Game Programming

Animated Sprites

Admin

- Exam, Bridgewater 2-step in Nov ()
- and Quiz Schedule
  - Any concerns?
- Next Project at midterm next week for night class.

Animated Sprites I

- This is something that many game engines hide from you, but you should understand how it is done even if you use an engine
- So far we have a static image moving around the screen as a sprite
  - All well and good, but we'd like to have an animated sprite
  - A first pass:
    - Use animated gifs
    - Why is this undesirable?

Animated Sprites I

- So far we have a static image moving around the screen as a sprite
  - All well and good, but we'd like to have an animated sprite
  - A first pass:
    - Use animated gifs
    - Why is this undesirable?
    - Well Animated gifs are self animating
      - Want to pause animations when pausing the game
      - Or at win/loss of game
  -

Animated Sprites II

- We need to create our own animated sprites
- Basic Idea:
  - Take the series of related images
  - Draw them on at a time till each has been displayed
    - Then start again
    - Have to be sure not to cycle the images too fast
  - Ebiten
    - We have to do this ourselves again

Animated Sprites II

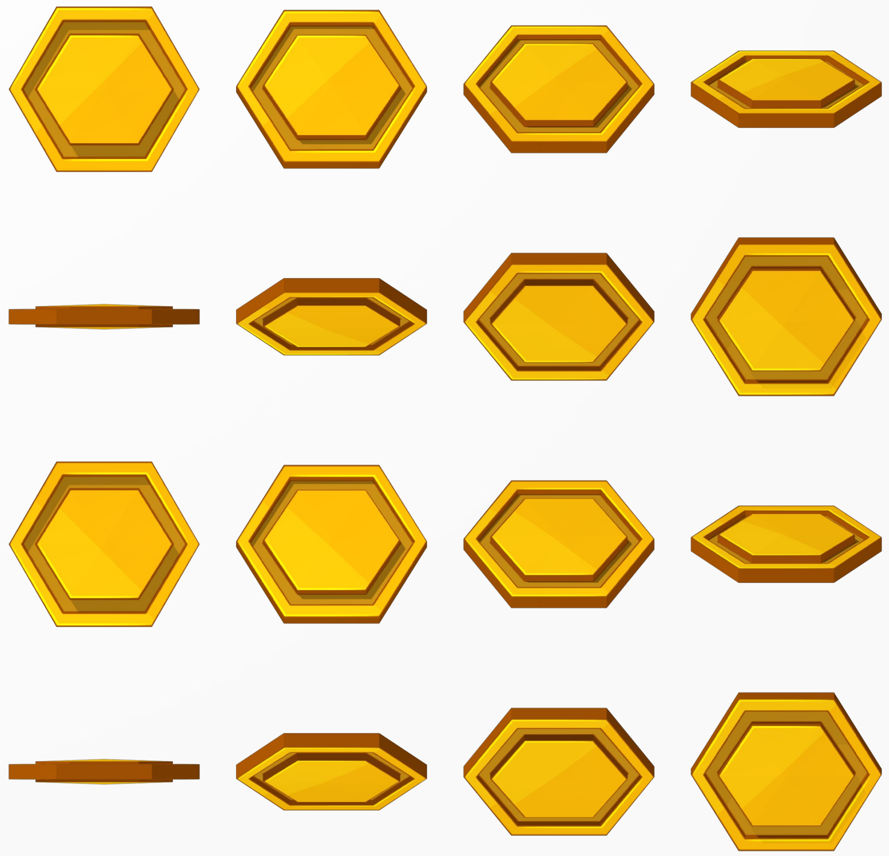
- We need to create our own animated sprites
- Basic Idea:
  - Take the series of related images
  - Draw them one at a time till each has been displayed
    - Then start again
    - Have to be sure not to cycle the images too fast
  - Libraries I've used in the past
    - Pygame: need to do this ourselves
    - Arcade: two animated sprite classes available to us, but recommendation to roll your own

Images or Spritesheets

- Once upon a time
  - DOS (includes windows 95/98) limited number of files per folder
    - So spritesheet
  - And then for 15 years or so
    - Lots of individual images used
  - Then/now for web games
    - Spritesheet
  - For local games
    - Which ever.
    - Lots of games use either approach.

