CS 428: Fall 2010

Introduction to Computer Graphics

OpenGL primer

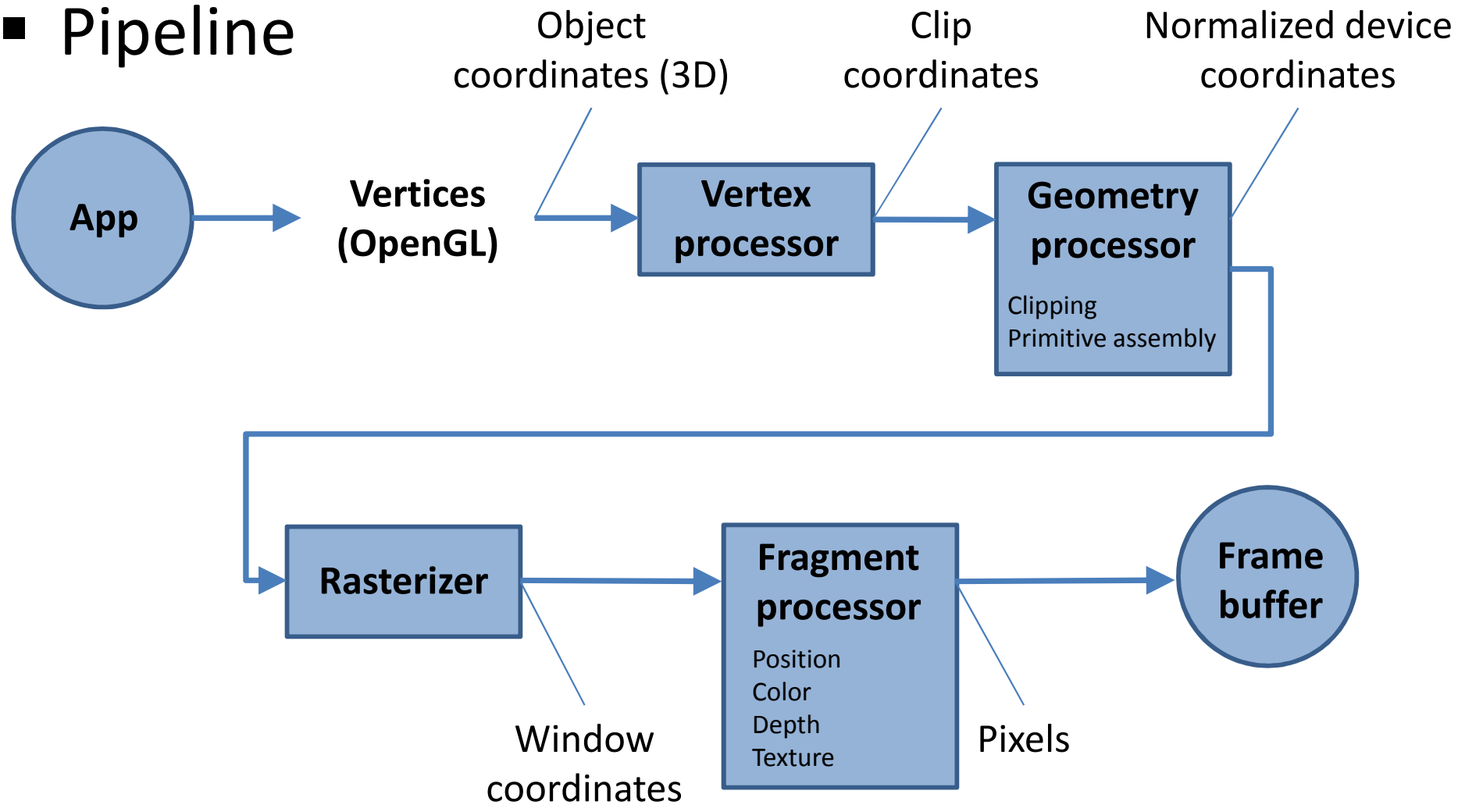
Graphics hardware

- Programmable
  {vertex, geometry, pixel} shaders
- Powerful

GeForce 285 GTX	GeForce 480 GTX	Intel Core2 Quad 3GHz
708 Gflops	1.45 TFlops	96 GFlops
166 GB/s	177 GB/s	21 GB/s
~200$ (board)	~450$ (board)	~330$ (chip)

