Godzilla VR Game

Introduction

The Godzilla VR Game is a first-person VR Experience in which the player is immersed in the role of a giant monster. The goal is to destroy a virtual city and its inhabitants beyond the point of repair. Players have the power to level buildings that crumble on impact and chase tiny screaming pedestrians to their tiny screaming deaths.

The player wins and the game ends when they have caused enough damage to fill a destruction meter, and the city is beyond repair. Of course, one can always play again. Although fundamentally destructive, our goal was to create an amusing and light-hearted VR game. The gameplay loop is relatively short - the damage meter can be filled quickly- but is satisfying and can easily be repeated for as long as the player likes.

The Godzilla VR Game was created in Unity 3D 2020.3.19f for Oculus Rift S, with input via Rift S Controllers.

Ideation:

Early Ideation:

All initial brainstorming may be found in this [miro board.]
Exploration of project scope may be found in this [document].

Personas:

Liz

CITY DESTRUCTOR PERSONA

Name: Liz

Gender: female

Age: 21

Occupation: student (bachelor's)

Gaming Experience: plays single player rpgs as well as casual, shorter games

VR Experience: just got a headset!

Motivation for Playing:

Liz just got a VR headset and is excited about all the new possibilites. As she's already invested in some story-heavy rpg's, she's tentative about picking up another long and intense game. She's looking for something short, simple, fun & engaging to introduce her to what her new system has to offer, without complicated a plot or mechanics.

Criteria Selecting Games:

- able to pick up and play, but easy enough to put down
- likes to have a world to explore
- timed challenges and other "beat your best" mechanics
- Immersion

Frustrations with Games:

- competitiveness in multiplayer games
- no longer has 60+ hours to sink into new titles

Florian

CITY DESTRUCTOR PERSONA

Name: Florian

Gender: Male

Age: 35

Occupation: Innovation Manager

Interests: Nerd tech stuff

Gaming Experience: Played since he was little

VR Experience: Was an early adopter for consumer VR headsets

Motivation for Playing:

Uses the little free time he has to blow off steam in (VR) Games.

Criteria Selecting Games

- physical excercise
- uncompicated startup
- fun and easy mechanics

Frustrations with Games

- long loading times
- long storylines bore him

Final Game Requirements:
As is to be expected, our scope narrowed over the course of the project as we got a better idea of the time it took to implement our ideas. In order to meet time requirements and save processing power, more complex ideas like an eye laser or moving cars were not implemented.

Ultimately, our final game requirements and interaction were close to the MVP described in our first scope document.

- Player is a Godzilla destroying a city by swiping through buildings and people
- Player only sees their hands and is able to move by making 'running motions' with their arms

- City consists of both interactive and static elements.
- Buildings are breakable and shatter into pieces.
- Pedestrians run and scream, die when touched by Godzilla hands.

- Destruction contributes to a 'destruction meter' and the player succeeds/ the game session is over when the meter is full

Implementation:

Team Members & Contributions:

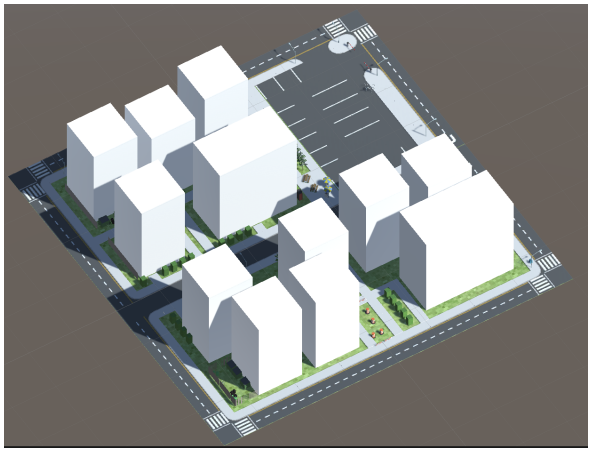
Paul Schramm	• Set up the project and created github repository • Implemented Movement Model • Implemented player hand models • Set up scene by implementing dramatic sky • Guide for participants during usability testing sessions
Talha Sarac	• Implemented destructible buildings • Implemented moving pedestrians • Implemented game mech**anics with "destruction bar"**
Caroline Kendrick	• built city scene • found & collected assets • initial documentation, organization, and time management • selected and created sound effects for city scene • usability testing analysis • storyboarding

Environment:

City Setting:

Source: CITY package | 255 pixel studios | Unity Asset Store

For the proof of concept, a sample city section was created using some of the package assets. Stand-in buildings - simple cuboids - were used in place of the buildings in the package, as we needed to use Blender to make the buildings shatter apart.

