Game Architecture

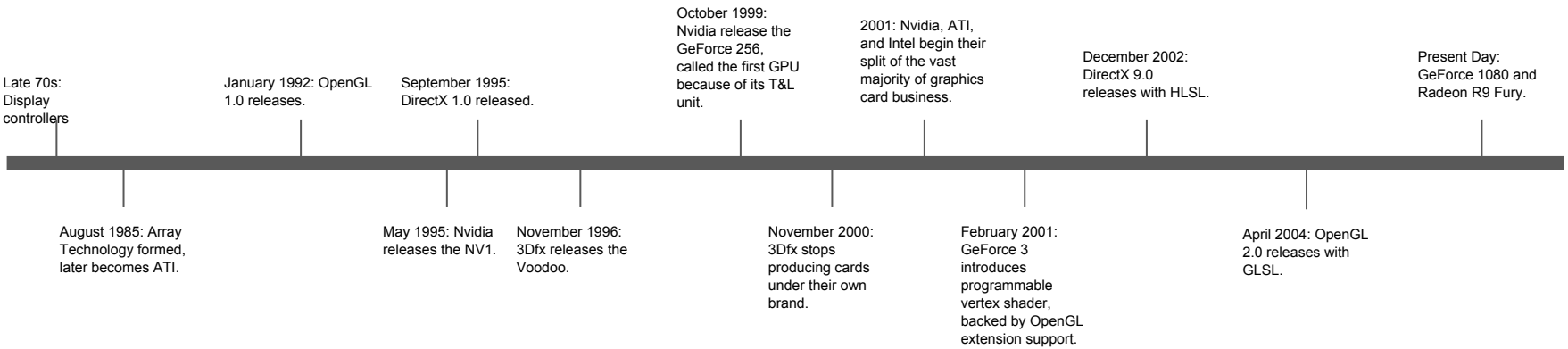
Graphics I

The Phantom Vertex

Today’s Agenda

- History
- Fixed Function Pipeline
- Programmable Pipeline - Part 1
- Scene Representation

History


Late 70s: Display controllers
August 1985: Array Technology formed, later becomes ATI.
January 1992: OpenGL 1.0 releases.
May 1995: Nvidia releases the NV1.
September 1995: DirectX 1.0 released.
November 1996: 3Dfx releases the Voodoo.
October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.
November 2000: 3Dfx stops producing cards under their own brand.
2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.
February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.
December 2002: DirectX 9.0 releases with HLSL.
April 2004: OpenGL 2.0 releases with GLSL.
Present Day: GeForce 1080 and Radeon R9 Fury.

History

Late 70s: Display controllers

August 1985: Array Technology formed, later becomes ATI.

January 1992: OpenGL 1.0 releases.

May 1995: Nvidia releases the NV1.

September 1995: DirectX 1.0 released.

November 1996: 3Dfx releases the Voodoo.

October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.

November 2000: 3Dfx stops producing cards under their own brand.

2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.

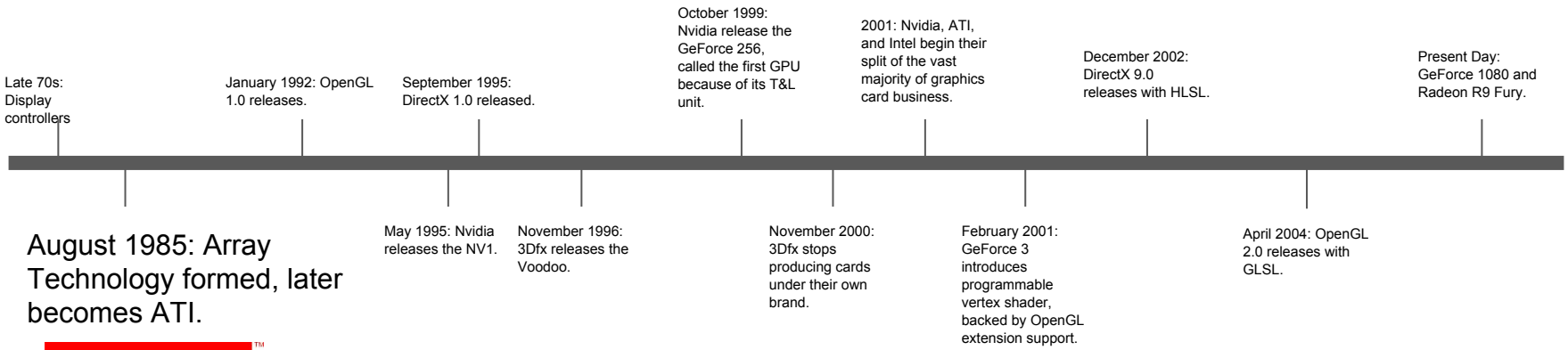
February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.

December 2002: DirectX 9.0 releases with HLSL.

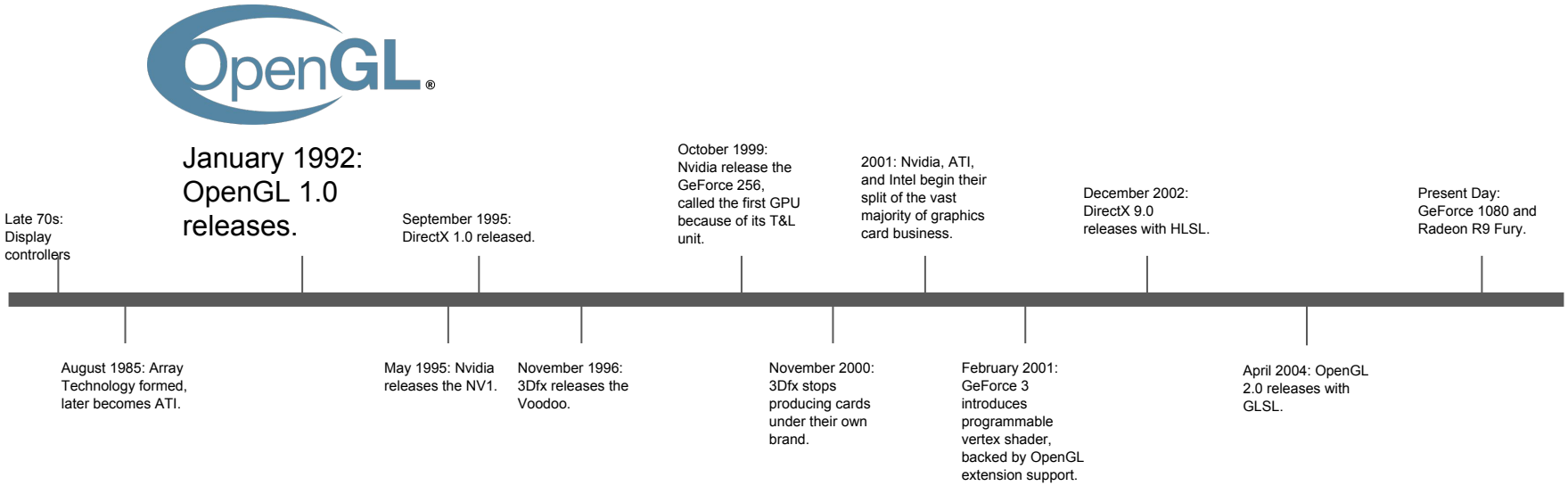
April 2004: OpenGL 2.0 releases with GLSL.

