

BUILD AN ASTEROIDS GAME IN PYGAME



WHY THIS COURSE?

- Learn how 2D games are structured
- Use Pygame to create some fun
 - Graphics
 - Animation
 - Sound
- Great project to learn more about coding in Python

YOU WILL LEARN:

1. Writing a game with Pygame
2. Using game loops
3. Including image and sound assets
4. Controlling your game through keyboard input

VERSIONS



Note:

- Python 3.9
- Pygame 2.0.1

OVERVIEW

- Many coders come to programming through a love of video games
- Pygame is a 2D game library that abstracts away operating system specific requirements
- Simple games can be up and running quickly
- Asteroids clone in about 250 lines of code

NEXT UP...



Getting started with Pygame

TABLE OF CONTENTS

1. Overview

▶ 2. Getting Started With Pygame

3. Game Structure

4. Sprites

5. Game Models

6. Frames Per Second (FPS)

7. Movement and Rotation

8. Obstacles

9. Shooting

10. Collisions

11. Splitting Rocks

12. Sound

13. Displaying Text

14. Summary

INSTALLATION

- Pygame is a third-party library

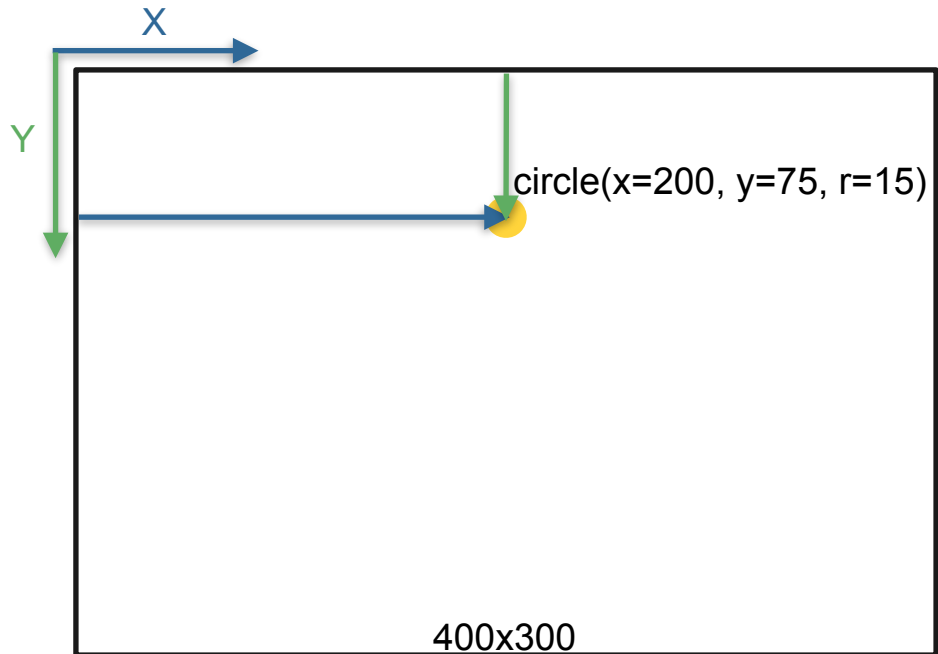
```
$ python -m pip install pygame==2.0.1
```


GAME LOOP

- Video games are structured as a large loop
- Each iteration of the loop:
 - Handles user input
 - Determines new positions of moveable assets
 - Renders a frame of the game
- For smooth animation you want at least 25 frames per second
- Modern gaming hardware frequently uses 60 FPS

COORDINATES

- Pygame screen origin is the top-left corner



NEXT UP...



Game structure

TABLE OF CONTENTS

1. Overview

2. Getting Started With Pygame

 **3. Game Structure**

4. Sprites

5. Game Models

6. Frames Per Second (FPS)

7. Movement and Rotation

8. Obstacles

9. Shooting

10. Collisions

11. Splitting Rocks

12. Sound

13. Displaying Text

14. Summary

PYTHON DIRECTORIES

- Python command can run a script or a directory
- To use the directory style, you need a `__main__.py` file

```
space_rocks/  
├── __main__.py  
└── game.py
```

- With this structure you can pass `python` the directory name to run it

```
$ python space_rocks
```

PYTHON DIRECTORIES

- Create a directory for your code:

```
$ mkdir space_rocks
```

NEXT UP...