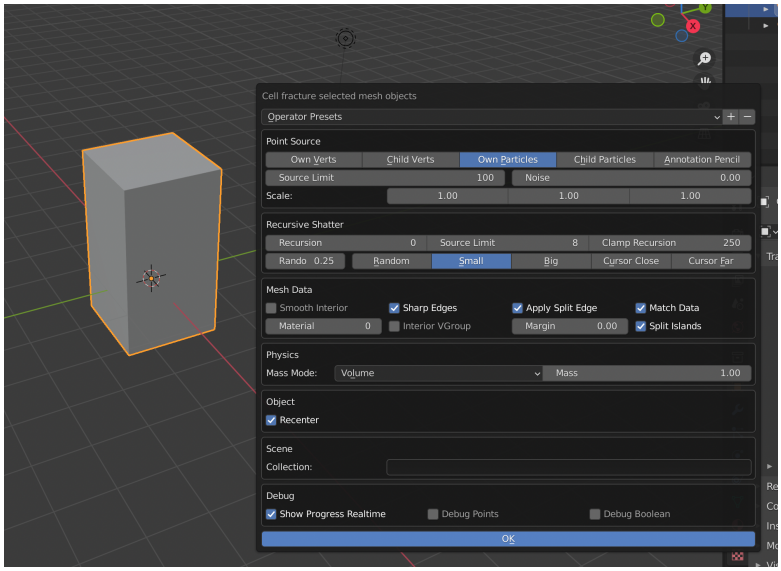


The section is made of a few city blocks, with assets like roads, shopping carts, streetlights, and plants to make it feel more realistic. While some of these items aren't interactive, they add to the visual interest of the scene and sell the core idea. Breakable buildings were then implemented into this scene.

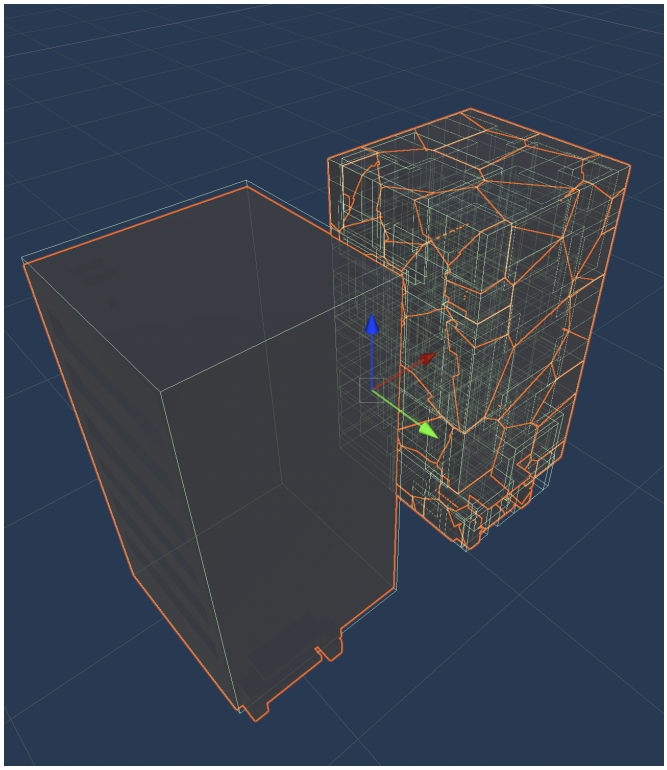
These city blocks can be rotated and tiled, in order to expand the city as much as needed, although a larger city greatly impacts the performance of the game.

