Wormy Game
3/22/2020

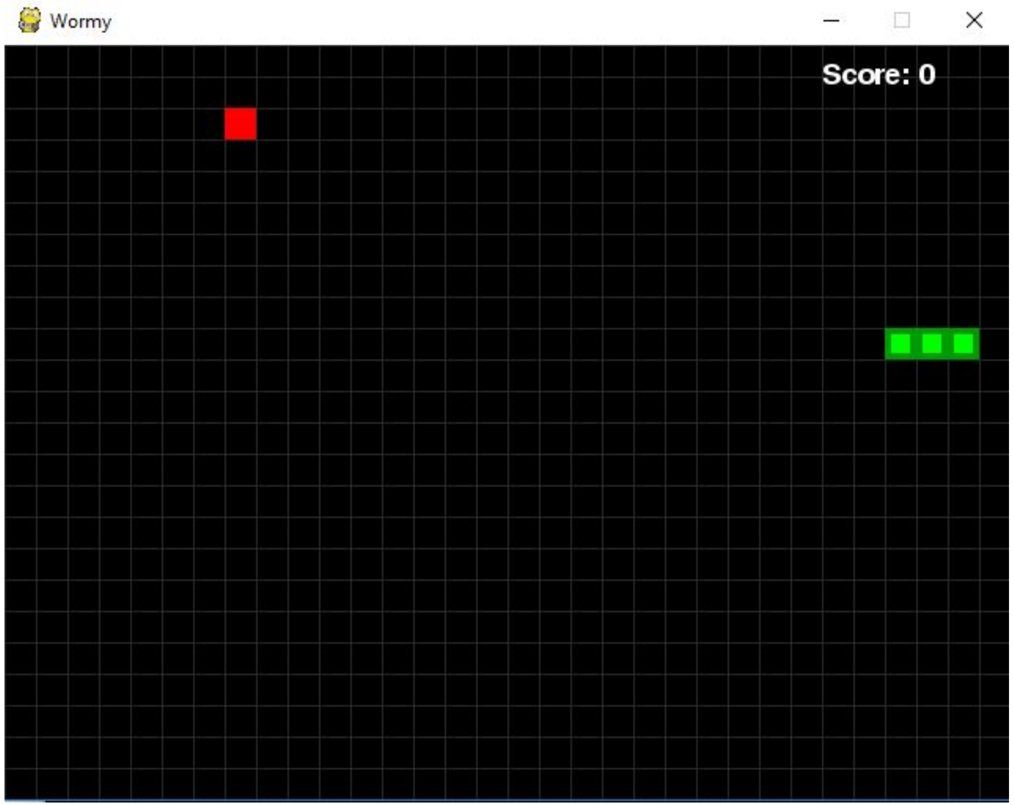
Exercise

```python
apple = {
    'color': 'red',
    'Size': 20
    }

banana = {
    'color': 'yellow',
    'size': 10
    }

fruit = [ apple, banana ]

print(fruit)

print(fruit[0])

print(fruit[-1])

print(fruit[0]['color'])

print(fruit[1]['size'])
```

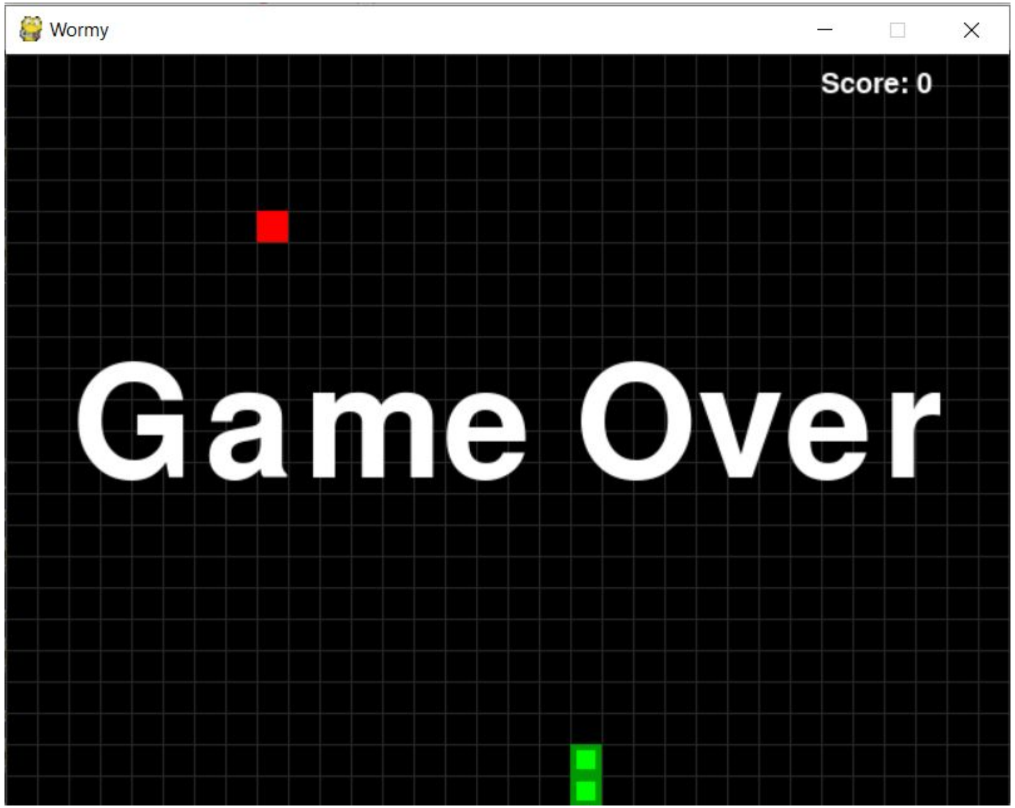
```
apple = {
    'color': ['red', 'yellow'],
    'size': [10, 20, 30]
    }

print(apple)

print(apple['color'])

print(apple['color'][1])
```

Game rules

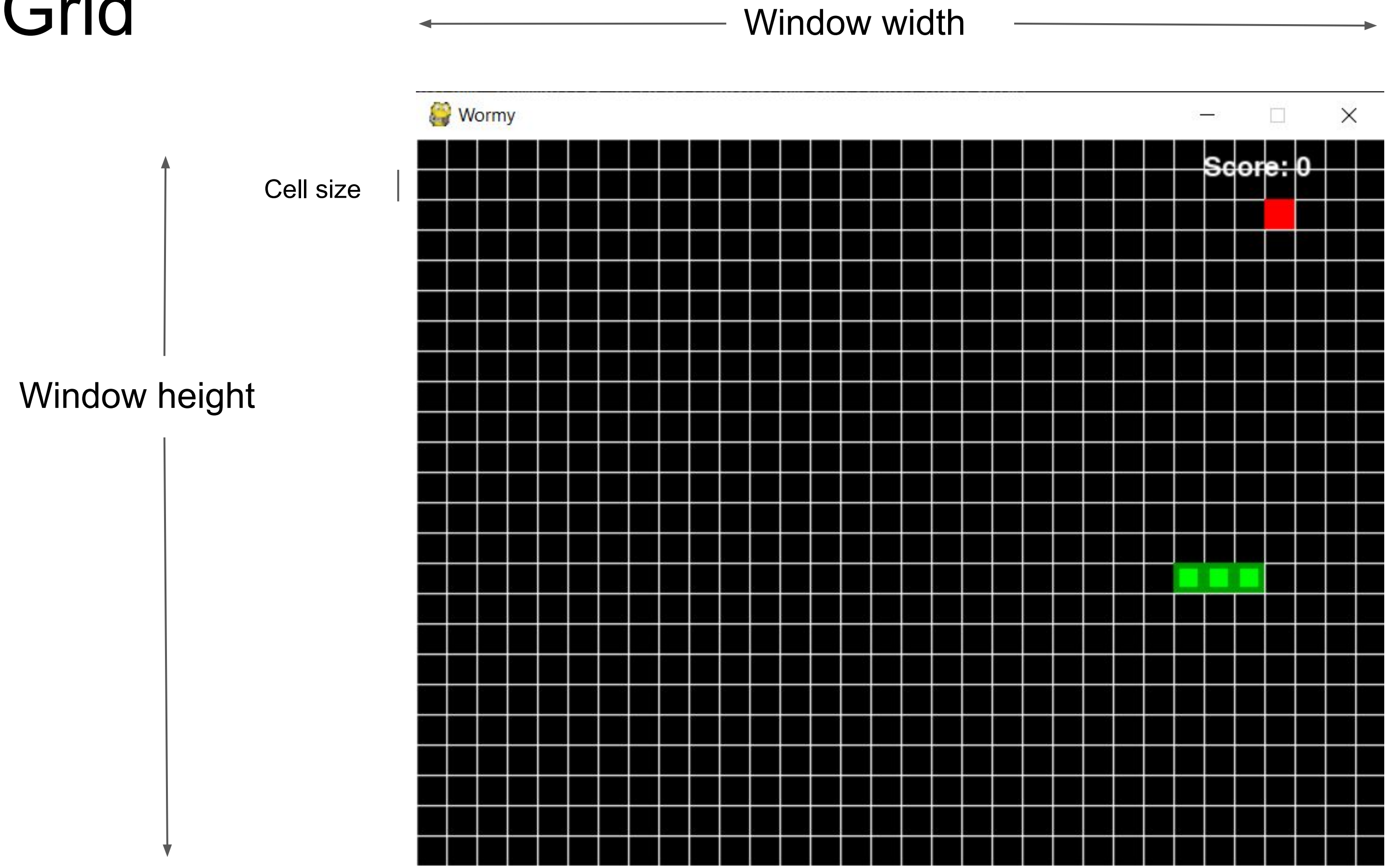
1. Eat the apple
2. Don’t hit the wall
3. Get the highest score



Game over

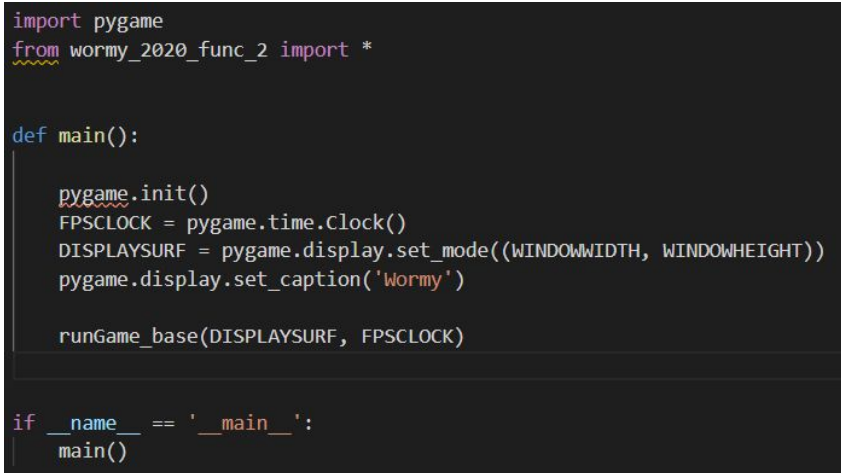


Grid



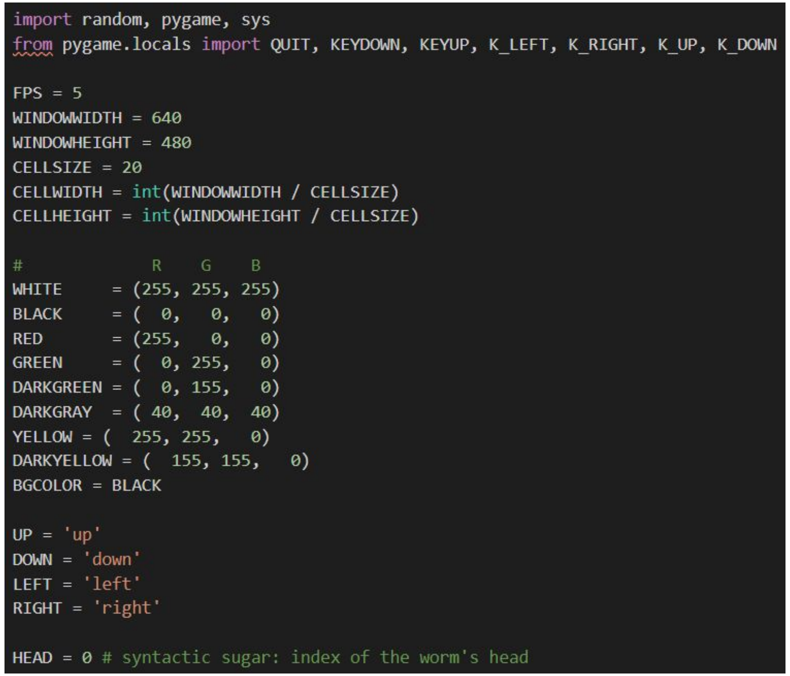
Main and func python script

wormy_main.py



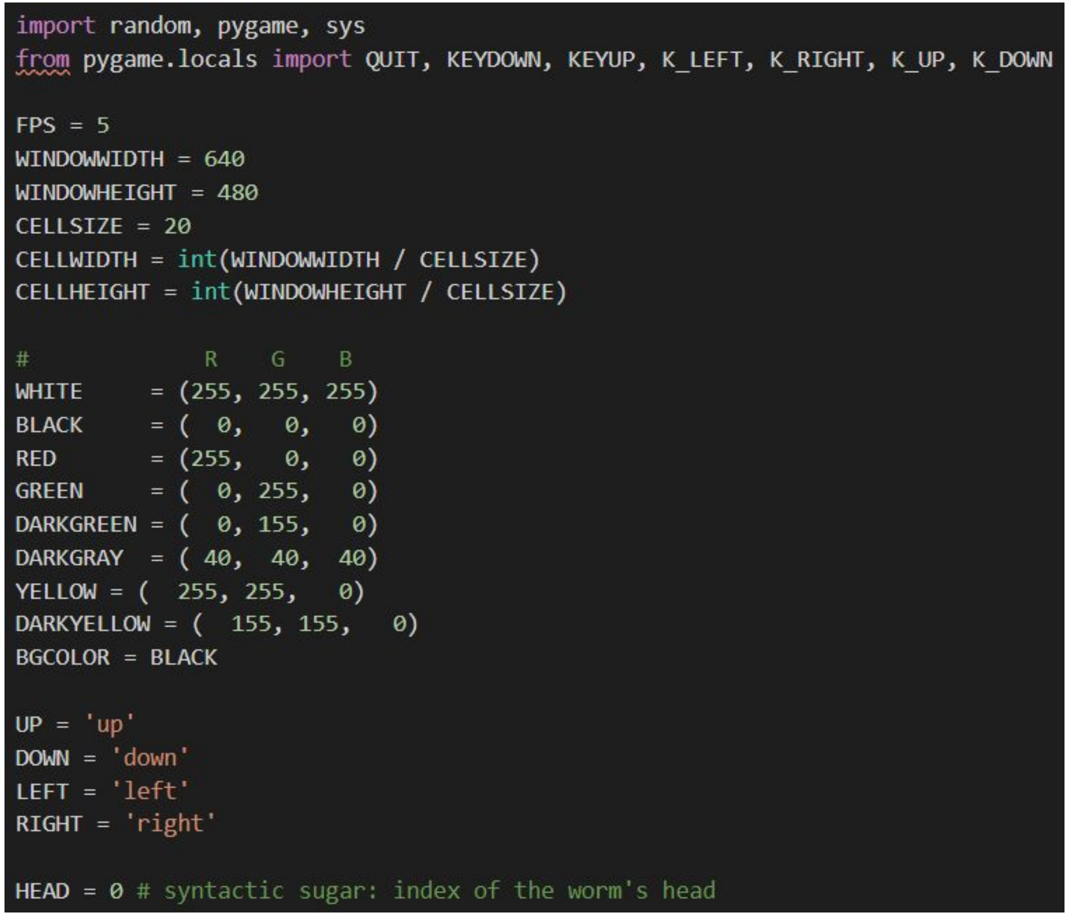
```python
import pygame
from wormy_2020_func_2 import *


def main():

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    pygame.display.set_caption('Wormy')

    runGame_base(DISPLAYSURF, FPSCLOCK)


if __name__ == '__main__':
    main()
```

