BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:		**Surname**	**Given Names**	**Student ID Number**
	1.	**Revano Chen**	**Dumac**	**2101699653**

Course Code	**: COMP6502**	**Course Name**	**: Introduction to Programming**
Class	**: L1AC**	**Name of Lecturer(s)**	: 1. Ida Bagus Kerthyayana 2. Tri Asih Budiono
Major	**: CS**		
Title of Assignment (if any)	: Jet Mission (Video Game)		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 7-11-2017**	**Submission Date**	**: 7-11-2017**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:
(Name of Student)

Dumac Revano Chen

“Jet Mission”
Student Name : Dumac Revano Chen

Student Id : 2101699653

I. Introduction

This game is created based on the simple aircraft shooting game. The objective of this game is to get a high score by shooting the meteor down and prevent getting struck by the meteor itself. The movement of the aircraft is based on the keyboard key pressed. This game run based on python 3 and Pygame, and the code used for this game is not copied from any other similar game that can be found on the online platform.

II. Game’s files and code review

Classes.py

This custom build module consists of a few classes which is used as the base of creating the objects in the game. These classes are built to be as flexible as possible so if any improvements are needed. Hence, redoing the whole code can be avoided. Here are the explanations for each class in this file:

Jet Class

This class is used as the base of the jet which players will control. This class affect the size, image, speed and the bullets that the jet will fire. The parameter screen in this class is used to decide on which surface the jet will be drawn. Besides this class also have four methods called as moveleft, moveup, moveright, movedown which control the movement of jet. Jet class also inherited the Sprite which is provided by the pygame library.

```
class Jet(Sprite):
    """initialize the jet"""

    def __init__(self, screen):
        Sprite.__init__(self)
        """initialize the Jet"""
        self.image = image.load("battlejet.png")
        self.image = pygame.transform.scale(self.image, (90, 50))
        self.rect = self.image.get_rect()
        self.rect.x = 10
        self.rect.y = 50
        self.screen = screen
        self.move_speed = 6
        """bullet"""
        self.firerates = 2
```

```
    def moveleft(self):
        self.rect.x -= self.move_speed
        display.flip()

    def moveright(self):
        self.rect.x += self.move_speed
        display.flip()

    def moveup(self):
        self.rect.y -= self.move_speed
        display.flip()

    def movedown(self):
        self.rect.y += self.move_speed
        display.flip()
```

Example of calling the jet class and the inherited method :

```
#creating a jet
jet1 = Jet(screen)
Jet_sprites = Group(jet1)
```

Example of jet method usage:

```
event.get()
"""moving the jet according to key pressed"""

key = pygame.key.get_pressed()
if key[K_LEFT] and jet1.rect.x>0:
    jet1.moveleft()

if key[K_RIGHT] and jet1.rect.x<=700:
    jet1.moveright()

if key[K_DOWN] and jet1.rect.y<=500:
    jet1.movedown()

if key[K_UP] and jet1.rect.y>0:
    jet1.moveup()
```

Star_bg Class

This “Star_bg” class is used as the base for the background in the whole game play, and it has a parameter background which affect the image displayed. This class has a method called as draw which will draw the background image too the screen. There are three parameters inside the draw method, screen parameter will decide on which surface the image will be drawn, x and y decide the coordinate position of the image.

```
class Star_bg:
    #resourse of the backgound setting
    def __init__(self,background):
        self.background=image.load(background)
        self.background=pygame.transform.scale(self.background,(800,600))
        self.background_size=self.background.get_size()
        self.background_rect=self.background.get_rect()
        self.width,self.height=self.background_size
    def draw(self,screen,x,y):
        screen.blit(self.background,(x,y))
```

Example of the star_bg class usage:

```
bg_image = Star_bg("star.gif")
```

Example of the star_bg’s draw method usage:

```
    """background move"""

    x -= 5
    x1 -= 5
    bg_image.draw(screen,x,y)
    bg_image.draw(screen,x1, y1)
    if x < -bg_image.width:
        x = 0
    if x1 < 0:
        x1 = bg_image.width
```

Bullet Class

This bullet class is used as the base of the bullet that is shot from the jet itself. This class has attribute to control the image, rectangle, surface, speed and almost anything related to bullets behavior. The parameter screen decides on which surface the bullet will be drawn, while startx and starty parameter decide the coordinate position where the bullet appear for the first time after it is called. This class also consist of method which will move the bullet's position like how a bullet used to move. Besides this class inherited the Sprite so it can use any method on the Sprites.

```
class Bullet(Sprite):
    def __init__(self,screen, startx, starty):
        Sprite. __init__(self)
        self.startx = startx
        self.starty = starty

        self.speedx = 20

        self.image = pygame.image.load("bullets.png")
        self.image = pygame.transform.scale(self.image,(40,40))
        self.rect=self.image.get_rect()
        self.rect.left = startx
        self.rect.top = starty
        self.rect.center = (startx,starty)
        self.screen = screen
    def movement(self):
        #self.screen.blit(self.image,[self.startx,self.starty])
        self.rect.left += self.speedx
```