Breakable Buildings:

First we found a way to break buildings in a natural way. Cell Fracture, an addon allows you to fracture meshes in a natural way using noise math



Now we have 100 pieces for each building and that's a lot to compute. So did two buildings. One of them is a single piece and one of them is fracturated. We are activating 100 pieces when the player interacts with it.



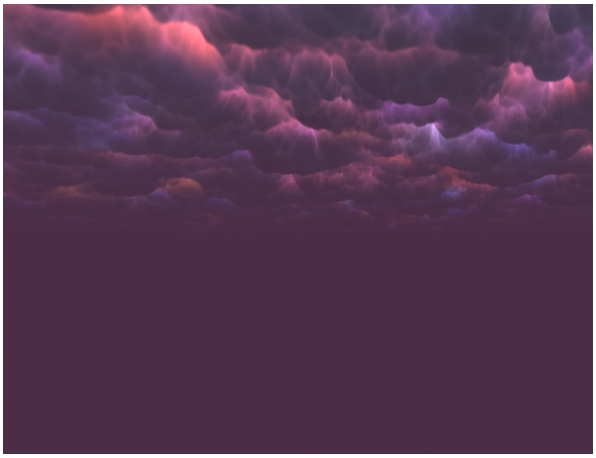
```
1 usage  ··· More
public void add(brick brick)
{


finditt.addPoint(1);
        if (_Bricks.Count > 10)
        {
            _Bricks[0].Sleep();
            _Bricks.Remove(_Bricks[0]);

            if (!_Bricks.Contains(brick))
            {
                _Bricks.Add(brick);
            }
        }
        else
        {
            if (!_Bricks.Contains(brick))
            {
                _Bricks.Add(brick);
            }

        }



}
```

Having 100 pieces at the same time is ok. But having 100 pieces with physics is too much for our cpu. To fix that we wrote a physics manager to optimize it. What it basically does is only allows 10 rigid bodies at one time

Sky:

We have a complicated sky shader that works based on tiling noise texture maps with a normal map also created with noise function. This asset was used as part of a previous project from the first semester called “Escape Tartarus” within the NUI course and was adapted to suit our needs. The asset was taken from the following website: http://vfxmike.blogspot.com/2018/07/dark-and-stormy.html

```
half4 SampleClouds ( float3 uv, half3 sunTrans, half densityAdd ){

    // wave distortion
    float3 coordsWave = float3( uv.xy *_TilingWave.xy + ( _TilingWave.zw * _Speed * _Time.y ), 0.0 );
    half3 wave = tex2Dlod( _WaveTex, float4(coordsWave.xy,0,0) ).xyz;

    // first cloud layer
    float2 coords1 = uv.xy * _Tiling1.xy + ( _Tiling1.zw * _Speed * _Time.y ) + ( wave.xy - 0.5 ) * _WaveDistort;
    half4 clouds = tex2Dlod( _CloudTex1, float4(coords1.xy,0,0) );
    half3 cloudsFlow = tex2Dlod( _FlowTex1, float4(coords1.xy,0,0) ).xyz;

    // set up time for second clouds layer
    float speed = _FlowSpeed * _Speed * 10;
    float timeFrac1 = frac( _Time.y * speed );
    float timeFrac2 = frac( _Time.y * speed + 0.5 );
    float timeLerp  = abs( timeFrac1 * 2.0 - 1.0 );
    timeFrac1 = ( timeFrac1 - 0.5 ) * _FlowAmount;
    timeFrac2 = ( timeFrac2 - 0.5 ) * _FlowAmount;

    // second cloud layer uses flow map
    float2 coords2 = coords1 * _Tiling2.xy + ( _Tiling2.zw * _Speed * _Time.y );
    half4 clouds2 = tex2Dlod( _CloudTex2, float4(coords2.xy + ( cloudsFlow.xy - 0.5 ) * timeFrac1,0,0)  );
    half4 clouds2b = tex2Dlod( _CloudTex2, float4(coords2.xy + ( cloudsFlow.xy - 0.5 ) * timeFrac2 + 0.5,0,0)  );
    clouds2 = lerp( clouds2, clouds2b, timeLerp);
    clouds += ( clouds2 - 0.5 ) * _Cloud2Amount * cloudsFlow.z;

    // add wave to cloud height
    clouds.w += ( wave.z - 0.5 ) * _WaveAmount;

    // scale and bias clouds because we are adding lots of stuff together
    // and the values could go outside 0-1 range
    clouds.w = clouds.w * _CloudScale + _CloudBias;

    // overhead light color
    float3 coords4 = float3( uv.xy * _TilingColor.xy + ( _TilingColor.zw * _Speed * _Time.y ), 0.0 );
    half4 cloudColor = tex2Dlod(  ColorTex, float4(coords4.xy,0,0)  );
```