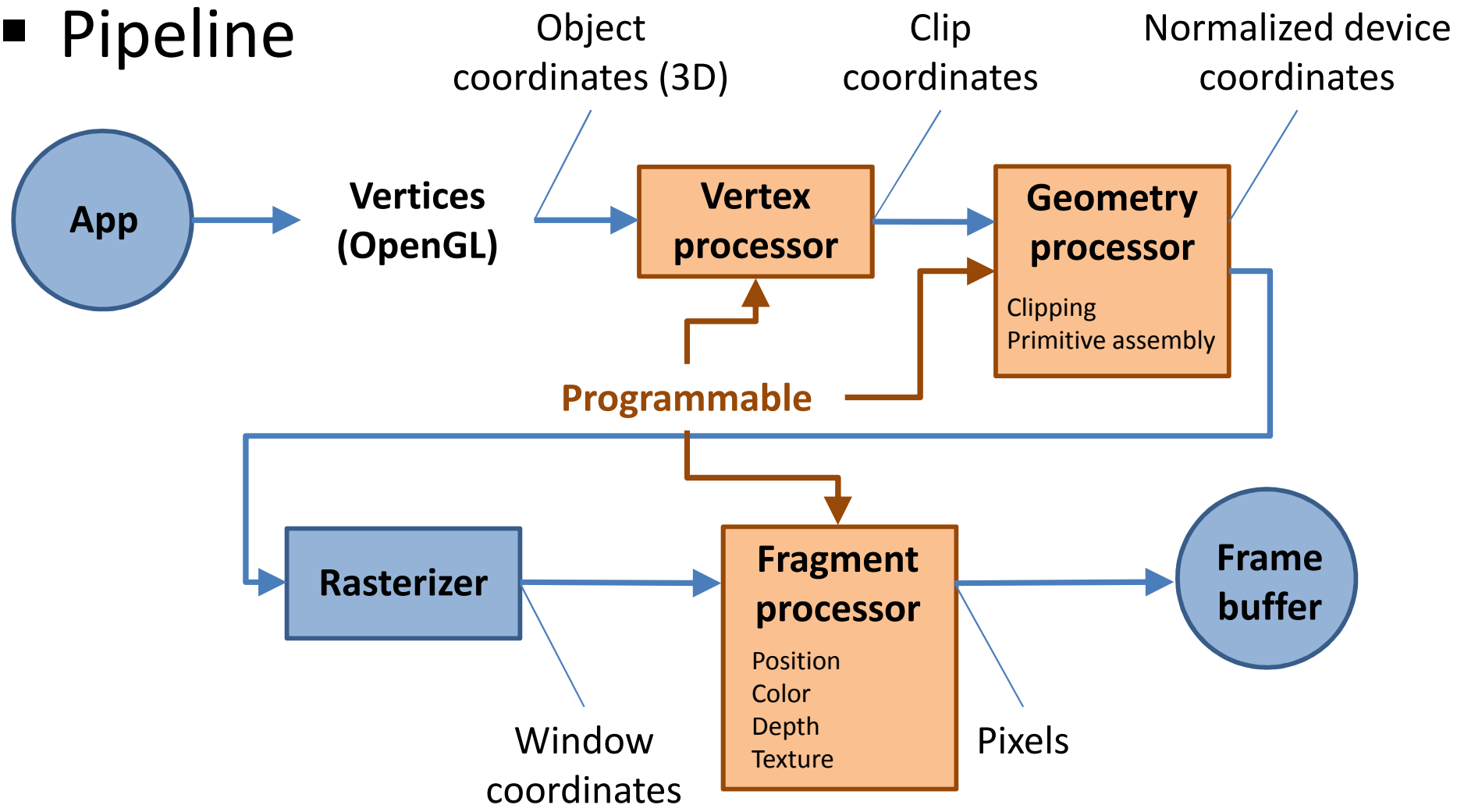
Hardware rendering

- Pipeline



Hardware rendering

- Pipeline

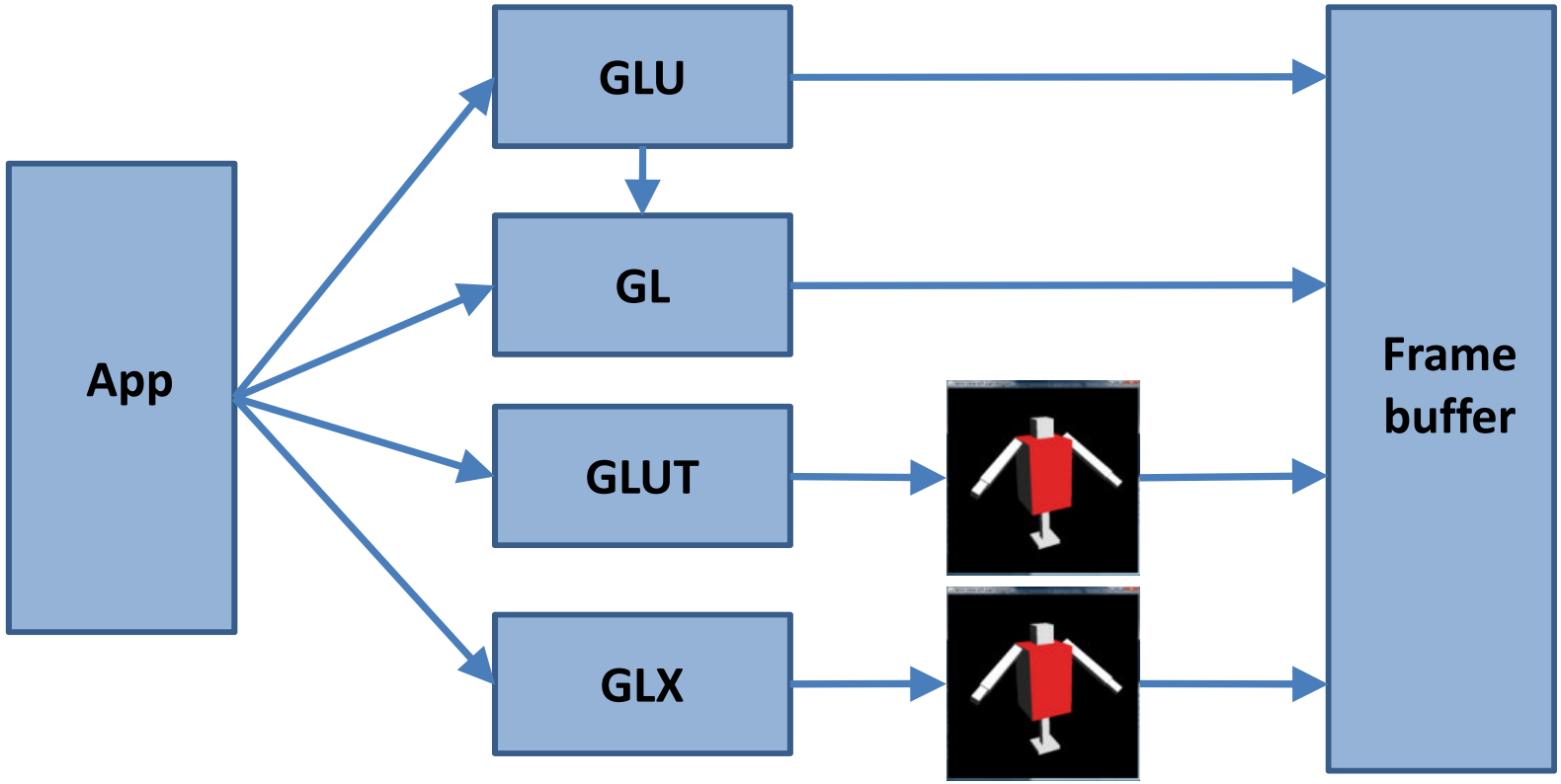


Hardware rendering

- Pipeline implemented in hardware
- Accessible via APIs: HW/device “independent”
- **GL** (1982, SGI) → **OpenGL** (1992)
  - Currently at version **4.1** (glsl version 4.10)
- **DirectX** (microsoft)
  - Currently at version **11**
- In **Java**
  - **Java** → **JOGL** → **OpenGL** (via JNI)

