Introduction to Pygame

Introduction

- Pygame is widely used and open-source.
- It is a Python library.
- Primarily used for creating 2D video games.
- Also supports multimedia application development.
- Offers various tools for graphics, sound, input, and other game-related tasks.
- Suitable for both beginners and experienced developers.
- Can be used to create simple games, interactive simulations, or complex applications.
- Provides an overview of Pygame's key features.

Pygame Libraries

- **Open-Source and Free**: Pygame is a free and open-source Python library for game development [6].
- **Based on SDL**: It is built on top of the Simple DirectMedia Layer (SDL) library, allowing for efficient multimedia handling [6].
- **Cross-Platform**: Pygame works on multiple platforms, making it versatile for game developers [6].

Pygame Libraries

- **Rich Multimedia Features**: It includes tools for handling graphics, sound, and input, enabling the creation of comprehensive multimedia applications [6].
- **User-Friendly**: Designed to be easy to use, it is suitable for both beginners and experienced developers [1].
- **Tutorials and Documentation**: There are many tutorials and extensive documentation available to help users get started and advance their skills [5].
- **Versatile Use Cases**: Pygame can be used to create simple games, interactive simulations, and complex applications [3].

Pygame Libraries

The Pygame library is composed of several modules, each responsible for a specific aspect of game development. The most fundamental modules include:

Pygame.display

This module handles the creation and management of the game window, including setting the window size, title, and background color.

Pygame.draw

This module provides functions for drawing basic shapes, such as lines, rectangles, and circles, as well as more complex shapes and images.

Pygame.event

The event module is responsible for handling user input, such as keyboard and mouse events, allowing your game to respond to player interactions.

Pygame Libraries

- LGT (Lightweight Game Toolkit)
- pygsear
- PYZZLE
- Spineless
- SPyRE
- pyso
- PyUI2
- Mirra

- PyGL2D
- Popup Gui
- TextWidget
- pyscroll
- pyTMX
- pygame.gfxdraw
- pygame.image
- pygame.joystick

pygame.org - libraries

Importing and Initializing Modules

Before you can start using Pygame, you need to import the necessary modules and initialize the library. This is typically done at the beginning of your Python script:

1 **Importing Pygame**

Import the Pygame library using the following line of code:

```
import pygame
```

2 **Initializing Pygame**

Initialize the Pygame library by calling the `pygame.init()` function. This sets up the necessary resources and prepares Pygame for use in your application.

3 **Quit Pygame**

When you're done with your Pygame application, be sure to call `pygame.quit()` to properly shut down the library and free up any allocated resources.

realpython.com - PyGame: A Primer on Game Programming in Python

Window

One of the first tasks in any Pygame project is to create a window where your game or application will be displayed. The `pygame.display` module provides several functions to help you with this:

Set Window Size

Use the `pygame.display.set_mode()` function to specify the width and height of your window. For example, `screen = pygame.display.set_mode((800, 600))` will create a window with a resolution of 800x600 pixels.

Set Window Title

You can change the title of the window using the `pygame.display.set_caption()` function. This allows you to provide a meaningful name for your application, making it easier to identify.

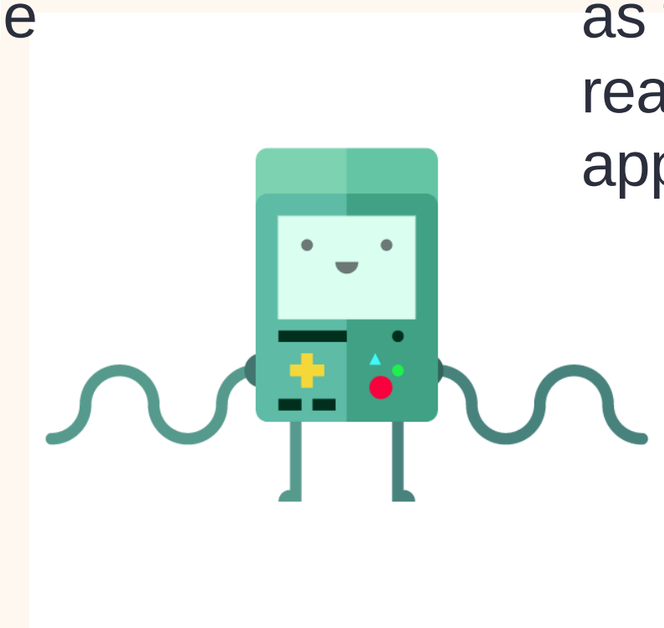
Update the Display

After making changes to the window or drawing objects, you need to update the display by calling `pygame.display.flip()` or `pygame.display.update()` to ensure the changes are visible to the player.

