Odd Semester (2021)



**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

**Assignment Cover Letter**

**(Individual Work**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | |  | |  | |
| **Student Information**: **Surname** | | | | | **Given Names**  **Dean Jourdan** | | **Student ID Number**  **2101705164** | |
| 1. | | **Dumais** |  | |
|  |  |
| **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) | : Final Project-The Snake 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 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. Dean Jourdan Dumais

**“The Snake Game”**

**Name : Dean Jourdan Dumais**

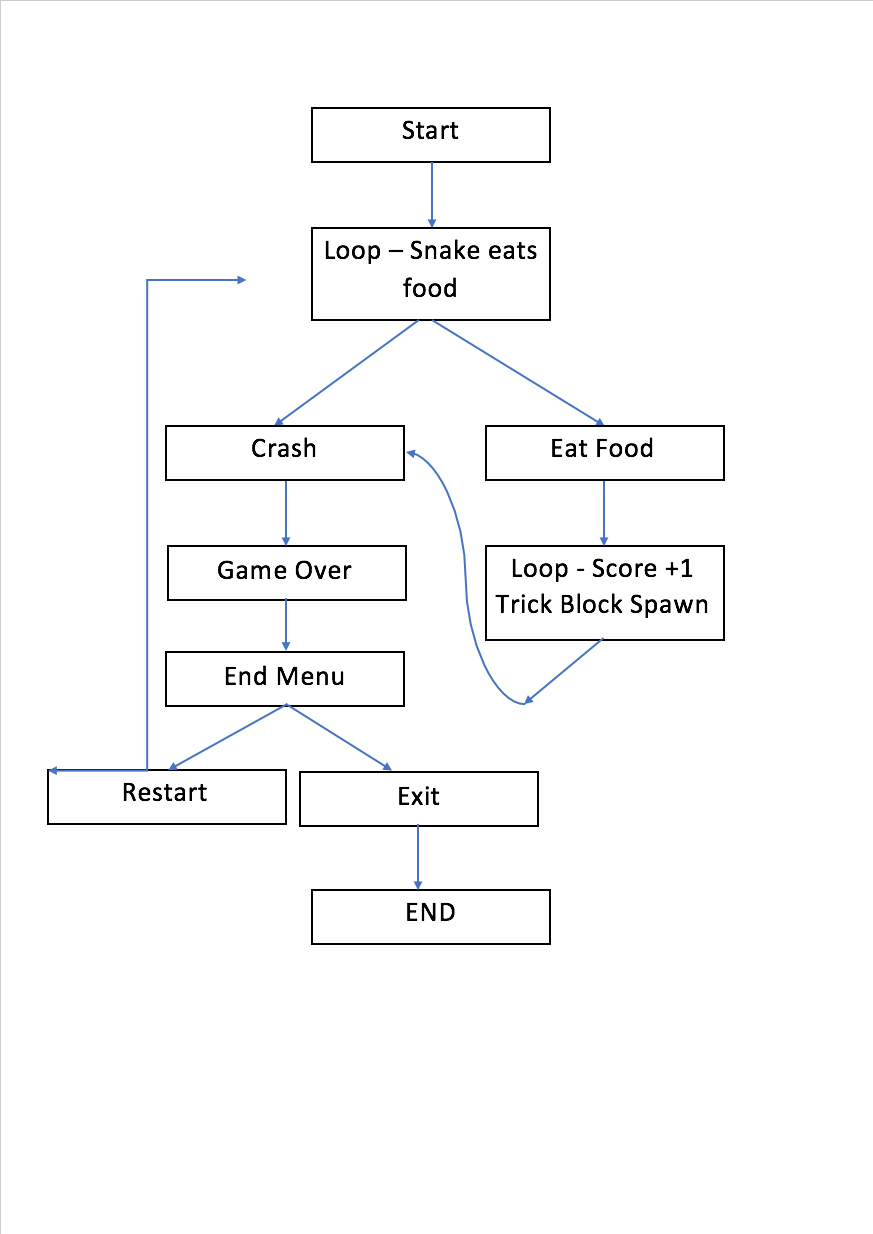
**ID : 2101705164**

1. **Description**

**The function of this program:**

This program is a remake of the classic Snake Game but with some modifications.

