DRAGON BALL - Z

A MINI PROJECT REPORT

Submitted by

Vijai K S 231501509

In partial fulfillment for the award of the degree of

BACHELOR OF

ENGINEERING

IN

Artificial Intelligence And Machine Learning

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2024 - 2025

BONAFIDE CERTIFICATE

Certified that this project report "**Dragon BALL - Z**" is the bonafide

work of **"SURYA M V (231501166), THIRUMALAI J (231501174), VIJAI**

K S (231501509)"

who carried out the project work under my
supervision.

Submitted for the Practical Examination held on _____

SIGNATURE **SIGNATURE**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

PROJECT SYNOPSIS

Background :

This project involves the creation of a simple 2D game inspired by the popular Dragon Ball franchise, developed using Python and the Pygame library. The primary objective of the game is for the player to control Goku (or another character) to catch falling Dragon Balls. The game operates within a basic environment where the player moves left and right at the bottom of the screen to catch the balls, earning points for each Dragon Ball caught. The game features a simple collision detection system and a basic scoring mechanism.

SYSTEM REQUIREMENTS

SYSTEM:

OS- Window 10 Home Single Language

(19045.2364 OS Build)

Language: English

Processor: Intel(R) Core(TM) i3-4005U CPU @ 1.70GHz 1.70 GHz

Memory: 8.00 GB RAM

SOFTWARE:

SQLite Community server 3.47.0

Python IDLE version 3.10.4

FUNCTIONS AND MODULES USED

Functions:

pygame.init() :

This function must be called before using any Pygame functionality.

pygame.display.set_mode() :

Used to create the main game window that will display all the graphics.

pygame.display.set_caption() :

It displays the name of the game in the title bar of the window.

pygame.Surface() :

Used to create the image for Goku and the Dragon Balls.

pygame.sprite.Sprite() :

Inherit from this class to create custom objects like Goku and Dragon Balls, which can be updated and drawn to the screen.

pygame.key.get_pressed() :

Used to check if the player presses the left or right arrow keys to move Goku.

pygame.event.get() :

Used to check for events like quitting the game or key presses.

pygame.sprite.spritecollide() :

Used to detect when Goku (the player) catches a Dragon Ball by checking for collisions between their sprites.

pygame.font.SysFont() :

Used to display the score on the screen.

screen.fill() :

Used to clear the screen at the beginning of each frame before drawing the updated game elements.

screen.blit() :

Used to display the player's sprite and other game elements on the screen.

pygame.time.Clock() :

Used to set the frames per second (FPS) to control how fast the game updates.

Goku.update() :

Called every frame to check if the player presses the arrow keys and move Goku accordingly.

DragonBall.update() :

Called every frame to make Dragon Balls fall, and reset their position once they go off-screen.

game() :

This function manages the flow of the game, including setting up sprites, updating game logic, and rendering the display.

spawn_dragon_ball() :

Generates a new Dragon Ball at a random position after one is caught by Goku.

game_over() :

Can be used to end the game, display the player's score, and optionally restart or quit the game.

Modules :

import pygame :

Main library for creating the game, handling graphics, sprites, input, timing, and events.

import random :

Used for random number generation, essential for randomizing the position, speed, and behavior of Dragon Balls and other game elements.

import sqlite3 :

Provides functionality for connecting to and interacting with an SQLite database to store and retrieve persistent data like high scores.

import pygame.time :

Part of Pygame, used to manage the game's timing and frame rate, and add delays (e.g., for the game-over screen).

import pygame.font :

Used to render text on the screen, such as scores, game-over messages, and other text-based feedback.

import math :

The math module in Python provides mathematical functions and constants that are useful for performing complex calculations in your game. Here's a summary of how math can be applied to enhance the functionality and dynamics of your Dragon Ball Catcher Game.

import tkinter :

Tkinter is the inbuilt python module that is used to create GUI applications.

USE OF TECHNOLOGY

SQLite :

SQLite is a popular embedded relational database management system (RDBMS) that is widely used in applications where a full-fledged database server is not required. It is serverless, self-contained, and can run directly from disk or in-memory. SQLite does not require a separate server process, making it easy to integrate into applications. The database is stored in a single file, which is often smaller than other databases. No installation or configuration is needed, making it easy to deploy and use. It supports most of the SQL standards, though it may not implement every feature found in larger RDBMS systems. SQL commands can be divided into following categories:

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Transaction Control Language (TCL)
- Session Control Commands
- System Control Commands Page

Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It is high-level built in data structures, combined with dynamic typing and dynamic binding which make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. Often, programmers fall in love with Python because of the increased productivity it provides. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

