Specification Document for AP Computer Science Final Project

Game of Aircraft Battle

June 16, 2019

Shawnigan Lake School

Uaena Tong

Table of Content

1. General Information

1.1 Introduction

For my final project in AP Computer Science class, I will be making an aircraft battle game, which is a classic shooting game. The gameplay focuses on keeping away from the enemies while annihilating their aircraft by shooting missiles. Players will control their aircraft, which is on the bottom of the frame, and has a goal to go to enemy's warship(ultimate boss) through nemesis planes safely. The environment in this game will revolve mainly around space; moreover, there is sound effect and background music to make player immersed in the game. Based on the classic operation, I am looking forward to making some improvements to make this game more enjoyable; such as improve damage by adding other elements to the game.

1.2 Gaming Instruction

Click mouse to start the game.

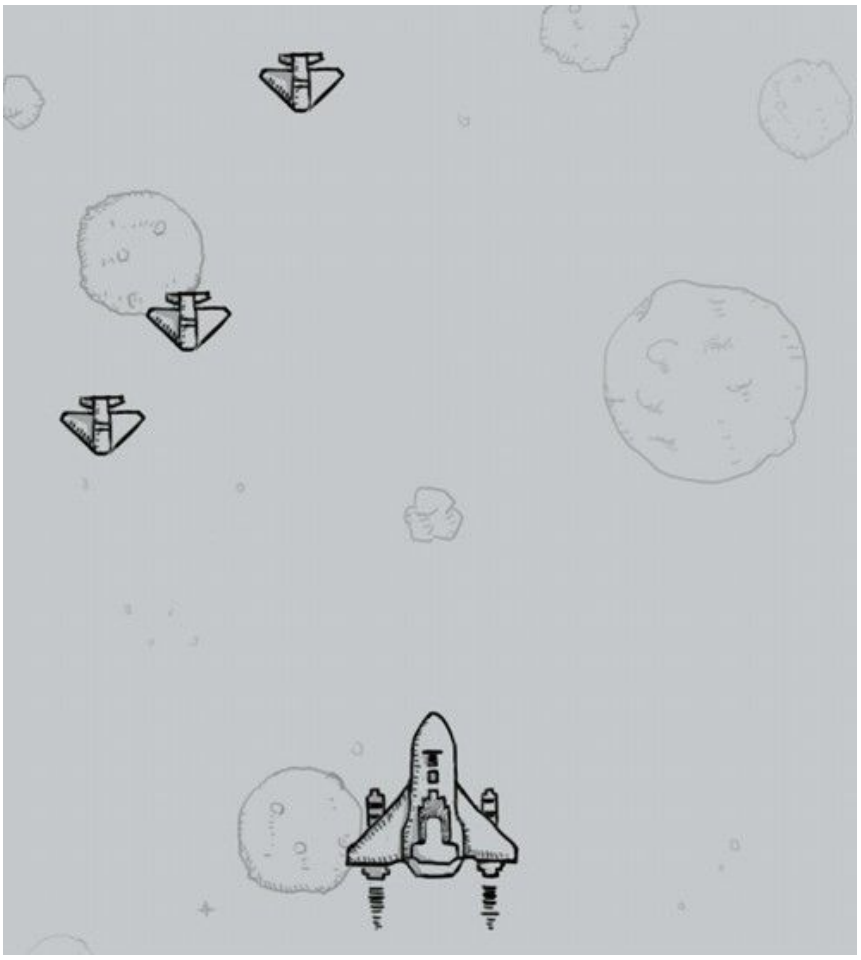
Use keyboard arrow keys ↑↓←→ to control the aircraft.

Score will increase when user destroy enemy's aircrafts.

Once user hit by enemy's aircrafts, the game ends.

Press the Space or click the mouse to shoot.

1.3 Example of Graphical User Interface



2. User Requirement

This program should be able to be used with PyCharm on their computer with high core-count 64-bit. If you are using McOS system, better to don't use McOS Mojave. Moreover, this project require users be able to import Pygame in their PyCharm successfully.

