Pygame

What is Pygame?

Pygame is a library, or a set of modules designed to write and simplify game development in Python. Its main role is to simplify and handle sound, graphic drawing, inputs like mouse and keyboard, and other functionality. The advantages and disadvantages of using pygame compared to other game development libraries or frameworks are listed below.

Advantages of Pygame:

- Runs on nearly every platform and operating system.
- Easy to learn and develop for beginners and skilled developers.
- Quick and easy to create and display graphics, handle inputs, simple collision detection, and audio.
- Large community to learn and communicate problems.
- Better for smaller game projects.

Disadvantages of Pygame:

- Difficulties handling three-dimensional graphics and development.
- Slower performance due to using Python language and large complex libraries.
- Not compatible with the web meaning the game can only be hosted locally on the machine.
- Physics engine capabilities are limited because collision and detection between sprites are simplified.

Now, to start our program, you must have the Python interpreter installed, pip or package installer for Python, preferably a code editor like Visual Studio Code and install the library using this command: "***python 3 –m pip install –U pygame –user***"

Core Concepts of Pygame:

There are multiple concepts within Pygame library, some that were already discussed in the previous section. Some of the key concepts that will be discussed are surface or graphics, sprites, events, game loop, collisions, physics, game mechanics/player movement, and sound.

What is a Surface?

The surface is the initial window of the game. Since Pygame is a desktop local run application, we would have to create a window with parameters like width, height, depth, and mask. However, the main parameters are specifying just width and height to size the window. We could include other graphics as well such as using their draw function to draw a rectangle with size (width, height) and positioning (x, y). We could also load images by including the directory it is in and displaying it onto the surface with positioning (x, y). The general overview of this section is that most graphics can be either drawn or loaded onto the surface with a positioning aspect. There are multiple other functions to draw and display different things on the surface where documentation would probably be better to look at as a developer.

What is a Sprite?

A sprite can be known as image/object/animation of a character that is able to be modified by different events such as collision. An example of a spite could be Pac-Man, it is an animated graphic that is able to interact with the ghosts in the game. The easiest way in Pygame to implement sprites is to create a class and give it features that it should have. Going back to our example, you would have to give the Pac-Man sprite a color, size, and set movement speeds. This sprite would have to then be drawn onto the screen and looped to check for specific events like when Pac-Man collects points on the screen.

What is an Event?

An event is an action that a user performs to achieve a specific result. The collection of events that a user performs all goes into something known as the event queue, which as its name implies goes off on a first in, first out basis. That means that whichever event the user performs first will be the one that gets processed and carried out first as well. Every element in the event queue is associated with an integer that represents what type of event it is.

What Event types are there?

The two main types of events are Keyboard events and Mouse events. With regards to keyboard events, the actions associated with them are “KEYDOWN” (pressing the key) and “KEYUP” (releasing the key). The most common keys are represented by a pre-defined integer followed by an underscore and then the key in question. Mouse events can, just like with keyboard events, be split up; however, in this case it will be “MOUSEBUTTONDOWN” (clicking the mouse button) and “MOUSEBUTTONUP” (releasing the mouse button).

How to implement player movement or game mechanics?

Pygame has a detection mechanism for when a key is pressed on the keyboard. The "pygame.KEYDOWN" and "pygame.KEYUP" functions are related to the event queue and will keep track of the keyboard related events. To determine which exact key was pressed, check the variable associated with that keyboard event. For Pygame specifically, the key for the letter "a" can be represented with a letter K, followed by an underscore and then the letter (K_a). Comparing the player's input to the pre-established list to check from will indicate to the program what key was pressed. For clicking buttons on a mouse, "MOUSEBUTTONDOWN" and "MOUSEBUTTONUP" are the functions built in. The attributes of the event objects can be used to check whether the input was a right click (button value will be 2), left click (button value will be 1), or the middle button was clicked (button value will be 3). Button values of 4 and 5 will additionally correspond to the mouse scrolling up and down. Having code flow that keeps the input of all these events in mind, movement can be implemented into a game. Creating a display surface object along with loading a player's sprite will allow for several factors to be considered such as speed and coordinates the sprite should be moved to, given the type of event that comes out of the queue along with what details it provides on that front. Utilizing Pygame documentary, simple movement mechanisms can be incorporated by taking this information and intertwining it with the features possible in the game.

What are Collisions & Physics in Pygame?

The collisions and physics are what an object does if it touches another object, and physics includes gravity or like when an object moves across space or the surface/screen. However, in Pygame, all the objects collision and physics are mathematically placed on the surface. Therefore, we would have to calculate the objects' velocity and positioning to display onto the screen. We would also need to calculate the position of the object to see any collision with other objects. Ultimately, when it comes to collisions and physics in Pygame, there will be many complex mathematical computations to determine speed and trajectory. A physics engine can be simpler with less calculations depending on the specifications and requirements of the game.

What is a Game Loop?

A game loop is simply just so the game keeps running up and updating the screen. A surface is a static screen for one part of the game, so a game loop is used to read events and inputs to update the surface. To perform a game loop, the developer just has to make a while true condition statement. The true condition statement just must be true for the game to keep running until the user decides to quit or exit.

How to implement audio and music in pygame?

A musical is crucial to a game as it can amplify a player's feeling and contributes better feedback in a game performance. Fortunately, Pygame makes audio simple to play and has adjustable settings. A developer must first import a mixer from Pygame and use functions like music load, set volume, and music play to allow audio to play throughout the game. This mixer/audio can also have other built-in functions that we would recommend following documentation to perform different actions like pausing and stopping.

Work Cited

https://www.pygame.org/wiki/about
https://www.geeksforgeeks.org/pygame-tutorial/
https://www.reddit.com/r/gamedev/comments/1g1xzd/pygame_proscons/
https://opensource.com/article/17/12/game-python-moving-player
http://www.codingwithruss.com/pygame/how-to-get-mouse-input-in-pygame/