

**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

**Assignment Cover Letter**

**(Individual Work)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | |  | |  | |
| **Student Information**: **Surname** | | | | **Given Names**  **Adrian** | | **Student ID Number**  **2101720682** | |
| 1. | | **Suharto** |  |
|  |  |
| **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) | : **Maze Simulator(Game)** | |  |  |  | |  |
| **Type of Assignment**  **Submission Pattern** | **: Final Project** |  |  |  | |  | |
| **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 myown 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)

Guntur Sandjaya

**“Maze Simulator”**

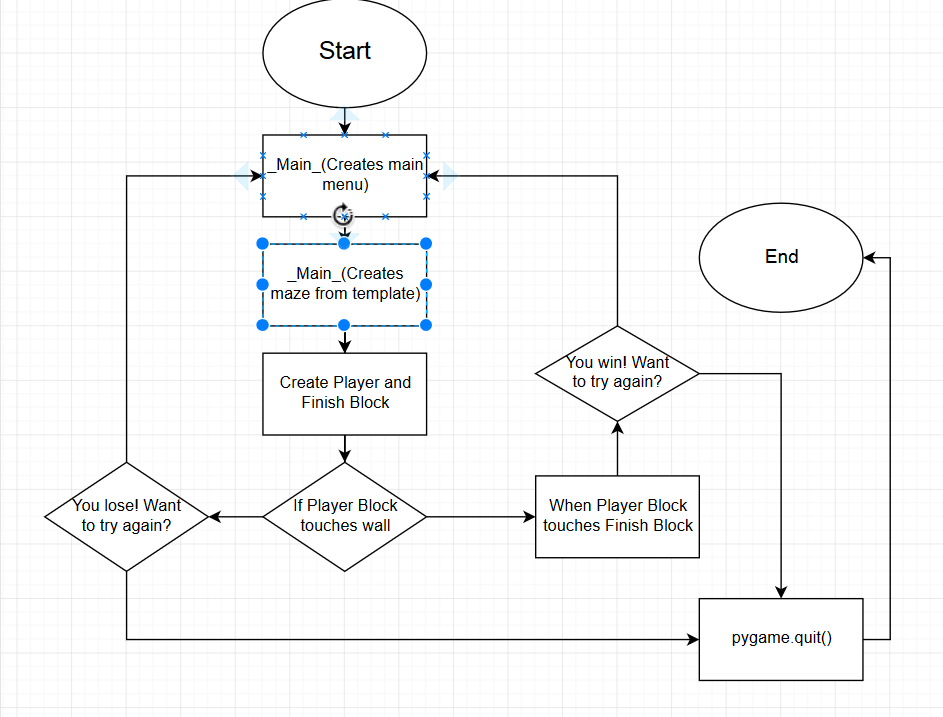
**Name: Adrian Pratama Suharto**

**ID : 2101720682**

1. Description

This program is a project that is made to help anyone to learn more about python, to have fun and start being interested in the world of programming. The reason this program was made is because it allows future programmers to think of several ways to make a solution or to create creative ideas that will help to make much more innovative programs, that hopefully can solve our problems that will arise in the future.

1. Flow Chart



1. Explanation of the Function

Class Box():

Class Box initiates the Box class that the player will use to play the program. The program itself will load an image called RedBox.png that will be used by the player, and it will make it into a hitbox by using the get\_rect() method. It will first be declared at the empty space at the top left of the maze, where it will be crushed to pieces if it hits the wall. The class also declares the position of the player block so it can move anywhere it pleases, as long it doesn’t touch the walls.

Class Maze():

Class Maze initiates the width and the height of the blocks that will be used to fill up the screen. It will split the 950 x 650 sized screen into a 19 x 13 space that will be filled by blocks with 50 x 50 size that are declared by the MazeWall class. The template that the class will use are declared by another module called template, where it uses def receive() that will declared a list of game templates that are available. Once declared the function game() will make an object out of a maze with the parameter that the def receive() has declared.

Class MazeWall():

Declares blocks and inputs the YellowBox.png that will be used by Maze function to create walls for the program. It also makes the hitbox for the maze by using get\_rect()

Class Finish():

Declares and load the image for the block that the player block must reach to win the game. The block appears either in three random locations, at the top right, bottom left, or bottom right of the maze.