3. Summary of Function

The user will use the arrow keys or WASD in order to play. Move around their character to avoid other flight. The user will not be in control of the timing of the release of the enemy in the way. Game over when the user gets hit by other flights.

4. Technical Specifications

4.1 User Control

Users control the aircraft with arrow keys, WASD and space

```
if event.key == pygame.K_w or event.key == pygame.K_UP:
```

Use pygame.K_ to check if the user press the key or not; than, user's plane will make reaction.

4.2 Collide Test

In order to determine the status of the game, we set every plane as a sprite group, and use `pygame.sprite.spritecollide` to check the collision of each plane. Spritecollide can check the collision between a spirit and another sprite in specific sprite group. It use as spritecollide(sprite, group, dokill, collided = None).

4.3 History Score

Under the python package--*store*, there is a txt file called rest; it records the highest score in the history. The score from the players will be compared with the score in *rest.txt*. If the current score is higher than the history highest, the current score will be the newest history highest score

4.4 Animation Effect

To improve the gaming experience, I drew the picture for every frame and saved them as a list. In the game, the list will plays when needed

4.5 Python Module

pygame:

A cross-platform set of Python modules design for writing video game.

os:

Provide a way of using operating system dependent functionality.

sys:

Provide access to some variable used or maintained by the interpreter and to function that interact strongly with the interpreter.

random:

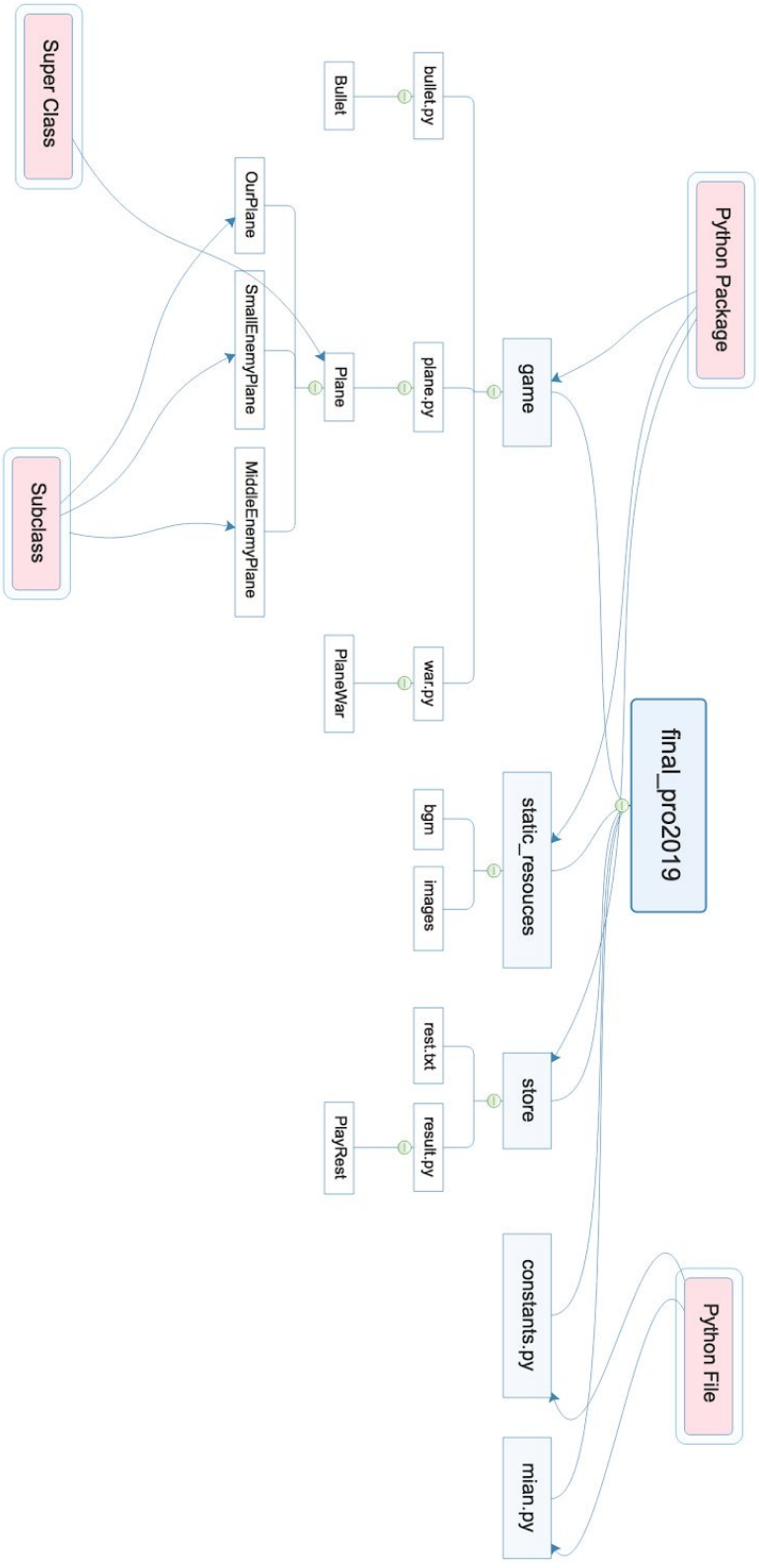
Get random numbers.

