SOFTWARE ENGINEERING

Project Title: Vaccinate Now to battle against COVID-19

Team Members:

Chandana Polakonda
Sai Keerthana Chitipothu
Gowtham Thotakura
Raina Maka
Anitha Nari

Goals and Objectives

Motivation:

Millions of individuals worldwide have been afflicted by the Covid-19 epidemic, which has had a significant influence on the world. Vaccination is one of the most efficient strategies to stop the virus from spreading. The efforts to increase vaccination rates have been hampered by persistent vaccine reluctance and false information in the public domain.

By developing a serious game that encourages vaccination and informs players about the value of getting vaccinated against Covid-19, our initiative seeks to address this problem. Vaccination is one of the best methods for preventing the virus from spreading. However, there is still vaccine hesitancy and misinformation circulating that has hindered the vaccination efforts.

Our project aims to address this issue by creating a serious game that promotes vaccination and educates users about the importance of getting vaccinated against Covid-19.

Significance:

The project is important because it will offer a fun and interactive way to encourage vaccination among a larger audience, especially those who are skeptical or have inaccurate information about vaccines.

Users can learn about the advantages of vaccination, the dangers of not getting vaccinated, and the significance of herd immunity by playing the game. In the end, this may aid in boosting vaccination rates, which may help stop the spread of Covid-19 and its variations.

In education for students a COVID-19 helps to relate games which can instruct players about the virus, how this virus spreads, and the procedure on how to avoid getting sick and attend the school regularly. This covid-19 game can educate the game players about the gravity of the pandemic and the importance of taking precautions by involving factual facts into gameplay.

The behavior changes which can be seen in players. They can adopt healthy habits like washing hands, wearing masks and by practicing social distancing of 6 feets.

Objectives:

The main objectives of this project are:

- To develop a fun and instructive game that will entice players to receive a COVID-19 vaccination.

- To disseminate accurate and current information regarding the advantages of vaccination and the significance of public health initiatives such as mask wear and social seclusion.
- To reach a wide audience, including those who may be reluctant or resistant to getting vaccinated, and provide an accessible and enjoyable way for them to learn about the importance of vaccination;
- To create a game that is both entertaining and educational, providing an enjoyable experience for players while also conveying important public health messages.
- To reach a wide audience, including those who may be reluctant or resistant to getting vaccinated.

Features:

The game is developed using Python and the Pygame library.
The following features are included in the project till now:

- Interactive game play where users can control the character and move around the game world.

The following features that are added in the final project:

- A mini-map to help users navigate the game world.

A minimap needs to be added to the current covid 19 fight game and it should be used throughout the game. It is a challenging task because with the movement of the character the mini map value changes.

- A storyline that promotes vaccination and encourages users to complete the game by getting vaccinated.

It is a stimulating step to track whether the gamer is vaccinated or not. It requires more research on the python code to develop it.
Storyline: You are to fight Covid by getting Vaccinated
You got infected with Covid virus
You got vaccinated.
You destroyed Covid.

- An inventory system to keep track of the user's vaccination status.

The implementation of quitting a game by the user is a challenging task as it requires the user’s mood for this task

- A better implementation of the story line is needed which resembles that the vaccination is mandatory for the gamer to complete the covid 19 fight game.

Background Work:

We have searched for various research papers to know the detailed functionality of the covid 19 games.
Different games were implemented for different purposes of providing awareness of the disease and how to prevent it.

The game which we are developing has covered some new features which are not present in the existing games.

Including this video game into the play store is under progress, so that many users can install and play this game and can be well informed about the importance of getting vaccinated and following important measures to stay immunized.

Till now our game application only works in desktop applications. So, we are also doing some background work to make our application in usable condition for mobiles and tablets too.

A background work was done to know how to upload our game application. To upload our Covid-19 fight game application into Google store, below are some of the important steps:

1. A developer account needs to be created.
2. A Google wallet Merchant account should be created and it should link with the user's developer account.
3. Next the app should be uploaded to the play store.
4. Now the android package kit should be uploaded.
5. The store listing to be prepared.
6. Content Rating should be added.
7. The pricing and the distribution model should be chosen.
8. Finally, the app needs to be published and is ready to install and is usable for the users to know the importance of vaccination and the steps to fight against Covid-19 disease.

For all the citations we have done the background work on how to create it.

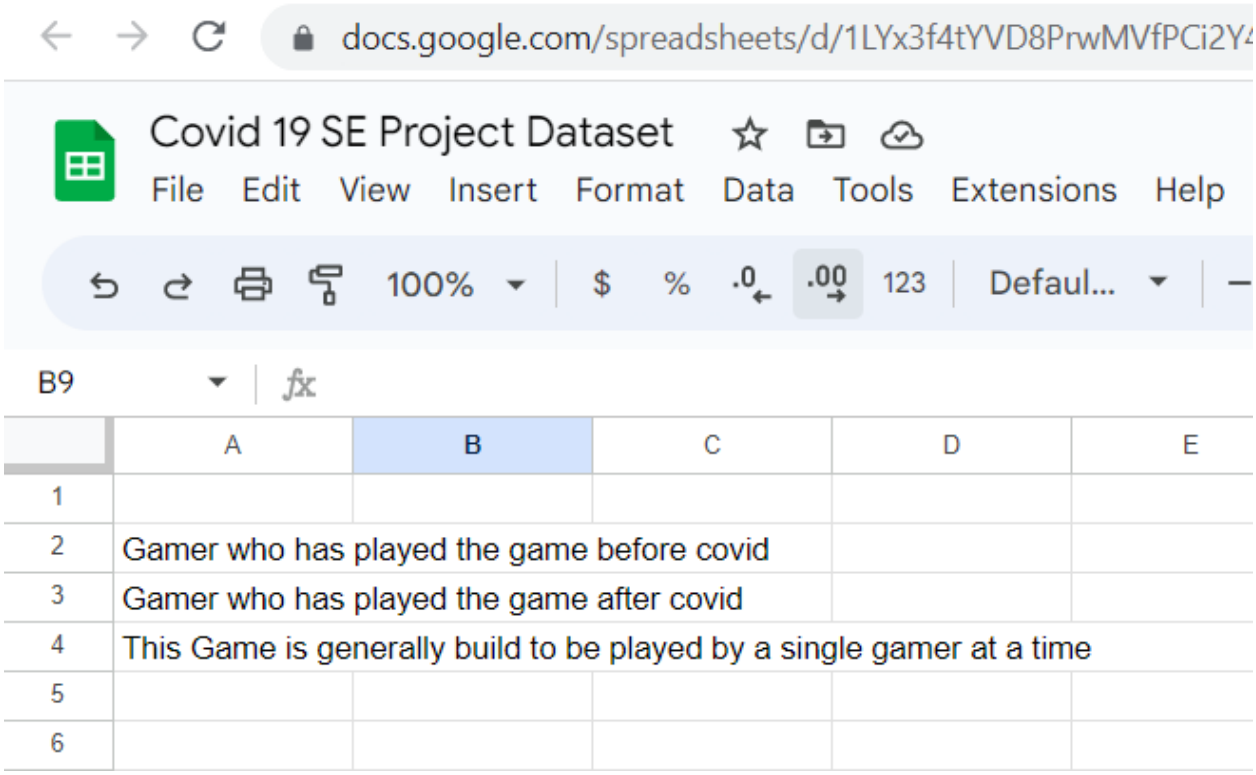
Creation of animations in our game to make the user Interface as user friendly, we have undergone many references to develop the animations.

