BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:		**Surname**	**Given Names**	**Student ID Number**
	1.	**Muhammad Zavier**	**Naufal**	**2101714843**

Course Code	**: COMP6335**	**Course Name**	**: Introduction to Programming**
Class	**: L1AC**	**Name of Lecturer(s)**	: 1. Ida Bagus Kerthyayana Manuaba 2. Jude Martinez
Major	**: CS**		
Title of Assignment (if any)	: House: Horror Game		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 6-11-2017**	**Submission Date**	**: 6-11-2017**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

1. Naufal Muhammad Zavier

Table of Contents

“House: Horror Game”

Name: Naufal Muhammad Zavier

ID: 2101714843

I. Description

The program I created is a horror game based on the online point and click franchise, “House”. The program is not a recreation of the game found online, but instead is a parody of it. The game runs with python3 and utilizes Pygame at its base. All of the code used is written from scratch, and no outside code is used for the game. The intention of this game is to scare people through utilizing jump scares and by creating an unsettling and unpredictable atmosphere. The game uses import modules and packages from Pygame, and also uses custom made modules, classes and function.

Note: You have to install the fonts “Diediedie” and “YouMurderer BB” (found in the folder) into your device for optimal viewing experience.

II. Design/Plan

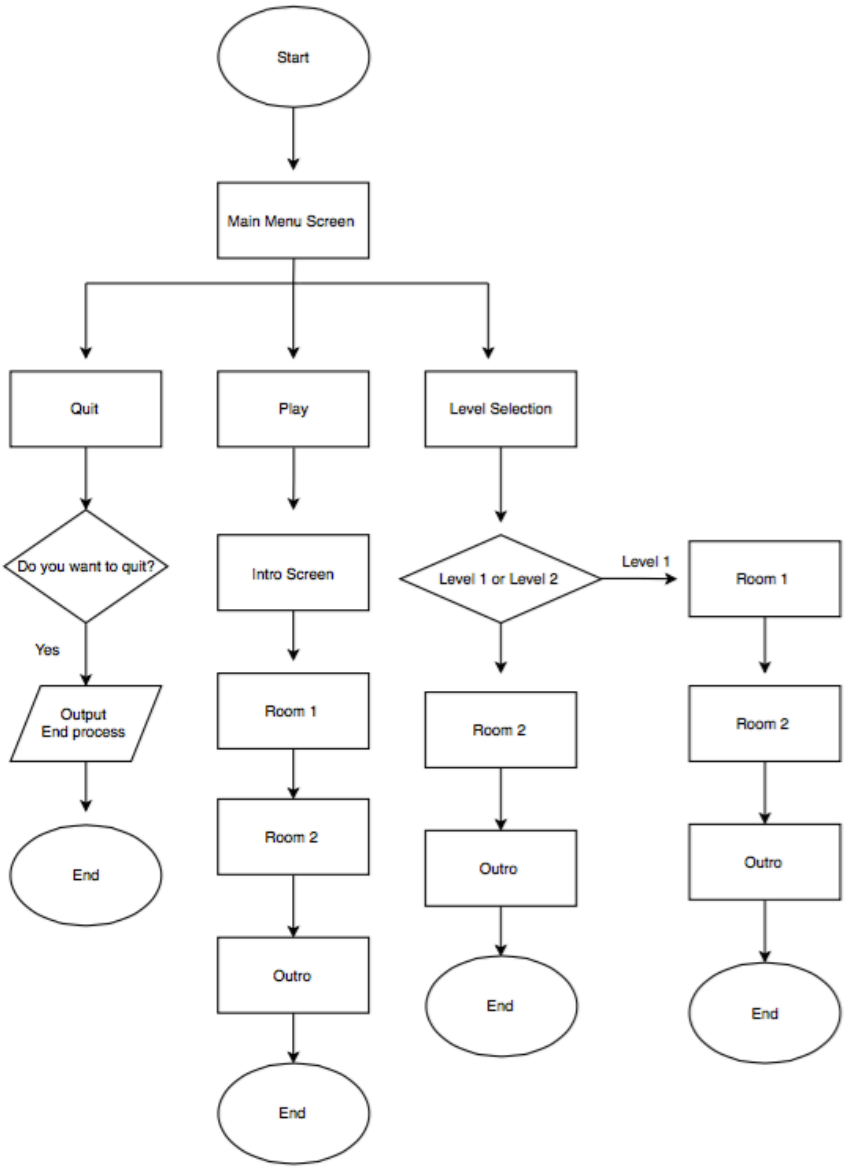

Figure 1 Project Flowchart

The Program's Code

Module – Start_Game.py

This module runs and starts the game. Having this module to run the game prevents issue with the outro of the game not appearing after the game is completed.

Module - HouseClasses.py

This module contains all the custom created classes that are designed to be reusable throughout the program. This module uses imports from pygame modules. These are the classes contained inside this module:

class imagesprite(Sprite):

```
class imagesprite(Sprite):

    '''This class is used to create and display an image as a sprite onto the screen'''

    def __init__(self, filename, xpos, ypos):  #fill in the parameters so that it is reusable
        Sprite.__init__(self)
        self.image = image.load(filename)
        self.rect = self.image.get_rect()
        self.rect.x = xpos
        self.rect.y = ypos
```

Figure 2: class imagesprite(Sprite)

The class imagesprite does exactly as it's labeled, it converts an image file into a rectangle/rect so that it obtains properties such as 'collidepoint' which is used frequently throughout the program. The parameters contain the variable filename, xpos and ypos, so that this class can be used repeatedly since you can set any image and position whenever you call this class. The parameters, filename, xpos and ypos as seen in Figure 1 corresponds to the position of the argument for filename, horizontal position and vertical position.
For example:

```
while True:
    family = imagesprite("Family.jpg", 100, 0)
    call_sprite(family, "Image", mainscreen)
    image_hover(family.rect), image_click(family.rect, e)
```

Figure 3: Example of class usage

The code given in Figure 2 calls the class imagesprite, and you can set the parameters with any image file in the folder in the format shown above, as well as its position in the window screen.

class textsprite(Sprite):

```
class textsprite(Sprite):

    '''This class is used to create and display a text as a sprite onto the screen'''

    def __init__(self, fontstyle, text, fontsize, xpos, ypos, R, G, B):
        Sprite.__init__(self)
        self.font = pygame.font.SysFont(fontstyle, fontsize)
        self.image = self.font.render(text, False, (R, G, B))
        self.rect = self.image.get_rect()
        self.rect.x = xpos
        self.rect.y = ypos
```

Figure 4: class textsprite(Sprite)

The second class is textsprite, which creates a text in form of a sprite, acquiring features that are found in rect. This class acts similar to the first class, imagesprite, and is used in a similar manner. The parameters allow you to fill in the desired font style, font size, position, color and text, so that is reusable. The font render is transformed into an image, and then later into a rect/sprite. This class is called in a similar fashion to imagesprite, here's an example:

```
play = textsprite("YouMurderer BB", "Play", 60, 100, 100, 255, 255, 255)
quit = textsprite("YouMurderer BB", "Quit", 60, 100, 300, 255, 255, 255)
level = textsprite("YouMurderer BB", "Level", 60, 100, 200, 255, 255, 255)
```

Figure 5: Example of textsprite usage

As seen in Figure 4, the class can be used more than once, and each time a different text is called with different values inputted in its parameters.

class MainScreen():

```
class MainScreen():

    '''This class is used to set an image as the background'''

    def __init__(self, imagefile):
        self.display = pygame.display.set_mode((1120, 590))
        pygame.display.set_caption("House")
        self.image = image.load(imagefile)
        self.display.blit(self.image, (0,0))
```

Figure 6: class MainScreen()

Shown in Figure 5, the class MainScreen() allows for the background to be an image instead of a color fill. The image loaded is based on what file is given in the parameter of imagefile, and the last line of the code will blit the image into the display. It is important to note that the image should be edited to fit the size given in display.set_mode, otherwise it will not be scaled properly.