4.6 Graphical User Interface with Detail of Each Screen

In this game, users have three status, each status will has different graphical user interface.

1. Ready 2. Playing 3. Over

4.7 Python File and Classes



4.8 Methods

bullet.py:

```
class Bullet(pygame.sprite.Sprite)
    def update(self, war)
```

plane.py:

```
class Plane(pygame.sprite.Sprite)
    def load_src(self)
    def image(self)
    def blit_me(self)
    def move_up(self)
    def move_down(self)
    def move_left(self)
    def move_right(self)
    def broken_down(self)
    def shoot(self)
```

war.py:

```
class PlaneWar(object)
    def bind_event(self)
    def add_small_enemies(self, num)
    def add_middle_enemies(self, num)
    def run_game(self)
```

result.py:

```
class PlayRest(object)
    def score(self)
    def score(self, value)
    def set_history(self)
    def get_max_core(self)
```

main.py:

```
def main()
```

4.9 Code

bullet.py

```
import pygame
import constants


class Bullet(pygame.sprite.Sprite):

    # Status of the bullets Ture: alive False: died
    active = True

    def __init__(self, screen, plane, speed=None):
        super().__init__()
        self.screen = screen
        # speed
        self.speed = speed or 10
        self.plane = plane

        # load the pic of the bullets
        self.image = pygame.image.load(constants.BULLET_IMG)

        # change the position of the bullets
        self.rect = self.image.get_rect()
        self.rect.centerx = plane.rect.centerx
        self.rect.top = plane.rect.top

        # shooting music
        self.shoot_sound = pygame.mixer.Sound(constants.BULLET_SHOOT_SOUND)
        self.shoot_sound.set_volume(0.3)
        self.shoot_sound.play()

    def update(self, war):
        self.rect.top -= self.speed
        # out of the window
        if self.rect.top < 0:
            self.remove(self.plane.bullets)
            print(self.plane.bullets)
        # show bullets
        self.screen.blit(self.image, self.rect)
        # check if hit enemies
```

```
        rest1 = pygame.sprite.spritecollide(self, war.small_enemies, False)
        print(rest1, 666)
        for r in rest1:
            # delete bullet
            self.kill()
            # flight explode
            r.broken_down()
            # score count
            war.rest.score += constants.SCORE_SHOOT_SMALL
            # save history
            war.rest.set_history()

        rest2 = pygame.sprite.spritecollide(self, war.middle_enemies, False)
        print(rest2, 888)
        for r in rest2:
            # delete bullet
            self.kill()
            # flight explode
            r.broken_down()
            # score count
            war.rest.score += constants.SCORE_SHOOT_MIDDLE
            # save history
            war.rest.set_history()
```