wormy_function.py



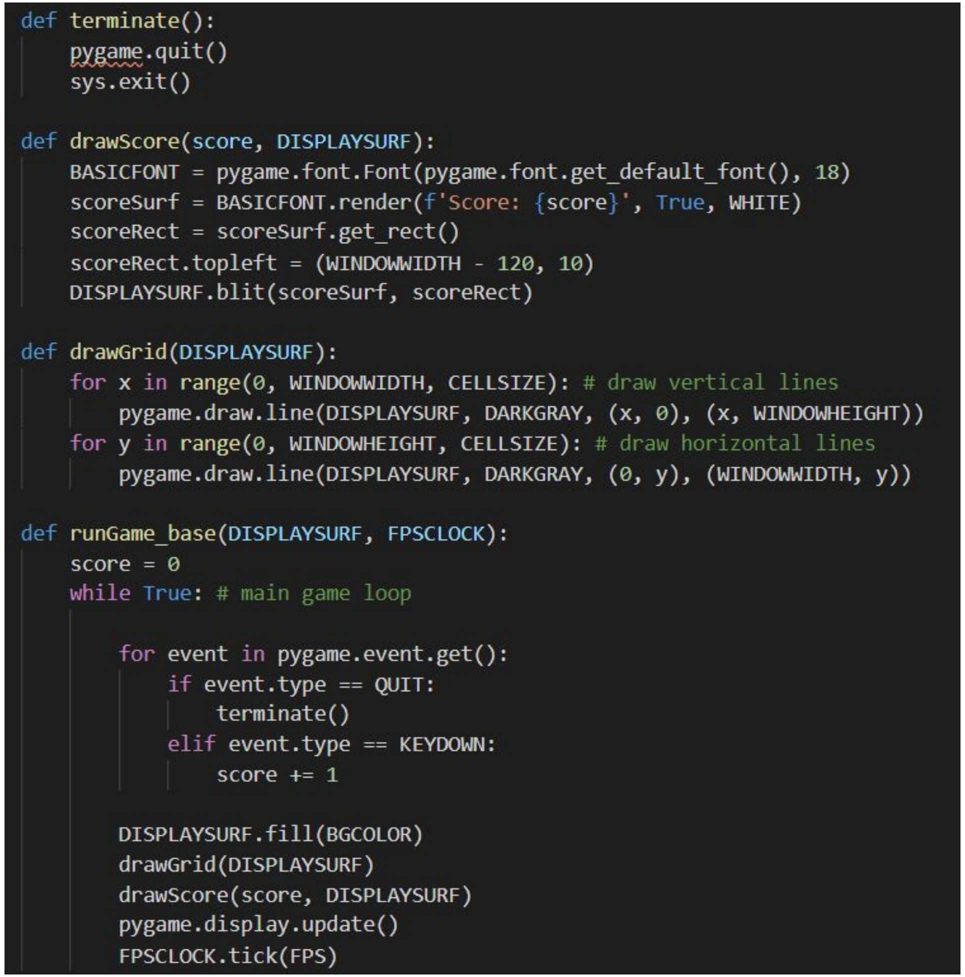
```python
import random, pygame, sys
from pygame.locals import QUIT, KEYDOWN, KEYUP, K_LEFT, K_RIGHT, K_UP, K_DOWN

FPS = 5
WINDOWWIDTH = 640
WINDOWHEIGHT = 480
CELLSIZE = 20
CELLWIDTH = int(WINDOWWIDTH / CELLSIZE)
CELLHEIGHT = int(WINDOWHEIGHT / CELLSIZE)

#             R    G    B
WHITE     = (255, 255, 255)
BLACK     = (  0,   0,   0)
RED       = (255,   0,   0)
GREEN     = (  0, 255,   0)
DARKGREEN = (  0, 155,   0)
DARKGRAY  = ( 40,  40,  40)
YELLOW = (  255, 255,   0)
DARKYELLOW = (  155, 155,   0)
BGCOLOR = BLACK

UP = 'up'
DOWN = 'down'
LEFT = 'left'
RIGHT = 'right'

HEAD = 0 # syntactic sugar: index of the worm's head
```

wormy_function.py
Game setup



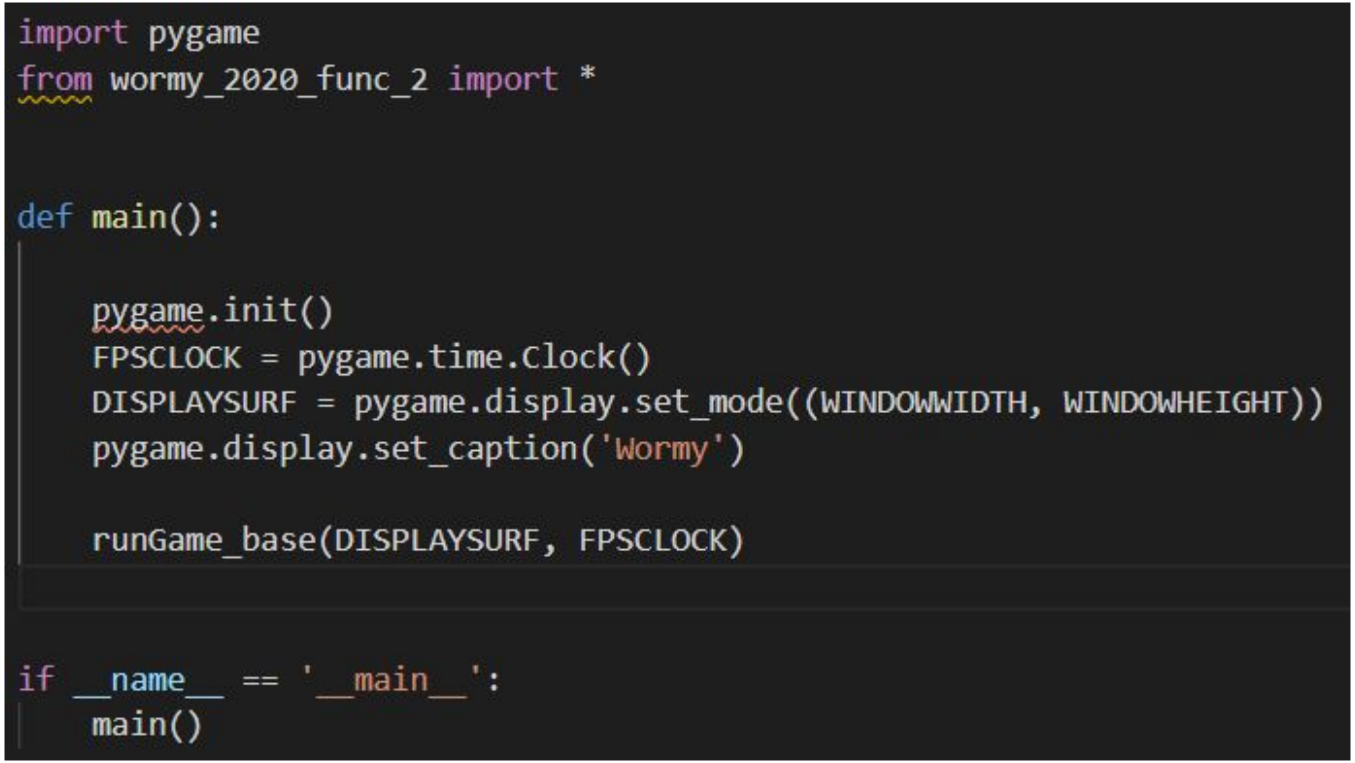
```python
import random, pygame, sys
from pygame.locals import QUIT, KEYDOWN, KEYUP, K_LEFT, K_RIGHT, K_UP, K_DOWN

FPS = 5
WINDOWWIDTH = 640
WINDOWHEIGHT = 480
CELLSIZE = 20
CELLWIDTH = int(WINDOWWIDTH / CELLSIZE)
CELLHEIGHT = int(WINDOWHEIGHT / CELLSIZE)

#             R    G    B
WHITE     = (255, 255, 255)
BLACK     = (  0,   0,   0)
RED       = (255,   0,   0)
GREEN     = (  0, 255,   0)
DARKGREEN = (  0, 155,   0)
DARKGRAY  = ( 40,  40,  40)
YELLOW = (  255, 255,   0)
DARKYELLOW = (  155, 155,   0)
BGCOLOR = BLACK

UP = 'up'
DOWN = 'down'
LEFT = 'left'
RIGHT = 'right'

HEAD = 0 # syntactic sugar: index of the worm's head
```

wormy_function continue



```python
def terminate():
    pygame.quit()
    sys.exit()

def drawScore(score, DISPLAYSURF):
    BASICFONT = pygame.font.Font(pygame.font.get_default_font(), 18)
    scoreSurf = BASICFONT.render(f'Score: {score}', True, WHITE)
    scoreRect = scoreSurf.get_rect()
    scoreRect.topleft = (WINDOWWIDTH - 120, 10)
    DISPLAYSURF.blit(scoreSurf, scoreRect)

def drawGrid(DISPLAYSURF):
    for x in range(0, WINDOWWIDTH, CELLSIZE): # draw vertical lines
        pygame.draw.line(DISPLAYSURF, DARKGRAY, (x, 0), (x, WINDOWHEIGHT))
    for y in range(0, WINDOWHEIGHT, CELLSIZE): # draw horizontal lines
        pygame.draw.line(DISPLAYSURF, DARKGRAY, (0, y), (WINDOWWIDTH, y))

def runGame_base(DISPLAYSURF, FPSCLOCK):
    score = 0
    while True: # main game loop

        for event in pygame.event.get():
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                score += 1

        DISPLAYSURF.fill(BGCOLOR)
        drawGrid(DISPLAYSURF)
        drawScore(score, DISPLAYSURF)
        pygame.display.update()
        FPSCLOCK.tick(FPS)
```

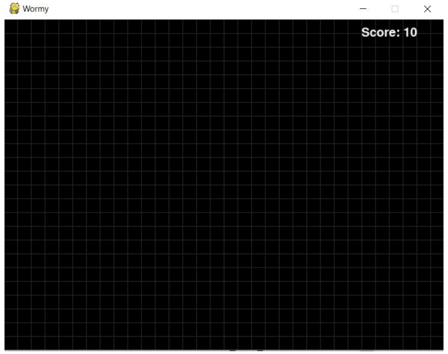
wormy_main.py



```
import pygame
from wormy_2020_func_2 import *


def main():

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    pygame.display.set_caption('Wormy')

    runGame_base(DISPLAYSURF, FPSCLOCK)


if __name__ == '__main__':
    main()
```

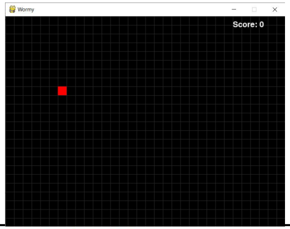
Wormy_2020_func.py: runGame_1

```
import pygame
from wormy_2020_func import *


def main():

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    pygame.display.set_caption('Wormy')

##    runGame_base(DISPLAYSURF, FPSCLOCK)
    runGame_1(DISPLAYSURF, FPSCLOCK)
##    runGame_show_apple(DISPLAYSURF, FPSCLOCK)
##    runGame_show_worm(DISPLAYSURF, FPSCLOCK)
##    showGameOverScreen_base(DISPLAYSURF)

##    while True:
##        runGame(DISPLAYSURF, FPSCLOCK)
##        showGameOverScreen(DISPLAYSURF)


if __name__ == '__main__':
    main()
```



```
def runGame_1(DISPLAYSURF, FPSCLOCK):
    score = 0
    while True: # main game loop

        for event in pygame.event.get():
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if event.key == K_LEFT:
                    score -= 1
                elif event.key == K_RIGHT:
                    score += 1
                elif event.key == K_UP:
                    score += 10
                elif event.key == K_DOWN:
                    score -= 10

        DISPLAYSURF.fill(BGCOLOR)
        drawGrid(DISPLAYSURF)
        drawScore(score, DISPLAYSURF)
        pygame.display.update()
        FPSCLOCK.tick(FPS)
```

Draw apple



wormy_2020_main.py

```python
import pygame
from wormy_2020_func import *


def main():

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    pygame.display.set_caption('Wormy')

##    runGame_base(DISPLAYSURF, FPSCLOCK)
##    runGame_1(DISPLAYSURF, FPSCLOCK)
    runGame_show_apple(DISPLAYSURF, FPSCLOCK)
##    runGame_show_worm(DISPLAYSURF, FPSCLOCK)
##    showGameOverScreen_base(DISPLAYSURF)

##    while True:
##        runGame(DISPLAYSURF, FPSCLOCK)
##        showGameOverScreen(DISPLAYSURF)


if __name__ == '__main__':
    main()
```

wormy_2020_func.py

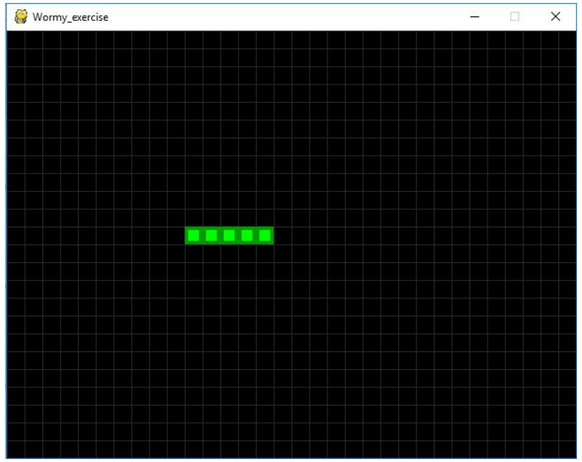
```python
class Apple(object):
    def __init__(self, cell_width, cell_height, cell_size):
        self.cell_width = cell_width
        self.cell_height = cell_height
        self.cell_size = cell_size
        self.update()

    def draw(self, DISPLAYSURF):
        x = self.Coord['x'] * self.cell_size
        y = self.Coord['y'] * self.cell_size
        appleRect = pygame.Rect(x, y, self.cell_size, self.cell_size)
        pygame.draw.rect(DISPLAYSURF, RED, appleRect)

    def update(self):
        self.Coord = {'x': random.randint(0, self.cell_width - 1), \
                      'y': random.randint(0, self.cell_height - 1)}
```

wormy_2020_func.py

```python
def runGame_show_apple(DISPLAYSURF, FPSCLOCK):
    score = 0
    apple = Apple(CELLWIDTH, CELLHEIGHT, CELLSIZE)
    while True: # main game loop

        for event in pygame.event.get(): # event handling
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if event.key == K_LEFT:
                    score -= 1
                elif event.key == K_RIGHT:
                    score += 1
                elif event.key == K_UP:
                    score += 10
                elif event.key == K_DOWN:
                    score -= 10
                else:
                    apple.update()

        DISPLAYSURF.fill(BGCOLOR)
        drawGrid(DISPLAYSURF)
        drawScore(score, DISPLAYSURF)
        apple.draw(DISPLAYSURF)
        pygame.display.update()
        FPSCLOCK.tick(FPS)
```

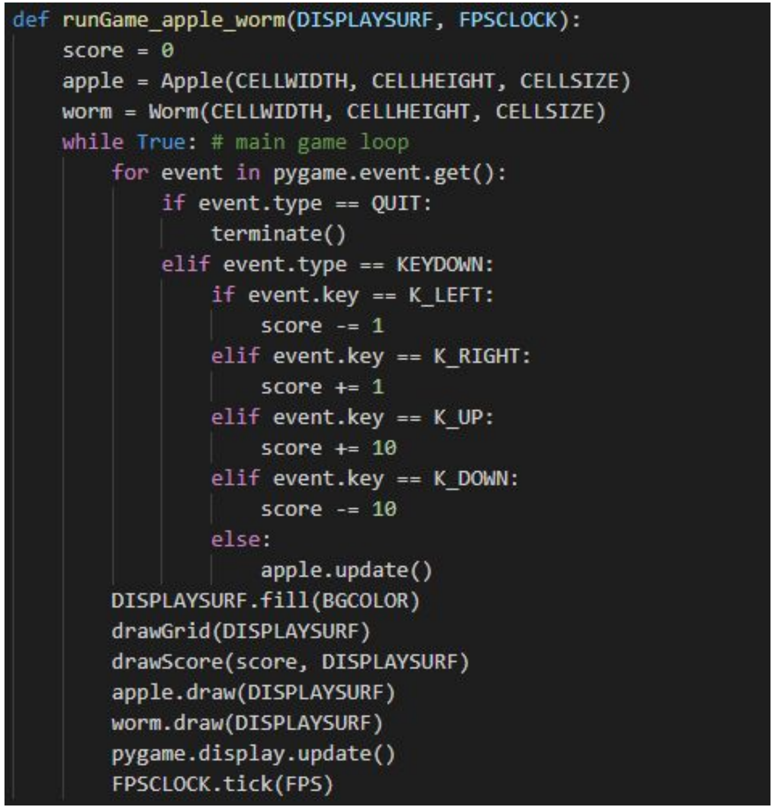
Worm class

```
class Worm(object):
    def __init__(self, cell_width, cell_height, cell_size):
        self.cell_width = cell_width
        self.cell_height = cell_height
        self.cell_size = cell_size
        self.direction = RIGHT
        # Set a random start point.
        margin = 5
        startx = random.randint(margin, cell_width - margin)
        starty = random.randint(margin, cell_height - margin)
        self.Coords = [{'x': startx,     'y': starty},
                       {'x': startx - 1, 'y': starty},
                       {'x': startx - 2, 'y': starty}]

    def draw(self, DISPLAYSURF):
        for coord in self.Coords:
            x = coord['x'] * self.cell_size
            y = coord['y'] * self.cell_size
            wormSegmentRect = pygame.Rect(x, y, self.cell_size, self.cell_size)
            pygame.draw.rect(DISPLAYSURF, DARKGREEN, wormSegmentRect)
            wormInnerSegmentRect = pygame.Rect(x + 4, y + 4, \
                                   self.cell_size - 8, self.cell_size - 8)
            pygame.draw.rect(DISPLAYSURF, GREEN, wormInnerSegmentRect)
```

