Guardian of the Galaxy

**ABSTRACT**

If we understand computer graphics simply, it deals with creation, manipulation and storage of images and objects. In this project we have used computer graphics concepts to develop a 2D single player game. The game we have developed is inspired from the 1978’s famous game ‘Space Invaders’. In this project we have tried to apply the same mechanics of the game in our project. For the development of the game we have used Python3 and Pygame module.

**Chapter 1**

**INTRODUCTION**

Computer graphics is responsible for the creation, manipulation, and storage of objects, images. Everything we see on our laptops, televisions, mobile screens is related to computer graphics. Computer graphics has several applications. Computer Art, Computer-aided drawing, Presentation Graphics, Entertainment, Education, Training, Visualisation, Image Processing, Machine Drawing, and Graphical User Interface are all major applications of computer graphics. To apply and understand more about the concepts of computer graphics we have developed a game. In this innovative project of computer graphics, we tried to learn more and more about the mechanics of game development and how computer graphics is important for its development. Computer graphics play an important role in game development. There has been a lot of development in computer graphics over many years. The performance in computer graphics has become much better than in comparison to a few decades back. In the late 20th century, the computer graphics were mainly pixellated. Now the advancement in technologies, opened the gates for the gaming world to develop high-resolution games. Gamers can play games in high-resolution on their mobile phones and computer.

In computer graphics, we learnt many basic concepts, and we applied those concepts in our game. We worked on a project to develop a 2D shooting game.

The Guardian of the Galaxy is a 2D single-player game.

The Guardian of the Galaxy is based upon 1978’s famous arcade game Space Invaders developed by Taito Corporation.

The game is developed using python.

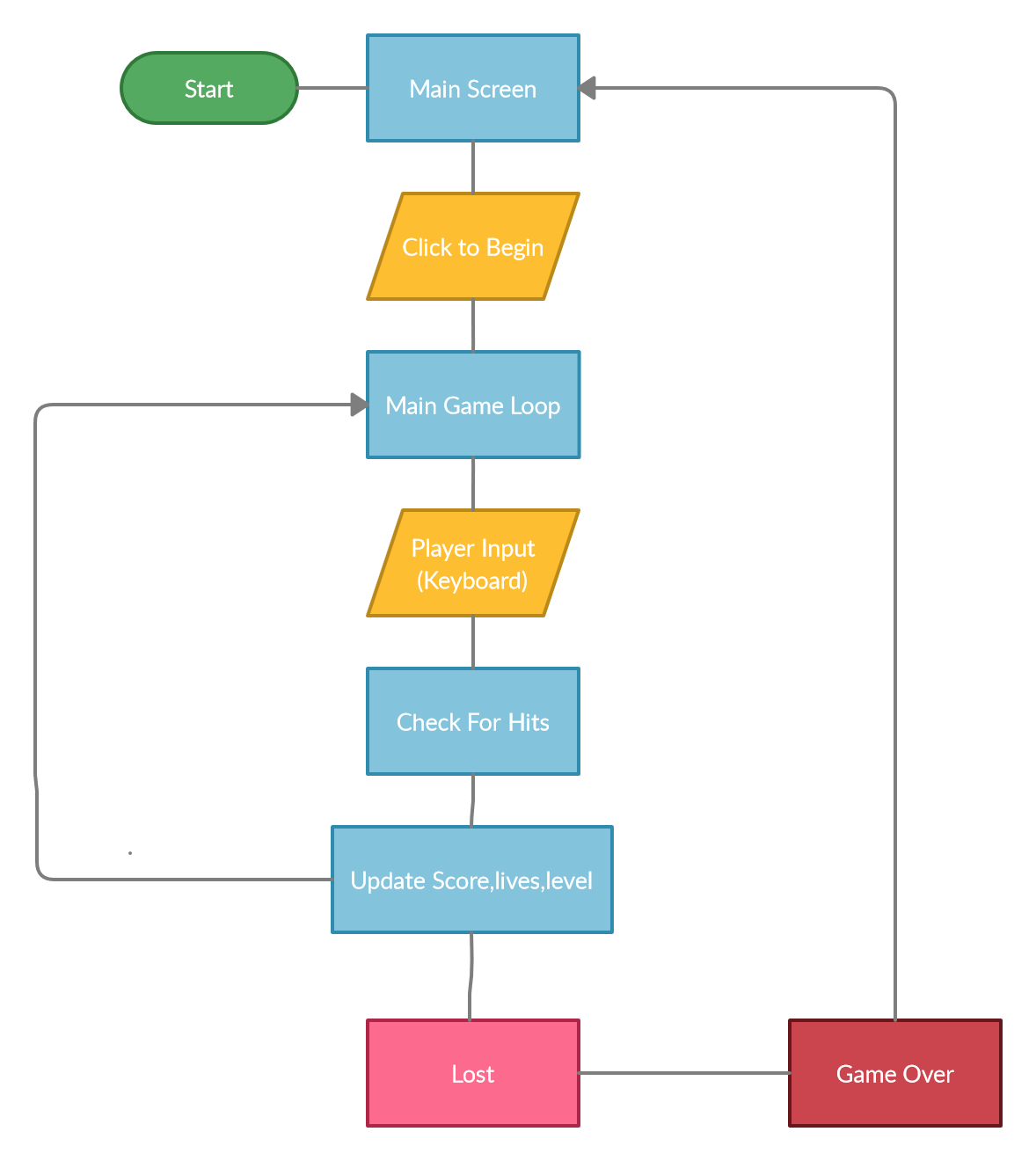
The game is simple to understand but challenging to play, you will be controlling a spaceship, you are going to face a lot of enemy ships, their main motive is to destroy your ship. You have to be careful and extra dodgy; your laser weapon is powerful; it can take out the enemy ship in a single fire. You have a powerful defence but it can only save you for some time so remember you are not immune to enemy fire.