plane.py

```
import random

import pygame

import constants
from game.bullet import Bullet

class Plane(pygame.sprite.Sprite):
    # pic of the plane
    plane_images = []
    # pic when plane explode
    destroy_images = []
    # explode m
    down_sound_src = None
    # status of the flight True:alive False: died
    active = True
```

```
    bullets = pygame.sprite.Group()

    def __init__(self, screen, speed=None):
        super().__init__()
        self.screen = screen
        # load static resources
        self.img_list = []
        self._destroy_img_list = []
        self._hit_img_list = []
        self.hit_sound = None
        self.down_sound = None
        self.load_src()

        # flying speed
        self.speed = speed or 15
        # get plane's position
        self.rect = self.img_list[0].get_rect()

        # plane's width and height
        self.plane_w, self.plane_h = self.img_list[0].get_size()

        # game window's width and height
        self.width, self.height = self.screen.get_size()

        # where plane apear
        self.rect.left = int((self.width - self.plane_w) / 2)
        self.rect.top = int(self.height / 2)

    def load_src(self):
        # pic of the plane
        for img in self.plane_images:
            self.img_list.append(pygame.image.load(img))
        # explode pic
        for img in self.destroy_images:
            self._destroy_img_list.append(pygame.image.load(img))
        # explode m
        if self.down_sound_src:
            self.down_sound = pygame.mixer.Sound(self.down_sound_src)

    @property
    def image(self):
        return self.img_list[0]
```

```
    def blit_me(self):
        self.screen.blit(self.image, self.rect)
    """action after user control """
    def move_up(self):
        self.rect.top -= self.speed

    def move_down(self):
        self.rect.top += self.speed

    def move_left(self):
        self.rect.left -= self.speed

    def move_right(self):
        self.rect.left += self.speed

    def broken_down(self):
        # play music
        if self.down_sound:
            self.down_sound.play()
        # show pic
        for img in self._destroy_img_list:
            self.screen.blit(img, self.rect)
        # plane died
        self.active = False

    def shoot(self):
        bullet = Bullet(self.screen, self, 15)
        self.bullets.add(bullet)


class OurPlane(Plane):
    # plane pic
    plane_images = constants.OUR_PLANE_IMG_LIST
    # explode pic
    destroy_images = constants.OUR_DESTROY_IMG_LIST
    # explode m
    down_sound_src = None

    def update(self, war):
        self.move(war.key_down)
        # switch pic, make the plane move
```

```
        if war.frame % 5:
            self.screen.blit(self.img_list[0], self.rect)
        else:
            self.screen.blit(self.img_list[1], self.rect)
        # check if it gets hit
        rest = pygame.sprite.spritecollide(self, war.enemies, False)
        if rest:
            # game end
            war.status = war.OVER
            war.enemies.empty()
            war.small_enemies.empty()