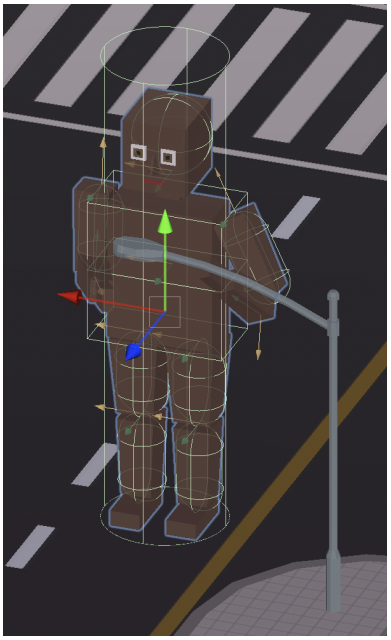
People:

Source package : [SIMPLE modular human | Characters | Unity Asset Store]

This package uses the built-in navigation system in unity but it has some animations that we need.

We used a baked navigation map for humans to walk. In order to do that we used invisible blocks for ai to calculate. We have 15 walk points for pedestrians. They select random point when they arrive in a spot.

```
if(!destermine_new_aim)
{
    int what_to_choose = UnityEngine.Random.Range(0, 2);



    walk = false;
    run = false;
    sit = false;
    steal = false;
    pick_up = false;



    if(what_to_choose == 0)
    {
        walk = true;

        int Which_point = UnityEngine.Random.Range(0, way_points.Count);
        aim_point = way_points[Which_point].gameObject;
        destermine_new_aim = true;
    }
    if (what_to_choose == 1)
    {
        run = true;

        int Which_point = UnityEngine.Random.Range(0, way_points.Count );
        aim_point = way_points[Which_point].gameObject;
        destermine_new_aim = true;
    }
```



```
if (checkleft) {
  int fornow = _peds.Count;

  for (int i = 0; i < _peds.Count; i++) {
    if (Vector3.Distance(handleft.transform.position,
        _peds[i].transform.position) < distance) {

      if (!_peds[i].iamalreadzdedmf) {

        finditt.addPoint(5);

      }

      if (!_Scream.isPlaying) {
        _Scream.Play();

      } else {

        if (!_Scream2.isPlaying) {
          _Scream2.Play();
        }

      }
      _peds[i].killthis();

      removeMe(_peds[i]);
```

To be able to hit pedestrians they have to have a rigidbody.

But having rigid bodys with animations is a big mistake. So we closed rigid bodies in pedestrians as long as they didn't die. But we have to detect hits anyway. So we scripted a code that calculates distance between godzilla hands and every pedestrian. If there is a pedestrian that is close enough we open the physics.

After a couple seconds we destroy those physics objects in order to gain some performance.

Gameplay:

User Interface:

Source: GUI PRO Kit - Fantasy RPG | Layer Lab | Unity Asset Store

The UI in VR works differently than regular GUI. It has to be a world space UI. In order to track camera movements we placed a child object in the player's center camera. And make the ui follow and make it look to the camera by using transform.lookat

We have a destruction bar that fills up when the player destroys things. To make the bar work we used a standard unity slider component. The way it works it takes a value between 0-1 and fills the bar accordingly. So in order to do that we wrote a script that counts points. Player gets 1 point for hitting a building and 5 points for hitting a pedestrian. Then we divide that point with our max point variable. This calculation allows us to change the game's difficulty by changing the max value.
We have set up the values in a way that the player has quick success and wants to play the game again.

