BINUS UNIVERSITY
BINUS INTERNATIONAL

BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:	**Surname**	**Given Names**	**Student ID Number**
1.	**Kho**	**Ryan**	**2101704672**

Course Code	**: COMP6502**	**Course Name**	**: Introduction to Programming**
Class	**: L1AC**	**Name of Lecturer(s)**	: 1. Bagus Kerthyayana 2.Tri Asih Budiono
Major	**: CS**		
Title of Assignment (if any)	: Unfair Maze (Game)		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 6-11-2017**	**Submission Date**	**: 6-11-2017**

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.

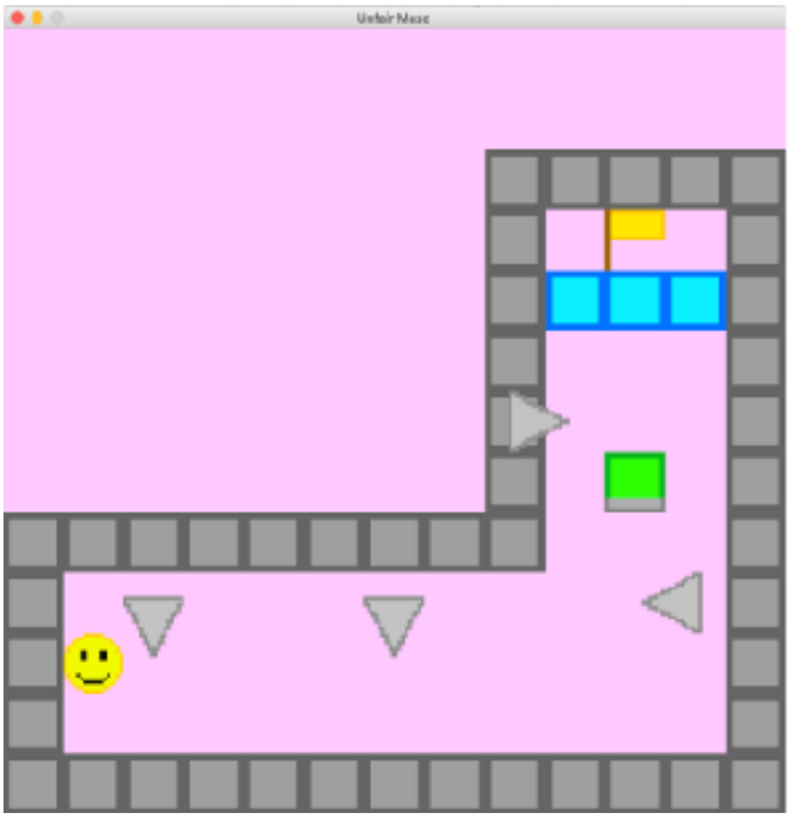
Signature of Student:

(Name of Student)
Ryan Anggada Kho

“Unfair Maze”
Name: Ryan Anggada Kho
ID: 2701704672

I. Description

The objective of this game is to move the character (which is a smiley face) to the flag safely without getting harmed by traps and hazards. The player must finish 5 levels of this game to complete. If the character touches the flag (which is the finish line), the player gets to play the next level otherwise, the character dies in one hit and must restart the level until the end.



