BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter
(Individual Work)

Student Information:	Surname	Given Names	Student ID Number
1.	**Nigo**	**Patrick**	**2301907183**

Course Code	**: COMP6502**	**Course Name**	**: Introduction to Programming**
Class	**: L1AC**	**Name of Lecturer(s)**	**: Ida Bagus Kerthyayana Manuaba**
Major	**: CS**		
Title of Assignment (if any)	**: Tightrope Dodge**		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 14/01/2020**	**Submission Date**	**: 14/01/2020**

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 submis-sion.
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)
1. Patrick Alvin Nigo

"Tightrope Dodge"

Name : Patrick Alvin Nigo

ID : 2301907183

I. Description

The function of this program:

This program is basically just a short little game in which is used to alleviate one's boredom. I have seen that little simple games can be used to help with that, so I have made this program. The program I have made is called 'Tightrope Dodge' In which you are a person on a unicycle on a tightrope, trying to dodge the falling blocks that are coming your way. The program itself is very simple to understand. You control the player with your left and right arrow keys. With the more blocks that you dodge, the bigger they get and the faster they fall, making a difficulty level in game.

II.a. Design/Plan

The game is made up of three files. The main code file (aaaa game.py), a sound file, and an image file for the player.

To start the game, a few things are done first. A window is made, the player image, the title, buttons to start or quit the game, and the sound is loaded in. The sound file is waiting to play whenever the player loses. Then comes the Start Menu loop of the game, which will then direct the player to the main game loop, or if the player wishes to, can exit the game.

One class is used, the **Player** class. This class is used to define the little player model/image that will be used for the game's character. More detail about the use of each class is specified in the page below

In the start menu loop, a simple equation is used where it calculated the mouse position is so that whenever the mouse is over the start or exit buttons, it knows where it is and will then light up whenever the mouse cursor is in its boundaries. If the player clicks on the buttons, a corresponding function will happen. Be it to start or to exit the game.

In the actual game loop itself, an imported library called random decides, on random, where the blocks will fall on the screen. The player then must dodge the blocks as they fall and to be careful not to hit them as that will end the game. They must also not move too far off the window as that will also end the game. When the player dodges a block successfully, a counter on the top will count up. But when a player gets hit by a block, they will be met with a Game over screen and will have buttons asking the player if they would like to play again or to quit the game.

II.b. Explanation of Each Function Inside the Class

Class Player:

player(image,x,y) function :

Here the **x** and **y** positions of the player could be set and with the **image** property, it will call upon a corresponding image to load in the player's character.

III.a. Lessons that Have Been Learned

1. ***Learning pygame***

   After I was done making this project. I have seen that I have learned the basics of pygame. Be it to draw a shape, creating displays, calling functions to your keyboard, inserting audio and images onto your game and well, how to make a game with python

2. ***One use of the random library***

   The random library is used to make all the random calculations you could ever need to have in a game. In my own experience, I used it to dictate in what position the blocks would fall into.

III.b. Problem that Have Been Overcome

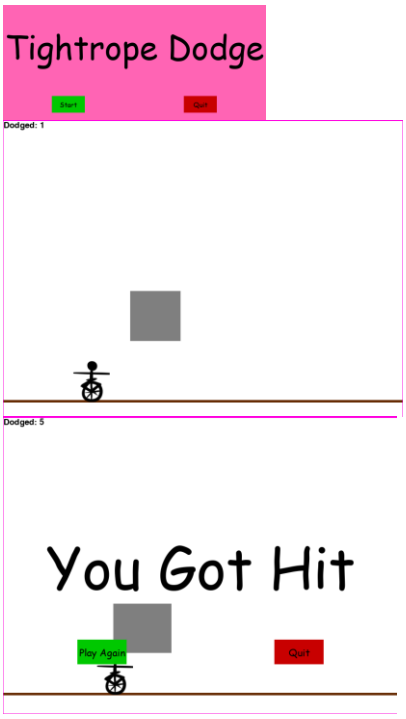
When I first started my project, I didn't really know what to make or do as I was very much still a beginner when it comes to coding or programming. I browsed through the internet to see what was possible for me at my current skill level. Thankfully I did find something that was simple enough for me to understand and that I thought I could recreate.

Whilst coding I did of course happen to run into some errors with my code and that proved to be quite frustrating. I eventually worked them out until my game was playable, and that was a good feeling.

