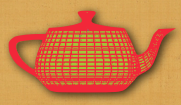
OpenGL 3D Graphics



Hans Dulimarta, Ph.D
dulimarh@cis.gvsu.edu
Associate Professor

Grand Valley State University

What is OpenGL?

What is OpenGL?

- Open Graphics Library

What is OpenGL?

- Open Graphics Library
  - API for 2D and 3D desktop graphics

What is OpenGL?

- Open Graphics Library
  - API for 2D and 3D desktop graphics
  - Silicon Graphics Inc. (1992)

What is OpenGL?

- Open Graphics Library
  - API for 2D and 3D desktop graphics
  - Silicon Graphics Inc. (1992)
  - KHRONOS GROUP (1994)

How to "create" a [2D] picture of [3D] objects? (if you are not a programmer)

Programming Paradigms

Programming Paradigms

- Painter vs. Photographer Paradigm

Programming Paradigms

- Painter vs. Photographer Paradigm
  - Painter: canvas, brush, color palette

Programming Paradigms

- Painter vs. Photographer Paradigm
  - Painter: canvas, brush, color palette
  - Photographer: **Objects**, **Light Source(s)**, **Cameras**

Programming Paradigms

- Painter vs. Photographer Paradigm
  - Painter: canvas, brush, color palette
  - Photographer: **Objects**, **Light Source(s)**, **Cameras**

Programming Paradigms

- Painter vs. Photographer Paradigm
  - Painter: canvas, brush, color palette
  - Photographer: **Objects**, **Light Source(s)**, **Cameras**

OpenGL APIs

- "Photographer" APIs
  - "create" **3D objects** and specify their properties (color, "type of material", ...)
  - specify properties of **light sources**
  - specify properties of the viewer's **camera**
    - image resolution, zoom/normal/macro lense
  - Place/**move** these objects, light sources, and the camera in a 3D world

Building Blocks of Graphical Objects

Building Blocks of Graphical Objects

- Points (Vertices)

Building Blocks of Graphical Objects

- Points (Vertices)

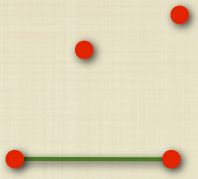
Building Blocks of Graphical Objects

- Points (Vertices)

- Lines

Building Blocks of Graphical Objects

- Points (Vertices)



- Lines

Building Blocks of Graphical Objects

- Points (Vertices)

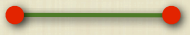
- Lines

- Planes (triangles or quads)

Building Blocks of Graphical Objects

- Points (Vertices)

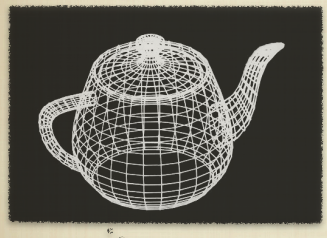
- Lines



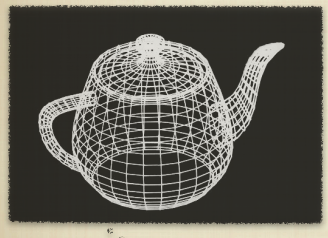
- Planes (triangles or quads)

The Utah Teapot

The Utah Teapot

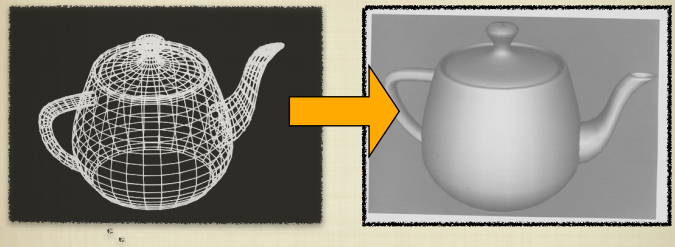


The Utah Teapot

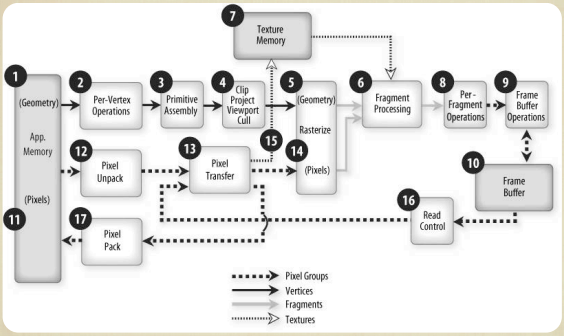


The Utah Teapot

The Utah Teapot



Graphics Pipeline (Simplified)