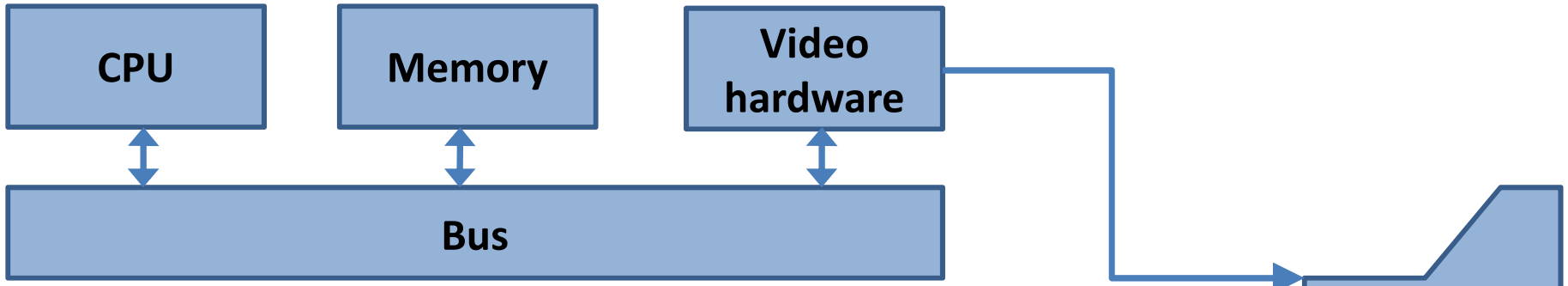
Hardware rendering

- Applications and APIs

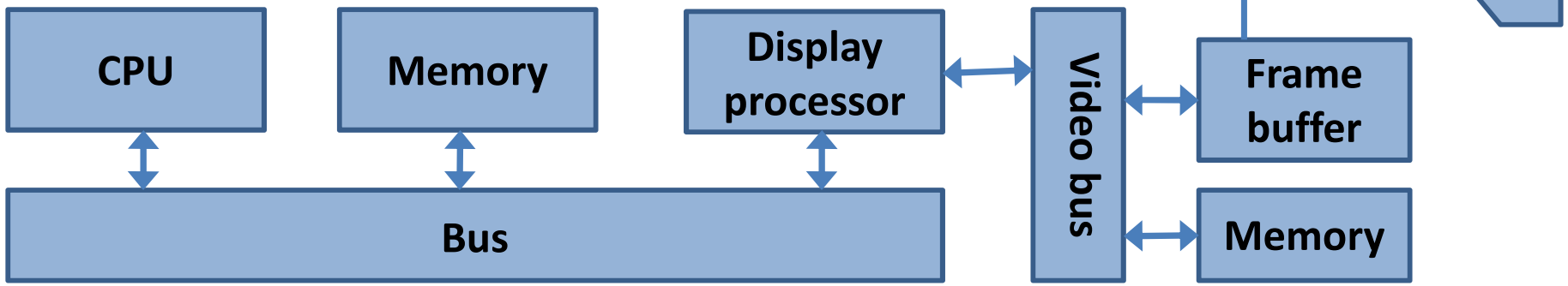


Display hardware

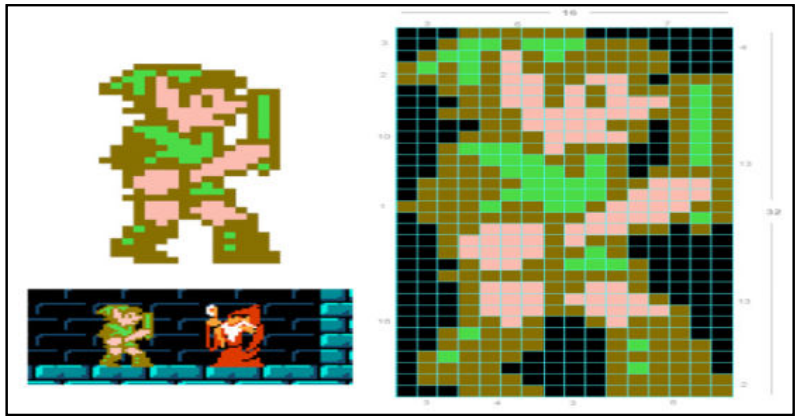
- Raster scan display

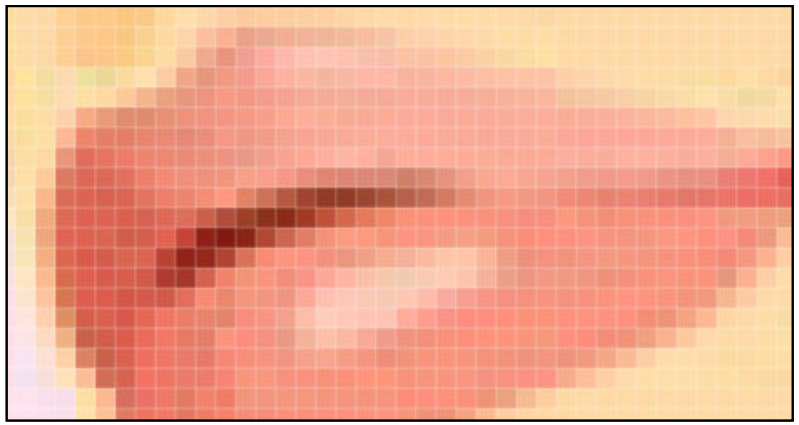


- Present architecture



Image

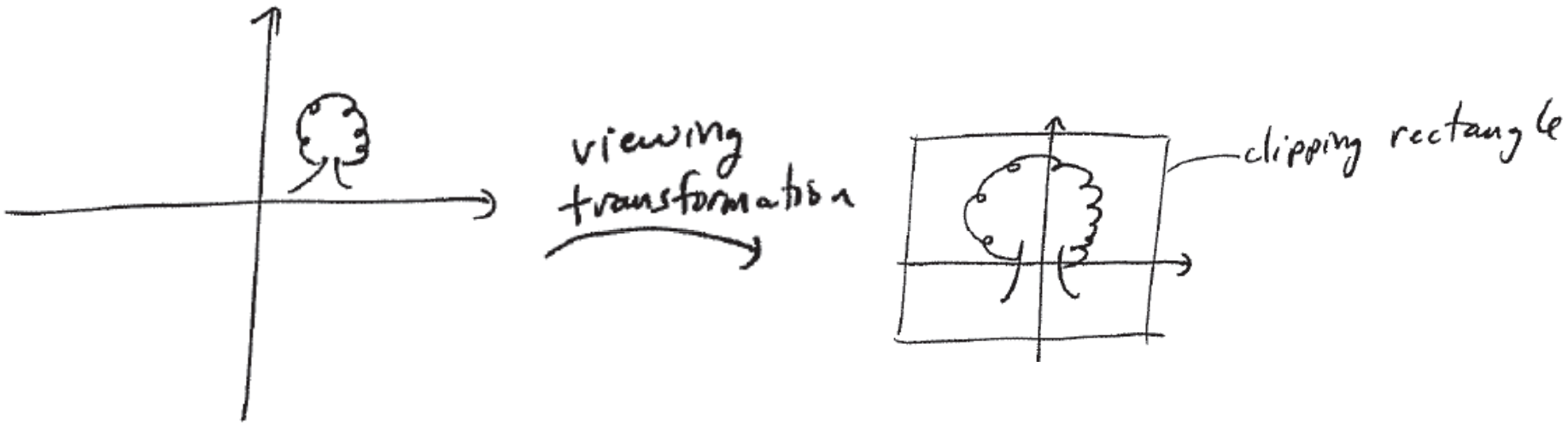




- Simply a grid of pixels
  - Each pixel stores R, G and B values
- A pixel is **not** a little square!
- Usually an array of 32-bit values
  - Sometimes 24-bit. Alignment makes access slower
  - Usually stores an alpha value as RGBA

