Worksheets

accompanying "Comparing Eye Tracker Input to Traditional Input for Measuring Player Performance, Immersion and Engagement in an Emergent Narrative".

Worksheets

MTA 201040

Aalborg University
Medialogy

Contents

1. Previous work

In this chapter, results from two papers referred to in our study are presented in more detail. Section 1.1 combines results from Ejdemyr's and Smith and Grahams' study, while section 1.2 presents a better overview of Jönsson's results. Some concessions had to be made while making Table 1.2, so where participants rated two conditions with the same value, those results were counted as "not in favour of" a condition.

1.1 Ejdemyr and Smith & Graham

Table 1.1 shows the participants' responses to the six questions by Ejdemyr and Smith and Graham. The questions were designed by Smith and Graham to capture the participants' opinions of the input modalities, and since Ejdemyr's study was similar to the one done by Smith and Graham, they used the same six questions [1, 2].

Ejdemyr designed a First-Person Adventure (FPA) in which players had to find six keys while avoiding enemies. The game had stationary and moving enemies the players had to avoid, and coming into contact with them decreased the players' score. Movement commands were issued using the WASD keys. [2].

Smith and Graham used three commercially available titles and made them work with Eye Tracker Input (ETI). They used a First-Person Shooter (FPS), a Role-Playing Game (RPG) and an Action/Arcade game.

In the FPS, players had to move along a corridor to reach a door. The corridor featured five enemies that they had to shoot. The players' orientation was controlled by the eyes when using ETI, or the mouse when using Traditional Input (TI). Movement commands were issued with the WASD keys, and shooting commands were issued using the right Ctrl button (ETI) or by left-clicking with the mouse (TI).

In the RPG, players had to enter a room by opening a door, then proceed to open and close two chests, and then exit the room. With ETI, players looked at a specific place or object to move to/interact with and clicked with the left mouse button. In TI they moved the mouse pointer to the desired place or object and clicked on it directly with the left mouse button.

The Arcade game featured targets (missiles) moving from the top of the screen towards the bottom, and players had to shoot these moving targets before they

reached the bottom of the screen. Control of the game was facilitated in the same way as the RPG [1].

In both studies, aside from input modality the ETI and the TI versions of the games were the same.

From the table we can see that ETI was more enjoyable in the FPA and RPG games but not in the FPS and Arcade game. The latter two required higher precision and faster reaction times, which are afforded more by TI than ETI. Most players felt more immersed using the ETI across all game genres. Except for the RPG, TI controls were reported to be more natural, but Ejdemyr suggests that might be due to more experience with TI controls than ETI controls.

Based on the answers it seems that the RPG condition was the one that benefited most from ETI. Among the 3D games it was the only one where the camera was not controlled by the players, but rather followed the players' avatar at a fixed distance and angle, just like in our game.

Author	Ejdemyr		Smith & Graham					
Genre	FPA		FPS		RPG		Arcade	
	TI	ETI	TI	ETI	TI	ETI	TI	ETI
Which did you enjoy playing with more?	28%	72%	58%	42%	17%	83%	58%	42%
Which was easier to learn?	95%	5%	67%	33%	33%	67%	67%	33%
Which was easier to use?	83%	17%	92%	8%	50%	50%	92%	8%
With which did you feel more immersed in the game world?	11%	89%	17%	83%	17%	83%	8%	92%
For which did the controls feel more natural?	61%	39%	75%	25%	33%	67%	58%	42%
Which would you prefer to use in the future?	5%	95%	67%	33%	33%	67%	58%	42%

Table 1.1: Test participants' preferences towards input modalities across four different game genres

1.2 Jönsson's thesis

As mentioned in our study, Jönsson compared TI and ETI in a Shoot 'em Up (SEU) game and an FPS. In the SEU with TI, aiming was controlled with the mouse and with ETI, aiming was controlled with gaze. The majority of players thought ETI was more accurate, more fun, more natural, and faster than TI, while TI was considered more difficult. With ETI, players were able to achieve a higher score.

In the FPS, three different setups were used. One where the mouse controlled the camera and the aiming (FPS Mouse, TI), one where the players' gaze controlled the camera and aiming (FPS Eye, ETI), and one where the camera was controlled with the mouse but aiming was controlled with the eyes (FPS Combined).

The majority of the players thought TI in FPS Mouse afforded better control, greater accuracy and that mouse input felt more natural. About half of the participants felt that the game was harder but more engaging with ETI, and more than half of them thought ETI was more fun to play with than TI.

The majority of the participants rated FPS Combined more natural, more engaging and easier to control than either FPS Eye or FPS Mouse. About half rated FPS Combined more fun and more accurate and easier than either FPS Eye or FPS Mouse. The players' ratings can be seen in Table 1.2. The SEU columns compare TI and ETI input modalities for the Shoot 'em Up game, FPS 1 columns compare TI and ETI used in the FPS Mouse and FPS Eye versions, and the FPS 2 columns compare these same TI and ETI scores to the scores of FPS Combined. We can see that aside from Combined, ETI was rated to be more fun tan TI across the board [3].

Game	SEU		FPS 1		FPS 2			
	TI	ETI	TI	ETI	TI	Combined	ETI	Combined
Which offered more control?	37,5%	62,5%	87,5%	12,5%	37,5%	62,5%	25%	72%
Which offered more accuracy?	25%	75%	62,5%	37,5%	50%	50%	62,5%	37,5%
Which was more fun?	0%	100%	37,5%	67,5%	25%	75%	50%	50%
Which was more engaging?	37,5%	62,5%	50%	50%	12,5%	87,5%	37,5%	62,5%
Which felt more natural?	25%	75%	62,5%	37,5%	25%	75%	75%	25%
Which was more difficult?	87,5%	12,5%	50%	50%	50%	50%	75%	25%
Which was more challenging?	100%	0%	50%	50%	37,5%	62,5%	75%	25%
Which was more straining?	75%	25%	25%	75%	87,5%	12,5%	87,5%	12,5%

Table 1.2: Responses comparing input modalities across 4 conditions from Jönsson's thesis

2. Design

In this chapter, we write about the design process and design decisions made while developing the game.

