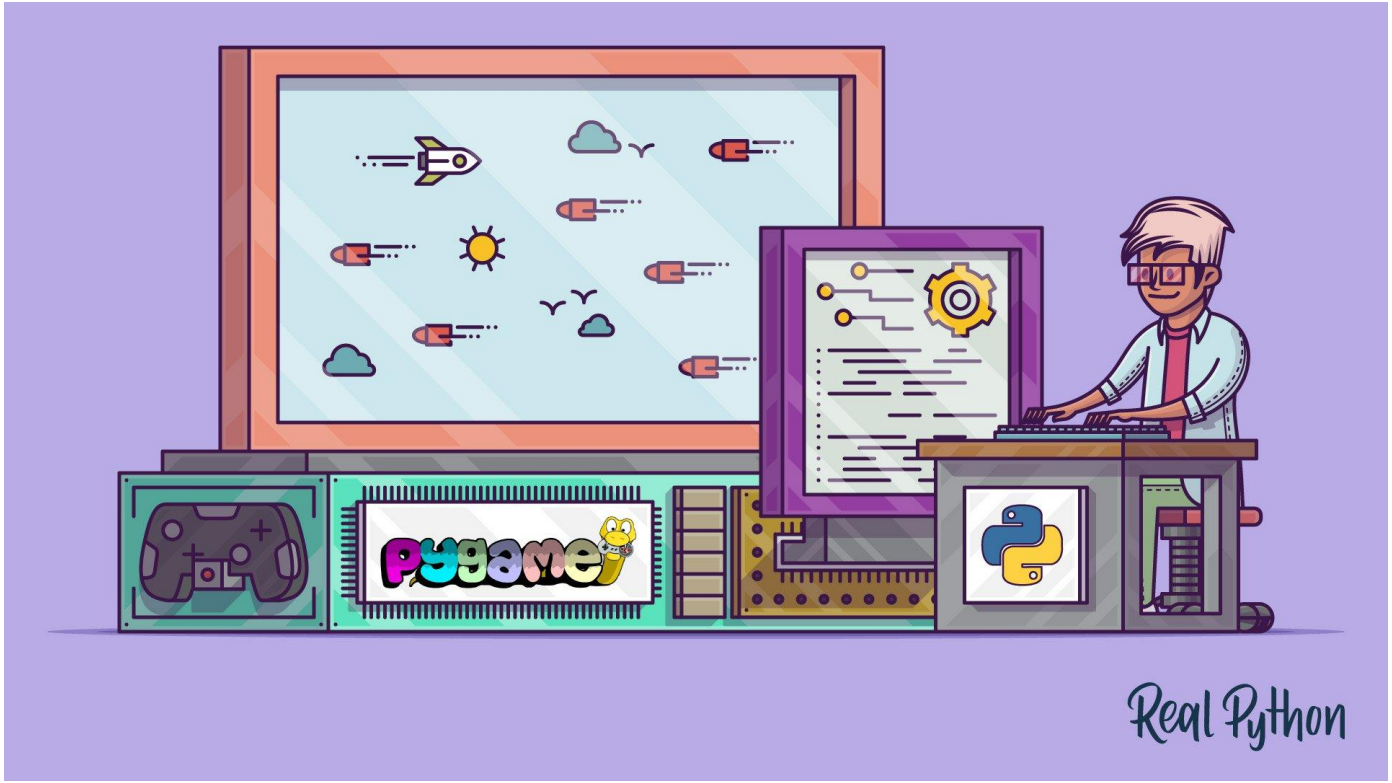
PyGame Primer



PyGame Primer

What will you learn how to do in PyGame?

- Draw items on your screen
- Play sound effects and music
- Handle user input
- Implement event loops
- Explore how game programming differs from standard procedural Python programming

PyGame Primer

Suggested background for this course

- User-defined functions
- Imports
- Loops
- Conditionals
- How to open files
- Basics of object-oriented Python

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

PYGAME BACKGROUND AND SETUP

What is PyGame?

- A Wrapper for the SDL library
  - Simple DirectMedia Layer
  - Cross-platform multimedia hardware support

PYGAME BACKGROUND AND SETUP

Installing PyGame on your platform

- Install using pip

```
$ python3 -m pip install pygame
```

- To verify the installation try out one of the examples

```
$ python3 -m pygame.examples.aliens
```

TABLE OF CONTENTS

BASIC PYGAME PROGRAM

First steps - aka the PyGame “Hello, World”

- Import and initialize the PyGame library
- Set up your display window
- Set up a game loop
- Scan and handle events
- Fill the window with a color
- Draw a circle
- Update the display window
- Exit PyGame

TABLE OF CONTENTS

PYGAME CONCEPTS

Initialization and Modules

- The PyGame library is composed of a number of modules
  - `display`
  - `joystick`
  - `music`
  - `key`
  - `event`
  - `image`