Example of usage:

```
mainscreen = MainScreen('unnamed.jpg')
```

Figure 7: Example of code usage

As seen in Figure 6, the parameters are filled in with the image name and type.

```
mainscreen.display.blit(mainscreen.image, (0,0))
```

Figure 8 Calling the background image

In order for the background image to appear in the window screen, you have to blit the variable given to the class as shown in Figure 7.

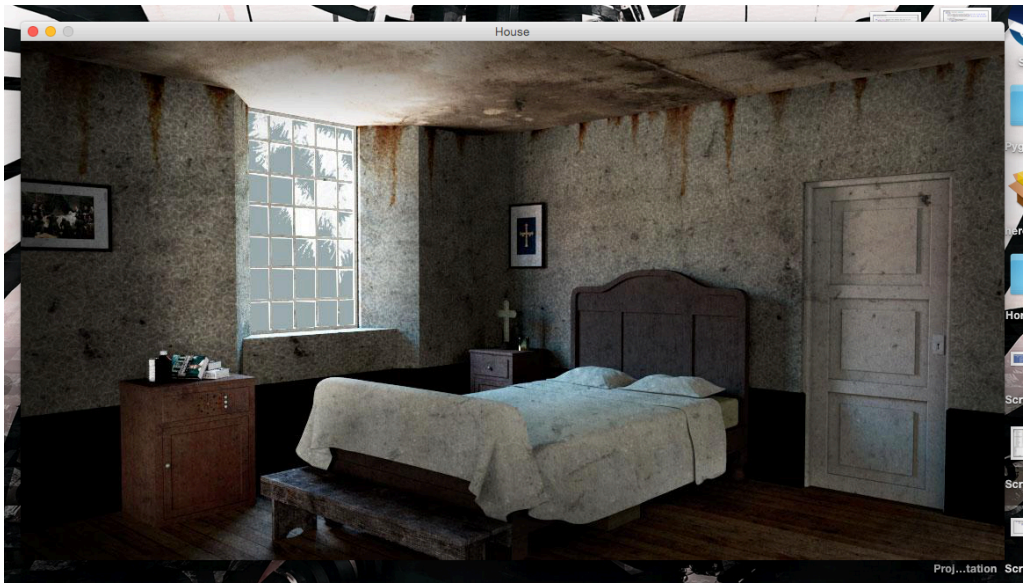

Figure 9: Image as background

As shown above in Figure 8, the image is the background of the entire screen.

class sound()

```
class sound():

    '''This class is used to play the music continouslt without interrruptions'''

    def __init__(self, musicfile):
        pygame.mixer.Sound(musicfile).play() #Use Sound instead of music as it won't
```

Figure 10: class sound()

This class is straightforward, this class plays the music inserted in the parameter continuously with no interruptions with other sound effects in the game. It used pygame.mixer.Sound instead of pygame. mixer.music as the former allows up to 8 sound files being played at the same time.

class jump()

```
class jump():

    '''This class is used to play the jumpscare sound effect'''

    def __init__(self):
        pygame.mixer.music.load("Jump2.wav") #Use mixer.music so
        pygame.mixer.music.play() #plays the sound file
```

Figure 11: class jump()

This class is specifically for playing jump scare sound effect, it used ‘music’ instead of ‘sound’ as sound has problems in playing new sound files.

class coloredbackground()

```
class coloredbackground():

    '''This class is used to create a colored background'''

    def __init__(self, R, G, B):
        self.screen = pygame.display.set_mode((1120, 590))
        pygame.display.set_caption("House")
        self.screen.fill((R, G, B))
```

Figure 12: class coloredbackground()

This class creates a screen in which the background is filled with a color based on the value given for parameter for R, G, B.

Module – HouseFunctions.py

This module contains custom built functions, in which like the classes are reusable. This module uses imports from the classes and pygame modules. These are the functions inside this module:

def text_button

```
def text_button(textname, mainscreen, screen_type, fontstyle, text, size, xpos, ypos)

    ''' This function changes the text's color and plays a sound effect when it is ho

    if textname.rect.collidepoint(mouse.get_pos()):
        sound("Click2.wav")
        txt = textsprite(fontstyle, text, size, xpos, ypos, 255, 0, 0)
        grouping = Group(txt)
        if screen_type == "Image":
            grouping.draw(mainscreen.display)
        if screen_type == "Fill":
            grouping.draw(mainscreen.screen)
    display.update()
```

Figure 13: def text_button

This function is used so that when a you hover on top of a text, it would turn the text's color into red and plays a sound effect while hovering. The function can be used for both backgrounds with color and image as shown in Figure 12. The code uses the textsprite class in order to work. The example is showcased in Figure 13 and Figure 14 on the next page.

