SPACE INVADER GAME

SUBMITTED TO:
RAJU KUMAR
COMPUTER GRAPHICS
DELHI TECHNOLOGICAL UNIVERSITY

SUBMITTED BY:
PARTICIPANT 01: VINNET
ROLL: 2K19/CO/427
PARTICIPANT 02: VIRENDRA JANGIR
ROLL: 2K19/CO/432
PARTICIPANT 03: ZISHNENDU SARKER
ROLL: 2K19/CO/450

CONTENTS

ABSTRACT

An enhancement for the old-style space intruders game that includes shooting projectiles at trespassers in a space setting. In our variety, the alien spaceship can likewise discharge shots at the client's spaceship. We need to overcome 5 columns of 10 aliens each, for example, 50 aliens in each round with outsider rates expanding as they are killed. We will likewise add safeguards as insurance for the client's spaceship in specific regions. The safeguards should be annihilated for the projectiles to hit the spaceship. The spaceship will have a specific number of lives. There will be 4 sorts of alien spaceships in the game which will have various varieties of focuses. A secret boat will likewise show up every once in a while and in case it is obliterated, countless focuses will be given. Be that as it may, it will not remain on the screen for quite a while. After the first round, the client spaceship will actually want to shoot various slugs all the while as trouble increments. The game ends if all the lives are consumed or the alien spaceships arrive at the base.

INTRODUCTION

The game depends on the famous 2D arcade game, Space Invaders, created in the '70s by Tomohiro Nishikado. There have been a few executions of the various variants of the game over the long haul. There have not been many 3D executions of this game. The execution examined in this report plans to deliver a 3D adaptation of the game, with a clever UI that carries more authenticity to the player.

BACKGROUND

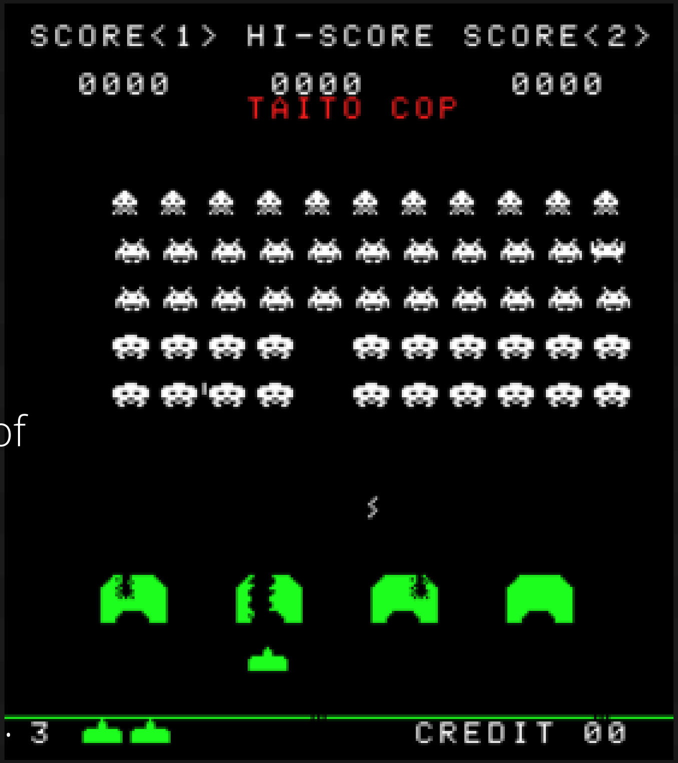
The report will begin with a description of the game, followed by information on how to play it. The talk will next shift to the game's technical features, including the software architecture. The implementation of several elements like collision detection, audio, and procedural modeling will also be presented. To increase the efficiency of rendering the models, optimization techniques were applied in game development. These will be noted as well. The stustudy will finish with suggestions on how the game may be improved in the future.

OBJECTIVE

The objective of the game is simply to destroy a formation of advancing enemy spaceships. The player moves on to the next level upon destroying all enemy ships. Each consecutive level gets tougher and tougher with enemies shooting at the player more frequently. The number of points obtained per enemy destroyed increases with every level. The player should aim to maximize his or her score.

ASPECT RATIO

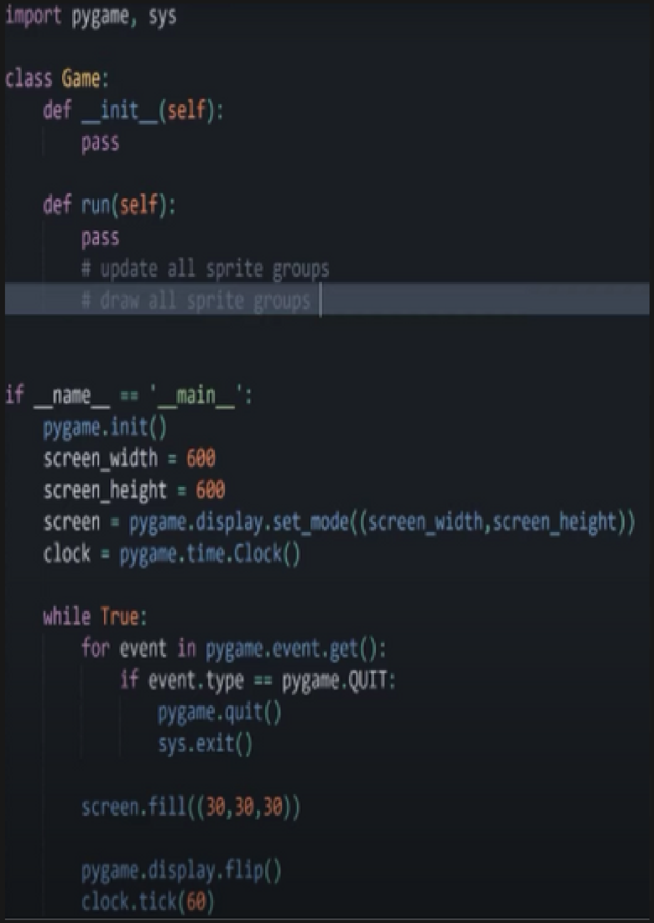
In most Space Invaders emulators, one bit of graphical information is drawn as one pixel on the screen. CRT “pixel” is slightly wider than it’s tall. The height is one scanline, which can be translated into one pixel, but the width can be more than that. That’s the case for Space Invaders. Because its resolution of 256 x 224 isn’t quite 4:3 like the monitor, the image gets stretched vertically as each pixel occupies a little more of the horizontal scanline.
Note that since the CRT monitor in Space Invaders is rotated, one pixel is in fact slightly taller than wide instead, after rotation.



WORKING PROCEDURE

BASIC SETUP

- Firstly, we have the basic pygame setup
- Then the display height and weight is setup
- Using those, we create a display surface
- Then we limit the frame rate
- In the actual game loop, we are checking for the escape button and when we are in the loop, we are drawing the color and anything else drawn in the game.
- Then we are creating a class "Game" where two methods are made. Then we create a reference in the main where we check the if statement for the name to reference main. Then call the game class in the loop which allows it to run.
- In the game class, we called two methods where init was the initial method of the class which contains the sprite groups of the player, obstacle, etc. And this is the framework of our game
- The next is used for updating and drawing all sprite groups
- At last, we created a game instance and run it.