**Motivation**

There has been a significant increase in the gaming world, the interests of developers, gamers, fans have been increasing a lot in the recent times. Due to the streaming platforms like Twitch and YouTube people are attracting towards games. And because of the pandemic situation people were forced to stay inside their homes, which also contributed in the growing interests of people in the gaming world. If we talk about India, around 80% of games played are for mobile phones. We have a huge audience for mobile games, but the problem is we don’t have many gaming studios and game developers. People do like to play games but there are not many companies for game developments. The main reason for building this game was to learn more about the gaming world and explore this path. As the students of Computer Engineering, we should always be innovative and should focus on more unique things.

**PROPOSED METHODOLOGY**

**FLOW CHART**

****

**Explanation:**

When you run the script, you’ll land on the main screen of the game where you’ll be asked to press the keyboard button or mouse button. This will start the game. From there the main game loop will start. The player will register the input from the keyboard which will be recorded. The game will record the hits and will update the results, scores, lives and levels in the game. If a player loses all 5 lives, then the game will be over and the player will be redirected to Main Screen.

**GAME ASSETS**

In the game we have used 9 assets which are as following:



This is our main ship, The Defender for fighting with the enemies, The weapon of our ship is Yellow laser which is powerful enough to eliminate the enemies in a single hit.



These are the three dangerous and vicious ships which are designed for only one task to destroy the Defender Ship, there weapons are not very powerful to damage the Defender Ship in a single hit.



We have used a static image in our game. The width and height of the background are fixed. It can be modified in the code.

**IMPLEMENTATION AND RESULT ANALYSIS**

Tools Used

Language: Python 3

To develop this game, we have used the python3 language.

Modules Used:

* Pygame – Pygame is a cross-platform module which contains several submodules to create video games.
* OS – for using the System functionalities.
* Time – The time module is used for time and clock
* Random – this module is used for random probability distribution

**CODE WITH EXPLANATION**

Firstly, we imported the following Library files into our program code, they are namely

* pygame
* os
* time
* random

import pygame

import os

import time

import random

pygame.font.init()

After importing these library files, the next thing we need to do is actually load all of our image assets into our script so that we can use them and display them on the screen.

