

**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

**Assignment Cover Letter**

**(Individual Work****)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | |  | |  | |
| **Student Information**: **Surname** | | | | | **Given Names**  **Nixon** | | **Student ID Number**  **2101693523** | |
| 1. | | **Louis** |  | |
|  |  |
| **Course Code** | **: COMP6335** |  |  | | **Course Name** | | **: Introduction to Programming** | |
| **Class** | **: L1BC** |  |  | | **Name of Lecturer(s)** | | **:** 1. Minaldi Loeis | |
|  |  |  |  | |  | | 2. Jude Martinez | |
| **Major** | **: CS** |  |  | |  | |  | |
| **Title of Assignment**  (if any) | : Snake Game | |  |  | |  | |  | |
| **Type of Assignment**    **Submission Pattern** | **: Final Project** |  |  | |  | |  | |
| **Due Date** | **: 7-11-2017** |  |  | | **Submission Date** | | **: 7-11-2017** | |

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)

1.Nixon Louis

**“Snake Game”**

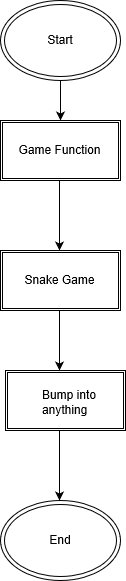
**Name : Nixon Louis**

**ID : 2101693523**

1. **Description**

**The function of this program:**

This program is to simulate an old game, to bring back memories and nostalgia of playing this game.

1. **Design/Plan**
2. **Flowchart**

**III.**

**b. Explanation of each function**

1. When the program starts it first checks if all of the functions in the program are working fine or not when it works/ if it fails

check\_errors = pygame.init()  
  
if check\_errors[1] > 0:  
  
 print("(!) Had {0} initializing errors, exiting...".format(check\_errors[1]))  
  
 sys.exit(-1)  
  
else:  
  
 print("(+) PyGame successfully initialized!")

2. After checking everything I created/implemented

-The play surface/screen

-The colors for the food, snake, scoreboard, background, and the game over sign

-The snake body and food

# Play surface  
  
playSurface = pygame.display.set\_mode((720, 460))  
  
pygame.display.set\_caption('Snake Game')  
  
  
  
# Colors  
  
red = pygame.Color(255, 0, 0) # gameover  
  
green = pygame.Color(0, 255, 0) #snake  
  
black = pygame.Color(0, 0, 0) #score  
  
white = pygame.Color(255, 255, 255) #background  
  
brown = pygame.Color(165, 42, 42) #food  
  
  
  
# FPS controller  
  
fpsController = pygame.time.Clock()  
  
  
  
# Snake Body and Food  
  
snakePos = [100, 50]  
  
snakeBody = [[100,50], [90,50], [80,50]]  
  
  
  
foodPos = [random.randrange(1,72)\*10,random.randrange(1,46)\*10]  
  
foodSpawn = True

3. After the first two functions I made the “Game Over” function and the “Show Score” function

def gameOver():  
  
 myFont = pygame.font.SysFont('monaco', 72)  
  
 GOsurf = myFont.render('Game over!', True, red)  
  
 GOrect = GOsurf.get\_rect()  
  
 GOrect.midtop = (360, 15)  
  
 playSurface.blit(GOsurf,GOrect)  
  
 showScore(0)  
  
 pygame.display.flip()  
  
  
  
 time.sleep(4)  
  
 pygame.quit() #pygame exit  
  
 sys.exit() #console exit  
  
  
  
def showScore(choice=1):  
  
 sFont = pygame.font.SysFont('monaco', 24)  
  
 Ssurf = sFont.render('Score : {0}'.format(score) , True, black)  
  
 Srect = Ssurf.get\_rect()  
  
 if choice == 1:  
  
 Srect.midtop = (80, 10)  
  
 else:  
  
 Srect.midtop = (360, 120)  
  
 playSurface.blit(Ssurf,Srect)

4. For the main logic I gave the functions for:

-The movement of the snake

-The food spawn (so it will be random)

-The limit of how far the snake can go

-If the snake hits itself the game will end

-The initial position of the snake

# Main Logic  
  
while True:  
 for event in pygame.event.get():  
  
 if event.type == pygame.QUIT:  
  
 pygame.quit()  
  
 sys.exit()  
  
 elif event.type == pygame.KEYDOWN:  
  
 if event.key == pygame.K\_RIGHT or event.key == ord('d'):  
  
 changeto = 'RIGHT'  
  
 if event.key == pygame.K\_LEFT or event.key == ord('a'):  
  
 changeto = 'LEFT'  
  
 if event.key == pygame.K\_UP or event.key == ord('w'):  
  
 changeto = 'UP'  
  
 if event.key == pygame.K\_DOWN or event.key == ord('s'):  
  
 changeto = 'DOWN'  
  
 if event.key == pygame.K\_ESCAPE:  
  
 pygame.event.post(pygame.event.Event(pygame.QUIT))

# validation of direction  
  
 if changeto == 'RIGHT' and not direction == 'LEFT':  
  
 direction = 'RIGHT'  
  
 if changeto == 'LEFT' and not direction == 'RIGHT':  
  
 direction = 'LEFT'  
  
 if changeto == 'UP' and not direction == 'DOWN':  
  
 direction = 'UP'  
  
 if changeto == 'DOWN' and not direction == 'UP':  
  
 direction = 'DOWN'  
  
  
  
 # Update snake position [x,y]  
  
 if direction == 'RIGHT':  
  
 snakePos[0] += 10  
  
 if direction == 'LEFT':  
  
 snakePos[0] -= 10  
  
 if direction == 'UP':  
  
 snakePos[1] -= 10  
  
 if direction == 'DOWN':  
  
 snakePos[1] += 10  
  
  
  
  
  
 # Snake body mechanism  
  
 snakeBody.insert(0, list(snakePos))  
  
 if snakePos[0] == foodPos[0] and snakePos[1] == foodPos[1]:  
  
 score += 1  
  
 foodSpawn = False  
  
 else:  
  
 snakeBody.pop()

#Food Spawn  
  
 if foodSpawn == False:  
  
 foodPos = [random.randrange(1,72)\*10,random.randrange(1,46)\*10]  
  
 foodSpawn = True  
  
  
  
 #Background  
  
 playSurface.fill(white)  
  
  
  
  
 #Draw Snake  
  
 for pos in snakeBody:  
  
 pygame.draw.rect(playSurface, green, pygame.Rect(pos[0],pos[1],10,10))  
  
  
  
 pygame.draw.rect(playSurface, brown, pygame.Rect(foodPos[0],foodPos[1],10,10))  
  
  
  
 # Bound  
  
 if snakePos[0] > 710 or snakePos[0] < 0:  
  
 gameOver()  
  
 if snakePos[1] > 450 or snakePos[1] < 0:  
  
 gameOver()  
  
  
  
 # Self hit  
  
 for block in snakeBody[1:]:  
  
 if snakePos[0] == block[0] and snakePos[1] == block[1]:  
  
 gameOver()

**IV. Problems Overcame while doing this project**

1. **Developing my skills to work individually**

**-I usually work better in a team**

1. **Making a game with the amount of skills learnt in the past 2-3 weeks**

**-very difficult with lots of free time put into it**

1. **Getting the idea of the game**

**-idea from github**