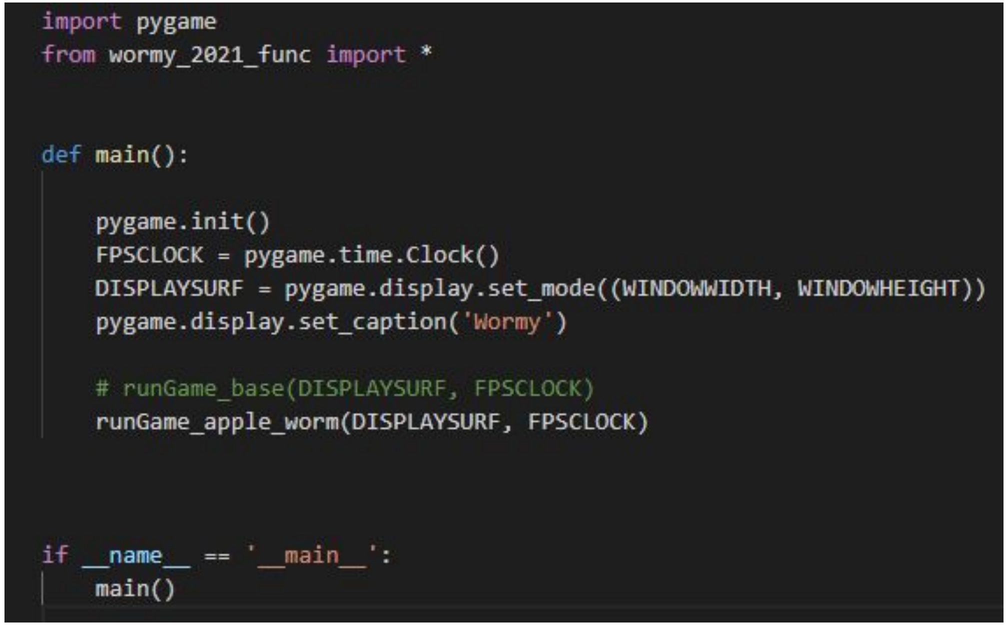


0	1	2	3	4

runGame_apple_worm



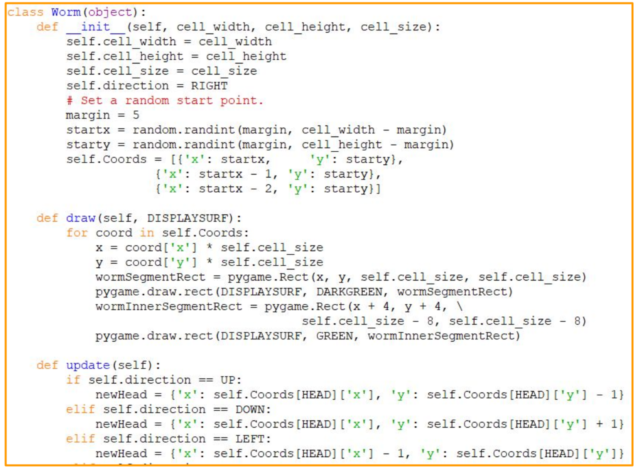
```python
def runGame_apple_worm(DISPLAYSURF, FPSCLOCK):
    score = 0
    apple = Apple(CELLWIDTH, CELLHEIGHT, CELLSIZE)
    worm = Worm(CELLWIDTH, CELLHEIGHT, CELLSIZE)
    while True: # main game loop
        for event in pygame.event.get():
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if event.key == K_LEFT:
                    score -= 1
                elif event.key == K_RIGHT:
                    score += 1
                elif event.key == K_UP:
                    score += 10
                elif event.key == K_DOWN:
                    score -= 10
                else:
                    apple.update()
        DISPLAYSURF.fill(BGCOLOR)
        drawGrid(DISPLAYSURF)
        drawScore(score, DISPLAYSURF)
        apple.draw(DISPLAYSURF)
        worm.draw(DISPLAYSURF)
        pygame.display.update()
        FPSCLOCK.tick(FPS)
```



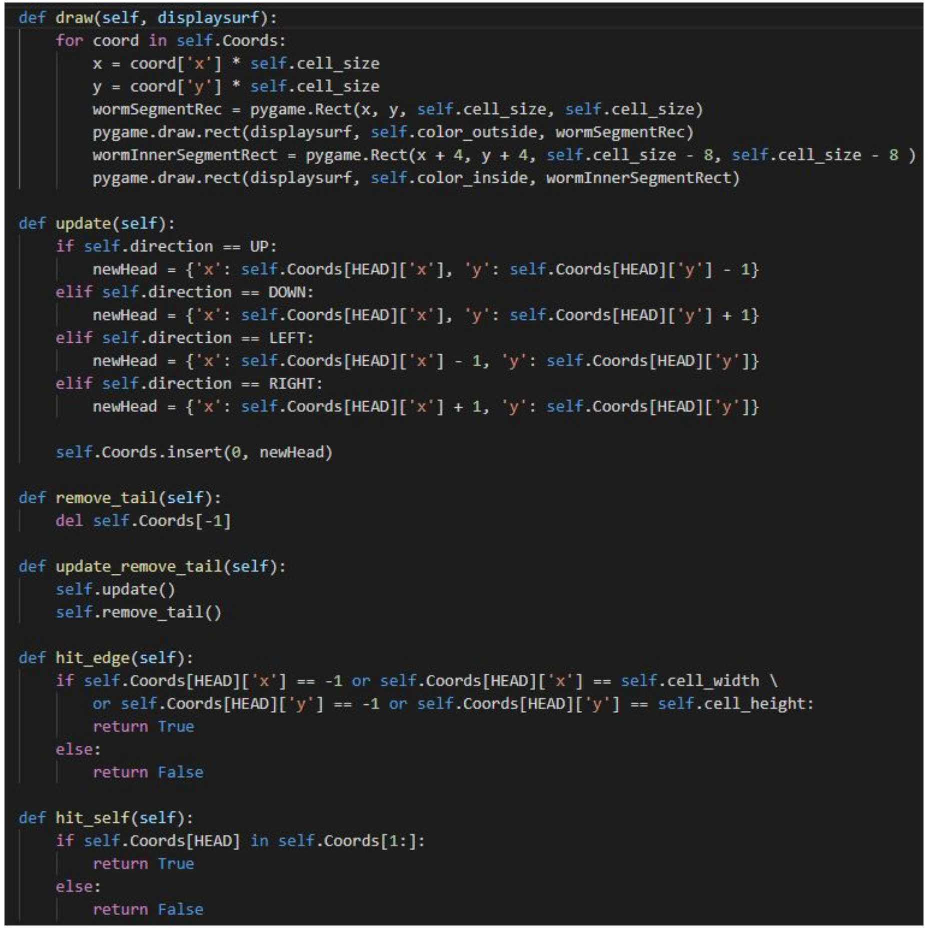
```python
import pygame
from wormy_2021_func import *


def main():

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    pygame.display.set_caption('Wormy')

    # runGame_base(DISPLAYSURF, FPSCLOCK)
    runGame_apple_worm(DISPLAYSURF, FPSCLOCK)



if __name__ == '__main__':
    main()
```

new	0	1	2	3	4

Worm class (cont)

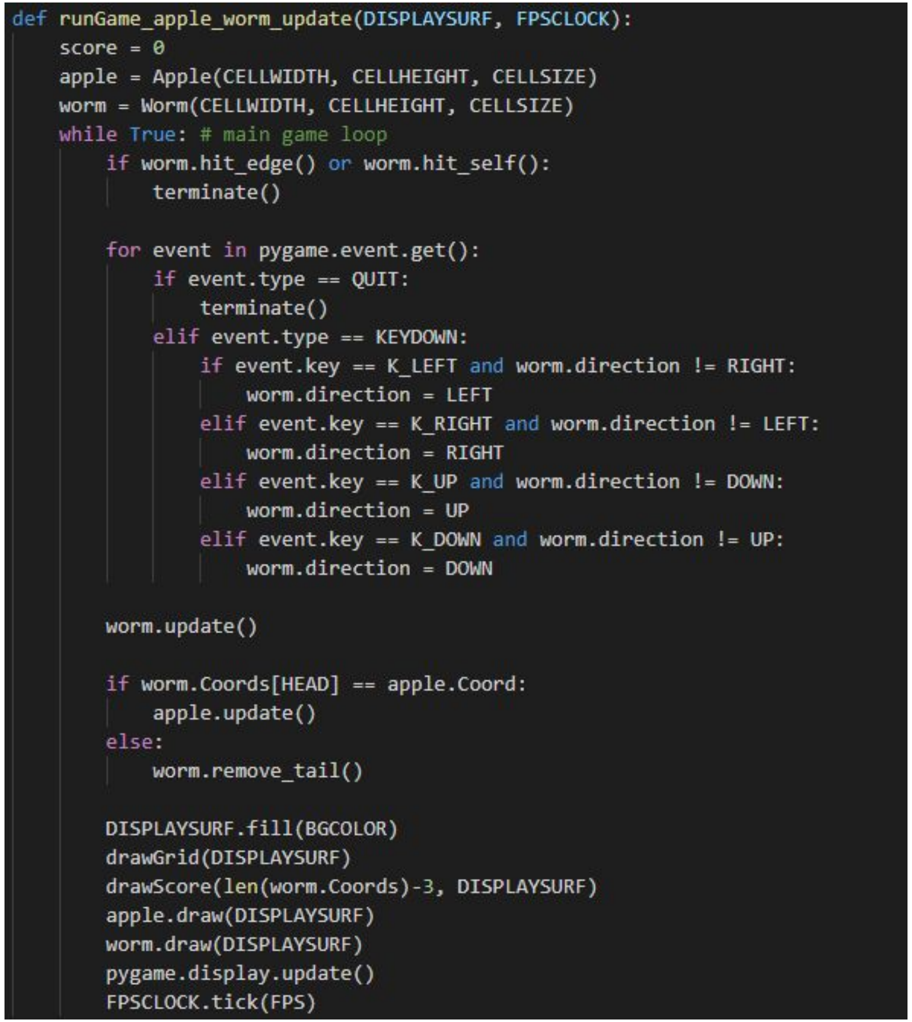


```python
class Worm(object):
    def __init__(self, cell_width, cell_height, cell_size):
        self.cell_width = cell_width
        self.cell_height = cell_height
        self.cell_size = cell_size
        self.direction = RIGHT
        # Set a random start point.
        margin = 5
        startx = random.randint(margin, cell_width - margin)
        starty = random.randint(margin, cell_height - margin)
        self.Coords = [{'x': startx,     'y': starty},
                       {'x': startx - 1, 'y': starty},
                       {'x': startx - 2, 'y': starty}]

    def draw(self, DISPLAYSURF):
        for coord in self.Coords:
            x = coord['x'] * self.cell_size
            y = coord['y'] * self.cell_size
            wormSegmentRect = pygame.Rect(x, y, self.cell_size, self.cell_size)
            pygame.draw.rect(DISPLAYSURF, DARKGREEN, wormSegmentRect)
            wormInnerSegmentRect = pygame.Rect(x + 4, y + 4, \
                                            self.cell_size - 8, self.cell_size - 8)
            pygame.draw.rect(DISPLAYSURF, GREEN, wormInnerSegmentRect)

    def update(self):
        if self.direction == UP:
            newHead = {'x': self.Coords[HEAD]['x'], 'y': self.Coords[HEAD]['y'] - 1}
        elif self.direction == DOWN:
            newHead = {'x': self.Coords[HEAD]['x'], 'y': self.Coords[HEAD]['y'] + 1}
        elif self.direction == LEFT:
            newHead = {'x': self.Coords[HEAD]['x'] - 1, 'y': self.Coords[HEAD]['y']}
```



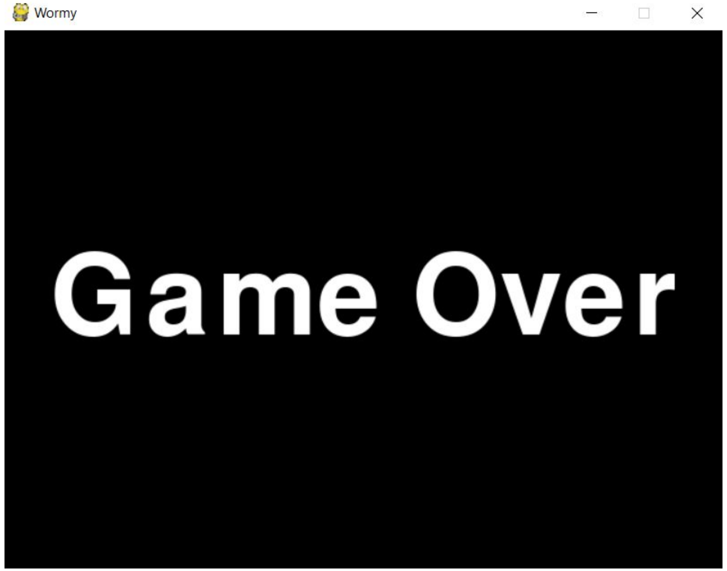
```python
def draw(self, displaysurf):
    for coord in self.Coords:
        x = coord['x'] * self.cell_size
        y = coord['y'] * self.cell_size
        wormSegmentRec = pygame.Rect(x, y, self.cell_size, self.cell_size)
        pygame.draw.rect(displaysurf, self.color_outside, wormSegmentRec)
        wormInnerSegmentRect = pygame.Rect(x + 4, y + 4, self.cell_size - 8, self.cell_size - 8 )
        pygame.draw.rect(displaysurf, self.color_inside, wormInnerSegmentRect)

def update(self):
    if self.direction == UP:
        newHead = {'x': self.Coords[HEAD]['x'], 'y': self.Coords[HEAD]['y'] - 1}
    elif self.direction == DOWN:
        newHead = {'x': self.Coords[HEAD]['x'], 'y': self.Coords[HEAD]['y'] + 1}
    elif self.direction == LEFT:
        newHead = {'x': self.Coords[HEAD]['x'] - 1, 'y': self.Coords[HEAD]['y']}
    elif self.direction == RIGHT:
        newHead = {'x': self.Coords[HEAD]['x'] + 1, 'y': self.Coords[HEAD]['y']}

    self.Coords.insert(0, newHead)

def remove_tail(self):
    del self.Coords[-1]

def update_remove_tail(self):
    self.update()
    self.remove_tail()

def hit_edge(self):
    if self.Coords[HEAD]['x'] == -1 or self.Coords[HEAD]['x'] == self.cell_width \
        or self.Coords[HEAD]['y'] == -1 or self.Coords[HEAD]['y'] == self.cell_height:
        return True
    else:
        return False

def hit_self(self):
    if self.Coords[HEAD] in self.Coords[1:]:
        return True
    else:
        return False
```