CODE

Main program code :

```
import
pygame
import
random
import math
import
sqlite3


#
Initialize
Pygame
pygame.init
()


# Set up
the display
WIDTH,
HEIGHT =
800,600
screen =
pygame.disp
lay.set_mod
```

```
e((WIDTH,
HEIGHT))
pygame.disp
lay.set_cap
tion("Epic
Turn-Based
Game")

# Colors
WHITE =
(255, 255,
255)
BLACK = (0,
0, 0)
RED = (255,
0, 0)
GREEN = (0,
255, 0)
BLUE = (0,
0, 255)
YELLOW =
(255, 255,
0)

# Fonts
font =
pygame.font
.Font(None,
36)
big_font =
pygame.font
```

```
.Font(None,
72)

# Database
setup
conn =
sqlite3.con
nect('game_
results.db'
)
c =
conn.cursor
()
c.execute('
''CREATE
TABLE IF
NOT EXISTS
players

(name TEXT
PRIMARY
KEY, wins
INTEGER,
losses
INTEGER,
level
INTEGER)'''
)
conn.commit
()
```

```
# Load
background
image
background_
image =
pygame.imag
e.load('bac
kground.png
')
background_
image =
pygame.tran
sform.scale
(background
_image,
(WIDTH,
HEIGHT))


class
Character:
    def
__init__(se
lf, name,
x, y,
color,
image_path,
is_player=F
alse):

self.name =
name
```

```
self.hp =
100

self.max_hp
= 100

self.level
= 1

self.exp =
0

self.x = x

self.y = y

self.origin
al_x = x

self.origin
al_y = y

self.color
= color

self.width
= 50

self.height
= 100
```

```
self.is_hit
= False

self.image
=
pygame.imag
e.load(imag
e_path)

self.image
=
pygame.tran
sform.scale
(self.image
,
(self.width
,
self.height
))

self.is_pla
yer =
is_player

self.abilit
ies = [

{"name":
"Basic
Attack",
```

