BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:	Surname	Given Names	Student ID Number
1.	**Tandra**	**Vincentius**	**2301894804**

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)	: Shoot the Target		
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 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)

1. Vincentius Gabriel Tandra

"Shoot the Targt"

Name : Vincentius Gabriel Tandra

ID : 2301894804

I. Description

The function of this program:

This program is an incremental game also known as an idle game which I named Shoot the Target. The players of the game will have a target at the centre of the screen and money which will go up whenever it is clicked. On the left side of the screen is a store menu which can be used to buy items with the money obtained from clicking the target. These items each work passively and give money every second in the background and have a default price which goes up with every purchase. This game has no storyline or ending and so can be played whenever, even if you get bored of clicking, you can let the game's passive income make money for you and stay satisfied watching the numbers get bigger and bigger.

II.a. Design/Plan

The game is made up of a number of files, these are the main file (project.py) , the different classes, which used are in different files and accessed by importing them and their respective variables from each other and a text file that contains all of the data. Since, the game uses text and numbers which change constantly, this text file is needed so it works dynamicly as it does.

First a few things have to be done, the game window is made, a caption and the display icon and image is loaded, the audio files are also loaded.

3 classes are used, the **Target** class, **DisplayText** class and the **StoreMenu** class, each are used to define the target, money, the store menu as a whole and a separate class used to create text. Incremental games use the feature of passive income and so this passive income which is defined as **BPS**(bullets per second) is given a default value of zero and increases when an item from the store menu is bought, this income gives money every second and so a time counter is established that adds the BPS amount and then resets when the game reaches 1000 miliseconds or 1 second, this process then repeats for the duration that the game. More detail about the use of each class is specified in the page below

In the main game loop, a math equation is used to confine the clicking of the target to the boundaries of the image, so that players cannot click outside of the target to gain money, two functions, used to draw the target image and to continuously update the store's values are called.

In the event handler where all the magic happens, if the game is closed, it saves all of it's data by rewriting the current values into the text file. There is a button click event where money increases by 1 when clicked, an audio sound is also played. After that, more button click events are made where each one corresponds to to when a store menu item is purchased. When clicked, the menu item's BPS is added to the default BPS and a sound is played. Additionally, the price of that item goes up whenever it is bought and continues to do so with every consecutive purchase.

II.b. Explanation of Each Function Inside the Class

Project.py

DisplayText.py

StoreMenu,py

Class Target:

Init (self,x,y,targetValue) function :

Here the **x** and **y** positions of the clickable target are defined and the **targetValue** property of the target which is the amount of money you get per click when the target is clicked once. Additionally, a new variable known as **shotCount** is added and set to it's default value which is 0, it is the number that displays the amount of money on the screen for the entirety of the time the game is running—e.g., if a player clicks once, the **shotCount** increases to 1 and when a store item is bought, this **shotCount** value goes up by the amount per second one item is worth. Lastly, when an instance of this class is created, two functions which will be explained further below are called. These functions are named **shotCounter** and **draw** respectively

shotCount(self) function :

This function opens a text file containing all of the data used for the game, and reads all of it as a variable known as **text**, every value of data is written in separate lines and so each on is split by \n. After that, each piece of data which is written in a format containing a string, a colon not separated by a space and empty strings is replaced so only the integer value of each piece of data is left and then used in the program to be assigned to different variables.

draw(self) function :

Here the target object, background color is added and additional text is drawn into the code by blitting each one, the **shotCount** value is also formatted to separate by comma after reaching 4 digits, --e.g., 999 becomes 1,000 and not 1000 so that larger numbers are easier to read.

Class DisplayText:

Init(self,message,color,x,y) function:

This was a class I created to simplify the process of adding text within the game. The message of the text, it's color and x and y locations are defined, the font and it's size are set and then rendered and blitted in.

Class StoreMenu:

init(self) function:

Here, two variables are defined, **menuRectFill** and **menuRectOutline** a set of rectangular coordinates which will be used to create the inner rectangle of the menu and the outline of the menu.

