University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

AVRET: Acrophobia Virtual Reality Exposure Therapy

A special problem in partial fulfillment

of the requirements for the degree of

Bachelor of Science in Computer Science

Submitted by:

Jethro Jake P. Matubis

June 2017

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "AVRET: Acrophobia Virtual Reality Exposure Therapy" prepared and submitted by Jethro Jake P. Matubis in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Marvin Ignacio, M.Sc. (*candidate*)
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (*candidate*)		
2. Avegail D. Carpio, M.Sc.		
3. Richard Bryann L. Chua, Ph.D. (*candidate*)		
4. Perlita E. Gasmen, M.Sc. (*candidate*)		
5. Ma. Sheila A. Magboo, M.Sc.		
6. Vincent Peter C. Magboo, M.D., M.Sc.		

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

Leonardo R. Estacio Jr., Ph.D.
Dean
College of Arts and Sciences

Abstract

Acrophobia, the fear of heights, is one of the most common forms of phobia. People with this condition are said to be induced with a feeling of restriction. In order to treat this, patients are usually subjected to cognitive behavior therapy. With the rise of virtual reality, a technique known as virtual reality exposure therapy emerged which can be used to treat patients with acrophobia. It has a number advantages over traditional methods while being able to produce the same results.

In this project, a system is proposed entitled, Acrophobia VRET, which aims to supplement therapists in treating patients with acrophobia.

Keywords: Virtual Reality Exposure Therapy, Acrophobia, Unity, Blender

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

Acrophobia, the fear of heights, is one of the most common forms of phobia. It is said that approximately five out of one-hundred people suffer from this condition [1]. Symptoms of this may range from having an accelerated heart rate, sweating and dizziness to experiencing hyperventilation and panic attacks. Acrophobic behavior typically involves the avoidance of a variety of height-related situations, including stairs, apartments, and offices located in high buildings, bridges, elevators and plane trips [2]. People with this condition are said to be induced with a feeling of restriction in his/her movements as compared to those suffering from other phobias [3].

In order to recover from acrophobia, the patient is conventionally subjected to in vivo exposure therapy, a form of cognitive behavior therapy (CBT). In CBT, the psychiatrist identifies the source of the patients fear and develops personal coping strategies to rehabilitate him/her. There are two kinds of in vivo exposure therapy: flooding and systematic desensitization. Flooding involves exposing the patient to the worst possible form of their phobia. Systematic desensitization, on the other hand, is where a patient is gradually exposed to the anxiety inducing object or event while engaging in a deep muscle relaxation technique taught by the therapist.

To further know about the process of treating patients with phobias, an interview was conducted at the psychiatry ward of the Philippine General Hospital [4]. It was done in the presence of the student's adviser, Sir Marvin John Ignacio, and the psychiatrist, Dra. Wayne Cortejos. The psychiatrist mentioned the standard procedure in conducting CBT. It begins with the therapist mapping out the main problems the patient is experiencing. Questions such as the date and scenario of the patient's first occurrence of the phobia are asked. These are asked with caution as to prevent the patients from retraumatizing themselves. In the proceeding sessions, therapist focuses on correcting the thoughts of the patient - making the patient see the situation from a different perspective. After the patient has recovered, a follow-up session is scheduled. The psychiatrist stressed that the time it takes to recover varies from patient to patient depending on his/her coping capabilities.

Although traditional methods such as flooding and systematic desensitization have proven to be effective, it is subjected to certain limitations. Access to these evidence-based treatments has been lacking due to the difficulty experienced by therapists in obtaining anxiety inducing stimuli when treating patients outside the clinic [5]. This has been found to be costly and time consuming. An example of this is in a study conducted by Emmelkamp et. al. where acrophobic patients who underwent in vivo exposure treatment were taken to a Magna Plaza and tasked to use its elevators in order to confront their phobia [6]. Parab et. al. states that the limitations of the real world bound the standard exposure therapy [7]. Patients with the fear of flying, for example, can only experience one takeoff and landing per flight. Such situations cannot be repeated due to environmental limitations. There is also a risk to the privacy of a patient when sessions are conducted in public places [7]. These limitations were then overcame with the introduction of virtual reality in exposure therapy.

Virtual reality (VR) is defined as an artificial environment that is created with software and presented to the user in such a way that the user suspends belief and accepts it as a real environment. [8]. It is primarily experienced through two of the five senses: sight and sound. The virtual reality environment provides appropriate responses in real time as the person explores its surroundings [9]. VR is now revolutionizing the field of medicine and healthcare in areas such as PTSD treatment, surgical management and exposure therapy [10]. The use of VR as a tool for therapists in exposure therapy has led to a process called virtual reality exposure therapy (VRET).

Recently, VRET has become a viable alternative in the treatment of fears and specific phobias including claustrophobia, acrophobia and arachnophobia [11]. In VRET, the natural tendency of patients to avoid the anxiety inducing event is bypassed by immersing users in simulations of the traumatic experiences which help the patient cope and overcome these [12]. VRET is completely controlled: the quality, intensity and frequency of the exposure are decided by the therapist, and can be stopped at any time if the patient does not tolerate it [13]. Additionally, the therapist and patient do not need to leave the consulting room. This implies a saving of time and money but also brings the benefit of not risking possible public embarrassment for the patient as well as preserving their confidentiality [14].

Emmelkamp et. al. aimed to evaluate the effectiveness of virtual reality exposure as compared to in vivo exposure in 33 patients suffering from acrophobia [6]. The patients were randomly assigned to each group and were required to fill-up questionnaires at pre-test, post-test, and follow-up. They were exposed to three different virtual environments which had been especially created for the project: a mall, a fire escape and a roof garden at top of a university building. The virtual environments used in treatment were exact copies of the real environments used in the exposure in vivo program. These were displayed through the Cybermind Visette Pro. Three sessions were conducted for both treatment conditions. These lasted approximately 1 hour and were held weekly. VR exposure was provided in a dark laboratory room at the Department of Clinical Psychology of the University of Amsterdam. Patients undertaking the VR exposure treatment were given breaks halfway each session in order to prevent motion sickness. Treatment activities during the in vivo exposure condition were held in the same three locations. Patients did activities such as using escalators, climbing stairs and walking on elevated heights while looking on ground level. They were allowed to be initially assisted by the therapist but were eventually let alone. An analysis of the results found VR exposure to be as effective as exposure in vivo on anxiety and avoidance. It was then stated that they now had considerable evidence that VR exposure is an effective treatment for patients with acrophobia.

Suarez et. al. developed a VRET system with a goal of designing a tool that can be used to treat intellectual disabilities on a general population diagnosed with acrophobia [15]. The system comprises of models designed with a software called Blender 3D 2.62. Blender is free open source 3D computer graphics software used to create films, visual effects, 3D applications or video games. The models were then exported to Unity 3D where all the needed scripts and functionalities are created for their virtual world. In the virtual environment, the patient will be situated outside the building where he/she is allowed to move freely inside and to look around. Once the patient enters the building, a panoramic elevator will be available for him/her to enter where he/she will confront the phobia. Suarez et. al. designed a 3D world wherein the patient will see plants, lights and landscape in a building through the panoramic elevator.

Dra. Cortejos mentioned that the inclusion of guided objectives in the VR environments for exposure therapy will be effective in rehabilitating patients. The absence of objectives will result to avoidance of the feared object or stressor. She also mentioned that it would be better for the patients to be exposed to more than one virtual environment [4].

B. Statement of the Problem

As stated, in the background, there are a variety of limitations in using traditional methods such as its difficulty to replicate due to the natural limitations of the environment. There is also a risk to the privacy of the patient as they are usually taken outside the clinic to be treated.

A problem with the VRET system of Suarez et. al. is that no explicit objectives were given for the patient to follow. Given the acrophobic patient's ability to move freely inside the virtual world and a natural tendency to avoid the stress-inducing stimuli, they are reluctant to enter the panoramic elevator which prevents them from being subjected to heights.

C. Objectives of the Study

This research aims to create a VRET system for acrophobia using an Android smartphone which has the following functionalities:

1. Allows the therapist to:
    - (a) Select among the following virtual environments:
        - i. Construction elevator
        - ii. Fire escape
        - iii. Glass bridge
    - (b) To view objectives accomplished by patient
2. Allows the patient to:
    - (a) Use the bluetooth controller to accomplish the following objectives in each virtual environment:
        - i. Construction elevator:
            - A. Enter the brick building

B. Find the keys

C. Find the crowbar in the room

D. Find the construction elevator

E. Enter the construction elevator

F. Turn on the construction elevator

G. Press the red button

ii. Fire escape:

A. Enter the brick building

B. Enter the elevator

C. Go to the 14th floor

D. Find the fire escape

E. Reach the 8th floor

F. Reach the 4th floor

G. Reach the bottom

H. Find the exit

iii. Glass bridge:

A. Enter the brick building

B. Enter the elevator

C. Go to the 13th floor

D. Enter the glass bridge

E. Save cat 1

F. Save cat 2

G. Save cat 3

H. Save cat 4

I. Return cats to parents

iv. Stop the simulation through an in-game button

v. To view objectives the patient has completed

D. Significance of the Project

This application provides virtual reality exposure therapy to patients suffering from acrophobia. The therapist is able to set the appropriate virtual environment based on his/her inferences on the patient. The application contains specific objectives for the patient to accomplish as well as tracks the progress of the patient through objectives in the virtual environment.

E. Scope and Limitations

1. This application is designed to treat Acrophobia and not other phobias.
2. This application will only contain a total of three virtual environments.
3. This application works on smartphones with minimum version of Android 4.4 (KitKat).
4. The brick building has a total of 14 floors.
5. The patient can only press the open button and 14th floor button in the elevator.
6. For the construction elevator virtual environment, the patient can only pick up the keys and the crowbar.
7. For the fire escape virtual environment, the patient must get the cats in the order of how they're placed in the bridge.
8. For the fire escape virtual environment, the patient cannot finish the level without saving the 4 cats.

F. Assumptions

1. The smartphone, controller and VR Headset will be provided by the therapist.

2. The smartphone needed to run the application must have a built-in accelerometer, magnetometer and gyroscope sensors.

3. The smartphone should be attached to a VR Headset.

4. The smartphone should be connected to a controller via bluetooth.

5. There is a therapist assisting the patient all throughout the therapy.

II. Review of Related Literature

In Sao Leopoldo, Brazil, Shunnaq et. al. proposed a model to assist the treatment of psychological disorders, enabling the use of techniques such as systematic desensitization, wherein a patient is gradually exposed to the object of fear that causes panic, anxiety, or flooding, in which the patient is exposed to the highest level of fear possible [16]. The tool makes use of the head-mounted display called Oculus Rift, a gadget usually used for entertainment. The pre-programmed phobias and their treatments are divided into modules enabling the option to add more levels. A total of 20 participants, aged 25 to 60 years old, was gathered to test the tool. After the participants were evaluated by the therapist, they were assigned the appropriate level according to the intensity of their phobia. They concluded that their model was effective and that virtual reality significantly affects human psychology, being that 80% of the participants were reported to experience anxiety.

Costa et. al. of the University Ontario Institute of Technology proposed the use of VR computer-assisted virtual environments (CAVE) in the treatment of acrophobia [17]. This application includes a physiological feedback mechanism to assess patient progress. To achieve a more authentic feeling, stereoscopic imaging was used to trick the user's brain into believing and seeing depth in the images. The CAVE consisted of four 2x2 meter walls, three Panasonic projectors, and a Vicon infrared tracking system. Through IR tracking markers, it was possible to track the user's position within the environment. It also makes use of the Neurosky Mindwave EEG which detects an increase in the user's relaxation levels to unlock additional levels consisting of buildings or other structures with greater height. However, it was replaced with a manual system of altitude unlocking due to a limited time frame. In the future, they plan to integrate computer operated fans that will blow air into the CAVE space to create an even more realistic experience for its users.

Leboucher et. al. assessed e-virtual reality in an acrophobic population [18]. Six individuals with acrophobia underwent six sessions of VRET. The first three underwent e-VRET sessions, which took place without any contact with hospital staff, while the last three underwent traditional sessions in the physical presence of a therapist (p-VRET). All sessions were conducted in a hospital to minimize expenses and further control the environment. A directional microphone was placed on the ceiling to provide communication between the therapist and patient, while a webcam allowed the therapist to observe the patient. To control the software remotely, TeamViewer 6 was used. No significant differences were found between the e-VRET and p-VRET sessions on anxiety, heart rate, presence, or therapeutic alliance. It was stated that the study was a simulation of what can be done over the Internet to which they concluded that e-virtual reality can be used to treat phobic disorders.

Schafer et. al. of the Heilbronn University in Germany, investigated the effect of additional body tracking with the use of avatars for its players [19]. The hardware selection used for the experiment was Oculus Rift, Microsoft Kinect for motion tracking, a Sony SixAxis controller, and a Polar Loop H7 Heart Rate Sensor. Two groups were formed with 21 persons each. In the first group, the subjects were led through a virtual height scenario with the avatar, while the second group experienced the same scenario without this feature. The first scene of the system allowed the user to become accustomed to the controls. The user was then led to a balcony where he/she will be asked to enter an unsecured section where one wrong step could lead to a deathly fall. After the procedure, participants then answered questionnaires to rate their overall experience. They concluded that through visual integration of a user's own body movement, the user feels more confirmed to act and navigate in the virtual reality like he is used to do in a real environment. They suggested that a simplification of distracting details can decrease the presence felt by the user.

Diemer et. al. tested whether a height challenge in VR elicited a comprehensive fear reaction in patients suffering from acrophobia compared to that of matched healthy controls [20]. The VR height scenario used was created with SourceEngine and presented via a head-mounted display. The participants were recruited through advertisements in local media, 80 in total. A number of measures were used to measure the patient's psychophysiological level such as heart rate, skin conductance level, and through the use of questionnaires. It was found that the physiological arousal of both groups did not differ in dangerous situations. They then concluded that a VR height scenario can activate subjective and physiological fear reactions which are determined by heart rate and skin conductance level in both acrophobic patients and healthy controls.

With cost-effectiveness in mind, Parab et. al. developed an application with the goal of helping patients overcome their phobias through the use of their developed virtual environment [7]. They chose three phobias to address: acrophobia, phasmophobia, and nyctophobia. Unity and Cardboard SDK were used to develop various virtual environments. An unspecified VR headset and Bluetooth controller was used to maneuver within the virtual environment in the android device, which displayed the virtual environment. The participants consisted of 50 students from the Atharva College of Engineering. All were supervised by a therapist and were asked to fill out a feedback form. An improvement to this system, as suggested by the users, was to add more levels and phobias to address.

Raghav et. al. aimed to address the efficacy of VRET in the treatment of long term trait dental anxiety [11]. Meeting the Phobia Checklist criteria of dental phobia, thirty participants, aged 18 to 50 years old, were divided into two groups: one that would receive VRET and another that would be assigned to the informational pamphlet (IP) control group. For VRET, a simulated dental environment was created, complete with a dental chair, overhead light and other dental instruments. There were five scenarios in which the user experienced, that ranges from patiently waiting for the dentist to arrive up to syringe injections. The IP group, on the other hand, were subjected to interventions. They were given pamphlets containing details about the standards of dental care and were given the chance to ask the researcher details about dental phobia. Anxiety was measured using the VAS-A, MDAS, and DFS survey. Participants then had scheduled follow-ups 1 week, 3 months, and 6 months after the treatment. However, the study was said to be a feasibility study. If proved successful, the VRET device produced can be rendered by any dental auxiliaries in a general dental practice which could make it a cost-effective solution.

Castro et. al. studied the efficacy of using virtual reality technique as compared to traditional cognitive behavioral treatment (CBT) in treating agoraphobia, the fear of crowded spaces or enclosed public places [21]. The two techniques were compared based on several clinical outcome measures and dropout rates. Only participants who have had agoraphobia for at least 5 years were recruited. The virtual environment consists of seven possible places that can produce phobic stimuli in agoraphobia patients that ranges from a square and a street to an airport building or an underground car park. Out of the 80 patients who were assigned different techniques, 37.5% left the study during the treatment. More than half of them belonged to the CBT group. Dropout rates increased at a six-month follow-up and the group with the highest rates was the CBT group. Both the traditional CBT and VRET were statistically effective, but the patients in the VRET group showed higher adherence rates, which implies a better therapy technique for chronic agoraphobia patients.

Gareth Walkom of the Nottingham Trent University conducted an experiment to see if VRET would decrease anxiety levels of people who stutter (PWS) [22]. This was done by gradually introducing them into a lecture theatre environment where they were to give a talk to an animated audience on a particular topic. Due to time constraints, however, only two sessions were conducted. A total of 6 participants was involved in VRET Session 1, 4 of whom also participated in VRET Session 2. Results were gathered throughout the VRET regarding anxiety level, body temperature, electrodermal activity (EDA), and comments related to the participants speech. An overall decrease in anxiety levels and improvement in speech was shown. Even though the results showed that the repeated sessions benefited the participants, it was stated that more testing should be conducted to receive even better results.

Miloff et. al. compared the effectiveness of VRET with the traditional one-session exposure therapy (OST) in which a patient is gradually exposed to the feared stimuli for up to 3 hours in a one-session format [23]. Participants, aged 18 and above, were recruited from the general public, screened, and randomly divided to either VR exposure therapy or traditional OST. The spiders used in the traditional OST session consisted of harmless varieties indigenous to the region and were classified according to size. The VR OST approach consisted of three zones with tasks for the user to complete: (1) looking at spiders; (2) interacting with spiders; and, (3) a task where the user is approached by a spider. As the user progresses, the more intense the spiders become. After the session, a 12 and 52 week follow-up was scheduled to each participant. In summary, it is stated that VRET may provide improvements in efficacy, access, standardization of protocols, and cost-effectiveness. It is also stated that the project will assist in the development of new methods for the delivery of evidence-based treatments.

III. Theoretical Framework

A. Acrophobia

Acrophobia, coming from the Greek word *akros* meaning ”at the end, topmost” and *phobos* meaning ”fear”, is an extreme or irrational fear of heights, especially when one is not particularly high up [24]. It is one of the most common forms of phobia with approximately one out of twenty people suffering from this condition with women twice as much as men [1]. The most widely accepted explanation is that acrophobia stems from a person's natural fear of falling and being injured or killed. The phobia occurs when fear is taken to an extreme, due possibly to unintentional learning, or the result of a traumatic experience. Typical symptoms of this include shortness of breath, rapid breathing, irregular heartbeat, sweating, and nausea when exposed to heights. There is also the possibility of a panic attack when the acrophobic is placed in a dangerous position. People with acrophobia typically avoid a variety of height-related situations, including stairs, apartments, and offices located in high buildings, bridges, elevators and plane trips [2]. They are said to be induced with a feeling of restriction in his/her movements as compared to those suffering from other forms of phobia [3].

Patients diagnosed with acrophobia may take medications as prescribed by the doctor. Three commonly used medications are beta blockers, anti-depressants and benzodiazepines [25]. Beta blockers are used for relieving performance anxiety by blocking the flow of adrenaline when a patient becomes anxious. Anti-depressants work by balancing neurotransmitters in the brain that affect mood and emotions which improves a patient's mood and concentration. Benzodiazepines are fast-acting anti-anxiety medications. However, they are sedating and addictive, so they are typically prescribed only when other medications have not worked.

Aside from taking medications, the patient may also undergo sessions supervised by a therapist. A well-established, highly effective, and lasting treatment method for treating patients with acrophobia is Cognitive Behavioral Therapy (CBT). It is focused on recognizing the negative thoughts and images associated with the fear of heights [26]. In this method, the therapist holds enclosed counseling sessions with the patient, and tries replacing the negative thoughts on the phobia with positive ones. The major purpose of this is to modify the fearful behavior associated with the phobia to a positive behavior by teaching different coping skills. Improvements are usually seen in 12 to 16 weeks, depending on the individual.