2.1 Visual direction

We decided to create low-poly assets for the game as an aesthectic choice. This also allows the objects to more easily be recognised from a distance. Furthermore, we wanted the game to be colourful for a more visually pleasing appearance. For inspiration, we have create a moodboard to help guide our visual direction, as seen on Figure 2.1.

Four biomes were made – farm, forest, desert, and village, with 4 objects native to each biome.

2.2 Gamifying the application

Originally, the game did not have any goals. Players were supposed to play until they thought they had played enough. The game would have played out the same, with the players moving about the map, activating objects and spawning new ones, creating a world unique to their playthrough. This would have made it harder to compare some results, and the lack of goal could have contributed to decreased engagement with the game. For this reason, we decided to implement additional elements: golden and dark objects.

Golden objects function similarly to regular objects, and they are made distinct by their golden texture. There are four golden objects in total, one belonging to each unique biome. Activating a golden object spawns text over it showing the player's progress: how many have they found of the four total as shown on Figure 2.2. Finding all of them enables them to finish the game, but should they choose to play more, they have the option.

Dark objects have a black texture, and activating them disables the player's cone of vision for a short amount of time as seen on Figure 2.3. This adds some level of challenge to the game, and enabled us to measure performance as well: we logged the dwell time on these objects and thus we were able to see whether people recognised these objects sooner in one condition than the other.