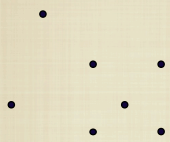
Graphics Pipeline

Graphics Pipeline

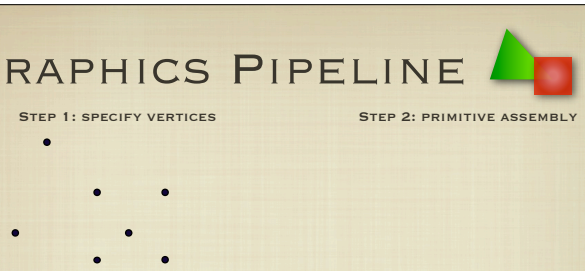
Step 1: specify vertices

Graphics Pipeline

Step 1: specify vertices



Graphics Pipeline



Step 1: specify vertices

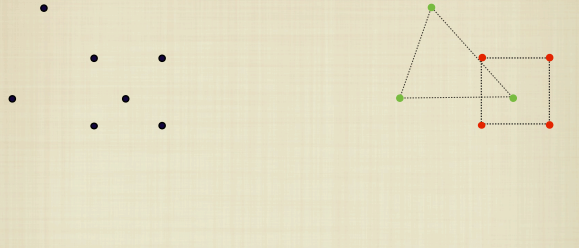
Step 2: primitive assembly

Graphics Pipeline

Step 1: specify vertices

Step 2: primitive assembly

Graphics Pipeline



Step 1: specify vertices

Step 2: primitive assembly

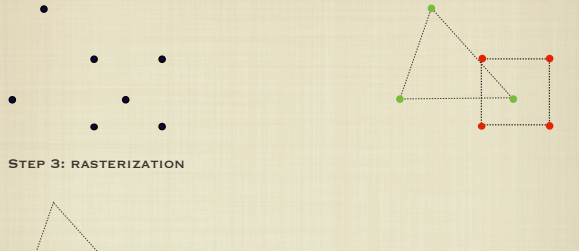
Graphics Pipeline

Step 1: specify vertices

Step 2: primitive assembly

Step 3: rasterization

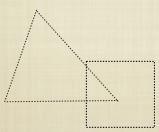
Graphics Pipeline



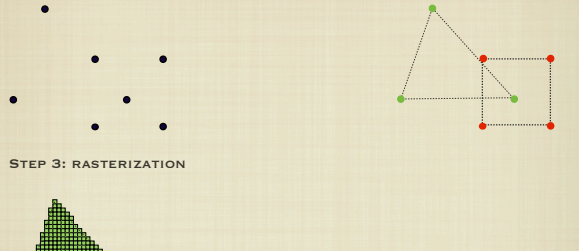
Step 1: specify vertices

Step 2: primitive assembly

Step 3: rasterization



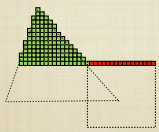
Graphics Pipeline



Step 1: specify vertices

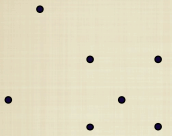
Step 2: primitive assembly

Step 3: rasterization

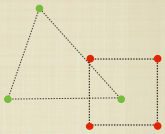


Graphics Pipeline

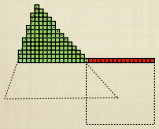
Step 1: specify vertices



Step 2: primitive assembly



Step 3: rasterization



Step 4: fragment operations

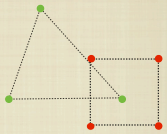
9

9

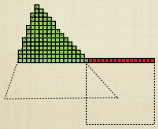
Graphics Pipeline

Step 1: specify vertices

Step 2: primitive assembly



Step 3: rasterization



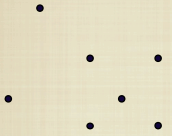
Step 4: fragment operations

9

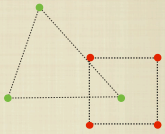
9

Graphics Pipeline

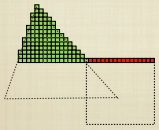
Step 1: specify vertices



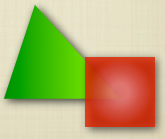
Step 2: primitive assembly



Step 3: rasterization



Step 4: fragment operations



9

9

OpenGL 1.x

10

10

OpenGL 1.x

- OpenGL 1.1 (1997)

10

10

OpenGL 1.x

- OpenGL 1.1 (1997)
  - Vertex Arrays

10

10

OpenGL 1.x

- OpenGL 1.1 (1997)
  - Vertex Arrays

10

OpenGL 1.x

- OpenGL 1.1 (1997)
  - Vertex Arrays
  - 2D Texture

10

OpenGL 1.x

- OpenGL 1.1 (1997)
  - Vertex Arrays
  - 2D Texture

10

OpenGL 1.x

- OpenGL 1.1 (1997)
  - Vertex Arrays
  - 2D Texture
- OpenGL 1.2 (March 1998)

10

OpenGL 1.x

- OpenGL 1.1 (1997)
  - Vertex Arrays
  - 2D Texture

- OpenGL 1.2 (March 1998)
  - 3D Texture

10

OpenGL 1.x

- OpenGL 1.1 (1997)
  - Vertex Arrays
  - 2D Texture

- OpenGL 1.2 (March 1998)
  - 3D Texture
  - Normal Vector Autoscale

10

OpenGL 1.x

- OpenGL 1.1 (1997)
  - Vertex Arrays
  - 2D Texture

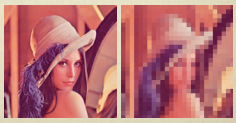
- OpenGL 1.2 (March 1998)
  - 3D Texture
  - Normal Vector Autoscale
  - Image Processing Functions