WIDTH, HEIGHT = 1280, 730

WIN = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("Guardian of the Galaxy")

These images are for the ENEMY Space Ships. Total there are 3 types of enemy ships, Namely: -

1. RED
2. GREEN
3. BLUE

RED\_SPACE\_SHIP = pygame.image.load(os.path.join("assets", "red\_ship.png"))

GREEN\_SPACE\_SHIP = pygame.image.load(os.path.join("assets", "green\_ship.png"))

BLUE\_SPACE\_SHIP = pygame.image.load(os.path.join("assets", "blue\_ship.png"))

This image is for the PLAYER Space Ship and it’s a Yellow Space ship.

YELLOW\_SPACE\_SHIP = pygame.image.load(os.path.join("assets", "spaceship.png"))

Then images for Lasers are also added, to make the game more interactive and attractive. Total there are 4 types of LASERS, Namely: -

1. RED
2. GREEN
3. BLUE
4. YELLOW

RED\_LASER = pygame.image.load(os.path.join("assets", "pixel\_laser\_red.png"))

GREEN\_LASER = pygame.image.load(os.path.join("assets", "pixel\_laser\_green.png"))

BLUE\_LASER = pygame.image.load(os.path.join("assets", "pixel\_laser\_blue.png"))

YELLOW\_LASER = pygame.image.load(os.path.join("assets", "pixel\_laser\_yellow.png"))

Then we added the Background image, to make the game more appealing

BG = pygame.transform.scale(pygame.image.load(os.path.join("assets", "background.png")), (WIDTH, HEIGHT))

After adding all the images next thing, we did was to create Classes namely of the following types: -

1. Laser
2. Ship
3. Player Ship
4. Enemy Ship

Class 1- LASER: -

This is the class for LASERs, it has the following functions: -

1. \_\_init\_\_
2. draw
3. move
4. off\_screen
5. collision

class Laser:

    def \_\_init\_\_(self, x, y, img):

        self.x = x

        self.y = y

        self.img = img

        self.mask = pygame.mask.from\_surface(self.img)

    def draw(self, window):

        window.blit(self.img, (self.x, self.y))

    def move(self, vel):

        self.y += vel

    def off\_screen(self, height):

        return not(self.y <= height and self.y >= 0)

    def collision(self, obj):

        return collide(self, obj)

Class 2- SHIP: -

This is the class for SHIPs, it has the following functions: -

1. \_\_init\_\_
2. draw
3. move\_lasers
4. cooldown
5. shoot
6. get\_width
7. get\_height

class Ship:

    COOLDOWN = 30

    def \_\_init\_\_(self, x, y, health=100):

        self.x = x

        self.y = y

        self.health = health

        self.ship\_img = None

        self.laser\_img = None

        self.lasers = []

        self.cool\_down\_counter = 0

    def draw(self, window):

        window.blit(self.ship\_img, (self.x, self.y))

        for laser in self.lasers:

            laser.draw(window)

    def move\_lasers(self, vel, obj):

        self.cooldown()

        for laser in self.lasers:

            laser.move(vel)

            if laser.off\_screen(HEIGHT):

                self.lasers.remove(laser)

            elif laser.collision(obj):

                obj.health -= 10

                self.lasers.remove(laser)

    def cooldown(self):

        if self.cool\_down\_counter >= self.COOLDOWN:

            self.cool\_down\_counter = 0

        elif self.cool\_down\_counter > 0:

            self.cool\_down\_counter += 1

    def shoot(self):

        if self.cool\_down\_counter == 0:

            laser = Laser(self.x, self.y, self.laser\_img)

            self.lasers.append(laser)

            self.cool\_down\_counter = 1

    def get\_width(self):

        return self.ship\_img.get\_width()

    def get\_height(self):

        return self.ship\_img.get\_height()

What we just did was create these move lasers, the idea was that these are moving the lasers so we call this every loop on all our enemies and on all our players or our player so that they can handle the movement of their own lasers notice that in the draw we're drawing all the laser so this should show up on the screen.

What we do is inside of cooldown we're making sure we're not shooting too fast and that we have at least a half a second delay before we can shoot that's what this does here we make sure that the cooldown counter is equal to zero then we make a laser and add it to the list if we press space right.

