SKY WARS

A Course Based Project Report Submitted in partial fulfillment of the requirements for the award of the degree of

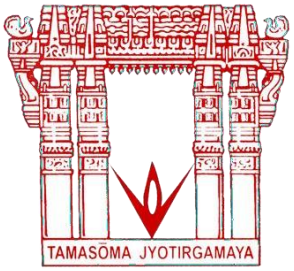
BACHELOR OF TECHNOLOGY IN
CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

Submitted by

Devanaboina Sanjana	21071A6682
Devvrat Yadav	21071A6683
G.Santhosh	21071A6686
Om Sai Thrishal	21071A66A1
P.Sai Teja	21071A66B6

Under the guidance of

P.Nethrasri
(Assistant Professor, Department of CSE-AIML & IoT, VNR VJIET)



DEPARTMENT OF

CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses Approved by AICTE, New Delhi,
Affiliated to JNTUH Recognized as "College with Potential for Excellence"
by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

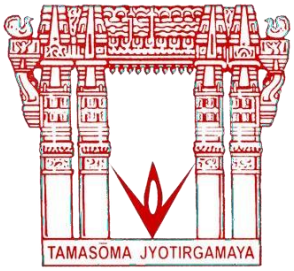
Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

DEPARTMENT OF
CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)



CERTIFICATE

This is to certify that the project report entitled "**SKY WARS"** is a bonafide work done under our supervision and is being submitted by **Ms. D. Sanjana (21071A6682), Mr. Devvrat Yadav (21071A6683), Mr. G. Santhosh (21071A6686), Mr.Om Sai Thrishal (21071A66A1), Mr. P. Sai Teja (21071A66B6)** in partial fulfillment for the award of the degree of Bachelor of Technology in CSE (Artificial Intelligence & Machine Learning), of the VNRVJIET, Hyderabad during the academic year 2022-2023. Certified further that to the best of our knowledge the work presented in this thesis has not been submitted to any other University or Institute for the award of any Degree or Diploma.

P.Nethrasri
Assistant Professor
CSE-AIML&IoT

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

DEPARTMENT OF
CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

DECLARATION

We declare that the major project work entitled "**SKY WARS**" submitted in the department of CSE(Artificial Intelligence & Machine Learning), Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **CSE(Artificial Intelligence & Machine Learning)** is a bonafide record of our own work carried out under the supervision of **P.Nethrasri, Assistant Professor**. Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree/diploma of any other institution or university previously.

Place: Hyderabad

D.Sanjana	**Devvrat Yadav**	**G.Santosh**	**Om Sai Thrishal**	**P.Sai Teja**
(21071A66682)	(21071A6683)	(21071A6686)	(21071A66A1)	(21071A66B6)

ABSTRACT

This is a simple game project, developed using the versatile Python programming language, offers a fun and easy experience for players of all ages. The game features straightforward gameplay mechanics and colorful graphics, making it an excellent choice for players of all ages and skill levels. By utilizing Python's well-established libraries, such as Ursina, this project delivers a smooth and enjoyable gaming experience. Whether you're looking for a quick distraction during a break or a way to pass the time on a rainy day, Python Simple Game is the perfect choice. This game project is a great introduction to game development and a fun way to learn programming. Get ready for some quick and easy fun with Python Simple Game! So grab a friend, sit back, and get ready to have some fun!

Table of Contents Page No

1. INTRODUCTION

This Python arcade game, built using the Ursina engine, takes players on a thrilling adventure through the sky. Players control a character and must dodge incoming obstacles while collecting points along the way. With simple yet addictive gameplay mechanics and stunning graphics, players will be fully immersed in this virtual world. The game uses keypresses to control character movement and shooting, allowing players to engage with the game in a fun and intuitive way. So buckle up, grab your keyboard, and get ready for an exciting journey through the sky with Python Arcade Game!

2.OBJECTIVE

The objective of the code is to create a simple game in Python using the Ursina engine. The code uses the Ursina engine to define an application window, create a player character, add obstacles in the form of flies, and allow the player to shoot at the flies. The code uses the time and random modules to generate flies at random intervals and provide a randomized starting position for each fly. The player character can be moved using the 'W', 'A', 'S', and 'D' keys and can shoot at the flies by pressing the 'space' key. The code also has a quit function that is triggered if the player character collides with a fly. The objective of the game is for the player to avoid the flies and shoot them before they reach the left side of the screen.