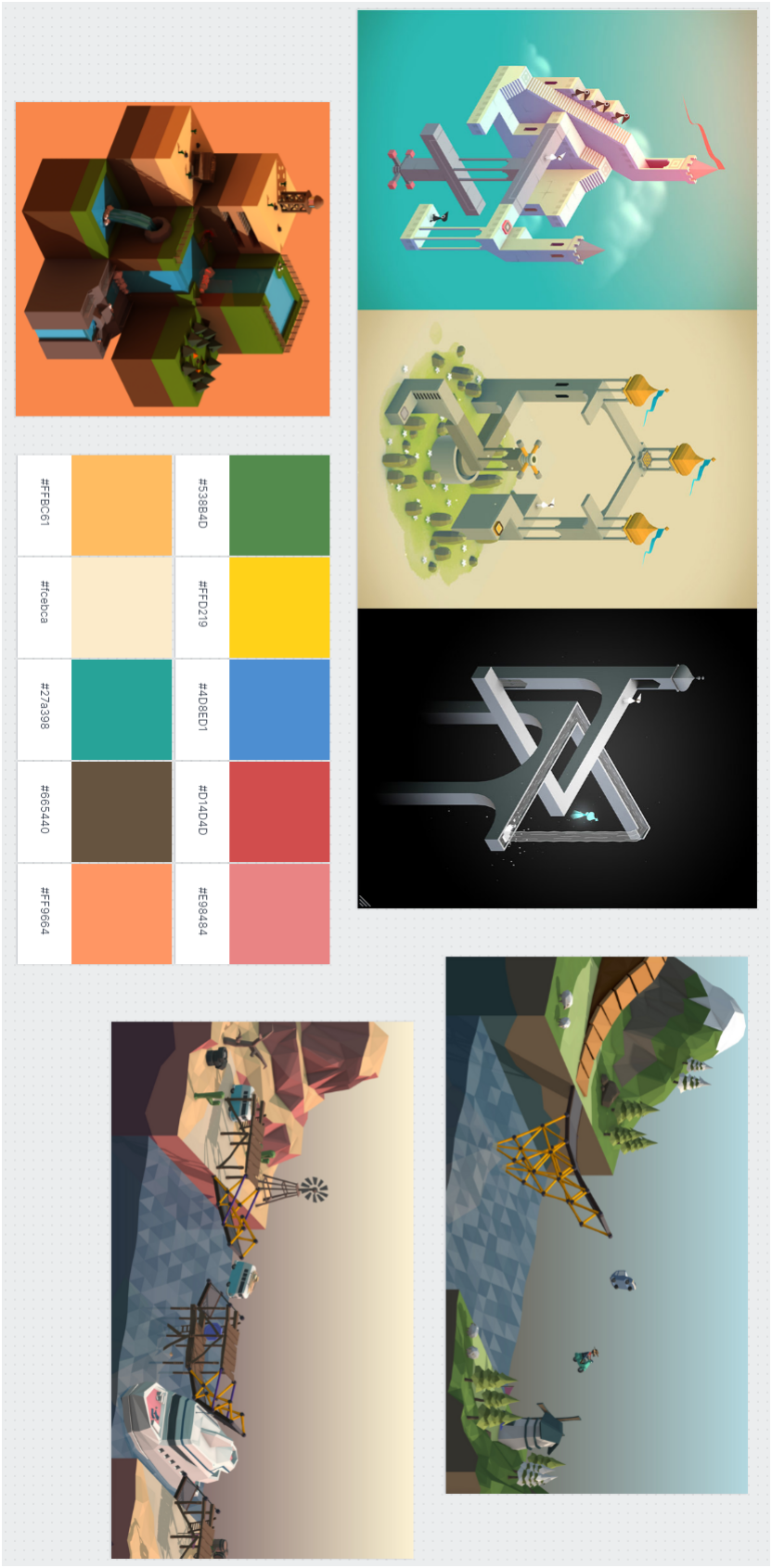


Figure 2.1: The moodboard we created for inspiration

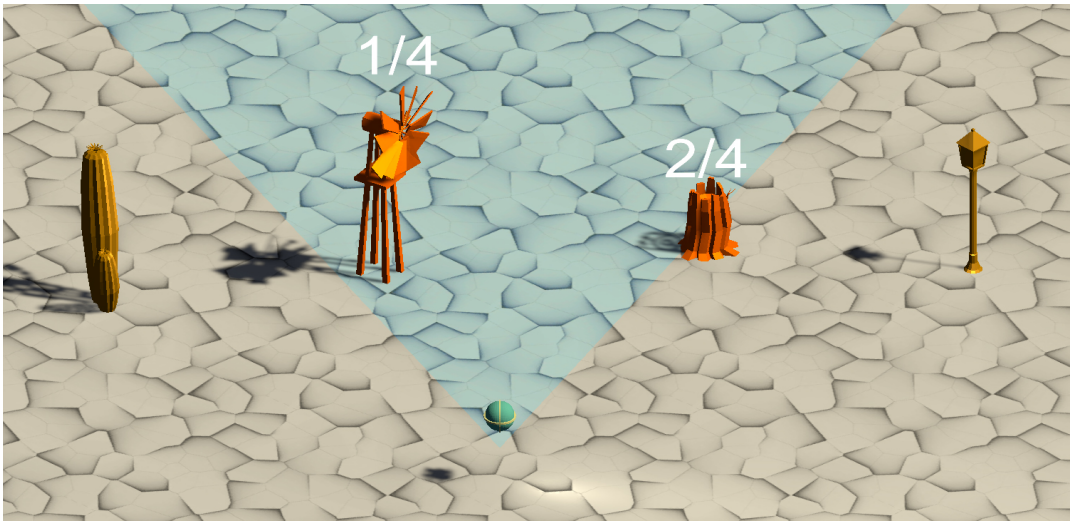


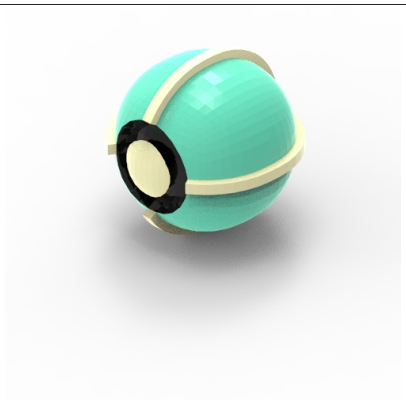
Figure 2.2: Each of the four golden objects, with the farm and forest objects activated – indicated with the progression above each activated object.

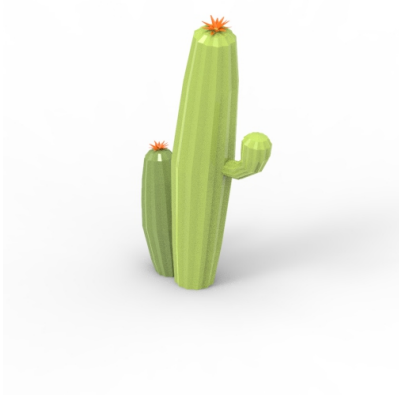
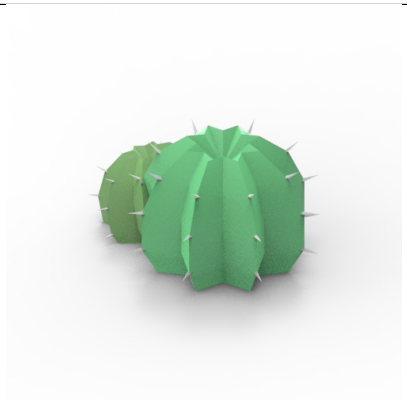
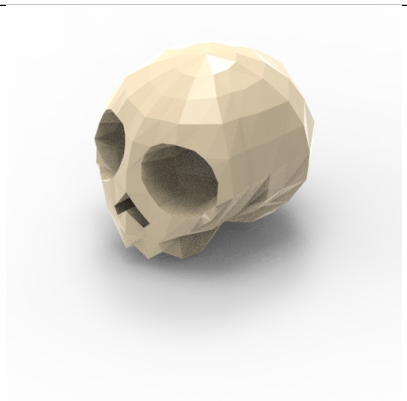
Figure 2.3: Examples of dark objects alongside a regular object. As the dark object was recently activated the cone of vision has been disabled, and the regular objects and ground stay desaturated.

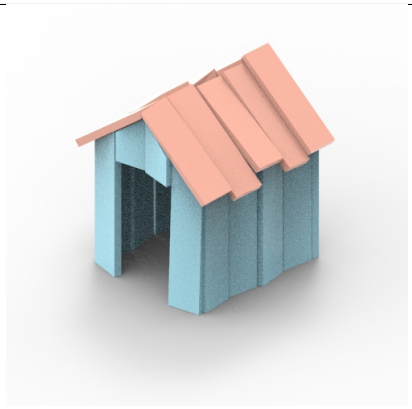
3. Game assets

This chapter features a list of the assets that were developed for this game.

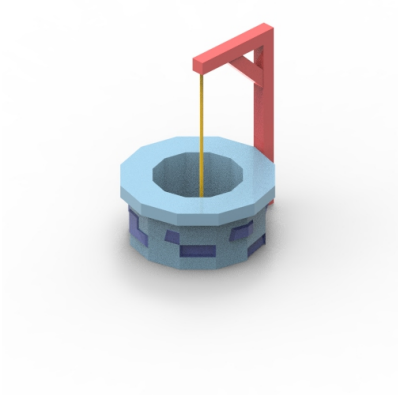
All the assets in the game were created by us using a trial version of ZBrush [4], academic license versions of Autodesk Maya 2017/2019 [5], and Substance Designer [6]. A complete list of assets can be found below. All assets (except two) were first sculpted in ZBrush after which their fbx files were exported. The fbx files were afterwards altered in Maya to create properly scaled UV-maps, apply materials, and to set the pivot of the model at the base of the mesh. Finally the models were scaled in Unity as their comparative scale was not considered during modelling. Their materials were handled by the Unity shader, which is explained in subsection 4.10

Object	Picture	Biome
Avatar		N/A
Palm tree		Desert

Tall cactus		Desert
Short cactus		Desert
Skull		Desert

Shed		Farm
Tractor		Farm
Windmill		Farm

Scarecrow		Farm
Lamppost		Village
Bench		Village

Well		Village
Trashcan		Village
Pine tree		Forest

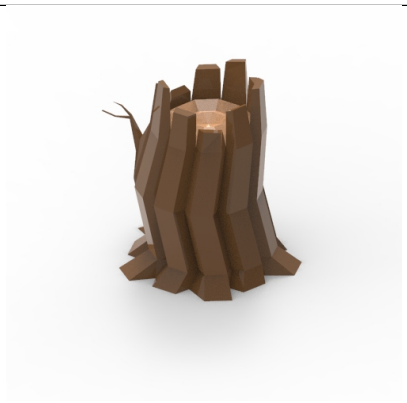
Tree		Forest
Tree 2		Forest
Tree stump		Forest

Table 3.1: Different assets used throughout the game.