Creating the Window

To create a window in Pygame, you'll need to use the `pygame.display.set_mode()` function. This function takes the width and height of the window as arguments and returns a Surface object that represents the window:

1. The `screen` variable you assigned in the previous step now represents the game window and allows you to draw graphics, update the display, and handle user input.



2. It's important to create the window and perform any necessary initialization before entering the main game loop, as this ensures the window is ready for use throughout your application.

Setting the Window Size

The size of your game window is an important consideration, as it can impact the overall appearance and playability of your application. Pygame provides several ways to set the window size:

Fixed Size

As mentioned in the previous section, you can set a fixed window size using the `pygame.display.set_mode()` function, passing in the desired width and height as a tuple: `screen = pygame.display.set_mode((800, 600))`.

Fullscreen

To create a fullscreen window, you can pass the `pygame.FULLSCREEN` flag as the second argument to `set_mode()`: `screen = pygame.display.set_mode((0, 0), pygame.FULLSCREEN)`.

Resizing the Window

If you want to allow the user to resize the window, you can pass the `pygame.RESIZABLE` flag to `set_mode()`:

`screen = pygame.display.set_mode((800, 600), pygame.RESIZABLE)`.

1

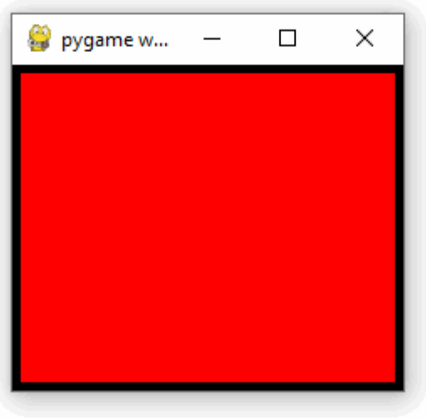
Detect Resize Event

In your main game loop, listen for the `pygame.VIDEORESIZE` event, which is triggered whenever the user resizes the window.

2

Get New Size

When the resize event is detected, you can retrieve the new window size using the `event.size` attribute, which returns the new width and height as a tuple.



3

Update Window

Finally, update the window size by calling `pygame.display.set_mode(event.size, pygame.RESIZABLE)` to reflect the new dimensions.

Changing the Window Name

In addition to setting the size and handling resizing, Pygame also allows you to change the title or name of the game window. This can be a useful way to provide context and branding for your application:

Set Window Name

Use the `pygame.display.set_caption()` function to set the window title. For example, `pygame.display.set_caption("My Awesome Game")` will change the window name to "My Awesome Game".

Update Window Name

You can update the window name at any time during your application's runtime, allowing you to provide dynamic titles or branding that change based on the game state or user interaction.

Setting the Background Color

The background color of your game window is an important aspect of the overall visual design and can greatly impact the player's experience. Pygame makes it easy to set the background color of the window:

Color Choices

Pygame uses the RGB color model, where each color is represented by a combination of red, green, and blue values ranging from 0 to 255. You can choose from a wide range of colors to set the background of your game window.

Setting the Background Color

Filling the Window

To set the background color, you can use the `screen.fill(color)` method, where `color` is a tuple or list containing the RGB values. For example, `screen.fill((255, 255, 255))` will set the background to white.

Updating the Display

After setting the background color, you need to call `pygame.display.flip()` or `pygame.display.update()` to ensure the changes are visible to the player.

Display Modes

In Pygame, display modes refer to the various ways you can set up the display window or screen. The display modes determine the size, type, and features of the window.

