IS71081A: 3D Virtual Environments and Animation (2018-19)

Coursework 2 - Group Unity Project

Group member:
Nima Jamalian, Habiba Sharaf, Elena Moroni, Gabriela Woch, Mariana Costa

Link to project file:
https://tinyurl.com/y3fp2w9r
Video demo of the project:
https://tinyurl.com/y3ts3srr
Submission date: 29/03/2019

Goldsmiths
UNIVERSITY OF LONDON

Table of Contents:

Introduction:

In this report, we first go over our game idea, explaining the concept of the game, our motivation for making this game, explaining how this project benefits from VR and finally cover our target audience. In the next section of the report we cover how we implemented the game, covering the code, graphics, animation and techniques which have been used in order to implement the game. In the last section we will cover our overall view of the project, the positive points and shortcomings, looking at the possible ways to develop the game furthermore. You can also find the contribution of each member of the team at the final section of the report.

Project overview:

Game Name:

- Smash Room VR

Platform & VR headset:

- PC(Windows), Oculus Rift

Game Genre:

- Actions,Sports, Simulation, VR, Casual

Game modes:

- Single Player

Game Pitch:

- In the game the player experiences VR simulation of being inside of a smash room. The main goal of the game is to break as many as objects that you can. The player can find a baseball bat inside the room and the player need to break the object using the bat.
- In the game, object will get thrown toward you from different direction, creating a more engaging experience, the player needs always to pay attention to when the next object will spawn.
- The ultimate goal of the game, is to simulate satisfying feeling for the player when they break game objects; same feeling that people will get when they go to actual smash room. The game is also much more safer experience cause you are in a virtual world, and you do not need worry about the safety procedures of smash room. The game also offers more engaging experience beauce you are able to do more in a VR simulation in comparison to real world for example feeling experience of having objects such as plate, laptop, chair thrown at you, and having ability to break is something you can not do in real world.

Motivation:

- In the UK, a survey of employees in junior and senior roles had stated that **more than a third** of the UK workforce is experiencing anxiety, depression with mental health problems being affected around one in six people in any week (Forster, 2017).
- Our main goal is to develop a virtual reality game that can contribute towards reducing

stress, while also inflincting fun by captivating its users into an immersive and engaging encounter.

- Boxing is one of the perfect sports to relieve symptoms of anxiety and depression by unleashing a lot of energy. By focusing the mind and body on precise punches, you improve concentration and help you forget the reasons why you are stressed (Fletcher, 2017 and Nall, 2019).
- When hitting a punching bag for example, your brain increases the production of endorphins, neurotransmitters which creates feel-good thoughts in your brain (Nall, 2019). By creating a game that has the elements of boxing and the smash room. Aiming and smashing an object would give the same stratification a punching a bag.

Why VR:

VR is an ideal platform for a Smash Room game because it can be used in situations which could otherwise cause health and safety risks and can also be used by people who are less able bodied to simulate the sensation of smashing something. With virtual reality you aren't in danger of being harmed by shattering objects or the stress of excess force on your joints. Also one can smash an unlimited amount of objects and impossible objects. Reality is limited to what is available to smash or smash with; however, virtual reality offers more possibility in terms of the 'weapons' one can choose to smash with, the locations and the objects to be destroyed. There is a lot more on offer in terms of creating a unique and personal experience.

Smash rooms are popular because they offer a moment of pleasure and satisfaction; it is not a solution for anger problems but a satisfying physical sensation that is intuitive; this is ideal to translate to virtual reality. Virtual Reality 'simulates a physical presence in the real world or an imagined world, allowing the user to interact with the world'. The key to the 'fun of it' in smash rooms is this cathartic sensation of smashing objects which can be simulated with VR as there is a sense of physical presence and interaction that is intuitive. Although interactions with games consoles can be learnt and become relatively intuitive they don't engage individuals in the same way that VR does. Corporeal engagement has been shown by research in neuroscience technology to incite 17% more of an emotional reaction than 360 degree video on a flat surface and 27% higher than 2D video. Virtual reality allows the sensation of release and satisfaction that is present in the smash room experience to be done in a controlled and curated environment. VR works best on activities or experiences that demand immersion. What this VR application will have to focus on then is the sensation of body ownership, place and responsiveness that the individual experiences to create the most believable sensation and intuitive interaction so the experience is fun and satisfying.

Closest Existing Games:

- Fruit Ninja VR: This game developed by HalfBrick Studios features similar style gameplay, cutting the fruits that spawn, as well as the way the controllers are used. However the main difference between Smash Room VR and Fruit Ninja VR is that we included the breaking effects to recreate the experience of a breaking room and providing a more entertaining experience.

- Rage Room :A game where the player gets to smash random things in a rage room experience, developed by Rockerm Reality Inc. Similar to our game, the player uses the controllers as a weapon to destroy the objects presented. The main difference resides that in Rage Room you may have to hit multiples times in order to destroy and the objects are already present in the room and mostly static.

- Beat Saber: While a musical game based on keeping the rhythm and melody, this game by Beat Games still has objects spawning directed at the player such as Smash Room VR . The main object is to use the weapon to cut in half the incoming Objects.

Key Features & Unique Selling Point:

- Combination of action and casual game
- Fun and engaging experience that kindles rebellion, where users can smash objects
- The player is immersed into a 3D environment that mimics the features integrated in a real-world “smash room” blurring the edges of reality
- Can help enhance the players reaction time and motor skills, potentially reducing anxiety through the act of smashing

Technical specifications:

- 3D Game
- Game engine: Unity, with Oculus Integration tool

Art Style and Logo:

The aim of this product was to replicate and replace a real smashroom experience. The VR space offers advantages that a real smashroom cannot fulfill, such as better safety and a more tailoring possibilities to create a unique experience. Therefore, in order to plant this game in the smashroom market the art style was adapted to be reminiscent of a real smashroom whilst distinctly showing that the game is set in a virtual space.

With the game space we chose to retain a realistic feeling, creating a context that was believable and didn’t detract from the game play. We observed that smashrooms are usually dilapidated; the room is simply a container for the activity without adornment. Therefore, we kept the design simple with a few cues to show that it was in some way an impossible space to make it distinct from a normal smashroom which shall be addressed in the VR room design section.

The logo for SmashRoom VR was again designed to be reminiscent of prior smashroom logo designs to clearly communicate what the game was. Using images that an audience will understand easily means that the logo can effectively work as a pitch that will attract individuals who are interested in smashrooms, VR or both. We researched logos from other smashroom companies and a common motif was a shattered effect on text or in the background and a baseball bat as one of the focal points. To show how SmashRoom VR is distinct the Oculus headset was made a key feature of the design and the ‘VR’ text was given a different colour to bring it to focus.

Target Market:

- Our target audience is the **fitness enthusiast** who is looking for a new physical experience to have fun as an alternative way to get physical.
- Although the age range would be for middle aged adults; the game can be a broad appeal that can also encourage the elderly to be more active as well.
- As the game involves a combination of the gym and boxing (which are the most efficient activities to reduce stress), it is intended to target the audiences who wants to find other ways to reduce their anxiety in their daily lives.

Plausibility Illusion:

Plausibility illusion aims to create a sense of presence within the space, where the player has influence over the virtual world. By tracking the players hand position, the actions conducted in the real world are replicated in the digital, granting the user a sense of embodiment. The environment has also been designed with the physical laws of gravity intact corresponding to that of reality, persuading the players to gain familiarity with the virtual space rapidly. In this manner the player becomes increasingly confident and interacts with objects in the space using their physical body, emitting the illusion of full body displacement. Newtons third law of physics states that ***"For every action, there is an equal and opposite reaction"***, this law indicates how actions in the digital space held by the player has notable impact on the objects within the environment. In Smash Room VR, the player has the capacity to break objects and the particles shatter, this is an illusion created to establish visual validation to trick the brain and persuade a multi-sensory experience. The user interfaces communicates the characters status throughout the game and guides the player towards the narrative, it allows them to remain aware of the actions they've performed.

Implementing the game:

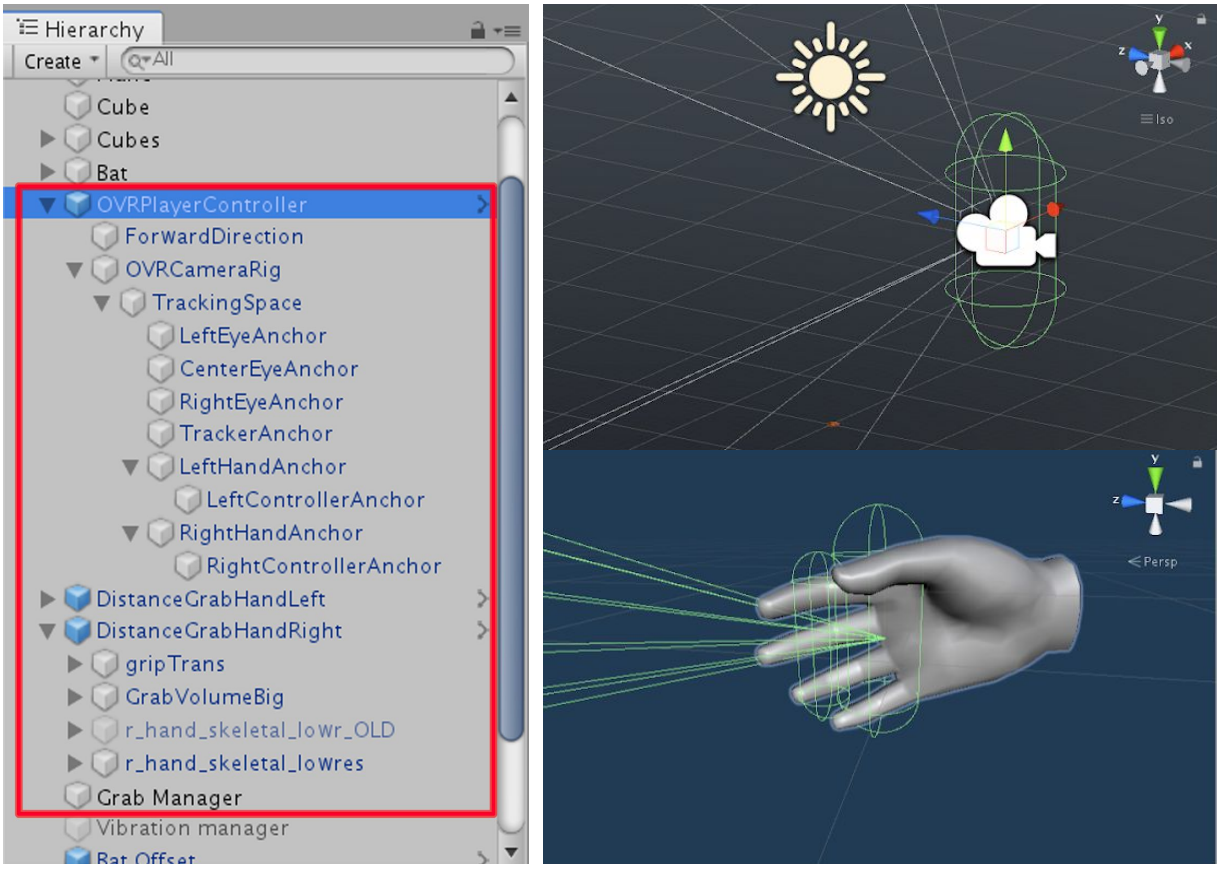
Introduction:

This section of the report will cover the process of producing a working demo of SmashRoom VR. We will cover the code used to create the mechanics necessary for gameplay and the assets, graphics and audio used to create the game scene. We will explain the choices and changes made in the process and conclude with an assessment of how successfully we achieved our aims.

