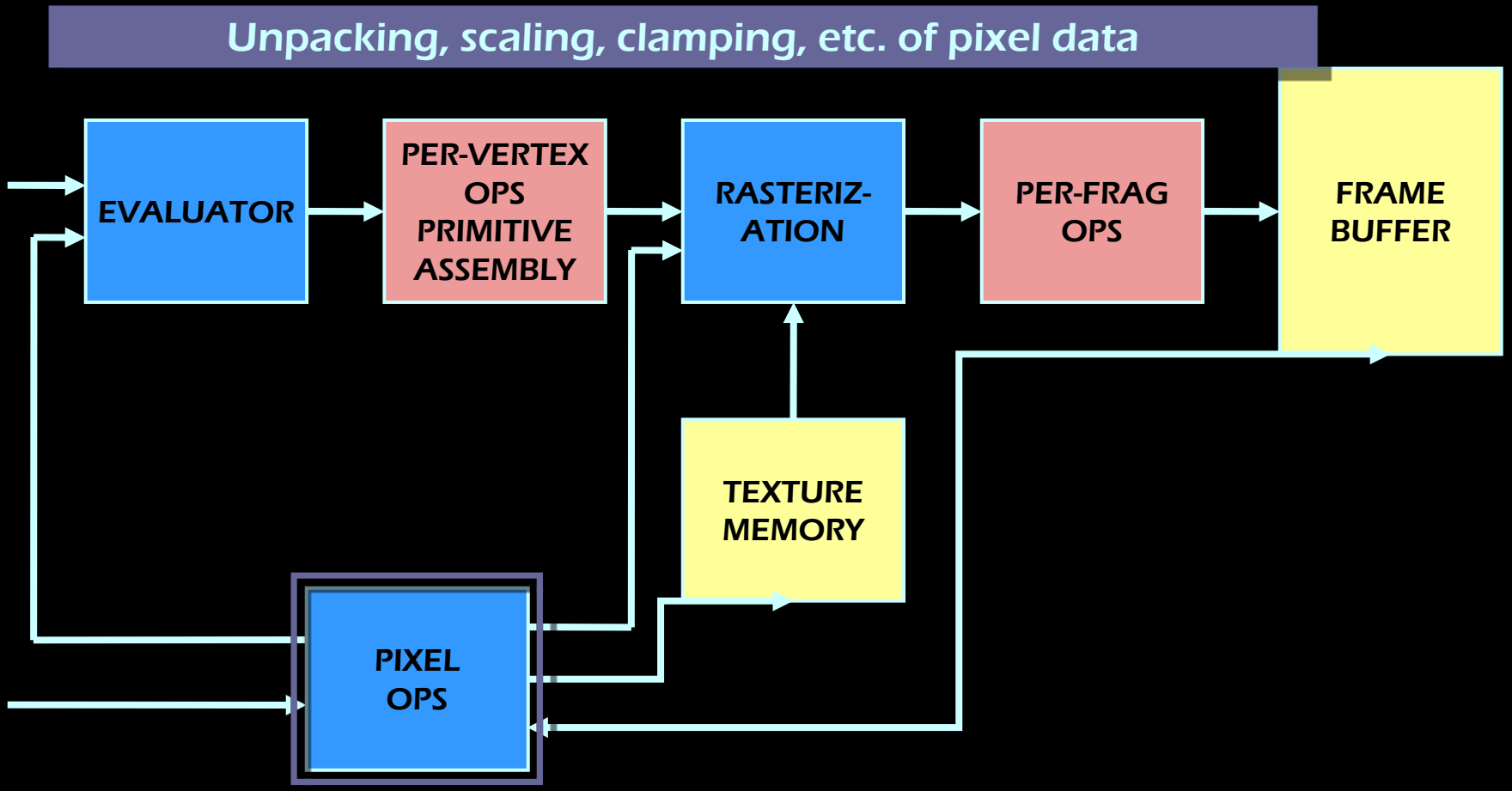
OpenGL Block Diagram



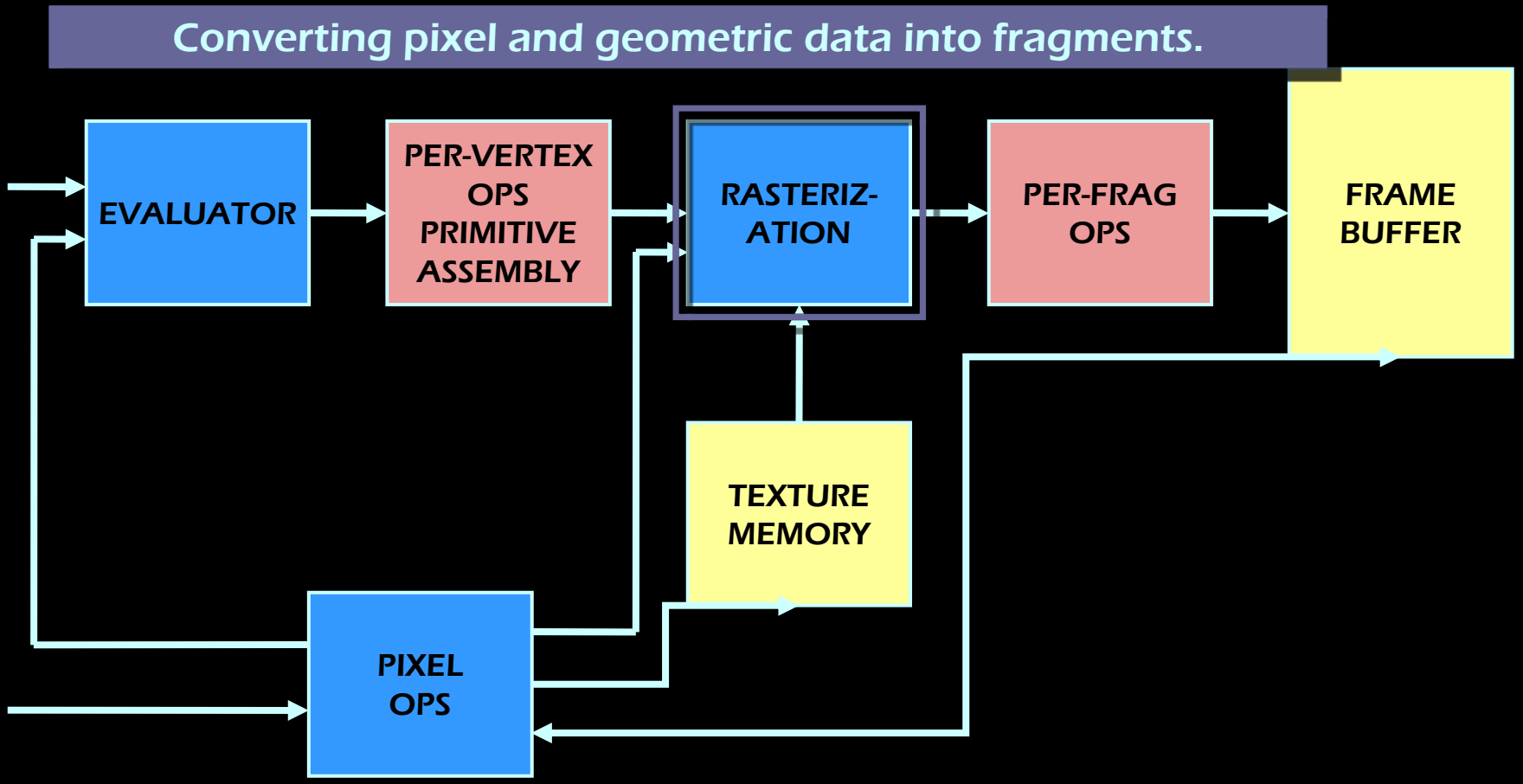
OpenGL Block Diagram



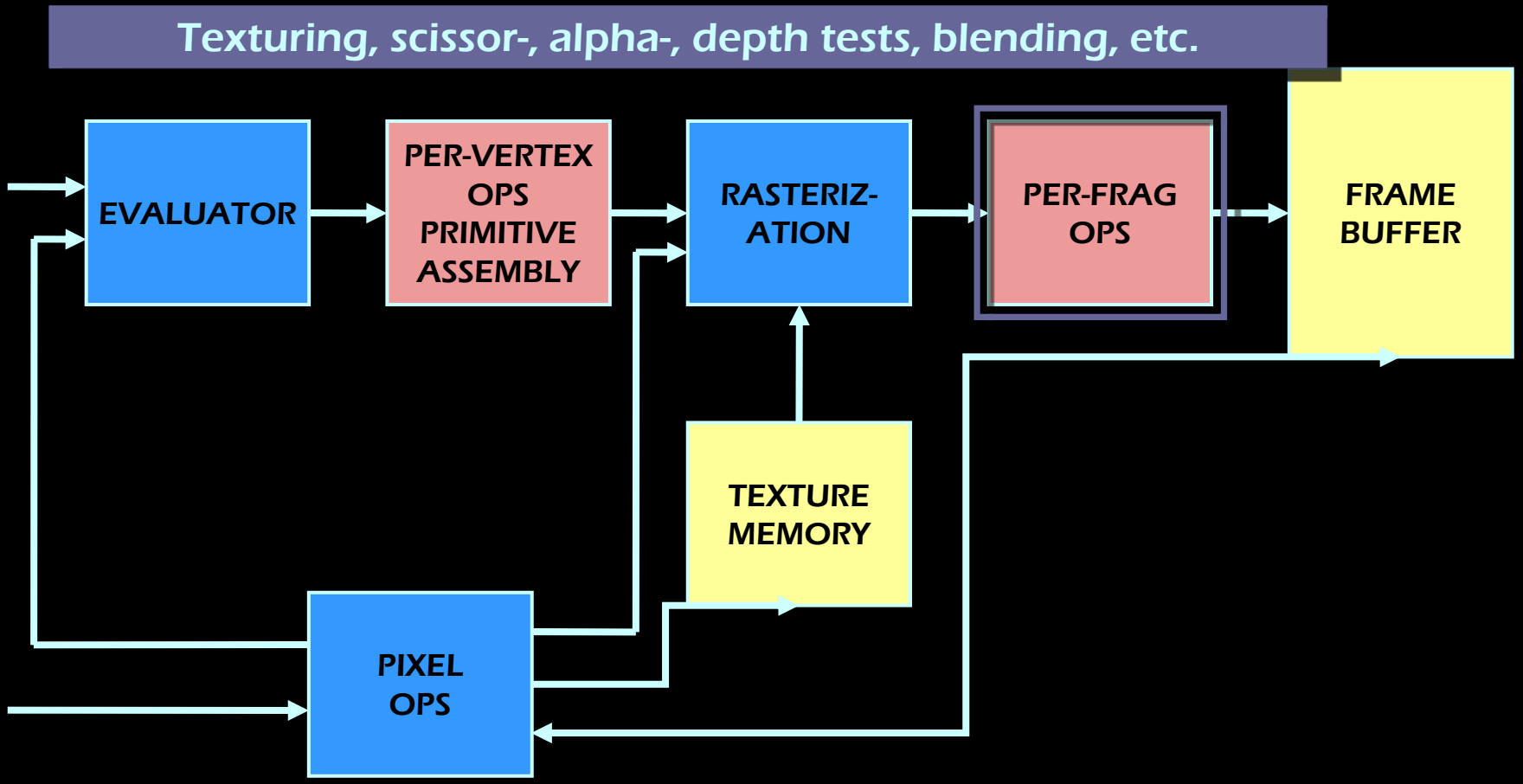
OpenGL Block Diagram



OpenGL Block Diagram

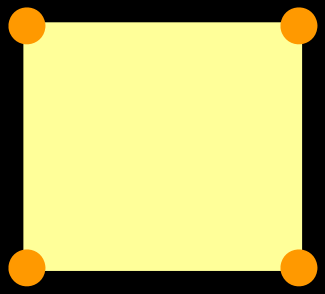


OpenGL Block Diagram



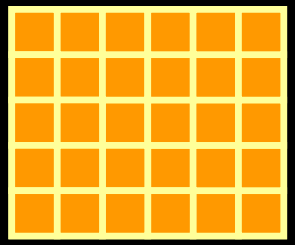
Rasterization

- Convert primitives into fragments



Vertices

Rasterization



Fragments

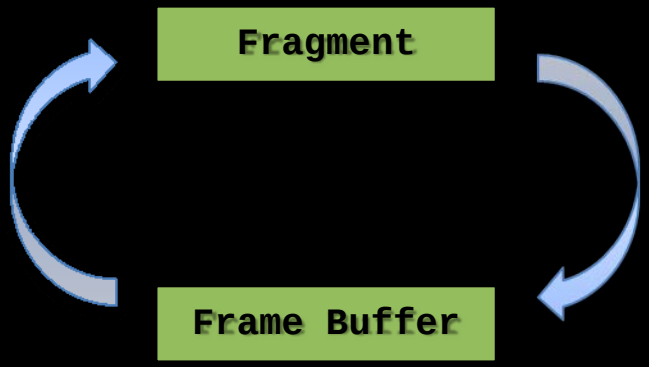
- Fragments - data necessary to *potentially* update a pixel in the frame buffer
  - depth
  - interpolated attributes
    - color
    - texture coords
    - etc
  - alpha
  - Normal