"damage":
20, "heal":
0,
"cooldown":
0},

{"name":
"ka me ha
me haa",
"damage":
35, "heal":
0,
"cooldown":
2},

{"name":
"senzu
bean",
"damage":
0, "heal":
25,
"cooldown":
3},

{"name":
"sprit
bomb",
"damage":
50, "heal":
0,
"cooldown":

```
5}
        ]

self.cooldo
wns = [0,
0, 0, 0]

self.partic
les = []

self.energy
_particles
= []

self.aura_p
articles =
[]



    def
draw(self,
screen):

screen.blit
(self.image
, (self.x,
self.y))
        #
Health bar

bar_width =
```

```
100

bar_height
= 10

outline_rec
t =
pygame.Rect
(self.x -
25, self.y
- 60,
bar_width,
bar_height)

fill_rect =
pygame.Rect
(self.x -
25, self.y
- 60,
int(self.hp
/
self.max_hp
*
bar_width),
bar_height)

pygame.draw
.rect(scree
n, RED,
outline_rec
t)
```

```
pygame.draw
.rect(scree
n, GREEN,
fill_rect)



health_text
=
font.render
(f"{self.na
me} HP:
{self.hp}",
True,
WHITE)

level_text
=
font.render
(f"Level
{self.level
}", True,
YELLOW)

screen.blit
(health_tex
t, (self.x
- 20,
self.y -
100))
```

```
screen.blit
(level_text
, (self.x,
self.y -
130))

        if
self.is_hit
:

pygame.draw
.rect(scree
n, RED,
(self.x -
5, self.y -
5,
self.width
+ 10,
self.height
+ 10), 3)

        #
Draw
particles
        for
particle in
self.partic
les:

particle.dr
aw(screen)
```

```
self.partic
les =
[particle
for
particle in
self.partic
les if
particle.li
fetime > 0]

        for
particle in
self.energy
_particles:

particle.dr
aw(screen)

        #
Draw aura
particles
        for
particle in
self.aura_p
articles:

particle.dr
aw(screen)
```

```
self.energy
_particles
= [p for p
in
self.energy
_particles
if
p.lifetime
> 0]

self.aura_p
articles =
[p for p in
self.aura_p
articles if
p.lifetime
> 0]

    def
take_damage
(self,
damage):

self.hp =
max(0,
self.hp -
damage)

return
self.hp <=
```

```
0

    def
heal(self,
amount):

self.hp =
min(self.ma
x_hp,
self.hp +
amount)

    def
gain_exp(se
lf,
amount):

self.exp +=
amount
        if
self.exp >=
100:

self.level_
up()

    def
level_up(se
lf):

self.level
```

```
+= 1

self.exp -=
100

self.max_hp
+= 20

self.hp =
self.max_hp
        for
ability in
self.abilit
ies:

ability["da
mage"] =
int(ability
["damage"]
* 1.1)

ability["he
al"] =
int(ability
["heal"] *
1.1)

    def
attack_anim
ation(self,
target):
```

```
frames = 60
        for
i in
range(frame
s):

progress =
i / frames

if progress
< 0.3:

# Charge up

self.charge
_up_animati
on()

elif
progress <
0.6:

# Release
energy
blast

self.energy
_blast_anim
ation(targe
t)
```

```
else:

# Impact
and
aftermath

self.impact
_animation(
target)


self.draw_f
rame(target
)

pygame.time
.delay(30)


target.is_h
it = False

    def
heal_animat
ion(self,
other_chara
cter):

frames = 60
        for
```

```
i in
range(frame
s):

progress =
i / frames

self.healin
g_aura_anim
ation()

self.draw_f
rame(other_
character)

pygame.time
.delay(30)

    def
draw_frame(
self,
other_chara
cter,
scale=1):

screen.blit
(background
_image, (0,
0))

scaled_imag
```

```
e =
pygame.tran
sform.scale
(self.image
,
(int(self.w
idth *
scale),
int(self.he
ight *
scale)))

screen.blit
(scaled_ima
ge, (self.x
-
(scaled_ima
ge.get_widt
h() -
self.width)
// 2,

self.y -
(scaled_ima
ge.get_heig
ht() -
self.height
) // 2))
        if
other_chara
cter:
```

```
other_chara
cter.draw(s
creen)

self.draw(s
creen)

draw_button
s(self)

pygame.disp
lay.flip()

    def
charge_up_a
nimation(se
lf):
        for
_ in
range(5):

angle =
random.unif
orm(0, 2 *
math.pi)

distance =
random.unif
orm(30, 50)
```

```
x = self.x
+
self.width
// 2 +
math.cos(an
gle) *
distance

y = self.y
+
self.height
// 2 +
math.sin(an
gle) *
distance

self.energy
_particles.
append(Ener
gyParticle(
x, y,
self.color)
)

    def
energy_blas
t_animation
(self,
target):

start_x =
```

```
self.x +
self.width

start_y =
self.y +
self.height
// 2

end_x =
target.x

end_y =
target.y +
target.heig
ht // 2
        for
_ in
range(10):

progress =
random.unif
orm(0, 1)

x = start_x
+ (end_x -
start_x) *
progress

y = start_y
+ (end_y -
start_y) *
```

```
progress


self.energy
_particles.
append(Ener
gyBlast(x,
y,
self.color)
)


    def
impact_anim
ation(self,
target):


target.is_h
it = True
        for
_ in
range(20):


angle =
random.unif
orm(0, 2 *
math.pi)


speed =
random.unif
orm(2, 5)


x =
```

```
target.x +
target.widt
h // 2

y =