A well known form of CBT is In Vivo Exposure Therapy. In this method, the patient is guided by the therapist into a situation where they will face their fears and anxieties without performing any rituals or avoidance tactics they might normally use. There are two kinds of in vivo exposure therapy: flooding and systematic desensitization. Flooding involves exposing the patient to the worst possible form of their phobia. Acrophobic patients may be brought to a building rooftop or a cliff to accomplish this. Systematic desensitization, soon later known as graduated exposure therapy, was developed Joseph Wolpe in 1958 [27]. In this method, a patient is gradually exposed to the anxiety inducing object or event while engaging in a deep muscle relaxation technique taught by the therapist.

B. Virtual Reality

The definition of virtual reality comes from the definition of virtual meaning near or close to and reality is what we experience as human beings. In technical terms, virtual reality is used to describe a three-dimensional, computer generated environment which can be explored and interacted with by a person [9]. It is displayed with the use of VR headsets and are primarily experienced through two of the five senses: sight and sound. In the case of headsets such as the HTC Vive and Oculus Rift, video is sent from the console or computer to the headset via a HDMI cable. For headsets that make use of smartphones (e.g. Google Cardboard and Samsung Gear VR), the video is displayed through the smartphone's screen. Internal components which are used for its head-tracking system are the gyroscope, accelerometer and a magnetometer. VR headsets also contain lenses placed between the user's eyes and the display. These focus and reshape the picture to create a stereoscopic 3D image by angling the two 2D images to mimic how each eye views the world. Overall, the virtual reality environment provides appropriate responses in real time as the person explores its surroundings.

C. Mobile Application

An application software designed to run on an Android-based smartphones

C..1 Google VR SDK

Google VR SDK allows virtual reality applications for Android and iOS to be built. This is integrated to Unity to provide it additional features such as spatialized audio, Daydream controller support, utilities and samples. A minimum version of Unity 5.2.1 is required to install this.

C..2 Google Cardboard

Google Cardboard is a virtual reality (VR) platform developed by Google for use with a head mount for a smartphone. It is made of low-cost viewers, with the reference design made of foldable cardboard 45mm plastic lenses, and a magnet or capacitive-taped lever to operate the screen. To use the platform, users run Cardboard-compatible applications on their phone, place the phone into the back of the viewer, and view content through the lenses.

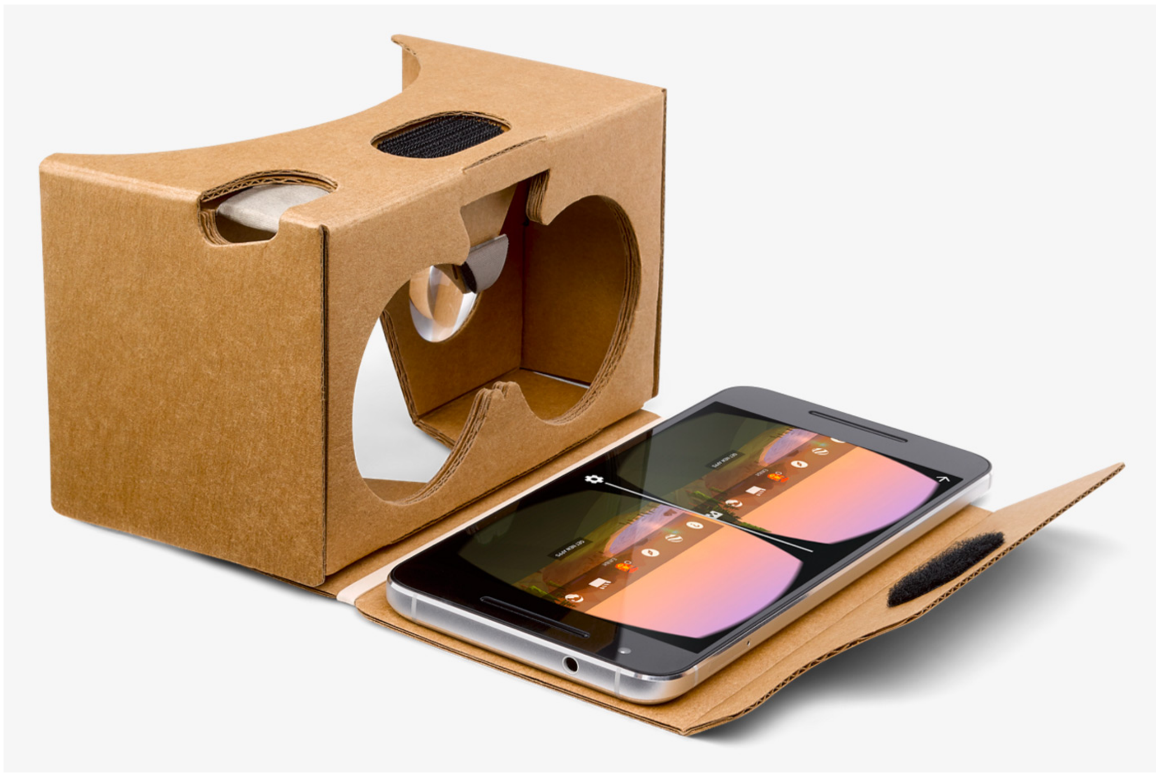


Figure 1: Google Cardboard

C..3 Android

Android is an operating system for smartphones created by Google that is based on Linux. Android is open-source and has a large community of developers extending various functionalities, written primarily in modified Java programming language.

D. Unity

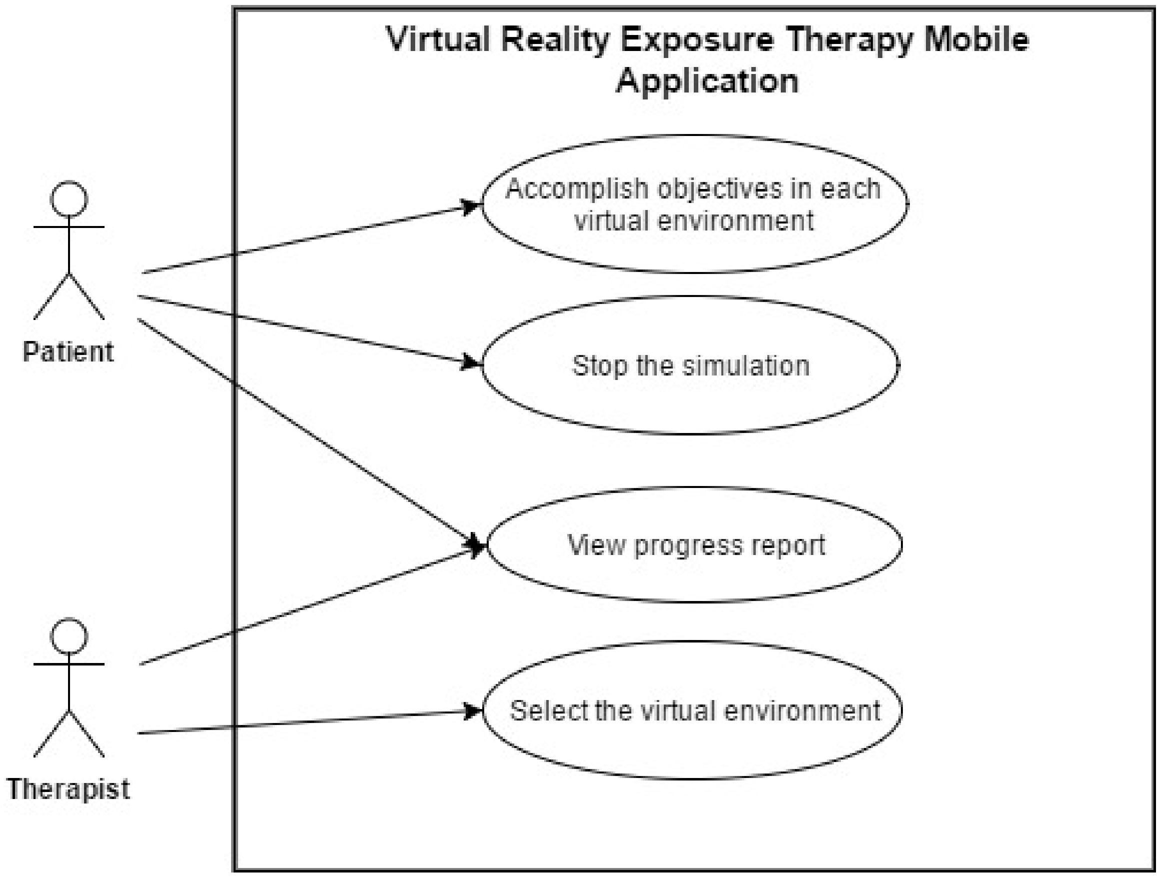
Unity is a game development platform. It is used to build high-quality 3D and 2D games that can be deployed across mobiles, desktops, virtual reality, augmented reality, consoles, or even the Web. It uses the C# programming language in making different scripts.

E. Blender

Blender is a free and open source 3D creation application. It supports the entirety of 3D pipeline: modeling, rigging, animation, simulation, rendering, compositing, motion tracking, video editing, and game creation [28]. Blender is cross-platform and runs equally well on Linux, Windows and Macintosh computers. It uses OpenGL as its interface.

IV. Design and Implementation

A. Use Case Diagram



Virtual Reality Exposure Therapy Mobile Application
Accomplish objectives in each virtual environment
Stop the simulation
View progress report
Select the virtual environment
Patient
Therapist

Figure 2: Use Case Diagram: Overview

The mobile application has two users: the patient and the therapist.

The patient can move inside the virtual environment and accomplish its objects.

If the patient cannot handle the anxiety, he/she can stop the simulation.

The therapist is the one who selects the virtual environment for the patient.

The therapist can stop the simulation anytime.

The therapist can also view the progress report of the simulation.

B. Context Case Diagram

There are two major entities that interact with the system: the therapist and the patient. The therapist selects the virtual environment. He is also able to receive a report containing the progress of the patient. The patient interacts with the virtual environment selected by the therapist, generating progression data to be viewed by the therapist.

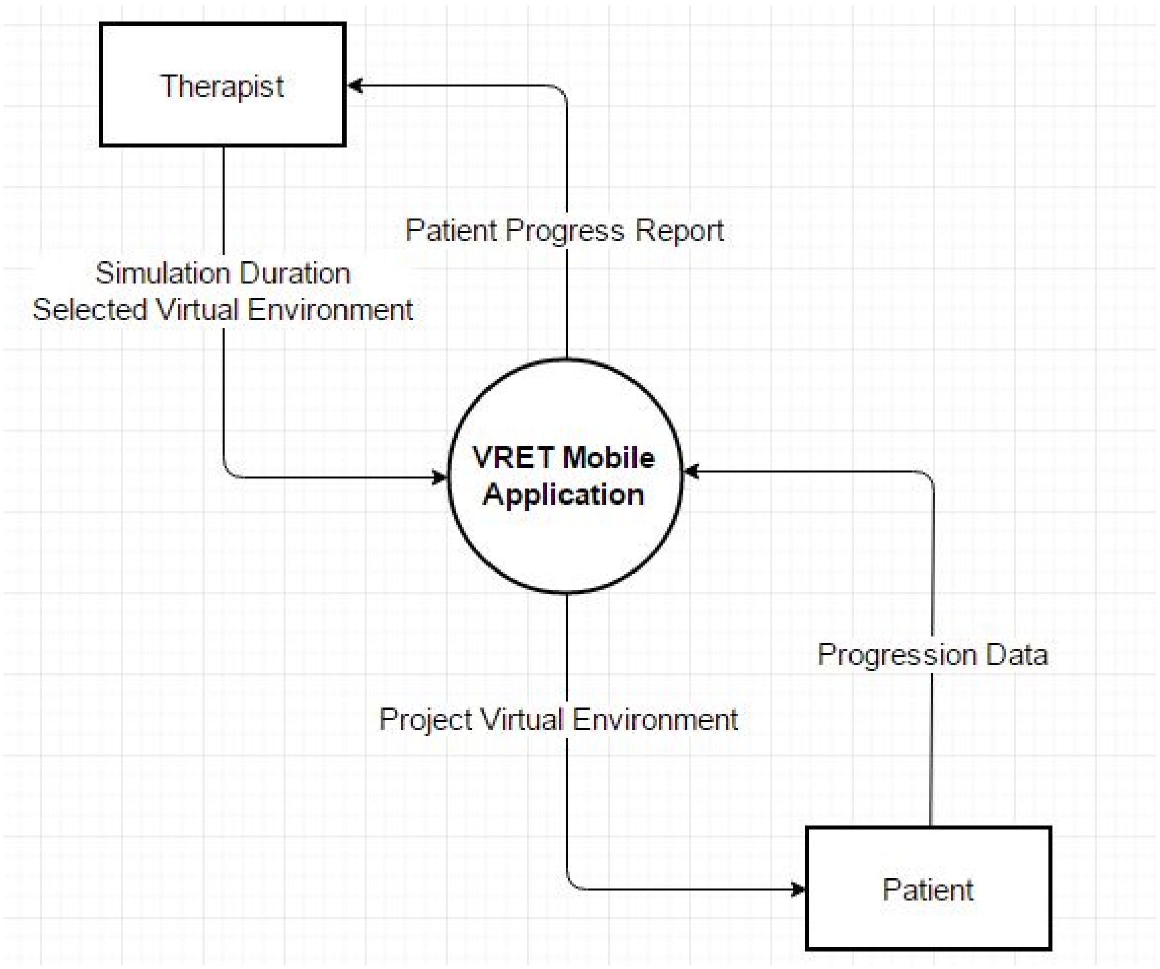
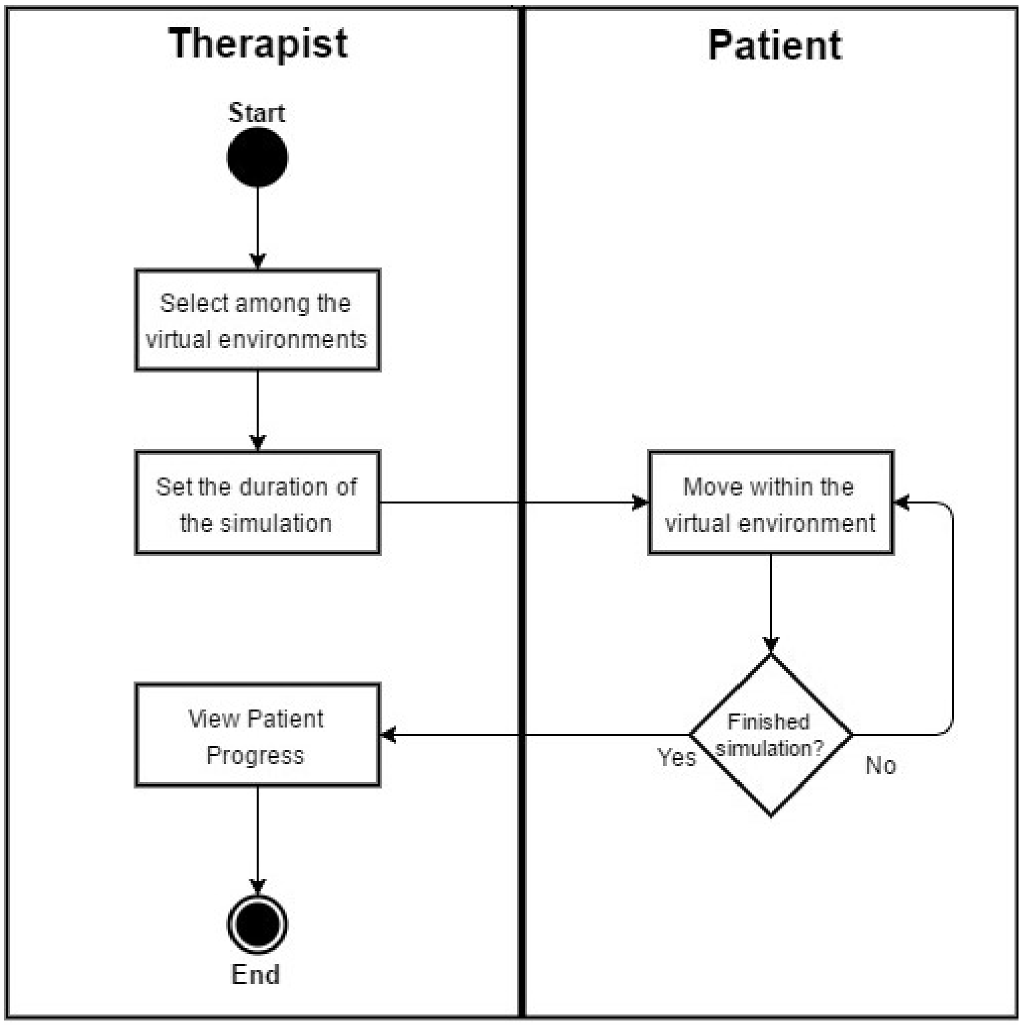


Figure 3: Context Case Diagram: Overview

C. Activity Diagram

The therapist configures the application by selecting among the three virtual environments. The patient is then able to dwell in the environment with actions depending on the virtual environment selected by the therapist. The therapist will then be able to view the progress of the patient whether the patient succeeds or not.



