PROJECT REPORT

Project Specification

- Name : Word Box Juice
- Genre : Space shooters
- Mode : Shoot ‘em up
- Scenario: Player control a spacecraft with the ability of moving and shooting. A random word is chosen from a text file, and the player must collect the specific number of letters which form the chosen word to obtain points. Other letters must be avoided or shot down. If the player gets hit, then the game is over.

Solution Design

The project was designed with the objective of providing a fun and classic gamified way of learning a set of given words. It only has one classic endless mode.

Discussion

- Main.py

Game class has the following functions:

a. init(): assign values to setup the game’s state,background,player character, word,tile,health,score,background music and sound effects.
b. display_score(): render the score as a text on the screen
c. display_life(): render the remaining life as a text on the screen
d. display_target(): render the required amount of characters to form the randomized word on the screen
e. generate_word(): get a random word from a read text file and create lists based on dictionary of the amount of character/letter(s) of the word. Calls the display function when not generating.
f. generate_tile(): If boolean condition is met, with the range of the length of alphabets, randomly select a number and create a letterTile instance accordingly, which group is then added to the primary tiles group. It also checks whether the cooldown time has passed and if so, the boolean state will be True to allow the condition to generate a tile to be met.
g. goodShipCollide(): adds scorepoint and play “glass shattering” explosion sound
h. badCollide(): deduct one lifepoint, check if there’s no more life left(if so, then call the end game function) and play “glass shattering” explosion sound

i. playMusic(): make a list from the background music directory, then randomly choose one to be played.
j. checkCollissionBetweenTileAndLaser(): call goodLaserCollide function upon a collission between tiles and laser group
k. checkCollissionBetweenTileAndPlayer(): upon collission between tile and player group, determine which group the tile belong to assign the letter to a variable, check if the letter is in target: if is, then call goodShipCollide function but if not, then call badCollide function.
l. background(): draw the background on the screen
m. run(): call all the functons for update and checking conditions

- ships.py

a. init() : assign values for object body members, width constraint, laser reload factors, laser group and laser sound effects.
b. get_input(): receive responses from pressed key arrows (left&right) for player ship's movement. Pressing the space key will result in shooting lasers if shooting cooldown condition(self.ready) is met.
c. recharge(): check whether cooldown time has passed and if so, player ship can shoot laser again.
d. constraint(): check whether the rect positions are more than screen width or height, and if so, move rect back at constraint limit
e. shoot_laser(): create a laser instance based on player ship's rect position and play the laser sound effect
f. update(): call for all four ships' functions (excluding init)

- letterTile.py

Contains a Super and many Subclasses.

The Super Class has the following functions:

a. Init() : assign values for image, rect, height constraint and speed
b. init subclass (): append it's subclasses to a list for randomization in generating tiles
c. Destroy(): once the tile breaks the constraint limit, the tile instance will get destroyed
d. update (): moves the tile downwards and calls the destroy function

The Subclasses have the following function:

a. init (): assign a new value for image, based on the alphabet letter

- laser.py

Contains the Laser class, which has several functions:

a. init (): assign values to laser instances such as it's image, rect, speed and boundary limit
b. destroy () : once the laser breakthrough the boundary limit, the instance will be destroyed
c. update (): moves the laser upwards and calls the destroy function

Evidence

Image 1.1 Collected a Letter

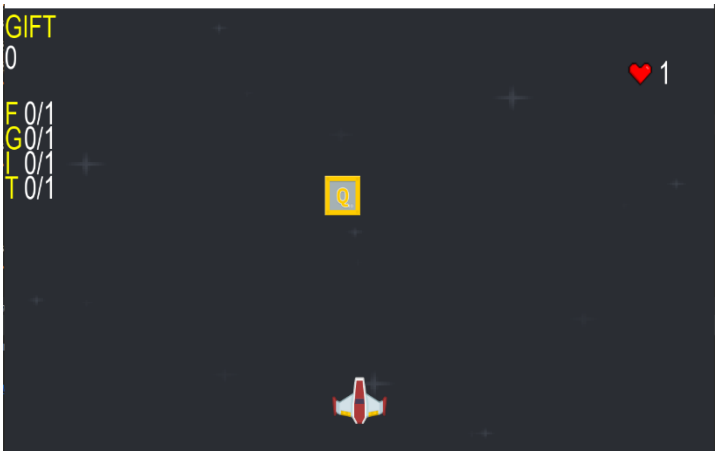

Image 1.2 Received 2 Damages

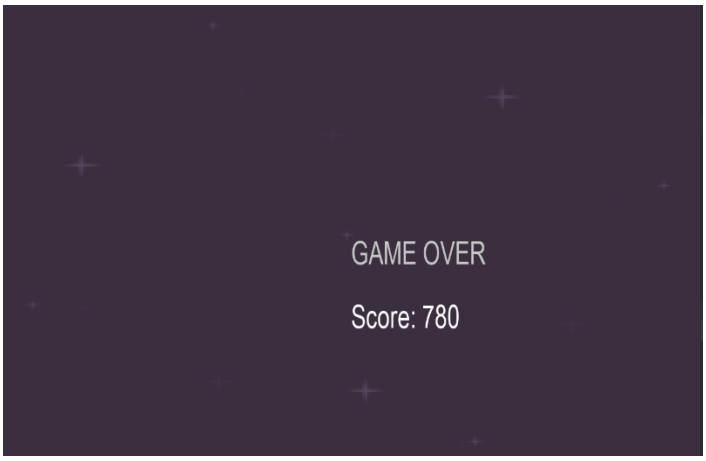

Image 1.3 Game Over Screen