CSE 3902: Basic Intro to 2D Graphics

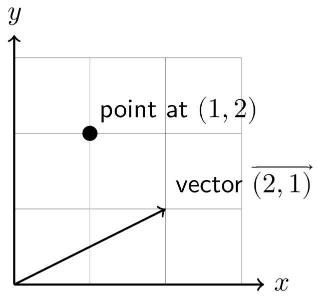
Justin Holewinski

The Ohio State University

Cartesian Coordinate Systems

Two interpretations of a set of coordinates:

- A *position* defining x and y coordinate values
- A *vector* defining a direction and magnitude



More on Vectors

The *length* (or *magnitude*) of a vector is defined as:

$$|\vec{v}| = \sqrt{{v_x}^2 + {v_y}^2}$$

How did we derive that?

What if we just want the direction? Normalize it!

$$\frac{\vec{v}}{|\vec{v}|}$$

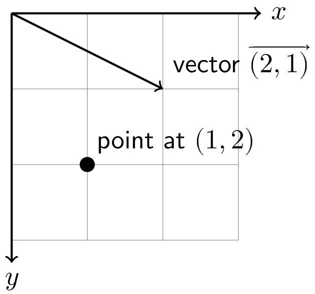
More on Vectors

Why do we care about the vector direction vs length?

- Think of it in terms of *direction* and *speed*
- Sometimes you only care about direction (“Am I heading north or south?”)
- Sometimes you only care about speed (“Is the player going fast enough to meet an objective?”)
- Sometimes you care about both (“What is the player’s next position?”)

MonoGame Coordinate System

MonoGame's coordinate system puts the origin at the *top* left



MonoGame Types

`Vector2`

- Two-dimensional vector (containing `X` and `Y` fields)
- Uses `float` coordinates
- Supports `Length` and `Normalize` methods

`Point`

- Two-dimensional position (containing `X` and `Y` fields)
- Uses `int` coordinates
- No `Length` or `Normalize` methods

`Rectangle`

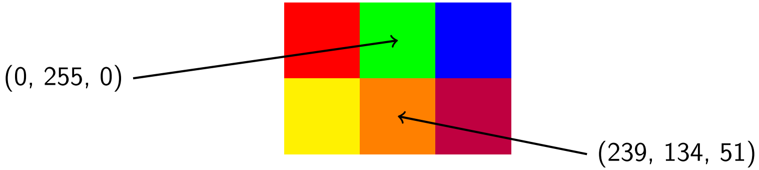
- Two-dimensional rectangle (containing `X`, `Y`, `Width`, and `Height` fields)
- Uses `int` coordinates

Raster Graphics

An image is divided into *pixels*, where each pixel is a single color

Color is typically represented with an 8-bit integer value per red/green/blue channel

- Forms a triple of integers: `{ red, green, blue }`
- 24 bits per pixel
- Usually extended to 32 bits by adding an 8-bit *alpha* channel for transparency



Textures

A *texture* is an image with a particular size usable by graphics hardware

- We only need to use 2D textures for this class
- Technically can be 1D, 1DArray, 2D, 2DArray, 3D, Cube, CubeArray
- Textures can also use non-8-bit RGB formats

MonoGame's *content pipeline* provides an easy way to create textures from image files

- Textures are stored in an `.xnb` file, an opaque format for the purposes of this class
- The content pipeline allows converting image files to `.xnb`
  - `.png`
  - `.jpg`
  - `.bmp`
- Add an image file to your project's `Content.mgcb` file, and MonoGame will automatically convert to `.xnb` when you build your project

Textures

Trivial to load in MonoGame using the `Content` object

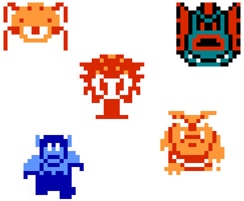
```
// "Content" is available through "Game"
// The given path is relative to "Content.mgcb"
Texture2D tex = Content.Load<Texture2D>("textures/player");
```

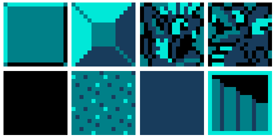
Used directly as argument to sprite drawing methods

Sprites

A *sprite* is a 2D image representing a single game entity

- Textures are used to store sprites
- They are sometimes synonymous with *image*, but a sprite is more narrow use of a texture



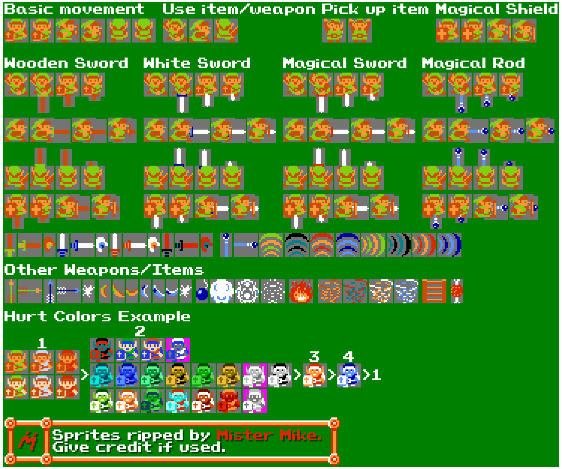


Sprite Source: https://www.spriters-resource.com/nes/legendofzelda/

Texture Atlas

Many sprites will be animated. How do we store animation?

Store all animation frames in the same texture! This is a *texture atlas*.



Sprite Source: https://www.spriters-resource.com/fullview/8366/

Texture Atlas

Why not just use a separate texture for each animation frame?

- Efficiency
- No need to change texture for each sprite draw command
- There is a (small) overhead associated with each texture object

Best Practices

- Keep sprite sizes consistent (e.g. 16x16 or 32x32)
    - Makes it easier to derive source rectangle for each frame
- Leave yourself some room between frames
    - Packing is good for efficiency, but unnecessary for this class
    - Padding makes it easier to visualize individual frames

SpriteBatch

MonoGame uses `SpriteBatch` to efficiently draw sprites

- Automatically handles batching of draw calls to minimize overhead
- Can sort draw calls based on layer, texture, program order, ...

```
SpriteBatch batch = new SpriteBatch(GraphicsDevice);
Texture2D tex = Content.Load<Texture2D>("textures/player");

batch.Begin();
// Draw "texture" at coordinate (10, 20) with same size as texture
batch.Draw(tex, new Vector2(10, 20), Color.White);
batch.End();
```

What's with the `Color.White` argument?

In a real example, you would not re-create the `SpriteBatch` nor re-load the `Texture2D` every frame!

SpriteBatch

There are a lot of overloads of `Draw`!

- Draw whole texture at full size
- Draw whole texture to scaled subset of window
- Draw subset of texture at full size
- Draw subset of texture to scaled subset of window
- ...
- Read the API reference!!!

https://docs.monogame.net/api/Microsoft.Xna.Framework.Graphics.SpriteBatch.html#methods

SpriteBatch

How do we draw just the frame we want with `SpriteBatch`?

Most of the sprite drawing in this class will be done with:

```
public void Draw(Texture2D texture,
                 Rectangle destinationRectangle,
                 Rectangle? sourceRectangle,
                 Color color)
```

Draw a portion of the texture to a portion of the window

- `SpriteBatch` takes care of any necessary scaling