Tools:

For the execution of the code any python executable environment can be used. We have used Visual Studio as our environment for the execution of the code.
Environments like pycharm, Jupyter, Google collab can be used.

Dataset:

As this is a Covid-19 fighter game, it does not include any Dataset for the development of the game but this game is played by a single gamer at a time. We have used python code to run and implement the source code.



	A	B	C	D	E
1					
2	Gamer who has played the game before covid				
3	Gamer who has played the game after covid				
4	This Game is generally build to be played by a single gamer at a time				
5					
6					

Figure a.

The above figure a shows the dataset that is being used for our game.

Implementation

Methodology:

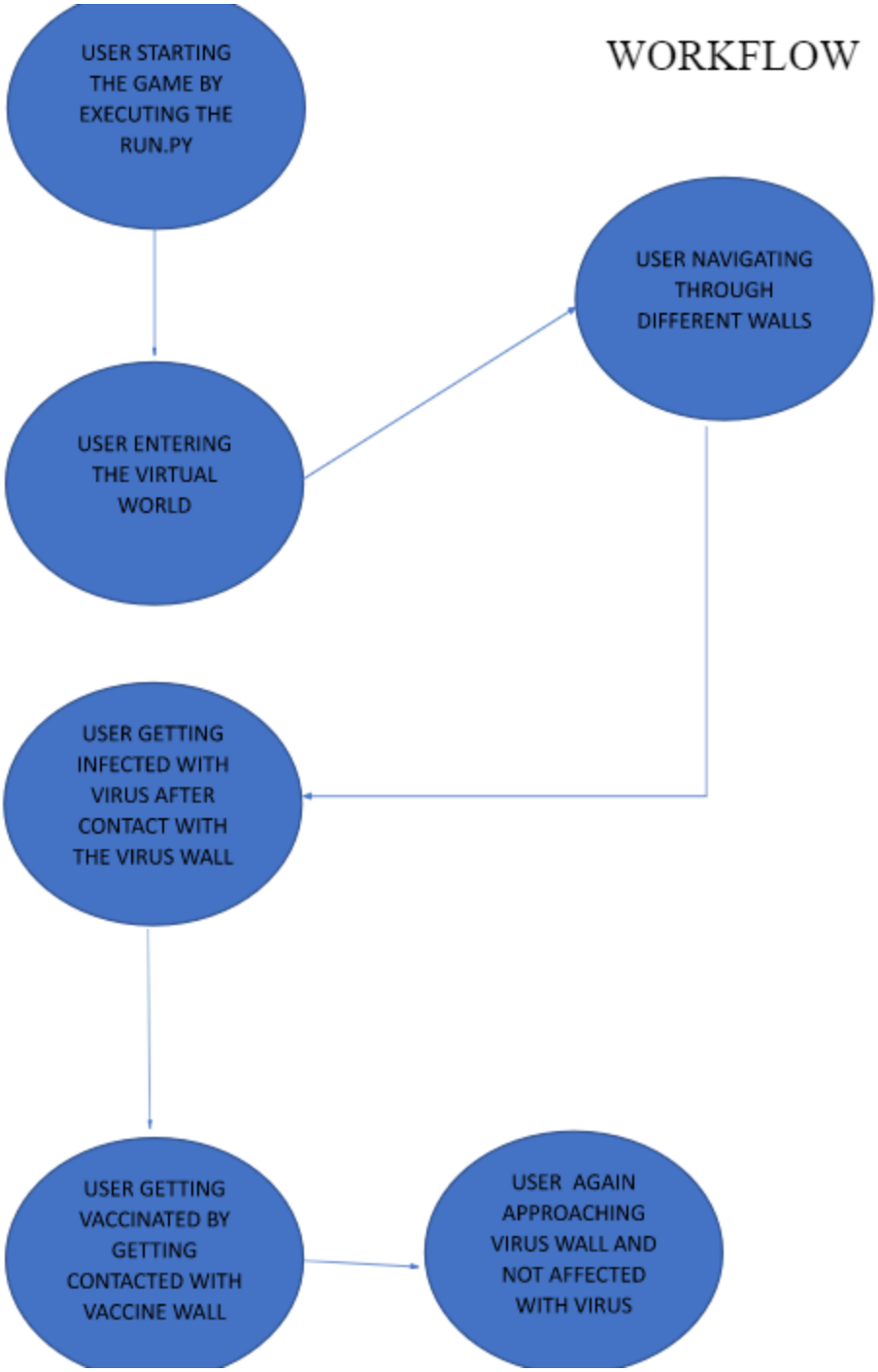


Figure:1

The above figure 1 explains the implementation of the covid-19 game. Initially the gamer starts the game and then enters the virtual world and then the gamer may get infected with the corona virus after getting contact with the virus wall. Next the gamer is vaccinated by contacting the

vaccine wall. Later the user again approaches the virus wall and is not affected with the virus. Finally, the gamer can complete the game only if he is vaccinated and is not infected anymore by the coronavirus.

```
from settings import *
text_map = [
    'AACACDAACAAA',
    'A.........E',
    'B..C.A.BBA.A',
    'A..A.A..cA.C',
    'A..B....A..A',
    'B..A..BAAA.C',
    'A..B...C...A',
    'ADAAAAEAAAcA'
]

world_map = {}
mini_map = set()
for j, row in enumerate(text_map):
    for i, char in enumerate(row):
        if char != '.':
            mini_map.add((i * MAP_TILE, j * MAP_TILE))
            if char == 'A':
                world_map[(i * TILE, j * TILE)] = 'a'
            elif char == 'B':
                world_map[(i * TILE, j * TILE)] = 'b'
            elif char == 'C':
                world_map[(i * TILE, j * TILE)] = 'c'
            elif char == 'E':
                world_map[(i * TILE, j * TILE)] = 'e'
            else:
                world_map[(i * TILE, j * TILE)] = 'd'
```

Fig2. Map.py code.