            war.middle_enemies.empty()
            self.broken_down()

    def move(self, key):
        """ user control """
        if key == pygame.K_w or key == pygame.K_UP:
            self.move_up()
        elif key == pygame.K_s or key == pygame.K_DOWN:
            self.move_down()
        elif key == pygame.K_a or key == pygame.K_LEFT:
            self.move_left()
        elif key == pygame.K_d or key == pygame.K_RIGHT:
            self.move_right()

    def move_up(self):
        # make sure is in the window
        super().move_up()
        if self.rect.top <= 0:
            self.rect.top = 0

    def move_down(self):
        super().move_down()
        if self.rect.top >= self.height - self.plane_h:
            self.rect.top = self.height - self.plane_h

    def move_left(self):
        super().move_left()
        if self.rect.left <= 0:
            self.rect.left = 0

    def move_right(self):
```

```
        super().move_right()
        if self.rect.left >= self.width - self.plane_w:
            self.rect.left = self.width - self.plane_w


class SmallEnemyPlane(Plane):
    # pics and music for small enemies
    plane_images = constants.SMALL_ENEMY_PLANE_IMG_LIST
    destroy_images = constants.SMALL_ENEMY_DESTROY_IMG_LIST
    down_sound_src = constants.SMALL_ENEMY_PLANE_DOWN_SOUND

    def __init__(self, screen, speed):
        super().__init__(screen, speed)
        self.init_pos()

    # random position
    def init_pos(self):
        self.rect.left = random.randint(0, self.width - self.plane_w)
        self.rect.top = random.randint(-5 * self.plane_h, -self.plane_h)

    def update(self, *args):
        super().move_down()

        # put on the screen
        self.blit_me()

        # reuse the plane
        if self.rect.top >= self.height:
            self.active = False
            # self.kill()
            self.reset()

    def reset(self):
        # make died plane alive again
        self.active = True
        # change position
        self.init_pos()

    def broken_down(self):
        super().broken_down()
        self.reset()
```

```
class MiddleEnemyPlane(Plane):
    # pics and music for small enemies
    # similar with the small flight but different speed
    plane_images = constants.MIDDLE_ENEMY_PLANE_IMG_LIST
    destroy_images = constants.MIDDLE_ENEMY_DESTROY_IMG_LIST
    down_sound_src = constants.MIDDLE_ENEMY_PLANE_DOWN_SOUND
    life = 0

    def __init__(self, screen, speed):
        super().__init__(screen, speed)
        self.init_pos()

    def init_pos(self):
        # random position
        self.rect.left = random.randint(0, self.width - self.plane_w)
        self.rect.top = random.randint(-15 * self.plane_h, -self.plane_h)

    def update(self, *args):
        super().move_down()

        self.blit_me()

        if self.rect.top >= self.height:
            self.active = False
            # self.kill()
            self.reset()

    def reset(self):
        self.active = True
        self.init_pos()

    def broken_down(self):
        super().broken_down()
        self.reset()
```

