LAUREA
UNIVERSITY OF APPLIED SCIENCES
Together we are stronger

Creating a training game for Urban Search and Rescue personnel

Tiira, Ville

2016 Leppävaara

Laurea University of Applied Sciences
Leppävaara

Creating a training game for Urban Search and Rescue Personnel

Ville Tiira
Degree Programme in BIT
Bachelor's Thesis
December, 2016

Laurea University of Applied Sciences
Degree Programme in Business Information Technology
Bachelor's Thesis

Abstract

Tiira, Ville

Creating a training game for Urban Search and Rescue

Year 2016 Pages 47

In today's world, where more and more people live in the cities, it becomes crucial that the people are prepared for disasters from both natural and human causes. Urban Search and Rescue is in a key role in saving lives when, for example, a building collapses, by helping to find and evacuate victims from the ruins and preventing further collapse. However, training for these situations is extremely difficult, as making the training realistic would require collapsing a building, which is both time consuming and expensive. To offer a cost-effective alternative, this thesis describes the research and development of a virtual training game aimed for Urban Search and Rescue personnel.

The thesis shortly introduces the multinational project INACHUS, in which Laurea University of Applied Sciences is taking part.The project is working to improve the tools and techniques used by Urban Search and Rescue personnel. The background of the target group and the benefits of using games for training purposes are described briefly and the methodology and tools used in the implementation are introduced.

The development process included a lot of research and testing as the game was built one functionality at a time, slowly expanding and improving the content, which was where the incremental life cycle was found the best alternative to work with. Meetings with the INACHUS representatives were held on a weekly basis, which allowed adjusting the focus point time to time and possibilities to showcase the current stage of the project and to plan for further implementations.

The end result of the thesis project is a prototype of a virtual training game, which allows the user to investigate a collapsed building and evacuate victims. Possibilities and limitations were researched to find out whether the chosen platform is a good choice for developing this type of game. As the game was not completed, suggestions for future implementations, that were thought of during the development process were also described.

Keywords: Unreal Engine, Game development, Urban Search and Rescue, Blender, INACHUS

Laurea-ammattikorkeakoulu **Tiivistelmä**
Degree Programme in Business Information Technology
Kandidaatin tutkielma

Tiira, Ville

Koulutuspelin kehitys urbaanille pelastushenkilökunnalle

Vuosi 2016 Sivumäärä 47

Nykymaailmassa, jossa yhä useampi asuu kaupungissa, tulee tärkeäksi, että ihmiset ovat valmistautuneita sekä luonnon, että ihmisen aiheuttamiin katastrofeihin. Urbaani pelastushenkilökunta on avainroolissa, esimerkiksi rakennuksen sortuessa, auttamalla löytämään ja evakuoimaan uhreja raunioista ja estämällä jatkosortumien tapahtumisen. Harjoittelu näitä tilanteita varten on tosin hyvin vaikeaa, sillä se vaatisi rakennuksen romahduttamisen, jotta harjoittelusta tulisi realistista, mikä on sekä kallista, että aikaa vievää. Tarjotakseen kustannustehoikkaan vaihtoehdon, tämä opinnäytetyö kuvaa urbaanille pelastuskunnalle suunnattua virtuaalisen koulutuspelin kehitystä.

Opinnäytetyö esittelee lyhyesti kansainvälisen INACHUS-projektin, johon Laurean ammattikorkeakoulu osallistuu. Projekti pyrkii parantamaan työkaluja ja tekniikoita urbaanin pelastushenkilökunnan käyttöön. Myös kohderyhmän taustaa ja pelien käyttämistä koulutustarkoituksiin kuvataan lyhyesti sekä työssä käytetty metodologia sekä työkalut esitellään.

Kehitysprosessi sisälsi paljon tutkimista ja kokeilua, sillä peli kehitettiin ominaisuus kerrallaan, hitaasti laajentuen ja parantaen sisältöä, missä inkrementaalinen elinkaari -metodologia todettiin parhaaksi vaihtoehdoksi. Tapaamisia INACHUS:n edustajien kanssa pidettiin viikoittain, mikä salli keskittymispisteen muuttamisen ajoittain, sekä antoi mahdollisuuden esitellä projektin senhetkistä tilannetta sekä seuraavien ominaisuuksien suunnittelun.

Opinnäytetyön lopputulos on virtuaalisen koulutuspelin prototyyppi, jossa käyttäjä voi tutkia romahtanutta rakennusta, sekä pelastaa uhreja. Työkalujen mahdollisuuksia ja rajoituksia tutkittiin, niiden soveltuvuuden arvioimiseksi. Koska peliä ei voitu viedä loppuun asti, jatkokehittämistä varten on annettu ehdotuksia, joita on kehittämisen aikana ajateltu.

Avainsanat: Unreal Engine, Pelinkehitys, Urbaani pelastushenkilökunta, Blender, INACHUS

Table of contents

Terms and abbreviations

3D	Three-dimensional form, an image with width, depth and height
Blueprint	A node editor inside Unreal Engine allowing scripting of game elements without a code language, such as C++
Trigger box	Actor used to cause an event to occur when interacted with, by the player or another object
UV mapping	Process in 3D modelling, in which a 2D image is applied on a 3D model for texturing purposes
Widget	Functionality in Unreal Engine used to create user interface components
C++	Programming language

1 Introduction

Every year, there are hundreds of earthquakes, which can cause buildings to collapse and people getting trapped under the wreckage. The earthquakes are measured using the Richter scale, a technique created by Charles Richter in 1934, which is based on the amplitudes and duration of the earthquake to calculate the magnitude (UPSeis 2007). Earthquakes above 6 in magnitude can cause a lot of damage to buildings in populated areas. Annually there are about a hundred earthquakes of this magnitude, according to UPSeis (UPSeis n.d.). Below is a map, showing the locations of earthquakes in the year 2016 until December 9th.

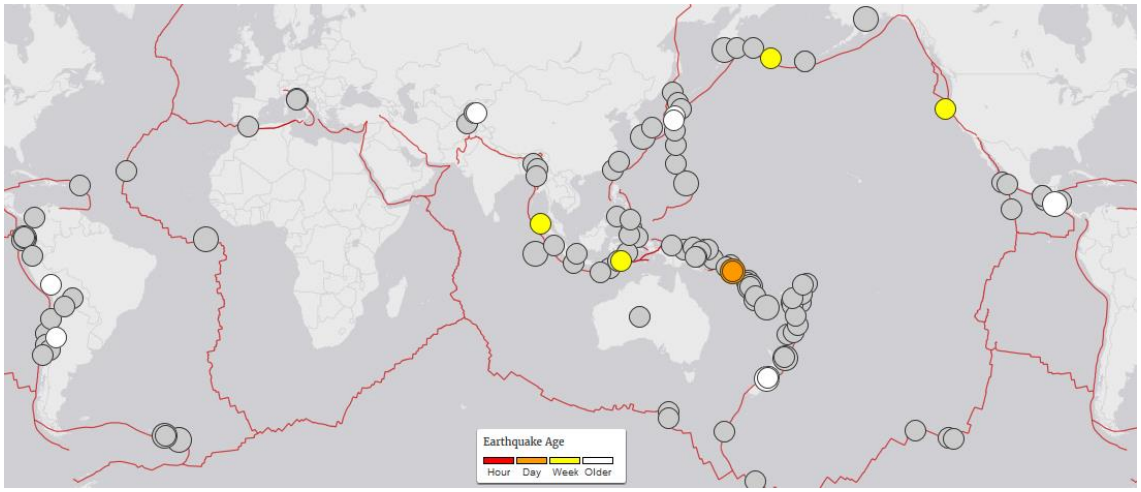

Figure 1: Map of earthquakes above 6 in magnitude over the year 2016. (USGS 2016)

In addition to earthquakes, there are other disasters, such as hurricanes, storms and terrorism that pose a threat to people living in cities. For this purpose, Urban Search and Rescue personnel are trained to work in these conditions to safely and effectively evacuate victims.

1.1 INACHUS

INACHUS is a multinational research and development project funded by the EU. Its objective is to research and develop tools to reduce the time required to respond, locate and rescue victims from collapsed buildings after a catastrophic event, such as an earthquake or a terrorist attack. Its primary goal is to develop various modules optimized and synchronized to make sure rapid and efficient decision making is possible in catastrophic events. Some of the modules that are developed in INACHUS are:

- Fast damage estimation from 3D airborne and ground based laser scans.
- Electromagnetic, vision, and chemical sensors for detection of trapped victims.
- Snake robot mechanism integrated with the sensors, to penetrate the rubble.
- The design of a communication network that enables the different modules to communicate whith each other quickly and safely.

Laurea University of Applied Sciences is participating in the project by developing a simulation module, which can be used to find most probable locations for air pockets and possibly survivors in the rubble and to determine the fastest and safest route to get to them. (INACHUS 2016)

In the beginning of the project, a small group of students, from Metropolia University of Applied Sciences, participated in the development by researching and creating content, such as the main menu and a hydration level progress bar, but were focused on investigating the workflow of importing collapse models from the 3D software Blender. (Skantz et al. 2016)

1.2 Research objectives and scope limitations

The assignment for the project was to research a virtual platform, for the purpose of creating a visual representation of a collapsed building and a training game for the Urban Search and Rescue personnel. The main focus was on finding out the capabilities using the chosen platform, to see whether it would be sufficient for this purpose and to develop the game as far as possible.

As the main focus of the project was to create as many functionalities as possible, visual aspects of the game were mostly untouched. Textures and special effects, such as smoke. were used at later stages to give the game a hint of realism and to make it more appealing to the user.

2 Training for rescue personnel

This section covers the background of the project. First by looking into rescue personnel training and then to existing virtual training tools and how these tools can be used to improve training.

2.1 On-site training

There are training facilities for firefighters around the world, such as the Uaill Training Centre in Cambuslang, Scotland. The training centre, which cost 22 million pounds to build, contains various types of buildings and scenarios, such as residential and industrial buildings and a motorway (Haagen 2013). However, the Urban Search and Rescue training is not fully included and as said on Haagen's website: "While Urban Search and Rescue training is increasingly vital, it is extremely difficult to realistically simulate" (Haagen 2016). For creating training environments that would be close to realistic, buildings need to be built and then collapsed, which is both time consuming and expensive.

2.2 Virtual training

Virtual learning is becoming more and more common as the technology improves. For example, Duolingo is a mobile application, which allows the user to learn a new language by the means of gamification. The game keeps track of the learning and once a lesson is learned, it rewards the user with points and allows them to progress further in their learning (Duolingo 2016). Gamification's benefits include making learning more fun, interesting, giving instant feedback and increasing motivation through friendly competition and sense of achievement (Pandey 2015).

Training in a virtual environment is already being used in U.S. military training to teach problem solving and leadership, train for war scenarios and improve attention span and teamwork (Ivankov 2015). Finland's military has been using a simulator to replace lectures and reduce costs and to provide a way to understand the big picture of operations. (Vainio 2016)

There are also some firefighting simulation games, such as FLAME-SIM, which emphasizes each firefighter's decision-making and communication in the field and allows the use of the same communication equipment as in the field, thus creating realism to the communication (FLAME-SIM 2013). Another example of a virtual training tool for firefighters is XVR, a tool used in training and assessing the commanders in a simulated scene, where they can investigate the scenes and make decisions on the best course of action. (XVR 2016)

However, there does not seem to be any virtual training games directed at the Urban Search and Rescue field, where training could be improved with a similar solution as used in the previous examples. Virtual training would not be sufficient alone, but the costs, safety and time management could be improved by sometimes training virtually instead of in the field. While the development of a virtual training tool may require funding, it would be reusable and distributed globally, where as on-site training will have only a set number of participants at a time and organizing it may have high costs.

3 Methodology

Planning was done mostly on a week-by-week basis, because it was difficult to plan for an exact end-result as the capabilities of the tools were unknown in the beginning. Due to this, using the incremental model was chosen as it allowed developing a new feature, testing it and finally showing the results in the next meeting. Each development cycle was therefore a week, during which possibilities for each functionality was researched, developed and tested.

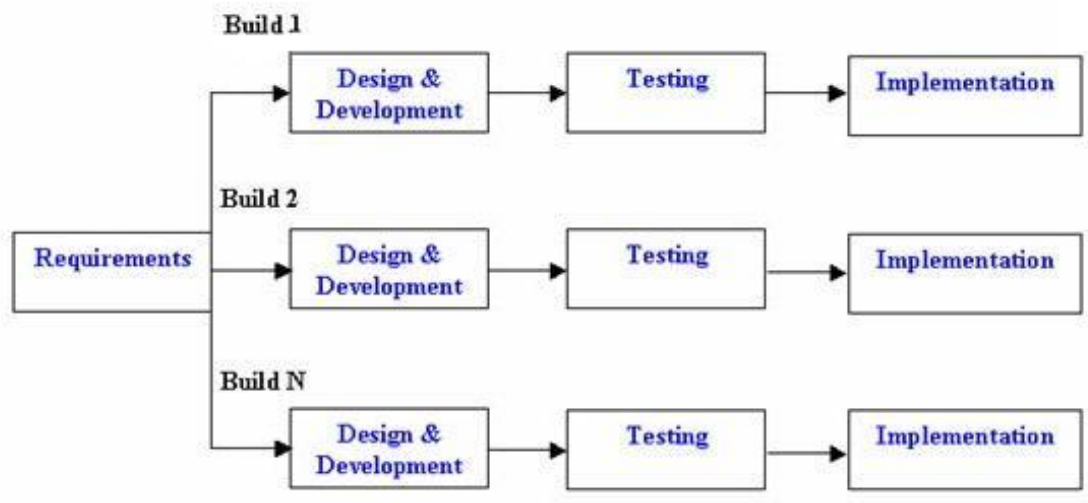

Figure 2: Incremental Life Cycle Model ((ISTQB, n.d.)

The advantages of using this model were the ability to gradually expand the functionality of a feature, without the need to finish it at once, allowing further development, where seen possible. (ISTQB, n.d.) For example, rescuing victims expanded from only finding victims to rescuing them by touch and later to rescuing them via a User interface widget.

4 Tools

The most important tools for this project were the game engine and the 3D modelling software. The first giving the basic tools and environment for building the game itself and the latter giving access to 3D models used in the game.