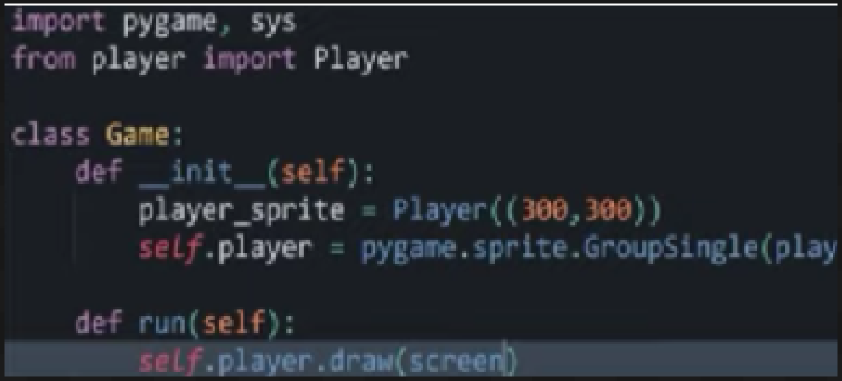


```python
import pygame, sys

class Game:
    def _init_(self):
        pass

    def run(self):
        pass
        # update all sprite groups
        # draw all sprite groups


if _name_ == '_main_':
    pygame.init()
    screen_width = 600
    screen_height = 600
    screen = pygame.display.set_mode((screen_width,screen_height))
    clock = pygame.time.Clock()

    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()

        screen.fill((30,30,30))

        pygame.display.flip()
        clock.tick(60)
```

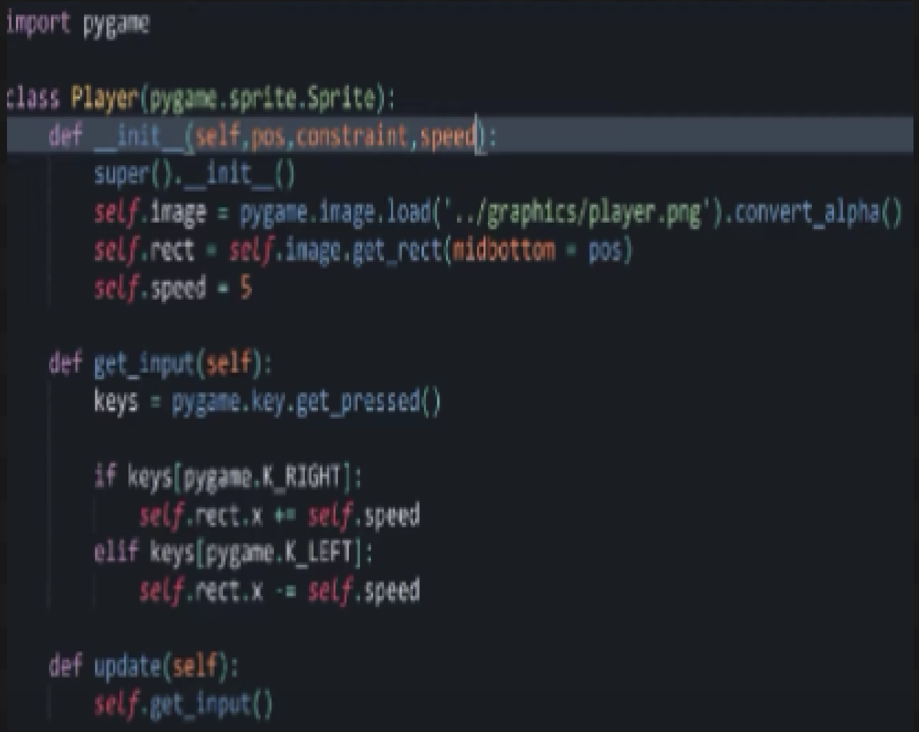
WORKING PROCEDURE

CREATING PLAYER

- SHOW THE IMAGE OF THE PLAYER



```
import pygame, sys
from player import Player

class Game:
    def __init__(self):
        player_sprite = Player((300,300))
        self.player = pygame.sprite.GroupSingle(play
    
    def run(self):
        self.player.draw(screen)
```



```
import pygame

class Player(pygame.sprite.Sprite):
    def __init__(self,pos,constraint,speed):
        super().__init__()
        self.image = pygame.image.load('../graphics/player.png').convert_alpha()
        self.rect = self.image.get_rect(midbottom = pos)
        self.speed = 5

    def get_input(self):
        keys = pygame.key.get_pressed()

        if keys[pygame.K_RIGHT]:
            self.rect.x += self.speed
        elif keys[pygame.K_LEFT]:
            self.rect.x -= self.speed

    def update(self):
        self.get_input()
```

- MOVE THE PLAYER

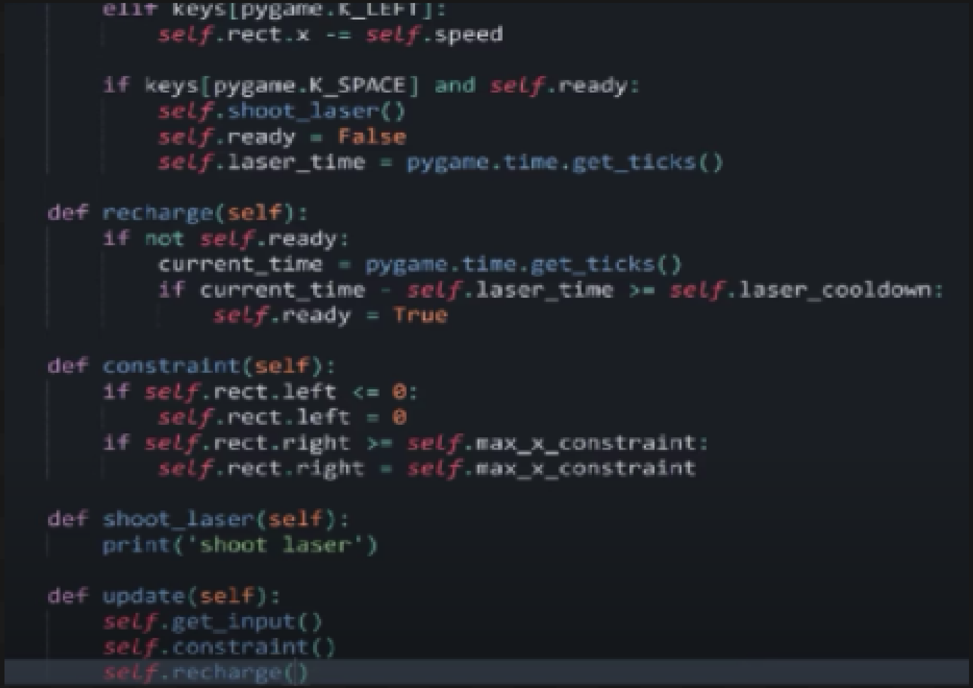
WORKING PROCEDURE

CREATING PLAYER

- CONSTRAINT PLAYER TO THE WINDOW

To constrain the player to the window, we have to define constraint and speed to the main method, we have added screen width. We also made a new method named “constraint”. And through the constraint method, we control the player inside the window

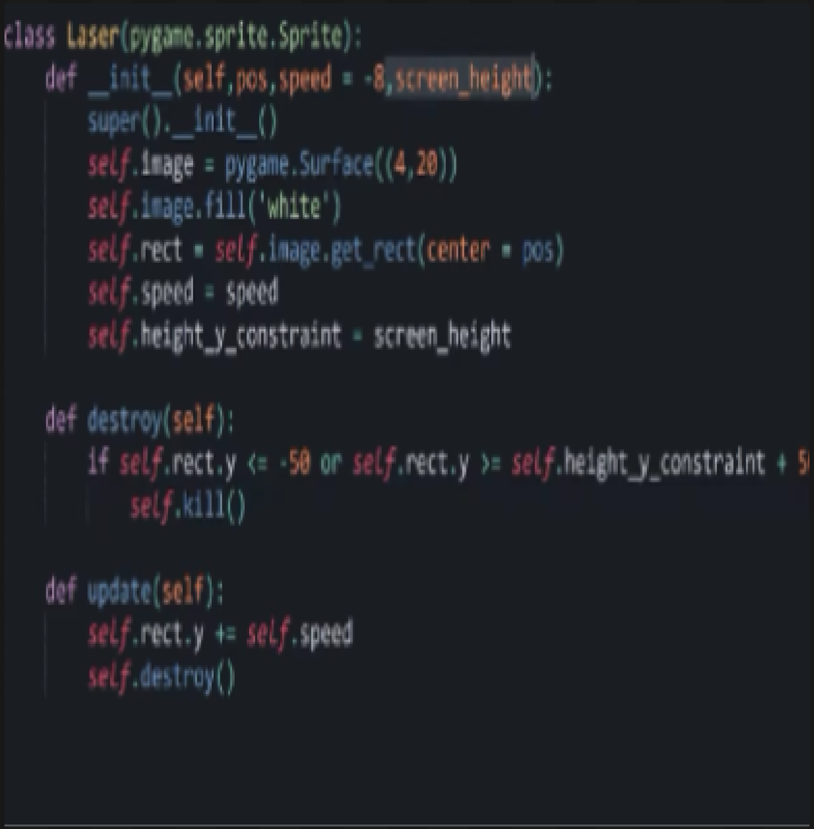
- SHOOT A LASER AND RECHARGE



```
        elif keys[pygame.K_LEFT]:
            self.rect.x -= self.speed

        if keys[pygame.K_SPACE] and self.ready:
            self.shoot_laser()
            self.ready = False
            self.laser_time = pygame.time.get_ticks()

    def recharge(self):
        if not self.ready:
            current_time = pygame.time.get_ticks()
            if current_time - self.laser_time >= self.laser_cooldown:
                self.ready = True

    def constraint(self):
        if self.rect.left <= 0:
            self.rect.left = 0
        if self.rect.right >= self.max_x_constraint:
            self.rect.right = self.max_x_constraint

    def shoot_laser(self):
        print('shoot laser')

    def update(self):
        self.get_input()
        self.constraint()
        self.recharge()
```

