CS1822

GAME PROJECT REPROT

Higashiuezato, Takase (2021)

Student ID: 100991373

INTRODUCTION 2

USER MANUAL 3

KEY CONCEPTS 4

TECHNICAL CONTENT 5

A MOMENT OF REFLECTION 7

CONCLUSION 8

Introduction

In this report, I will be introducing our game project. This report is divided in to five parts. The first section will explain all the features and prepose of our game. For example, how is this game going to work, what kind of role should know before start game.
The second section will describe mainly summary of how use sprites, collisions, and vector class in this game.
The next section will present my member's code which I picked up particularly interesting part. The final section will be a moment of reflection.

User Manual

Our project is Asteroids. This is classic game. In this game you navigate a spaceship in a simple way by using keyboard.

Role of this game:

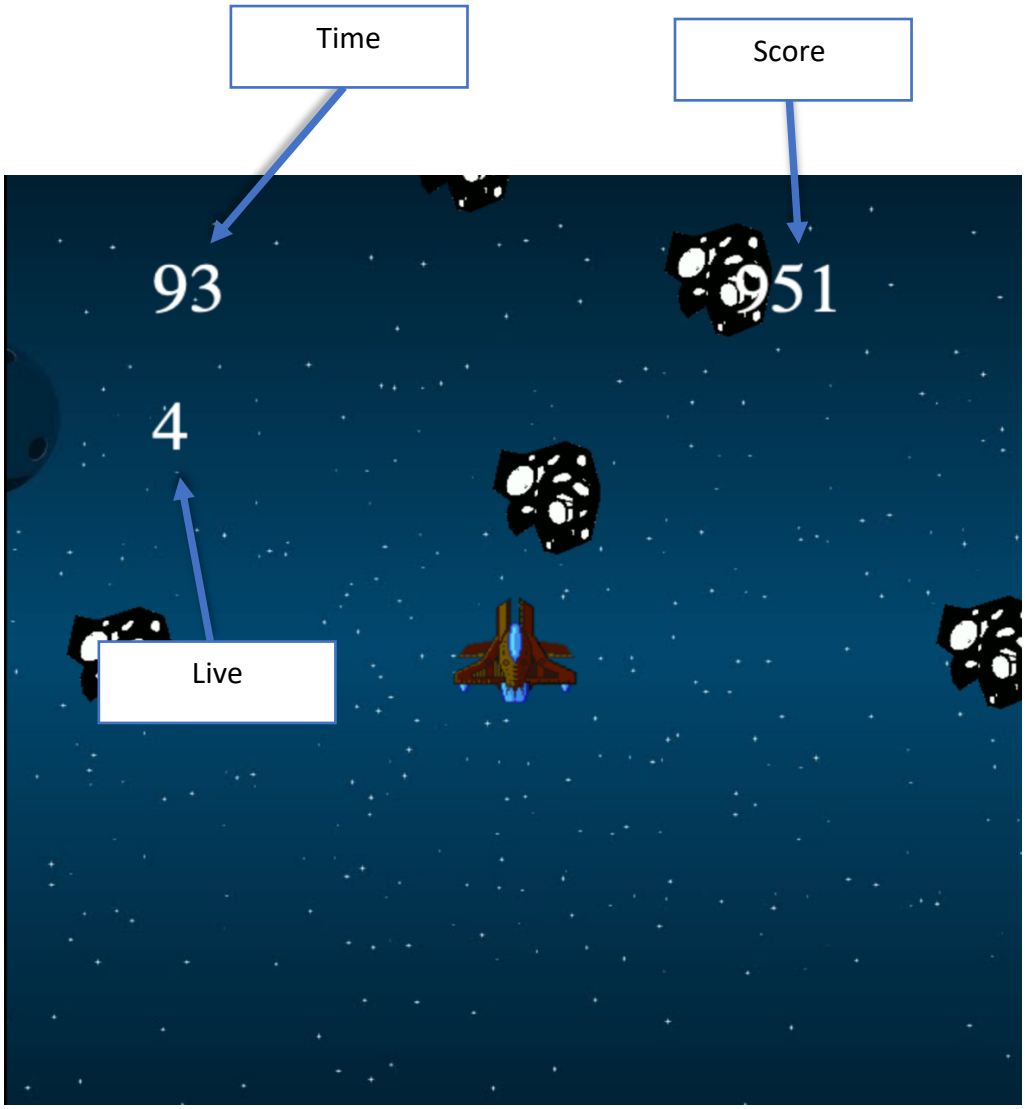
1 Player spaceship

2 Meteorite

Planning

Gantt chart

Interface Design
Welcome screen



Key Concepts

Sectors

: Event

: Sprites

: Vectors

: Collisions

: Bouncing Ball

: Colliding Balls

Interaction class

```
def hit(self, spaceship, meteorite): #provisional instance val
    distance = spaceship.pos.copy().subtract(meteorite.pos)
    return distance.length() <= spaceship.radius + meteorite.radius