war.py

```
import sys

import constants
from game.plane import OurPlane, SmallEnemyPlane, MiddleEnemyPlane
```

```
from store.result import PlayRest


class PlaneWar(object):
    # game statuse
    READY = 0
    PLAYING = 1
    OVER = 2
    status = READY

    our_plane = None

    frame = 0  # frame rate

    small_enemies = pygame.sprite.Group()
    middle_enemies = pygame.sprite.Group()
    enemies = pygame.sprite.Group()

    # game result
    rest = PlayRest()

    def __init__(self):
        # initialize the game
        pygame.init()
        self.width, self.height = 480, 852
        # make screen
        self.screen = pygame.display.set_mode((self.width, self.height))
        # window's title
        pygame.display.set_caption('Avro Arrow')

        # load back ground pic
        self.bg = pygame.image.load(constants.BG_IMG)
        self.bg_over = pygame.image.load(constants.BG_IMG_OVER)
        # game title
        self.img_game_title = pygame.image.load(constants.IMG_GAME_TITLE)
        self.img_game_title_rect = self.img_game_title.get_rect()
        # width and height
        t_width, t_height = self.img_game_title.get_size()
        self.img_game_title_rect.topleft = (int((self.width - t_width) / 2),
                                            int(self.height / 2 - t_height))
        # start botton
        self.btn_start = pygame.image.load(constants.IMG_GAME_START_BTN)
```

```
        self.btn_start_rect = self.btn_start.get_rect()
        btn_width, btn_height = self.btn_start.get_size()
        self.btn_start_rect.topleft = (int((self.width - btn_width) / 2),
                                       int(self.height / 2 + btn_height))
        # text setting
        self.score_font = pygame.font.SysFont('arial', 88)

        # load bgm
        pygame.mixer.music.load(constants.BG_MUSIC)
        pygame.mixer.music.play(-1)         # repeat bgm
        pygame.mixer.music.set_volume(0.2)  # set volume

        # our plane setting
        self.our_plane = OurPlane(self.screen, speed=10)

        self.clock = pygame.time.Clock()

        self.key_down = None

    def bind_event(self):
        for event in pygame.event.get():
            # quit game
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            elif event.type == pygame.MOUSEBUTTONDOWN:
                # click to go into the game
                if self.status == self.READY:
                    self.status = self.PLAYING
                    self.rest.score = 0
                elif self.status == self.PLAYING:
                    # click mouse the shoot
                    self.our_plane.shoot()
                elif self.status == self.OVER:
                    self.status = self.READY
                    self.add_small_enemies(6)
            elif event.type == pygame.KEYDOWN:
                # key board control
                self.key_down = event.key
                # control by arrows and awsd
                if self.status == self.PLAYING:
                    if event.key == pygame.K_w or event.key == pygame.K_UP:
```

```
                    self.our_plane.move_up()
                elif event.key == pygame.K_s or event.key == pygame.K_DOWN:
                    self.our_plane.move_down()
                elif event.key == pygame.K_a or event.key == pygame.K_LEFT:
                    self.our_plane.move_left()
                elif event.key == pygame.K_d or event.key == pygame.K_RIGHT:
                    self.our_plane.move_right()
                elif event.key == pygame.K_SPACE:
                    # shoot
                    self.our_plane.shoot()

    def add_small_enemies(self, num):
        # add 6 small enemies, speed 4
        for i in range(num):
            plane = SmallEnemyPlane(self.screen, 4)
            plane.add(self.small_enemies, self.enemies)

    def add_middle_enemies(self, num):
        # add 8 middle enemies, speed 8
        for i in range(num):
            plane = MiddleEnemyPlane(self.screen, 8)
            plane.add(self.middle_enemies, self.enemies)

    def run_game(self):
        while True:
            # frame rate
            self.clock.tick(60)
            self.frame += 1
            if self.frame >= 60:
                self.frame = 0
            self.bind_event()

            # update status
            if self.status == self.READY:
                # show background
                self.screen.blit(self.bg, self.bg.get_rect())
                # title
                self.screen.blit(self.img_game_title, self.img_game_title_rect)
                # start button
                self.screen.blit(self.btn_start, self.btn_start_rect)
                self.key_down = None
            elif self.status == self.PLAYING:
```

```
            # in the game
            self.screen.blit(self.bg, self.bg.get_rect())
            # show plane
            self.our_plane.update(self)
            # show bul;lets
            self.our_plane.bullets.update(self)
            # show enemies
            self.small_enemies.update()
            self.middle_enemies.update()
            # score
            score_text = self.score_font.render(
                'score: {0}'.format(self.rest.score),
                False,
                constants.TEXT_SOCRE_COLOR
            )
            self.screen.blit(score_text, score_text.get_rect())

        elif self.status == self.OVER:
            # game end
            # background of the ending
            self.screen.blit(self.bg_over, self.bg_over.get_rect())
            # score count
            # total score
            score_text = self.score_font.render(
                '{0}'.format(self.rest.score),
                False,
                constants.TEXT_SOCRE_COLOR
            )
            score_text_rect = score_text.get_rect()
            text_w, text_h = score_text.get_size()
            # text position
            score_text_rect.topleft = (
                int((self.width - text_h) / 2),
                int(self.height / 2 + 30)
            )
            self.screen.blit(score_text, score_text_rect)
            # history score
            score_his = self.score_font.render(
                '{0}'.format(self.rest.get_max_core()),
                False,
                constants.TEXT_SOCRE_COLOR
            )
```

```
            self.screen.blit(score_his, (288, 40))

        pygame.display.flip()
```

