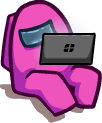
******COMPUTER SCIENCE PROJEC****T**

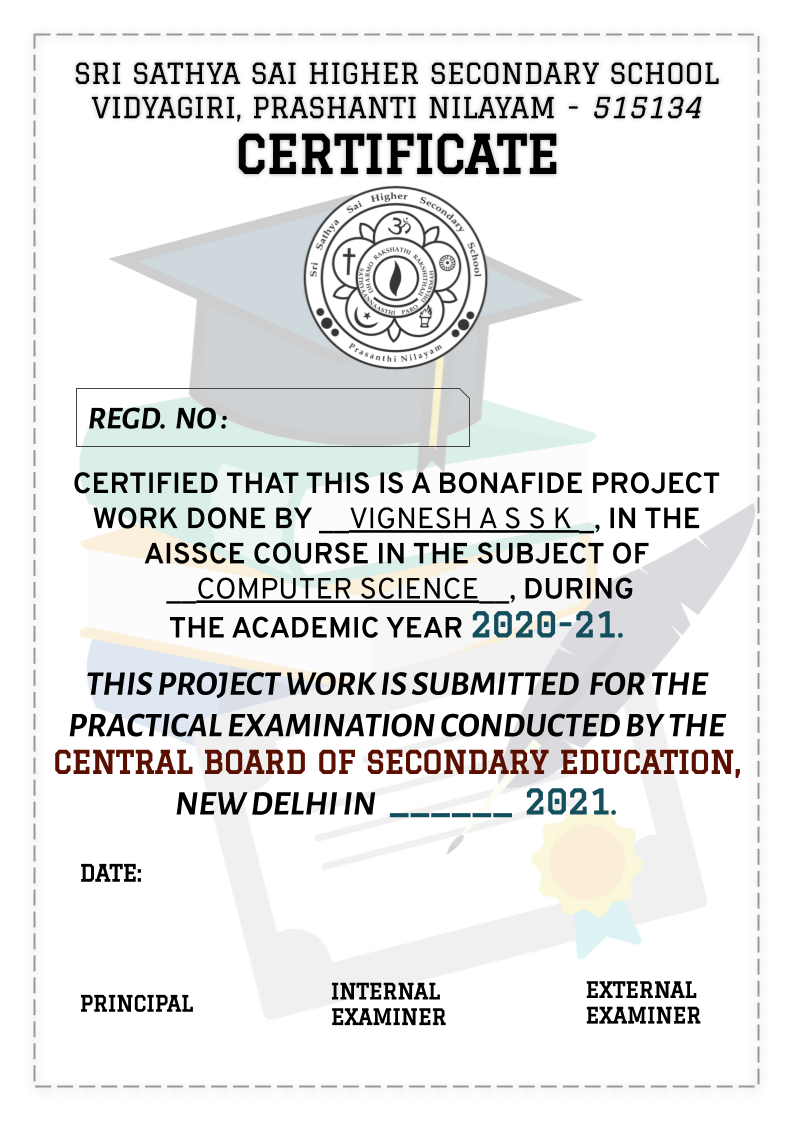
2 0 2 0 – 2 1

*RPG finish tasks or kill all social deduction game.*



**VIGNESH A S S K**

**XII MPC-CS**





Dedicated At Thy Lotus Feet

ACKNOWLEDGEMENT

**I would like to express my gratitude to our dear lord for giving me the strength to do this project.**

**I would like to thank Sri Venkateswar Prusty sir for teaching us with so much love and patience**

**I would like to thank principal sir for his support in the completion of this project.**

**I am most grateful to my mother for always being with me and encouraging me.**

**I thank all my friends for their help in completion for this project.**

**Table of Contents**

**ACKNOWLEDGEMENT4**

**AIM6**

**INTRODUCTION6**

**THEORY7**

**IMPLEMENTATION8**

Player movement8

Wall collisions8

Tasks9

Imposter10

Multiplayer11

Ray casting12

File structure and functions13

Main.py module13

Randomizer.py module14

Tasks.py module15

Walk\_anim.py module18

Free\_play.py module19

Online\_muiltiplayer.py module20

Networking/server.py module20

Networking/client.py module21

Rays/map\_coll.py module21

Collision\_points.dat22

**PROGRAM CODE23**

**PHOTO GALLERY118**

**FUTURE IMPLEMENTATION122**

**BIBLOGRAPHY123**

**THANK YOU124**

AIM

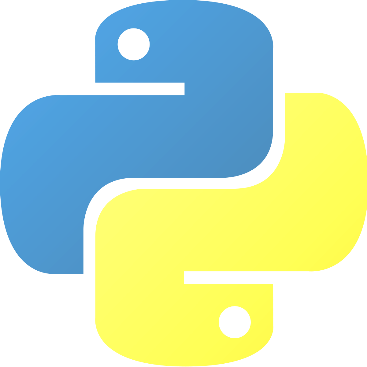
****T*o make a LAN multiplayer game called among us using python’s graphics engine pygame.*

INTRODUCTION

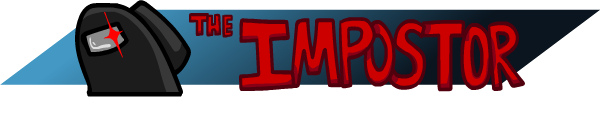
Among us is a RPG (Role Playing Game) multiplayer game. The original game was made by Innersloth. It was a top trending game on twitch.tv and YouTube in 2020. The game relies on players mind. This simple game attracted a lot of players around the world.

Space travel, betrayal, murder, maintenance. These are just a few exciting elements involved in among us. Solving the murder mystery and winning is the main aim of the crew. Killing the crew is the main aim of the imposter. The crew must perform various tasks to win. There are few resources that the players can interact with to find the most suspicious player. There is none that the players can trust in this game.

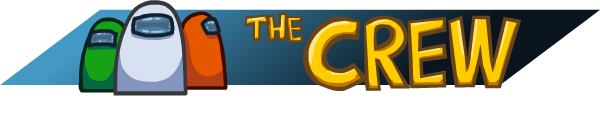
The main map here is “the skeld”. It is a spaceship that is on its way to predestined location. The map has various rooms. The tasks are located in the rooms. The overall shape of the map is a turtle-shaped. There are few moving parts in the map, to make that much more interesting.

Among us has been around for a while now. This game is an inspiration from among us. Even though it is not fully functional like among us. It like the development stages of the game. The language used to make this game is python. Python is a quite simple language, but can do some amazing stuff. Pygame library was used here. It is python graphics library that uses C and Assembly code for core functions. This library was designed to be used for making games.

THEORY

There are two types of people here:

The **Imposter** has to kill the crew. He must blend in with the crew and try not be suspicious. The imposter is not given any tasks so he must fake them to appear doing them. If he is able to kill crew, with one left to tell the tale, he **wins**. The imposter also has the ability to sabotage parts of map, so that the crewmates come together.



The other is the **crew**. The crew must complete all the given tasks to win. Some of the tasks are in order so the players have to complete a task to revel the next task. If the crew fine anyone suspicious they also have the ability to call an emergency meeting.

When there is dead body nearby the crew can report the body and call a meeting. The crew and the imposter can discuss and decide to vote out a player or skip to continue later. The crew win by either completing all their tasks or by voting out the imposter during the game.

IMPLEMENTATION

PLAYER MOVEMENT

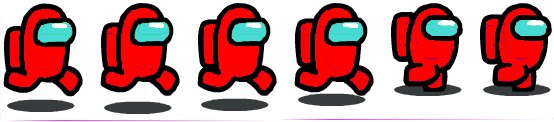
To make the game look like the player is moving around map, the map is made to move around the player putting the player constant. The player is always in the center of the window. The map is made of different images put together in their correct position. The position of the images have an offset which makes the player look like he is in a different position. The player here is a sprite. His image changes based on a walk cycle, to give the effect of the player walking around the map.

Fig. 1 half walk cycle of players.

WALL COLLISIONS

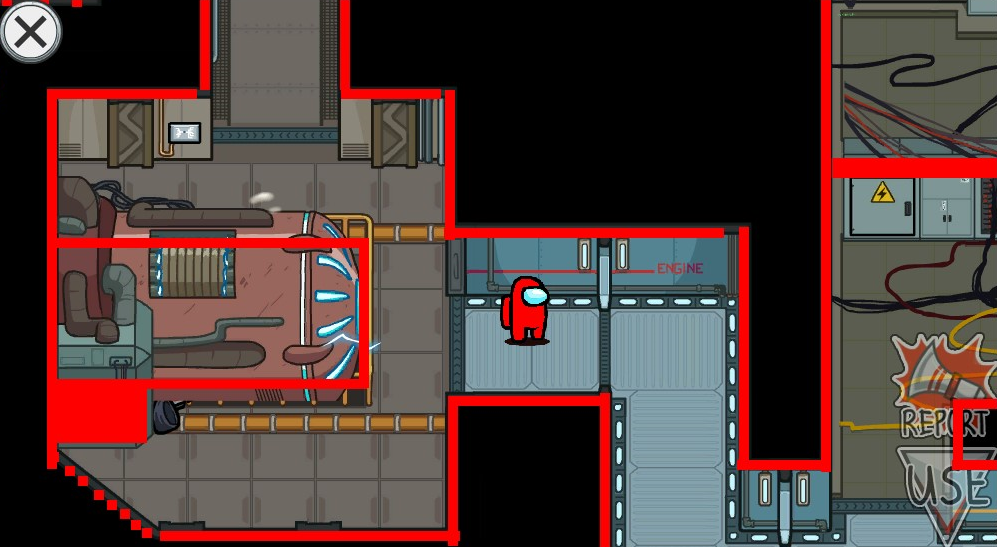
For the player to stay within the map and within the rooms, there are walls. The walls here are another sprite group. In order for them not to be seen by the player, they are blit behind the map. When the player sprite and the wall sprite collide, it can be detected by **pygame.sprite.collide\_rect (player, wall, False).** This gives in binary whether the two given sprites have collided. If the player has collided, we find which side of the player has collided with which side of the wall. Based on the side collided, the movement in that direction will disallowed.

Fig. 2 collision walls when displayed.

TASKS

Each player is given a set of tasks to complete. The tasks to be done is decided by the client itself. Only the number of tasks completed is sent to the server. Each tasks is a function in the class **Tasks**. Every task on the map is sprite. So when the player collides with any task sprite, his **USE** button highlights and he can perform the task. The player call the task in the class when he presses on the button, which is also a sprite. When the tasks is completed the tasks bar on top goes up by a little number. When the task bar reaches maximum the crewmates win.

IMPOSTER

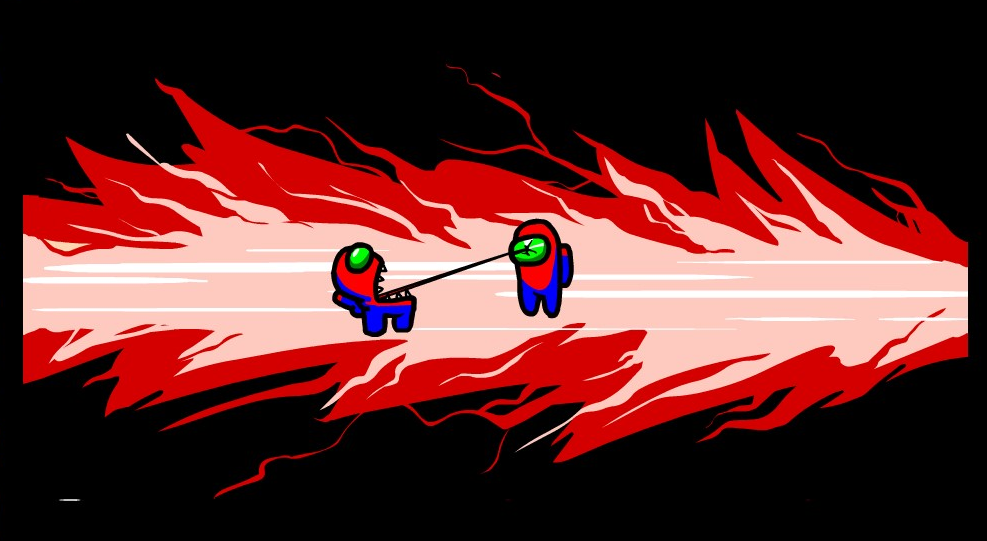
At the starting of the game, the server decides a player to the imposter. The imposter is given three extra abilities than the crew. He has the ability to kill players and sabotage parts of the map. When the imposter come near any player they get an option kill them. The killed player gets an animation on his screen saying that he was killed. When players come near a dead body they can report the body to discuss who could be the imposter and vote him out. The imposter wins if he kills all the players.

Fig. 3 imposter killing an innocent crewmate.

The imposter also has another ability to vent. There are few vents around the map. The imposter can hide in a vent to stay hidden. The crew can’t see the imposter in the vent.

MULTIPLAYER

Since among us is a multiplayer game, we have a server. Here the server is in the LAN. So only the clients in the LAN can connect to the server. The server and the client i.e. the player here are in constant communication with each other. The client sends his player info like the position, color, what he is doing, etc. The server sends the same info of all other player to the client. So that the player can blit all of that on the screen.

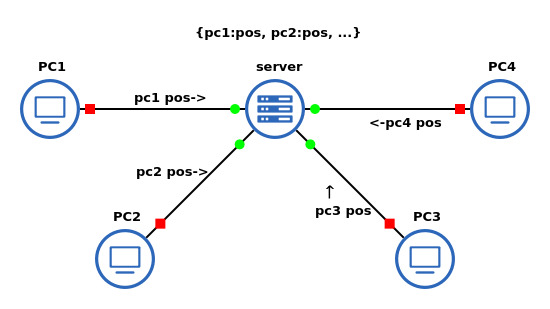
During voting, the players also send their info on whom they have voted for and if they have voted. And the server sends if he has been voted out. The killed/voted out players then become ghosts who can roam around the map with no collision. They can complete their tasks to win.

Fig. 4 server client model used here.

RAY CASTING

In order for the players to have a limited a vision, the concept of ray casting/ shadow casting was implemented here. Since all the wall positions were already known, a line was drawn from the player to each end of wall. Then the length of the line was calculated. If the length of the line was less than a certain value, a transparent polygon was drawn connecting the points. And the other parts were made a bit darker. This method is still in development and future tasks to do. This method was not as efficient as we are drawing line with each end point of the wall. So if the map was a huge one, this method would have led to inefficiency.

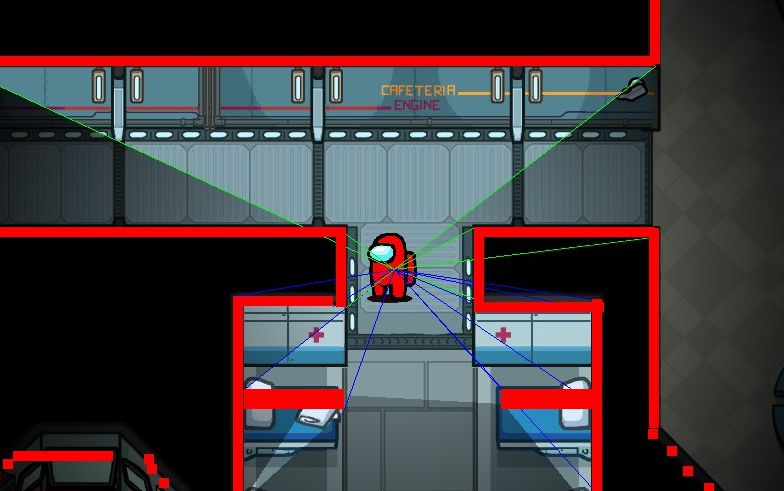


Fig. 5 ray casting in its initial stages.

FILE STRUCTURE AND FUNCTIONS

The total game is divided into six modules. Each module contains few function and classes. Below is a brief information on each function.

Main.py Module

class Screens():

Contains different screens like title screen, intro screen, etc.

def \_\_init\_\_():

Decides which screen must come based on the user input.

def TitleScreen():

Displays a intro screen before coming to the main menu screen.

def mainScreen():

Displays the main screen for the user to choose if he wants to free\_play or play online.

def colorchanger():

Takes a surface, color as input and return the surface with the white in the surface changed to the given color.

Randomizer.py Module

def getWiring():

Returns the fix wiring tasks to be done by the player.

def getPowerTo():

Returns divert to and accept power tasks to be done by the player.

def getDownload():

Returns the download tasks to be done by the player.

def getWiring():

Returns the fix wiring tasks to be done by the player.

def getChoice():

Based on random it returns if the task to be done is fual engine or align engine or medbay.

def getGarbage():

Based on random it returns if the task to be done is empty garbage or a random task from a list of tasks.

def getComp():

Returns a random task to be done from a list of tasks.

def getAllTasks():

Returns all the tasks to be done by the player using the above functions.

Tasks.py Module

class Tasks():

This class contains all the tasks/mini games that are there in the game.

def blitRotate():

Takes a surface, position, angle and returns a surface with the image on the surface rotated in the middle of the image at the given position.

def draw\_dashed\_line():

Takes surface, start pos, end pos, of a line and returns a dashed line with the given width.

def swipeCard():

Perform the swipe card mini game in the admin room when this function is called.

def fixWiring():

Perform the fix wiring mini game that is present in different parts of the map when it is called.

def emptyGarbage():

Perform the empty garbage mini game that is present in different parts of the map when this function is called.

def upload():

Perform the upload mini game that is present in the admin room when this function is called.

def Download():

Perform the download mini game that is present in different parts of the map when this function is called.

def clearLeaves():

Perform the clear leaves mini game that is present in the oxygen room when this function is called.

def alignEngine():

Perform the align Engine mini game that is present in the engine rooms when this function is called.

def calibrate():

Perform the calibrate mini game that is present in the electrical room when this function is called.

def chartCourse():

Perform the chart course mini game that is present in the navigation room when this function is called.

def weapons():

Perform the weapons mini game that is present in the weapons room when this function is called.

def divertPower():

Perform the divert power mini game that is present in the electrical room when this function is called.

def fualEngine():

Perform the fuel engine mini game that is present in the engine room when this function is called.

def fillCan():

Perform the fill can mini game that is present in the storage room when this function is called.

def inspectSample():

Perform the inspect sample mini game that is present in the med bay room when this function is called.

def primeShield():

Perform the prime shield mini game that is present in the shield room when this function is called.

def stabSteering():

Perform stabilize steering mini game that is present in the navigation room when this function is called.

def unlockManifolds():

Perform unlock manifolds mini game that is present in the reactor room when this function is called.

def starReactor():

Perform start reactor mini game that is present in the reactor room when this function is called.

def acceptPower():

Perform accept power mini game that is present in different parts of the map when this function is called.

def medbayScan():

Perform med bay scan mini game that is present in med bay room when this function is called.

def electrical():

Perform electrical mini game that is present in electrical room when this function is called during the sabotage.

def oxygen():

Perform oxygen mini game that is present in oxygen room when this function is called during the sabotage.

walk\_anim.py Module

class Player():

This class main player class. It is derived from the pygame.sprite.Sprite class.

def \_\_init\_\_():

Since this is a sprite class, it must have two main variable - image and rect. The rect is used for collisions and image for the image.

def update():

This function is called in each frame. It detects if any of the w, a, s and d key is pressed and change the image based on it. If the player is dead, the dead images are used instead of the usual walk cycle.

def colorchange():

Same as the function in the main class.

Free\_play.py Module

class Free\_play():

This class is called in the main.py when the player chooses the free play mode.

def run():

This function runs the main free play mode. It first initializes many variable for varies purpose. It runs a while loop that runs the game. It blits the map and the player from the Player class. All the collisions, player walk, the tasks calling happens in the main loop.

online\_muiltiplayer.py Module

class online():

This class is called in the main.py when the player chooses the online mode.

def run():

This function runs the online multiplayer. It first connects to the server through the client.py. If it connects to the server, it runs a loop that shows the all the players in the lobby. If the number of players is equal to a certain number greater than 2, the loop breaks and start another loop for the game. Here the players can perform all their tasks, call meetings, imposter can kill, sabotage. It is here that the game decides who has won or lost.

networking/server.py Module

def start():

This function starts a python socket server. It runs a loop searching for any incoming connections. It then start a thread for each client connected.

def handle\_client():

This function handles each connection in different thread. It adds the message received to the dictionary. This starts a while loop till the client sends a message to disconnect. The server receives the info from each client and sends the dictionary containing all players’ information.

networking/client.py Module

def getIp():

This function gets ip address from the user to connect to the server.

def getName():

This function gets a name to display for the player after it has connected to the server successfully.

class client():

def \_\_init\_\_():

This constructer initializes the server connection. And also get the ip address and name to connect with to the server. Here tkinter library is used to take input and output.

def send():

This function takes the message to send to the server and returns the server’s result.

rays/map\_coll.py Module

This module is similar to the free\_play.py, but with added shadow casting. A line is draw to each end of the wall sprites. If the length of the line is less than a value then a transparent polygon is draw in that area, while the other part is made dark at the starting.

collision\_points.dat file

This file contains all the wall positions. It is stored in this file using pickle library. The points are then found by reading the file with the pickle.