Above code from the fig2 is used to define world map and mini map from the text map. Firstly, the code imports all the constants from the settings module. The size of the map, size of the player etc., code has variable text_map which has strings representing grid of characteres

```python
import pygame
from settings import *
from map import world_map
def mapping(a, b):
    return (a // TILE) * TILE, (b // TILE) * TILE
def ray_casting(sc, player_pos, player_previous_direction_horizon, textures):
    ox, oy = player_pos
    xm, ym = mapping(ox, oy)
    player_current_direct                                              HALF_FOV
    for ray in range(NUM_
        sin_a = math.sin(player_current_direction_horizon)
        cos_a = math.cos(player_current_direction_horizon)
        sin_a = sin_a if sin_a else 0.00003
        cos_a = cos_a if cos_a else 0.00003
        # verticals
        x, dx = (xm + TILE, 1) if cos_a >= 0 else (xm, -1)
        for i in range(0, WIDTH, TILE):
            depth_v = (x - ox) / cos_a
            yv = oy + depth_v * sin_a
            tile_v = mapping(x + dx, yv)
            if tile_v in world_map:
                texture_v = world_map[tile_v]
                break
            x += dx * TILE
```

(variable) player_current_direction_horizon: Any

Fig.3 ray_casting.py code

```python
    # horizontals
    y, dy = (ym + TILE, 1) if sin_a >= 0 else (ym, -1)
    for i in range(0, HEIGHT, TILE):
        depth_h = (y - oy) / sin_a
        xh = ox + depth_h * cos_a
        tile_h = mapping(xh, y + dy)
        if tile_h in world_map:
            texture_h = world_map[tile_h]
            break
        y += dy * TILE
    # projection
    depth, offset, texture = (depth_v, yv, texture_v) if depth_v < depth_h else (depth_h, xh, texture_
    offset = int(offset) % TILE
    depth *= math.cos(player_previous_direction_horizon - player_current_direction_horizon)
    depth = max(depth, 0.0003)
    proj_height = min(int(PROJ_COEFF / depth), 2 * HEIGHT)

    wall_column = textures[texture].subsurface(offset * TEXTUR                    XTURE_HEIGH
    wall_column = pygame.transform.scale(wall_column, (SCALE, proj_height))
    sc.blit(wall_column, (ray * SCALE, HALF_HEIGHT - proj_height // 2))

    #cur_angle += DELTA_ANGLE
    player_current_direction_horizon += DELTA_ANGLE
```

(variable) proj_height: int

Fig.4 ray_casting.py code

We are using ray casting technique for the 3D game design. The pictures fig3, fig4 are the code for the 3D game designing. It calculates the distances and the texture of the walls in game and to project it on the screen. 2 function definitions helps to achieve it mapping and raycasting.

```
from settings import *
import pygame
import math
class Accomplice:
    def __init__(self):
        self.x, self.y = player_pos
        self.perspective = player_previous_direction_horizon
        self.sensitivity = 0.004
    @property
    def pos(self):
        return (self.x, self.y)
    def movement(self):
        self.keys_control()
        self.mouse_control()
    def keys_control(self):
        sin_a = math.sin(self.perspective)
        cos_a = math.cos(self.perspective)
        keys = pygame.key.get_pressed()
```

Fig.5 accomplice.py code

```
        if keys[pygame.K_ESCAPE]:
            exit()
        if keys[pygame.K_w]:
            self.x += player_speed * cos_a
            self.y += player_speed * sin_a
        if keys[pygame.K_s]:
            self.x += -player_speed * cos_a
            self.y += -player_speed * sin_a
        if keys[pygame.K_a]:
            self.x += player_speed * sin_a
            self.y += -player_speed * cos_a
        if keys[pygame.K_d]:
            self.x += -player_speed * sin_a
            self.y += player_speed * cos_a
        if keys[pygame.K_LEFT]:
            self.perspective -= 0.02
        if keys[pygame.K_RIGHT]:
            self.perspective += 0.02
    def mouse_control(self):
        if pygame.mouse.get_focused():
            angle_variation = pygame.mouse.get_pos()[0] - HALF_WIDTH
            pygame.mouse.set_pos((HALF_WIDTH, HALF_HEIGHT))
            self.perspective += angle_variation * self.sensitivity
```

Fig.6 accomplice.py code

The picture above fig5, fig6 are the code for the accomplice module. The code is used for to control the movement of the player. It initializes players position, field of view, sensitivity. The code has pos method by that we will get the position, control and rotation using the mouse by movement method. Method keys_control updates movement and rotation, while method mouse_control controls its rotation and movement of the mouse.

```python
import math

# game settings
WIDTH = 1200
HEIGHT = 800
HALF_WIDTH = WIDTH // 2
HALF_HEIGHT = HEIGHT // 2
#FPS = 60
TILE = 100
TEXT_POS = (WIDTH - 1025, 5)
TEXT_POS2 = (WIDTH - 1025, 40)
TEXT_POS3 = (WIDTH - 1025, 80)
TEXT_POS4 = (WIDTH - 1025, 120)
TEXT_POS5 = (WIDTH - 1025, 160)

# minimap settings
MAP_SCALE = 5
MAP_TILE = TILE // MAP_SCALE
MAP_POS = (0, HEIGHT - HEIGHT // MAP_SCALE)

# ray casting settings
FOV = math.pi / 3
HALF_FOV = FOV / 2
NUM_RAYS = 300
MAX_DEPTH = 800
DELTA_ANGLE = FOV / NUM_RAYS
DIST = NUM_RAYS / (2 * math.tan(HALF_FOV))
PROJ_COEFF = 3 * DIST * TILE
SCALE = WIDTH // NUM_RAYS
```

Fig.7 settings.py code

```
# texture settings (1200 x 1200)
TEXTURE_WIDTH = 1200
TEXTURE_HEIGHT = 1200
TEXTURE_SCALE = TEXTURE_WIDTH // TILE

# player settings
initial_position_height = HALF_HEIGHT - 220
initial_position_width = HALF_WIDTH - 360
player_pos = (initial_position_width, initial_position_height)
player_previous_direction_horizon = -3.15
player_speed = 2

# colors
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (220, 0, 0)
GREEN = (0, 80, 0)
BLUE = (0, 0, 255)
DARKGRAY = (40, 40, 40)
PURPLE = (120, 0, 120)
SKYBLUE = (0, 186, 255)
YELLOW = (220, 220, 0)
SANDY = (244, 164, 96)
```