CREATING A LASER

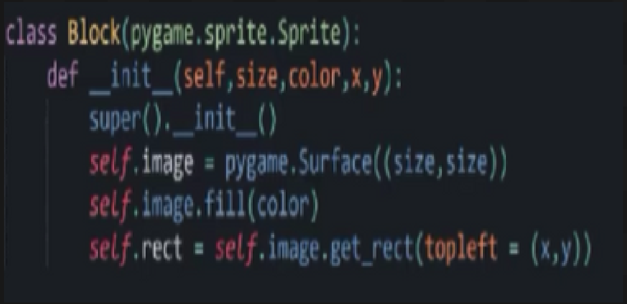
- CREATING A CLASS A LASER
- MOVING OF THE LASER



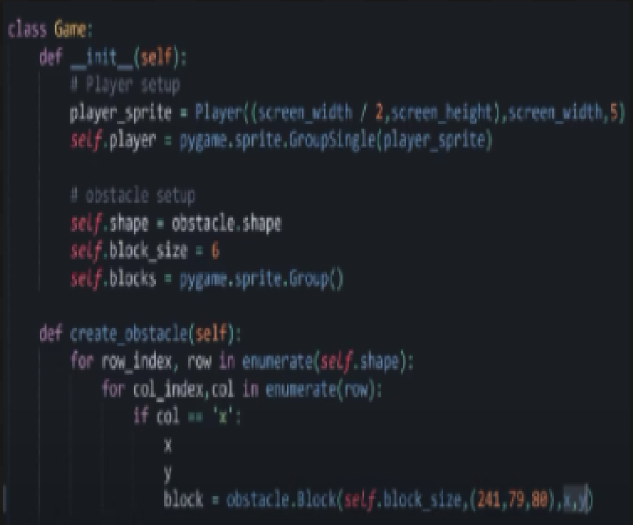
```
class Laser(pygame.sprite.Sprite):
    def __init__(self,pos,speed = -8,screen_height):
        super().__init__()
        self.image = pygame.Surface((4,20))
        self.image.fill('white')
        self.rect = self.image.get_rect(center = pos)
        self.speed = speed
        self.height_y_constraint = screen_height

    def destroy(self):
        if self.rect.y <= -50 or self.rect.y >= self.height_y_constraint + 5
            self.kill()

    def update(self):
        self.rect.y += self.speed
        self.destroy()
```

CREATING OBSTACLES

- CREATING AN OBSTACLES
- SHAPING OF AN OBSTACLES
- PACEMENT OF THE OBSTACLES
- CREATE MULTIPLE OBSTACLES

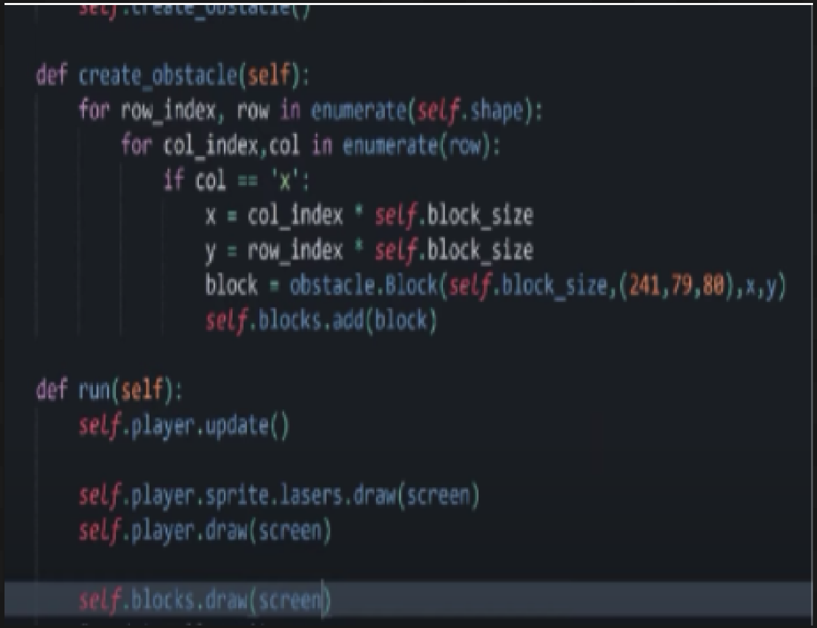


```
class Block(pygame.sprite.Sprite):
    def __init__(self,size,color,x,y):
        super().__init__()
        self.image = pygame.Surface((size,size))
        self.image.fill(color)
        self.rect = self.image.get_rect(topleft = (x,y))
```



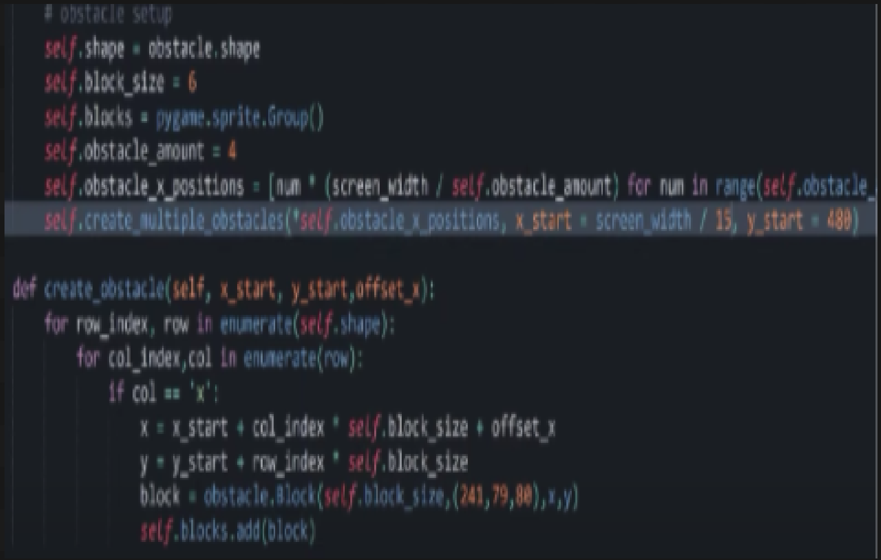
```
class Game:
    def __init__(self):
        # Player setup
        player_sprite = Player((screen_width / 2,screen_height),screen_width,5)
        self.player = pygame.sprite.GroupSingle(player_sprite)

        # obstacle setup
        self.shape = obstacle.shape
        self.block_size = 6
        self.blocks = pygame.sprite.Group()

    def create_obstacle(self):
        for row_index, row in enumerate(self.shape):
            for col_index,col in enumerate(row):
                if col == 'x':
                    x
                    y
                    block = obstacle.Block(self.block_size,(241,79,80),x,y)
```

