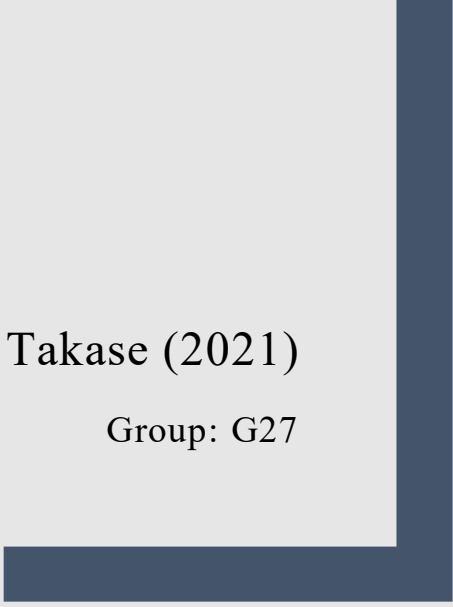


CS1822

ASTEROIDS GAME

Higashiuezato, Takase (2021)

Group: G27



<u>INTRODUCTION</u>	<u>2</u>
<u>USER MANUAL</u>	<u>3</u>
<u>KEY CONCEPTS</u>	<u>8</u>
<u>TECHNICAL CONTENT</u>	<u>10</u>
<u>A MOMENT OF REFLECTION</u>	<u>14</u>
<u>CONCLUSION</u>	<u>15</u>

Introduction

In this report, I will be introducing our game project. This report is divided into 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 a player should know before starting this game.

The second section will describe mainly a summary of how to use sprites, collisions, and vector class in this game.

The next section will present my member's code which I picked up a particularly interesting part. The final section will be a moment of reflection.

User Manual

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

Welcome screen: Canvas size: 700px x 700px.

If a player clicks the welcome screen, the game will be started.



Role of this game:

Scoring is based on timing. level increases and gain point the longer you survive the game. Control the direction of the player by using your keyboard to avoid meteorites.

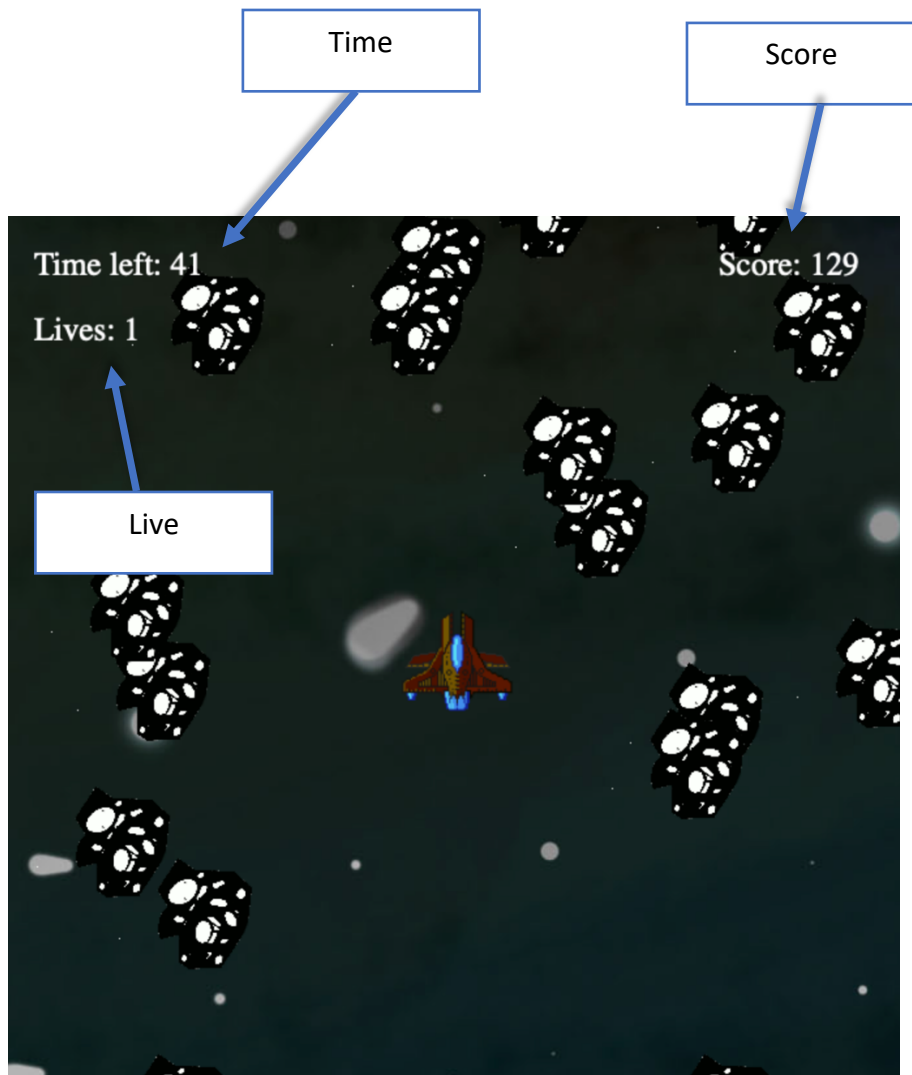
Player Lives: start with 3, lose life when hit with a meteorite.