4.1 Blender

Blender is an open source software that is widely used for 3D modelling (Blender n.d.). The models used in the game are prepared and then imported from Blender. The tool was chosen as the buildings used in the game were built and simulated within Blender. Importing models from Blender into Unreal Engine is also well supported, which made the decision easier.

4.1.1 Bullet Constraint Builder

The building collapses were simulated with Bullet Constraint Builder, an add-on being developed by INACHUS, which allows creating structural depencies between building elements. With the add-on, it is possible to simulate realistic collapses, caused by earthquakes or explosions. (Kostack & Walter, 2015)

The collapsed building model used in the game, was INACHUS's real life case about Pyne Gould Corporation building in Christchurch, New Zealand, that collapsed in 2011 during an earthquake. The building was modelled and simulated using the Bullet Constraints Builder in Blender, by referencing the model sources of the PGC building's engineer, with results that came close to reality.

Figure 3: PGC building collapsed in 2011

Figure 4: Simulated building collapse model imported to Unreal Engine

4.2 Unreal Engine

Unreal Engine 4 is a game engine published by Epic Games. It is free to use until the gross product revenue reaches 3,000 USD per game per calendar quarter, after which, a 5% royalty will be paid to Epic Games (Epic games, 2016). It contains tools for creating visually appealing scenes and logic for games, and was chosen for its easy to use User Interface and the lack of need for coding. Another reason for selecting Unreal engine is the ability to test the game from the very beginning of the project. Majority of the project is done by using the in-built functionalities of the engine, such as the blueprints and physics.

4.2.1 Considered alternatives

First alternative for Unreal Engine that was considered was Blender's game engine as it was already used for the 3D modelling, but it was discarded as the visual quality and usability were not at the same level with Unreal Engine. Unity was also considered, but was not chosen because of the need to write C++ for basically everything in the game.

4.2.2 Blueprint visual scripting

The Blueprints Visual Scripting system in Unreal Engine allows creating gameplay elements from within Unreal Editor by using a node-based interface without the need of coding in a programming language, such as C++ or Java.

Fundamentally, it works in a similar way as basic programming, with functions like if-statements and for-loops. For example, to print text to the screen by walking into a specific spot in the level, you can use a to detect when the player enters the designated area. Then in a blueprint, add an event to start when the trigger box is entered and connect it to a print text function and give it a value.

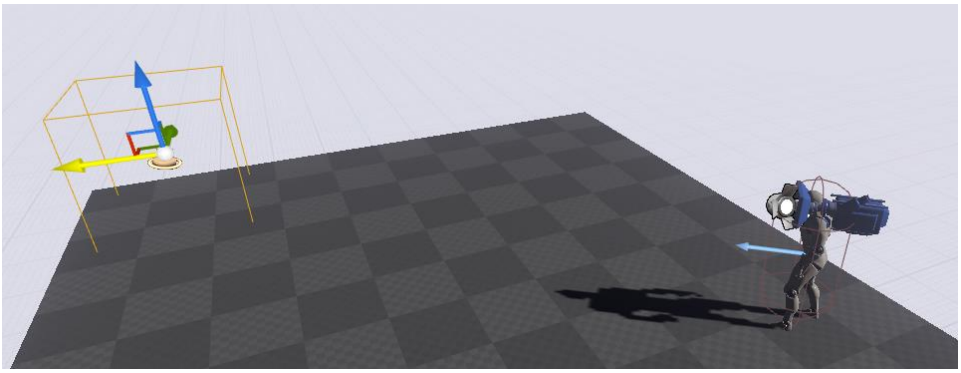

Figure 5: Blueprint demonstration scene

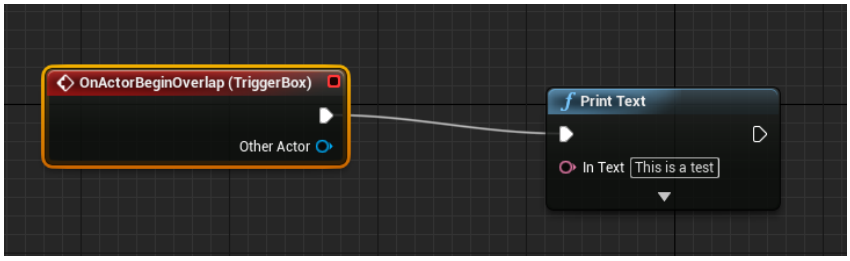

Figure 6: Blueprint for the demonstration

Figure 7: Text is printed when the player enters the trigger box

5 Practical implementation

This section describes the practical implementation of the project, using Unreal Engine. The game content is split into categories by functionality for easier understanding. Many of the features reference each other and they were not developed to the final stages at once, but in stages, improving them as new techniques and ideas came up.

5.1 Features

The game went through many changes and improvements during the project, but the development process could be divided into four major builds. Between each of these builds, there were additional versions that were developed on a weekly basis. The chart below displays the timeline during which these builds were created in and what they contained.

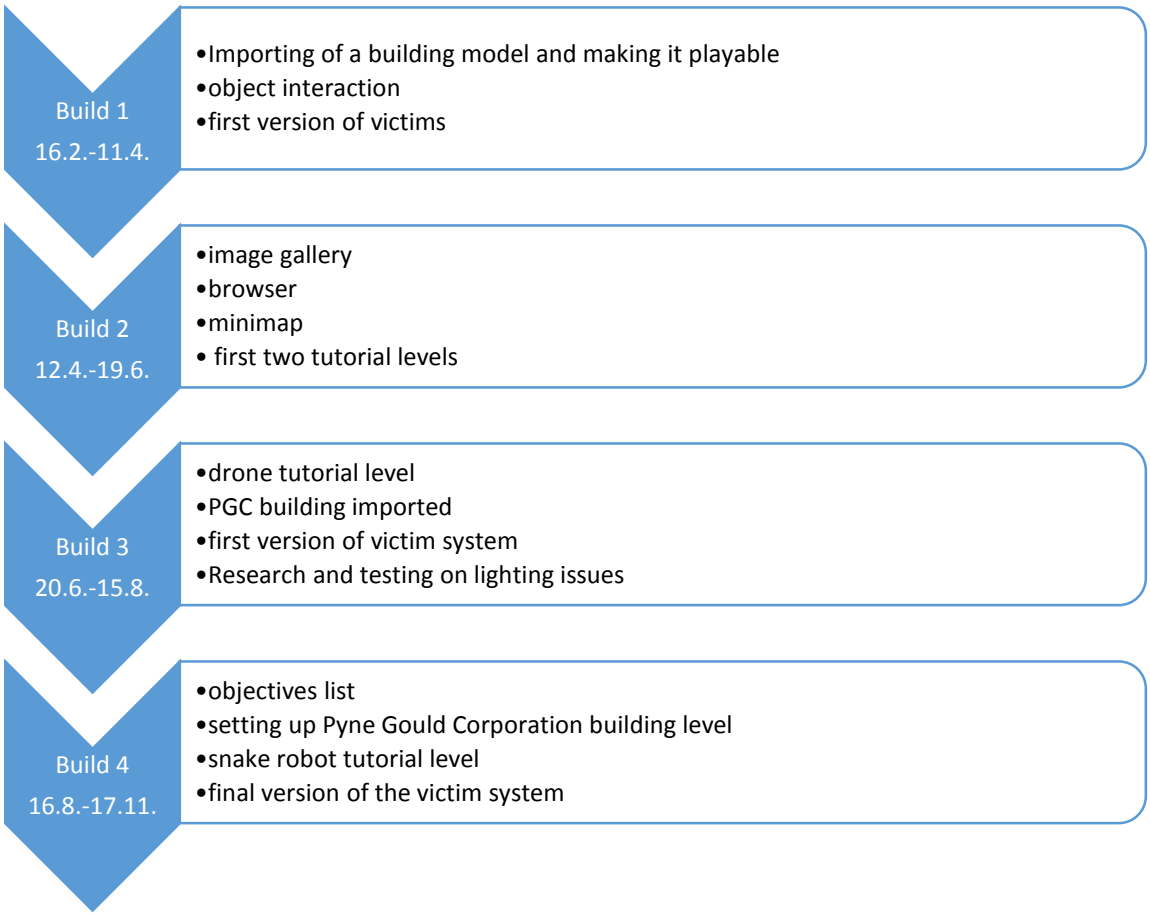

Figure 8: Timeline and descriptions of the different stages of the game

5.2 Building the environment

The environment is the scene, that the player sees and it sets the tone and theme of the game. For example, peaceful meadow and a dark cave give the player entirely different feelings. This part describes the different tasks needed to create the environments in the game.

5.2.1 Importing from blender

As 3D modelling in Unreal Engine is not supported beyond basic shapes, using another program for this task is necessary. Importing 3D-models from Blender to Unreal Engine is convenient as they both support FBX-file format. The transferred objects retain their proportions, material assignments and UV mapping. Once the model is being imported to Unreal Engine, there are multiple options to choose from depending on the need, such as keeping the previous settings of the model and whether the model should be treated as a single object, even if there are separate parts.

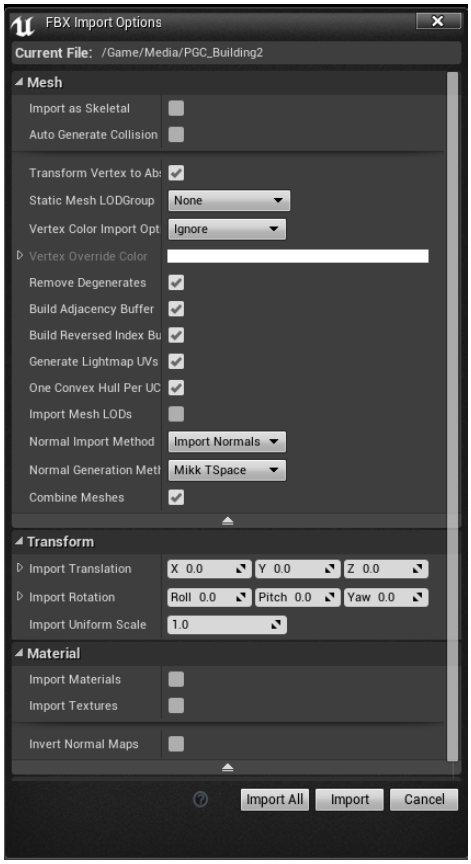

Figure 9: Import options in Unreal Engine

5.2.2 Collision settings for the building

When models are imported into UE4, they do not have any collision boxes, which means that any object or player character that was to collide with it, would go through the object. As the buildings used in the game are imported models, they require collisions to happen in order to allow the player to walk through the level without falling through the floor or walking through walls. Unreal Engine has multiple options for simplified collision boxes, for different types of objects.

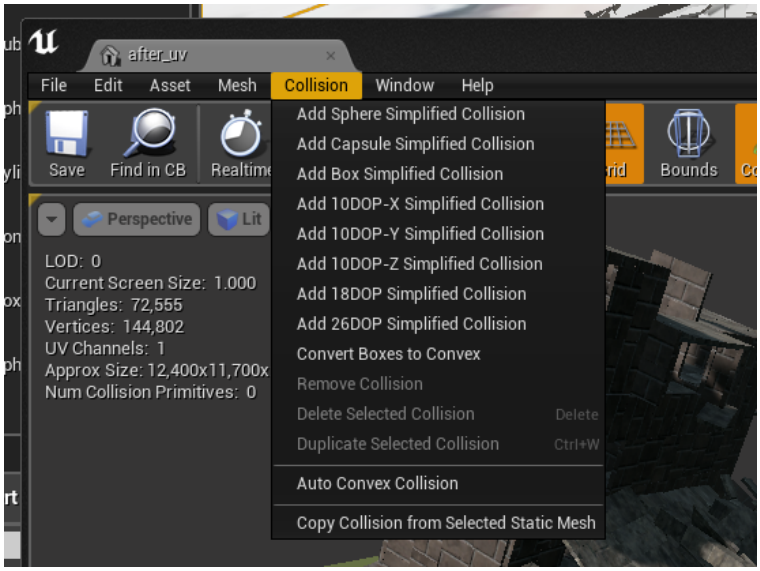

Figure 10: List of collision options

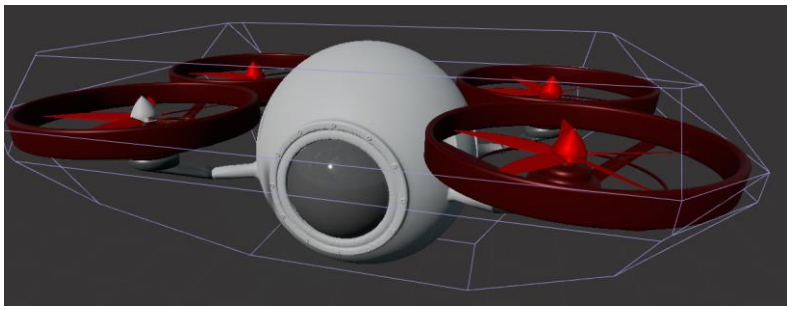

Figure 11: Example of a simplified collision box.

As the buildings are complicated models, a special setting for collisions is required to allow free movement outside and inside of the building. The option “Use complex collision as simple”, creates collisions along the surfaces of the model.

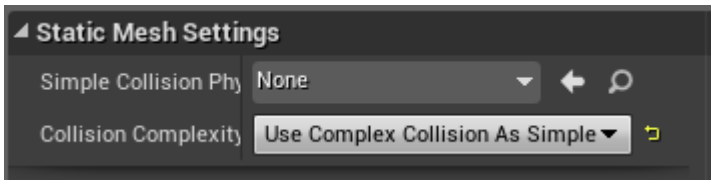

Figure 12: Collision setting being used for the building models.

The “Use complex collision as simple”-setting is not completely ideal for every game, as some collisions, for example falling objects may react unnaturally to colliding with a model using this setting. During testing, dropped objects bounced off the ground or were sent flying on impact. For this game’s purposes, it causes no issues and is therefore considered the best alternative.

5.2.3 Building the basic level layout

Each level serves a different purpose, which would reflect on the layout of the level. The collapsed building level, for example, is large and complex, when a tutorial for moving objects does not require a large area or a complicated structure. Before starting the creation process, it was important to plan what was necessary for the level's purpose and how the user would move in the level and finally a rough layout was drawn.