Fig.8 settings.py code

The above picture fig7, fig8 are the codes of the setting module. In this code we define the settings and constants of the game which uses ray casting to create 3D. It consists of window and title size, mini map scale and position, settings of raycasting and FOV, player settings and texture and color constants.

```python
import pygame
from settings import *
from ray_casting import ray_casting
from map import mini_map

class Drawing:
    def __init__(self, sc, sc_map):
        self.sc = sc
        self.sc_map = sc_map
        self.font = pygame.font.SysFont('Arial', 36, bold=True)
        self.textures = {'a': pygame.image.load('C:/Users/HP/Downloads/covidFight/covidFightAssignment/img/wp06.png').convert(),
                         'b': pygame.image.load('C:/Users/HP/Downloads/covidFight/covidFightAssignment/img/wp07.png').convert(),
                         'c': pygame.image.load('C:/Users/HP/Downloads/covidFight/covidFightAssignment/img/wp05.png').convert(),
                         'd': pygame.image.load('C:/Users/HP/Downloads/covidFight/covidFightAssignment/img/wp08.png').convert(),
                         'e': pygame.image.load('C:/Users/HP/Downloads/covidFight/covidFightAssignment/img/wp03.png').convert(),
                         'm': pygame.image.load('C:/Users/HP/Downloads/covidFight/covidFightAssignment/img/C01.png').convert()
                         }

    def background(self, view):
        sky_offset = -5 * math.degrees(view) % WIDTH
        self.sc.blit(self.textures['m'], (sky_offset, 0))
        self.sc.blit(self.textures['m'], (sky_offset - WIDTH, 0))
        self.sc.blit(self.textures['m'], (sky_offset + WIDTH, 0))
        pygame.draw.rect(self.sc, DARKGRAY, (0, HALF_HEIGHT, WIDTH, HALF_HEIGHT))

    def world(self, player_pos, player_previous_direction_horizon):
        ray_casting(self.sc, player_pos, player_previous_direction_horizon, self.textures)
```

Fig.9 drawing.py code

```python
def display_storyline_with_payoffs(self, text, x, y,covid,isvacc):
    isvaccinated = isvacc
    indicator_num = 0
    display_text1 = '  '
    render1 = self.font.render(display_text1, 0, WHITE)
    value1 = int(x)
    value2 = int(y)
    counter = 0

    if value1 >= 80 and value1 <=230:
        if value2 >= 600:
            counter = 1
            if isvaccinated == True:
                display_text1 ="You destroyed Covid."
                render1 = render1 = self.font.render(display_text1, 0, SANDY)
            else:
                covid[1] = 1
                indicator_num = 4
                display_text1 = "You got infected with Covid virus."
                render1 = render1 = self.font.render(display_text1, 0, BLUE)
```

Fig.10 drawing.py code

```
if value1 >= 475 and value1 <= 630:
    if value2 <= 190:
        counter = 1
        if isvaccinated == True:
            display_text1 ="You destroyed Covid."
            render1 = render1 = self.font.render(display_text1, 0, SANDY)
        else:
            covid[2] = 1
            indicator_num = 4
            display_text1 = "You got infected with Covid virus."
            render1 = render1 = self.font.render(display_text1, 0, BLUE)

if value1 >= 669 and value1 <= 800:
    if value2 >= 300 and value2 <= 435:
        counter = 1
        if isvaccinated == True:
            display_text1 ="You destroyed Covid."
            render1 = render1 = self.font.render(display_text1, 0, SANDY)
        else:
            covid[3] = 1
            indicator_num = 4
            display_text1 = "You got infected with Covid virus."
            render1 = render1 = self.font.render(display_text1, 0, BLUE)
```

Fig.11 drawing.py code

```
if value1 >= 964 and value1 <= 1122:
    if value2 >= 600 and value2 <= 695:
        counter = 1
        if isvaccinated == True:
            display_text1 ="You destroyed Covid."
            render1 = render1 = self.font.render(display_text1, 0, SANDY)
        else:
            covid[4] = 1
            indicator_num = 4
            display_text1 = "You got infected with Covid virus."
            render1 = render1 = self.font.render(display_text1, 0, BLUE)

if value1 >= 584 and value1 <= 700:
    if value2 >= 579 and value2 <= 690:
        isvaccinated = True
        counter = 1
        indicator_num = 3
        display_text1 = "You got vaccinated."
        for i in covid:
            covid[i] = 0
        render1 = render1 = self.font.render(display_text1, 0, SANDY)
```

Fig.12 drawing.py code

```
    if value1 >= 1000 and value1 <= 1085:
        if value2 >= 95 and value2 <= 205:
            isvaccinated = True
            counter = 1
            indicator_num = 3
            display_text1 = "You got vaccinated."
            for i in covid:
                covid[i] = 0
            render1 = render1 = self.font.render(display_text1, 0, SANDY)

    if counter == 0:
        display_text1 = text
        render1 = self.font.render(display_text1, 0, WHITE)

    self.sc.blit(render1, TEXT_POS)
    self.payoff_accomplice(indicator_num, covid)
    return isvaccinated

def payoff_accomplice(self, indicator_num,covid):
    display_text1 = '  '
    payoff_render1 = self.font.render(display_text1, 0, WHITE)
```

Fig.13 drawing.py code

```
    if indicator_num == 3:
        pos =[TEXT_POS2,TEXT_POS3,TEXT_POS4,TEXT_POS5]
        line = 0
        for i in covid:
            if covid[i] == 1:
                display = 'PAY OFF '+str(i)+': You destroyed Covid'
                payoff_render1 = self.font.render(display, 0, WHITE)
                self.sc.blit(payoff_render1, pos[line])
                line += 1



def mini_map(self, accomplice):
    self.sc_map.fill(BLACK)
    map_x, map_y = accomplice.x // MAP_SCALE, accomplice.y // MAP_SCALE
    pygame.draw.line(self.sc_map, YELLOW, (map_x, map_y), (map_x + 12 * math.cos(accomplice.perspective),
                                                           map_y + 12 * math.sin(accomplice.perspective)), 2)
    pygame.draw.circle(self.sc_map, BLUE, (int(map_x), int(map_y)), 5)
    for x, y in mini_map:
        pygame.draw.rect(self.sc_map, RED, (x, y, MAP_TILE, MAP_TILE))
    self.sc.blit(self.sc_map, MAP_POS)
```