II. Explanation of Each Function/Class

1. class Character(Sprite)
   def __init__(self, position)   #to initialize the class of the character and its position
   Sprite.__init__(self)   #inheritance sprite of PyGame
   self.images = [image.load(“hero.png”), image.load(“dead_hero.png”)]   #set an array filled with two different images representing the status of the character
   self.image = self.images[0] #initialise the sprite image of the character
   self.rect = self.image.get_rect()   #get the position of the image
   self.rect.left = position[0]
   self.rect.top = position[1]   #these place the sprite to a location by coordinates

```
def moveChar(self, action)
self.action = action
```

2. def Grouping_Sprites()

```
#groups all of the sprites so that they are easily assigned and drawn to the screen
for block in range(len(block_position[level])):
    a_block = Block(block_position[level][block], "normal")
    all_block.add(a_block)

for hazard_block in range(len(hazard_block_position[level])):
    a_hazard_block = Block(hazard_block_position[level][hazard_block], "hazard")
    all_hazard_block.add(a_hazard_block)

for door in range(len(door_position[level])):
    a_door = Door(door_position[level][door])
    all_door.add(a_door)

for troll_door in range(len(troll_door_position[level])):
    a_troll_door = Spike(troll_door_position[level][troll_door])
    all_troll_door.add(a_troll_door)

for spike in range(len(spike_position[level])):
    a_spike = Spike(spike_position[level][spike])
    all_spike.add(a_spike)

for button in range(len(button_position[level])):
    a_button = Button(button_position[level][button], "normal")
    all_button.add(a_button)

for troll_button in range(len(troll_button_position[level])):
    a_troll_button = Button(troll_button_position[level][troll_button], "troll")
    all_troll_button.add(a_troll_button)

for flag in range(len(flag_position[level])):
    a_flag = Flag(flag_position[level][flag])
    all_flag.add(a_flag)

for bullet_up in range(len(bullet_up_position[level])):
    a_bullet_up = Bullet(bullet_up_position[level][bullet_up], "up")
    all_bullet_up.add(a_bullet_up)

for bullet_down in range(len(bullet_down_position[level])):
    a_bullet_down = Bullet(bullet_down_position[level][bullet_down], "down")
    all_bullet_down.add(a_bullet_down)

for bullet_left in range(len(bullet_left_position[level])):
    a_bullet_left = Bullet(bullet_left_position[level][bullet_left], "left")
    all_bullet_left.add(a_bullet_left)

for bullet_right in range(len(bullet_right_position[level])):
    a_bullet_right = Bullet(bullet_right_position[level][bullet_right], "right")
    all_bullet_right.add(a_bullet_right)
```

III. Problem that Have Been Overcome

Making this program with PyGame is challenging. First of all, when I start to make sprites for the PyGame, calling the sprite with unique instructions are hard to implement, loading sounds are a chore and I always get a syntax and logic errors. This causes some of the problem to be resolved in few weeks, which causes serious delay. With the addition of my laziness and bad time management, the deadline of this project is coming close. I was losing hope, but I never give up. Luckily, with the help of one of my friend, Georgius, he was able to fix my program from errors and explained to me clearly why my program cannot run what I expect. During the long period of time, I gained programming knowledge and I have figured out that I can assign a sprite to reveal the one that is touched by a character, instead of revealing everything.

IV. Source Code

```
//importing the pygame package
from pygame import *
from pygame.sprite import *
from pygame.mixer import *



//initialise pygame
pygame.init()
//initiaise the mixer of the pygame to be able to load sound
mixer.init()
//initialises the frames per second
clock = pygame.time.Clock()

//setting variables with different sound files
game_background = Sound("background.wav")
intro_background = Sound("troll_background.wav")
death_background = Sound("death_marioworld.wav")
```

```
game_complete = Sound("game_clear.wav")
death = Sound("roblox_death.wav")
victory_sound = Sound("victory.wav")
troll_sound = Sound("troll_laugh.wav")

//sets the initial value of level and number of deaths
level = 0
death_count = 0