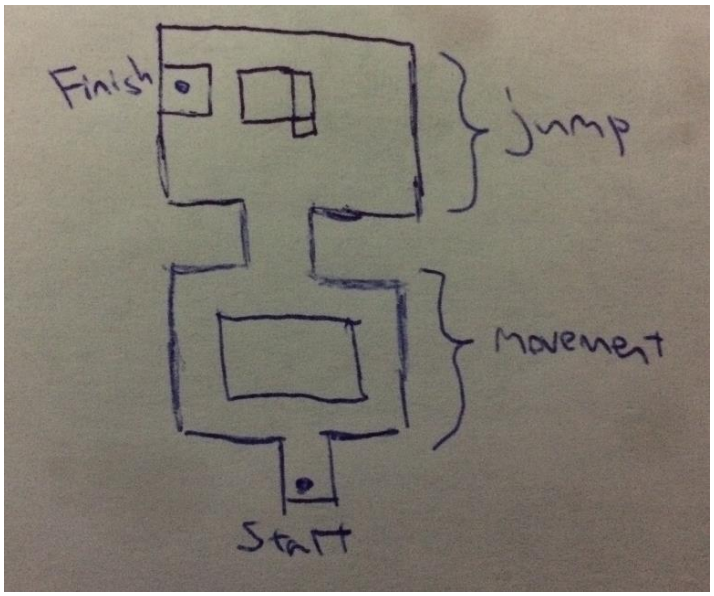

Figure 13: Layout planning of the first tutorial level

When new level was created, the first thing to do was to create a ground to walk on. After that came the walls, and other elements required for the level in question, such as blocks to stand on for movement tutorial level and obstacles for the drone tutorial.

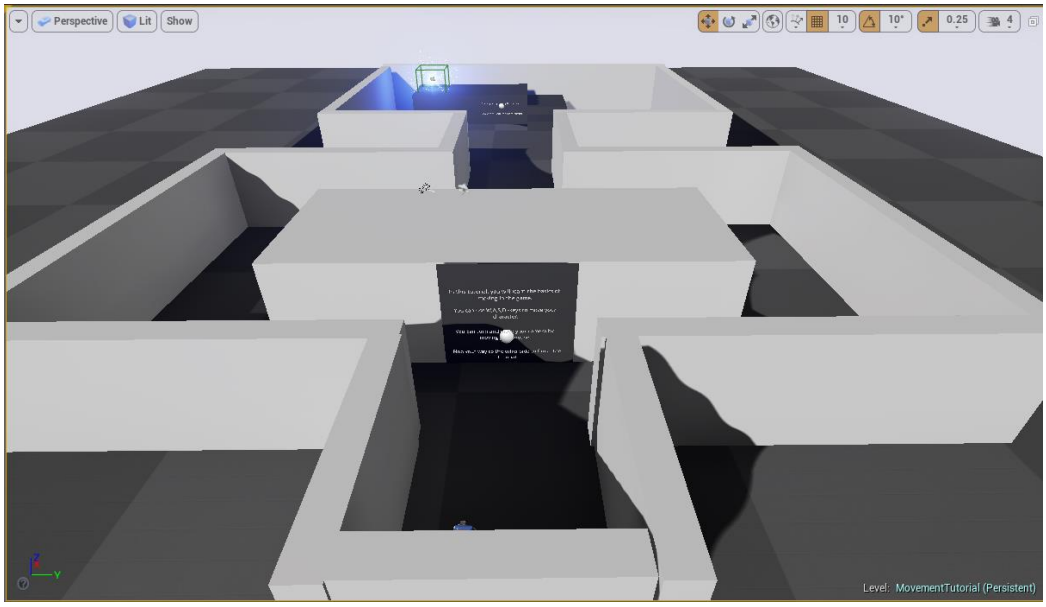

Figure 14: Overhead view of the first tutorial level

5.2.4 Lighting

Lighting is in a large role in making the game visually appealing and required for shadows and texturing purposes. However, due to the way the building was created from a massive amount of pieces to recreate the collapse, there were problems in making a functioning UV mapping. The UV mapping would have to be made separately to each piece in the building model to prevent it from overlapping, causing the process to consume a massive amount of time, as for example the PGC building was built from about 6000 pieces. This theory was tested, by taking a part of a wall from the PGC building and UV mapping half of it manually and the rest automatically. Once imported to Unreal Engine, the manually UV mapped side worked well with lighting, but the other half remained completely black, as shown below.

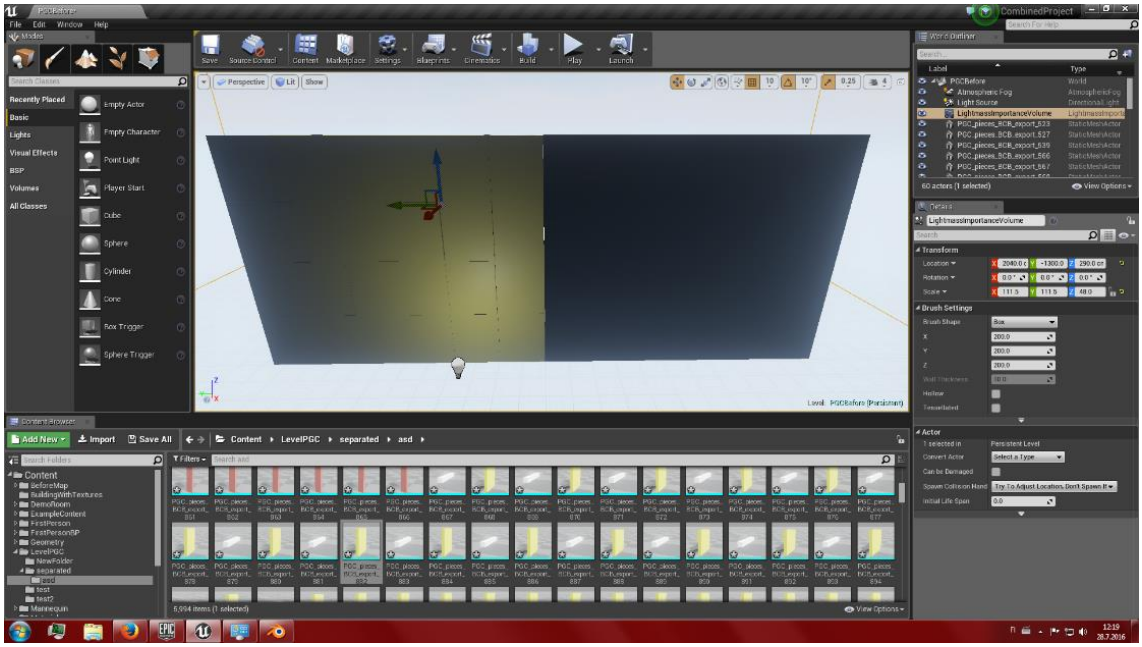

Figure 15: Test result of UV mapping a part of a wall

The best lighting quality for the game at this stage was accomplished by using the editor's lighting preview settings, before the lighting is built. With this setting the shadows are not as realistic as when the it is properly built, but it allows developing and playing in a decent lighting and no error messages are shown, once the game is played from a standalone file. Additionally, once the lighting is built, the texturing becomes chaotic, due to the UV mapping problems, as seen in the image below.

Figure 16: The PGC level after lighting is built.

5.3 Level database

Some levels that include tasks that require passing information, such as variables, from one blueprint to another, needs a level database. The database stores variables that are easy to

reference and to modify within another blueprint. For example, the victim blueprint is highly dependent on this functionality.

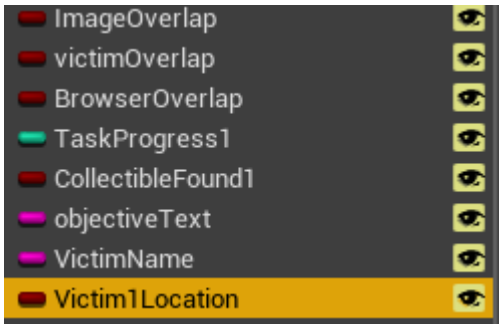

Figure 17: List of variables shared through the database

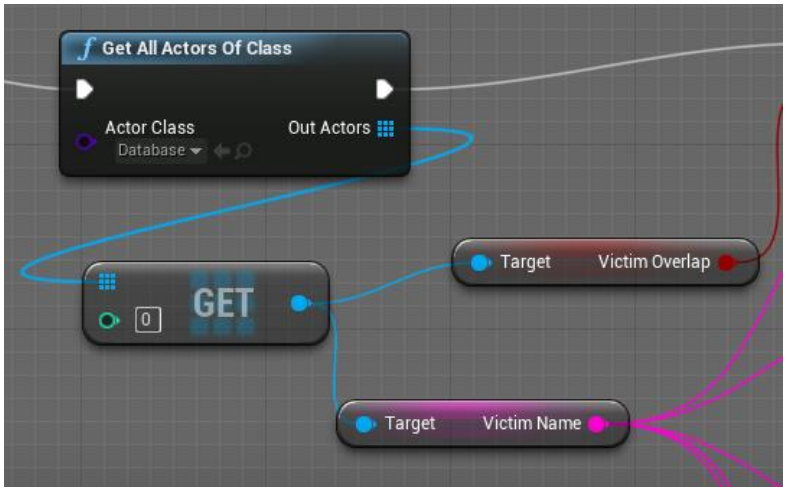

Figure 18: Example of referencing to variables in the database.

5.4 Playable character

The character, also called as pawn, represents the player's body, which is used to move and interact in the game world. Unreal Engine has a mannequin character model inbuilt and it is used as the character in the game, to reduce the time consumption of creating, importing and setting up the animations for a new character, such as a fireman. Nearly all actions in the game have connections to the character, but the following functionalities are tied closer to the character itself.

5.4.1 Picking up objects

In a building collapse site, there is debris, such as pieces of walls, ceiling and furniture, that needs to be moved, to be able to search for victims and to access cavities. Due to this it was

important to build a functionality, that would allow the user to move objects, in order to clear a path, to the victims buried under the rubble.

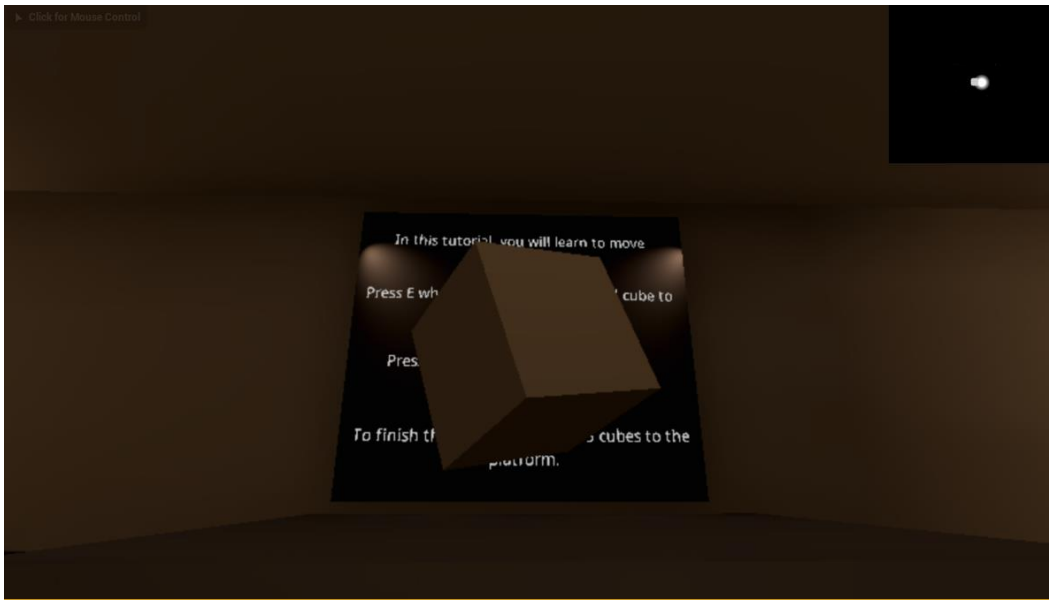

Figure 19: A cube being carried in the pick-up tutorial level

There are three stages in carrying objects: picking up, carrying and dropping. A scene component was attached to the character's camera, to mark the position where the carried object would be positioned in. Also a physics handle was included to allow moving physics objects.

When the pickup-key is pressed, it checks whether an object is already being carried and then proceeds to attempt to pick up a targeted object.

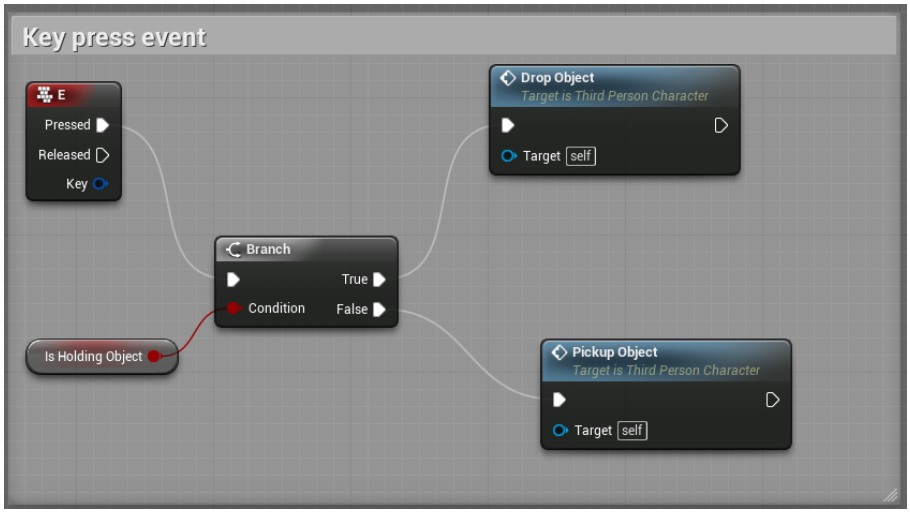

Figure 20: Key press event checks if an object is already being held.

The Pickup Object - function checks for objects in a straight line from the camera and checks if the object is suitable for picking up.