Therapist
Patient
Start
Select among the virtual environments
Set the duration of the simulation
Move within the virtual environment
Finished simulation?
Yes
No
View Patient Progress
End

Figure 4: Activity Diagram: Overview

D. Flowchart

The following flowcharts depict the activities the patient can do in each virtual environment:

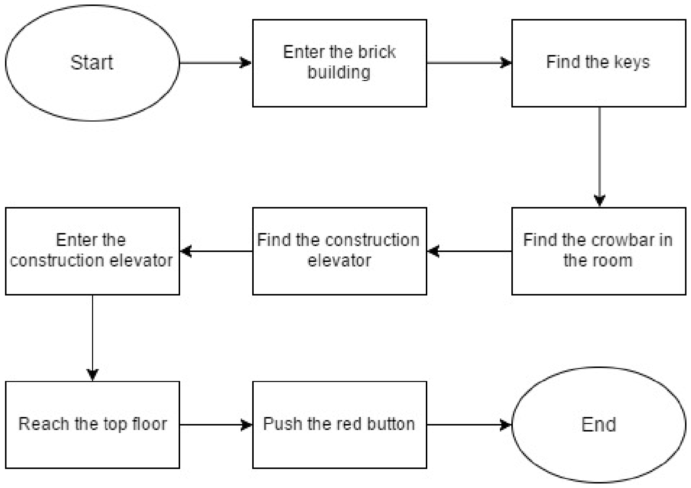

Start
Enter the brick building
Find the keys
Find the crowbar in the room
Find the construction elevator
Enter the construction elevator
Reach the top floor
Push the red button
End

Figure 5: Flowchart of the patient in the construction elevator

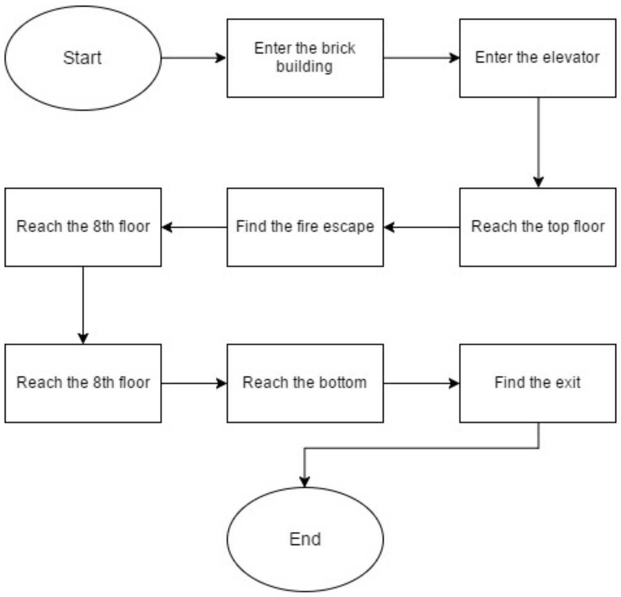


Figure 6: Flowchart of the patient in the fire escape

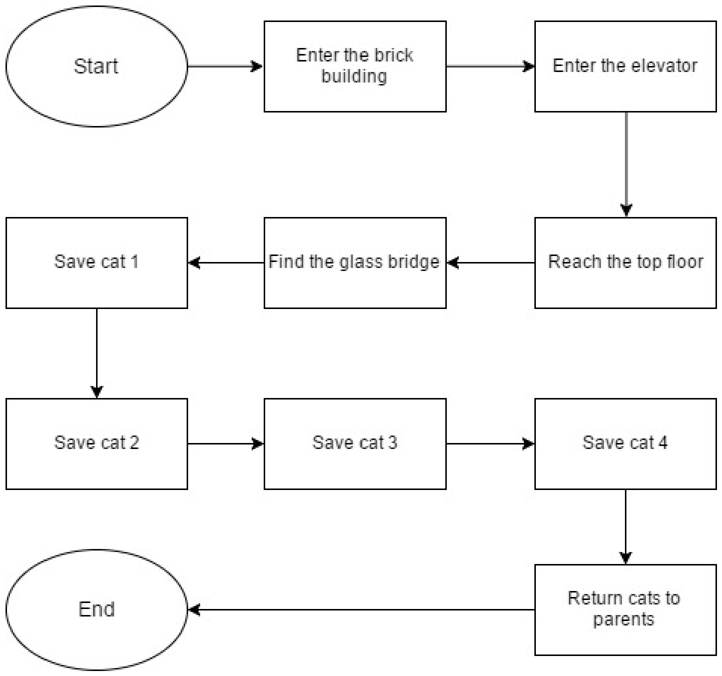


Figure 7: Flowchart of the patient in the glass bridge

E. Technical Architecture

The smartphone that will used to run the mobile application requires the following:

1. Android OS version 4.4 (KitKat) or higher
2. Accelerometer
3. Gyroscope
4. Magnetometer

V. Results

Acrophobia VRET is a mobile application that exposes patients to three virtual environment: construction elevator, glass bridge and fire escape. The application produces a progress report that shows the objectives done by the patient and the time it took to complete each objective.

A. Main Menu

When the application is started, the main menu will show.

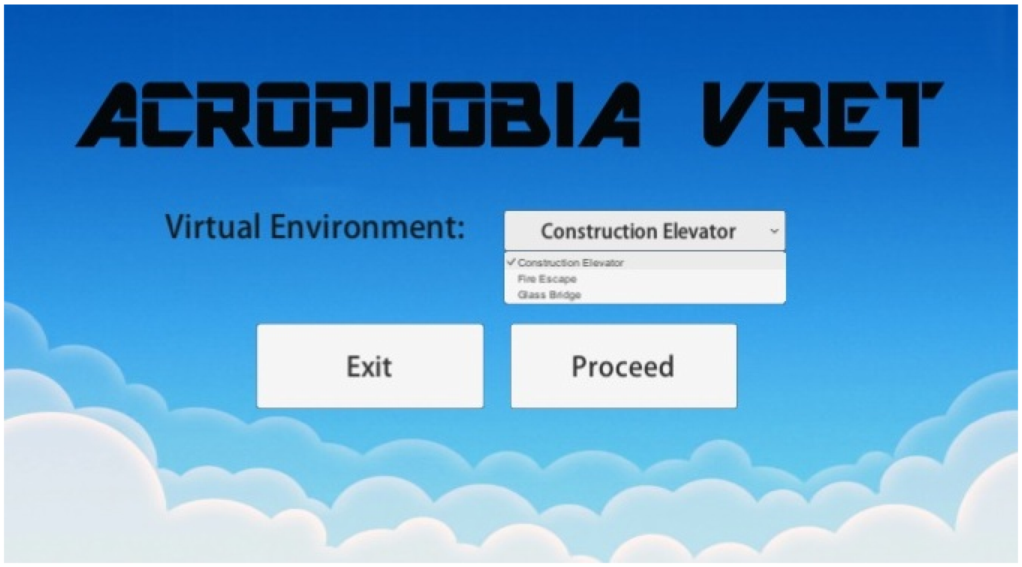


Figure 8: Main Menu

B. Virtual Environment

Upon clicking the proceed button, the selected virtual environment will load. The objectives can be seen at the top of the camera screen. The objective displayed will change upon completing it.

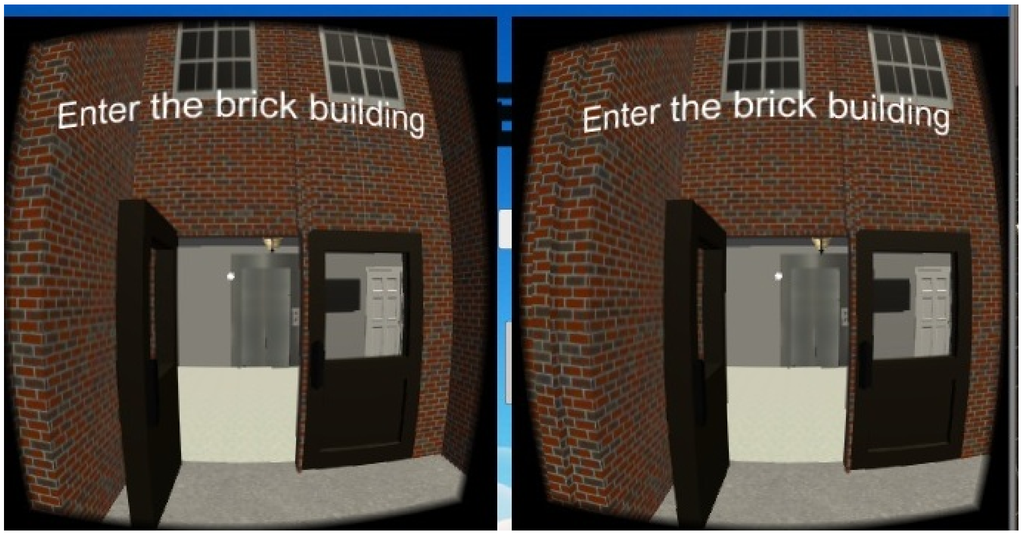


Figure 9: Construction Elevator Sample Scene 1

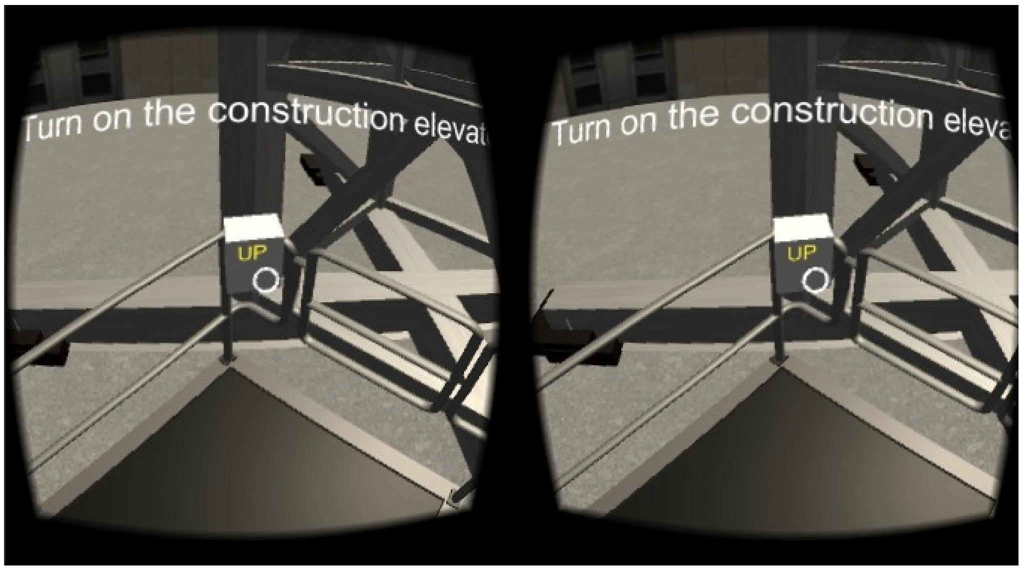

Turn on the construction elevat
Turn on the construction eleva
UP
UP

Figure 10: Construction Elevator Sample Scene 2

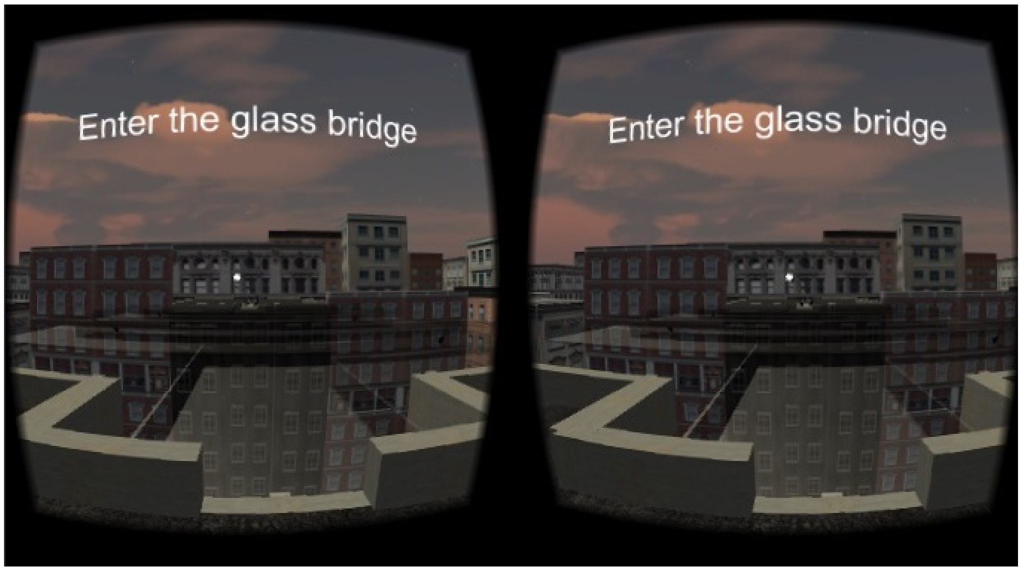


Figure 11: Glass Bridge Sample Scene 1

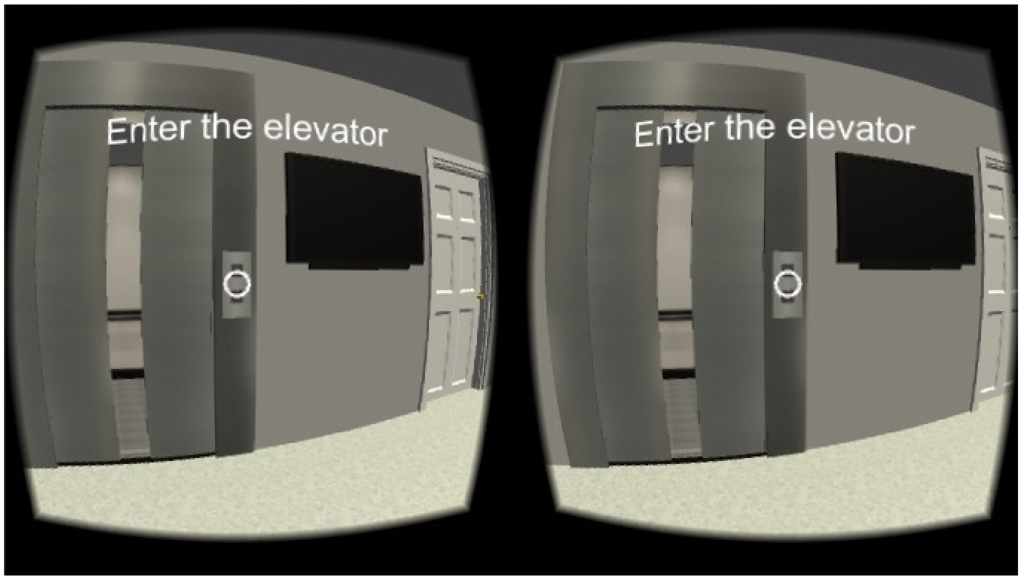


Figure 12: Glass Bridge Sample Scene 2

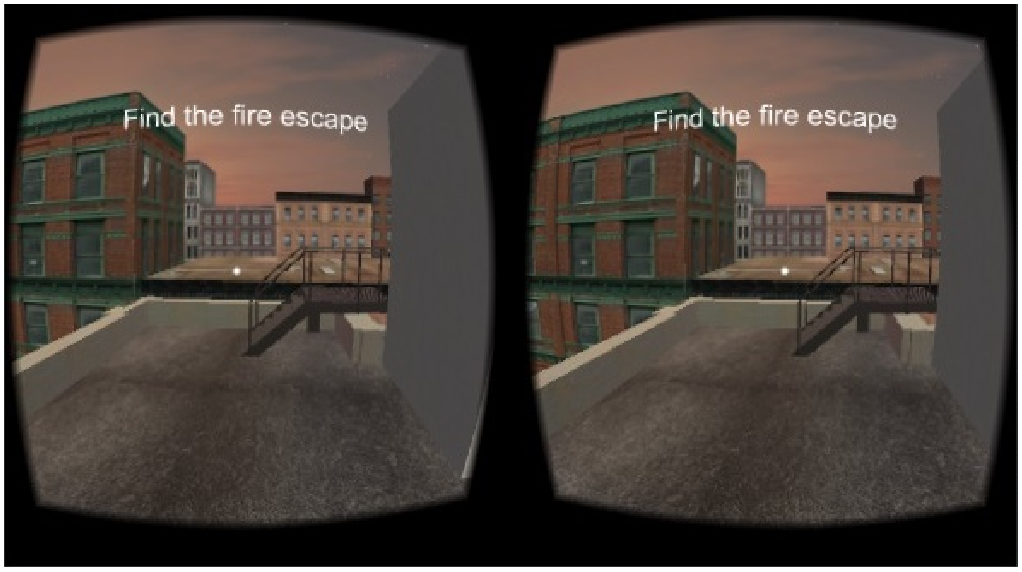


Figure 13: Fire Escape Sample Scene 1

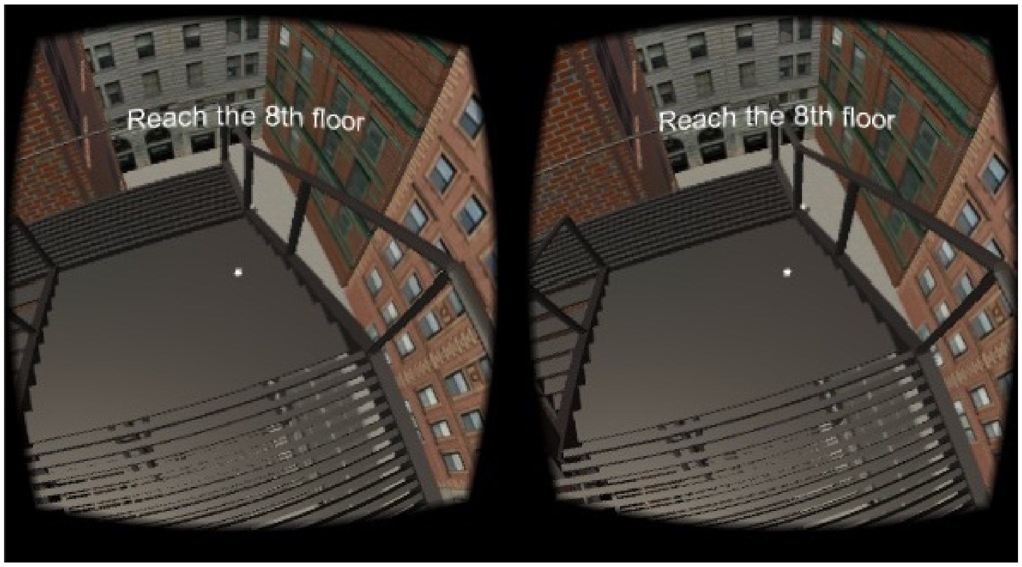


Figure 14: Fire Escape Sample Scene 2

C. Stop Button

The patient can stop the simulation when he/she is in panic.

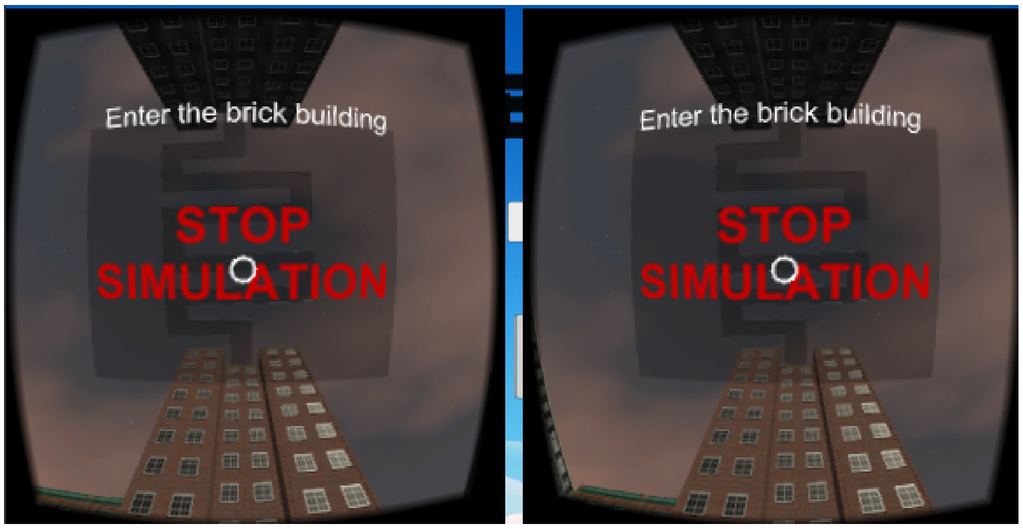


Figure 15: Stop Button

D. Progress Report

Once the patient has finished the virtual environment or has pressed the stop button, a list of objectives done will be presented.

