Final Project Report
“Flappy Plane”
Algorithm and Programming

Lecturer:

Jude Joseph Lamug Martinez

Tirza Gabriella

2602109870

CLASS L1AC

Computer Science Major

Binus International University

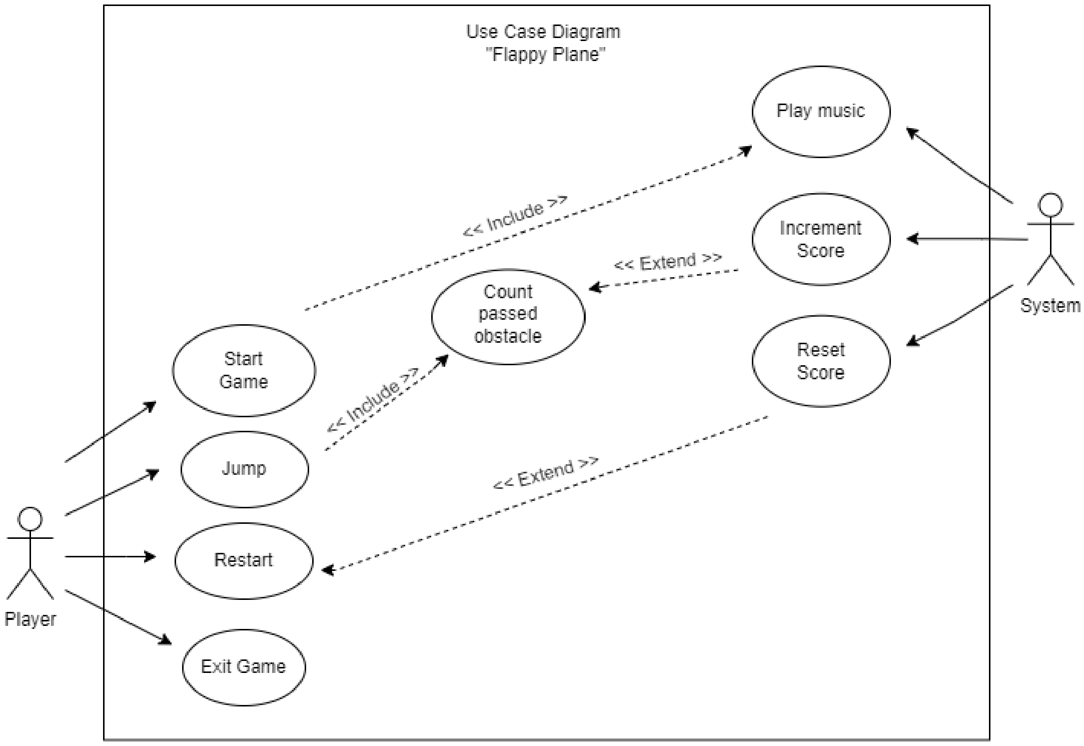
2022

Table of Contents

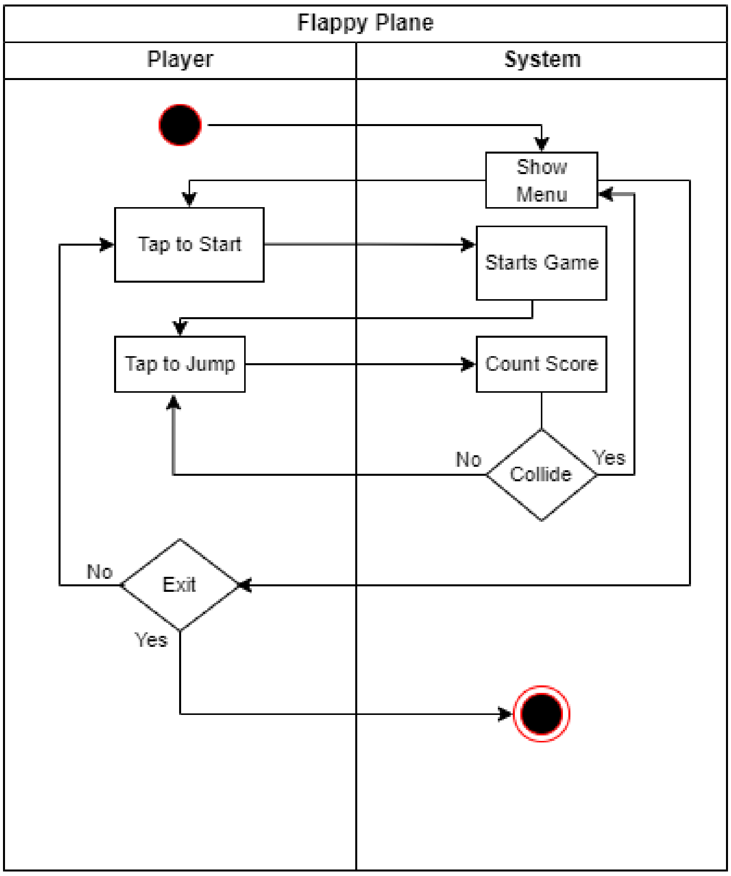
I. Brief Description

For the final project assignment, I’ve decided on making a game called “flappy plane”. Before deciding to make this game, I got much confusion about what topic I’m gonna make that suits the best with my skills. I’m also considering what project it’s not too complicated but not too simple and it suits the criteria. I had to shuffle through many ideas that I scrapped, such as typing tests and rock paper scissors. In the end, I decided not to use them because they didn’t fulfill the criteria of the assignment. After a number of considerations, I choose “flappy plane” a game that uses PyGame as its main module. Basically, it is my own rendition of the famous flappy bird game. I choose this game because it uses a basic implementation of classes, modules, functions, instance variables, and objects. But the most important reason is that it suits me the most with my basic skill of python, which was taught by Mr. Jude