Fig.14 drawing.py code

The above picture fig10, fig11, fig,12, fig13 and fig14 are the codes for the drawing module. It contains 3 methods background method, world method and display storyline with payoff method. The background method is responsible for the moon on the sky and the ground. World method is

the reason for the walls in the game. Finally the display story line with payoff method is the reason for the tracking of the players position on the map.

```
import pygame
from settings import *
from accomplice import Accomplice
import math
from map import world_map
from drawing import Drawing
pygame.init()
sc = pygame.display.set_mode((WIDTH, HEIGHT))
sc_map = pygame.Surface((WIDTH // MAP_SCALE, HEIGHT // MAP_SCALE))
clock = pygame.time.Clock()
accomplice = Accomplice()
drawing = Drawing(sc, sc_map)
text = "Storyline: You are to fight Covid by getting vaccinated."
covid = {1:0, 2:0, 3:0, 4:0}
isvaccinated = False
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            exit()
    accomplice.movement()
    sc.fill(BLACK)

    drawing.background(accomplice.perspective)
    drawing.world(accomplice.pos, accomplice.perspective)
    isvaccinated = drawing.display_storyline_with_payoffs(text, accomplice.x, accomplice.y,covid, isvaccin
    drawing.mini_map(accomplice)

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

Fig.15 main.py code

The picture above fig15 is the code for the main.py module. It combines all the modules map.py, raycasting, accomplish, settings.py, and drawing.py.

The game is built using the Pygame library in Python. The main.py file contains the main loop of the game, which handles events, updates game objects, and renders the game window.

The game uses ray casting techniques to render a 3D view of a 2D map on the game window.

The ray_casting.py file contains the implementation of the ray casting algorithm, which casts rays from the player's position to the edges of the screen and calculates the distance and height of walls to create a 3D perspective view.

The settings.py file contains various settings such as window size, map size, player settings, color codes, etc. These settings are used throughout the game to control the behavior and appearance of the game objects.

The drawing.py file contains the Drawing class, which handles the rendering of textures, background, and world objects on the game window. It uses the Pygame library to load and display textures, and the ray_casting.py file to calculate the 3D perspective view.

The map.py file contains the definition of the game map, which is represented as a 2D grid of characters. Each character represents a type of wall or obstacle in the game world, and the corresponding textures are assigned to them in the world map dictionary.
We have tested the code by making to play the game by the users and getting feedback from them and playing the game on different pcs i.e., compatibility testing.

Project Management:

The project is a game developed using the Pygame library, which provides a simple way to create 2D games in Python.

The project follows a modular approach, with different functionalities organized into separate files (main.py, settings.py, drawing.py, ray_casting.py, and map.py) to improve code readability and maintainability.

The project follows an Agile methodology for development.

Software approach used for Covid 19 game (Spiral Methodology):

The project has been developed using an incremental approach, where different functionalities such as movement, rendering, and collision detection were developed incrementally and integrated into the game as they were completed.

Results:

The game is a working 3D raycasting project with simple movement and rendering features. The world's walls and background are displayed in the game window, and the player can move the character by pressing the keyboard characters (W, A, S, and D). The walls in the game are rendered based on the layout specified in the "map.py" file, and the game uses raycasting to produce a 3D perspective effect. The "settings.py" file provides a number of options that can be changed to change the way the game looks and plays.

Analysis:

The game features a first-person perspective view with ray casting techniques to render a 3D world. The player can control a character to move around the game world.

The game has several components, including a main file (main.py), a settings file (settings.py), a drawing file (drawing.py), and a map file (map.py), each of which contributes to different functionalities of the game.

The main.py file contains the main game loop that updates the game state, handles events, and renders the game world. It uses the Accomplice class from the accomplice.py file to control the movement of the player character. It also uses the Drawing class from the drawing.py file to draw the game world on the screen.