result.py

```
import constants

# players score result


class PlayRest(object):
    __score = 0
    __life = 3
    __blood = 1000

    @property
    def score(self):
        return self.__score

    @score.setter
    def score(self, value):
        if value < 0:
            return None
        self.__score = value

    def set_history(self):
        if int(self.get_max_core()) < self.score:
            with open(constants.PLAY_RESULT_STORE_FILE, 'w') as f:
                f.write('{0}'.format(self.score))

    def get_max_core(self):
        rest = 0
        with open(constants.PLAY_RESULT_STORE_FILE, 'r') as f:
            r = f.read()
            if r:
                rest = r
        return rest
```

constants.py

```
import os
```

```
import pygame

# root of the project
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
# static files directory
ASSETS_DIR = os.path.join(BASE_DIR, 'static_resources')

# background pic
BG_IMG = os.path.join(ASSETS_DIR, 'images/background.png')
BG_IMG_OVER = os.path.join(ASSETS_DIR, 'images/game_over.png')
# title pic
IMG_GAME_TITLE = os.path.join(ASSETS_DIR, 'images/game_title.png')
# start btn
IMG_GAME_START_BTN = os.path.join(ASSETS_DIR, 'images/game_start.png')
# bgm
BG_MUSIC = os.path.join(ASSETS_DIR, 'bgm/game.wav')
# colore of the text
TEXT_SOCRE_COLOR = pygame.Color(255, 255, 255)

# score when hit the enemise
SCORE_SHOOT_SMALL = 10
SCORE_SHOOT_MIDDLE = 15
# score result
PLAY_RESULT_STORE_FILE = os.path.join(BASE_DIR, 'store/rest.txt')


# static file of our plane
OUR_PLANE_IMG_LIST = [
    os.path.join(ASSETS_DIR, 'images/hero1.png'),
    os.path.join(ASSETS_DIR, 'images/hero2.png')
]
OUR_DESTROY_IMG_LIST = [
    os.path.join(ASSETS_DIR, 'images/hero_broken1.png'),
    os.path.join(ASSETS_DIR, 'images/hero_broken2.png'),
    os.path.join(ASSETS_DIR, 'images/hero_broken3.png'),
    os.path.join(ASSETS_DIR, 'images/hero_broken4.png'),
]

# bullet oic and sound
BULLET_IMG = os.path.join(ASSETS_DIR, 'images/bullet1.png')
ENEMISE_BULLET_IMG = os.path.join(ASSETS_DIR, 'images/bullet2.png')
BULLET_SHOOT_SOUND = os.path.join(ASSETS_DIR, 'bgm/bullet.wav')
```

```
# small enemise info
SMALL_ENEMY_PLANE_IMG_LIST = [os.path.join(ASSETS_DIR, 'images/enemy1.png')]
SMALL_ENEMY_DESTROY_IMG_LIST = [
    os.path.join(ASSETS_DIR, 'images/enemy1_broken1.png'),
    os.path.join(ASSETS_DIR, 'images/enemy1_broken2.png'),
    os.path.join(ASSETS_DIR, 'images/enemy1_broken3.png'),
    os.path.join(ASSETS_DIR, 'images/enemy1_broken4.png'),
]
# small enemise falling sound
SMALL_ENEMY_PLANE_DOWN_SOUND = os.path.join(ASSETS_DIR, 'bgm/enemy1_down.wav')


# middle enemies info
MIDDLE_ENEMY_PLANE_IMG_LIST = [os.path.join(ASSETS_DIR, 'images/enemy2.png')]
MIDDLE_ENEMY_DESTROY_IMG_LIST = [
    os.path.join(ASSETS_DIR, 'images/enemy2_broken1.png'),
    os.path.join(ASSETS_DIR, 'images/enemy2_broken2.png'),
    os.path.join(ASSETS_DIR, 'images/enemy2_broken3.png'),
    os.path.join(ASSETS_DIR, 'images/enemy2_broken4.png'),
]
# middle enemied music
MIDDLE_ENEMY_PLANE_DOWN_SOUND = os.path.join(ASSETS_DIR, 'bgm/enemy2_down.wav')
```

main.py

```
from game.war import PlaneWar

def main():
    # entre the game
    war = PlaneWar()
    # add the enemies
    war.add_small_enemies(6)
    war.add_middle_enemies(4)
    war.run_game()


if __name__ == '__main__':
    main()
```

5. Set Up Instruction

1) Download and install python3.7 interpreter from: https://www.python.org/downloads/
2) Add Python 3.7 to PATH when you install(Windows). There will be an option on the bottom of the window.
3) Download and install PyCharm from: https://www.jetbrains.com/pycharm/
4) Import pygame in your laptop

Windows:

Open up your Comment Prompt.