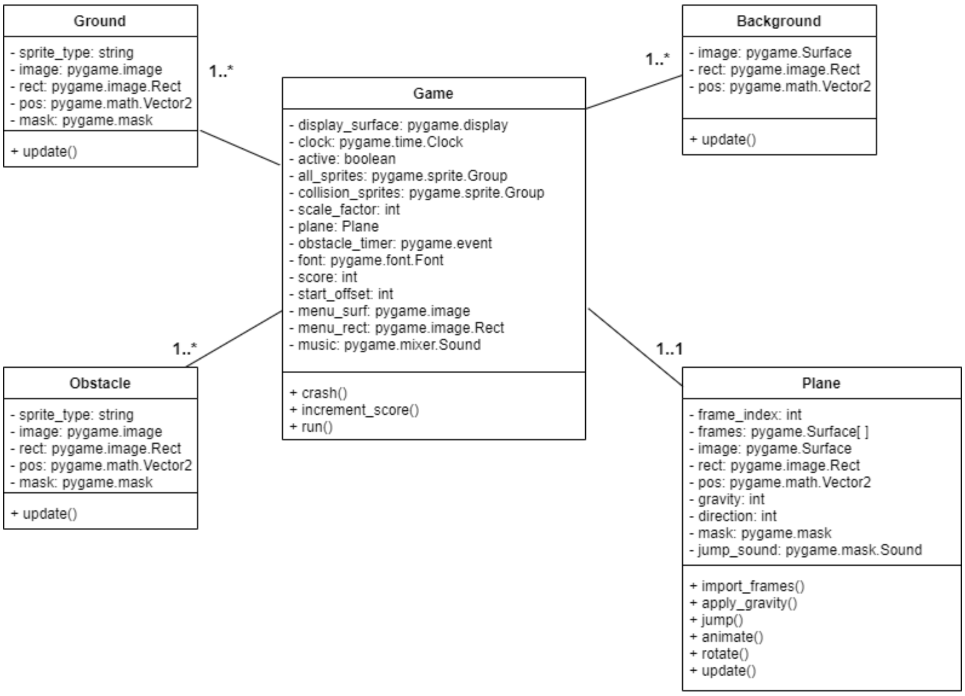
II. Use-Case Diagram



III. Activity Diagram



IV. Class Diagram



V. Modules

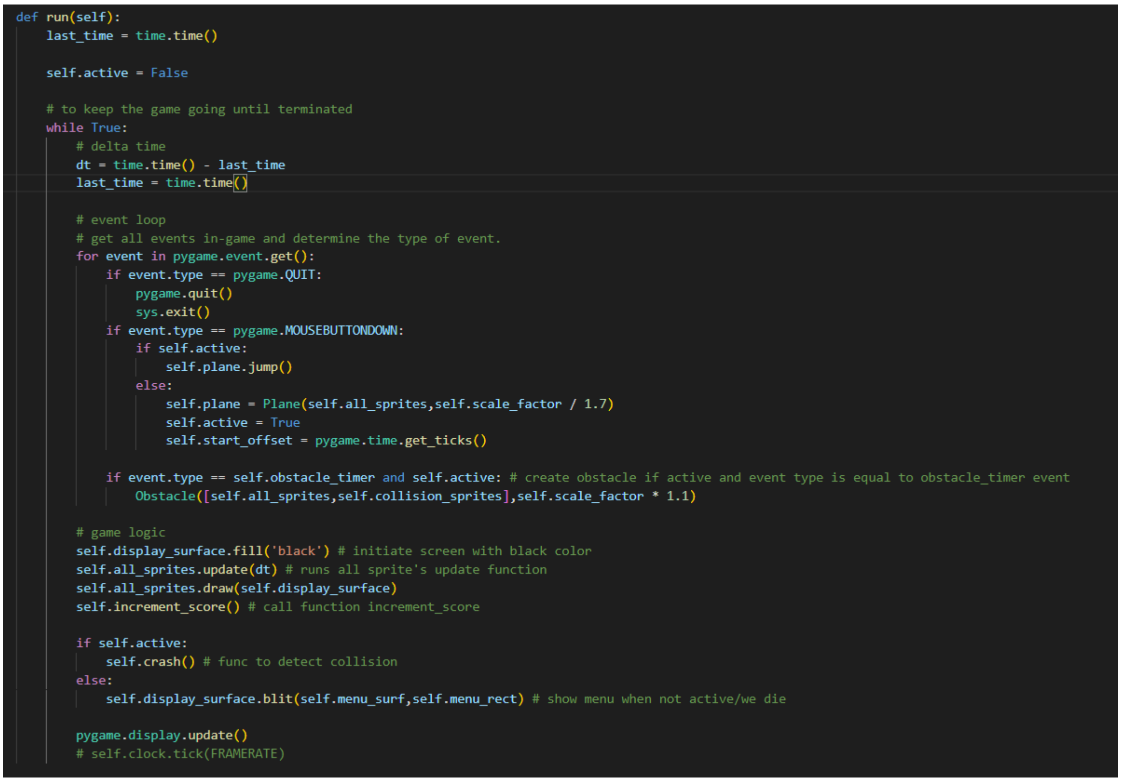
```python
import pygame, time, sys
```

```python
from random import choice, randint
```

Modules:

- PyGame - all the basic functionalities of the game. That includes sprite movements, event handling, and the basic logic of the game.
- sys - module to manipulate the virtual environment (used to exit the code through sys.exit)
- time - module used to get Clock, get the current time, get ticks
- random - module to use choice, and randint