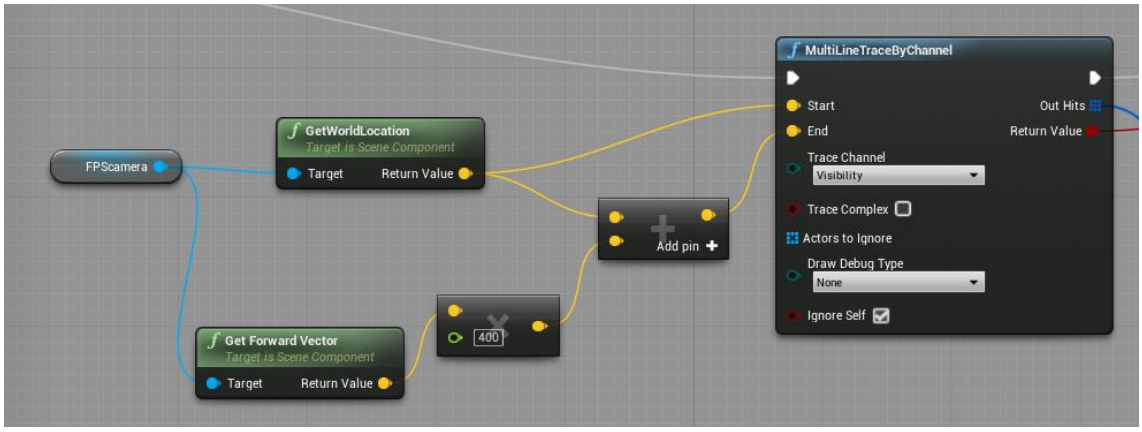

Figure 21: Line trace checks for objects in front of the camera.

The object needs to simulate physics, in other words, the object will be affected by gravity or other forces. In addition, the object's weight needs to within the specified range to be able to simulate objects that are too heavy to lift.

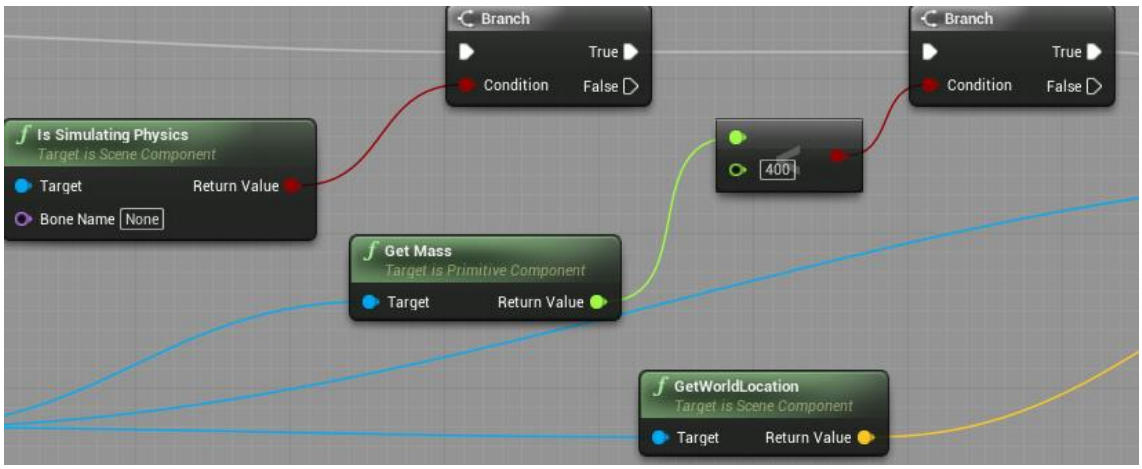

Figure 22: Target object is examined, whether it is suitable to be picked up.

After the object is successfully examined, it will be picked up and its location information is passed on to the physics handle to pick it up. The Is Holding Object – variable's value is set to True for the key press event.

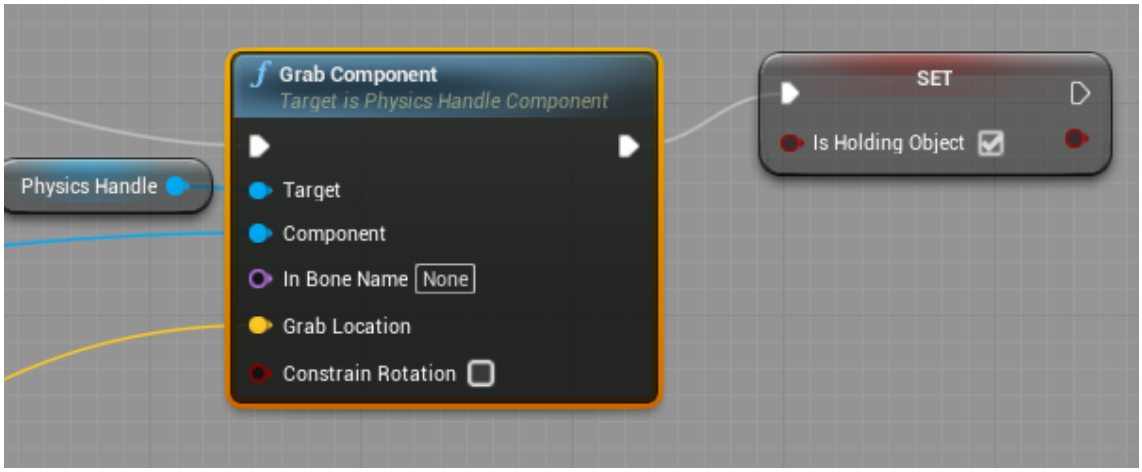

Figure 23: Object is picked up.

Once the object is picked up, it will need to carry the character's movement and rotation, in order for it to be moved appropriately. Setting the object's location to match the physics handle's location constantly by using Event tick, the object will remain in the set position until released.

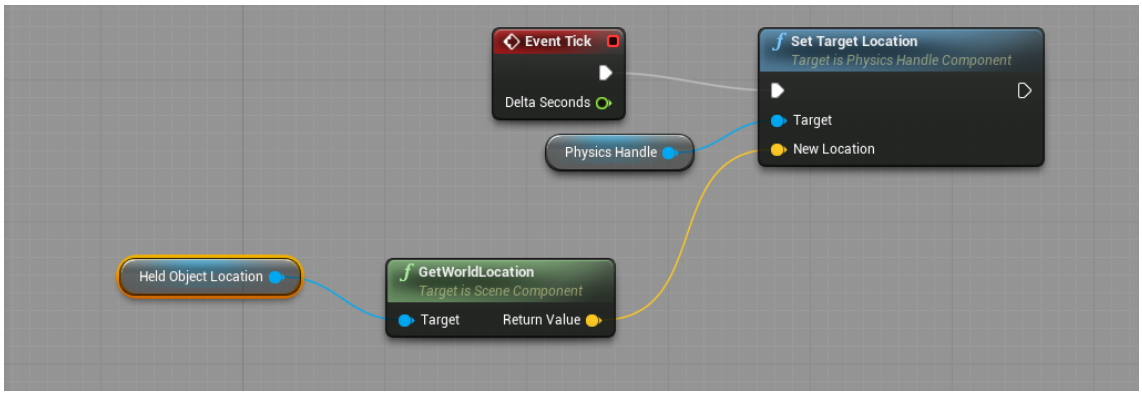

Figure 24: Updating the object's location to follow the player's movement

When the pick-up key is pressed again, the object will be released and the IsHoldingObject-variable's value changes to false to be ready to pick up another object if necessary.

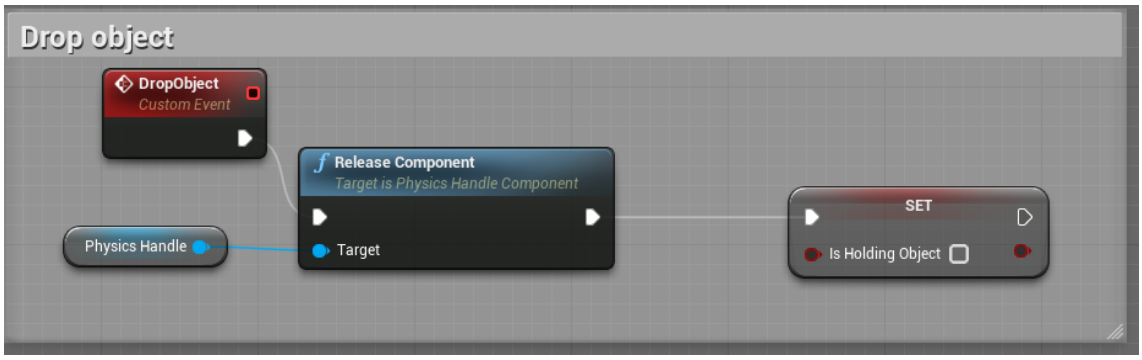

Figure 25: Drop object event releases the carried object

5.4.2 Switching pawn

In the game, there are three different playable pawns; the human, the drone and the snake robot. Each of these can be used for different tasks, like scouting the area using the drone and entering small spaces using the snake robot. In the PGC level, for example, it is possible to use both the human and the drone for moving in the scene.

To allow playing with multiple pawns in the same level, ability to switch pawn in-game is required. The most convenient way of doing this is by assigning a key for the action, for example pressing “T” will toggle between human and drone as shown below.

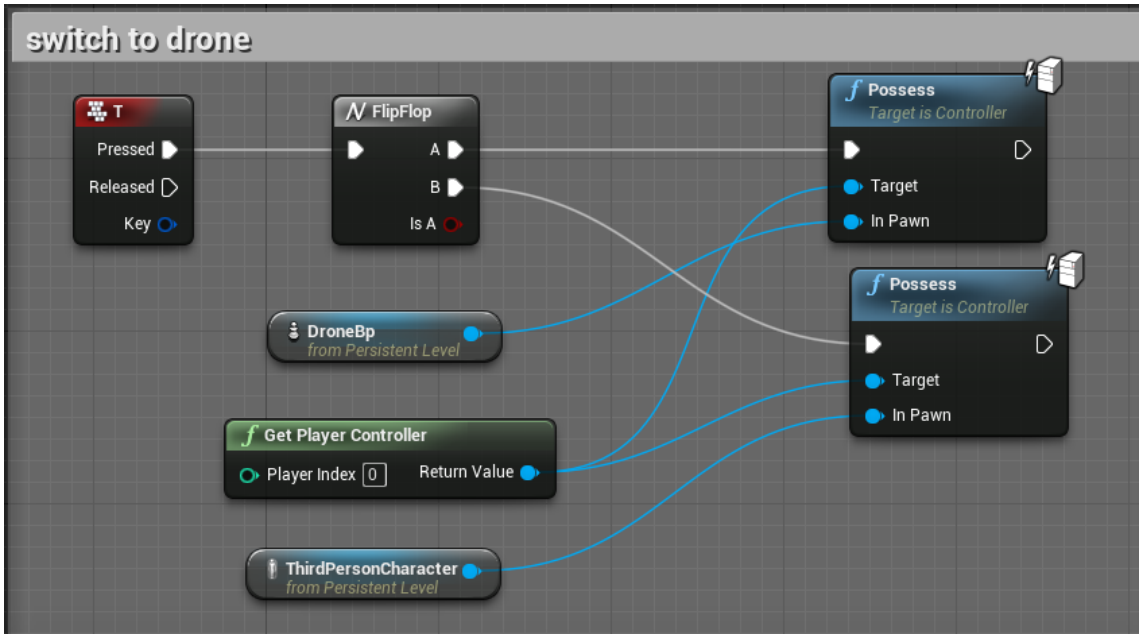

Figure 26: Blueprint for switching between human and drone pawns

5.4.3 Minimap

A mini-map is commonly used in games, as a way to identify, where the player is at the moment and to show points of interests in the surrounding area. A new HUD widget was created to add the mini-map and any future content, such as health bars or usable tools, onto the screen.

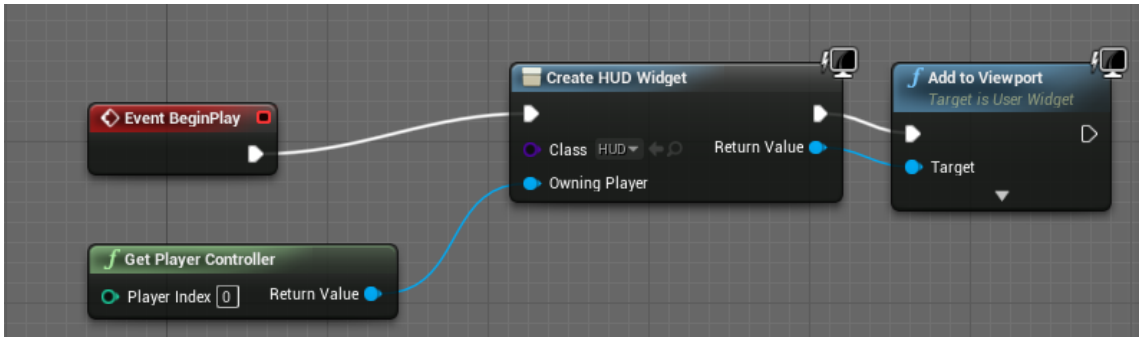

Figure 27: The HUD is shown when the game starts.

To get an image of the surroundings, a scene capture component, which works like a camera, was attached to the character and placed high above it. The component captures the scene constantly and saves it as a texture that is used in the HUD widget. The end result is an aerial view of the surrounding area, which updates in real-time.

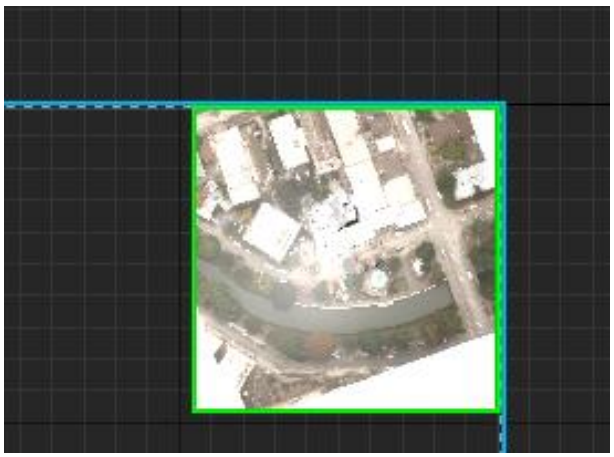

Figure 28: The image captured by the scene component used in the widget.

5.5 Victim system

The victims, represented by mannequins, are the main task in the game. The system can be split into two parts: the individual victims and the user actions through the victim widget.

5.5.1 Victims

The purpose of the victim blueprint is to give each victim a set of statuses, such as pulse and consciousness level, which would be used in determining the course of action for the rescue operation.

When the level starts, the randomizer chooses a unique set of statuses to each victim, by selecting an integer in the specified range and then checking which text variable the integer matches with and then repeats the process for other statuses.

