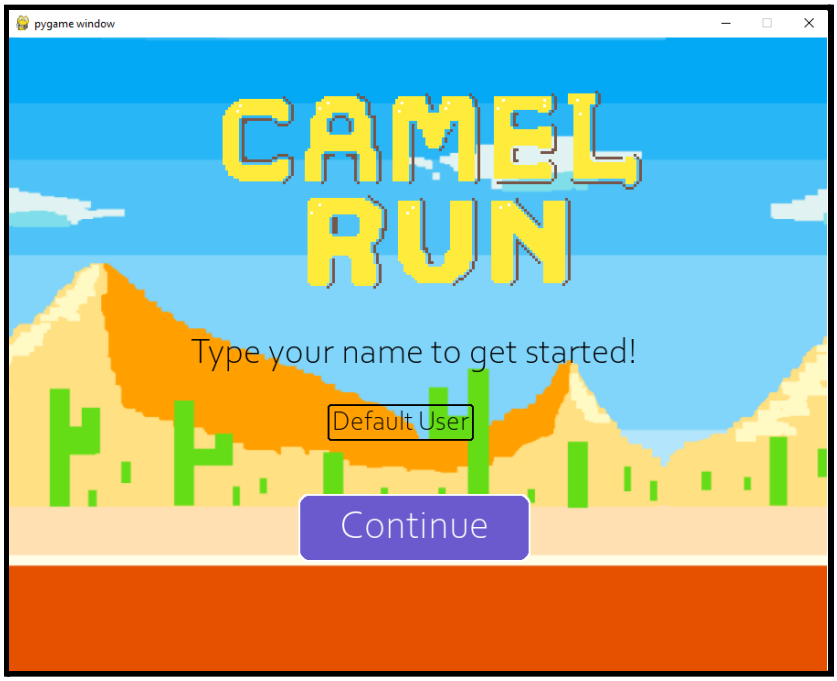
TEAM MEMBERS: Tahsin Tasnim and Jun Yi He Wu



PITCH / CLEAR STATEMENT

Camel Run is a fun, pleasant, and visually and auditorily appealing game to lay back and play on a late afternoon that can be equally competitive and relaxing. Through the powerful and complex methods offered by pygame, we were able to create a fully-functioning running game inspired by the mythical Google Dinosaur! Although our original idea was to recreate Google Dinosaur, our game (Camel Run) expanded beyond the features found within the previously mentioned game. We have three levels, each level with a different character, a different theme, and a signature soundtrack. The first level is a camel in a desert, hence the name of this game! The second level is a flamingo on a beach, and the third level is a dinosaur in a jungle. The dinosaur level is our favorite! All the characters and themes were drawn by us! As you go through the game, you will collect coins and these coins will allow you to unlock these levels. The first level is the default for a new user and requires 0 coins, the second level needs 150 coins for access, and the third requires 250 coins for access.. If you enter your name at the beginning of the game, you can even save your progress, your high score, and your total coins! Hope you enjoy it, and please leave 5 stars on the App Store and Google Play!

FILE COMPONENT

CLASSES

Button Class -

- Same button class as created with graphics.py in class, but updated with pygame constructors (e.g. pygame.draw.rect)
- Creates button label, gives the button a color, gives the button borders and makes the edges of the button round.
- Button.isOver
    - If the mouse if over the button, then it changes to a different color (more interactive)
        - Using if event.type == pygame.MOUSEMOTION:
- Button.isClicked
    - Makes the click sound from pygame.mixer
    - Change the button to a different color to indicate that it is clicked
    - Returns True

Player Class -

- Accesses the images of the characters, and has specific features such as jumping, ducking, and dying alongside the corresponding sound effect (jump sound, duck sound, etc…)
- The character images are a Camel, a Flamingo and a Dinosaur (drawn by us). Within pygame these images, once created, join the Sprite class, and the player class itself is supercharged with super(). According to Real Python, super() gives "you access to methods in a superclass from the subclass that inherits from it."
- The player moves using the self.update() method, which utilizes key presses from the main event queue
    - The player can move from right to left, jump and duck. However, there are boundaries. The player cannot move outside of the screen, the player cannot double jump or double duck.
- When your character "dies" after colliding, "blood_splatter.png" is blotted onto the screen for visual effect.

Obstacle Class -

- Also a sprite class, and it's also supercharged with pygame super().
- This class accesses a set of images which includes a rock, a bird, and poop. They have no other specific function aside from moving leftwards from the right, and being killed out of existence once they go over the left edge. The object's sprites are randomly spawned within the screen (within a boundary near the player's jump and duck limit), and move at random speed with a range of 5 to 20.

Coin Class -

- Also a sprite class and it's also supercharged with super()
- It loads the image of a coin, and moves it from right to left (like the obstacle class).

- There is a method to “consume” the coin when the player collides with it.
- The coin spawns at a random spot (within a boundary near the player’s jump and duck limit), and moves at a random speed (range of 5 to 20).

Main() -

- createText()
    - Convenience function. In pygame, there is no option to directly draw text, so we need to make it into a font and then render the font onto the surface.

- BeginPage()
    - Creates the initial page in which the user could interact by typing their username. This information is stored in the User Record (user_record.txt) to later add and update the player’s total coins and high score.
    - There is file processing occurring within this function to see if the player is a new member or a returning player.
    - It returns name, coins, and highscore.