Example of bullet class usage:

```
if key[K_SPACE] and len(bullets) <= jet1.firerates+(scores/4000):
    bullet = Bullet(screen, jet1.rect.x+50, jet1.rect.y+42)
    bullets.add(bullet)
```

Asteroid Class

Asteroid Class Basically has the same concept with the bullet class, however instead of having a fixed speed Asteroid speed is determined by a parameter.

```
class Asteroid(Sprite):
    """initialize the Asteroid"""
    def __init__(self, screen, width, height, speedx, startx, starty):
        Sprite.__init__(self)
        self.startx = startx
        self.starty = starty

        self.speedx = speedx

        self.image = pygame.image.load("meteor.png")
        self.image = pygame.transform.scale(self.image, (width, height))
        self.rect = self.image.get_rect()
        self.rect.left = startx
        self.rect.top = starty
        self.screen = screen

    def movement(self):
        """method to move the Asteoid"""
        self.rect.left -= self.speedx
```

Example of Asteroid class usage:

```
    """generate asteroid randomly"""
    if pygame.time.get_ticks() - asteroid_timer >= 200:
        asteroid = Asteroid(screen, 50, 50, random.randint(1,4)*6, 800, (random.randint(1,28) * 20))
        asteroid_group.add(asteroid)
        asteroid_timer = pygame.time.get_ticks()

    """update the movement of asteroid"""
    for asteroid in asteroid_group:
        asteroid.movement()
        if asteroid.rect.right <= 0:
            asteroid_group.remove(asteroid)
        if groupcollide(Jet_sprites,asteroid_group,dokilla=True,dokillb=True):
            menu.lose_menu(Button,run_game,scores)
```

Button class

This class is used as the base to insert the button image in the game menu, the image parameter decide which image will be used as the image of the button.

```
class Button(Sprite):
    """initialize the button"""
    def __init__(self,image):
        Sprite. __init__(self)
        self.button=pygame.image.load(image)
        self.button=pygame.transform.scale(self.button,(300,150))
```

Example of button class usage in the menu:

```
# object button for quit and start
start_button = Button("quit button.png")
return_button = Button("pause button.png")
```

This file includes of some custom build function that specialized on displaying menu interface. To help the menu function some outside library like sys also imported. Here are the explanation of the each function contained in the menu.py.

menu_screen function

This function is used to make the first menu which player will encounter after the game run. This function also has two parameters with it, Button parameter is a parameter which must be filled with class name which will determine the button displayed in the screen, while run_game button will decide what action will be done after specified user event detected.

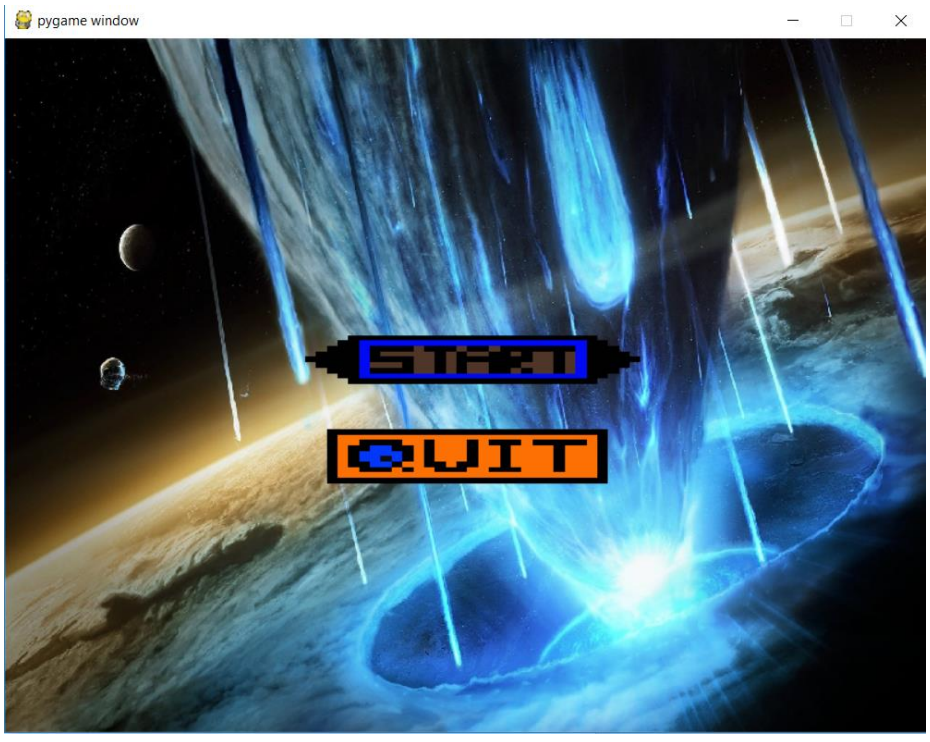
```
from pygame import *
import sys
import pygame

def menu_screen(Button,run_game):
    """make the screen for menu"""
    display.set_caption("Jet Mission")
    screen = pygame.display.set_mode((800, 600))
    #object button for quit and start
    start_button = Button("New Piskel.png")
    quit_button = Button("quit button.png")
    #image for the menu's backgound
    bg_image=pygame.image.load("asteroid_wall.jpg")
    bg_image=pygame.transform.scale(bg_image, (800, 600))
```

```
pygame.init()

while True:
    rect_start= draw.rect(screen, (0, 0, 0), (250, 200, 300, 150))
    rect_quit = draw.rect(screen, (0, 0, 0), (250, 300, 300, 150))
    screen.blit(bg_image,(0,0))

    screen.blit(start_button.button,(250,200))
    screen.blit(quit_button.button,(250,300))

    ev=event.wait()

    if ev.type == MOUSEBUTTONDOWN:
        if rect_start.collidepoint(mouse.get_pos()):
            run_game()
        if rect_quit.collidepoint(mouse.get_pos()):
            sys.exit()

    if ev.type == QUIT:
        sys.exit()

    display.update()
```