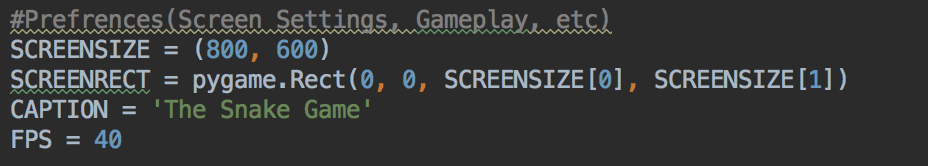
**II.a. Design/Plan**

**FlowChart**

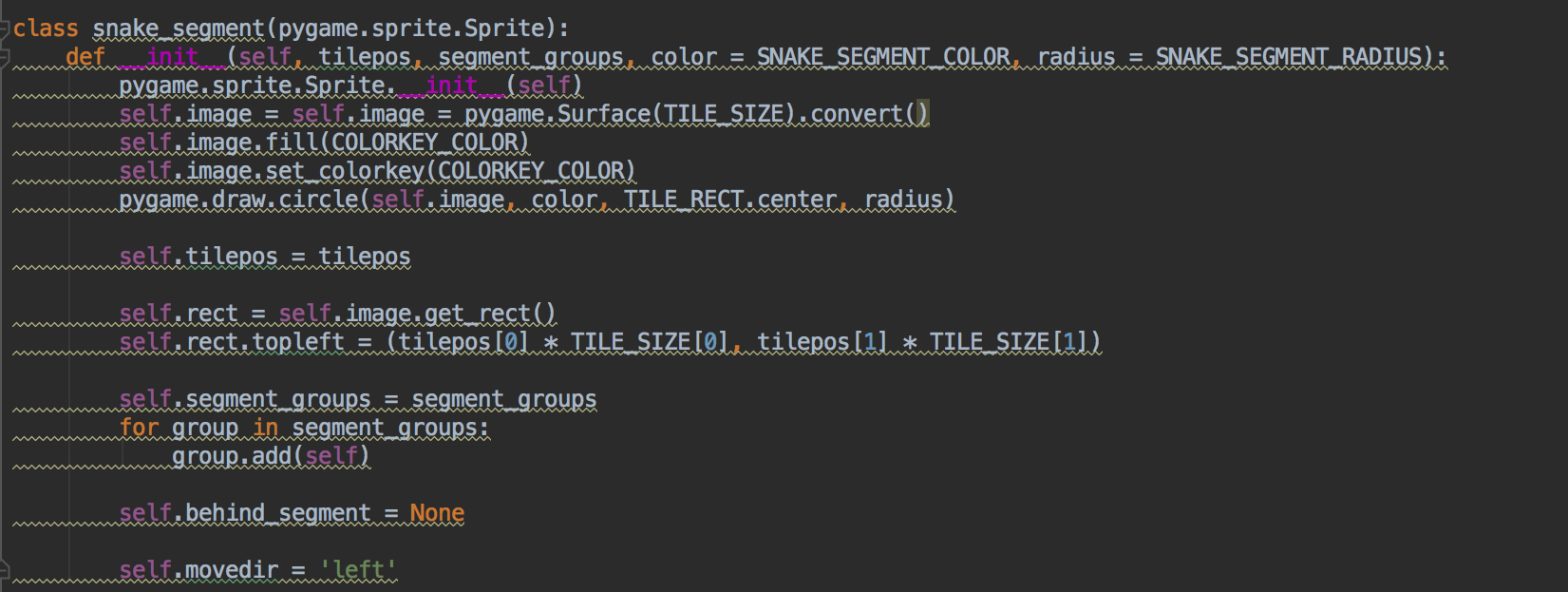