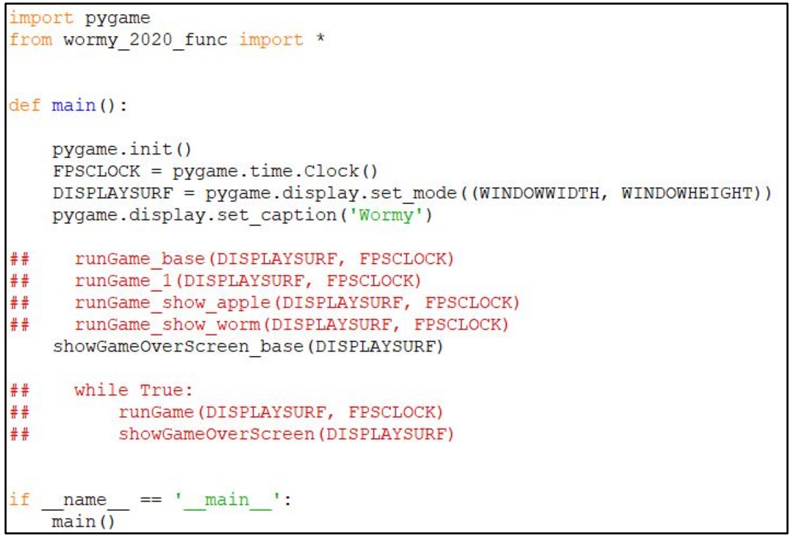
runGame_apple_worm_update



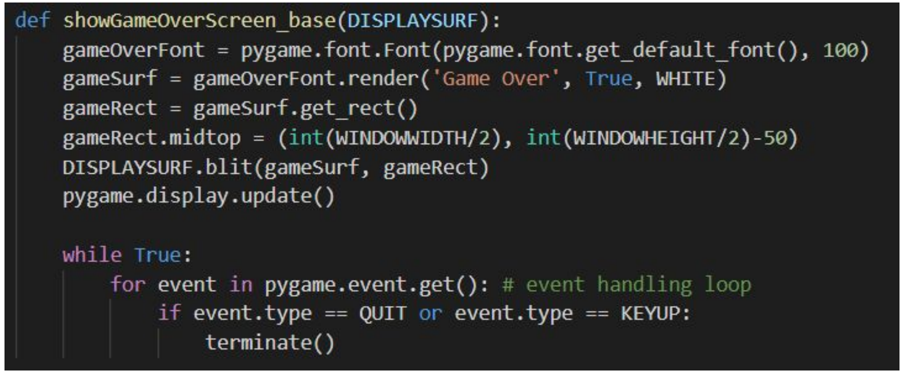
```
def runGame_apple_worm_update(DISPLAYSURF, FPSCLOCK):
    score = 0
    apple = Apple(CELLWIDTH, CELLHEIGHT, CELLSIZE)
    worm = Worm(CELLWIDTH, CELLHEIGHT, CELLSIZE)
    while True: # main game loop
        if worm.hit_edge() or worm.hit_self():
            terminate()

        for event in pygame.event.get():
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if event.key == K_LEFT and worm.direction != RIGHT:
                    worm.direction = LEFT
                elif event.key == K_RIGHT and worm.direction != LEFT:
                    worm.direction = RIGHT
                elif event.key == K_UP and worm.direction != DOWN:
                    worm.direction = UP
                elif event.key == K_DOWN and worm.direction != UP:
                    worm.direction = DOWN

        worm.update()

        if worm.Coords[HEAD] == apple.Coord:
            apple.update()
        else:
            worm.remove_tail()

        DISPLAYSURF.fill(BGCOLOR)
        drawGrid(DISPLAYSURF)
        drawScore(len(worm.Coords)-3, DISPLAYSURF)
        apple.draw(DISPLAYSURF)
        worm.draw(DISPLAYSURF)
        pygame.display.update()
        FPSCLOCK.tick(FPS)
```

show game over screen



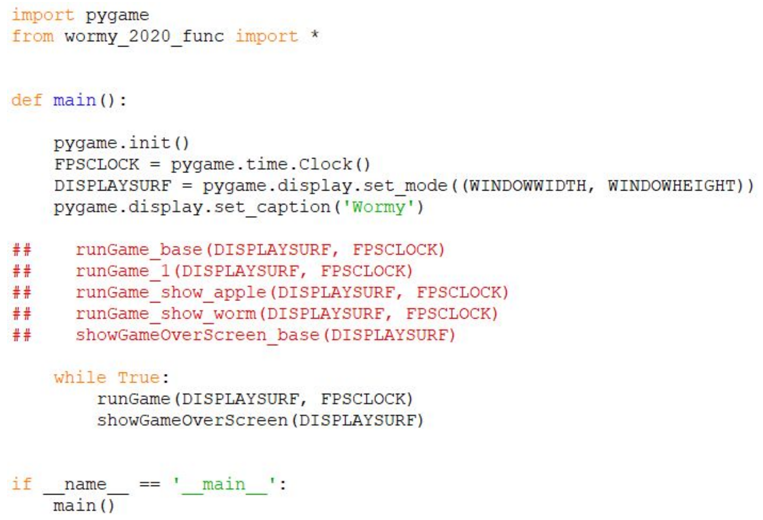


```
import pygame
from wormy_2020_func import *


def main():

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    pygame.display.set_caption('Wormy')

##    runGame_base(DISPLAYSURF, FPSCLOCK)
##    runGame_1(DISPLAYSURF, FPSCLOCK)
##    runGame_show_apple(DISPLAYSURF, FPSCLOCK)
##    runGame_show_worm(DISPLAYSURF, FPSCLOCK)
    showGameOverScreen_base(DISPLAYSURF)

##    while True:
##        runGame(DISPLAYSURF, FPSCLOCK)
##        showGameOverScreen(DISPLAYSURF)


if __name__ == '__main__':
    main()
```

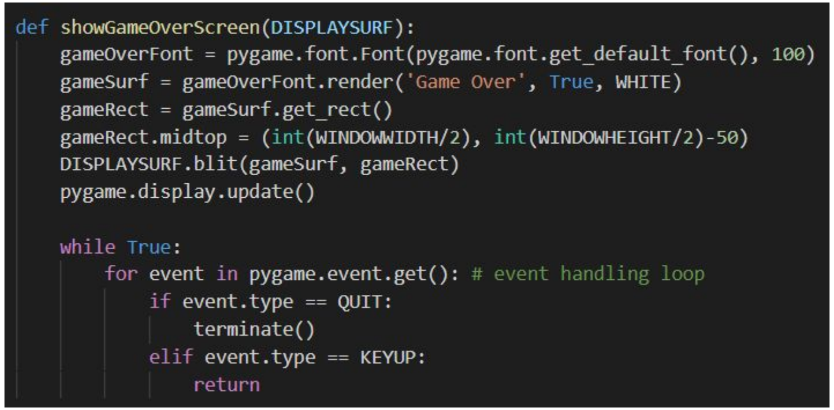


```
def showGameOverScreen_base(DISPLAYSURF):
    gameOverFont = pygame.font.Font(pygame.font.get_default_font(), 100)
    gameSurf = gameOverFont.render('Game Over', True, WHITE)
    gameRect = gameSurf.get_rect()
    gameRect.midtop = (int(WINDOWWIDTH/2), int(WINDOWHEIGHT/2)-50)
    DISPLAYSURF.blit(gameSurf, gameRect)
    pygame.display.update()

    while True:
        for event in pygame.event.get(): # event handling loop
            if event.type == QUIT or event.type == KEYUP:
                terminate()
```

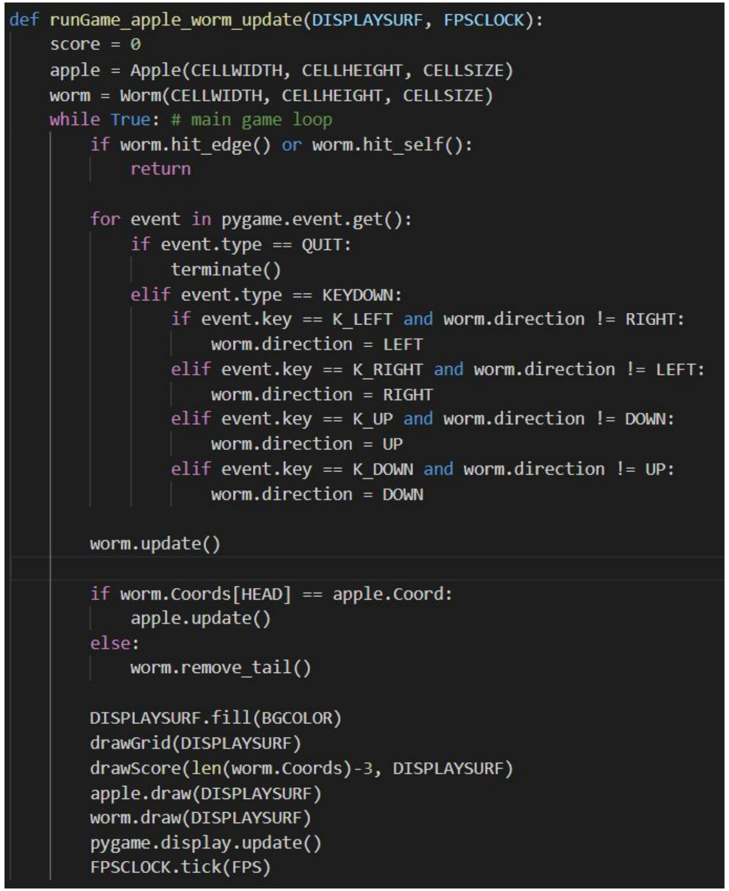
Put them together



```
import pygame
from wormy_2020_func import *


def main():

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    pygame.display.set_caption('Wormy')

##      runGame_base(DISPLAYSURF, FPSCLOCK)
##      runGame_1(DISPLAYSURF, FPSCLOCK)
##      runGame_show_apple(DISPLAYSURF, FPSCLOCK)
##      runGame_show_worm(DISPLAYSURF, FPSCLOCK)
##      showGameOverScreen_base(DISPLAYSURF)

    while True:
        runGame(DISPLAYSURF, FPSCLOCK)
        showGameOverScreen(DISPLAYSURF)


if __name__ == '__main__':
    main()
```

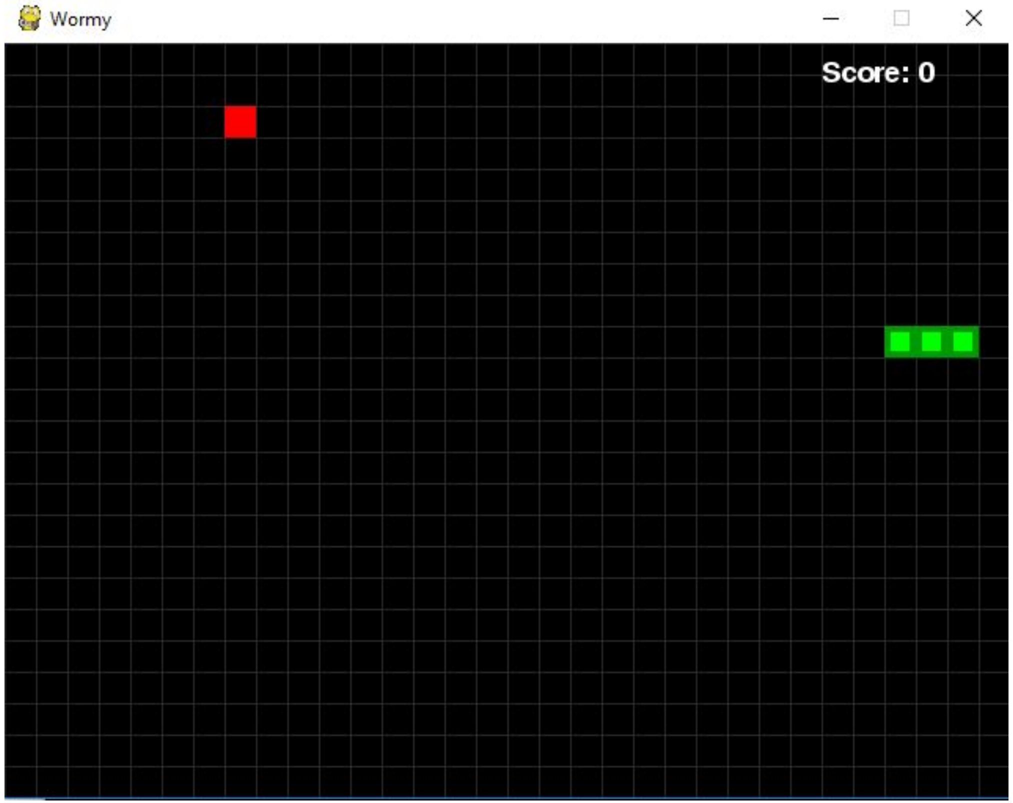


```
def showGameOverScreen(DISPLAYSURF):
    gameOverFont = pygame.font.Font(pygame.font.get_default_font(), 100)
    gameSurf = gameOverFont.render('Game Over', True, WHITE)
    gameRect = gameSurf.get_rect()
    gameRect.midtop = (int(WINDOWWIDTH/2), int(WINDOWHEIGHT/2)-50)
    DISPLAYSURF.blit(gameSurf, gameRect)
    pygame.display.update()

    while True:
        for event in pygame.event.get(): # event handling loop
            if event.type == QUIT:
                terminate()
            elif event.type == KEYUP:
                return
```

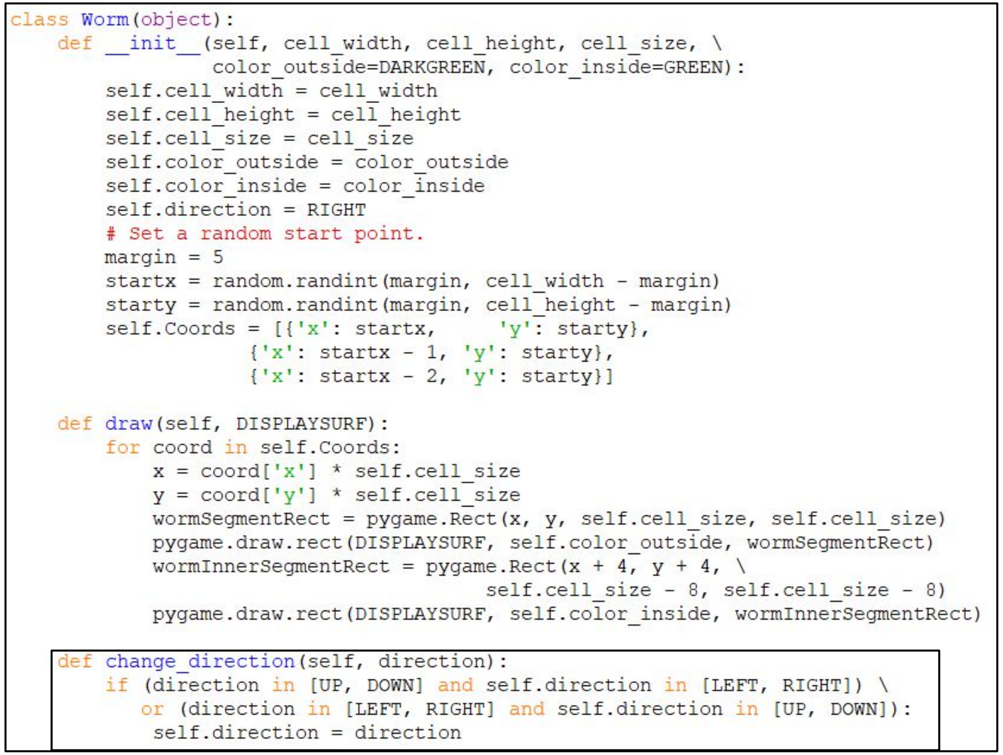


```
def runGame_apple_worm_update(DISPLAYSURF, FPSCLOCK):
    score = 0
    apple = Apple(CELLWIDTH, CELLHEIGHT, CELLSIZE)
    worm = Worm(CELLWIDTH, CELLHEIGHT, CELLSIZE)
    while True: # main game loop
        if worm.hit_edge() or worm.hit_self():
            return

        for event in pygame.event.get():
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if event.key == K_LEFT and worm.direction != RIGHT:
                    worm.direction = LEFT
                elif event.key == K_RIGHT and worm.direction != LEFT:
                    worm.direction = RIGHT
                elif event.key == K_UP and worm.direction != DOWN:
                    worm.direction = UP
                elif event.key == K_DOWN and worm.direction != UP:
                    worm.direction = DOWN

        worm.update()

        if worm.Coords[HEAD] == apple.Coord:
            apple.update()
        else:
            worm.remove_tail()

        DISPLAYSURF.fill(BGCOLOR)
        drawGrid(DISPLAYSURF)
        drawScore(len(worm.Coords)-3, DISPLAYSURF)
        apple.draw(DISPLAYSURF)
        worm.draw(DISPLAYSURF)
        pygame.display.update()
        FPSCLOCK.tick(FPS)
```