3.DESIGN

3.1. REQUIREMENTS

1. Python: This code requires Python 3.x installed on the system.
2. Ursina: The code uses the Ursina engine to build the game, so Ursina must be installed in the system.
3. Graphics card: The code requires a graphics card with good performance to display graphics and animations smoothly.
4. Operating System: The code is written in a way that it should work on most operating systems including Windows, MacOS, and Linux.
5. Hard Disk: A minimum of 50 MB of free space on the hard disk is required to install the required libraries and run the code.
6. RAM: The code requires a minimum of 2 GB of RAM to run smoothly, but 4 GB or higher is recommended for better performance.

3.2. DIAGRAM

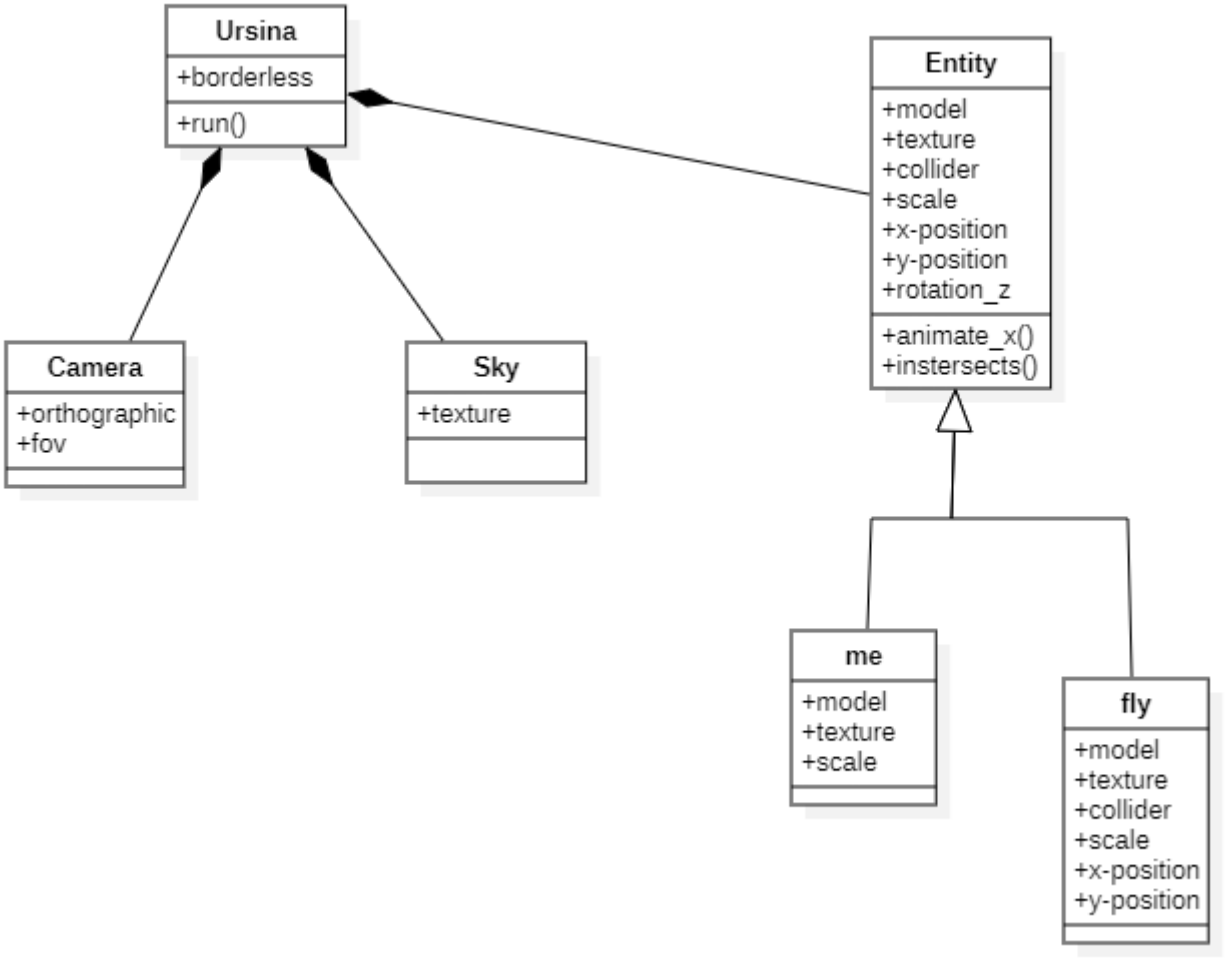


FIG.1. CLASS DIAGRAM

4. IMPLEMENTATION

4.1. CODE

```
from ursina import *
import random
import time

app=Ursina(borderless=False)
camera.orthographic=True
camera.fov=20
me = Entity(
    model='quad',
    texture='assets\pl',
    scale=(3,2),
)

skybox_image = load_texture("assets\sky.png")
Sky(texture=skybox_image)

fly=Entity(
    model='cube',
    texture='assets\\mon',
    collider='box',
    scale=4,
    x=20,
    y=-10
)
flies=[]
def newFly():
    new=duplicate(
        fly,
        y=-5+(5124*time.dt)%15
    )
    flies.append(new)
    invoke(newFly,delay=1)
newFly()
```

```
def update():
    me.x -=held_keys['a']*6*time.dt
    me.x +=held_keys['d']*6*time.dt
    me.y -=held_keys['s']*6*time.dt
    me.y +=held_keys['w']*6*time.dt
    b=held_keys['s']*20
    a=held_keys['w']*-20
    if(a!=0):
        me.rotation_z=a
    else:
        me.rotation_z=b
    for fly in flies:
        fly.x-=4*time.dt
        touch=fly.intersects()
        if touch.hit:
            flies.remove(fly)
            destroy(fly)
        #t=me.intersects(fly)
        if me.intersects().hit==True:
            quit()

def input(key):
    if(key=='space'):
        e=Entity(
            y=me.y,
            x=me.x+2,
            model='cube',
            texture='assets\Bul',
            collider='cube'
        )
        e.animate_x(
            30,
            duration=2,
            curve=curve.linear
        )
        invoke(destroy,e,delay=2)

app.run()
```