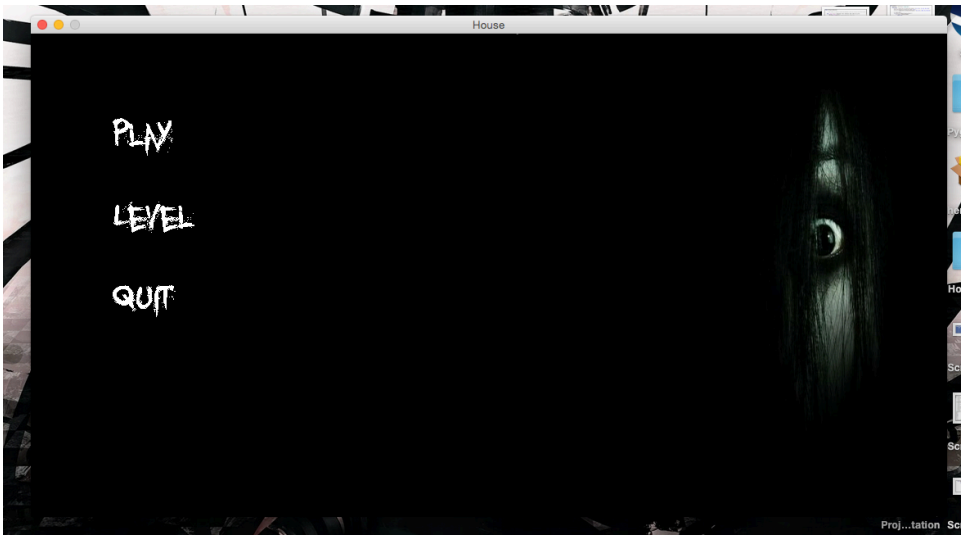

Figure 14: Before Hover

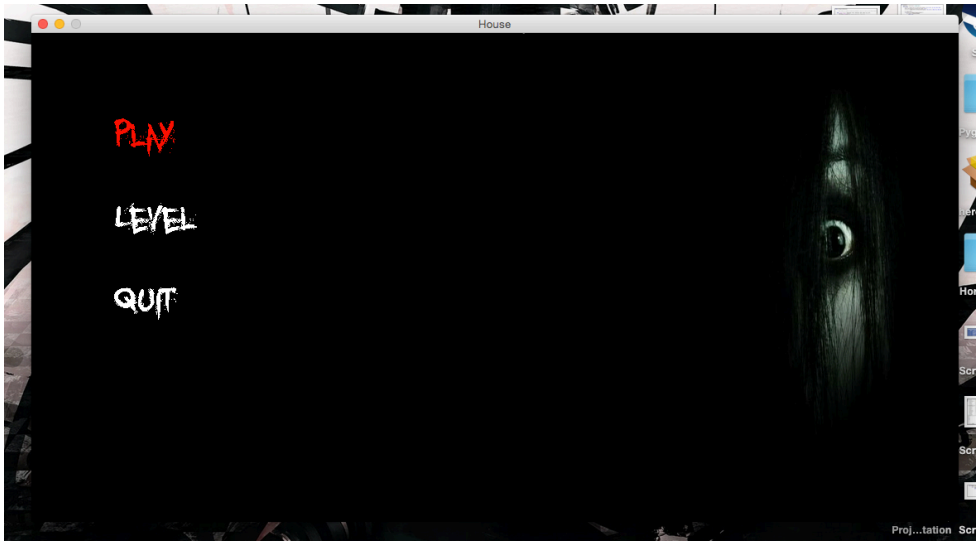

Figure 15: After Hover

def call_sprite

```
def call_sprite(text, screen_type, mainscreen):

    '''This function displays the sprite into the screen'''

    text_group = Group(text)
    if screen_type == "Image":
        text_group.draw(mainscreen.display)
    if screen_type == "Fill":
            text_group.draw(mainscreen.screen)
    display.update()
```

This function makes the sprite appear on the screen of the game. You fill in the parameters accordingly and the sprite (text and image) will appear on the screen. However, there are certain limitations to this function, as the function text_button would not change the text's color when using the call_sprite function.

Example of the call_sprite:

Figure 16: Picture Sprite Example

def Ghost()

```
def Ghost(screen_name, screen_display, image_call, xpos, ypos):

    '''This function calls the Jumpscare with the ghost and the sound'''

    screen_name= MainScreen(screen_display)
    jumpscare = imagesprite(image_call, xpos, ypos)
    pop = jump ()
    while True:
        ghost = Group(jumpscare)
        ghost.draw(screen_name.display)
        ghost_wait = event.wait()
        if ghost_wait.type == MOUSEBUTTONDOWN:
            break
        display.update()
```

Figure 17: def Ghost()

The Ghost() function calls the jump scare which plays the jump scare effect and shows the ghost's face in the middle of the screen. The sound defined by pop in Figure 16 uses the jump() class and is put before the while True loop in order to prevent the sound for playing more than once. Inside the while True loop, the ghost's sprite will be displayed on the screen and you have to click the mouse button for it to close. The parameter is set so that this function is reusable for every level of the game. The ghost sprite is show case below in Figure 17.

Figure 18: the Ghost Sprite

def reset() and def reset_kharon()

```
def reset():
    '''This function resets the values for the event triggers to it's original value'''
    global kharon, picturecount, door_unlock, key, scare, lives_remaining, bathroom_event
    kharon = []
    picturecount = 0
    door_unlock = 0
    key = 0
    lives_remaining = 3
    bathroom_event = 0

def reset_kharon():

    '''A reset specifically for the ouija board; resets value of kharon and scare'''

    global kharon, scare
    kharon = []
    scare = 0
```

Figure 19: def reset() and def reset_kharon()

The reset function is pretty straightforward; this function uses global so that when this function is called, it would 'reset' the value of the event triggers in game. For example, when you lost in a level, returning to the level you would still be able to play the level as the event trigger is reset to its original value. The 'reset value' is defined by the global value in Figure 18. The function reset_kharon() is specifically for the Ouija board. When you press 'Goodbye' in the Ouija board, it would reset the value back to zero so that you won't accidentally clear the level or accidentally fail the level as when the scare is equal to three, the player is transported to the game over screen.

def image_hover and def image_click

```
def image_hover(rect_name):

    '''Plays a sound effect when an image/text is hovered'''

    if rect_name.collidepoint(mouse.get_pos()):
        pygame.mixer.Sound("Click2.wav").play()

def image_click(rect_name, event):

    '''Plays a sound effect when an image/text is hovered'''

    if rect_name.collidepoint(mouse.get_pos()):
        if event.type == MOUSEBUTTONDOWN:
            pygame.mixer.Sound('Click.wav').play()
```

Figure 20: def image_hover() and def image_click()

These two functions, image_hover() and image_click() would play a sound effect every time a sprite or a rect is hovered or clicked. These functions are straightforward, you enter the desired parameters and it would play the sound effect defined in the function.

def transition_screen()

```
def transition_screen(text):

    '''Creates a transitional screen with black background and a text'''

    black = coloredbackground(0, 0, 0)
    transition_text = textsprite("YouMurderer BB", text, 100, 440, 170, 255, 255, 255)
    call_sprite(transition_text, "Fill", black)
    display.update()
```

Figure 21: def transition_screen()

This creates the transition screen for continuing to the next level or game over, this code creates the background color of black and the text will be filled in when called and used in another function such as Gameover(), which is shown below:

def Gameover() and def game_completed()

```
def Gameover():

    '''Creates the Game Over screen'''

    while True:
        transition_screen("Game Over")
        e = event.wait()
        if e.type == MOUSEBUTTONDOWN:
            reset()
            import PyHouseHorror; PyHouseHorror.menu()
        display.update()
```

Figure 22: def Gameover()

This function creates the Game over screen and as shown by Figure 21, it utilizes the transitional screen from the transition_screen() function. When you click the mouse button, it would transport you back to the menu screen. As showcased below in Figure 22. The function game_complete is identical, however showcases the text “Gam Completed” instead.

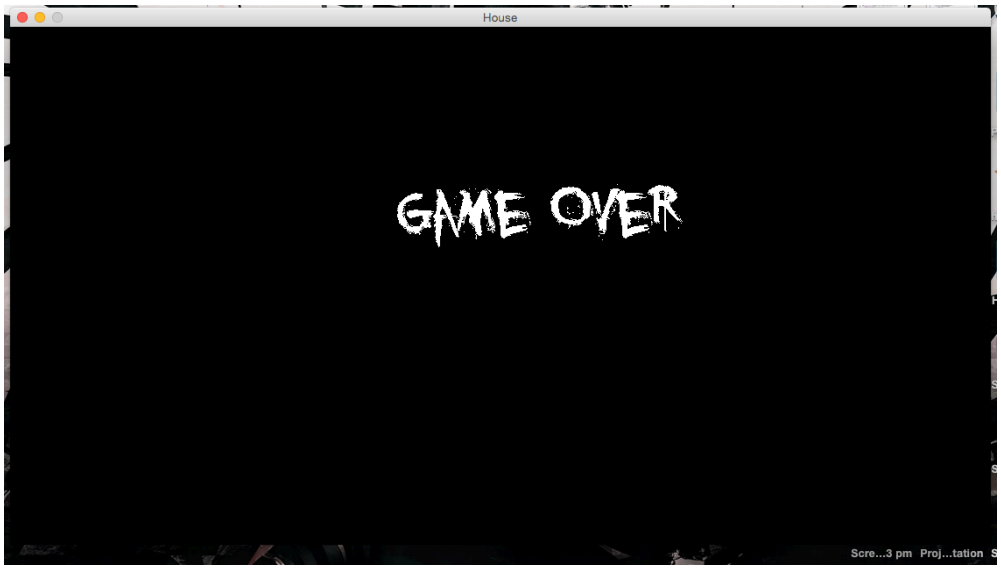

Figure 23: Gameover screen

Module – HouseHub.py

This module contains the function in which hosts the menu screen, level selection screen, introduction and the ending screen. This module uses the previous classes and functions as it’s foundation. It imports the previous two modules and the pygame module.