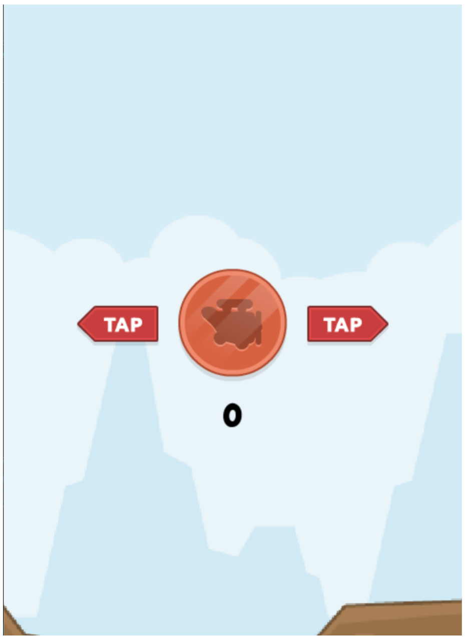
VI. Essential Algorithm



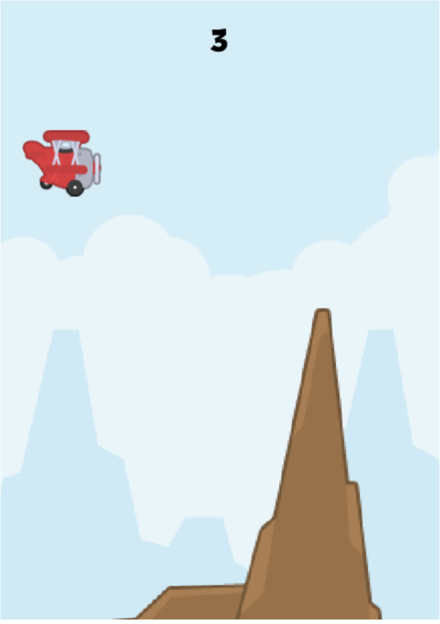
```python
def run(self):
    last_time = time.time()

    self.active = False

    # to keep the game going until terminated
    while True:
        # delta time
        dt = time.time() - last_time
        last_time = time.time()

        # event loop
        # get all events in-game and determine the type of event.
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if self.active:
                    self.plane.jump()
                else:
                    self.plane = Plane(self.all_sprites,self.scale_factor / 1.7)
                    self.active = True
                    self.start_offset = pygame.time.get_ticks()

            if event.type == self.obstacle_timer and self.active: # create obstacle if active and event type is equal to obstacle_timer event
                Obstacle([self.all_sprites,self.collision_sprites],self.scale_factor * 1.1)

        # game logic
        self.display_surface.fill('black') # initiate screen with black color
        self.all_sprites.update(dt) # runs all sprite's update function
        self.all_sprites.draw(self.display_surface)
        self.increment_score() # call function increment_score

        if self.active:
            self.crash() # func to detect collision
        else:
            self.display_surface.blit(self.menu_surf,self.menu_rect) # show menu when not active/we die

        pygame.display.update()
        # self.clock.tick(FRAMERATE)
```

Start with the delta time, later on, delta time will be used to calculate the movement of the sprites. Then, we run the game continuously until the game is stopped (while true). Then we read all the events and determine what kind of event it is. If the event type equals quit, then it will exit the game. If the event equals mouse-click, then we continue on to another checking. If the game's active status is true then it will jump. Else, if the game's active status is false then it will make the plane and set the game's active status to true. Outside of the mouse-click checking, create an obstacle if active and the event type is equal to obstacle_timer event (custom event to time when obstacles appear). For the game logic, first, it initiated the screen with black color and runs all sprite's update functions (each sprite has its own update function to determine its movement). And then, we use functions to increment the score. Then finally we check the game status. When the active status is true then it will run the crash function, which is to detect crashes. Else, when it's not active (we die) it will show the menu button.

