DROP!

CPE 202L

OBJECT ORIENTED PROGRAMMING

GROUP 2A

Prepared By:

Aguilar, Paul Christian D.

De Jesus, Ronie P.

Pagtalunan, Jonas D.

Santos, Kristian A.

BSCpE-2B

December, 2019

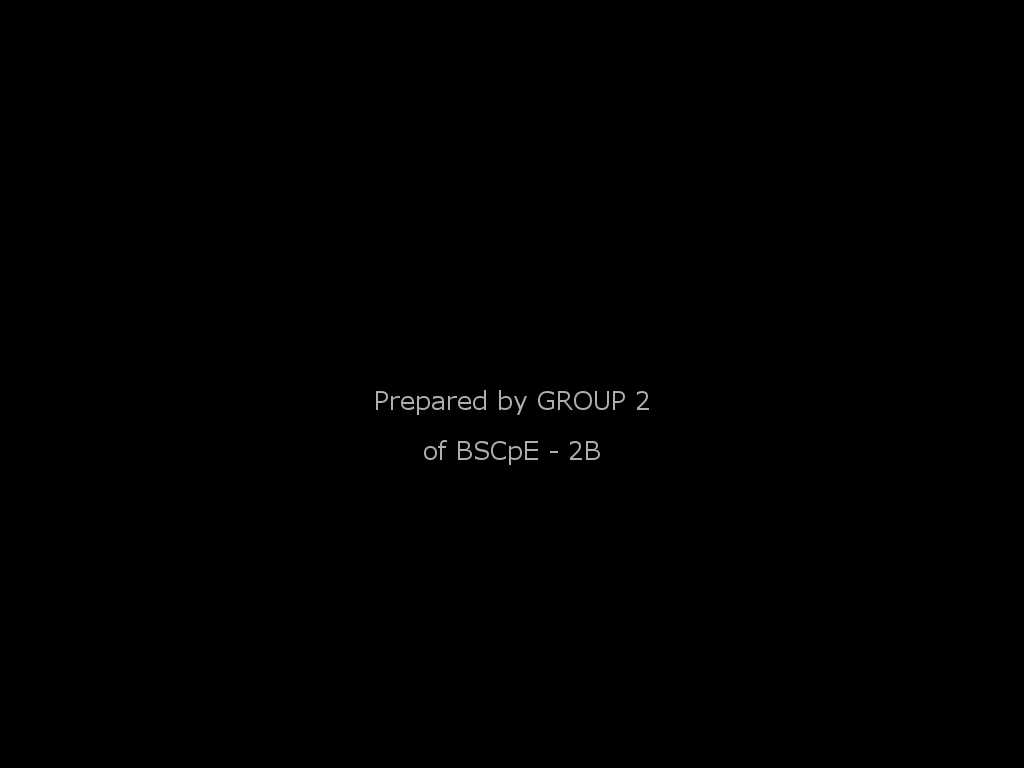
INTRODUCTION

DROP! is a game developed in Python using the Pygame library that has one goal: Get the astronaut down the platforms as fast as you can and avoid the spikes at the top of the screen. There is only one game mode: Endless. The longer the play time, the faster the platforms will go up which increases the chance of getting impaled by the spikes. There are 2 powerups: Teleport and Slowdown Platforms. Teleport powerup instantly moves the player position one platform down while the slowdown platforms powerup halves the speed of the platforms.