Next, each item in the store menu is created, each item have identical properties, the amount owned, their price, the amount of money they give per second defined as their bps and their rectangle outline. The amount owned and their price is assigned to each property in the text file respectively.

Next the text and all values are displayed onto the screen, they each have a title, description, price and amount owned which are each given rectangular coordinates once again so it is easier to blit in each one later, the buy button is also given an outline for the click event latr in the code. They are also given their own font and fontsize.

update(self) function:

First the **menuRectFill** and **menuRectOutline** rectangles are drawn in and then each item's text, and rectangles are drawn in.

III.a. Lessons that Have Been Learned

1. ***Learning the ins and outs of pygame***

   After making this project, I have learned the basics of using the pygame library such as creating the display, creating shapes, inserting text, audio and making use of the events. It was fun to make because coding in general is not a very visual experience and learning how the logic and numbers translate into making a game was a very refreshing learning experience

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

   At first, i learned that pygame can store coordinates in rectangles using the pygame.Rect method. However, it cannot be used to store circular coordinates which the target used, will be a circle. After looking it up, I learned that there's a way to calculate the boundaries of a circular object using a mathematical formula. The mouse click event is then confined to the circular object's circumference so that the target can't be clicked anywhere else but on the target itself. I find this useful because it might come in handy in the future.

3. ***One use of file handling***

   I wasn't very aware of the use of file handling or in this case text files besides for the exercises that we do in class but without it, I wouldn't be able to make dynamic text for the game that also saves when the program is quit. By being able to apply this programming concept in making this game, I better learned how to use it and how I might be able to use it in the future.

III.b. Problem that Have Been Overcome

When I first started on this project, without very much refereence I was only able to create a circle which did something when clicked while trying to make another kind of game. After realizing I could make an incremental game instead, I found it difficult at first to conceptualize the code beyond that point. I needed to add a way to be able to display dynamic text, change the circle into an image to suit the game and more. The dynamic text was one problem I had no idea how to fix and so I took reference from a YouTube video making a similar type of game. I learnt that using text files allowed this and also to bee able to save the game data so it wouldn't restart whenever closed, I also was unable to decide on a window size for the game and liked the size useed in the video. The design aspect of the game was also somewhat difficult to come up with since I don't consider myself the most creative person in that department and so I got my brother to make the target image and took reference from other popular incremental games such as Cookie Clicker and the YouTube video for things like the store menu to make things more interesting.

Resources :

- https://www.youtube.com/watch?v=9Wk4gTHucYU
- https://stackoverflow.com (used for equations, fixing errors and other pygame issues)
- https://orteil.dashnet.org/cookieclicker/ (inspiration for game, design and price adjustment)
- https://www.dafont.com/press-start.font (for font used for ingame text)
- Timotius Ariel Tandra 2301894810 (target image, color choice)