****

PROGRAM CODE

1. **Main.py………………………………………24**
2. **Walk\_anim.py…………………………28**
3. **Tasks.py……………………………………30**
4. **Randomizer.py………………………62**
5. **Free\_play.py…………………………63**
6. **Online\_muiltiplayer.py80**
7. **Server.py………………………………114**
8. **Ray casting function.115**
9. **Client.py………………………………116**

MAIN.PY

import pygame

from pygame.locals import \*

import random, time

from free\_play import Free\_play

from online\_muiltiplayer import online

pygame.init()

pygame.mixer.init()

size =[1000, 550]

screen = pygame.display.set\_mode(size)

pygame.display.set\_icon(pygame.image.load("idle.png"))

pygame.display.set\_caption("Among US Project")

clock = pygame.time.Clock()

fps = 50

pygame.mixer.music.load("bgs/Among Us Theme.wav")

Free\_play = Free\_play()

Theme = pygame.mixer.Sound('bgs/Among Us Theme.wav')

Theme.set\_volume(0.5)

pygame.mixer.Channel(0).play(Theme)

class Screens():

    def \_\_init\_\_(self):

        if self.TitleScreen() != 0:

            while True:

                if pygame.mixer.Channel(0).get\_busy() == 0:

                    pygame.mixer.Channel(0).play(Theme)

                self.nextScreen = self.mainScreen()

                if self.nextScreen == 1:

                    try:

                        online().run()

                    except:

                        pass

                elif self.nextScreen == 2:

                    pygame.mixer.music.stop()

                    Free\_play.run()

                else:

                    pygame.mixer.Channel(2).play(pygame.mixer.Sound('bgs/Among Us General Sounds/Panel\_GenericDisappear.wav'))

                    time.sleep(0.5)

                    pygame.quit()

                    exit()

    def TitleScreen(self):

        im1 = pygame.image.load("models/titleScreen/1.png")

        im2 = pygame.image.load("models/titleScreen/2.jpeg")

        b = 0

        a = 0

        g = 1

        while True:

            b += g

            if b == 255:

                g = -1

            if b == 0:

                return 1

            screen.fill(0)

            imcopy = im1.copy()

            imcopy.fill((255, 255, 255, b), None, pygame.BLEND\_RGBA\_MULT)

            screen.blit(im2, (-50, 0))

            screen.blit(imcopy, (291, 208))

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def mainScreen(self):

        im1 = pygame.image.load("models/titleScreen/1.png")

        online = pygame.image.load("images/main\_screen/2.png")

        freeplay = pygame.image.load("images/main\_screen/3.png")

        online\_rect = online.get\_rect()

        freeplay\_rect = freeplay.get\_rect()

        hover1 = 0

        hover2 = 0

        colors = [(255, 0, 0), (0, 0, 255), (0, 255, 0), (255, 255, 0), (255, 128, 0),

                (255, 255, 255), (255, 0, 255), (0, 255, 255), (102, 51, 0), (0, 204, 0)]

        plys = [self.colorchanger(pygame.image.load(f'images/main\_screen/mainscreenCrew{i}.png'), random.choice(colors)) for i in range(1,7)]

        bliting = [[random.choice(plys), random.randint(-500, 1000), random.randint(-100, 560)] for i in range(len(plys))]

        while True:

            screen.fill(0)

            if pygame.mixer.Channel(0).get\_busy() == 0:

                    pygame.mixer.Channel(0).play(Theme)

            for i in range(len(bliting)):

                bliting[i][1] += 1

                if bliting[i][1] > 1000:

                    bliting[i][1] = -100

                    bliting[i][2] = random.randint(-100, 560)

            for i in range(len(bliting)):

                screen.blit(bliting[i][0], (bliting[i][1], bliting[i][2]))

            screen.blit(im1, (281, 74))

            screen.blit(online, (583, 298))

            screen.blit(freeplay, (239, 298))

            if 584<pygame.mouse.get\_pos()[0]<774 and 299<pygame.mouse.get\_pos()[1]<377:

                if hover1 == 0:

                    hover1 += 1

                    pygame.mixer.Channel(1).play(pygame.mixer.Sound('bgs/Among Us General Sounds/UI\_Hover.wav'))

                if pygame.mouse.get\_pressed()[0]:

                    pygame.mixer.Channel(2).play(pygame.mixer.Sound('bgs/Among Us General Sounds/UI\_Select.wav'))

                    return 1

            else:

                hover1 = 0

            if 239<pygame.mouse.get\_pos()[0]<429 and 299<pygame.mouse.get\_pos()[1]<377:

                if hover2 == 0:

                    hover2 += 1

                    pygame.mixer.Channel(1).play(pygame.mixer.Sound('bgs/Among Us General Sounds/UI\_Hover.wav'))

                if pygame.mouse.get\_pressed()[0]:

                    pygame.mixer.Channel(2).play(pygame.mixer.Sound('bgs/Among Us General Sounds/UI\_Select.wav'))

                    return 2

            else:

                hover2 = 0

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def colorchanger(self, surface, color):

        """Fill all pixels of the surface with color, preserve transparency."""

        surface = surface.convert\_alpha()

        w, h = surface.get\_size()

        r, g, b = color

        for x in range(w):

            for y in range(h):

                if surface.get\_at((x,y)) == (255, 255, 255, 255):

                    surface.set\_at((x, y), pygame.Color(r, g, b, 255))

        return surface

Screens()

WALK\_ANIM.PY

import pygame

from pygame.locals import \*

import random

class Player(pygame.sprite.Sprite):

    def \_\_init\_\_(self, location = "images/Sprites/idle.png"):

        pygame.sprite.Sprite.\_\_init\_\_(self)

        self.image = pygame.image.load(location)

        self.image = pygame.transform.scale(self.image, (78-20,103-26))

        self.location = location

        self.rect = self.image.get\_rect()

        self.speed = 0.5

        self.move = 1

        self.flip = 0

        self.x = 0

        self.y = 0

        self.dead\_move = 1

    def update(self, secCam=0, color=(255, 0, 0), in\_vent = False, Not\_Alive = False):

        self.rect.x = 1000//2

        self.rect.y = 550//2

        keys = pygame.key.get\_pressed()

        if secCam == 1:

            self.rect.bottomright = (0, 0)

        if keys[K\_w] or keys[K\_a] or keys[K\_s] or keys[K\_d]:

            if pygame.mixer.Channel(3).get\_busy() == 0:

                pygame.mixer.Channel(3).play(pygame.mixer.Sound(f'bgs/Player/Footsteps/Metal/FootstepMetal0{random.randint(1,8)}.wav'))

        if keys[K\_d]:

            self.flip = 1

            self.image = pygame.image.load(f"images/Sprites/Walk/walkcolor00{int(self.move)}.png")

            self.move += self.speed

            self.x = 1

        elif keys[K\_a]:

            self.flip = 0

            self.image = pygame.image.load(f"images/Sprites/Walk/walkcolor00{int(self.move)}.png")

            self.move += self.speed

            self.x = 1

        elif keys[K\_w]:

            self.image = pygame.image.load(f"images/Sprites/Walk/walkcolor00{int(self.move)}.png")

            self.move += self.speed

            self.y = 1

        elif keys[K\_s]:

            self.image = pygame.image.load(f"images/Sprites/Walk/walkcolor00{int(self.move)}.png")

            self.move += self.speed

            self.y = 1

        else:

            self.image = pygame.image.load(self.location)

            self.move = 1

            self.x, self.y = 0, 0

        if self.move == 13:

            self.move = 1

        if Not\_Alive:

            self.dead\_move += 0.5

            if int(self.dead\_move) == 49:

                self.dead\_move = 1

            self.image = pygame.image.load(f'images/Sprites/Ghost/ghostbob{int(self.dead\_move)}.png')

        if self.flip == 0:

            self.image = pygame.transform.flip(self.image, True, False)

        if in\_vent:

            self.image = pygame.image.load('models/maps/4.png')

        self.image = pygame.transform.scale(self.image, (78-25,103-30)) #(78-20,103-26))

        self.image = self.colorchanger(self.image, color)

    def colorchanger(self, surface, color):

        """Fill all pixels of the surface with color, preserve transparency."""

        surface = surface.convert\_alpha()

        w, h = surface.get\_size()

        r, g, b = color

        for x in range(w):

            for y in range(h):

                if surface.get\_at((x,y)) == (255, 0, 0, 255):

                    surface.set\_at((x, y), pygame.Color(r, g, b, 255))

                elif surface.get\_at((x,y)) == (254, 0, 0, 127):

                    surface.set\_at((x, y), pygame.Color(r, g, b, 127))

                elif surface.get\_at((x,y)) == (254, 0, 0, 126):

                    surface.set\_at((x, y), pygame.Color(r, g, b, 126))

        return surface

TASKS.PY

import pygame

from pygame.locals import \*

import random, math, numpy

pygame.init()

clock = pygame.time.Clock()

fps = 50

size =[1000, 550]

screen = pygame.display.set\_mode(size)

close = pygame.image.load("models/buttons/close.png")

class Tasks():

    def \_\_init\_\_(self):

        self.tasks = [0 for i in range(20)]

    def blitRotate(self, surf, image, pos, originPos, angle):

        # calcaulate the axis aligned bounding box of the rotated image

        w, h       = image.get\_size()

        box        = [pygame.math.Vector2(p) for p in [(0, 0), (w, 0), (w, -h), (0, -h)]]

        box\_rotate = [p.rotate(angle) for p in box]

        min\_box    = (min(box\_rotate, key=lambda p: p[0])[0], min(box\_rotate, key=lambda p: p[1])[1])

        max\_box    = (max(box\_rotate, key=lambda p: p[0])[0], max(box\_rotate, key=lambda p: p[1])[1])

        # calculate the translation of the pivot

        pivot        = pygame.math.Vector2(originPos[0], -originPos[1])

        pivot\_rotate = pivot.rotate(angle)

        pivot\_move   = pivot\_rotate - pivot

        # calculate the upper left origin of the rotated image

        origin = (pos[0] - originPos[0] + min\_box[0] - pivot\_move[0], pos[1] - originPos[1] - max\_box[1] + pivot\_move[1])

        # get a rotated image

        rotated\_image = pygame.transform.rotate(image, angle)

        # rotate and blit the image

        surf.blit(rotated\_image, origin)

        # draw rectangle around the image

        #pygame.draw.rect(surf, (255, 0, 0), (\*origin, \*rotated\_image.get\_size()),2)

    def draw\_dashed\_line(self, surf, color, start\_pos, end\_pos, width=5, dash\_length=10):

        x1, y1 = start\_pos

        x2, y2 = end\_pos

        dl = dash\_length

        if (x1 == x2):

            ycoords = [y for y in range(y1, y2, dl if y1 < y2 else -dl)]

            xcoords = [x1] \* len(ycoords)

        elif (y1 == y2):

            xcoords = [x for x in range(x1, x2, dl if x1 < x2 else -dl)]

            ycoords = [y1] \* len(xcoords)

        else:

            a = abs(x2 - x1)

            b = abs(y2 - y1)

            c = round(math.sqrt(a\*\*2 + b\*\*2))

            dx = dl \* a / c

            dy = dl \* b / c

            xcoords = [x for x in numpy.arange(x1, x2, dx if x1 < x2 else -dx)]

            ycoords = [y for y in numpy.arange(y1, y2, dy if y1 < y2 else -dy)]

        next\_coords = list(zip(xcoords[1::2], ycoords[1::2]))

        last\_coords = list(zip(xcoords[0::2], ycoords[0::2]))

        for (x1, y1), (x2, y2) in zip(next\_coords, last\_coords):

            start = (round(x1), round(y1))

            end = (round(x2), round(y2))

            pygame.draw.line(surf, color, start, end, width)

    def swipeCard(self):

        spbg = pygame.image.load("models/tasks/Swipe Card/admin\_BG.png")

        spbg1 = pygame.image.load("models/tasks/Swipe Card/admin\_sliderTop.png")

        spbg2 = pygame.image.load("models/tasks/Swipe Card/admin\_sliderBottom.png")

        spbg3 = pygame.image.load("models/tasks/Swipe Card/admin\_Wallet.png")

        spbg4 = pygame.image.load("models/tasks/Swipe Card/admin\_walletFront.png")

        spbg5 = pygame.image.load("models/tasks/Swipe Card/admin\_Card.png")

        doing = 0

        swipeDon = 0

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(spbg, (259, 32))

            screen.blit(spbg2, (259, 169))

            screen.blit(spbg3, (268, 363))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            if 286 < pygame.mouse.get\_pos()[0] < 507 and pygame.mouse.get\_pressed()[0]:

                doing = 1

            if doing == 1:

                if pygame.mouse.get\_pressed()[0]:

                    if 201 < pygame.mouse.get\_pos()[0] < 700 and 150 < pygame.mouse.get\_pos()[1] < 250:

                        screen.blit(spbg5, (pygame.mouse.get\_pos()[0], 150))

                        if 600 < pygame.mouse.get\_pos()[0] < 680:

                            self.tasks[0] = 1

                            return 1

                else:

                    screen.blit(spbg5, (201, 150))

            else:

                screen.blit(spbg5, (285, 376))

            screen.blit(spbg1, (259, 32))

            screen.blit(spbg4, (279, 446))

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[0] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def fixWiring(self):

        fw1 = pygame.image.load("models/tasks/Fix Wiring/electricity\_wiresBaseBack.png")

        fw2 = pygame.image.load("models/tasks/Fix Wiring/electricity\_wires1.png")

        fw3 = pygame.image.load("models/tasks/Fix Wiring/electricity\_wires1.png")

        red, green, blue, yellow = 0, 0, 0, 0

        while True:

            screen.fill((0, 0, 0))

            screen.blit(fw1, (264, 28))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            if 269 < pygame.mouse.get\_pos()[0] < 300 and 130 < pygame.mouse.get\_pos()[1] < 200 and pygame.mouse.get\_pressed()[0] and red != 2:

                red = 1

            elif 269 < pygame.mouse.get\_pos()[0] < 300 and 230 < pygame.mouse.get\_pos()[1] < 300 and pygame.mouse.get\_pressed()[0] and green != 2:

                green = 1

            elif 269 < pygame.mouse.get\_pos()[0] < 300 and 335 < pygame.mouse.get\_pos()[1] < 400 and pygame.mouse.get\_pressed()[0] and blue != 2:

                blue = 1

            elif 269 < pygame.mouse.get\_pos()[0] < 300 and 440 < pygame.mouse.get\_pos()[1] < 500 and pygame.mouse.get\_pressed()[0] and yellow != 2:

                yellow = 1

            if red == 1:

                pygame.draw.line(screen, (255, 0, 0), (269, 130), pygame.mouse.get\_pos(), 20)

                #(735, 129)

                if 735 < pygame.mouse.get\_pos()[0] < 800 and 130 < pygame.mouse.get\_pos()[1] < 200 and pygame.mouse.get\_pressed()[0]:

                    red = 2

                    redpos = pygame.mouse.get\_pos()

                screen.blit(fw2, (pygame.mouse.get\_pos()[0]-5, pygame.mouse.get\_pos()[1]-10))

            elif green == 1:

                pygame.draw.line(screen, (0, 255, 0), (269, 230), pygame.mouse.get\_pos(), 20)

                if 735 < pygame.mouse.get\_pos()[0] < 800 and 230 < pygame.mouse.get\_pos()[1] < 300 and pygame.mouse.get\_pressed()[0]:

                    green = 2

                    greenpos = pygame.mouse.get\_pos()

                screen.blit(fw2, (pygame.mouse.get\_pos()[0]-5, pygame.mouse.get\_pos()[1]-10))

            elif blue == 1:

                pygame.draw.line(screen, (0, 0, 255), (269, 335), pygame.mouse.get\_pos(), 20)

                if 735 < pygame.mouse.get\_pos()[0] < 800 and 335 < pygame.mouse.get\_pos()[1] < 400 and pygame.mouse.get\_pressed()[0]:

                    blue = 2

                    bluepos = pygame.mouse.get\_pos()

                screen.blit(fw2, (pygame.mouse.get\_pos()[0]-5, pygame.mouse.get\_pos()[1]-10))

            elif yellow == 1:

                pygame.draw.line(screen, (255, 255, 0), (269, 440), pygame.mouse.get\_pos(), 20)

                if 735 < pygame.mouse.get\_pos()[0] < 800 and 440 < pygame.mouse.get\_pos()[1] < 500 and pygame.mouse.get\_pressed()[0]:

                    yellow = 2

                    yellowpos = pygame.mouse.get\_pos()

                screen.blit(fw2, (pygame.mouse.get\_pos()[0]-5, pygame.mouse.get\_pos()[1]-10))

            if red == 2:

                pygame.draw.line(screen, (255, 0, 0), (269, 130), redpos, 20)

                screen.blit(fw2, (redpos[0]-10, redpos[1]-10))

            if green == 2:

                pygame.draw.line(screen, (0, 255, 0), (269, 230), greenpos, 20)

                screen.blit(fw2, (greenpos[0]-10, greenpos[1]-10))

            if blue == 2:

                pygame.draw.line(screen, (0, 0, 255), (269, 335), bluepos, 20)

                screen.blit(fw2, (bluepos[0]-10, bluepos[1]-10))

            if yellow == 2:

                pygame.draw.line(screen, (255, 255, 0), (269, 440), yellowpos, 20)

                screen.blit(fw2, (yellowpos[0]-10, yellowpos[1]-10))

            if red == green == blue == yellow == 2:

                self.tasks[1] = 1

                return 1

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[1] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def emptyGarbage(self):

        eg1 = pygame.image.load("models/tasks/Empty Garbage/garbage\_Base.png")

        eg2 = pygame.image.load("models/tasks/Empty Garbage/garbage\_lightShadow.png")

        eg3 = pygame.image.load("models/tasks/Empty Garbage/button.png")

        eg4 = pygame.image.load("models/tasks/Empty Garbage/garbage\_leverBars.png")

        eg5 = pygame.image.load("models/tasks/Empty Garbage/garbage\_leverHandle.png")

        gar1 = pygame.image.load("models/tasks/Empty Garbage/Nova pasta/diamond.png")

        gar2 = pygame.image.load("models/tasks/Empty Garbage/Nova pasta/garbage\_1.png")

        gar3 = pygame.image.load("models/tasks/Empty Garbage/Nova pasta/garbage\_2.png")

        gar4 = pygame.image.load("models/tasks/Empty Garbage/Nova pasta/garbage\_3.png")

        gar5 = pygame.image.load("models/tasks/Empty Garbage/Nova pasta/garbage\_4.png")

        gar6 = pygame.image.load("models/tasks/Empty Garbage/Nova pasta/garbage\_5.png")

        gar7 = pygame.image.load("models/tasks/Empty Garbage/Nova pasta/garbage\_6.png")

        gar8 = pygame.image.load("models/tasks/Empty Garbage/Nova pasta/teleporter.png")

        gar9 = pygame.image.load("models/tasks/Empty Garbage/Nova pasta/totem.png")

        garPos = []

        garba = [gar1, gar2, gar3, gar4, gar5, gar6, gar7, gar8, gar9]

        for i in range(1,20):

                garPos.append([garba[random.randint(0,8)], [random.randint(261, 539), random.randint(270, 469)]])

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(eg1, (256, 28))

            screen.blit(eg2, (259, 32))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                self.tasks[2] = 0

                return 0

            if 652 < pygame.mouse.get\_pos()[0] < 717 and 63 < pygame.mouse.get\_pos()[1] < 168 and pygame.mouse.get\_pressed()[0]:

                screen.blit(eg3, (651, 68))

                tot = 0

                for i in range(len(garPos)):

                    garPos[i][1][1] += 5

                    tot += garPos[i][1][1]

                if tot > 550 \* len(garPos):

                    self.tasks[2] = 1

                    return 1

            else:

                screen.blit(eg3, (651, 60))

            for i in garPos:

                screen.blit(i[0], i[1])

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[2] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def upload(self, Download = 0):

        #upload

        uplDon = 0

        down = 1

        man = 1

        speed = 0

        up1 = pygame.image.load("models/tasks/Upload Data/dataTransfer\_Base.png")

        up2 = pygame.image.load("models/tasks/Upload Data/dataTransfer\_progressBar.png")

        up3 = pygame.image.load("models/tasks/Upload Data/dataTransfer\_downloadButton.png")

        up4 = pygame.image.load("models/tasks/Upload Data/dataTransfer\_folderOpen0001.png")

        up4 = pygame.transform.scale(up4, (10, 10))

        up5 = pygame.image.load("models/tasks/Upload Data/dataTransfer\_uploadButton.png")

        while uplDon <= 300:

            screen.fill((0, 0, 0))

            screen.blit(up1, (256, 90))

            screen.blit(up2, (316, 288))

            screen.blit(up3, (462, 316))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                self.tasks[3] = 1

                return 0

            if Download == 1:

                screen.blit(up5, (462, 316))

            if 465 < pygame.mouse.get\_pos()[0] < 553 and 321 < pygame.mouse.get\_pos()[1] < 340 and pygame.mouse.get\_pressed()[0]:

                down = 0

            if down == 0:

                uplDon += 1 #360, 636

                if Download != 0:

                    screen.blit(pygame.image.load(f"images/Sprites/Walk/walkcolor00{int(man)}.png"), (360+speed,191))

                else:

                    screen.blit(pygame.transform.flip(pygame.image.load(f"images/Sprites/Walk/walkcolor00{int(man)}.png"), True, False), (636-speed,191))

                man += 0.5

                speed += 5

                if speed > 250:

                    speed = 0

                if man == 13:

                    man = 1

                screen.blit(pygame.transform.scale(up4, (10\*(uplDon//8), 10)), (319, 291))

            '''if down != 0:

                if Download != 1:

                    screen.blit(up4, (pygame.mouse.get\_pos()[0], pygame.mouse.get\_pos()[1]))

                else:

                    screen.blit(up5, (pygame.mouse.get\_pos()[0], pygame.mouse.get\_pos()[1]))'''

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[3] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

        self.tasks[3] = 1

        return 1

    def Download(self, Download):

        #upload

        uplDon = 0

        down = 1

        man = 1

        speed = 0

        up1 = pygame.image.load("models/tasks/Upload Data/dataTransfer\_Base.png")

        up2 = pygame.image.load("models/tasks/Upload Data/dataTransfer\_progressBar.png")

        up3 = pygame.image.load("models/tasks/Upload Data/dataTransfer\_downloadButton.png")

        up4 = pygame.image.load("models/tasks/Upload Data/dataTransfer\_folderOpen0001.png")

        up4 = pygame.transform.scale(up4, (10, 10))

        up5 = pygame.image.load("models/tasks/Upload Data/dataTransfer\_uploadButton.png")

        while uplDon <= 300:

            screen.fill((0, 0, 0))

            screen.blit(up1, (256, 90))

            screen.blit(up2, (316, 288))

            screen.blit(up3, (462, 316))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                self.tasks[4] = 1

                return 0

            if Download == 1:

                screen.blit(up5, (462, 316))

            if 465 < pygame.mouse.get\_pos()[0] < 553 and 321 < pygame.mouse.get\_pos()[1] < 340 and pygame.mouse.get\_pressed()[0]:

                down = 0

            if down == 0:

                uplDon += 1 #360, 636

                if Download != 0:

                    screen.blit(pygame.image.load(f"images/Sprites/Walk/walkcolor00{int(man)}.png"), (360+speed,191))

                else:

                    screen.blit(pygame.transform.flip(pygame.image.load(f"images/Sprites/Walk/walkcolor00{int(man)}.png"), True, False), (636-speed,191))

                man += 0.5

                speed += 5

                if speed > 250:

                    speed = 0

                if man == 13:

                    man = 1

                screen.blit(pygame.transform.scale(up4, (10\*(uplDon//8), 10)), (319, 291))

            '''if down != 0:

                if Download != 1:

                    screen.blit(up4, (pygame.mouse.get\_pos()[0], pygame.mouse.get\_pos()[1]))

                else:

                    screen.blit(up5, (pygame.mouse.get\_pos()[0], pygame.mouse.get\_pos()[1]))'''

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[4] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

        self.tasks[4] = 1

        return 1

    def clearLeaves(self):

        tot = 0

        cllev1 = pygame.image.load("models/tasks/Clean O2 Filter/o2\_bgBase.png")

        cllev2 = pygame.image.load("models/tasks/Clean O2 Filter/o2\_bgTop.png")

        levPos = [(random.randint(390, 649), random.randint(109, 400)) for i in range(7)]

        leaves = [pygame.image.load(f"models/tasks/Clean O2 Filter/o2\_leafs/o2\_leaf{i}.png") for i in range(1,8)]

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(cllev1, (263, 29))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            for i in range(len(leaves)):

                if levPos[i][0] < pygame.mouse.get\_pos()[0] < levPos[i][0]+100:

                    if levPos[i][1] < pygame.mouse.get\_pos()[1] < levPos[i][1]+100:

                        if pygame.mouse.get\_pressed()[0]:

                            levPos[i] = pygame.mouse.get\_pos()[0]-50, pygame.mouse.get\_pos()[1]-50

                if levPos[i][0] < 309 and 157 < levPos[i][1] < 340:

                    levPos[i] = (0, -200)

                    tot += 1

                screen.blit(leaves[i], levPos[i])

                screen.blit(cllev2, (263, 29))

            if tot == 7:

                self.tasks[5] = 1

                return 1

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[5] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def alignEngine(self):

        enDon = 1

        enp = [132, 205]

        count = 1

        en1 = pygame.image.load("models/tasks/Align Engine Output/engineAlign\_base.png")

        en2 = pygame.image.load("models/tasks/Align Engine Output/engineAlign\_slider.png")

        en3 = pygame.image.load("models/tasks/Align Engine Output/engineAlign\_engine.png")

        en4 = pygame.image.load("models/tasks/Align Engine Output/engineAlign\_engine\_green.png")

        en5 = pygame.image.load("models/tasks/Align Engine Output/green.png")

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(en5, (286, 54))

            screen.blit(en1, (253, 22))

            screen.blit(en4, (220, 170))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            if pygame.mouse.get\_pressed()[0] and 126 < pygame.mouse.get\_pos()[1] < 379 and 614 < pygame.mouse.get\_pos()[0] < 687:

                if enp[1]-20 < pygame.mouse.get\_pos()[1] < enp[1]+20:

                    enp[1] = pygame.mouse.get\_pos()[1]

                if 240 < pygame.mouse.get\_pos()[1] < 260:

                    count += 1

                    if count == 100:

                        self.tasks[6] = 1

                        return 1

                else:

                    count = 0

            screen.blit(en2, (611, enp[1]))

            screen.blit(en3, (220, enp[1]-73))

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[6] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def calibrate(self):

        calDon = 1

        cal1 = pygame.image.load("models/tasks/Calibrate Distributor/CalibratorBaseWWires.png")

        cal2 = pygame.image.load("models/tasks/Calibrate Distributor/calibratorButton.png")

        cal3 = pygame.image.load("models/tasks/Calibrate Distributor/calibratorGauge.png")

        cal4 = pygame.image.load("models/tasks/Calibrate Distributor/calibratorSpin1.png")

        w, h = cal4.get\_size()

        angles = [random.randint(0, 360) for i in range(3)]

        angle1, angle2, angle3 = [random.randint(0, 132) for i in range(3)]

        calDoing = [0, 0, 0]

        cal5 = pygame.image.load("models/tasks/Calibrate Distributor/calibratorSpin2.png")

        cal6 = pygame.image.load("models/tasks/Calibrate Distributor/calibratorSpin3.png")

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(cal1, (258, 24))

            screen.blit(cal2, (609, 124))

            screen.blit(cal2, (609, 271))

            screen.blit(cal2, (609, 421))

            screen.blit(cal3, (597, 385))

            screen.blit(cal3, (597, 231))

            screen.blit(cal3, (597, 82))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            if calDoing[0] == 0:

                angle1 += 1

                if angle1 > 132:

                    angle1 = 0

            yellow = pygame.surface.Surface([angle1, 32])

            yellow.fill((255, 255, 0))

            screen.blit(yellow, (597, 82))

            if calDoing[1] == 0:

                angle2 += 1

                if angle2 > 132:

                    angle2 = 0

            blue = pygame.surface.Surface([angle2, 32])

            blue.fill((0, 102, 204))

            screen.blit(blue, (597, 231))

            if calDoing[2] == 0:

                angle3 += 1

                if angle3 > 132:

                    angle3 = 0

            cyan = pygame.surface.Surface([angle3, 32])

            cyan.fill((102, 255, 255))

            screen.blit(cyan, (597, 385))

            angles = [i+1 if i !=360 else 0 for i in angles]

            if calDoing[0] == 0:

                self.blitRotate(screen, cal4, (369, 123), (w/2, h/2), angles[0])

            else:

                self.blitRotate(screen, cal4, (369, 123), (w/2, h/2), 0)

            if calDoing[1] == 0:

                self.blitRotate(screen, cal5, (369, 276), (w/2, h/2), angles[1])

            else:

                self.blitRotate(screen, cal5, (369, 276), (w/2, h/2), 0)

            if calDoing[2] == 0:

                self.blitRotate(screen, cal6, (369, 422), (w/2, h/2), angles[2])

            else:

                self.blitRotate(screen, cal6, (369, 422), (w/2, h/2), 0)

            if 612 < pygame.mouse.get\_pos()[0] < 716 and 128 < pygame.mouse.get\_pos()[1] < 152 and calDoing[0] == 0:

                if pygame.mouse.get\_pressed()[0]:

                    screen.blit(cal2, (609, 130))

                    if 0.1 < angles[0]/360 < 0.9:

                        angles = [random.randint(0, 360) for i in range(3)]

                        calDoing = [0 for i in range(3)]

                    else:

                        if calDoing[1] == calDoing[2] == 0:

                            calDoing[0] = 1

                else:

                    screen.blit(cal2, (609, 124))

            elif 612 < pygame.mouse.get\_pos()[0] < 716 and 276 < pygame.mouse.get\_pos()[1] < 300 and calDoing[1] == 0:

                if pygame.mouse.get\_pressed()[0]:

                    screen.blit(cal2, (609, 278))

                    if 0.1 < angles[1]/360 < 0.9:

                        angles = [random.randint(0, 360) for i in range(3)]

                        calDoing = [0 for i in range(3)]

                    else:

                        if calDoing[0] == 1 and calDoing[2] == 0:

                            calDoing[1] = 1

                else:

                    screen.blit(cal2, (609, 271))

            elif 612 < pygame.mouse.get\_pos()[0] < 716 and 427 < pygame.mouse.get\_pos()[1] < 450 and calDoing[2] == 0:

                if pygame.mouse.get\_pressed()[0]:

                    screen.blit(cal2, (609, 427))

                    if 0.1 < angles[2]/360 < 0.9:

                        angles = [random.randint(0, 360) for i in range(3)]

                        calDoing = [0 for i in range(3)]

                    else:

                        if calDoing[0] == calDoing[1] == 1:

                            calDoing[2] = 1

                else:

                    screen.blit(cal2, (609, 421))

            if calDoing[0] == calDoing[1] == calDoing[2] == 1:

                self.tasks[7] = 1

                return 1

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[7] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def chartCourse(self):

        ccDon = 1

        cc1 = pygame.image.load("models/tasks/Chart Course/nav\_chartCourse\_base.png")

        cc2 = pygame.image.load("models/tasks/Chart Course/nav\_chartCourse\_checkPt.png")

        cc3 = pygame.image.load("models/tasks/Chart Course/nav\_chartCourse\_checkPtShadow.png")

        cc4 = pygame.image.load("models/tasks/Chart Course/nav\_chartCourse\_end.png")

        cc5 = pygame.image.load("models/tasks/Chart Course/nav\_chartCourse\_endShadow.png")

        cc6 = pygame.image.load("models/tasks/Chart Course/nav\_chartCourse\_ship.png")

        cc7 = pygame.image.load("models/tasks/Chart Course/nav\_chartCourse\_start.png")

        cc8 = pygame.image.load("models/tasks/Chart Course/nav\_chartCourse\_startShadow.png")

        angle = 0

        ccpos = [344, 430, 507, 597, 673]

        ccPos = [(i, random.randint(178, 353)) for i in ccpos]

        ship = [0, 0, 0, 0, 0]

        while True:

            screen.fill((0, 0, 0, 5))

            angle += 1

            screen.blit(cc1, (253, 118))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            for i in ccPos[1:]:

                screen.blit(cc2, (i[0]-cc2.get\_size()[0]//2, i[1]-cc2.get\_size()[1]//2))

                screen.blit(cc8, (i[0]-cc7.get\_size()[0]//2, i[1]-cc7.get\_size()[1]//2))

            screen.blit(cc7, (ccPos[0][0]-cc7.get\_size()[0]//2, ccPos[0][1]-cc7.get\_size()[1]//2))

            self.blitRotate(screen , cc4, ccPos[-1], (cc4.get\_size()[0]//2, cc4.get\_size()[1]//2), angle)

            for i in range(len(ccPos)-1):

                self.draw\_dashed\_line(screen, (0, 0, 0), ccPos[i], ccPos[i+1])

            if ship[0] == 0:

                screen.blit(cc6, (ccPos[0][0]-17, ccPos[0][1]-22))

                if ccPos[0][0]-20 < pygame.mouse.get\_pos()[0] < ccPos[0][0]+20 and ccPos[0][1]-20 < pygame.mouse.get\_pos()[1] < ccPos[0][1]+20:

                    if pygame.mouse.get\_pressed()[0]:

                        ship[0] = 1

            elif ship[1] == 0:

                screen.blit(cc6, (ccPos[1][0]-17, ccPos[1][1]-22))

                if ccPos[1][0]-20 < pygame.mouse.get\_pos()[0] < ccPos[1][0]+20 and ccPos[1][1]-20 < pygame.mouse.get\_pos()[1] < ccPos[1][1]+20:

                    if pygame.mouse.get\_pressed()[0]:

                        ship[1] = 1

            elif ship[2] == 0:

                screen.blit(cc6, (ccPos[2][0]-17, ccPos[2][1]-22))

                if ccPos[2][0]-20 < pygame.mouse.get\_pos()[0] < ccPos[2][0]+20 and ccPos[2][1]-20 < pygame.mouse.get\_pos()[1] < ccPos[2][1]+20:

                    if pygame.mouse.get\_pressed()[0]:

                        ship[2] = 1

            elif ship[3] == 0:

                screen.blit(cc6, (ccPos[3][0]-17, ccPos[3][1]-22))

                if ccPos[3][0]-20 < pygame.mouse.get\_pos()[0] < ccPos[3][0]+20 and ccPos[3][1]-20 < pygame.mouse.get\_pos()[1] < ccPos[3][1]+20:

                    if pygame.mouse.get\_pressed()[0]:

                        ship[3] = 1

            else:

                self.tasks[8] = 1

                return 1

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[8] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def weapons(self):

        waDon = 1

        wa1 = pygame.image.load("models/tasks/Clear Asteroids/weapons\_base.png")

        wa2 = pygame.image.load("models/tasks/Clear Asteroids/weapons\_target.png")

        ast1 = pygame.image.load("models/tasks/Clear Asteroids/weapons\_asteroid1.png")

        ast2 = pygame.image.load("models/tasks/Clear Asteroids/weapons\_asteroid2.png")

        ast3 = pygame.image.load("models/tasks/Clear Asteroids/weapons\_asteroid3.png")

        ast4 = pygame.image.load("models/tasks/Clear Asteroids/weapons\_asteroid4.png")

        ast5 = pygame.image.load("models/tasks/Clear Asteroids/weapons\_asteroid5.png")

        ((322,92), (674, 453))

        astPos = [[(i\*322)+322\*2, random.randint(92, 400)] for i in range(10)]

        tarPos = [wa1.get\_size()[0]//2 + 247, wa1.get\_size()[1]//2 + 23]

        asDes = 0

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(wa1, (247, 23))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            for i in range(len(astPos)):

                astPos[i][0] -= 5

            if  277 < astPos[0][0] < 660:

                screen.blit(ast1, astPos[0])

            if  277 < astPos[5][0] < 660:

                screen.blit(ast1, astPos[5])

            if  277 < astPos[1][0] < 660:

                screen.blit(ast2, astPos[1])

            if  277 < astPos[6][0] < 660:

                screen.blit(ast2, astPos[6])

            if  277 < astPos[2][0] < 660:

                screen.blit(ast3, astPos[2])

            if  277 < astPos[7][0] < 660:

                screen.blit(ast3, astPos[7])

            if  277 < astPos[3][0] < 660:

                screen.blit(ast4, astPos[3])

            if  277 < astPos[8][0] < 660:

                screen.blit(ast4, astPos[8])

            if  277 < astPos[4][0] < 660:

                screen.blit(ast5, astPos[4])

            if  277 < astPos[9][0] < 660:

                screen.blit(ast5, astPos[9])

            if 322 < pygame.mouse.get\_pos()[0] < 674 and 92 < pygame.mouse.get\_pos()[1] < 453:

                if pygame.mouse.get\_pressed()[0]:

                    tarPos = pygame.mouse.get\_pos()

                    for i in range(len(astPos)):

                        if astPos[i][0] < tarPos[0] < astPos[i][0]+100 and astPos[i][1] < tarPos[1] < astPos[i][1]+100:

                            astPos[i] = [0, -200]

                            asDes += 1

            screen.blit(wa2, (tarPos[0] - wa2.get\_size()[0]//2, tarPos[1] - wa2.get\_size()[1]//2))

            if asDes > 9:

                self.tasks[9] = 1

                return 1

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[9] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def divertPower(self, station):

        dpDon = 1

        dp1 = pygame.image.load("models/tasks/Divert Power/electricity\_Divert\_Base.png")

        dp2 = pygame.image.load("models/tasks/Divert Power/electricity\_Divert\_switch.png")

        dpBut = [[288+54\*i, 387] for i in range(8)]

        dpto = [0, 0, 0, 0, 0, 0, 0, 0]

        dpto[station] = 1

        to = 0

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(dp1, (249, 26))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            for i in dpBut:

                ind = dpBut.index(i)

                if dpto[ind] == 1:

                    if i[0] < pygame.mouse.get\_pos()[0] < i[0]+45 and i[1] < pygame.mouse.get\_pos()[1] < i[1]+30:

                        if pygame.mouse.get\_pressed()[0]:

                            dpBut[ind][1] = pygame.mouse.get\_pos()[1]-15

                            if dpBut[ind][1] < 350:

                                dpto[ind] = 0

                            if dpBut[ind][1] > 450:

                                dpBut[ind][1] = 445

                screen.blit(dp2, i)

            for i in range(len(dpto)):

                if dpto[i] == 1:

                    to = i

                sur = pygame.surface.Surface([15, 50])

                sur.fill((255, 255, 0)) #301, 227, 354, 228

                screen.blit(sur, (305 + i\*54, 227))

            if dpto[to] == 0:

                screen.blit(sur, (305 + to\*54, 200))

                self.tasks[10] = 1

                return 1

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[10] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def fualEngine(self):

        feDon = 1

        fe1 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_fillBase.png")

        fe2 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_wires.png")

        fe3 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_buttonBase.png")

        fe4 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_Button.png")

        fe5 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_Light.png")

        fe6 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_LightRed.png")

        en = 0

        while True:

            screen.fill((0, 0, 0))

            sur = pygame.surface.Surface([90, 65])

            sur.fill((255, 255, 0))

            screen.blit(sur, (498+en, 45))

            #444, 118, 527, 432

            sur = pygame.surface.Surface([83, 314])

            sur.fill((255, 255, 0))

            screen.blit(sur, (444, 432-int(en)\*5))

            screen.blit(fe1, (324, 23))

            if 432-int(en)\*5 < 118:

                self.tasks[11] = 1

                return 1

            screen.blit(fe2, (661, 397))

            screen.blit(fe3, (711, 356))

            screen.blit(fe4, (735, 379))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            if 735 < pygame.mouse.get\_pos()[0] < 811 and 379 < pygame.mouse.get\_pos()[1] < 456:

                if pygame.mouse.get\_pressed()[0]:

                    screen.blit(fe4, (735, 382))

                    en += 0.5

            screen.blit(fe5, (777, 335))

            screen.blit(fe6, (741, 335))

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[11] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def fillCan(self):

        en = 0

        fe1 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_fillBase.png")

        fe2 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_wires.png")

        fe3 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_buttonBase.png")

        fe4 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_Button.png")

        fe5 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_Light.png")

        fe6 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_LightRed.png")

        fc1 = pygame.image.load("models/tasks/Fuel Engines/engineFuel\_gasCanBase.png")

        while True:

            screen.fill((0, 0, 0, 5))

            sur = pygame.surface.Surface([300, 320]) #344, 80

            sur.fill((255, 255, 0))

            screen.blit(sur, (344, 413-int(en)\*5))

            screen.blit(fc1, (324, 23))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            if 413-int(en)\*5 < 105:

                self.tasks[12] = 1

                return 1

            screen.blit(fe2, (661, 397))

            screen.blit(fe3, (711, 356))

            screen.blit(fe4, (735, 379))

            if 735 < pygame.mouse.get\_pos()[0] < 811 and 379 < pygame.mouse.get\_pos()[1] < 456:

                if pygame.mouse.get\_pressed()[0]:

                    screen.blit(fe4, (735, 382))

                    en += 0.5

            screen.blit(fe5, (777, 335))

            screen.blit(fe6, (741, 335))

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[12] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def inspectSample(self):

        isDon = 1

        is1 = pygame.image.load("models/tasks/Inspect Sample/medBay\_back.png")

        is2 = pygame.image.load("models/tasks/Inspect Sample/medBay\_panelCenter.png")

        is3 = pygame.image.load("models/tasks/Inspect Sample/medBay\_glassBack.png")

        is4 = pygame.image.load("models/tasks/Inspect Sample/medBay\_glassFrontTestTubes.png")

        is5 = pygame.image.load("models/tasks/Inspect Sample/medBay\_dispenser.png")

        is6 = pygame.image.load("models/tasks/Inspect Sample/medBay\_liquid\_filled.png")

        is7 = pygame.image.load("models/tasks/Inspect Sample/medBay\_panelBottom.png")

        is8 = pygame.image.load("models/tasks/Inspect Sample/medBay\_buttonConfirm.png")

        is9 = pygame.image.load("models/tasks/Inspect Sample/medBay\_liquid\_anom.png")

        is10 = pygame.image.load("models/tasks/Inspect Sample/medBay\_sampleButton\_green.png")

        is11 = pygame.image.load("models/tasks/Inspect Sample/medBay\_sampleButton\_red.png")

        is12 = pygame.image.load("models/tasks/Inspect Sample/medBay\_liquid\_filled.png")

        anom = random.randint(0,4)

        before = 0

        fills = [0, 0, 0, 0, 0]

        statfilling = 0

        fillingDon = 0

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(is1, (250, 22))

            screen.blit(is2, (257, 274))

            screen.blit(is3, (285, 221))

            screen.blit(is4, (285, 184))

            screen.blit(is7, (248, 406))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            if statfilling == 1:

                for i in range(len(fills)):

                    if fills[i] == 100:

                        screen.blit(is6, (360+63\*i, 233))

                        if fillingDon == 1:

                            before += 1

                            if anom == i and before > 1000:

                                screen.blit(is9, (360+63\*i, 233))

                                screen.blit(is11, (360+63\*i, 419))

                                if (360+63\*i) < pygame.mouse.get\_pos()[0] < (40+360+63\*i) and (419) < pygame.mouse.get\_pos()[1] < (40+419):

                                    if pygame.mouse.get\_pressed()[0]:

                                        self.tasks[13] = 1

                                        return 1

                            elif before > 1000:

                                screen.blit(is10, (360+63\*i, 419))

                                if (360+63\*i) < pygame.mouse.get\_pos()[0] < (40+360+63\*i) and (419) < pygame.mouse.get\_pos()[1] < (40+419):

                                    if pygame.mouse.get\_pressed()[0]:

                                        anom = random.randint(0,4)

                                        before = 0

                                        fills = [0, 0, 0, 0, 0]

                                        statfilling = 0

                                        fillingDon = 0

                    if fills[i] < 100:

                        screen.blit(is5, (355+60\*i, 25))

                        fills[i] += 2

                        break

            if 652 < pygame.mouse.get\_pos()[0] < 684 and 468 < pygame.mouse.get\_pos()[1] < 498 and pygame.mouse.get\_pressed()[0]:

                statfilling = 1

            tot = 0

            for i in fills:

                tot += i

            if tot == 500:

                fillingDon = 1

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[13] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def primeShield(self):

        psDon = 1

        ps1 = pygame.image.load("models/tasks/Prime Shields/shield\_screen.png")

        ps2 = pygame.image.load("models/tasks/Prime Shields/shield\_Panel.png")

        ps3 = pygame.image.load("models/tasks/Prime Shields/shield\_Panel\_red.png")

        ps4 = pygame.image.load("models/tasks/Prime Shields/shield\_Gauge100.png")

        psPos = [(417, 203), (538, 131), (541, 273), (419, 346), (295, 276), (295, 132), (416, 60)]

        psred = [random.randint(0, 1) for i in range(7)]

        angle = 0

        while True:

            screen.fill((0, 0, 0, 5))

            angle += 1

            screen.blit(ps1, (238, 25))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            for i in range(len(psPos)):

                screen.blit(ps2, psPos[i])

                if psred[i] == 1:

                    screen.blit(ps3, psPos[i])

                    if psPos[i][0] < pygame.mouse.get\_pos()[0] < psPos[i][0]+152 and psPos[i][1] < pygame.mouse.get\_pos()[1] < psPos[i][1]+152:

                        if pygame.mouse.get\_pressed()[0]:

                            psred[i] = 0

            self.blitRotate(screen , ps4, (493, 268), (ps4.get\_size()[0]//2, ps4.get\_size()[1]//2), angle)

            #screen.blit(ps4, pygame.mouse.get\_pos())

            tot = 0

            for i in psred:

                tot += i

            if tot == 0:

                self.tasks[14] = 1

                return 1

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[14] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def stabSteering(self):

        ssDon = 1

        ss1 = pygame.image.load("models/tasks/Stabilize Steering/nav\_stabilize\_base.png")

        ss2 = pygame.image.load("models/tasks/Stabilize Steering/nav\_stabilize\_graph.png")

        ss3 = pygame.image.load("models/tasks/Stabilize Steering/nav\_stabilize\_target.png")

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(ss2, (261, 42))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            if pygame.mouse.get\_pressed()[0]:

                if 251 < pygame.mouse.get\_pos()[0] < 730 and 34 < pygame.mouse.get\_pos()[1] < 511:

                    pygame.draw.line(screen, (255, 255, 255), (pygame.mouse.get\_pos()[0],34), (pygame.mouse.get\_pos()[0],511), 5)

                    pygame.draw.line(screen, (255, 255, 255), (251, pygame.mouse.get\_pos()[1]), (730, pygame.mouse.get\_pos()[1]), 5)

                    screen.blit(ss3, (pygame.mouse.get\_pos()[0]-ss3.get\_size()[0]//2, pygame.mouse.get\_pos()[1]-ss3.get\_size()[1]//2))

                    if 487 < pygame.mouse.get\_pos()[0] < 500 and 269 < pygame.mouse.get\_pos()[1] < 281:

                        if pygame.mouse.get\_pressed()[0]:

                            self.tasks[15] = 1

                            return 1

            screen.blit(ss1, (241, 23))

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[15] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def unlockManifolds(self):

        umDon = 1

        um1 = pygame.image.load("models/tasks/Unlock Manifolds/reactorPanel.png")

        um2 = pygame.image.load("models/tasks/Unlock Manifolds/reactorPanelGlass.png")

        um3 = pygame.image.load("models/tasks/Unlock Manifolds/reactorWire.png")

        um4 = pygame.image.load("models/tasks/Unlock Manifolds/red.png")

        nums = {i:pygame.image.load(f"models/tasks/Unlock Manifolds/reactorButton{i}.png") for i in range(1, 11)}

        numPos = []

        numDid = 1

        while len(numPos) != 10:

            i = random.randint(1, 10)

            if i not in numPos:

                numPos.append(i)

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(um1, (255, 151))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            for i in range(1, 11):

                if i > 5:

                    screen.blit(nums[numPos[i-1]], (292+(i-6)\*85, 278))

                    if numPos[i-1] < numDid:

                        screen.blit(um4, (292+(i-6)\*85, 278))

                    if 292+(i-6)\*85 < pygame.mouse.get\_pos()[0] < 83+292+(i-6)\*85 and 278 < pygame.mouse.get\_pos()[1] < 278+82:

                        if pygame.mouse.get\_pressed()[0]:

                            if numDid == numPos[i-1]:

                                numDid += 1

                else:

                    screen.blit(nums[numPos[i-1]], (292+(i-1)\*85, 192))

                    if numPos[i-1] < numDid:

                        screen.blit(um4, (292+(i-1)\*85, 192))

                    if 292+(i-1)\*85 < pygame.mouse.get\_pos()[0] < 82+292+(i-1)\*85 and 192 < pygame.mouse.get\_pos()[1] < 192+82:

                        if pygame.mouse.get\_pressed()[0]:

                            if numDid == numPos[i-1]:

                                numDid += 1

            if numDid == 11:

                self.tasks[16] = 1

                return 1

            screen.blit(um2, (288, 187))

            screen.blit(um3, (208, 89))