Present Day: GeForce 1080 and Radeon R9 Fury.

History


Late 70s: Display controllers
January 1992: OpenGL 1.0 releases.
September 1995: DirectX 1.0 released.
October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.
2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.
December 2002: DirectX 9.0 releases with HLSL.
Present Day: GeForce 1080 and Radeon R9 Fury.
August 1985: Array Technology formed, later becomes ATI.
May 1995: Nvidia releases the NV1.
November 1996: 3Dfx releases the Voodoo.
November 2000: 3Dfx stops producing cards under their own brand.
February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.
April 2004: OpenGL 2.0 releases with GLSL.

History


Late 70s: Display controllers
August 1985: Array Technology formed, later becomes ATI.
January 1992: OpenGL 1.0 releases.
May 1995: Nvidia releases the NV1.
September 1995: DirectX 1.0 released.
November 1996: 3Dfx releases the Voodoo.
October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.
November 2000: 3Dfx stops producing cards under their own brand.
2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.
February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.
December 2002: DirectX 9.0 releases with HLSL.
April 2004: OpenGL 2.0 releases with GLSL.
Present Day: GeForce 1080 and Radeon R9 Fury.

History

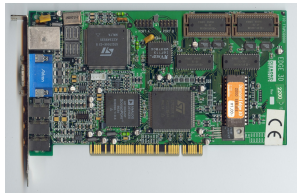
Late 70s: Display controllers

August 1985: Array Technology formed, later becomes ATI.

January 1992: OpenGL 1.0 releases.

September 1995: DirectX 1.0 released.

May 1995: Nvidia releases the NV1.



November 1996: 3Dfx releases the Voodoo.

October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.

November 2000: 3Dfx stops producing cards under their own brand.

2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.

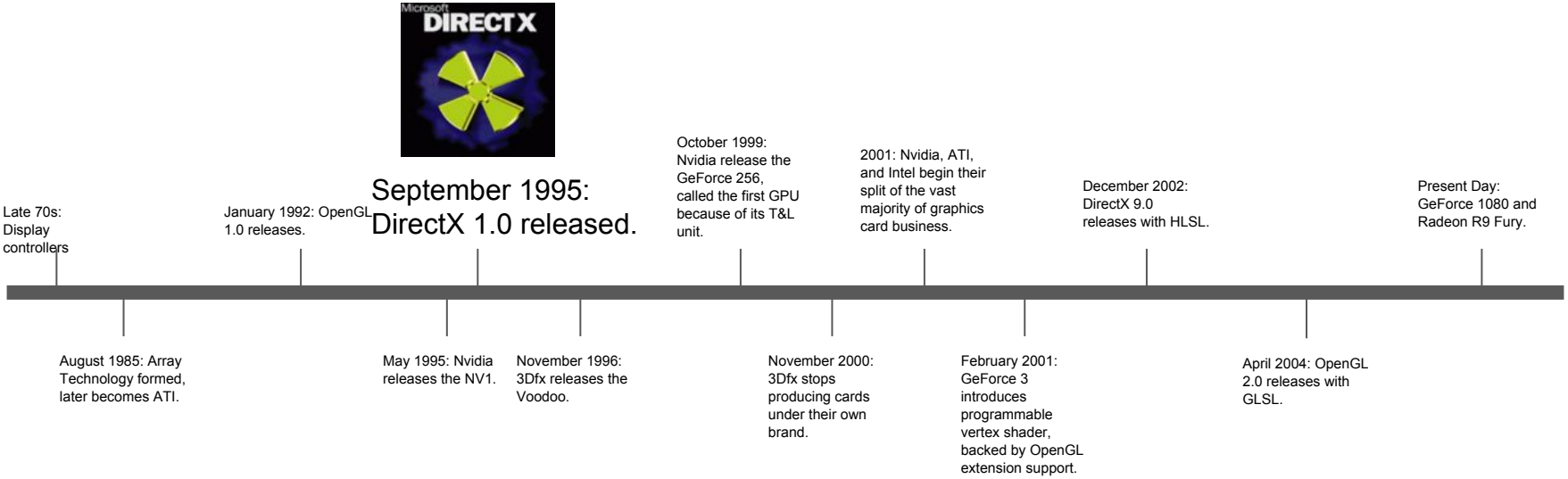
February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.

December 2002: DirectX 9.0 releases with HLSL.

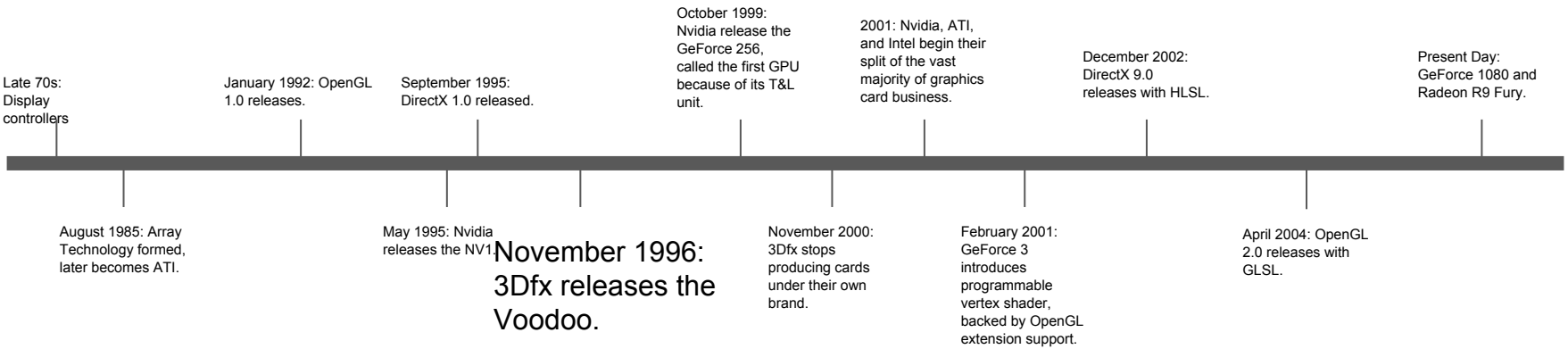
April 2004: OpenGL 2.0 releases with GLSL.