If we call this method if we shoot so then we set the cooldown counter to 1 the cooldown counter goes through does its loop and then it lets us shoot again once it resets to zero so here this is the move laser for the player this is a little bit different what this is doing is checking if the laser collides with every single enemy which we pass in as OBJ's right if it does we remove that objects remove that enemy and we remove the laser.

Class 3- PLAYER SHIP: -

This is the class for PLAYER SHIP, it has the following functions: -

1. \_\_init\_\_
2. move\_lasers
3. draw
4. healthbar

class Player(Ship):

    def \_\_init\_\_(self, x, y, health=100):

        super().\_\_init\_\_(x, y, health)

        self.ship\_img = YELLOW\_SPACE\_SHIP

        self.laser\_img = YELLOW\_LASER

        self.mask = pygame.mask.from\_surface(self.ship\_img)

        self.max\_health = health

    def move\_lasers(self, vel, objs):

        self.cooldown()

        for laser in self.lasers:

            laser.move(vel)

            if laser.off\_screen(HEIGHT):

                self.lasers.remove(laser)

            else:

                for obj in objs:

                    if laser.collision(obj):

                        objs.remove(obj)

                        if laser in self.lasers:

                            self.lasers.remove(laser)

    def draw(self, window):

        super().draw(window)

        self.healthbar(window)

    def healthbar(self, window):

        pygame.draw.rect(window, (255,0,0), (self.x, self.y + self.ship\_img.get\_height() + 10, self.ship\_img.get\_width(), 10))

        pygame.draw.rect(window, (0,255,0), (self.x, self.y + self.ship\_img.get\_height() + 10, self.ship\_img.get\_width() \* (self.health/self.max\_health), 10))

Class 4- ENEMY SHIP: -

This is the class for ENEMY SHIP, it has the following functions: -

1. \_\_init\_\_
2. move
3. shoot
4. collide

class Enemy(Ship):

    COLOR\_MAP = {

                "red": (RED\_SPACE\_SHIP, RED\_LASER),

                "green": (GREEN\_SPACE\_SHIP, GREEN\_LASER),

                "blue": (BLUE\_SPACE\_SHIP, BLUE\_LASER)

                }

    def \_\_init\_\_(self, x, y, color, health=100):

        super().\_\_init\_\_(x, y, health)

        self.ship\_img, self.laser\_img = self.COLOR\_MAP[color]

        self.mask = pygame.mask.from\_surface(self.ship\_img)

    def move(self, vel):

        self.y += vel

    def shoot(self):

        if self.cool\_down\_counter == 0:

            laser = Laser(self.x-20, self.y, self.laser\_img)

            self.lasers.append(laser)

            self.cool\_down\_counter = 1

def collide(obj1, obj2):

    offset\_x = obj2.x - obj1.x

    offset\_y = obj2.y - obj1.y

    return obj1.mask.overlap(obj2.mask, (offset\_x, offset\_y)) != None

This is our MAIN () function of the Program, We have defined the FPS(Frames Per Second) =60, our game start from level 0 and give a total of 5 LIVES to the users to play and once these LIVES become 0 (Zero),the game comes to an end and a “Game Over ” message is displayed.

def main():

    run = True

    FPS = 60

    level = 0

    lives = 5

    main\_font = pygame.font.SysFont("comicsans", 50)

    lost\_font = pygame.font.SysFont("comicsans", 60)

    enemies = []

    wave\_length = 5

    enemy\_vel = 1

    player\_vel = 5

    laser\_vel = 5

The Player can restart the game after that from fresh. The Enemy Velocity is set to 1 and Player Velocity is set to 5, this helps the players to move quickly and defend itself from the enemy. The velocity of the laser is set to be same as that of player i.e., 5.

    player = Player(300, 630)

    clock = pygame.time.Clock()

    lost = False

    lost\_count = 0

    def redraw\_window():

        WIN.blit(BG, (0,0))

        lives\_label = main\_font.render(f"Lives: {lives}", 1, (255,255,255))

        level\_label = main\_font.render(f"Level: {level}", 1, (255,255,255))

        WIN.blit(lives\_label, (10, 10))

        WIN.blit(level\_label, (WIDTH - level\_label.get\_width() - 10, 10))

        for enemy in enemies:

            enemy.draw(WIN)

        player.draw(WIN)

        if lost:

            lost\_label = lost\_font.render("Thanks For Playing but Game Over !!!", 1, (255,255,255))