Example of called menu_screen function:

```
menu.menu_screen(Button,run_game)
```

when code is executed window like below will appear on the screen



lose_menu function

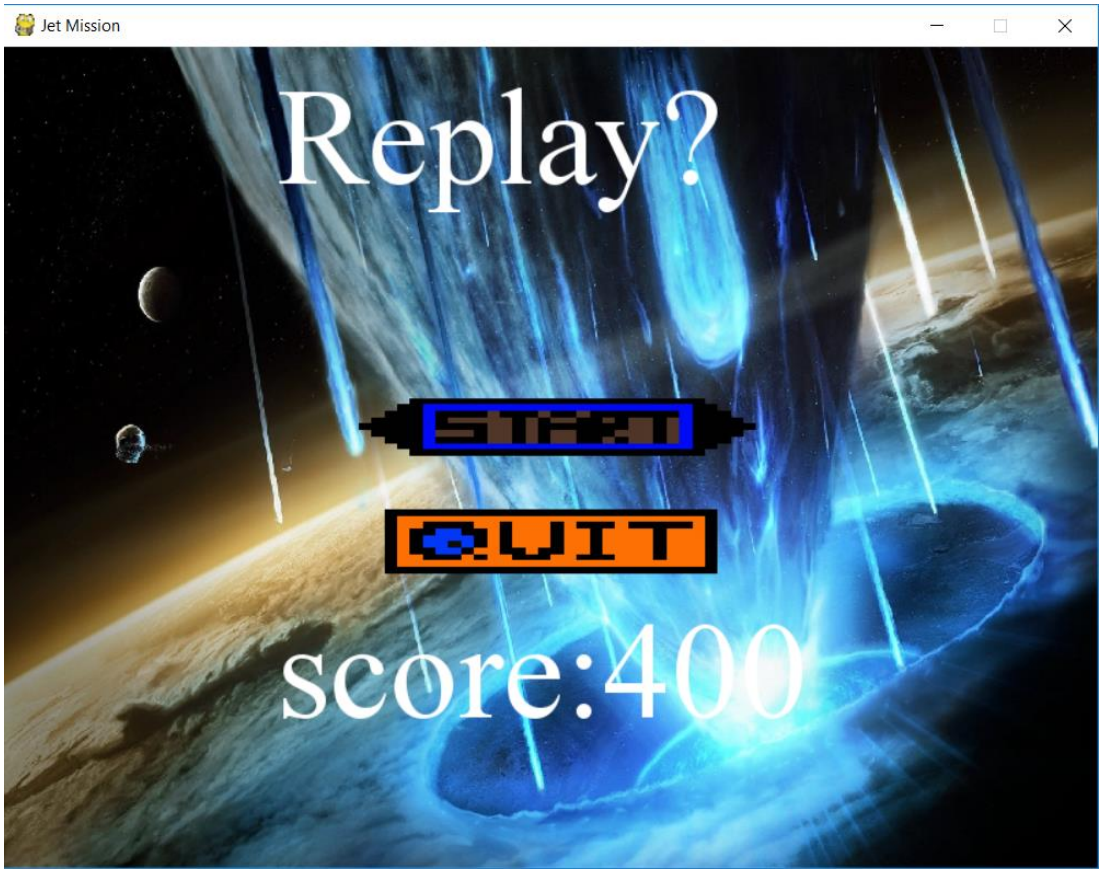
this function is a function that is used to create the losing menu, just like the menu_screen this menu consists of display set which is part of pygame library. Though it is almost the same with the menu_screen function this menu has a parameter called score, this parameter will be filled with an interger variable. This interger variable will then be drawn onto the screen as final results.

```
def lose_menu(Button,run_game,score):
    """make the screen for menu"""
    display.set_caption("Jet Mission")
    screen = pygame.display.set_mode((800, 600))
    font=pygame.font.SysFont("times new roman",100)
    text=font.render("Replay?",True,(255,255,255))
    score_text=font.render("score:"+str(score),True,(255,255,255))

    # object button for quit and start
    start_button = Button("New Piskel.png")
    quit_button = Button("quit button.png")

    # image for the menu's backgound
    bg_image = pygame.image.load("asteroid_wall.jpg")
    bg_image = pygame.transform.scale(bg_image, (800, 600))

    pygame.init()

    while True:
        rect_start = draw.rect(screen, (0, 0, 0), (250, 200, 300, 150))
        rect_quit = draw.rect(screen, (0, 0, 0), (250, 300, 300, 150))
        screen.blit(bg_image, (0, 0))
        screen.blit(text,(200,10))
        screen.blit(start_button.button, (250, 200))
        screen.blit(quit_button.button, (250, 300))
        screen.blit(score_text,(200,400))

        ev = event.wait()

        if ev.type == MOUSEBUTTONDOWN:
            if rect_start.collidepoint(mouse.get_pos()):
                run_game()
            if rect_quit.collidepoint(mouse.get_pos()):
                sys.exit()

        if ev.type == QUIT:
            sys.exit()

        display.update()
```