//set class for playable sprite character
class Character(Sprite):
    def __init__(self, position):
        Sprite.__init__(self)
            self.images = [image.load("hero.png"),
image.load("dead_hero.png")]
        //set the image
        self.image = self.images[0]
        //get the position of image
        self.rect = self.image.get_rect()
        //position the image by coordinates
        self.rect.left = position[0]
        self.rect.top = position[1]


   //function to move the character
   def moveChar(self, action):
        //initialise the action
        self.action = action

        //move the character by 6 pixels depending on its action
        if self.action == "up":
            self.rect.top -= 6
        elif self.action == "down":
            self.rect.top += 6
        elif self.action == "left":
            self.rect.left -= 6
        elif self.action == "right":
            self.rect.left += 6

        //function to swap the character image to death if the
character is dead
    def dead(self):
        self.image = self.images[1]
```

```
//set class for block and hazard block
class Block(Sprite):
    def __init__(self, position, status):
        Sprite.__init__(self)
        self.status = status
                    self.images  =  [image.load("block.png"),
image.load("hazard_block.png")]

        //choose what type of block
        if self.status == "normal":
            self.image = self.images[0]
        elif self.status == "hazard":
            self.image = self.images[1]

        self.rect = self.image.get_rect()
        self.rect.left = position[0]
        self.rect.top = position[1]

//set class for the door
class Door(Sprite):
    def __init__(self, position):
        Sprite.__init__(self)
        self.image = image.load("block_door.png")
        self.rect = self.image.get_rect()
        self.rect.left = position[0]
        self.rect.top = position[1]

//set class for the flag
class Flag(Sprite):
    def __init__(self, position):
        Sprite.__init__(self)
        self.image = image.load("flag.png")
        self.rect = self.image.get_rect()
        self.rect.left = position[0]
        self.rect.top = position[1]

//set class for the spike that is hidden to the player
class Spike(Sprite):
    def __init__(self, position):
        Sprite.__init__(self)
                   self.images  =  [image.load("spike_hidden.png"),
image.load("spike.png")]
```

```
        self.image = self.images[0]
        self.rect = self.image.get_rect()
        self.rect.left = position[0]
        self.rect.top = position[1]

    //reveals the spike
    def reveal(self):
        self.image = self.images[1]

//set class for the button
class Button(Sprite):
    def __init__(self, position, status):
        Sprite.__init__(self)
        self.status = status
                    self.images  =  [image.load("button.png"),
image.load("flag.png")]

        //the type of the button used
        if self.status == "normal":
            self.image = self.images[0]
        elif self.status == "troll":
            self.image = self.images[1]

        self.rect = self.image.get_rect()
        self.rect.left = position[0]
        self.rect.top = position[1]

//set class for the bullet
class Bullet(Sprite):
    def __init__(self, position, point):
        Sprite.__init__(self)
        //set the direction of the bullet pointing at
        self.point = point
                    self.images  =  [image.load("bullet_up.png"),
image.load("bullet_down.png"),  image.load("bullet_left.png"),
image.load("bullet_right.png")]

        //load image of bullet in proper pointed direction
        if self.point == "up":
            self.image = self.images[0]
        elif self.point == "down":
            self.image = self.images[1]
        elif self.point == "left":
            self.image = self.images[2]
        elif self.point == "right":
            self.image = self.images[3]

        self.rect = self.image.get_rect()
        self.rect.left = position[0]
        self.rect.top = position[1]

    //shoots the bullet in 12 pixels per frame
     def shoot(self, direction):
        //direction to shoot at
        self.direction = direction

        if self.direction == "up":
            self.rect.top -= 12
        elif self.direction == "down":
            self.rect.top += 12