Display Modes

Basic Setup: The most common function used to set the display mode is pygame.display.set_mode(size, flags=0, depth=0). This function initializes the display and returns a surface object that represents the visible part of the window.

- size: A tuple specifying the width and height of the window (e.g., (800, 600)).
- flags: Optional flags that control the display mode (e.g., pygame.FULLSCREEN).
- depth: Optional parameter to set the color depth (in bits per pixel).

Windowed and Fullscreen Modes: Pygame supports both windowed and fullscreen modes. By default, the display is windowed. You can switch to fullscreen mode using the pygame.FULLSCREEN flag.

pygame.org - Setting Display Modes

Colour Object

RGB Values

Pygame allows you to create and manipulate colors using RGB (Red, Green, Blue) values, which range from 0 to 255 for each component.

Predefined Colors

Pygame also includes a variety of predefined color names, such as "red", "green", and "blue", which you can use to quickly set the color of your objects.

Blending Colors

By mixing different amounts of red, green, and blue, you can create a wide range of custom colors to suit your game's aesthetic.

Event Objects

1 **Event Capture**

Pygame can detect and handle various events, such as mouse clicks, key presses, and window resizing, allowing you to create interactive and responsive games.

2 **Event Queues**

Pygame maintains an event queue, which stores all the events that occur during the game loop, so you can process them in the order they were received.

3 **Event Types**

Pygame provides a wide range of event types, including QUIT, MOUSEBUTTONDOWN, KEYDOWN, and more, which you can use to handle different user interactions.

Keyboard events

1 **Key Detection**

Pygame allows you to detect when specific keys are pressed, released, or held down, enabling you to create complex and responsive keyboard controls.

2 **Modifier Keys**

Pygame can also detect modifier keys like Shift, Ctrl, and Alt, which you can use to create additional functionality or shortcuts in your game.

3 **Key States**

Pygame provides methods to check the current state of a key, whether it's pressed, released, or held down, allowing you to create seamless and responsive controls.

Mouse events

1 **Mouse Position**

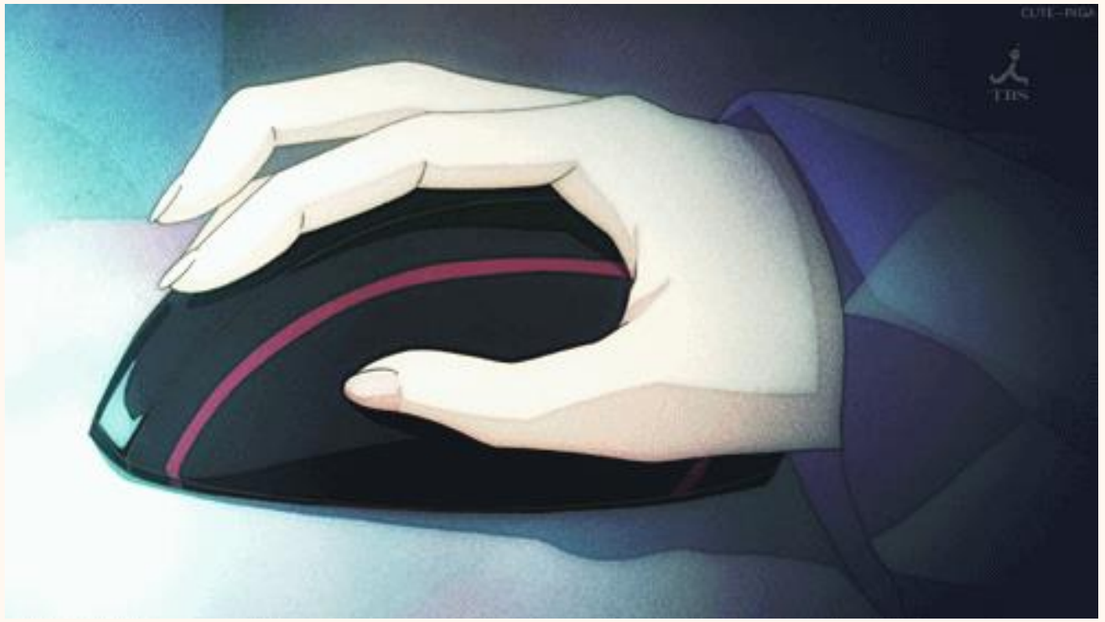
Pygame can retrieve the current position of the mouse cursor, enabling you to create games that respond to the user's mouse movements.