Player Model:

Source: VR Monster Hands | Astronaut | Unity Asset Store

The implementation of the monster hands consisted of swapping the left and right hand models given by the standard camera rig with the ones provided within the asset bundle. We decided for the green option since it was most fitting for the story we wanted to convey

Movement:

The movement was implemented by the SwingingArmMotion Script which is attached to the main camera rig. It propels the user towards the direction they are looking at. To do that both of the inner squeeze buttons of the Rift S Controllers must be pressed simultaneously. The amount of forward motion is dictated by the distance the user moves their hands. The script keeps track of the user's controller position in the world and compares it every frame.

Sound:
Background Music: “Silly Talk” by Florian Stracker, from The Hero’s Path, Unity Asset Store
City Traffic Noises: “City Traffic Sound Effect” by RoyaltyFreeSounds
Ambulance: From FreeSoundEffects
Car Honking: From J Gaming

Scream: Wilhelm Scream | stock sound effect

The backing track of the game is a number of city sounds layered over the musical track. The musical track, “Silly Talk” was chosen for its upbeat sound, and high tempo. We wanted something that sounded fun and energetic, that complemented the chosen visuals for the game, and encouraged the player to fill the destruction bar quickly. The background sound elements were combined into one track and are set to loop in the game until the player has completed their destruction.

The scream is triggered when the player knocks over a person. We use the same trigger that we used to detect active physics on peoples and send a message to audio manager.

User Testing:

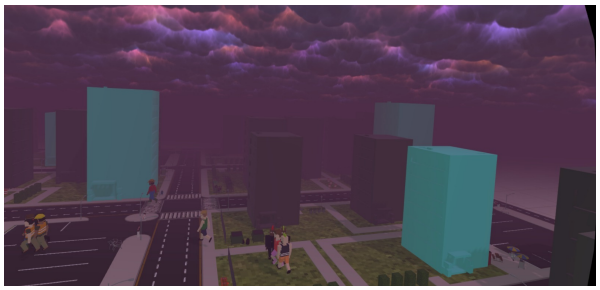
Test Description:

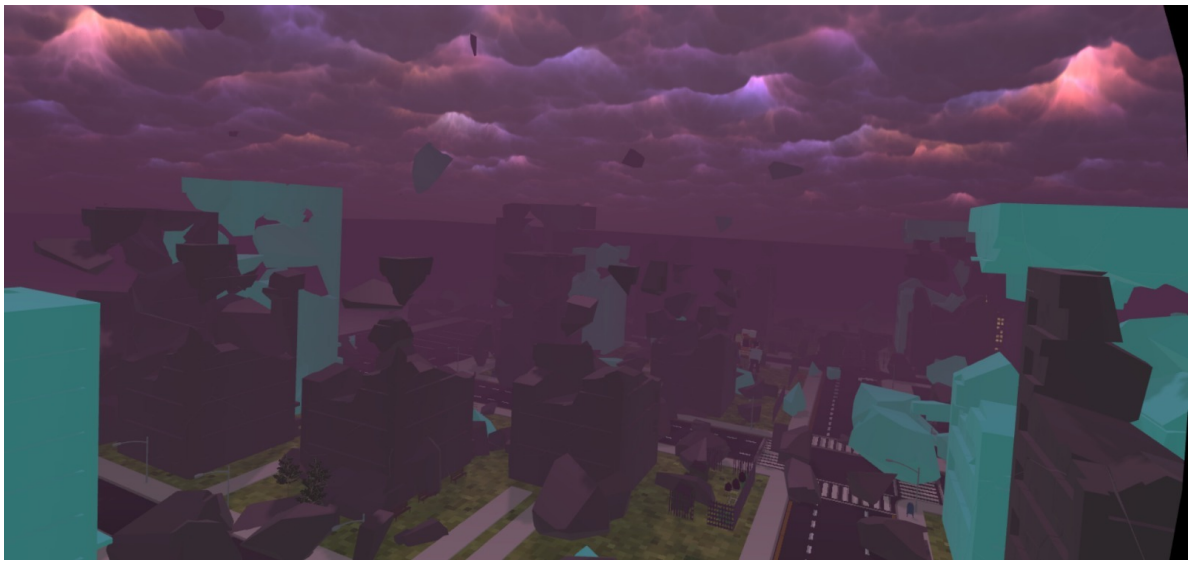
Our proof of concept was tested by three people. Participants played the game for a period between 1 - 5 minutes and were able to walk around the city while destroying buildings and people. When participants were finished playing, they completed the System Usability Scale questionnaire as well as three open-ended questions about the experience:

- What about this game did you like?
- What about this game did you dislike, or would change/add?
- Do you have any additional comments?

The questions of the SUS were modified slightly to be more relevant to a game instead of a website or product.

State of Product (Proof of Concept):





A video of the project at the time of user testing/proof of concept may be seen here.

For Proof of Concept/User Testing, participants were able to walk around a small city and smash buildings, as well as chase people down and kill them. Monster hands were not yet implemented, so the user's hands are represented by the white cubes seen in the pictures above.

There were a handful of known issues going into testing: as the movement was tied to hand gestures, players would move forward while trying to destroy a building or person. The pieces of the buildings were unpredictable and would float away or stay in place after the building was destroyed. Additionally, sound and destruction scoring were not yet implemented.

SUS Results & Analysis:
The raw SUS data may be found here.

SUS scores by Participant

Participant	**Score**	**Rating**
Participant 1	90	Excellent
Participant 2	77.5	Good
Participant 3	82.5	Excellent
Average Score	83.3	Excellent

While our proof-of-concept scored very well in this small usability test, it should be noted that for a reliable result this test should have included more (8 - 12) people. With only three participants, we received confirmation that our game was enjoyable and understandable to some, but can't confidently assert this. More valuable was the verbal and written feedback of our participants about their experience with the game.

Additional Feedback:
All participants reacted positively to the core concept of the Godzilla game. They found the idea of smashing a city fun and enjoyable. One participant mentioned that the game provided stress relief, and noted the base interaction was satisfying.

Problems & Implemented Solutions:

1) Movement caused motion sickness:
   More specifically, participants noted that bending down to swat at pedestrians was disorienting/uncomfortable. One mentioned the speed of movement also made them queasy. To solve this, movement was slowed down and the scale of objects in the scene was adjusted. Pedestrians were made taller in relation to the player, so that the player did not have to get too low to kill them.

2) Destruction causes forward movement:

To prevent the player from moving forward when swiping with their arms, the movement controls were changed so that the player has to depress the trigger on the front of the Oculus Rift controller while moving their arms. Swiping or grabbing without depressing the trigger does not cause movement. Addressing this issue will hopefully also address issues of motion sickness, as unexpected movement can also result in nausea.

3) Unbalanced Sound:
In addition to the initial round of user testing, we tested our game with our peers when the product was a little more polished. When sound was implemented, we received feedback that it was unbalanced - the traffic and car honk sounds of the soundscape were too loud compared to other sounds effects and the backing music. Based on this feedback, the volume of the background elements was brought down.

Final Product

A video of the final product may be found here.

Outlook

This documentation should help anyone that wants to develop this project any further. Improvements could be done in terms of city size, degree of destruction & overall game performance. The game should be tested with more users to refine the feedback. Additional features could be implemented in the form of:

- Laser Eyes

- Moving Cars / Helicopters
- Additional Menu + UI Elements (Pause, Replay, Settings)
- Sound when user destroys buildings
- More variety in the models of the buildings

Lessons Learned

- Gained a (slightly) better understanding of estimating scope of a project and the length of time it takes to complete a VR project
- Video tutorials that can be found on the internet are extremely helpful and better than a few years ago
- Some GitHub problems were hard to solve which forced us to work with local copies from time to time. We used Unity Packages to make that easier.
- The clear vision we had from the beginning was helpful for the team during the implementation phase