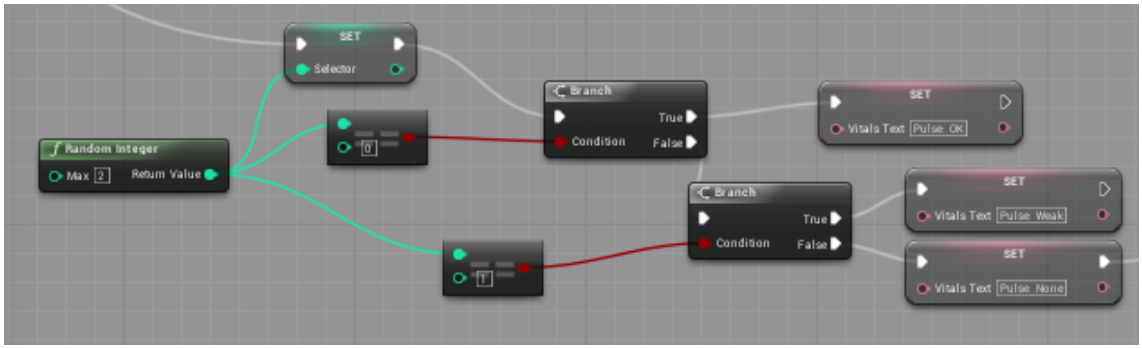

Figure 29: Randomizer for victim's pulse.

In addition to the statuses in text format, there is also a progress bar, which represents the vitality of the victim. In the earlier stages, created by the students from Metropolia, the bar represented the victim's hydration level, but was remodelled for the use as a vitality bar. The bar decreases constantly giving a time limit for the player to rescue every victim in time. Once fully depleted, the victim will no longer be able to be rescued.

When the level starts, the blueprint starts to check, if the victim is still alive and if it is not rescued yet. If either of these are true, the victim will disappear.

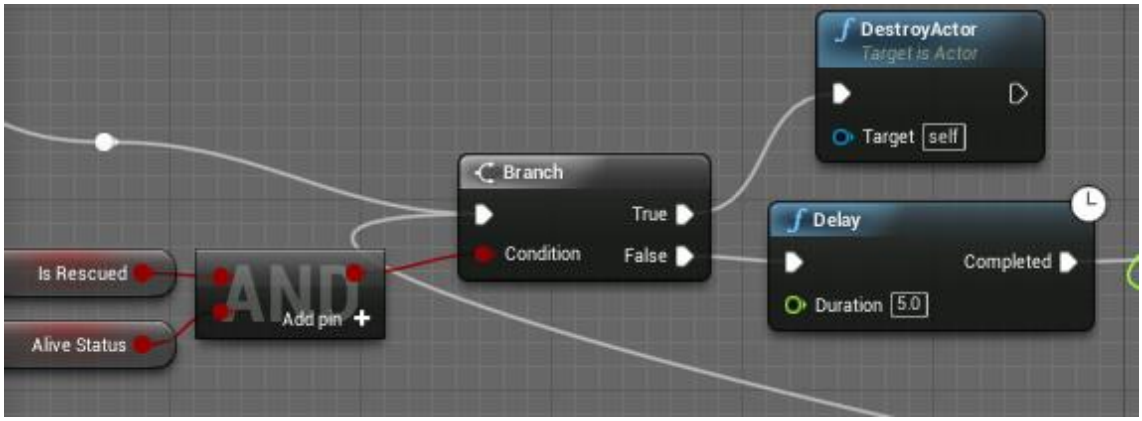

Figure 30: Possibility for victim's rescue is checked.

If the victim is still in need of rescue, its vitality decreases every 5 seconds and the amount is checked. If it reaches 0, the victim's alive status will change to false.

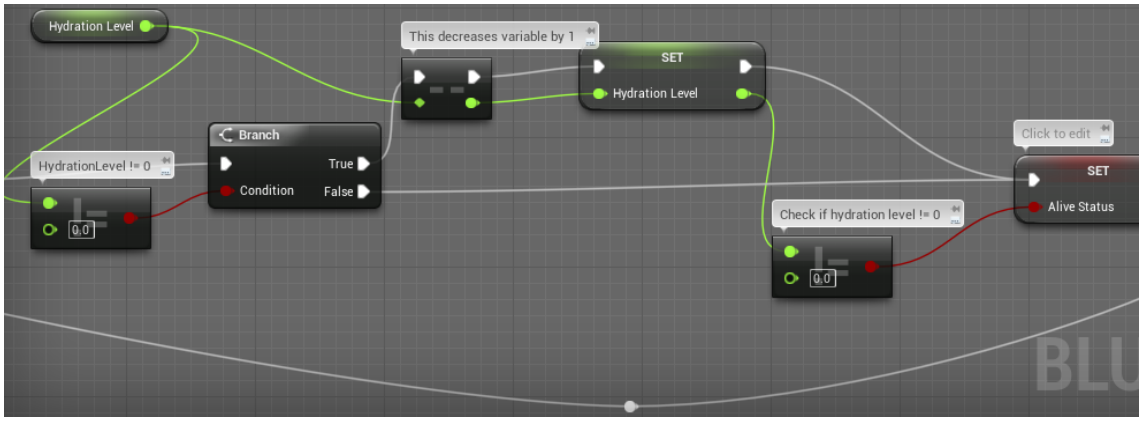

Figure 31: Vitality level is checked and decreased over time.

5.5.2 User actions through the victim widget

When the user approaches a victim, a text appears in the screen instructing to press a key to open the victim rescue widget.

Figure 32: Victim found in the collapse site.

The widget is the User Interface for the victim rescue, which collects and presents the data about the victim and allows user interaction. As there are multiple victims with different statuses, each of them have their own widget to avoid problems with rescuing victims, such as rescuing a victim makes another victim disappear instead of the intended one.

As the key to help the victim is pressed, the blueprint identifies the victim and opens the corresponding widget.

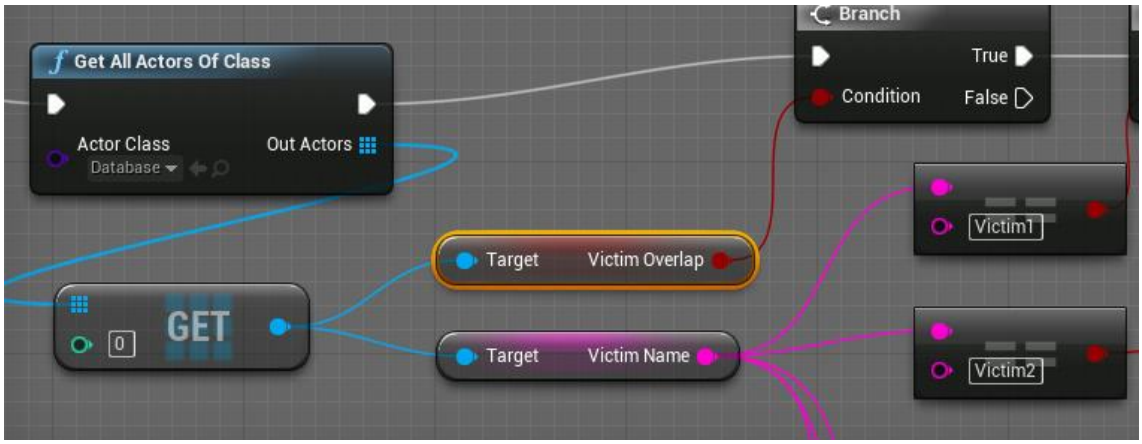

Figure 33: Victim is identified

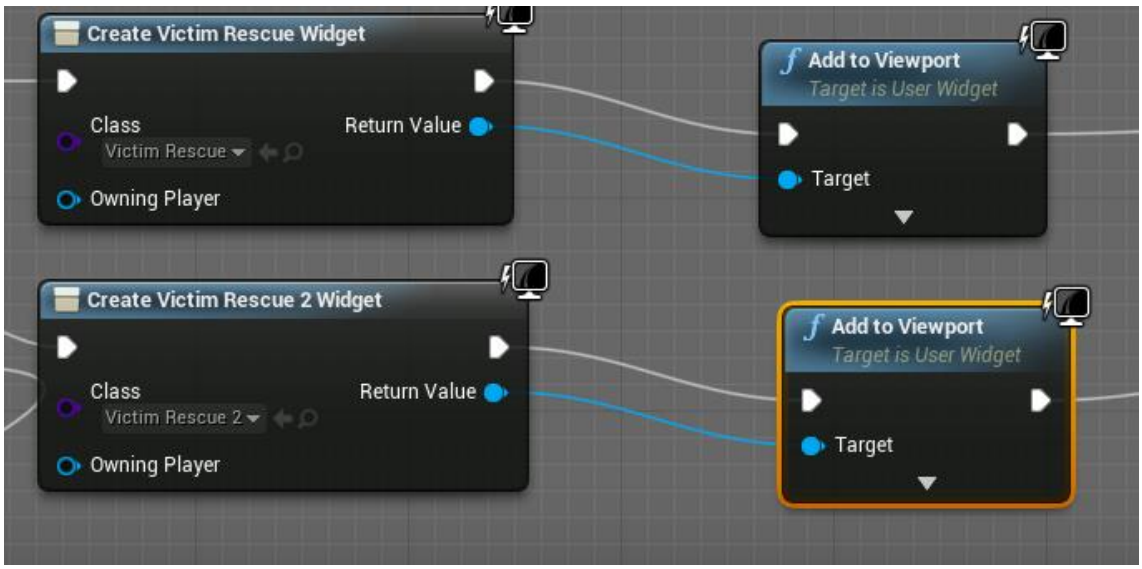

Figure 34: Corresponding victim widget is opened

When the widget is first opened, it collects the victim's status information from the victim blueprint and references to the objective message in the level database by using the Get All Actors Of Class -function. The collected status information is placed into the corresponding fields in the widget, which will be shown to the user as the widget opens.

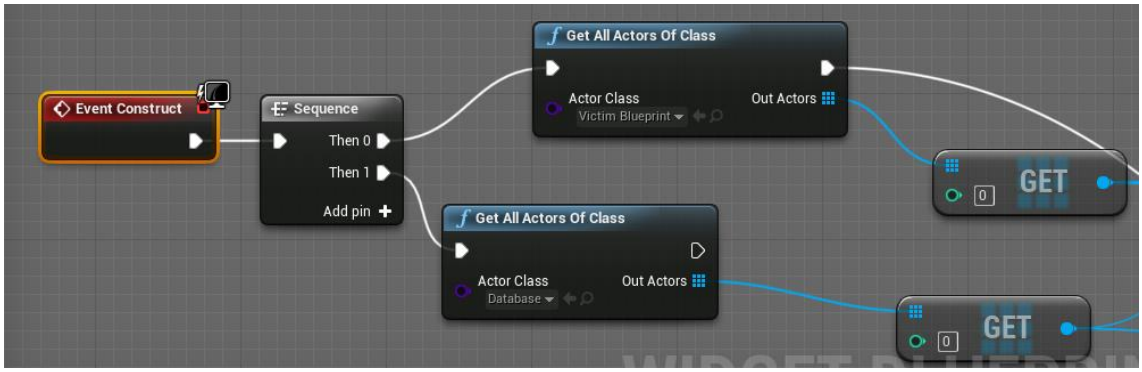

Figure 35: The widget collects data from two sources

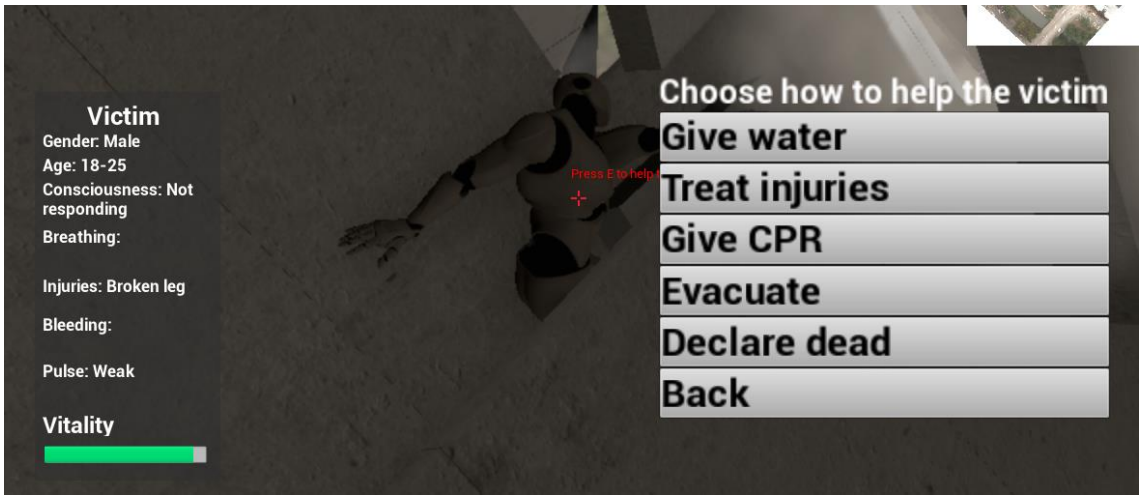

Figure 36: Victim statuses shown to the user.

The Give water-button allows the player to fill the victim's vitality bar. Once clicked, the widget gets the hydration level from the victim blueprint and sets the value to 100.

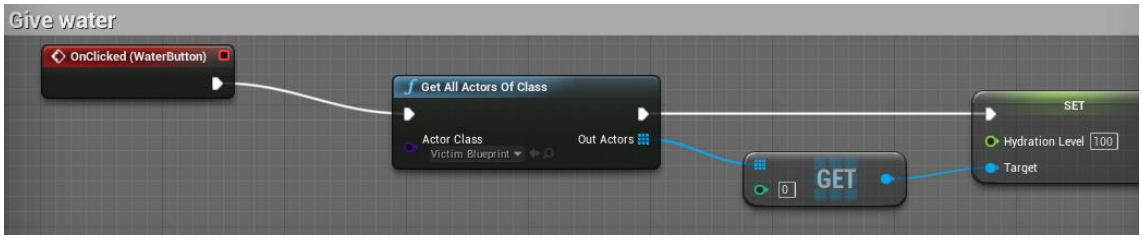

Figure 37: Giving water fills up the vitality bar.

Treatment can be given to an injured victim by pressing the Treat injuries-button. When pressed, the widget will check if there is any need for treatment and then changes the status text to "Emergency treatment given".