Main function:

Holds the pygame.init() function that initializes pygame, display.set\_mode that initializes the windows, and declares Black and White RGB color.

def menu():

Creates caption for the window, also starts the first interface that will appear when you first start the program. The sentence will appear at the middle of the screen.

def tryagain()/lose():

The tryagain function are executed when the player block collides with the finish block, while the lose function are executed when the player block collides with the wall block. Both functions opens up a new window that declares whether you win or lose the game. If you lose the game, the player block dies horribly with a horrible scream.

game():

Loads the sound of the game, creates a loop that allows continual movement of the player block, randomize the template that will be chosen by the maze, and creates the object of the classes. Also creates the maze by calling the object that initialized the MazeClass. It also generates grouping of the sprites that comes from the objects. After all parts of the program are declared, assigned, and called, the program will update the display. If the player block touches either the finish block or the wall block, the program will go to lose/tryagain function.

template.py:

holds the def receive() that lists all available template for the maze program to use.

1. Source Code:

Maze.py:

**import** random  
**from** pygame **import** \*  
**from** pygame.sprite **import** \*  
**from** template **import** receive  
*#import modules  
#create Box Class for the Box player***class** Box(Sprite):  
 **def** \_\_init\_\_(self):*#initialize the sprite* Sprite.\_\_init\_\_(self)  
 self.image= pygame.image.load(**"RedBox.png"**)*#load the image* self.rect = self.image.get\_rect()*#creates hitbox* self.rect.left = self.rect.top = 60*#initial position* **def** moveRight(self):*#move the box to the right* self.rect.left += 2  
 **def** moveLeft(self):*#move the box to the left* self.rect.left -= 2  
 **def** moveUp(self):*#move the box forward* self.rect.top -= 2  
 **def** moveDown(self):*#move the box backward* self.rect.top += 2  
  
**class** Maze(Sprite):*#makes the maze* **def** \_\_init\_\_(self,grid):  
 *#initializes class as well as asking for input from the template.py* Sprite.\_\_init\_\_(self)  
 self.M = 19 *#amount of the blocks at the row* self.N = 13 *#amount of the blocks at the column* self.maze = grid *#accepts parameter* **def** create(self,surface,image): *#creates the wall* self.mazewall = Group()*#groups the wall* bx = 0*#x axis of the blocks* by = 0*#y axis of the blocks* **for** i **in** range(0,self.M\*self.N):*#ranges the amount of blocks need to be declared* **if** self.maze[bx + (by\*self.M)]== 1:*#* tempwall = MazeWall(bx\*50,by\*50)  
 self.mazewall.add(tempwall)*#adds wall for every row* bx = bx+1  
 **if** bx > self.M-1:*#resets the x axis blocks to 0* bx = 0  
 by = by+1*#goes to the next colum* **return** self.mazewall  
  
**class** Finish(Sprite):*#creates the finish class* **def** \_\_init\_\_(self):  
 Sprite.\_\_init\_\_(self)  
 self.image = pygame.image.load(**"FinishBox.png"**).convert()*#loads the class image* self.rect = self.image.get\_rect()*#creates hitbox* x = (850,550)  
 y = (850,50)  
 z = (50,550)  
 rand = [x,y,z]*#list of available position* (self.rect.left,self.rect.top) = rand[random.randint(0,2)]*#randomize the position***class** MazeWall(Sprite):*#class for the maze blocks* **def** \_\_init\_\_(self,x,y):  
 Sprite.\_\_init\_\_(self)  
 self.image= pygame.image.load(**"YellowBox.png"**).convert()*#load the image* self.rect = self.image.get\_rect()*#creates hitbox* self.rect.top = y  
 self.rect.left = x  
  
**def** text\_object(text, font):*#renders the font* textSurface = font.render(text, **True**, (BLACK))  
 **return** textSurface, textSurface.get\_rect()  
  
*#main function*pygame.init()*#initialize everything*display\_width = 950  
display\_height = 650  
display = pygame.display.set\_mode((display\_width,display\_height),HWSURFACE,0)*#initialize the window*BLACK = (0,0,0)*#values for RGB*WHITE = (255,255,255)  
  