The settings.py file contains various settings and configurations for the game, such as screen dimensions, minimap settings, ray casting settings, player settings, and color definitions.
The drawing.py file contains the Drawing class, which handles drawing of the game world on the screen. It uses textures from the textures dictionary, which are loaded from image files, to render the walls and sky of the game world.

The map.py file contains the definition of the game world, which is represented as a text-based map in the text_map variable. The world_map dictionary is generated based on the text_map variable, which maps coordinates to corresponding wall textures.

	N	Boys (%) (N = 1,246)	Girls (%) (N =1,694)	P
Age, t1				$p = 0.123$
12	161	6	5	
13	201	8	6	
14	187	6	7	
15	213	8	7	
16	688	22	25	
17	1,097	37	37	
18+	389	13	14	
Socioeconomic status, compared to others at t2				**$p < 0.001$**
Better off	349	30	19	
About the same	1043	64	71	
Worse off	133	6	10	
Gaming, t1				**$p < 0.001$**
A lot more	686	41	14	
A little more	762	35	23	
About the same	501	14	21	
A little less	84	4	3	
A lot less	32	1	1	
No gaming	652	4	38	
Gaming, t2				**$p < 0.001$**
A lot more	172	20	9	
A little more	255	22	18	
About the same	348	30	25	
A little less	141	15	8	
A lot less	65	9	3	
No gaming	306	4	36	
Physical activity status, t1				**$p = 0.010$**
Active	1718	69	64	
Inactive	872	31	36	
Physical activity status, t2				**$p = 0.011$**
Active	615	60	52	
Inactive	498	40	48	
Categorization of gaming change over two time points				**$p < 0.001$**
More and increasing	298	37	17	
More and decreasing	377	43	23	
Increasing	106	5	11	
About the same	133	6	14	
Less	126	7	12	
No gaming	192	2	24	

Table a

Table a explains the study variables of girls and boys

The above table (a) shows the frequency of the players who has played the covid 19 fight game before, during the pandemic. The data that we have taken is of Norway and it shows from the table that in the initial days of Covid 19, only few boys were playing the game during April 2020. Considering the gender boys who played were 41%, 35% of the boys played very less and coming to girls about 14% played more and 23% of them played very less compared to boys.

As in the pandemic children were not allowed to play outside, so this completely reduced the physical activities and this resulted in poor health and very poor mental status of the children. By playing covid 19 games by children of any age groups and adults, it increases the awareness and also the mental stability of the people.

Discussion:

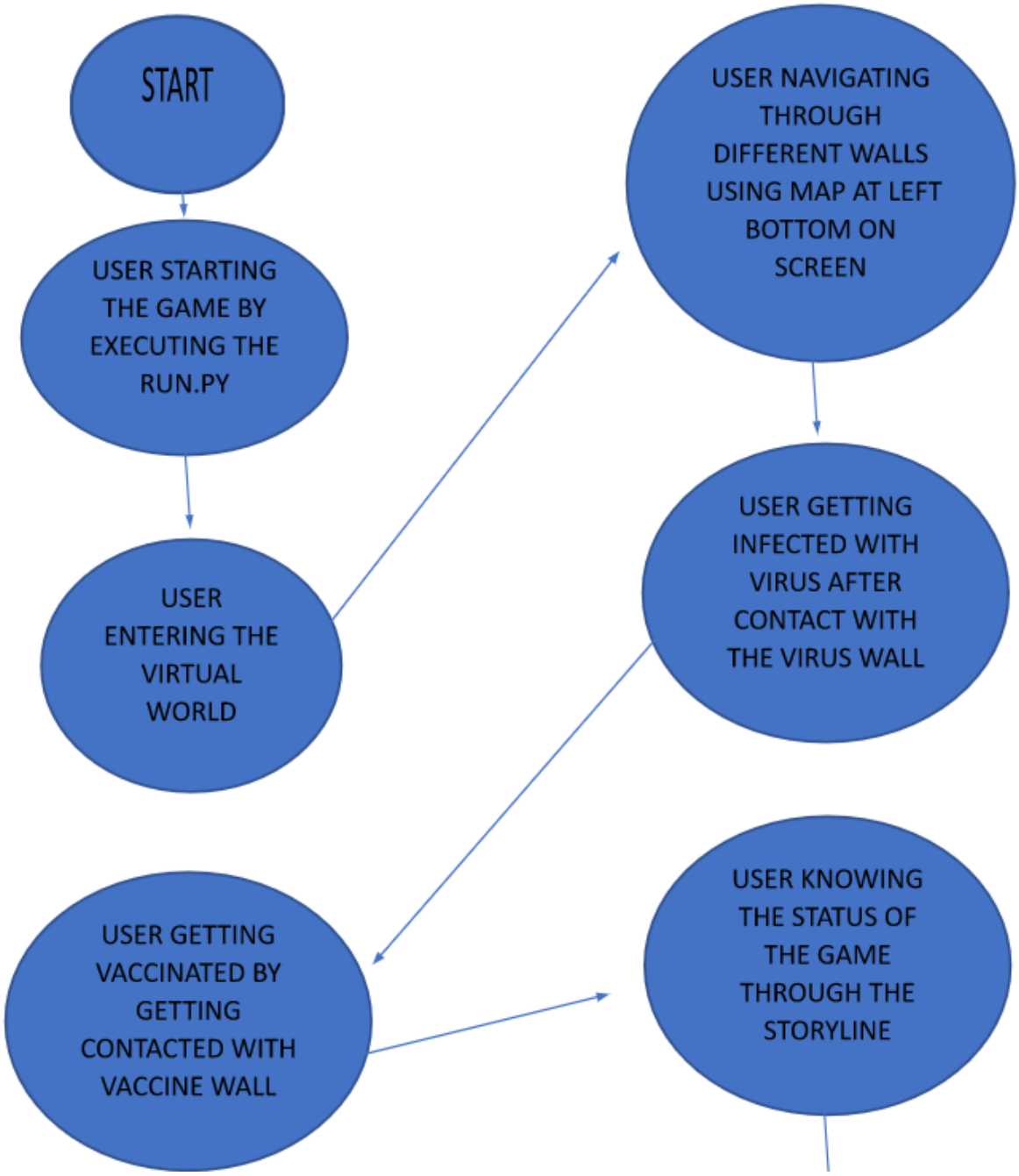
The game is implemented as a simple maze-like environment, where the player can navigate through the maze using ray casting techniques to render the walls and other objects in the game world. The player can control the character to move around and explore the maze.

The game uses Pygame, a popular game development library for Python, to handle graphics and user input. Pygame provides a simple way to create 2D games in Python and includes functionalities for handling events, rendering graphics, playing sounds, and more.
The game world is defined as a text-based map in the map.py file, where each character in the text_map variable represents a different type of object in the game world. The world_map dictionary is generated based on the text_map variable and is used to map coordinates to corresponding wall textures in the game world.

The ray casting technique is used to render the walls and other objects in the game world. Ray casting is a popular technique for rendering 3D graphics in 2D environments, such as first-person perspective games, where the view of the world is projected onto a 2D screen. The game uses a fixed number of rays that are cast from the player's position to simulate the perspective view of the game world. The ray_casting function in the drawing.py file implements the ray casting algorithm and renders the walls and other objects based on the world_map dictionary.

The game also includes a basic player movement system, where the player can control the character to move around the game world using the arrow keys or other input methods. The player movement is implemented in the Accomplice class in the accomplice.py file, which updates the player's position based on the input and handles collision detection with walls in the game world.

Workflow:


START
USER STARTING THE GAME BY EXECUTING THE RUN.PY
USER ENTERING THE VIRTUAL WORLD
USER NAVIGATING THROUGH DIFFERENT WALLS USING MAP AT LEFT BOTTOM ON SCREEN
USER GETTING INFECTED WITH VIRUS AFTER CONTACT WITH THE VIRUS WALL
USER GETTING VACCINATED BY GETTING CONTACTED WITH VACCINE WALL
USER KNOWING THE STATUS OF THE GAME THROUGH THE STORYLINE

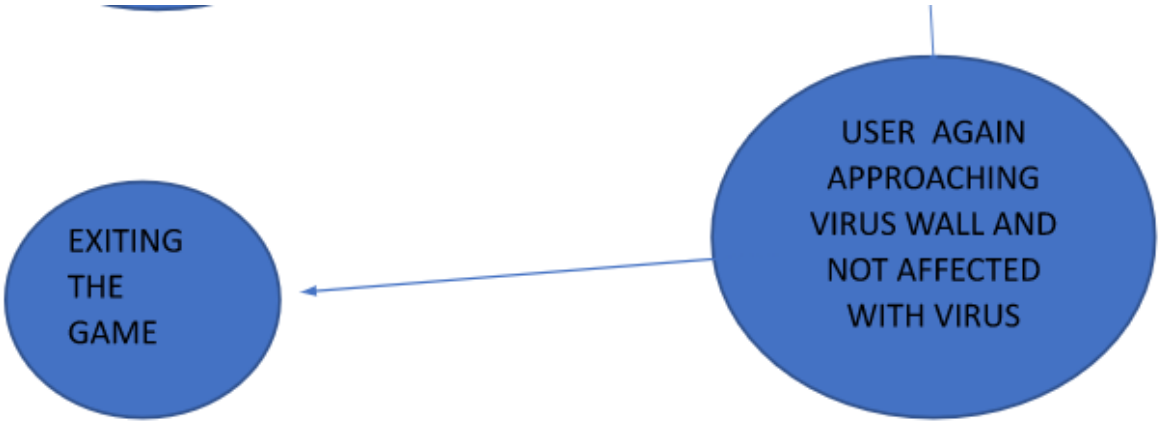