PLACEMENT OF THE OBSTACLES

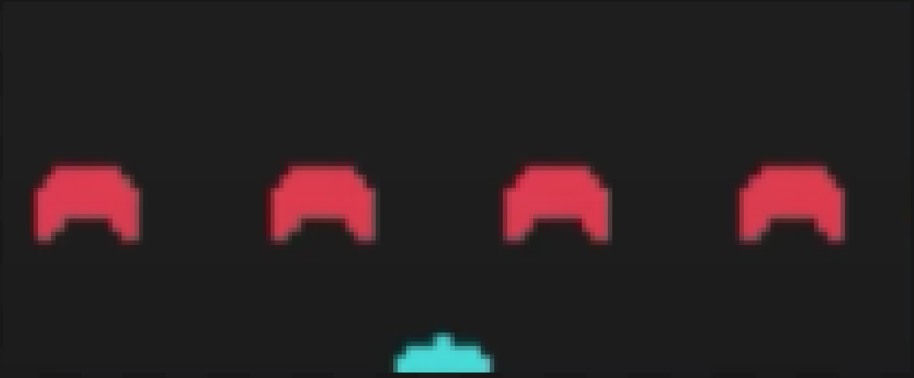


```
    self.create_obstacle()

def create_obstacle(self):
    for row_index, row in enumerate(self.shape):
        for col_index,col in enumerate(row):
            if col == 'x':
                x = col_index * self.block_size
                y = row_index * self.block_size
                block = obstacle.Block(self.block_size,(241,79,80),x,y)
                self.blocks.add(block)

def run(self):
    self.player.update()

    self.player.sprite.lasers.draw(screen)
    self.player.draw(screen)

    self.blocks.draw(screen)
```

CREATING MULTIPLE OBSTACLES

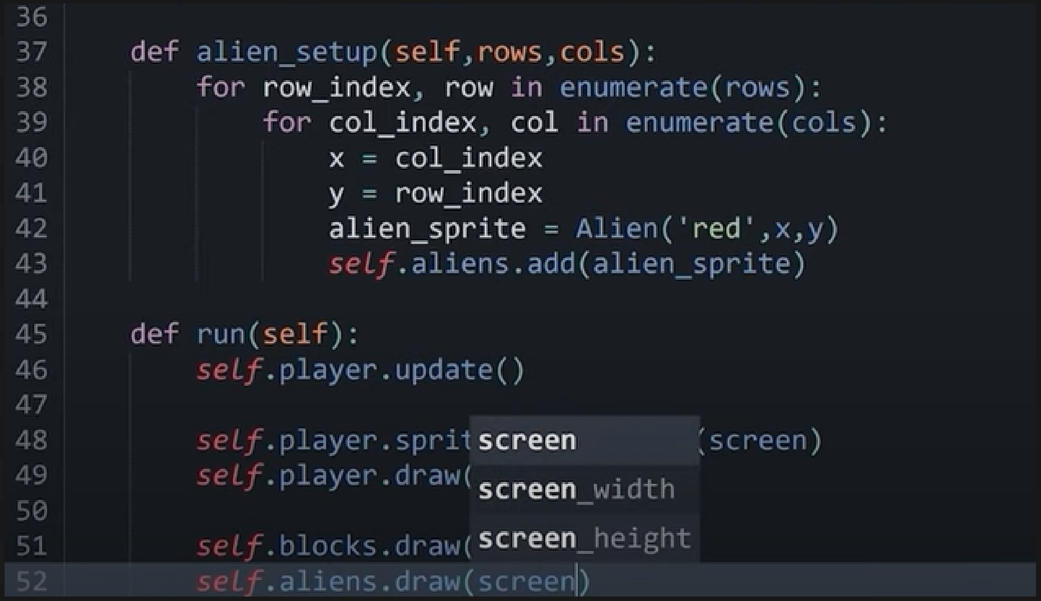


```
    # obstacle setup
    self.shape = obstacle.shape
    self.block_size = 6
    self.blocks = pygame.sprite.Group()
    self.obstacle_amount = 4
    self.obstacle_x_positions = [num * (screen_width / self.obstacle_amount) for num in range(self.obstacle_
    self.create_multiple_obstacles(*self.obstacle_x_positions, x_start = screen_width / 15, y_start = 480)

def create_obstacle(self, x_start, y_start,offset_x):
    for row_index, row in enumerate(self.shape):
        for col_index,col in enumerate(row):
            if col == 'x':
                x = x_start + col_index * self.block_size + offset_x
                y = y_start + row_index * self.block_size
                block = obstacle.Block(self.block_size,(241,79,80),x,y)
                self.blocks.add(block)
```

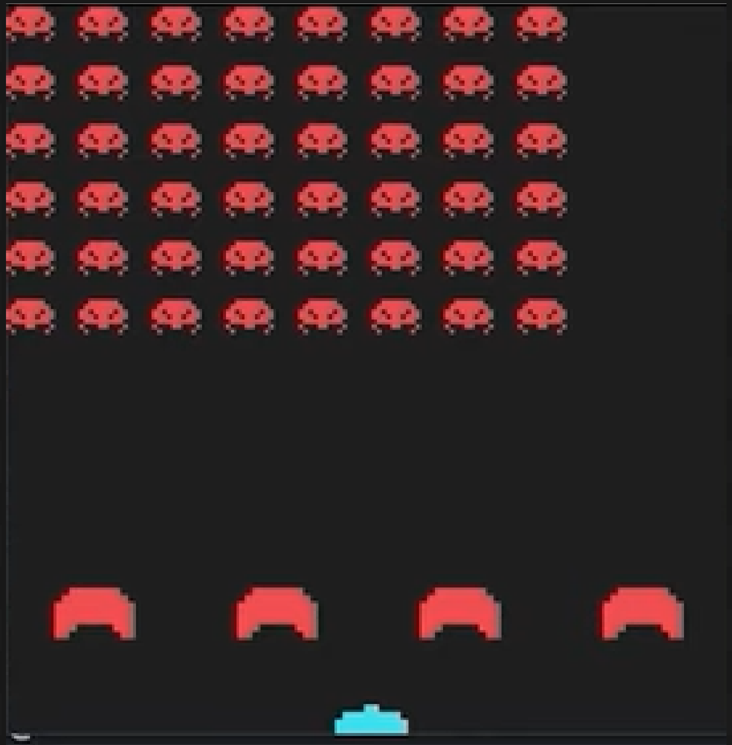
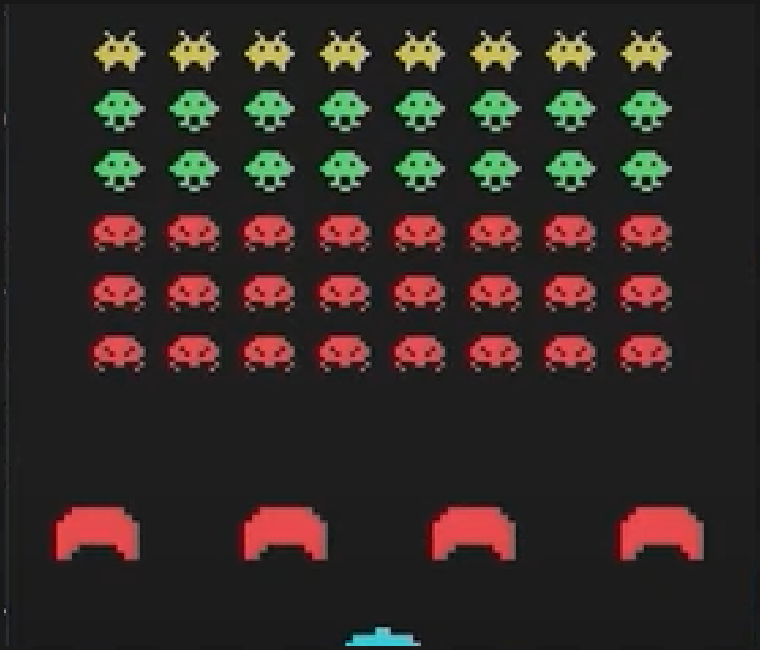