        elif self.direction == "left":
            self.rect.left -= 12
        elif self.direction == "right":
            self.rect.left += 12

          //stops  the  bullet  from  going  beyond  certain  place
horizontally
    def limit_left(self, stop_left, back_left):
        self.stop_left = stop_left

        //place the bullet back if stopped
        if self.rect.left == self.stop_left:
            self.rect.left = back_left

    //stops it in vertical
    def limit_top(self, stop_top, back_top):
        self.stop_top = stop_top

        //place the bullet back if stopped
        if self.rect.top == stop_top:
            self.rect.top = back_top

//class for font sprite
class TextDisplay(Sprite):
    def __init__(self, text, font, size, position, color):
        Sprite.__init__(self)
        //font type and size
        self.font = pygame.font.Font(font, size)
        //renders it with chosen text and color
        self.image = self.font.render(text, True, color)
        //positions the text
        self.rect = self.image.get_rect()
        self.rect.left = position[0]
        self.rect.top = position[1]

//initialise group of sprites
all_block = Group()
all_hazard_block = Group()
all_door = Group()
all_troll_door = Group()
all_spike = Group()
all_button = Group()
all_troll_button = Group()
all_flag = Group()
all_bullet_up = Group()
all_bullet_down = Group()
all_bullet_left = Group()
all_bullet_right = Group()
grouped_char = Group()

//set the start position of character in 5 levels
char_position = [(60, 600), (360, 660), (60, 60), (660, 60), (660,
360)]

//set the position of the blocks in 5 levels except level 4
block_position = [[], [], [], [], []]
block_position[0] = [(0, 480), (0, 540), (0, 600), (0, 660), (0,
720), (60, 480), (60, 720), (120, 480), (120, 720), (180, 480),
(180, 720), (240, 480), (240, 720), (300, 480), (300, 720), (360,
480), (360, 720), (420, 480), (420, 720), (480, 120), (480, 180),
(480, 240), (480, 300), (480, 360), (480, 420), (480, 480), (480,
720), (540, 120), (540, 720), (600, 120), (600, 720), (660, 120),
(660, 720), (720, 120), (720, 180), (720, 240), (720, 300), (720,
360), (720, 420), (720, 480), (720, 540), (720, 600), (720, 660),
(720, 720)]
block_position[1] = [(180, 360), (180, 420), (180, 480), (180,
540), (180, 600), (180, 660), (180, 720), (240, 0), (240, 60),
(240, 120), (240, 180), (240, 240), (240, 300), (240, 360), (240,
720), (300, 0), (300, 720), (360, 0), (360, 720), (420 , 0), (420,
720), (480, 0), (480, 60), (480, 120), (480, 180), (480, 240),
(480, 300), (480, 360), (480, 720), (540, 360), (540, 420), (540,
480), (540, 540), (540, 600), (540, 660), (540, 720)]
block_position[2] = [(60, 420), (60, 480), (60, 540), (60, 600),
(60, 660), (120, 420), (120, 660), (180, 420), (180, 660), (240,
420), (240, 660), (300, 660)]
block_position[4] = [(0, 240), (0, 300), (0, 360), (0, 420), (0,
480), (0, 540), (0, 600), (0, 660), (60, 240), (120, 240), (540,
0), (540, 240), (600, 0), (600, 240), (600, 300), (600, 360),
(600, 420), (660, 0), (720, 0), (720, 60), (720, 120), (720, 180),
(720, 240), (720, 300), (720, 360), (720, 420)]

//set the hazard block position in 5 level
hazard_block_position = [[], [], [], [], []]
hazard_block_position[2] = [(0, 0), (0, 60), (0, 120), (0, 180),
(0, 240), (0, 300), (60, 0), (60, 300), (120, 0), (120, 60), (120,
120), (120, 300), (180, 0), (180, 300), (240, 0), (240, 180),
(240, 240), (240, 300), (240, 360), (300, 0), (360, 0), (360,
240), (360, 300), (360, 360), (360, 420), (360, 660), (420, 0),
(420, 240), (420, 420), (420, 660), (480, 0), (480, 60), (480,
120), (480, 180), (480, 240), (480, 420), (480, 660), (540, 420),
(540, 660), (600, 420), (600, 660), (660, 420), (660, 660), (720,
420), (720, 480), (720, 540), (720, 600), (720, 660)]
hazard_block_position[3] = [(0, 0), (0, 60), (0, 120), (0, 180),
(0, 240), (0, 300), (0, 360), (0, 420), (0, 480), (0, 540), (0,
600), (0, 660), (60, 0), (60, 420), (60, 660), (120, 0), (120,
420), (120, 660), (180, 0), (180, 240), (180, 420), (180, 660),
(240, 0), (240, 240), (240, 420), (240, 660), (300, 0), (300,
240), (300, 420), (300, 660), (360, 0), (360, 240), (360, 420),
(360, 660), (420, 0), (420, 240), (420, 420), (420, 660), (480,
0), (480, 240), (480, 420), (480, 660), (540, 0), (540, 240),