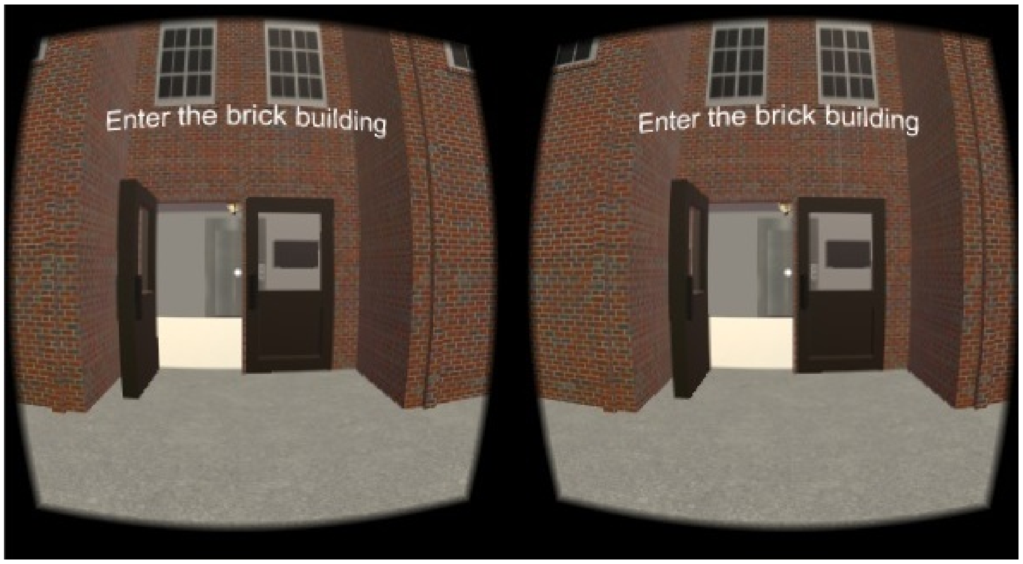
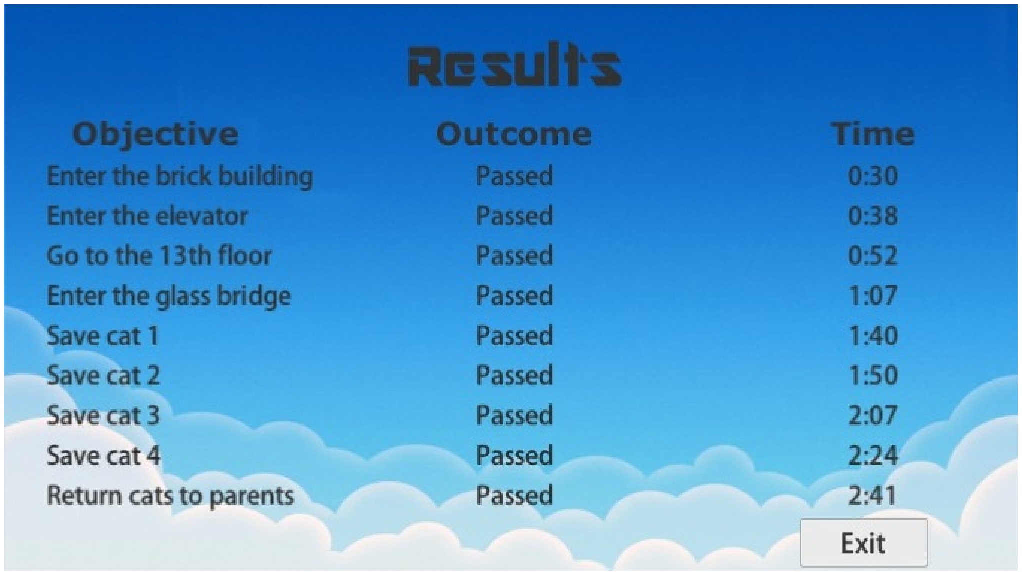


Figure 16: Progress Report Sample 1



Results

Objective	Outcome	Time
Enter the brick building	Passed	0:30
Enter the elevator	Passed	0:38
Go to the 13th floor	Passed	0:52
Enter the glass bridge	Passed	1:07
Save cat 1	Passed	1:40
Save cat 2	Passed	1:50
Save cat 3	Passed	2:07
Save cat 4	Passed	2:24
Return cats to parents	Passed	2:41

Figure 17: Progress Report Sample 2

VI. Discussions

Acrophobia VRET is a program designed to be a supplementary tool for therapists in treating patients with acrophobia. It makes use of an Android smartphone equipped with an accelerometer, magnetometer and a gyroscope to run its virtual environments. In order for the patient to be fully immersed in the environment, he/she must be wearing a VR headset to which the Android phone is attached to. Example of these VR headsets include VR Box and Google Cardboard.

Two programs were used in the development of this application: Blender and Unity. Blender is a free and open source 3D creation application. It is through Blender that the brick building model was created, along with its interiors which includes the elevator and the furniture. Unity, on the other hand, is a game development platform. It is here where the created models were added and integrated with scripts that allows for player movement, level selection and interaction with other objects. Unity also features an asset store which contains a list of free models, scripts, scenes and many more. This is where the street elements and the other remaining buildings were downloaded.

The application is built with the following functionalities: a selection system for the therapist to choose a virtual environment, the simulation of said environments and a progress reporting system that can be viewed by both the patient and therapist.

The first functionality was developed using Unity's UI panel. In this panel, the developer is given the option to create buttons, dropdown lists and text which can serve as labels or titles. The developer is also allowed to set pictures to serve as the menu's wallpaper and is also able to add background music. Being the first scene to be viewed by the user when the application opens, it is necessary to set the index of the menu scene to 0 in the build settings of the application. The dropdown contains the names of each virtual environment as choices and are used by a script to load the player to the chosen scene after pressing the proceed button.

The second functionality is where the player is tasked to accomplish a specific set of objectives which vary depending on the virtual environment selected. These objectives were made using a combination of invisible box colliders and targetable objects that triggers the next objective on contact or on click by the player. Unity does not allow virtual reality support by default. In order to enable this, a package must be downloaded named, Google VR sdk or Cardboard sdk. The former being the latest version while the latter being an old version. Once the developer has installed the package, he/she will be able to make use of the VR camera contained in the package. To enable player movement, a script was created that moves the player towards the direction he/she is facing. In order for the player to target an object, a reticle, which is also contained in the package, must be dragged to the vr camera. An event system is then added to the object to be made targetable. If the patient cannot handle the anxiety generated by being exposed to the phobia, a stop button was created positioned above him/her. This will end the simulation and load the progress report.

The last functionality is the progress report. This shows what the patient has and hasn't completed during the simulation. Objectives that are completed are marked as 'done' while objectives that aren't are marked as 'not done'. The progress report also includes the time the patient was able to complete the objective. This can be used by the therapist to check for improvements in the patient. The data displayed in the timer is taken from a script in each of the virtual environments where completing an objective there will save the current time. The progress report simply takes the recorded time and displays it.

VII. Conclusions

Acrophobia VRET is an application that provides virtual reality exposure therapy to patients suffering from acrophobia. The therapist is able to set the appropriate virtual environment based on his/her inferences on the patient. Each virtual environment contains specific objectives for the patient to accomplish. A progress report will be displayed for for both the therapist and the patient to view.

This application was developed with two programs: Blender for modeling and Unity for scripting. It was created for the benefit of the therapist and the patient.

For the patient, this application will give them the benefit of privacy. Acrophobic stimulants are usually located in public places. With the help of this application, the patient will not need to have the risk of being exposed to the public. Objectives will be displayed within the virtual environment as to serve as a guide for the patient. He/she will also be given an option to stop the simulation in case he/she cannot handle the stress.

For the therapist, this application will serve as a supplementary tool that he/she can use to save time and money. The therapist, along with the patient will not have to leave the clinic in order to obtain the anxiety inducing stimuli: heights. From the comfort of the clinic, the therapist will be able to choose 3 virtual environment: construction elevator, fire escape and a glass bridge. After the patient the simulation ends, the therapist will be able to monitor the progress of the patient thanks to the progress report.

This application is not meant to be a replacement for the traditional method of treating patients with acrophobia but an alternative.

VIII. Recommendations

Acrophobia VRET has already adequately addressed the problems it aimed to solve such as the addition of objectives in each of the virtual environments for the patient to follow. It can be used by therapists as a supplementary tool in treating patients diagnosed with acrophobia. There are still, however, some ways to improve the system and make it more useful to its users.

One is the replacement of the construction elevator into a panoramic elevator. Since patients who will use the system are assumed to be diagnosed with acrophobia, it does not seem realistic they are tasked to act like a construction worker given the objectives in the construction elevator virtual environment Construction workers are exposed to heights everyday. It reduces the realism of the environment. A panoramic elevator is suggested because it is more likely to be experienced by a common civilian. An example of this is the elevator located at the center of Robinson's Manila, Philippines.

Another possible improvement is the reduction of models in each of the virtual environment. The program seems to work fine, without lags, when it is played through a laptop or a desktop but when it was played using a budget friendly smartphone, complete with the three necessary requirements for virtual reality, noticeable lags occur. It only when the program is played using a high-end device such as the Samsung Galaxy S7, where this program was tested, do this lags disappear. A possible reduction in the models could be the deletion of the additional floors in the middle of the brick building. These will not be accessed by the patient so it is not needed in the virtual environment.

One more improvement, though minor, could be a prompt in the screen whenever objectives are not done in a linear manner. For example, in the glass bridge virtual environment, the patient must first save cat 1 before he/she is allowed to save cat 2. Not doing this in the proper order will render cat 2 untargetable and thus may slow down the objective progression of the patient.

A number of improvements can be added to this application. The addition of the improvements suggested above can greatly improve the application.

IX. Bibliography

[1] "Acrophobia Statistics." http://healthresearchfunding.org/acrophobia-statistics/.

[2] C. Coelho, A. Waters, T. Hine, and G. Wallis, "The use of virtual reality in acrophobia research and treatment," in *Journal of Anxiety Disorders*, pp. 563–574, Jan 2009.

[3] R. Menzies, "Height phobia in: G. l. c. davey (ed.), phobias," in *A handbook of theory, research and treatment*, pp. 129–138, 1997.

[4] D. W. Cortejos. Private Interview, March 2017.

[5] T. Davis, T. Ollendick, and L. st, "Intensive one-session treatment of specific phobias," in *New York: Springer*, 2012.

[6] P. Emmelkamp, M. Krijn, A. Hulsbosch, S. de Vries, M. Schuemie, and C. van der Mast, "Virtual reality treatment versus exposure in vivo: a comparative evaluation in acrophobia," *Behaviour Research and Therapy*, vol. 40, no. 5, pp. 509 – 516, 2002.

[7] T. Parab, D. Pawar, and A. Chaudhari, "A cost effective approach towards virtual reality: Phobia exposure therapy," in *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, pp. 670–674, Mar 2016.

[8] "Virtual Reality Definition." http://whatis.techtarget.com/definition/virtual-reality.

[9] "What is Virtual Reality?." https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html.

[10] "10 ways virtual reality is revolutionizing medicine and healthcare." http://www.techrepublic.com/article/10-ways-virtual-reality-is-revolutionizing-medicine-and-healthcare/.

[11] K. Raghav, A. Van Wijk, F. Abdullah, M. N. Islam, M. Bernatchez, and A. De Jongh, "Efficacy of virtual reality exposure therapy for treatment of dental phobia: a randomized control trial," *BMC Oral Health*, vol. 16, no. 1, p. 25, 2016.

[12] A. Rizzo, A. Hartholt, M. Grimani, A. Leeds, and M. Liewer, "Virtual reality exposure therapy for combat-related posttraumatic stress disorder," *Computer*, vol. 47, pp. 31–37, July 2014.

[13] C. Repetto, A. Gaggioli, F. Pallavicini, P. Cipresso, S. Raspelli, and G. Riva, "Virtual reality and mobile phones in the treatment of generalized anxiety disorders: a phase-2 clinical trial," *Personal and Ubiquitous Computing*, vol. 17, no. 2, pp. 253–260, 2013.

[14] C. M. Coelho, A. M. Waters, T. J. Hine, and G. Wallis, "The use of virtual reality in acrophobia research and treatment," *Journal of Anxiety Disorders*, vol. 23, no. 5, pp. 563 – 574, 2009.

[15] A. Surez, M. Santamara, and E. Claudio, "Virtual reality: A tool for treating phobias of heights," in *Eleventh LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI2013) Innovation in Engineering, Technology and Education for Competitiveness and Prosperity*, Aug 2013.

[16] S. Shunnaq and M. Raeder, "Virtualphobia: A model for virtual therapy of phobias," in *2016 XVIII Symposium on Virtual and Augmented Reality (SVR)*, pp. 59–63, June 2016.

[17] J. P. Costa, J. Robb, and L. E. Nacke, "Physiological acrophobia evaluation through in vivo exposure in a vr cave," in *2014 IEEE Games Media Entertainment*, pp. 1–4, Oct 2014.

[18] P. Leboucher, F. Levy, G. Rautureau, and R. Jouvent, "E-virtual reality exposure therapy in acrophobia: A pilot study.," in *J Telemed Telecare*, pp. 215–20, Aug 2016.

[19] P. Schfer, M. Koller, J. Diemer, and G. Meixner, "Development and evaluation of a virtual reality-system with integrated tracking of extremities under the aspect of acrophobia," in *2015 SAI Intelligent Systems Conference (IntelliSys)*, pp. 408–417, Nov 2015.

[20] J. Diemer, N. Lohkamp, A. Mhlberger, and P. Zwanzger, "Fear and physiological arousal during a virtual height challengeeffects in patients with acrophobia and healthy controls," *Journal of Anxiety Disorders*, vol. 37, pp. 30 – 39, 2016.

[21] W. P. Castro, M. J. R. Snchez, C. T. P. Gonzlez, J. M. Bethencourt, J. A. de la Fuente Portero, and R. G. Marco, "Cognitive-behavioral treatment and antidepressants combined with virtual reality exposure for patients with chronic agoraphobia," *International Journal of Clinical and Health Psychology*, vol. 14, no. 1, pp. 9 – 17, 2014.

[22] G. Walkom, "Virtual reality exposure therapy: To benefit those who stutter and treat social anxiety," in *2016 International Conference on Interactive Technologies and Games (ITAG)*, pp. 36–41, Oct 2016.

[23] A. Miloff, P. Lindner, W. Hamilton, L. Reuterskiöld, G. Andersson, and P. Carlbring, "Single-session gamified virtual reality exposure therapy for spider phobia vs. traditional exposure therapy: study protocol for a randomized controlled non-inferiority trial," *Trials*, vol. 17, no. 1, p. 60, 2016.

[24] "Understanding Acrophobia." http://www.psychologistanywhereanytime.com/phobias_psychologist_and_psychologists/psychologist_acrophobia.htm.

[25] "Acrophobia Pills and Medicines." http://www.changethatsrightnow.com/acrophobia/medication-and-drugs/.

[26] "Acrophobia: Fear of Height - Causes, Symptoms and Treatment." https://www.healthtopia.net/disease/mental-health/phobia/acrophobia-causes-symptoms-treatment.

[27] J. Wolpe, "Psychotherapy by reciprocal inhibition," in *California: Stanford University Press*, 1958.

[28] "Blender." https://www.blender.org/.