Sprites

TABLE OF CONTENTS

1. Overview

2. Getting Started With Pygame

3. Game Structure

 **4. Sprites**

5. Game Models

6. Frames Per Second (FPS)

7. Movement and Rotation

8. Obstacles

9. Shooting

10. Collisions

11. Splitting Rocks

12. Sound

13. Displaying Text

14. Summary

GAME ASSETS

- When rendering graphics you can:
 - Use drawing primitives
 - Use image files
- A **sprite** is an image in your game
- An image file may contain one or more images, possibly different frames in an animation sequence
- When playing sound you can:
 - Use sound primitives
 - Use sound files
- Sprites and sound files are called **assets**

GAME ASSETS

- Pygame provides methods for loading and manipulating assets
- You'll want to create some reusable utility functions to support your code

```
space_rocks/  
├── __main__.py  
├── assets  
│   ├── sprites  
│   │   └── space.png  
├── game.py  
└── utils.py
```

NEXT UP...



Game models

TABLE OF CONTENTS

1. Overview

2. Getting Started With Pygame

3. Game Structure

4. Sprites

 **5. Game Models**

6. Frames Per Second (FPS)

7. Movement and Rotation

8. Obstacles

9. Shooting

10. Collisions

11. Splitting Rocks

12. Sound

13. Displaying Text

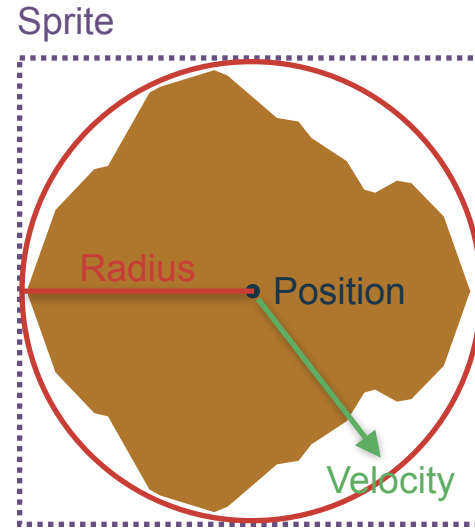
14. Summary

GAME MODELS

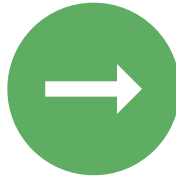
- Space Rocks things:
 - Ship
 - Asteroid (various sizes)
 - Bullets
- Each thing can be an object:
 - Attributes for position, velocity
 - Methods for drawing, detecting collisions
- Start with a generic object to contain this info: `GameObject`

GAME OBJECT

- Object attributes:
 - Sprite
 - Position
 - Velocity
 - Radius



NEXT UP...



FPS

TABLE OF CONTENTS

1. Overview

2. Getting Started With Pygame

3. Game Structure

4. Sprites

5. Game Models

 **6. Frames Per Second (FPS)**

7. Movement and Rotation

8. Obstacles

9. Shooting

10. Collisions

11. Splitting Rocks

12. Sound

13. Displaying Text

14. Summary

FRAMES PER SECOND

- Infinite game loop
- Each iteration calls `pygame.display.flip()`
- Get as many frames as your computer can handle
- Need to put in a control so display is consistent between computers

NEXT UP...



Controlling the ship

TABLE OF CONTENTS

1. Overview

2. Getting Started With Pygame

3. Game Structure

4. Sprites

5. Game Models

6. Frames Per Second (FPS)

 **7. Movement and Rotation**

8. Obstacles

9. Shooting

10. Collisions

11. Splitting Rocks

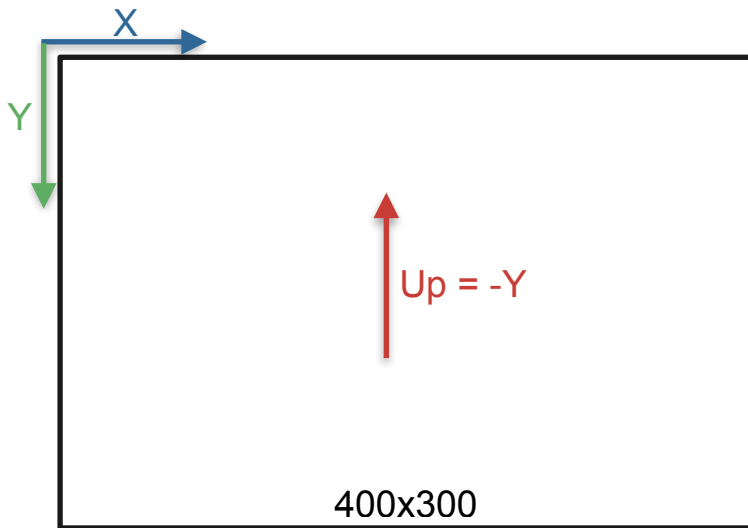
12. Sound

13. Displaying Text

14. Summary

CONTROLLING THE SHIP

- Extend `GameObject` and make it ship specific
- Add input handling to game to control the direction of the ship
- Ship needs to be able to rotate to move in different directions



MOVING THE SHIP

- Ship only has a single rocket
- Only support acceleration
- Slowing down done through turning and accelerating in the other direction

NEXT UP...



Rocks!