Example of called lose_menu function:

```
if groupcollide(Jet_sprites,asteroid_group,dokilla=True,dokillb=True):
    menu.lose_menu(Button,run_game,scores)
```

If this function is triggered a window similar with the picture below will be shown:



Pause_menu function

Theoretically this function work just like two menu's functions before. However at here to trigger the program, flagging is used in order to close the program, so that it didn't affect the main game.

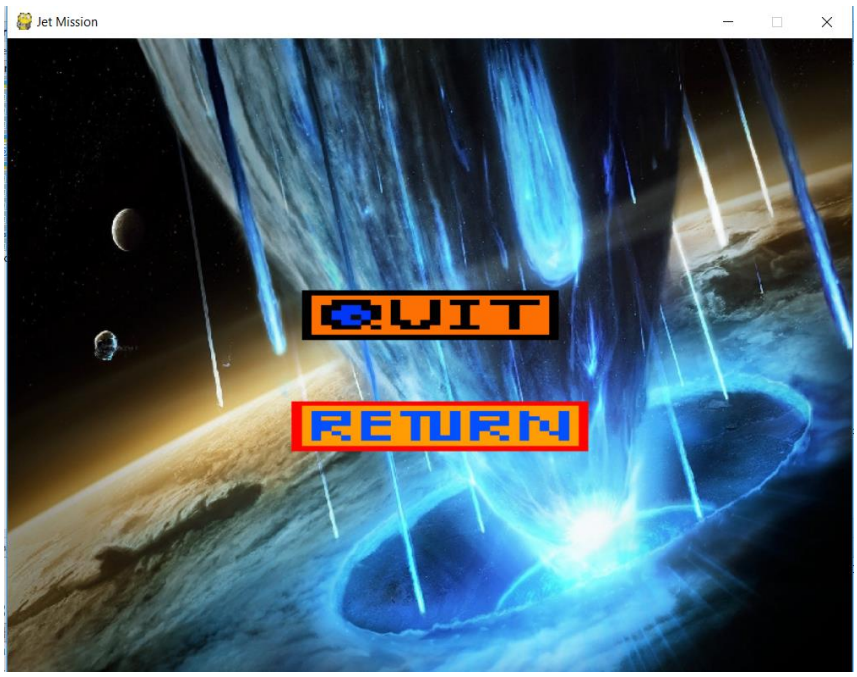
```
def pause_menu(Button,run_game):
    """pause_menu"""
    #make the screen display
    display.set_caption("Jet Mission")
    screen = pygame.display.set_mode((800, 600))

    # object button for quit and start
    start_button = Button("quit button.png")
    return_button = Button("pause button.png")

    # image for the menu's backgound
    bg_image = pygame.image.load("asteroid_wall.jpg")
    bg_image = pygame.transform.scale(bg_image, (800, 600))


    pygame.init()
    paused=True #pause flag
```

```
    while paused:
        rect_start = draw.rect(screen, (0, 0, 0), (250, 200, 300, 150))
        rect_return = draw.rect(screen, (0, 0, 0), (250, 300, 300, 150))
        screen.blit(bg_image, (0, 0))

        screen.blit(start_button.button, (250, 200))
        screen.blit(return_button.button, (250, 300))

        ev = event.wait()

        if ev.type == MOUSEBUTTONDOWN:
            if rect_start.collidepoint(mouse.get_pos()):
                menu_screen(Button,run_game)
            if rect_return.collidepoint(mouse.get_pos()):
                paused = False #flag become  False

        if ev.type == QUIT:
            sys.exit()


        display.update()
```

The example of pause_menu called function:

```
if key[K_p]:
    menu.pause_menu(Button,run_game)
```

When this code is executed windows similar with below image will appear:



App.py

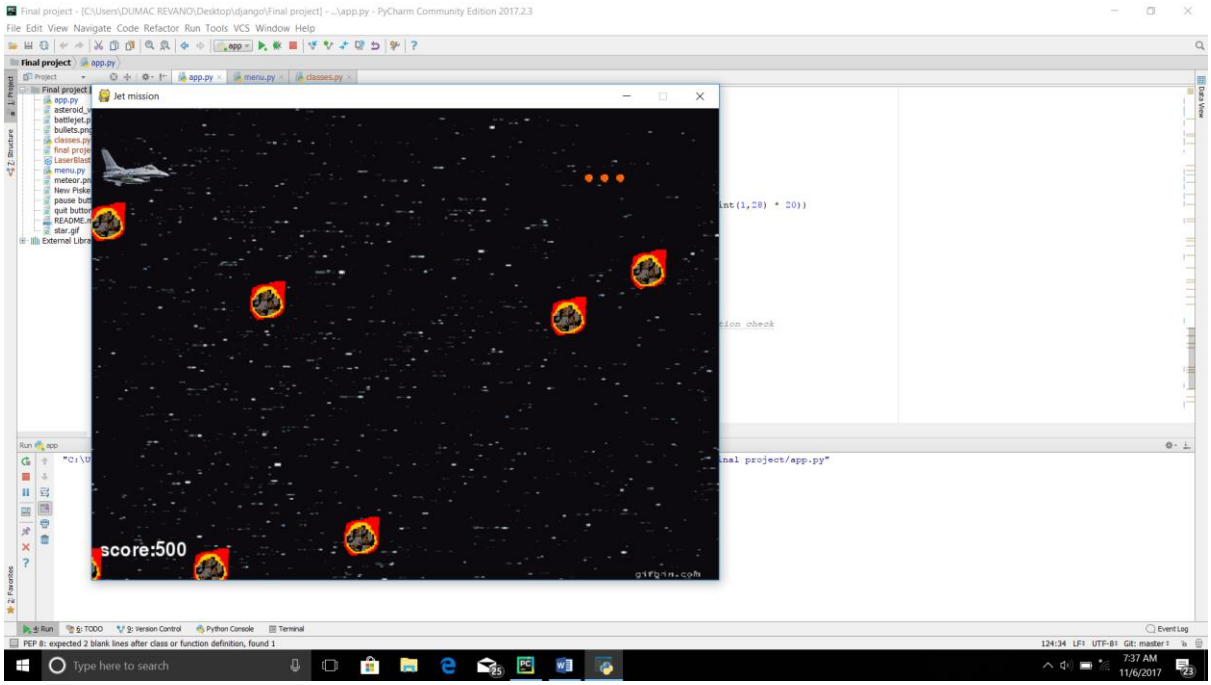
This python file consists only of one function, the run_game function. The run_game function control everything in the main game play. This function affect the moving background in the game, collide check amongst sprite,scoring, ext.

```
def run_game():
    """game play interface"""
    screen = pygame.display.set_mode((800, 600))
    display.set_caption("Jet mission")


    scores = 0
    theClock = pygame.time.Clock()
    bg_image = Star_bg("star.gif")

    #coordinate of moving background
    x = 0
    y = 0
    x1 = bg_image.width
    y1 = 0

    pygame.init()


    #creating a jet
    jet1 = Jet(screen)
    Jet_sprites = Group(jet1)

    #create asteroid object group
    asteroid_group = Group()

    #create bullets object Group
    bullets = Group()



    Fps = 40
    asteroid_timer = pygame.time.get_ticks()
    while True:
        theClock.tick(Fps)
        Fps += 0.01#game phase goes faster after every frame

        """background move"""
```