target.y +
target.heig
ht // 2

self.energy
_particles.
append(Impa
ctParticle(
x, y,
self.color,
angle,
speed))

    def
healing_aur
a_animation
(self):
        for
_ in
range(5):

angle =
random.unif
orm(0, 2 *
math.pi)
```

```
distance =
random.unif
orm(0,
self.width
// 2)

x = self.x
+
self.width
// 2 +
math.cos(an
gle) *
distance

y = self.y
+
self.height
+
math.sin(an
gle) *
distance

self.aura_p
articles.ap
pend(AuraPa
rticle(x,
y, GREEN))



class
Particle:
```

```
    def
__init__(se
lf, x, y,
color,
move_up=Fal
se):

self.x = x

self.y = y

self.color
= color

self.radius
=
random.rand
int(2, 5)

self.lifeti
me =
random.rand
int(20, 40)

self.move_u
p = move_up
        if
move_up:

self.speed
=
```

```
random.unif
orm(1, 3)

self.angle
=
random.unif
orm(-0.5,
0.5)

else:

self.speed
=
random.unif
orm(2, 5)

self.angle
=
random.unif
orm(0, 2 *
math.pi)

    def
draw(self,
screen):

self.lifeti
me -= 1
        if
self.move_u
p:
```

```
self.y -=
self.speed

self.x +=
math.sin(se
lf.angle) *
0.5

else:

self.x +=
math.cos(se
lf.angle) *
self.speed

self.y +=
math.sin(se
lf.angle) *
self.speed

pygame.draw
.circle(scr
een,
self.color,
(int(self.x
),
int(self.y)
),
self.radius
)
```

```
    def
take_damage
(self,
damage):

self.hp =
max(0,
self.hp -
damage)

return
self.hp <=
0

    def
heal(self,
amount):

self.hp =
min(self.ma
x_hp,
self.hp +
amount)

    def
gain_exp(se
lf,
amount):

self.exp +=
```

```
amount
        if
self.exp >=
100:

self.level_
up()

    def
level_up(se
lf):

self.level
+= 1

self.exp -=
100

self.max_hp
+= 20

self.hp =
self.max_hp
        for
ability in
self.abilit
ies:

ability["da
mage"] =
int(ability
```

```
["damage"]
* 1.1)

ability["he
al"] =
int(ability
["heal"] *
1.1)

class
EnergyParti
cle:
    def
__init__(se
lf, x, y,
color):

self.x = x

self.y = y

self.color
= color

self.size =
random.rand
int(2, 5)

self.lifeti
me =
random.rand
```

```
int(10, 20)

    def
draw(self,
screen):

self.lifeti
me -= 1

pygame.draw
.circle(scr
een,
self.color,
(int(self.x
),
int(self.y)
),
self.size)

class
EnergyBlast
:
    def
__init__(se
lf, x, y,
color):

self.x = x

self.y = y
```

```
self.color
= color

self.size =
random.rand
int(5, 10)

self.lifeti
me =
random.rand
int(20, 30)

    def
draw(self,
screen):

self.lifeti
me -= 1

pygame.draw
.circle(scr
een,
self.color,
(int(self.x
),
int(self.y)
),
self.size)

class
ImpactParti
```

```
cle:
    def
__init__(se
lf, x, y,
color,
angle,
speed):

self.x = x

self.y = y

self.color
= color

self.angle
= angle

self.speed
= speed

self.size =
random.rand
int(2, 5)

self.lifeti
me =
random.rand
int(20, 30)

    def
```

```
draw(self,
screen):

self.lifeti
me -= 1

self.x +=
math.cos(se
lf.angle) *
self.speed

self.y +=
math.sin(se
lf.angle) *
self.speed

pygame.draw
.circle(scr
een,
self.color,
(int(self.x
),
int(self.y)
),
self.size)

class
AuraParticl
e:
    def
__init__(se
```

```
lf, x, y,
color):

self.x = x

self.y = y

self.color
= color

self.size =
random.rand
int(2, 5)

self.lifeti
me =
random.rand
int(20, 30)

self.speed
=
random.unif
orm(1, 2)

    def
draw(self,
screen):

self.lifeti
me -= 1
```

```
self.y -=
self.speed

self.x +=
random.unif
orm(-0.5,
0.5)

pygame.draw
.circle(scr
een,
self.color,
(int(self.x
),
int(self.y)
),
self.size)

def
create_enem
y(player_le
vel):
    enemy =
Character("
Enemy",
650, 400,
RED,
'enemy.png'
)

enemy.level
```

```
= max(1,
player_leve
l - 1)  #
Enemy level
is player
level - 1,
but at
least 1

    # Scale
enemy stats
based on
level

enemy.max_h
p = 100 +
(enemy.leve
l - 1) * 20

enemy.hp =
enemy.max_h
p

    for
ability in
enemy.abili
ties:

ability["da
mage"] =
int(ability
```

```
["damage"]
* (1 + 0.1
*
(enemy.leve
l - 1)))

ability["he
al"] =
int(ability
["heal"] *
(1 + 0.1 *
(enemy.leve
l - 1)))

    return
enemy

def
draw_button
(screen,
text, x, y,
width,
height,
color,
text_color=
BLACK):

pygame.draw
.rect(scree
n, color,
(x, y,
```

```
width,
height))

pygame.draw
.rect(scree
n, WHITE,
(x, y,
width,
height), 2)

text_surfac
e =
font.render
(text,
True,
text_color)

text_rect =
text_surfac
e.get_rect(
center=(x +
width // 2,
y + height
// 2))

screen.blit
(text_surfa
ce,
text_rect)
    return
pygame.Rect
```

```
(x, y,
width,
height)

def
draw_button
s(player):
    buttons
= []
    for i,
ability in