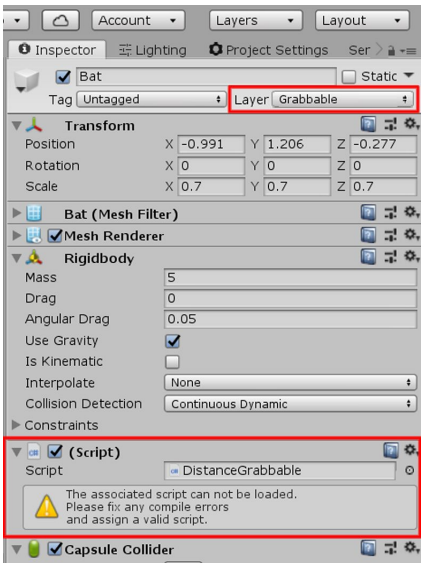
Oculus Setup:

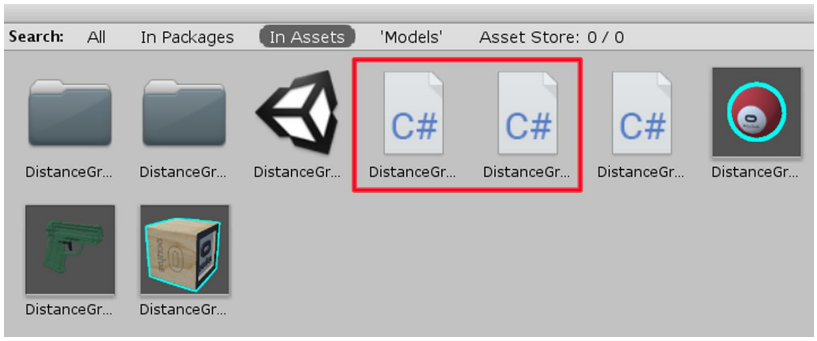
When setting up the Oculus Rift for the camera and the controllers in Unity, we followed a series of tutorials from the user name Valem (2019) whose methods were considered to be the most recent since *Unity* has a tendency to change its own code and functionality within every update. As many of us were not well experienced with programming, we eventually used a Unity package from the Asset Store called 'Oculus Integration' (Unity Asset Store [2], 2019) where ready-made packages such as camera controllers and functioning hands were available to use in the game. We set up the player's camera using 'OVRPlayerController' and then had to pick between two hands that had different setups; one being able to grab an object up close (AvatarGrabberLeft/AvatarGrabberRight) whilst the other has the ability to grab an object at a distance (DistanceGrabHandLeft/ DistanceGrabHandRight). After testing both functions we decided to choose the 'DistanceGrabHand' system as it's very hard to pick something up from

the ground and overall, grabbing from afar improves user performance. Assets that would be picked up will need to have a script that is associated with that accompanying setup.



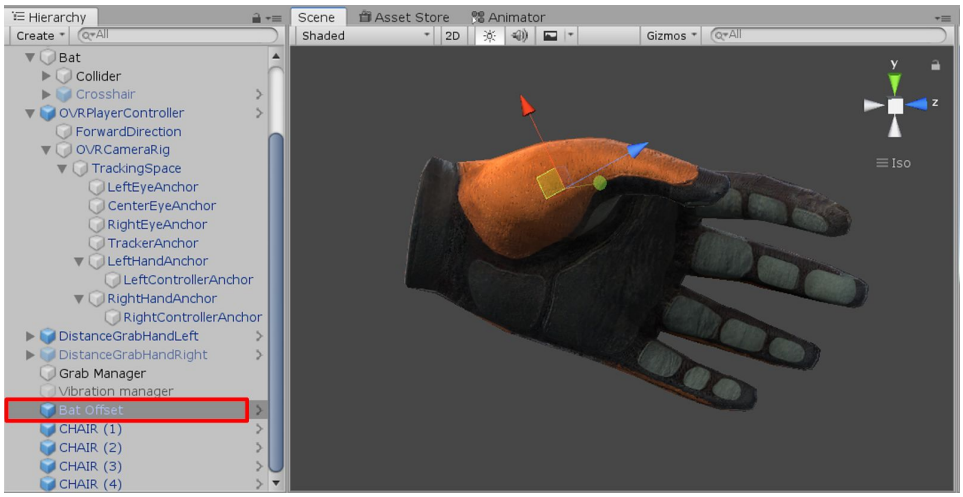
Assets that would be picked up will need to have a script that is associated with that companying setup and in the Layer system, the gameobject would need to be assigned to a new version called ‘Grabbable’.



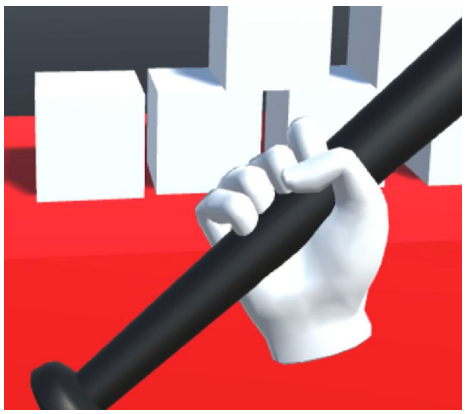


The hand is equipped with the script ‘DistanceGrabber’ whilst the selected assets are equipped with ‘DistanceGrabbable’. These all come from the ‘Oculus Integration’ package.

For the bat model in particular, we created an empty gameobject where this would be the bat’s offset. After a few attempts to adjust the position, the hand would now naturally hold onto the bat in a way a normal human would. It was important to get the offset right as it will affect the way the player will swing the bat. Done right the performance from the player will come naturally to them and achieve that plausibility.



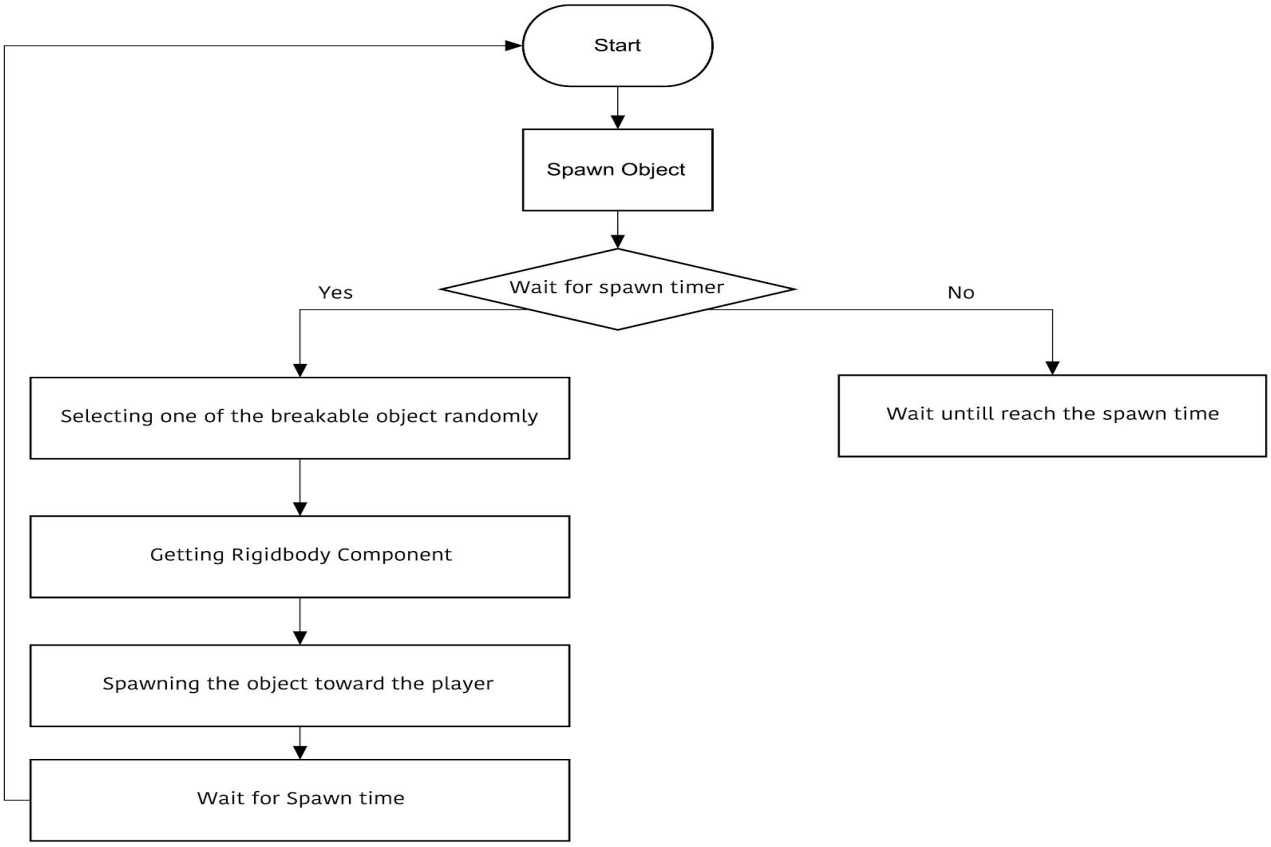
Before



After

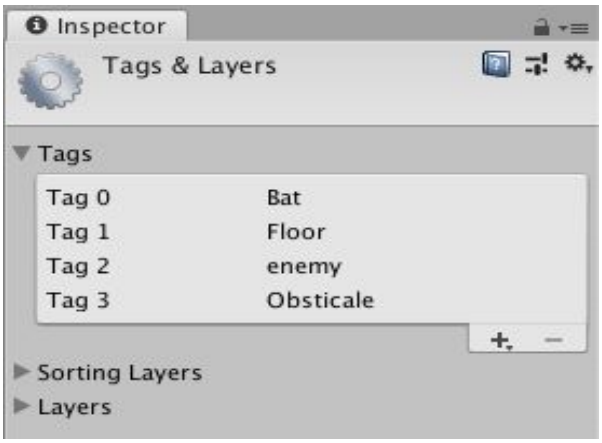
Spawner:

The purpose of this script, is to spawn random object at certain random rate, the script also gives you the ability to change the direction which objects get spawn in the game. For our game we set the spawning location in a way, which all the objects get spawn toward the player. This code simulates the effect of object getting thrown toward the player. You can find flowchart explaining how the code works below.