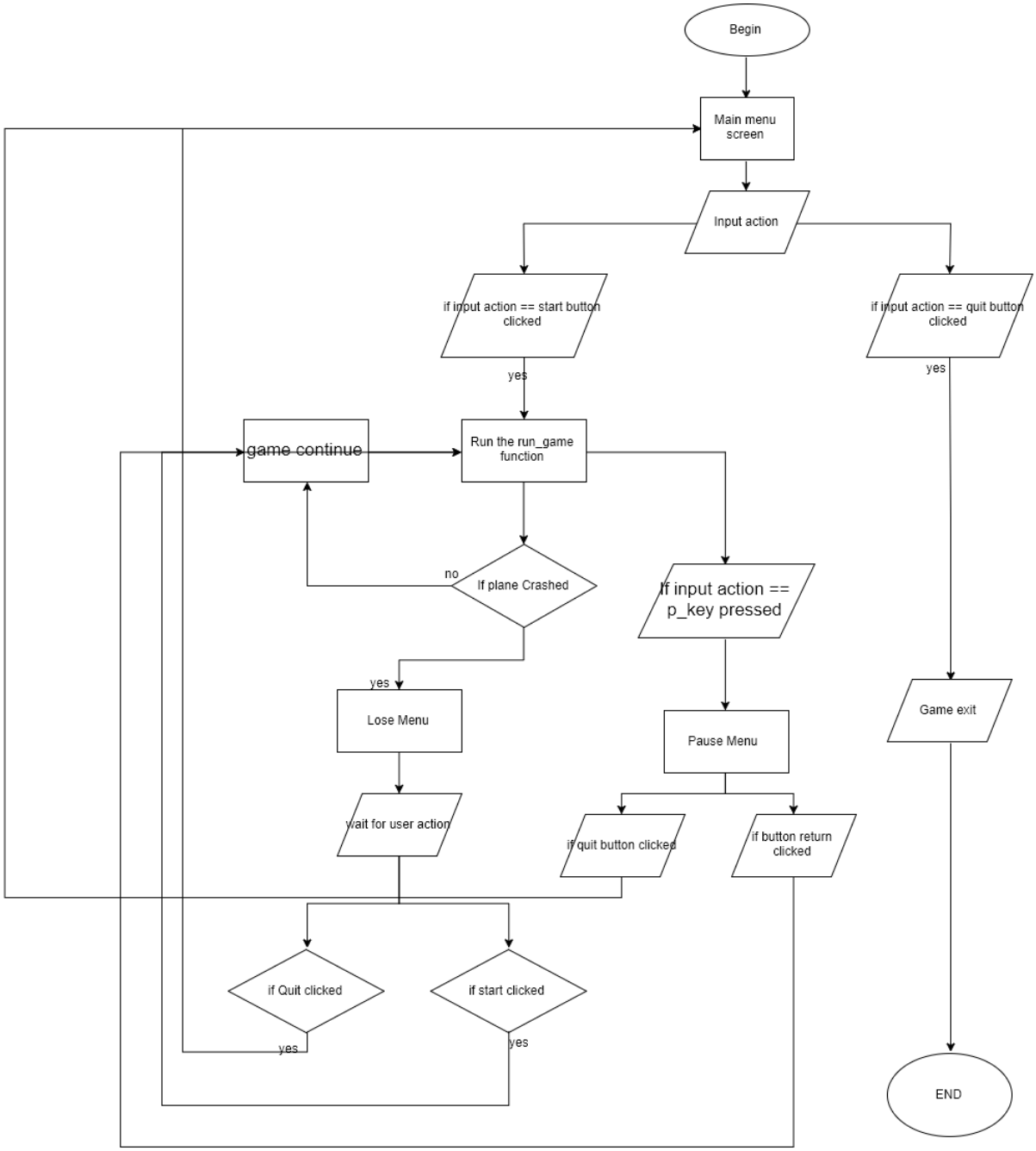
```
"""background move"""

x -= 5
x1 -= 5
bg_image.draw(screen,x,y)
bg_image.draw(screen,x1, y1)
if x < -bg_image.width:
    x = 0
if x1 < 0:
    x1 = bg_image.width

# create score board
font=pygame.font.SysFont("Times New Romans",36)
score_board=font.render("score:"+str(scores),True,(255,255,255))
# update refered to the word's method
screen.blit(score_board,(10,550))



Jet_sprites.draw(screen)

bullets.draw(screen)

asteroid_group.draw(screen)
display.update()#update jet and screen view

"""moving the jet according to key pressed"""

key = pygame.key.get_pressed()
if key[K_LEFT] and jet1.rect.x>0:
    jet1.moveleft()

if key[K_RIGHT] and jet1.rect.x<=700:
    jet1.moveright()

if key[K_DOWN] and jet1.rect.y<=500:
    jet1.movedown()

if key[K_UP] and jet1.rect.y>0:
    jet1.moveup()

if key[K_SPACE] and len(bullets) <= jet1.firerates+(scores/4000):
    bullet = Bullet(screen, jet1.rect.x+50, jet1.rect.y+42)
    bullets.add(bullet)
    pygame.mixer.music.load("LaserBlast.wav")
    pygame.mixer.music.play()

if key[K_ESCAPE]:
    menu.menu_screen(Button,run_game)

if key[K_p]:
    menu.pause_menu(Button,run_game)


"""generate asteroid randomly"""
if pygame.time.get_ticks() - asteroid_timer >= 200:
    asteroid = Asteroid(screen, 50, 50, random.randint(1,4)*6, 800, (random.randint(1,28) * 20))
    asteroid_group.add(asteroid)
    asteroid_timer = pygame.time.get_ticks()
```

```
        """update the movement of asteroid"""
        for asteroid in asteroid_group:
            asteroid.movement()
            if asteroid.rect.right <= 0:
                asteroid_group.remove(asteroid) #remove after screen
            if groupcollide(Jet_sprites,asteroid_group,dokilla=True,dokillb=True):#collition check
                menu.lose_menu(Button,run_game,scores)

        """update bullet movement on screen"""
        for bullet in bullets:
            bullet.movement()
            if bullet.rect.left > 800:
                bullets.remove(bullet)
            if groupcollide(bullets,asteroid_group,dokilla=True,dokillb=True):
                scores += 100

menu.menu_screen(Button,run_game)

#------------------SPECIAL THANKS to Pier,Excel,georgius,William,Nicander,Nicolas,Andy,Guntur---------------
```