4.2. OUTPUT

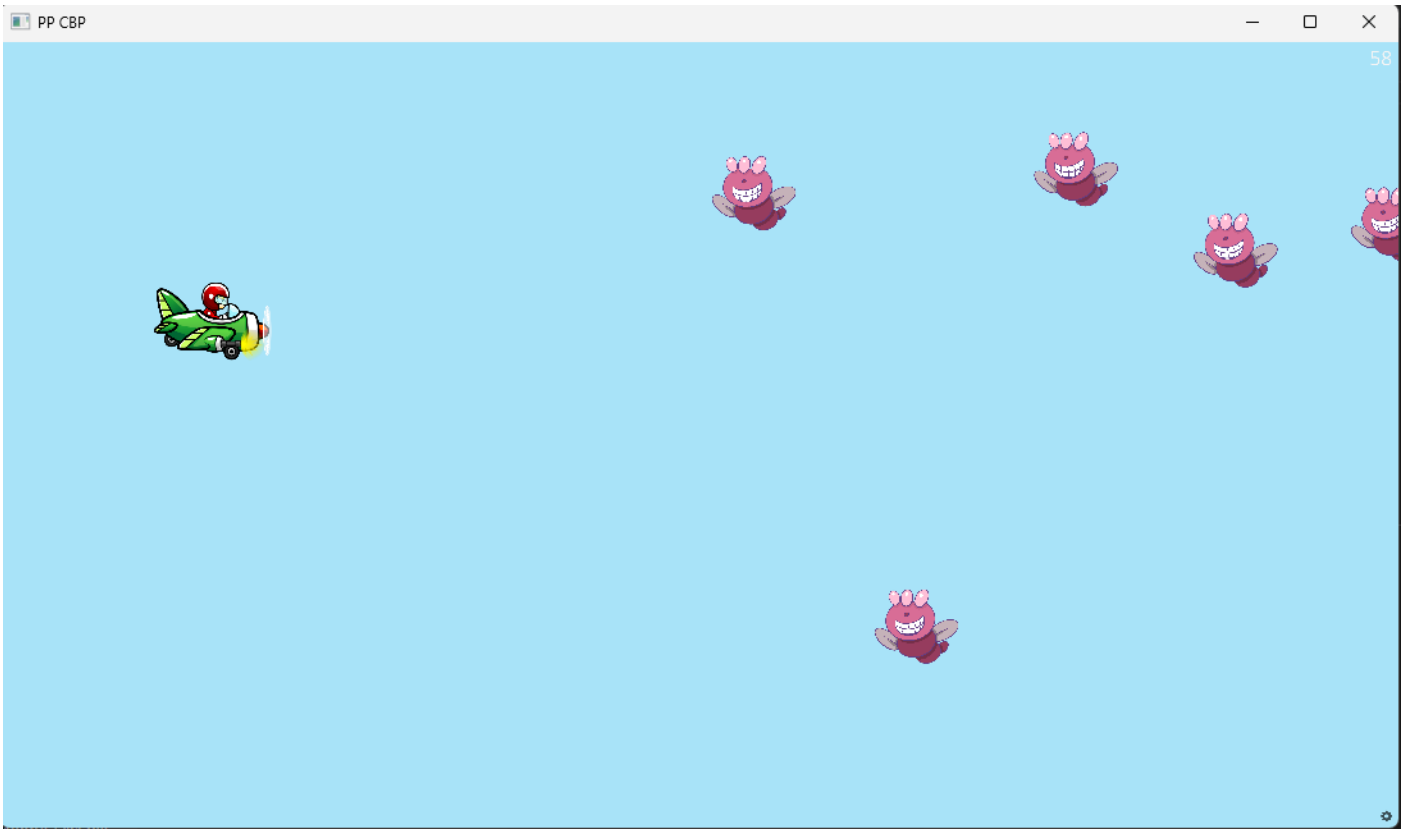


FIG.2.