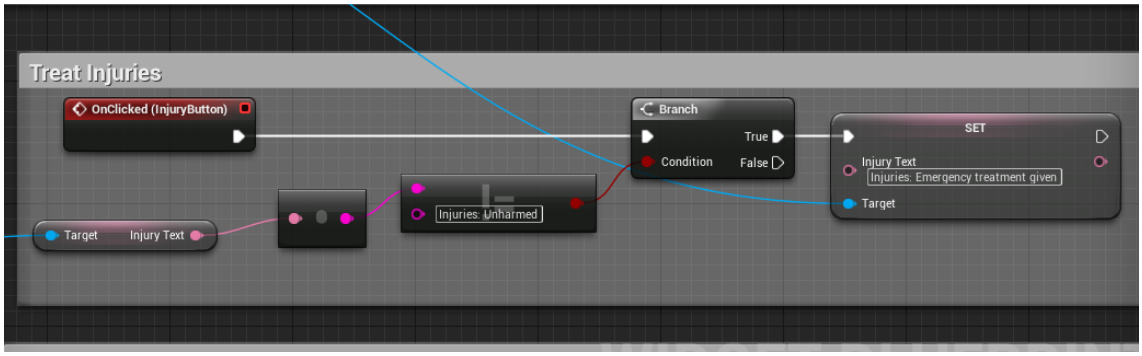

Figure 38: The value of the text variable is checked before changing it.

If the victim has no vitals, the player has three chances to revive the victim by pressing the Give CPR-button. The widget will check the victim's vitals status and the amount of attempts so far. If either of these does not match the requirements, a message will appear on the screen informing the player that the victim is unable to be revived.

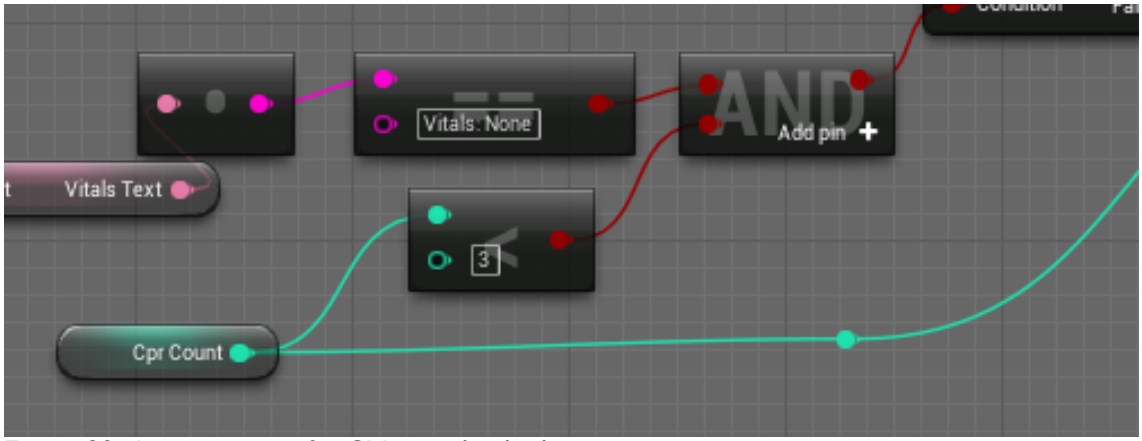

Figure 39: Requirements for CPR are checked.

If the victim has no vitals and the three tries were not used yet, the Cpr count- tracker is incremented and then a random value between 0 and 9 is picked. If the value is above 8, the victim's status will change to "OK", above 4 will change to "Weak" and below that will have no change. The objective text will tell the user whether their attempt in reviving the victim has succeeded or not.

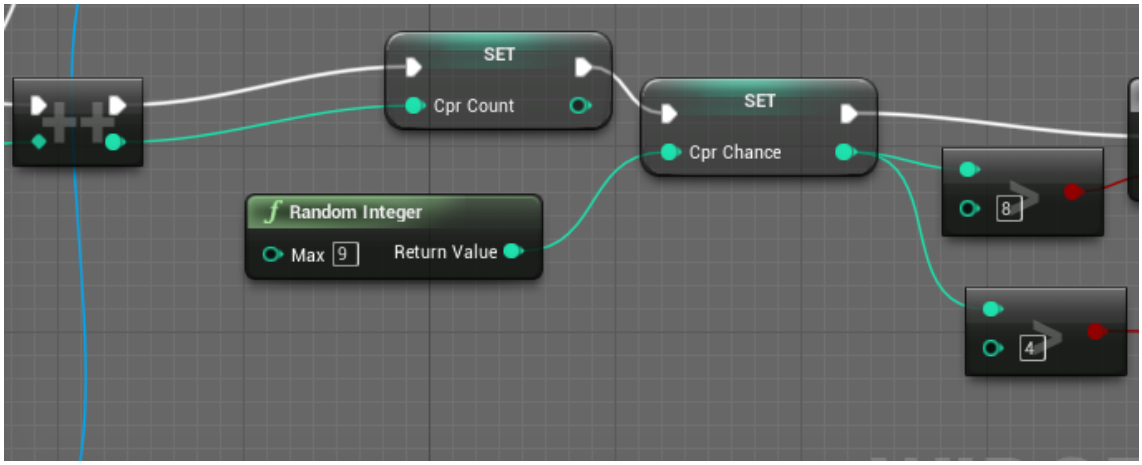

Figure 40: The CPR count is incremented and the random integer is chosen.

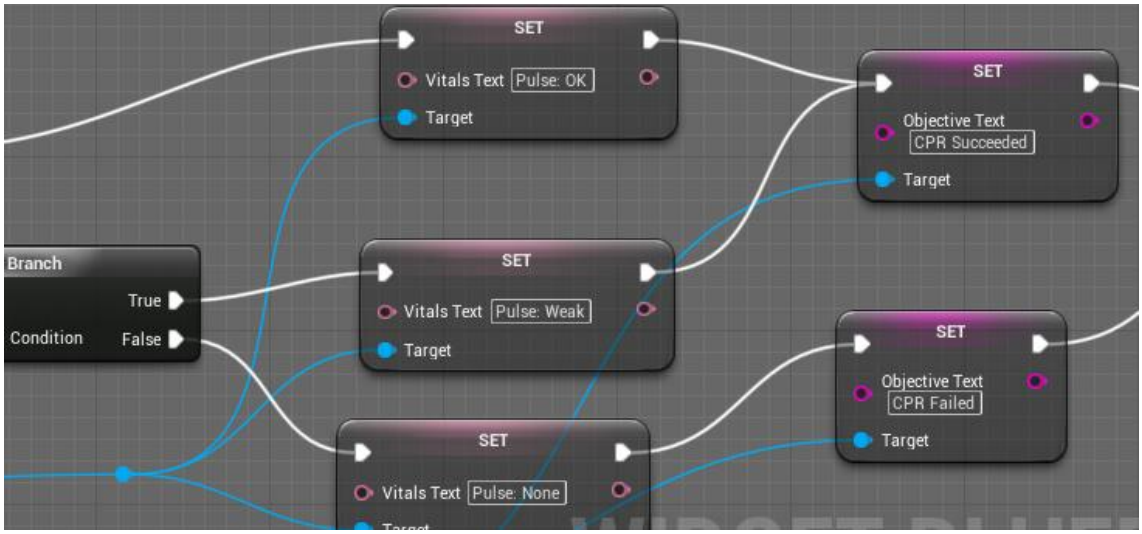

Figure 41: Vitals -value is changed and objective text is displayed.

Rescuing the victim is done by pressing the Evacuate-button. The widget checks if the victim is still alive and then sets the victim as rescued and updates the objectives list with the located victim.

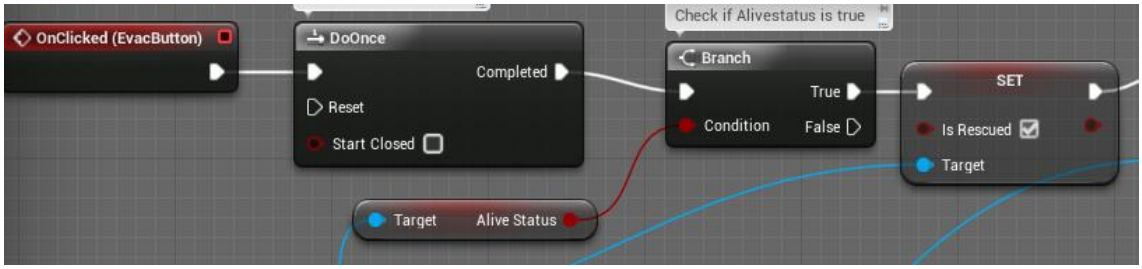

Figure 42: Before rescuing, the victim's livelihood is checked.

The Declare dead- button works the same way as Evacuate, but will not add a point to the objectives. It is intended to be used when a victim has not been rescued in time, but at the prototype stage, it has no special function.

5.6 Levels

Splitting the game into separate levels serves multiple purposes. It helps making the game lighter for the computer, as everything is not loaded at once. To make the game easier for new players, especially for those with no previous experience from games, it was important to make introductory levels explaining the controls and mechanics, to avoid the player getting overwhelmed. In addition, using levels allows the use of different environments, suitable for the purpose of each task and also creating variation to the game's visual side.

5.6.1 Movement tutorial

First tutorial focuses on simple movement using the human pawn. The player gets guidance to moving and looking around as the game starts.

Figure 43: First image the player gets when the game starts

When the player reaches the next area, another information board instructs the player to jump to the portal to finish the tutorial. At first the player had to jump up the stairs and then cross a gap to reach the portal, but it proved to be too difficult for more inexperienced players and therefore it was made simpler by removing the gap and preventing the player from falling off the structure.

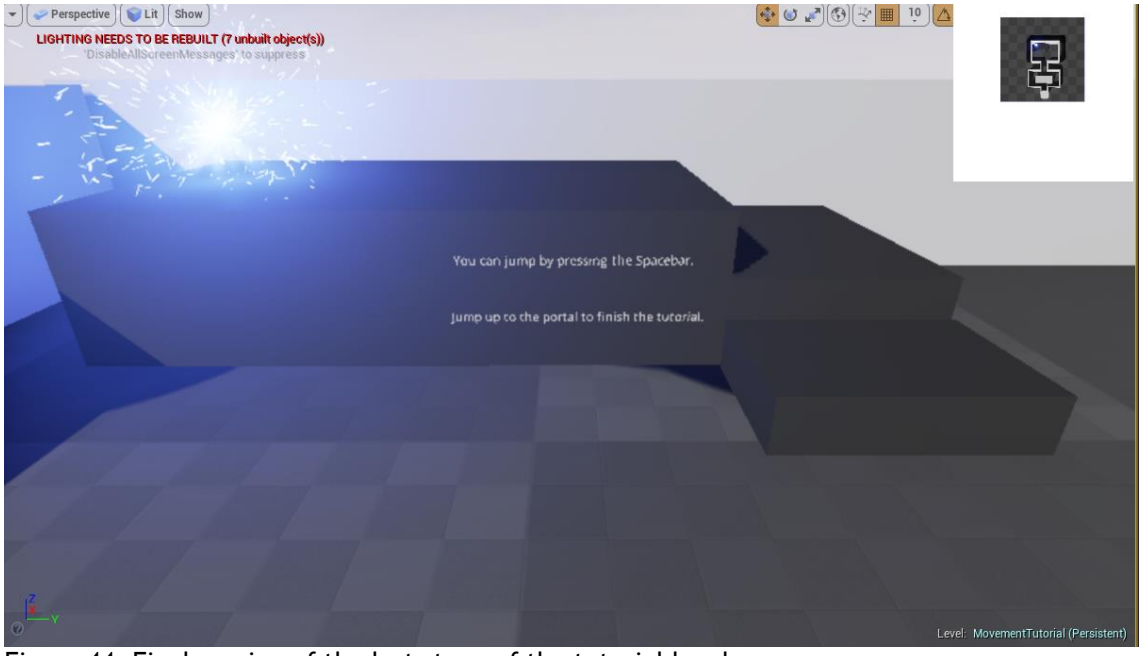

Figure 44: Final version of the last stage of the tutorial level

5.6.2 Tutorial for moving objects

The second tutorial focuses on moving objects. The goal is to move all cubes to the platform to pass the level.

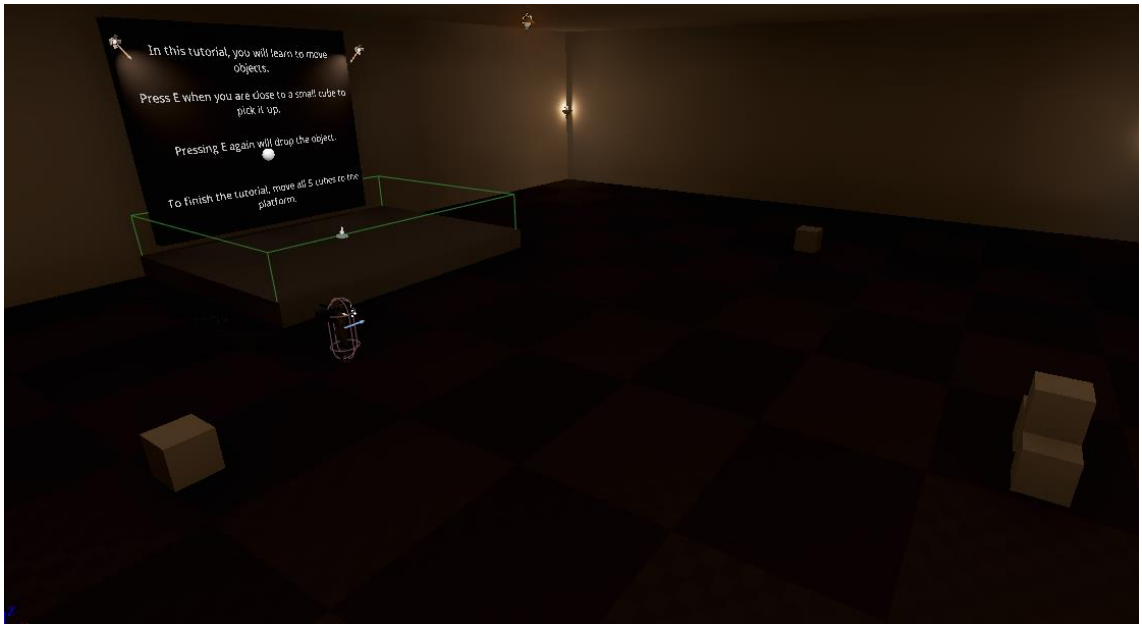

Figure 45: Overview of the level

A trigger box was placed on the platform to know when all objects are moved correctly. It checks the number of objects inside itself and when the number is 5, it starts the next level.