#if ship radius + met radius bigger than or equal to distance between ship pos and met pos
```

Technical content

```
def collide(self, spaceship, meteorite):
    if self.hit(spaceship, meteorite):
        mete_exist = meteorite in self.in_collision
        self.score.on_hit()
        if not mete_exist:
            self.in_collision.add(meteorite)
            self.is_mete_nolonger_visible.add(meteorite)
            self.collision_occured = True
            self.count += 1
            score_value = self.score.score_val – 100
```

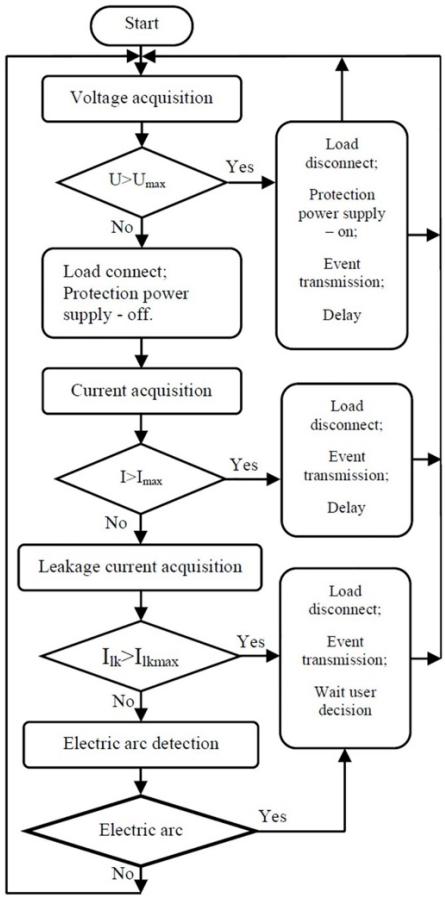
this is collide method. Which is for determine whether collision is occurred among player ship and meteorite. If hit has been occurred, we need to check meteorite is already in the in_collision set. Otherwise, score hit will be called. Then we going to check if the meteorite was exist, we need to add the mete to in_collision set. At the same time, we need to add it to nolonger_visible set. This is because on draw_handler method, it will useful to make them invisible. After that we defined collision_occured is True.

```
def hit(self, spaceship, meteorite):
    distance = spaceship.pos.copy().subtract(meteorite.pos)
     return distance.length() <= spaceship.radius + meteorite.radius
```