enumerate(p
layer.abili
ties):

color =
GREEN if
player.cool
downs[i] ==
0 else RED

button =
draw_button
(screen,
ability["na
me"], 50 +
i*180, 500,
170, 50,
color,
WHITE)
```

```
buttons.app
end(button)
    return
buttons

def
show_messag
e(message,
color=BLACK
):
    text =
big_font.re
nder(messag
e, True,
color)

text_rect =
text.get_re
ct(center=(
WIDTH // 2,
HEIGHT //
2))

screen.blit
(text,
text_rect)

pygame.disp
lay.flip()

pygame.time
```

```
.delay(1000
)

def
get_user_in
put():

input_box =
pygame.Rect
(WIDTH // 2
- 100,
HEIGHT // 2
- 16, 200,
32)

color_inact
ive =
pygame.Colo
r('lightsky
blue3')

color_activ
e =
pygame.Colo
r('dodgerbl
ue2')
    color =
color_inact
ive
    active
= False
```

```
    text =
''
    done =
False

    while
not done:
        for
event in
pygame.even
t.get():

if
event.type
==
pygame.QUIT
:

pygame.quit
()

return None

if
event.type
==
pygame.MOUS
EBUTTONDOWN
:

if
```

```
input_box.c
ollidepoint
(event.pos)
:

active =
not active

else:

active =
False

color =
color_activ
e if active
else
color_inact
ive

if
event.type
==
pygame.KEYD
OWN:

if active:

if
event.key
==
```

```
pygame.K_RE
TURN:

done = True

elif
event.key
==
pygame.K_BA
CKSPACE:

text =
text[:-1]

else:

text +=
event.unico
de



screen.fill
(BLACK)

txt_surface
=
font.render
(text,
True,
color)
```

```
width =
max(200,
txt_surface
.get_width(
) + 10)

input_box.w
= width

screen.blit
(txt_surfac
e,
(input_box.
x + 5,
input_box.y
+ 5))

pygame.draw
.rect(scree
n, color,
input_box,
2)



prompt_text
=
font.render
("Enter
your
name:",
True,
```

```
WHITE)

screen.blit
(prompt_tex
t, (WIDTH
// 2 - 100,
HEIGHT // 2
- 50))



pygame.disp
lay.flip()

    return
text

def
update_play
er_record(n
ame, won):

c.execute("
SELECT *
FROM
players
WHERE
name=?",
(name,))
    player
=
c.fetchone(
```

```
)
    if
player:
        if
won:

c.execute("
UPDATE
players SET
wins = wins
+ 1, level
= ? WHERE
name=?",
(player[3]
+ 1, name))

else:

c.execute("
UPDATE
players SET
losses =
losses + 1
WHERE
name=?",
(name,))
    else:
        if
won:

c.execute("
```

```
INSERT INTO
players
VALUES (?,
1, 0, 1)",
(name,))

else:

c.execute("
INSERT INTO
players
VALUES (?,
0, 1, 1)",
(name,))

conn.commit
()



def
show_player
_stats(name
):

c.execute("
SELECT *
FROM
players
WHERE
name=?",
(name,))
```

```
    player
=
c.fetchone(
)
    if
player:

stats_text
= f"Player:
{player[0]}
| Wins:
{player[1]}
| Losses:
{player[2]}
| Level:
{player[3]}
"
    else:

stats_text
= f"New
player:
{name}"

    text =
font.render
(stats_text
, True,
WHITE)

text_rect =
```

```
text.get_re
ct(center=(
WIDTH // 2,
30))

screen.blit
(text,
text_rect)

pygame.disp
lay.flip()

pygame.time
.delay(3000
)

def
show_main_m
enu(player_
name):

screen.fill
(BLACK)
    title =
big_font.re
nder(f"Welc
ome,
{player_nam
e}!", True,
WHITE)
```

```
screen.blit
(title,
(WIDTH // 2
-
title.get_w
idth() //
2, 100))



play_button
=
draw_button
(screen,
"Play",
WIDTH // 2
- 100, 250,
200, 50,
GREEN,
WHITE)

leaderboard
_button =
draw_button
(screen,
"Leaderboar
d", WIDTH
// 2 - 100,
320, 200,
50, BLUE,
WHITE)
```

```
pygame.disp
lay.flip()

    while
True:
        for
event in
pygame.even
t.get():

if
event.type
==
pygame.QUIT
:

return
"quit"

if
event.type
==
pygame.MOUS
EBUTTONDOWN
:

if
play_button
.collidepoi
nt(event.po
```

```
s):

return
"play"

elif
leaderboard
_button.col
lidepoint(e
vent.pos):

return
"leaderboar
d"

def
show_leader
board():

screen.fill
(BLACK)
    title =
big_font.re
nder("Leade
rboard",
True,
WHITE)

screen.blit
(title,
(WIDTH // 2
```

```
-
title.get_w
idth() //
2, 50))

    #
Updated SQL
query to
calculate
win
percentage

c.execute("
""SELECT
name, wins,
losses,
level, CASE
WHEN (wins
+ losses) >
0 THEN
ROUND(CAST(
wins AS
FLOAT) /
(wins +
losses) *
100, 2)ELSE
0
        END
as
win_percent
age
```

```
FROM
players

WHERE (wins
+ losses) >
0

ORDER BY
win_percent
age DESC,
level DESC,
wins DESC

LIMIT 10
    """)
    players
=
c.fetchall(
)



y_offset =
120
    for i,
player in
enumerate(p
layers, 1):

name, wins,
losses,
```

```
level,
win_percent
age =
player

player_text
= f"{i}.
{name}:
{win_percen
tage}% (W:
{wins}, L:
{losses},
Lvl:
{level})"

text_surfac
e =
font.render
(player_tex
t, True,
WHITE)

screen.blit
(text_surfa
ce, (WIDTH
// 2 -