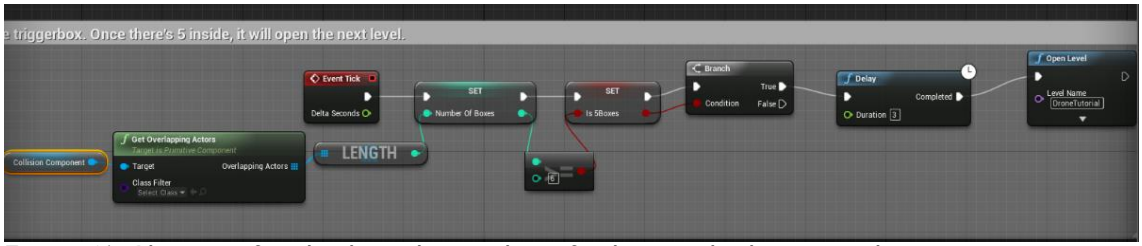

Figure 46: Blueprint for checking the number of cubes inside the trigger box

5.6.3 Drone tutorial

The next tutorial teaches the player to fly a drone by controlling the height in addition to the already learned controls. The level is completed once all blue sparks are collected.

Figure 47: Drone tutorial overview

To make the user fly through each spark instead of flying straight to the last spark, a counter was implemented to check how many had been collected so far, by incrementing the value of the counter by one. An event tick function checks the value of the counter. Once it recognizes that all sparks are collected, the next level is opened.

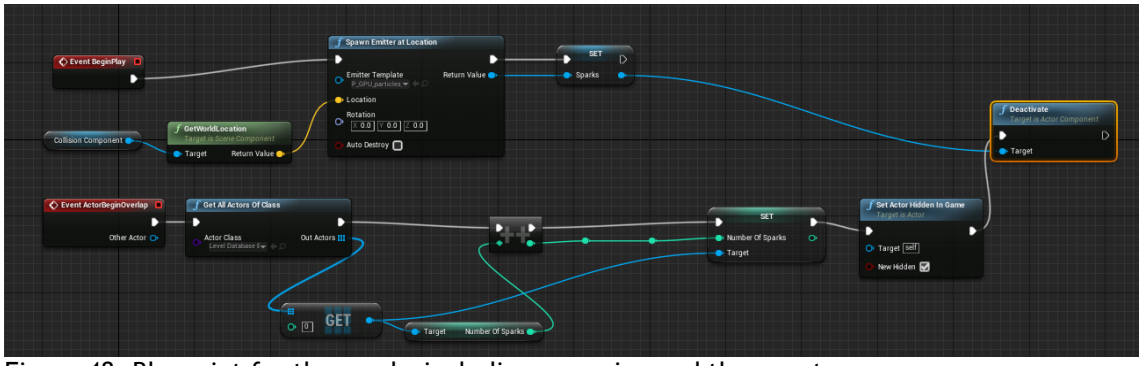

Figure 48: Blueprint for the sparks including spawning and the counter

5.6.4 Snake robot tutorial

The last tutorial in the prototype is for showcasing the snake robot that is being developed by INACHUS, explore the collapse site in search of victims. The player enters a hole, which represents a small cavity that would not be otherwise accessible and searches for victims underneath. after each victim is located, the tutorial level is completed.

Figure 49: First view to the snake robot tutorial level

Figure 50: The way inside the tunnel is visible only with the flashlight

5.6.5 Pyne Gould Corporation building level

The main level of the game is the PGC building level, where the main task is to find and rescue all victims before their vitality is depleted. The player can use both the human character and the drone to search the area.

The level contains the collapsed Pyne Gould Corporation building and the surrounding area has been visualized with a snapshot from Google Maps of the neighbourhood. For more realism, a recording of the actual collapse site was added to the background.


Figure 51: Aerial view of the collapse site.

A fire truck model was found and downloaded from clara.io-website and brought into the scene for extra authenticity. The materials used in the model were either inbuilt Unreal Engine materials or created within Unreal Engine to colour the different parts of the model.

To make it easier to access the higher levels of the building, an access point to the roof was created. As the player enters the trigger box shown below, they are moved to the roof directly in front of them by changing the character position to the target point placed on the roof.

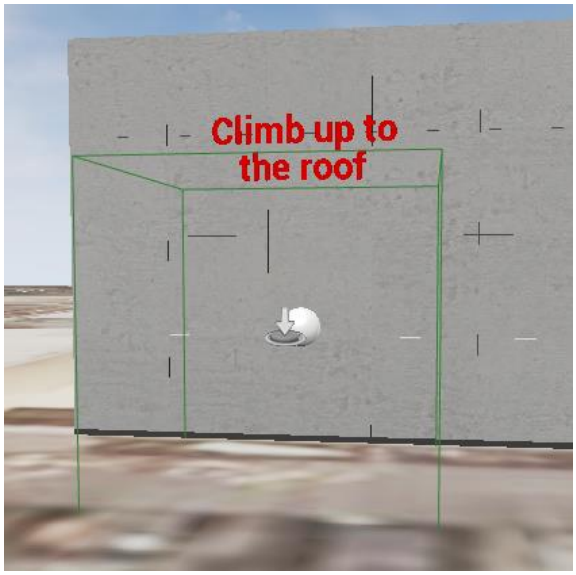

Figure 52: Trigger box with guidance for the player

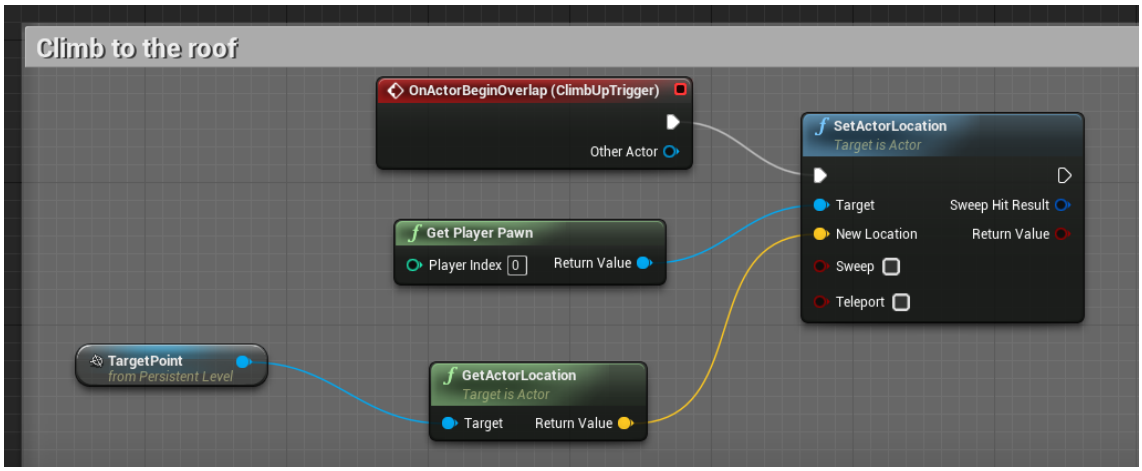

Figure 53: Blueprint for moving the player to the roof.

5.7 Media

As the game uses a building model from a real-life case, it was deemed important to include information about the disaster. Different types of media were considered, including video, pictures and news reports.

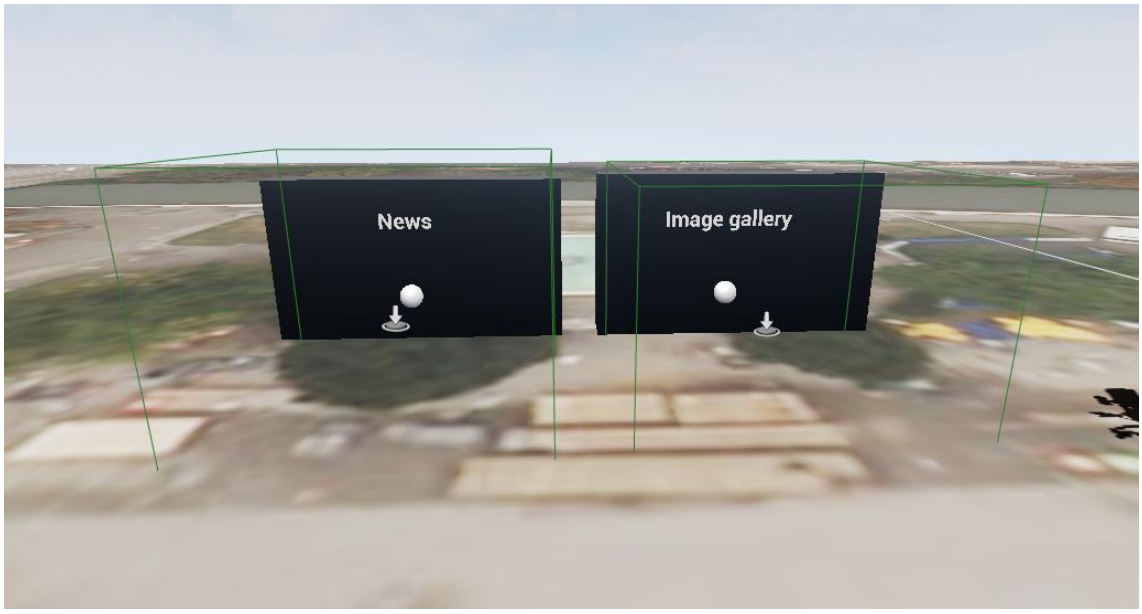

Figure 54: Media section with trigger boxes in the PGC level

5.7.1 Image gallery

The best way to give the player a reference to how the building looked like in reality compared to the one in the game was to show pictures. For this purpose, an image gallery was created by using a widget blueprint. The widget opens when the player presses a key inside the trigger box.

Figure 55: The image gallery during gameplay

The widget has images that rotate by pressing the arrow buttons. Each picture was given a value inside the widget blueprint to allow changing images. When a button is pressed, the index value is increased or decreased and the widget sets the image accordingly.

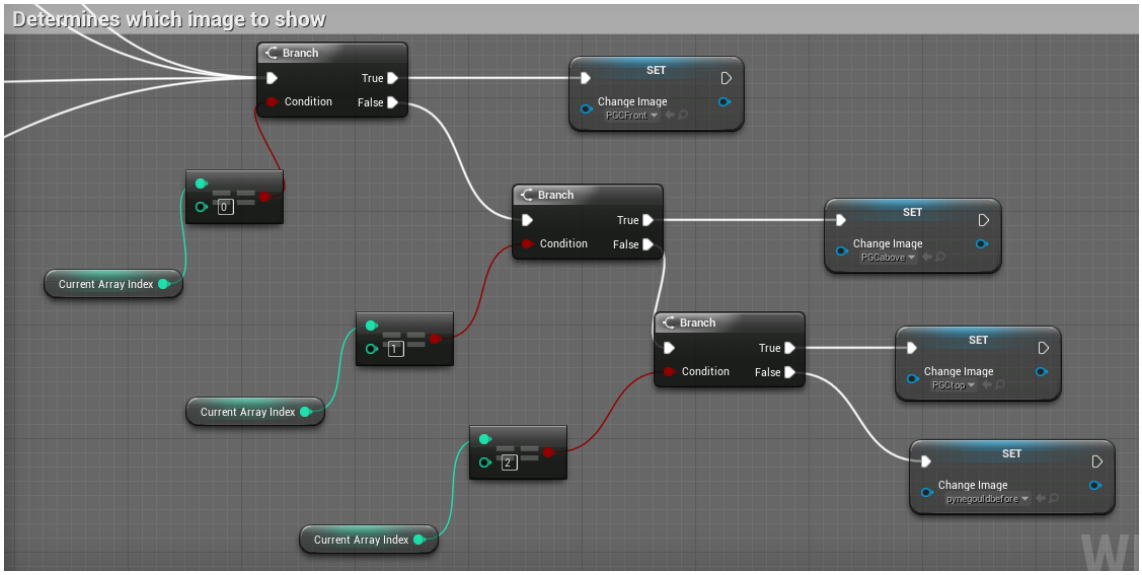

Figure 56: The blueprint checks the value of the index and shows the correct image

Once the value of the index reaches the end of the array, it is set to the other end again, to create a loop.

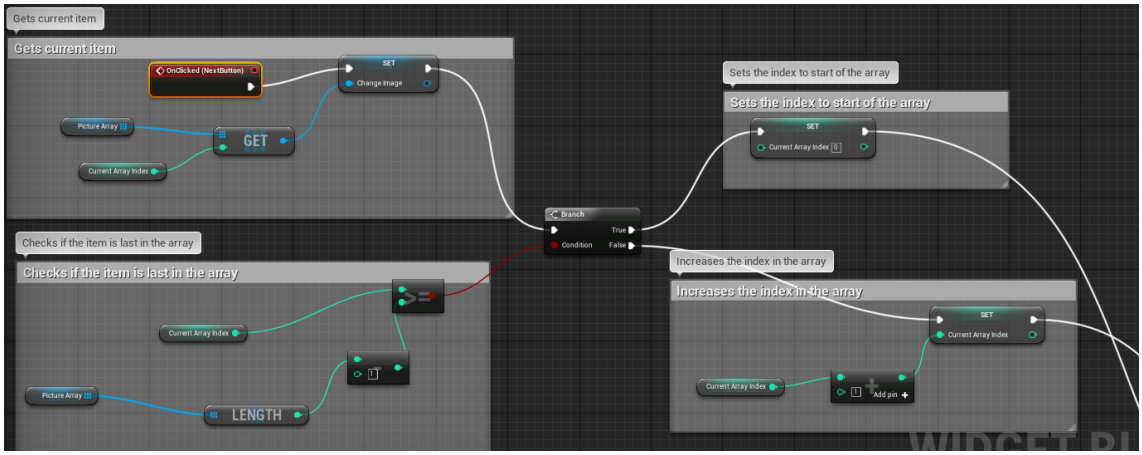

Figure 57: The blueprint for the Next button

As there is no static image in the widget, it would first show blank screen when it is opened in gameplay. To correct this, an event was added to set the first image and corresponding array index when the widget is opened.