4. Implementation

Since the game itself is procedurally generated, the main goal of the implementation for the game was to create an authoring system for us instead of manually placing and modifying the game. This consisted of several major parts. The terrain generation, the object generation, and a simple but powerful custom event system based on the Unite 2017 talk by Ryan Hipple [7].

4.1 Terrain generation

The system first creates two random offsets for the perlin noise map, then generates the vertex array for the mesh, applies the float values generated by the noise function to the y position of the vertices, seen in line 13 in Code 4.1, and then generates the mesh in the method shown on line 26. We added the option to scale the distance of the vertices and to scale the strength of the noise values applied. The final steps in the terrain generation were the addition of circular plateaus, where the terrain is smoothed out, and the addition of predetermined biomes, which can influence the initial object generation in the game. The initial biome map is saved in a scriptable object to decrease dependency between behaviour scripts.

Code 4.1: Terrain Generation

```
1  void CreateShape()
2  {
3      vertices = new Vector3[vertexAmount];
4      uv = new Vector2[vertexAmount];
5      oneTerrain.terrain = new TerrainInfo[(zSize + 1) * (xSize + 1)];
6      oneTerrain.verticesX = xSize + 1;
7      oneTerrain.verticesZ = zSize + 1;
8
9      for (int i = 0, z = 0; z <= zSize; z++)
10     {
11         for (int x = 0; x <= xSize; x++)
12         {
13             float y = Mathf.PerlinNoise(x * settings.perlinScale +
                   perlinXOffset,
14                             z * settings.perlinScale + perlinZOffset) *
15                     settings.perlinStrength;
16             oneTerrain.terrain[i] = new TerrainInfo(false, y,
                   BiomeType.Plains);
```

```
17            vertices[i] = new Vector3(settings.vertexDistance * x, y,
                  settings.vertexDistance * z);
18            uv[i] = new Vector2(x/(float)xSize,z/(float)zSize);
19            i++;
20        }
21    }
22
23    MakeTriangles();
24 }
25
26 void MakeTriangles()
27 {
28    triangles = new int[xSize * zSize * 6];
29
30    int vert = 0;
31    int tris = 0;
32    for (int z = 0; z < zSize; z++)
33    {
34        for (int x = 0; x < xSize; x++)
35        {
36            triangles[tris + 0] = vert + 0;
37            triangles[tris + 1] = vert + xSize + 1;
38            triangles[tris + 2] = vert + 1;
39            triangles[tris + 3] = vert + 1;
40            triangles[tris + 4] = vert + xSize + 1;
41            triangles[tris + 5] = vert + xSize + 2;
42
43            vert++;
44            tris += 6;
45        }
46
47        vert++;
48    }
49 }
```

4.2 Object Pipeline

To make the object pipeline smoother, we have used prefabs and prefab variants in the game. For each biome we had an empty prefab with the interactable object script on it, and for each variant we added the graphics component which stored the collider, as well as the GazeAware component from the Tobii API.

The first step of object pipeline was to create an authoring tool for the possible objects to spawn into the game. We made a simple database with a custom editor window, seen in Figure 4.1. We used three databases, one for the regular object

spawns, one for the dark- and finally one for the golden objects. The database allowed us to easily modify the biome, safe distance and biome influence of the objects without having to individually look through the prefabs.