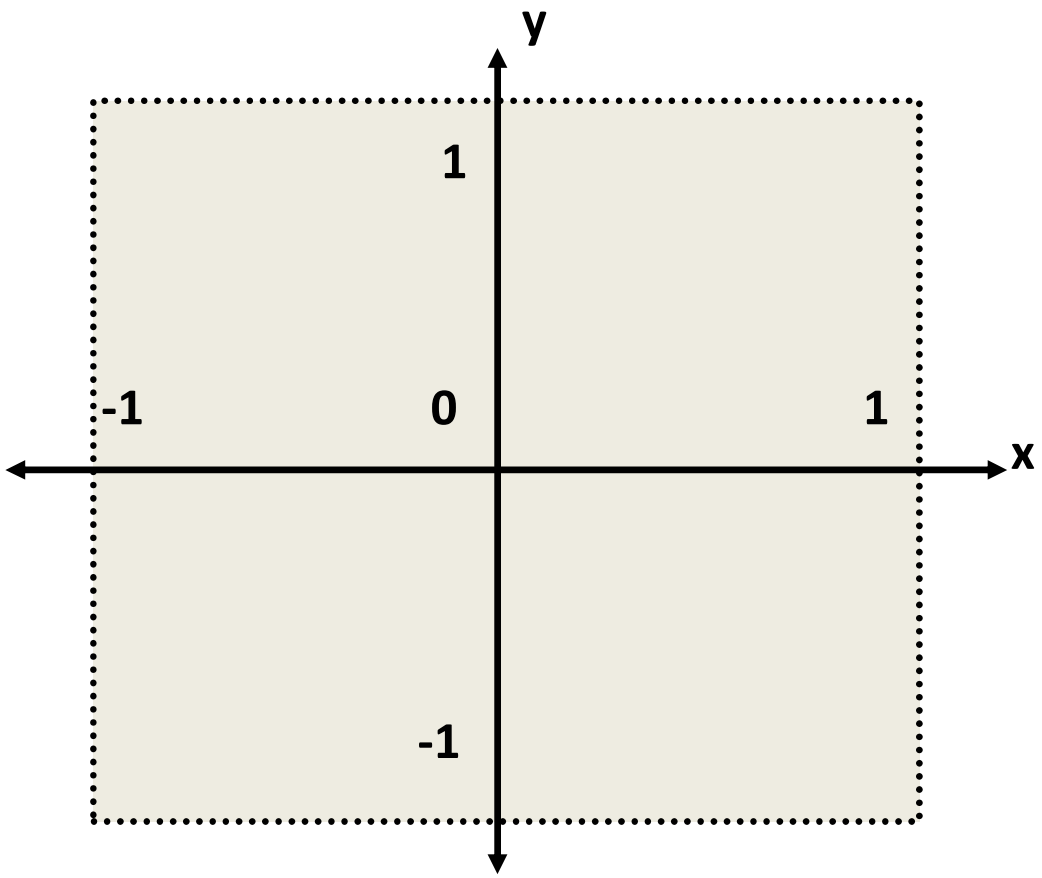
Windows

- Geometry (3D) seen through a 2D **window**



Window Coordinates

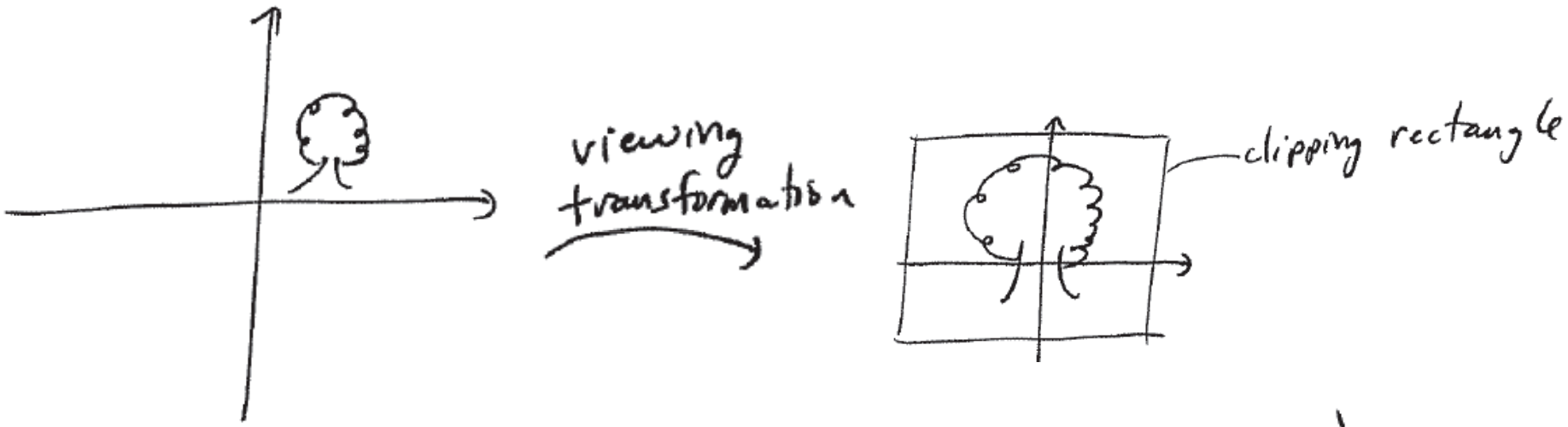
[-1,1] x [-1,1]



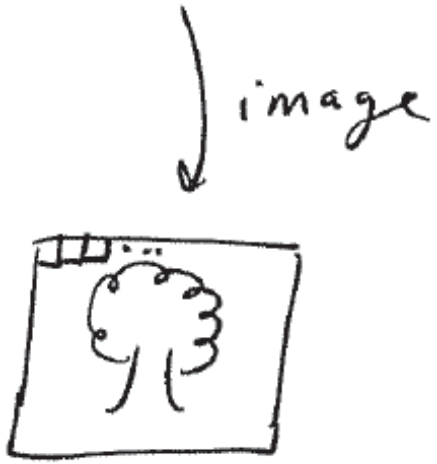

```
gl.glOrtho(-1,1,-1,1, …);
```

Windows

- Geometry (3D) seen through a 2D **window**



- Projection onto that 2D **window**
  - Results in pixel image
  - Pixels stored in 2D array of values
  - Low level `v = getp(x,y)` and `setp(x,y,v)`