PYGAME CONCEPTS

Initialization and Modules

- Importing the PyGame library
  - `import pygame`
- Initializing PyGame
  - `pygame.init()`
  - Initializes all of the included PyGame modules

PYGAME CONCEPTS

Displays and Surfaces

- `Surface`
  - A class that defines a rectangular area on which you can draw
- `display`
  - Can be a window or full screen
  - Created using `.set_mode()`
  - Contents of a `Surface` are pushed to the display when `pygame.display.flip()` is called

PYGAME CONCEPTS

Images and Rects

- `image` module
  - Allows the loading and saving of images
  - Images are loaded into `Surface` objects and then manipulated
- `Rect` class
  - A special class for storing rectangular coordinates
  - Used for managing and moving around on-screen objects
  - Used to manage collisions

TABLE OF CONTENTS

TABLE OF CONTENTS

THE TUTORIAL GAME

Goals of the game

- Avoid the incoming obstacles
  - Player starts on left side of the screen
  - Obstacles enter randomly on the right and move left in a straight line
- The player can move left, right, up or down to avoid the obstacles
- The player cannot move off the screen
- The game ends either when the player is hit by an obstacle or when the user closes the window

THE TUTORIAL GAME

Not goals of the game

- No multiple lives
- No scorekeeping
- No player attack capabilities
- No advancing levels
- No boss characters

These are all excellent potential ways to explore growing your PyGame skills

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

THE GAME LOOP

Important things the game loop does

- Processes user input
- Updates the state of all game objects
- Updates the display and audio output
- Maintains the speed of the game

THE GAME LOOP

Every cycle of the game loop is called a frame

- The quicker you can do things each cycle, the faster your game will run
- Frames continue until some condition to exit the game is met
    - Your design will look for two conditions
        - The player collides with an obstacle
        - The player closes the window

PROCESSING EVENTS

Events are placed in an event queue

- Every event has an event type
  - Keypresses are event type `KEYDOWN`
  - Closing the window is event type `QUIT`
- To access the list of active events
  - `pygame.event.get()`

TABLE OF CONTENTS

DRAWING ON THE SCREEN

Two commands you have used so far

- `screen.fill()` to fill the background
- `pygame.draw.circle()` to draw a circle

DRAWING ON THE SCREEN

Now for a third way

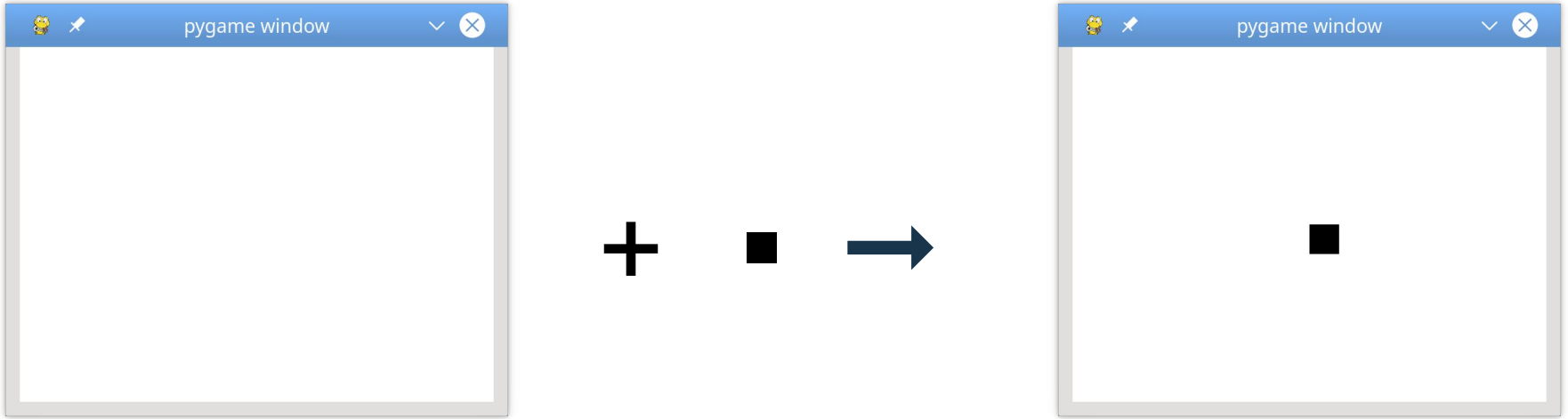
- You can use a `Surface`
  - A rectangular object on which you can draw
    - Like a blank sheet of paper
    - The `screen` object is a `Surface`
    - You will create several more for your game

TABLE OF CONTENTS

BLOCK TRANSFER

How do you get your new Surface to appear?