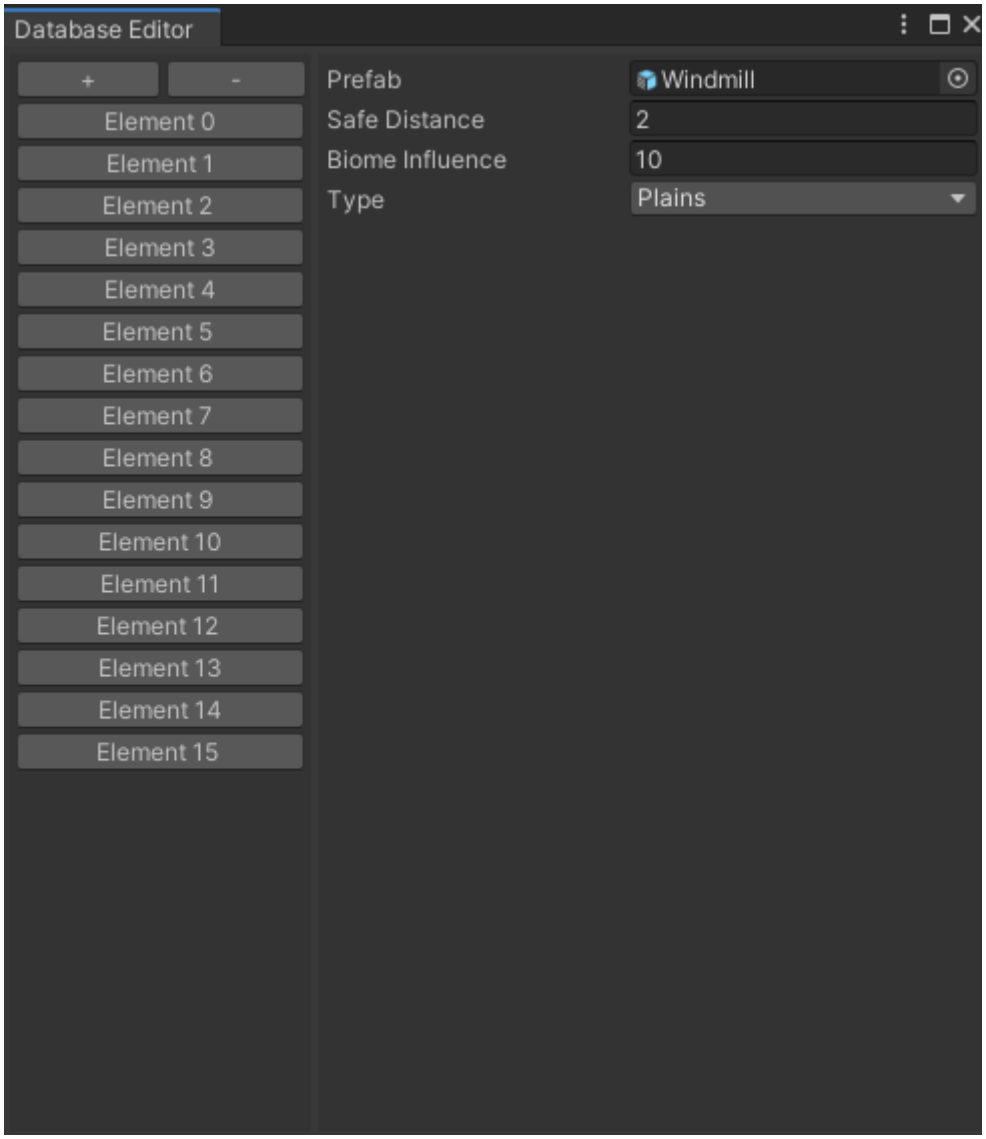


Figure 4.1: The database editor window

4.3 Object Spawning

Object spawning is done in several steps. First there is the initial object spawn, which can be completely random, or it can be influenced by the initial spawn

biomes. In the second case, line 13 in code 4.2, the system checks the initial biomes created by the terrain generator, and based on those uses a weighed random generator to spawn the objects. If there is a biome allocated to the coordinate, the object generator will have a higher chance of spawning an object of that biome.

Code 4.2: Initial object spawning

```
1  public void GenerateSafeEnvironmentWhile(int amount, EnvironmentDatabase
       database, bool ignoreBiomes = false)
2  {
3      int errorCount = 0;
4      while (objectsSpawned < amount && errorCount < maxErrors)
5      {
6          //spawn candidate
7          Vector3 spawnPointCandidate = new Vector3(Random.Range(0f, xSize),
               5f, Random.Range(0f, zSize));
8
9          //Biome check
10         int[] biomeRatios = new int[4];
11         if (!ignoreBiomes)
12         {
13             biomeRatios = CheckInitialBiomes(spawnPointCandidate.x,
                   spawnPointCandidate.z, 5);
14         }
15
16         biomeRatios = AddBaseRatios(biomeRatios, 15);
17         int objectTypeCandidate =
               ReturnRandomWeightedBiomeIndex(biomeRatios);
18
19         List<ObjectData> datas =
20             database.environmentObjects.FindAll(data => data.type ==
                   (BiomeType) objectTypeCandidate);
21         if (datas.Count == 0)
22         {
23             errorCount++;
24             continue;
25         }
26
27         ObjectData selectedObj = datas[Random.Range(0, datas.Count)];
28
29         spawnPointCandidate.y = 0f;
30         Vector3 clampedSpawnPointCandidate = new Vector3(
31             Mathf.Clamp(spawnPointCandidate.x, selectedObj.safeDistance,
                   xSize - selectedObj.safeDistance),
32             0f,
33             Mathf.Clamp(spawnPointCandidate.z, selectedObj.safeDistance,
                   zSize - selectedObj.safeDistance));
```

```
34
35          if (CheckOverlapsForObject(clampedSpawnPointCandidate,
                selectedObj.safeDistance))
36          {
37              InteractableObject o = Instantiate(selectedObj.prefab,
                    clampedSpawnPointCandidate, RandomRotation())
38                  .GetComponent<InteractableObject>();
39              o.type = selectedObj.type;
40              o.biomeInfluenceRadius = selectedObj.biomeInfluence;
41              objectsSpawned++;
42              errorCount = 0;
43          }
44          else
45          {
46              errorCount++;
47          }
48      }
49  }
```

The object spawning runs either until enough objects are created or, to avoid infinite loops, until an error threshold is reached. The error count goes up every time the algorithm cannot find a spawn object candidate or the position of the new object would overlap with an already spawned one. Selecting the object candidates is done on line 19, 20 and 27, where the algorithm creates a list of objects of the desired biome from the asset database, and chooses a random one. After this, the spawnpoint candidate is clamped down to be inside the game area, then an overlap check is performed to see if the spawnpoint candidate is valid. If there are no other objects in the way, the object is instantiated and its values are initialized.

The second type of object spawning happens during gameplay, after object activation. This is very similar to the first method, but it differs in two main things: Firstly the object's biome is predetermined, and based on the last activated object, and second: The algorithm has to make sure the object does not spawn where it is immediately visible. For this we use a spawn gizmo which is a set distance in front of the player's movement direction, and a method to project the screen's corner coordinates on the ground to see if the spawn gizmo is inside or not. In code 4.3 on line 4 we make a quaternion to rotate the rectangle created by the projected points, aligning them with the x and z axes to simplify the boundary check. If the spawn gizmo is inside, a corner coordinate based on the player position is returned, otherwise the normal spawn gizmo position is used.

Code 4.3: Bumping the spawnpoint out of view

```
1  Vector3 BumpOutOfRotatedCamera(Vector3 pos, int quadrant)
2  {
```

```
    Camera camera = Camera.main;
    Quaternion rot =
        Quaternion.AngleAxis(-camera.transform.rotation.eulerAngles.y,
        Vector3.up);
    Vector3 adjustedPos = new Vector3(pos.x, pos.y, pos.z);
    Vector3 bottomLeft = rot * CameraPointProjectedOnGround(camera, 0f, 0f);
    Vector3 topRight = rot * CameraPointProjectedOnGround(camera,
        camera.pixelWidth, camera.pixelHeight);
    Vector3 rotatedPos = rot * pos;
    if (rotatedPos.x > bottomLeft.x && rotatedPos.x < topRight.x &&
        rotatedPos.z > bottomLeft.z &&
        rotatedPos.z < topRight.z)
    {
        adjustedPos.x = (quadrant & 1) != 1 ? bottomLeft.x : topRight.x;
        adjustedPos.z = (quadrant & 2) != 2 ? bottomLeft.z : topRight.z;
        return Quaternion.Inverse(rot) * adjustedPos;
    }

    return adjustedPos;
}
```

4.4 Interactable Objects

The script *InteractableObject* is responsible for the interactivity in the game. It essentially houses methods that are called in various stages of interaction. The first three: OnLookingEnter, OnLookingStay and OnLookingExit get called when the player's Field of View hits an object, stays in view or leaves the object respectively. In a similar vein in code 4.4, OnFocusEnter, OnFocusStay and OnFocusExit are called when the player looks at an object or hovers over an object with their mouse. Each interactable object is part of a runtime set, which is used to create the biome maps during gameplay. The script is also the base script for the golden and dark objects, as the interaction with those is identical, and the only changes are in their behaviour when interacted with.