2 **Mouse Buttons**

Pygame can detect when the user clicks or releases the left, middle, or right mouse buttons, allowing you to create point-and-click interactions.

3 **Mouse Wheel**

Pygame can also detect mouse wheel scrolling, which you can use to implement features like zooming or scrolling in your game.



Drawing Shapes

Rectangles

Pygame allows you to draw filled or outlined rectangles of any size and position on the screen.

Circles

You can also draw filled or outlined circles, with customizable radius and position.

Polygons

Pygame supports the creation of polygons with any number of sides, allowing you to draw complex shapes.

Lines

Pygame enables you to draw straight lines between any two points on the screen.

Load image

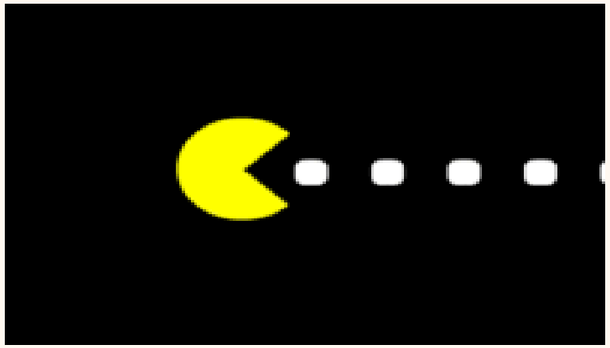
1 2 3

Image Loading

Pygame can load and display various image formats, including PNG, JPG, and BMP, enabling you to incorporate visuals into your games.

Image Manipulation

Once loaded, you can manipulate the image, such as resizing, rotating, or blending it with other elements on the screen.



Transparent Images

Pygame supports transparent images, allowing you to create games with complex backgrounds and overlapping elements.

Displaying a Text in Window

Font Selection	Pygame allows you to choose from a variety of built-in font styles or load custom fonts, enabling you to create unique text displays.
Text Rendering	Pygame can render text on the screen with customizable size, color, and positioning, making it easy to add text-based elements to your games.
Text Animation	By dynamically updating the text's position, color, or size, you can create scrolling, blinking, or other animated text effects.

Moving an Image - with Numeric Key Pads

Key Detection

Pygame can detect when the user presses specific keys on the numeric keypad, allowing you to move an image in the corresponding direction.

1

Boundary Detection

Pygame can also check if the image is reaching the edge of the screen, and adjust its movement accordingly to prevent it from going off-screen.

3

Image Position

By updating the image's position based on the key press, you can create smooth and responsive movement of the image on the screen.

2

Moving an Image - Mouse

1 2 3

Mouse Position

Pygame can retrieve the current position of the mouse cursor, which you can use to update the position of an image on the screen.

Mouse Events

By combining mouse position tracking with mouse event handling, you can create games where the user can click and drag an image to move it around.