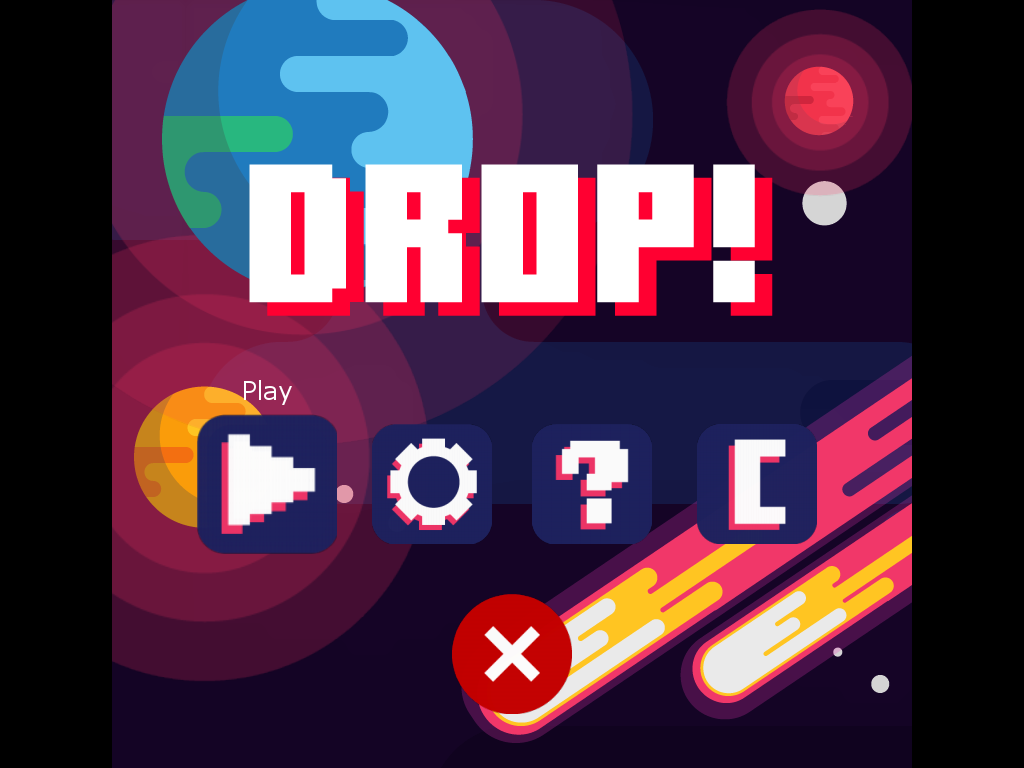
It is a program that accepts input from the mouse and keyboard. For the mouse (left click), it is used to select an icon that a player wants whether it is the Play Button, Credits Button, Settings Button, and the Exit Button. As for the keyboard, it is used to control the astronaut in the game loop. The “left” and “right” directional button is used for the astronaut to move and go left and right, the “R” button is used to restart the game if the game is over, the “Space” button is used to pause the game, and “Esc” button is used to exit and go to the main menu of the game.

The application produces multimedia features and implementations as a result of the inputs of the user. For every valid input from the user, the screen updates and display the corresponding visual output to the operators. The program also includes sound effects and background music to improve the multimedia experience of the user. These sound effects and animations are created whenever the inputs of the handler satisfy the conditions needed. An example of this is when the player dies in the game. It can react interactively through animations and button reactions to make it user-friendly. With all of these, the visual outputs formed on the screen are linked with its functions to make it understandable for the operator like the symbols of the main menu buttons.

MANUAL FOR DROP! GAME

 The title of the application is “DROP!”. This tells the main goal of the game which is to move downward as much as the players can to continue playing the game. To know more about the software, the following manual will explain all the details regarding the game.

Upon launching the application, a short credits animation will show. This consists of the number and section of the developers, a group of graphics designers and their roles, and a list of logic programmers and their functions. The credits part acknowledges the developers and their efforts for the creation of the application. In this part, the program will not entertain any input from the user aside from the “escape” button on the keyboard. This command will stop the credits animation without finishing everything in its iteration. After displaying all the contributors to the game, the animation will end, and a new interface will show up.