Present Day: GeForce 1080 and Radeon R9 Fury.

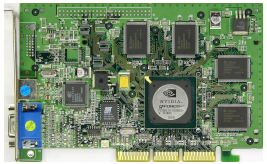
History


Late 70s: Display controllers
August 1985: Array Technology formed, later becomes ATI.
January 1992: OpenGL 1.0 releases.
May 1995: Nvidia releases the NV1.
September 1995: DirectX 1.0 released.
November 1996: 3Dfx releases the Voodoo.
October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.
November 2000: 3Dfx stops producing cards under their own brand.
2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.
February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.
December 2002: DirectX 9.0 releases with HLSL.
April 2004: OpenGL 2.0 releases with GLSL.
Present Day: GeForce 1080 and Radeon R9 Fury.

History


Late 70s: Display controllers
August 1985: Array Technology formed, later becomes ATI.
January 1992: OpenGL 1.0 releases.
September 1995: DirectX 1.0 released.
May 1995: Nvidia releases the NV1.
November 1996: 3Dfx releases the Voodoo.
October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.
November 2000: 3Dfx stops producing cards under their own brand.
2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.
February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.
December 2002: DirectX 9.0 releases with HLSL.
April 2004: OpenGL 2.0 releases with GLSL.
Present Day: GeForce 1080 and Radeon R9 Fury.

History



October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.

Late 70s: Display controllers

August 1985: Array Technology formed, later becomes ATI.

January 1992: OpenGL 1.0 releases.

May 1995: Nvidia releases the NV1.

September 1995: DirectX 1.0 released.

November 1996: 3Dfx releases the Voodoo.

November 2000: 3Dfx stops producing cards under their own brand.

2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.

February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.

December 2002: DirectX 9.0 releases with HLSL.

April 2004: OpenGL 2.0 releases with GLSL.

Present Day: GeForce 1080 and Radeon R9 Fury.

History

Late 70s: Display controllers

August 1985: Array Technology formed, later becomes ATI.

January 1992: OpenGL 1.0 releases.

May 1995: Nvidia releases the NV1.

September 1995: DirectX 1.0 released.

November 1996: 3Dfx releases the Voodoo.

October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.

November 2000: 3Dfx stops producing cards under their own brand.

2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.

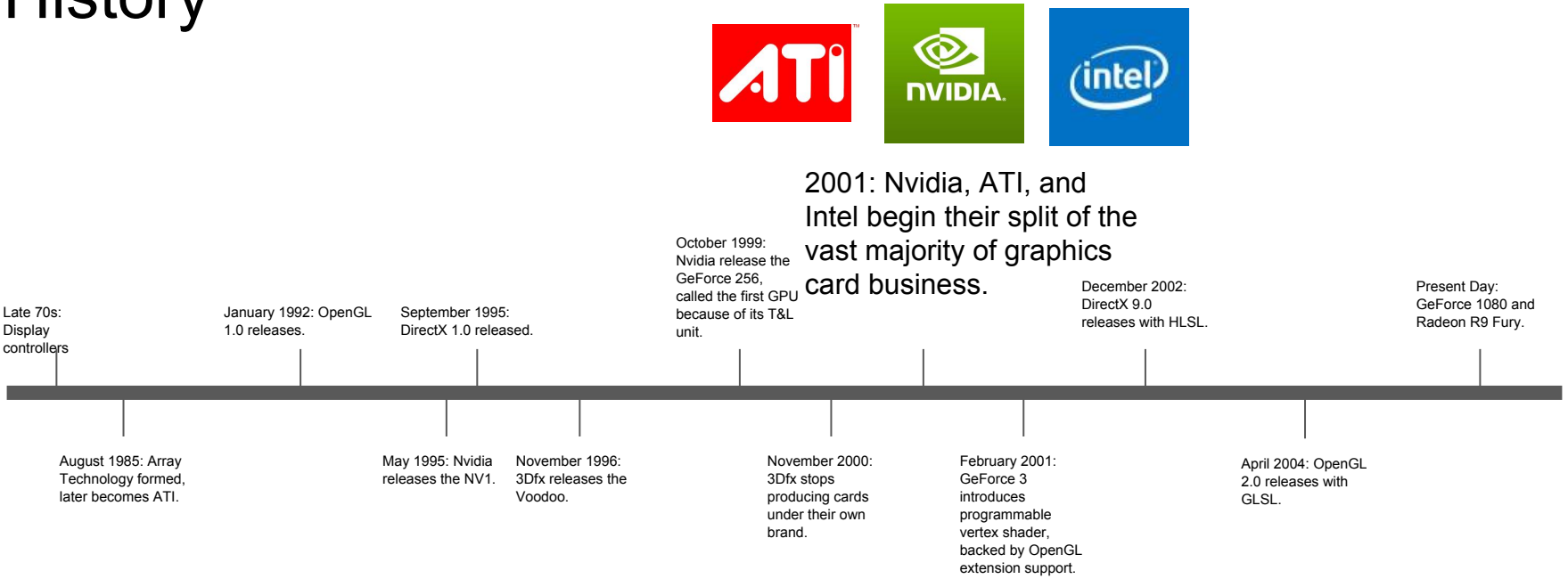
February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.

December 2002: DirectX 9.0 releases with HLSL.

April 2004: OpenGL 2.0 releases with GLSL.

Present Day: GeForce 1080 and Radeon R9 Fury.

History


Late 70s: Display controllers
August 1985: Array Technology formed, later becomes ATI.
January 1992: OpenGL 1.0 releases.
May 1995: Nvidia releases the NV1.
September 1995: DirectX 1.0 released.
November 1996: 3Dfx releases the Voodoo.
October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.
November 2000: 3Dfx stops producing cards under their own brand.
2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.
February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.
December 2002: DirectX 9.0 releases with HLSL.
April 2004: OpenGL 2.0 releases with GLSL.
Present Day: GeForce 1080 and Radeon R9 Fury.

History

Late 70s: Display controllers

August 1985: Array Technology formed, later becomes ATI.

January 1992: OpenGL 1.0 releases.

May 1995: Nvidia releases the NV1.

September 1995: DirectX 1.0 released.

November 1996: 3Dfx releases the Voodoo.

October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.

November 2000: 3Dfx stops producing cards under their own brand.

2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.

February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.

December 2002: DirectX 9.0 releases with HLSL.

April 2004: OpenGL 2.0 releases with GLSL.