def menu()

```
def menu():

    '''Opens the menu screen containing, the features, play, levels and quit'''

    menuscreen = MainScreen("Menu.png")
    play = textsprite("YouMurderer BB", "Play", 60, 100, 100, 255, 255, 255)
    quit = textsprite("YouMurderer BB", "Quit", 60, 100, 300, 255, 255, 255)
    level = textsprite("YouMurderer BB", "Level", 60, 100, 200, 255, 255, 255)
    while True:

        #'''Lines 19 to 27 creates the text with rect features, and changes colors when hovered.'''

        play_text = Group(play)
        play_text.draw(menuscreen.display)
        text_button(play, menuscreen, "Image", "YouMurderer BB", "Play", 60, 100, 100)
        quit_text = Group(quit)
        quit_text.draw(menuscreen.display)
        text_button(quit, menuscreen, "Image", "YouMurderer BB", "Quit", 60, 100, 300)
        level_text = Group(level)
        level_text.draw(menuscreen.display)
        text_button(level, menuscreen, "Image", "YouMurderer BB", "Level", 60, 100, 200)
        menu_wait = event.wait()
        image_click(play.rect, menu_wait), image_click(quit.rect, menu_wait), image_click(level.rect, menu_wait)
        if play.rect.collidepoint(mouse.get_pos()):
            if menu_wait.type == MOUSEBUTTONDOWN:
                while True:
                    Ghost(menuscreen, "Menu.png", "Jumpscare.png", 0, 0)
                    intro()
        if quit.rect.collidepoint(mouse.get_pos()):
            if menu_wait.type == MOUSEBUTTONDOWN:
                pygame.quit()
                break
        if level.rect.collidepoint(mouse.get_pos()):
            if menu_wait.type == MOUSEBUTTONDOWN:
                levelscreen()
    display.update()
```

Figure 24: def menu()

This function creates the menu in which has three texts in the form of sprites in which when pressed, executes its proper function. For example the text ‘play’ would open the intro() function containing the intro, and while the text ‘quit’ would quit the game while the text ‘level’ would take the player into the level selection screen via the levelscreen() function. Each text would make sound when hovered and would make a difference sound when click as shown by the usage of the image_click and image. When calling the text however, I did not use the call_sprite() function as the text_button() function would not execute its function correctly and the red color will flicker instead of being a solid red color. Here is the picture of the menu screen:

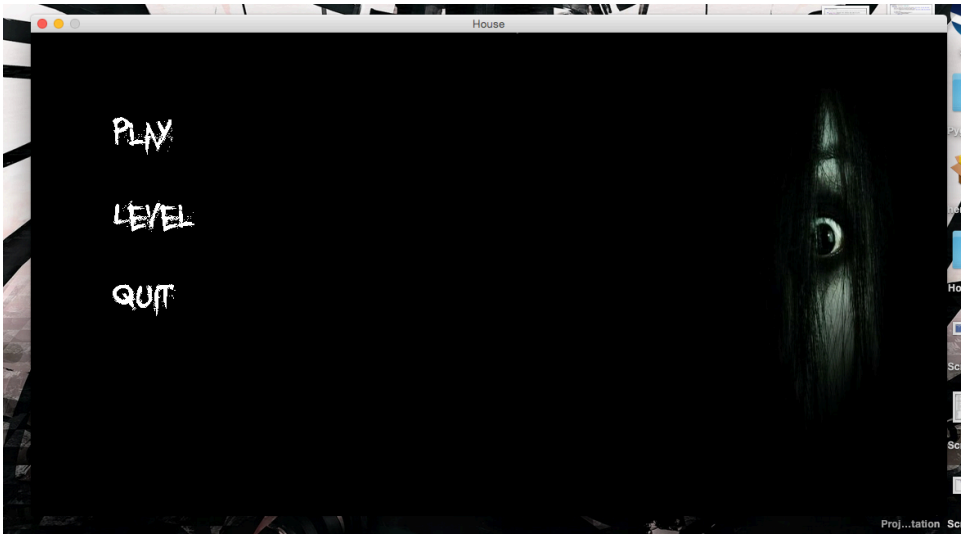

Figure 25: Menu screen

def levelscreen()

```
def levelscreen():

    '''Opens the level selection screen'''

    selection_screen = coloredbackground(0, 0, 0,)
    while True:
        level1 = textsprite("Diediedie", "Level 1", 50, 50, 100, 255, 255, 255)
        level1_text = Group(level1)
        level1_text.draw(selection_screen.screen)
        text_button(level1, selection_screen, "Fill", "Diediedie", "Level 1", 50, 50, 100)
        level2 = textsprite("Diediedie", "Level 2", 50, 350, 100, 255, 255, 255)
        level2_text = Group(level2)
        level2_text.draw(selection_screen.screen)
        text_button(level2, selection_screen, "Fill", "Diediedie", "Level 2", 50, 350, 100)
        back = textsprite("Diediedie", "Back", 25, 0, 0, 255, 255, 255)
        back_text = Group(back)
        back_text.draw(selection_screen.screen)
        text_button(back, selection_screen, "Fill", "Diediedie", "Back", 25, 0, 0)
        level_wait = event.wait()
        image_click(level1.rect, level_wait), image_click(level2.rect, level_wait), image_click(back.rect, level_wait)
        if level1.rect.collidepoint(mouse.get_pos()):
            if level_wait.type == MOUSEBUTTONDOWN:
                import Room_One; Room_One.transition_screen1()
        if level2.rect.collidepoint(mouse.get_pos()):
            if level_wait.type == MOUSEBUTTONDOWN:
                import Room_Two; Room_Two.transition_screen2()
        if back.rect.collidepoint(mouse.get_pos()):
            if level_wait.type == MOUSEBUTTONDOWN:
                menu()
        if level_wait.type == QUIT:
            pygame.quit()
            break
        display.update()
```

Figure 26: def levelscreen()

This function calls the level selection screen, in which you can select which level you want to play. It uses the textsprite class in a similar fashion as the one used in def menu(). Once a

level text is clicked, it would transport you into the selected level. There is also a back text which serves as a way to go back to the main menu via the menu() function. Same with the menu screen, when the text is hovered or clicked, it would play a sound effect. The level selection screen is shown in the next page in Figure 26.