- `.blit()` stands for Block Transfer
  - Copies the contents of one `Surface` onto another `Surface`



BLOCK TRANSFER

How do you get your new Surface to appear?

- `.blit()` stands for Block Transfer
  - Copies the contents of one `Surface` onto another `Surface`
  - It takes two arguments
    - The `Surface` to draw
    - The location at which to draw it on the source `Surface`

UPDATING THE SCREEN

How to draw the changes?

- `pygame.display.flip()`
  - Updates the entire screen
  - Including everything that's been drawn since the previous flip

TABLE OF CONTENTS

THE TUTORIAL GAME

Goals of the game

- Avoid the incoming obstacles
  - Player starts on left side of the screen
  - Obstacles enter randomly on the right and move left in a straight line
- The player can move left, right, up or down to avoid the obstacles
- The player cannot move off the screen
- The game ends either when the player is hit by an obstacle or when the user closes the window

SPRITES

A 2D representation of something on the screen

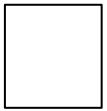
- PyGame provides a `Sprite` class
  - Advantages of it being a class
    - Creating multiple instances
    - Access to its built-in methods
    - Can be extended to allow new methods

RGB COLOR

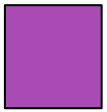
Computers represent everything as numbers

- RGB is a color model that converts colors into numbers that a computer can process
- RGB has 3 color channels
  - Red
  - Green
  - Blue
- The values per channel go from 0 to 255

Black
(**0**, **0**, **0**)



White
(**255**, **255**, **255**)



Cool Purple
(**170**, **74**, **181**)

RGB COLOR

Computers represent everything as numbers

- Google search - RGB to Hex

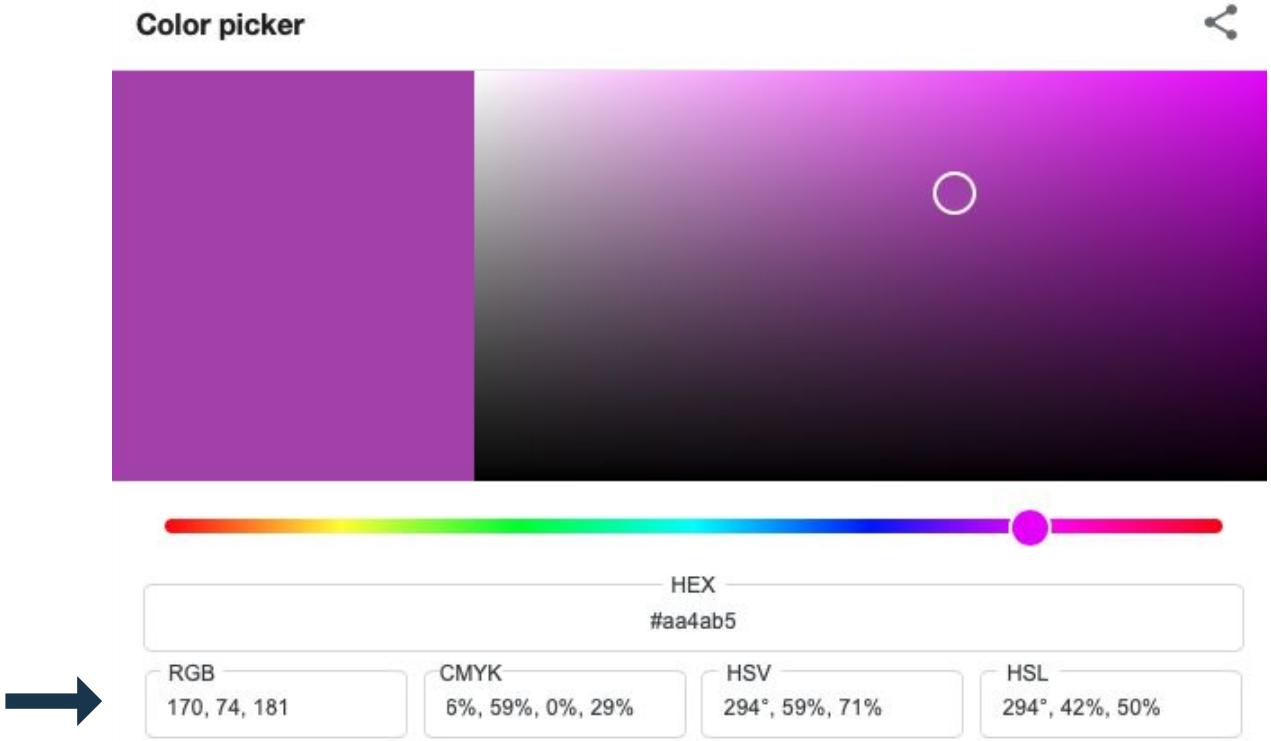


TABLE OF CONTENTS

USER INPUT

Adding keyboard control

- `pygame.event.get()`
  - Returns a list of the events in the event queue
  - Which you currently are scanning for KEYDOWN event types
- `pygame.event.get_pressed()`
  - Returns a dictionary containing all the KEYDOWN events in the queue
  - Put this in your game loop right after the event handling loop
    - This will return keys pressed at the beginning of every frame