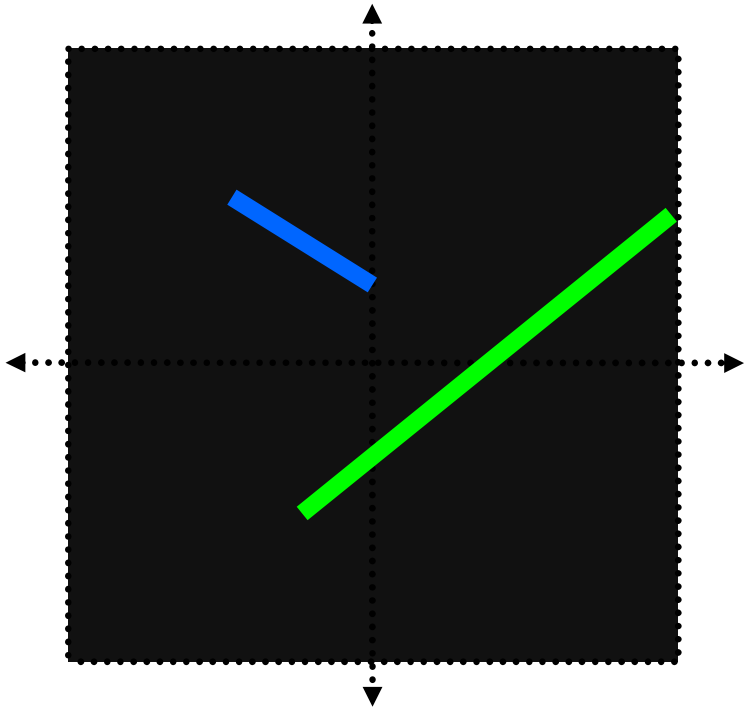


OpenGL API

- **Geometry**
  - Points
  - Primitives
    - Collection of points + type (point, line, polygon)

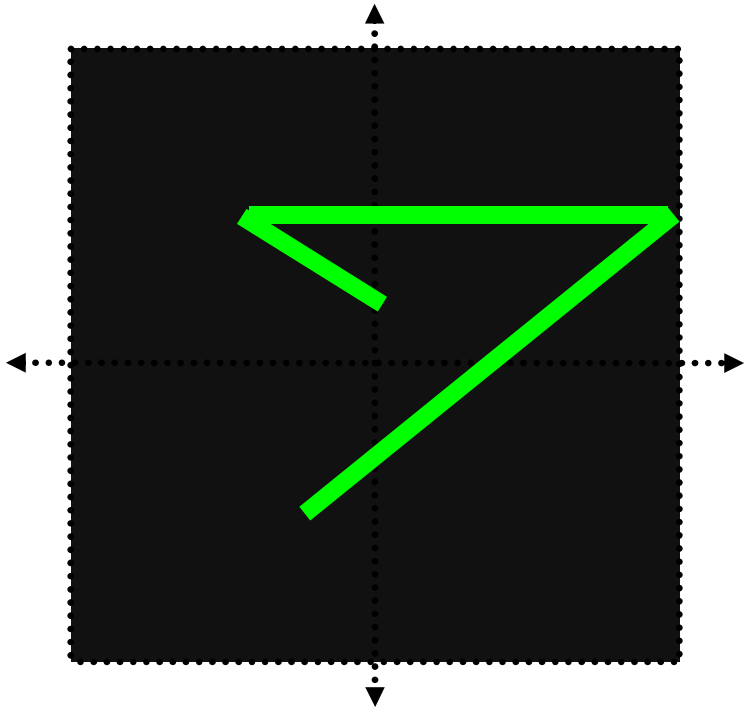
Lines

```
gl.glColor3d(0,1,0);

gl.glBegin(GL.GL_LINES);

gl.glVertex2d(-0.25, -0.5);
gl.glVertex2d(1, 0.5);

gl.glColor3d(0,0,1);
gl.glVertex2d(-0.5, 0.5);
gl.glVertex2d(0, 0.25);

gl.glEnd();
```



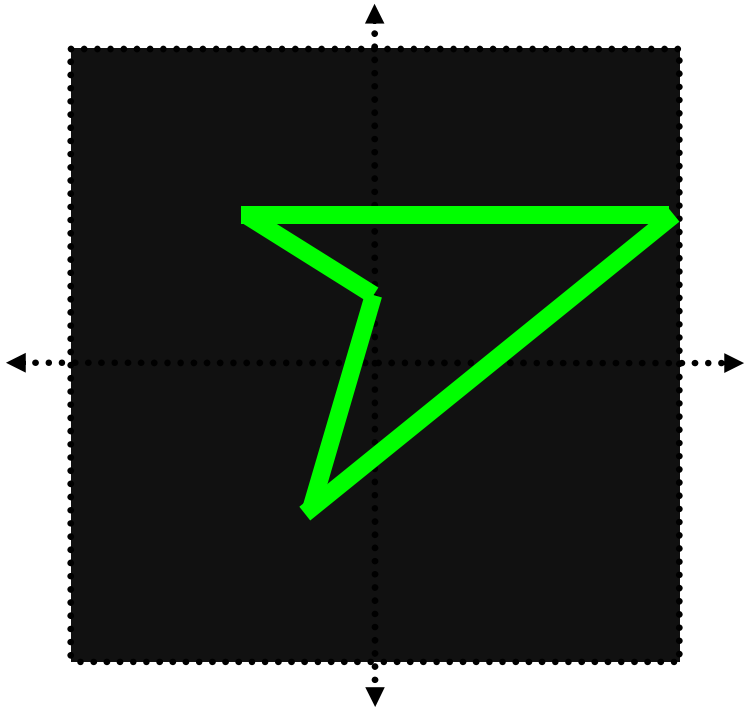
Strips

```
gl.glColor3d(0,1,0);

gl.glBegin(GL.GL_LINE_STRIP);

gl.glVertex2d(-0.25, -0.5);
gl.glVertex2d(1, 0.5);
gl.glVertex2d(-0.5, 0.5);
gl.glVertex2d(0, 0.25);

gl.glEnd();
```



