BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:		**Surname**	**Given Names**	**Student ID Number**
	1.	**Azzahra**	**Rainamira**	**2301900391**

Course Code	**: COMP6056**	**Course Name**	**: Introduction to Programming**
Class	**: L1CC**	**Name of Lecturer(s)**	**: Ida Bagus Kerthyayana**
Major	**: CS**		
Title of Assignment (if any)	: Maze Game		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 17-01-20**	**Submission Date**	**: 14-01-20**

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.

Signature of Student: (Name of Student)
1. Rainamira Azzahra

"Maze Game"

Name : Rainamira Azzahra

ID : 2301900391

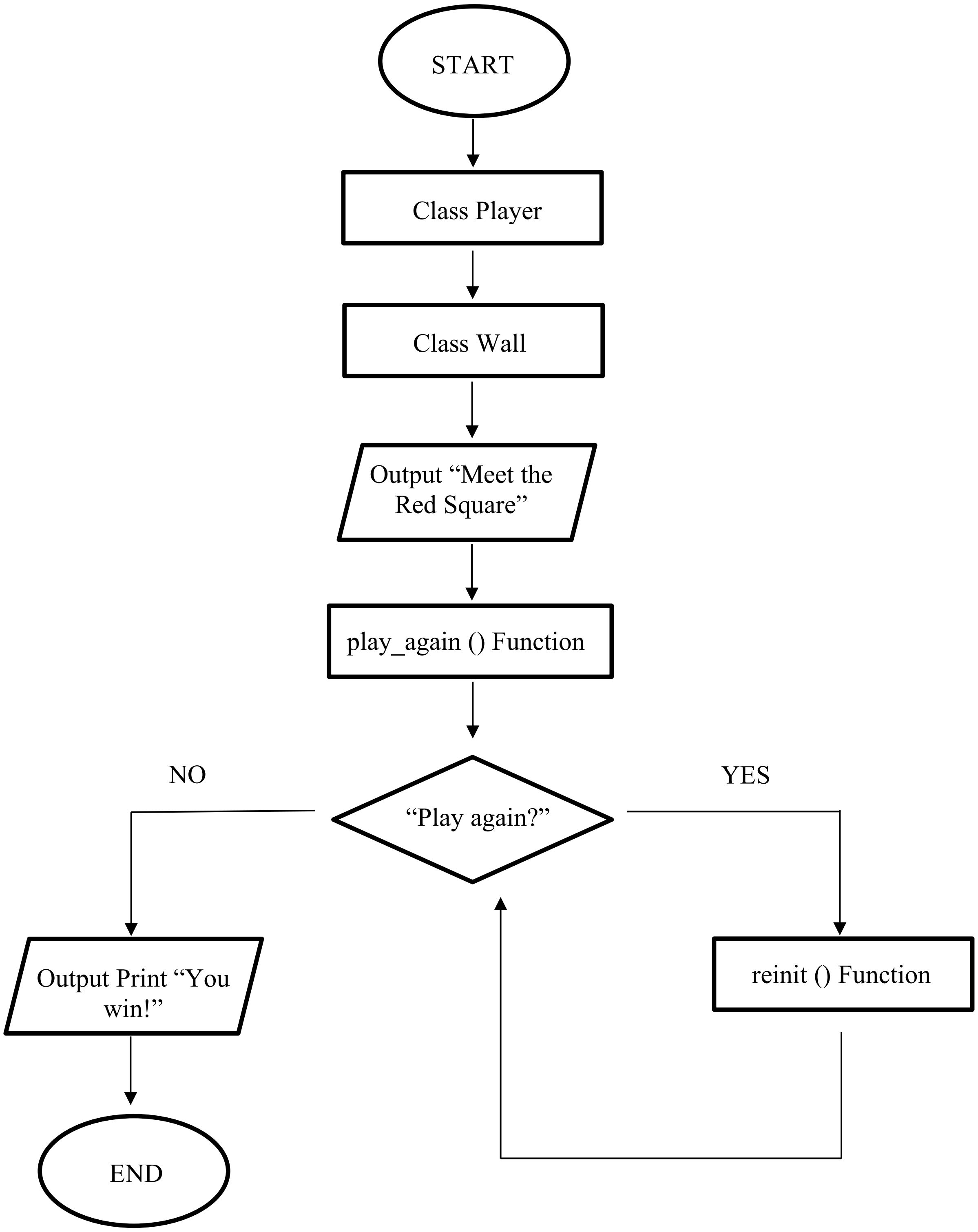
I. Description

The function of this program:

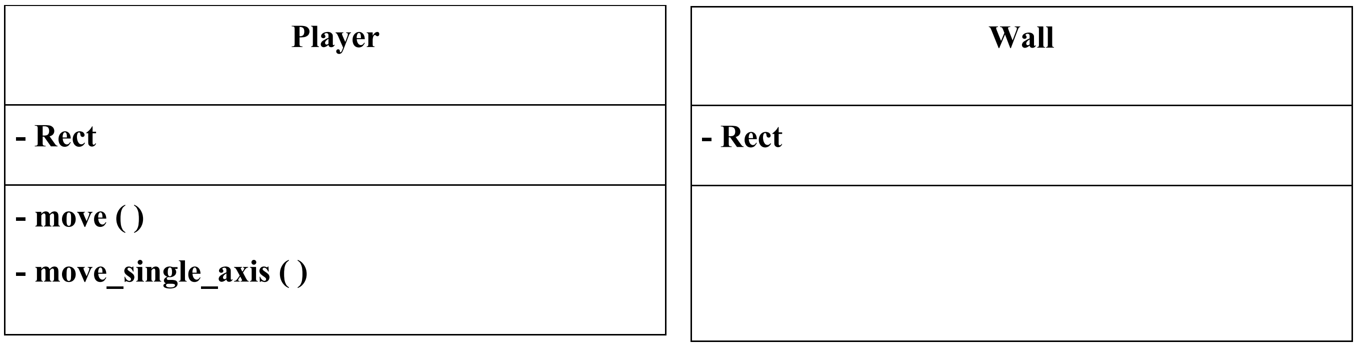
The purpose of this program is for the player to make their way from the start point to the end point (red rectangle). The layout of the maze is such that the player must follow a series of narrow winding corridors with many false turns and dead-ends until they eventually reach the finish point.

II. Solution Design

Project's Hierarchy Chart



Class Diagram



III. Explanation of Each Function

a. Inside the Class

1) Player Class

♦ **_init_ (self):**

- Creates Player rectangle.
- Assign the rectangle position, size, width, and height.

♦ **move (self, dx, dy) :**

- Moves each axis separately. First move along the X axis, test for a collision, move out, then move along Y axis, test for a collision, and move out. So that, this function checks for collisions both times.

♦ **move_single_axis (self, dx, dy) :**

- Moves a rectangle by adding the axis to move out.
- Make sure player rectangle do not overlap with wall rectangle.
- Check if player rectangle border meet with wall rectangle border then move out based on velocity.
- The area covered by a rectangle does not include the right- and bottom-most edge of pixels. If player rectangle bottom border is another rectangle's top border, the two meet but do not overlap.

2) Wall Class

♦ **_init_ (self, pos):**

- Create Wall rectangle.
- Assign/formats the rect position, size, width, and height.
- Add each Wall rectangle to empty walls list that stored all the wall.