Code 4.4: Eye-interaction of objects

```
if (settings.eyeInput)
{
    if (_gazeAware.HasGazeFocus)
    {
        totalDwelltime += Time.deltaTime;
        if (!hadFocusLastFrame)
        {
            OnFocusEnter();
```

```
            hadFocusLastFrame = true;
        }

        if (!isActivated && pbt.activateReady)
        {
            activationTimer += Time.deltaTime;
            OnFocusStay();
        }
    }
    else
    {
        if (hadFocusLastFrame)
        {
            OnFocusExit();
            activationTimer = 0f;
            hadFocusLastFrame = false;
        }
    }
}
```

4.5 Golden and Dark objects

Interaction is the same across all objects, the only difference is that the dark and golden objects fire different events when activated, and have slightly altered behaviour. The dark objects are spawned and initialized before the game in the same manner as the regular environment objects, but the player cannot spawn more of them. The golden objects are the hidden objective of the game. The way they spawn is whenever a regular object spawns, it increases the chance to also spawn a golden object, shown in code 4.5. Golden object spawns are made so they spawn when their respective biomes are active, in a position where the biome different from their own, and away from the player, to increase the motivation to explore.

Code 4.5: Golden Object manager

```
public void TrySpawnGoldenObject(SpawnInfo info)
{
    int typeAsIndex = (int) info.type;
    spawnPercentages[typeAsIndex] += spawnPercentIncrease / 100f;

    if (hasSpawned[typeAsIndex]) return;

    if (!(Random.value < spawnPercentages[typeAsIndex])) return;
    hasSpawned[typeAsIndex] = true;
    GoldenSpawnEvent.Invoke(info.type);
```

```
11 }
```

4.6 Eye tracking

Eye tracking is done through the Tobii API. Before starting, the user can select the eye-tracking interaction option, and if an eye tracker is connected and the Tobii software is running, it will be enabled, otherwise the game will default to mouse interaction. On the interactable object prefabs we use a GazeAware component, which tells the *InteractableObject* script whether the given object is being looked at or not.

4.7 Event system

We implemented a custom event system for two main reasons: To make it more modular and to allow for easier development. We made three different event types, one that takes no arguments, one that takes a *BiomeType* enum as an argument and finally one that takes a *SpawnInfo* class containing the Biome Type, spawn location and influence radius of an object.

The event system is made out of three parts: The event definition, the event listener and a custom *UnityEvent* to store the event response. Code 4.6 shows the structure of the event system for the spawn type events. The event itself, on line 2 is a scriptable object with a list of listeners registered to it, and a public method to raise the event. Line 29 is the event listener, which can store a reference to the custom event and has a UnityEvent of type "SpawnInfo" for the response (Line 51). When the custom event gets raised, the listener invokes the event response.

Code 4.6: Spawn Type event structure

```
1  [CreateAssetMenu(fileName = "NewSpawnEvent", menuName = "Spawn Game Event",
       order = 8)]
2  public class SpawnTypeEvent : ScriptableObject
3  {
4      private List<SpawnTypeListener> eventListeners = new
           List<SpawnTypeListener>();
5
6      public void Raise(SpawnInfo info)
7      {
8          for (int i = eventListeners.Count - 1; i >= 0; i--)
9          {
10             eventListeners[i].OnEventRaised(info);
11         }
12     }
```

```
13
14      public void RegisterListener(SpawnTypeListener listener)
15      {
16          if (!eventListeners.Contains(listener))
17          {
18              eventListeners.Add(listener);
19          }
20      }
21
22      public void UnregisterListener(SpawnTypeListener listener)
23      {
24          if (eventListeners.Contains(listener))
25              eventListeners.Remove(listener);
26      }
27  }
28
29  public class SpawnTypeListener : MonoBehaviour
30  {
31      public SpawnTypeEvent Event;
32      public SpawnTypeUnityEvent Response;
33
34      private void OnEnable()
35      {
36          Event.RegisterListener(this);
37      }
38
39      private void OnDisable()
40      {
41          Event.UnregisterListener(this);
42      }
43
44      public void OnEventRaised(SpawnInfo info)
45      {
46          Response.Invoke(info);
47      }
48  }
49
50  [System.Serializable]
51  public class SpawnTypeUnityEvent : UnityEvent<SpawnInfo>
52  {
53
54  }
```

Using the event system is simple. We can create an event through the context menu in the project folder, then define a UnityEvent in the class where we want to call an event from. In that UnityEvent we can drag in the previously created

custom event, and select its Raise() method for invokation. On the object where we want to listen to the event, we can add an event listener, drag in the event to listen to, and for the response we can drag in public methods from scripts on the object that either match the signature of the event call or have no arguments. A use of the spawn type events is shown on figure 4.2.