**II.b. Explanation of Each function**

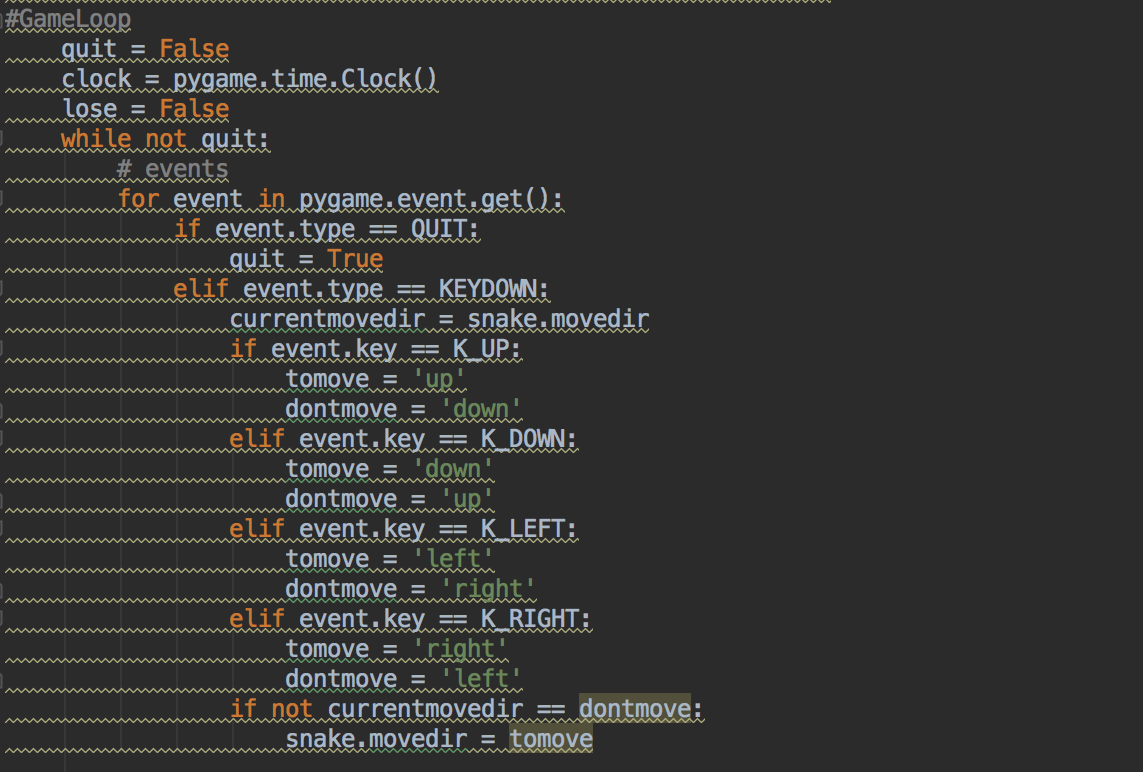
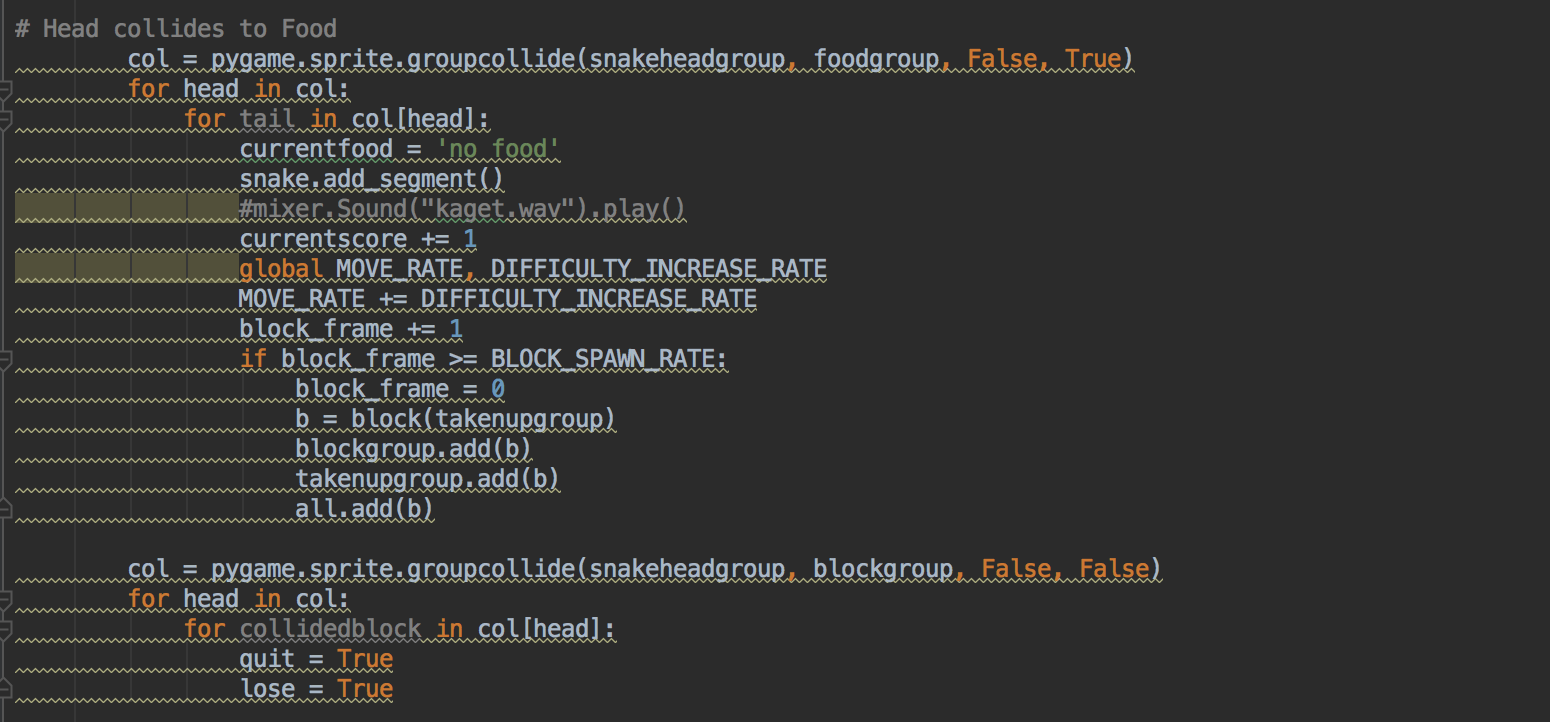
1. **Screen Settings :**