text_surfac
e.get_width
() // 2,
y_offset))
```

```
y_offset +=
40



back_button
=
draw_button
(screen,
"Back",
WIDTH // 2
- 100, 500,
200, 50,
RED, WHITE)



pygame.disp
lay.flip()


    while
True:
        for
event in
pygame.even
t.get():


if
event.type
==
pygame.QUIT
:
```

```
return
"quit"

if
event.type
==
pygame.MOUS
EBUTTONDOWN
:

if
back_button
.collidepoi
nt(event.po
s):

return
"back"



def
play_game(p
layer_name)
:
    player
=
Character(p
layer_name,
WIDTH *
0.2, HEIGHT
* 0.6,
```

```
BLUE,
'player.png
',
is_player=T
rue)

    # Fetch
player
level from
database

c.execute("
SELECT
level FROM
players
WHERE
name=?",
(player_nam
e,))
    result
=
c.fetchone(
)
    if
result:

player.leve
l =
result[0]

    enemy =
```

```
create_enem
y(player.le
vel)
    enemy.x
= WIDTH *
0.8 -
enemy.width
    enemy.y
= HEIGHT *
0.6

    clock =
pygame.time
.Clock()

player_turn
= True

    def
end_game(wi
nner):

show_messag
e(f"{winner
} wins!",
GREEN if
winner ==
player_name
else RED)

print(f"{wi
```

```
nner}
wins!")  #
Print the
winner

update_play
er_record(p
layer_name,
winner ==
player_name
)

pygame.time
.delay(2000
)

return
False  #
Ends the
game loop

    running
= True
    while
running:
        for
event in
pygame.even
t.get():

if
```

```
event.type
==
pygame.QUIT
:

return

if
event.type
==
pygame.MOUS
EBUTTONDOWN
and
player_turn
:

mouse_pos =
pygame.mous
e.get_pos()

buttons =
draw_button
s(player)

for i,
button in
enumerate(b
uttons):

if
button.coll
```

```
idepoint(mo
use_pos)
and
player.cool
downs[i] ==
0:

ability =
player.abil
ities[i]

if
ability["da
mage"] > 0:

if
enemy.take_
damage(abil
ity["damage
"]):

player.atta
ck_animatio
n(enemy)

running =
end_game(pl
ayer_name)

else:
```

```
player.atta
ck_animatio
n(enemy)

show_messag
e(f"Player
used
{ability['n
ame']} for
{ability['d
amage']}
damage!")

if
ability["he
al"] > 0:

player.heal
(ability["h
eal"])

player.heal
_animation(
enemy)

show_messag
e(f"Player
healed for
{ability['h
eal']}
HP!")
```

```
player.cool
downs[i] =
ability["co
oldown"]

player_turn
= False

break

        if
not
player_turn
and
running:

# Enemy
turn logic

available_a
bilities =
[i for i,
cd in
enumerate(e
nemy.cooldo
wns) if cd
== 0]

if
available_a
```

```
bilities:

chosen_abil
ity =
random.choi
ce(availabl
e_abilities
)

ability =
enemy.abili
ties[chosen
_ability]

if
ability["da
mage"] > 0:

if
player.take
_damage(abi
lity["damag
e"]):

enemy.attac
k_animation
(player)

running =
end_game("E
nemy")
```

```
else:

enemy.attac
k_animation
(player)

show_messag
e(f"Enemy
used
{ability['n
ame']} for
{ability['d
amage']}
damage!")

if
ability["he
al"] > 0:

enemy.heal(
ability["he
al"])

enemy.heal_
animation(p
layer)

show_messag
e(f"Enemy
healed for
```

```
{ability['h
eal']}
HP!")

enemy.coold
owns[chosen
_ability] =
ability["co
oldown"]

else:

show_messag
e("Enemy is
stunned!")

player_turn
= True

        #
Reduce
cooldowns

player.cool
downs =
[max(0, cd
- 1) for cd
in
player.cool
downs]
```

```
enemy.coold
owns =
[max(0, cd
- 1) for cd
in
enemy.coold
owns]

        if
running:

screen.blit
(background
_image, (0,
0))  # Draw
background

player.draw
(screen)

enemy.draw(
screen)

buttons =
draw_button
s(player)



# Draw
cooldown
timers
```

```
for i,
cooldown in
enumerate(p
layer.coold
owns):

if cooldown
> 0:

cooldown_te
xt =
font.render
(str(cooldo
wn), True,
WHITE)

screen.blit
(cooldown_t
ext,
(buttons[i]
.centerx -
cooldown_te
xt.get_widt
h() // 2,
buttons[i].
bottom +
5))



# Draw
```

```
experience
bar

exp_bar_wid
th = 200

exp_bar_hei
ght = 20

exp_bar_x =
WIDTH // 2
-
exp_bar_wid
th // 2

exp_bar_y =
50

pygame.draw
.rect(scree
n, WHITE,
(exp_bar_x,
exp_bar_y,
exp_bar_wid
th,
exp_bar_hei
ght), 2)

pygame.draw
.rect(scree
n, BLUE,
```

```
(exp_bar_x,
exp_bar_y,
int(player.
exp / 100 *
exp_bar_wid
th),
exp_bar_hei
ght))

exp_text =
font.render
(f"EXP:
{player.exp
}/100",
True,
WHITE)

screen.blit
(exp_text,
(exp_bar_x
+
exp_bar_wid
th // 2 -
exp_text.ge
t_width()
// 2,
exp_bar_y +
exp_bar_hei
ght + 5))
```

```
pygame.disp
lay.flip()

clock.tick(
60)

    # End
of battle
    if
player.hp >
0:

exp_gain =
random.rand
int(20, 50)
+
(enemy.leve
l * 5)  #
More exp
for higher
level
enemies

player.gain
_exp(exp_ga
in)

show_messag
e(f"You
gained
{exp_gain}
```