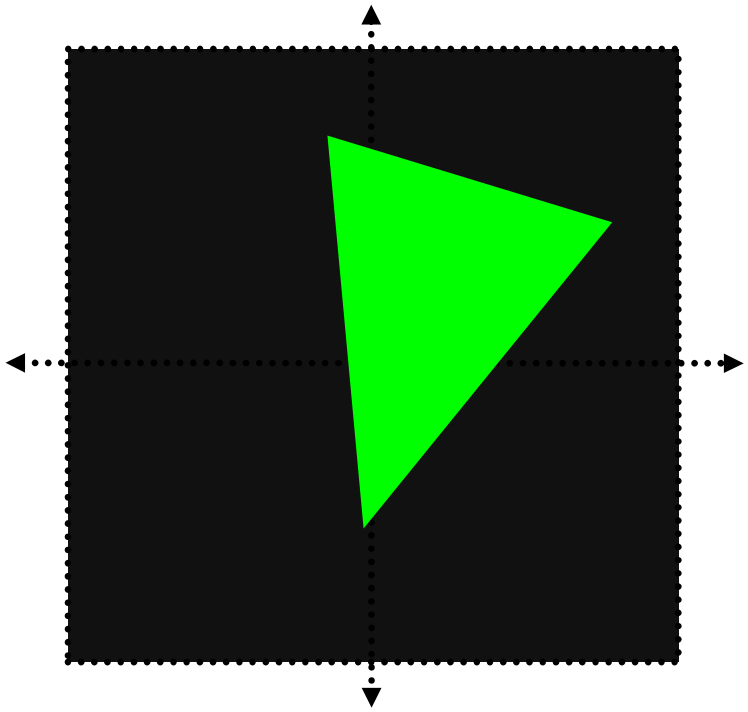
Loops

```
gl.glColor3d(0,1,0);

gl.glBegin(GL.GL_LINE_LOOP);

gl.glVertex2d(-0.25, -0.5);
gl.glVertex2d(1, 0.5);
gl.glVertex2d(-0.5, 0.5);
gl.glVertex2d(0, 0.25);

gl.glEnd();
```



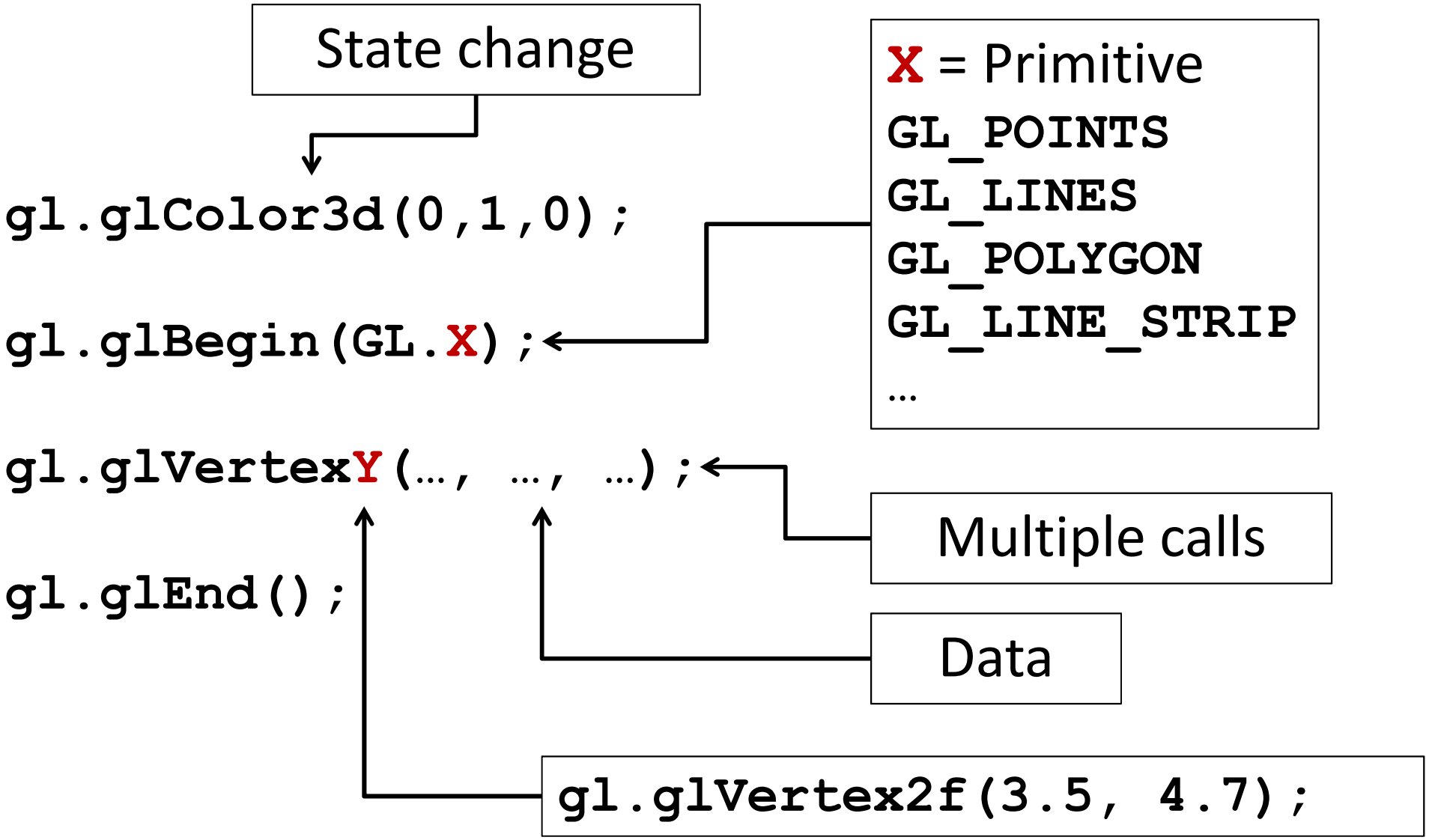
Polygons

```
gl.glColor3d(0,1,0);

gl.glBegin(GL.GL_POLYGON);

gl.glVertex2d(0, -0.5);
gl.glVertex2d(0.75, 0.5);
gl.glVertex2d(-0.25, 0.75);

gl.glEnd();
```