            WIN.blit(lost\_label, (WIDTH/2 - lost\_label.get\_width()/2, 350))

        pygame.display.update()

    while run:

        clock.tick(FPS)

        redraw\_window()

        if lives <= 0 or player.health <= 0:

            lost = True

            lost\_count += 1

        if lost:

            if lost\_count > FPS \* 3:

                run = False

            else:

                continue

        if len(enemies) == 0:

            level += 1

            wave\_length += 5

            for i in range(wave\_length):

                enemy = Enemy(random.randrange(50, WIDTH-100), random.randrange(-1500, -100), random.choice(["red", "blue", "green"]))

                enemies.append(enemy)

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                quit()

        keys = pygame.key.get\_pressed()

        if keys[pygame.K\_a] and player.x - player\_vel > 0: # left

            player.x -= player\_vel

        if keys[pygame.K\_d] and player.x + player\_vel + player.get\_width() < WIDTH:

            player.x += player\_vel

        if keys[pygame.K\_w] and player.y - player\_vel > 0:

            player.y -= player\_vel

        if keys[pygame.K\_s] and player.y + player\_vel + player.get\_height() + 15 < HEIGHT:

            player.y += player\_vel

        if keys[pygame.K\_SPACE]:

            player.shoot()

        for enemy in enemies[:]:

            enemy.move(enemy\_vel)

            enemy.move\_lasers(laser\_vel, player)

            if random.randrange(0, 2\*60) == 1:

                enemy.shoot()

            if collide(enemy, player):

                player.health -= 10

                enemies.remove(enemy)

            elif enemy.y + enemy.get\_height() > HEIGHT:

                lives -= 1

                enemies.remove(enemy)

        player.move\_lasers(-laser\_vel, enemies)

The MAIN\_MENU () function is a menu function which display the msg “**Press the mouse to begin...**”, and if the mouse is pressed then the game eventually starts.Else, the game donot start and the Value FALSE is assigned to the RUN variable.

def main\_menu():

    title\_font = pygame.font.SysFont("comicsans", 70)

    run = True

    while run:

        WIN.blit(BG, (0,0))

        title\_label = title\_font.render("Welcome to the Guardian of the Galaxy...", 1, (255,255,255))

        WIN.blit(title\_label, (WIDTH/2 - title\_label.get\_width()/2, 350))

        pygame.display.update()

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                run = False

            if event.type == pygame.MOUSEBUTTONDOWN:

                main()

    pygame.quit()

main\_menu()

**SNIPPETS FROM THE GAMEPLAY**











**Objectives**

The Main objective for developing this game are:

* To learn the basic concepts of computer graphics
* To learn the mechanism of 2D game.
  1. Advantages of Proposed Algorithm

The game we have developed is simple and easy to understand. We have used Python3 to develop this game and have used Pygame module. We used Python3 because of its various advantages like easy to use, versatile, well-structured and readable. Python has a large support and is free to use. The most significant advantage of using python is it has numerous modules which can be easily downloaded and added to our program whereas C++ can be a headache. C++ needs special IDE’s and development tools. The code we have written is maintainable, portable, scalable, and reusable.

**CONCLUSION AND FUTURE ENHANCEMENT**

The game was successfully implemented, it fulfils all the basic requirements for a single player game. The implementation and design of the system was simple and can be easily modified. The game resolutions can be modified from the code. The game can be further improved for multi-players. It can be modified to give a better user interface. More levels can be added in the game. A menu can also be added for providing different difficulty levels (Easy, Medium, Hard). More spaceships can also be added to provide a better gaming experience.

**REFERENCES**

<https://www.pygame.org/>

<https://www.geeksforgeeks.org/introduction-to-computer-graphics/>

<https://pythonprogramming.net/pygame-python-3-part-1-intro/>

<https://www.javatpoint.com/pygame>

<https://www.researchgate.net/publication/343319593_THE_DESIGN_AND_IMPLEMENTATION_OF_THE_2D_ARCADE_GAME_DUAL_SPACE_INVADERS>