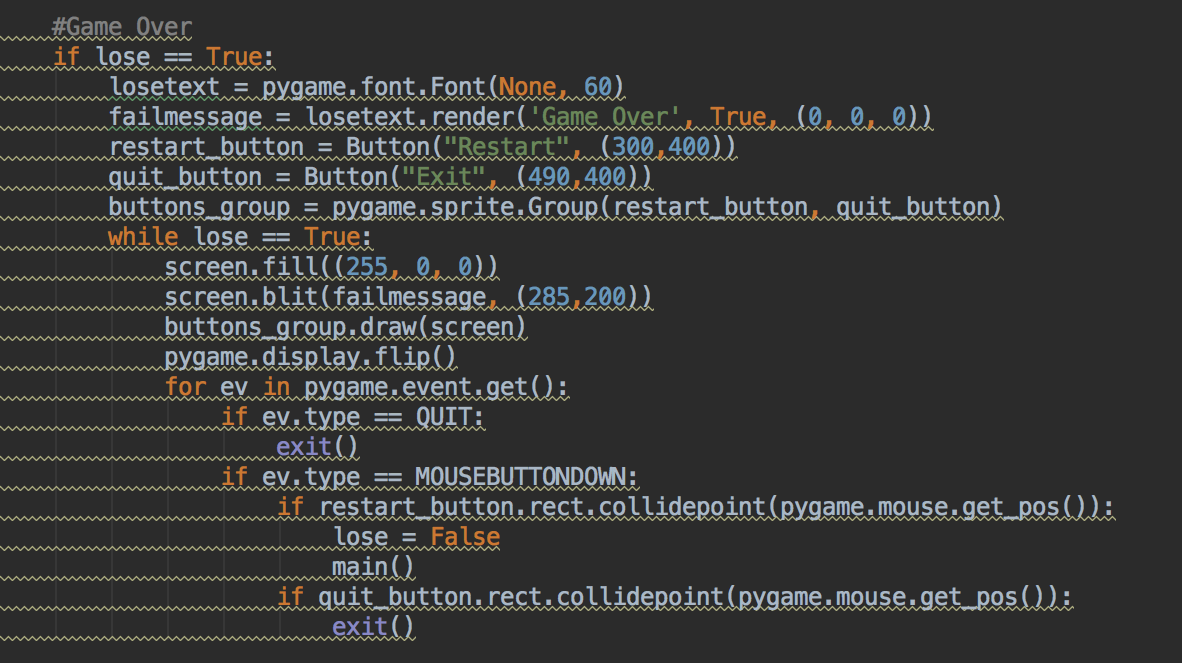
**I use an (800, 600) window**



1. **Snake Settings :**