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[16] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def starReactor(self):

        srDon = 1

        sr1 = pygame.image.load("models/tasks/Start Reactor/simonSaysBase.png")

        sr2 = pygame.image.load("models/tasks/Start Reactor/simonSaysButtonsShadow.png")

        sr3 = pygame.image.load("models/tasks/Start Reactor/simonSaysLightsIndicationWShadows.png")

        sr4 = pygame.image.load("models/tasks/Start Reactor/simonSaysScreen.png")

        sr5 = pygame.image.load("models/tasks/Start Reactor/ssbutton.png")

        sr6 = pygame.image.load("models/tasks/Start Reactor/ssbuttonblue.png")

        sr7 = pygame.image.load("models/tasks/Start Reactor/ssbuttonred.png")

        srDoing = 1

        sr = [random.randint(0,8) for i in range(srDoing)]

        l = 0

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(sr1, (243, 118))

            screen.blit(sr1, (520, 118))

            screen.blit(sr2, (572, 195))

            screen.blit(pygame.image.load(f"models/tasks/Start Reactor/{srDoing}.png"), (290, 163))

            screen.blit(pygame.image.load(f"models/tasks/Start Reactor/{len(sr)}.png"), (564, 163))

            screen.blit(sr4, (290, 199))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            for i in range(9):

                #screen.blit(sr5, (575+i\*50, 200))

                if i >= 6:

                    screen.blit(sr5, (575+(i-6)\*50, 300))

                    if i == sr[0]:

                        screen.blit(sr6, (295+(i-6)\*50, 300))

                elif i >= 3:

                    screen.blit(sr5, (575+(i-3)\*50, 250))

                    if i == sr[0]:

                        screen.blit(sr6, (295+(i-3)\*50, 250))

                else:

                    screen.blit(sr5, (575+i\*50, 200))

                    if i == sr[0]:

                        screen.blit(sr6, (295+i\*50, 200))

            butPress = None

            for i in range(9):

                if i>=6:

                    if 575+(i-6)\*50 < pygame.mouse.get\_pos()[0] < 40+575+(i-6)\*50 and 300 < pygame.mouse.get\_pos()[1] < 300+40:

                        if pygame.mouse.get\_pressed()[0]:

                            butPress = i

                elif i>=3:

                    if 575+(i-3)\*50 < pygame.mouse.get\_pos()[0] < 40+575+(i-3)\*50 and 250 < pygame.mouse.get\_pos()[1] < 250+40:

                        if pygame.mouse.get\_pressed()[0]:

                            butPress = i

                else:

                    if 575+i\*50 < pygame.mouse.get\_pos()[0] < 40+575+i\*50 and 200 < pygame.mouse.get\_pos()[1] < 200+40:

                        if pygame.mouse.get\_pressed()[0]:

                            butPress = i

            if butPress != None:

                if butPress == sr[0]:

                    del sr[0]

                    if len(sr) == 0:

                        srDoing += 1

                        sr = [random.randint(0,8) for i in range(srDoing)]

                        l = 0

                    if srDoing == 6:

                        self.tasks[17] = 1

                        return 1

                else:

                    l += 1

                    if l > 10:

                        srDoing = 1

                        sr = [random.randint(0,8) for i in range(srDoing)]

                        l = 0

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[17] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def acceptPower(self):

        apDon = 1

        ap1 = pygame.image.load("models/tasks/Accept Power/1.png")

        ap2 = pygame.image.load("models/tasks/Accept Power/2.png")

        ap3 = pygame.image.load("models/tasks/Accept Power/3.png")

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(ap2, (160, 58))

            screen.blit(ap1, (482, 198))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            if 458 < pygame.mouse.get\_pos()[0] < 552 and 238 < pygame.mouse.get\_pos()[1] < 311 and pygame.mouse.get\_pressed()[0]:

                self.tasks[18] = 1

                return 1

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[18] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def medbayScan(self):

        msDon = 1

        ms1 = pygame.image.load("models/tasks/Submite Scan/medbayScan\_panelBottom.png")

        ms2 = pygame.image.load("models/tasks/Submite Scan/medbayScan\_panelTop.png")

        ms3 = pygame.image.load("models/tasks/Submite Scan/medbayScan\_wires.png")

        num = 1

        n = 0

        while True:

            screen.fill((0, 0, 0, 5))

            screen.blit(ms2, (250, 7))

            screen.blit(ms3, (719, 72))

            screen.blit(ms1, (250, 372))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            num += 0.1

            if num >= 9:

                num = 1

                n += 1

            if n > 2:

                self.tasks[19] = 1

                return 1

            screen.blit(pygame.image.load(f"models/scan/{int(num)}.png"), (488, 256))

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[19] = 0

                    return 0

            pygame.display.update()

            clock.tick(fps)

    def electrical(self):

        e1 = pygame.image.load("models/sabotages/e1.png")

        e2 = pygame.image.load("models/sabotages/e2.png")

        e3 = pygame.image.load("models/sabotages/e3.png")

        on = [0, 0, 0, 0, 0]

        while True:

            screen.fill(0)

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            screen.blit(e1, (310, 75))

            for i in range(len(on)):

                if on[i] == 0:

                    screen.blit(e2, (334+80\*i, 373)) #334, 373, 414, 373, 334, 356

                else:

                    screen.blit(e3, (334+80\*i, 356))

            if sum(on) == 5:

                return 1

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[20] = 0

                    return 0

                if event.type == pygame.MOUSEBUTTONDOWN:

                    for i in range(len(on)):

                        if 334+80\*i < pygame.mouse.get\_pos()[0] < 359+80\*i and 373 < pygame.mouse.get\_pos()[1] < 473:

                            if on[i] == 0:

                                on[i] = 1

            pygame.display.update()

            clock.tick(fps)

    def oxygen(self):

        o2 = pygame.image.load("models/sabotages/o2.png")

        o2 = pygame.transform.scale(o2, (360, 500))

        num = ''

        resul = random.randint(10000, 99999)

        font\_size = 48

        font = pygame.font.Font('freesansbold.ttf', font\_size)

        Text1 = str(resul)

        text1 = font.render(Text1, True, (255, 255, 0))

        textRect1 = text1.get\_rect()

        while True:

            screen.fill(0)

            screen.blit(o2, (261, 29))

            screen.blit(text1, (711, 241))

            Text2 = num

            text2 = font.render(Text2, True, (0, 255, 0))

            textRect2 = text2.get\_rect()

            screen.blit(text2, (373, 75))

            screen.blit(close, (100, 25))

            if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                return 0

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.tasks[21] = 0

                    return 0

                if event.type == pygame.MOUSEBUTTONDOWN:

                    if len(num) < 5:

                        if 318 < pygame.mouse.get\_pos()[0] < 392 and 155 < pygame.mouse.get\_pos()[1] < 229:

                            num += '1'

                        if 404 < pygame.mouse.get\_pos()[0] < 480 and 153 < pygame.mouse.get\_pos()[1] < 231:

                            num += '2'

                        if 491 < pygame.mouse.get\_pos()[0] < 568 and 155 < pygame.mouse.get\_pos()[1] < 229:

                            num += '3'

                        if 318 < pygame.mouse.get\_pos()[0] < 392 and 243 < pygame.mouse.get\_pos()[1] < 318:

                            num += '4'

                        if 403 < pygame.mouse.get\_pos()[0] < 480 and 243 < pygame.mouse.get\_pos()[1] < 318:

                            num += '5'

                        if 491 < pygame.mouse.get\_pos()[0] < 568 and 243 < pygame.mouse.get\_pos()[1] < 318:

                            num += '6'

                        if 318 < pygame.mouse.get\_pos()[0] < 392 and 336 < pygame.mouse.get\_pos()[1] < 409:

                            num += '7'

                        if 403 < pygame.mouse.get\_pos()[0] < 480 and 336 < pygame.mouse.get\_pos()[1] < 409:

                            num += '8'

                        if 491 < pygame.mouse.get\_pos()[0] < 568 and 336 < pygame.mouse.get\_pos()[1] < 409:

                            num += '9'

                        if 405 < pygame.mouse.get\_pos()[0] < 478 and 425 < pygame.mouse.get\_pos()[1] < 500:

                            num += '0'

                    if 318 < pygame.mouse.get\_pos()[0] < 392 and 425 < pygame.mouse.get\_pos()[1] < 500:

                        num = num[:-1]

                    if 491 < pygame.mouse.get\_pos()[0] < 568 and 425 < pygame.mouse.get\_pos()[1] < 500:

                        if num == str(resul):

                            return 1

                        else:

                            num = ''

            pygame.display.update()

            clock.tick(fps)

RANDOMIZER.PY

import random

wiring = [5, 10, 17, 42, 37, 25, 31]

powerTo = [36, 34, 2, 15, 11, 17, 6, 30]

dowloadable = [4, 1, 12, 19, 23]

garbage = [3, 9]

tasks = [0, 14, 13, 16, 45, 43]

comp = [40, 8, 26]

def getWiring():

    return [[random.choice(wiring)]]

def getPowerTo():

    return [[24, random.choice(powerTo)]]

def getDownload():

    return [[random.choice(dowloadable), 38]]

def getChoice():

    x = random.choice([0, 1, 2])

    if x == 0:

        return [['Fual engines', 21, 35, 21, 32]]

    elif x == 1:

        return [['Allign Engines', 49, 33]]

    else:

        return [['MedBay', random.choice([28, 27])]]

def getGarbage():

    if random.choice([True, False]):

        return [['Empty Garbage', random.choice(garbage), 20]]

    else:

        return [[random.choice(tasks)]]

def getComp():

    return [[random.choice(comp)]]

def getAllTasks():

    AllTasks = getWiring() + getPowerTo() + getDownload() + getChoice() + getGarbage() + getComp()

    return AllTasks

FREE\_PLAY.PY

import pygame

from pygame.locals import \*

from walk\_anim import Player

import pickle

from Tasks import Tasks

from threading import Thread

pygame.init()

clock = pygame.time.Clock()

fps = 60

size =[1000, 550]

screen = pygame.display.set\_mode(size)

screen.set\_colorkey('#000000')

class Free\_play():

    def \_\_init\_\_(self):

        pass

    def run(self):

        from walk\_anim import Player

        from Tasks import Tasks

        tasksToDo = None

        a = 0

        b = 0

        c = 0

        cam\_on = pygame.image.load("models/map parts/cam-on.png")

        cam\_off = pygame.image.load("models/map parts/cam-off.png")

        redirect = pygame.image.load("models/map parts/redirect.png")

        electric = pygame.image.load("models/map parts/electric.png")

        upload = pygame.image.load("models/map parts/weapons/upload.png")

        #cafeteria

        bg = pygame.image.load('models/map parts/PC Computer - Among Us - Skeld Cafeteria.png')

        caflev = pygame.transform.flip(pygame.image.load("models/map parts/oxygen/o2-4.png"), True, False)

        cafup = upload

        cafred = pygame.transform.flip(pygame.image.load("models/map parts/weapons/redirect.png"), True, False)

        #weapons

        caf\_weap = pygame.image.load("models/map parts/weapons/caf-weapons.png")

        weapons1 = pygame.image.load("models/map parts/weapons/weapons-1.png")

        weapons2 = pygame.image.load("models/map parts/weapons/weapons-2.png")

        weapons3 = pygame.image.load("models/map parts/weapons/weapons-3.png")

        weapons4 = pygame.image.load("models/map parts/weapons/weapons-4.png")

        weapons5 = pygame.image.load("models/map parts/weapons/weapons-5.png")

        weapons6 = pygame.image.load("models/map parts/weapons/weapons-6.png")

        weapons7 = pygame.image.load("models/map parts/weapons/weapons-7.png")

        weapons8 = pygame.image.load("models/map parts/weapons/weapons-8.png")

        weapons9 = pygame.image.load("models/map parts/weapons/weapons-9.png")

        weapons10 = pygame.image.load("models/map parts/weapons/weapons-10.png")

        weaponselectric = pygame.image.load("models/map parts/weapons/redirect.png")

        weaponsupload = upload

        weaponsgreenscreen = pygame.image.load("models/map parts/weapons/greenscreen.png")

        #oxygen

        wep\_o2\_nav\_she = pygame.image.load("models/map parts/oxygen/wep-ox-nav-she.png")

        o21 = pygame.image.load("models/map parts/oxygen/o2-1.png")

        o22 = pygame.image.load("models/map parts/oxygen/o2-2.png")

        o23 = pygame.image.load("models/map parts/oxygen/o2-3.png")

        o24 = pygame.image.load("models/map parts/oxygen/o2-4.png")

        o25 = pygame.image.load("models/map parts/oxygen/o2-5.png")

        o26 = pygame.image.load("models/map parts/oxygen/o2-6.png")

        o27 = pygame.image.load("models/map parts/oxygen/o2-7.png")

        oredirect = redirect

        #navigation

        nav1 = pygame.image.load("models/map parts/navigation/nav-1.png")

        nav2 = pygame.image.load("models/map parts/navigation/nav-2.png")

        nav3 = pygame.image.load("models/map parts/navigation/nav-3.png")

        nav4 = pygame.image.load("models/map parts/navigation/nav-4.png")

        nav5 = pygame.image.load("models/map parts/navigation/nav-5.png")

        nav6 = pygame.image.load("models/map parts/navigation/nav-6.png")

        nav7 = pygame.image.load("models/map parts/navigation/nav-7.png")

        nav8 = pygame.image.load("models/map parts/navigation/nav-8.png")

        nav9 = pygame.image.load("models/map parts/navigation/nav-9.png")

        navredirect = redirect

        navelectric = electric

        #admin

        caf\_ad\_st = pygame.image.load("models/map parts/admin/admin-4.png")

        admin1 = pygame.image.load("models/map parts/admin/admin-1.png")

        admin2 = pygame.image.load("models/map parts/admin/admin-2.png")

        admin3 = pygame.image.load("models/map parts/admin/admin-3.png")

        admin4 = pygame.image.load("models/map parts/admin/admin-5.png")

        admin5 = pygame.image.load("models/map parts/admin/admin-6.png")

        admin6 = pygame.image.load("models/map parts/admin/admin-7.png")

        admin7 = pygame.image.load("models/map parts/admin/admin-8.png")

        admin8 = pygame.image.load("models/map parts/admin/admin-9.png")

        admin9 = pygame.image.load("models/map parts/admin/admin-10.png")

        admin10 = pygame.image.load("models/map parts/admin/admin-10.png")

        adminelec = electric

        adminupl = upload

        adminox = o23

        #storage

        sto1 = pygame.image.load("models/map parts/storage/storage-1.png")

        sto2 = pygame.image.load("models/map parts/storage/storage-2.png")

        sto3 = pygame.image.load("models/map parts/storage/storage-3.png")

        sto4 = pygame.image.load("models/map parts/storage/storage-4.png")

        sto5 = pygame.image.load("models/map parts/storage/storage-5.png")

        sto6 = electric

        sto7 = pygame.image.load("models/map parts/storage/storage-7.png")

        #commnication

        comm1 = pygame.image.load("models/map parts/communication/comm-1.png")

        comm2 = pygame.image.load("models/map parts/communication/comm-2.png")

        comm3 = pygame.image.load("models/map parts/communication/comm-3.png")

        comm4 = pygame.image.load("models/map parts/communication/comm-4.png")

        comm5 = pygame.image.load("models/map parts/communication/comm-5.png")

        comm6 = pygame.image.load("models/map parts/communication/comm-6.png")

        comm7 = pygame.image.load("models/map parts/communication/comm-7.png")

        comm8 = pygame.image.load("models/map parts/communication/comm-8.png")

        comm9 = pygame.image.load("models/map parts/communication/comm-9.png")

        comm10 = electric

        comm11 = upload

        #electric

        ele1 = pygame.image.load("models/map parts/electric/ele-1.png")

        ele2 = pygame.image.load("models/map parts/electric/ele-2.png")

        ele3 = pygame.image.load("models/map parts/electric/ele-3.png")

        ele4 = pygame.image.load("models/map parts/electric/ele-4.png")

        ele5 = pygame.image.load("models/map parts/electric/ele-5.png")

        ele6 = pygame.image.load("models/map parts/electric/ele-6.png")

        ele7 = electric

        ele8 = redirect

        ele9 = upload

        #lowerengine

        low1 = pygame.image.load("models/map parts/engine/eng-1.png")

        low2 = pygame.image.load("models/map parts/engine/eng-2.png")

        low3 = pygame.image.load("models/map parts/engine/eng-3.png")

        low4 = pygame.image.load("models/map parts/engine/eng-4.png")

        low5 = pygame.image.load("models/map parts/engine/eng-5.png")

        low6 = pygame.image.load("models/map parts/engine/eng-6.png")

        low7 = pygame.image.load("models/map parts/engine/eng-7.png")

        low8 = pygame.image.load("models/map parts/engine/eng-8.png")

        low9 = pygame.image.load("models/map parts/engine/eng-10.png")

        low10 = pygame.image.load("models/map parts/engine/eng-11.png")

        low11 = pygame.image.load("models/map parts/engine/eng-12.png")

        low12 = pygame.image.load("models/map parts/engine/eng-13.png")

        low13 = pygame.image.load("models/map parts/engine/eng-9.png")

        lowred = redirect

        #security

        sec1 = pygame.image.load("models/map parts/security/sec-1.png")

        sec2 = pygame.image.load("models/map parts/security/sec-2.png")

        sec3 = pygame.image.load("models/map parts/security/sec-3.png")

        sec4 = pygame.image.load("models/map parts/security/sec-4.png")

        sec5 = pygame.image.load("models/map parts/security/sec-5.png")

        sec6 = pygame.image.load("models/map parts/security/sec-6.png")

        secele = electric

        #medbay

        med1 = pygame.image.load("models/map parts/medbay/med-1.png")

        med2 = pygame.image.load("models/map parts/medbay/med-2.png")

        med3 = pygame.image.load("models/map parts/medbay/med-3.png")

        med4 = pygame.image.load("models/map parts/medbay/med-4.png")

        #shields

        she1 = pygame.image.load("models/map parts/shields/she-1.png")

        she2 = pygame.image.load("models/map parts/shields/she-2.png")

        she3 = pygame.image.load("models/map parts/shields/she-3.png")

        she4 = pygame.image.load("models/map parts/shields/she-4.png")

        she5 = pygame.image.load("models/map parts/shields/she-5.png")

        she6 = pygame.image.load("models/map parts/shields/she-6.png")

        she7 = pygame.image.load("models/map parts/shields/she-7.png")

        she8 = pygame.image.load("models/map parts/shields/she-8.png")

        she9 = pygame.image.load("models/map parts/shields/she-9.png")

        she10 = redirect

        #reactor

        rec1 = pygame.image.load("models/map parts/reactor/rec-1.png")

        rec2 = pygame.image.load("models/map parts/reactor/rec-2.png")

        rec3 = pygame.image.load("models/map parts/reactor/rec-3.png")

        class Sprite(pygame.sprite.Sprite):

            def \_\_init\_\_(self, sizex = 10, sizey = 10, surface = 'default'):

                pygame.sprite.Sprite.\_\_init\_\_(self)

                if surface == 'default':

                    self.image = pygame.Surface((sizex, sizey))

                    self.image.fill((255, 0, 0))

                else:

                    self.image = surface

                self.rect = self.image.get\_rect()

        q = open('collision\_points.dat', 'rb')

        coll\_loc = pickle.load(q)

        collision = [Sprite(k, l) for i,j,k,l in coll\_loc]

        player = Player()

        players = pygame.sprite.Group()

        players.add(player)

        wall\_group = pygame.sprite.Group()

        for i in range(len(collision)):

            wall\_group.add(collision[i])

        topos = []

        sps = []

        #buttons

        tasks = Sprite(surface = pygame.image.load("models/buttons/tasks.png"))

        taskson = pygame.image.load("models/buttons/tasks.png")

        tasksoff = pygame.image.load("models/buttons/tasks\_off.png")

        report = Sprite(surface = pygame.image.load("models/buttons/report.png"))

        reporton = pygame.image.load("models/buttons/report.png")

        reportoff = pygame.image.load("models/buttons/report\_off.png")

        sabotage = Sprite(surface = pygame.image.load("models/buttons/sabotage.png"))

        vent = Sprite(surface = pygame.image.load("models/buttons/vent.png"))

        kill = Sprite(surface = pygame.image.load("models/buttons/kill.png"))

        killon = pygame.image.load("models/buttons/kill.png")

        killoff = pygame.image.load("models/buttons/kill\_off.png")

        security = Sprite(surface = pygame.image.load("models/buttons/security.png"))

        mousebut = Sprite()

        mousebut.rect.x, mousebut.rect.y = 0, 0

        tasks.rect.x, tasks.rect.y = 897, 446

        report.rect.x, report.rect.y = 892, 333

        sabotage.rect.x, sabotage.rect.y = 897, 446

        vent.rect.x, vent.rect.y = 897, 446

        kill.rect.x, kill.rect.y = 789, 444

        security.rect.x, security.rect.y = 789, 444

        button\_group = pygame.sprite.Group()

        button\_group.add(tasks)

        button\_group.add(report)

        button\_group.add(sabotage)

        button\_group.add(vent)

        button\_group.add(kill)

        button\_group.add(security)

        but = [tasks, report, sabotage, vent, kill]

        mous\_grp = pygame.sprite.Group()

        mous\_grp.add(mousebut)

        imposter = False

        if imposter:

            tasks.rect.x, tasks.rect.y = (0, -200)

        else:

            sabotage.rect.x, sabotage.rect.y = 0, -200

            kill.rect.x, kill.rect.y = 0, -200

            vent.rect.x, vent.rect.y = 0, -200

        tskpos = [(1290, 405, 10, 10), (1290, 233, 10, 10), (1478, 397, 10, 10),

                    (920, 256, 10, 10), (841, 180, 10, 10), (117, 121, 10, 10),

                    (1260, 787, 10, 10), (1107, 817, 10, 10), (1035, 827, 10, 10),

                    (964, 860, 10, 10), (1761, 919, 10, 10), (1878, 791, 10, 10),

                    (1967, 776, 10, 10), (2028, 811, 10, 10), (2073, 1021, 10, 10),

                    (1409, 1422, 10, 10), (1176, 1769, 10, 10), (1056, 1760, 10, 10),

                    (922, 1972, 10, 10), (887, 1771, 10, 10), (612, 1981, 10, 10),

                    (254, 1724, 10, 10), (-256, 1432, 10, 10), (-266, 1201, 10, 10),

                    (-206, 1201, 10, 10), (-111, 1201, 10, 10), (54, 1201, 10, 10),

                    (-67, 997, 10, 10), (-13, 936, 10, 10), (-518, 815, 10, 10),

                    (-475, 833, 10, 10), (-719, 956, 10, 10), (-990, 1626, 10, 10),

                    (-1007, 1617, 10, 10), (-978, 1330, 10, 10), (-1001, 588, 10, 10),

                    (-917, 284, 10, 10), (639, 1091, 10, 10), (738, 1072, 10, 10),

                    (808, 1265, 10, 10), (1012, 1265, 10, 10), (1078, 1088, 10, 10),

                    (362, 1294, 10, 10), (-1316, 1020, 10, 10), (-1156, 799, 10, 10),

                    (-1366, 710, 10, 10), (-1273, 648, 10, 10), (-1278, 1344, 10, 10),

                    (340, 402, 220, 150), (-1071, 610, 10, 10)]

        ToDo = {5:1, 4:3, 3:2, 0:9, 1:3, 2:18, 6:18, 8:5, 9:2, 10:1, 11:18, 12:3, 13:8,

                14:15, 15:18, 16:14, 17:18, 19:3, 21:12, 20:2, 42:1, 23:1, 24:10, 25:1,

                26:7, 32:11, 33:6, 34:18, 43:17, 45:16, 31:1, 30:18, 35:11, 36:18, 27:19,

                28:13, 37:1, 38:4, 40:0, 49:6}

        DoTo = []

        taskmgr = [Sprite(k+40, l+40) for i,j,k,l in tskpos]

        task\_group = pygame.sprite.Group()

        for i in range(len(taskmgr)):

            task\_group.add(taskmgr[i])

        dead\_bodys = [(100, 100, 100, 100)]

        f = []

        Tasks = Tasks()