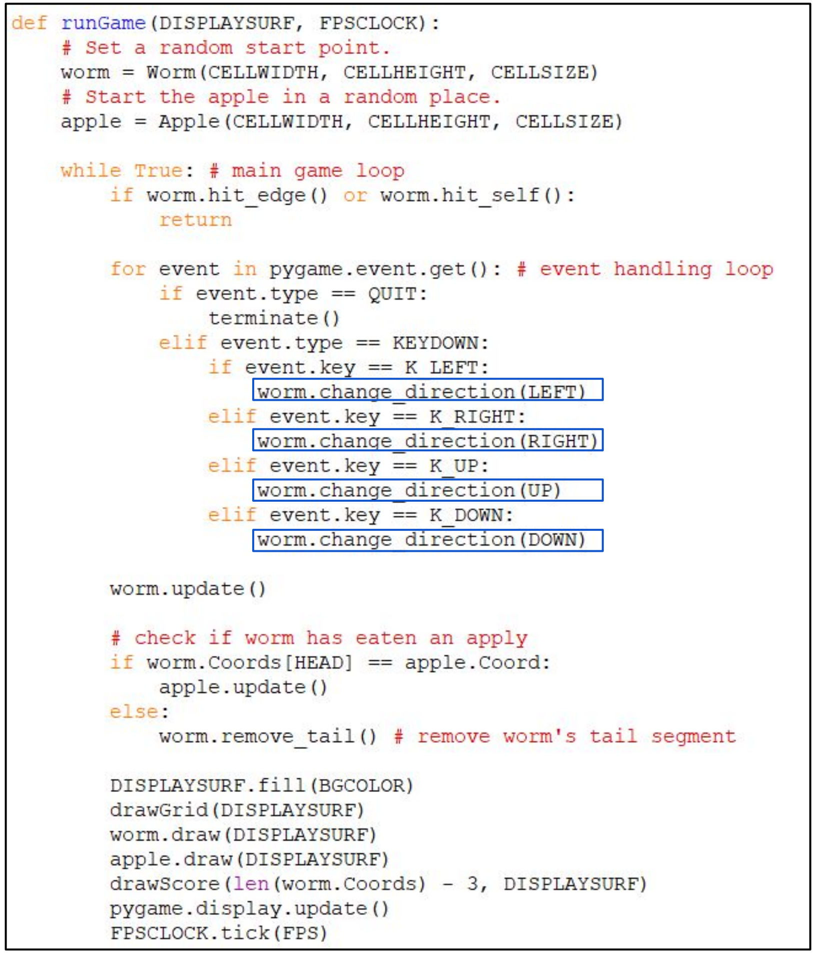
Finally



Optimize worm class

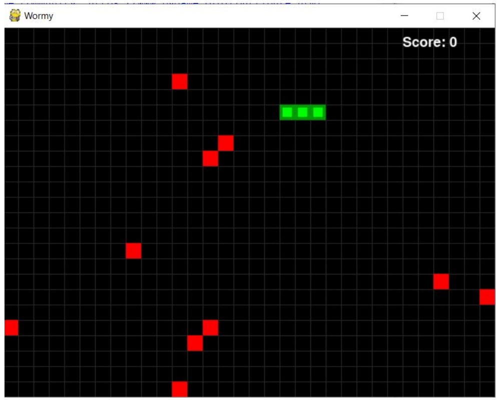


```python
class Worm(object):
    def __init__(self, cell_width, cell_height, cell_size, \
                 color_outside=DARKGREEN, color_inside=GREEN):
        self.cell_width = cell_width
        self.cell_height = cell_height
        self.cell_size = cell_size
        self.color_outside = color_outside
        self.color_inside = color_inside
        self.direction = RIGHT
        # Set a random start point.
        margin = 5
        startx = random.randint(margin, cell_width - margin)
        starty = random.randint(margin, cell_height - margin)
        self.Coords = [{'x': startx,     'y': starty},
                       {'x': startx - 1, 'y': starty},
                       {'x': startx - 2, 'y': starty}]

    def draw(self, DISPLAYSURF):
        for coord in self.Coords:
            x = coord['x'] * self.cell_size
            y = coord['y'] * self.cell_size
            wormSegmentRect = pygame.Rect(x, y, self.cell_size, self.cell_size)
            pygame.draw.rect(DISPLAYSURF, self.color_outside, wormSegmentRect)
            wormInnerSegmentRect = pygame.Rect(x + 4, y + 4, \
                                        self.cell_size - 8, self.cell_size - 8)
            pygame.draw.rect(DISPLAYSURF, self.color_inside, wormInnerSegmentRect)

    def change_direction(self, direction):
        if (direction in [UP, DOWN] and self.direction in [LEFT, RIGHT]) \
           or (direction in [LEFT, RIGHT] and self.direction in [UP, DOWN]):
            self.direction = direction
```



```python
def runGame(DISPLAYSURF, FPSCLOCK):
    # Set a random start point.
    worm = Worm(CELLWIDTH, CELLHEIGHT, CELLSIZE)
    # Start the apple in a random place.
    apple = Apple(CELLWIDTH, CELLHEIGHT, CELLSIZE)

    while True: # main game loop
        if worm.hit_edge() or worm.hit_self():
            return

        for event in pygame.event.get(): # event handling loop
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if event.key == K_LEFT:
                    worm.change_direction(LEFT)
                elif event.key == K_RIGHT:
                    worm.change_direction(RIGHT)
                elif event.key == K_UP:
                    worm.change_direction(UP)
                elif event.key == K_DOWN:
                    worm.change_direction(DOWN)

        worm.update()

        # check if worm has eaten an apply
        if worm.Coords[HEAD] == apple.Coord:
            apple.update()
        else:
            worm.remove_tail() # remove worm's tail segment

        DISPLAYSURF.fill(BGCOLOR)
        drawGrid(DISPLAYSURF)
        worm.draw(DISPLAYSURF)
        apple.draw(DISPLAYSURF)
        drawScore(len(worm.Coords) - 3, DISPLAYSURF)
        pygame.display.update()
        FPSCLOCK.tick(FPS)
```

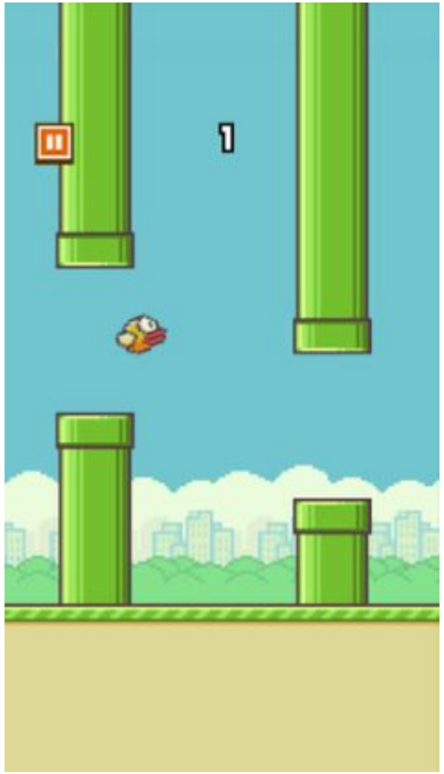
Add multiple apples

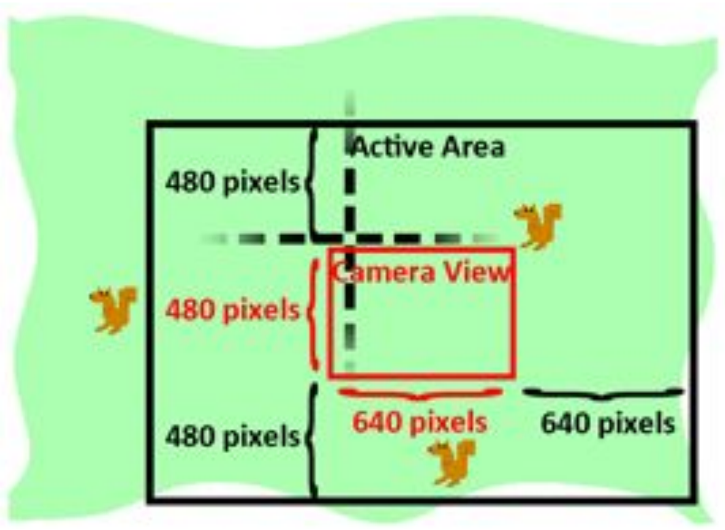


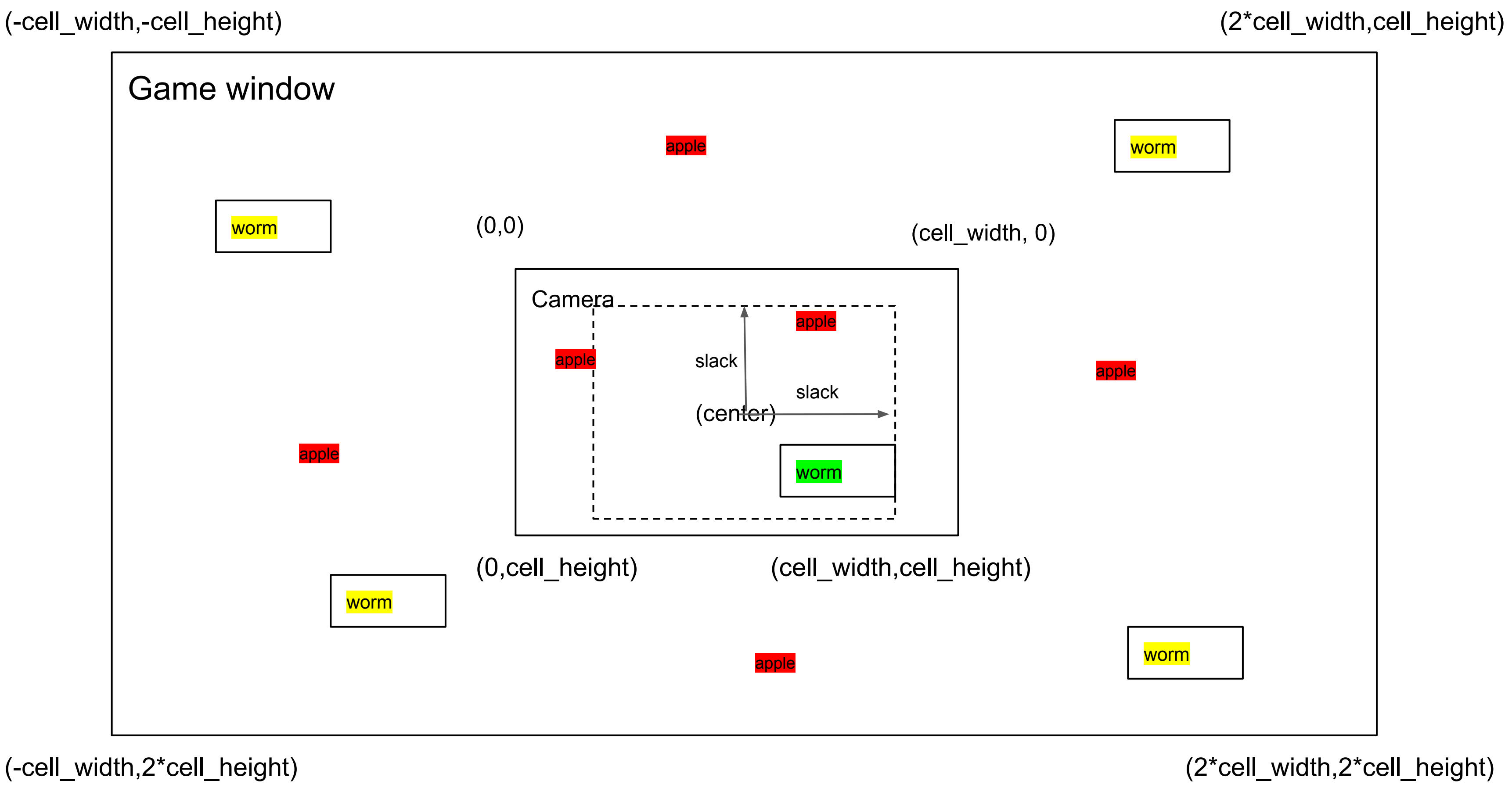
```
def main():

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    pygame.display.set_caption('Wormy')

##    runGame_base(DISPLAYSURF, FPSCLOCK)
##    runGame_1(DISPLAYSURF, FPSCLOCK)
##    runGame_show_apple(DISPLAYSURF, FPSCLOCK)
##    runGame_show_worm(DISPLAYSURF, FPSCLOCK)
##    showGameOverScreen_base(DISPLAYSURF)

    while True:
##        runGame(DISPLAYSURF, FPSCLOCK)
        runGame_multi_apple(DISPLAYSURF, FPSCLOCK, 10)
##        runGame_camera_move(DISPLAYSURF, FPSCLOCK, 100)
##        runGame_camera_move_multipe_apple_worm(DISPLAYSURF, FPSCLOCK, 100)
        showGameOverScreen(DISPLAYSURF)
```

```
def runGame_multi_apple(DISPLAYSURF, FPSCLOCK, num_apple):
    worm = Worm(CELLWIDTH, CELLHEIGHT, CELLSIZE)
    apples = [Apple(CELLWIDTH, CELLHEIGHT, CELLSIZE) for i in range(num_apple)]

    while True: # main game loop
        if worm.hit_edge() or worm.hit_self():
            return

        for event in pygame.event.get(): # event handling loop
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if event.key == K_LEFT:
                    worm.change_direction(LEFT)
                elif event.key == K_RIGHT:
                    worm.change_direction(RIGHT)
                elif event.key == K_UP:
                    worm.change_direction(UP)
                elif event.key == K_DOWN:
                    worm.change_direction(DOWN)

        worm.update()

        # check if worm has eaten an apply
        apple_bite = False
        for i in range(len(apples)-1, -1, -1):
            apple = apples[i]
            if worm.Coords[HEAD] == apple.Coord:
                del apples[i]
                apple_bite = True
                break
        if not apple_bite:
            worm.remove_tail()

        DISPLAYSURF.fill(BGCOLOR)
        drawGrid(DISPLAYSURF)
        worm.draw(DISPLAYSURF)

        for apple in apples:
            apple.draw(DISPLAYSURF)
```