X. Appendix

A. Forms

B. Source Code

```
  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

/**
 * To be attach to the hinge of the door (Its pivot point)
 */
public class DoorRotate : MonoBehaviour {

        //For rotation. Parang if naclick yung object na naka-point dun ang reticle, set this to true para matrigge
        private bool isClicked = false;
        public Quaternion dir;
        public float smoothSpeed = 0.5f;

        // Use this for initialization
        void Start () {
                dir =  Quaternion.Euler(new Vector3(0, 0, 0));
        }

        // Update is called once per frame
        void Update () {
                if(isClicked == true){
                //if(true){
                        if (Input.GetAxis ("Vertical") == 0 && Input.GetAxis ("Horizontal") == 0) {
//                              dir.x = -135;
//                              dir.y = 0;
//                              dir.z = 0;
                                transform.rotation = Quaternion.Slerp(transform.rotation, dir, Time.deltaTime * sm
                                //transform.rotation = Quaternion.Lerp (from.rotation, to.rotation, Time.time * spe
                        }
                }
        }

        public void setIsClickedToTrue(){
                isClicked = true;
        }
}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Lerp : MonoBehaviour {

        public GameObject wall; //Object to move
        private Vector3 startPos;
        private Vector3 endPos;
        private Vector3 originalPosition;

        private float distance = 5f;


        // Use this for initialization
        void Start () {
                originalPosition = transform.position;
                startPos = transform.position;
                endPos = transform.position + new Vector3(0, 5f, 0);
                //startPos = transform.position;
                //endPos = transform.position + Vector3.up * distance;
        }

        // Update is called once per frame
        void Update () {

        }

        IEnumerator MoveObject(Vector3 source, Vector3 target, float overTime){
                if(transform.position.y > target.y - 0.0001f){
                        GetComponent<Renderer> ().material.SetColor("_Color", Color.yellow);
                }else{
                        GetComponent<Renderer> ().material.SetColor("_Color", Color.red);
                        float startTime = Time.time;
                        while(Time.time < startTime + overTime){
                                transform.position = Vector3.Lerp(transform.position, target, (Time.time - startTim
                                yield return null;
```

```
                        }
                        transform.position = target;
                }
        }

        public void Action (){
                StartCoroutine (MoveObject(startPos, endPos, 100
        }
}

/**

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Lerp : MonoBehaviour {

        public GameObject wall; //Object to move

        private Vector3 startPos;
        private Vector3 endPos;
        private float distance = 5f;

        private float lerpTime = 2f; //Gaano katagal para lumipat
        private float currentLerpTime = 0; //Updates the lerp tim

        private bool keyHit = false;

        // Use this for initialization
        void Start () {
                startPos = wall.transform.position;
                endPos = wall.transform.position + Vector3.up * d
        }

        // Update is called once per frame
        void Update () {
                if (Input.GetKeyDown (KeyCode.A)) {
                        keyHit = true;
                }

                if (keyHit == true) {
                        currentLerpTime += Time.deltaTime;
                        if(currentLerpTime >= lerpTime){
                                currentLerpTime = lerpTime;
                        }

                        float Percentage = currentLerpTime / lerp
                        transform.position = Vector3.Lerp (startP
                }
        }
}

**/

//////////////////////////////////////
//Script created by Alexander Ameye //
//Version: 2.2 (FREE)
//////////////////////////////////////

using System.Collections;
using UnityEngine;

public class ModifiedDoor : MonoBehaviour
{
        // INSPECTOR SETTINGS
        [Header("Door Settings")]
        [Tooltip("The initial angle of the door/window.")]
        public float InitialAngle = 0.0F;
        [Tooltip("The amount of degrees the door/window rotates."
        public float RotationAngle = 90.0F;
        public bool isDoorOpen;

        public enum PositionOfHinge
        {
```

```
        Left,
        Right,
}
public PositionOfHinge HingePosition;

public enum SideOfRotation
{
        Left,
        Right,
}
public SideOfRotation RotationSide;

[Tooltip("Rotating speed of the door/window.")]
public float Speed = 3F;
[Tooltip("0 = infinite times")]
public int TimesMoveable = 0;

// PRIVATE SETTINGS
private int n = 0; //For 'TimesMoveable' loop.
[HideInInspector] public bool Running = false;

// DEBUGGING
[Header("Debug Settings")]
[Tooltip("Visualizes the position of the hinge in-game by a colored cube.")]
public bool VisualizeHinge = false;
[Tooltip("The color of the visualization of the hinge.")]
public Color HingeColor = Color.cyan;

//Define an initial and final rotation.
private Quaternion FinalRot, InitialRot;
private int State;

//Create a hinge.
GameObject hinge;

// START FUNCTION
void Start ()
{
        isDoorOpen = false;

        //Give the object the name "Door" for future reference.
        gameObject.tag = "Door";

        //Create a hinge.
        hinge = new GameObject();
        hinge.name = "hinge";

        //Calculate sine/cosine of initial angle (needed for hinge positioning).
        float CosDeg = Mathf.Cos ((transform.eulerAngles.y * Mathf.PI) / 180);
        float SinDeg = Mathf.Sin ((transform.eulerAngles.y * Mathf.PI) / 180);

        //Read transform (position/rotation/scale) of the door.
        float PosDoorX = transform.position.x;
        float PosDoorY = transform.position.y;
        float PosDoorZ = transform.position.z;

        float RotDoorX = transform.localEulerAngles.x;
        float RotDoorZ = transform.localEulerAngles.z;

        float ScaleDoorX = transform.localScale.x;
        float ScaleDoorZ = transform.localScale.z;

        //Create a placeholder of the hinge's position/rotation.
        Vector3 HingePosCopy = hinge.transform.position;
        Vector3 HingeRotCopy = hinge.transform.localEulerAngles;

        // HINGE LEFT
        if (HingePosition == PositionOfHinge.Left)
        {
                // CALCULATE
                if (transform.localScale.x > transform.localScale.z)
                {
                        HingePosCopy.x = (PosDoorX - (ScaleDoorX / 2 * CosDeg));
                        HingePosCopy.z = (PosDoorZ + (ScaleDoorX / 2 * SinDeg));
                        HingePosCopy.y = PosDoorY;

                        HingeRotCopy.x = RotDoorX;
                        HingeRotCopy.y = -InitialAngle;
                        HingeRotCopy.z = RotDoorZ;
                }

                else
                {
                        HingePosCopy.x = (PosDoorX + (ScaleDoorZ / 2 * SinDeg));
                        HingePosCopy.z = (PosDoorZ + (ScaleDoorZ / 2 * CosDeg));
                        HingePosCopy.y = PosDoorY;

                        HingeRotCopy.x = RotDoorX;
                        HingeRotCopy.y = -InitialAngle;
                        HingeRotCopy.z = RotDoorZ;
        }
        }
```

```
        // RIGHT
        if (HingePosition == PositionOfHinge.Right)
        {
                // CALCULATE
                if(transform.localScale.x > transform.loc
                {
                        HingePosCopy.x = (PosDoorX + (Sca
                        HingePosCopy.z = (PosDoorZ - (Sca
                        HingePosCopy.y = PosDoorY;

                        HingeRotCopy.x = RotDoorX;
                        HingeRotCopy.y = -InitialAngle;
                        HingeRotCopy.z = RotDoorZ;
                }

                else
                {
                        HingePosCopy.x = (PosDoorX - (Sca
                        HingePosCopy.z = (PosDoorZ - (Sca
                        HingePosCopy.y = PosDoorY;

                        HingeRotCopy.x = RotDoorX;
                        HingeRotCopy.y = -InitialAngle;
                        HingeRotCopy.z = RotDoorZ;
                }
        }

        // HINGE POSITIONING
        hinge.transform.position = HingePosCopy;
        transform.parent = hinge.transform;
        hinge.transform.localEulerAngles = HingeRotCopy;

        // DEBUGGING
        if (VisualizeHinge == true)
        {
                GameObject cube = GameObject.CreatePrimit
                cube.transform.position = HingePosCopy;
                cube.transform.localScale = new Vector3(0
                cube.GetComponent<Renderer>().material.co
        }

        // ERROR CODES (UN-COMMENT THIS WHEN YOU'RE NOT E
        /*if (Mathf.Abs(InitialAngle) + Mathf.Abs(Rotatio
        {
                UnityEditor.EditorUtility.DisplayDialog (
                UnityEditor.EditorApplication.isPlaying =
        }*/

        // ANGLES
        if (RotationSide == SideOfRotation.Left)
        {
                InitialRot = Quaternion.Euler (0, -Initia
                FinalRot = Quaternion.Euler(0, -InitialAn
        }

        if (RotationSide == SideOfRotation.Right)
        {
                InitialRot = Quaternion.Euler (0, -Initia
                FinalRot = Quaternion.Euler(0, -InitialAn
        }
}

// OPEN FUNCTION
public IEnumerator Open ()
{
        if (n < TimesMoveable || TimesMoveable == 0)
        {
                        //Change state from 1 to 0 and ba
                        if (hinge.transform.rotation == (

                        //Set 'FinalRotation' to 'FinalRo
                        Quaternion FinalRotation = ((Stat

                if (FinalRotation == FinalRot) {
                        Debug.Log ("Door Open");
                        isDoorOpen = true;
                } else {
                        Debug.Log ("Door Closed");
                        isDoorOpen = false;
                }

        //Make the door/window rotate until it is fully c
        while (Mathf.Abs(Quaternion.Angle(FinalRotation,
        {
                                Running = true;
                                hinge.transform.rotation

        yield return new WaitForEndOfFrame();
        }
```

```
                                Running = false;

                                if (TimesMoveable == 0) n = 0;

                                else n++;
                        }
                }

                public void OpenCloseDoor (){
                        if (Running == false) {
                                StartCoroutine (Open());
                        }
                }

                public void CE_OpenCloseDoor (){ //Construction Elevator
                        bool isCERunning = GameObject.Find("Construction Elevator").GetComponent<CE_Up>().isRunning;
                        bool hasCrowbar = GameObject.Find("Crowbar").GetComponent<CE_Obj3>().hasCrowbar;
                        if(hasCrowbar){
                                if (Running == false) {
                                        if(!isCERunning){
                                                StartCoroutine (Open());
                                        }

                                }
                        }
                }

                public void CE_BuildingInterior_OpenCloseDoor (){ //Construction Elevator
                        bool isKeyTaken = GameObject.Find("Keys").GetComponent<CE_Obj2>().isKeyTaken;
                        if (Running == false) {
                                if(isKeyTaken){
                                        StartCoroutine (Open());
                                }

                        }
                }
}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Move : MonoBehaviour {

        public Color color;
        public float delayTime = 0.3f;
        public KeyCode beginDemo; //Timer is linked through keypress. Can create delay before and after the press.

        public float transitionSpeed = 2f;


        public Transform target;
        public float smoothTime = 0.1F;
        private Vector3 velocity = Vector3.zero;

        //For rotation. Parang if naclick yung object na naka-point dun ang reticle, set this to true para matrigge
        private bool isClicked = false;
        public Quaternion dir;
        public float smoothSpeed = 0.5f;

        // Use this for initialization
        void Start () {

        }

        // Update is called once per frame
        void Update () {
                if(isClicked == true){
                        if (Input.GetAxis ("Vertical") == 0 && Input.GetAxis ("Horizontal") == 0) {
                                dir.y = -135;
                                transform.rotation = Quaternion.Lerp (transform.rotation, dir, Time.deltaTime * smo
                        }
                }

                //Vector3 targetPosition = target.TransformPoint(new Vector3(0, 2.33f, 4.71f));
                //transform.position = Vector3.SmoothDamp(transform.position, targetPosition, ref velocity, smoothT


                /*
                //Input.GetKey is for rapid fire raw. For every frame? Hinohold?
                if (Input.GetKey(beginDemo)) { //If key has been pressed
                        //StartCoroutine (processTask());
                        transform.Translate (0, transitionSpeed * Time.deltaTime, 0);
                }
                */

                /*
                if (Input.GetKey(beginDemo)) { //Pwede for elevator pero you have to add another conditon: if nasa
                        transform.Translate (0, transitionSpeed * Time.deltaTime, 0);
                }
                */
```

```
        }
        IEnumerator processTask(){ // This is called a co-routine
                yield return new WaitForSeconds (delayTime); //1.
                //transform.Translate (0, transitionSpeed * Time.
                transform.position += new Vector3(0f, 2f, 0f);
                //transform.Rotate (0, 0, 3); //Rotate 3 degrees
        }

        IEnumerator changeColor(){ // This is called a co-routine
                yield return new WaitForSeconds (delayTime); //1.
                GetComponent<Renderer> ().material.SetColor ("_Co
        }

        IEnumerator changeColorOfAnother(){ // This is called a c
                yield return new WaitForSeconds (delayTime); //1.

                var go = GameObject.Find("Sphere");
                //go.transform.Translate(0, 1, 0);

                go.GetComponent<Renderer> ().material.SetColor ("

        }

        IEnumerator doorLikeRotation(){ // This is called a co-ro
                yield return new WaitForSeconds (0); //for delay
                var go = GameObject.Find("FrontDoor");
                var hinge = go.transform.Find ("Hinge_Door_001");
                hinge.transform.rotation = Quaternion.Euler(45, 4

                /*
                for (var i = 0; i < 100; i++)
                {
                        go.transform.rotation = Quaternion.Euler(
                        yield return new WaitForSeconds (delayTim
                }
                */
        }

        public void Action(){
                //StartCoroutine (processTask());
                StartCoroutine (changeColor());
                StartCoroutine (changeColorOfAnother());
                //StartCoroutine (doorLikeRotation());
        }

        public void setIsClickedToTrue(){
        //Set the key to Space for space bar
                isClicked = true;
        }
}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SlenderAppear : MonoBehaviour {

        // Use this for initialization
        void Start () {

        }

        void OnTriggerEnter() {
                StartCoroutine (appear());
        }

        IEnumerator appear(){
                GetComponent<BoxCollider> ().enabled = false;
                GetComponent<AudioSource>().Play();
                GameObject.Find("Slenderman2").transform.Translat
                yield return new WaitForSeconds (0.5f); //Time be
                Destroy(GameObject.Find("Slenderman2"));
        }
}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SlenderDisappear : MonoBehaviour {
        public GameObject Slenderman;

        // Use this for initialization
        void Start () {
                Slenderman = GameObject.Find("Slenderman");
        }

        void OnTriggerEnter() {
                StartCoroutine (disappear());
                GetComponent<BoxCollider> ().enabled = false;
                GameObject.Find("SlenderAppear").GetComponent<BoxCollider> ().enabled = true;
        }

        IEnumerator disappear(){
                float randomNumber = Random.Range (4f,7f);
                Debug.Log (randomNumber);
                yield return new WaitForSeconds (randomNumber); //Time before slendy disappears
                GetComponent<AudioSource>().Play();
                Destroy (Slenderman);
        }
}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SmoothRotation : MonoBehaviour {
        public Quaternion dir;
        public float smoothSpeed = 0.5f;

        // Use this for initialization
        void Start () {

        }

        // Update is called once per frame
        void Update () {
                if (Input.GetAxis ("Vertical") == 0 && Input.GetAxis ("Horizontal") == 0) {
                        dir.y = -135;
                        transform.rotation = Quaternion.Lerp (transform.rotation, dir, Time.deltaTime * smoothSpeed
                }
        }
}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class StopPlane : MonoBehaviour {

        public GameObject ObjectToBeAttached;
        public GameObject StopText;

        // Use this for initialization
        void Start () {
        }

        // Update is called once per frame
        void Update () {
                transform.rotation = Quaternion.Euler(transform.rotation.eulerAngles.x, ObjectToBeAttached.transform
        }

        public void boldText(){
                StopText.GetComponent<TextMesh>().fontStyle = FontStyle.Bold;
        }

        public void unboldText(){
                StopText.GetComponent<TextMesh>().fontStyle = FontStyle.Normal;
        }
}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//Construction Elevator Objective 1
public class CE_Obj1 : MonoBehaviour {

        private CE_ObjectivesManager CE_ObjectivesManager;

        void Start () {
                CE_ObjectivesManager = GameObject.Find("CE_ObjectivesCanvas").GetComponent<CE_ObjectivesManager>();
        }

        //Switch to objective 2 (e.g. Objective n switch to Objective n+1)
        //Theoretical: Save object 1's completion time here
        void OnTriggerEnter() {
                Destroy(gameObject);
                Debug.Log ("Player Has Entered Brick Building");

                CE_ObjectivesManager.objective1 = false;
                CE_ObjectivesManager.objective2 = true;
                CE_ObjectivesManager.objective3 = false;
                CE_ObjectivesManager.objective4 = false;
                CE_ObjectivesManager.objective5 = false;
                CE_ObjectivesManager.objective6 = false;
                CE_ObjectivesManager.objective7 = false;

                //BoxCollider nextTarget = GameObject.Find ("ObjectCE2").GetComponent<BoxCollider> ();
```

```
                //nextTarget.enabled = true;
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//Construction Elevator Objective 2
public class CE_Obj2 : MonoBehaviour {

        private CE_ObjectivesManager CE_ObjectivesManager;
        public bool keyTaken;

        void Start () {
                keyTaken = false;
                CE_ObjectivesManager = GameObject.Find("CE_Objec
        }

        public void SwitchToObj3() {
                gameObject.GetComponent<MeshRenderer> ().enabled
                Debug.Log ("Player Has Found The Keys");

                CE_ObjectivesManager.objective1 = false;
                CE_ObjectivesManager.objective2 = false;
                CE_ObjectivesManager.objective3 = true;
                CE_ObjectivesManager.objective4 = false;
                CE_ObjectivesManager.objective5 = false;
                CE_ObjectivesManager.objective6 = false;
                CE_ObjectivesManager.objective7 = false;
                keyTaken = true;

        }

}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

//Construction Elevator Objective 2
public class CE_Obj3 : MonoBehaviour {

        private CE_ObjectivesManager CE_ObjectivesManager;
        public bool hasCrowbar;

        void Start () {
                hasCrowbar = false;
                CE_ObjectivesManager = GameObject.Find("CE_Objec
        }

        public void SwitchToObj4() {
                gameObject.GetComponent<MeshRenderer> ().enabled
                Debug.Log ("Player Has Found The Crowbar");

                CE_ObjectivesManager.objective1 = false;
                CE_ObjectivesManager.objective2 = false;
                CE_ObjectivesManager.objective3 = false;
                CE_ObjectivesManager.objective4 = true;
                CE_ObjectivesManager.objective5 = false;
                CE_ObjectivesManager.objective6 = false;
                CE_ObjectivesManager.objective7 = false;
                hasCrowbar = true;

                BoxCollider nextTarget = GameObject.Find ("Objec
                nextTarget.enabled = true;
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CE_Obj4 : MonoBehaviour {

        private CE_ObjectivesManager CE_ObjectivesManager;

        void Start () {
                CE_ObjectivesManager = GameObject.Find("CE_Objec
        }

        void OnTriggerEnter() {
                Destroy(gameObject);
                Debug.Log ("Player Has Found The Construction Ele

                CE_ObjectivesManager.objective1 = false;
                CE_ObjectivesManager.objective2 = false;
                CE_ObjectivesManager.objective3 = false;
                CE_ObjectivesManager.objective4 = false;
                CE_ObjectivesManager.objective5 = true;
                CE_ObjectivesManager.objective6 = false;
                CE_ObjectivesManager.objective7 = false;

                //BoxCollider nextTarget = GameObject.Find ("Objective2").GetComponent<BoxCollider>();
                //nextTarget.enabled = true;
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CE_Obj5 : MonoBehaviour {

        private CE_ObjectivesManager CE_ObjectivesManager;

        void Start () {
                CE_ObjectivesManager = GameObject.Find("CE_ObjectivesCanvas").GetComponent<CE_ObjectivesManager>();
        }

        void OnTriggerEnter() {
                Destroy(gameObject);
                Debug.Log ("Player Has Entered The Construction Elevator");

                CE_ObjectivesManager.objective1 = false;
                CE_ObjectivesManager.objective2 = false;
                CE_ObjectivesManager.objective3 = false;
                CE_ObjectivesManager.objective4 = false;
                CE_ObjectivesManager.objective5 = false;
                CE_ObjectivesManager.objective6 = true;
                CE_ObjectivesManager.objective7 = false;

                //BoxCollider nextTarget = GameObject.Find ("Objective2").GetComponent<BoxCollider>();
                //nextTarget.enabled = true;
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CE_Obj6 : MonoBehaviour {

        private CE_ObjectivesManager CE_ObjectivesManager;

        void Start () {
                CE_ObjectivesManager = GameObject.Find("CE_ObjectivesCanvas").GetComponent<CE_ObjectivesManager>();
        }

        void OnTriggerEnter() {
                Debug.Log ("Player Has Reached The Top");

                CE_ObjectivesManager.objective1 = false;
                CE_ObjectivesManager.objective2 = false;
                CE_ObjectivesManager.objective3 = false;
                CE_ObjectivesManager.objective4 = false;
                CE_ObjectivesManager.objective5 = false;
                CE_ObjectivesManager.objective6 = false;
                CE_ObjectivesManager.objective7 = true;

                Destroy(gameObject);
                //BoxCollider nextTarget = GameObject.Find ("Objective2").GetComponent<BoxCollider>();
                //nextTarget.enabled = true;
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CE_Obj7 : MonoBehaviour {

        private CE_ObjectivesManager CE_ObjectivesManager;

        void Start () {
                CE_ObjectivesManager = GameObject.Find("CE_ObjectivesCanvas").GetComponent<CE_ObjectivesManager>();
        }

        public void Finish() {
                Debug.Log ("Player Has Finished Level");

                CE_ObjectivesManager.objective1 = false;
                CE_ObjectivesManager.objective2 = false;
                CE_ObjectivesManager.objective3 = false;
                CE_ObjectivesManager.objective4 = false;
                CE_ObjectivesManager.objective5 = false;
                CE_ObjectivesManager.objective6 = false;
                CE_ObjectivesManager.objective7 = false;
                CE_ObjectivesManager.objective8 = true;
        }
}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
//using UnityEngine.VR;

//[RequireComponent(typeof(Text))]
public class CE_ObjectivesManager: MonoBehaviour {

        private Text textField;
        public Camera cam;

        //public int indexOfScene;

        public bool objective1;
        public bool objective2;
        public bool objective3;
        public bool objective4;
        public bool objective5;
        public bool objective6;
        public bool objective7;
        public bool objective8;

        public float counter = 0;
        public bool timer;
        public static bool isStopped;

        public static float objective1_time = 0;
        public static float objective2_time = 0;
        public static float objective3_time = 0;
        public static float objective4_time = 0;
        public static float objective5_time = 0;
        public static float objective6_time = 0;
        public static float objective7_time = 0;
        public static float objective8_time = 0;

        public static string objective1_remark;
        public static string objective2_remark;
        public static string objective3_remark;
        public static string objective4_remark;
        public static string objective5_remark;
        public static string objective6_remark;
        public static string objective7_remark;
        public static string objective8_remark;

        // Use this for initialization
        void Start() {
                isStopped = false;
                objective1 = true;
                objective2 = false;
                objective3 = false;
                objective4 = false;
                objective5 = false;
                objective6 = false;
                objective7 = false;
                objective8 = false;

                objective1_remark = "n/a";
                objective2_remark = "n/a";
                objective3_remark = "n/a";
                objective4_remark = "n/a";
                objective5_remark = "n/a";
                objective6_remark = "n/a";
                objective7_remark = "n/a";
                objective8_remark = "n/a";

                textField = GetComponent<Text>();
                timer = true;

                if (cam == null) {
                        cam = Camera.main;
                }

                if (cam != null) {
                        // Tie this to the camera, and do not kee
                        transform.SetParent(cam.GetComponent<Tran
                }

                //Debug.Log ("Index: " + controller.indexOfScene)
        }

        // Update is called once per frame
        void Update () {
                Debug.Log ("Index Of Scene: " + controller.indexOfScene);

                //int upToObjectives =

                if (timer) {
                        counter += Time.deltaTime;

                        //Debug.Log ("Time:" + counter);
                } else {
                        counter = counter;
                        Debug.Log ("Stop");
                }

                if (objective1) {
                        textField.text = "Enter the brick building";
                        objective1 = false;
                }

                if (objective2) {
                        objective1_time = counter;
                        textField.text = "Find the keys";
                        objective1_remark = "Done";
                        objective2 = false;

                }
                if (objective3) {
                        objective2_time = counter;
                        textField.text = "Find the crowbar in the room";
                        objective2_remark = "Done";
                        objective3 = false;
                }
                if (objective4) {
                        objective3_time = counter;
                        textField.text = "Find the construction elevator";
                        objective3_remark = "Done";
                        objective4 = false;
                }
                if (objective5) {
                        objective4_time = counter;
                        textField.text = "Enter the construction elevator";
                        objective4_remark = "Done";
                        objective5 = false;
                }
                if (objective6) {
                        objective5_time = counter;
                        textField.text = "Turn on the construction elevator";
                        objective5_remark = "Done";
                        objective6 = false;
                }
                if (objective7) {
                        objective6_time = counter;
                        textField.text = "Press the red button";
                        objective6_remark = "Done";
                        objective7 = false;
                }
                if (objective8) {
                        objective7_time = counter;
                        textField.text = "LEVEL FINISHED";
                        objective7_remark = "Done";
                        objective8 = false;

                        //SceneManager.LoadScene (4);
                        Application.LoadLevel ("Progress Report");
                }
        }

        public void EndSimulation (){
                isStopped = true;
                Application.LoadLevel ("Progress Report");
        }
}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CE_PlatformSticker : MonoBehaviour {
        public GameObject ObjectToBeAttached;

        void Start(){

        }

        void OnTriggerEnter(Collider yungMagcocontact) {
                //So that the player will go up with the elevator
                yungMagcocontact.gameObject.transform.parent = ObjectToBeAttached.transform;
                Debug.Log("Player Sticks To Construction Elevator");
        }

        void OnTriggerExit(Collider yungMagcocontact) {
                yungMagcocontact.gameObject.transform.parent = nu
                Debug.Log("Player Does Not Stick To Construction
        }
}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CE_Up : MonoBehaviour {

        public GameObject ObjectToMove;
        private Vector3 startPos;
        private Vector3 endPos;
        private float distance = 103.05f;

        public GameObject ConstructionElevatorDoor;
        public GameObject ObjectToBeAttached;

        //Para kapag clinick yung button habang nag-aanimate, wa
        public bool isRunning;
        private bool isDoorOpen;
        public bool isOnTop;

        // Use this for initialization
        void Start () {
                startPos = transform.position;
                endPos = transform.position + Vector3.up * distan
                isOnTop = false;
                isRunning = false;
        }

        IEnumerator GoUp(Vector3 source, Vector3 target, float ov
                Debug.Log ("Construction Elevator Starts Going Up
                ConstructionElevatorDoor.transform.parent.parent
                isRunning = true;