Figure b: Workflow for Covid 19 game

The above figure b explains the implementation of the covid-19 game. Initially the gamer starts the game denoted by start shape and then enters the virtual world and then the gamer may get infected with the coronavirus after getting contact with the virus wall. Next the gamer is vaccinated by contacting the vaccine wall. The location of the player is denoted by a blue dot and the story line is also displayed at each level of the game. Later the user again approaches the virus wall and is not affected with the virus. Finally, the gamer can complete the game only if he is vaccinated and is not infected anymore by the coronavirus this is denoted by stop symbol.

Work Completed Increment 1:

Responsibility	Description	Team Members & Contribution
Map module	The map module where the world map and the mini map is based on the text mapping.	Chandana Polakonda 20%
Ray casting Module	The covid game code looks to be a part in a program which	Keerthana 20%

	uses the ray casting method and has a 3D interface of the Gameworld.	
User Interface Settings Main	Worked on the creation Of the user interface, settings, main modules.	Gowtham 20%
Accomplish	The accomplish module determine the movement of the character and how to control the game	Raina 20%
Navigation of Character	Anitha has worked on the navigation of the characters in the game. Each movement resembles the player directions which represents whether the player is vaccinated or not, whether the player can be able to quit it. Vaccination wall and the immunity wall.	Anitha Nari 20%

- Created a Virtual Environment.
- Created a in-game character which represents the user(item-1)
- Created various walls representing different scenarios(item-2)
- Adding the items to the virtual environment
- Allowing player to navigate through the virtual environment created
- Developing code to change the situations when user approaches different walls

1. Code for character when approaching the virus wall. 2. Code for character when approaching the vaccine wall.

- Developing code for updating the status of the character.
- Updating the status of the character depending upon the different scenarios

Work completed Increment 2:

- A mini-map to help users navigate the game world.
- A storyline that promotes vaccination and encourages users to complete the game by getting vaccinated.

- An inventory system to keep track of the user's vaccination status.

Responsibility	**Description**	**Issue/Concerns**	**Team Members**
Map	A mini map Creation to navigate game world	A minimap is added to the current covid 19 fight game and it should be used throughout the game. It is a challenging task because with the movement of the character the mini map value changes. It shows the players which are denoted by blue dots. It describes the location of the player in the game.	Chandana Polakonda
Tracking system	Creation of system to track vaccination status	It is a stimulating step to track whether the gamer is vaccinated or not. It requires more research on the python code to develop it.	Gowtham
Menu	Implementing a menu at the beginning and at the end to quit for the players in the game to make it more user friendly.	The implementation of quitting a game by the user is a challenging task as it requires the users mood for this task	Raina
Customization	Customizing the in-game characters	Different modules need to be used for the customization of the characters for eye catching so that it looks pleasant for the user while playing the covid 19 game	Anitha Nari

Story Line	Story Line for Covid 19 fight application	A better implementation of the story line is added which resembles that the vaccination is mandatory for the gamer to complete the covid 19 fight game.	Sai Keerthana

Chandana Polakonda will be working on a mini- map to help the users for the navigation of the real world. As a mini-map needs to be added to the current covid 19 fight game. It should be used throughout the game. Whereas with the movement of the character the mini map value should changes is a challenging task

Raina will be working on implementing a menu at the beginning and at the end to quit for the players in the game to make it more user friendly. The user in a game can quit it at any time while playing and it is a challenging task as this process requires the user's mood for this task.

In future Gowtham will be working on a tracking system. This is a process where the user's data is captured whether the gamer is vaccinated or not and it will be displayed on the gamers screen while playing the covid 19 game.

Anitha Nari will be working on Customizing the in-game characters as different modules are needed to be used for the customization of the characters for eye catching so that it looks pleasant for the user while playing the covid 19 game and it will be user friendly.