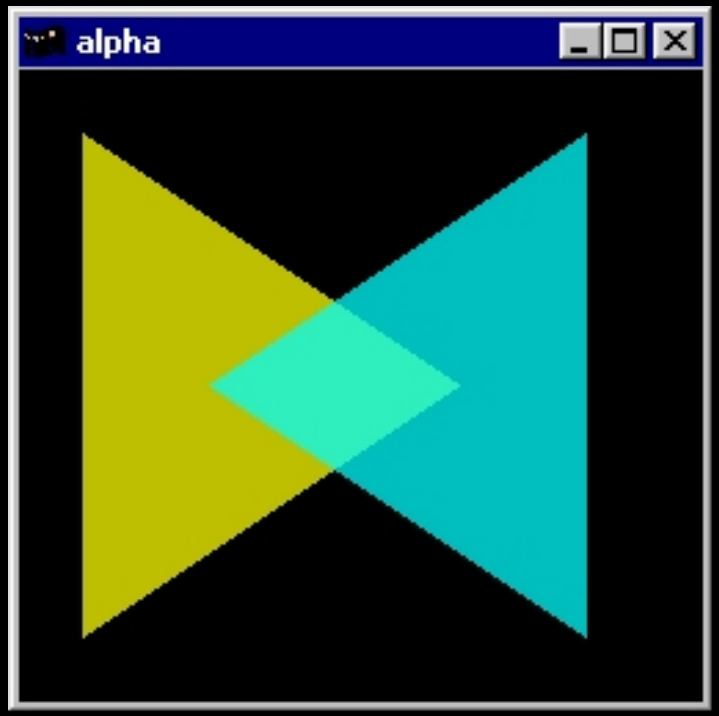
Rasterization

- glHint
  - Antialiasing, perspective correction

Blending

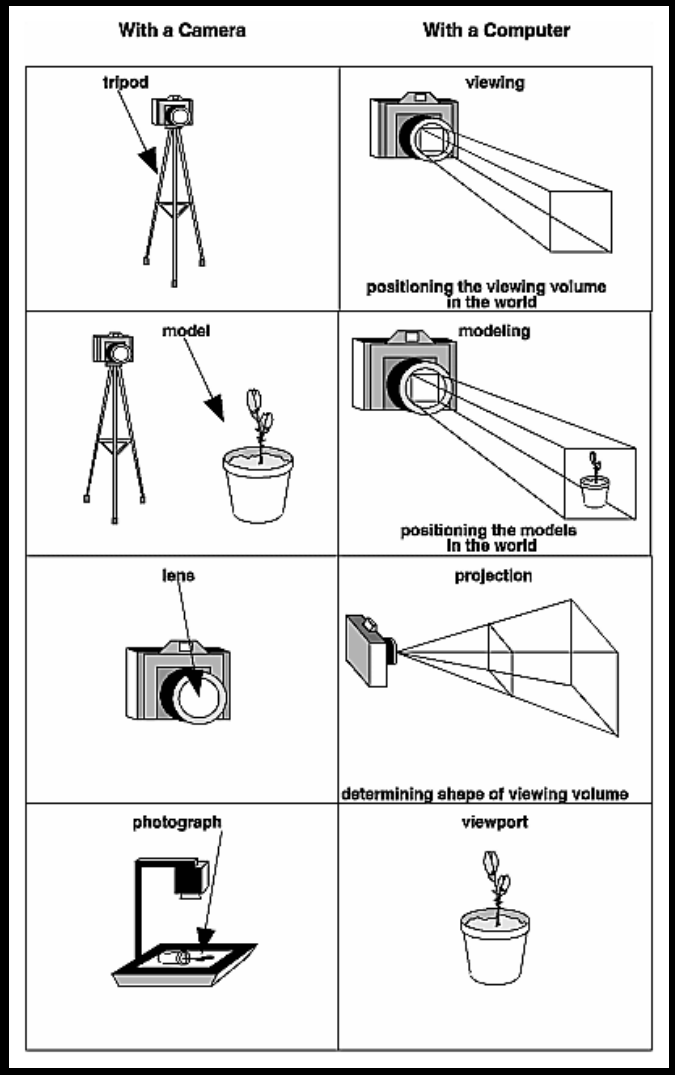
- Combine colour value of fragment with pixel value stored in framebuffer
  - Order dependent





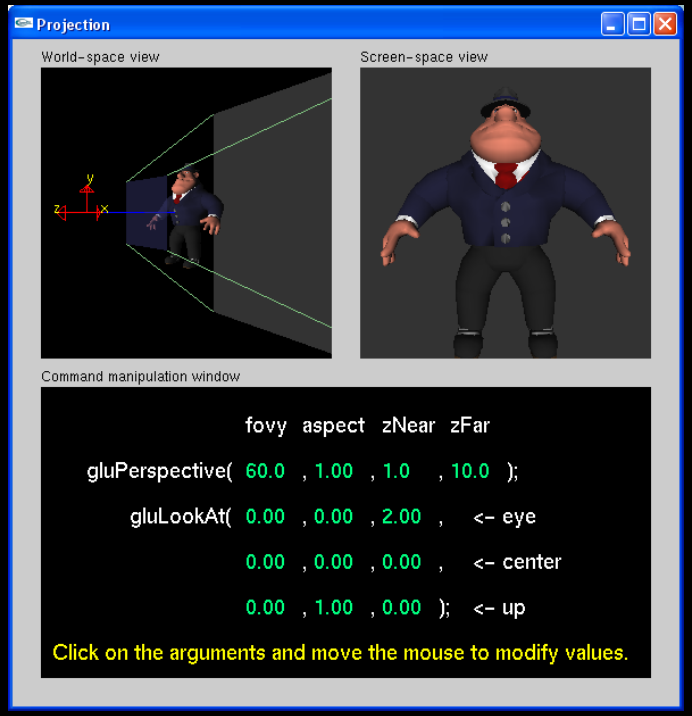
Transforms

- View transform
- Model transform
- Projection transform
- Viewport transform



View Transform

- Viewpoint/Camera
  - `glOrtho( L, R, B, T, N, F )`
  - `gluPerspective( fovy, aspect, zNear, zFar )`