Collision detection System:

One of the most important aspects of our game is interaction with objects in a virtual world, therefore we need always to be aware what the objects in the game are colliding with. We used unity tag system in order to track different objects collision, in the game they are 4 tag, each tag is assigned to its specific object.

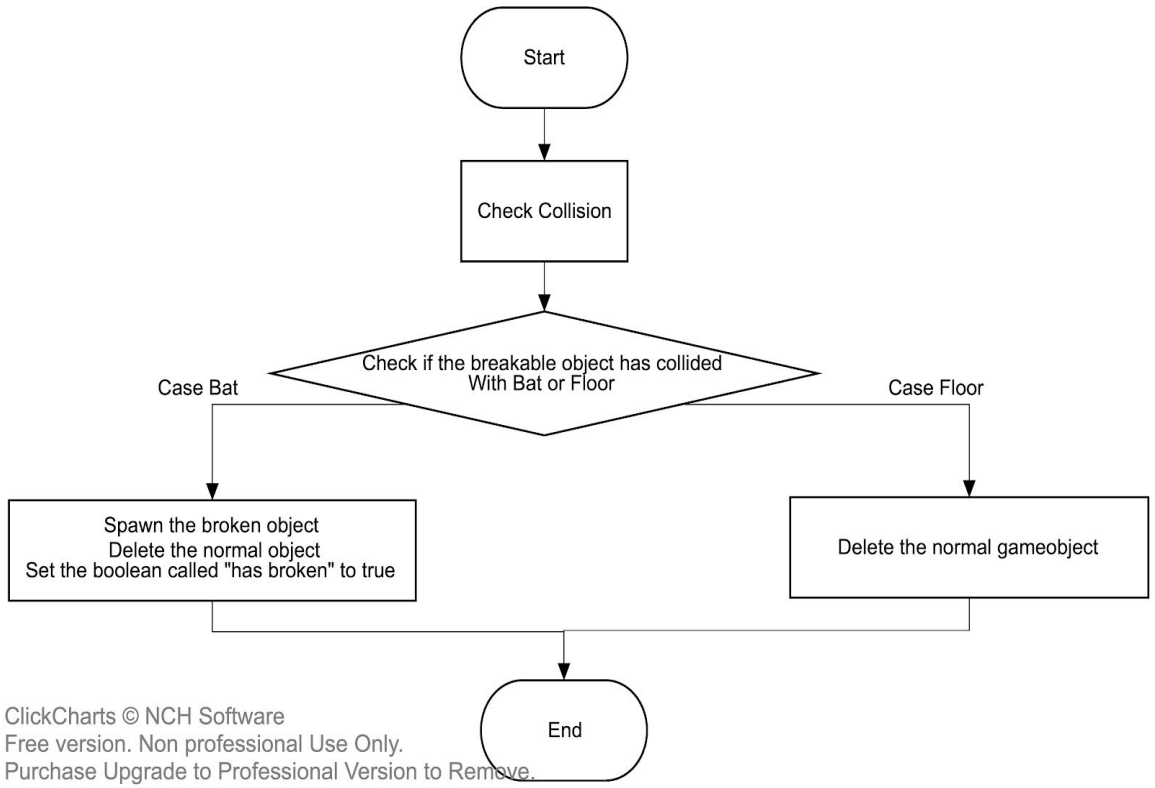


Tag system has been used in different scripts in game, in order to detect certain object collision. For example, in the bat script we are using unity tag system in order to check if the bat has collided with the obstacles, if the condition is true, then we add point to the score. Same method has been used in other scripts as well, whenever you find the word CompareTag or .Tag in the script, it means that the tag system is being used.

```
1  public void OnCollisionEnter(Collision collision)
2      {
3          if (collision.gameObject.CompareTag("Obstacle"))
4          {
5              Score.scoreValue += 1;
6              //Debug.Log("hit");
7          }
8      }
```

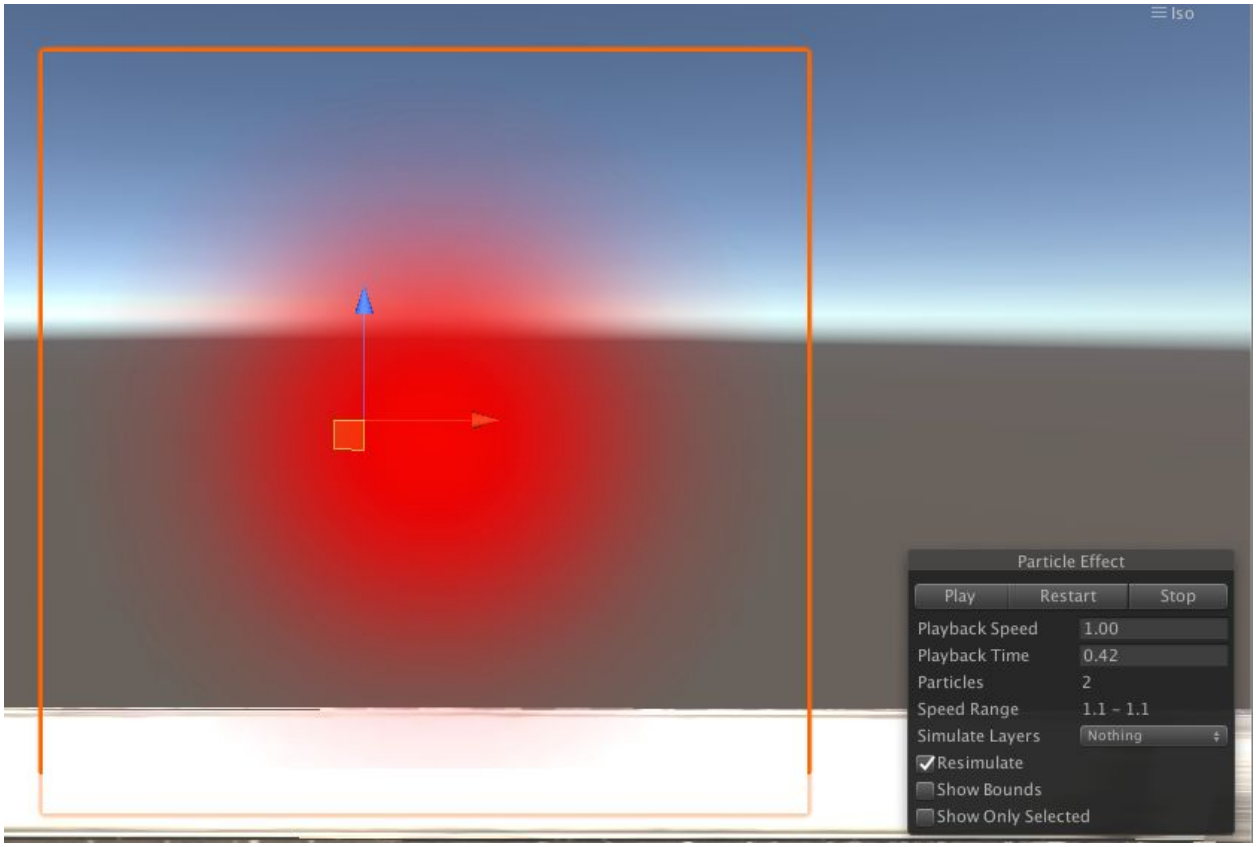
Smashing mechanics:

The main purpose of the ShatterOnCollision script is to break the objects when the bat collides with them. In the game we have two models for each object; one is the 'normal' game object and the other one is the 'broken' game object. The shattering script replaces the 'normal' object with the 'broken' object and then the physics would let the pieces disperse naturally. When the breaking script worked, we realized on the first test run that the harder we hit the target, the more the 'broken' object duplicated. This was due to the collision being triggered twice (apparently this did not happen in the older version of Unity which was a couple months back). For a while this took trial and error to fix it until thankfully the tutors and teaching assistant of the VR course came to help fix the dilemma by using a boolean called **has_hit** in order to prevent objects double breaking. You can find the flowchart explaining how the code works below.



Efficiency:
We wanted our game to run smoothly even on lower end computer that are capable of handling VR. Therefore we implemented our game in a way which is more efficient and it uses less computing power. In the game if the player misses hitting an object, that object will get deleted after few second. We used the same method for game object broken pieces, they will be on displayer for the player to see but after few second they will get deleted from the game, making the game run more smoothly and more efficient.

Enemy Object:
The purpose of enemy script, is to detect if the player has hit the canon ball (enemy object), if yes the code display a red particle effect and decreases one health from the player.



Audio Setup:
Sound effect are a necessary component to achieve plausibility. If an object doesn't make a sound when it gets smashed into pieces, then the virtual experience is instantly ruined. To manage this, with the guidance of Valem's (2019) tutorials, all it took was the 'AudioSource' as the setup component and use of an individual script to call it out.

```
1 public class PlayFeedback : MonoBehaviour
2  {
3      public AudioClip Breaking;
4          // Start is called before the first frame update
5
```

```
6           void Start()
7     {
8       Debug.Log("Audio On");
9      GetComponent<AudioSource>().PlayOneShot(Breaking);
10      }
11
12  }
```

The script will be attached to the broken object and the selected audio will play out automatically when the asset appears in the game. The only downside is that the audio source can only play one song and due to some time constraints, we didn't have time to further develop the script to play more than one clip.

Haptic Feedback/ Triggering Vibration:
In reality; when we lift something, we feel the weight and gravity of that object. In virtual reality, we only feel the weight of the controllers and anything else that we pick up in the scene doesn't add on. This is the same with touching. When we touch something in the real world; we feel the pressure, in virtual reality we feel nothing. So; in response to this conundrum, applying vibration in the controllers is the closest feature that for now technology can offer us. In our project we intend to trigger the vibration in the controllers using haptics feedback each time the player successfully hits the objects. This would give that little bit of satisfaction for the user and much more sense of presence.