Keerthana in future will be working on the better implementation of the story line which resembles that the vaccination is mandatory for the gamer to complete the covid 19 fight game. The game can be completed for the user only if he is vaccinated.

Preliminary Results:

Our project is on Covid 19 fight game. The below YouTube link shows the game video of covid 19. Once the player hits the Vaccine wall then he will be vaccinated and his immunity will be increased.

https://www.youtube.com/watch?v=1Q8eoscWLIs

The game is a working 3D raycasting project with simple movement and rendering features. The world's walls and background are displayed in the game window, and the player can move the character by pressing the keyboard characters (W, A, S, and D). The walls in the game are rendered based on the layout specified in the "map.py" file, and the game uses raycasting to produce a 3D perspective effect. The "settings.py" file provides a number of options that can be changed to change the way the game looks and plays.

Image 1:

Image 1 for the real environment.

Image 2:

Image 2 is the vaccination for covid-19 booster

Image 3:

Image 3 shows the wall of vaccination of the Immunized wall

Image 4:

Image 4 shows the not vaccinated walls of the gamers.

Image 5:

Image 5 shows the not immunized wall.

Image 6:

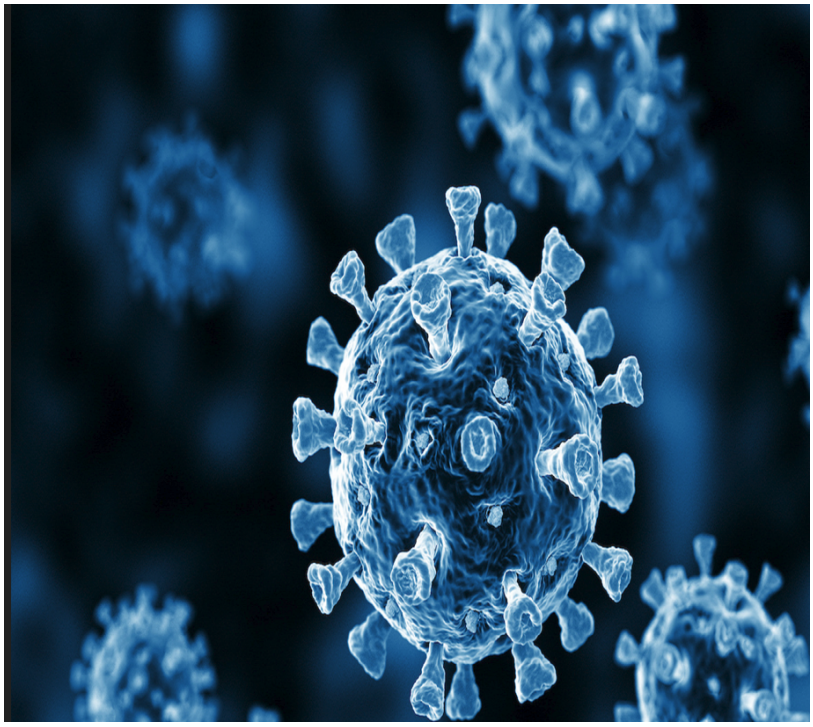


Image 6 represents the coronavirus wall.

Image 7:

Image 7 shows the players which are denoted by blue dot. It describes the location of the player in the game.

Story Line :

Image 1

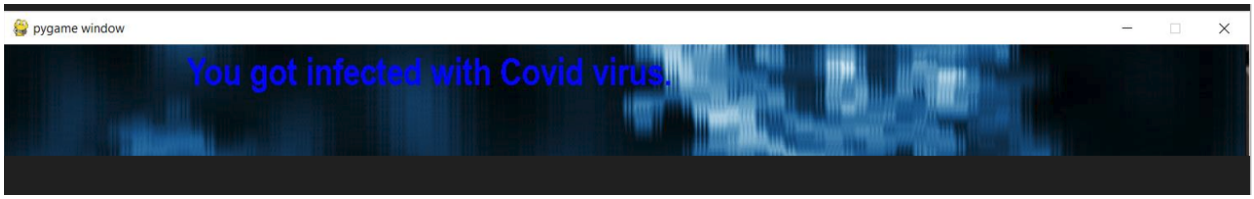


Image 2

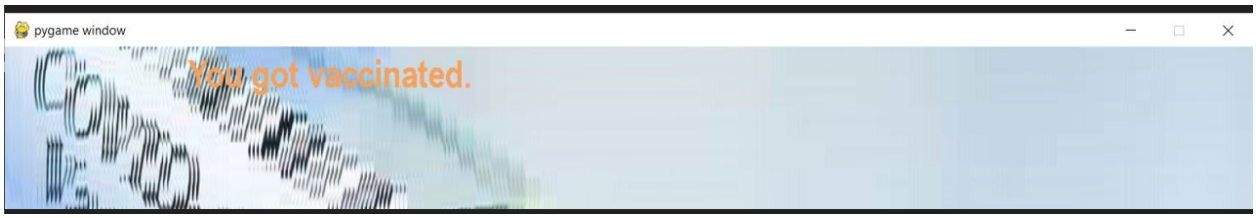


Image 3

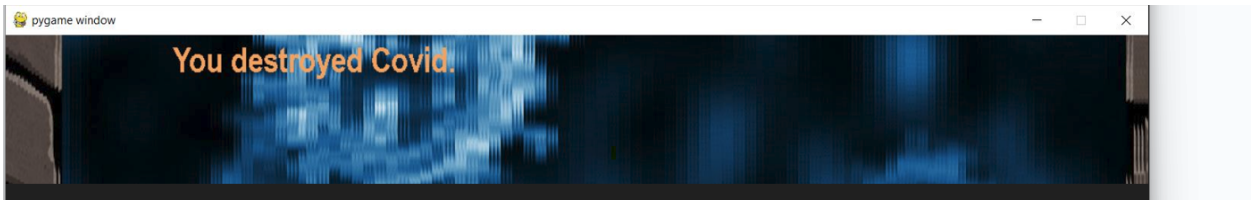


Image 4

The Images 1, 2, 3 & 4 describes the story line for the covid 19 game as given below:

- Storyline: You are to fight Covid by getting Vaccinated
- You got infected with Covid virus
- You got vaccinated.

- You destroyed Covid.

Relation of the project with class lecture:

Covid 19 game involves a waterfall model methodology to develop software game, one that covers many phases like requirements gathering, design, implementation, testing, and maintenance. By this way, we developed a project from zero percentage to 100 percentage. However in future we can go on and develop more features into this project with multiple players and so on.

We have included the incremental approach for the project. Initially we have gathered all the information of the project. Done deep analysis on it, figured out the required tools. Also created a use case diagram that describes the flow of our project. Next the tasks were created among the team members. Our team mates have conducted scrum meetings for the discussion, status and updates on the project progress. By this process all the conflicts were cleared and we were able to completed the tasks on time.

Conclusion:

The project is a maze-like game using Pygame and ray casting technique, there are some more features need to be added in the game such as storyline that promotes vaccination and encourages users to complete the game by getting vaccinated, inventory system to keep track of the user's vaccination status and mini-map to help users navigate the game world.

Demo Video:

https://www.youtube.com/watch?v=2o4Zww-fPVE

GitHub source for Source code:

https://github.com/anith462/Software-Engineering

References/ Citations:

A; D. D. H. A. T. (n.d.). Serious games for health: Three steps forwards. Advances in simulation (London, England). Retrieved April 16, 2023, from https://pubmed.ncbi.nlm.nih.gov/29450004/

British Journal of Educational Technology - ResearchGate. (n.d.). Retrieved April 17, 2023, from https://www.researchgate.net/journal/British-Journal-of-Educational-Technology-1467-8535

(PDF) serious games for health: Three steps forwards - ResearchGate. (n.d.). Retrieved April 17, 2023, from https://www.researchgate.net/publication/313341833_Serious_games_for_health_three_steps_forwards

Haug, E., Mæland, S., Lehmann, S., Bjørknes, R., Fadnes, L. T., Sandal, G. M., & Skogen, J. C. (2022, January 18). Increased gaming during COVID-19 predicts physical inactivity among youth in Norway-a two-wave longitudinal cohort study. Frontiers. Retrieved April 30, 2023, from https://www.frontiersin.org/articles/10.3389/fpubh.2022.812932/full

Changes and correlates of screen time in adults and children during the ... (n.d.). Retrieved May 1,2023,from https://www.thelancet.com/journals/eclinm/article/PIIS2589-5370(22)00182-1/fulltext

Readme file:

- Download the covidfight zip file.
- Upload all the modules in visual studios.
- Install pygame by using the command >> pip install pygame
- Run the main.py module.
- To execute the file, use the command >>python3 main.py in the terminal.