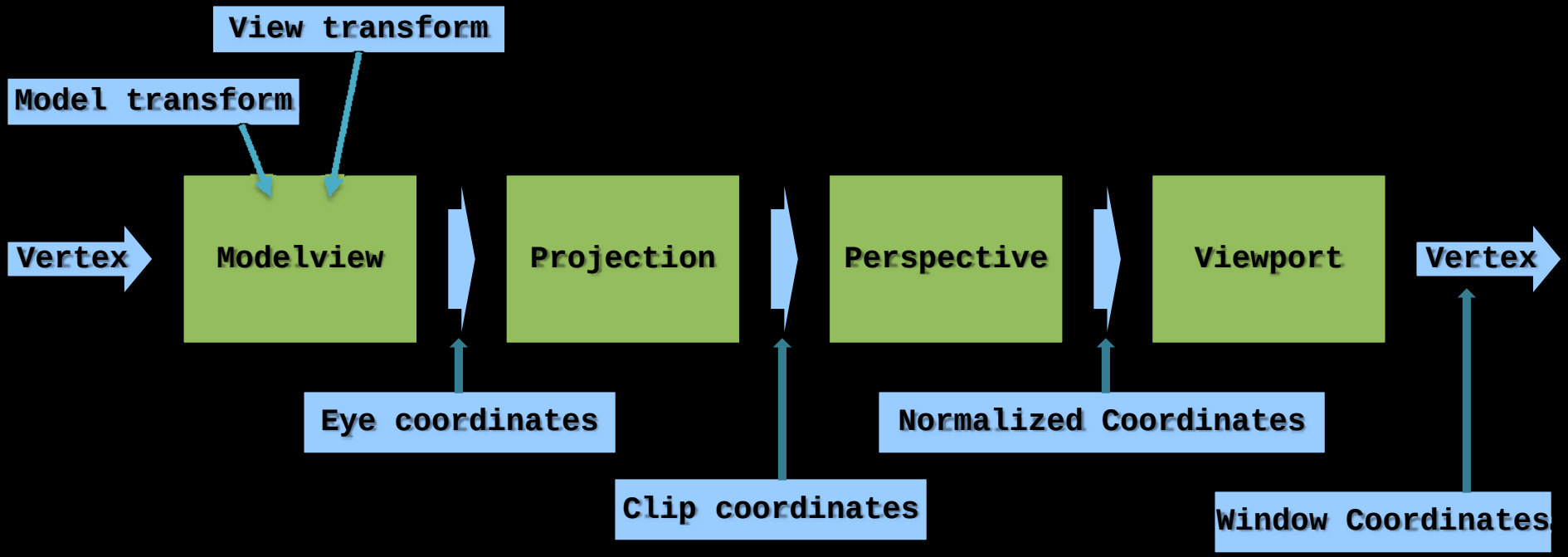
Model Transform

- Management
  - **`GlLoadMatrix/Identity`**
  - **`glPush/PopMatrix`**

- Manipulation
  - **`glTranslate`**
  - **`glRotate`**
  - **`glScale`**

Transforms Put Together

- Model + View transform = Modelview transform
- Projection transform
- Viewport transform



Colours

- glColor, glClearColor
  - RGB, RGBA,
  - Execute with glVertex, glClear

Rendering with Colours

```
glBegin(GL_TRIANGLES);

glColor4f( 1.0, 0.0, 0.0, 1.0 );
glVertex3f( 1.0, 0.0, 0.0);
glVertex3f(-0.5, 1.0, 0.5);
glVertex3f( 0.0,-0.5,-0.2);

glColor4f( 0.0, 0.0, 1.0, 0.5 );
glVertex3f( 1.0,-0.4, 0.0);
glVertex3f( 0.0, 0.6, 0.0);
glVertex3f(-0.6,-0.2, 0.4);

glEnd();
```

Rendering with Colour Array

```
static GLfloat vertices[] = {  1.0, 0.0, 0.0,
                               -0.5, 1.0, 0.5,
                               0.0,-0.5,-0.2,
                               1.0,-0.4, 0.0,
                               0.0, 0.6, 0.0,
                               -0.6,-0.2, 0.4 };
static GLfloat colors[] = {  1.0, 0.0, 0.0, 1.0,
                             1.0, 0.0, 0.0, 1.0,
                             1.0, 0.0, 0.0, 1.0,
                             0.0, 0.0, 1.0, 0.5,
                             0.0, 0.0, 1.0, 0.5,
                             0.0, 0.0, 1.0, 0.5 };
```

```
glVertexPointer( 3, GL_FLOAT, 0, vertices );
glColorPointer( 3, GL_FLOAT, 0, colors );

glEnableClientState(GL_VERTEX_ARRAY|GL_COLOR_ARRAY);
glDrawArrays( GL_TRIANGLES, 0, 6 );
glDisableClientState(GL_VERTEX_ARRAY|GL_COLOR_ARRAY);
```

Rendering with Interleaved Arrays

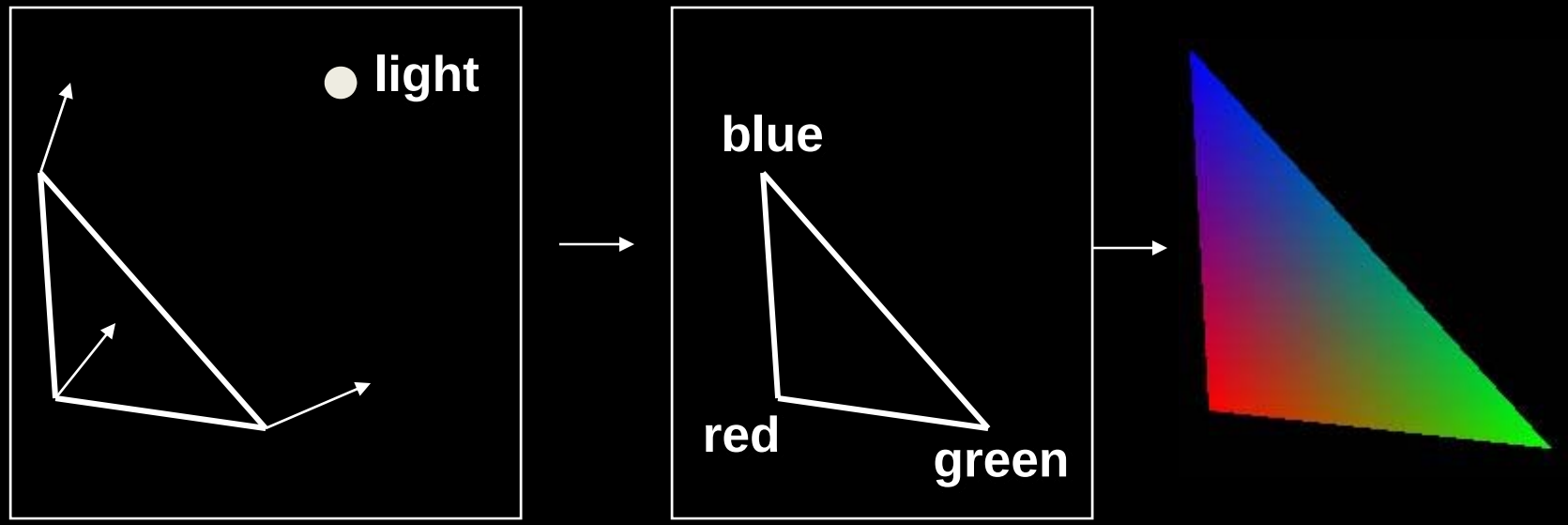
```
static GLfloat colverts[] = {  1.0, 0.0, 0.0, 1.0,
                               1.0, 0.0, 0.0,
                               1.0, 0.0, 0.0, 1.0,
                              -0.5, 1.0, 0.5,
                               1.0, 0.0, 0.0, 1.0,
                               0.0,-0.5,-0.2,
                               0.0, 0.0, 1.0, 0.5,
                               1.0,-0.4, 0.0,
                               0.0, 0.0, 1.0, 0.5,
                               0.0, 0.6, 0.0,
                               0.0, 0.0, 1.0, 0.5,
                              -0.6,-0.2, 0.4 };
```

```
glInterleavedArrays( GL_C4F_V3F, 0, colverts );
glDrawArrays( GL_TRIANGLES, 0, 6 );
glDisableClientState(GL_VERTEX_ARRAY|GL_COLOR_ARRAY);
```