- MenuPage()
    - Creates and draws the buttons for the menu. When the button is clicked, it changes the color of the button and the respective function is called.
        - The buttons are ‘Introduction’ button, ‘See Skins’ button, and ‘Start Game’ button.
    - MenuPage() is also the “central” function that is called most often for the player to come back to after the end of all the other functions, such as coming back to Menu after reading the instructions in Introduction and after losing a game to begin a new run

- Intro()
    - Displaying the instructions and purpose of the games, and shows the arrow keys (for how to play the game).
    - When the player clicks the ‘Got it!’ button, the screen brings the user back to the Menu.

- CharDisplay()
    - Draws the three character skins and the corresponding buttons to choose a character. This provides the option for the player to choose which character/theme they want to play (if they reach the coin milestone: flamingo/beach (150 coins) and dinosaur/jungle (250 coins)).
    - If the ‘Done’ button is clicked, the screen brings the user back to Menu.

- Game()
    - Main while loop of the game that turns the play time itself
    - The variable “level” is passed from CharDisplay() to Game(), and depending on the “level” it will play the corresponding background music. If the level == 1 then the program plays the desert soundtrack, if the level == 2 the program plays the beach soundtrack and if the level == 3 the program plays the jungle soundtrack.
    - Depending on the level, it will also give the corresponding character image and the background image (similar to the if statements for the background music).
    - All the classes interact within this function
        - The player class creates and displays the player’s character onto the screen, and moves it with sound effects based on the keys that are pressed (player.update()).
        - The obstacle class creates and displays the obstacles to avoid in order to for the player to stay alive, which move from varying positions and varying speeds leftward from the right (obstacles.update())
        - The coins class creates and displays the coins to collide with to increase your coin count by 10 coins. They similarly move from varying positions and varying speeds leftward from the right (coins.update())
        - The score increases by 60 for every run within the while loop
        - If and when the player collides with an obstacle:
            - The player visibly dies (player.die() to show blood splatter) and sound effects are played to match (ouch sound, the losing music and the collide sound).
            - The game score is compared to the highest score - if the game score is greater, the highest score is updated.
            - Coins collected in that game session are added to user’s total coins
            - The User Record is updated with the latest numbers.
            - Run is set to False, breaking the while loop and calling MenuPage to give the user the option to play again.
            - If the player chooses, they can also just hit X and quit the game.

All of these functions have their own while loop to access the event queue and track user interaction, including if the user hit QUIT. All of these functions also have their own background, and pass the parameters of screen, name, coin, level, and highscore as parameters. Only BeginPage() returns anything to begin a flow that maintains the user’s information.

KIND OF TESTING & FUTURE VERSION

In order for our game to function properly, we re-ran the code multiple times to catch for any errors found within the IDLE window. Aside from that, we checked the “screen” (window) to see

if objects and sprites were blitted correctly, and also tested the sound to see if they were playing and adjusting the volume. One thing that we used extensively throughout our code was "print statements" to trace information travel from one function to another and make sure there were no errors or loss of data as the program progressed. This helped to trim down the code and solve certain errors that the file was giving previously. We also had several people be test players for our game, for example Dean Arcelus (who was giving out late-night study snacks at Cro), and Tahsin's and Jun's friends.

Although there are certain "future" modifications we would like to implement, our current code stands firm without these modifications. However, these modifications will certainly make the game more complex and entertaining. Since we already track the highest score, one of the improvements we were thinking of was to add a leaderboard. This wouldn't be difficult as we already store the highscore alongside the player's name and total coins, and we only need to rank those high scores and display it somewhere within the Menu (when the user clicks the Leadership Board' button within the Menu). We currently have 3 characters, 3 settings and 3 soundtrack, but it would be incredible if we could add more. It would also be an improvement if we could cross character, themes and soundtracks together (for example, a camel in a jungle).

An additional setting that would make our game more interesting is an underwater area. This would change the dynamics of the player class a bit, since moving forward and backward, jumping, and ducking physically do not happen the same way underwater as in air. In terms of obstacles, we wanted to have more obstacles that were "themed" according to the setting. We could have inclines, holes, and even have obstacles that would not kill the player and end the game, but instead damage the player or slow them down. We were also thinking that if we were able to put "boundaries" for some obstacles such as the poop or the rock to make them stay on the ground, it would be more realistic. Implementing a 'Store' in which you could use accumulated coins to buy certain items (such as items of clothing to put on your character) is also a consideration. Our current game is set up such that when you reach 150 coins, or 250 coins you will unlock the "levels" but that value is not subtracted from your total coins (you are not buying the characters, only accessing them). Future implementation in which you can "use" your coins will make the game more interactive, enjoyable, and practical. Lastly, we would like to expand it to Google Play and App Store, and popularize the game within Connecticut College!

RESOURCES

Alexpinho98. Jun 22, 2013. "*How to make the background continuously scrolling with Pygame?*" [Forum]. Stackoverflow, https://stackoverflow.com/questions/17240442/how-to-make-the-background-continuously-scrolling-with-pygame

Clear Code. May 14, 2020. "*Python / Pygame tutorial: Getting text input*" [Video]. YouTube, https://www.youtube.com/watch?v=Rvcyf4HsWiw

Jon Fincher. "*PyGame: A Primer on Game Programming in Python*" [Article]. Real Python, https://realpython.com/pygame-a-primer/#note-on-sources

Tech with Tim. Feb 3, 2018. "*Pygame Tutorial #2 - Jumping and Boundaries*" [Video]. YouTube, https://www.youtube.com/watch?v=2-DNswzCkqk

Tech with Tim. Mar 14, 2018. "*How to Create a Button in Pygame [CODE IN DESCRIPTION]*" [Video]. YouTube, https://www.youtube.com/watch?v=4_9twnEduFA