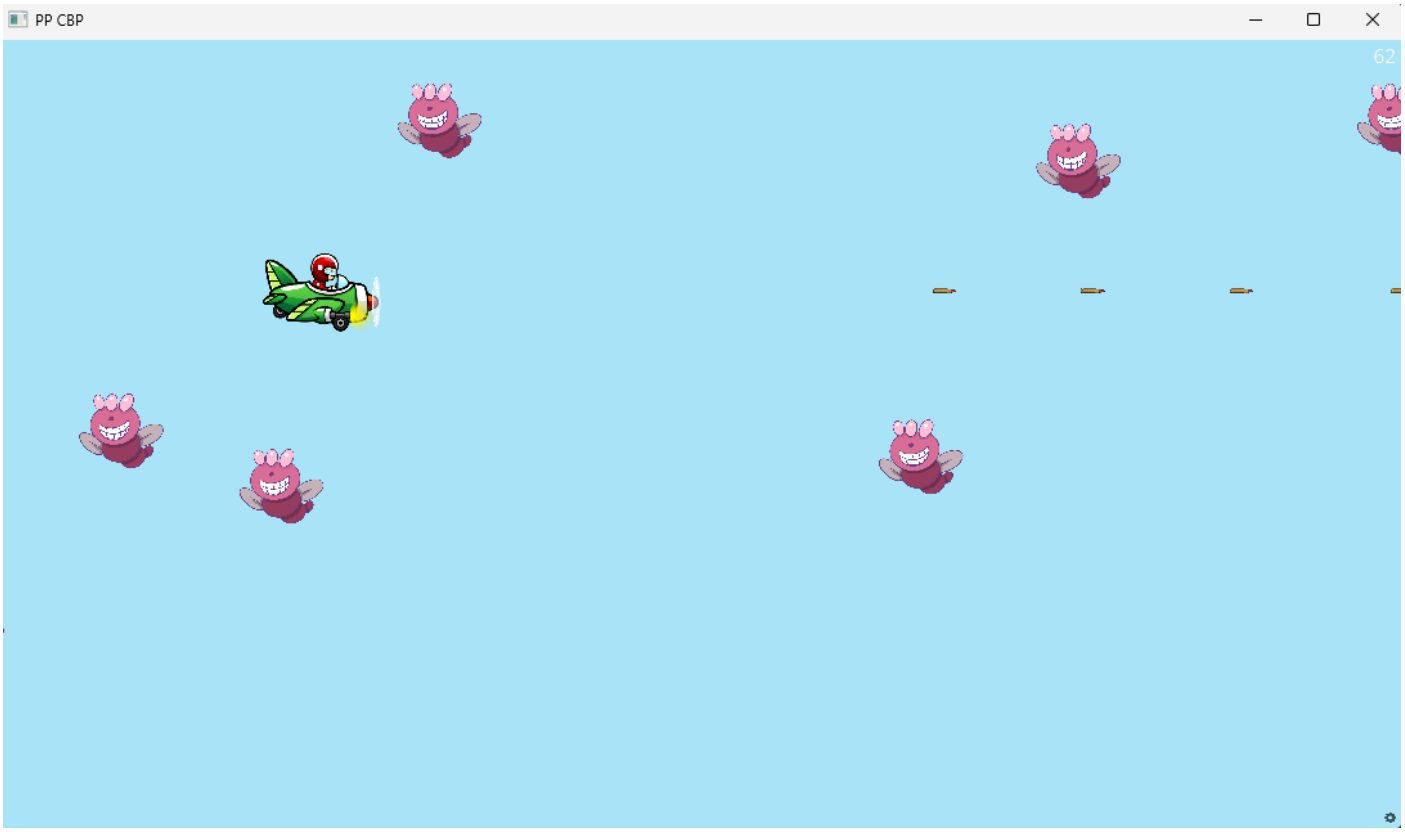


FIG.3.

4.3 COMPONENTS

4.3.1

FIG.4

4.3.2

FIG.5

4.3.3

FIG.6

5. TESTING

TEST CASES	RESULT
Movement: Test the movement of the character in all four directions using the 'W', 'A', 'S', 'D' keys. The character should move smoothly in all directions and should not go out of the screen.	PASS
Shooting: Test the shooting functionality by pressing the 'space' key. A bullet should be fired and should move across the screen until it reaches the end.	PASS
Obstacle collision: Test the obstacle collision by creating a fly and checking if the character collides with it. The game should quit when the character collides with the fly.	FAIL
Fly generation: Test the fly generation by checking if new flies are generated at regular intervals. The flies should appear at different heights and should move across the screen.	PASS
Fly destruction: Test the destruction of the flies by checking if the flies disappear from the screen when they reach the end.	PASS
Graphics: Test the graphics by checking if the skybox, character, flies, and bullets are displayed correctly and have the correct texture.	PASS
Orthographic camera: Test the orthographic camera by checking if the camera displays the correct perspective and does not distort the objects on the screen.	PASS
FOV: Test the field of view (FOV) by checking if the camera has the correct FOV set to 20.	PASS
Input function: Test the input function by pressing the 'W', 'A', 'S', 'D', and 'space' keys and verifying that the expected actions are performed.	PASS

6. RESULTS

The result of the above code is a simple game where the player must control a character and avoid incoming flies while shooting them. When the code is executed, a window will appear with a player character and randomly generated flies. The player can move the character using the 'W', 'A', 'S', and 'D' keys and shoot at the flies by pressing the 'space' key. As the game progresses, more flies will be generated, and the player must avoid them while shooting them. If the player character collides with a fly, the game will quit. The result of the code is a simple game that provides a basic demonstration of game development using the Ursina engine and Python.

7. CONCLUSION