                float startTime = Time.time;
                while(Time.time < startTime + overTime){
                        transform.position = Vector3.Lerp(transfo
                        yield return null;
                        if(transform.position.y > target.y - 0.01
                                break;
                        }
                }
                Debug.Log ("Construction Elevator Has Reached Top
                ConstructionElevatorDoor.transform.parent.parent
                isRunning = false;
                transform.position = target;
                isOnTop = true;


        }

        public void Up (){
                isDoorOpen = GameObject.Find("InviConElevatorDoor
                bool hasCrowbar = GameObject.Find("Crowbar").GetC
                if (hasCrowbar) {
                        if(!isDoorOpen){
                                if (!isRunning) {
                                        StartCoroutine (GoUp (sta
                                }
                        }
                }
        }

}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ElevDoorLeft : MonoBehaviour {

        public GameObject ObjectToMove; //Object to move
        private Vector3 startPos;
        private Vector3 endPos;
        public float distance = 2f;

        //Para kapag clinick yung button habang nag-aanimate, wa
        public bool isRunning;
        private bool isMovingElevator;

        public float delayTime = 3f;

        // Use this for initialization
        void Start () {
                startPos = transform.position;
                endPos = transform.position + Vector3.forward * d
                isRunning = false;
        }

        // Update is called once per frame
        void Update () {
        }

        IEnumerator MoveObject(Vector3 source, Vector3 target, float overTime){
                isRunning = true;
                //Opening
                Debug.Log ("Elevator Door Open");
                float startTime = Time.time;
                while(Time.time < startTime + overTime){
                        transform.position = Vector3.Lerp(source, target, (Time.time - startTime) / overTime);
                        yield return null;
                }
                transform.position = target;
                //GetComponent<Renderer> ().material.SetColor ("_Color", Color.red);

                //Delay
                yield return new WaitForSeconds (delayTime);
                //Close
                //GetComponent<Renderer> ().material.SetColor ("_Color", Color.yellow);
                startTime = Time.time;
                while(Time.time < startTime + overTime){
                        transform.position = Vector3.Lerp(target, source, (Time.time - startTime) / overTime);
                        yield return null;
                }
                transform.position = source;
                Debug.Log ("Elevator Door Close");
                isRunning = false;
        }

        public void OpenClose (){
                isMovingElevator = GameObject.Find("ElevatorInterior").GetComponent<UpDown>().isRunning;
                Debug.Log ("isMovingElevator: " + isMovingElevator);
                if(!isMovingElevator){
                        if (!isRunning) {
                                StartCoroutine (MoveObject(startPos, endPos, 3)); //TIME IT TAKES TO COMPLETE
                        }
                }
        }
}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ElevDoorRight : MonoBehaviour {

        public GameObject ObjectToMove; //Object to move
        private Vector3 startPos;
        private Vector3 endPos;
        public float distance = 2f;

        //Para kapag clinick yung button habang nag-aanimate, walang mangyayari
        private bool isRunning;
        private bool isMovingElevator;

        public float delayTime = 3f;

        // Use this for initialization
        void Start () {
                startPos = transform.position;
                endPos = transform.position - Vector3.forward * distance;
                isRunning = false;
        }

        // Update is called once per frame
        void Update () {
        }

        IEnumerator MoveObject(Vector3 source, Vector3 target, float overTime){
                isRunning = true;
                //Opening
                float startTime = Time.time;
                while(Time.time < startTime + overTime){
                        transform.position = Vector3.Lerp(source, target, (Time.time - startTime) / overTime);
                        yield return null;
                }
                transform.position = target;
                //GetComponent<Renderer> ().material.SetColor ("_Color", Color.red);

                //Delay
                yield return new WaitForSeconds (delayTime);
                //Close
                //GetComponent<Renderer> ().material.SetColor ("_Color", Color.yellow);
                startTime = Time.time;
                while(Time.time < startTime + overTime){
                        transform.position = Vector3.Lerp(target, source, (Time.time - startTime) / overTime);
                        yield return null;
                }
                transform.position = source;
                isRunning = false;
        }
```

```
        public void OpenClose (){
                isMovingElevator = GameObject.Find("ElevatorInter
                if(!isMovingElevator){
                        if (!isRunning) {
                                StartCoroutine (MoveObject(startP
                        }
                }
        }
}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlatformSticker : MonoBehaviour {
        public GameObject ObjectToBeAttached;

        void Start(){

        }

        void OnTriggerEnter(Collider yungMagcocontact) {
                //So that the player will go up with the elevator
                yungMagcocontact.gameObject.transform.parent = O
        }

        void OnTriggerExit(Collider yungMagcocontact) {
                yungMagcocontact.gameObject.transform.parent = n
        }
}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class UpDown : MonoBehaviour {

        public GameObject ObjectToMove;
        private Vector3 startPos;
        private Vector3 endPos;
        private float distance = 101.98f;

        //Para kapag clinick yung button habang nag-aanimate, wal
        public bool isRunning;
        private bool isOpeningDoor;
        private bool isOnTop;



        // Use this for initialization
        void Start () {
                startPos = transform.position;
                endPos = transform.position - Vector3.down * dist
                isOnTop = false;
                isRunning = false;
        }

        IEnumerator GoUp(Vector3 source, Vector3 target, float ov
                Debug.Log ("Elevator Starts Going Up");
                isRunning = true;
                float startTime = Time.time;
                while(Time.time < startTime + overTime){
                        transform.position = Vector3.Lerp(transfo
                        yield return null;
                        if(transform.position.y > target.y - 0.01
                                break;
                        }
                }
                Debug.Log ("Elevator Has Reached Top");
                isRunning = false;
                transform.position = target;
                isOnTop = true;
                //TopElevatorDoorLeft, TopElevatorDoorLeft, Eleva
                //Open doors when top floor is reached
                //Used for open elev doors when top floor reached
                GameObject.Find("ElevatorDoorLeft").GetComponent<
                GameObject.Find("ElevatorDoorRight").GetComponent
                GameObject.Find("TopElevatorDoorLeft").GetCompone
                GameObject.Find("TopElevatorDoorRight").GetCompor
        }

        IEnumerator GoDown(Vector3 source, Vector3 target, float
                Debug.Log ("Elevator Starts Going Down");
                isRunning = true;
                float startTime = Time.time;
                while(Time.time < startTime + overTime){
                        transform.position = Vector3.Lerp(target,
                        yield return null;
                        if(transform.position.y < source.y + 0.01){
                                break;
                        }
                }
                Debug.Log ("Elevator Has Reached Bottom");
                isRunning = false;
                isOnTop = false;
                transform.position = source;
        }

        public void Up (){
                isOpeningDoor = GameObject.Find("ElevatorDoorLeft").GetComponent<ElevDoorLeft>().isRunning; //can g
                if(!isOpeningDoor){
                        if (!isRunning) {
                                StartCoroutine (GoUp (startPos, endPos, 150));
                        }
                }
        }

        public void Down (){
                isOpeningDoor = GameObject.Find("ElevatorDoorLeft").GetComponent<ElevDoorLeft>().isRunning; //can g
                if(!isOpeningDoor){
                        if(isOnTop){
                                if (!isRunning) {
                                        StartCoroutine (GoDown (startPos, endPos, 20));
                                }
                        }
                }
        }
}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FE_Obj1 : MonoBehaviour {

        private FE_ObjectivesManager FE_ObjectivesManager;

        void Start () {
                FE_ObjectivesManager = GameObject.Find("FE_ObjectivesCanvas").GetComponent<FE_ObjectivesManager>();
        }

        void OnTriggerEnter() {
                Debug.Log ("Player Has Entered Brick Building");

                FE_ObjectivesManager.objective1 = false;
                FE_ObjectivesManager.objective2 = true;
                FE_ObjectivesManager.objective3 = false;
                FE_ObjectivesManager.objective4 = false;
                FE_ObjectivesManager.objective5 = false;
                FE_ObjectivesManager.objective6 = false;
                FE_ObjectivesManager.objective7 = false;
                FE_ObjectivesManager.objective8 = false;
                FE_ObjectivesManager.objective9 = false;

                BoxCollider nextTarget = GameObject.Find ("Objective2").GetComponent<BoxCollider>();
                nextTarget.enabled = true;

                Destroy(gameObject);
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FE_Obj2 : MonoBehaviour {

        private FE_ObjectivesManager FE_ObjectivesManager;

        void Start () {
                FE_ObjectivesManager = GameObject.Find("FE_ObjectivesCanvas").GetComponent<FE_ObjectivesManager>();
        }

        void OnTriggerEnter() {
                Debug.Log ("Player Has Entered Elevator");

                FE_ObjectivesManager.objective1 = false;
                FE_ObjectivesManager.objective2 = false;
                FE_ObjectivesManager.objective3 = true;
                FE_ObjectivesManager.objective4 = false;
                FE_ObjectivesManager.objective5 = false;
                FE_ObjectivesManager.objective6 = false;
                FE_ObjectivesManager.objective7 = false;
                FE_ObjectivesManager.objective8 = false;
                FE_ObjectivesManager.objective9 = false;

                BoxCollider nextTarget = GameObject.Find ("Objective3").GetComponent<BoxCollider>();
```

```
                nextTarget.enabled = true;
                Destroy(gameObject);
        }
}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FE_Obj3 : MonoBehaviour {
        private FE_ObjectivesManager FE_ObjectivesManager;

        void Start () {
                FE_ObjectivesManager = GameObject.Find("FE_Objec
        }

        void OnTriggerEnter() {
                FE_ObjectivesManager.objective1 = false;
                FE_ObjectivesManager.objective2 = false;
                FE_ObjectivesManager.objective3 = false;
                FE_ObjectivesManager.objective4 = true;
                FE_ObjectivesManager.objective5 = false;
                FE_ObjectivesManager.objective6 = false;
                FE_ObjectivesManager.objective7 = false;
                FE_ObjectivesManager.objective8 = false;
                FE_ObjectivesManager.objective9 = false;

                BoxCollider nextTarget = GameObject.Find ("Object
                nextTarget.enabled = true;

                //To prevent player from going to elevator again
                GameObject.Find("Elevator_Button").GetComponent<

                Destroy(gameObject);
        }
}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FE_Obj4 : MonoBehaviour {

        private FE_ObjectivesManager FE_ObjectivesManager;

        void Start () {
                FE_ObjectivesManager = GameObject.Find("FE_Objec
        }

        void OnTriggerEnter() {
                FE_ObjectivesManager.objective1 = false;
                FE_ObjectivesManager.objective2 = false;
                FE_ObjectivesManager.objective3 = false;
                FE_ObjectivesManager.objective4 = false;
                FE_ObjectivesManager.objective5 = true;
                FE_ObjectivesManager.objective6 = false;
                FE_ObjectivesManager.objective7 = false;
                FE_ObjectivesManager.objective8 = false;
                FE_ObjectivesManager.objective9 = false;

                CharacterLookWalk.speed = 3.0f;

                BoxCollider nextTarget = GameObject.Find ("Object
                nextTarget.enabled = true;
                Destroy(gameObject);
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FE_Obj5 : MonoBehaviour {
        private FE_ObjectivesManager FE_ObjectivesManager;

        void Start () {
                FE_ObjectivesManager = GameObject.Find("FE_Objec
        }

        void OnTriggerEnter() {
                FE_ObjectivesManager.objective1 = false;
                FE_ObjectivesManager.objective2 = false;
                FE_ObjectivesManager.objective3 = false;
                FE_ObjectivesManager.objective4 = false;
                FE_ObjectivesManager.objective5 = false;
                FE_ObjectivesManager.objective6 = true;
                FE_ObjectivesManager.objective7 = false;
                FE_ObjectivesManager.objective8 = false;
                FE_ObjectivesManager.objective9 = false;

                BoxCollider nextTarget = GameObject.Find (”Objective6”).GetComponent<BoxCollider>();
                nextTarget.enabled = true;
                Destroy(gameObject);
        }

}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FE_Obj6 : MonoBehaviour {

        private FE_ObjectivesManager FE_ObjectivesManager;

        void Start () {
                FE_ObjectivesManager = GameObject.Find(”FE_ObjectivesCanvas”).GetComponent<FE_ObjectivesManager>();
        }

        void OnTriggerEnter() {
                FE_ObjectivesManager.objective1 = false;
                FE_ObjectivesManager.objective2 = false;
                FE_ObjectivesManager.objective3 = false;
                FE_ObjectivesManager.objective4 = false;
                FE_ObjectivesManager.objective5 = false;
                FE_ObjectivesManager.objective6 = false;
                FE_ObjectivesManager.objective7 = true;
                FE_ObjectivesManager.objective8 = false;
                FE_ObjectivesManager.objective9 = false;

                BoxCollider nextTarget = GameObject.Find (”Objective7”).GetComponent<BoxCollider>();
                nextTarget.enabled = true;
                Destroy(gameObject);
        }

}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FE_Obj7 : MonoBehaviour {

        private FE_ObjectivesManager FE_ObjectivesManager;

        void Start () {
                FE_ObjectivesManager = GameObject.Find(”FE_ObjectivesCanvas”).GetComponent<FE_ObjectivesManager>();
        }

        void OnTriggerEnter() {
                FE_ObjectivesManager.objective1 = false;
                FE_ObjectivesManager.objective2 = false;
                FE_ObjectivesManager.objective3 = false;
                FE_ObjectivesManager.objective4 = false;
                FE_ObjectivesManager.objective5 = false;
                FE_ObjectivesManager.objective6 = false;
                FE_ObjectivesManager.objective7 = false;
                FE_ObjectivesManager.objective8 = true;
                FE_ObjectivesManager.objective9 = false;

                CharacterLookWalk.speed = 5.0f;

                BoxCollider nextTarget = GameObject.Find (”Objective8”).GetComponent<BoxCollider>();
                nextTarget.enabled = true;

                Destroy(gameObject);
                Destroy (GameObject.Find (”Group_A”));
        }

}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FE_Obj8 : MonoBehaviour {

        private FE_ObjectivesManager FE_ObjectivesManager;

        void Start () {
                FE_ObjectivesManager = GameObject.Find(”FE_ObjectivesCanvas”).GetComponent<FE_ObjectivesManager>();
        }

        void OnTriggerEnter() {
                Destroy(gameObject);
                FE_ObjectivesManager.objective1 = false;
                FE_ObjectivesManager.objective2 = false;
                FE_ObjectivesManager.objective3 = false;
                FE_ObjectivesManager.objective4 = false;
                FE_ObjectivesManager.objective5 = false;
                FE_ObjectivesManager.objective6 = false;
                FE_ObjectivesManager.objective7 = false;
                FE_ObjectivesManager.objective8 = false;
                FE_ObjectivesManager.objective9 = true;
        }

}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
//using UnityEngine.VR;

//[RequireComponent(typeof(Text))]
public class FE_ObjectivesManager : MonoBehaviour {

        private Text textField;
        public Camera cam;

        //public int indexOfScene;

        public bool objective1;
        public bool objective2;
        public bool objective3;
        public bool objective4;
        public bool objective5;
        public bool objective6;
        public bool objective7;
        public bool objective8;
        public bool objective9;

        public float counter = 0;
        public bool timer;
        public static bool isStopped;

        public static float objective1_time = 0;
        public static float objective2_time = 0;
        public static float objective3_time = 0;
        public static float objective4_time = 0;
        public static float objective5_time = 0;
        public static float objective6_time = 0;
        public static float objective7_time = 0;
        public static float objective8_time = 0;
        public static float objective9_time = 0;

        public static string objective1_remark;
        public static string objective2_remark;
        public static string objective3_remark;
        public static string objective4_remark;
        public static string objective5_remark;
        public static string objective6_remark;
        public static string objective7_remark;
        public static string objective8_remark;
        public static string objective9_remark;

        // Use this for initialization
        void Start() {
                isStopped = false;
                objective1 = true;
                objective2 = false;
                objective3 = false;
                objective4 = false;
                objective5 = false;
                objective6 = false;
                objective7 = false;
                objective8 = false;
                objective8 = false;
                objective9 = false;

                objective1_remark = ”n/a”;
                objective2_remark = ”n/a”;
                objective3_remark = ”n/a”;
                objective4_remark = ”n/a”;
                objective5_remark = ”n/a”;
                objective6_remark = ”n/a”;
                objective7_remark = ”n/a”;
                objective8_remark = ”n/a”;
                objective9_remark = ”n/a”;

                textField = GetComponent<Text>();
                timer = true;

                if (cam == null) {
                        cam = Camera.main;
                }
                if (cam != null) {
                        // Tie this to the camera, and do not keep the local orientation
                        transform.SetParent(cam.GetComponent<Transform>(), true);
                }

                //Debug.Log ("Index: " + controller.indexOfScene);
        }

        // Update is called once per frame
        void Update () {
                //Debug.Log ("Index Of Scene: " + controller.indexOfScene);

                //int upToObjectives =

                if (timer) {
                        counter += Time.deltaTime;
                        //Debug.Log ("Time:" + counter);
                } else {
                        counter = counter;
                        Debug.Log ("Stop");
                }

                if (objective1) {
                        textField.text = "Enter the brick building";
                        objective1 = false;
                }

                if (objective2) {
                        objective1_time = counter;
                        textField.text = "Enter the elevator";
                        objective1_remark = "Done";
                        objective2 = false;
                }
                if (objective3) {
                        objective2_time = counter;
                        textField.text = "Go to the 13th floor";
                        objective2_remark = "Done";
                        objective3 = false;
                }
                if (objective4) {
                        objective3_time = counter;
                        textField.text = "Find the fire escape";
                        objective3_remark = "Done";
                        objective4 = false;
                }
                if (objective5) {
                        objective4_time = counter;
                        textField.text = "Reach the 8th floor";
                        objective4_remark = "Done";
                        objective5 = false;
                }
                if (objective6) {
                        objective5_time = counter;
                        textField.text = "Reach the 4th floor";
                        objective5_remark = "Done";
                        objective6 = false;
                }
                if (objective7) {
                        objective6_time = counter;
                        textField.text = "Reach the bottom";
                        objective6_remark = "Done";
                        objective7 = false;
                }
                if (objective8) {
                        objective7_time = counter;
                        textField.text = "Find the exit";
                        objective7_remark = "Done";
                        objective8 = false;
                }
                if (objective9) {
                        objective8_time = counter;
                        textField.text = "LEVEL FINISHED";
                        objective8_remark = "Done";
                        objective9 = false;

                        //SceneManager.LoadScene (4);
                        Application.LoadLevel ("Progress Report");
                }
        }

        public void EndSimulation (){
                isStopped = true;
                Application.LoadLevel ("Progress Report");
        }
}


 using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GB_Obj1 : MonoBehaviour {

        private GB_ObjectivesManager GB_ObjectivesManager;

        void Start () {
                GB_ObjectivesManager = GameObject.Find("GB_Objec
        }
        /**
         * Put times in everything
         * Main menu, selection of environment
         * Progress Report, pag natapos yung simulation
         * STOP
         * - Do the others
         * MONDAY - Documentation
         *         - mga pinabago nung proposal
         *         - Chapter 1 and 5
         *         - 6,7,8 lagyan ng unting laman
         * TUESDAY - WORST CASE
         */

        void OnTriggerEnter() {
                Debug.Log ("Player Has Entered Brick Building");

                GB_ObjectivesManager.objective1 = false;
                GB_ObjectivesManager.objective2 = true;
                GB_ObjectivesManager.objective3 = false;
                GB_ObjectivesManager.objective4 = false;
                GB_ObjectivesManager.objective5 = false;
                GB_ObjectivesManager.objective6 = false;
                GB_ObjectivesManager.objective7 = false;
                GB_ObjectivesManager.objective8 = false;
                GB_ObjectivesManager.objective9 = false;
                GB_ObjectivesManager.objective10 = false;

                BoxCollider nextTarget = GameObject.Find ("Objec
                nextTarget.enabled = true;

                Destroy(gameObject);
        }
}


 using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GB_Obj2 : MonoBehaviour {

        private GB_ObjectivesManager GB_ObjectivesManager;

        void Start () {
                GB_ObjectivesManager = GameObject.Find("GB_Objec
        }

        void OnTriggerEnter() {
                GB_ObjectivesManager.objective1 = false;
                GB_ObjectivesManager.objective2 = false;
                GB_ObjectivesManager.objective3 = true;
                GB_ObjectivesManager.objective4 = false;
                GB_ObjectivesManager.objective5 = false;
                GB_ObjectivesManager.objective6 = false;
                GB_ObjectivesManager.objective7 = false;
                GB_ObjectivesManager.objective8 = false;
                GB_ObjectivesManager.objective9 = false;
                GB_ObjectivesManager.objective10 = false;

                BoxCollider nextTarget = GameObject.Find ("Objec
                nextTarget.enabled = true;
                Destroy(gameObject);
        }

}

 using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GB_Obj3 : MonoBehaviour {

        private GB_ObjectivesManager GB_ObjectivesManager;

        void Start () {
                GB_ObjectivesManager = GameObject.Find("GB_Objec
        }

        void OnTriggerEnter() {
                GB_ObjectivesManager.objective1 = false;
                GB_ObjectivesManager.objective2 = false;
                GB_ObjectivesManager.objective3 = false;
                GB_ObjectivesManager.objective4 = true;
                GB_ObjectivesManager.objective5 = false;
                GB_ObjectivesManager.objective6 = false;
                GB_ObjectivesManager.objective7 = false;
                GB_ObjectivesManager.objective8 = false;
                GB_ObjectivesManager.objective9 = false;
                GB_ObjectivesManager.objective10 = false;

                BoxCollider nextTarget = GameObject.Find ("Objective4").GetComponent<BoxCollider>();
                nextTarget.enabled = true;

                //To prevent player from going to elevator again when objective reached.
                GameObject.Find("Elevator_Button").GetComponent<MeshCollider>().enabled = false;

                Destroy(gameObject);
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GB_Obj4 : MonoBehaviour {

        private GB_ObjectivesManager GB_ObjectivesManager;

        void Start () {
                GB_ObjectivesManager = GameObject.Find("GB_ObjectivesCanvas").GetComponent<GB_ObjectivesManager>();
        }

        void OnTriggerEnter() {
                Debug.Log ("Player Has Entered Elevator");

                GB_ObjectivesManager.objective1 = false;
                GB_ObjectivesManager.objective2 = false;
                GB_ObjectivesManager.objective3 = false;
                GB_ObjectivesManager.objective4 = false;
                GB_ObjectivesManager.objective5 = true;
                GB_ObjectivesManager.objective6 = false;
                GB_ObjectivesManager.objective7 = false;
                GB_ObjectivesManager.objective8 = false;
                GB_ObjectivesManager.objective9 = false;
                GB_ObjectivesManager.objective10 = false;

                Destroy(gameObject);
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GB_Obj5 : MonoBehaviour {

        private GB_ObjectivesManager GB_ObjectivesManager;

        void Start () {
                GB_ObjectivesManager = GameObject.Find("GB_ObjectivesCanvas").GetComponent<GB_ObjectivesManager>();
        }

        public void SwitchToObj6() {
                Debug.Log ("Cat 1 Taken");

                GB_ObjectivesManager.objective1 = false;
                GB_ObjectivesManager.objective2 = false;
                GB_ObjectivesManager.objective3 = false;
                GB_ObjectivesManager.objective4 = false;
                GB_ObjectivesManager.objective5 = false;
                GB_ObjectivesManager.objective6 = true;
                GB_ObjectivesManager.objective7 = false;
                GB_ObjectivesManager.objective8 = false;
                GB_ObjectivesManager.objective9 = false;
                GB_ObjectivesManager.objective10 = false;

                MeshCollider nextTarget = GameObject.Find ("Cat2").GetComponent<MeshCollider>();
                nextTarget.enabled = true;

                Destroy(gameObject);
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GB_Obj6 : MonoBehaviour {

        private GB_ObjectivesManager GB_ObjectivesManager;
```

```
        void Start () {
                GB_ObjectivesManager = GameObject.Find("GB_Objec
        }

        public void SwitchToObj7() {
                Debug.Log ("Cat 2 Taken");

                GB_ObjectivesManager.objective1 = false;
                GB_ObjectivesManager.objective2 = false;
                GB_ObjectivesManager.objective3 = false;
                GB_ObjectivesManager.objective4 = false;
                GB_ObjectivesManager.objective5 = false;
                GB_ObjectivesManager.objective6 = false;
                GB_ObjectivesManager.objective7 = true;
                GB_ObjectivesManager.objective8 = false;
                GB_ObjectivesManager.objective9 = false;
                GB_ObjectivesManager.objective10 = false;

                MeshCollider nextTarget = GameObject.Find ("Cat3"
                nextTarget.enabled = true;

                Destroy(gameObject);
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GB_Obj7 : MonoBehaviour {

        private GB_ObjectivesManager GB_ObjectivesManager;

        void Start () {
                GB_ObjectivesManager = GameObject.Find("GB_Objec
        }

        public void SwitchToObj8() {
                Debug.Log ("Cat 3 Taken");

                GB_ObjectivesManager.objective1 = false;
                GB_ObjectivesManager.objective2 = false;
                GB_ObjectivesManager.objective3 = false;
                GB_ObjectivesManager.objective4 = false;
                GB_ObjectivesManager.objective5 = false;
                GB_ObjectivesManager.objective6 = false;
                GB_ObjectivesManager.objective7 = false;
                GB_ObjectivesManager.objective8 = true;
                GB_ObjectivesManager.objective9 = false;
                GB_ObjectivesManager.objective10 = false;

                MeshCollider nextTarget = GameObject.Find ("Cat4"
                nextTarget.enabled = true;

                Destroy(gameObject);
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GB_Obj8 : MonoBehaviour {

        private GB_ObjectivesManager GB_ObjectivesManager;

        void Start () {
                GB_ObjectivesManager = GameObject.Find("GB_Objec
        }

        public void SwitchToObj9() {
                Debug.Log ("Cat 4 Taken");

                GB_ObjectivesManager.objective1 = false;
                GB_ObjectivesManager.objective2 = false;
                GB_ObjectivesManager.objective3 = false;
                GB_ObjectivesManager.objective4 = false;
                GB_ObjectivesManager.objective5 = false;
                GB_ObjectivesManager.objective6 = false;
                GB_ObjectivesManager.objective7 = false;
                GB_ObjectivesManager.objective8 = false;
                GB_ObjectivesManager.objective9 = true;
                GB_ObjectivesManager.objective10 = false;

                MeshCollider nextTarget = GameObject.Find ("CatF
                nextTarget.enabled = true;

                nextTarget = GameObject.Find ("CatMother").GetCor
                nextTarget.enabled = true;