10

OpenGL 1.x

- OpenGL 1.1 (1997)
  - Vertex Arrays
  - 2D Texture

- OpenGL 1.2 (March 1998)
  - 3D Texture
  - Normal Vector Autoscale
  - Image Processing Functions



10

OpenGL 1.x

11

OpenGL 1.x

- OpenGL 1.3 (August 2001)
  - Compressed Texture

11

OpenGL 1.x

- OpenGL 1.3 (August 2001)
  - Compressed Texture
- OpenGL 1.4 (July 2002)
  - Fog

11

OpenGL 1.x

- OpenGL 1.3 (August 2001)
  - Compressed Texture
- OpenGL 1.4 (July 2002)
  - Fog

11

OpenGL 1.x

- OpenGL 1.3 (August 2001)
  - Compressed Texture
- OpenGL 1.4 (July 2002)
  - Fog
  - Buffer Objects

11

OpenGL 2.x, 3.x, 4.x

- Programmable Shaders
  - Vertex Shaders
  - Fragment Shaders
  - Geometry Shaders
- more GPU support

12

Using OpenGL Libraries

- Supporting Libraries
  - GLU (GL Utility)
    - Quadrics, Texture functions, ...
  - GLUT (GL Utility Toolkit)
    - Window Management functions & Event Handling
      - Display callback
      - I/O callbacks (mouse, keyboard, ...)
      - Idle callback, resize callback, ...

13

OpenGL Primitives

14

OpenGL Primitives

- Each primitive is specified using glBegin(___) and glEnd()

14

OpenGL Primitives

- Each primitive is specified using glBegin(___) and glEnd()
- Types of Primitives

14

OpenGL Primitives

- Each primitive is specified using glBegin(___) and glEnd()
- Types of Primitives
  - GL_POINTS

14

OpenGL Primitives

- Each primitive is specified using glBegin(___) and glEnd()
- Types of Primitives
  - GL_POINTS
  - GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP

14

OpenGL Primitives

- Each primitive is specified using glBegin(___) and glEnd()
- Types of Primitives
  - GL_POINTS
  - GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP
  - GL_TRIANGLES, GL_TRIANGLE_FAN, GL_TRIANGLE_STRIP

14

OpenGL Primitives

- Each primitive is specified using glBegin(___) and glEnd()
- Types of Primitives
  - GL_POINTS
  - GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP
  - GL_TRIANGLES, GL_TRIANGLE_FAN, GL_TRIANGLE_STRIP
  - GL_QUADS, GL_QUAD_STRIP

14

OpenGL Line Primitives

15

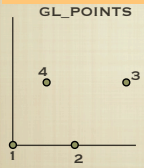
OpenGL Line Primitives

```
glBegin(GL_POINTS)
   glColor3f (1, 0, 0); // red
   glVertex2f (0, 0);
   glVertex2f (1, 0);
   glVertex2f (2,1);
   glVertex2f (1, 0.5);
glEnd();
```


15

OpenGL Line Primitives

```
glBegin(GL_POINTS)
   glColor3f (1, 0, 0); // red
   glVertex2f (0, 0);
   glVertex2f (1, 0);
   glVertex2f (2,1);
   glVertex2f (1, 0.5);
glEnd();
```

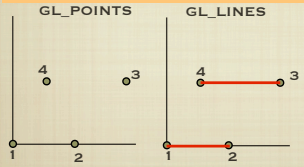


OpenGL Line Primitives

```
glBegin(GL_LINES)
   glColor3f (1, 0, 0); // red
   glVertex2f (0, 0);
   glVertex2f (1, 0);
   glVertex2f (2,1);
   glVertex2f (1, 0.5);
glEnd();
```

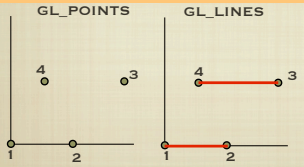
OpenGL Line Primitives

```
glBegin(GL_LINES)
   glColor3f (1, 0, 0); // red
   glVertex2f (0, 0);
   glVertex2f (1, 0);
   glVertex2f (2,1);
   glVertex2f (1, 0.5);
glEnd();
```



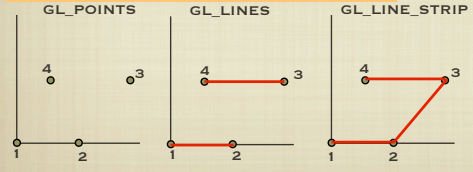
OpenGL Line Primitives

```
glBegin(GL_LINE_STRIP)
   glColor3f (1, 0, 0); // red
   glVertex2f (0, 0);
   glVertex2f (1, 0);
   glVertex2f (2,1);
   glVertex2f (0.5, 1);
glEnd();
```



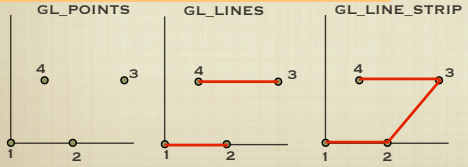
OpenGL Line Primitives

```
glBegin(GL_LINE_STRIP)
   glColor3f (1, 0, 0); // red
   glVertex2f (0, 0);
   glVertex2f (1, 0);
   glVertex2f (2,1);
   glVertex2f (0.5, 1);
glEnd();
```