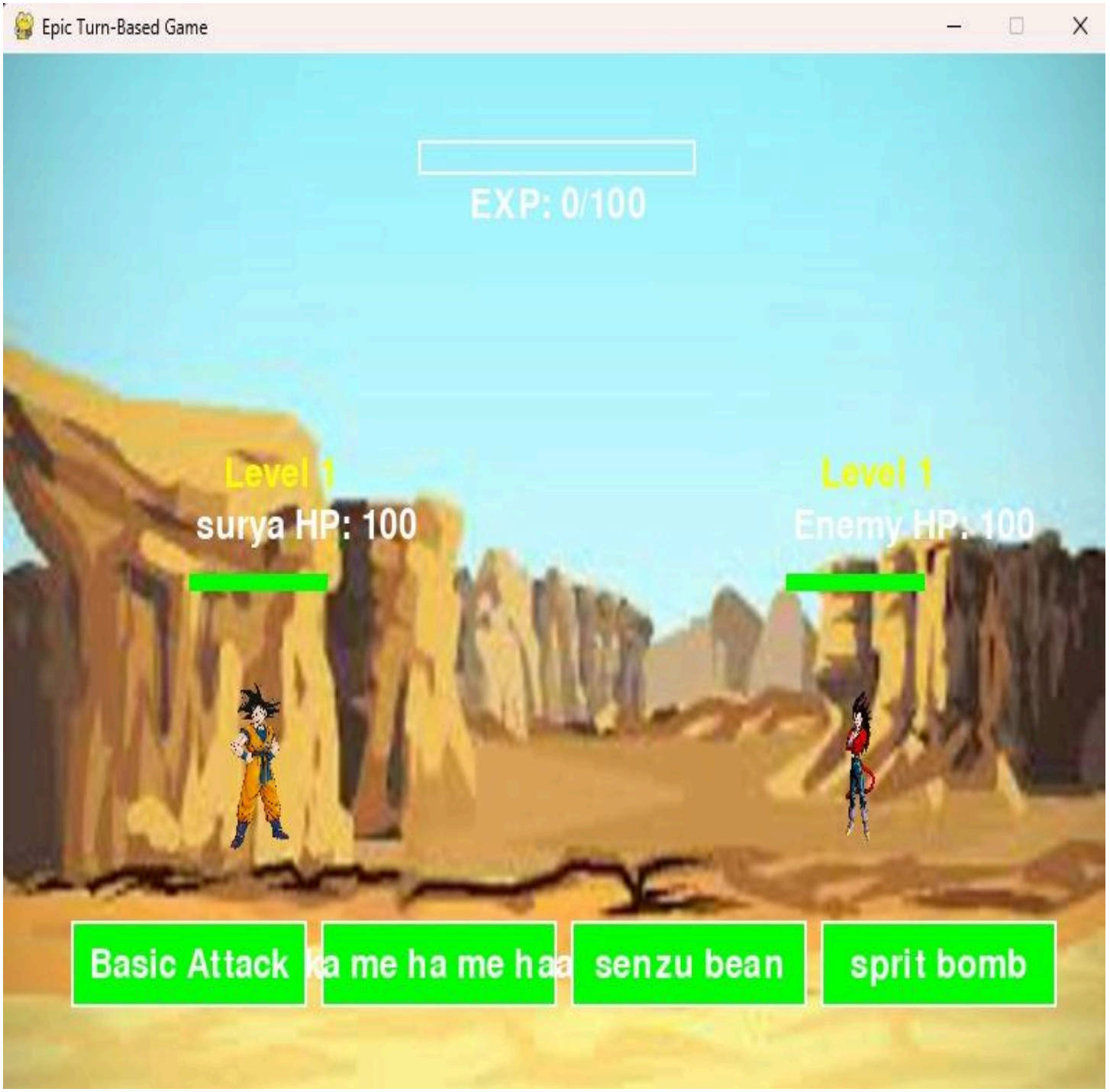
```
EXP!")
        if
player.exp
>= 100:

show_messag
e(f"Level
Up! You are
now level
{player.lev
el}!")



def main():

player_name
=
get_user_in
put()
    if not
player_name
:

return

    while
True:

action =
show_main_m
enu(player_
```

```
name)

        if
action ==
"quit":

break

elif action
== "play":

play_game(p
layer_name)

elif action
==
"leaderboar
d":

if
show_leader
board() ==
"quit":

break



pygame.quit
()

conn.close(
```