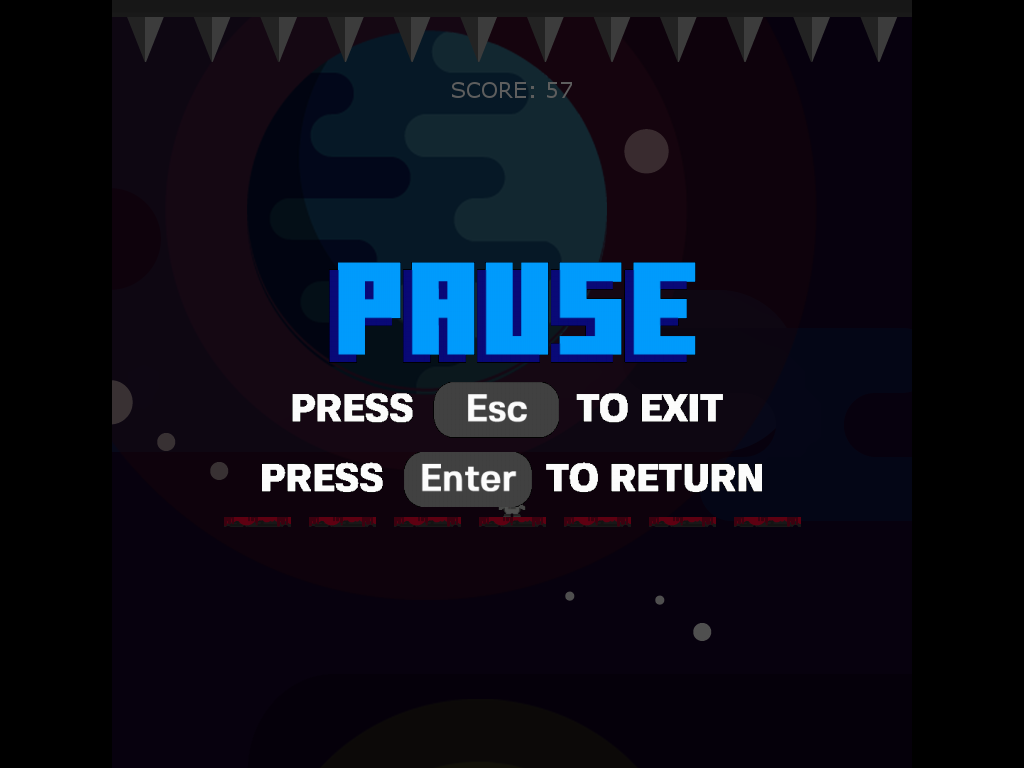
Once the introduction finishes, the main menu will be available for the users. The picture above illustrates the title of the game which is “DROP!”, and a set of buttons for the interface. In this portion, the program can accept inputs from the users by clicking a button on the screen. Each button responses to the commands of the user in two ways. First is when the mouse pointer hovers above the button will display its function. The other one is if the operator clicks above the button, then its designated purpose will occur, and these would be discussed one-by-one below.

These represent the play icon which means play will begin the game part of the program, the gear icon or settings will provoke the settings interface, the question mark shows the instructions of the game, the “C” button again triggers the credits or acknowledgment animation of the lists of game developers, and the “X” button will exit the application.



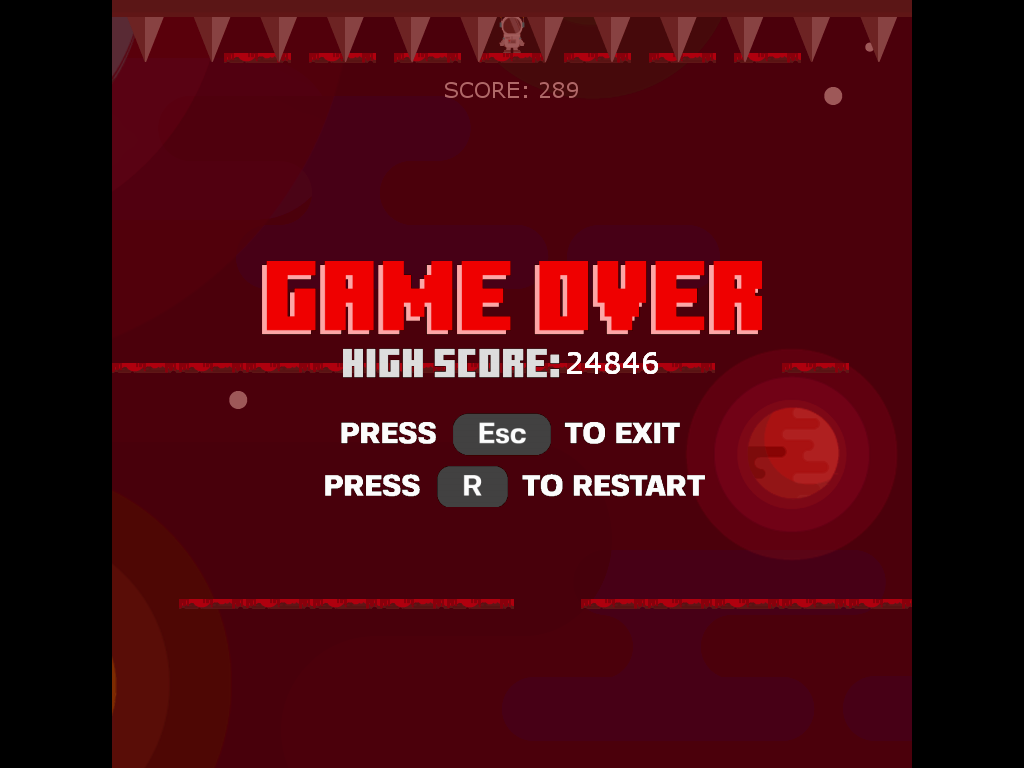
In case the user clicked on the play button on the main menu, the program will change the display and run the game logic. But before the interface in the picture above displays, an initializing animation of “READY”, “SET”, and, “DROP” animation will show up and the application will not accept any inputs. After this portion, the game immediately begins, and the user can start playing it.