Player's score: based on time survived. If there is a collision, the score will stop for a few seconds. The score will increase as the time decreases from 35 seconds to 25 seconds, to 15 seconds.

Game: Timer: 50 seconds total, the game is won if player lasts till timer ends Score Counter: score increases every second, as the level increases the counter increases.

: Interface Design

We drew the meteorite sprite by GIMP which is one of the convenient tools for drawing our picture on the computer.



When you won this game, this screen will show up with how many points you got. There should be a restart button, if you want to restart this game, click the button on the left, the welcome screen will appear with lives go to 3 again, everything resets.



When you time over or lost all live, you this screen will show up. There should be the restart button, if you want to restart this game, click the button on the left, the welcome screen will appear with lives go to 3 again, everything resets.



Key Concepts

We used CodeSkulptor3 to implement this game. You can use Python and SimpleGUI in this module.

```
try:
    import simple gui
except ImportError:
    import SimpleGUICS2Pygame.simpleguics2pygame as simplegui
```

Sprites

We drew the meteorite sprite by GIMP which is one of the convenient tools for drawing our picture on the computer. We acquired the rights to use our own sprites online, we have used personal.rhul.ac.uk. when it comes to loading image, we managed to clarify the dimension and image's centre position rather than whenever call them up one by one.

```
IMG_backgournd_DIMS = (1890, 1417)
```

```
IMG_backgournd_CENTRE = (IMG_backgournd_DIMS[0]/2,
    IMG_backgournd_DIMS[1]/2)
```

Vector class

All games are required to make use of Vector class. We used the vector class when it is necessary to make a movement. For example, we used them on the `rand_meteorite` method. when you want the meteorites to flow from top of the screen to bottom, we can say random meteorites velocity is `Vector(0, random.randint(3, 5))`. That means the vector of x is 0, they keep same position of vertical, the vector of y is random number between 3 to 5.

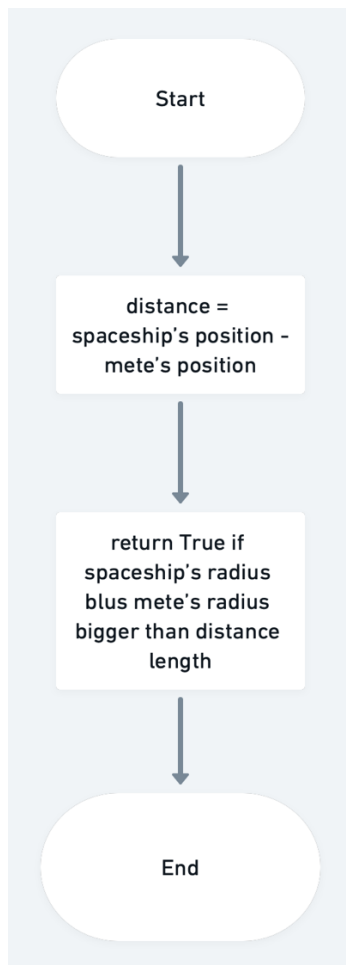
Interaction class

```
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 a variable it's called distance. It indicates a distance from the spaceship's position to the meteorite's position. The spaceship's position is copied. This is because we don't want to impact the original variable. If it is changed, the position would not stay in place. After that, we going to subtract the meteorite's position. Thus, we can get appropriate distance.

If the 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



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 collision method. Which is to determine whether a collision has occurred among player ship and meteorite. If a hit has 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. Because of draw_handler method, it is useful to make them invisible. After that we defined collision_occured as True.

Control flow



