Lab 6 – Developer Usage Documentation

Anyone will be able to use or adapt our final project. From the code alone and the installation of Python and Pygame 3.2 and Visual Studio (or similar IDE), any developer is be able to play our game and adapt the game's environment to upgrade or make new versions of the game as they see fit. Our code is reasonably well documented and because it is written in python it is a little bit easier to pick up and manipulate as Python has a smaller learning curve.

Would any user be able to adapt and implement anything they wanted to just from our code? Not necessarily. Our final project will accomplish everything in the design document labeled as Priority 1. Any additional priorities (labeled 2 or 3) may be implemented in the future or have a pseudo-code outline to a greater or lesser extent.

Some extra features are not even handled by our requirements and design. For example, if someone wished to implement wall jumping there is no existing requirement or area of the code that deals with this concept. Therefore it would fall upon the future user/developer to break down these tasks and figure out an implementation on their own.

Our use case for extending the game is adding a level. The requirements and design do not specify a number of levels the game must have. There can be potentially any reasonable amount of levels, provided the artwork and tile maps to do so. Say you wanted to create a new level.

The work required to make a new level is:

- Make or find new art assets for the background, platforms, and other pieces of the level.
- Create a new Level_XX class the inherits from Level.
  - Where XX is the number of the level
- Create at least 2 level arrays for level options. This is required if you want procedural level generation. For example,
  - ```
    level = [[50, 50, 2, 600, 450],
             [50, 50, 1, 700, 450],
             [300, 100, 4, 500, 500],
             [50, 50, 1, 450, 550] ]
    ```
  - Each element (platform) in the level (which needs to have an art asset associated with it, like asset1.png) is a sublist consisting of integers are its pixel positions.
- Add the platforms to the platform list.
  - ```
    for platform in level:
        block = Platform(platform[0],
                         platform[1],
                         platform[2])
        block.rect.x = platform[3]
        block.rect.y = platform[4]
        block.player = self.player
        self.platform_list.add(block)
    ```
- Add the level to the list so it can be played. This is in main.
  - ```
    level_list = []
    level_list.append( Level_01(player) )
    ```

See below for more ways to extend the game.
    ```

README

Welcome new user! You are about to enter a fun and exciting new world of developing games in Python. To get started do the following:

1. Install Visual Studio (or any IDE). Heck, even VIM will do it for Python.
2. Install Python 3.2. This may require you to uninstall the current version of python you are using.
3. Install Pygame 3.2 and place it inside the same folder as your Python installation.
4. Open Visual Studio and create a new Python project.
    a. This will require you to download the Python development libraries. Visual Studio should prompt you to do this automatically.
5. Inside the window that has now opened copy and paste the main code of PythonApplication3 into the window and save it. (Found here: https://github.com/drexel-game-devs/Project_Folder/tree/master/Code/PythonApplication3/PythonApplication3)
6. Navigate to where you project folder is located. Create and save the remaining python files there.
7. Manipulate the code as you see fit.
    a. Any new artwork you prefer can be placed inside https://github.com/drexel-game-devs/Project_Folder/tree/master/Code/PythonApplication3/PythonApplication3
        i. The file paths must be edited in the code in lines such as `pygame.image.load(filename.png)`
    b. Any new functionality you want the game to have can be added by way of Python functions found in the file https://github.com/drexel-game-devs/Project_Folder/tree/master/Code/PythonApplication3/PythonApplication3/PythonApplication.py. This includes the following classes:
        i. Player – the main character that the user controls
        ii. Bullet – the shooting functionality
        iii. Mob – the AI enemies which the main player fights
        iv. Platform – solid blocks which the player can jump on
        v. Level – the environment for the levels
            1. Level_01 – the tile maps for level one. These can be edited or added to by inheriting from Level and calling it Level_XX, where XX is the number of the level.
    c. Functionality on a higher level of the game can be found in the file https://github.com/drexel-game-devs/Project_Folder/tree/master/Code/PythonApplication3/PythonApplication3/Title_Screen.py
        i. This includes the title screen and pause functions. If you want to change what happens during these two states of the game, this code can be edited accordingly.