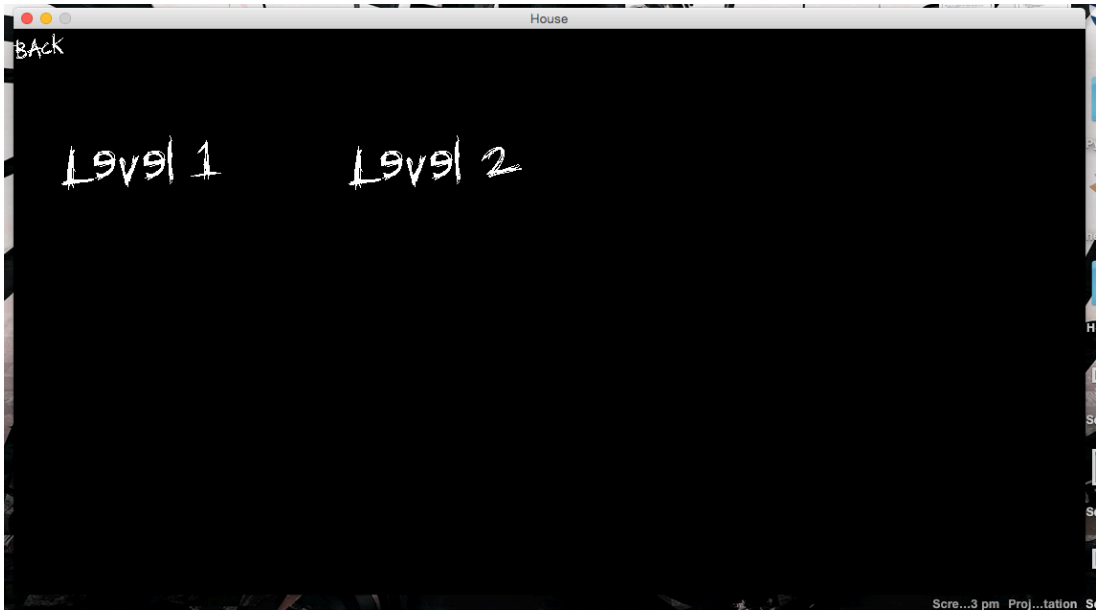

Figure 27: The level selection screen

def intro and outro()

```
def intro():

    '''Opens the introduction screen with the text containing the back story'''

    mainscreen = coloredbackground(0, 0, 0,)
    intro = textsprite("Diediedie", "This is your first night moving in into this board house", 30, 0, 0, 255, 255, 255)
    skip = textsprite("YouMurderer BB", "Skip", 35, 1000, 0, 255, 255, 255)
    while True:
        intro_text = Group(intro, skip)
        intro_text.draw(mainscreen.screen)
        intro_wait = event.wait()
        skip_button = text_button(skip, mainscreen, "Fill", "YouMurderer BB", "Skip", 35, 1000, 0)
        image_hover(intro.rect)
        image_click(intro.rect, intro_wait)
        if skip.rect.collidepoint(mouse.get_pos()):
            if intro_wait.type == MOUSEBUTTONDOWN:
                import Room_One; Room_One.transition_screen1()
        if intro_wait.type == MOUSEBUTTONDOWN:
            intro1 = textsprite("Diediedie", "The place has a history with the supernatural", 30, 0, 100, 255, 255, 255)
            call_sprite(intro1, "Fill", mainscreen)
            if intro_wait.type == MOUSEBUTTONDOWN:
                intro2 = textsprite("Diediedie", "It is said that the restless spirit still haunts this place to this day", 30, 0, 200, 255, 255, 255)
                call_sprite(intro2, "Fill", mainscreen)
                if intro_wait.type == MOUSEBUTTONDOWN:
                    intro3 = textsprite("Diediedie", "You woke up stressed and locked in your room", 30, 0, 300, 255, 255, 255)
                    call_sprite(intro3, "Fill", mainscreen)
                    if intro_wait.type == MOUSEBUTTONDOWN:
                        intro4 = textsprite("Diediedie", "If you want to live, escape the house.", 30, 0, 400, 255, 255, 255)
                        call_sprite(intro4, "Fill", mainscreen)
                        continue_text = textsprite("YouMurderer BB", "Continue", 35, 1000, 500, 255, 255, 255)
                        continue_call = Group(continue_text)
                        continue_call.draw(mainscreen.screen)
                        text_button(continue_text, mainscreen, "Fill", "YouMurderer BB", "Continue", 35, 1000, 500,)
                        if continue_text.rect.collidepoint(mouse.get_pos()):
                            if intro_wait.type == MOUSEBUTTONDOWN:
                                import Room_One; Room_One.transition_screen1()
        display.update()
```

Figure 28: def intro()

This function opens the intro screen in which the text displaying the introduction and background of the story will appear. Once you click on the screen, the text would appear for you to read. There is also a skip button which would take you straight to the first level via the imported transition_screen1() function. This function heavily uses textsprite button for the text to appear. Figure 28 in the following page will showcase the intro screen.

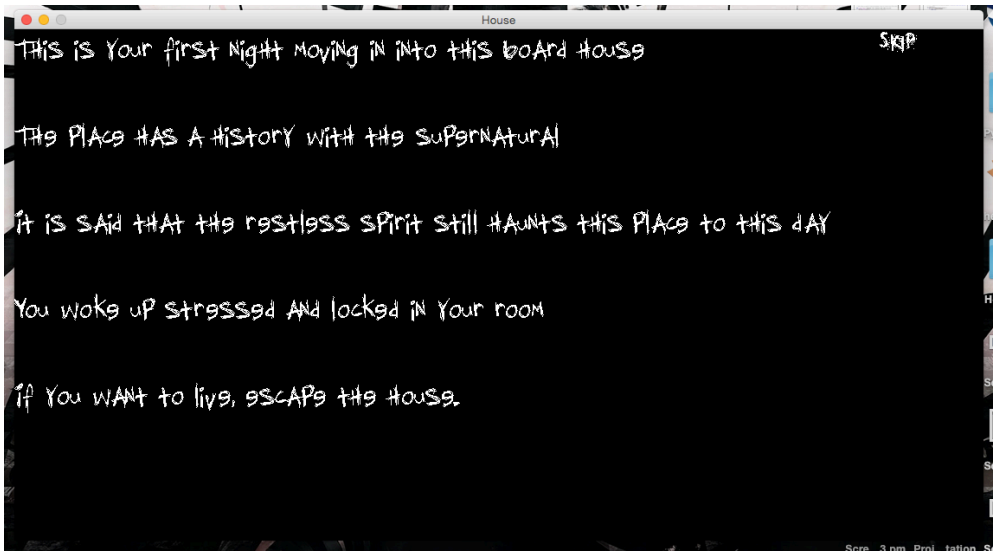

Figure 29: Intro screen

Def outro() behaves identically to the intro() function, however contains a different set of text and the skip button would transfer the player back to menu screen via the menu() function. The outro function() however is executed at the very last scene of the game.

def pause_quit()

```
def pause_quit(eventname, mainscreen):
    if eventname.type == KEYDOWN:
            if eventname.key == K_SPACE:
                while True:
                    pause = textsprite("Diediedie", "Paused", 50, 520, 295, 255, 255, 255)
                    menu_button = textsprite("Diediedie", "Menu", 50, 0, 0, 255, 255, 255)
                    menu_text = Group(menu_button)
                    menu_text.draw(mainscreen.display)
                    text_button(menu_button, mainscreen, "Image", "Diediedie", "Menu", 50, 0, 0)
                    call_sprite(pause, "Image", mainscreen)
                    bar = event.wait()
                    if menu_button.rect.collidepoint(mouse.get_pos()):
                        if bar.type == MOUSEBUTTONDOWN:
                            import HouseHub; HouseHub.menu()
                    if bar.type == KEYDOWN:
                        if bar.key == K_SPACE:
                            break

    if eventname.type == QUIT:
        pygame.quit()
```