Present Day: GeForce 1080 and Radeon R9 Fury.

History

Late 70s: Display controllers

August 1985: Array Technology formed, later becomes ATI.

January 1992: OpenGL 1.0 releases.

May 1995: Nvidia releases the NV1.

September 1995: DirectX 1.0 released.

November 1996: 3Dfx releases the Voodoo.

October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.

November 2000: 3Dfx stops producing cards under their own brand.

2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.

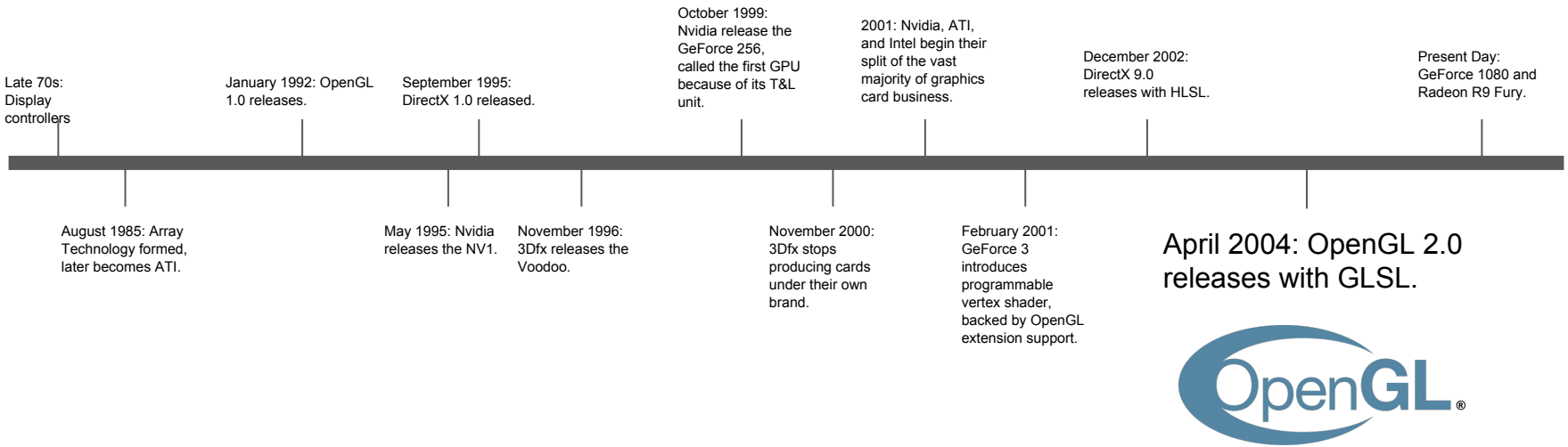
February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.

December 2002: DirectX 9.0 releases with HLSL.

April 2004: OpenGL 2.0 releases with GLSL.

Present Day: GeForce 1080 and Radeon R9 Fury.

History


Late 70s: Display controllers
August 1985: Array Technology formed, later becomes ATI.
January 1992: OpenGL 1.0 releases.
May 1995: Nvidia releases the NV1.
September 1995: DirectX 1.0 released.
November 1996: 3Dfx releases the Voodoo.
October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.
November 2000: 3Dfx stops producing cards under their own brand.
2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.
February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.
December 2002: DirectX 9.0 releases with HLSL.
April 2004: OpenGL 2.0 releases with GLSL.
Present Day: GeForce 1080 and Radeon R9 Fury.

History

Late 70s: Display controllers

August 1985: Array Technology formed, later becomes ATI.

January 1992: OpenGL 1.0 releases.

May 1995: Nvidia releases the NV1.

September 1995: DirectX 1.0 released.

November 1996: 3Dfx releases the Voodoo.

October 1999: Nvidia release the GeForce 256, called the first GPU because of its T&L unit.

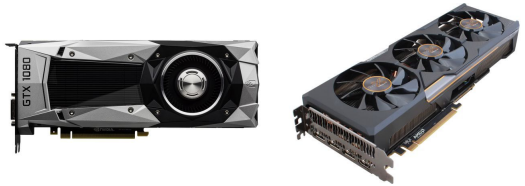
November 2000: 3Dfx stops producing cards under their own brand.

2001: Nvidia, ATI, and Intel begin their split of the vast majority of graphics card business.

February 2001: GeForce 3 introduces programmable vertex shader, backed by OpenGL extension support.

December 2002: DirectX 9.0 releases with HLSL.

April 2004: OpenGL 2.0 releases with GLSL.



Present Day: GeForce 1080 and Radeon R9 Fury.

Modern APIs

Images courtesy of Arstechnica, Wikimedia, and Apple

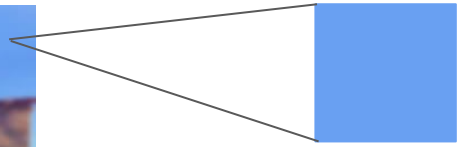
The Big Picture

- We have a virtual world, and we need a way to see it.
- Everything in the scene needs to be displayed on the screen.
- This series of lectures explains this process, from beginning to end.
- But first...

Disclaimers

- Every individual topic in this(ese) lecture(s) could be an entire lecture by itself.
- This is a crash course in the graphics pipeline.
- It contains only the essentials.
- And some relevant deep dives.
- Always feel free to dive deeper!

Concepts: Framebuffer

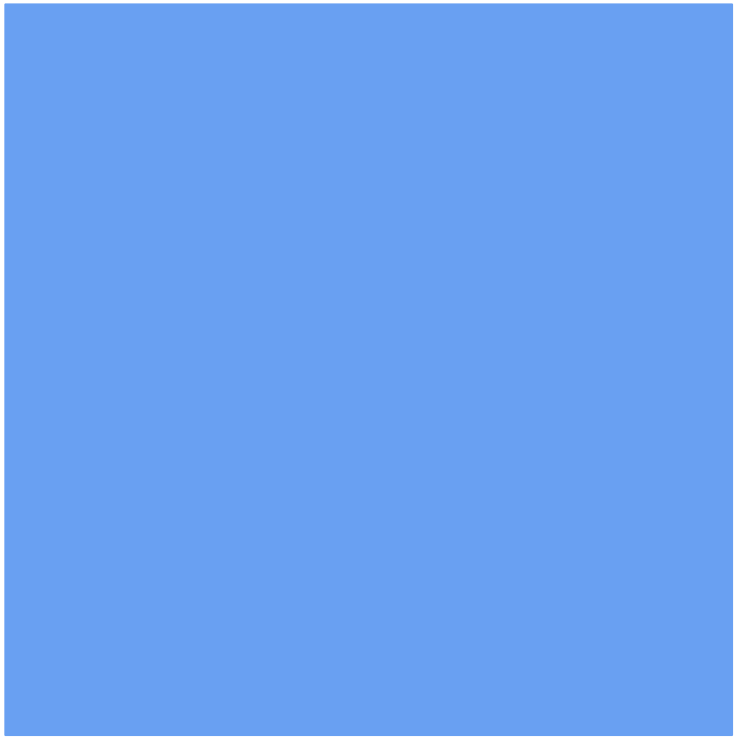


R: 105 G: 160 B: 242

0x69 0xA0 0xF2

Image courtesy of Vicarious Visions.