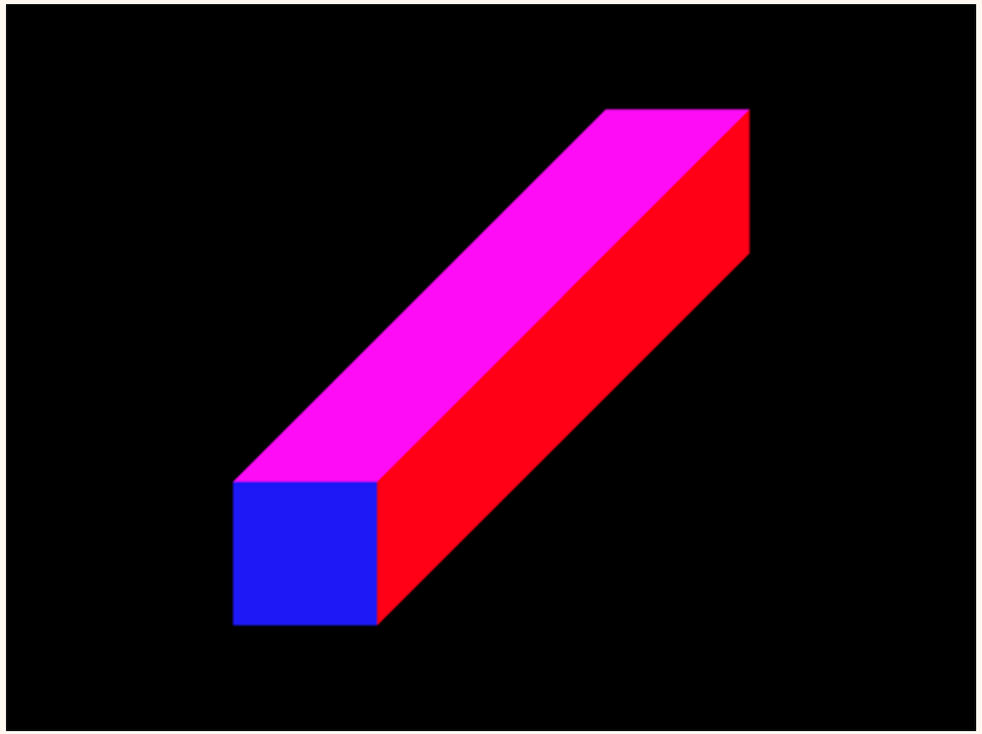
Cursor Change

Pygame also allows you to change the appearance of the mouse cursor, such as displaying a custom icon or making it invisible, to enhance the user experience.

Moving Rectangular Objects

1 **Tracking Position**

Use Pygame's coordinate system to track the location of the rectangular object on the screen.



2 **Responding to Input**

Detect user input, such as keyboard or mouse events, to control the movement of the object.

3 **Updating the Display**

Continuously redraw the object at its new position to create the illusion of movement.

Text as Buttons

Button Creation

Use Pygame's font and text rendering capabilities to create interactive text-based buttons.

Event Handling

Detect when the user clicks or hovers over a button and trigger the desired action.

Visual Feedback

Provide visual cues, such as color changes or animations, to indicate the button's state.

Transforming Images

1 **Load Image**

Use Pygame's image loading functions to import images into your project.

2 **Apply Transformations**

Scale, rotate, or flip the images to create dynamic and interactive visuals.

3 **Render Transformed Images**

Continuously update the display to show the transformed images in real-time.