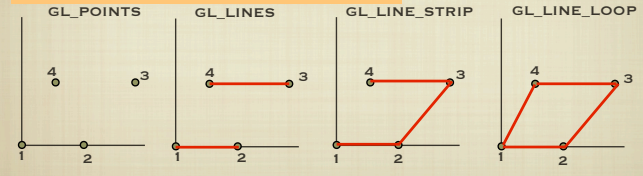
OpenGL Line Primitives

```
glBegin(GL_LINE_STRIP)
   glColor3f (1, 0, 0); // red
   glVertex2f (0, 0);
   glVertex2f (1, 0);
   glVertex2f (2,1);
   glVertex2f (0.5, 1);
glEnd();
```



OpenGL Line Primitives

```
glBegin(GL_LINE_STRIP)
   glColor3f (1, 0, 0); // red
   glVertex2f (0, 0);
   glVertex2f (1, 0);
   glVertex2f (2,1);
   glVertex2f (0.5, 1);
glEnd();
```



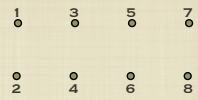
OpenGL Triangle Primitives

OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLES)
   glVertex2f (0, 1); // 1
   glVertex2f (0, 0); // 2
   glVertex2f (1, 1); // 3
   glVertex2f (1, 0); // 4
   glVertex2f (2, 1); // 5
   glVertex2f (2, 0); // 6
   glVertex2f (3, 1); // 7
   glVertex2f (3, 0); // 8
glEnd();
```


OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLES)
   glVertex2f (0, 1); // 1
   glVertex2f (0, 0); // 2
   glVertex2f (1, 1); // 3
   glVertex2f (1, 0); // 4
   glVertex2f (2, 1); // 5
   glVertex2f (2, 0); // 6
   glVertex2f (3, 1); // 7
   glVertex2f (3, 0); // 8
glEnd();
```

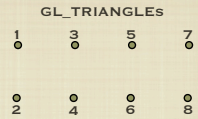


OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLES)
   glVertex2f (0, 1); // 1
   glVertex2f (0, 0); // 2
   glVertex2f (1, 1); // 3
   glVertex2f (1, 0); // 4
   glVertex2f (2, 1); // 5
   glVertex2f (2, 0); // 6
   glVertex2f (3, 1); // 7
   glVertex2f (3, 0); // 8
glEnd();
```

OpenGL Triangle Primitives

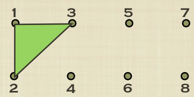
```
glBegin(GL_TRIANGLES)
   glVertex2f (0, 1); // 1
   glVertex2f (0, 0); // 2
   glVertex2f (1, 1); // 3
   glVertex2f (1, 0); // 4
   glVertex2f (2, 1); // 5
   glVertex2f (2, 0); // 6
   glVertex2f (3, 1); // 7
   glVertex2f (3, 0); // 8
glEnd();
```



OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLES)
    glVertex2f(0, 1); // 1
    glVertex2f(0, 0); // 2
    glVertex2f(1, 1); // 3
    glVertex2f(1, 0); // 4
    glVertex2f(2, 1); // 5
    glVertex2f(2, 0); // 6
    glVertex2f(3, 1); // 7
    glVertex2f(3, 0); // 8
glEnd();
```

GL_TRIANGLES

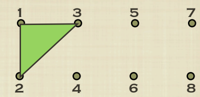


FRONT FACING

OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLES)
    glVertex2f(0, 1); // 1
    glVertex2f(0, 0); // 2
    glVertex2f(1, 1); // 3
    glVertex2f(1, 0); // 4
    glVertex2f(2, 1); // 5
    glVertex2f(2, 0); // 6
    glVertex2f(3, 1); // 7
    glVertex2f(3, 0); // 8
glEnd();
```

GL_TRIANGLES

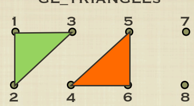


FRONT FACING

OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLES)
    glVertex2f(0, 1); // 1
    glVertex2f(0, 0); // 2
    glVertex2f(1, 1); // 3
    glVertex2f(1, 0); // 4
    glVertex2f(2, 1); // 5
    glVertex2f(2, 0); // 6
    glVertex2f(3, 1); // 7
    glVertex2f(3, 0); // 8
glEnd();
```

GL_TRIANGLES



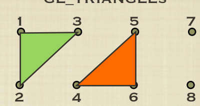
FRONT FACING

BACK FACING

OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLES)
    glVertex2f(0, 1); // 1
    glVertex2f(0, 0); // 2
    glVertex2f(1, 1); // 3
    glVertex2f(1, 0); // 4
    glVertex2f(2, 1); // 5
    glVertex2f(2, 0); // 6
    glVertex2f(3, 1); // 7
    glVertex2f(3, 0); // 8
glEnd();
```

GL_TRIANGLES



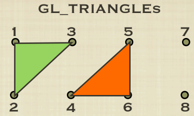
FRONT FACING

BACK FACING

OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLES)
    glVertex2f(0, 1); // 1
    glVertex2f(0, 0); // 2
    glVertex2f(1, 1); // 3
    glVertex2f(1, 0); // 4
    glVertex2f(2, 1); // 5
    glVertex2f(2, 0); // 6
    glVertex2f(3, 1); // 7
    glVertex2f(3, 0); // 8
glEnd();
```