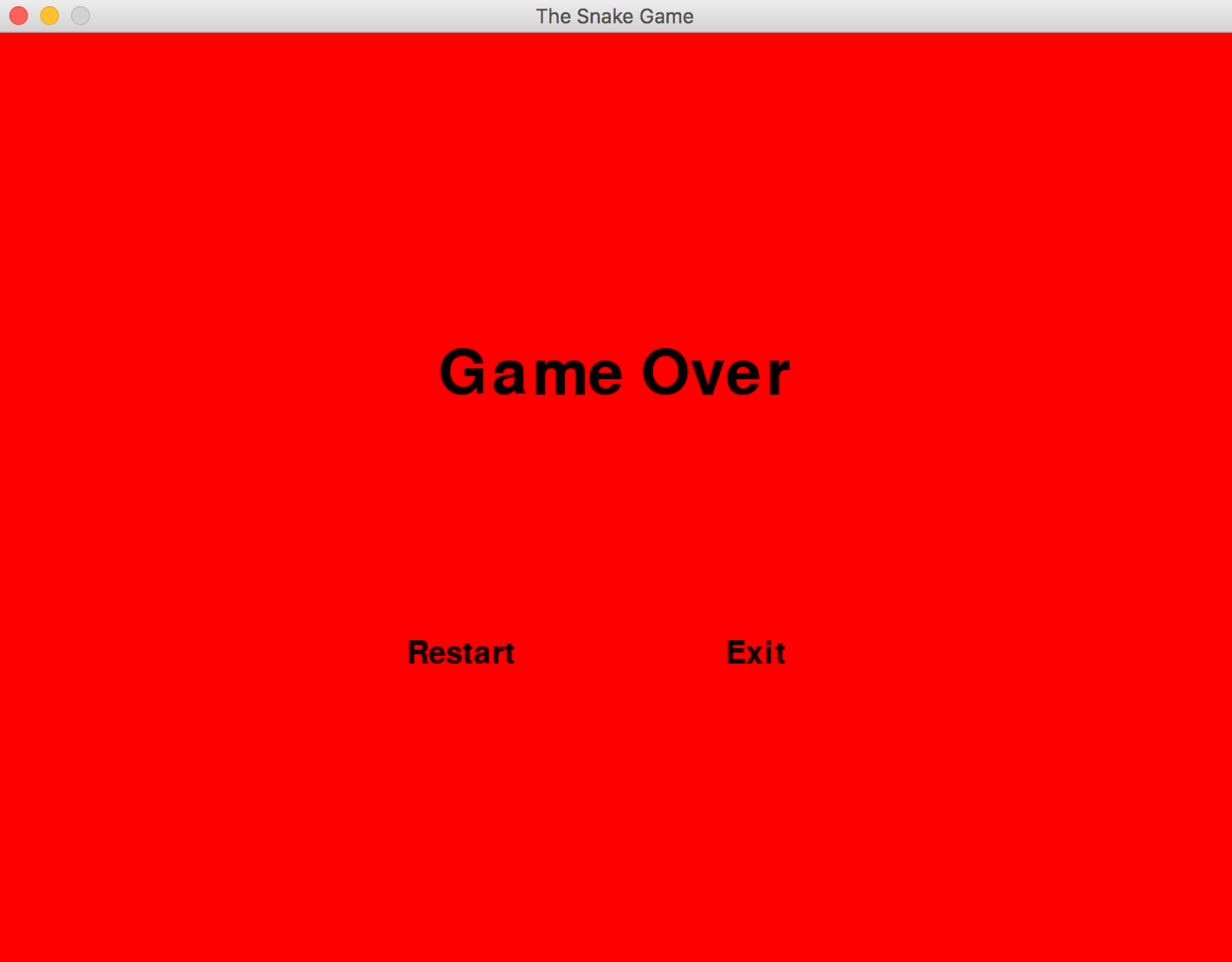
**I use Class for the snake’s Body which will add up each time it eats the food.**