USER INPUT

Adding an `update()` method to `Player`

- Defining movement behavior of the sprite based off the keys pressed
- Using `rect.move_ip()`

TABLE OF CONTENTS

STAYING ON THE SCREEN

You may have noticed two small problems

- The player rectangle can move off the screen
- The player can move very fast if a key is held down

TABLE OF CONTENTS

TABLE OF CONTENTS

SPRITE GROUPS

`pygame.sprite.Group()`

- A super useful class
  - Holds and manages multiple `Sprite` objects
  - Includes methods for collision detection
  - Makes updating positions and rendering sprites easier

TABLE OF CONTENTS

CUSTOM EVENTS

How to have enemies appear at regular intervals?

- The event loop looks for random events and deals with them appropriately
- What about events that are not defined by PyGame?
  - You define custom events to handle as you see fit
    - PyGame defines events internally as integers
    - The last event PyGame reserves is called `USEREVENT`
      - `pygame.USEREVENT + 1` will ensure its unique

CUSTOM EVENTS

What about timing?

- `pygame.time` a module for monitoring time
  - `pygame.time.set_timer()`
    - Called outside the game loop
    - Can fire custom events throughout the entire game

TABLE OF CONTENTS

COLLISION DETECTION

Checking for collisions can be difficult

- PyGame makes this non-trivial math problem of determining when two sprites overlap easy
  - PyGame has several collision detection methods available
  - `pygame.sprite.spritecollideany()`
    - Arguments accepted are a `Sprite` and a `Group`
    - Checks every object in the `Group` if its `.rect` intersects with the `.rect` of the `Sprite`
    - Returns `True` if detected, and `False` otherwise

TABLE OF CONTENTS

SPRITE IMAGES

PyGame module for loading images and transferring to a `Surface`

- `pygame.image.load("filename.png")`
  - Loads an image from the disk
  - Returns a `Surface`
- `.convert()`
  - Optimizes the `Surface`, making future `.blit()` calls faster
- `.set_colorkey()`
  - Used to indicate the color PyGame will render as transparent

TABLE OF CONTENTS

ADDING BACKGROUND IMAGES

Let's make some clouds

- Create the `Cloud` class
- Add an image of a cloud to it
- Create a method `.update()` that moves the `cloud` toward the left side of the screen
- Create a custom event and handler to create new `cloud` objects at a set time interval
- Add the newly created `cloud` objects to a new `Group` called `clouds`
- Update and draw the `clouds` in your game loop

TABLE OF CONTENTS

GAME SPEED

Adjusting the frame rate

- The `time` module contains a `Clock` object
  - `pygame.time.Clock()`
- The `Clock` object contains a method to limit the maximum frame rate
  - `.tick()` takes an argument of frames per second

TABLE OF CONTENTS

ADDING MUSIC

Using the pygame `mixer` module

- The `mixer` module handles all the sound-related activities
  - `pygame.mixer.init()` sets up the mixer for playback
  - `pygame.mixer.music` sub-module for music playback
    - `.load("filename.ogg")` supports .ogg and .Mod
    - `.play(loops=-1)`
    - `.stop()`
  - `pygame.mixer.quit()` stops all sound playback

TABLE OF CONTENTS

ADDING SOUND EFFECTS

Attaching sounds to specific events

- Create an instance of the `.Sound` class for each sound effect
  - `pygame.mixer.Sound("filename.ogg")`
    - Audio files can be `.wav` or `.ogg`
    - `.play()` triggers the sound effect
    - `.stop()`
    - `.set_volume()` takes a `float` in the range 0.0 to 1.0

TABLE OF CONTENTS

A NOTE ON SOURCES

Resources for your own games

- Here are some sources for music, sound, and art
  - OpenGameArt.org - sounds, sound effects, sprites, and other artwork
  - Kenney.nl - sounds, sound effects, sprites, and other artwork
  - Game Art 2D - sprites and other artwork
  - CC Mixter - sounds and sound effects
  - Freesound - sounds and sound effects

A NOTE ON SOURCES

Licenses and proper attribution

- Make sure to pay attention to the license requirements
  - Attribution
  - Link to the license
  - Different requirements for commercial release

TABLE OF CONTENTS

CONGRATULATIONS!

YOU COMPLETED THE COURSE

THANKS FOR WATCHING!

MAKE SURE YOU PRACTICE WITH WHAT YOU HAVE LEARNED