GL_TRIANGLES



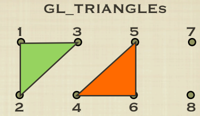
FRONT FACING

BACK FACING

OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_STRIP)
    glVertex2f(0, 1); // 1
    glVertex2f(0, 0); // 2
    glVertex2f(1, 1); // 3
    glVertex2f(1, 0); // 4
    glVertex2f(2, 1); // 5
    glVertex2f(2, 0); // 6
    glVertex2f(3, 1); // 7
    glVertex2f(3, 0); // 8
glEnd();
```

GL_TRIANGLES



FRONT FACING

BACK FACING

OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_STRIP)
   glVertex2f (0, 1); // 1
   glVertex2f (0, 0); // 2
   glVertex2f (1, 1); // 3
   glVertex2f (1, 0); // 4
   glVertex2f (2, 1); // 5
   glVertex2f (2, 0); // 6
   glVertex2f (3, 1); // 7
   glVertex2f (3, 0); // 8
glEnd();
```

GL_TRIANGLES

FRONT FACING

BACK FACING

OpenGL Triangle Primitives

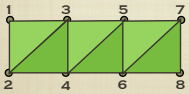
```
glBegin(GL_TRIANGLE_STRIP)
   glVertex2f (0, 1); // 1
   glVertex2f (0, 0); // 2
   glVertex2f (1, 1); // 3
   glVertex2f (1, 0); // 4
   glVertex2f (2, 1); // 5
   glVertex2f (2, 0); // 6
   glVertex2f (3, 1); // 7
   glVertex2f (3, 0); // 8
glEnd();
```

GL_TRIANGLES

FRONT FACING

BACK FACING

GL_TRIANGLE_STRIP



OpenGL Triangle Primitives

OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_FAN)
   glVertex2f (0,  0);  // 1

   glVertex2f (0, -4);  // 2
   glVertex2f (2, -3);  // 3
   glVertex2f (4,  0);  // 4
   glVertex2f (3,  2);  // 5
   glVertex2f (1,  4);  // 6
glEnd();
```

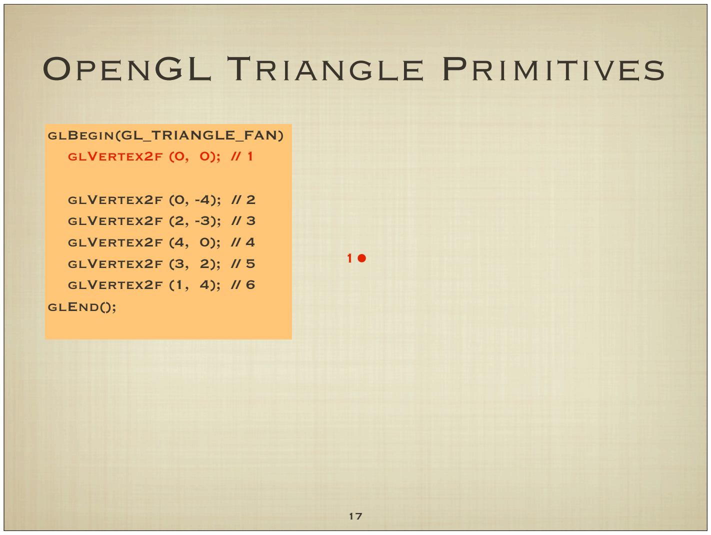

OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_FAN)
   glVertex2f (0,  0);  // 1

   glVertex2f (0, -4);  // 2
   glVertex2f (2, -3);  // 3
   glVertex2f (4,  0);  // 4
   glVertex2f (3,  2);  // 5
   glVertex2f (1,  4);  // 6
glEnd();
```

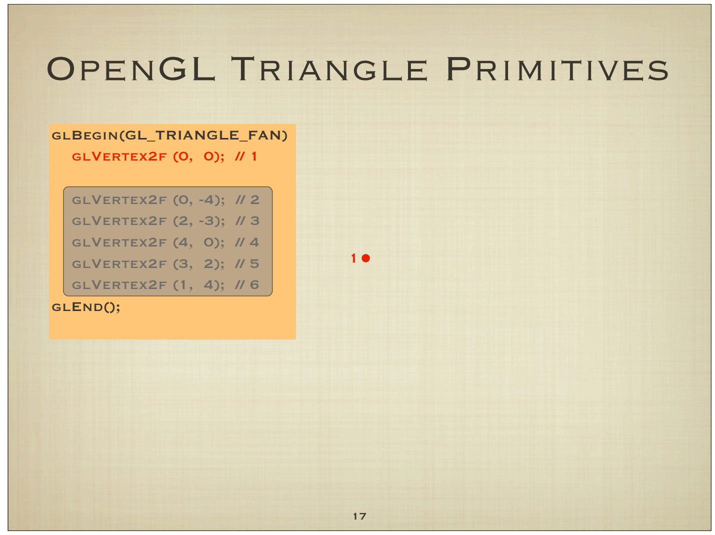

OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_FAN)
   glVertex2f (0,  0);  // 1