When starting to build the code, we looked at multiple tutorials which showed many ways of triggering the function. The first attempt, followed a method of adding one line of code within the shattering script, based on an already addressed class **'OVRInput' (**which applies only to the Oculus control and gamepad that is assigned to the prefab hands). Note the script shown below was an old script before it was changed for the final product.

```
1 void OnCollisionEnter()
2   {
3       GameObject.Instantiate(replacement, transform.position, 4transform.rotation);
5       Destroy(gameObject);
6
7
8       OVRInput.SetControllerVibration(0.5f, 0.5f,
9ovrGrabbable.grabbedBy.GetController());
10
11  }
```

This didn't work. The second attempt was through a tutorial that used the same Integration package that we were using. To note, many other methods that talk about applying haptic feedback were based on some other Oculus setups, creating a new gameobject and script called Vibration Manager. This allowed us to set up a new function 'public void TriggerVibration' where we ended up taking the audio input clip to match the sound for the vibration and to be its own controller. To call the function, we needed to have some reference to the optic manager which we don't want. So, we used 'singleton' instead, when there won't be more than one

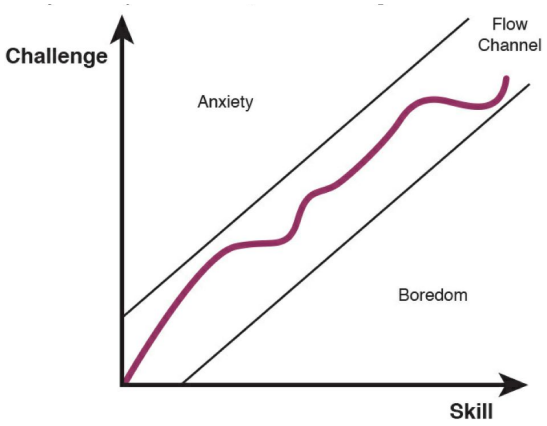
instance of our script. During this process, we had to tamper with some other ready-made scripts to set up references that we later called a class or function.

```
1 public class VibrationManager : MonoBehaviour
2 {
3
4   public class VibrationManager singleton;
5
6  // Start is called before the first frame update
7   void Start()
8   {
9
10      if (singleton && singleton != this)
11              Destroy(this);
12      else singleton = this;
13
14   }
15
16    // Update is called once per frame
17   public void TriggerVibration(AudioClip vibrationAudio, 18OVRInput.Controller controller)
19    {
20
21       OVRHapticsClip clip = new OVRHapticsClip(vibrationAudio);
22
23      if(controller == OVRInput.Controller.LTouch)
24      {
25              //Trigeer Left Controller
26         OVRHaptics.LeftChannel.Preempt(clip);
27      }
28
29      else if (controller == OVRInput.Controller.RTouch)
30      {
31              //Trigger Right Controller
32         OVRHaptics.RightChannel.Preempt(clip);
33      }
34   }
35 }
```

When we tested this, it unfortunately completely damaged the entire oculus controller including both the camera and the hand controllers. It was hard to tell what was the possible reason why it failed, but it became more jarring when reverting the scripts back to their original state and didn't fix the problem.

By the time we discovered this, we couldn't afford to try again as at that point we were running out of time. We can guarantee that if more people worked on this function and tried other methods, we would have been able to succeed. But sadly, it became too risky when it came to working around some of the dense readymade scripts, especially when there was potential to ruin the project when other people's work was included. In the end due to time constraints we had to abandon the task.

Core Fundamentals:



Psychologist Mihaly Csikszentmihalyi, discusses how balance between challenge and skill can be attained through his theory of the “FlowState”. This is a fundamental element game designers implement to craft the ideal level design to accommodate the players skill development in relation to the arising challenges/obstacles.

The flow channel aims to describe the subsequent: If the game features level design with high intensity challenges at a constant, then the player will be subjected to a state of anxiety. Similarly, the lack of skill evolution could resolute to the player being bored. When developing “Smash Room VR” it was essential to ensure that balance was obtained through the challenges that aided and motivated skill growth. Therefore, the game featured a timer, health and scoring systems to equipp the experience with the balance required to keep the player engaged.

Timer:

The timer embedded into *SmashRoom VR* is known as a “warning timer”, it attempts to increase tension and excitement. The player becomes conscious that a significant action is to be held once the timer has reached its final countdown, enabling them to become more focused and challenged. Meaning, that it manipulates the players mental state, perchance improving or decline the player’s progression within the gameplay.

The Integration:

In-order to link the user interface on the Unity finder to the script (*using UnityEngine.UI;)* had to be encoded, as depicted on line five.

First an integer variable was created called “countDownStartvalue” in-order to store the gameplay time which is 120 seconds, this means that the timer will start at two minutes.
The timer is meant to count down the remaining time from 120 seconds, in-order to-do so a conditional statement has to be performed that clarifies: *if the timer is greater than zero, then the timer should decrease by the second.* However, the game is meant to stop when the timer ends; therefore, an “if else” statement is executed to stop the game when the timer reaches 0:00 and launches the game over panel in the centre of the screen.

Timer Script explanations:

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; // linking the user interface on the finer to the script

public class Timer : MonoBehaviour
{
    int countDownStartVlaue = 120; // this is the game time stored inside a varible
    public Text timerUI; // this refers to the text in unity, the actual font that is subjected to change in accordance to the time
    public GameObject gameOver1; // this is a varible that stores the  "game over" illustration, that appears when the time is over

    void Start()
    {
        countDownTimer();
    }

    void countDownTimer()        // creating an if statement with condtions
    {
        if (countDownStartVlaue > 0)  //if the timer is greater than zero
        {
            Debug.Log("counting down"); // show the count down in the unity debug pannel to check if the countdown is functioning
            TimeSpan spanTime = TimeSpan.FromSeconds(countDownStartVlaue); // transforming the 120 seconds to minutes to be displayed as a timer
            timerUI.text = "" + spanTime.Minutes + ":" + spanTime.Seconds;
            countDownStartVlaue--; // if the timer is greater than zero, remove 1 second every second
            Invoke("countDownTimer", 1.0f);
        }
        else  //otherwise
        {
            timerUI.text = "0";       // if the timer is equal to zero
            gameOver1.gameObject.SetActive(true); // the game over pannel is going to appear on the screen
            Time.timeScale = 0; // the game will stop
        }
    }
}
```

Score:

The scoring system is a game design fundamental that bridges the player to the game and has a significant influence on the player's satisfaction. Scoring provides the player with positive feedback acting as a reward system that kindles the player's motivation. This system not only indicates what actions could help the player achieve the objective of the game, but also increase the like-ability of the game being replayed.

The Integration:

In order to achieve a score-point, the bat must collide with an obstacle and not the enemy. A script called "Score" was composed and attached to text visible in the game scene responsible for the games score. By using the expression (using UnityEngine.UI;), the script enables the use of code statements not visible otherwise. In the score script a variable called "scoreValue" is set to zero, as this is the game's starting score. On the bats script an if statement was created stating that if any collisions occur with the obstacles , then the games score increases.

Score Script and explanations:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; //enabling certain functions on the script related to the user inteface

public class Score : MonoBehaviour
{
    public static int scoreValue = 0; // creating a varible to hold the games score and it it set to zero
    Text score; //Creating a text varible called score to locate it on the finder

    void Start()
    {
        score = GetComponent<Text>(); // locating the score text on the game display

    }


    void Update()
    {
        score.text = "" + scoreValue; //every second the game will update and the score  will increase in realtion to the bat script
    }
}

```

Health System:

Health is an attribute that is assigned in role playing games to characters and indicates the functionality of the player. Some actions in the game world have consequences which are embodied through the loss of a health point. The health system helps inform the user on the current status of their player and guides them to identify the game's rules and objectives.

The Integration:

Every-time the player collides with an enemy, a heart is deducted and after three collisions the game ends. This was applied in-order create a time limit influenced through the characters the character actions to challenge and test the players on their ability to focus and engage.

Health System Script and explanations:

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

public class GameController : MonoBehaviour
{
    public GameObject heart1, heart2, heart3, gameOver;  //creating four varibles that represent the three hearts and game over panel visiable in the game
    public static int health; //creating a varible called health

    // Start is called before the first frame update
    void Start()
    {
        health = 3; // if the players health is equal to three
        heart1.gameObject.SetActive(true); // then all three hearts are activated
        heart2.gameObject.SetActive(true);
        heart3.gameObject.SetActive(true);
        gameOver.gameObject.SetActive(false); // the game over panel wont appear

    }

    // Update is called once per frame
    void Update()
    {
        if (health > 3)
            health = 3;

        switch (health) //is a statement that lists possibilities with an action for each possibility and
                        // and a default action, in the circumstance than nothing is true.
        {
            case 3:
                heart1.gameObject.SetActive(true); // When all three hearts are set to true then all lives are available
                heart2.gameObject.SetActive(true);
                heart3.gameObject.SetActive(true);
                break;

            case 2:
                heart1.gameObject.SetActive(true); //When two hearts are true and one heart is disabled, a life is deduced
                heart2.gameObject.SetActive(true);
                heart3.gameObject.SetActive(false);
                break;

            case 1:
                heart1.gameObject.SetActive(true); //When one heart is active, and two statements are false the on life is remaning
                heart2.gameObject.SetActive(false);
                heart3.gameObject.SetActive(false);
                break;