(540, 420), (540, 660), (600, 0), (600, 240), (600, 660), (660,
0), (660, 240), (660, 660), (720 ,0), (720 ,60), (720, 120), (720,
180), (720, 240), (720, 300), (720, 360), (720, 420), (720, 480),
(720, 540), (720, 600), (720, 660)]
hazard_block_position[4] = [(0, 720), (60, 720), (120, 720), (180,
240), (180, 420), (180, 720), (240, 240), (240, 420), (240, 720),
(300, 180), (300, 240), (300, 420), (300, 720), (360, 120), (360,
180), (360, 360), (360, 420), (360, 720), (420, 60), (420, 120),
(420, 300), (420, 420), (420, 720), (480, 0), (480, 60), (480,
240), (480, 420), (480, 720), (540, 420), (540, 720), (600, 420),
(600, 720), (660, 420), (660, 720), (720, 420), (720, 480), (720,
540), (720, 600), (720, 660), (720, 720)]
```

```
//set the door position only in level 1 and 2
door_position = [[], [], [], [], []]
door_position[0] = [(540, 240), (600, 240), (660, 240)]
door_position[1] = [(300, 120), (360, 120), (420, 120)]

troll_door_position = [[], [], [], [], []]
troll_door_position[2] = [(180, 480), (180, 540), (180, 600)]

//set the spike position in 5 levels
spike_position = [[], [], [], [], []]
spike_position[0] = [(60, 540), (60, 660), (300, 600), (480, 540),
(600, 480), (600, 660), (660, 660)]
spike_position[1]  =  [(240,  420),  (240,  660),  (300,  300),  (360,
180), (360, 600), (420, 300), (480, 420), (480, 660)]
spike_position[2]  =  [(360,  600),  (420,  600),  (480,  540),  (600,
480), (600, 540)]
spike_position[3] = [(60, 240), (120, 540), (120, 600), (180, 60),
(180, 180), (180, 360), (300, 360), (300, 480), (300, 540), (360,
60), (360, 120), (480, 300), (540, 120)]
spike_position[4] = [(60, 360), (60, 420), (180, 480), (180, 540),
(240, 300), (360, 540), (360, 600), (480, 540), (540, 120), (540,
540), (600, 480), (600, 660)]

//set the button position only in level 1 and 2
button_position = [[], [], [], [], []]
button_position[0] = [(600, 420)]
button_position[1] = [(360, 480)]

//set the fake button position only in level 3
troll_button_position = [[], [], [], [], []]
troll_button_position[2] = [(660, 600)]

//set the flag (finish line) position in 5 levels
flag_position = [[], [], [], [], []]
flag_position[0] = [(600, 180)]
flag_position[1] = [(360, 60)]
flag_position[2] = [(120, 540)]
flag_position[3] = [(60, 540)]
flag_position[4] = [(660, 540)]

//set the bullets position in 5 levels
bullet_up_position = [[], [], [], [], []]
```

```
bullet_up_position[4] = [(180, 780), (420, 780), (600, 780)]

bullet_down_position = [[], [], [], [], []]
bullet_down_position[0] = [(120, 480), (360, 480)]
bullet_down_position[4] = [(300, -60), (540, -60)]

bullet_left_position = [[], [], [], [], []]
bullet_left_position[0] = [(780, 540)]
bullet_left_position[1] = [(600, 240), (600, 600)]
bullet_left_position[2] = [(720, 60), (720, 120)]
bullet_left_position[4]  =  [(780,  180),  (780,  300),  (780,  480),
(780, 660)]

bullet_right_position = [[], [], [], [], []]
bullet_right_position[0] = [(420, 360)]
bullet_right_position[1] = [(120, 120), (120, 420), (120, 540)]
bullet_right_position[4] = [(-60, 540), (-60, 60)]



//group the character
character = Character(char_position[level])
grouped_char.add(character)

//groups all of each of the sprites
def Grouping_Sprites():
    for block in range(len(block_position[level])):
        a_block = Block(block_position[level][block], "normal")
        all_block.add(a_block)

    for hazard_block in range(len(hazard_block_position[level])):
               a_hazard_block  =  Block(hazard_block_position[level]
[hazard_block], "hazard")
        all_hazard_block.add(a_hazard_block)

    for door in range(len(door_position[level])):
        a_door = Door(door_position[level][door])
        all_door.add(a_door)

    for troll_door in range(len(troll_door_position[level])):
                  a_troll_door  =  Spike(troll_door_position[level]
[troll_door])
        all_troll_door.add(a_troll_door)
    for spike in range(len(spike_position[level])):
        a_spike = Spike(spike_position[level][spike])
        all_spike.add(a_spike)

    for button in range(len(button_position[level])):
                a_button = Button(button_position[level][button],
"normal")
        all_button.add(a_button)

    for troll_button in range(len(troll_button_position[level])):
              a_troll_button = Button(troll_button_position[level]