OpenGL primitives

Geometry specification

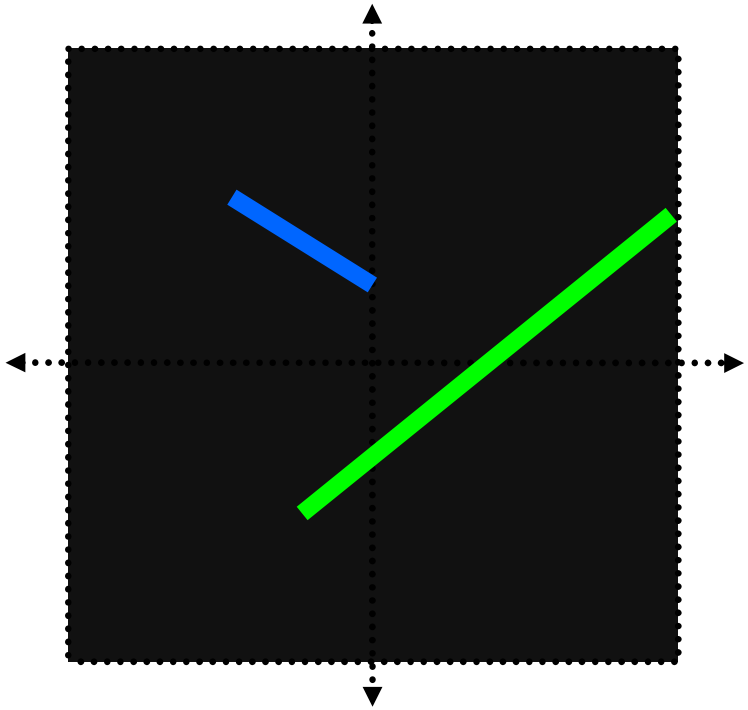


OpenGL API

- **Geometry**
  - Points
  - Primitives
    - Collection of points + type (point, line, polygon)
- **State (!)**
  - OpenGL runtime is a **huge** state machine
  - Output of geometry (on screen) controlled by state

Simple state change example

```
gl.glColor3d(0,1,0);

gl.glBegin(GL.GL_LINES);

gl.glVertex2d(-0.25, -0.5);
gl.glVertex2d(1, 0.5);

gl.glColor3d(0,0,1);
gl.glVertex2d(-0.5, 0.5);
gl.glVertex2d(0, 0.25);

gl.glEnd();
```



OpenGL State machine

- OpenGL state influences many things
  - Draw color
  - Point size
  - Line width
  - Lighting conditions (en- or disabled), light sources
  - Material properties
  - Current model-view transformation (matrix stack)
  - Current projective transformation (matrix stack)
  - …

Animation

- **Draw loop + double buffering**
  - **Init:** Hold on to two images/pixel buffers
  - **(1)** Draw into so-called back buffer while showing front buffer on screen
  - **(2)** When done drawing, or some fixed time has passed (e.g. $1/30^{th}$ of a second)
    - Swap buffers
  - **(3)** Repeat (go to step **(1)**)

Interaction

- Very similar to the previous principle
  - **Init:** Hold on to two images/pixel buffers
    Start with no user input
  - **(1a)** Draw scene into back buffer, take user input into account, while showing front buffer on screen
  - **(1b)** Gather user input (mouse, gamepad, etc.)
  - **(2)** When done drawing, or some fixed time has passed (e.g. 1/30th of a second)
    - Swap buffers
  - **(3)** Repeat (go to step **(1)**)

JOGL/OpenGL code + demo