Figure 58: Setting the first image

5.7.2 Browser

For additional information about the disaster, a browser window was added to the game using a widget with an inbuilt functionality allowing the creation of a fully functional browser. It was added to a widget that would open when a key is pressed inside a trigger box. The widget contains a search bar and an exit button in addition to the browser window itself.

Figure 59: The browser shows news about the disaster

5.8 Menus

Menus are used as a simple User Interface for selecting different actions. In the game, there are two menus, the main menu and pause menu.

5.8.1 Main menu

The main menu is the first view the player gets when the game is started and it contains the play menu and options. It was originally developed by the Metropolia students with a connection to the Highrise level, that was used in the beginning for development and testing.

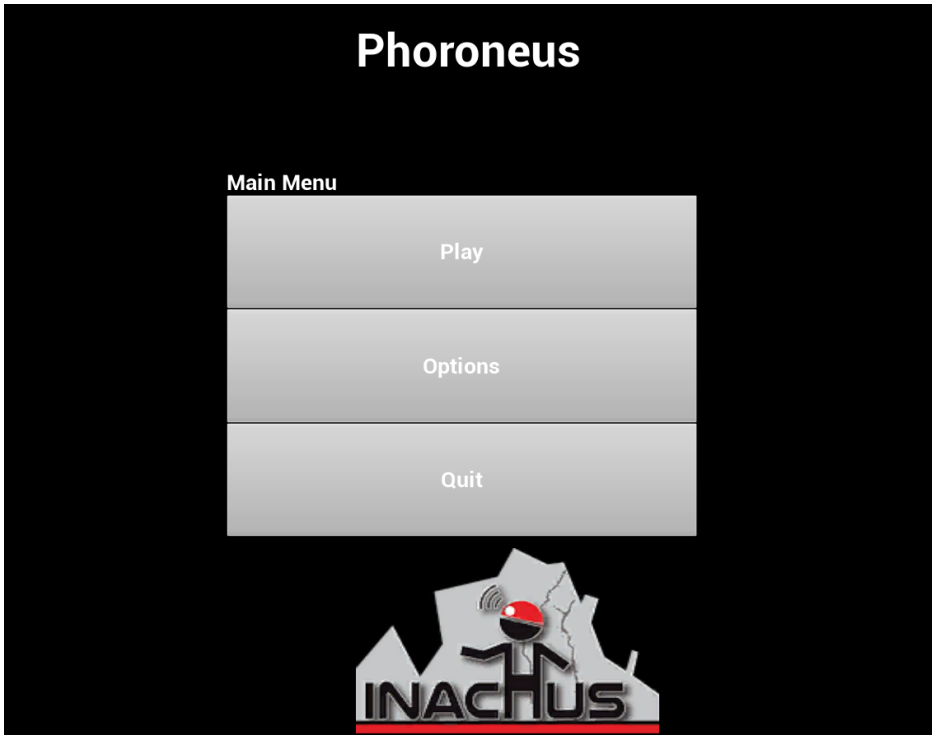

Figure 60: Main menu

Changes were made to the structure and the content of the menu, as the tutorials and the PGC level were created and they were separated from each other. A special menu view was created to display the level selection.

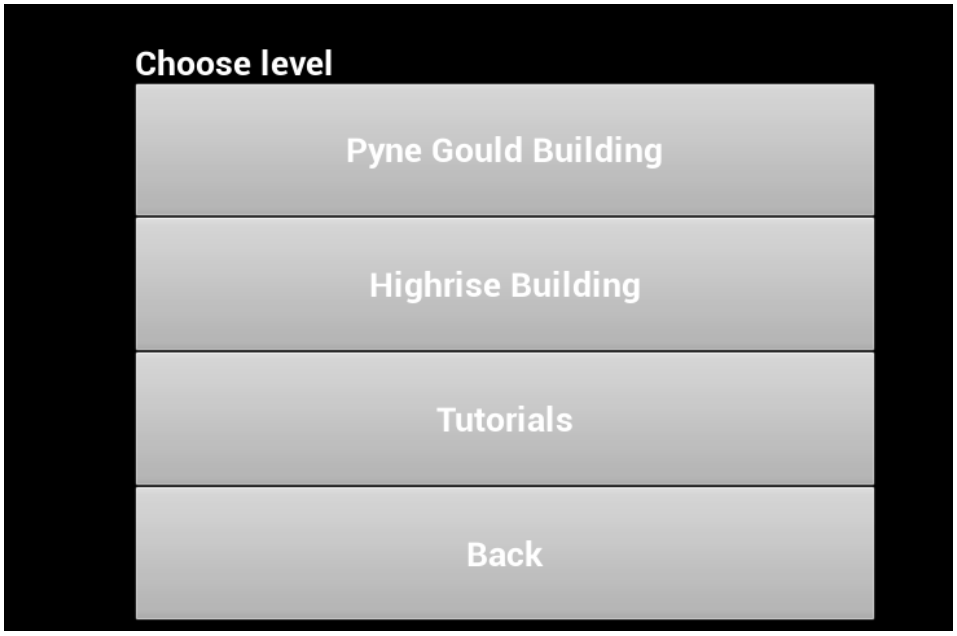

Figure 61: Level selection menu

Making the menu switch the view, instead of having all of the buttons on the screen creates a simpler user interface. Switching between the menu views is done by toggling the visibility of the views as shown below. Once the Play-button is pressed, it hides the first menu view, and goes back to it when the back-button is pressed.

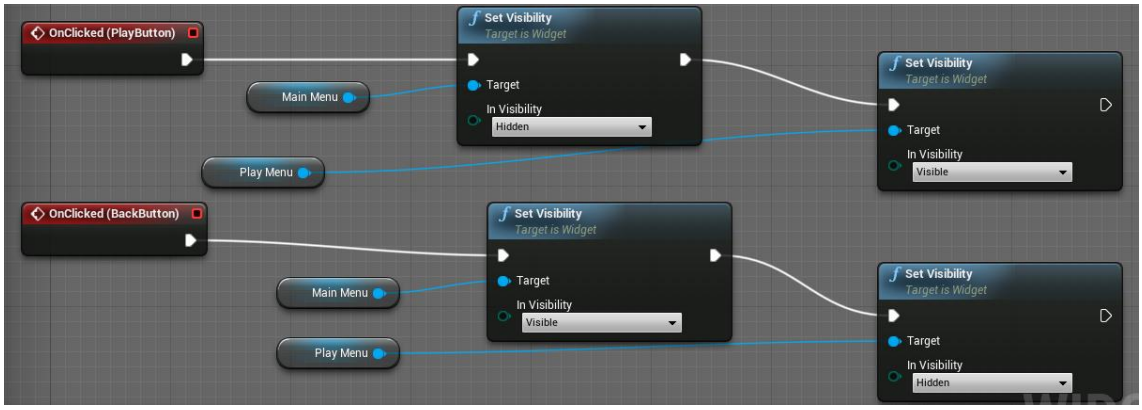

Figure 62: The visibility is toggled when the buttons are pressed.

5.8.2 Pause menu and objective list

A pause menu was created by using a widget, as a way for the player to pause the game, go back to the main menu or to close the game.

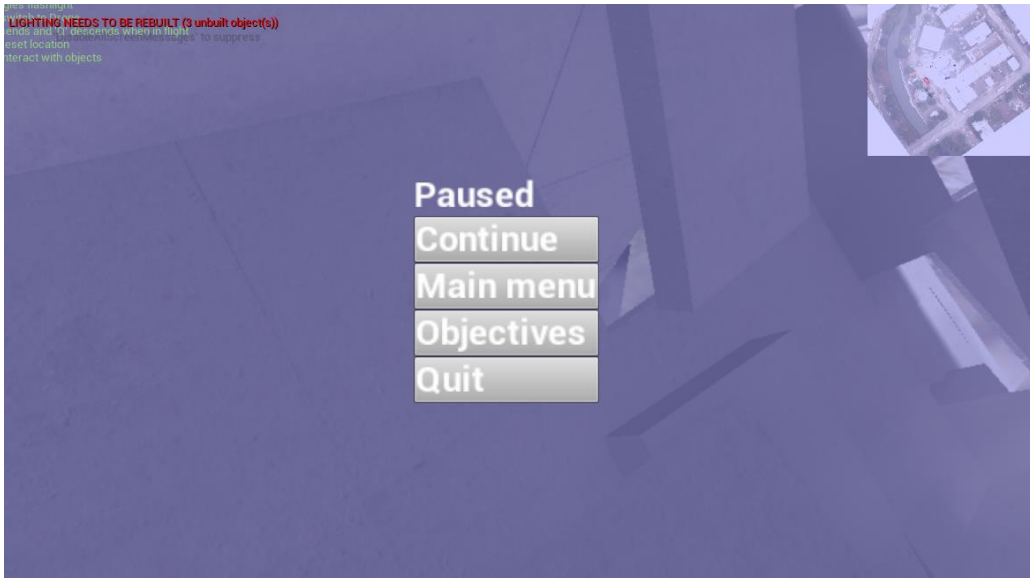

Figure 63: Pause menu

Pressing the Objectives -button opens the objective list with the same technique as the main menu, by toggling the visibility of the views. The objectives list checks the level database and collects objective completion data, such as the number of victims rescued and displays it for

the player. At the prototype phase, its main purpose, besides counting rescued victims, is to demonstrate what it could contain in the future.

Figure 64: Objectives list

6 Evaluation

Due to time limitations, there was no playtesting arranged with rescue personnel, but the game was tested by a few different people in different stages of the process, and the feedback was used to further improve the game. There were no major issues, that would stop the player from continuing, found in the gameplay and the functionalities, such as the victim system, were found easy to use even for inexperienced players.

The Unreal Engine proved to be an excellent choice for making the game, as the blueprint system made creating the game logic very simple. The learning curve was not too steep, which will help the next person who would continue the game development. The capabilities supported all requirements as there were no functionalities that were left out due to the inability to create it within Unreal Engine.

7 Conclusion

As the project begun, I had imagined finishing the game completely, but I feel that the project was a success nonetheless. A good base for future implementations was created and as there were no complications at this stage, the game development could continue even further, especially if C++ was used for functionalities that may not be as easy to develop without it.

Once completed, the game could be used by rescue personnel for training purposes and thus reduce costs in equipment and logistics and improve time management and safety for the training. Even if the game would be licensed and require fees, instead of offering it for free, the costs would not rise extremely high.

Working on this project taught me a lot about different aspects of game designing and a new way to solve problems, by identifying the need, splitting it into smaller pieces and then finding solutions for the smaller sections, which could be applied to many fields as well.

8 Future improvement suggestions

During the project, many functionalities and improvements were discussed, but due to time limitations were not implemented in the game. Having a small, interdisciplinary team working on the game would be ideal, to add more knowledge to the game, for example for accurate first aid functionalities. The following list contains some of the most impacting suggestions.

- Expand the victim rescue system, for example, different types of treatments depending on the victim's status and locating the victims by sounds or other signals.
- Improve the victim conditions system, for example, seriously injured victims require more urgent attention than relatively unharmed.
- Simplify adding more victims, as there is a lot of manual work for each new victim added.
- Switch the victim and playable character models to a more human look, instead of mannequins.
- Research ways to build proper lighting with the building models.
- Add more objectives, as the only objective right now is to rescue victims. Supporting the building from further collapsing, by using the shoring technique, is a good example of the possibilities.
- Introduce techniques and tools used by the Urban Search and Rescue personnel to raise awareness.

References

Blender. No date. Blender. Accessed 3.12.2016. https://www.blender.org/about/

Duolingo. 2016. Accessed 27 November 2016. https://www.duolingo.com/

Epic games. 2016. Accessed 9 November 2016. https://www.unrealengine.com/faq

FLAME-SIM. 2013. FLAME-SIM LLC. Accessed 25 August 2016. http://www.flame-sim.com/

Haagen. 2013. A world-leading Fire and Rescue training facility has been completed in Strathclyde. Accessed 27 November 2016. http://haagen.com/news/haagen-advises-and-builds-the-uks-most-advanced-fire-training-complex/

Haagen. 2016. USaR Firefighter Training. Accessed 27 November 2016. http://haagen.com/product/usar/

Inachus. 2016. Expected impact. Inachus. Accessed 9 December 2016. http://www.inachus.eu/impact

ISTQB. No date. What is Incremental model- advantages, disadvantages and when to use it? ISTQB Exam Certification. Accessed 23 November 2016. http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/

Ivanov, A. 2015. How the U.S. uses video games for military training. Version daily. Accessed 26 November 2016. http://www.versiondaily.com/how-the-u-s-uses-video-games-for-military-training/

Kostack, K., Walter, O. 2015. Bullet Constraints Builder for collapse simulation. Blender. Accessed 7 December 2016. https://www.blender.org/conference/2015/presentations/190

Pandey, A. 2015. Top 6 Benefits Of Gamification In eLearning. ELearning Industry. Accessed 27 November 2016. https://elearningindustry.com/top-6-benefits-of-gamification-in-elearning

Skantz, A., Törrönen, T., Hassinen, M. & Vaittinen, T. 2016. Phoroneus. Unpublished work.

USGS. 2016. Earthquakes map. Accessed 9 December 2016. http://earthquake.usgs.gov/earthquakes/map

UPSeis. 2007. How Are Earthquake Magnitudes Measured? UPSeis. Accessed 9 December 2016. http://www.geo.mtu.edu/UPSeis/intensity.html

UPSeis. No date. Earthquake Magnitude Scale. UPSeis. Accessed 9 December 2016. http://www.geo.mtu.edu/UPSeis/magnitude.html

Vainio, M. 2016. Suomen armeija rakentaa täydellistä sotilasta tietokonepelillä. YleX. Accessed 23 November 2016. http://yle.fi/ylex/uutiset/suomen_armeija_rakentaa_taydellista_sotilasta_tietokonepelilla/3-8690747

XVR. 2016. XVR on scene. XVR. Accessed 9 December 2016. http://www.xvr-sim.com/en/XVR_Platform/XVR_On_Scene/

Figures

Appendix

Below are links to videos made of the game to display the end result of the project. Also download link for the game can be found below. The game will be available for a year after the publish of the thesis.

Tutorial levels
https://www.youtube.com/watch?v=xmup-b_UoD0

Pyne Gould Corporation level
https://www.youtube.com/watch?v=-ap9x6bz0cg

Download link for the game
https://1drv.ms/f/s!An_aRlulJVbggslnet7f_BsqpjOqqA