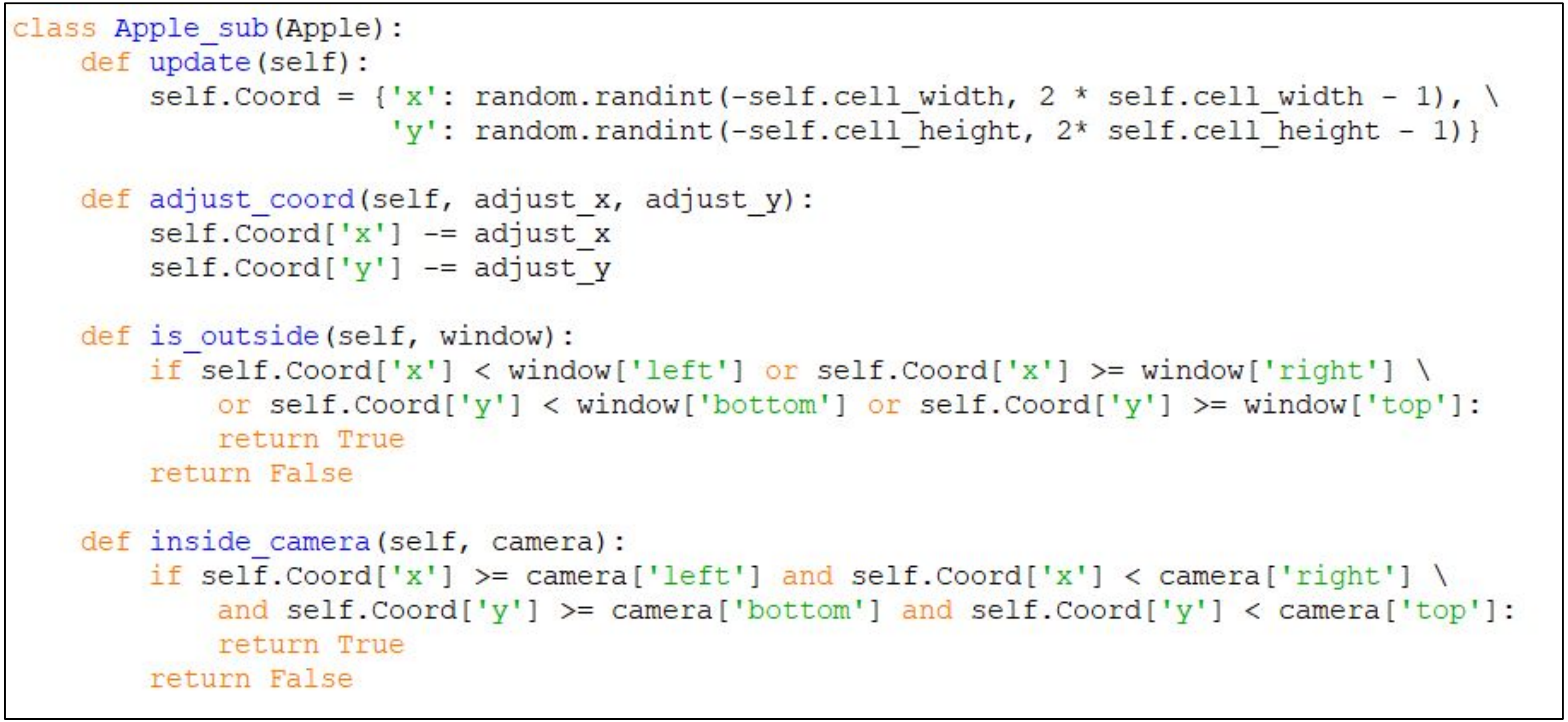
Moving camera





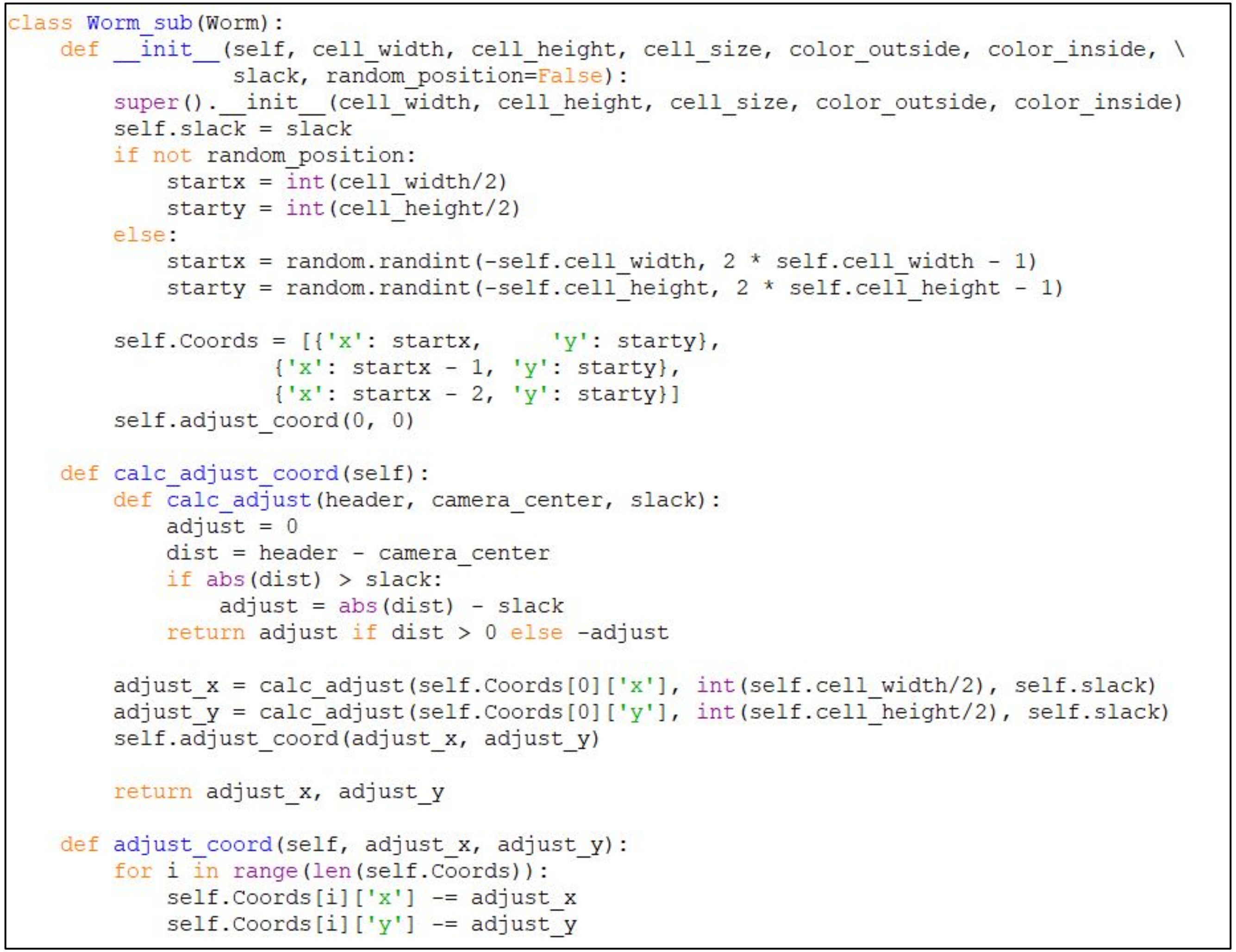

(-cell_width,-cell_height)
(2*cell_width,cell_height)
Game window
apple
worm
worm
(0,0)
(cell_width, 0)
Camera
apple
apple
slack
slack
(center)
apple
apple
worm
apple
(0,cell_height)
(cell_width,cell_height)
worm
apple
worm
(-cell_width,2*cell_height)
(2*cell_width,2*cell_height)

Apple_sub class



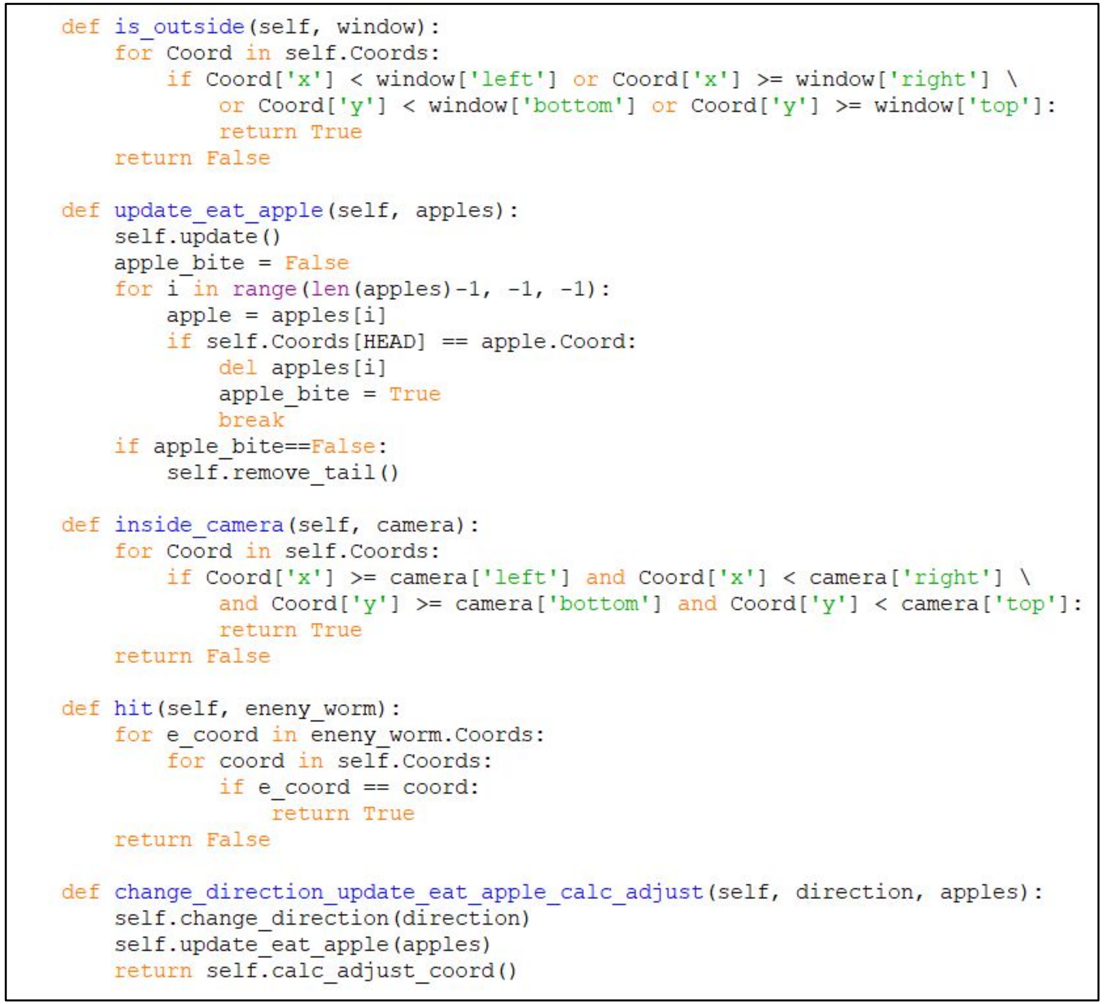
```python
class Apple_sub(Apple):
    def update(self):
        self.Coord = {'x': random.randint(-self.cell_width, 2 * self.cell_width - 1), \
                      'y': random.randint(-self.cell_height, 2* self.cell_height - 1)}

    def adjust_coord(self, adjust_x, adjust_y):
        self.Coord['x'] -= adjust_x
        self.Coord['y'] -= adjust_y

    def is_outside(self, window):
        if self.Coord['x'] < window['left'] or self.Coord['x'] >= window['right'] \
            or self.Coord['y'] < window['bottom'] or self.Coord['y'] >= window['top']:
            return True
        return False

    def inside_camera(self, camera):
        if self.Coord['x'] >= camera['left'] and self.Coord['x'] < camera['right'] \
            and self.Coord['y'] >= camera['bottom'] and self.Coord['y'] < camera['top']:
            return True
        return False
```

Worm_sub class



```
class Worm_sub(Worm):
    def __init__(self, cell_width, cell_height, cell_size, color_outside, color_inside, \
                 slack, random_position=False):
        super().__init__(cell_width, cell_height, cell_size, color_outside, color_inside)
        self.slack = slack
        if not random_position:
            startx = int(cell_width/2)
            starty = int(cell_height/2)
        else:
            startx = random.randint(-self.cell_width, 2 * self.cell_width - 1)
            starty = random.randint(-self.cell_height, 2 * self.cell_height - 1)

        self.Coords = [{'x': startx,     'y': starty},
                    {'x': startx - 1, 'y': starty},
                    {'x': startx - 2, 'y': starty}]
        self.adjust_coord(0, 0)

    def calc_adjust_coord(self):
        def calc_adjust(header, camera_center, slack):
            adjust = 0
            dist = header - camera_center
            if abs(dist) > slack:
                adjust = abs(dist) - slack
            return adjust if dist > 0 else -adjust

        adjust_x = calc_adjust(self.Coords[0]['x'], int(self.cell_width/2), self.slack)
        adjust_y = calc_adjust(self.Coords[0]['y'], int(self.cell_height/2), self.slack)
        self.adjust_coord(adjust_x, adjust_y)

        return adjust_x, adjust_y

    def adjust_coord(self, adjust_x, adjust_y):
        for i in range(len(self.Coords)):
            self.Coords[i]['x'] -= adjust_x
            self.Coords[i]['y'] -= adjust_y
```

Worm_sub class (cont)



```
    def is_outside(self, window):
        for Coord in self.Coords:
            if Coord['x'] < window['left'] or Coord['x'] >= window['right'] \
                or Coord['y'] < window['bottom'] or Coord['y'] >= window['top']:
                return True
        return False

    def update_eat_apple(self, apples):
        self.update()
        apple_bite = False
        for i in range(len(apples)-1, -1, -1):
            apple = apples[i]
            if self.Coords[HEAD] == apple.Coord:
                del apples[i]
                apple_bite = True
                break
        if apple_bite==False:
            self.remove_tail()

    def inside_camera(self, camera):
        for Coord in self.Coords:
            if Coord['x'] >= camera['left'] and Coord['x'] < camera['right'] \
                and Coord['y'] >= camera['bottom'] and Coord['y'] < camera['top']:
                return True
        return False

    def hit(self, eneny_worm):
        for e_coord in eneny_worm.Coords:
            for coord in self.Coords:
                if e_coord == coord:
                    return True
        return False

    def change_direction_update_eat_apple_calc_adjust(self, direction, apples):
        self.change_direction(direction)
        self.update_eat_apple(apples)
        return self.calc_adjust_coord()
```

runGame_camera_move

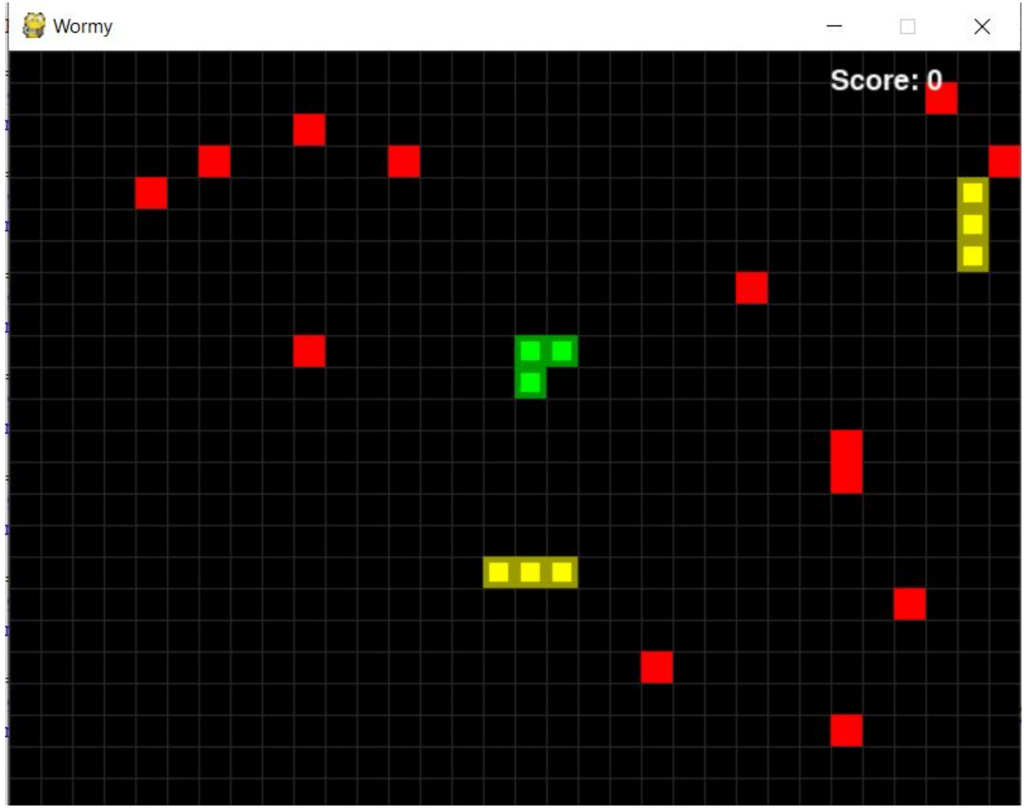
```
def runGame_camera_move(DISPLAYSURF, FPSCLOCK, num_apple):
    slack = 8
    worm = Worm_sub(CELLWIDTH, CELLHEIGHT, CELLSIZE, DARKGREEN, GREEN, slack)
    apples = [Apple_sub(CELLWIDTH, CELLHEIGHT, CELLSIZE) for _ in range(num_apple)]
    window = {'left': -CELLWIDTH, 'right': 2 * CELLWIDTH,\
              'bottom': -CELLWIDTH, 'top': 2 * CELLHEIGHT }
    camera = {'left': 0, 'right': CELLWIDTH, \
              'bottom': 0, 'top': CELLHEIGHT }

    while True: # main game loop
        adjust_x, adjust_y = 0, 0
        for i in range(len(apples)-1, -1, -1):
            if apples[i].is_outside(window):
                del apples[i]
        while len(apples) < num_apple:
            apple = Apple_sub(CELLWIDTH, CELLHEIGHT, CELLSIZE)
            if not apple.inside_camera(camera):
                apples.append(apple)

        for event in pygame.event.get(): # event handling loop
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if event.key == K_LEFT:
                    adjust_x, adjust_y = worm.change_direction_update_eat_apple_calc_adjust(LEFT, apples)
                elif event.key == K_RIGHT:
                    adjust_x, adjust_y = worm.change_direction_update_eat_apple_calc_adjust(RIGHT, apples)
                elif event.key == K_UP:
                    adjust_x, adjust_y = worm.change_direction_update_eat_apple_calc_adjust(UP, apples)
                elif event.key == K_DOWN:
                    adjust_x, adjust_y = worm.change_direction_update_eat_apple_calc_adjust(DOWN, apples)

        DISPLAYSURF.fill(BGCOLOR)
        drawGrid(DISPLAYSURF)
        worm.draw(DISPLAYSURF)

        for apple in apples:
            apple.adjust_coord(adjust_x, adjust_y)
            apple.draw(DISPLAYSURF)

        drawScore(len(worm.Coords) - 3, DISPLAYSURF)
        pygame.display.update()
        FPSCLOCK.tick(FPS)
```

```
def runGame_multi_apple(DISPLAYSURF, FPSCLOCK, num_apple):
    worm = Worm(CELLWIDTH, CELLHEIGHT, CELLSIZE)
    apples = [Apple(CELLWIDTH, CELLHEIGHT, CELLSIZE) for i in range(num_apple)]

    while True: # main game loop
        if worm.hit_edge() or worm.hit_self():
            return

        for event in pygame.event.get(): # event handling loop
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if event.key == K_LEFT:
                    worm.change_direction(LEFT)
                elif event.key == K_RIGHT:
                    worm.change_direction(RIGHT)
                elif event.key == K_UP:
                    worm.change_direction(UP)
                elif event.key == K_DOWN:
                    worm.change_direction(DOWN)

        worm.update()

        # check if worm has eaten an apply
        apple_bite = False
        for i in range(len(apples)-1, -1, -1):
            apple = apples[i]
            if worm.Coords[HEAD] == apple.Coord:
                del apples[i]
                apple_bite = True
                break
        if not apple_bite:
            worm.remove_tail()

        DISPLAYSURF.fill(BGCOLOR)
        drawGrid(DISPLAYSURF)
        worm.draw(DISPLAYSURF)

        for apple in apples:
            apple.draw(DISPLAYSURF)
```