Animated Sprite

- I’ve got this animated coin spritesheet
  - https://opengameart.org/content/spinning-coin-animation-atlas
  - Coin_Spin_Animation_A.png
  - I’ve got it in a subfolder of my standard Assets subfolder called Things
  - Image is 2048x2048
    - With four images per row/column
    - Means that each image is 512 pixels
    -



Ebitengine Animation

- Ebitengine
  - Do the animation yourself
    - Often hidden by engines like Unity/Unreal/Godot etc.
  - When working with a sprite sheet
    - Use ebiten.Image SubImage method
    - func (i *Image) SubImage(r image.Rect) image.Image
    - image.Rectangle
      - Pass lower left x, lower left y, upper right x and upper right y

Utility function

- You might use this or a similar utility function in all of your projects

```go
func LoadEmbeddedImage(folderName string, imageName string) *ebiten.Image {
	embeddedFile, err := EmbeddedAssets.Open(path.Join("assets", folderName, imageName))
	if err != nil {
		log.Fatal("failed to load embedded image ", imageName, err)
	}
	ebitenImage, _, err := ebitenutil.NewImageFromReader(embeddedFile)
	if err != nil {
		fmt.Println("Error loading tile image:", imageName, err)
	}
	return ebitenImage
}
```

The Exemplar Project

- For this next few slides you can find the complete stripped down project here:
- https://github.com/jsantore/AnimatedSprite

The Beginning

```
bed"
t"
hub.com/hajimehoshi/ebiten/v2"
hub.com/hajimehoshi/ebiten/v2/ebitenutil"
age"
"
h"



bed assets/*
eddedAssets embed.FS



DOW_WIDTH   = 1000
DOW_HEIGHT  = 1000
N_DIMENSION = 512.0
ME_COUNT    = 4
```

Game Struct, layout and main

```go
type AnimatedSpriteDemo struct {
	CoinImage  *ebiten.Image
	xFrame     int
	yFrame     int
	FrameDelay int
}

func (demo AnimatedSpriteDemo) Layout(outsideWidth, outsideHeight int) (screenWidth, screenHeight int) {
	return outsideWidth, outsideHeight
}

func main() {
	coin := LoadEmbeddedImage("", "Coin_Spin_Animation_A.png")
	demo := AnimatedSpriteDemo{CoinImage: coin} //xFrame and yFrame deliberately 0
	ebiten.SetWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT)
	ebiten.SetWindowTitle("Sprite animation on Sprite Sheet")
	err := ebiten.RunGame(&demo)
	if err != nil {
	fmt.Println("Error running game:", err)
	}
}
```

Update and Draw

```
func (demo *AnimatedSpriteDemo) Update() error {
        demo.xFrame += 1
        if demo.xFrame >= FRAME_COUNT {
          demo.xFrame = 0
          demo.yFrame += 1
          if demo.yFrame >= FRAME_COUNT {
            demo.yFrame = 0
        }}
        return nil
}

func (demo *AnimatedSpriteDemo) Draw(screen *ebiten.Image) {
        op := &ebiten.DrawImageOptions{}
        op.GeoM.Translate(COIN_DIMENSION/2, COIN_DIMENSION/2)
        frameX := demo.xFrame * COIN_DIMENSION
        frameY := demo.yFrame * COIN_DIMENSION
        screen.DrawImage(demo.CoinImage.SubImage(image.Rect(frameX, frameY,
                        frameX+COIN_DIMENSION, frameY+COIN_DIMENSION)).(*ebiten.Image), op)
}
```

That’s pretty fast

- How could we fix that?

That’s pretty fast

- How could we fix that?
  - Only call update_animation a few times per second
- Updated __init__ and update

```
func (demo *AnimatedSpriteDemo) Update() error {
	demo.FrameDelay += 1
	if demo.FrameDelay%5 == 0 {  //adjust this to speed up or slow down the animation
		demo.xFrame += 1
		if demo.xFrame >= FRAME_COUNT {
				demo.xFrame = 0
				demo.yFrame += 1
				if demo.yFrame >= FRAME_COUNT {
				demo.yFrame = 0
				}
		}
	}
	return nil
}
```

List of images

- The other way is to use a folder full of images
  - For this I'll use the victory dance set from the raccoon
    - https://opengameart.org/content/cute-raccoon-2d-game-sprite-and-animations
    - Then grab the images out of the victory dance folder.
  - I've got them in Assets/raccoon
  - If we look at them we see that we have 14 images.