TABLE OF CONTENTS

1. Overview

2. Getting Started With Pygame

3. Game Structure

4. Sprites

5. Game Models

6. Frames Per Second (FPS)

7. Movement and Rotation

 **8. Obstacles**

9. Shooting

10. Collisions

11. Splitting Rocks

12. Sound

13. Displaying Text

14. Summary

MUST GO CRASH

- Not much fun without some obstacles
- Extend `GameObject` with code for rocks
- Careful about where rocks appear: avoid instant death

I LIKE TO MOVE IT, MOVE IT

- Obstacles no fun if they're standing still
- Add code to [Rock](#) objects to have them move around

NEXT UP...



Pew, pew!

TABLE OF CONTENTS

1. Overview

2. Getting Started With Pygame

3. Game Structure

4. Sprites

5. Game Models

6. Frames Per Second (FPS)

7. Movement and Rotation

8. Obstacles

 **9. Shooting**

10. Collisions

11. Splitting Rocks

12. Sound

13. Displaying Text

14. Summary

FASTER THAN A SPEEDING...


- Next step is to add shooting
- Add a `Bullet` game object
- Add ability to shoot
- Track bullets in the game logic

NEXT UP...



Bang!

TABLE OF CONTENTS

- 1. Overview**
- 2. Getting Started With Pygame**
- 3. Game Structure**
- 4. Sprites**
- 5. Game Models**
- 6. Frames Per Second (FPS)**
- 7. Movement and Rotation**
- 8. Obstacles**
- 9. Shooting**
-  **10. Collisions**

- 11. Splitting Rocks
- 12. Sound
- 13. Displaying Text
- 14. Summary

COLLISIONS

- Bullets can collide with rocks
 - Remove rock
- Rocks can collide with the ship
 - Remove ship

NEXT UP...



Boulders, rocks, and gravel

TABLE OF CONTENTS

1. Overview

2. Getting Started With Pygame

3. Game Structure

4. Sprites

5. Game Models

6. Frames Per Second (FPS)

7. Movement and Rotation

8. Obstacles

9. Shooting

10. Collisions

 **11. Splitting Rocks**

12. Sound

13. Displaying Text

14. Summary

SPLITTING ROCKS

- Currently rocks are destroyed instantaneously
- Rocks break into smaller rocks
- **Rock** constructor needs to change:
 - Randomly generated rock, and
 - Smaller rock in specific location

NEXT UP...



Make some noise

TABLE OF CONTENTS

- 1. Overview**
- 2. Getting Started With Pygame**
- 3. Game Structure**
- 4. Sprites**
- 5. Game Models**
- 6. Frames Per Second (FPS)**
- 7. Movement and Rotation**
- 8. Obstacles**
- 9. Shooting**
- 10. Collisions**

11. Splitting Rocks



12. Sound

13. Displaying Text

14. Summary

SOUND

- Sound files are assets
- Pygame provides methods for loading and playing sounds

NEXT UP...



Showing info

TABLE OF CONTENTS

- 1. Overview**
- 2. Getting Started With Pygame**
- 3. Game Structure**
- 4. Sprites**
- 5. Game Models**
- 6. Frames Per Second (FPS)**
- 7. Movement and Rotation**
- 8. Obstacles**
- 9. Shooting**
- 10. Collisions**

11. Splitting Rocks

12. Sound

 **13. Displaying Text**

14. Summary

TEXT CONTENT

- Detect win & loss events
- Show player a message
- Text is a special kind of graphic
- Need to load fonts
- Render text on screen

NEXT UP...



Summary

TABLE OF CONTENTS

- 1. Overview**
- 2. Getting Started With Pygame**
- 3. Game Structure**
- 4. Sprites**
- 5. Game Models**
- 6. Frames Per Second (FPS)**
- 7. Movement and Rotation**
- 8. Obstacles**
- 9. Shooting**
- 10. Collisions**

- 11. Splitting Rocks**
- 12. Sound**
- 13. Displaying Text**
-  **14. Summary**

SUMMARY

- Game loops and Pygame basics
- Sprites and images
 - Bit blitting
 - Rotation
 - Movement
- Keyboard control
- Sound
- Fonts and screen text
- Game-y awesomeness

WHAT'S NEXT?

- Some modifications you can try out:
 - Fix the bug where firing the gun after dying crashes the game:
 - Quicker way: add an extra check for “`ship is None`”
 - Better way: add context to the ship: **alive** or **dead** mode
 - Add multiple lives
 - Have the bullets kill the ship if it accelerates into one
 - Add scoring
 - Add a “play again” action
 - Add an “instant replay”

FURTHER INVESTIGATION

- Pygame documentation:
<https://www.pygame.org/docs/>
- Course: Make a 2D Side-Scroller Game with Pygame
<https://realpython.com/courses/pygame-primer/>

Thanks!