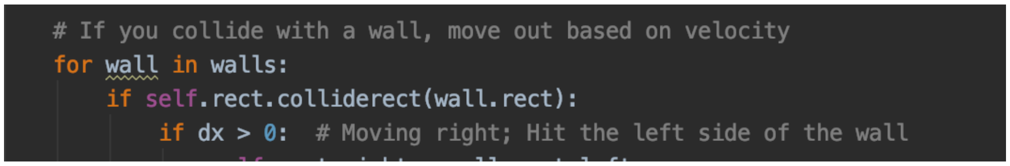
b. Outside the Class

- ♦ **message_to_screen (msg, color) :**
  - Create a new Surface with the specified text rendered on it then blit this image (Surface) of the text onto another Surface.
- ♦ **reinit ():**
  - Allows to change global variable (player, end_rect) from inside a function using global.
  - Holds the level layout in a list of string then parse the level string.
  - Checks every column in row if it read “W” or “E”.
  - If it read “W”, it will connect to Wall Class and give the position information for the wall rectangle then the x,y position increased by 16.
  - If it read “E”, it will create an End rectangle.
  - If it checks a new row, x = y = 0, It will keep going and stop in the last row.
- ♦ **play_again ():**
  - Create a display window 320 pixels wide by 240 pixels high called screen.
  - Create a new Surface with the specified text which is “Play again?” with the color rendered on it.
  - Formats the text axis and size.
  - pygame.draw.rect(). Draw rectangle as a Surface on the game window, formats the Surface color and position.
  - textscreen.blit(). Blit the Surface of the text onto another Surface.
  - pygame.display.flip() tells Pygame to make the most recently drawn screen visible. When we move the game elements around, pygame.display.flip() will continually update the display to show the new positions of elements and hide the old ones, creating the illusion of smooth movement.

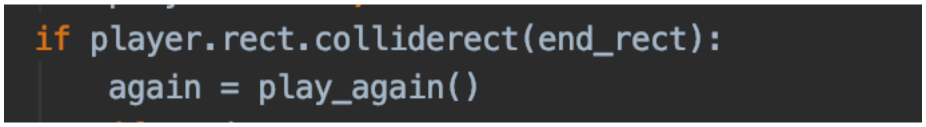
- Check if the key pressed (pygame.MOUSEBUTTONDOWN) by reading the event.key attribute. If the mouse button down was pressed once in the rectangle area, the game repeated.

IV. Lessons that Have Been Learned

1. *The use of " rect.colliderect() ":*



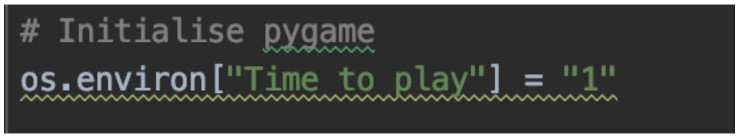
```
# If you collide with a wall, move out based on velocity
for wall in walls:
    if self.rect.colliderect(wall.rect):
        if dx > 0:  # Moving right; Hit the left side of the wall
```



```
if player.rect.colliderect(end_rect):
    again = play_again()
```

I found an easy way to check the collision for the player rect and wall rect, also the player rect and end rect.

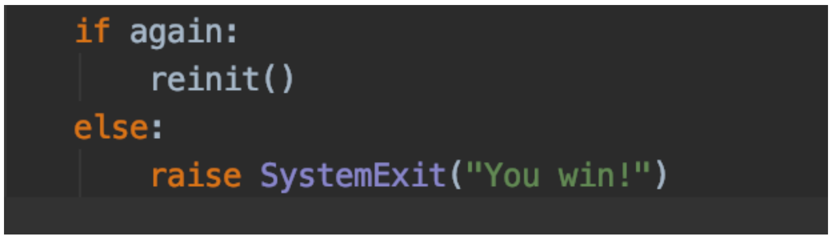
2. *The use of "os.environ ":*



```
# Initialise pygame
os.environ["Time to play"] = "1"
```

I did some research, and I found out about os.environ. It returns a dictionary having user's environmental variable as key and their values as value.

3. *Closing/Ending the program :*



```
if again:
    reinit()
else:
    raise SystemExit("You win!")
```

I found a fun way to print a message in the exit status after the program terminated.

Resources :

- https://www.daniweb.com/programming/threads/504827/maze-problem ("Play Again" text)

- https://www.pygame.org/project-Rect+Collision+Response-1061-.html
 (background code with the algorithms)

- https://stackoverflow.com (website I used when I was trying to fix the errors)