In conclusion, the code provides a basic demonstration of game development using the Ursina engine and Python. It creates a simple game where the player must control a character and avoid incoming flies while shooting them. The code demonstrates the use of various functions and modules, such as time and random, to generate obstacles and provide a randomized starting position for each fly. The code also demonstrates the use of keyboard input to control the player character and shoot at the flies. The result of the code is a simple game that can serve as a starting point for further development or as a demonstration of game development using Python and the Ursina engine.

8.FUTURE SCOPE

Here are some potential areas for future expansion for the code:

1. **Increased game complexity:** More game elements can be added, such as power-ups, multiple types of obstacles, or enemies, to increase the complexity of the game.
2. **Improved graphics:** The graphics can be improved by adding more detailed textures, animations, and special effects.
3. **Sound effects:** Sound effects can be added to enhance the user experience and make the game more engaging.
4. **Multiplayer:** Multiplayer functionality can be added so that multiple players can play the game simultaneously.
5. **Scoring system:** A scoring system can be added to keep track of the player's progress and to make the game more challenging.
6. **Different levels:** Different levels with increasing difficulty can be added to the game.
7. **Customizable character:** A character customization system can be added so that players can personalize their characters.
8. **Improved movement:** The movement of the character can be improved by adding jumping, crouching, and other actions.
9. **Improved shooting:** The shooting functionality can be improved by adding different types of bullets and weapons.
10. **Mobile compatibility:** The game can be optimized for mobile devices and ported to Android and iOS platforms.

These are just a few potential areas for future expansion. There are many other possibilities, and the future scope of the code will depend on the specific requirements and goals of the project.

REFERENCES

- **https://www.instagram.com/pythonlearnerr/**
- **https://www.ursinaengine.org/documentation.html**