   glVertex2f (0, -4);  // 2
   glVertex2f (2, -3);  // 3
   glVertex2f (4,  0);  // 4
   glVertex2f (3,  2);  // 5
   glVertex2f (1,  4);  // 6
glEnd();
```

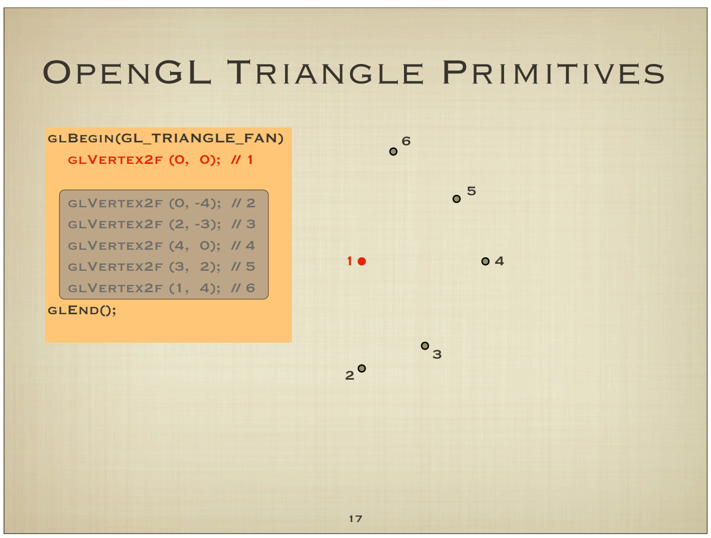
1



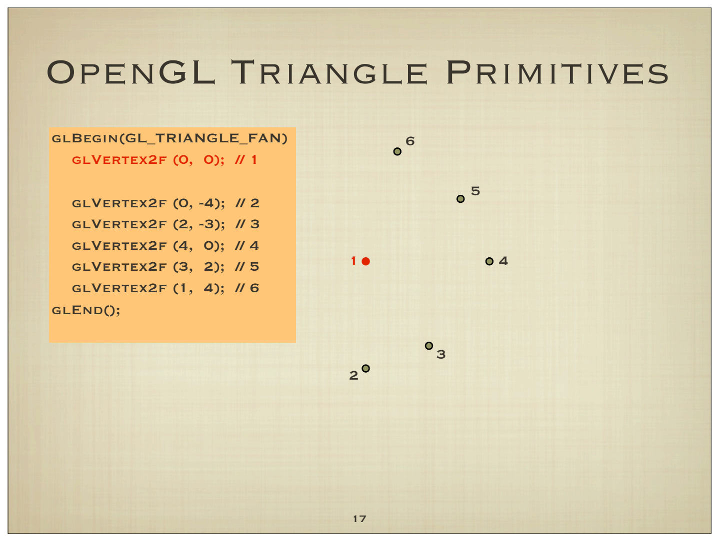
17



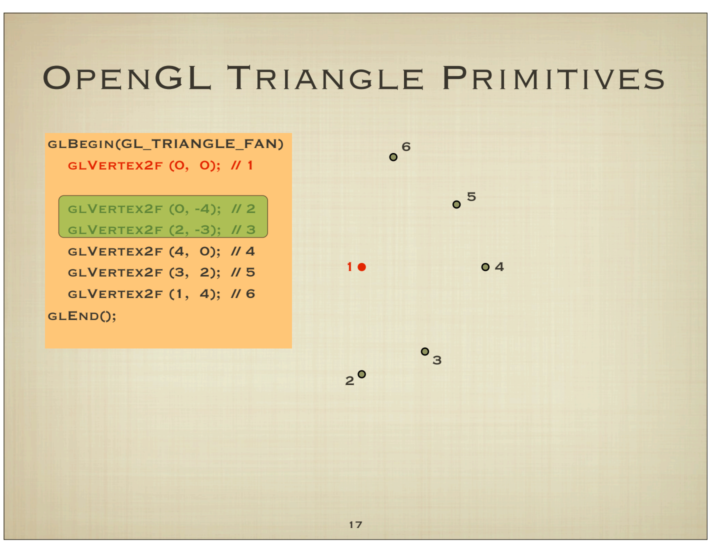
17



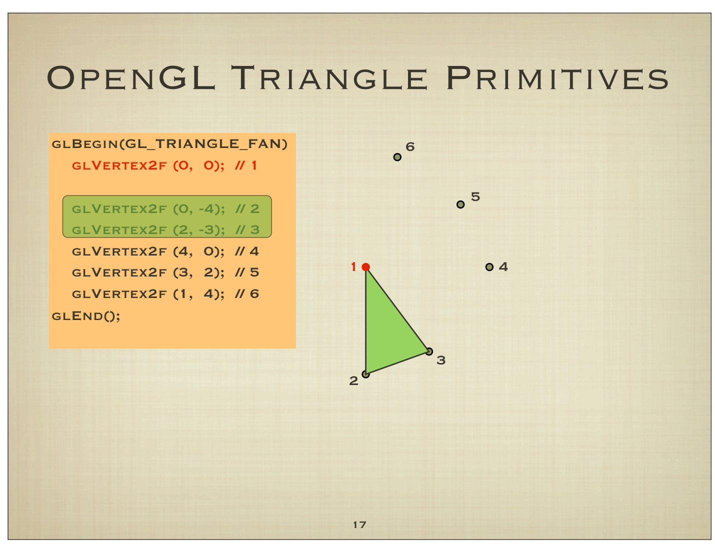
17



17



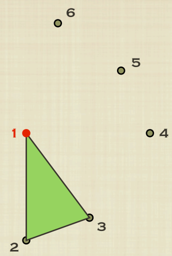
17



17

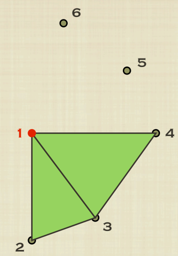
OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_FAN)
    glVertex2f (0, 0); // 1

    glVertex2f (0, -4); // 2
    glVertex2f (2, -3); // 3
    glVertex2f (4, 0); // 4
    glVertex2f (3, 2); // 5
    glVertex2f (1, 4); // 6
glEnd();
```