            case 0:
                heart1.gameObject.SetActive(false); //When all heats are false then the game is over
                heart2.gameObject.SetActive(false);
                heart3.gameObject.SetActive(false);
                gameOver.gameObject.SetActive(true);// the game over panel appears
                Time.timeScale = 0; // the game stops
                break;
        }
    }
}
```

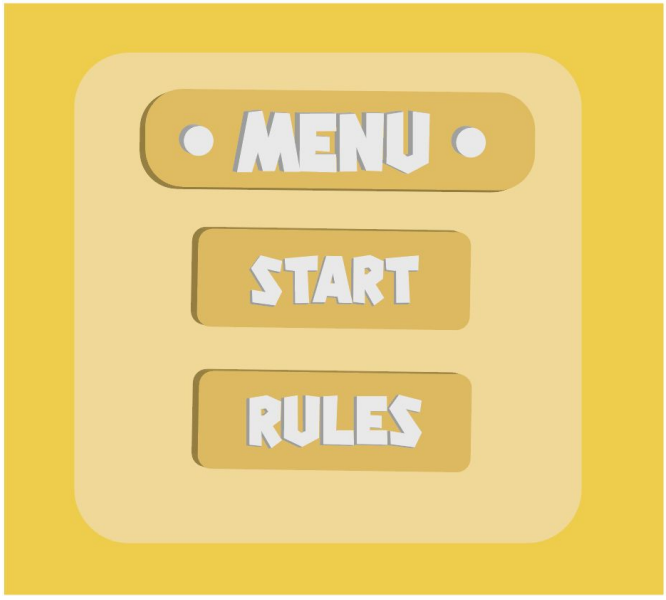
User Interface:
Smash Room Vr employs spatial UI, this type of interface aids the games narrative and provides essential information to the player. This interface is positioned within the game world and offers the user all the visual feedback required for progression. The interface can easily contribute towards increasing the game's immersion and limiting the players confusion.

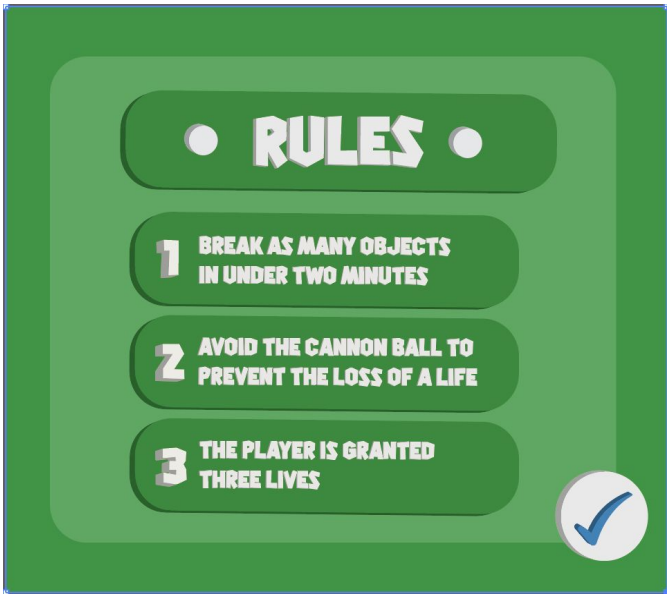
Implementation:
The interface on the menu panel is composed of Unity buttons, which enables movement between scenes. The start function found on the menu scene would transmit the player into the game scene, which would automatically start the game. By selecting the rules button on the menu page a new scene is apparent, one that educates the player in the games rules and depicts the games constraints. A start button function is also positioned on the rules scene at the bottom right corner to encourage the player to start the game once they have understood the objectives.

Unfortunately, the menu and rules scenes were not implemented into the game, due to the time constraints that limited our ability to assign functions to the physical game controller which would encourage the desired interactions.
The UI was programmed for laptop and mobile, rather than for VR as I presumed that Unity had a universal approach to generating its UI.

The games spatial interface informs the player about their current health, score and the time remaining. This helps contribute towards creating an immersive experience, that offers the player adequate feedback when required. All the UI assets were designed on illustrator and imported into Unity for enforcement.

Images below:

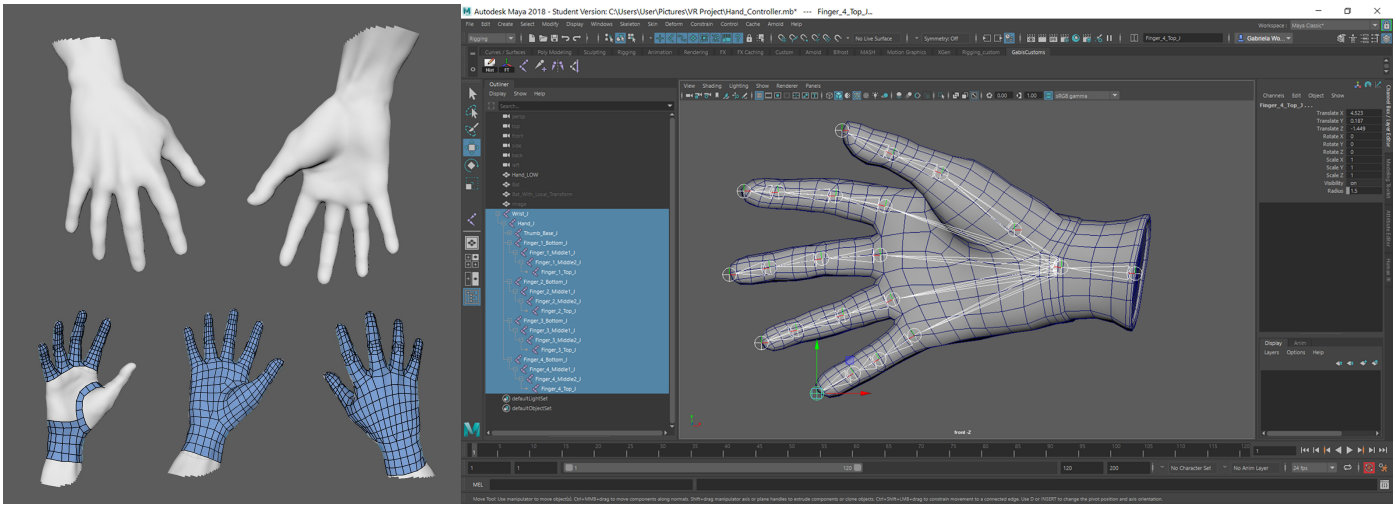


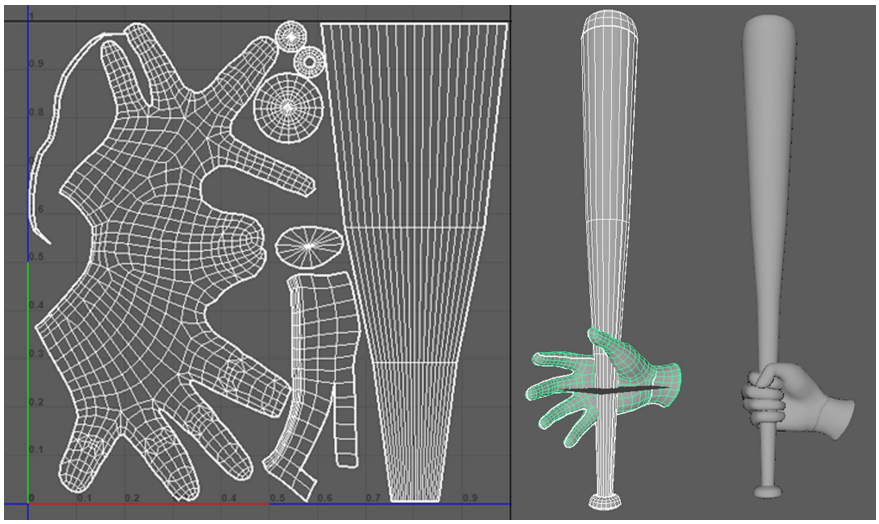


Assets, Graphics and Audio:

Hand/Gloves:

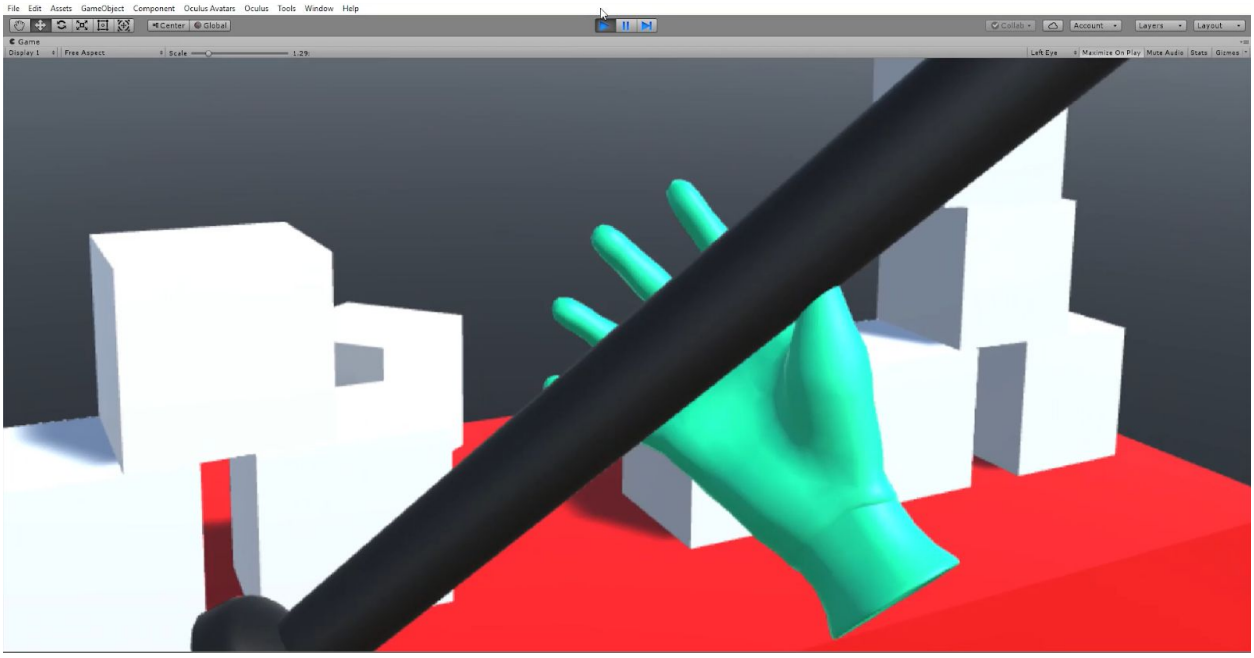
When we started the project, we wanted to have our own fully customizable hand that the player would control. This would allow us to have more freedom in the design aspect, as we wanted it to look more like a hard-wear glove as it would seem more realistic for the audience, to make them think that they are geared up for the smash room. Using an old hand model made by one of the artists, we built around it using the 3D software *Maya*. By working on the same file, we also built the main bat tool. After applying joints inside the hand and creating two animations (one clenching its fist and the other wrapping around the bat), we decided to test this out in *Unity* before giving the model more detail.





Pipeline Process - Maya

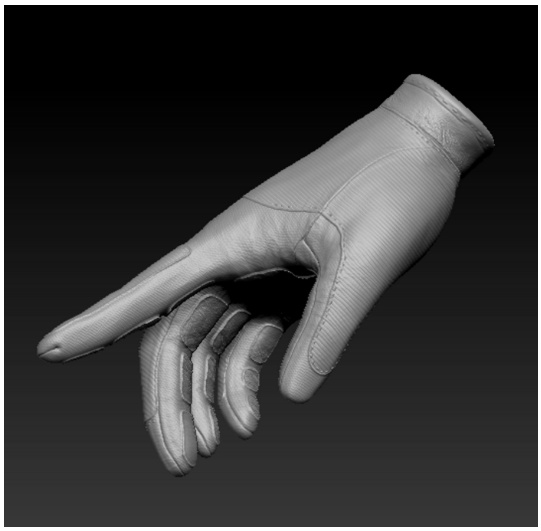
Upon loading the model into *Unity*, we soon realized that swapping the Prefab model (‘DistanceGrabHandLeft’ and ‘DistanceGrabHandRight’) for our customizable hand became more complicated than expected. After carefully swapping the 3D asset and applying the same scripts that the Prefab had, our customizable hand moved perfectly and was able to grab the object. However, we didn’t know how to trigger the animation. Our initial plan was to track down how the Prefab activates its finger gestures and animation. Upon inspection, we soon found out that it was functioning through ‘Inverse Kinematics’, a way to control the joints basing off where the joints touch an object. Because this method is highly advanced and too risky to tamper with in the scripts, we decided to abandon the task for the customizable model and made the decision to use the Prefab model that was already built in, to use as the main controllers.



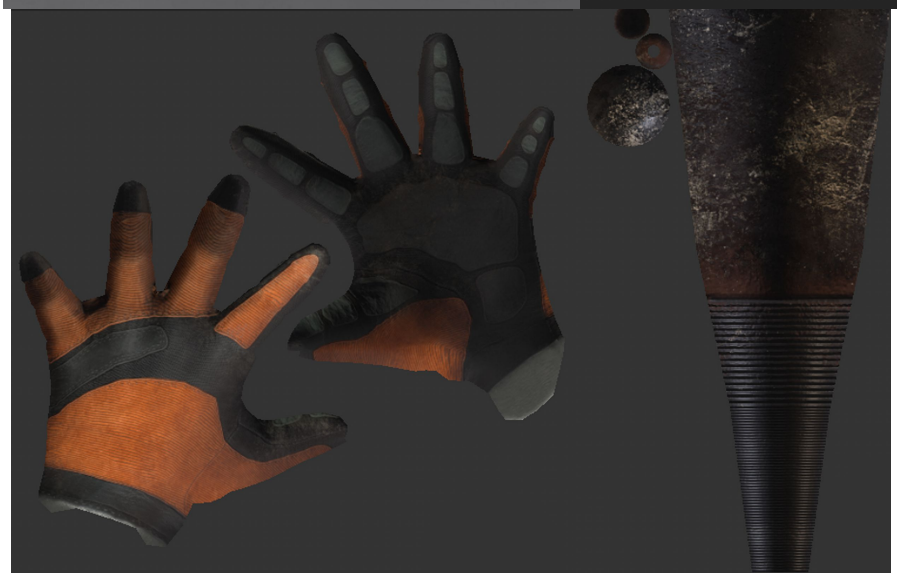
1st Play Test with Customizable Model - *Unity*

We exported the default hand using a script from the Asset store ‘FBX Exporter’ (Unity Asset Store [1], 2019) and sculpted the details to the model using *Zbrush*. Adding texture such as sport materials, in a rough way to make it look realistic but worn out for that gritty design. It was then taken to a rendering software *Marmoset Toolbag 3* to make use OF the baking system and then transferred it to *Substance Painter* where the colours and textures were finally added. This pipeline was the same for the bat model, adding bumps and scratches to make it appear that it has been chipped away due to its long usage. The two models took the most time to create as they were the ones that the player will see up close in the Oculus Rift.

Image Reference: Biltwell Inc. (n.d.)



High Polygon Model - *Zbrush*

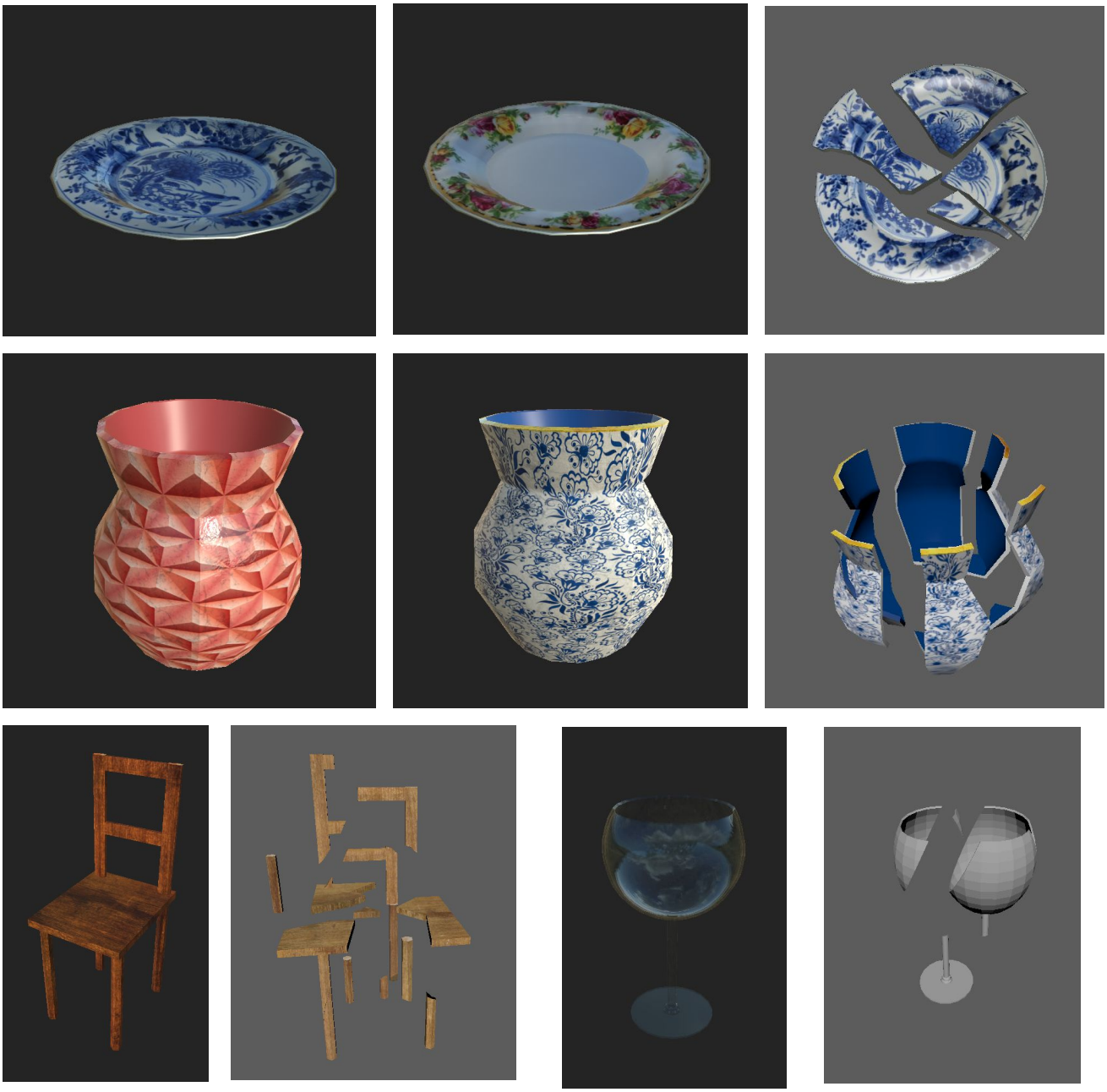


Final Render Model with Textures - *Substance Painter*

Breakable objects:

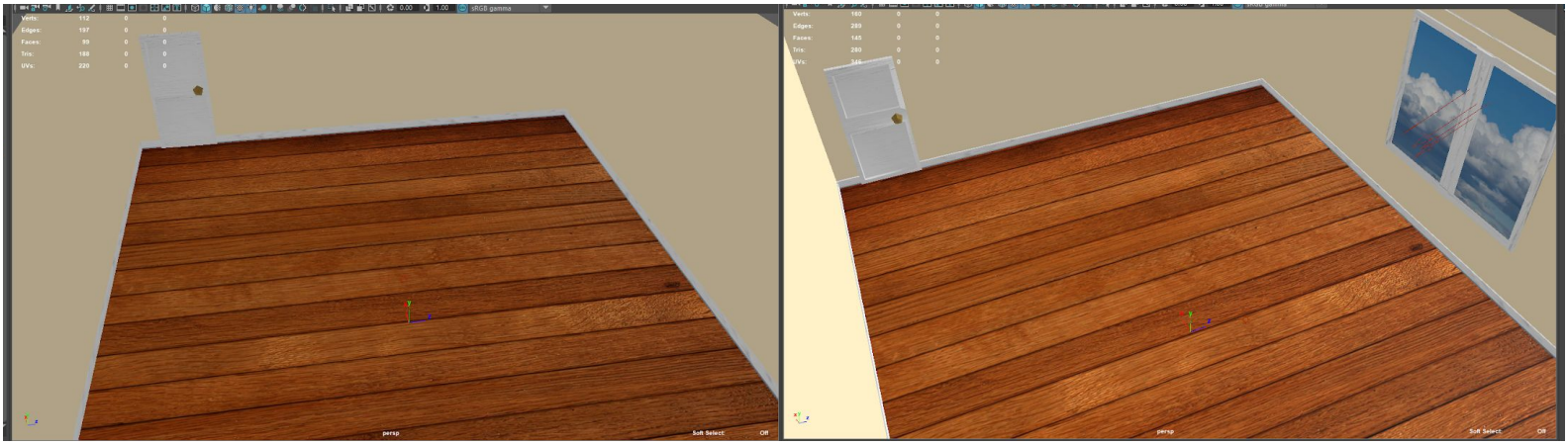
In order to recreate the experience of smashing objects and keep plausibility, we decided to give our models a duplicate that will replace the original after collision with the bat , This duplicate will be divided into several parts in a way the object would in reality - by allowing us to simulate the physics for each parts, without having to use multiple animations that would have to be rendered live.

We added a model of a jar, with two different textures, a plate also with two textures, a balloon glass and a chair. These objects were modelled in Maya, keeping in mind that the player won't be able to see them for a long time as they will be hit and destroyed fast, otherwise the mesh would be replaced for a shattered version that after collision with the floor would be destroyed from the game. For the plates we used image references (**Royal Albert - OLD COUNTRY ROSES PLATE; 1stdibs - Fine Chinese Porcelain Blue and White Plate, Kangxi Period and Mark Circa 1700**)



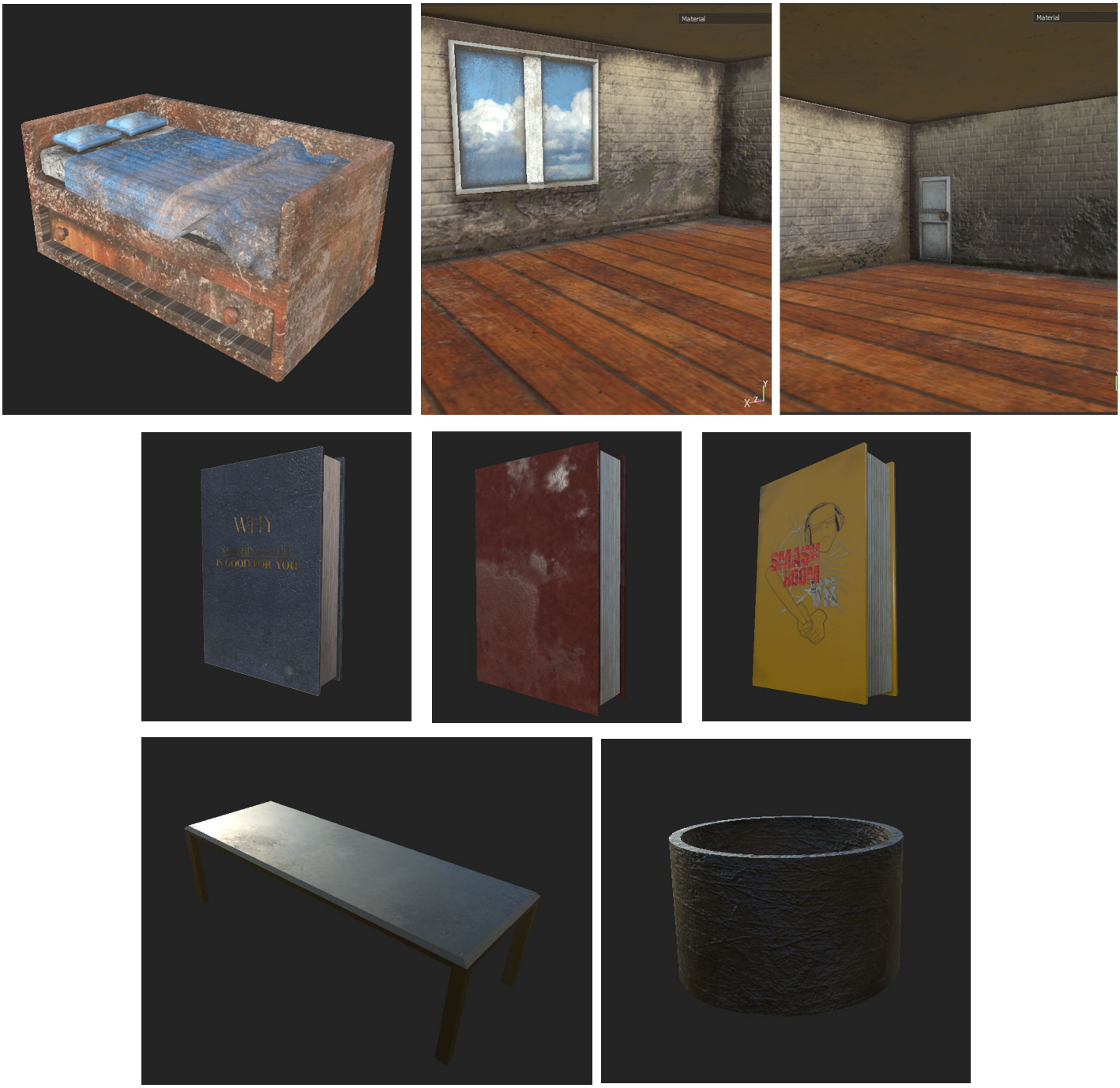
VR Room Design:

The VR room was modelled as a simple square room in Maya to contain the gameplay and keep the player in one position. The design was kept basic to keep the focus on the core gameplay. Initially when designing the space it was claustrophobic. Although the room isn't a key part of the experience it was important that it added to the experience and allowed the player to destress. Adding a window to the smashroom created the illusion of an open space and natural light. This didn't detract from the core gameplay but made a significant difference to the feel of the space. Furthermore, the view from the window of the cloudscape makes it clear that this is virtual space. There is a sense of surrealism from being up in the clouds which makes smashroom VR distinct from a normal smashroom. This simple design could be easily added to and changed with different textures, this opens up the potential that the game has to be adapted for individual experiences. A player would be able to tailor the experience to an environment that would be the most cathartic for them. Although this could have broken the plausibility of the game it worked well as the space wasn't photorealistic and therefore it didn't break the immersive experience within the context we created.



Textures:

To match the high-quality detail from the hand controller and the bat, we took the new imported models to Substance Painter and gave them the extra detail and texture.



Final Render and Textures - *Substance Painter 3*

Audio:

Soundtrack:

We wanted to have a soundtrack that enhanced the effects of the stress relieving feeling the game was intended to give. Initially we thought of classical music because it is generally thought of as soothing and there is a large range of classical music that lacks vocals which would allow it to accompany the game play without distracting the player. However, with more research it was found that classical music has also been shown to enhance genes involved in dopamine secretion. (ScienceDaily, 2019) It is unlikely that the soundtrack would have the effect of enhancing dopamine secretion within the two minutes of gameplay that Smashroom VR has but it showed that our decision to use this genre was justified. This also lead to further research in the possible effects the music choice could have on the stress relieving effects of the game.

It was important that the soundtrack of the game fitted with the high activity of the gameplay and slow classical music would have created a disjointed experience between the visuals and audio breaking the plausibility of the context. Therefore, we looked at dramatic, fast-paced classical music to accompany the game. We found that ‘music of 120 - 130 beats per minute increases . . . blood pressure and heart rate’ (Edworthy and Waring, 2006), which is similar to the effects of cardiovascular exercise. Studies show that exercise that heightens an individual's heart rate, ‘activates the metabolic pathway that replenishes . . . glutamate and GABA’ which are often depleted in people who have a ‘major depressive disorder’ (ScienceDaily, 2019). The effect of choosing music with a fast tempo allowed the game to further simulate the experience of exercising by increasing the player ‘s heart rate. This isn’t to say that the game can be offered as a solution to mental health issues. However, this illustrates the measured decision taken to have a soundtrack that increases the efficacy of the game and shows that there is reason to believe this audio could help relieve stress momentarily.

The songs used were; Carmen, ‘L’amour est un oiseau rebelle’ (YouTube, 2019), Vivaldi, The Four Seasons, Concerto No. 2 "Summer" (YouTube, 2019) Carmina Burana, O Fortuna (YouTube, 2019), Mozart, Symphony No. 25 in G minor (YouTube, 2019) and Rossini, William Tell Overture (YouTube, 2019). The clips were transferred to MP3 files then clipped and edited so only the climax and build up of the songs were part of the mix made. This was to ensure that in the two minutes of gameplay there was an array of songs to stop it being monotonous and that each song contributed to elevating the player’s heart rate. The editing was done on adobe audition from which the final track was exported and put in game. During the demo people commented that at certain points the rhythm of the music seemed to synchronise with gameplay. This was completely by chance but shows how the music was appropriate for the game play because it was interpreted as being associated to it and not as a distraction.

Sound Effects:
In order to mimic the breaking objects in real life, we needed to add sound effects that would be timed with the smashing motion. In Adobe Premiere Pro, the sound effect files were cut, edited and clean to fit our game needs.
The sounds used were converted from parts of the following videos: “Loud Bang Sound Effect” by Need Sound Effects (Youtube 2016), “Rage Room” by CBC News (Youtube 2015), “Ultimate Glass SFX Library Computer Smashing” by Frank Bry (Youtube 2011), “Smash Plate Sound Effects All Sounds” (Youtube 2018) and “Breaking Glass Sound Effect” (Youtube 2010).

Demo:

The demo ran smoothly despite issues we had had the day before with the game file becoming corrupt. It was well received and once players got into the rhythm of the game the play was intuitive and easy. Here is a link to footage of the demo as well as clips of the process showing some of the problems we encountered **https://tinyurl.com/y3ts3srr**

Feedback/Ways for Improvement:

What did the audience like:

- They became really motivated to swing the bat and dodge the enemies, thus becoming more activated.
- The audience enjoyed the background music as it help lighten up the mood.
- The scenery achieved the gritty appeal.
- Appeal that the VR experience is a game than a chore.

What could have been Improved:

- Without the haptic feedback on the controller, the game lost a lot of its plausibility.
- The spawning objects didn’t appear natural as they didn’t rotate or move at different heights when coming towards the player.
- Could have let players carry more than one bat.
- More audio for each breakable item would be better as some did became repetitive and boring after a while.
- The game became too easy when the ‘enemy’ didn’t move towards the player.