Concepts: Framebuffer



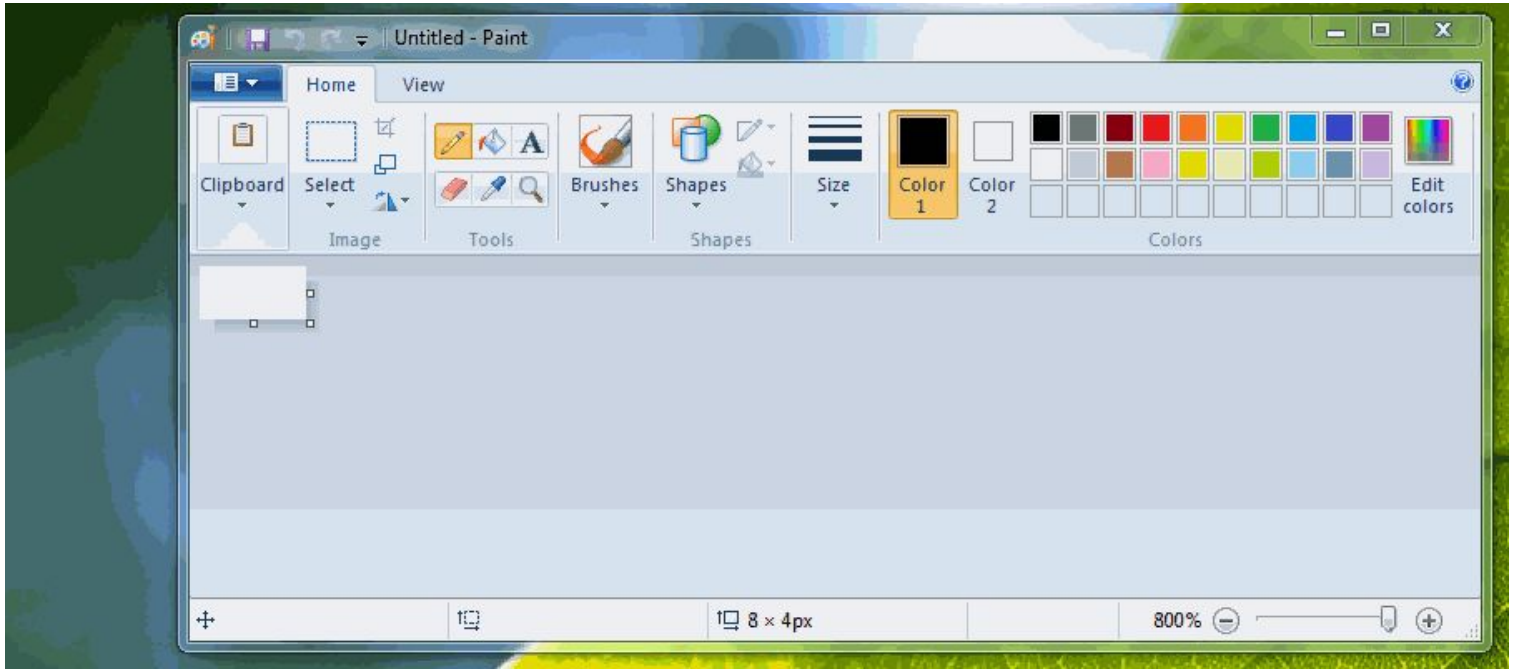
=

{ 0x69, 0xA0, 0xF2, 0x69, 0xA0, 0xF2,
0x69, 0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2, 0x69,
0xA0, 0xF2, 0x69, 0xA0, 0xF2...

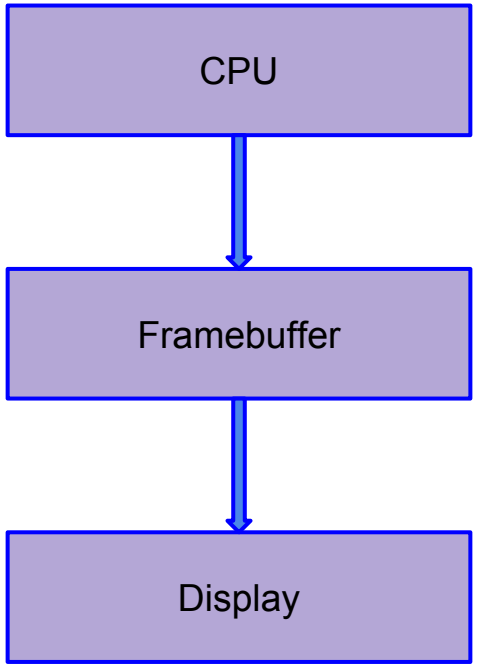
Images are nothing more than arrays of numbers.

Their appearance depends solely on how you interpret the data.

Which means...

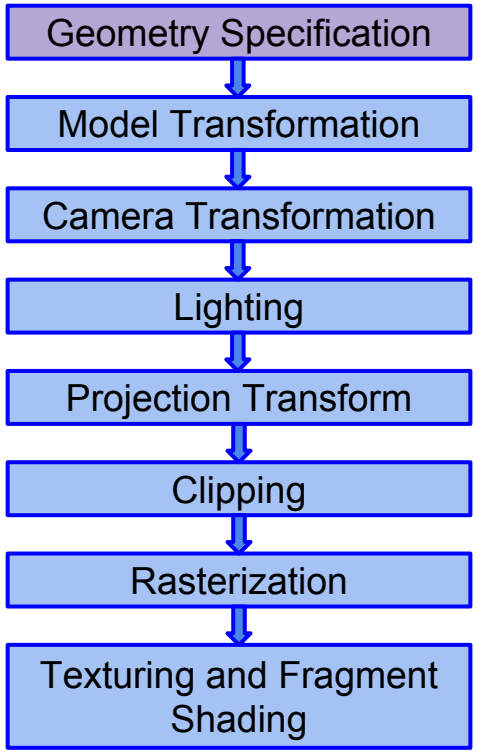

Untitled - Paint
Home
View
Clipboard
Select
Image
Tools
Brushes
Shapes
Shapes
Size
Color 1
Color 2
Edit colors
Colors
8 × 4px
800%

Pre-GPU Pipeline



Fixed Function Pipeline

CPU
Fixed Hardware



Fixed Function Pipeline

- OpenGL Immediate Mode
- API calls for transformations
- API calls for triangles
- API calls for vertex coloring
- API calls for texturing

Fixed Function Pipeline

- **OpenGL Immediate Mode**
- API calls for transformations
- API calls for triangles
- API calls for vertex coloring
- API calls for texturing

Specify each individual vertex and its attributes with individual API calls.