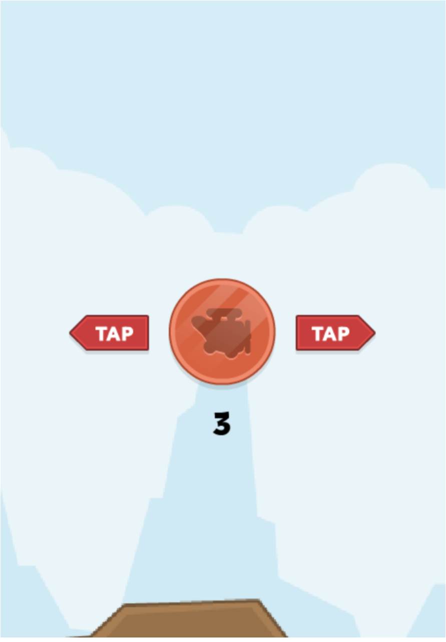
VII. Screenshot of application



This is the first thing that will be seen if we start the "Flappy Plane" game. In the first place, it shows the start button, when we click it, it goes directly to the game (game starts). Music starts when the game object is instantiated and another voice will also trigger each time the plane jumps.



Every time we pass any obstacles (up and down) the system will increment the score. When the plane crash (hit the obstacles) the system will show the start button again with the score below, so we can choose whether you want to play again or not.



VIII. Lessons learned

a. Error handling

One of the most frustrating parts is when it comes to handling errors. It's quite challenging when we have to debug our own program. But in that case, I have my own way to figure out what went wrong in my code. The first thing is to write down things using the print function where I expect things to be. If it's printed out at the console, it means that I was right in thinking so, if not then something went wrong and I need to check it again. The second thing I do is when I face an error in my calculation about the distance between any obstacle, the gravity of the plane, the speed of the screen, etc. So I tried changing the value one by one until I got the right one.

b. Reflection

From this project, I have acquired a lot of knowledge as a programmer, especially in making a game using PyGame. It feels like programming can be challenging, fun, and interesting at the same time, but there's still tension when it comes to deadlines. However, I am satisfied with how my program ended up being, even though there are many improvements that can still be done (for example: making an exit button) to the existing program. After all, this is one of my first python projects and I will take the lessons I've learned (classes, functions, OOP).

GitHub link: https://github.com/tirzagabriella/FinalProject-FlappyPlane.git

Video demo:

https://drive.google.com/file/d/11LHrjVi76QO_E9aIGywi9CTS8eIlkMUv/view?usp=share_link