- We also made a scaling script which increases the size of the objects as they get spawn toward you, this would give the player feeling that object are getting thrown at them from distance. However we did not have enough time to test the script in order to get the scaling size correctly therefore we decided to remove it from the demo, and work on it in the future.

Conclusion:

In conclusion, SmashRoom Vr has been successfully developed using a diverse range of softwares and prowess. The group managed to collaboratively engage and work together to solve unforeseen issues by locating ideal solutions. As a result, the game has smooth functionality with the capacity to immerse players into a compelling and fun encounter. Immersion was attained through the photo-realistic models, physics and audio validation system encoded into the game. One element that would have contributed towards enhanced immersion is the hand controller vibrations, which was failed to be incorporated. Nevertheless, player satisfaction is guaranteed as all three illusions of place, plausibility and embodiment are apparent.

Contribution:

Nima Jamalian: Responsible for programming, such as making spawner and bat script, implementing unity tag collision system. I also contributed to create health, timer,smashing and other scripts in the game with Habiba. Troubleshooting unity errors and bugs in the game. Rebuild the game after our project file got corrupted with Elena. I also wrote the report for game idea pitch and implementing the game(spawner, breaking, tag collision system).

Gabriela Woch: Was in charge of setting up the Oculus device in every session and took charge in building the customizable hand and bat as well as applying the detail and textures for the Prefab version. I created the scripts that enabled the breaking of the objects into pieces, as well as triggering the audio afterwards. I was responsible for attempting to build the haptics feedback for the controllers but also helped set up all of the materials into *Unity.* With Mariana Costa, we both built extra texture maps for the remaining assets.

Elena Adams: Responsible for creating video documentation in Adobe PremierePro, the soundtrack for the game in Adobe Audition and building the VR room in Maya. I also worked with Mariana Costa to create whole and shattered versions of all the breakable assets for the game in Maya. Wrote in the report on the choice to use VR, the art style, the VR room design, and the soundtrack. I supported Nima when repairing the game after the file had been corrupted.

Mariana Costa: Worked on the Spawner model, and the breakable assets (whole and shattered versions) in Maya, together with Elena Adams. Built textures for these objects as well as the remaining in game assets with Gabriela Woch. Also responsible for edition of the Sound Effects for the game. On the report wrote about the similar existent games, sound effects and the breakable assets.

Habiba Youssef: Was responsible for programming the game's timer, health system, scoring system and user interface. In addition, to working with Nima on script errors to ensure the game functioned smoothly and made alternations in relation to his code to reduce bugs or glitches. Finally, I modelling assets (the bed and books) in Maya for Gabby to texture and designed the

user interface using Adobe illustrator. In the report I wrote about the games unique selling points, plausibility illusion, the timer, health system, user interface and the conclusion.

Softwares:

Adobe Illustrator:
Building User Interface and the main menu.

Adobe Photoshop:
Concept Design, creating Logos and use to create albedo texture maps for the assets.

Adobe Premiere:
Audio Editing.

Audition:
Audio Editing

Marmoset Toolbag 3:
Applied only to the hand and bat; used to fully bake the normal maps.

Maya:
Asset building and creating its UV maps.

Substance Painter:
Final texturing and painting for the objects.

Unity:
The main engine of the game, built in using C# scripts.

Zbrush:
Sculpt models in more detail (Applied only to the hand controllers and the bat).

ClickChart:
Used for creating flowchart for explaining the code in the report

Reference List:

Resources that were used to help with the development.

Adobe Blog. (2019). *How Virtual Reality Will Change How We Learn and How We Teach | Adobe Blog*. [online] Available at: https://theblog.adobe.com/virtual-reality-will-change-learn-teach/ [Accessed 25 Feb. 2019].

Biltwell Inc. (n.d.) MOTO GLOVES - ORANGE/BLACK/YELLOW. [online] Available at: https://www.biltwellinc.com/p854/buy/helmets-amp-gear/gloves/moto/-moto-gloves-orange-black-yellow/[Accessed 8th March. 2019].

Royal Albert - OLD COUNTRY ROSES PLATE [online] Available at: https://www.royalalbert.com/media/catalog/product/cache/11/image/1200x1200/9df78eab33525d08d6e5fb8d27136e95/r/o/royal-albert-old-country-roses-plate-798901568025.jpg [Accessed 18th March. 2019].

1stdibs - Fine Chinese Porcelain Blue and White Plate, Kangxi Period and Mark Circa 1700 [online] Available at: https://www.1stdibs.com/furniture/asian-art-furniture/ceramics/fine-chinese-porcelain-blue-white-plate-kangxi-period-mark-circa-1700/id-f_11077691/

CBS. (2012). "Anger room" truly a good outlet for anger? Available from: https://www.youtube.com/watch?v=IpOoa5oIheY [Accessed 21st February 2019].

Edworthy, J. and Waring, H. (2006). The effects of music tempo and loudness level on treadmill exercise. *Ergonomics*, 49(15), pp.1597-1610.

Evan Daley. (2017). *How to Fracture and Shatter Gameobjects.* [online]. Available from: https://www.youtube.com/watch?v=xqQkiHI5U9o [Accessed 8th March 2019].

Fletcher, B. (2017). Could boxing be a secret weapon in the fight against anxiety and depression? Available from: https://www.netdoctor.co.uk/healthy-living/fitness/a27734/boxing-mindfulness-help-anxiety-and-depression/ [Accessed 19th February 2019].

Forbes.com. (2019). *Five Reasons Why Virtual Reality Is A Game-Changer*. [online] Available at: https://www.forbes.com/sites/robertadams/2016/03/21/5-reasons-why-virtual-reality-is-a-game-changer/#c190d241bed9 [Accessed 25 Feb. 2019].

Forster, K. (2017). *Third of UK workers experiencing anxiety, depression or stress, survey finds.* [online] Available at: https://www.independent.co.uk/news/health/uk-workers-depression-stress-anxiety-survey-a7827656.html [Accessed 25 Feb. 2019].

Marketing Dive. (2019). *Report: VR delivers big on engagement, emotional response*. [online]