This function opens pauses the game and also can close the game when the quit button is pressed. There is also an option to back to the menu screen.


```
def pause_quit(eventname, mainscreen):
    if eventname.type == KEYDOWN:
            if eventname.key == K_SPACE:
                while True:
                    pause = textsprite("Diediedie", "Paused", 50, 520, 295, 255, 255, 255)
                    menu_button = textsprite("Diediedie", "Menu", 50, 0, 0, 255, 255, 255)
                    menu_text = Group(menu_button)
                    menu_text.draw(mainscreen.display)
                    text_button(menu_button, mainscreen, "Image", "Diediedie", "Menu", 50, 0, 0)
                    call_sprite(pause, "Image", mainscreen)
                    bar = event.wait()
                    if menu_button.rect.collidepoint(mouse.get_pos()):
                        if bar.type == MOUSEBUTTONDOWN:
                            import HouseHub; HouseHub.menu()
                    if bar.type == KEYDOWN:
                        if bar.key == K_SPACE:
                            break

    if eventname.type == QUIT:
        pygame.quit()
```


Figure 30: def pause_quit()

Module – Room_One.py and Room_Two.py

This module contains the function of the entire first level. It uses import from the Classes, HouseFunctions and pygame module. Since the second function, screen1() is long, it would be divided and explained into different parts.

def transition_screen1() and def transition_screen2()

```
def transition_screen1():

    '''Opens the transition screen for the first Room'''

    music = sound("BackgroundMusic.wav")
    while True:
        transition_screen("Room 1")
        transition_wait = event.wait()
        display.update()
        if transition_wait.type == MOUSEBUTTONDOWN:
            screen1()
```

Figure 31: Transition screen code

This function is the start of every level, which uses the transition_screen() function to create a the transition screen. Here is an example of it below:

Figure 32: Room 1 Transition Screen()

def screen1() and def screen2()

Rectangle Placement

```
while True:

    picture = pygame.draw.rect(Bedroom.display, ((0, 0, 0)), (0, 150, 100, 100))
    picturecross = pygame.draw.rect(Bedroom.display, ((0, 0, 0)), (545, 170, 50, 90))
    picturedoor = pygame.draw.rect(Bedroom.display, ((0, 0, 0)), (870, 140, 200, 390))
    drawer = pygame.draw.rect(Bedroom.display, ((0, 0, 0)), (510 , 350, 65, 50))
    ouija = pygame.draw.rect(Bedroom.display, ((0, 0, 0)), (670 ,535, 40, 25))
    medicine = pygame.draw.rect(Bedroom.display, ((0, 0, 0)), (145 , 340, 90, 50))
    cupboard = pygame.draw.rect(Bedroom.display, ((0, 0, 0)), (115 , 385, 150, 160))
    grab_key = textsprite("Diediedie","Do you want to grab the? (Yes)", 45, 300, 100, 255, 255, 255)
    Bedroom.display.blit(Bedroom.image, (0,0))
```

Room 1 and Room 2 utilizes the point and click feature by clicking on the rectangle. However, the rectangle for these two screen are invisible, so it gives the illusion that when the user clicks on a component of the background, such as a picture, it would act as a rect and open it's features. The background screen has to be displayed last to make the rectangle drawn invisible. If it's not the last the rectangles would be visible shown in the example below.

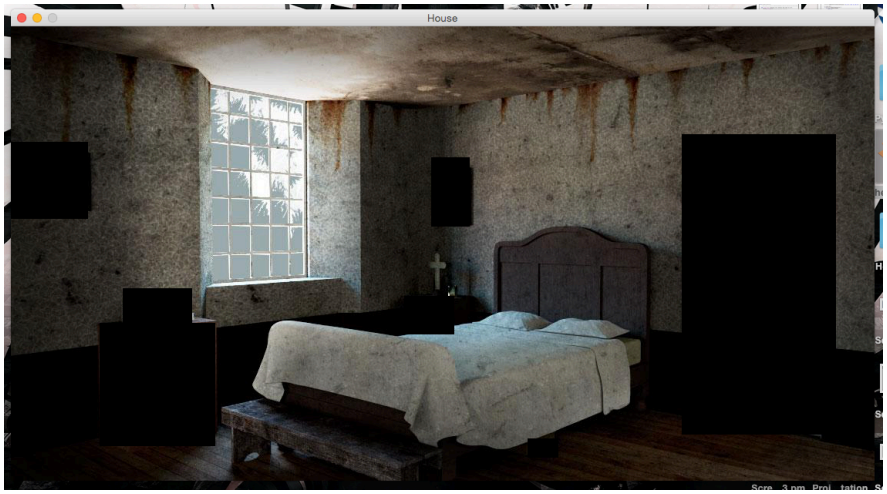

Figure 33: Rectangle Visible on the screen

However if the code "mainscreen.display.blit…" is put after the rectangle, it would become invisible.

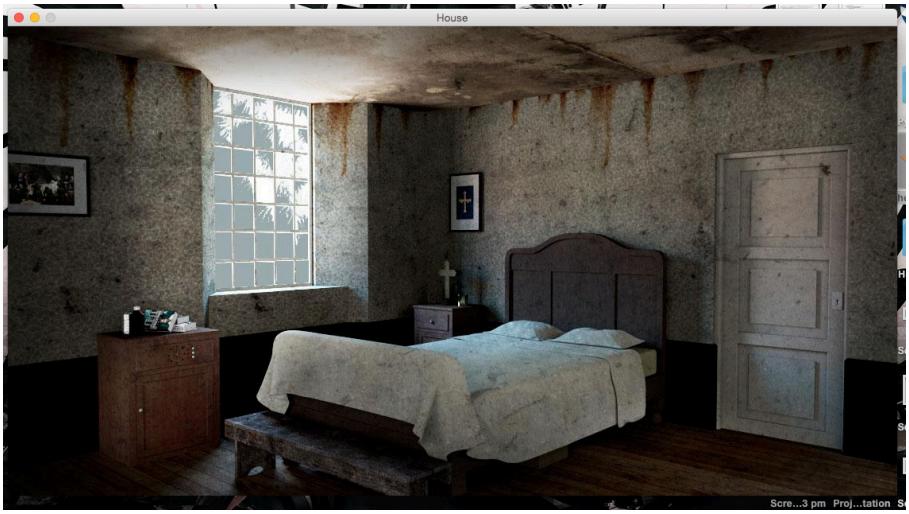

Figure 34: Rectangle invisible

The two levels utilizes the rect as the point and click as it has the mouse.get_pos() property and it can open events that are defined.

Clicking On The Invisible Rectangles

Clicking on the invisible rectangles would open events such as text or a new sprite.

```
if drawer.collidepoint(mouse.get_pos()):
    if key == 0 and picturecount < 3:
        if room1_wait.type == MOUSEBUTTONDOWN:
            while True:
                blocking_drawer = textsprite("Diediedie","Something is blocking the drawer", 45, 300, 400, 255, 255, 255)
                call_sprite(blocking_drawer, "Image", Bedroom)
                a = event.wait()
                if a.type == MOUSEBUTTONDOWN:
                    break
```

Figure 35: Clicking on the Text Code

For example, the code shown in Figure 35 would open up a text when the rectangle in place of the drawer is clicked on. The code drawer.collidepoint(mouse.get_pos()) is the argument the allows the player to click on the invisible rectangle. In Figure 36 the picture of the cross is clicked, and the text will appear based on the code shown in Figure 35. When you click on the mouse button again when the text is visible on the screen, the text will disappear. The "MOUSEBUTTONDOWN" argument creates a new while True loop in which the sprite would only appear if the loop is activated. To make the sprite disappear, the while loop breaks when the player presses "MOUSEBUTTONDOWN" again.