Program incidentals

- Even smaller than last time

```go
import (
	"embed"
	"fmt"
	"github.com/hajimehoshi/ebiten/v2"
	"github.com/hajimehoshi/ebiten/v2/ebitenutil"
	"log"
	"path"
)

//go:embed assets/*
var EmbeddedAssets embed.FS

type AnimatedSpriteDemo2 struct {
	Raccoon    []*ebiten.Image
	Frame      int
	FrameDelay int
}
```

Main and Load

```go
func main() {
	frames := LoadAllRaccoons()
	ebiten.SetWindowSize(1000, 1000)
	ebiten.SetWindowTitle("sliceOfImeages")
	demo := AnimatedSpriteDemo2{
		Raccoon:    frames,
		Frame:      0,
		FrameDelay: 0,
	}
	ebiten.RunGame(&demo)
}

func LoadAllRaccoons() []*ebiten.Image {
	all_frames := make([]*ebiten.Image, 14, 20)
	suffix_list := []string{"01", "03", "05", "07", "09", "11", "13", "15", "17", "19", "21", "23", "25", "27"}
	for index, suffix := range suffix_list {
		filename := fmt.Sprintf("victory-dance00%s.png", suffix)
		frame_pict := LoadEmbeddedImage("victory-dance", filename)
		all_frames[index] = frame_pict
	}
	return all_frames
}
```

Game Interface

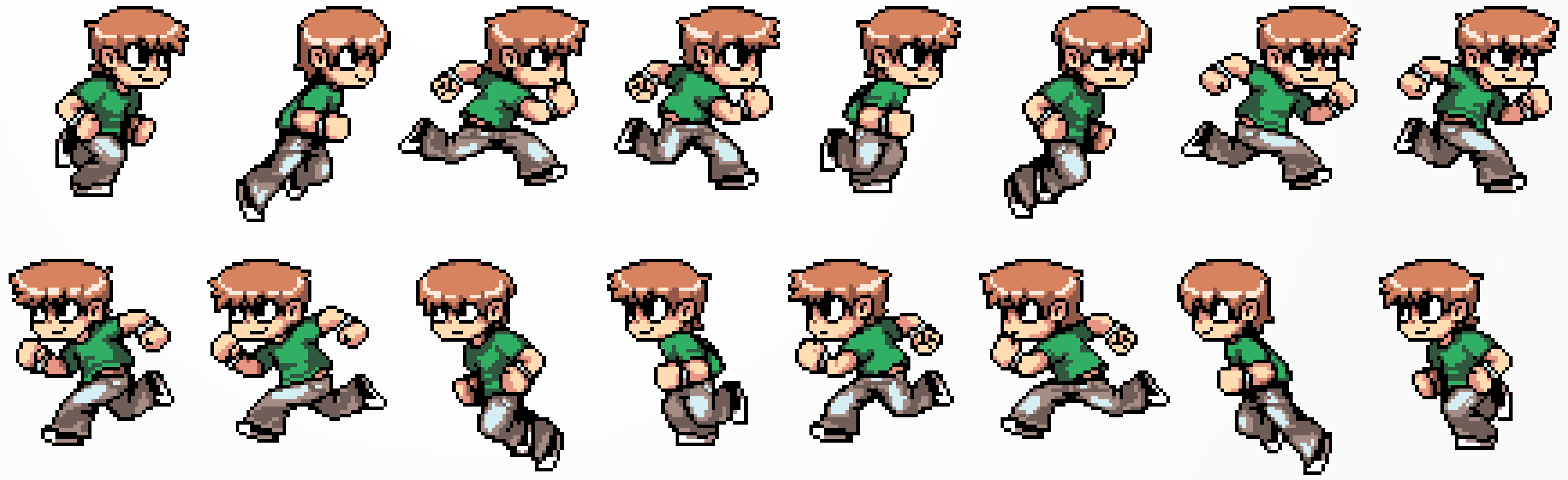
```
func (demo *AnimatedSpriteDemo2) Update() error {
	demo.FrameDelay += 1
	if demo.FrameDelay%5 == 0 {
	demo.Frame += 1
	if demo.Frame >= len(demo.Raccoon) {
	demo.Frame = 0
	}
	}
	return nil
}

func (demo AnimatedSpriteDemo2) Draw(screen *ebiten.Image) {
	drawOps := ebiten.DrawImageOptions{}
	drawOps.GeoM.Reset()
	//drawOps.GeoM.Translate(float64(WINDOW_WIDTH/2), float64(WINDOW_HEIGHT/2))
	screen.DrawImage(demo.Raccoon[demo.Frame], &drawOps)
}

func (demo AnimatedSpriteDemo2) Layout(outsideWidth, outsideHeight int) (screenWidth, screenHeight int) {
	return outsideWidth, outsideHeight
}
```