        #secCam

        secCam = 0

        secC1 = pygame.image.load("models/tasks/Security Camera/sec-2.png")

        secC2 = pygame.image.load("models/tasks/Security Camera/sec-1.png")

        secC3 = pygame.image.load("models/tasks/Security Camera/sec-3.png")

        secCNum = 1

        while True:

            c += 1

            wall\_group.draw(screen)

            mousebut.rect.x, mousebut.rect.y = pygame.mouse.get\_pos()

            mous\_grp.draw(screen)

            task\_group.draw(screen)

            screen.fill((0, 0, 0))

            before\_pos = a, b

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    return 1

                    exit()

                if event.type == pygame.MOUSEBUTTONDOWN:

                    for i in range(len(but)):

                        smashhit = pygame.sprite.collide\_rect(mousebut, but[i])

                        if smashhit and pygame.mouse.get\_pressed()[0]:

                            if tasksToDo in ToDo and not imposter:

                                pygame.mixer.Channel(4).play(pygame.mixer.Sound(f'bgs/Among Us General Sounds/task\_Inprogress.wav'))

                                if ToDo[tasksToDo] == 0:

                                    Tasks.swipeCard()

                                elif ToDo[tasksToDo] == 1:

                                    Tasks.fixWiring()

                                elif ToDo[tasksToDo] == 2:

                                    Tasks.emptyGarbage()

                                elif ToDo[tasksToDo] == 3:

                                    Tasks.upload()

                                elif ToDo[tasksToDo] == 4:

                                    Tasks.Download(1)

                                elif ToDo[tasksToDo] == 5:

                                    Tasks.clearLeaves()

                                elif ToDo[tasksToDo] == 6:

                                    Tasks.alignEngine()

                                elif ToDo[tasksToDo] == 7:

                                    Tasks.calibrate()

                                elif ToDo[tasksToDo] == 8:

                                    Tasks.chartCourse()

                                elif ToDo[tasksToDo] == 9:

                                    Tasks.weapons()

                                elif ToDo[tasksToDo] == 10:

                                    Tasks.divertPower(1)

                                elif ToDo[tasksToDo] == 11:

                                    Tasks.fualEngine()

                                elif ToDo[tasksToDo] == 12:

                                    Tasks.fillCan()

                                elif ToDo[tasksToDo] == 13:

                                    Tasks.inspectSample()

                                elif ToDo[tasksToDo] == 14:

                                    Tasks.primeShield()

                                elif ToDo[tasksToDo] == 15:

                                    Tasks.stabSteering()

                                elif ToDo[tasksToDo] == 16:

                                    Tasks.unlockManifolds()

                                elif ToDo[tasksToDo] == 17:

                                    Tasks.starReactor()

                                elif ToDo[tasksToDo] == 18:

                                    Tasks.acceptPower()

                                elif ToDo[tasksToDo] == 19:

                                    Tasks.medbayScan()

                                pygame.mixer.Channel(4).play(pygame.mixer.Sound(f'bgs/Among Us General Sounds/task\_Complete.wav'))

            keys = pygame.key.get\_pressed()

            if keys[K\_w]:

                b += 3

            if keys[K\_a]:

                a += 5

            if keys[K\_s]:

                b -= 3

            if keys[K\_d]:

                a -= 5

            #weapons

            screen.blit(bg, (0+a,0+b))

            screen.blit(weapons1, (1100+a, 260+b))

            screen.blit(weapons4, (1137+a, 204+b))

            screen.blit(weapons5, (1136+a, 286+b))

            screen.blit(weapons6, (1383+a, 463+b))

            screen.blit(weapons2, (1122+a, 256+b))

            screen.blit(caf\_weap, (979+a, 362+b))

            screen.blit(weapons3, (980+a, 193+b))

            screen.blit(weapons7, (1132+a, 242+b))

            #o2

            screen.blit(wep\_o2\_nav\_she, (1209+a, 652+b))

            screen.blit(o21, (875+a, 707+b))

            screen.blit(o22, (1006+a, 802+b))

            screen.blit(o24, (941+a, 830+b))

            screen.blit(o25, (1067+a, 722+b))

            screen.blit(o23, (1090+a, 806+b))

            if c%2 == 0:

                screen.blit(o26, (891+a, 777+b))

            else:

                screen.blit(o27, (891+a, 777+b))

            screen.blit(oredirect, (1229+a, 771+b))

            #navigation

            screen.blit(nav2, (1809+a, 798+b))

            screen.blit(nav1, (1808+a, 742+b))

            screen.blit(nav3, (1808+a, 752+b))

            screen.blit(nav4, (2008+a, 835+b))

            screen.blit(nav5, (2017+a, 1040+b))

            screen.blit(nav6, (2023+a, 930+b))

            screen.blit(nav7, (2066+a, 905+b))

            screen.blit(nav8, (2021+a, 812+b))

            screen.blit(nav9, (1946+a, 770+b))

            screen.blit(navredirect, (1866+a, 771+b))

            screen.blit(navelectric, (1747+a, 898+b))

            #admin

            screen.blit(caf\_ad\_st, (414+a, 958+b))

            screen.blit(admin1, (561+a, 1115+b))

            screen.blit(admin2, (578+a, 1056+b))

            screen.blit(admin3, (582+a, 1064+b))

            screen.blit(admin4, (824+a, 1248+b))

            screen.blit(admin5, (824+a, 1259+b))

            screen.blit(admin6, (999+a, 1259+b))

            screen.blit(admin7, (870+a, 1259+b))

            screen.blit(admin9, (831+a, 1081+b))

            screen.blit(admin10, (998+a, 1081+b))

            screen.blit(admin8, (838+a, 1099+b))

            screen.blit(adminelec, (645+a, 1094+b))

            screen.blit(adminupl, (749+a, 1084+b))

            screen.blit(adminox, (1078+a, 1095+b))

            #cafeteria

            screen.blit(caflev, (913+a, 226+b))

            screen.blit(cafup, (832+a, 159+b))

            screen.blit(cafred, (93+a, 104+b))

            #storage

            screen.blit(sto1, (80+a, 1249+b))

            screen.blit(sto2, (194+a, 1471+b))

            screen.blit(sto4, (279+a, 1700+b))

            screen.blit(sto5, (603+a, 1959+b))

            screen.blit(sto6, (365+a, 1298+b))

            screen.blit(sto7, (637+a, 1485+b))

            #communication

            screen.blit(comm1, (663+a, 1739+b))

            screen.blit(comm2, (662+a, 1696+b))

            screen.blit(comm3, (676+a, 1729+b))

            if c%2 == 0:

                screen.blit(comm8, (696+a, 1756+b))

            else:

                screen.blit(comm9, (696+a, 1756+b))

            screen.blit(comm10, (1046+a, 1752+b))

            screen.blit(comm11, (856+a, 1736+b))

            #electric

            screen.blit(ele3, (-694+a, 1147+b))

            screen.blit(ele6, (-302+a, 1293+b))

            screen.blit(ele1, (-304+a, 1353+b))

            screen.blit(ele4, ( 35+a, 1187+b))

            screen.blit(ele5, (-294+a, 1395+b))

            screen.blit(ele7, (-118+a, 1187+b))

            screen.blit(ele8, (-227+a, 1183+b))

            screen.blit(ele9, (-280+a, 1180+b))

            screen.blit(ele2, (-314+a, 1159+b))

            #security

            screen.blit(sec1, (-1055+a, 723+b))

            screen.blit(sec2, (-700+a, 729+b))

            screen.blit(sec3, (-620+a, 753+b))

            screen.blit(sec4, (-574+a, 809+b))

            screen.blit(sec5, (-475+a, 793+b))

            screen.blit(sec6, (-664+a, 780+b))

            screen.blit(secele, (-737+a, 945+b))

            #lowerengine

            screen.blit(low1, (-1097+a, 1269+b))

            screen.blit(low3, (-832+a, 1432+b))

            if c%2 == 0:

                screen.blit(low4, (-1088+a, 1394+b))

                screen.blit(low5, (-1078+a, 1607+b))

            else:

                screen.blit(low4, (-1087+a, 1389+b))

                screen.blit(low5, (-1077+a, 1606+b))

            screen.blit(low2, (-987+a, 1583+b))

            screen.blit(low6, (-993+a, 1592+b))

            if c%7 == 0:

                screen.blit(low9, (-825+a, 1548+b))

                screen.blit(low11, (-895+a, 1411+b))

            elif c%8 == 0:

                screen.blit(low10, (-863+a, 1449+b))

                screen.blit(low12, (-895+a, 1594+b))

            screen.blit(lowred, (-975+a, 1341+b))

            #upper engine

            screen.blit(low13, (-1099+a, 256+b))

            screen.blit(low3, (-838+a, 407+b))

            if c%2 == 0:

                screen.blit(low4, (-1089+a, 348+b))

            else:

                screen.blit(low4, (-1088+a, 349+b))

            screen.blit(low2, (-993+a, 556+b))

            screen.blit(low5, (-1081+a, 563+b))

            screen.blit(low6, (-997+a, 569+b))

            if c%7 == 0:

                screen.blit(low9, (-846+a, 383+b))

                screen.blit(low12, (-884+a, 539+b))

            elif c%8 == 0:

                screen.blit(low10, (-918+a, 535+b))

                screen.blit(low11, (-805+a, 433+b))

            screen.blit(ele8, (-906+a, 291+b))

            #medbay

            screen.blit(med1, (-706+a, 362+b))

            screen.blit(med2, (-401+a, 563+b))

            screen.blit(med3, (-142+a, 958+b))

            screen.blit(med4, (-52+a, 871+b))

            #shields

            screen.blit(she1, (1106+a, 1436+b))

            screen.blit(she2, (1103+a, 1429+b))

            screen.blit(she3, (1093+a, 1362+b))

            lig = [(1149, 1454), (1164, 1436), (1177, 1417), (1196, 1404), (1494, 1542),

                    (1494, 1513), (1495, 1483)]

            for i in lig[::-1]:

                screen.blit(she9, (i[0]+a, i[1]+b))

            screen.blit(she5, (1397+a, 1598+b))

            screen.blit(she6, (1131+a, 1436+b))

            screen.blit(she7, (1153+a, 1713+b))

            screen.blit(she10,(1425+a, 1411+b))

            #reactor

            screen.blit(rec2, (-1505+a, 636+b))

            screen.blit(rec3, (-1483+a, 1187+b))

            #cams

            if secCam == 0:

                screen.blit(cam\_off, (-15+a, 385+b))

                screen.blit(cam\_off, (-790+a, 927+b))

                screen.blit(cam\_off, (577+a, 1076+b))

                screen.blit(cam\_off, (1652+a, 878+b))

            else:

                screen.blit(cam\_on, (-15+a, 385+b))

                screen.blit(cam\_on, (-790+a, 927+b))

                screen.blit(cam\_on, (577+a, 1076+b))

                screen.blit(cam\_on, (1652+a, 878+b))

            #collision

            for i in range(len(collision)):

                collision[i].rect.x, collision[i].rect.y = coll\_loc[i][0]+a, coll\_loc[i][1]+b

            coll1 = pygame.surface.Surface([150, 100])

            h = coll1.get\_rect()

            hit = pygame.sprite.spritecollide(player, wall\_group, False)

            did = 0

            prev = a, b

            #tasks

            screen.blit(weaponselectric, (1491+a, 376+b))

            screen.blit(weaponsupload, (1276+a, 216+b))

            screen.blit(weapons10, (1274+a , 360+b))

            s = pygame.surface.Surface([10, 10])

            screen.blit(s, pygame.mouse.get\_pos())

            security.rect.x, security.rect.y = 0, -200

            for i in range(len(taskmgr)):

                taskmgr[i].rect.x, taskmgr[i].rect.y = tskpos[i][0]+a, tskpos[i][1]+b

            todo = 0

            for i in range(len(taskmgr)):

                if pygame.sprite.collide\_rect(player, taskmgr[i]) == 1:

                    todo = 1

                    tasksToDo = i

                    tasks.image = taskson

                    if i == 29:

                        security.rect.x, security.rect.y = 789, 444

                        if pygame.mouse.get\_pressed()[0]:

                            cc = a, b

                            secCam = 1

                    else:

                        security.rect.x, security.rect.y = 0, -200

                else:

                    tasks.image = tasksoff

            if todo == 1:

                tasks.image = taskson

            else:

                tasksToDo = None

            #dead

            dead = []

            dead\_grp = pygame.sprite.Group()

            reportbut = 0

            for i in range(len(dead)):

                ded = dead[i]

                dead[i] = Sprite(100, 100)

                dead[i].rect.x, dead[i].rect.y = ded[0]+a, ded[1]+b

                dead\_grp.add(dead[i])

            dead\_grp.draw(screen)

            for i in dead:

                if pygame.sprite.collide\_rect(player, i) == 1:

                    reportbut = 1

            if reportbut == 1:

                report.image = reporton

            else:

                report.image = reportoff

            for i in collision:

                if pygame.sprite.collide\_rect(player, i):

                    if keys[pygame.K\_w]:

                        if abs(player.rect.top - i.rect.bottom) < 10 and hit:

                            b = before\_pos[1]

                    if keys[pygame.K\_a]:

                        if abs(player.rect.left - i.rect.right) < 10 and hit:

                            a = before\_pos[0]

                    if keys[pygame.K\_s]:

                        if abs(player.rect.bottom - i.rect.top) < 10 and hit:

                            b = before\_pos[1]

                    if keys[pygame.K\_d]:

                        if abs(player.rect.right - i.rect.left) < 10 and hit:

                            a = before\_pos[0]

            players.draw(screen)

            coll = a, b

            #on the player

            a, b = prev

            screen.blit(weapons8, (1133+a, 509+b))

            screen.blit(weaponsgreenscreen, (1304+a, 278+b))

            screen.blit(weapons9, (1394+a, 409+b))

            screen.blit(she8, (1397+a, 1448+b))

            screen.blit(she4, (1129+a, 1635+b))

            screen.blit(comm5, (772+a, 1935+b))

            screen.blit(comm4, (792+a, 1970+b))

            screen.blit(comm6, (671+a, 1849+b))

            screen.blit(comm7, (1041+a, 1846+b))

            screen.blit(sto3, (439+a, 1447+b))

            screen.blit(low3, (-832+a, 1432+b))

            if c%2 == 0:

                screen.blit(low4, (-1088+a, 1394+b))

                screen.blit(low5, (-1078+a, 1607+b))

            else:

                screen.blit(low4, (-1087+a, 1389+b))

                screen.blit(low5, (-1077+a, 1606+b))

            if c%7 == 0:

                screen.blit(low9, (-825+a, 1548+b))

                screen.blit(low11, (-895+a, 1411+b))

            elif c%8 == 0:

                screen.blit(low10, (-863+a, 1449+b))

                screen.blit(low12, (-895+a, 1594+b))

            if c%2 == 0:

                screen.blit(low4, (-1089+a, 348+b))

            else:

                screen.blit(low4, (-1088+a, 349+b))

            screen.blit(rec1, (-1466+a, 824+b))

            screen.blit(low5, (-1074+a, 563+b))

            a, b = coll

            #buttons

            button\_group.draw(screen)

            if secCam == 1:

                screen.blit(secC1, (-200, 0))

                screen.blit(secC2, (809, 245))

                screen.blit(secC3, (82, 230))

                if 809 < pygame.mouse.get\_pos()[0] < 809+60 and 245 < pygame.mouse.get\_pos()[1] < 245+60 and pygame.mouse.get\_pressed()[0]:

                    secCNum += 0.5

                    if secCNum > 5:

                        secCNum = 1

                if 82 < pygame.mouse.get\_pos()[0] < 82+60 and 230 < pygame.mouse.get\_pos()[1] < 230+60 and pygame.mouse.get\_pressed()[0]:

                    secCNum -= 0.5

                    if secCNum < 1:

                        secCNum = 4.5

                if int(secCNum) == 1:

                    a, b = (1365, -720)

                elif int(secCNum) == 2:

                    a, b = (785, -99)

                elif int(secCNum) == 3:

                    a, b = 5, -825

                else:

                    a, b = -1025, -660

                close = pygame.image.load("models/buttons/close.png")

                screen.blit(close, (100, 25))

                if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                    secCam = 0

                    a, b = cc

                    cc = None

            #wall\_group.draw(screen)

            close = pygame.image.load("models/buttons/close.png")

            screen.blit(close, (0, 0))

            if 0 < pygame.mouse.get\_pos()[0] < 50 and 0 < pygame.mouse.get\_pos()[1] < 50 and pygame.mouse.get\_pressed()[0]:

                return 1

            players.update(secCam)

            pygame.display.update()

            clock.tick(fps)

ONLINE\_MUILTIPLAYER.PY

import pygame

from pygame.locals import \*

import random

class online():

    def \_\_init\_\_(self):

        pass

    def run(self):

        from walk\_anim import Player

        import pickle

        from Tasks import Tasks

        from threading import Thread

        from randomizer import getAllTasks

        from networking.client import client

        import time

        connect = client()

        life = eval(connect.send((0, 0, 1, 0, (0, 0, 0), False, False)))

        ownpos = eval(life[str(f"b'{connect.name}'")])[-2]

        imposter = eval(life[str(f"b'{connect.name}'")])[6]

        My\_color = connect.color

        pygame.init()

        clock = pygame.time.Clock()

        fps = 60

        size =[1000, 550]

        screen = pygame.display.set\_mode(size)

        screen.set\_colorkey('#000000')

        font\_size = 18

        font = pygame.font.Font('freesansbold.ttf', font\_size)

        in\_lobby = True

        a = 0

        b = 0

        c = 0

        def colorchanger(surface, color):

            """Fill all pixels of the surface with color, preserve transparency."""

            surface = surface.convert\_alpha()

            w, h = surface.get\_size()

            r, g, b = color

            for x in range(w):

                for y in range(h):

                    if surface.get\_at((x,y)) == (255, 0, 0, 255):

                        surface.set\_at((x, y), pygame.Color(r, g, b, 255))

            return surface

        class Sprite(pygame.sprite.Sprite):

            def \_\_init\_\_(self, sizex = 10, sizey = 10, surface = 'default'):

                pygame.sprite.Sprite.\_\_init\_\_(self)

                if surface == 'default':

                    self.image = pygame.Surface((sizex, sizey))

                    self.image.fill((255, 0, 0))

                else:

                    self.image = surface

                self.rect = self.image.get\_rect()

        coll\_loc = [(202, 294, 10, 341), (204, 289, 10, 10), (223, 281, 10, 10),

                    (241, 272, 10, 10), (255, 262, 10, 10), (276, 256, 10, 10),

                    (290, 249, 10, 10), (306, 245, 10, 10), (322, 240, 10, 10),

                    (332, 235, 10, 10), (350, 228, 10, 10), (368, 223, 10, 10),

                    (376, 223, 214, 10), (589, 224, 10, 10), (606, 230, 10, 10),

                    (629, 236, 10, 10), (643, 245, 10, 10), (661, 254, 10, 10),

                    (676, 259, 10, 10), (692, 264, 10, 10), (708, 271, 10, 10),

                    (726, 278, 10, 10), (740, 287, 10, 10), (752, 294, 10, 10),

                    (765, 299, 10, 10), (765, 303, 10, 290), (213, 613, 10, 10),

                    (223, 628, 10, 10), (231, 642, 10, 10), (243, 652, 477, 10),

                    (728, 645, 10, 10), (742, 631, 10, 10), (753, 616, 10, 10),

                    (765, 601, 10, 10), (315, 310, 100, 60)]

        collision = [Sprite(k, l) for i,j,k,l in coll\_loc]

        player = Player()

        players = pygame.sprite.Group()

        players.add(player)

        wall\_group = pygame.sprite.Group()

        for i in range(len(collision)):

            wall\_group.add(collision[i])

        lob1 = pygame.image.load("models/map parts/lobby/1.png")

        lob2 = pygame.image.load("models/map parts/lobby/2.png")

        lob3 = pygame.image.load("models/map parts/lobby/3.png")

        lob4 = pygame.image.load("models/map parts/lobby/4.png")

        while in\_lobby:

            wall\_group.draw(screen)

            screen.fill(0)

            before\_pos = a, b

            server\_info = connect.send((-a, -b, player.move, player.flip, My\_color))

            for i in range(len(collision)):

                collision[i].rect.x,collision[i].rect.y=coll\_loc[i][0]+a,coll\_loc[i][1]+b

            keys = pygame.key.get\_pressed()

            if keys[K\_w]:

                b += 3

            if keys[K\_a]:

                a += 5

            if keys[K\_s]:

                b -= 3

            if keys[K\_d]:

                a -= 5

            screen.blit(lob1, (-125+a, 0+b))

            screen.blit(lob2, (153+a, 617+b))

            screen.blit(lob3, (311+a, 279+b))

            screen.blit(lob4, (880+a+random.random()\*10, 728+b+random.random()))

            screen.blit(pygame.transform.rotate(lob4, 25), (-104+a+random.random()\*10, 702+b+random.random()))

            hit = pygame.sprite.spritecollide(player, wall\_group, False)

            for i in collision:

                if pygame.sprite.collide\_rect(player, i):

                    if keys[pygame.K\_w]:

                        if abs(player.rect.top - i.rect.bottom) < 17 and hit:

                            b = before\_pos[1]

                    if keys[pygame.K\_a]:

                        if abs(player.rect.left - i.rect.right) < 17 and hit:

                            a = before\_pos[0]

                    if keys[pygame.K\_s]:

                        if abs(player.rect.bottom - i.rect.top) < 17 and hit:

                            b = before\_pos[1]

                    if keys[pygame.K\_d]:

                        if abs(player.rect.right - i.rect.left) < 17 and hit:

                            a = before\_pos[0]

            try:

                server\_info = eval(server\_info)

                for i in server\_info:

                    server\_info[i] = eval(server\_info[i])

                for i in server\_info:

                    if i != str(f"b'{connect.name}'"):

                        font1 = pygame.font.Font('freesansbold.ttf', 10)

                        Tet = i[2:-1]

                        tet = font.render(Tet, True, (255, 255, 255))

                        tetRect = tet.get\_rect()

                        screen.blit(tet, (int(server\_info[i][0])+490+a, int(server\_info[i][1])+265+b))

                        player2 = pygame.transform.flip(pygame.image.load(f"images/Sprites/Walk/walkcolor00{int(server\_info[i][2])}.png"), not server\_info[i][3], False)

                        if int(server\_info[i][2]) == 1:

                            player2 = pygame.image.load('idle.png')

                        player2 = pygame.transform.scale(player2, (78-25,103-30))

                        player2 = colorchanger(player2, server\_info[i][4])

                        screen.blit(player2, (int(server\_info[i][0])+500+a, int(server\_info[i][1])+275+b))

            except Exception as e:

                pass

            players.draw(screen)

            if len(server\_info) == 4:

                in\_lobby = False

                start = True

                sin = 0

                pygame.mixer.Channel(5).play(pygame.mixer.Sound(f'bgs/Among Us General Sounds/Roundstart\_MAIN.wav'))

                while start:

                    screen.fill(0)

                    sin += 1

                    if sin > 200:

                        start = False

                    screen.blit(pygame.image.load("models/shhhhhh.png"), (253, 26))

                    pygame.display.update()

                    clock.tick(fps)

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    connect.send("disconnect")

                    return 3

            players.update(color = My\_color)

            pygame.display.update()

            clock.tick(fps)

        AllTasks = getAllTasks()

        Text1 = 'fixWiring '

        Text2 = 'divert and accept power '

        Text3 = 'Download and Upload '

        Text4 = f'{AllTasks[3][0]} '

        if str(AllTasks[4][0])[0].isalpha():

            Text5 = f'{AllTasks[4][0]} '

        else:

            Text5 = 'Usual Tasks '