                Destroy(gameObject);
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GB_Obj9 : MonoBehaviour {

        private GB_ObjectivesManager GB_ObjectivesManager;

        void Start () {
                GB_ObjectivesManager = GameObject.Find("GB_ObjectivesCanvas").GetComponent<GB_ObjectivesManager>();
        }

        public void SwitchToObj10() {
                Debug.Log ("Cats Returned");

                GB_ObjectivesManager.objective1 = false;
                GB_ObjectivesManager.objective2 = false;
                GB_ObjectivesManager.objective3 = false;
                GB_ObjectivesManager.objective4 = false;
                GB_ObjectivesManager.objective5 = false;
                GB_ObjectivesManager.objective6 = false;
                GB_ObjectivesManager.objective7 = false;
                GB_ObjectivesManager.objective8 = false;
                GB_ObjectivesManager.objective9 = false;
                GB_ObjectivesManager.objective10 = true;

                MeshRenderer nextTarget = GameObject.Find ("InviCat1").GetComponent<MeshRenderer>();
                nextTarget.enabled = true;

                nextTarget = GameObject.Find ("InviCat2").GetComponent<MeshRenderer>();
                nextTarget.enabled = true;

                nextTarget = GameObject.Find ("InviCat3").GetComponent<MeshRenderer>();
                nextTarget.enabled = true;

                nextTarget = GameObject.Find ("InviCat4").GetComponent<MeshRenderer>();
                nextTarget.enabled = true;
        }

}

  using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
//using UnityEngine.VR;

//[RequireComponent(typeof(Text))]
public class GB_ObjectivesManager: MonoBehaviour {

        private Text textField;
        public Camera cam;

        //public int indexOfScene; //Binabato ng main menu yung index of scene. Para mala

        public bool objective1;
        public bool objective2;
        public bool objective3;
        public bool objective4;
        public bool objective5;
        public bool objective6;
        public bool objective7;
        public bool objective8;
        public bool objective9;
        public bool objective10;

        public float counter = 0;
        public bool timer;
        public static bool isStopped;

        public static float objective1_time = 0;
        public static float objective2_time = 0;
        public static float objective3_time = 0;
        public static float objective4_time = 0;
        public static float objective5_time = 0;
        public static float objective6_time = 0;
        public static float objective7_time = 0;
        public static float objective8_time = 0;
        public static float objective9_time = 0;
        public static float objective10_time = 0;

        public static string objective1_remark;
        public static string objective2_remark;
        public static string objective3_remark;
        public static string objective4_remark;
        public static string objective5_remark;
        public static string objective6_remark;
        public static string objective7_remark;
        public static string objective8_remark;
        public static string objective9_remark;
        public static string objective10_remark;

        // Use this for initialization
        void Start() {
                isStopped = false;
                objective1 = true;
                objective2 = false;
                objective3 = false;
                objective4 = false;
                objective5 = false;
                objective6 = false;
                objective7 = false;
                objective8 = false;
                objective8 = false;
                objective9 = false;
                objective10 = false;

                objective1_remark = "n/a";
                objective2_remark = "n/a";
                objective3_remark = "n/a";
                objective4_remark = "n/a";
                objective5_remark = "n/a";
                objective6_remark = "n/a";
                objective7_remark = "n/a";
                objective8_remark = "n/a";
                objective9_remark = "n/a";
                objective10_remark = "n/a";

                textField = GetComponent<Text>();
                timer = true; //Para pag nagrun siya. True kaagad

                if (cam == null) {
                        cam = Camera.main;
                }

                if (cam != null) {
                        // Tie this to the camera, and do not kee
                        transform.SetParent(cam.GetComponent<Tran
                }

                //Debug.Log ("Index: " + controller.indexOfScene)
        }

        // Update is called once per frame
        void Update () {
                //Debug.Log ("Index Of Scene: " + controller.inde

                //int upToObjectives =


                if (timer) {
                        counter += Time.deltaTime;
                        //Debug.Log ("Time:" + counter);
                } else {
                        counter = counter;
                        Debug.Log ("Stop");
                }

                /**
                 *  Objective n's time will be recorded at object
                 */

                if (objective1) {
                        textField.text = "Enter the brick buildin
                        objective1 = false;
                }

                if (objective2) {
                        objective1_time = counter;
                        textField.text = "Enter the elevator";
                        objective1_remark = "Done";
                        objective2 = false;
                }

                if (objective3) {
                        objective2_time = counter;
                        textField.text = "Go to the 13th floor";
                        objective2_remark = "Done";
                        objective3 = false;
                }
                if (objective4) {
                        objective3_time = counter;
                        textField.text = "Enter the glass bridge'
                        objective3_remark = "Done";
                        objective4 = false;
                }
                if (objective5) {
                        objective4_time = counter;
                        textField.text = "Save cat 1";
                        objective4_remark = "Done";
                        objective5 = false;
                }
                if (objective6) {
                        objective5_time = counter;
                        textField.text = "Save cat 2";
                        objective5_remark = "Done";
                        objective6 = false;
                }
                if (objective7) {
                        objective6_time = counter;
                        textField.text = "Save cat 3";
                        objective6_remark = "Done";
                        objective7 = false;
                }
                if (objective8) {
                        objective7_time = counter;
                        textField.text = "Save cat 4";
                        objective7_remark = "Done";
                        objective8 = false;
                }
                if (objective9) {
                        objective8_time = counter;
                        textField.text = "Return cats to parent cats";
                        objective8_remark = "Done";
                        objective9 = false;
                }
                if (objective10) {
                        objective9_time = counter;
                        textField.text = "LEVEL FINISHED";
                        objective9_remark = "Done";
                        objective10 = false;

                        //SceneManager.LoadScene (4);
                        Application.LoadLevel ("Progress Report");
                }
        }

        public void EndSimulation (){
                isStopped = true;
                Application.LoadLevel ("Progress Report");
        }
}
```

```
  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class h_enter : MonoBehaviour {
        public GameObject hallelujahSource;
        public GameObject applauseSource;
        public GameObject JethroLight;
        public ParticleSystem fireworks;

        // Use this for initialization
        void Start () {
                hallelujahSource = GameObject.Find("hallelujahSource");
                applauseSource = GameObject.Find("applauseSource");
                JethroLight = GameObject.Find("JethroLight");
        }

        void OnTriggerEnter() {
                hallelujahSource.GetComponent<AudioSource> ().Play ();
                applauseSource.GetComponent<AudioSource> ().Play ();
                fireworks.Play ();
                GetComponent<BoxCollider> ().enabled = false;
                JethroLight.GetComponent<Light> ().enabled = true;
                Debug.Log ("Should have played sound");
        }
}
```

```
  using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class h_exit : MonoBehaviour {
        public GameObject hallelujahSource;
        public GameObject applauseSource;
        public GameObject JethroLight;
        public ParticleSystem fireworks;

        // Use this for initialization
        void Start () {
                hallelujahSource = GameObject.Find("hallelujahSource");
                applauseSource = GameObject.Find("applauseSource");
                JethroLight = GameObject.Find("JethroLight");
        }

        void OnTriggerEnter() {
                hallelujahSource.GetComponent<AudioSource> ().Sto
                applauseSource.GetComponent<AudioSource> ().Stop
                fireworks.Stop ();
                JethroLight.GetComponent<Light> ().enabled = fals
                GameObject.Find("hallelujahTriggerEnter").GetCom
        }
}
```

```
  using System.Collections;
using UnityEngine;

public class CharacterLookWalk : MonoBehaviour {

        // VR  Main Camera
        public Transform vrCamera;
        // How fast to move
        public static float speed = 5.4f;
        //Should I move forward or not
        public bool moveForward;
        //CharacterController script
        private CharacterController cc;

        // Use this for initialization
        void Start () {
                //Find the CharacterController
                cc = GetComponent<CharacterController>();
        }

        // Update is called once per frame
        void Update () {

                // If it is in contact with the ground? Not sure
                if (cc.isGrounded) {
                        // If left click
                        //if (Input.GetMouseButton(1) || Input.G
                        if (Input.GetMouseButton(0)) {
                                // Move forward
                                moveForward = true;
                        } else {
                                // Stop moving
                                moveForward = false;
                        }

                        // Check to see if I should move
                        if (moveForward) {
                                // Fine the forward direction
                                Vector3 forward = vrCamera.Transf
                                // Tell CharacterController to mo
                                cc.SimpleMove (forward * speed);
                        }
                }else {
                        // In the case of falling
                        Vector3 downward = vrCamera.TransformDir
                        cc.SimpleMove (downward * speed);
                }
        }
}
```

```
  using UnityEngine;
using System.Collections;

[RequireComponent(typeof(Rigidbody))]
[RequireComponent(typeof(CapsuleCollider))]
public class VRMoveController : MonoBehaviour
{
        public Camera cam;
        public float groundCheckDistance = 0.01f; // distance for
        public float stickToGroundHelperDistance = 0.5f; // stops
        public float slowDownRate = 2000f; // rate at which the c
        private Rigidbody m_RigidBody;
        private CapsuleCollider m_Capsule;
        private float m_YRotation;
        private Vector3 groundContactNormal;
        public float CurrentTargetSpeed = 0f;
    private Vector3 spawnPoint;

        public Vector3 Velocity {
                get { return m_RigidBody.velocity; }
        }

        private void Start ()
        {
                m_RigidBody = GetComponent<Rigidbody> ();
                m_Capsule = GetComponent<CapsuleCollider> ();
        spawnPoint = transform.position;
        }

        private void FixedUpdate ()
        {
```

```
		Vector2 input = GetInput ();

		//if (input.x == 0 || input.y == 1) {

		//	m_RigidBody.constraints = RigidbodyConstraints.FreezePosition;

		//} else {

		//	m_RigidBody.constraints = RigidbodyConstraints.None;
		//	m_RigidBody.constraints = RigidbodyConstraints.FreezeRotation;

			GroundCheck ();
			if ((Mathf.Abs (input.x) > float.Epsilon || Mathf.Abs (input.y) > float.Epsilon)) {
				//Debug.Log("got input "+input);
				// always move along the camera forward as it is the direction that it being aimed at
				Vector3 desiredMove = cam.transform.forward * input.y + cam.transform.right * input.x;
				desiredMove = Vector3.ProjectOnPlane (desiredMove, groundContactNormal).normalized;

				desiredMove.x = desiredMove.x * CurrentTargetSpeed;
				desiredMove.z = desiredMove.z * CurrentTargetSpeed;
				desiredMove.y = desiredMove.y * CurrentTargetSpeed;
				if (m_RigidBody.velocity.sqrMagnitude <
					(CurrentTargetSpeed * CurrentTargetSpeed)) {
					m_RigidBody.AddForce (desiredMove, ForceMode.Impulse);
				}
				StickToGroundHelper ();
			}
	if (transform.position.y < -5f) {
	    transform.position = spawnPoint;
	}
		//}
	}

	private void StickToGroundHelper ()
	{
		RaycastHit hitInfo;
		if (Physics.SphereCast (transform.position, m_Capsule.radius, Vector3.down, out hitInfo,
					((m_Capsule.height / 2f) - m_Capsule.radius) +
			stickToGroundHelperDistance)) {
			if (Mathf.Abs (Vector3.Angle (hitInfo.normal, Vector3.up)) < 85f) {
				m_RigidBody.velocity = Vector3.ProjectOnPlane (m_RigidBody.velocity, hitInfo.normal);
			}
		}
	}
	/// sphere cast down just beyond the bottom of the capsule to see if the capsule is colliding round the bott
	private void GroundCheck()
	{

		RaycastHit hitInfo;
		if (Physics.SphereCast(transform.position, m_Capsule.radius, Vector3.down, out hitInfo,
				((m_Capsule.height/2f) - m_Capsule.radius) + groundCheckDistance))
		{
			groundContactNormal = hitInfo.normal;
		}
		else
		{
			groundContactNormal = Vector3.up;
		}
	}

	private Vector2 GetInput ()
	{
		float x,y;
		if (Input.GetMouseButton(0)){
			x = 0;
			y = 1;
		} else {
			x = Input.GetAxis("Horizontal");
			y = Input.GetAxis("Vertical");
		}
		Vector2 input = new Vector2(x,y);
		return input;
	}
}



/**

using UnityEngine;
using System.Collections;

[RequireComponent(typeof(Rigidbody))]
[RequireComponent(typeof(CapsuleCollider))]
public class VRMoveController : MonoBehaviour
{
	public Camera cam;
	public float groundCheckDistance = 0.01f; // distance for checking if the controller is grounded ( 0.01f se
	public float stickToGroundHelperDistance = 0.5f; // stops the character
	public float slowDownRate = 20f; // rate at which the controller comes to a stop when there is no input
	private Rigidbody m_RigidBody;
```

```
	private CapsuleCollider m_Capsule;
	private float m_YRotation;
	private Vector3 groundContactNormal;
	public float CurrentTargetSpeed = 0f;
	private Vector3 spawnPoint;

	public Vector3 Velocity {
		get { return m_RigidBody.velocity; }
	}
	private void Start ()
	{
		m_RigidBody = GetComponent<Rigidbody> ();
		m_Capsule = GetComponent<CapsuleCollider> ();
		spawnPoint = transform.position;
	}

	private void FixedUpdate ()
	{
		Vector2 input = GetInput ();

		//if (input.x == 0 || input.y == 1) {

		//	m_RigidBody.constraints = RigidbodyConst

		//} else {

		//	m_RigidBody.constraints = RigidbodyConst
		//	m_RigidBody.constraints = RigidbodyConst

			GroundCheck ();
			if ((Mathf.Abs (input.x) > float.Epsilon
				//Debug.Log("got input "+input);
				// always move along the camera f
				Vector3 desiredMove = cam.transfo
				desiredMove = Vector3.ProjectOnPl

				desiredMove.x = desiredMove.x * C
				desiredMove.z = desiredMove.z * C
				desiredMove.y = desiredMove.y * C
				if (m_RigidBody.velocity.sqrMagn
					(CurrentTargetSpeed * Cu
					m_RigidBody.AddForce (des
				}
				StickToGroundHelper ();
			}

	if (transform.position.y < -5f) {
	    transform.position = spawnPoint;
	}
		//}
	}

	private void StickToGroundHelper ()
	{
		RaycastHit hitInfo;
		if (Physics.SphereCast (transform.position, m_Ca
					((m_Capsule.height
			stickToGroundHelperDistance)) {
			if (Mathf.Abs (Vector3.Angle (hitInfo.nor
				m_RigidBody.velocity = Vector3.Pr
			}
		}
	}
	/// sphere cast down just beyond the bottom of the capsul
	private void GroundCheck()
	{

		RaycastHit hitInfo;
		if (Physics.SphereCast(transform.position, m_Cap
				((m_Capsule.height/2f) - m
		{
			groundContactNormal = hitInfo.normal;
		}
		else
		{
			groundContactNormal = Vector3.up;
		}
	}

	private Vector2 GetInput ()
	{
		float x,y;
		if (Input.GetMouseButton(0)){
			x = 0;
			y = 1;
		} else {
			x = Input.GetAxis("Horizontal");
			y = Input.GetAxis("Vertical");
		}
```

```
                Vector2 input = new Vector2(x,y);
                return input;
        }
}

**/


  using System.Collections;
using UnityEngine;
public class YouMove : MonoBehaviour {
        private float speed = 3f;
        private float jumpforce = 20f;
        private float  gravity = 30f;
        private Vector3 moveDir = Vector3.zero;

        void Start () {

        }


        void Update () {
                CharacterController Controller = gameObject.GetComponent<CharacterController >();

                if (Controller.isGrounded) {
                        moveDir = new Vector3 (Input.GetAxis ("Horizontal"), 0, Input.GetAxis ("Vertical"));

                        moveDir = transform.TransformDirection (moveDir);

                        moveDir *= speed;

                        if (Input.GetButtonDown ("Jump")) {
                                moveDir.y = jumpforce;
                        }
                }

                moveDir.y -= 4 * gravity * Time.deltaTime;
                Controller.Move (moveDir * Time.deltaTime);
        }
}


  using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
//using UnityEngine.VR;

public class canvasManager : MonoBehaviour {

        public Canvas CE_Progress;
        public Canvas FE_Progress;
        public Canvas GB_Progress;
        public Button CE_exit;
        public Button CE_menu;
        public Button FE_exit;
        public Button FE_menu;
        public Button GB_exit;
        public Button GB_menu;
        float minutes = 0;
        float seconds = 0;
        string exactTime;
        //public int indexOfScene;

        //======================functions for canvas switching======================//

        //function to make Construction Elevator Canvas visible
        public void CE_ProgressOn() {
                CE_Progress.enabled = true;
                FE_Progress.enabled = false;
                GB_Progress.enabled = false;
        }

        //function to make Fire Escape Canvas visible
        public void FE_ProgressOn() {
                FE_Progress.enabled = true;
                CE_Progress.enabled = false;
                GB_Progress.enabled = false;
        }

        //function to make Glass Bridge Canvas visible
        public void GB_ProgressOn() {
                GB_Progress.enabled = true;
                FE_Progress.enabled = false;
                CE_Progress.enabled = false;
        }

        public void exitMenuListener(){
                Button btn1 = CE_exit.GetComponent<Button> ();
                btn1.onClick.AddListener (exitApp);

                Button btn2 = FE_exit.GetComponent<Button> ();
                btn2.onClick.AddListener (exitApp);

                Button btn3 = GB_exit.GetComponent<Button> ();
                btn3.onClick.AddListener (exitApp);

                Button btn4 = CE_menu.GetComponent<Button> ();
                btn4.onClick.AddListener (returnToMenu);

                Button btn5 = FE_menu.GetComponent<Button> ();
                btn5.onClick.AddListener (returnToMenu);

                Button btn6 = GB_menu.GetComponent<Button> ();
                btn6.onClick.AddListener (returnToMenu);
        }

        void exitApp(){
                Application.Quit();
        }

        void returnToMenu(){
                Application.LoadLevel ("Menu");
        }

        void convertSecondstoMinutes (float timer){
                int minutes = Mathf.FloorToInt(timer / 60F);
                int seconds = Mathf.FloorToInt(timer − minutes *
                exactTime = string.Format("{0:0}:{1:00}", minutes
        }

        // Use this for initialization
        void Start () {
                //VRSettings.enabled = false;
                Debug.Log ("Index: " + controller.indexOfScene);


                if (controller.indexOfScene == 1) { //Glass Bridg
                        GB_ProgressOn ();
                        Text remark1 = GameObject.Find ("GB_Outco
                        remark1.text = GB_ObjectivesManager.objec

                        Text time1 = GameObject.Find ("GB_Time1")
                        convertSecondstoMinutes (GB_ObjectivesMan
                        time1.text = exactTime;

                        Text remark2 = GameObject.Find ("GB_Outco
                        remark2.text = GB_ObjectivesManager.objec

                        Text time2 = GameObject.Find ("GB_Time2")
                        convertSecondstoMinutes (GB_ObjectivesMan
                        time2.text = exactTime;

                        Text remark3 = GameObject.Find ("GB_Outco
                        remark3.text = GB_ObjectivesManager.objec

                        Text time3 = GameObject.Find ("GB_Time3")
                        convertSecondstoMinutes (GB_ObjectivesMan
                        time3.text = exactTime;

                        Text remark4 = GameObject.Find ("GB_Outco
                        remark4.text = GB_ObjectivesManager.objec

                        Text time4 = GameObject.Find ("GB_Time4")
                        convertSecondstoMinutes (GB_ObjectivesMan
                        time4.text = exactTime;

                        Text remark5 = GameObject.Find ("GB_Outco
                        remark5.text = GB_ObjectivesManager.objec

                        Text time5 = GameObject.Find ("GB_Time5")
                        convertSecondstoMinutes (GB_ObjectivesMan
                        time5.text = exactTime;

                        Text remark6 = GameObject.Find ("GB_Outco
                        remark6.text = GB_ObjectivesManager.objec

                        Text time6 = GameObject.Find ("GB_Time6")
                        convertSecondstoMinutes (GB_ObjectivesMan
                        time6.text = exactTime;

                        Text remark7 = GameObject.Find ("GB_Outco
                        remark7.text = GB_ObjectivesManager.objec

                        Text time7 = GameObject.Find ("GB_Time7")
                        convertSecondstoMinutes (GB_ObjectivesMan
                        time7.text = exactTime;

                        Text remark8 = GameObject.Find ("GB_Outco
                        remark8.text = GB_ObjectivesManager.objec

                        Text time8 = GameObject.Find ("GB_Time8")
                        convertSecondstoMinutes (GB_ObjectivesMan
                        time8.text = exactTime;
                Text remark9 = GameObject.Find ("GB_Outcome9").GetComponent<Text> ();
                remark9.text = GB_ObjectivesManager.objective9_remark;

                Text time9 = GameObject.Find ("GB_Time9").GetComponent<Text> ();
                convertSecondstoMinutes (GB_ObjectivesManager.objective9_time);
                time9.text = exactTime;

                Text isStoppedText = GameObject.Find ("GB_isStopped").GetComponent<Text> ();
                if (GB_ObjectivesManager.isStopped) {
                        isStoppedText.text = "Simulation Aborted: Yes";
                } else {
                        isStoppedText.text = "Simulation Aborted: No";
                }
        }

        if (controller.indexOfScene == 2) { //Fire Escape
                FE_ProgressOn ();
                Text remark1 = GameObject.Find ("FE_Outcome1").GetComponent<Text> ();
                remark1.text = FE_ObjectivesManager.objective1_remark;

                Text time1 = GameObject.Find ("FE_Time1").GetComponent<Text> ();
                convertSecondstoMinutes (FE_ObjectivesManager.objective1_time);
                time1.text = exactTime;

                Text remark2 = GameObject.Find ("FE_Outcome2").GetComponent<Text> ();
                remark2.text = FE_ObjectivesManager.objective2_remark;

                Text time2 = GameObject.Find ("FE_Time2").GetComponent<Text> ();
                convertSecondstoMinutes (FE_ObjectivesManager.objective2_time);
                time2.text = exactTime;

                Text remark3 = GameObject.Find ("FE_Outcome3").GetComponent<Text> ();
                remark3.text = FE_ObjectivesManager.objective3_remark;

                Text time3 = GameObject.Find ("FE_Time3").GetComponent<Text> ();
                convertSecondstoMinutes (FE_ObjectivesManager.objective3_time);
                time3.text = exactTime;

                Text remark4 = GameObject.Find ("FE_Outcome4").GetComponent<Text> ();
                remark4.text = FE_ObjectivesManager.objective4_remark;

                Text time4 = GameObject.Find ("FE_Time4").GetComponent<Text> ();
                convertSecondstoMinutes (FE_ObjectivesManager.objective4_time);
                time4.text = exactTime;

                Text remark5 = GameObject.Find ("FE_Outcome5").GetComponent<Text> ();
                remark5.text = FE_ObjectivesManager.objective5_remark;

                Text time5 = GameObject.Find ("FE_Time5").GetComponent<Text> ();
                convertSecondstoMinutes (FE_ObjectivesManager.objective5_time);
                time5.text = exactTime;

                Text remark6 = GameObject.Find ("FE_Outcome6").GetComponent<Text> ();
                remark6.text = FE_ObjectivesManager.objective6_remark;

                Text time6 = GameObject.Find ("FE_Time6").GetComponent<Text> ();
                convertSecondstoMinutes (FE_ObjectivesManager.objective6_time);
                time6.text = exactTime;

                Text remark7 = GameObject.Find ("FE_Outcome7").GetComponent<Text> ();
                remark7.text = FE_ObjectivesManager.objective7_remark;

                Text time7 = GameObject.Find ("FE_Time7").GetComponent<Text> ();
                convertSecondstoMinutes (FE_ObjectivesManager.objective7_time);
                time7.text = exactTime;

                Text remark8 = GameObject.Find ("FE_Outcome8").GetComponent<Text> ();
                remark8.text = FE_ObjectivesManager.objective8_remark;

                Text time8 = GameObject.Find ("FE_Time8").GetComponent<Text> ();
                convertSecondstoMinutes (FE_ObjectivesManager.objective8_time);
                time8.text = exactTime;

                Text isStoppedText = GameObject.Find ("FE_isStopped").GetComponent<Text> ();
                if (FE_ObjectivesManager.isStopped) {
                        isStoppedText.text = "Simulation Aborted: Yes";
                } else {
                        isStoppedText.text = "Simulation Aborted: No";
                }
        }
        if (controller.indexOfScene == 3) {//Construction Elevator
                CE_ProgressOn ();
                Text remark1 = GameObject.Find ("CE_Outcome1").GetComponent<Text> ();
                remark1.text = CE_ObjectivesManager.objective1_remark;

                Text time1 = GameObject.Find ("CE_Time1").GetComponent<Text> ();
                convertSecondstoMinutes (CE_ObjectivesManager.objective1_time);
                time1.text = exactTime;

                Text remark2 = GameObject.Find ("CE_Outcome2").GetComponent<Text> ();
                remark2.text = CE_ObjectivesManager.objective2_remark;
```

```
                Text time2 = GameObject.Find ("CE_Time2")
                convertSecondstoMinutes (CE_ObjectivesMar
                time2.text = exactTime;

                Text remark3 = GameObject.Find ("CE_Outco
                remark3.text = CE_ObjectivesManager.objec

                Text time3 = GameObject.Find ("CE_Time3")
                convertSecondstoMinutes (CE_ObjectivesMar
                time3.text = exactTime;

                Text remark4 = GameObject.Find ("CE_Outco
                remark4.text = CE_ObjectivesManager.objec

                Text time4 = GameObject.Find ("CE_Time4")
                convertSecondstoMinutes (CE_ObjectivesMar
                time4.text = exactTime;

                Text remark5 = GameObject.Find ("CE_Outco
                remark5.text = CE_ObjectivesManager.objec

                Text time5 = GameObject.Find ("CE_Time5")
                convertSecondstoMinutes (CE_ObjectivesMar
                time5.text = exactTime;

                Text remark6 = GameObject.Find ("CE_Outco
                remark6.text = CE_ObjectivesManager.objec

                Text time6 = GameObject.Find ("CE_Time6")
                convertSecondstoMinutes (CE_ObjectivesMar
                time6.text = exactTime;

                Text remark7 = GameObject.Find ("CE_Outco
                remark7.text = CE_ObjectivesManager.objec

                Text time7 = GameObject.Find ("CE_Time7")
                convertSecondstoMinutes (CE_ObjectivesMar
                time7.text = exactTime;

                Text isStoppedText = GameObject.Find ("C
                if (CE_ObjectivesManager.isStopped) {
                        isStoppedText.text = "Simulation
                } else {
                        isStoppedText.text = "Simulation
                }
        }
        exitMenuListener();
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
//using UnityEngine.VR;
using UnityEngine.SceneManagement;

public class controller : MonoBehaviour {
        public Button something;
        public Dropdown VirtualEnvironment;
        public static int indexOfScene = 0; //1-GB, 2-FE, 3-CE
        // Use this for initialization
        void Start () {
                //disableVr ();
                indexOfScene = 0; //THIS MIGHT AFFECT SOMETHING I
                Button btn = something.GetComponent<Button>     (
                btn.onClick.AddListener (someFunction);
        }
        void Awake() {
                DontDestroyOnLoad(transform.gameObject);
        }

        // Update is called once per frame
        void Update () {

        }

        public void someFunction(){
                switch(VirtualEnvironment.value){
                case 0:{
                        indexOfScene = 3;
                        Debug.Log ("You selected Construc
                        //enableVr ();
                Application.LoadLevel ("Construction Elevator");
                break;
            }
            case 1:{
                indexOfScene = 2;
                Debug.Log ("You selected Fire Escape");
                //enableVr ();
                Application.LoadLevel ("Fire Escape");
                break;
            }
            case 2:{
                indexOfScene = 1;
                Debug.Log ("You selected Glass Bridge");
                //enableVr ();
                Application.LoadLevel ("Glass Bridge");
                break;
            }
        }

        //enableVr ();  // IF HINDI GUMANA TO, PUT IT IN THE START OF THE LOADING Other scene
        //Application.LoadLevel ("Construction Elevator");
        //WHEN YOU PROCEED TO PROGRESS REPORT || STOP BUTTON || TIMER REACHED, disable VR *****
}
```

```
public void QuitApplication(){
        Application.Quit ();
}

/*
IEnumerator LoadDevice(string newDevice, bool enable)
{
        VRSettings.LoadDeviceByName(newDevice);
        yield return null;
        VRSettings.enabled = enable;
}

void enableVr() //gamitin
{
        StartCoroutine(LoadDevice("cardboard", true));
}

void disableVr()
{
        StartCoroutine(LoadDevice("", false));
}
*/
}
```