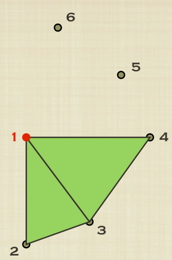
OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_FAN)
    glVertex2f (0, 0); // 1

    glVertex2f (0, -4); // 2
    glVertex2f (2, -3); // 3
    glVertex2f (4, 0); // 4
    glVertex2f (3, 2); // 5
    glVertex2f (1, 4); // 6
glEnd();
```



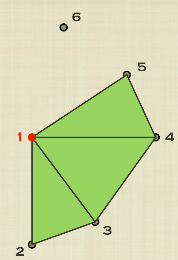
OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_FAN)
    glVertex2f (0, 0); // 1

    glVertex2f (0, -4); // 2
    glVertex2f (2, -3); // 3
    glVertex2f (4, 0); // 4
    glVertex2f (3, 2); // 5
    glVertex2f (1, 4); // 6
glEnd();
```



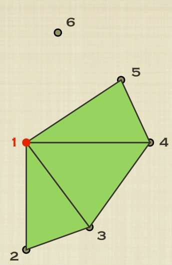
OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_FAN)
    glVertex2f (0, 0); // 1

    glVertex2f (0, -4); // 2
    glVertex2f (2, -3); // 3
    glVertex2f (4, 0); // 4
    glVertex2f (3, 2); // 5
    glVertex2f (1, 4); // 6
glEnd();
```



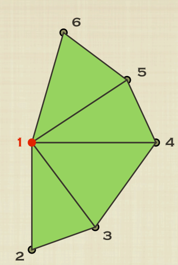
OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_FAN)
    glVertex2f (0, 0); // 1

    glVertex2f (0, -4); // 2
    glVertex2f (2, -3); // 3
    glVertex2f (4, 0); // 4
    glVertex2f (3, 2); // 5
    glVertex2f (1, 4); // 6
glEnd();
```



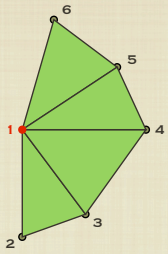
OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_FAN)
    glVertex2f (0, 0); // 1

    glVertex2f (0, -4); // 2
    glVertex2f (2, -3); // 3
    glVertex2f (4, 0); // 4
    glVertex2f (3, 2); // 5
    glVertex2f (1, 4); // 6
glEnd();
```



OpenGL Triangle Primitives

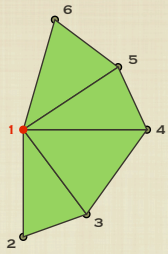
```
glBegin(GL_TRIANGLE_FAN)
   glVertex2f (0,  0);  // 1

   glVertex2f (0, -4);  // 2
   glVertex2f (2, -3);  // 3
   glVertex2f (4,  0);  // 4
   glVertex2f (3,  2);  // 5
   glVertex2f (1,  4);  // 6
glEnd();
```



17

OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_FAN)
   glVertex2f (0,  0);  // 1

   glVertex2f (0, -4);  // 2
   glVertex2f (2, -3);  // 3
   glVertex2f (4,  0);  // 4
   glVertex2f (3,  2);  // 5
   glVertex2f (1,  4);  // 6
glEnd();
```

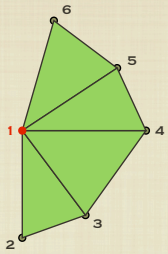


17

17

OpenGL Triangle Primitives

```
glBegin(GL_TRIANGLE_FAN)
   glVertex2f (0,  0);  // 1

   glVertex2f (0, -4);  // 2
   glVertex2f (2, -3);  // 3
   glVertex2f (4,  0);  // 4
   glVertex2f (3,  2);  // 5
   glVertex2f (1,  4);  // 6
glEnd();
```