1. **The Game Loop, The Game loop I for, while, if, elif commands and also mouse events.**
2. **Collision, to end the game when it collides to the wall of the trick block.**

**If you hit a wall or brick it will go to your game over menu**

**IIIB. Game Appearance**

**Game Appearance**

**Game Over Menu**

**IV. Main Code :**

#CODE IS ORIGINALLY FROM DANIEL WESTBROOK  
#I DID SOME MODIFICATION REGARDING THE GAMEPLAY, GAME CONCEPT, SOME APPEARANCE CHANGE AND THE ADDING  
#SOME MENUS, BUG CLEAN UPS, AND ETC  
  
  
import pygame  
from pygame.locals import \*  
import random  
import time  
  
#Prefrences(Screen Settings, Gameplay, etc)  
screensize = (800, 600)  
SCREENRECT = pygame.Rect(0, 0, SCREENSIZE[0], SCREENSIZE[1])  
CAPTION = 'The Snake Game'  
FPS = 40  
  
START\_TILE = (20, 20) #Snake's Starting Point  
START\_SEGMENTS = 7 #Starting Snake Size  
  
MOVE\_RATE = 2  
DIFFICULTY\_INCREASE\_RATE = 1 #Snake Speed Increasing everytime it eats the food  
MOVE\_THRESHOLD = 5  
BLOCK\_SPAWN\_RATE = 1 #Spawn Trick Block  
  
TILE\_SIZE = (10, 10)  
TILE\_RECT = pygame.Rect(0, 0, TILE\_SIZE[0], TILE\_SIZE[1])  
  
SCREENTILES = ((SCREENSIZE[0] / TILE\_SIZE[0]) - 1, (SCREENSIZE[1] / TILE\_SIZE[1]) - 1)  
  
SNAKE\_HEAD\_RADIUS = 5  
SNAKE\_SEGMENT\_RADIUS = 4  
FOOD\_RADIUS = 4  
  
BG\_Colour = (255, 255, 255)  
SNAKE\_HEAD\_COLOR = (219, 79, 70)  
SNAKE\_SEGMENT\_COLOR = (36, 200, 243)  
FOOD\_COLOR = (0, 255, 0)  
BLOCK\_COLOR = (0, 0, 150)  
COLORKEY\_COLOR = (255, 255, 0)  
  
SCORE\_COLOR = (0, 0, 0) #Score Test Colour  
SCORE\_POS = (20, 20) # Score position on Screen  
SCORE\_PREFIX = 'Score: ' #Score text template  
  
MOVE\_VECTORS = {'left' : (-1, 0),  
 'right' : (1, 0),  
 'up' : (0, -1),  
 'down' : (0, 1)  
 }  
MOVE\_VECTORS\_PIXELS = {'left' : (-TILE\_SIZE[0], 0),  
 'right' : (TILE\_SIZE[0], 0),  
 'up' : (0, -TILE\_SIZE[1]),  
 'down' : (0, TILE\_SIZE[1])  
 }  
  
#-----------------------------------------------------------------  
# Game Objects  
class Button(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, message, coor):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 f = pygame.font.Font(None, 30)  
 self.image = f.render(message, True, (0, 0, 0))  
 self.rect = self.image.get\_rect()  
 self.rect.center = coor  
  
  
class snake\_segment(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, tilepos, segment\_groups, color = SNAKE\_SEGMENT\_COLOR, radius = SNAKE\_SEGMENT\_RADIUS):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image = self.image = pygame.Surface(TILE\_SIZE).convert()  
 self.image.fill(COLORKEY\_COLOR)  
 self.image.set\_colorkey(COLORKEY\_COLOR)  
 pygame.draw.circle(self.image, color, TILE\_RECT.center, radius)  
  
 self.tilepos = tilepos  
  
 self.rect = self.image.get\_rect()  
 self.rect.topleft = (tilepos[0] \* TILE\_SIZE[0], tilepos[1] \* TILE\_SIZE[1])  
  