Shading

- Activate shading
  - glShadeModel, flat or smooth
  - Requires normalized normals
- Automatic normalization
  - glEnable(GL_NORMALIZE)
  - Expensive!

Smooth Shading



Textures

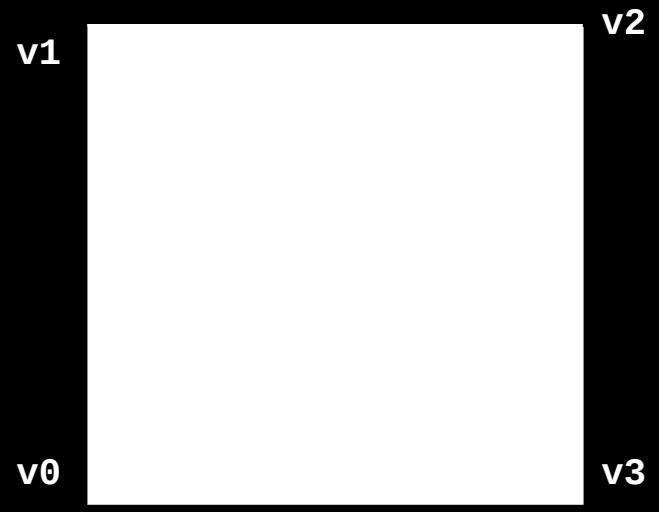
- Texture states
  - Targets: 1D, 2D, 3D texture
    - **`glEnable / glDisable`**
  - **`glTexEnv{if}[v],`** e g **`GL_TEXTURE_ENV_MODE`**
    - Decal, replace, modulate, (blend, add, combine)
  - **`glTexParameter{if}[v]`**
    - e g border, interpolation, level-of-details
- Texture state sets
  - **`glGenTextures`**
  - **`glBindTexture`**

Texture Data

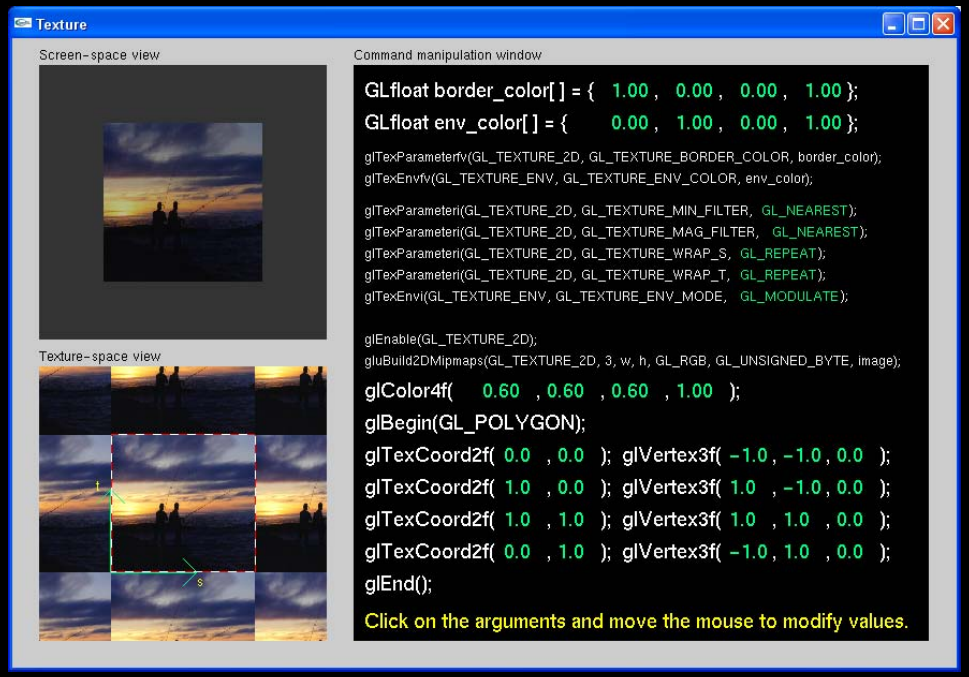
- Load data into memory
  - **`glTexImage{123}D / glTexSubImage{123}D`**
    - Internal format, data format, border, size, type, etc