The user can click vector buttons on the keyboard (left, right) to lead the player across the screen. It will not take inputs up and down since the program does not provoke vertical movements. This is because the goal of the user is to travel down as much as possible while generated platforms move upward towards the spike on top of the screen. These spikes will kill the player upon contact so to remain playing, the operator should lead the player left and right to proceed on the platforms farther from the spikes. While the player moves down the screen, the score will increase. As time passes the difficulty increases and platforms rise faster and faster.



Aside from directional keyboard buttons, the program accepts inputs using the space bar button during the game loop. The operator can click the space bar at any time of the game if the player is still alive.

 If the user clicked the space bar on the keyboard, the Pause interface will show up and interrupt the game. A translucent screen will display while keeping the game frozen at the back. The progress reached by the operator will halt and can be available if he or she wishes to continue by pressing the enter button of the keyboard and it will return to the interrupted game loop with the user data. But if the escape button on the keyboard is pressed while on the pause screen, the program will go back to the main menu and the game data will be lost.

Another input the program can receive during the game loop is the escape button on the keyboard. When clicked, a translucent screen will occur that asks if the user wants to exit like in the picture above. While this disruption is active, the game will freeze, and the progress data will remain in case the operator wants to go back to the game. The user can do so by pressing the enter button on the keyboard. This way, the user can continue the game. But if the user pressed the escape button when this menu shows up, then the program will redirect to the main menu of the application and progress data will be deleted. The users cannot input anything else to the program aside from the designated buttons for the pause and game exit menus.

In the event the player touches the spikes on the top of the screen, the game loop will stop. When this happens, a translucent interrupt screen will display showing the game is over and the current high score on the duration of the application is running. While the game over interface is available, the operator can choose whether to restart or to revert to the main menu. If the user clicked the escape button on the keyboard, the program will show the main menu. But if the “r” button on the keyboard is clicked, the game data will reset and begin a new game loop.

 The user can track his or her progress through the high score display of the game over screen on the duration of the application is running, and if it is restarted also. Only the highest score among all the complete game loops will be displayed on this screen.

Going back to the main menu interface, in case the user presses the gear button to access the settings of the game, the options interface will be displayed on the screen. The picture above shows what will happen when the operators clicked the gear button. In this part of the program, the user can change the settings of the game. The available setting is turning on and off the sounds of the application.

To change the status of the music setting of the application, click above the on and off button. The interactive button will show if the music is on or off so the user can remove the music if he or she wishes. Users can exit the settings menu by clicking the red “x” button on the upper right corner of its interface. The settings menu can only be accessed in the main menu screen.

Another menu that users can access the main menu is the “HOW TO PLAY” portion of the application. Game operators can open the instructions menu by pressing the question mark “?” icon on the main menu. This can help new users learn the mechanics of the game, learn its features and how to play better. The operators of the application will not need to input anything in this menu except when he or she wishes to exit. Users can exit this menu by clicking the red “x” button on the upper right corner of the screen.