```
def show_rand_mete(self):
    if not Game_restart:
        self.meteorite.add(rand_meteorite())
    if self.countdown.get_time_left() < 35 and not Game_end:
        self.score.score_up(2)
        self.meteorite.add(rand_meteorite())
    if self.countdown.get_time_left() < 25 and not Game_end:
        self.score.score_up(4)
        self.meteorite.add(rand_meteorite())
    if self.countdown.get_time_left() < 15 and not Game_end:
        self.score.score_up(8)
        self.meteorite.add(rand_meteorite())
    else:
        self.meteorite.discard(rand_meteorite())
        self.meteorite.add(reset_rand_meteorite())
```

This method for showing random meteorites, at the same time we check the time whether exceed the specific period. Then it will call the score up method from the score class.

On the other hand, we set up reset_rand_meteorite method. This is because we thought only a random method is not enough for continuing the game and time running could be slow.

```
def button():
    global Game_started, Game_end, Game_restart
    Game_started = False
    Game_end = False
    Game_restart = True
    spaceship.lives = 3
    countdown.time_left = 50
    score.score_val = 0
    spaceship.pos = Vector(CANVAS_DIMS[0]/2, CANVAS_DIMS[1]/2)
```

We made a button function for the reset button. We have four Boolean variables. These are useful to know the player's activities. In this case, we need to initialise global since we want to change the original variables. It has to be a brand new condition as the player click the button. The spaceship will return to a default position.