CREATING ALIENS

- Placements of aliens
- Setup of aliens
- The difference in color of aliens
- Dymanic of aliens

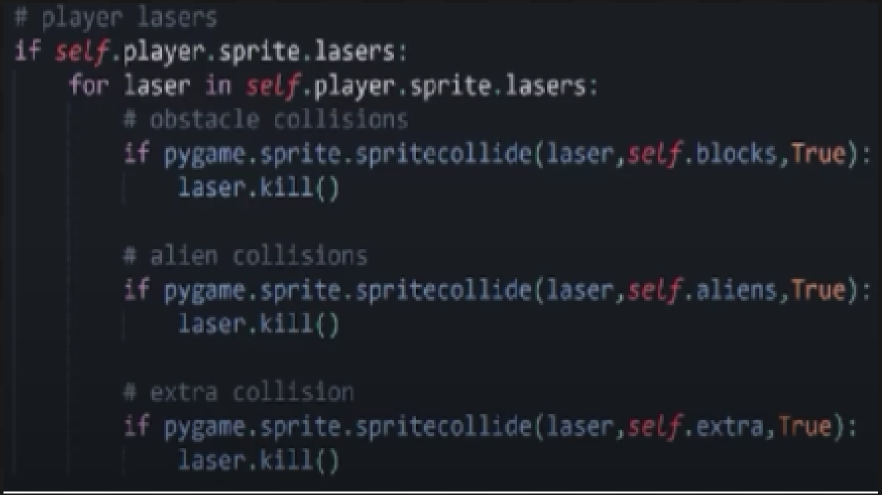


```
    def alien_setup(self,rows,cols):
        for row_index, row in enumerate(rows):
            for col_index, col in enumerate(cols):
                x = col_index
                y = row_index
                alien_sprite = Alien('red',x,y)
                self.aliens.add(alien_sprite)

    def run(self):
        self.player.update()

        self.player.sprit screen          (screen)
        self.player.draw( screen_width

        self.blocks.draw( screen_height
        self.aliens.draw(screen)
```

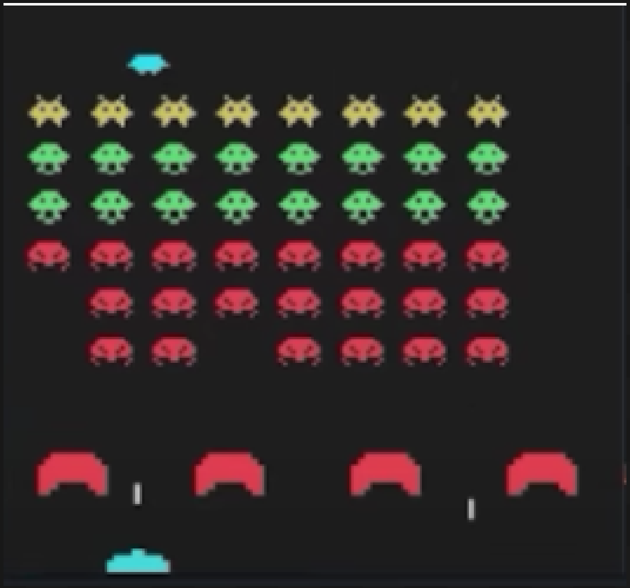



COLLISIONS

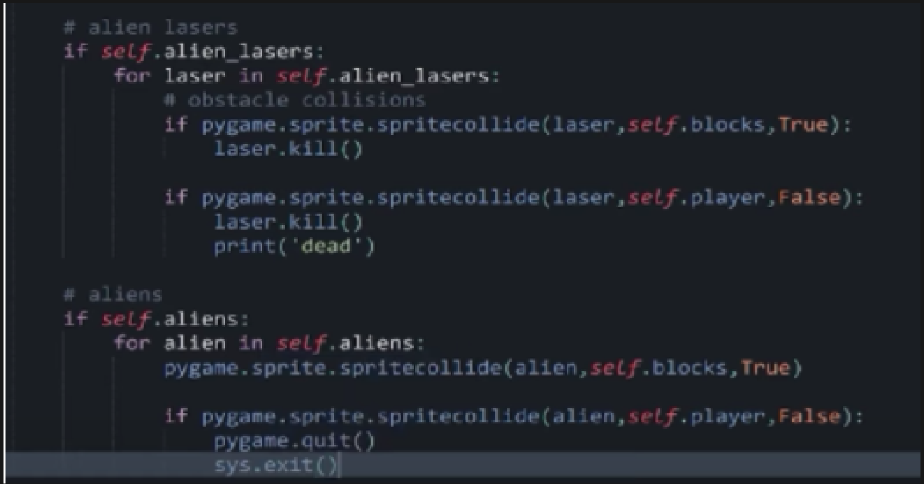
1.PLAYERS LASER



```python
# player lasers
if self.player.sprite.lasers:
    for laser in self.player.sprite.lasers:
        # obstacle collisions
        if pygame.sprite.spritecollide(laser,self.blocks,True):
            laser.kill()

        # alien collisions
        if pygame.sprite.spritecollide(laser,self.aliens,True):
            laser.kill()

        # extra collision
        if pygame.sprite.spritecollide(laser,self.extra,True):
            laser.kill()
```

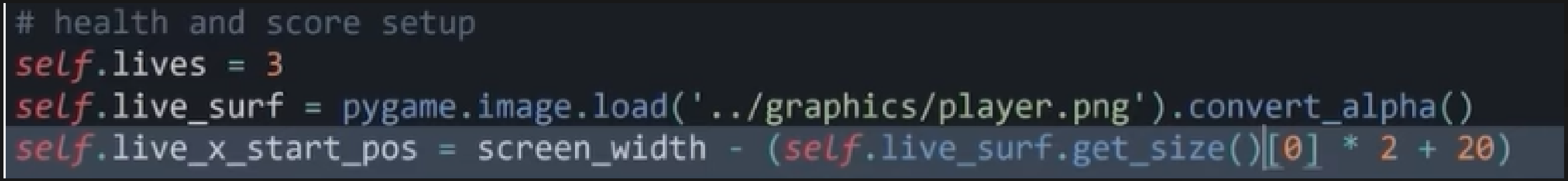


2. ALIEN LASERS

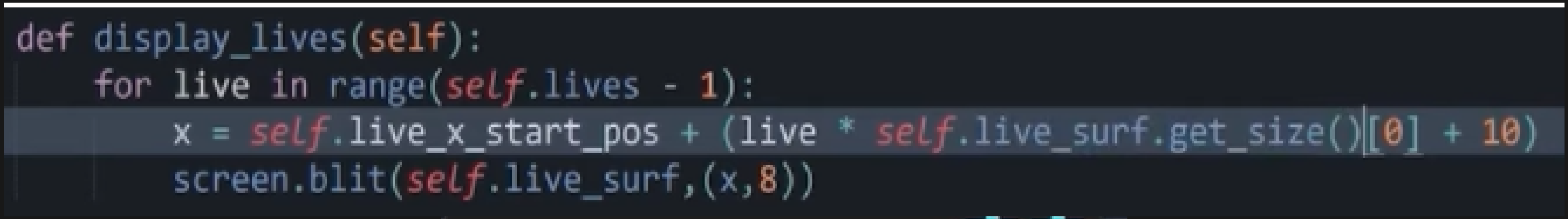


```python
# alien lasers
if self.alien_lasers:
    for laser in self.alien_lasers:
        # obstacle collisions
        if pygame.sprite.spritecollide(laser,self.blocks,True):
            laser.kill()

        if pygame.sprite.spritecollide(laser,self.player,False):
            laser.kill()
            print('dead')

# aliens
if self.aliens:
    for alien in self.aliens:
        pygame.sprite.spritecollide(alien,self.blocks,True)

        if pygame.sprite.spritecollide(alien,self.player,False):
            pygame.quit()
            sys.exit()
```