XI. Acknowledgment

Finally, tapos na ang SP ko. As of now yung inaalala ko is yung grade ko sa isang subject na deliks ako. Piling ko yun pa yung maghahadlang sa 'kin sa pag-graduate on time.

Pero back to the acknowledgement, first of all, I would to thank God for giving the opportunity to study in the best school (naks) in the Philippines. I thank Him for everything that has happened to me, whether good or bad. I know that everything that happens is according to His divine will.

I would like to thank my parents who supported me all throughout. There were times na hindi talaga kinaya. Gusto ko nang sumuko talaga but they were there to comfort me and support me with any decision I make. I wouldn't be where I am today if it wasn't for them. Thank you for guiding me and loving me through the years. I love both from the bottom of my heart.

To my siblings who also supported me, gave me hugs and prayed for me. Thank you very much. Malilibre ko na kayo oh!

To Sir Ignacio, I owe you a lot Sir. Seryoso Sir! Akala ko na talaga na hindi ako makakapagdefend that day but you pushed me to continue and developed an effective strategy haha. Super hassle, wala na po akong tulog nung araw na yun. Pero it was all well worth it, thanks to you! Cabalen. Nakapag-propose and defend in one semester?? Whaat. Iba ka talaga, Sir! Cabalen.

To the VRET team, as I am writing this acknowledgement right now, you are here with me at my home. Seryoso. We shared the same struggles. We experienced the same sleepless nights. But thanks to the guidance of our beloved Sir Ignacio, we were able to finish our SP's. Congratulations to us! We should eat at Cabalen some time.

To papi, grabe talaga yung support mo. Super na-aappreciate ko talaga. Kapag nasstress ako, salamat sa pakikinig sa akin. Salamat din sa napakaraming prayers at verses na binigay mo. One year na oh. I gat u (Y) (Y)

And finally, to the Kumpanero Light team. Thank you for the good times we shared throughout this college life. Hindi ko ata mapapasa yung mga ibang subjects kung hindi ko kayo naging katropa. Sana magkita pa rin tayo every now and then kahit hindi na magkakaklase. ¡3