When this code is executed it will show the game play:



III. Game's flowchart



IV. Game's source code

Classes.py

```
from pygame import *
from pygame.sprite import *

class Jet(Sprite):
    """initialize the jet"""

    def __init__(self, screen):
        Sprite.__init__(self)
        """initialize the Jet"""
        self.image = image.load("battlejet.png")
        self.image = pygame.transform.scale(self.image, (90, 50))
        self.rect = self.image.get_rect()
        self.rect.x = 10
        self.rect.y = 50
        self.screen = screen
        self.move_speed = 6
        """bullet"""
        self.firerates = 2

    def moveleft(self):
        self.rect.x -= self.move_speed
        display.flip()

    def moveright(self):
        self.rect.x += self.move_speed
        display.flip()

    def moveup(self):
        self.rect.y -= self.move_speed
        display.flip()

    def movedown(self):
        self.rect.y += self.move_speed
        display.flip()




class Star_bg:
    #resourse of the backgound setting
    def __init__(self,background):
        self.background=image.load(background)
        self.background=pygame.transform.scale(self.background,(800,600))
        self.background_size=self.background.get_size()
        self.background_rect=self.background.get_rect()
        self.width,self.height=self.background_size
    def draw(self,screen,x,y):
        screen.blit(self.background,(x,y))

class Bullet(Sprite):
    def __init__(self,screen, startx, starty):
        Sprite. __init__(self)
        self.startx = startx
        self.starty = starty

        self.speedx = 20

        self.image = pygame.image.load("bullets.png")
        self.image = pygame.transform.scale(self.image,(40,40))
        self.rect=self.image.get_rect()
        self.rect.left = startx
```

```
        self.rect.top = starty
        self.rect.center = (startx,starty)
        self.screen = screen
    def movement(self):
        #self.screen.blit(self.image,[self.startx,self.starty])
        self.rect.left += self.speedx

class Asteroid(Sprite):
    """initialize the Asteroid"""
    def __init__(self, screen, width, height, speedx, startx, starty):
        Sprite.__init__(self)
        self.startx = startx
        self.starty = starty

        self.speedx = speedx

        self.image = pygame.image.load("meteor.png")
        self.image = pygame.transform.scale(self.image, (width, height))
        self.rect = self.image.get_rect()
        self.rect.left = startx
        self.rect.top = starty
        self.screen = screen

    def movement(self):
        """method to move the Asteoid"""
        self.rect.left -= self.speedx


class Button(Sprite):
    """initialize the button"""
    def __init__(self,image):
        Sprite. __init__(self)
        self.button=pygame.image.load(image)
        self.button=pygame.transform.scale(self.button,(300,150))
```

Menu.py

```
from pygame import *
import sys
import pygame

def menu_screen(Button,run_game):
    """make the screen for menu"""
    display.set_caption("Jet Mission")
    screen = pygame.display.set_mode((800, 600))
    #object button for quit and start
    start_button = Button("New Piskel.png")
    quit_button = Button("quit button.png")
    #image for the menu's backgound
    bg_image=pygame.image.load("asteroid_wall.jpg")
    bg_image=pygame.transform.scale(bg_image, (800, 600))


    pygame.init()

    while True:
        rect_start= draw.rect(screen, (0, 0, 0), (250, 200, 300, 150))
        rect_quit = draw.rect(screen, (0, 0, 0), (250, 300, 300, 150))
        screen.blit(bg_image,(0,0))

        screen.blit(start_button.button,(250,200))
        screen.blit(quit_button.button,(250,300))

        ev=event.wait()

        if ev.type == MOUSEBUTTONDOWN:
            if rect_start.collidepoint(mouse.get_pos()):
                run_game()
            if rect_quit.collidepoint(mouse.get_pos()):
                sys.exit()

        if ev.type == QUIT:
            sys.exit()

        display.update()

def pause_menu(Button,run_game):
    """pause_menu"""
    #make the screen display
    display.set_caption("Jet Mission")
    screen = pygame.display.set_mode((800, 600))

    # object button for quit and start
    start_button = Button("quit button.png")
    return_button = Button("pause button.png")

    # image for the menu's backgound
    bg_image = pygame.image.load("asteroid_wall.jpg")
    bg_image = pygame.transform.scale(bg_image, (800, 600))


    pygame.init()
    paused=True #pause flag
    while paused:
        rect_start = draw.rect(screen, (0, 0, 0), (250, 200, 300, 150))
        rect_return = draw.rect(screen, (0, 0, 0), (250, 300, 300, 150))
        screen.blit(bg_image, (0, 0))

        screen.blit(start_button.button, (250, 200))
        screen.blit(return_button.button, (250, 300))

        ev = event.wait()
```

```
        if ev.type == MOUSEBUTTONDOWN:
            if rect_start.collidepoint(mouse.get_pos()):
                menu_screen(Button,run_game)
            if rect_return.collidepoint(mouse.get_pos()):
                paused = False #flag become  False

        if ev.type == QUIT:
            sys.exit()


        display.update()

def lose_menu(Button,run_game,score):
    """make the screen for menu"""
    display.set_caption("Jet Mission")
    screen = pygame.display.set_mode((800, 600))
    font=pygame.font.SysFont("times new roman",100)
    text=font.render("Replay?",True,(255,255,255))
    score_text=font.render("score:"+str(score),True,(255,255,255))

    # object button for quit and start
    start_button = Button("New Piskel.png")
    quit_button = Button("quit button.png")

    # image for the menu's backgound
    bg_image = pygame.image.load("asteroid_wall.jpg")
    bg_image = pygame.transform.scale(bg_image, (800, 600))

    pygame.init()

    while True:
        rect_start = draw.rect(screen, (0, 0, 0), (250, 200, 300, 150))
        rect_quit = draw.rect(screen, (0, 0, 0), (250, 300, 300, 150))
        screen.blit(bg_image, (0, 0))
        screen.blit(text,(200,10))
        screen.blit(start_button.button, (250, 200))
        screen.blit(quit_button.button, (250, 300))
        screen.blit(score_text,(200,400))

        ev = event.wait()

        if ev.type == MOUSEBUTTONDOWN:
            if rect_start.collidepoint(mouse.get_pos()):
                run_game()
            if rect_quit.collidepoint(mouse.get_pos()):
                sys.exit()

        if ev.type == QUIT:
            sys.exit()

        display.update()
```