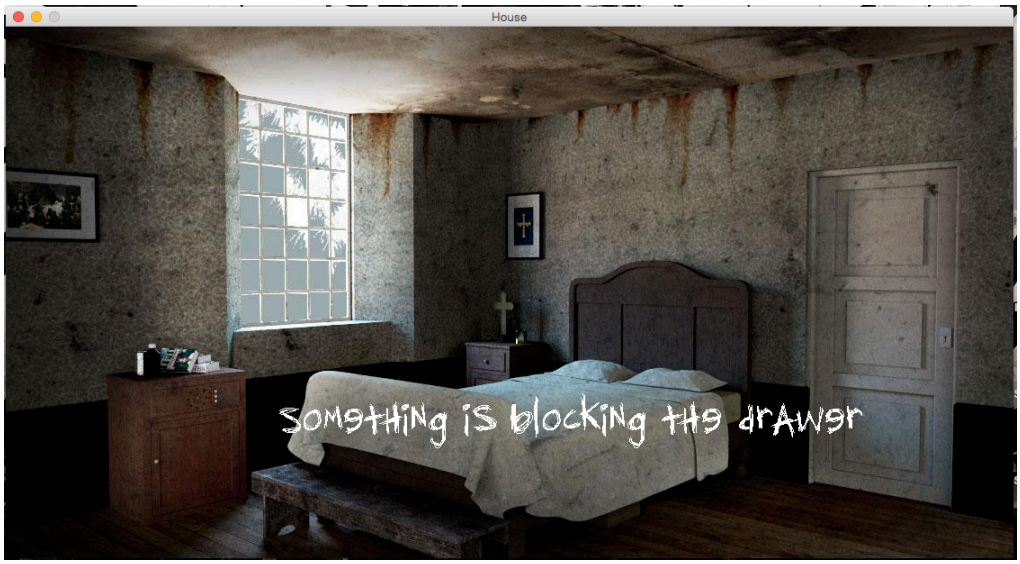

Figure 36: Text Appearing on Screen

Some events would not activate depending on the values of the "event triggers" defined by the global variables in the reset() function. If the value of key == 0 for example, the Ouija board would not open.

Room 1 – Obstacle – Ouija Board

The level starts to deviate in terms of the obstacle the player has to face. In the first room, the player has to enter a code in the Ouija board and hangman hybrid. The Ouija board heavily uses the textsprite function to call every individual letter so that it can use the collidepoint to detect the collision for each letter when clicked on. Some letters would increase the "scare" value in which when scare = 3, a jump scare would occur and the player would be greeted in the game over screen.

```
A = textsprite("Diediedie", "A", 50, 250, 275, 255, 255, 255)
B = textsprite("Diediedie", "B", 50, 300, 275, 255, 255, 255)
C = textsprite("Diediedie", "C", 50, 350, 275, 255, 255, 255)
D = textsprite("Diediedie", "D", 50, 400, 275, 255, 255, 255)
E = textsprite("Diediedie", "E", 50, 450, 275, 255, 255, 255)
F = textsprite("Diediedie", "F", 50, 500, 275, 255, 255, 255)
```

Figure 37: The code for defining each letter

As seen in Figure 37, the letters have to be individually defined using textsprite so that when highlighted, it can change color and when clicked on, it would either increase the "scare" value or the progress value "kharon".

```
if C.rect.collidepoint(mouse.get_pos()):
    C = textsprite("Diediedie", "C", 50, 350, 275, 255, 0, 0)
    if ouija_wait.type == MOUSEBUTTONDOWN:
        clicknoise = sound("Click.wav")
        scare += 1
else:
    C = textsprite("Diediedie", "C", 50, 350, 275, 255, 255, 255)
```

Figure 38: Example of code for each individual letter (except 'K', 'H', 'A', 'R', 'O' and 'N')

For each letter except for the ones mention in the caption for Figure 38, when the letter is hovered, it will change the text color to red, and when clicked on will play a sound effect and would add value to 'scare'. I did not use the text_button () function for this one as the hovering of each letters will create a significant time delay, and when clicked on there is also a delay in the command execution. To prevent further lag, I had to input the code manually for each individual letter.

```
if N.rect.collidepoint(mouse.get_pos()):
    N = textsprite("Diediedie", "N", 50, 250, 350, 255, 0, 0)
    if ouija_wait.type == MOUSEBUTTONDOWN:
        if "N" not in kharon:
            kharon.append("N")
        clicknoise = sound("Click.wav")
        N = textsprite("Diediedie", "N", 50, 650, 200, 255, 255, 255)
        text = Group(N)
        text.draw(Bedroom.display)
```

Figure 39: Code for letters 'K', 'H', 'A', 'R', 'O' and 'N'

When the letters specified in the caption in Figure 39 is clicked it would append its respective character into the empty list of kharon if the respective character is not in the list. If the respective character is already inside the list, it would not append the character, to prevent players from finishing the game by continuously clicking on when letter. After the respective character has been clicked on, the character will be displayed on top of the Ouija board, spelling out the code.

Figure 40: Ouija Board

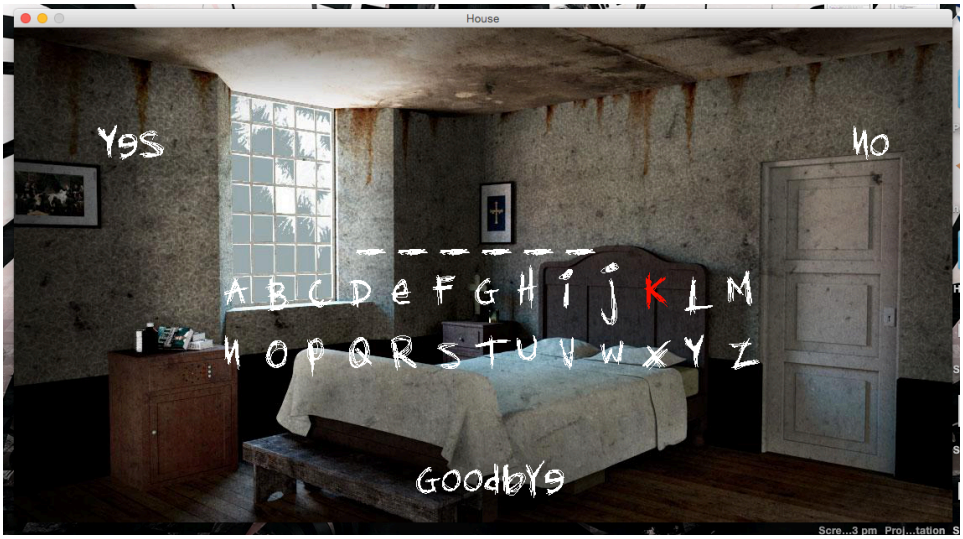

Figure 41: When a letter is hovered

Figure 42: When the characters "K", "H" and "R" is entered.

```
if goodbye.rect.collidepoint(mouse.get_pos()):
    goodbye = textsprite("Diediedie", "Goodbye", 50, 480, 500, 255, 0, 0)
    if ouija_wait.type == MOUSEBUTTONDOWN:
        reset_kharon()
        break
```

When the text “Goodbye” is clicked on, it would call the reset_kharon() function, resetting the values for kharon and scare back to its global value. The while True loop for the Ouija board will break, and take the user back to the level.

```
if "K" in kharon:
    if "H" in kharon:
        if "A" in kharon:
            if "R" in kharon:
                if "O" in kharon:
                    if "N" in kharon:
                        door_unlock += 1
                        unlock_sound = sound("Door Unlock.wav")
                        kharon_wait = event.wait()
                        if kharon_wait.type == MOUSEBUTTONDOWN:
                            while True:
                                warning_event = event.wait()
                                display.update()
                                pass
                        break
```

Figure 43: If statement for completion

If the respective letters is inside the list of kharon, the player would be able to go through the door and enter the next level as the value for door_unlock is increased by one. Originally the ‘if’ statement is only one line and uses “and” to combine all six letters. However, that statement only counts towards the last letter in the argument. So, when the letter “N” has been clicked in the Ouija board, the game automaticlly and open the door for the next level even though the other values in the list is not present. To prevent that problem, the if statements is followed by another if statement to make sure that each statement is passed through correctly so that the door_unlock value would not be added. With this, the order does not matter when clicking on the respective letters.