        Text6 = 'Main Tasks '

        for i in range(len(AllTasks)):

            for j in range(len(AllTasks[i])):

                if str(AllTasks[i][j])[0].isalpha():

                    del AllTasks[i][j]

                    break

        for i in range(len(AllTasks)):

            if AllTasks[i][0] == 4:

                del AllTasks[i][0]

        tasksToDo = None

        a = 0

        b = 0

        c = 0

        cam\_on = pygame.image.load("models/map parts/cam-on.png")

        cam\_off = pygame.image.load("models/map parts/cam-off.png")

        redirect = pygame.image.load("models/map parts/redirect.png")

        electric = pygame.image.load("models/map parts/electric.png")

        upload = pygame.image.load("models/map parts/weapons/upload.png")

        vent\_pic = pygame.image.load("models/map parts/vent.png")

        #cafeteria

        bg = pygame.image.load('models/map parts/PC Computer - Among Us - Skeld Cafeteria.png')

        caflev = pygame.transform.flip(pygame.image.load("models/map parts/oxygen/o2-4.png"), True, False)

        cafup = upload

        cafred = pygame.transform.flip(pygame.image.load("models/map parts/weapons/redirect.png"), True, False)

        #weapons

        caf\_weap = pygame.image.load("models/map parts/weapons/caf-weapons.png")

        weapons1 = pygame.image.load("models/map parts/weapons/weapons-1.png")

        weapons2 = pygame.image.load("models/map parts/weapons/weapons-2.png")

        weapons3 = pygame.image.load("models/map parts/weapons/weapons-3.png")

        weapons4 = pygame.image.load("models/map parts/weapons/weapons-4.png")

        weapons5 = pygame.image.load("models/map parts/weapons/weapons-5.png")

        weapons6 = pygame.image.load("models/map parts/weapons/weapons-6.png")

        weapons7 = pygame.image.load("models/map parts/weapons/weapons-7.png")

        weapons8 = pygame.image.load("models/map parts/weapons/weapons-8.png")

        weapons9 = pygame.image.load("models/map parts/weapons/weapons-9.png")

        weapons10 = pygame.image.load("models/map parts/weapons/weapons-10.png")

        weaponselectric = pygame.image.load("models/map parts/weapons/redirect.png")

        weaponsupload = upload

        weaponsgreenscreen = pygame.image.load("models/map parts/weapons/greenscreen.png")

        #oxygen

        wep\_o2\_nav\_she = pygame.image.load("models/map parts/oxygen/wep-ox-nav-she.png")

        o21 = pygame.image.load("models/map parts/oxygen/o2-1.png")

        o22 = pygame.image.load("models/map parts/oxygen/o2-2.png")

        o23 = pygame.image.load("models/map parts/oxygen/o2-3.png")

        o24 = pygame.image.load("models/map parts/oxygen/o2-4.png")

        o25 = pygame.image.load("models/map parts/oxygen/o2-5.png")

        o26 = pygame.image.load("models/map parts/oxygen/o2-6.png")

        o27 = pygame.image.load("models/map parts/oxygen/o2-7.png")

        oredirect = redirect

        #navigation

        nav1 = pygame.image.load("models/map parts/navigation/nav-1.png")

        nav2 = pygame.image.load("models/map parts/navigation/nav-2.png")

        nav3 = pygame.image.load("models/map parts/navigation/nav-3.png")

        nav4 = pygame.image.load("models/map parts/navigation/nav-4.png")

        nav5 = pygame.image.load("models/map parts/navigation/nav-5.png")

        nav6 = pygame.image.load("models/map parts/navigation/nav-6.png")

        nav7 = pygame.image.load("models/map parts/navigation/nav-7.png")

        nav8 = pygame.image.load("models/map parts/navigation/nav-8.png")

        nav9 = pygame.image.load("models/map parts/navigation/nav-9.png")

        navredirect = redirect

        navelectric = electric

        #admin

        caf\_ad\_st = pygame.image.load("models/map parts/admin/admin-4.png")

        admin1 = pygame.image.load("models/map parts/admin/admin-1.png")

        admin2 = pygame.image.load("models/map parts/admin/admin-2.png")

        admin3 = pygame.image.load("models/map parts/admin/admin-3.png")

        admin4 = pygame.image.load("models/map parts/admin/admin-5.png")

        admin5 = pygame.image.load("models/map parts/admin/admin-6.png")

        admin6 = pygame.image.load("models/map parts/admin/admin-7.png")

        admin7 = pygame.image.load("models/map parts/admin/admin-8.png")

        admin8 = pygame.image.load("models/map parts/admin/admin-9.png")

        admin9 = pygame.image.load("models/map parts/admin/admin-10.png")

        admin10 = pygame.image.load("models/map parts/admin/admin-10.png")

        adminelec = electric

        adminupl = upload

        adminox = o23

        #storage

        sto1 = pygame.image.load("models/map parts/storage/storage-1.png")

        sto2 = pygame.image.load("models/map parts/storage/storage-2.png")

        sto3 = pygame.image.load("models/map parts/storage/storage-3.png")

        sto4 = pygame.image.load("models/map parts/storage/storage-4.png")

        sto5 = pygame.image.load("models/map parts/storage/storage-5.png")

        sto6 = electric

        sto7 = pygame.image.load("models/map parts/storage/storage-7.png")

        #commnication

        comm1 = pygame.image.load("models/map parts/communication/comm-1.png")

        comm2 = pygame.image.load("models/map parts/communication/comm-2.png")

        comm3 = pygame.image.load("models/map parts/communication/comm-3.png")

        comm4 = pygame.image.load("models/map parts/communication/comm-4.png")

        comm5 = pygame.image.load("models/map parts/communication/comm-5.png")

        comm6 = pygame.image.load("models/map parts/communication/comm-6.png")

        comm7 = pygame.image.load("models/map parts/communication/comm-7.png")

        comm8 = pygame.image.load("models/map parts/communication/comm-8.png")

        comm9 = pygame.image.load("models/map parts/communication/comm-9.png")

        comm10 = electric

        comm11 = upload

        #electric

        ele1 = pygame.image.load("models/map parts/electric/ele-1.png")

        ele2 = pygame.image.load("models/map parts/electric/ele-2.png")

        ele3 = pygame.image.load("models/map parts/electric/ele-3.png")

        ele4 = pygame.image.load("models/map parts/electric/ele-4.png")

        ele5 = pygame.image.load("models/map parts/electric/ele-5.png")

        ele6 = pygame.image.load("models/map parts/electric/ele-6.png")

        ele7 = electric

        ele8 = redirect

        ele9 = upload

        #lowerengine

        low1 = pygame.image.load("models/map parts/engine/eng-1.png")

        low2 = pygame.image.load("models/map parts/engine/eng-2.png")

        low3 = pygame.image.load("models/map parts/engine/eng-3.png")

        low4 = pygame.image.load("models/map parts/engine/eng-4.png")

        low5 = pygame.image.load("models/map parts/engine/eng-5.png")

        low6 = pygame.image.load("models/map parts/engine/eng-6.png")

        low7 = pygame.image.load("models/map parts/engine/eng-7.png")

        low8 = pygame.image.load("models/map parts/engine/eng-8.png")

        low9 = pygame.image.load("models/map parts/engine/eng-10.png")

        low10 = pygame.image.load("models/map parts/engine/eng-11.png")

        low11 = pygame.image.load("models/map parts/engine/eng-12.png")

        low12 = pygame.image.load("models/map parts/engine/eng-13.png")

        low13 = pygame.image.load("models/map parts/engine/eng-9.png")

        lowred = redirect

        #security

        sec1 = pygame.image.load("models/map parts/security/sec-1.png")

        sec2 = pygame.image.load("models/map parts/security/sec-2.png")

        sec3 = pygame.image.load("models/map parts/security/sec-3.png")

        sec4 = pygame.image.load("models/map parts/security/sec-4.png")

        sec5 = pygame.image.load("models/map parts/security/sec-5.png")

        sec6 = pygame.image.load("models/map parts/security/sec-6.png")

        secele = electric

        #medbay

        med1 = pygame.image.load("models/map parts/medbay/med-1.png")

        med2 = pygame.image.load("models/map parts/medbay/med-2.png")

        med3 = pygame.image.load("models/map parts/medbay/med-3.png")

        med4 = pygame.image.load("models/map parts/medbay/med-4.png")

        #shields

        she1 = pygame.image.load("models/map parts/shields/she-1.png")

        she2 = pygame.image.load("models/map parts/shields/she-2.png")

        she3 = pygame.image.load("models/map parts/shields/she-3.png")

        she4 = pygame.image.load("models/map parts/shields/she-4.png")

        she5 = pygame.image.load("models/map parts/shields/she-5.png")

        she6 = pygame.image.load("models/map parts/shields/she-6.png")

        she7 = pygame.image.load("models/map parts/shields/she-7.png")

        she8 = pygame.image.load("models/map parts/shields/she-8.png")

        she9 = pygame.image.load("models/map parts/shields/she-9.png")

        she10 = redirect

        #reactor

        rec1 = pygame.image.load("models/map parts/reactor/rec-1.png")

        rec2 = pygame.image.load("models/map parts/reactor/rec-2.png")

        rec3 = pygame.image.load("models/map parts/reactor/rec-3.png")

        q = open('collision\_points.dat', 'rb')

        coll\_loc = pickle.load(q)

        collision = [Sprite(k, l) for i,j,k,l in coll\_loc]

        player = Player()

        players = pygame.sprite.Group()

        players.add(player)

        wall\_group = pygame.sprite.Group()

        for i in range(len(collision)):

            wall\_group.add(collision[i])

        topos = []

        sps = []

        #buttons

        tasks = Sprite(surface = pygame.image.load("models/buttons/tasks.png"))

        taskson = pygame.image.load("models/buttons/tasks.png")

        tasksoff = pygame.image.load("models/buttons/tasks\_off.png")

        report = Sprite(surface = pygame.image.load("models/buttons/report.png"))

        reporton = pygame.image.load("models/buttons/report.png")

        reportoff = pygame.image.load("models/buttons/report\_off.png")

        sabotage = Sprite(surface = pygame.image.load("models/buttons/sabotage.png"))

        vent = Sprite(surface = pygame.image.load("models/buttons/vent.png"))

        kill = Sprite(surface = pygame.image.load("models/buttons/kill.png"))

        killon = pygame.image.load("models/buttons/kill.png")

        killoff = pygame.image.load("models/buttons/kill\_off.png")

        security = Sprite(surface = pygame.image.load("models/buttons/security.png"))

        mapbut = Sprite(surface = pygame.image.load("models/buttons/map.png"))

        mousebut = Sprite()

        mousebut.rect.x, mousebut.rect.y = 0, 0

        tasks.rect.x, tasks.rect.y = 897, 446

        report.rect.x, report.rect.y = 892, 333

        sabotage.rect.x, sabotage.rect.y = 897, 446

        vent.rect.x, vent.rect.y = 897, 446

        kill.rect.x, kill.rect.y = 789, 444

        security.rect.x, security.rect.y = 789, 444

        mapbut.rect.x, mapbut.rect.y = 929, 74

        button\_group = pygame.sprite.Group()

        button\_group.add(tasks)

        button\_group.add(report)

        button\_group.add(sabotage)

        button\_group.add(vent)

        button\_group.add(kill)

        button\_group.add(security)

        button\_group.add(mapbut)

        but = [tasks, report, sabotage, vent, kill]

        mous\_grp = pygame.sprite.Group()

        mous\_grp.add(mousebut)

        if imposter:

            tasks.rect.x, tasks.rect.y = (0, -200)

            vent.rect.x, vent.rect.y = 0, -200

        else:

            sabotage.rect.x, sabotage.rect.y = 0, -200

            kill.rect.x, kill.rect.y = 0, -200

            vent.rect.x, vent.rect.y = 0, -200

        tskpos = [(1290, 405, 10, 10), (1290, 233, 10, 10), (1478, 397, 10, 10),

                    (920, 256, 10, 10), (841, 180, 10, 10), (117, 121, 10, 10),

                    (1260, 787, 10, 10), (1107, 817, 10, 10), (1035, 827, 10, 10),

                    (964, 860, 10, 10), (1761, 919, 10, 10), (1878, 791, 10, 10),

                    (1967, 776, 10, 10), (2028, 811, 10, 10), (2073, 1021, 10, 10),

                    (1409, 1422, 10, 10), (1176, 1769, 10, 10), (1056, 1760, 10, 10),

                    (922, 1972, 10, 10), (887, 1771, 10, 10), (612, 1981, 10, 10),

                    (254, 1724, 10, 10), (-256, 1432, 10, 10), (-266, 1201, 10, 10),

                    (-206, 1201, 10, 10), (-111, 1201, 10, 10), (54, 1201, 10, 10),

                    (-67, 997, 10, 10), (-13, 936, 10, 10), (-518, 815, 10, 10),

                    (-475, 833, 10, 10), (-719, 956, 10, 10), (-990, 1626, 10, 10),

                    (-1007, 1617, 10, 10), (-978, 1330, 10, 10), (-1001, 588, 10, 10),

                    (-917, 284, 10, 10), (639, 1091, 10, 10), (738, 1072, 10, 10),

                    (808, 1265, 10, 10), (1012, 1265, 10, 10), (1078, 1088, 10, 10),

                    (362, 1294, 10, 10), (-1316, 1020, 10, 10), (-1156, 799, 10, 10),

                    (-1366, 710, 10, 10), (-1273, 648, 10, 10), (-1278, 1344, 10, 10),

                    (340, 402, 220, 150), (-1071, 610, 10, 10)]

        ToDo = {5:1, 4:3, 3:2, 0:9, 1:3, 2:18, 6:18, 8:5, 9:2, 10:1, 11:18, 12:3, 13:8,

                14:15, 15:18, 16:14, 17:18, 19:3, 21:12, 20:2, 42:1, 23:3, 24:10, 25:1,

                26:7, 32:11, 33:6, 34:18, 43:17, 45:16, 31:1, 30:18, 35:11, 36:18, 27:19,

                28:13, 37:1, 38:4, 40:0, 49:6, 7:21, 22:20, 100:100}

        DoTo = []

        taskmgr = [Sprite(k+40, l+40) for i,j,k,l in tskpos]

        task\_group = pygame.sprite.Group()

        for i in range(len(taskmgr)):

            task\_group.add(taskmgr[i])

        f = []

        Tasks = Tasks()

        #secCam

        secCam = 0

        secC1 = pygame.image.load("models/tasks/Security Camera/sec-2.png")

        secC2 = pygame.image.load("models/tasks/Security Camera/sec-1.png")

        secC3 = pygame.image.load("models/tasks/Security Camera/sec-3.png")

        secCNum = 1

        #maps

        show\_map = False

        map1 = pygame.image.load("models/maps/1.png")

        #deadpos

        dead = []

        adminPanel = False

        #alive player position

        Player\_Pos = []

        killed\_player\_index = None

        DeadPlayers = []

        AmIDEAD = False

        DEADPOS = []

        other\_players\_group = pygame.sprite.Group()

        in\_vent = False

        near\_vent = False

        in\_vent\_time = 0

        sabotages = []

        dead\_grp = pygame.sprite.Group()

        sabo\_on = False

        sab\_fixed = False

        Emergencys = []

        #voting

        vs1 = pygame.image.load('models/voting/1.png')

        vs2 = pygame.image.load('models/voting/2.png')

        vs3 = pygame.image.load('models/voting/3.png')

        vs4 = pygame.image.load('models/voting/4.png')

        vs5 = pygame.image.load('models/voting/5.png')

        vs6 = pygame.image.load('models/voting/6.png')

        vs7 = pygame.image.load('models/voting/7.png')

        voted = None

        pressed\_on = -10

        Should\_I\_vote = False

        kick\_screen = False

        amount\_name = 0

        game\_status = None

        while True:

            c += 1

            sab\_fixed = False

            wall\_group.draw(screen)

            mousebut.rect.x, mousebut.rect.y = pygame.mouse.get\_pos()

            mous\_grp.draw(screen)

            other\_players\_group.draw(screen)

            task\_group.draw(screen)

            dead\_grp.draw(screen)

            screen.fill((0, 0, 0))

            before\_pos = a, b

            oo = []

            for i in AllTasks:

                if len(i) != 0:

                    oo.append(i[0])

                else:

                    oo.append(-10)

            if len(sabotages)>0:

                oo.append(sabotages[0][0])

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    connect.send("disconnect")

                    return 1

                if event.type == pygame.MOUSEBUTTONDOWN:

                    f.append(pygame.mouse.get\_pos())

                    if vent.rect.colliderect(mousebut.rect):

                        in\_vent = not in\_vent

                    if sabotage.rect.colliderect(mousebut.rect):

                        sabo\_on = not sabo\_on

                    #divertTo

                    divert\_To = {36:0, 34:1, 2:2, 15:3, 11:4, 6:5, 30:6, 17:0}

                    for i in range(len(but)):

                        smashhit = pygame.sprite.collide\_rect(mousebut, but[i])

                        if smashhit and pygame.mouse.get\_pressed()[0]:

                            if tasksToDo in ToDo and not imposter and tasksToDo in oo:

                                score = 0

                                sabsfixing = 0

                                pygame.mixer.Channel(4).play(pygame.mixer.Sound(f'bgs/Among Us General Sounds/task\_Inprogress.wav'))

                                if ToDo[tasksToDo] == 0:

                                    score = Tasks.swipeCard()

                                elif ToDo[tasksToDo] == 1:

                                    score = Tasks.fixWiring()

                                elif ToDo[tasksToDo] == 2:

                                    score = Tasks.emptyGarbage()

                                elif ToDo[tasksToDo] == 3:

                                    score = Tasks.upload()

                                elif ToDo[tasksToDo] == 4:

                                    score = Tasks.Download(1)

                                elif ToDo[tasksToDo] == 5:

                                    score = Tasks.clearLeaves()

                                elif ToDo[tasksToDo] == 6:

                                    score = Tasks.alignEngine()

                                elif ToDo[tasksToDo] == 7:

                                    score = Tasks.calibrate()

                                elif ToDo[tasksToDo] == 8:

                                    score = Tasks.chartCourse()

                                elif ToDo[tasksToDo] == 9:

                                    score = Tasks.weapons()

                                elif ToDo[tasksToDo] == 10:

                                    score = Tasks.divertPower(divert\_To[AllTasks[1][1]])

                                elif ToDo[tasksToDo] == 11:

                                    score = Tasks.fualEngine()

                                elif ToDo[tasksToDo] == 12:

                                    score = Tasks.fillCan()

                                elif ToDo[tasksToDo] == 13:

                                    score = Tasks.inspectSample()

                                elif ToDo[tasksToDo] == 14:

                                    score = Tasks.primeShield()

                                elif ToDo[tasksToDo] == 15:

                                    score = Tasks.stabSteering()

                                elif ToDo[tasksToDo] == 16:

                                    score = Tasks.unlockManifolds()

                                elif ToDo[tasksToDo] == 17:

                                    score = Tasks.starReactor()

                                elif ToDo[tasksToDo] == 18:

                                    score = Tasks.acceptPower()

                                elif ToDo[tasksToDo] == 19:

                                    score = Tasks.medbayScan()

                                elif len(sabotages)>0:

                                    if ToDo[sabotages[0][0]] == 21:

                                        sabsfixing = Tasks.oxygen()

                                    elif ToDo[sabotages[0][0]] == 20:

                                        sabsfixing = Tasks.electrical()

                                if sabsfixing == 1:

                                    sab\_fixed = True

                                if score == 1:

                                    pygame.mixer.Channel(4).play(pygame.mixer.Sound(f'bgs/Among Us General Sounds/task\_Complete.wav'))

                                    ids = oo.index(tasksToDo)

                                    del AllTasks[ids][0]

            keys = pygame.key.get\_pressed()

            if not in\_vent and not adminPanel and not Should\_I\_vote:

                if keys[K\_w]:

                    b += 5

                if keys[K\_a]:

                    a += 8

                if keys[K\_s]:

                    b -= 5

                if keys[K\_d]:

                    a -= 8

            #weapons

            screen.blit(bg, (0+a,0+b))

            screen.blit(weapons1, (1100+a, 260+b))

            screen.blit(weapons4, (1137+a, 204+b))

            screen.blit(weapons5, (1136+a, 286+b))

            screen.blit(weapons6, (1383+a, 463+b))

            screen.blit(weapons2, (1122+a, 256+b))

            screen.blit(caf\_weap, (979+a, 362+b))

            screen.blit(weapons3, (980+a, 193+b))

            screen.blit(weapons7, (1132+a, 242+b))

            #o2

            screen.blit(wep\_o2\_nav\_she, (1209+a, 652+b))

            screen.blit(o21, (875+a, 707+b))

            screen.blit(o22, (1006+a, 802+b))

            screen.blit(o24, (941+a, 830+b))

            screen.blit(o25, (1067+a, 722+b))

            screen.blit(o23, (1090+a, 806+b))

            if c%2 == 0:

                screen.blit(o26, (891+a, 777+b))

            else:

                screen.blit(o27, (891+a, 777+b))

            screen.blit(oredirect, (1229+a, 771+b))

            #navigation

            screen.blit(nav2, (1809+a, 798+b))

            screen.blit(nav1, (1808+a, 742+b))

            screen.blit(nav3, (1808+a, 752+b))

            screen.blit(nav4, (2008+a, 835+b))

            screen.blit(nav5, (2017+a, 1040+b))

            screen.blit(nav6, (2023+a, 930+b))

            screen.blit(nav7, (2066+a, 905+b))

            screen.blit(nav8, (2021+a, 812+b))

            screen.blit(nav9, (1946+a, 770+b))

            screen.blit(navredirect, (1866+a, 771+b))

            screen.blit(navelectric, (1747+a, 898+b))

            #admin

            screen.blit(caf\_ad\_st, (414+a, 958+b))

            screen.blit(admin1, (561+a, 1115+b))

            screen.blit(admin2, (578+a, 1056+b))

            screen.blit(admin3, (582+a, 1064+b))

            screen.blit(admin4, (824+a, 1248+b))

            screen.blit(admin5, (824+a, 1259+b))

            screen.blit(admin6, (999+a, 1259+b))

            screen.blit(admin7, (870+a, 1259+b))

            screen.blit(admin9, (831+a, 1081+b))

            screen.blit(admin10, (998+a, 1081+b))

            screen.blit(admin8, (838+a, 1099+b))

            screen.blit(adminelec, (645+a, 1094+b))

            screen.blit(adminupl, (749+a, 1084+b))

            screen.blit(adminox, (1078+a, 1095+b))

            #cafeteria

            screen.blit(caflev, (913+a, 226+b))

            screen.blit(cafup, (832+a, 159+b))

            screen.blit(cafred, (93+a, 104+b))

            #storage

            screen.blit(sto1, (80+a, 1249+b))

            screen.blit(sto2, (194+a, 1471+b))

            screen.blit(sto4, (279+a, 1700+b))

            screen.blit(sto5, (603+a, 1959+b))

            screen.blit(sto6, (365+a, 1298+b))

            screen.blit(sto7, (637+a, 1485+b))

            #communication

            screen.blit(comm1, (663+a, 1739+b))

            screen.blit(comm2, (662+a, 1696+b))

            screen.blit(comm3, (676+a, 1729+b))

            if c%2 == 0:

                screen.blit(comm8, (696+a, 1756+b))

            else:

                screen.blit(comm9, (696+a, 1756+b))

            screen.blit(comm10, (1046+a, 1752+b))

            screen.blit(comm11, (856+a, 1736+b))

            #electric

            screen.blit(ele3, (-694+a, 1147+b))

            screen.blit(ele6, (-302+a, 1293+b))

            screen.blit(ele1, (-304+a, 1353+b))

            screen.blit(ele4, ( 35+a, 1187+b))

            screen.blit(ele5, (-294+a, 1395+b))

            screen.blit(ele7, (-118+a, 1187+b))

            screen.blit(ele8, (-227+a, 1183+b))

            screen.blit(ele9, (-280+a, 1180+b))

            screen.blit(ele2, (-314+a, 1159+b))