App.py

```
from pygame import *
import menu
import random

from classes import *



def run_game():
    """game play interface"""
    screen = pygame.display.set_mode((800, 600))
    display.set_caption("Jet mission")


    scores = 0
    theClock = pygame.time.Clock()
    bg_image = Star_bg("star.gif")

    #coordinate of moving background
    x = 0
    y = 0
    x1 = bg_image.width
    y1 = 0

    pygame.init()


    #creating a jet
    jet1 = Jet(screen)
    Jet_sprites = Group(jet1)

    #create asteroid object group
    asteroid_group = Group()

    #create bullets object Group
    bullets = Group()



    Fps = 40
    asteroid_timer = pygame.time.get_ticks()
    while True:
        theClock.tick(Fps)
        Fps += 0.01#game phase goes faster after every frame

        """background move"""

        x -= 5
        x1 -= 5
        bg_image.draw(screen,x,y)
        bg_image.draw(screen,x1, y1)
        if x < -bg_image.width:
            x = 0
        if x1 < 0:
            x1 = bg_image.width

        # create score board
        font=pygame.font.SysFont("Times New Romans",36)
        score_board=font.render("score:"+str(scores),True,(255,255,255))
        # update refered to the word's method
        screen.blit(score_board,(10,550))



        Jet_sprites.draw(screen)
```

```
        bullets.draw(screen)

        asteroid_group.draw(screen)
        display.update()#update jet and screen view

        event.get()
        """moving the jet according to key pressed"""

        key = pygame.key.get_pressed()
        if key[K_LEFT] and jet1.rect.x>0:
            jet1.moveleft()

        if key[K_RIGHT] and jet1.rect.x<=700:
            jet1.moveright()

        if key[K_DOWN] and jet1.rect.y<=500:
            jet1.movedown()

        if key[K_UP] and jet1.rect.y>0:
            jet1.moveup()

        if key[K_SPACE] and len(bullets) <= jet1.firerates+(scores/4000):
            bullet = Bullet(screen, jet1.rect.x+50, jet1.rect.y+42)
            bullets.add(bullet)
            pygame.mixer.music.load("LaserBlast.wav")
            pygame.mixer.music.play()

        if key[K_ESCAPE]:
            menu.menu_screen(Button,run_game)

        if key[K_p]:
            menu.pause_menu(Button,run_game)


        """generate asteroid randomly"""
        if pygame.time.get_ticks() - asteroid_timer >= 200:
            asteroid = Asteroid(screen, 50, 50, random.randint(1,4)*6, 800,
(random.randint(1,28) * 20))
            asteroid_group.add(asteroid)
            asteroid_timer = pygame.time.get_ticks()

        """update the movement of asteroid"""
        for asteroid in asteroid_group:
            asteroid.movement()
            if asteroid.rect.right <= 0:
                asteroid_group.remove(asteroid) #remove after screen
            if
groupcollide(Jet_sprites,asteroid_group,dokilla=True,dokillb=True):#collition check
                menu.lose_menu(Button,run_game,scores)

        """update bullet movement on screen"""
        for bullet in bullets:
            bullet.movement()
            if bullet.rect.left > 800:
                bullets.remove(bullet)
            if groupcollide(bullets,asteroid_group,dokilla=True,dokillb=True):
                scores += 100

menu.menu_screen(Button,run_game)

#-----------------SPECIAL THANKS to
Pier,Excel,georgius,William,Nicander,Nicolas,Andy,Guntur--------------------------
"""Acknowledgement:
LaserBlast.wav(shooting sound) http://soundbible.com/472-Laser-Blasts.html
"""
```