[troll_button], "troll")
        all_troll_button.add(a_troll_button)

    for flag in range(len(flag_position[level])):
        a_flag = Flag(flag_position[level][flag])
        all_flag.add(a_flag)

    for bullet_up in range(len(bullet_up_position[level])):
         a_bullet_up = Bullet(bullet_up_position[level][bullet_up],
"up")
        all_bullet_up.add(a_bullet_up)

    for bullet_down in range(len(bullet_down_position[level])):
               a_bullet_down = Bullet(bullet_down_position[level]
[bullet_down], "down")
        all_bullet_down.add(a_bullet_down)

    for bullet_left in range(len(bullet_left_position[level])):
               a_bullet_left = Bullet(bullet_left_position[level]
[bullet_left], "left")
        all_bullet_left.add(a_bullet_left)

    for bullet_right in range(len(bullet_right_position[level])):
              a_bullet_right = Bullet(bullet_right_position[level]
[bullet_right], "right")
        all_bullet_right.add(a_bullet_right)

//sets the caption, display size and color
display.set_caption("Unfair Maze")
screen = display.set_mode((780, 780))
screen.fill((255, 200, 255))
```

```
//action to move the character
move_left = False
move_right = False
move_up = False
move_down = False

//sets the limit coordinates of the bullet and back position
up_limit = [[], [], [], [], []]
up_back = [[], [], [], [], []]
down_limit = [[], [], [], [], []]
down_back = [[], [], [], [], []]
left_limit = [[], [], [], [], []]
left_back = [[], [], [], [], []]
right_limit = [[], [], [], [], []]
right_back = [[], [], [], [], []]

up_limit[0] = 840
up_back[0] = 840
down_limit[0] = 840
down_back[0] = 420
left_limit[0] = 420
left_back[0] = 780
right_limit[0] = 840
right_back[0] = 420
left_limit[1] = -60
left_back[1] = 600
right_limit[1] = 840
right_back[1] = 120
left_limit[2] = 120
left_back[2] = 720
up_limit[4] = 420
up_back[4] = 780
down_limit[4] = 420
down_back[4] = -60
left_limit[4] = -60
left_back[4] = 780
right_limit[4] = 840
right_back[4] = -60

//booleans for the loop
intro = False
alive = True
```

```
victory = False
win = False
exit_flag = False
death_loop = True
death_loop2 = False

//declare it to integer for the time
death_time = 0
victory_time = 0
complete_time = 0

//assign variables with texts
all_text = Group()
victory_text = TextDisplay("You win!", "SuperMario256.ttf", 80 ,
(200, 150), (255, 255, 0))
death_text = TextDisplay("You died!", "SuperMario256.ttf", 80,
(200, 150), (255, 0, 0))
title = TextDisplay("Unfair Maze", "GibsonBold.ttf", 90, (140,
100), (255, 165, 0))
start = TextDisplay("Start the game", "MyriadProLight.ttf", 45,
(260, 300), (50, 205, 50))
retry = TextDisplay("Retry?", "MyriadProLight.ttf", 45, (350,
330), (50, 205, 50))
next_level = TextDisplay("Go to the next level!",
"MyriadProLight.ttf", 45, (200, 300), (50, 205, 50))
complete = TextDisplay("Congratulations!", "MyriadProLight.ttf",
45, (170, 300), (50, 205, 50))

//calls the function for grouping sprites
Grouping_Sprites()

//adds the text for the main menu
all_text.add(start)
all_text.add(title)

//main menu
intro = True
intro_background.play()
while intro:
    for e in event.get():
        if e.type == QUIT:
            pygame.quit()
            intro = False
```

```
                exit_flag = True

            if e.type == MOUSEBUTTONDOWN:
                if start.rect.collidepoint(mouse.get_pos()):
                    //starts the game if start text is clicked
                    all_text = Group()
                    intro = False
                    intro_background.stop()
                    game_background.play()

        //display text to the screen
        all_text.draw(screen)
        display.update()

    while exit_flag == False:
        screen.fill((255, 200, 255))

        //display all the sprites to the screen
        all_text.draw(screen)
        grouped_char.draw(screen)
        all_block.draw(screen)
        all_hazard_block.draw(screen)
        all_door.draw(screen)
        all_troll_door.draw(screen)
        all_spike.draw(screen)
        all_bullet_up.draw(screen)
        all_bullet_down.draw(screen)
        all_bullet_left.draw(screen)
        all_bullet_right.draw(screen)
        all_button.draw(screen)
        all_troll_button.draw(screen)
        all_flag.draw(screen)
        display.update()