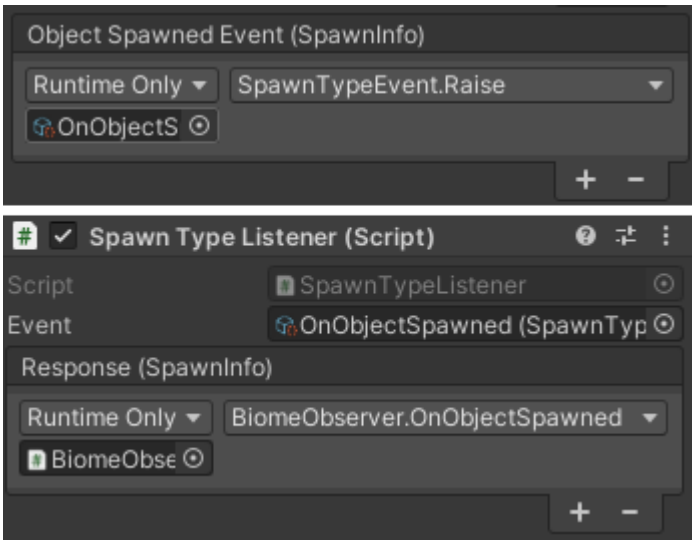


Figure 4.2: Event call on the object manager (top) and response on the biome observer (bottom).

4.8 Game Manager

The Game Manager is a singleton responsible for storing the game's current state, which can be MainMenu, GamePlay, Pause, and GameEnd. The script also handles transitions between the game states, done in two stages: The script has methods when exiting a state and when entering a state, making the transitions easier to maintain. Finally the game manager is responsible for setting up the level and initial object generation, as well as handling the UI for the game.

4.9 Substance graphs

The different textures utilised in the prototype were all created using Substance Designer [6]. The program features a powerful node-graph system which allows different texture maps to be generated and output in high quality. Though it is possible to have substance graph textures generated at runtime in Unity, the cost in terms of processing power was too high. For this reason the only textures generated at runtime were biomemaps which could be done at a low resolution, leaving little impact on the performance.

A texture was created for the ground as well as for each of the four biomes: forest, farm, village, and desert. Another benefit with Substance was that by generating procedurally generated textures, the program also ensured seamless tiling of the textures. This meant that the textures could be scaled and tiled without fearing seams to appear, no matter how small/big of an area any biome would occupy.

4.9.1 Ground texture

The ground texture which exists outside any of the biomes is the simplest of the five. The main part of the texture generation can be seen in Figure 4.3. The irregular shapes are created by running an edge detection on a "Cell" node and applying a flood fill to create the bevelled effect in random directions. As Unity's Universal Render Pipeline does not support height maps, a coloured ambient occlusion map was used to create the sense of depth in the material.

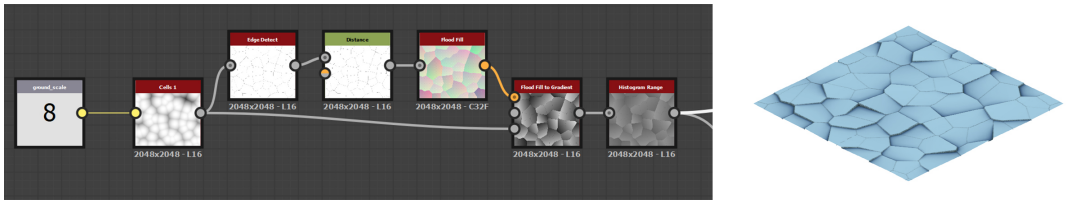


Figure 4.3: Substance graph cutout of the ground texture map generation as well as outputted texture.

4.9.2 Forest texture

The biome textures consisted of a few more nodes than the ground texture, shown in Figure 4.4. To create the leaves a few shape and transform nodes were utilised to create the overall silhouette of the leaf. The same process was used to create the "groove", marking the skeleton of the leaf, in the middle. Once the silhouette and skeleton is combined, by passing it through a "Tile Sampler" node the surface is populated with 900 leaves. The values of the tile sampler are tweaked to add randomness to the direction and size of the leaves and by changing their brightness we create depth. The levels of the height map are changed and used for the roughness and metallic map, leaving the top leaves shinier than the bottom ones, creating some depth through those maps instead. With the randomised brightness of the leaves, a gradient map containing colours from yellow to red was used to give the leaves randomised colours as well.

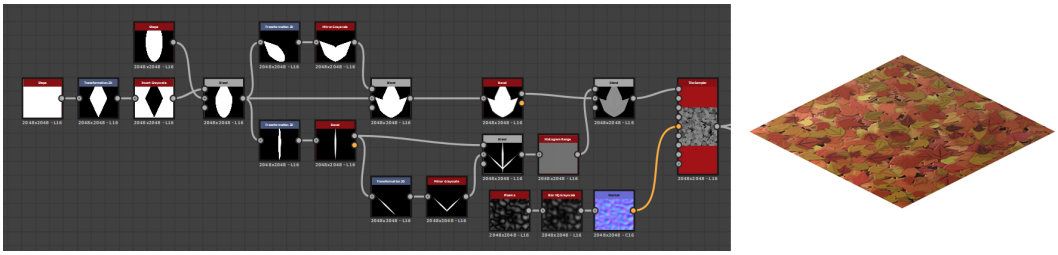


Figure 4.4: Substance graph cutout of the forest texture map generation as well as outputted texture.

4.9.3 Farm texture

The farm texture is generated in a similar way to the forest. The top half of the graph, seen in Figure 4.6 uses shape, gradient, and warp nodes to create two different grass shapes. These are set as inputs in the tile sampler which populates the texture with 96x96 blades of grass. Though there is some randomisation to the size and direction, the "DisplacementMap", "RotationMap", and "MaskMap" inputs are used to control the randomness and creating the general "flow" through the grass. The masks are generated, shown on the bottom half of the graph, with noise maps and a blurred "Cell" node and then the values are adjusted for the desired outcome. Like with the forest, the metallic and roughness maps are used to create depth. The colour of the texture is made with a gradient map, making the base of the blades darker and the tips lighter in colour. A bit of noise is added to act as ground underneath the grass, but cannot be seen at the angle and distance it is viewed at in the final implementation.