For the credits button “C” on the main menu, the users of the game can access the credits animation of the program. Like on the launch of the application, it will not respond to any input from the user until the animation ends other than the “escape” button on the keyboard. The escape button can exit the credits animation without finishing it. This acknowledgment portion tells the operators who are the contributors to the game in case they missed information at the beginning of the program.

There are five buttons on the main menu screen and the fifth one is the red “x” button. The button is located on the bottom center of the main menu interface. This can invoke the application to end and exit it.

To better understand the flow of the program:

**DEVELOPMENT**

The idea of “DROP!” came from Melissa, a member of the graphics team. She said that it’s from an old Nokia phone game and is not popular today which made it suitable as a game to build.

The team initially is not knowledgeable of the library known as “Pygame” which made them watch tutorials on the internet on how to build a game using the said library.

Kidscancode offers a tutorial on how to build a platformer game. He taught how to build a simple game of jumping across different platforms. From his part 5 tutorial, the game logic programmers started modifying the concept from upward jumping to moving left and right to fall down the platforms (that continuously float upwards.)

The programmers divided up different screens of different functions into modules, each screen loop now has a specified module for easier access and modification.

The module named ***main.py*** contains the main game loop and the core program logic to make the game work. Inside of it contains a Game class which has different methods: new() which creates an instance of the game screen, run() which runs the game loop, events() which checks for user keyboard and mouse input, update() which checks for any changes in the attributes of the Game class and draw() which displays the graphics and logic to the window/screen. This is the script that should be run initially.

Another module named ***main\_menu.py*** handles most of the navigation across different screens: Play (which points towards the *main* module which would start gameplay), Settings (this is where you may disable in-game music but not the sound effects), How-to-play screen which outputs the instructions on how to play the game and access the credits screen. This is the only module where users can exit the game. It consists of similar scene looping functions with the main module: new(), run(), events(), update() and draw(). These scene functions act similarly as the scene functions found in the main module. Different classes that make up the different components of the main menu screen and are assigned inside the attributes of the MainMenu class. There is a class for the Logo, Planets and their animated glow, buttons and the MainMenu (which is a class) scene itself that handles different animation for better display. ***Main.py***and ***main\_menu.py*** are the two vital modules that handle the general events of the game.

Like the first module, **main\_menu.py** has initialized attributes in the \_\_init\_\_() function that describes the appearance of different objects in the main menu screen inside the general class. This includes the sounds, background display, screen objects, and their respective animations, and others. After declaring the default attributes of the objects of the module, the run() function dictates the general codes for it like music loop and calling looping other functions. Next is the update() function which renews the display on the screen according to the designated properties in it. The draw() function contains the implementation of display objects, buttons, and animation which makes up the screen. It also has the interactive response of the buttons when the mouse pointer hovers above them and the music options when it is designated to play. The last function of this class is the event() which catches the inputs of the user when in the main menu screen. Since this is the main menu, it will navigate to other modules according to the input of the user.

The second class in this module is Logo which assigns the attributes of the logo. PlanetVenus class manipulates the planet properties to display to animate its appearance. This class has functions that make its glow move. It is also the same as the PlanetEarth and PlanetMars class since they only manipulate other objects with similar attributes. The main menu buttons also have their properties and they have their separate class to assign the characteristics of these buttons when they are hovered over or not. The class also has the SettingsScreen and HowToPlay that has the traits of the settings and howtoplay modules, respectively. Finally, the main menu screen also has the comets class that controls the attributes of these objects.

**Bin Package:**

**NON-SCENE MODULES:**

***\_\_init\_.py*** is used to tell Python that the entire directory that this script is contained in a package ready to be used by the main module outside the directory.

**Settings.py** is where the repeatedly used static functions and constant variables used across different modules are contained.

**Sprite.py** is where the sprites are defined for them to be used inside the game using classes and objects. These are the entities that exist inside the game: Platforms, players, powerups and the spike. Each entity has a separate class. To change images for the player sprite when its standing and when it moves, a part of StevePaget’s pygame\_function module was used which uses gif spritesheets instead of loading individual images.