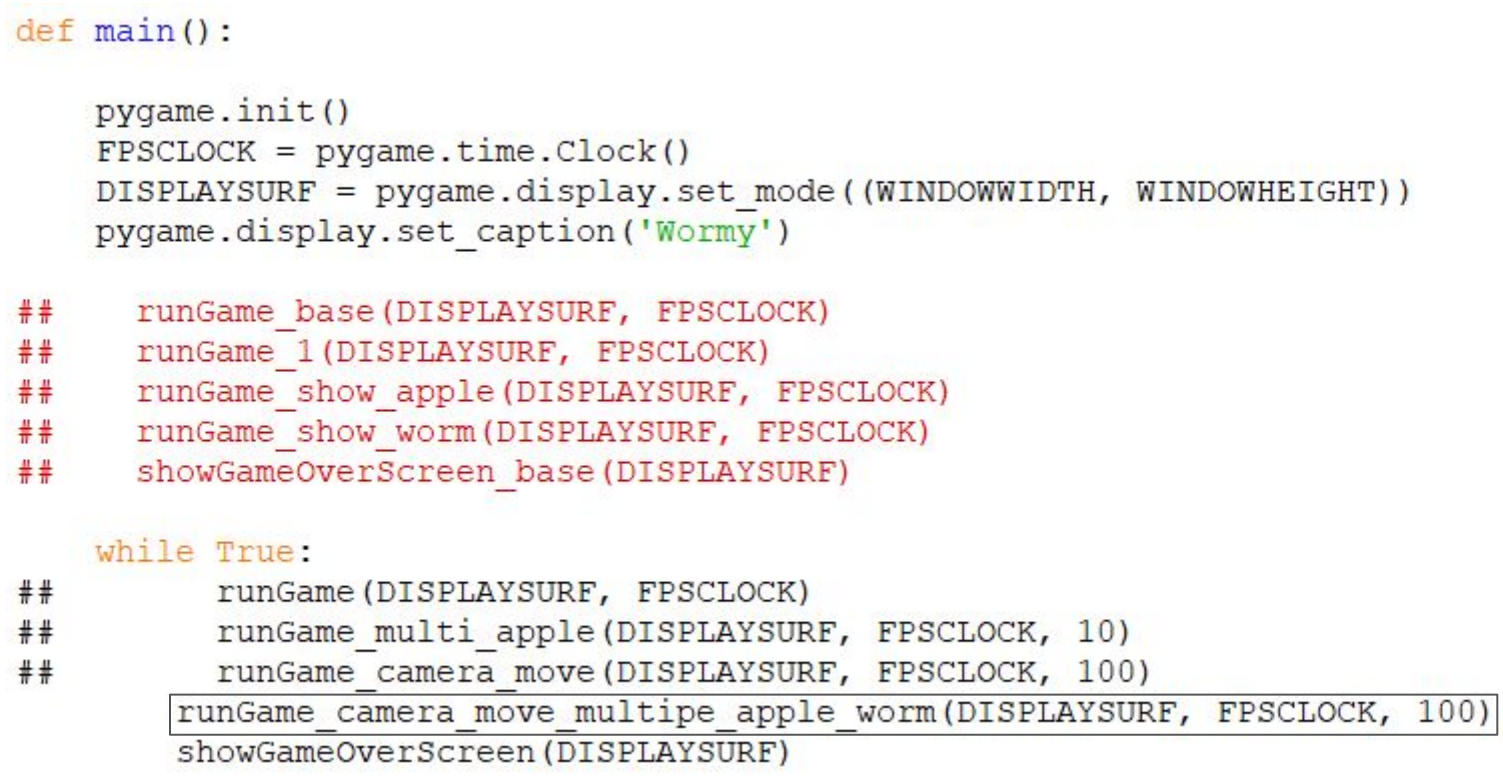
Main function

```
def main():

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    pygame.display.set_caption('Wormy')

##    runGame_base(DISPLAYSURF, FPSCLOCK)
##    runGame_1(DISPLAYSURF, FPSCLOCK)
##    runGame_show_apple(DISPLAYSURF, FPSCLOCK)
##    runGame_show_worm(DISPLAYSURF, FPSCLOCK)
##    showGameOverScreen_base(DISPLAYSURF)

    while True:
##        runGame(DISPLAYSURF, FPSCLOCK)
##        runGame_multi_apple(DISPLAYSURF, FPSCLOCK, 10)
        runGame_camera_move(DISPLAYSURF, FPSCLOCK, 100)
##        runGame_camera_move_multipe_apple_worm(DISPLAYSURF, FPSCLOCK, 100)
        showGameOverScreen(DISPLAYSURF)


if __name__ == '__main__':
    main()
```

multiple_apple_worm_moving_camera



Main function

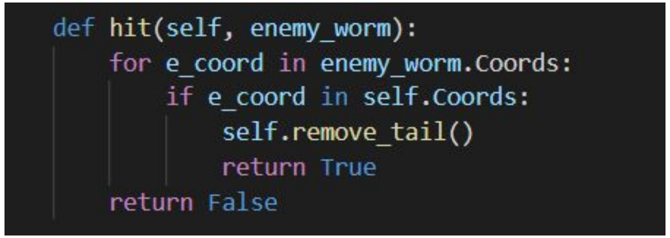


```
def main():

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    pygame.display.set_caption('Wormy')

##    runGame_base(DISPLAYSURF, FPSCLOCK)
##    runGame_1(DISPLAYSURF, FPSCLOCK)
##    runGame_show_apple(DISPLAYSURF, FPSCLOCK)
##    runGame_show_worm(DISPLAYSURF, FPSCLOCK)
##    showGameOverScreen_base(DISPLAYSURF)

    while True:
##        runGame(DISPLAYSURF, FPSCLOCK)
##        runGame_multi_apple(DISPLAYSURF, FPSCLOCK, 10)
##        runGame_camera_move(DISPLAYSURF, FPSCLOCK, 100)
        runGame_camera_move_multipe_apple_worm(DISPLAYSURF, FPSCLOCK, 100)
        showGameOverScreen(DISPLAYSURF)
```

```python
def runGame_camera_move_multipe_apple_worm(DISPLAYSURF, FPSCLOCK, num_apple):
    # Set a random start point.
    slack = 8
    num_worm = 20
    count = 0
    worm = Worm_sub(CELLWIDTH, CELLHEIGHT, CELLSIZE, DARKGREEN, GREEN, slack)
    enemy_worms = [Worm_sub(CELLWIDTH, CELLHEIGHT, CELLSIZE, DARKYELLOW, YELLOW, slack, True) for _ in range(num_worm)]
    apples = [Apple_sub(CELLWIDTH, CELLHEIGHT, CELLSIZE) for _ in range(num_apple)]

    window = {'left': -CELLWIDTH, 'right': 2 * CELLWIDTH,\
              'bottom': -CELLWIDTH, 'top': 2 * CELLHEIGHT }
    camera = {'left': 0, 'right': CELLWIDTH, \
              'bottom': 0, 'top': CELLHEIGHT }

    while True: # main game loop
        count += 1
        adjust_x, adjust_y = 0, 0

        for i in range(len(apples)-1, -1, -1):
            if apples[i].is_outside(window):
                del apples[i]
        while len(apples) < num_apple:
            apple = Apple_sub(CELLWIDTH, CELLHEIGHT, CELLSIZE)
            if not apple.inside_camera(camera):
                apples.append(apple)

        for i in range(len(enemy_worms)-1, -1, -1):
            if enemy_worms[i].is_outside(window):
                del enemy_worms[i]
        while len(enemy_worms) < num_worm:
            w = Worm_sub(CELLWIDTH, CELLHEIGHT, CELLSIZE, DARKYELLOW, YELLOW, slack, True)
            if not w.inside_camera(camera):
                enemy_worms.append(w)
        for event in pygame.event.get(): # event handling loop
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if event.key == K_LEFT:
                    adjust_x, adjust_y = worm.change_direction_update_eat_apple_calc_adjust(LEFT, apples)
                elif event.key == K_RIGHT:
                    adjust_x, adjust_y = worm.change_direction_update_eat_apple_calc_adjust(RIGHT, apples)
                elif event.key == K_UP:
                    adjust_x, adjust_y = worm.change_direction_update_eat_apple_calc_adjust(UP, apples)
                elif event.key == K_DOWN:
                    adjust_x, adjust_y = worm.change_direction_update_eat_apple_calc_adjust(DOWN, apples)

        DISPLAYSURF.fill(BGCOLOR)
        drawGrid(DISPLAYSURF)
        worm.draw(DISPLAYSURF)

        for apple in apples:
            apple.adjust_coord(adjust_x, adjust_y)
            apple.draw(DISPLAYSURF)

        for w in enemy_worms:
            if worm.hit(w):
                return
            if not count % 10:
                w.change_direction(random.choice([LEFT, RIGHT, UP, DOWN]))
            w.update_remove_tail()
            w.adjust_coord(adjust_x, adjust_y)
            w.draw(DISPLAYSURF)

        drawScore(len(worm.Coords) - 3, DISPLAYSURF)
        pygame.display.update()
        FPSCLOCK.tick(FPS)
```

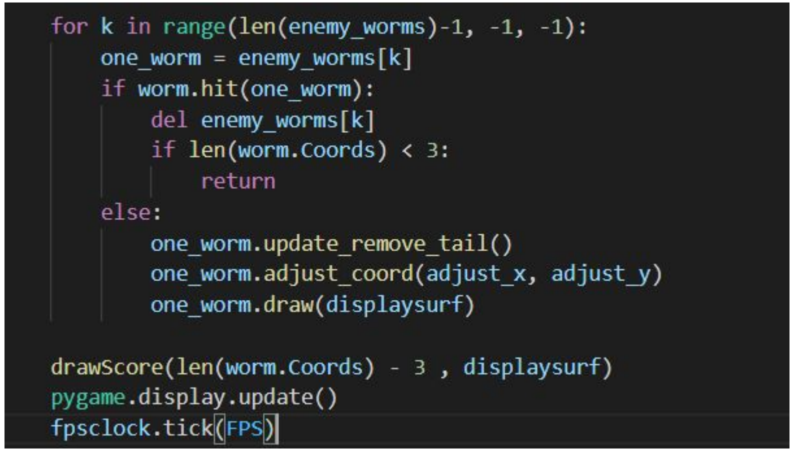
Player worm endures the enemy worm hit

Worm_sub class



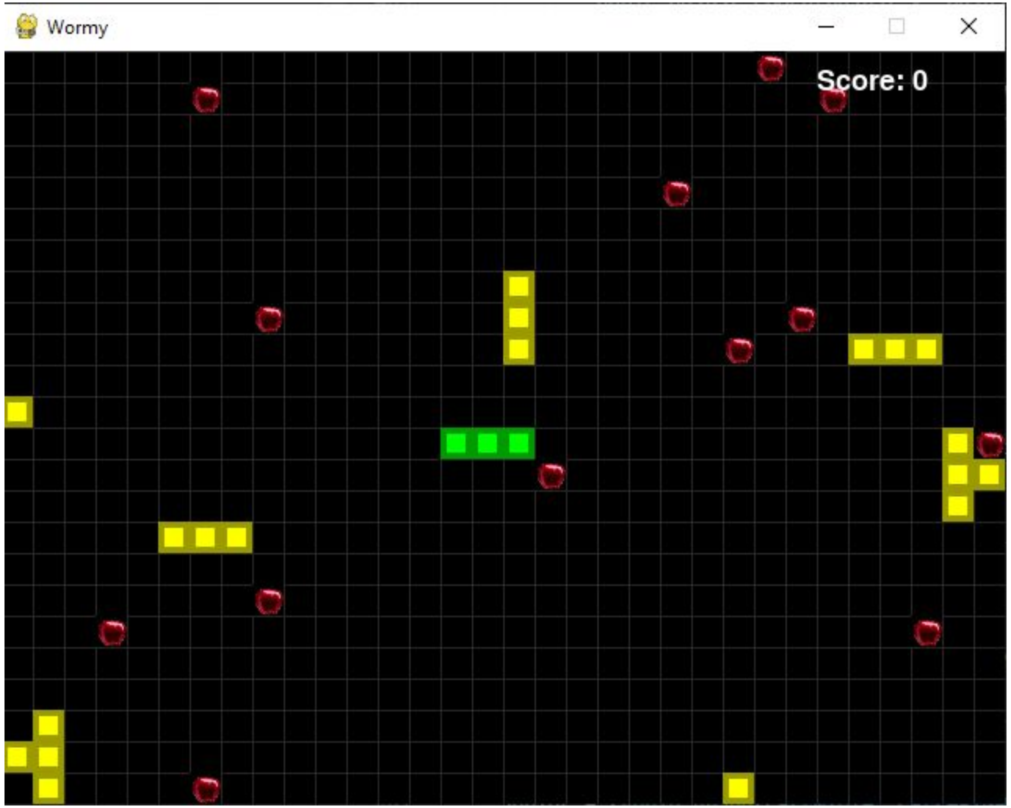
```
def hit(self, enemy_worm):
    for e_coord in enemy_worm.Coords:
        if e_coord in self.Coords:
            self.remove_tail()
            return True
    return False
```

run_game_camera_move_multiple_apple_worm

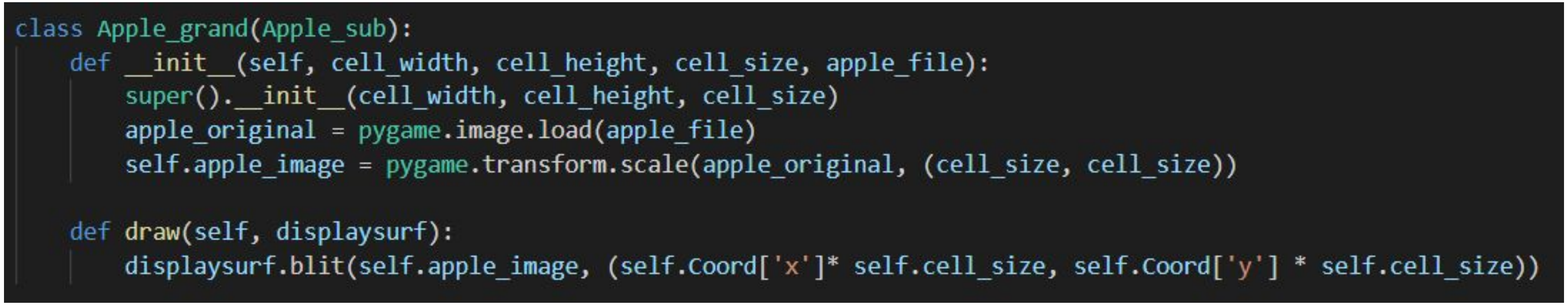


```
for k in range(len(enemy_worms)-1, -1, -1):
    one_worm = enemy_worms[k]
    if worm.hit(one_worm):
        del enemy_worms[k]
        if len(worm.Coords) < 3:
            return
    else:
        one_worm.update_remove_tail()
        one_worm.adjust_coord(adjust_x, adjust_y)
        one_worm.draw(displaysurf)

drawScore(len(worm.Coords) - 3 , displaysurf)
pygame.display.update()
fpsclock.tick(FPS)
```