Input: "pip" enter

Input: "install pygame" enter

Mac:

Open Terminal

Input "curl https://bootstrap.pypa.io/get-pip.py > get-pip.py" enter

Input "sudo pip install pygame" enter

Type your password in. You will not see your password show on the screen. Once your type it in, Just hit enter.

5) Release the .zip file in your laptop.
6) Open the file with PyCharm (If you use Mac or Linux, you can just run the file in the Terminal by changing the classpath to python)
7) Go into main.py
8) Right click your mouse
9) Choose "Run 'main' "

Tip: For the **MacOS Mojave** users, the window may be blank when you run "main." Follow this instruction to run the game.

1) Download and install Miniconda from: https://conda.io/miniconda.html
2) Run sh in your Terminal: "bash Miniconda3-latest-MacOSX-x86_64.sh"

3) Reopen the Terminal and check if onda is in your laptop or not by type “conda --version” (If it works, it will shows the version of conda)
4) Create a new environment called snakes by typing in “conda create --name snakes python=3.7”
5) Input “source activate snakes” to activate the new virtual environment
6) Run the code.
7) When you finish testing, type “source deactivate” to return back to the original environment.

6. Timeline

May 1 - draft timeline complete

May 23 - research complete

June 6 - draft of specific document complete

First working code in test cycle

Test plan draft complete

June 9 - Second prototype complete

June 12 - Final testing cycle and refinement of programme

June 15 - Final specific document complete

Presentation and Reflection complete

June 16 - Complete the final copy of documentation

June 17 - Present to the class

7. References

https://www.youtube.com/watch?v=AdUZArA-kZw

https://www.youtube.com/watch?v=E-WhAS6qzsU

https://www.dl-sounds.com/royalty-free/off-limits/

8. Reflection

April:

I was a little bit shocked by what we are going to do for our final project, which is using another language to make something. I had no idea what should I make. I thought a lot and finally decided that I am going to use python which I used before. However, I was struggling with either making a web crawler, a website or a game.

Trying something new was my goal for the project because I want to learn more skills. I made crawler and website before, which would not be that challenging. Moreover, I found out there were many people in our class who wanted to make a website. Therefore, I chose an area which I have never involved before--game.

May:

At the beginning of this month, I watched some tutorial videos online discontinuity because I want to prepare my AP exam at the same time, and I was afraid to mix up Java and Python together.

After my AP, I decided to code something easy with the tutorial first, but I met my first biggest challenge also. My pygame doesn't work on my laptop. It's because my laptop's system is MacOS Mojave, which is not compatible with pygame. To solve this problem, I did much research and asked many people for help. I follow the instructions above to run my pygame in virtual environment.

June:

I finished half of the code at the beginning of this month, but I haven't tested them yet because I didn't have picture and music for my game at that time. Therefore, these codes were mostly my planning. It was not impressed when I found out there were many bugs in my code after I prepared all the static sources I need.

The second challenge I had was about the background music. I found some pieces of music in .mp3 form and put them into my static sources file. After I ran my

code, the music didn't play on my laptop. I checked my laptop's volume first, it was good. Then, I check my codes, I found out a little bug -- there was no ".music" after "mixer." When I thought I fixed it, it still didn't work. I struggled with that for an afternoon, checked my code again and again, and did research online. In the end, I found out the reason on Pygame's website. Pygame.mixer only accepts the audio in .wav and .ogg form.

I made many errors other than the two I listed above. Honestly, I am not a very good programmer, but I will keep improving myself until the day my works can win the approval from people.