Character image

- Now lets look at a left right walking sprite
  - I'm using the sprite sheet from this javascript tutorial
  - https://blaiprat.github.io/jquery.animateSprite/
  -

Examining the Image

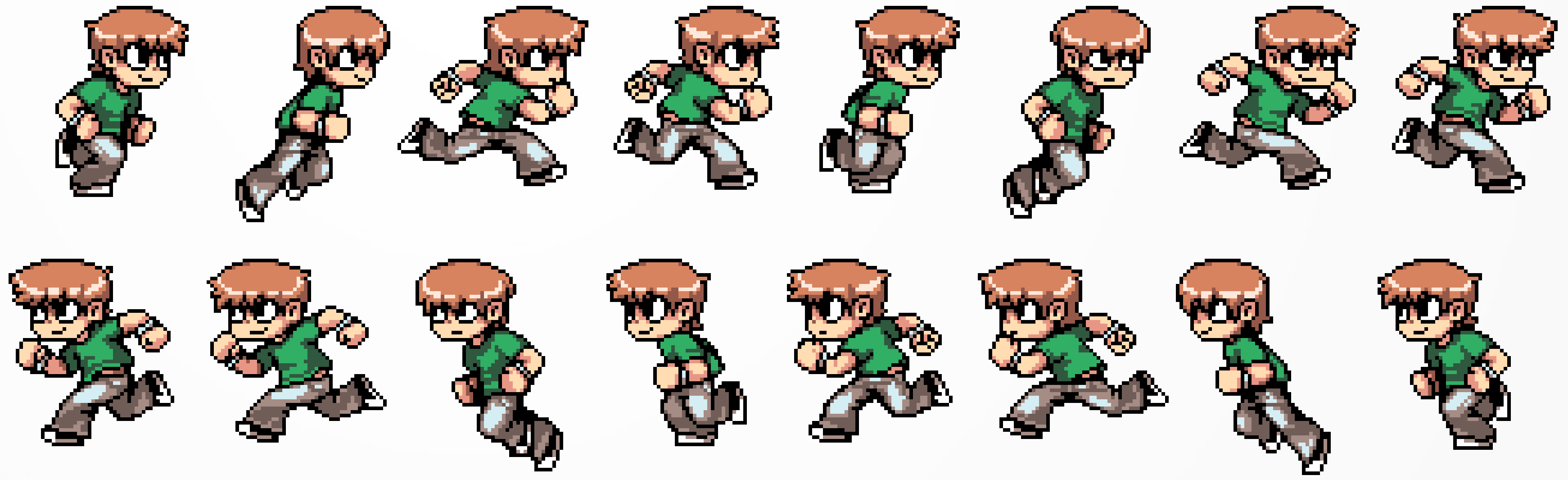
- Image:
-



- Dimensions: 864x280 pixels
  - Two rows: 280/2→ image height is 140 pixels
  - Eight Columns: 864/8 → image width is 108 pixels

Examining the Image

- Image:
-



- Dimensions: 864x280 pixels
  - Two rows: 280/2→ image height is 140 pixels
  - Eight Columns: 864/8 → image width is 108 pixels

Enumerated Type

- What is an enumerated type?

Enumerated Type

- What is an enumerated type?
  - In most languages: a constrained finite set of values for limited number of options
    - Days of the week popular example
    - Maybe direction of travel in games

Enumerated Type in go

- Go doesn't have an exact enumerated type instead use const

```
const(
    UP = iota
    DOWN
    LEFT
    RIGHT
)
```

- Iota is a special keyword that evaluates to zero
  - Others will be incremented by one. Eg: LEFT is 2

Lets take a look

- Lets have a look and try it out
- https://github.com/jsantore/AnimatedSprite
-