Resources :

- https://python-forum.io/Thread-PyGame-PyGame-Colors (used to see what other colors I can use)
- https://stackoverflow.com (used for seeing what I could do to fix my code)
- https://pythonprogramming.net/ (used for tutorials to help with code and other inspirations on what I could do)

Screenshots :



Video : https://youtu.be/dzzlca0tv5A

V. Source Code

```
#code from pythonprogamming.net
#imports
import pygame
import time
import random



#initialize pygame
pygame.init()


#window size
display_width = 800
display_height = 600


#colors
black = (0,0,0)
white = (255,255,255)
red = (200,0,0)
green = (0,200,0)
bright_red = (255,0,0)
bright_green = (0,255,0)
brown = (100,40,0)
pink = (255,100,180)


#color for block to be dodged
block_color = (127,127,127)


#size of player in game 73 so that its not too big
player_width = 73


#game screen
gameDisplay = pygame.display.set_mode((display_width,display_height))
pygame.display.set_caption('Tightrope Dodge')
clock = pygame.time.Clock()


#image of player
playerImg = pygame.image.load('player.png')


#counter of objects dodged
```

```
def blocks_dodged(count):
    font = pygame.font.SysFont(None, 25) #size of font
    text = font.render("Dodged: "+str(count), True, black)
    gameDisplay.blit(text,(0,0))


#blocks to be dodged with its x y, width height and colour


def blocks(blockx, blocky, blockw, blockh, color):
    pygame.draw.rect(gameDisplay, color, [blockx, blocky, blockw, blockh])


#the player
class Player():

    def player(image,x,y):
        gameDisplay.blit(playerImg,(x,y))


#to put text
def text_objects(text, font):
    textSurface = font.render(text, True, black)
    return textSurface, textSurface.get_rect()


#code for quit button
def quitgame():
    pygame.quit()


#button function
def button(msg,x,y,w,h,ic,ac,action=None):
    mouse = pygame.mouse.get_pos()
    click = pygame.mouse.get_pressed()
    if x+w > mouse[0] > x and y+h > mouse[1] > y:
        pygame.draw.rect(gameDisplay, ac,(x,y,w,h))


        if click[0] == 1 and action != None:
            action()
    else:
        pygame.draw.rect(gameDisplay, ic,(x,y,w,h))


    smallText = pygame.font.SysFont("comicsansms",20) #size of font
    textSurf, textRect = text_objects(msg, smallText)
    textRect.center = ( (x+(w/2)), (y+(h/2)) )
```

```
    gameDisplay.blit(textSurf, textRect)


#sound
hit_sound = pygame.mixer.Sound("hit.wav")


#how to indicate that the player lost
def hit():

    #calling the sound when hit
    pygame.mixer.Sound.play(hit_sound)
    largeText = pygame.font.SysFont("comicsansms",115) #size of font
    TextSurf, TextRect = text_objects("You Got Hit", largeText)
    TextRect.center = ((display_width/2),(display_height/2))
    gameDisplay.blit(TextSurf, TextRect)



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


#game over screen buttons
        button("Play Again",150,450,100,50,green,bright_green,game_loop)
        button("Quit",550,450,100,50,red,bright_red,quitgame)


        pygame.display.update()
        clock.tick(15) #refresh rate


#start menu
def start_menu():


    intro = True


    while intro:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
```

```
        gameDisplay.fill(pink)
        #for the title
        largeText = pygame.font.SysFont("comicsansms",100)
        TextSurf, TextRect = text_objects("Tightrope Dodge", largeText)
        TextRect.center = ((display_width/2),(display_height/2))
        gameDisplay.blit(TextSurf, TextRect)
#start or quit buttons
        button("Start",150,450,100,50,green,bright_green,game_loop)
        button("Quit",550,450,100,50,red,bright_red,quitgame)



        pygame.display.update()
        clock.tick(15)



#loop of game so that it runs
def game_loop():
    x = (display_width * 0.45)
    y = (display_height * 0.8)

    x_change = 0

 #to indicate that the blocks would generate randomly
    block_startx = random.randrange(0, display_width)
    block_starty = -600 #so that blocks form off screen
    block_speed = 4 #starting speed
    block_width = 100
    block_height = 100

    blockCount = 1

    dodged = 0

    gameExit = False

    while not gameExit:

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

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_LEFT:
                    x_change = -5
                if event.key == pygame.K_RIGHT:
```

```
                    x_change = 5
            #to make the player move

            if event.type == pygame.KEYUP:
                if event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT:
                    x_change = 0

        x += x_change
        gameDisplay.fill(white)

        #to make the blocks that will fall
        blocks(block_startx, block_starty, block_width, block_height,
block_color)


        #adding a rope to the screen
        rope = pygame.Rect(0,558,800,5)
        pygame.draw.rect(gameDisplay,brown,rope)
        pygame.display.flip()



        block_starty += block_speed
        p = Player()
        p.player(x,y)
        blocks_dodged(dodged)

        #to calculate if the player goes out of bounds
        if x > display_width - player_width or x < 0:
            hit()

        if block_starty > display_height:
            block_starty = 0 - block_height
            block_startx = random.randrange(0,display_width)
            #to count up how many blocks dodged
            dodged += 1
            #make blocks fall faster
            block_speed += 1
            #make blocks bigger
            block_width += (dodged * 1.2)

#to calculate if a block hits a player
        if y < block_starty+block_height:

            if x > block_startx and x < block_startx + block_width or
x+player_width > block_startx and x + player_width < block_startx+block_width:
```

```
                hit()

        pygame.display.update()
        clock.tick(60)


#all the loops
start_menu()
game_loop()
pygame.quit()
quit()
```