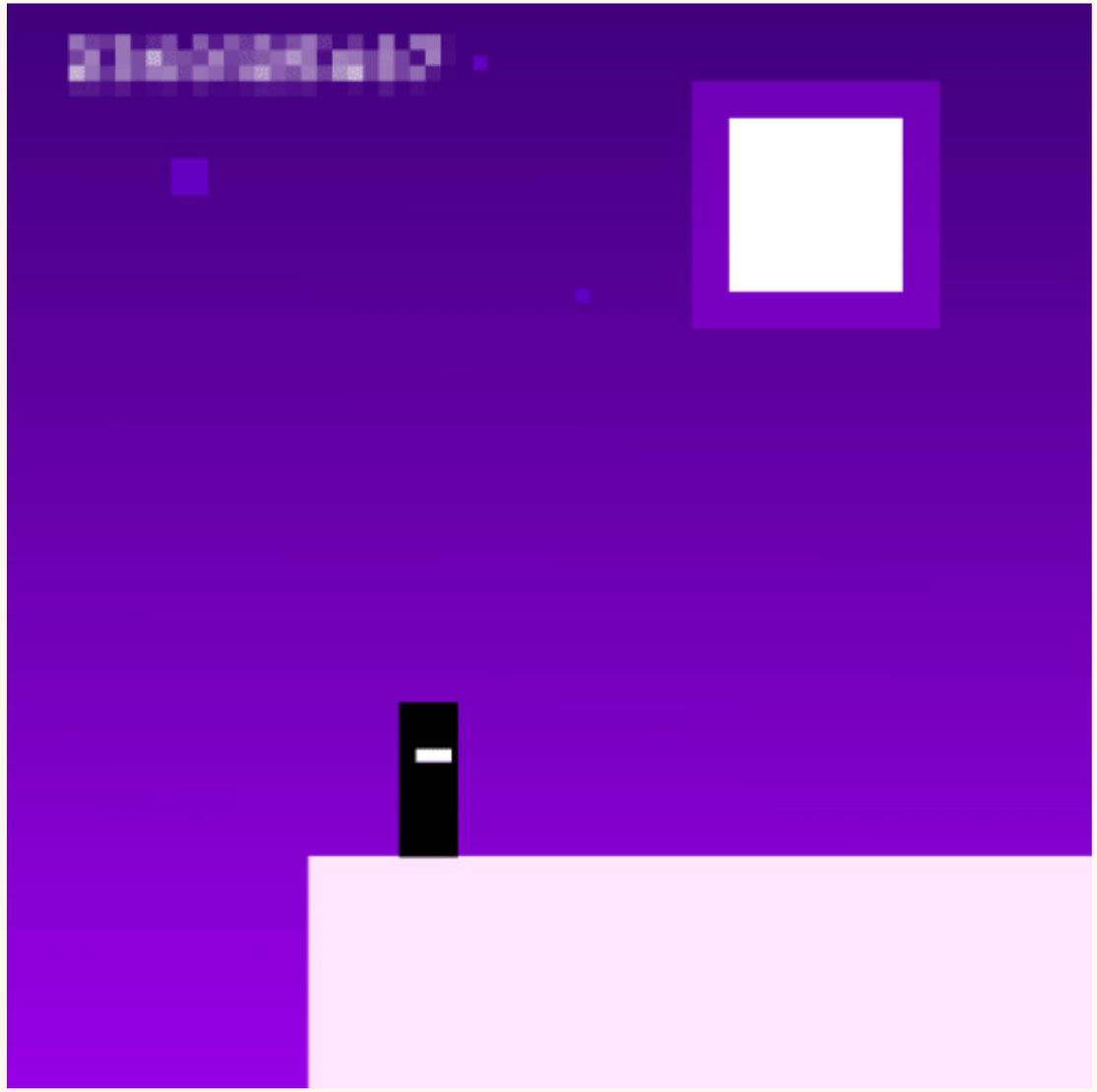
Game Loop Fundamentals

Continuous Execution

The game loop runs repeatedly, ensuring your game or application remains active and responsive.

Updating Game State

Update the positions, attributes, and behaviors of game objects based on the user's actions.