 self.segment\_groups = segment\_groups  
 for group in segment\_groups:  
 group.add(self)  
  
 self.behind\_segment = None  
  
 self.movedir = 'left'  
  
 def add\_segment(self): #to add the length of the snake  
 seg = self  
 while True:  
 if seg.behind\_segment == None:  
 x = seg.tilepos[0]  
 y = seg.tilepos[1]  
 if seg.movedir == 'left':  
 x += 1  
 elif seg.movedir == 'right':  
 x -= 1  
 elif seg.movedir == 'up':  
 y += 1  
 elif seg.movedir == 'down':  
 y -= 1  
 seg.behind\_segment = snake\_segment((x, y), seg.segment\_groups)  
 seg.behind\_segment.movedir = seg.movedir  
 break  
 else:  
 seg = seg.behind\_segment  
  
 def update(self):  
 pass  
  
 def move(self):  
 self.tilepos = (self.tilepos[0] + MOVE\_VECTORS[self.movedir][0], self.tilepos[1] + MOVE\_VECTORS[self.movedir][1])  
 self.rect.move\_ip(MOVE\_VECTORS\_PIXELS[self.movedir])  
 if self.behind\_segment != None:  
 self.behind\_segment.move()  
 self.behind\_segment.movedir = self.movedir  
  
class snake\_head(snake\_segment):  
 def \_\_init\_\_(self, tilepos, movedir, segment\_groups):  
 snake\_segment.\_\_init\_\_(self, tilepos, segment\_groups, color = SNAKE\_HEAD\_COLOR, radius = SNAKE\_HEAD\_RADIUS)  
 self.movedir = movedir  
 self.movecount = 0  
  
 def update(self):  
 self.movecount += MOVE\_RATE  
 if self.movecount > MOVE\_THRESHOLD:  
 self.move()  
 self.movecount = 0  
  
class food(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, takenupgroup):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image = self.image = pygame.Surface(TILE\_SIZE).convert()  
 self.image.fill(COLORKEY\_COLOR)  
 self.image.set\_colorkey(COLORKEY\_COLOR)  
 pygame.draw.circle(self.image, FOOD\_COLOR, TILE\_RECT.center, FOOD\_RADIUS)  
  
 self.rect = self.image.get\_rect()  
 while True:  
 self.rect.topleft = (random.randint(0, SCREENTILES[0]) \* TILE\_SIZE[0], random.randint(0, SCREENTILES[1]) \* TILE\_SIZE[1])  
 for sprt in takenupgroup:  
 if self.rect.colliderect(sprt):  
 continue # collision, food cant go here  
 break # no collision, food can go here  
  
class block(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, takenupgroup):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image = self.image = pygame.Surface(TILE\_SIZE).convert()  
 self.image.fill(BLOCK\_COLOR)  
  
 self.rect = self.image.get\_rect()  
 while True:  
 self.rect.topleft = (random.randint(0, SCREENTILES[0]) \* TILE\_SIZE[0], random.randint(0, SCREENTILES[1]) \* TILE\_SIZE[1])  
 for sprt in takenupgroup:  
 if self.rect.colliderect(sprt):  
 continue # collision, food cant go here  
 break # no collision, food can go here  
  
#------------------------------------------------------  
#Game Conditions  
def main():  
 pygame.init()  
 screen = pygame.display.set\_mode(screensize)  
 pygame.display.set\_caption(CAPTION)  
 bg = pygame.Surface(screensize).convert()  
 bg.fill(BG\_Colour)  
 screen.blit(bg, (0, 0))  
  
 snakegroup = pygame.sprite.Group()  
 snakeheadgroup = pygame.sprite.Group()  
 foodgroup = pygame.sprite.Group()  
 blockgroup = pygame.sprite.Group()  
 takenupgroup = pygame.sprite.Group()  
 all = pygame.sprite.RenderUpdates()  
  
 snake = snake\_head(START\_TILE, 'right', [snakegroup, all, takenupgroup])  
 snakeheadgroup.add(snake)  
 for index in range(START\_SEGMENTS):  
 snake.add\_segment()  
  
 currentfood = 'no food'  
  
 block\_frame = 0  
  
 currentscore = 0  
  
 pygame.display.flip()  
  