V. Source Code

```
#imports
import pygame
import os.path
from os import path
import math
from Target import Target
from DisplayText import DisplayText
from StoreMenu import StoreMenu
from StoreMenu import target1
from Target import Target
from Target import red
from Target import lightBlue
from Target import targetImage
```

```
pygame.init()
win = pygame.display.set_mode((950, 600))
pygame.display.set_caption('Shoot the Target')
running = True
clock = pygame.time.Clock()
timecounter = 0
pygame.display.set_icon(targetImage)


#SFX
click_effect = pygame.mixer.Sound('pew.wav')
buy_effect = pygame.mixer.Sound('chaching.wav')

#instance of store menu
storemenu1 = StoreMenu()

#Definition of BPS
BPS = 0
BPS = int(target1.text[1])
commaBPS = '{:,}'.format(BPS)
BPSFont = pygame.font.Font('press_start.ttf',20)
BPSDisplay = BPSFont.render(commaBPS , True, lightBlue)
BPSRect = BPSDisplay.get_rect(center=(525,100))
win.blit(BPSDisplay,BPSRect)


# Main Game Loop
while running:
    x = pygame.mouse.get_pos()[0]
    y = pygame.mouse.get_pos()[1]

    sqx = (x - 525)**2
    sqy = (y -300)**2
    target1.draw()
    storemenu1.update()
    for event in pygame.event.get():

        if event.type == pygame.QUIT:
            newFile = open('btarget.txt','w')
            newFile.write('MONEY: ' + str(target1.shotCount) + '\n')
            newFile.write('BPS: ' + str(BPS) + '\n')
            newFile.write('SLINGS: ' + str(storemenu1.slingsOwned) + '\n')
            newFile.write('SLINGS_PRICE: ' + str(storemenu1.slingsPrice) + '\n')
            newFile.write('REVOLVERS: '  + str(storemenu1.revolversOwned) + '\n')
            newFile.write('REVOLVERS_PRICE: ' + str(storemenu1.revolv-
ersPrice) + '\n')
```

```
            newFile.write('RIFLES: '  + str(storemenu1.riflesOwned) + '\n')
            newFile.write('RIFLES_PRICE: ' + str(storemenu1.riflesPrice) + '\n')
            newFile.write('MINIGUNS: '  + str(storemenu1.minigunsOwned) + '\n')
            newFile.write('MINIGUNS_PRICE: ' + str(storemenu1.minigun-
sPrice) + '\n')
            newFile.close()
            running = False

        #counter goes up when target is clicked
        if event.type == pygame.MOUSEBUTTONUP:
            if math.sqrt(sqx + sqy) < 90:
                target1.shotCount += target1.targetValue
                click_effect.play()


        #when the buy sling button is clicked
        if event.type == pygame.MOUSEBUTTONUP and storemenu1.buySlingOutline.col-
lidepoint(pygame.mouse.get_pos()):
            if storemenu1.slingsPrice <= target1.shotCount:
                target1.shotCount -= storemenu1.slingsPrice
                storemenu1.slingsOwned += 1
                BPS += storemenu1.slingbps
                storemenu1.slingsPrice += int(storemenu1.sling-
sPrice // (4/100 * storemenu1.slingsPrice))
                buy_effect.play()

        #when the buy revolver button is clicked
        if event.type == pygame.MOUSEBUTTONUP and storemenu1.buyRevolverOut-
line.collidepoint(pygame.mouse.get_pos()):
            if storemenu1.revolversPrice <= target1.shotCount:
                target1.shotCount -= storemenu1.revolversPrice
                storemenu1.revolversOwned += 1
                BPS += storemenu1.revolverbps
                storemenu1.revolversPrice += int(storemenu1.revolv-
ersPrice // 6.5/100 * storemenu1.revolversPrice)
                buy_effect.play()

        #when the buy rifle button is clicked
        if event.type == pygame.MOUSEBUTTONUP and storemenu1.buyRifleOutline.col-
lidepoint(pygame.mouse.get_pos()):
            if storemenu1.riflesPrice <= target1.shotCount:
                target1.shotCount -= storemenu1.riflesPrice
                storemenu1.riflesOwned += 1
                BPS += storemenu1.riflebps
                storemenu1.riflesPrice += int(storemenu1.ri-
flesPrice // 90/100 * storemenu1.riflesPrice)
                buy_effect.play()
```

```
        #when the buy minigun button is clicked
        if event.type == pygame.MOUSEBUTTONUP and storemenu1.buyMinigunOut-
line.collidepoint(pygame.mouse.get_pos()):
            if storemenu1.minigunsPrice <= target1.shotCount:
                target1.shotCount -= storemenu1.minigunsPrice
                storemenu1.minigunsOwned += 1
                BPS += storemenu1.minigunbps
                storemenu1.minigunsPrice += int(storemenu1.minigun-
sPrice // 600/100 * storemenu1.minigunsPrice)
                buy_effect.play()

    #counter used to count bps
    timecounter += clock.tick()
    if timecounter >= 1000:
        target1.shotCount += BPS
        timecounter = 0


    pygame.display.update()
pygame.quit()
```