Texture Coordinates

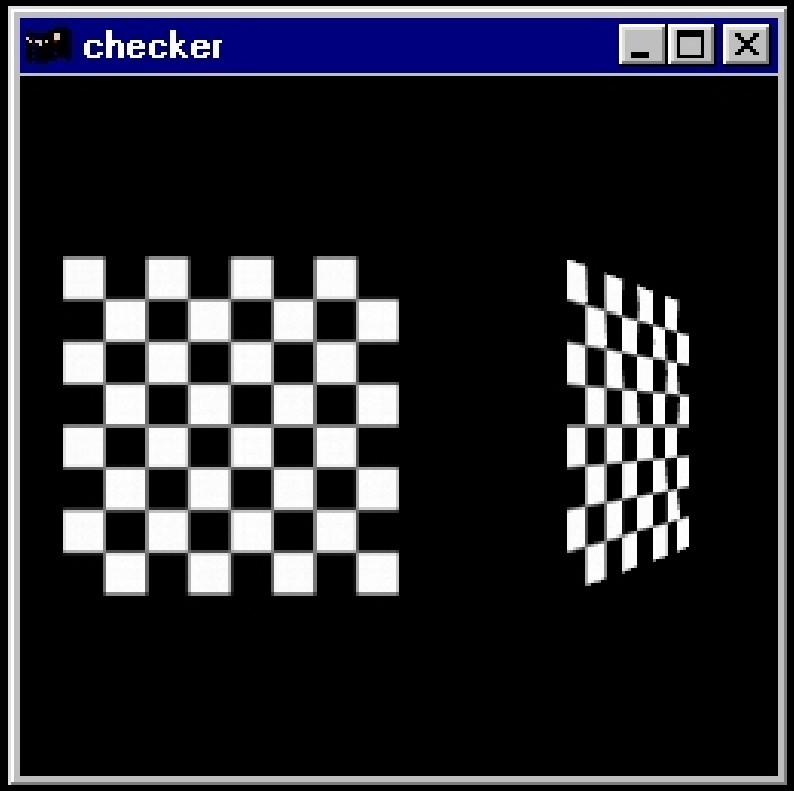
- Vertex-by-vertex
  - **`glTexCoord`**
- Vertex arrays
  - **`glTexCoordPointer`**
- Automatic coordinates
  - **`glTexGen`**