```
sound = simplegui.load_sound(  
    'http://personal.rhul.ac.uk/zkac/332/BoxCat-Games-Battle-Boss.mp3')
```

We have used background sounds from chosic.com.

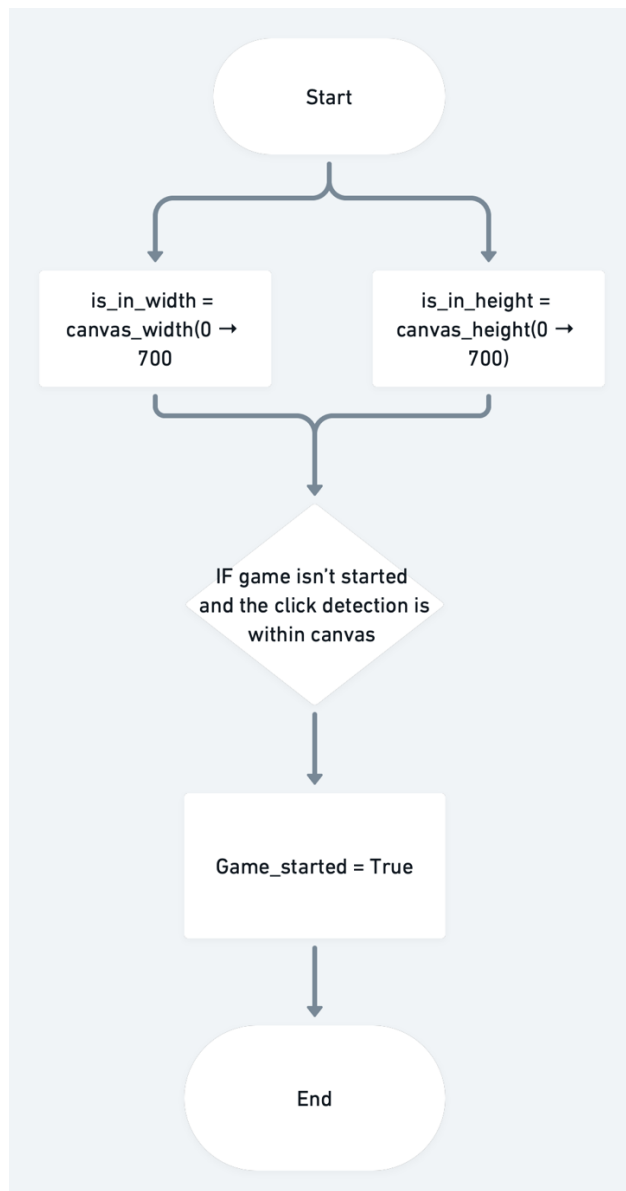
We want to stop music as the game over or player win screen shows up, we used `sound.pause()` function in the `draw_handler` method. In addition, when the player clicks the reset game button, the music should play once more, we used `sound.play()` function in button method.

<https://www.chosic.com/download-audio/29726/>

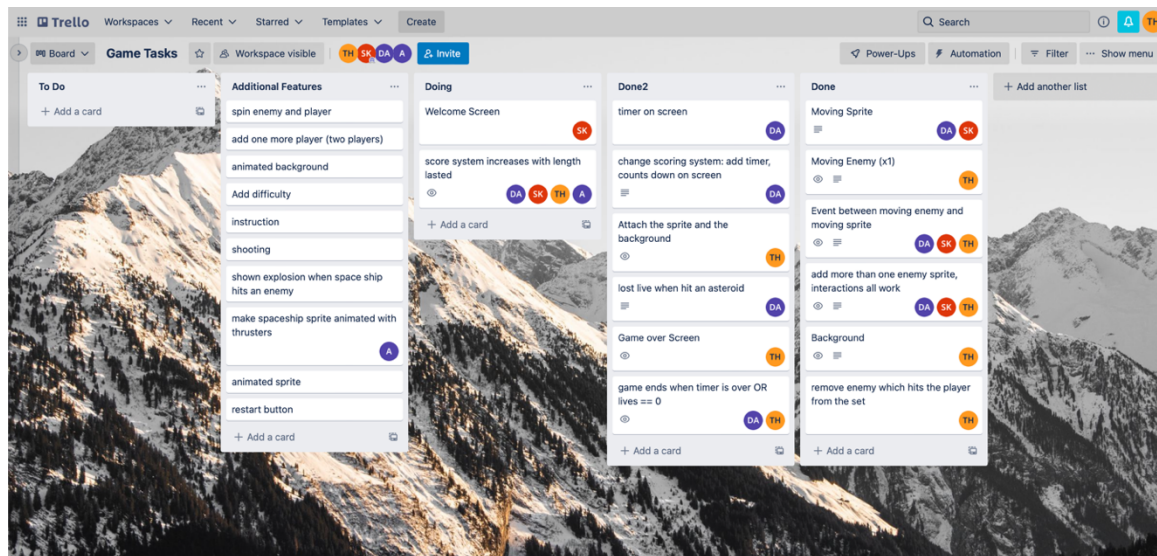
```
def click(pos):  
    global Game_started  
    screen = [WIDTH, HEIGHT]  
    is_in_width = (screen[0] - screen[0]) < pos[0] < (screen[0]) # 0 -> 700  
    is_in_height = (screen[1] - screen[1]) < pos[1] < (screen[1]) # 0 -> 700  
    if not Game_started and is_in_width and is_in_height:  
        Game_started = True
```

This is click method to detect whether the game screen has been clicked. We defined `is_in_width` and `is_in_height` variables. The purpose of this is to check whether a click has been made within the scope of the canvas. Both width and height are 700, thus we just need to detect between 0 to 700.

Control flow



A moment of reflection



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 what is the difficult part and what we could do better. We always try to catch up with every milestone so that make a graphical game. And we submitted the weekly video every week.

One of the struggling parts was thinking concept of this game from scratch. We had to set roles and design. We thought some assumptions then we picked up the thing that is closer to the ideal game we want to implement.

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

We faced with how we would remove an enemy rather than every enemy if there was a collision. The solution is when the draw handler is called, we specified what group we don't want to show on screen. In order to do that, we defined no longer visible boolean variable.

We learned a lot of things from this project. The thing is to 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 stress and it's going to be rough around the edges. Although the time is limited, and we have different module assignments, so we talked about a lot then managed time and thoroughly committed to completing tasks as quickly as possible.

Conclusion

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

References:

Farukh Khalilov <https://youtu.be/TtA-9TGwLLQ> (YouTube) Retrieved 12 Mar 2022

<https://opengameart.org/> (OpenGameArt) Retrieved 6 Mar 2022