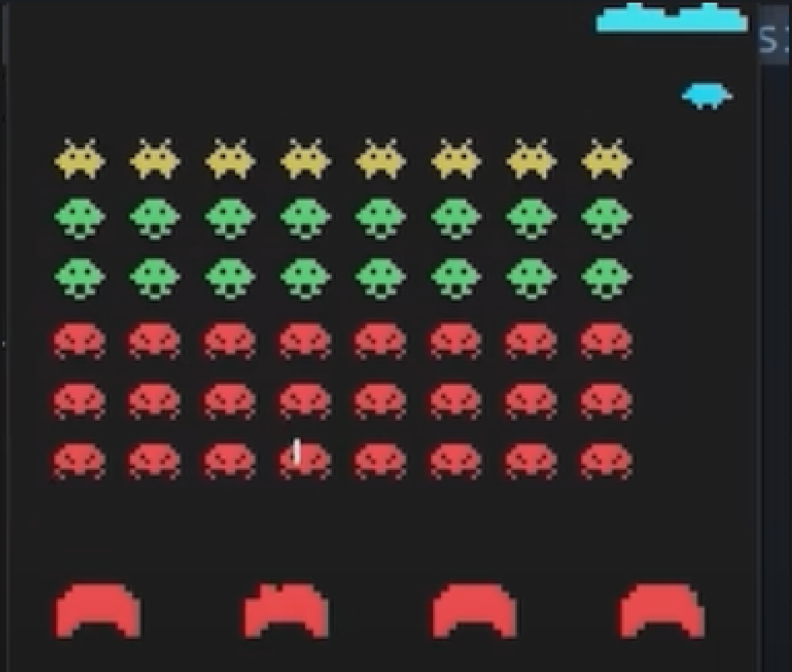
ADDING THE HEALTH SYSTEM



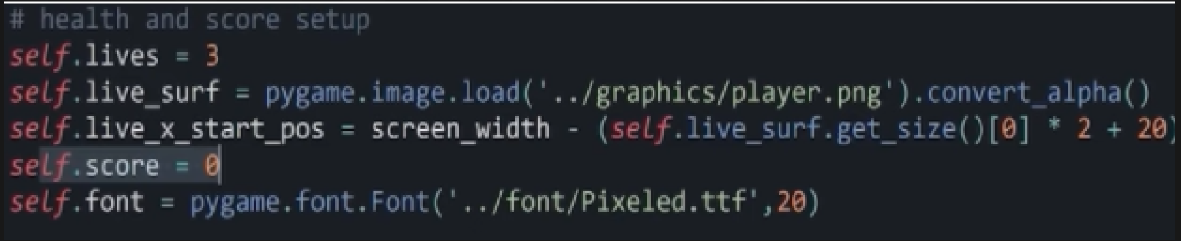
```
# health and score setup
self.lives = 3
self.live_surf = pygame.image.load('../graphics/player.png').convert_alpha()
self.live_x_start_pos = screen_width - (self.live_surf.get_size()[0] * 2 + 20)
```



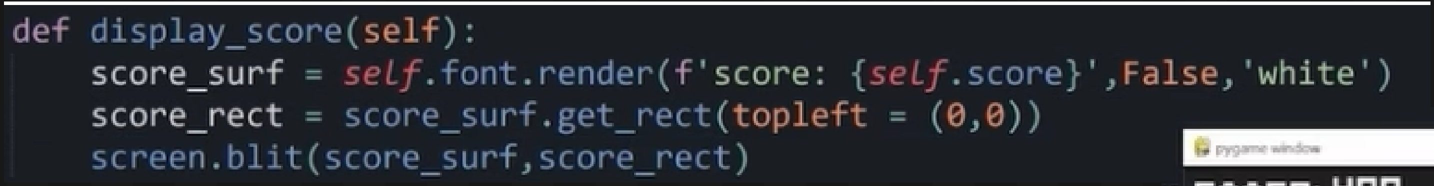
```
def display_lives(self):
    for live in range(self.lives - 1):
        x = self.live_x_start_pos + (live * self.live_surf.get_size()[0] + 10)
        screen.blit(self.live_surf,(x,8))
```



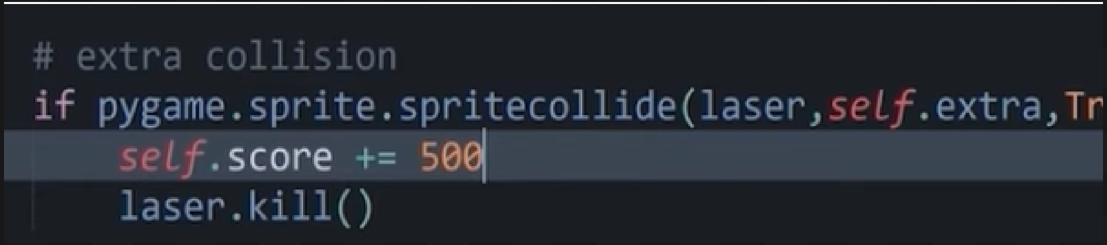
ADDING SCORING SYSTEM



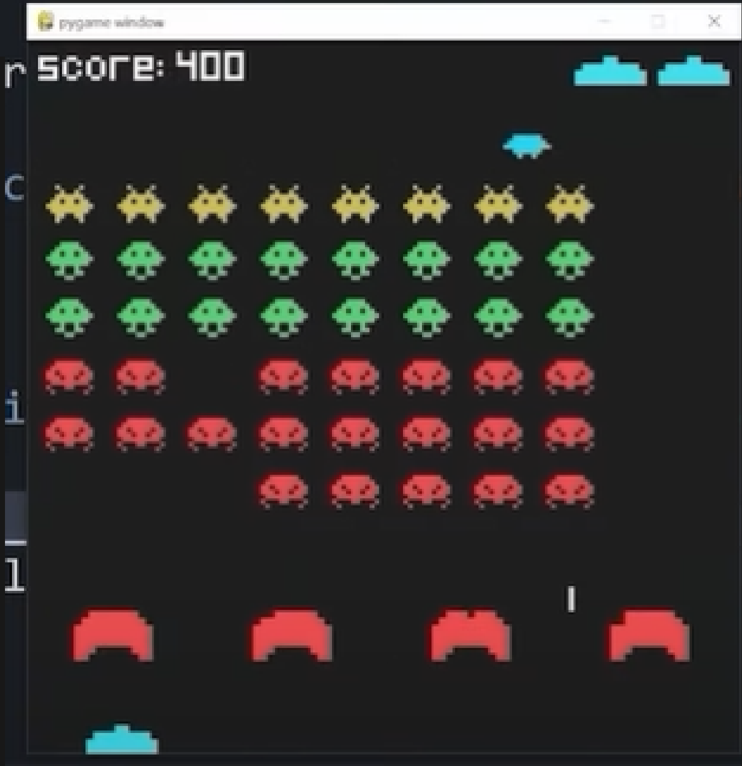
```python
# health and score setup
self.lives = 3
self.live_surf = pygame.image.load('../graphics/player.png').convert_alpha()
self.live_x_start_pos = screen_width - (self.live_surf.get_size()[0] * 2 + 20)
self.score = 0
self.font = pygame.font.Font('../font/Pixeled.ttf',20)
```