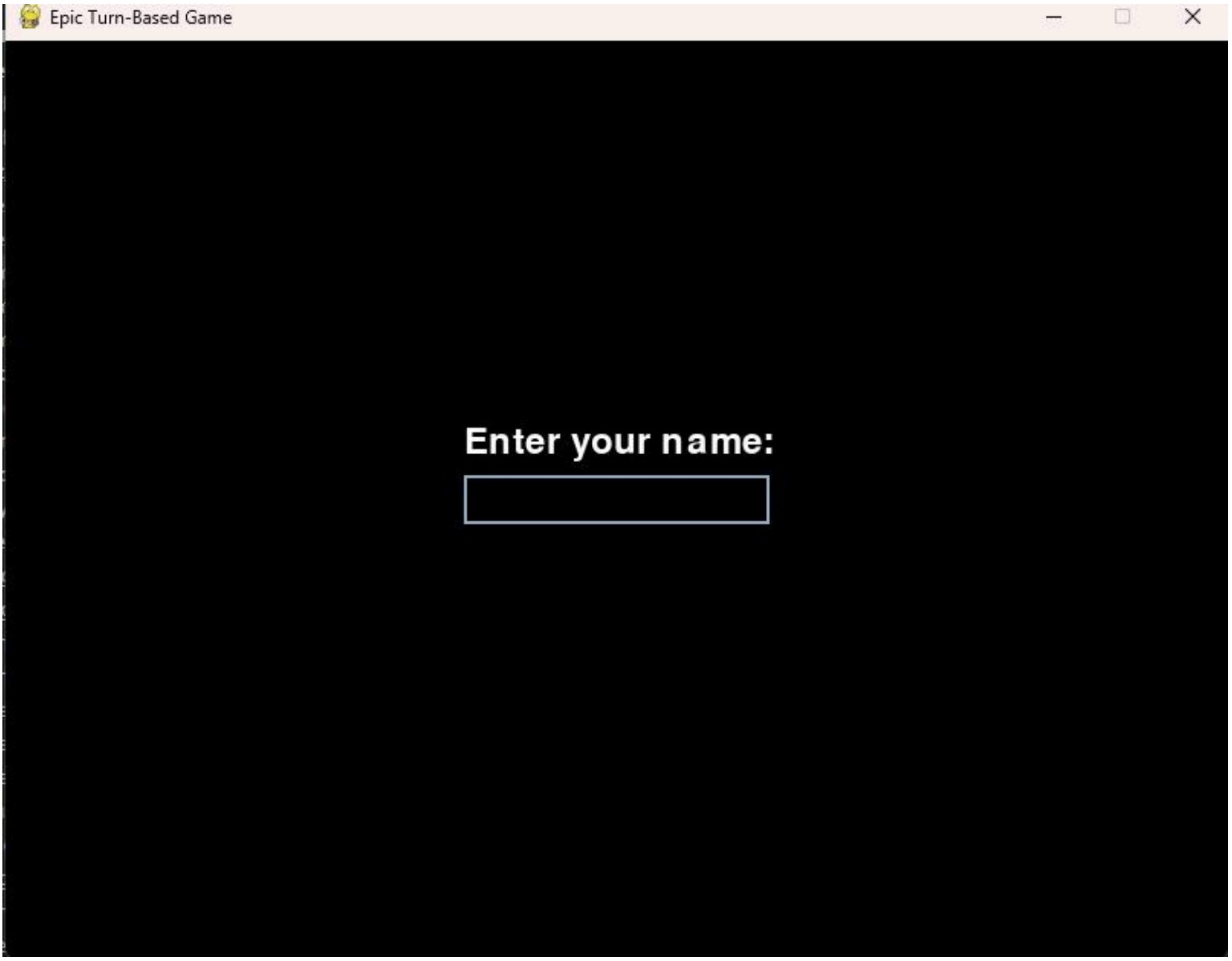
```
)


if __name__

==

"__main__":

    main()
```

OUTPUT

Main Interface :



User Details :



Bibliography :

To Develop this project many references were used :

https://www.bandainamcoent.com/games/dragon-ball-z-kakarot

https://realpython.com/python-gui-tkinter/

https://stackoverflow.com/

https://www.geeksforgeeks.org/

https://youtu.be/wBvYUMxV1Xw?feature

https://youtu.be/pd-0G0MigUA?feature=shared