            #security

            screen.blit(sec1, (-1055+a, 723+b))

            screen.blit(sec2, (-700+a, 729+b))

            screen.blit(sec3, (-620+a, 753+b))

            screen.blit(sec4, (-574+a, 809+b))

            screen.blit(sec5, (-475+a, 793+b))

            screen.blit(sec6, (-664+a, 780+b))

            screen.blit(secele, (-737+a, 945+b))

            #lowerengine

            screen.blit(low1, (-1097+a, 1269+b))

            screen.blit(low3, (-832+a, 1432+b))

            if c%2 == 0:

                screen.blit(low4, (-1088+a, 1394+b))

                screen.blit(low5, (-1078+a, 1607+b))

            else:

                screen.blit(low4, (-1087+a, 1389+b))

                screen.blit(low5, (-1077+a, 1606+b))

            screen.blit(low2, (-987+a, 1583+b))

            screen.blit(low6, (-993+a, 1592+b))

            if c%7 == 0:

                screen.blit(low9, (-825+a, 1548+b))

                screen.blit(low11, (-895+a, 1411+b))

            elif c%8 == 0:

                screen.blit(low10, (-863+a, 1449+b))

                screen.blit(low12, (-895+a, 1594+b))

            screen.blit(lowred, (-975+a, 1341+b))

            #upper engine

            screen.blit(low13, (-1099+a, 256+b))

            screen.blit(low3, (-838+a, 407+b))

            if c%2 == 0:

                screen.blit(low4, (-1089+a, 348+b))

            else:

                screen.blit(low4, (-1088+a, 349+b))

            screen.blit(low2, (-993+a, 556+b))

            screen.blit(low5, (-1081+a, 563+b))

            screen.blit(low6, (-997+a, 569+b))

            if c%7 == 0:

                screen.blit(low9, (-846+a, 383+b))

                screen.blit(low12, (-884+a, 539+b))

            elif c%8 == 0:

                screen.blit(low10, (-918+a, 535+b))

                screen.blit(low11, (-805+a, 433+b))

            screen.blit(ele8, (-906+a, 291+b))

            #medbay

            screen.blit(med1, (-706+a, 362+b))

            screen.blit(med2, (-401+a, 563+b))

            screen.blit(med3, (-142+a, 958+b))

            screen.blit(med4, (-52+a, 871+b))

            #shields

            screen.blit(she1, (1106+a, 1436+b))

            screen.blit(she2, (1103+a, 1429+b))

            screen.blit(she3, (1093+a, 1362+b))

            lig = [(1149, 1454), (1164, 1436), (1177, 1417), (1196, 1404),

                    (1494, 1542), (1494, 1513), (1495, 1483)]

            for i in lig[::-1]:

                screen.blit(she9, (i[0]+a, i[1]+b))

            screen.blit(she5, (1397+a, 1598+b))

            screen.blit(she6, (1131+a, 1436+b))

            screen.blit(she7, (1153+a, 1713+b))

            screen.blit(she10,(1425+a, 1411+b))

            #reactor

            screen.blit(rec2, (-1505+a, 636+b))

            screen.blit(rec3, (-1483+a, 1187+b))

            #cams

            screen.blit(cam\_off, (-15+a, 385+b))

            screen.blit(cam\_off, (-790+a, 927+b))

            screen.blit(cam\_off, (577+a, 1076+b))

            screen.blit(cam\_off, (1652+a, 878+b))

            #vent

            near\_vent = False

            vent\_pos = [(-360, 955), (-530, 1130), (-292, 1241), (-1334, 797),

                        (-766, 349), (-1233, 1166), (-765, 1689), (1290, 1770),

                        (1870, 1094), (1269, 279), (1869, 833), (833, 543), (743, 1380)]

            vent\_rect = [vent\_pic.get\_rect() for i in vent\_pos]

            for i in range(len(vent\_rect)):

                vent\_rect[i].x, vent\_rect[i].y = vent\_pos[i][0]+a, vent\_pos[i][1]+b

                if vent\_rect[i].colliderect(player.rect) and imposter:

                    near\_vent = True

            if imposter:

                if near\_vent:

                    sabotage.rect.x, sabotage.rect.y = 0, -200

                    vent.rect.x, vent.rect.y = 897, 446

                else:

                    in\_vent = False

                    vent.rect.x, vent.rect.y = 0, -200

                    sabotage.rect.x, sabotage.rect.y = 897, 446

            p = 0

            for i in vent\_rect:

                p += 1

            for i in vent\_pos:

                screen.blit(vent\_pic, (i[0]+a, i[1]+b))

            #collision

            for i in range(len(collision)):

                collision[i].rect.x,collision[i].rect.y=coll\_loc[i][0]+a,coll\_loc[i][1]+b

            coll1 = pygame.surface.Surface([150, 100])

            h = coll1.get\_rect()

            hit = pygame.sprite.spritecollide(player, wall\_group, False)

            did = 0

            prev = a, b

            #tasks

            screen.blit(weaponselectric, (1491+a, 376+b))

            screen.blit(weaponsupload, (1276+a, 216+b))

            screen.blit(weapons10, (1274+a , 360+b))

            s = Sprite(10, 10)

            s.image.fill((255, 255, 255))

            s.rect.center = pygame.mouse.get\_pos()

            security.rect.x, security.rect.y = 0, -200

            for i in range(len(taskmgr)):

                taskmgr[i].rect.x,taskmgr[i].rect.y=tskpos[i][0]+a,tskpos[i][1]+b

            todo = ()

            for i in range(len(taskmgr)):

                if pygame.sprite.collide\_rect(player, taskmgr[i]) == 1:

                    todo = 1, i

                    tasksToDo = i

                    if i == 29:

                        security.rect.x, security.rect.y = 789, 444

                        if pygame.mouse.get\_pressed()[0]:

                            cc = a, b

                            secCam = 1

                    elif i == 39:

                        todo = 1, i

                        if pygame.mouse.get\_pressed()[0]:

                            adminPanel = True

                    else:

                        security.rect.x, security.rect.y = 0, -200

                    if i == 48:

                        todo = 1, i

                        if sabotage.rect.colliderect(mousebut.rect) or tasks.rect.colliderect(mousebut.rect) and len(sabotages)==0:

                            if pygame.mouse.get\_pressed()[0]:

                                Emergencys.append(i)

                else:

                    tasks.image = tasksoff

            if len(todo) > 0:

                if todo[1] in oo or todo[1] == 48:

                    tasks.image = taskson

            else:

                tasksToDo = None

            if not AmIDEAD:

                for i in collision:

                    if pygame.sprite.collide\_rect(player, i):

                        if keys[pygame.K\_w]:

                            if abs(player.rect.top - i.rect.bottom) < 17 and hit:

                                b = before\_pos[1]

                        if keys[pygame.K\_a]:

                            if abs(player.rect.left - i.rect.right) < 17 and hit:

                                a = before\_pos[0]

                        if keys[pygame.K\_s]:

                            if abs(player.rect.bottom - i.rect.top) < 17 and hit:

                                b = before\_pos[1]

                        if keys[pygame.K\_d]:

                            if abs(player.rect.right - i.rect.left) < 17 and hit:

                                a = before\_pos[0]

            players.draw(screen)

            coll = a, b

            #on the player

            a, b = prev

            screen.blit(weapons8, (1133+a, 509+b))

            screen.blit(weaponsgreenscreen, (1304+a, 278+b))

            screen.blit(weapons9, (1394+a, 409+b))

            screen.blit(she8, (1397+a, 1448+b))

            screen.blit(she4, (1129+a, 1635+b))

            screen.blit(comm5, (772+a, 1935+b))

            screen.blit(comm4, (792+a, 1970+b))

            screen.blit(comm6, (671+a, 1849+b))

            screen.blit(comm7, (1041+a, 1846+b))

            screen.blit(sto3, (439+a, 1447+b))

            screen.blit(low3, (-832+a, 1432+b))

            if c%2 == 0:

                screen.blit(low4, (-1088+a, 1394+b))

                screen.blit(low5, (-1078+a, 1607+b))

            else:

                screen.blit(low4, (-1087+a, 1389+b))

                screen.blit(low5, (-1077+a, 1606+b))

            if c%7 == 0:

                screen.blit(low9, (-825+a, 1548+b))

                screen.blit(low11, (-895+a, 1411+b))

            elif c%8 == 0:

                screen.blit(low10, (-863+a, 1449+b))

                screen.blit(low12, (-895+a, 1594+b))

            if c%2 == 0:

                screen.blit(low4, (-1089+a, 348+b))

            else:

                screen.blit(low4, (-1088+a, 349+b))

            screen.blit(rec1, (-1466+a, 824+b))

            screen.blit(low5, (-1074+a, 563+b))

            a, b = coll

            if secCam == 1:

                screen.blit(secC1, (-200, 0))

                screen.blit(secC2, (809, 245))

                screen.blit(secC3, (82, 230))

                if 809 < pygame.mouse.get\_pos()[0] < 809+60 and 245 < pygame.mouse.get\_pos()[1] < 245+60 and pygame.mouse.get\_pressed()[0]:

                    secCNum += 0.5

                    if secCNum > 5:

                        secCNum = 1

                if 82 < pygame.mouse.get\_pos()[0] < 82+60 and 230 < pygame.mouse.get\_pos()[1] < 230+60 and pygame.mouse.get\_pressed()[0]:

                    secCNum -= 0.5

                    if secCNum < 1:

                        secCNum = 4.5

                if int(secCNum) == 1:

                    a, b = (1365, -720)

                elif int(secCNum) == 2:

                    a, b = (785, -99)

                elif int(secCNum) == 3:

                    a, b = 5, -825

                else:

                    a, b = -1025, -660

                close = pygame.image.load("models/buttons/close.png")

                screen.blit(close, (100, 25))

                if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                    secCam = 0

                    a, b = cc

                    cc = None

            #Multiplayer

            Player\_Pos = []

            dead = []

            DEADPOS = []

            voting = False

            #DeadPlayers = []

            sumTasks = 0

            for i in AllTasks:

                if len(i) != 0:

                    sumTasks += 1

            if AmIDEAD:

                server\_info = connect.send((-MyDeadPos[0], -MyDeadPos[1], player.move,

                            player.flip, My\_color, AmIDEAD, (imposter, sabotages),

                            killed\_player\_index, DeadPlayers, in\_vent, sumTasks, False,

                            [], voted, game\_status))

            else:

                MyDeadPos = a, b

                server\_info = connect.send((-a, -b, player.move, player.flip, My\_color,

                            AmIDEAD, (imposter, sabotages), killed\_player\_index, DeadPlayers,

                            in\_vent, sumTasks, sab\_fixed, Emergencys, voted, game\_status))

            Emergencys = []

            try:

                server\_info = eval(server\_info)

                ownpos = eval(server\_info[str(f"b'{connect.name}'")])[-2]

                if imposter:

                    sumTasks = 0

                else:

                    sumTasks = 6 - sumTasks

                for i in server\_info:

                    server\_info[i] = eval(server\_info[i])

                    if server\_info[i][6][0]:

                        imposterName = i

                for i in server\_info:

                    if server\_info[i][14] != None:

                        uwon = None

                        if server\_info[i][14]:

                            uwon = 'won'

                            if imposter:

                                uwon = 'lost'

                        else:

                            uwon = 'lost'

                            if imposter:

                                uwon = 'won'

                        status = 0

                        pygame.mixer.Channel(4).play(pygame.mixer.Sound(f'bgs/Among Us General Sounds/{uwon}.wav'))

                        while status != 500:

                            status += 1

                            screen.blit(pygame.image.load(f'images/{uwon}.png'), (0,0))

                            pygame.display.update()

                            clock.tick(fps)

                        connect.send("disconnect")

                        return 1

                    if len(server\_info[i][12]) > 0:

                        if "report" in server\_info[i][12]:

                            pygame.mixer.Channel(4).play(pygame.mixer.Sound(f'bgs/Among Us General Sounds/report\_Bobdyfound.wav'))

                        else:

                            pygame.mixer.Channel(4).play(pygame.mixer.Sound(f'bgs/Among Us General Sounds/alarm\_emergencymeeting.wav'))

                        Should\_I\_vote = True

                        Emergencys = server\_info[i][12]

                    if i != str(f"b'{connect.name}'"):

                        if i != imposterName:

                            sumTasks += 6-server\_info[i][10]

                        if not server\_info[i][5]:

                            if not server\_info[i][9]:

                                font1 = pygame.font.Font('freesansbold.ttf', 10)

                                Tet = i[2:-1]

                                tet = font.render(Tet, True, (255, 255, 255))

                                tetRect = tet.get\_rect()

                                screen.blit(tet, (int(server\_info[i][0])+490+a, int(server\_info[i][1])+265+b))

                                player2 = pygame.transform.flip(pygame.image.load(f"images/Sprites/Walk/walkcolor00{int(server\_info[i][2])}.png"), not server\_info[i][3], False)

                                if int(server\_info[i][2]) == 1:

                                    player2 = pygame.image.load('idle.png')

                                if i == imposterName:

                                    sabotages = server\_info[i][6][1]

                        else:

                            player2 = pygame.image.load("images/Sprites/Death/Dead0033.png")

                            dead.append((int(server\_info[i][0])+530+a, int(server\_info[i][1])+305+b))

                            DEADPOS.append((int(server\_info[i][0])+530+a, int(server\_info[i][1])+305+b))

                        if not server\_info[i][9]:

                            player2 = pygame.transform.scale(player2, (78-25,103-30))

                            player2 = colorchanger(player2, server\_info[i][4])

                            screen.blit(player2, (int(server\_info[i][0])+500+a, int(server\_info[i][1])+275+b))

                    if server\_info[i][7] != None:

                        dead.append((int(server\_info[i][0])+530+a, int(server\_info[i][1])+305+b))

                        DEADPOS.append((int(server\_info[i][0])+530+a, int(server\_info[i][1])+305+b))

                        DeadPlayers.append(server\_info[i][7])

                        if server\_info[i][7] == ownpos and not AmIDEAD:

                            DeadPlayers.append(ownpos)

                            AmIDEAD = True

                            bg1 = pygame.image.load("images/death/bg.png")

                            lo = 1

                            start = True

                            while start:

                                lo+= 0.2

                                if int(lo) == 22:

                                    start = False

                                    lo = 1

                                screen.fill(0)

                                screen.blit(bg1, (27, 64))

                                screen.blit(pygame.image.load(f"images/death/{int(lo)}.png"), (334, 242))

                                screen.blit(pygame.image.load(f"images/death/de{int(lo)}.png"), (512, 218))

                                pygame.display.update()

                                clock.tick(fps)

                    if server\_info[i][11]:

                        sabotages = []

                    Player\_Pos.append((int(server\_info[i][0])+530+a, int(server\_info[i][1])+305+b))

            except Exception as e:

                pass

            #Player Pos

            other\_players\_group = pygame.sprite.Group()

            kill\_but = 0

            for i in range(len(Player\_Pos)):

                if i != ownpos:

                    pp = Player\_Pos[i]

                    Player\_Pos[i] = Sprite(100, 100)

                    Player\_Pos[i].rect.center = pp[0], pp[1]

                    other\_players\_group.add(Player\_Pos[i])

            canKill = False

            killed\_player\_index = None

            for i in range(len(Player\_Pos)):

                if i != ownpos:

                    if pygame.sprite.collide\_rect(player, Player\_Pos[i]) and i not in DeadPlayers:

                        canKill = True

                        if pygame.sprite.collide\_rect(s, kill) and pygame.mouse.get\_pressed()[0]:

                            killed = Player\_Pos[i].rect.x, Player\_Pos[i].rect.y

                            killed\_player\_index = i

                            break

            if canKill:

                kill.image = killon

            else:

                kill.image = killoff

            #dead

            dead\_grp = pygame.sprite.Group()

            reportbut = 0

            for i in range(len(dead)):

                try:

                    ded = dead[i][0], dead[i][1]

                    dead[i] = Sprite(100, 100)

                    dead[i].rect.center = ded[0], ded[1]

                    dead\_grp.add(dead[i])

                except:

                    pass

            for i in dead:

                if pygame.sprite.collide\_rect(player, i) == 1:

                    reportbut = 1

                    if pygame.mouse.get\_pressed()[0] and report.rect.colliderect(mousebut.rect):

                        Emergencys.append("report")

            if reportbut == 1:

                report.image = reporton

            else:

                report.image = reportoff

            if secCam != 1:

                if imposter:

                    text0 = font.render('Fake Tasks', True, (255, 0, 0))

                    textrect0 = text0.get\_rect()

                    screen.blit(text0, (10, 70-font\_size))

                text1 = font.render(Text1 + f'({len(AllTasks[0])})', True, (255, 255, 255))

                if len(AllTasks[0]) == 0:

                    text1 = font.render(Text1, True, (0, 255, 0))

                text2 = font.render(Text2 + f'({len(AllTasks[1])})', True, (255, 255, 255))

                if len(AllTasks[1]) == 0:

                    text2 = font.render(Text2, True, (0, 255, 0))

                text3 = font.render(Text3+f'({len(AllTasks[2])})', True, (255, 255, 255))

                if len(AllTasks[2]) == 0:

                    text3 = font.render(Text3, True, (0, 255, 0))

                text4 = font.render(Text4+f'({len(AllTasks[3])})', True, (255, 255, 255))

                if len(AllTasks[3]) == 0:

                    text4 = font.render(Text4, True, (0, 255, 0))

                text5 = font.render(Text5+f'({len(AllTasks[4])})', True, (255, 255, 255))

                if len(AllTasks[4]) == 0:

                    text5 = font.render(Text5, True, (0, 255, 0))

                text6 = font.render(Text6+f'({len(AllTasks[5])})', True, (255, 255, 255))

                if len(AllTasks[5]) == 0:

                    text6 = font.render(Text6, True, (0, 255, 0))

                textRect1 = text1.get\_rect()

                textRect2 = text2.get\_rect()

                textRect3 = text3.get\_rect()

                textRect4 = text4.get\_rect()

                textRect5 = text5.get\_rect()

                textRect6 = text6.get\_rect()

                screen.blit(text1, (10, 70))

                screen.blit(text2, (10, 70+font\_size))

                screen.blit(text3, (10, 70+font\_size\*2))

                screen.blit(text4, (10, 70+font\_size\*3))

                screen.blit(text5, (10, 70+font\_size\*4))

                screen.blit(text6, (10, 70+font\_size\*5))

                if 929 < pygame.mouse.get\_pos()[0] < 1000 and 74 < pygame.mouse.get\_pos()[1] < 154 and pygame.mouse.get\_pressed()[0]:

                    show\_map = True

                if show\_map:

                    screen.blit(map1, (0, 0))

                    screen.blit(pygame.image.load("models/maps/4.png"), ((-a/3657)\*1000 + 500, (-b/2058)\*550 + 60))

                    ts\_im = pygame.image.load("models/maps/3.png")

                    ts\_imx = pygame.image.load("models/maps/3.png").get\_size()[0]//2

                    ts\_imy = pygame.image.load("models/maps/3.png").get\_size()[1]//2

                    map\_task\_pos = {5:(411, 24), 10:(882, 231), 17:(683, 471), 42:(485, 348),

                                    37:(565, 290), 25:(372, 290), 31:(191, 249), 24:(339, 315),

                                    36:(169, 74), 34:(118, 356), 2:(806, 102), 15:(790, 377),

                                    11:(907, 204), 6:(725, 201), 30:(267, 217), 38:(609, 290),

                                    4:(783, 82), 1:(740, 55), 12:(928, 201), 19:(628, 470),

                                    23:(314, 316), 21:(466, 490), 35:(108, 178), 32:(124, 456),

                                    49:(98, 175), 33:(104, 457), 28:(399, 239), 27:(366, 263),

                                    40:(666, 345), 8:(659, 215), 26:(408, 316), 0:(754, 123),

                                    14:(975, 255), 13:(954, 214), 16:(720, 490), 45:(20, 194),

                                    43:(28, 282), 3:(645, 59), 9:(644, 235), 20:(533, 544)}

                    sab\_pos = {22:(309, 349), 7:(655, 218)}

                    for i in sabotages:

                        if i[0] in sab\_pos:

                            screen.blit(pygame.image.load("models/maps/4.png"), i[1])

                    for i in oo:

                        if i in map\_task\_pos:

                            screen.blit(ts\_im, (map\_task\_pos[i][0]-ts\_imx, map\_task\_pos[i][1]-ts\_imy))

                    close = pygame.image.load("models/buttons/close.png")

                    screen.blit(close, (100, 25))

                    if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                        show\_map = False

                if sabo\_on and len(sabotages) == 0:

                    sab\_pos = {22:(309, 349), 7:(655, 218)}

                    door\_sab\_pos = [(490, 101), (305, 191), (115, 101), (115, 377), (206, 238),

                                    (292, 414), (458, 422)]

                    sabs = {22:pygame.image.load("models/maps/7.png"), 7:pygame.image.load("models/maps/6.png")}

                    close\_doors = pygame.image.load("models/maps/5.png")

                    door1\_pos = [(-283, 553), (-928, 1256), (-314, 1628), (425, 1230), (426, 941)]

                    door2\_pos = [(1, 357), (-701, 355), (-697, 900), (91, 1668), (623, 1476), (956, 358)]

                    screen.blit(pygame.image.load("models/maps/2.png"), (0, 0))

                    for i in door\_sab\_pos:

                        screen.blit(close\_doors, i)

                    for i in sab\_pos:

                        screen.blit(sabs[i], sab\_pos[i])

                        if sab\_pos[i][0] < pygame.mouse.get\_pos()[0] < sab\_pos[i][0]+60 and sab\_pos[i][1] < pygame.mouse.get\_pos()[1] < sab\_pos[i][1]+60:

                            if pygame.mouse.get\_pressed()[0]:

                                sabotages.append((i,sab\_pos[i]))

                    close = pygame.image.load("models/buttons/close.png")

                    screen.blit(close, (100, 25))

                    if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                        sabo\_on = False

            if adminPanel:

                screen.blit(pygame.image.load("models/maps/8.png"), (0,0))

                close = pygame.image.load("models/buttons/close.png")

                for i in server\_info:

                    screen.blit(pygame.image.load("models/mini.png"), ((server\_info[i][0]/3657)\*1000 + 500, (server\_info[i][1]/2058)\*550 + 60))

                screen.blit(close, (100, 25))

                if 117 < pygame.mouse.get\_pos()[0] < 155 and 41 < pygame.mouse.get\_pos()[1] < 78 and pygame.mouse.get\_pressed()[0]:

                    adminPanel = False

            #buttons

            button\_group.draw(screen)

            if Should\_I\_vote:

                Emergencys = []

                screen.blit(vs1, (77, -10))

                names = list(server\_info.keys())

                totVotes = 0

                votes = []

                for i in range(len(server\_info)):

                    if i <= 4:

                        screen.blit(vs2, (121, 100+74\*i))

                        try:

                            screen.blit(colorchanger(vs4, server\_info[names[i]][4]), (128, 106+74\*i))

                        except:

                            pass

                        Text = font.render(names[i], True, (0,0,0))

                        Textrect = Text.get\_rect()

                        screen.blit(Text, (190, 116+74\*i))

                        try:

                            if server\_info[names[i]][13] != None:

                                totVotes += 1

                                votes.append(server\_info[names[i]][13])

                                screen.blit(vs6, (121, 100+74\*i))

                        except:

                            pass

                        if 121 < pygame.mouse.get\_pos()[0] < 121 + 346 and 100+74\*i < pygame.mouse.get\_pos()[1] < 160+74\*i:

                            if pygame.mouse.get\_pressed()[0]:

                                pressed\_on = i

                        if voted == None and not AmIDEAD:

                            screen.blit(vs3, (354, 105+74\*(pressed\_on)))

                            if 105+74\*pressed\_on < pygame.mouse.get\_pos()[1] < 150+74\*pressed\_on:

                                if 355 < pygame.mouse.get\_pos()[0] < 402:

                                    if pygame.mouse.get\_pressed()[0]:

                                        voted = names[pressed\_on]

                                if 410 < pygame.mouse.get\_pos()[0] < 456:

                                    if pygame.mouse.get\_pressed()[0]:

                                        pressed\_on = -10

                    else:

                        screen.blit(vs2, (491, 100+74\*(i-5)))

                        screen.blit(colorchanger(vs4, server\_info[names[i]][4]), (491, 106+74\*(i-5)))

                        Text = font.render(names[i], True, (0,0,0))

                        Textrect = Text.get\_rect()

                        screen.blit(Text, (500, 116+74\*(i-5)))

                        try:

                            if server\_info[names[i]][13] != None:

                                totVotes += 1

                                votes.append(server\_info[names[i]][13])

                                screen.blit(vs6, (491, 100+74\*(i-5)))

                        except:

                            pass

                        if 491 < pygame.mouse.get\_pos()[0] < 491 + 346 and 100+74\*(i-5) < pygame.mouse.get\_pos()[1] < 160+74\*(i-5):

                            if pygame.mouse.get\_pressed()[0]:

                                pressed\_on = i

                        if voted == None and not AmIDEAD:

                            screen.blit(vs3, (728, 105+74\*(pressed\_on-5)))

                            if 105+74\*(pressed\_on-5) < pygame.mouse.get\_pos()[1] < 150+74\*(pressed\_on-5):

                                if 355 < pygame.mouse.get\_pos()[0] < 402:

                                    if pygame.mouse.get\_pressed()[0]:

                                        voted = names[pressed\_on]

                                if 410 < pygame.mouse.get\_pos()[0] < 456:

                                    if pygame.mouse.get\_pressed()[0]:

                                        pressed\_on = -10

                if AmIDEAD:

                    screen.blit(pygame.image.load("models/voting/7.png"), (110, 22))

                if totVotes == len(server\_info)-len(DeadPlayers):

                    Should\_I\_vote = False

                    kick\_screen = True

                    amount\_name = 0

                    a, b = 0, 0

                    voted = None

                    Emergencys = []

                    got\_votes = {}

                    for i in votes:

                        got\_votes[i] = votes.count(i)

                    prev\_max = (0, '')

                    max\_votes = (0,'')

                    for i in got\_votes:

                        if max\_votes[0] <= got\_votes[i]:

                            prev\_max = max\_votes

                            max\_votes = (got\_votes[i], i)

                    if max\_votes[0] == prev\_max[0]:

                        Tie = True

                    else:

                        Tie = False

                        if max\_votes[1][2:-1] == connect.name:

                            AmIDEAD = True

                        MyDeadPos = -1000, -1000

            if kick\_screen:

                screen.fill(0)

                amount\_name += 0.1

                a, b = 0, 0

                if not Tie:

                    if server\_info[max\_votes[1]][6][0]:

                        time.sleep(5)

                        game\_status = True

                    Text = font.render((max\_votes[1][2:-1]+'  WAS EJECTED')[:int(amount\_name)], True, (255,255,255))

                else:

                    Text = font.render('NO ONE WAS EJECTED'[:int(amount\_name)], True, (255,255,255))

                Textrect = Text.get\_rect()

                screen.blit(Text, (363, 243))

                if amount\_name > len(max\_votes[1])+50:

                    kick\_screen = False

            if len(sabotages)>0:

                if c%50 == 0:

                    pygame.mixer.Channel(4).play(pygame.mixer.Sound(f'bgs/Among Us General Sounds/Alarm\_sabotage.wav'))

                    screen.fill((255, 0, 0))

            if len(server\_info)-len(DeadPlayers) == 2 and imposter:

                time.sleep(5)

                game\_status = False

            if sumTasks == len(server\_info)\*6 - 6:

                time.sleep(5)

                game\_status = True

            else:

                pygame.draw.rect(screen, (0, 255, 0), (16, 19, sumTasks\*10, 30))

                for i in range(len(server\_info)):

                    pygame.draw.rect(screen, (255, 255, 255), (16, 19, ((i+1)\*6)\*10, 30), 5)

            players.update(secCam, My\_color, in\_vent, AmIDEAD)

            s.update()

            pygame.display.update()

            clock.tick(fps)

SERVER.PY

import socket

import threading

HEADER = 64

PORT = 5050

SERVER = "192.168.43.151" #socket.gethostbyname(socket.gethostname())

ADDR = (SERVER, PORT)

FORMAT = 'utf-8'

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.bind(ADDR)

player\_pos = {}

def handle\_client(conn, addr, name='supper'):

    print(f"[NEW CONNECTION] {addr} {name} connected.")

    #player\_pos contains the position of each user

    player\_pos[name] = (0,0)

    player\_index = len(player\_pos)-1

    msg = conn.recv(2048).decode(FORMAT)

    player\_pos[name] = eval(msg)

    if player\_index == 0:

        player\_pos[name] = player\_pos[name][:-1] + (True, )

    player\_pos[name] = str(player\_pos[name])

    conn.send(str(player\_pos).encode(FORMAT))

    connected = True

    while connected:

        #disconnect properly only if disconnect message comes

        msg = conn.recv(2048).decode(FORMAT)

        #print(player\_pos)

        if 'disconnect' in msg.lower():

            connected = False

        else:

            #print(f"[{addr}] {msg}")

            player\_pos[name] = msg[:-1] + ', ' + str(player\_index) +', "' + name + '")'

            conn.send(str(player\_pos).encode(FORMAT))

    conn.close()

    del player\_pos[name]

    print(f"[SERVER] {addr} lost connection..")

def start():

    #listen for incomming connections

    server.listen()

    print(f"[LISTENING] server is listening on {SERVER}")

    while True:

        #getting the address of the user

        conn, addr = server.accept()

        if len(player\_pos) > 10:

            conn.send('hey um 10 players are playing'.encode(FORMAT))

        #starting the thread for a perticular address

        name = conn.recv(2048).decode(FORMAT)

        conn.send(f'hey {name} u are connected. ENJOY.'.encode(FORMAT))

        thread = threading.Thread(target=handle\_client, args=(conn, addr, name))

        thread.start()

        print(f"[ACTIVE CONNECTIONS] {threading.activeCount()-1}")

print("[STARTING] server is starting...")

start()

SHADOW CASTING FUNCTION

#putting these in the while loop of game does the job

view = pygame.Surface.convert\_alpha(pygame.Surface([1000, 550]))

    view.fill((0, 0, 0, 150))

    for i in range(255, 0, -1):

        pygame.draw.circle(view, (0,0,0,i), player.rect.center, i\*3)

    for i in range(len(collision)):

        if ((player.rect.center[0]-collision[i].rect.center[0])\*\*2 + (player.rect.center[1]-collision[i].rect.center[1])\*\*2)\*\*(1/2) <= 250:

            try:

                if collision[i].rect.center[1] < player.rect.center[1]:

                    pygame.draw.line(screen, (0, 255, 0), player.rect.center, (coll\_loc[i][0]+a, coll\_loc[i][1]+b))

                    pygame.draw.line(screen, (0, 255, 0), player.rect.center, (coll\_loc[i][0]+a+coll\_loc[i][2], coll\_loc[i][1]+b+coll\_loc[i][3]))

                else:

                    pygame.draw.line(screen, (0, 0, 255), player.rect.center, (coll\_loc[i][0]+a, coll\_loc[i][1]+b))

                    pygame.draw.line(screen, (0, 0, 255), player.rect.center, (coll\_loc[i][0]+a+coll\_loc[i][2], coll\_loc[i][1]+b+coll\_loc[i][3]))

                pygame.draw.polygon(view, (0, 255, 0, 0), (player.rect.center, (coll\_loc[i][0]+a, coll\_loc[i][1]+b), (coll\_loc[i-1][0]+a, coll\_loc[i-1][1]+b)))

                pygame.draw.polygon(view, (0, 255, 0, 0), (player.rect.center, (coll\_loc[i][0]+a+coll\_loc[i][2], coll\_loc[i][1]+b+coll\_loc[i][3]), (coll\_loc[i+1][0]+a, coll\_loc[i+1][1]+b)))

            except:

                pass

CLIENT.PY

import socket

import time

from tkinter import \*

import random

root = Tk()

e = Entry(root, width=50)

e.insert(0, "192.168.43.151")

e.pack()

IP = 0

NAME = '0'

def getIp():

    global IP

    IP = e.get()

    root.destroy()

    print(IP)

def getName():

    global NAME

    NAME = e.get()

    root.destroy()

def create():

    global root, e

    root = Tk()

    e = Entry(root, width=50)

    e.pack()

class client():

    def \_\_init\_\_(self):

        self.HEADER = 64

        self.PORT = 5050

        self.FORMAT = 'utf-8'

        self.DISCONNECT\_MESSAGE = "DISCONNECT!"

        while IP==0:

            myButton = Button(root, text="Enter Server IP", command=getIp)

            myButton.pack()

            root.mainloop()

        self.SERVER = IP #"192.168.43.151"

        create()

        self.ADDR = (self.SERVER, self.PORT)    #"192.168.43.245"

        self.client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

        self.client.connect(self.ADDR)

        while NAME == '0':

            myButton = Button(root, text="Enter Name", command=getName)

            myButton.pack()

            root.mainloop()

        self.name = NAME

        while 'change it' in self.send(self.name.encode(self.FORMAT)):

            create()

            myButton = Button(root, text="Name Already taken\nEnter another", command=getName)

            myButton.pack()

            self.name = NAME

            root.mainloop()

        self.color = random.choice([(255, 0, 0), (0, 0, 255), (0, 255, 0), (255, 255, 0), (255, 128, 0), (0, 0, 0), (255, 255, 255), (255, 0, 255), (0, 255, 255), (102, 51, 0), (0, 204, 0)])

    def send(self,msg):

        self.client.send(str(msg).encode(self.FORMAT))

        return self.client.recv(2048).decode(self.FORMAT)

if \_\_name\_\_ == "\_\_main\_\_":

    a = client()

    while True:

        mess = input('enter message')

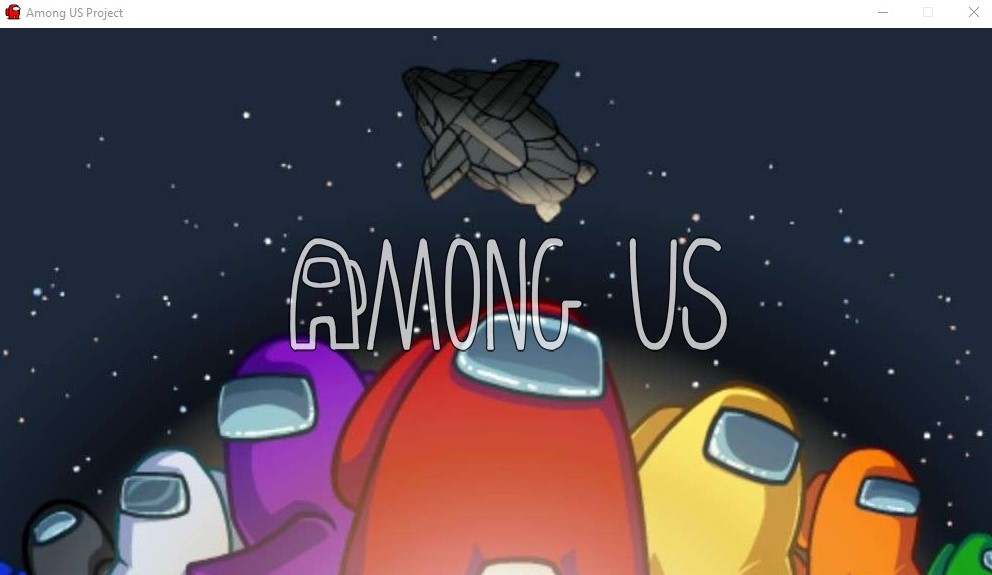
        if mess != '':

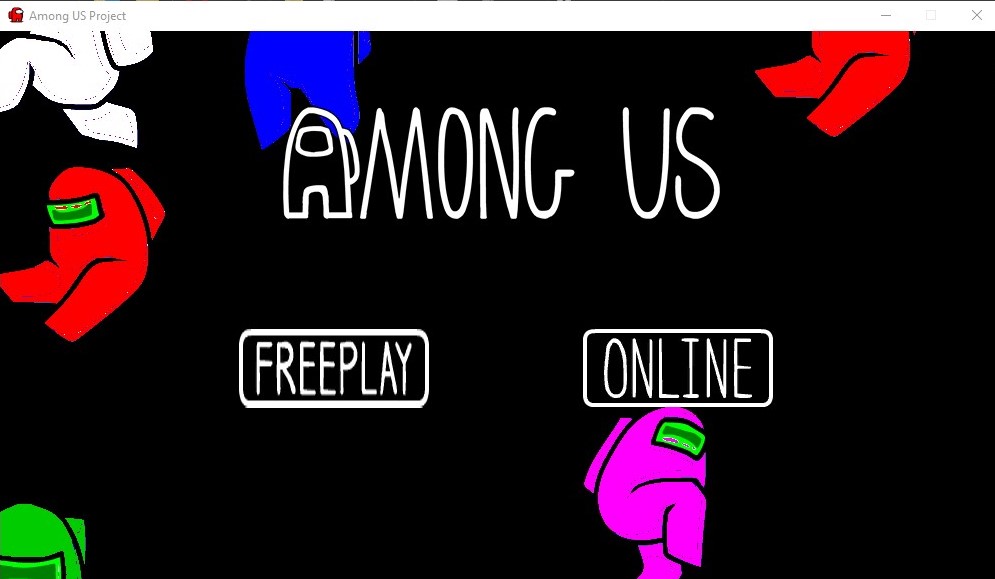
            a.send(mess)

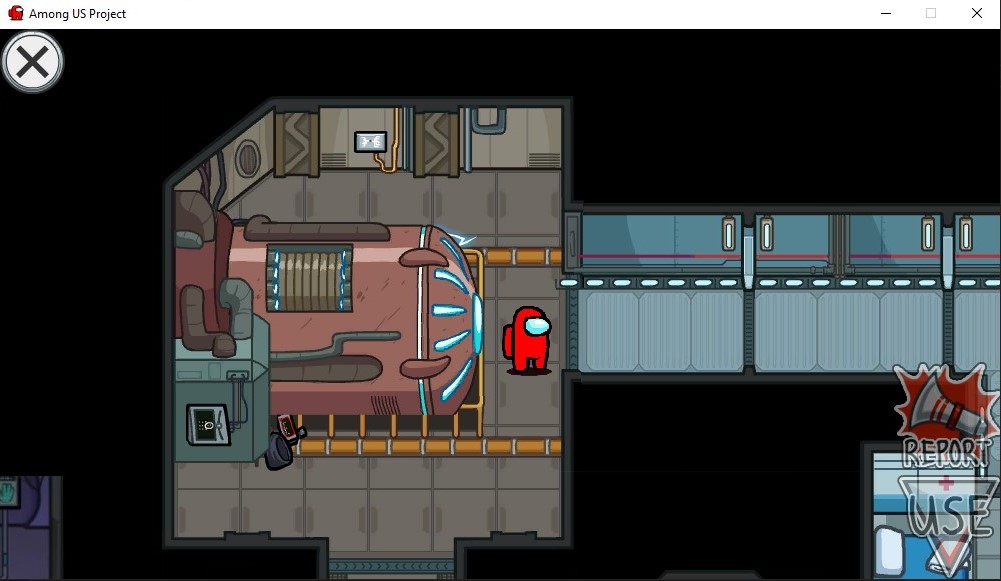
        else:

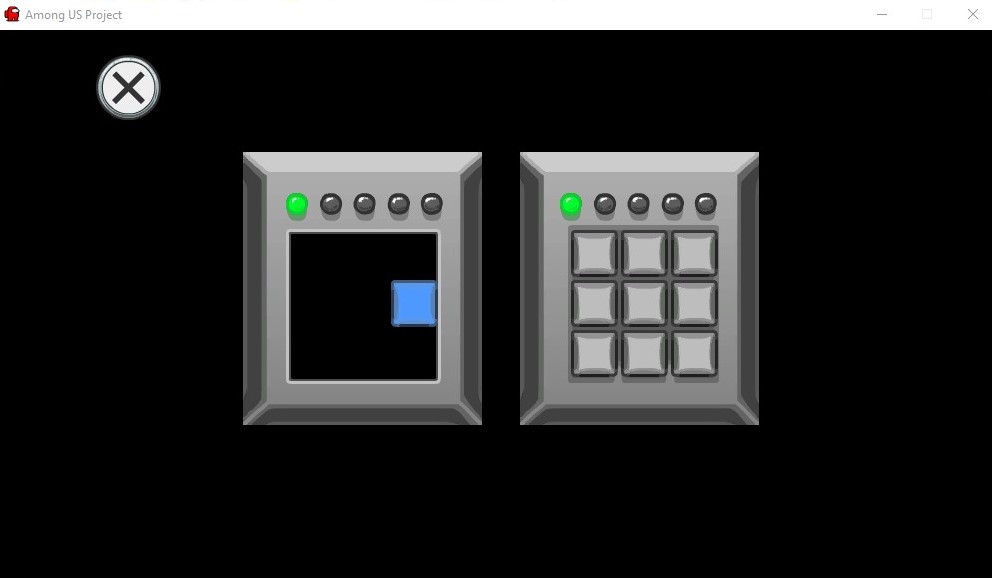
            a.send('disconnect')

            break

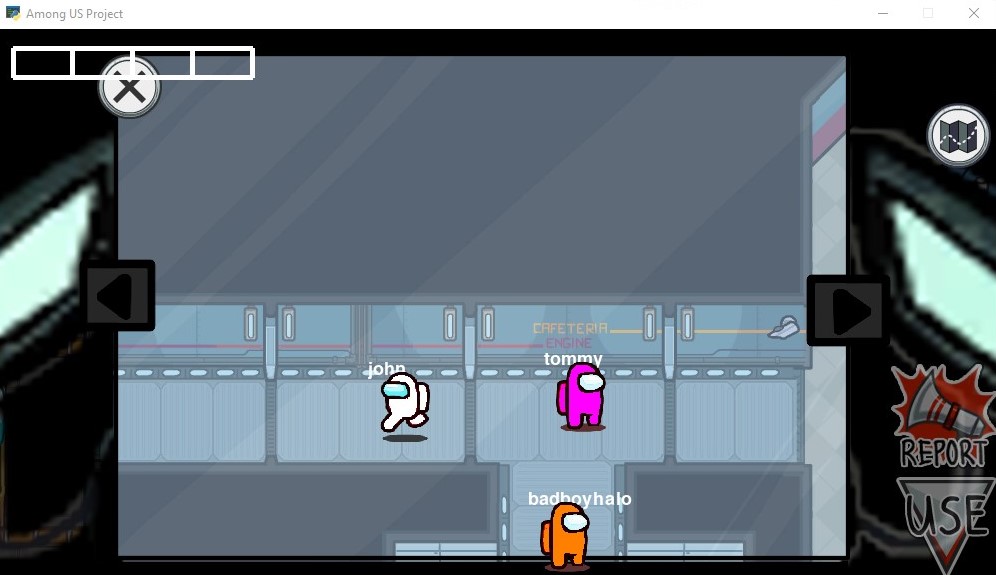
PHOTO GALLERY

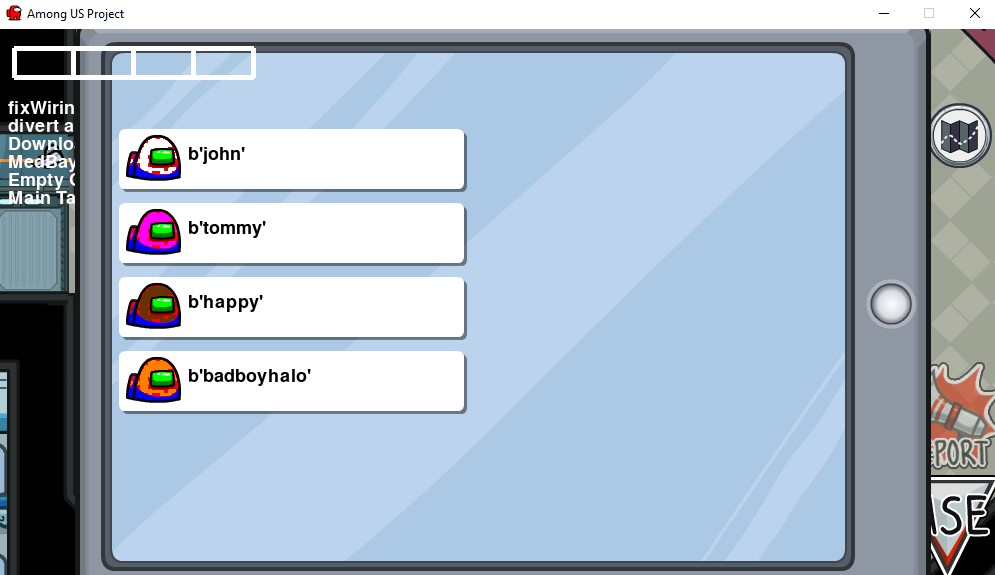
****

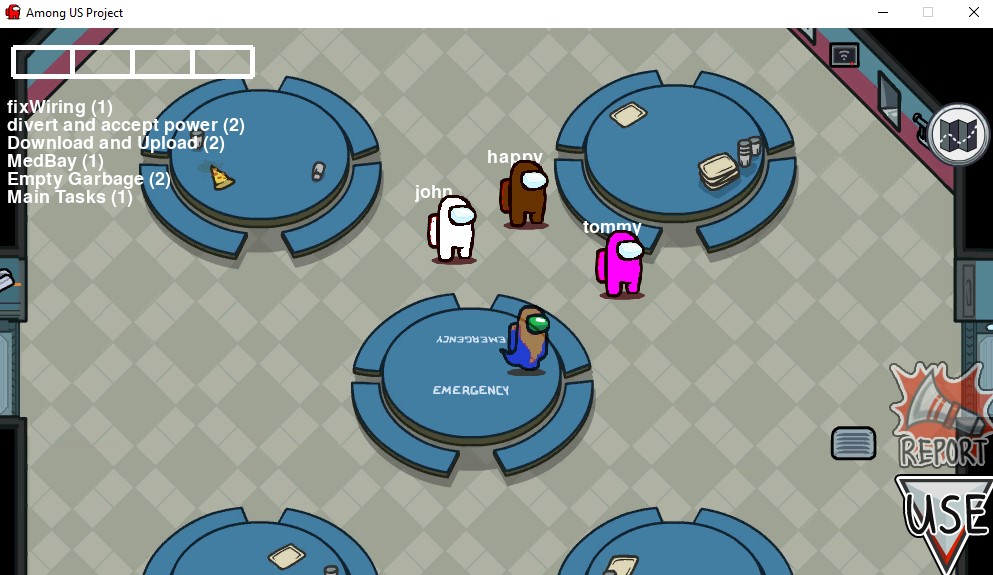
****

****

****

****

****

****

FUTURE ENHANCEMENTS

* Adding shadow casting to game.
* ****Putting chatting screen.
* Improving performance.
* Making the code more OOP.
* Making LAN to global.

BIBLIOGRAPHY

This game would not have been possible without the help from the following:

* Python forms
* Socket programming by tech with tim
* Sprites and music from spriters-resources.com
* And all the various forms and community of devalopers.

 THANK YOU



Scan this qr code or click on the link below to access the github page containing the code and images for this project. <https://github.com/allamvignesh/among_us_pygame.git>