Available at:
https://www.marketingdive.com/news/report-vr-delivers-big-on-engagement-emotional-response/430113/ [Accessed 25 Feb. 2019].

Mental Health Foundation. (2019). *Stressed nation: 74% of UK 'overwhelmed or unable to cope' at some point in the past year*. [online] Available at: https://www.mentalhealth.org.uk/news/stressed-nation-74-uk-overwhelmed-or-unable-cope-some-point-past-year [Accessed 25 Feb. 2019].

Nall, R. (2019). Why Is Hitting a Punching Bag Good to Relieve Stress? Available from: https://www.livestrong.com/article/354110-why-is-hitting-a-punching-bag-good-to-relieve-stress/ [Accessed 19th February 2019].

Rebar et al. (2015). *A meta-meta-analysis of the effect of physical activity on depression and anxiety in non-clinical adult populations*. Available from: https://www.tandfonline.com/doi/abs/10.1080/17437199.2015.1022901 [Accessed 19th February 2019].

ScienceDaily. (2019). *Listening to classical music modulates genes that are responsible for brain functions*. [online] Available at: https://www.sciencedaily.com/releases/2015/03/150313083410.htm [Accessed 27 Mar. 2019].

ScienceDaily. (2019). *This is your brain on exercise: Vigorous exercise boosts critical neurotransmitters, may help restore mental health*. [online] Available at: https://www.sciencedaily.com/releases/2016/02/160225101241.htm [Accessed 27 Mar. 2019].

Unity Asset Store [1]. (2019). *FBX Exporter.* [online]. Available from: https://assetstore.unity.com/packages/essentials/fbx-exporter-101408 [Accessed 12th March 2019].

Unity Asset Store [2]. (2019). *Oculus Integration.* [online]. Available from: https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022 [Accessed 1st March 2019].

Valem [1]. (2019). *How to make a VR game in Unity - Part 1 - Setup, Hand presence, Grabbing. object.* [online]. Available from: https://www.youtube.com/watch?v=sKQOlqNe_WY&t=53s [Accessed 1st March 2019].

Valem [2]. (2019). *How to make a VR game in Unity - Part 2 - Custom Hand, Distance grab.* [online]. Available from: https://www.youtube.com/watch?v=rnOR1OANIAU [Accessed 1st March 2019].

Valem [3]. (2019). *How to make a VR game in Unity - Part 3 - VR Shooter.* [online]. Available from: https://www.youtube.com/watch?v=98gfkursxYI&t=375s [Accessed 8th March 2019].

Valem [4]. (2019). *How to make a VR game in Unity - Part 4 - User Interface.* [online]. Available from: https://www.youtube.com/watch?v=h7z4E8Qy8Ks&t=275s [Accessed 8th March 2019].

Valem [5]. (2019). How to make a VR game in Unity - Part 5 - Controller Vibration. [online]. Available from: https://www.youtube.com/watch?v=9KJqZBoc8m4&t=212s [Accessed 15th March 2019].

YouTube. (2019). *Antonio Vivaldi - "Summer" from four seasons*. [online] Available at: https://www.youtube.com/watch?v=g65oWFMSoK0 [Accessed 27 Mar. 2019].

YouTube. (2019). *Carl Orff - O Fortuna ~ Carmina Burana*. [online] Available at: https://www.youtube.com/watch?v=GXFSK0ogeg4&t=165s [Accessed 27 Mar. 2019].

YouTube. (2019). *Carmen: "L'amour est un oiseau rebelle" (Elina Garanca)*. [online] Available at: https://www.youtube.com/watch?v=K2snTkaD64U [Accessed 27 Mar. 2019].

YouTube. (2019). *Rossini: William Tell Overture: Final*. [online] Available at: https://www.youtube.com/watch?v=c7O91GDWGPU&t=87s [Accessed 27 Mar. 2019].

YouTube. (2019). *Wolfgang Amadeus Mozart - Symphony No. 25 in G minor*. [online] Available at: https://www.youtube.com/watch?v=7lC1lRz5Z_s [Accessed 27 Mar. 2019].

YouTube. (2016). *Loud Bang Sound Effect*. [online] Available at: https://youtu.be/q2XBBC0_f2Y [Accessed 25 Mar. 2019].

YouTube. (2015). *Rage Room* [online] Available at: https://youtu.be/sCb_JFGQpto [Accessed 27 Mar. 2019].

YouTube. (2011). *Ultimate Glass SFX Library Computer Smashing* [online] Available at: https://youtu.be/MerSrBzP3kI[Accessed 25 Mar. 2019].

YouTube. (2018).*Smash Plate Sound Effects All Sounds*
[online] Available at: https://www.youtube.com/watch?v=X1bcMltR-uE[Accessed 25 Mar. 2019].

YouTube. (2010).*Breaking Glass Sound Effect*
[online] Available at: https://www.youtube.com/watch?v=Dc1cN7FE-pk[Accessed 25 Mar. 2019]

Bibliography:

Additional resources that weren't included in this assignment.

Brackeys. (2017). SHATTER / DESTRUCTION in Unity (Tutorial). [online]. Available from: https://www.youtube.com/watch?v=EgNV0PWVaS8&t=95s [Accessed 8th March 2019].

Developers [1]. (2019). Haptic [online]. Available from: https://developer.oculus.com/documentation/unity/latest/concepts/unity-haptics/?fbclid=IwAR0rOblJ-j5-cOdVwSNEciJJ65ONplo-zYrMYGt8ahXHAbxWINBxUeZM97o [Accessed 20th March 2019].

Developers [2]. (2019). Haptic Feedback [online]. Available from: https://developer.oculus.com/documentation/pcsdk/latest/concepts/dg-input-touch-haptic/ [Accessed 20th March 2019].

Eric Hodgson. (2018). Oculus Grabber Tutorial 3: Adding Haptic Feedback. [online]. Available from: https://www.youtube.com/watch?v=JnPHZHuEPg4 [Accessed 16th March 2019].

Gloomglow Gaming. (2019). *[Oculus Rift | Unity] Hand Tracking & Button Setup* [online]. Available from: https://www.youtube.com/watch?v=9azwMqTaVCk [Accessed 1st March 2019].

Unity Asset Store. (2019). *VRTK - Virtual Reality Toolkit - [VR Toolkit].* [online]. Available from: https://assetstore.unity.com/packages/tools/integration/vrtk-virtual-reality-toolkit-vr-toolkit-64131 [Accessed 1st March 2019].

Valem. (2017). *Unity Tutorial: VR, Oculus Avatar and Grabbing Object setup IN 5 MINUTES.* [online]. Available from: https://www.youtube.com/watch?v=sxvKGVDmYfY&t=77s [Accessed 1st March 2019].

VRGameDev. (2017). *Beginner VR Unity Tutorial: SDK, Unity Setup And Hand Interaction* [online]. Available from: https://www.youtube.com/watch?v=Gq3K48FozlA [Accessed 1st March 2019].

Wavyeye. (2018). *Unity Oculus Rift Avatar Haptics and Left Hand Control Set Up.* [online]. Available from: https://www.youtube.com/watch?v=6_lo6pUfe1k&t=460s [Accessed 1st March 2019].

YouTube. (2015). *Smashing Sound Effect*
[online] Available at:https://youtu.be/RhsgmhhOX0U[Accessed 25 Mar. 2019].

YouTube. (2017). *Smashing Computers | Fliplash*
[online] Available at: https://youtu.be/FOc2zU1Lg3U[Accessed 25 Mar. 2019].

YouTube. (2017).*Game Over Sound Effects All Sounds*
[online] Available at: https://www.youtube.com/watch?v=6SkQJmenc60[Accessed 25 Mar. 2019].