Fixed Function Pipeline

- OpenGL Immediate Mode
- **API calls for transformations**
- API calls for triangles
- API calls for vertex coloring
- API calls for texturing

```
glLoadIdentity();
glTranslatef(
    0.0f,1.0f,2.0f);
glRotatef(
    45.0f,0.0f,1.0f,0.0f);
```

Fixed Function Pipeline

- OpenGL Immediate Mode
- API calls for transformations
- **API calls for triangles**
- API calls for vertex coloring
- API calls for texturing

```
glBegin(GL_TRIANGLES);
glVertex3f(
    0.0f, 1.0f, 0.0f);
glVertex3f(
    -1.0f, -1.0f, 0.0f);
glVertex3f(
    1.0f, -1.0f, 0.0f);
glEnd();
```

Fixed Function Pipeline

- OpenGL Immediate Mode
- API calls for transformations
- API calls for triangles
- **API calls for vertex coloring**
- API calls for texturing

```
// Red.
glColor3f(
    1.0f, 0.0f, 0.0f);
```

Fixed Function Pipeline

- OpenGL Immediate Mode
- API calls for transformations
- API calls for triangles
- API calls for vertex coloring
- **API calls for texturing**

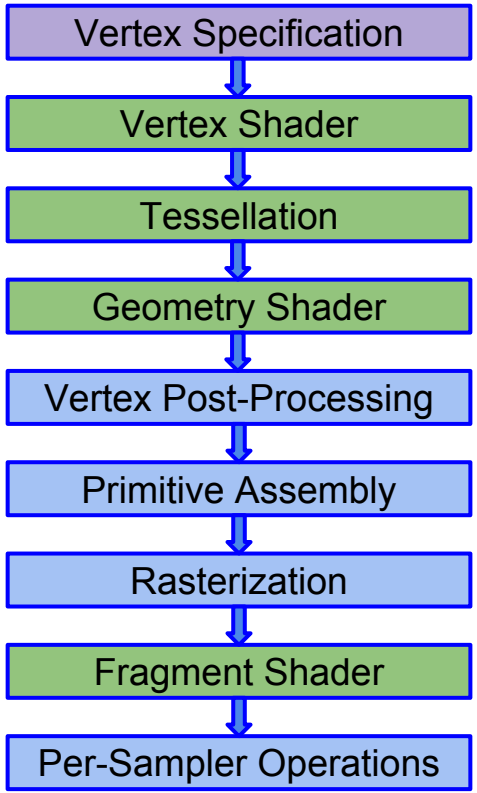
```
// Lower-left of texture.
glTexCoord2f(
    0.0f, 0.0f);
```

Fixed Function Pipeline

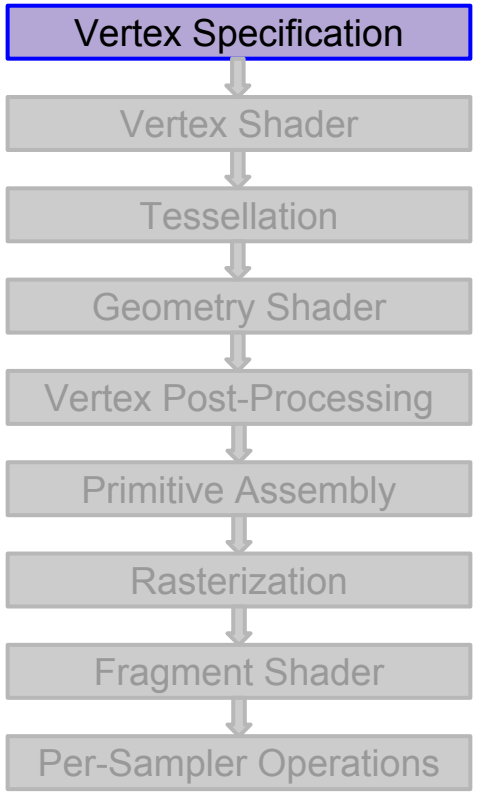
$$f(a, b, c) \rightarrow (x, y, z)$$

Programmable Pipeline

- CPU
- Programmable
- Fixed Hardware



Programmable Pipeline



Scene Representation

- How do we describe objects in the scene?
- How do we describe scene topology?
- How do we describe where things are?
- How do we describe how we see things?

Scene Representation

- **How do we describe objects in the scene?**
- **How do we describe scene topology?**
- How do we describe where things are?
- How do we describe how we see things?

Tessellation

- Tiling without gaps or overlap
- Breaking a surface into polygons
- Necessary for rendering
    - The hardware is made to work with polygons.

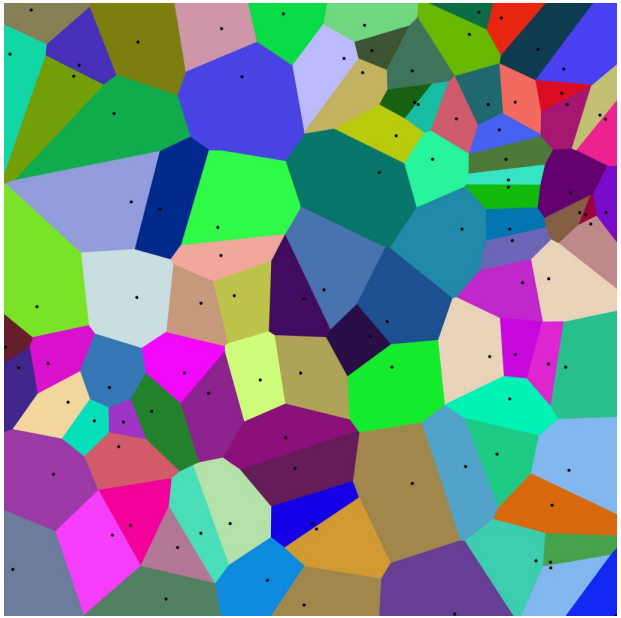


Image courtesy of Wikipedia.

Triangulation

- Tessellating a surface into triangles.
- Triangles are ideal to work with.
  - Almost all modern graphics hardware is designed for triangle rasterization.
- There are still cons.
  - Tessellation is normally fixed.
  - Discrete polygons make blocky silhouettes.

