Displays, color, and coordinates,

And a homework assignment

Homework assignment

- Please read chapter 2 for a week from Thursday
- There will be a quiz on the major topics
  - The quiz will NOT ask you to name specific WebGL functions

Today

- What are computer displays and how do they work?
- What is color, how do we see it, and how is it represented?
- What are coordinates?

Computer displays

- What type of computer displays can you think of?

Pixels

- **Pic**ture **El**elemnt
- Little tiny areas of light that make up a computer display
- Almost always represented internally as a grid of numbers

Atari 2600

- VERY limited hardware
- 4kB ROM
- 128 bytes RAM
- Video:
    - Two 8-bit horizontal lines
    - One 1-, 2-, 4-, or 8-pixel wide "ball"
    - Two 1-, 2-, 4-, or 8-pixel wide "missiles"
    - Foreground, background, and 2 player colors

https://prilik.com/blog/wideNES

Framebuffer

- Almost all modern displays use a framebuffer
- Framebuffer: chunk of memory that stores the color for each pixel
- This can make life a whole lot easier
- For example, VGA/DOS computer graphics store data starting at 0xA0000000

```
char *vga = (char*) 0xA0000000;
vga[10] = 32;
```

Color

- Many of these displays are color displays
- Some (Ninja Turtles, Vectrex) are pre-drawn color
- Others allow one of a selection of colors to be drawn in certain places
- Others allow any color to be drawn anywhere
- Almost all of these do this by allowing you to specify an amount of red, green, and blue for each pixel

How do we see color?

- Very complicatedly
- tl;dr:
- Doubling the RGB value will not make a color twice as bright
- You can't just add R+G+B to figure out brightness
- The same RGB values might look different on different computers
  - Or to different people
  - Or under different lighting

How many possible colors?

- Different systems have different numbers of possible colors
- Ranges from 2 colors to over a billion

1-bit color

- A given pixel can be one of two colors
- Typically uses one bit per pixel
  - 1 byte = 8 pixels

2-bit color

- Allows a total of four different colors
- Usually grayscale
- Usually four pixels to a byte

In between

- Devices such as the Atari 2600, NES, and similar era devices allow a choice of colors from a small set of available colors

8-bit color

- Allows for 256 different colors
- Typically implemented as **indexed color** or **palleted mode**
- Each pixel stores a number from zero to 255
- Separately, a 256-element table maps from the number to a color to be displayed
- VGA graphics on the PC used this

15/16 bit color

- Instead of using an index, represent colors directly
- Use 5 bits for the possible red values
- Use 5 or 6 bits for the possible green values
- Use 5 bits for the possible blue values
- Sometimes referred to as 5/5/5 color or 5/6/5

24-bit color

- 8 bits each for red, green, and blue
- Total of 256 distinct values for each, for a total of 16 million distinct colors
- Among digital art people, also referred to as 8 bit color

32-bit color

- 8 bits each for red, green, and blue
- Total of 256 distinct values for each, for a total of 16 million distinct colors
- Among digital art people, also referred to as 8 bit color
- Also has 8 bits that aren't used to indicate color
- Almost always used instead of 24-bit color

10-bit color *twitch*

- 10-bit color (should be called 30-bit color...) has 10 bits each for red, green, and blue
- Over a billion distinct colors possible
- Just about the high end of modern displays

16-bit color

- 16 bit per channel color
- Used as an intermediate step in some image editing programs
- Helps avoid banding and posterization

Coordinates

- How do we specify which pixel we're talking about?
- Remember the CRT scan pattern?

Pixel/screen coordinates

- Origin is in the upper-left
- X-axis increases to the right
- Y-axis increases down the screen
- Width is equal to the number of pixels
- Height is equal to the number of pixels

Normalized device coordinates

- OpenGL was designed so programs don't depend on the size of our screen
- Instead, graphics that we draw are drawn in normalized device coordinates
- X ranges from -1 to 1 (increases to the right)
- Y ranges from -1 to 1 (increases going up the screen(!))