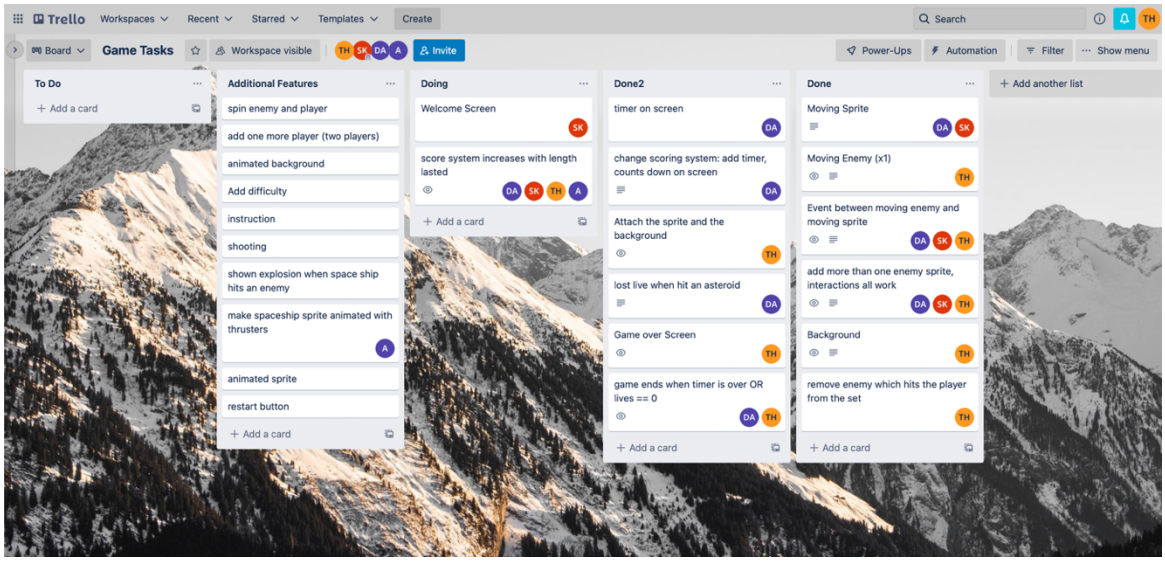
This method for detecting the collision between player and enemy.
first, we defined variable it's called distance. It indicates distance from spaceship's position to meteorite's position. The spaceship's position is copied. This is because we don't want to impact to original variable. If it is changed, the position would not stay in place. After that, we going to subtract meteorite's position. Thus, we can get appropriate the distance.

If spaceship's radius and meteorite's radius amount are bigger than or equal to the distance between each position. in other words, if the distance is smaller than or equal to each radius, the method will return True. Otherwise, False.

Control flow


Start
Voltage acquisition
U>Umax
Yes
No
Load disconnect;
Protection power supply – on;
Event transmission;
Delay
Load connect;
Protection power supply - off.
Current acquisition
I>Imax
Yes
No
Load disconnect;
Event transmission;
Delay
Leakage current acquisition
Ilk>Ilkmax
Yes
No
Load disconnect;
Event transmission;
Wait user decision
Electric arc detection
Electric arc
Yes
No

A moment of reflection



How well did your group work? Why?
We set up this list and assigned roles to each member. By doing so, it's easy to see and organise specific tasks to implement within the allotted time.
We often communicated in Microsoft teams, discussed about what is difficult part and what we could better. We always try to catch up every milestone so that making graphical game.

What was hard about writing the program?

The one of the struggling parts was thinking concept of this game from scratch. We had to set role and design. We thought some assumption then we picked up thing that closer to ideal game we want to implement.

In terms of source of game, we already have various of material on Moodle. We initially looked through all videos to solid fundamental of the game sector, after that we put them together so that match our specification (requirements).

What would you do differently if you started again?

Did the game end up doing what you wanted to do?
We learned a lot of things from this project. The thing is do tasks as soon as possible. After meeting with TA, we immediately worked on each task. TA told me if you procrastinate everything, at the end of the day you would have to finish with stressful and it's going to be rough around the edges. Although the time is limited, and we have different module assignment, so we talked about a lot then managed time and thoroughly committed to completing tasks as quickly as possible.

Link to git repo

Conclusion

In conclusion, we were able to get this opportunity to implement dynamic game in group. I always patriated group meeting every time. And I'm so glad that I have managed to organise this group smoothly. It was not easy, but I learnt a lot of skills in terms of group work and python fundamentals.