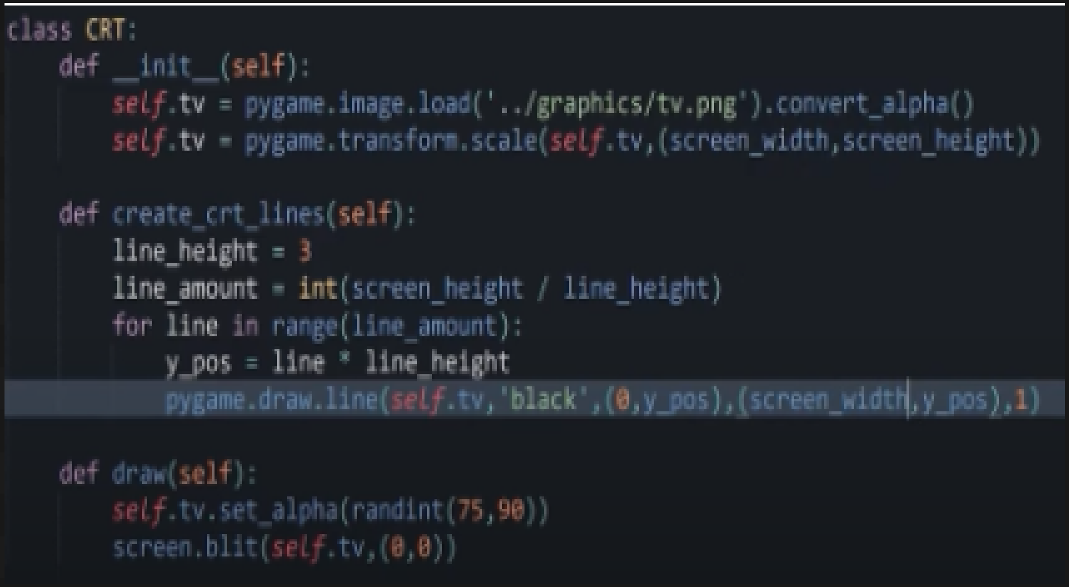
```python
def display_score(self):
    score_surf = self.font.render(f'score: {self.score}',False,'white')
    score_rect = score_surf.get_rect(topleft = (0,0))
    screen.blit(score_surf,score_rect)
```



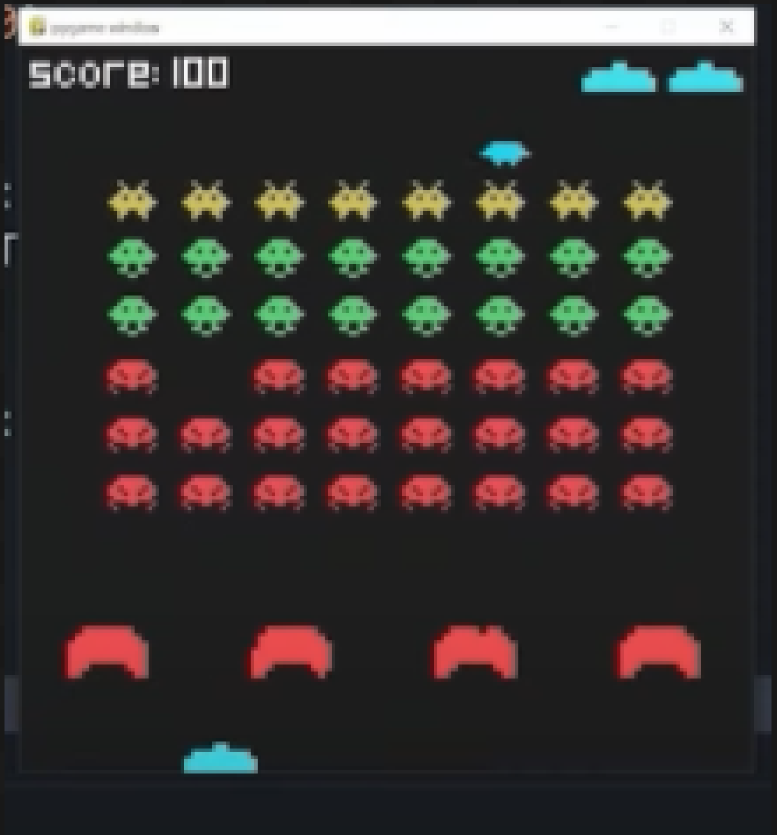
```python
# extra collision
if pygame.sprite.spritecollide(laser,self.extra,Tr
    self.score += 500
    laser.kill()
```



ADDING CRT STYLING

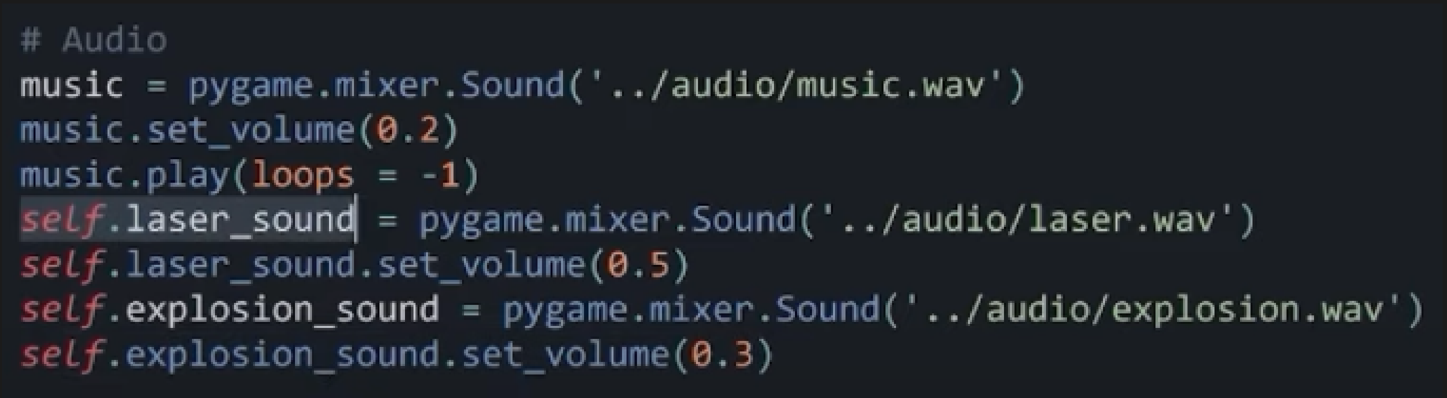


```
class CRT:
    def __init__(self):
        self.tv = pygame.image.load('../graphics/tv.png').convert_alpha()
        self.tv = pygame.transform.scale(self.tv,(screen_width,screen_height))

    def create_crt_lines(self):
        line_height = 3
        line_amount = int(screen_height / line_height)
        for line in range(line_amount):
            y_pos = line * line_height
            pygame.draw.line(self.tv,'black',(0,y_pos),(screen_width,y_pos),1)

    def draw(self):
        self.tv.set_alpha(randint(75,90))
        screen.blit(self.tv,(0,0))
```

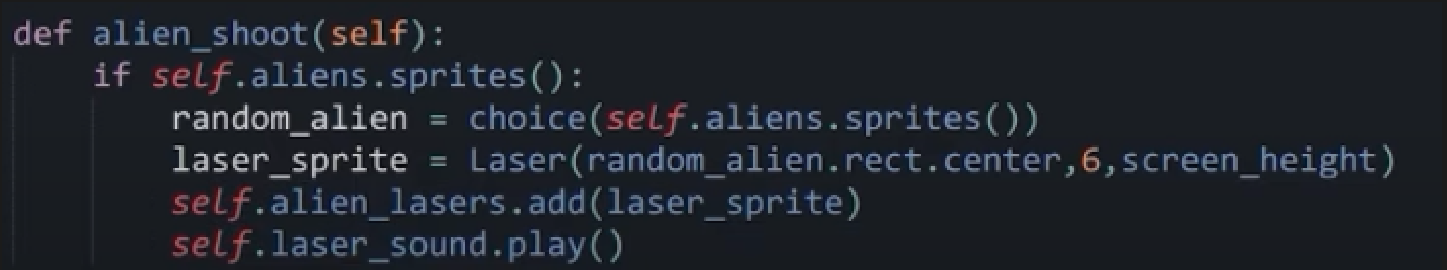


- Create a class called crt using pygame.
- It’s not inherited from any class.
- Setting up init method
- We initialize image and rect for the CRT
- Created a draw function
- Creating crt as an instance of crt class