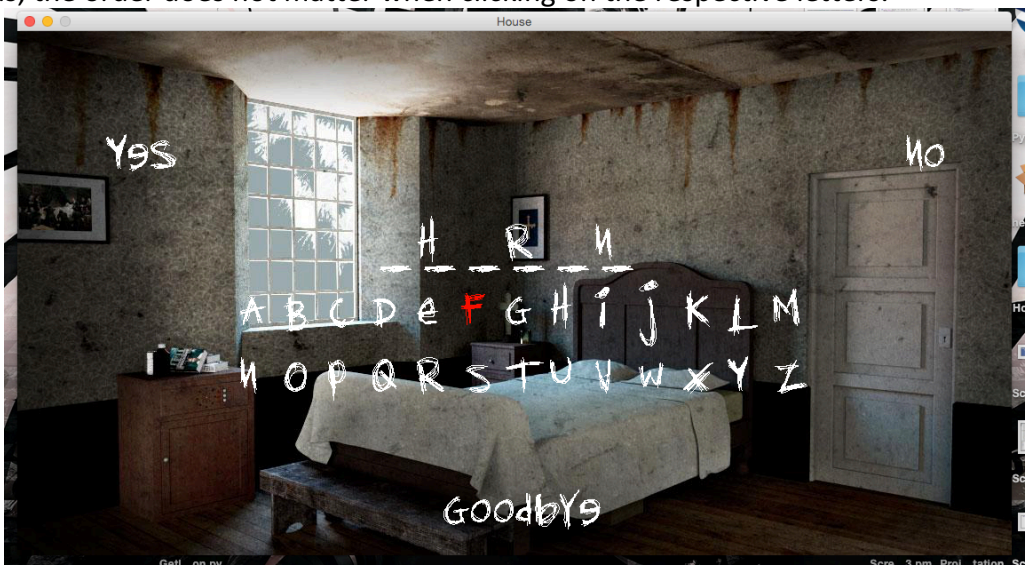

Figure 44: Showing that order does not matter

Room 2 – Obstacle - Puzzle

```
def Puzzle1():

    '''Opens the first screen of the puzzle'''

    music = sound("BackgroundMusic.wav")
    screen = pygame.display.set_mode((1120, 560))
    pygame.display.set_caption("There's no cure for loneliness")
    background_image = pygame.image.load("Bathroom.jpg")
    puzzle_pos = imagesprite('Position1.png', 299, 111)
    puzzle = imagesprite('puzzleholder.png', 300, 100)
    puzzle1 = imagesprite('puzzle1.png', randint(0, 800), randint(0, 300))
    while True:
        screen.blit(background_image,(0,0))
        puzzle_call = Group(puzzle1, puzzle)
        puzzle_call.draw(screen)
        puzzle_wait = event.wait()
        if puzzle1.rect.collidepoint(mouse.get_pos()):
            if puzzle_wait.type == MOUSEBUTTONDOWN:
                image_click(puzzle1.rect, puzzle_wait)
                while True:
                    for event1 in pygame.event.get():
                        if event1.type == pygame.QUIT:
                            pygame.quit()
                    mx,my = pygame.mouse.get_pos()
                    screen.blit(background_image,(0,0))
                    puzzle_call = Group(puzzle)
                    puzzle_call.draw(screen)
                    screen.blit(puzzle1.image, (mx-100,my-100))
                    pygame.display.flip()
                    move_wait = event.wait()
                    if puzzle_pos.rect.collidepoint(mouse.get_pos()):
                        if move_wait.type == MOUSEBUTTONDOWN:
                            image_click(puzzle_pos.rect, move_wait)
                            Puzzle2()
                    if move_wait.type == QUIT:
                        pygame.quit()
                        break
        display.update()
```

For the puzzle obstacle, you can only do one puzzle at a time as to prolong the game and to create suspense. The puzzle is randomly spawned in range of 0 to 800 horizontally and 0 to 300 vertically so that it would spawn in range of the screen. Each puzzle piece contains a different set of captions on top of the window screen in order to 'break the fourth wall' and tell the story in a different way. Each puzzle piece contains a different caption. When the puzzle sprite is clicked, the sprite that was clicked disappears and the position of the sprite is equal to the position of the mouse (mx and my). This means as the mouse moves, the puzzle also moves. Once the mouse clicks on the invisible rectangle of puzzle_pos, the screen would transition over the next screen via puzzle2() function. In this screen, the puzzle has been set correctly to its proper position, giving the player an indication that the puzzle has been set correctly and the next puzzle piece is given to solve.

Figure 45: The first piece of the puzzle

Figure 46: The second piece of the puzzle, with the first piece completed.

Figure 47: The entire puzzle completed

Room 2 – Lives, Hints and Events

The difference between the first room and the second room is that the latter contains three live points. When the player hits a rectangle that contains a jump scare, the value of their 'lives_remaining' will decrease by one point. If the value of lives_remaining is equal to zero, the player would be transported to the game over screen.

```
lives_text = textsprite("YouMurderer BB", "Lives: " + str(lives_remaining) + "/3", 50, 950, 0, 255, 0, 0,)
```

Figure 48: The text sprite that displays the amount of lives left

```
if mirror.collidepoint(mouse.get_pos()):
    if room2_wait.type == MOUSEBUTTONDOWN:
        Ghost(bathroom, 'Bathroom.jpg', 'Scare.png', 250, 0)
        lives_remaining -= 1
```

Figure 49: Example of jump scare 'trap'

As seen in Figure 47, if the player clicks on the mirror rectangle, their remaining lives would be deducted by one.

```
if lives_remaining == 0:
    Gameover()
```

Figure 50: Code for game over trigger

If the lives_remaning is equal to zero, the player would be transported to the game over screen. In addition, the position of the ghost changes depending on the value of the variable bathroom_event. This variable increases if the player progresses and clicks on the right rectangle and as the value increase, the position of the ghost changes.

```
if bathroom_event == 0:
    if vase1.collidepoint(mouse.get_pos()):
        if room2_wait.type == MOUSEBUTTONDOWN:
            lives_remaining -= 1
            Ghost(bathroom, 'Bathroom.jpg', 'Scare.png', 250, 0)
```

Figure 51: Bathroom event = 0

For example, in the example above, if the bathroom_event variable value is equal to zero, the vase acts as trap. However, if the bathroom_event == 3, the vase1 rectangle would not activate an event.

```
if showerhead.collidepoint(mouse.get_pos()):
    if room2_wait.type == MOUSEBUTTONDOWN:
        while True:
            pygame.display.set_caption("I didn't know that things would go this bad...")
            hint_wait = event.wait()
            if hint_wait.type == MOUSEBUTTONDOWN:
                pygame.display.set_caption("House")
                break
```

Some hints are given in the window's caption, in the example shown above, the caption will change when this object is clicked, and will change back to the original caption after it loop is broken when the mouse button is clicked.