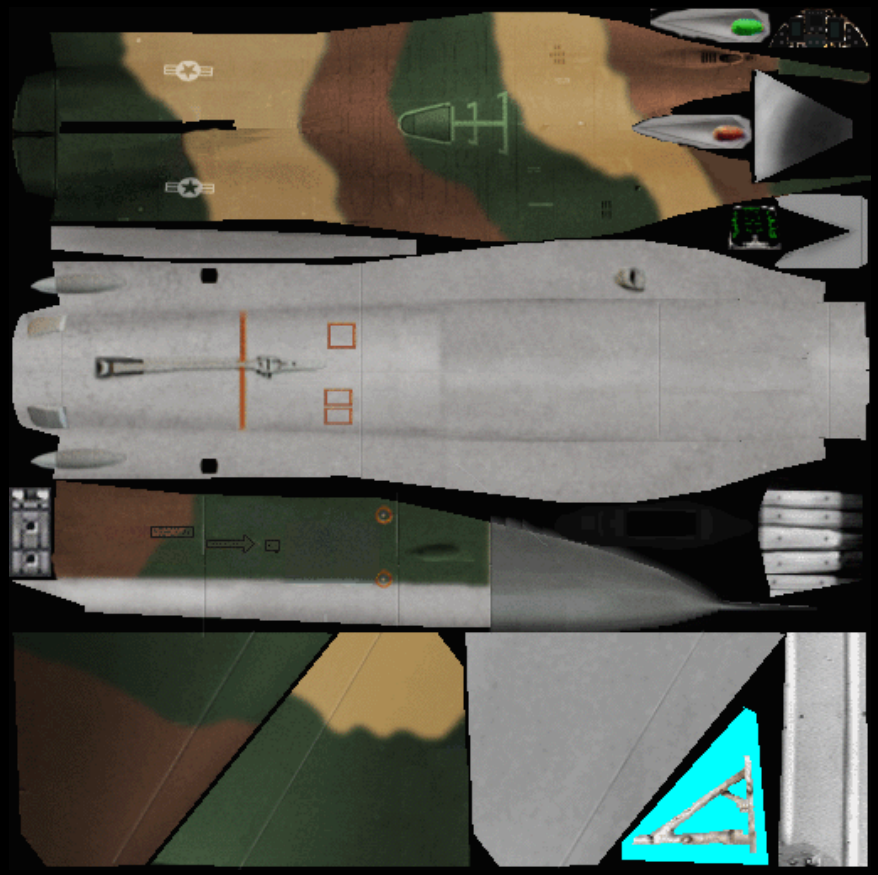
2D Texture Example

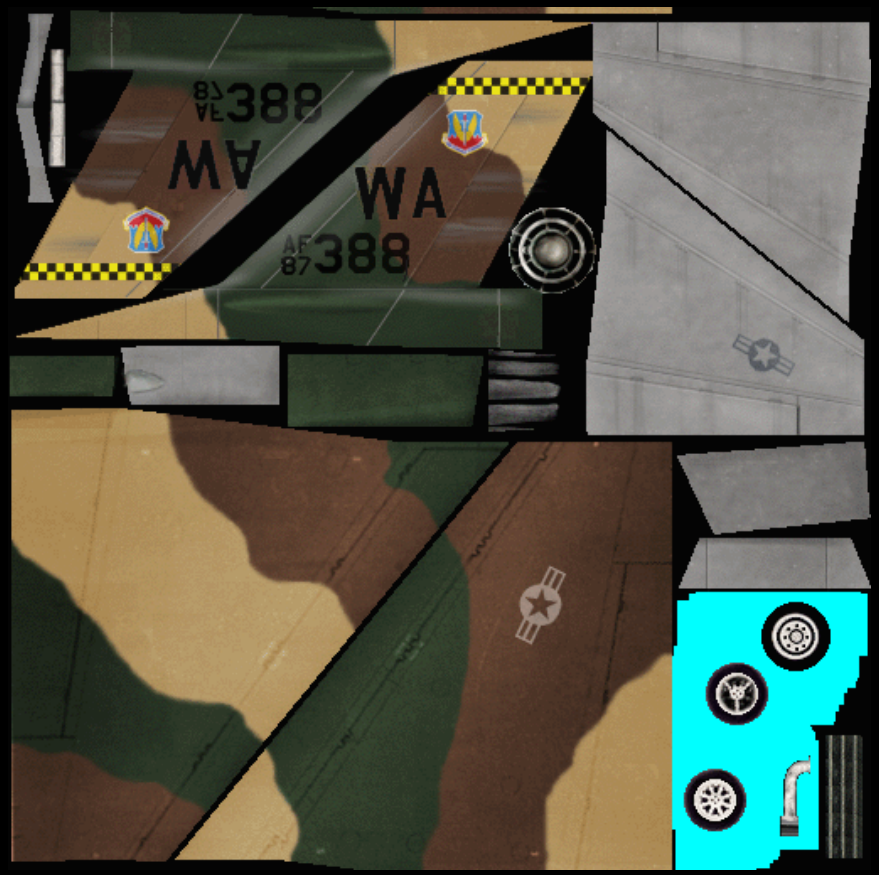


2D Texture Code Example



2D Texture Example





2D Texture Example

Lighting

- glLightModel
  - E g local or infinitively far away, two sided lighting

- Light sources
  - Eight sources
  - Head light default (GL_LIGHT0)
- Parameters
  - glLight{if}[v](*source*, *pname*, *value(s)*)
  - Ambient, diffuse, specular, position (direction), spot direction, cutoff, attenuation

Material

- Object materials
  - Activated by lighting
- Parameters
  - glMaterial{if}[v](*face*, *pname*, *value(s)*)
  - Diffuse, ambient-diffuse,
    specular, shininess, emission

GLU (OpenGL Utility Library)

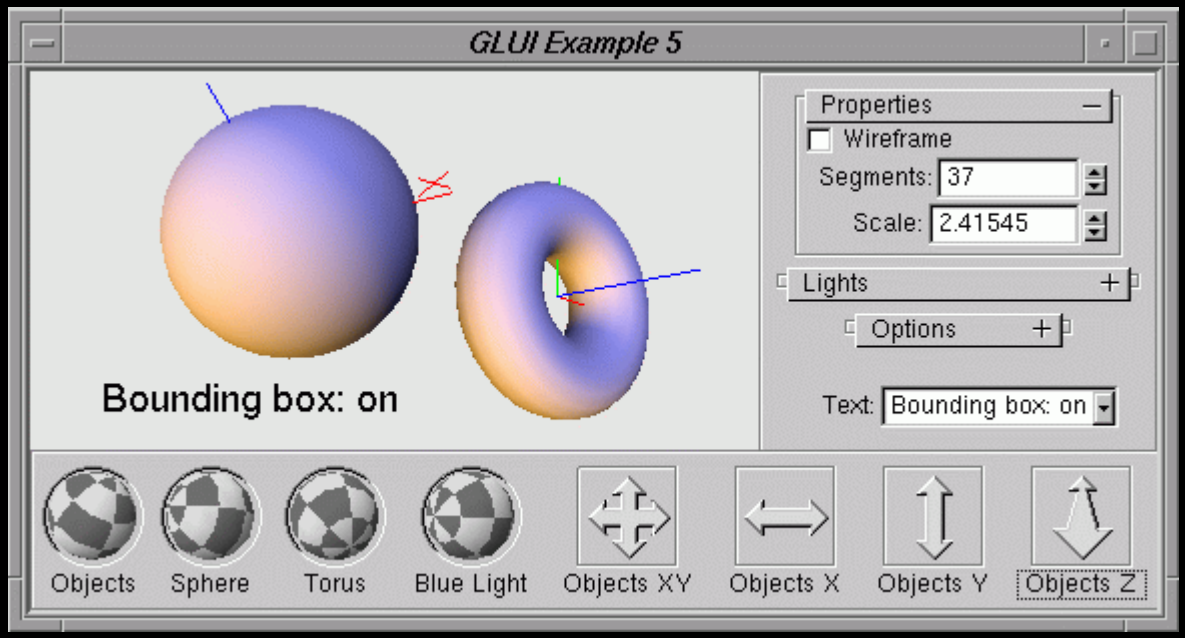
- Composed geometries
  - gluSphere, gluCylinder, etc
- Tesselators, Nurbs
  - Subdivide complex polygons
  - Nurb lines, nurb surfaces
- Camera management
  - gluPerspective, gluLookat

GLUT (OpenGL Utility Toolkit)

- Window handling
  - Init context, display mode, etc
  - Create windows, set size, etc
  - Entry point is glutMainLoop
- Events
  - Mouse, keyboard, display, etc
  - Call backs (hooks)
  - Post redisplay
- Complex objects
  - Sphere, torus, teapot, etc

GLUI

- User Interface library
  - Add-on to OpenGL
  - Portable (uses OpenGL)
  - Easy to use



GLEW (Extension Wrangler Library)

- Cross-platform open-source C/C++ extension loading library
- Header exposes complete set
  - Standard functions for any OpenGL version
  - All extensions
- Functions to check capabilities
  - Actual version of OpenGL
  - Actually supported functions and extensions