**ADDITIONAL SCENE MODULES:**

The **confirm\_exit.py** module runs when the user pressed the “escape” button while playing the game. It interrupts the gameplay and displays a translucent screen asking if the user wishes to discontinue the game or not through keyboard input (ESCAPE to go back to the game, RETURN to continue playing). With this, the module contains initialized attributes that make up the **confirm\_exit.py** screen. It also changes the status of the gameplay through the new() function to freeze. While the run() function organizes the order of the specified functions to run. Then, the update() function handles the animation of the display with preset attributes of the module. The draw() function arranges the appearance of the display before updating. Lastly, the events() and the wait\_key\_event\_pause\_screen() functions process the inputs of the user according to the allowable inputs in them.

**Credits.py** is responsible for displaying the acknowledgements of the people who worked on building the game. It works similarly with other scene modules: new(), run(), event(), update() and new() functions are also found inside. This is executed if the user presses the “C” button in the main menu screen using the left mouse button.

**Pause\_screen.py** is responsible for displaying the pause sequence of the game. It is executed once the user presses the SPACEBAR button when playing the game. It is similar with the other scene modules and this one has a “wait” function to freeze the screen (if it is stuck in a wait loop function the screen freezes because there are no updates and draw function that executes.) Once the user inputs the

After building the package and its modules, the logic programmers had to ask for the graphics programmers to insert their graphics inside the game. There were a few bugs that occurred, especially the “hollowed platforms” bug where the player would go through the platforms and ignores all kinds of collisions. This was only fixed by settings the players y-position back to 20 pixels upwards then the player sprite had enough time to detect the platforms again.

Graphics bugs were present the moment the graphics team implemented their work, the majority were scale bugs which were fixed by the game logic team by Pygame’s built-in image function: “smoothscale”. The game logic team also made the animated main menu glow for the planets and the scrolling background.

**REFERENCES:**

**FOR SOUNDS:**

[FUSEHIVE – AAA Audio for Video Games]. (2017, November 12). *INTERFACE SELECT / BUTTON CLICK SFX - Daily FREE UI Sound Effect Download* [Video File]. Retrieved from <https://www.youtube.com/watch?v=G4awm69_CpA>

[James Yevon]. (2019, February 2). *Pokemon Mystery Dungeon Explorers Sound Effects- Part 37* [Video File]. Retrieved from: <https://www.youtube.com/watch?v=oxmLPJdibSI>

[Shinx]. (2015, February 20). *036 - Amp Plains - (Pokémon Mystery Dungeon - Explorers of Sky)* [Video File]. Retrieved from: <https://www.youtube.com/watch?v=DDz5UR-WWhY&t=25s>

[Shinx]. (2015, February 20). *147 - Mission Failure - (Pokémon Mystery Dungeon - Explorers of Sky)* [Video File]. Retrieved from: <https://www.youtube.com/watch?v=L3hsbseh9yQ>

[Shinx]. (2015, February 20). *002 - Top Menu Theme - (Pokémon Mystery Dungeon - Explorers of Sky)* [Video File]. Retrieved from: <https://www.youtube.com/watch?v=Yx1OWa8NM30>

**FOR GAME LOGIC:**

Published by admin. GitHub. StevePaget/Pygame\_Functions. Retrieved December 11, 2019 from [https://github.com/StevePaget/Pygame\_Functions/blob/master/pygame\_functions.py](https://github.com/StevePaget/Pygame_Functions/blob/master/pygame_functions.py?fbclid=IwAR1AjY7DoBe0t9KGEDPJzh-7c5lJNhBLNR-JLm0hwtJWMM2AkOCi6y5bzHY)

Published by admin on January 20, & Admin. (2017, February). GitHub. kidscancode/pygame\_tutorials. Retrieved December 11, 2019, from [https://github.com/kidscancode/pygame\_tutorials/tree/master/platform](https://github.com/kidscancode/pygame_tutorials/tree/master/platform?fbclid=IwAR1vAt7Ubuiy6rTIG3hQhEiYRPM0TiOxEdIHBRe8kzPNnIzXoA4Zgq9IAlU)