Add apple image

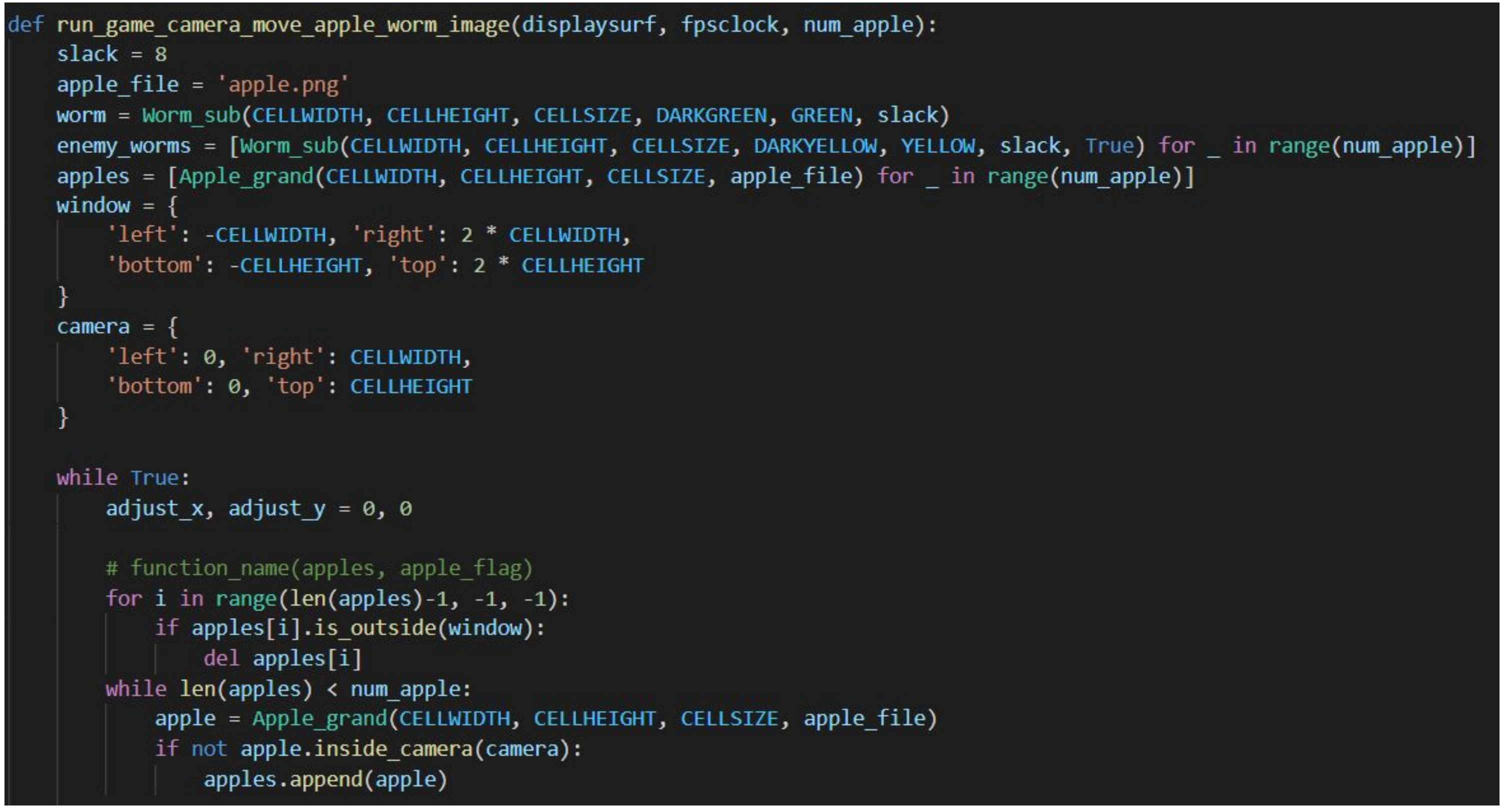


Grandchild class of Apple



```
class Apple_grand(Apple_sub):
    def __init__(self, cell_width, cell_height, cell_size, apple_file):
        super().__init__(cell_width, cell_height, cell_size)
        apple_original = pygame.image.load(apple_file)
        self.apple_image = pygame.transform.scale(apple_original, (cell_size, cell_size))

    def draw(self, displaysurf):
        displaysurf.blit(self.apple_image, (self.Coord['x']* self.cell_size, self.Coord['y'] * self.cell_size))
```

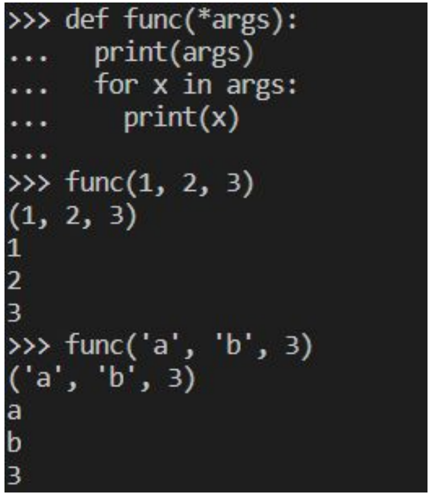
run_game_camera_move_apple_worm_image



```python
def run_game_camera_move_apple_worm_image(displaysurf, fpsclock, num_apple):
    slack = 8
    apple_file = 'apple.png'
    worm = Worm_sub(CELLWIDTH, CELLHEIGHT, CELLSIZE, DARKGREEN, GREEN, slack)
    enemy_worms = [Worm_sub(CELLWIDTH, CELLHEIGHT, CELLSIZE, DARKYELLOW, YELLOW, slack, True) for _ in range(num_apple)]
    apples = [Apple_grand(CELLWIDTH, CELLHEIGHT, CELLSIZE, apple_file) for _ in range(num_apple)]
    window = {
        'left': -CELLWIDTH, 'right': 2 * CELLWIDTH,
        'bottom': -CELLHEIGHT, 'top': 2 * CELLHEIGHT
    }
    camera = {
        'left': 0, 'right': CELLWIDTH,
        'bottom': 0, 'top': CELLHEIGHT
    }

    while True:
        adjust_x, adjust_y = 0, 0

        # function_name(apples, apple_flag)
        for i in range(len(apples)-1, -1, -1):
            if apples[i].is_outside(window):
                del apples[i]
        while len(apples) < num_apple:
            apple = Apple_grand(CELLWIDTH, CELLHEIGHT, CELLSIZE, apple_file)
            if not apple.inside_camera(camera):
                apples.append(apple)
```

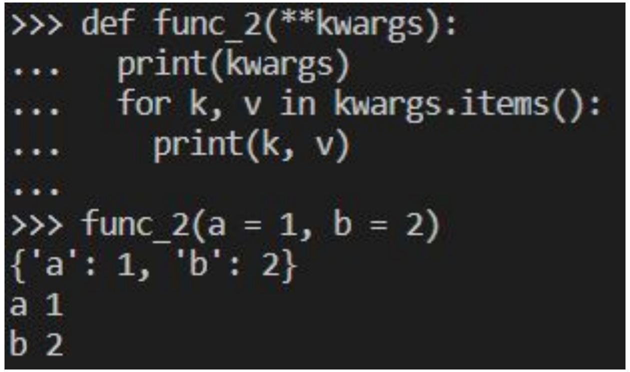
Python function input

input *args



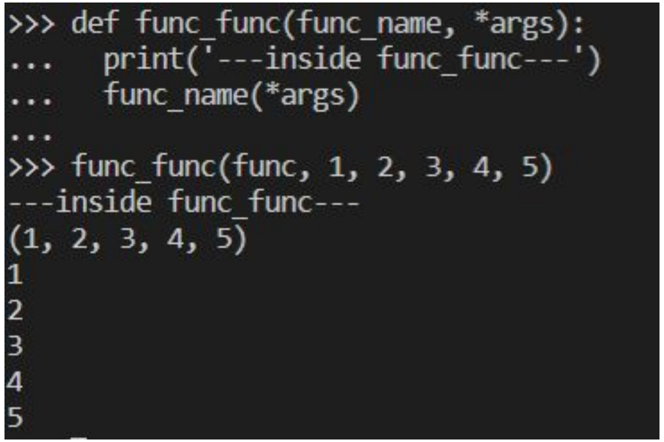
```
>>> def func(*args):
...   print(args)
...   for x in args:
...     print(x)
...
>>> func(1, 2, 3)
(1, 2, 3)
1
2
3
>>> func('a', 'b', 3)
('a', 'b', 3)
a
b
3
```

input *kwargs



```
>>> def func_2(**kwargs):
...   print(kwargs)
...   for k, v in kwargs.items():
...     print(k, v)
...
>>> func_2(a = 1, b = 2)
{'a': 1, 'b': 2}
a 1
b 2
```

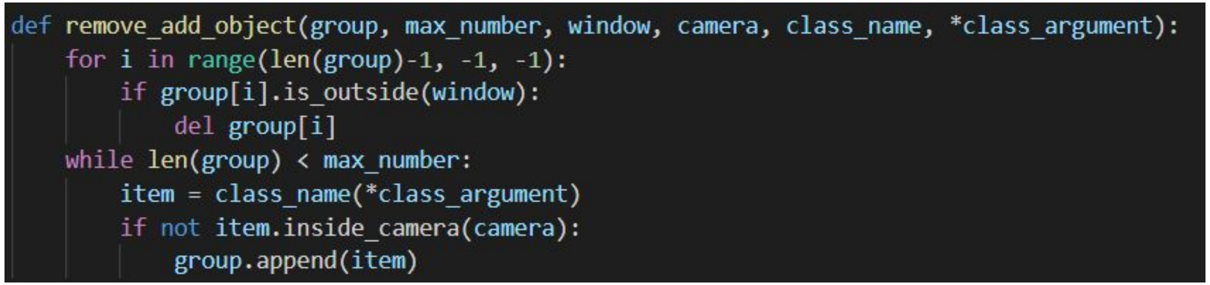
input func



```
>>> def func_func(func_name, *args):
...   print('---inside func_func---')
...   func_name(*args)
...
>>> func_func(func, 1, 2, 3, 4, 5)
---inside func_func---
(1, 2, 3, 4, 5)
1
2
3
4
5
```

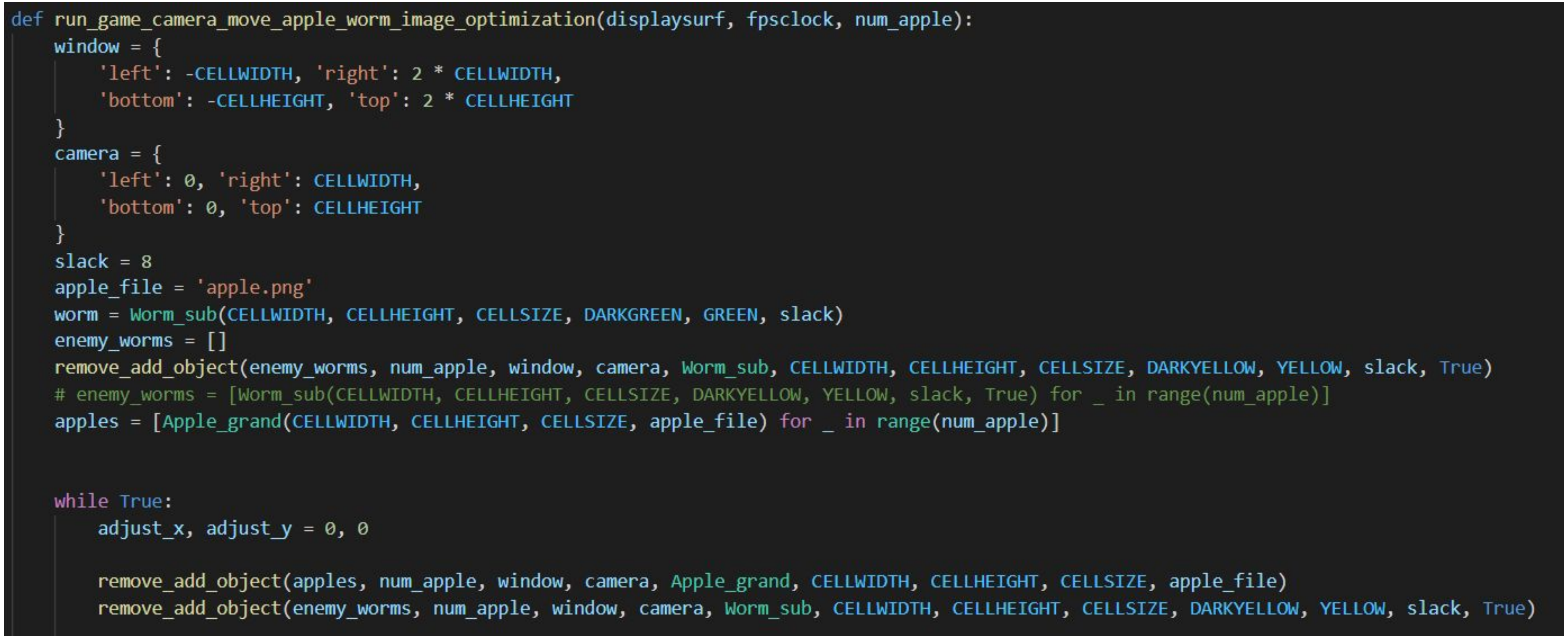
Optimization

Python function input: class and class_argument



```python
def remove_add_object(group, max_number, window, camera, class_name, *class_argument):
    for i in range(len(group)-1, -1, -1):
        if group[i].is_outside(window):
            del group[i]
    while len(group) < max_number:
        item = class_name(*class_argument)
        if not item.inside_camera(camera):
            group.append(item)
```

Optimization continue



```python
def run_game_camera_move_apple_worm_image_optimization(displaysurf, fpsclock, num_apple):
    window = {
        'left': -CELLWIDTH, 'right': 2 * CELLWIDTH,
        'bottom': -CELLHEIGHT, 'top': 2 * CELLHEIGHT
    }
    camera = {
        'left': 0, 'right': CELLWIDTH,
        'bottom': 0, 'top': CELLHEIGHT
    }
    slack = 8
    apple_file = 'apple.png'
    worm = Worm_sub(CELLWIDTH, CELLHEIGHT, CELLSIZE, DARKGREEN, GREEN, slack)
    enemy_worms = []
    remove_add_object(enemy_worms, num_apple, window, camera, Worm_sub, CELLWIDTH, CELLHEIGHT, CELLSIZE, DARKYELLOW, YELLOW, slack, True)
    # enemy_worms = [Worm_sub(CELLWIDTH, CELLHEIGHT, CELLSIZE, DARKYELLOW, YELLOW, slack, True) for _ in range(num_apple)]
    apples = [Apple_grand(CELLWIDTH, CELLHEIGHT, CELLSIZE, apple_file) for _ in range(num_apple)]


    while True:
        adjust_x, adjust_y = 0, 0

        remove_add_object(apples, num_apple, window, camera, Apple_grand, CELLWIDTH, CELLHEIGHT, CELLSIZE, apple_file)
        remove_add_object(enemy_worms, num_apple, window, camera, Worm_sub, CELLWIDTH, CELLHEIGHT, CELLSIZE, DARKYELLOW, YELLOW, slack, True)
```