17

17

Vertex Arrays

18

Vertex Arrays

- Without Vertex Arrays

18

Vertex Arrays

- Without Vertex Arrays

- glBegin(GL_LINE_STRIP);
  glVertex (10, 5, 6);
  glVertex (2, 1, 4);
  glVertex (…);
  …. more lines ….
  glEnd();

  // (N + 2) calls

18

Vertex Arrays

- Without Vertex Arrays

```
glBegin(GL_LINE_STRIP);
glVertex (10, 5, 6);
glVertex (2, 1, 4);
glVertex (…);
…. more lines ….
glEnd();
```

// (N + 2) calls

- With vertex arrays

18

Vertex Arrays

- Without Vertex Arrays

```
glBegin(GL_LINE_STRIP);
glVertex (10, 5, 6);
glVertex (2, 1, 4);
glVertex (…);
…. more lines ….
glEnd();
```

// (N + 2) calls

- With vertex arrays

```
float[] PTS =
{10, 5, 6, 2, 1, 4, …..};
glVertexPointer (3,
GL_FLOAT, 0, PTS);
glDrawArrays
(GL_LINE_STRIP, 0,
N_POINTS);
```

// only 2 calls

18

Photographer Paradigm

19

Photographer Paradigm

- Select a Camera

19

Photographer Paradigm

- Select a Camera
  - Pixel Resolutions

19

Photographer Paradigm

- Select a Camera
  - Pixel Resolutions
  - Type of Lens

19

Photographer Paradigm

- Select a Camera
  - Pixel Resolutions
  - Type of Lens
  - OpenGL PROJECTION_MATRIX

19

Photographer Paradigm

- Select a Camera
  - Pixel Resolutions
  - Type of Lens
  - OpenGL PROJECTION_MATRIX

- Arrange Objects (in 3D world) & Place the camera in front of the objects

19

Photographer Paradigm

- Select a Camera
  - Pixel Resolutions
  - Type of Lens
  - OpenGL PROJECTION_MATRIX

- Arrange Objects (in 3D world) & Place the camera in front of the objects
  - OpenGL MODEL_VIEW_MATRIX

19

Transformation Matrices

20

Transformation Matrices

- Homogeneous Coordinates

20

Transformation Matrices

- Homogeneous Coordinates
  - As an "extension" to the 3D Cartesian Coordinates: promotion to 4D

20

Transformation Matrices

- Homogeneous Coordinates
  - As an "extension" to the 3D Cartesian Coordinates: promotion to 4D
    - (5,2,-6) ==> (5,2,-6,1)

20

Transformation Matrices

- Homogeneous Coordinates
  - As an "extension" to the 3D Cartesian Coordinates: promotion to 4D
    - (5,2,-6) ==> (5,2,-6,1)
    - (5,2,-6) ==> (20,8,-24, 4)

20

Transformation Matrices

- Homogeneous Coordinates
  - As an "extension" to the 3D Cartesian Coordinates: promotion to 4D
    - (5,2,-6) ==> (5,2,-6,1)
    - (5,2,-6) ==> (20,8,-24, 4)
- [Linear] Transformation Matrices

20

Transformation Matrices

- Homogeneous Coordinates
  - As an "extension" to the 3D Cartesian Coordinates: promotion to 4D
    - (5,2,-6) ==> (5,2,-6,1)
    - (5,2,-6) ==> (20,8,-24, 4)
- [Linear] Transformation Matrices
  - 3x3 matrices In Cartesian coordinate frames

20

Transformation Matrices

- Homogeneous Coordinates
  - As an "extension" to the 3D Cartesian Coordinates: promotion to 4D
    - (5,2,-6) ==> (5,2,-6,1)
    - (5,2,-6) ==> (20,8,-24, 4)
- [Linear] Transformation Matrices
  - 3x3 matrices In Cartesian coordinate frames
  - 4x4 matrices in Homogeneous coordinate frames

20

Linear Transformations

21

Linear Transformations

- 4x4 Matrices

Linear Transformations

- 4x4 Matrices
- Identity Matrix (“No Transformation”)

Linear Transformations

- 4x4 Matrices
- Identity Matrix (“No Transformation”)
- Scaling

Linear Transformations

- 4x4 Matrices
- Identity Matrix (“No Transformation”)
- Scaling
- Translation

Linear Transformations

- 4x4 Matrices
- Identity Matrix (“No Transformation”)
- Scaling
- Translation
- Rotation

Composite Transformations & Matrix Multiplications

- OpenGL Commands

```
glTranslate*(4, 5, 2);
glRotate*(30, 0, 1, 0);
glTranslate*(0, 0, 3);
```

Texture Mapping

Each vertex is assoc.'ed with two coordinates:
(3.0, 1.6, 0.4): 3D position of the vertex
(0.1, 0.8): 2D position of the texel to copy from