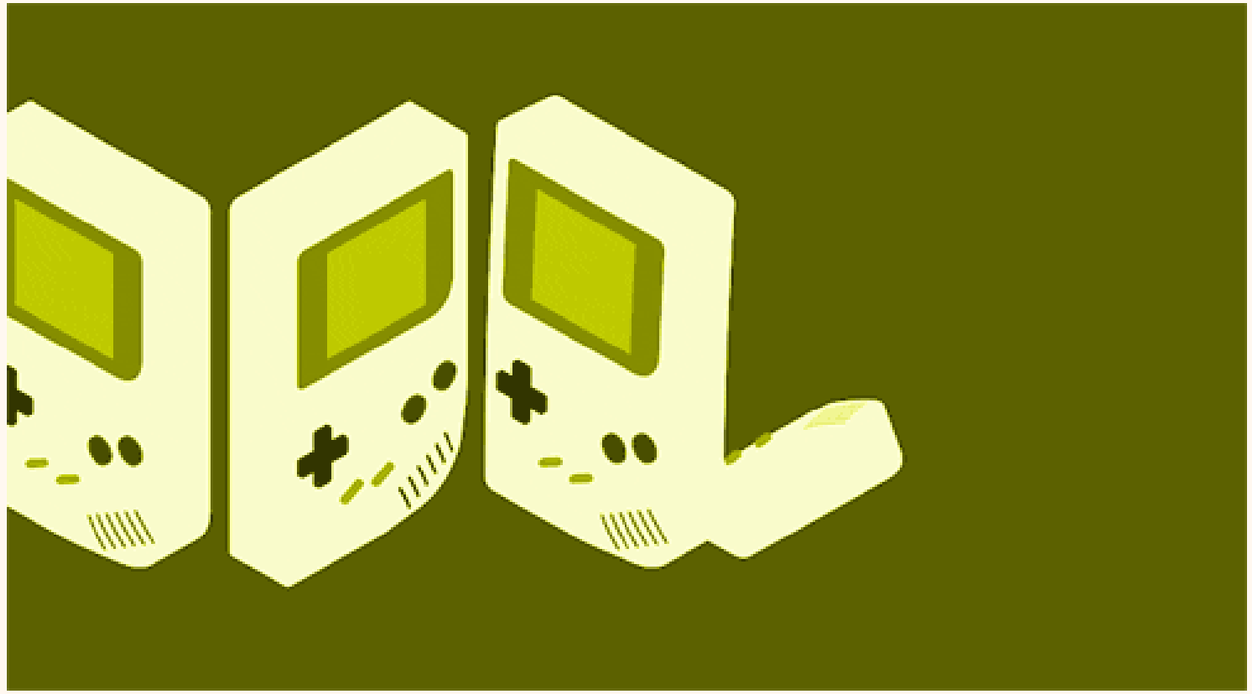
Game Loop Fundamentals

Handling Events

Within the game loop, you'll process user input, such as keyboard or mouse events.

Rendering the Frame

Redraw the game scene with the updated positions and transformations to create the illusion of motion.



Creating Surfaces

Surface Creation

Use Pygame's Surface class to create a canvas for your game or application.

Loading Images

Load and blit images onto the Surface to create visually appealing graphics.

Background Color

Set a background color for the Surface to provide a solid foundation for your visuals.

Blitting

Efficiently draw and update elements on the Surface using Pygame's blit function.

Timing and Animation

1 **Pygame Clock**

Use the Pygame Clock object to track time and control the frame rate of your game.

2 **Time Calculations**

Determine the time elapsed between frames to update game objects and create smooth animations.

3 **Animated Sequences**

Leverage the time-based updates to create animated sequences, such as sprite sheet animations.

Cursor Handling

Cursor Customization

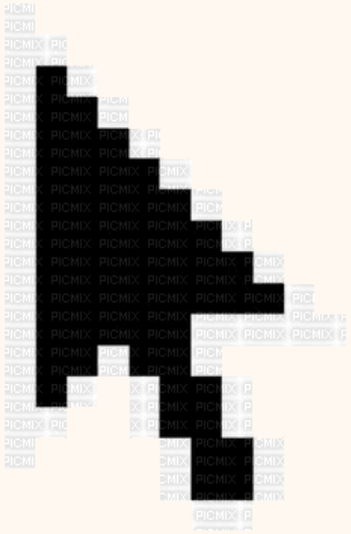
Use Pygame's cursor functions to change the appearance of the mouse cursor in your game or application.

Cursor Tracking

Monitor the cursor's position and respond to user interactions, such as clicks or hover events.

Cursor Visibility

Control the visibility of the cursor, hiding it when necessary to create a more immersive experience.



Error Handling

1

Try-Except Blocks

Wrap your code in try-except blocks to catch and handle common exceptions, such as file not found or unsupported image formats.

2

Logging and Debugging

Use Pygame's logging functions to track errors and provide valuable information for troubleshooting and debugging your code.

3

Error Messages

Display meaningful error messages to the user, guiding them towards a resolution or providing instructions for reporting the issue.

Introducing PyOpenGL

OpenGL Elements	Description
Vertices	The basic building blocks of 3D graphics, used to define the shape of objects.
Primitives	The geometric shapes, such as points, lines, and triangles, that are used to construct 3D models.
Transformation Matrices	Matrices used to apply scaling, rotation, and translation to 3D objects.
Lighting	Controls the illumination of 3D scenes, creating realistic shadows and highlights.