**def** menu():*#function for menu* pygame.display.set\_caption(**"Welcome to a-MAZE-ing World"**)*#caption for the window* apple = **True  
 while** apple:  
 largeText = pygame.font.Font(**None**, 80)*#declares the font template* textSurf, textRect = text\_object(**"PRESS SPACE TO START"**, largeText)*#asks for input* textRect.center = ((display\_width/2), (display\_height/2))  
 display.fill((WHITE))*#refill the background with white* display.blit(textSurf, textRect)*#blits the window* **for** events **in** pygame.event.get():  
 keys = key.get\_pressed()*#gets the keys to check for input* **if** events.type == pygame.QUIT:  
 pygame.quit()  
 **if** keys[K\_SPACE]:  
 apple = **False** pygame.display.flip()  
  
**def** tryagain():*#function for trying the game again* apple = **True  
 while** apple:  
 largeText = pygame.font.Font(**None**, 60)  
 textSurf, textRect = text\_object(**"You win! Do you want to Continue?"**, largeText)  
 textRect.center = ((display\_width/2), (display\_height/2))  
 display.fill((WHITE))  
 display.blit(textSurf, textRect)  
 **for** events **in** pygame.event.get():  
 keys = key.get\_pressed()  
 **if** keys[K\_q]:*#pressing q quits the game* pygame,quit()  
 quit()  
 **if** keys[K\_c]:*#pressing c starts another game* game()  
 **if** events.type == pygame.QUIT:*#pressing quit leaves the game* pygame.quit()  
 pygame.display.flip()  
  
**def** lose():*#function for trying the game* apple = **True** pygame.mixer.music.load(**"glass.wav"**)*#loads the sound of losing* pygame.mixer.music.play()*#plays it* **while** apple:  
 largeText = pygame.font.Font(**None**, 60)  
 textSurf, textRect = text\_object(**"You lose! Do you want to Continue?"**, largeText)  
 textRect.center = ((display\_width/2), (display\_height/2))  
 display.fill((WHITE))  
 display.blit(textSurf, textRect)  
 **for** events **in** pygame.event.get():  
 keys = key.get\_pressed()  
 **if** keys[K\_q]:  
 pygame,quit()  
 quit()  
 **if** keys[K\_c]:  
 game()  
 **if** events.type == pygame.QUIT:  
 pygame.quit()  
 pygame.display.flip()  
  
**def** game():*#starting the game function* pygame.mixer.music.load(**"Solution.wav"**)*#loads the music for the game* pygame.mixer.music.play(-1)*#loops the game sound* pygame.mixer.music.set\_volume(1)*#sets the volume* running = **True** x = random.randint(0,4)*#randomized number between 0-4* y = receive()*#receive the list* z = y[x]*#getting value of the randomized number and use it to get the list's value* player = Box()*#initialize box class* maze = Maze(z)*#initialize maze class using the template* finish = Finish()*#initialize finish class* mazewallgroup = maze.create(display,image)*#create maze* sprites = Group(player)*#grouping the sprite* sprite = Group(finish)*#grouping the sprite* **while** running:  
 keys = pygame.key.get\_pressed()  
 **if** keys[K\_RIGHT]:  
 player.moveRight()  
 **if** spritecollideany(player,mazewallgroup):  
 lose()  
 **if** keys[K\_LEFT]:  
 player.moveLeft()  
 **if** spritecollideany(player,mazewallgroup):  
 lose()  
 **if** keys[K\_UP]:  
 player.moveUp()  
 **if** spritecollideany(player,mazewallgroup):  
 lose()  
 **if** keys[K\_DOWN]:  
 player.moveDown()  
 **if** spritecollideany(player,mazewallgroup):  
 lose()  
 **if** keys[K\_ESCAPE]:  
 running = **False  
 if** spritecollideany(player,sprite):  
 tryagain()  
 **for** event **in** pygame.event.get():  
 **if** event == pygame.QUIT:  
 pygame.quit()  
 pygame.event.pump()*#get event* display.fill(BLACK)  
 sprites.draw(display)*#display everything* sprite.draw(display)  
 mazewallgroup.draw(display)  
 pygame.display.flip()  
  
menu()  
game()

template.py:

**def** receive():  
 a= [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,  
 1,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,1,  
 1,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1,1,0,1,  
 1,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,0,1,  
 1,0,1,0,0,0,1,0,1,0,1,1,1,0,1,0,1,0,1,  
 1,0,1,0,1,0,1,0,1,0,0,0,0,0,1,0,1,0,1,  
 1,0,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,0,1,  
 1,0,0,0,0,0,1,0,1,0,1,0,0,0,1,0,0,0,1,  
 1,0,1,0,1,1,1,1,1,0,1,0,1,0,1,1,1,0,1,  
 1,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,1,  
 1,0,1,1,0,1,1,0,1,1,0,1,1,0,1,0,1,0,1,  
 1,0,1,0,0,0,1,0,0,0,0,0,1,0,1,0,1,0,1,  
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]  
  
 b =[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,  
 1,0,0,0,0,0,1,0,0,0,1,0,1,0,0,0,0,0,1,  
 1,1,1,0,1,1,1,1,1,0,1,0,1,1,1,0,1,1,1,  
 1,0,0,0,0,0,1,0,0,0,1,0,1,0,1,0,1,0,1,  
 1,1,1,0,1,0,0,0,1,1,1,0,1,0,1,0,1,0,1,  
 1,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,1,0,1,  
 1,0,1,0,1,0,1,0,1,0,1,0,1,1,1,1,1,0,1,  
 1,0,1,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,1,  
 1,1,1,1,1,0,1,0,1,0,1,0,1,1,0,1,1,1,1,  
 1,0,0,0,0,0,1,0,1,0,1,0,1,0,0,0,0,0,1,  
 1,1,1,0,1,1,1,1,1,0,1,1,1,1,1,1,1,0,1,  
 1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,1,  
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]  
  
 c =[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,  
 1,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,1,  
 1,0,1,0,1,0,1,0,1,1,1,1,1,0,1,0,1,0,1,  
 1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,1,0,1,  
 1,1,1,1,1,1,1,0,1,1,1,0,1,1,1,1,1,0,1,  
 1,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,1,  
 1,1,1,0,1,1,1,1,1,0,1,1,1,0,1,0,1,0,1,  
 1,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,1,0,1,  
 1,1,1,0,1,1,1,0,1,1,1,1,1,1,1,1,1,0,1,  
 1,0,0,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,1,  
 1,1,1,1,1,0,1,0,1,1,1,0,1,0,1,1,1,0,1,  
 1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0,1,  
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]  
  
 d =[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,  
 1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,1,  
 1,0,1,1,1,0,1,1,1,1,1,0,1,1,1,0,1,1,1,  
 1,0,1,0,1,0,1,0,0,0,1,0,0,0,0,0,0,0,1,  
 1,0,1,0,1,0,1,0,1,1,1,0,1,1,1,1,1,0,1,  
 1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0,1,  
 1,1,1,0,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,  
 1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,1,  
 1,1,1,0,1,1,1,0,1,1,1,0,1,1,1,1,1,0,1,  
 1,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,  
 1,1,1,1,1,1,1,0,1,0,1,0,1,1,1,0,1,1,1,  
 1,0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,0,0,1,  
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]  
  
 e =[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,  
 1,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,  
 1,1,1,0,1,1,1,0,0,0,1,0,1,0,0,0,0,0,1,  
 1,0,0,0,0,0,1,0,1,0,0,0,1,0,1,0,1,0,1,  
 1,1,1,1,0,1,1,0,1,0,1,0,0,0,1,1,1,1,1,  
 1,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,  
 1,1,0,1,1,1,1,0,1,1,1,0,1,0,0,0,1,0,1,  
 1,0,0,0,0,0,0,0,1,0,0,0,1,0,1,0,1,0,1,  
 1,1,1,0,1,1,1,1,1,1,0,1,1,1,1,0,0,0,1,  
 1,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,1,  
 1,1,1,0,1,0,0,0,1,1,1,1,0,1,1,1,1,0,1,  
 1,0,0,0,1,0,1,0,1,0,0,0,0,0,0,0,1,0,1,  
 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]  
  
  
 list = [a,b,c,d,e]  
 **return** list

1. References:

* Excelino
* William
* Georgius
* Aldi
* python.spot.com
* stackoverflow
* Wikipedia
* Youtube
* Github