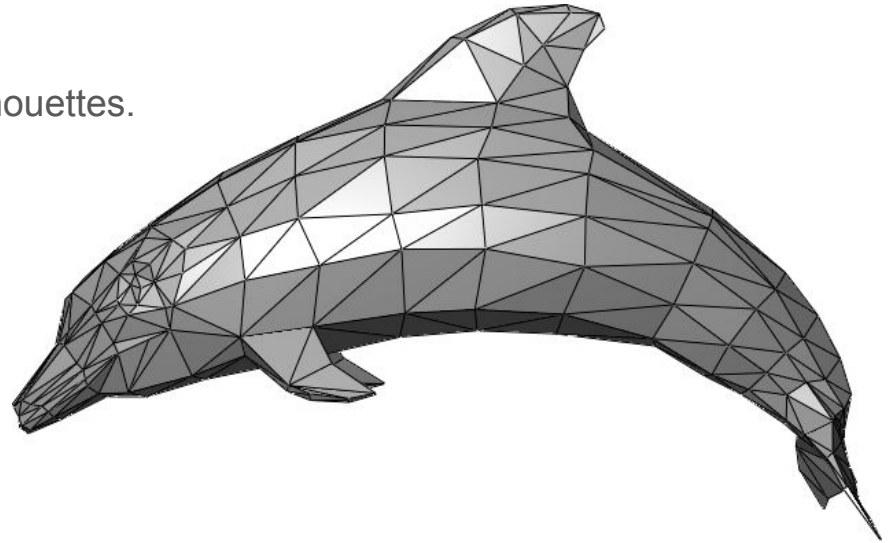


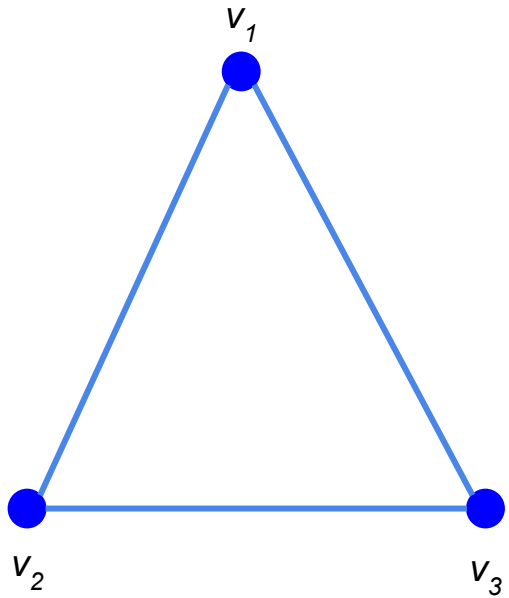
Image courtesy of Crhschn via Wikimedia Commons.

Triangles

- Representation
- Always planar
- Winding order
- Normal calculation

Triangles

- **Representation**
  - Three points, each known as a vertex
- Always planar
- Winding order
- Normal calculation



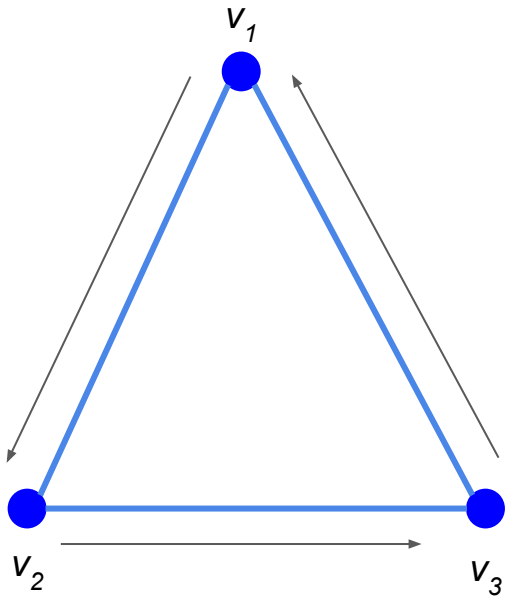
$t = \{ v_1, v_2, v_3 \}$

Triangles

- Representation
- **Always planar**
  - Three points of a triangle always lie on the same plane.
  - This is better than a quad, which can be non-planar and create rendering artifacts.
- Winding order
- Normal calculation

Triangles

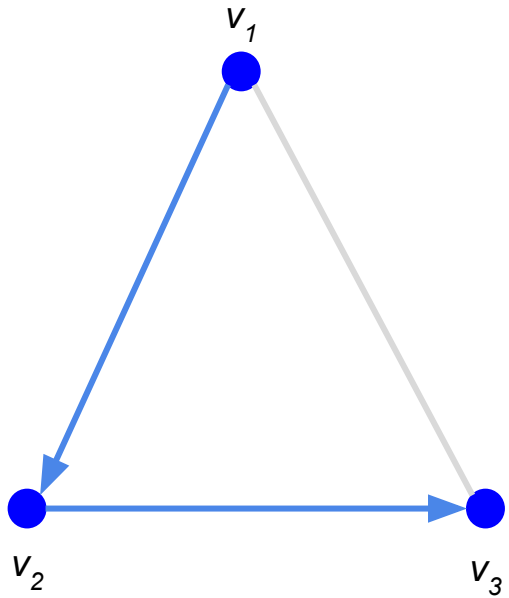
- Representation
- Always planar
- **Winding order**
  - Dictates which side is the ‘front’, and which is the ‘back.’
- Normal calculation



$t = \{ v_1, v_2, v_3 \}$

Triangles

- Representation
- Always planar
- Winding order
- **Normal calculation**
  - Simply cross one edge with another.



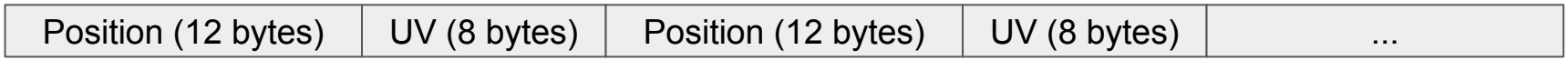
$N = (v_1, v_2) \times (v_2, v_3)$

Vertex Specification

- A can vertex can contain a lot of different information:
  - Position, UV, color, normal, binormal, tangent, bone weights, baked lighting, etc.
- The set of data contained in a vertex is called a **vertex format**.
- In OpenGL, each datum is called a **vertex attribute**.
  - glVertexAttributePointer
  - glEnableVertexAttribArray
- The attribute array index corresponds to the data’s location in the vertex shader specification.