        for ev in event.get():
            if ev.type == QUIT:
                pygame.quit()
                exit_flag = True

            //character will move if keys are hold or tapped
            if ev.type == KEYDOWN:
                if ev.key == K_UP or ev.key == K_w:
                    move_up = True
                elif ev.key == K_DOWN or ev.key == K_s:
                    move_down = True

                elif ev.key == K_LEFT or ev.key == K_a:
                    move_left = True

                elif ev.key == K_RIGHT or ev.key == K_d:
                    move_right = True

            //character will stop moving if keys are released
            if ev.type == KEYUP:
                if ev.key == K_UP or ev.key == K_w:
                    move_up = False

                elif ev.key == K_DOWN or ev.key == K_s:
                    move_down = False

                elif ev.key == K_LEFT or ev.key == K_a:
                    move_left = False

                elif ev.key == K_RIGHT or ev.key == K_d:
                    move_right = False

            if ev.type == MOUSEBUTTONDOWN:
                //retries the level if retry text is clicked
                if retry.rect.collidepoint(mouse.get_pos()):
                    alive = True
                    death_loop = True
                    game_background.play()
                    all_text = Group()
                    Grouping_Sprites()
                    character = Character(char_position[level])
                    grouped_char.add(character)

                    //move on to the next level if next level text is
    clicked
                elif next_level.rect.collidepoint(mouse.get_pos()):
                    all_text = Group()
                    Grouping_Sprites()
                    game_background.play()
                    character = Character(char_position[level])
                    grouped_char.add(character)
```

```
    //character can only move if he is alive
    if move_up == True and alive == True:
        character.moveChar("up")
                    if spritecollideany(character, all_block) or
spritecollideany(character, all_door):
            character.moveChar("down")

    if move_down == True and alive == True:
        character.moveChar("down")
                    if spritecollideany(character, all_block) or
spritecollideany(character, all_door):
            character.moveChar("up")

    if move_left == True and alive == True:
        character.moveChar("left")



       //so that character cannot penetrate through the block or
door
     if spritecollideany(character, all_block) or
spritecollideany(character, all_door):
            character.moveChar("right")

    if move_right == True and alive == True:
        character.moveChar("right")
                    if spritecollideany(character, all_block) or
spritecollideany(character, all_door):
            character.moveChar("left")

    //character will die if spike that is hidden collided
    if spritecollideany(character, all_spike):
        character.dead()
        alive = False

         //character will die if fake door (spike and door
characteristic) is collided
    if spritecollideany(character, all_troll_door):
        character.dead()
        alive = False

    //removes the door if button is pressed
    for button_pressed in all_button:
        if button_pressed.rect.colliderect(character):
            all_door = Group()
            all_button = Group()

       //removes the fake door if button (disguised as flag) is
collided
    for troll_button_pressed in all_troll_button:
        if troll_button_pressed.rect.colliderect(character):
            all_troll_door = Group()
            all_troll_button = Group()
            troll_sound.play()

    //reveals a collided spike if the character collides it
    for spike_invisible in all_spike:
        if spike_invisible.rect.colliderect(character):
            spike_invisible.reveal()


    for troll_door_invisible in all_troll_door:
        if troll_door_invisible.rect.colliderect(character):
            troll_door_invisible.reveal()

    //character dies if it collides the bullet
    if spritecollideany(character, all_bullet_up):
        character.dead()
        alive = False

    if spritecollideany(character, all_bullet_down):