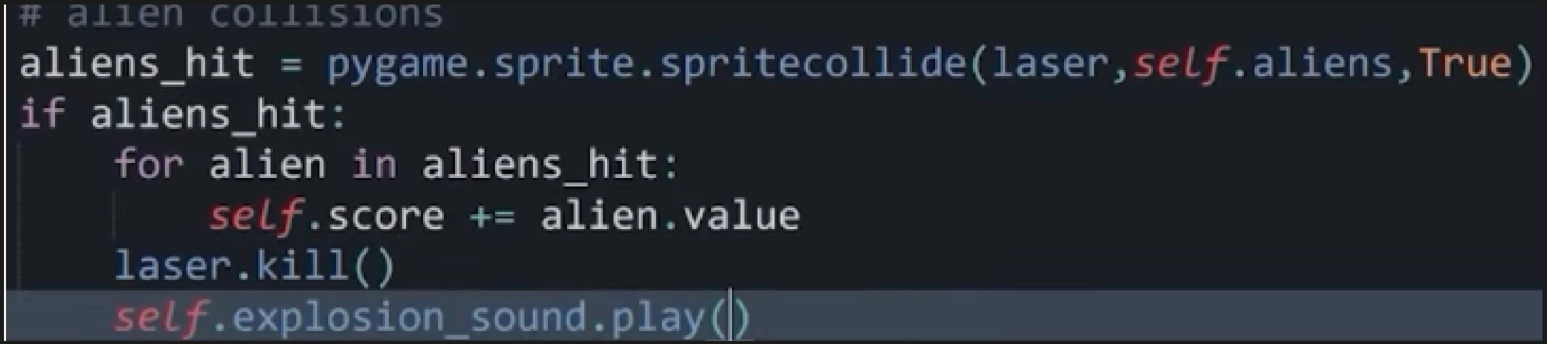
ADDING MUSIC AND SOUND



```python
# Audio
music = pygame.mixer.Sound('../audio/music.wav')
music.set_volume(0.2)
music.play(loops = -1)
self.laser_sound = pygame.mixer.Sound('../audio/laser.wav')
self.laser_sound.set_volume(0.5)
self.explosion_sound = pygame.mixer.Sound('../audio/explosion.wav')
self.explosion_sound.set_volume(0.3)
```

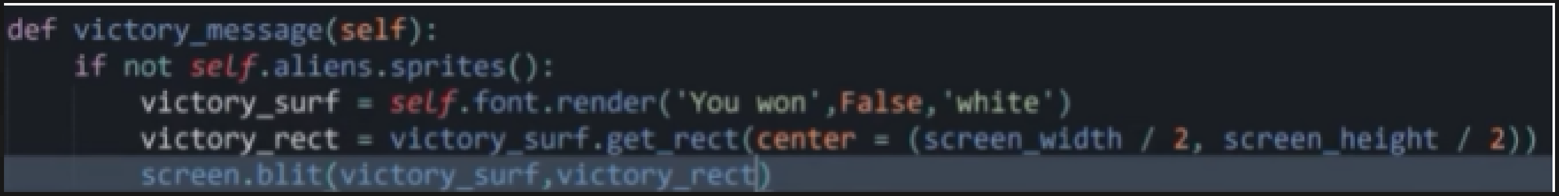


```python
def alien_shoot(self):
    if self.aliens.sprites():
        random_alien = choice(self.aliens.sprites())
        laser_sprite = Laser(random_alien.rect.center,6,screen_height)
        self.alien_lasers.add(laser_sprite)
        self.laser_sound.play()
```

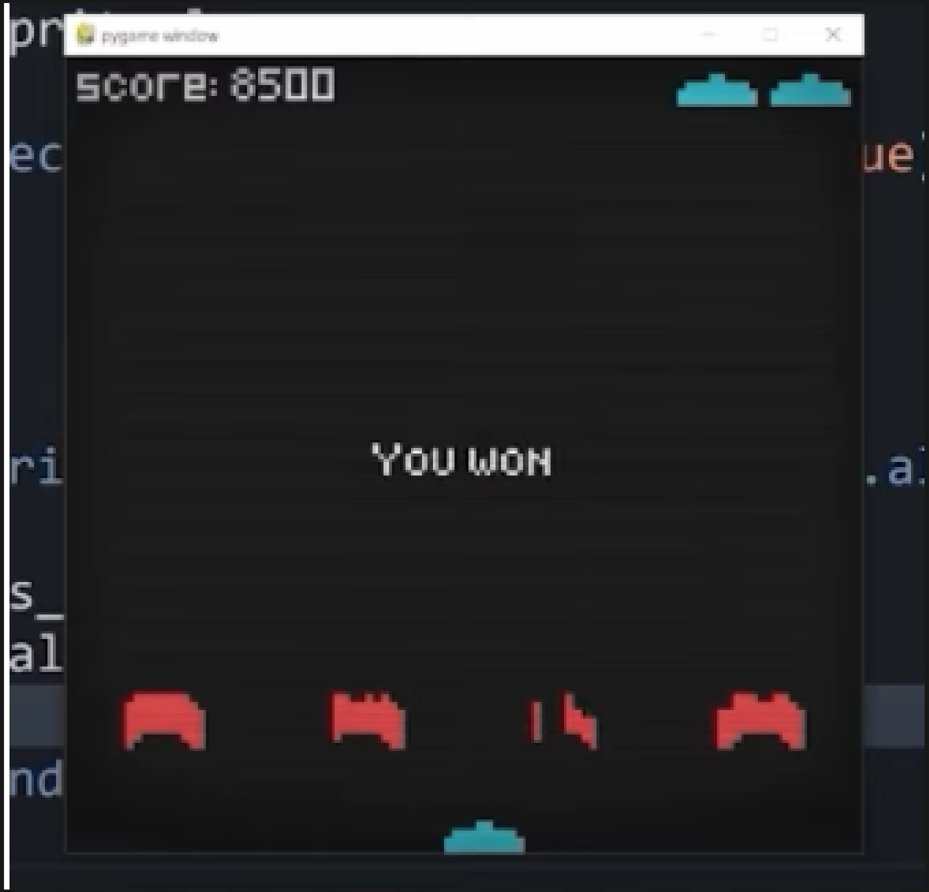


```python
# alien collisions
aliens_hit = pygame.sprite.spritecollide(laser,self.aliens,True)
if aliens_hit:
    for alien in aliens_hit:
        self.score += alien.value
    laser.kill()
    self.explosion_sound.play()
```

ADDING VICTORY SCREEN



```
def victory_message(self):
    if not self.aliens.sprites():
        victory_surf = self.font.render('You won',False,'white')
        victory_rect = victory_surf.get_rect(center = (screen_width / 2, screen_height / 2))
        screen.blit(victory_surf,victory_rect)
```



CONCLUSION

If we were to summarize the entire experience of this third-year project in a word, it would be an adventure. Looking back to the past few months, we can truly assert it has been a journey that started slowly, without much progress due to the lack of technical knowledge required for this project but took off while making progress with the research.

We began the project with absolutely no idea on how graphics work, how a collision detection system works, or even how to display an image on the screen and, by the end of the project, we got to a point where we could create our own or modify existing game-specific algorithms to suit our needs. What is more, this was our first attempt at building a graphics-based game and we believe that achieving a single-player space shooter arcade game with a smart powerup module and AI for NPCs is a good start if we decide to go down the game development path.

REFERENCES

[1] Pygame. URL: http://www.pygame.org/hifi.html. Last accessed: 04/04/2016

[2] The ESA 2015 Report. URL: http://www.theesa.com/wp-content/uploads/2015/04/ESAEssentialFacts2015.pdf. Last accessed: 05/04/2016

[3] Video Games Industry sales. URL: http://vgsales.wikia.com/wiki/Video_game_industry. Last accessed: 05/04/2016

[4] Video Game Industry sales in Japan. URL: http://vgsales.wikia.com/wiki/Video_games_in_Japan. Last accessed: 05/04/2016

[5] Unity3D. URL: http://blogs.unity3d.com/2014/09/03/documentationunity-scriptinglanguagesandyou/. Last accessed: 28/04/2016

THANK YOU!

-VINEET

-VIRENDRA JANGIR

-ZISHNENDU SARKER