V. Source Code

MazeCode.py

```python
import os
import pygame

pygame.init()

# The dimension of the game
display_width = 320
display_height = 240

# Class for the player
class Player(object):
    def __init__(self):
        self.rect = pygame.Rect(32, 32, 16, 16)

    def move(self, dx, dy):
        # Move each axis separately. Note that this checks for collisions both times.
        if dx != 0:
            self.move_single_axis(dx, 0)
        if dy != 0:
            self.move_single_axis(0, dy)

    def move_single_axis(self, dx, dy):
        # Move the rect
        self.rect.x += dx
        self.rect.y += dy

        # If you collide with a wall, move out based on velocity
        for wall in walls:
            if self.rect.colliderect(wall.rect):
                if dx > 0:  # Moving right; Hit the left side of the wall
                    self.rect.right = wall.rect.left
                if dx < 0:  # Moving left; Hit the right side of the wall
                    self.rect.left = wall.rect.right
                if dy > 0:  # Moving down; Hit the top side of the wall
                    self.rect.bottom = wall.rect.top
                if dy < 0:  # Moving up; Hit the bottom side of the wall
                    self.rect.top = wall.rect.bottom


# Nice class to hold a wall rect
class Wall(object):
    def __init__(self, pos):
        walls.append(self)
        self.rect = pygame.Rect(pos[0], pos[1], 16, 16)
```

```python
font = pygame.font.SysFont(None, 50)
def message_to_screen(msg,color):
    screen_text = font.render(msg, True, color)
    screen.blit(screen_text, [680/2, display_height/2])

# Initialise pygame
os.environ["Time to play"] = "1"

# Set up the display
pygame.display.set_caption("Get to the red square!")
screen = pygame.display.set_mode((display_width, display_height))

clock = pygame.time.Clock()
walls = []  # List to hold the walls
player = None
end_rect = None


def reinit():
    global player, end_rect
    player = Player()  # Create the player
    # Holds the level layout in a list of strings.
    level = [
        "WWWWWWWWWWWWWWWWWWWW",
        "W                  W",
        "W         WWWWWWW  W",
        "W   WWWW        E  W",
        "W   W        WWWW  W",
        "W WWW  WWWW        W",
        "W   W     W W      W",
        "W   W     W   WWW WW",
        "W   WWW WWW   W W  W",
        "W     W   W   W W  W",
        "WWW   W   WWWWW W  W",
        "W W      WW        W",
        "W W   WWWW   WWW   W",
        "W     W        W   W",
        "WWWWWWWWWWWWWWWWWWWW",
    ]
    # Parse the level string above. W = wall, E = exit
    x = y = 0
    for row in level:
        for col in row:
            if col == "W":
                Wall((x, y))
            if col == "E":
                end_rect = pygame.Rect(x, y, 16, 16)
            x += 16
        y += 16
        x = 0
reinit()


bigfont = pygame.font.Font(None, 72)
smallfont = pygame.font.Font(None, 45)

def play_again():
    SCREEN_WIDTH = display_width
    SCREEN_HEIGHT = display_height
    textscreen = screen
    text = bigfont.render('Play again?', 13, (0, 0, 0))
    textx = SCREEN_WIDTH / 2 - text.get_width() / 2
    texty = SCREEN_HEIGHT / 2 - text.get_height() / 2
    textx_size = text.get_width()
    texty_size = text.get_height()
    pygame.draw.rect(textscreen, (255, 0, 0), ((textx - 5, texty - 5),
                                               (textx_size + 10, texty_size +
                                                10)))
    textscreen.blit(text, (SCREEN_WIDTH / 2 - text.get_width() / 2,
                           SCREEN_HEIGHT / 2 - text.get_height() / 2))
    #clock = pygame.time.Clock()
    pygame.display.flip()
    in_main_menu = True
    while in_main_menu:
        clock.tick(50)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                in_main_menu = False
                return False
            elif event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                x, y = event.pos
                if x >= textx - 5 and x <= textx + textx_size + 5:
                    if y >= texty - 5 and y <= texty + texty_size + 5:
                        in_main_menu = False
                        return True


running = True
while running:

    clock.tick(60)

    for e in pygame.event.get():
        if e.type == pygame.QUIT:
            running = False
        if e.type == pygame.KEYDOWN and e.key == pygame.K_ESCAPE:
            running = False

    # Move the player if an arrow key is pressed
    key = pygame.key.get_pressed()
    if key[pygame.K_LEFT]:
        player.move(-2, 0)
    if key[pygame.K_RIGHT]:
        player.move(2, 0)
    if key[pygame.K_UP]:
        player.move(0, -2)
    if key[pygame.K_DOWN]:
        player.move(0, 2)
    if player.rect.colliderect(end_rect):
        again = play_again()
        if again:
            reinit()
        else:
            raise SystemExit("You win!")

    # Draw the scene
    screen.fill((0, 0, 0))
    for wall in walls:
        pygame.draw.rect(screen, (255, 255, 255), wall.rect)
    pygame.draw.rect(screen, (255, 0, 0), end_rect)
    pygame.draw.rect(screen, (255, 200, 0), player.rect)
    pygame.display.flip()
```