Vertex Format Example

```
struct vertex
{
     vec3 position;
     vec2 uv;
};
```

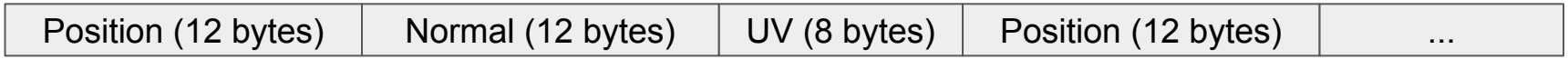



```
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, sizeof(vertex), 0);
glEnableVertexAttribArray(0);

glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, sizeof(vertex), ((char*)NULL + (12)));
glEnableVertexAttribArray(1);
```

Vertex Format Example

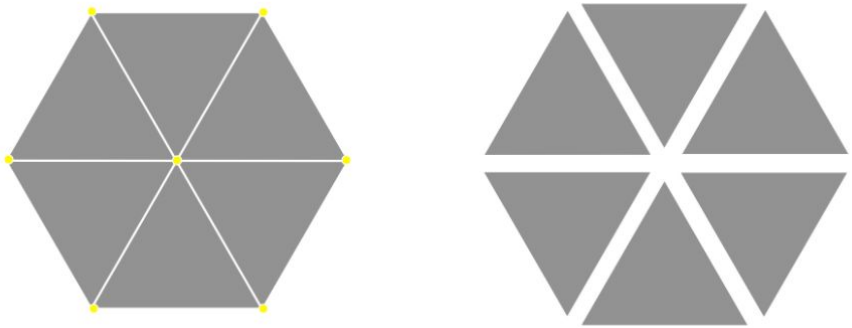
```
struct vertex
{
     vec3 position;
     vec3 normal;
     vec2 uv;
};
```




```
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, sizeof(vertex), 0);
glEnableVertexAttribArray(0);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, sizeof(vertex), ((char*)NULL + (12)));
glEnableVertexAttribArray(1);
glVertexAttribPointer(2, 2, GL_FLOAT, GL_FALSE, sizeof(vertex), ((char*)NULL + (24)));
glEnableVertexAttribArray(2);
```

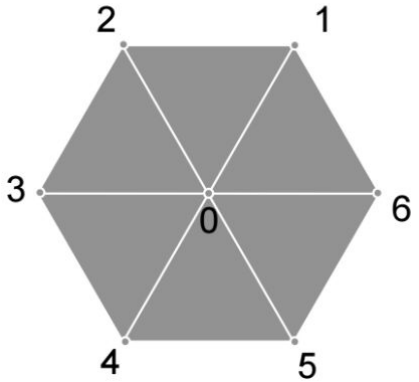
Collections of Triangles

- A naive approach would be to duplicate verts.
  - For each triangle, there would be three verts.
  - But this is memory intensive, and would hog bandwidth on the GPU.



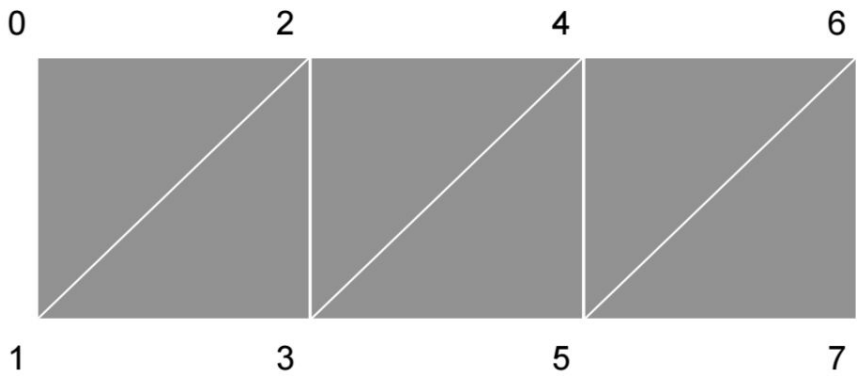
Indexed Triangle Lists

- Instead, we could store each vertex once.
- Triangles are specified by supplying three vertex indices.
  - In the example here, the triangles are…
    - (0,1,2) (0,2,3) (0,3,4)
      (0,4,5) (0,5,6) (0,6,1)



Triangle Strips

- If we want to be even more efficient, we can use **triangle strips**.
  - Similar to an indexed triangle list, but after the first three indices, each subsequent index forms a triangle with the two previous indices.
  - Or, as an example…
    - Index list (0,1,2,3,4,5,6,7) make triangles…
    - (0,1,2) (1,3,2) (2,3,4) (3,5,4) (4,5,6) (5,7,6)
  - Essentially (n-2, n-1, n), but two previous values swap places to maintain winding order.



Vertex Cache

- Modern hardware is optimized for triangle strips, but they’re not essential.
  - It’s much harder to specify a model as one giant triangle strip.
- A GPU maintains a cache of transformed vertices.
  - Just because two triangles share a vertex doesn’t mean the cache helps.
  - The vertex may have been evicted between assembling triangles.
- All hardware is different.
  - Some offline tools are available to optimize vertex data for a target hardware’s vertex cache.

OpenGL

- In OpenGL, vertex data is stored in **buffers**.
  - glGenBuffers
  - glBindBuffer
  - glBufferData
- Data may be **separate**, or **interleaved**.
  - glVertexAttribPointer
- Buffer state can be encapsulated in single **vertex array object**.
  - glGenVertexArrays
  - glBindVertexArray
  - Only recalls buffer state; does not actually store data.
- This is how we get data to the GPU.

OpenGL

- glDrawElements actually issues a draw call.
  - The bound vertex buffers and index buffers are read.
    - Or you can pass in an index buffer.
  - The primitive type passed to glDrawElements determines how the data is read. For example:
    - GL_POINTS - Each index is a primitive.
    - GL_LINE_STRIP - Each index and the previous index makes a line segment.
    - GL_TRIANGLES - Every group of three indices make a triangle.
    - GL_TRIANGLE_STRIP - Refer back to triangle strip slide.
    - GL_QUADS - Every group of four indices make a quad.

Want to learn more?

- For more on the history of the GPU:
  - http://www.techspot.com/article/650-history-of-the-gpu/
- For more on the graphics pipeline:
  - https://fgiesen.wordpress.com/2011/07/09/a-trip-through-the-graphics-pipeline-2011-index/
  - https://www.opengl.org/wiki/Rendering_Pipeline_Overview
- For reference material on OpenGL:
  - https://www.opengl.org/sdk/docs/man/
  - https://www.khronos.org/opengl/wiki

End of Lecture

- Homework 3 is due next Thursday, 2/16.
- This lecture covered only a third of it.
- The next two lectures will cover the rest.
- You are, of course, free to move ahead.