        character.dead()
        alive = False

    if spritecollideany(character, all_bullet_left):
        character.dead()
        alive = False

    if spritecollideany(character, all_bullet_right):
        character.dead()
        alive = False

    if spritecollideany(character, all_hazard_block):
        character.dead()
        alive = False
    //shoots all of the bullets in given direction and limit
    for shot_bullet_up in all_bullet_up:
        shot_bullet_up.shoot("up")
        shot_bullet_up.limit_top(up_limit[level], up_back[level])

    for shot_bullet_down in all_bullet_down:
        shot_bullet_down.shoot("down")
                    shot_bullet_down.limit_top(down_limit[level],
down_back[level])

    for shot_bullet_left in all_bullet_left:
        shot_bullet_left.shoot("left")
                    shot_bullet_left.limit_left(left_limit[level],
left_back[level])

    for shot_bullet_right in all_bullet_right:
        shot_bullet_right.shoot("right")
                   shot_bullet_right.limit_left(right_limit[level],
right_back[level])

    //player wins the game
    if spritecollideany(character, all_flag) and level < 4:
        all_flag = Group()
        game_background.stop()
        grouped_char = Group()
        victory_sound.play(0)
        victory_time = pygame.time.get_ticks()
        victory = True
       death_history = TextDisplay("You have died {}
times!".format(death_count), "MyriadProLight.ttf", 50, (200, 500),
(255, 0, 255))

    //player completes the game
    if spritecollideany(character, all_flag) and level == 4:
        all_flag = Group()
        game_background.stop()
        grouped_char = Group()
        game_complete.play(0)
        complete_time = pygame.time.get_ticks()
              death_final = TextDisplay("You died a total of {}
times!".format(death_count), "MyriadProLight.ttf", 50, (200, 500),
(255, 0, 255))
```

```
        win = True


      //display victory
     if pygame.time.get_ticks() - victory_time >= 2800 and level <
4:
        while victory:
            all_block = Group()
            all_hazard_block = Group()
            all_spike = Group()
            all_bullet_up = Group()
            all_bullet_down = Group()
            all_bullet_left = Group()
            all_bullet_right = Group()
            all_text.add(victory_text)
            all_text.add(next_level)
            all_text.add(death_history)
            level += 1
            victory = False

    //character dies
    if alive == False and death_loop == True:
        death_loop = False
        death_count += 1
                  death_counter  =  TextDisplay("You  have  died  {}
times!".format(death_count), "MyriadProLight.ttf", 50, (170 ,500),
(255, 0, 255))
        game_background.stop()
        death.play(0)
        death_background.play(0)
        death_time = pygame.time.get_ticks()
        death_loop2 = True

    //death screen showed up
    if pygame.time.get_ticks() - death_time >= 2850:
        while death_loop2:
            all_text.add(death_text)
            all_text.add(retry)
            all_text.add(death_counter)

            all_block = Group()
            all_hazard_block = Group()
            all_spike = Group()
            all_bullet_up = Group()
            all_bullet_down = Group()
            all_bullet_left = Group()
            all_bullet_right = Group()
            all_door = Group()
            all_troll_door = Group()
            all_button = Group()
            all_troll_button = Group()
            all_flag = Group()
            grouped_char = Group()
            display.update()
            death_loop2 = False


          //player  completes  the  game  and  gets  a  celebration  and
reward
      if level == 4 and pygame.time.get_ticks() - complete_time >=
5400:
        while win:
            all_block = Group()
            all_hazard_block = Group()
            all_spike = Group()
            all_bullet_up = Group()
            all_bullet_down = Group()
            all_bullet_left = Group()
            all_bullet_right = Group()
            all_text.add(victory_text)
            all_text.add(complete)
            all_text.add(death_final)
            win = False


       //sets the program’s frames 60 per seconds (that means the
bullet is moving 720 pixels per seconds)
     clock.tick(60)
```