#----------------------------------------------------------  
#GameLoop  
 quit = False  
 clock = pygame.time.Clock()  
 lose = False  
 while not quit:  
 # events  
 for event in pygame.event.get():  
 if event.type == QUIT:  
 quit = True  
 elif event.type == KEYDOWN:  
 currentmovedir = snake.movedir  
 if event.key == K\_UP:  
 tomove = 'up'  
 dontmove = 'down'  
 elif event.key == K\_DOWN:  
 tomove = 'down'  
 dontmove = 'up'  
 elif event.key == K\_LEFT:  
 tomove = 'left'  
 dontmove = 'right'  
 elif event.key == K\_RIGHT:  
 tomove = 'right'  
 dontmove = 'left'  
 if not currentmovedir == dontmove:  
 snake.movedir = tomove  
  
# clearing  
 all.clear(screen, bg)  
# updates  
 all.update()  
  
 if currentfood == 'no food':  
 currentfood = food(takenupgroup)  
 foodgroup.add(currentfood)  
 takenupgroup.add(currentfood)  
 all.add(currentfood)  
  
 pos = snake.rect.topleft  
 if pos[0] < 0:  
 quit = True  
 lose = True  
 if pos[0] >= SCREENSIZE[0]:  
 quit = True  
 lose = True  
 if pos[1] < 0:  
 quit = True  
 lose = True  
 if pos[1] >= SCREENSIZE[1]:  
 quit = True  
 lose = True  
  
# Head collides to Food  
 col = pygame.sprite.groupcollide(snakeheadgroup, foodgroup, False, True)  
 for head in col:  
 for tail in col[head]:  
 currentfood = 'no food'  
 snake.add\_segment()  
  
 currentscore += 1  
 global MOVE\_RATE, DIFFICULTY\_INCREASE\_RATE  
 MOVE\_RATE += DIFFICULTY\_INCREASE\_RATE  
 block\_frame += 1  
 if block\_frame >= BLOCK\_SPAWN\_RATE:  
 block\_frame = 0  
 b = block(takenupgroup)  
 blockgroup.add(b)  
 takenupgroup.add(b)  
 all.add(b)  
  
# Head Collides to trick block  
 col = pygame.sprite.groupcollide(snakeheadgroup, blockgroup, False, False)  
 for head in col:  
 for collidedblock in col[head]:  
 quit = True  
 lose = True  
  
 #Scoring System  
 d = screen.blit(bg, SCORE\_POS, pygame.Rect(SCORE\_POS, (50, 100)))  
 f = pygame.font.Font(None, 25)  
 scoreimage = f.render(SCORE\_PREFIX + str(currentscore), True, SCORE\_COLOR)  
 d2 = screen.blit(scoreimage, SCORE\_POS)  
  
 # Draw Game  
 dirty = all.draw(screen)  
 dirty.append(d)  
 dirty.append(d2)  
  
 # Display Update  
 pygame.display.update(dirty)  
  
 # waiting  
 clock.tick(FPS)  
  
 #Game Over  
 if lose == True:  
 losetext = pygame.font.Font(None, 60)  
 failmessage = losetext.render('Game Over', True, (0, 0, 0))  
 restart\_button = Button("Restart", (300,400))  
 quit\_button = Button("Exit", (490,400))  
 buttons\_group = pygame.sprite.Group(restart\_button, quit\_button)  
 while lose == True:  
 screen.fill((255, 0, 0))  
 screen.blit(failmessage, (285,200))  
 buttons\_group.draw(screen)  
 pygame.display.flip()  
 for ev in pygame.event.get():  
 if ev.type == QUIT:  
 exit()  
 if ev.type == MOUSEBUTTONDOWN:  
 if restart\_button.rect.collidepoint(pygame.mouse.get\_pos()):  
 lose = False  
 main()  
 if quit\_button.rect.collidepoint(pygame.mouse.get\_pos()):  
 exit()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Main code is from

Westbrooke, Daniel. “Mini Snake.” *Pixelated Awesome*, 24 Jan. 2009, projects.pixelatedawesome.com/minisnake/releases/minisnake\_V1.0.tar.gz.

But I did some modifications.