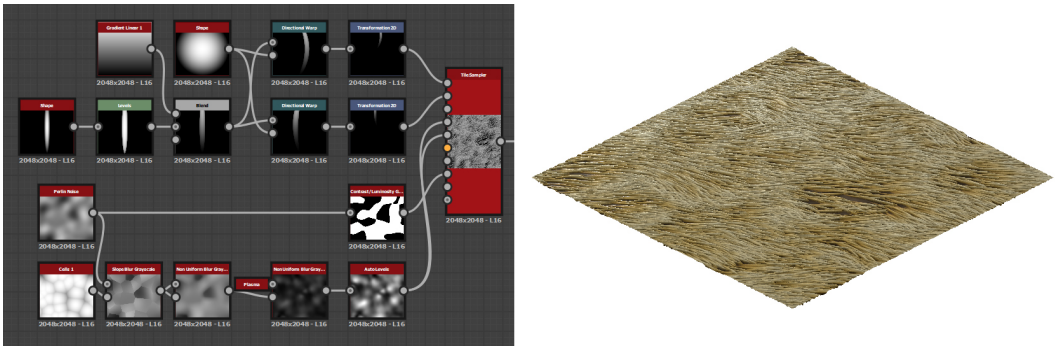


Figure 4.5: Substance graph cutout of the farm texture map generation as well as outputted texture.

4.9.4 Village texture

The village cobblestone texture was created by filling the map with a 12x12 grid of squares, offset by 0.5 for every other row and given a random brightness. The greyscale value is to colour the stones at random. By applying adding different

types of blurring and bevelling to the edges, chips and cracks were added. Two randomised flood fills were used: one to vary the surface angles of the stones, and one to create deeper indents/erosion/missing parts of the stones. Lastly some noise was added via a cloud and fractal sum node to add roughness to the surface of the stones. The same nodes were also used to create dirt between the rocks, mainly shown where the stones are very chipped/cracked/eroded.

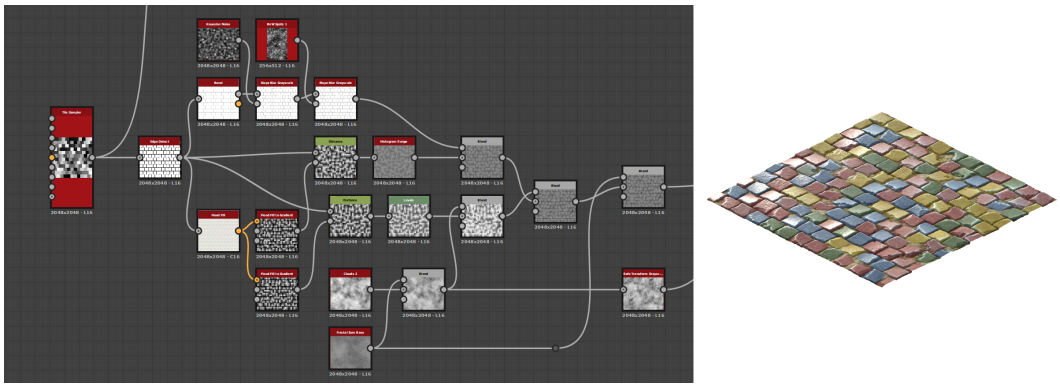


Figure 4.6: Substance graph cutout of the village texture map generation as well as outputted texture.

4.9.5 Desert texture

The desert texture had two generated components: the rocks and the sand. For the rocks, four different rocks were created as seen in Figure 4.7. The rocks were based on paraboloids which have a cell node subtracked for a more rocklike shape. By blending with some flood fills planes were created on the surface, and a bevel tapered the shape to prevent the edge from going straight down into the ground. The tile sampler placed the rocks around in a range of randomised size and rotations.

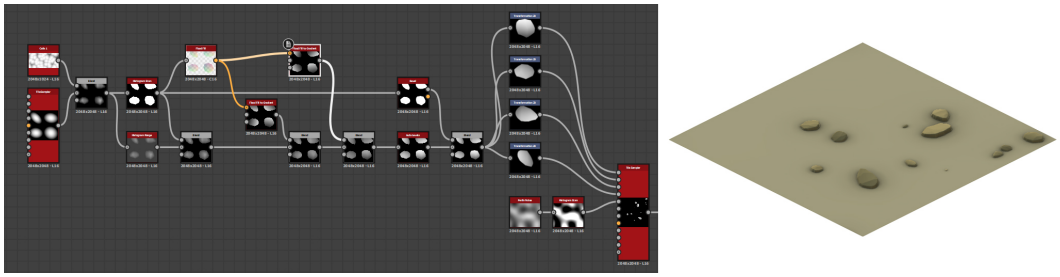


Figure 4.7: Substance graph cutout of the rock generation used for the desert biome as well as output of the rocks without the sand.

The sand in which the rocks were placed in was a bit simpler to make, as shown in Figure 4.8. The tile sampler placed the different dunes on the texture, and a directional warp, with a plasma node as the intensity input, added waves to

them. The intensity/height of the dunes is tapered by the histogram range node and the rocks and sand is added together for the final texture.

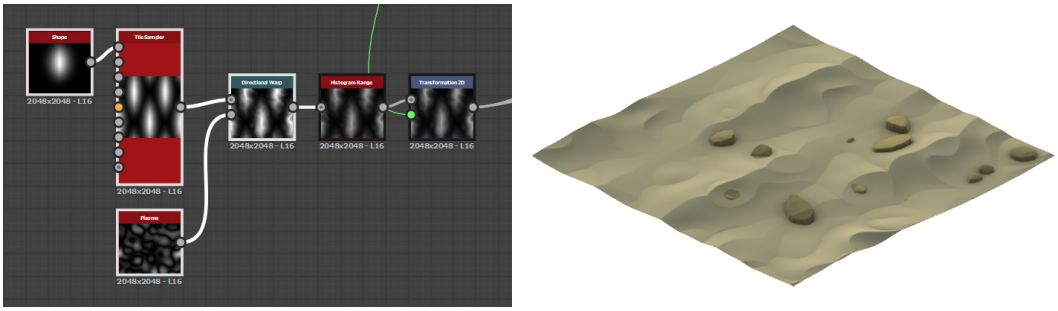


Figure 4.8: Substance graph cutout of the desert texture map generation as well as outputted texture.

4.10 Unity shaders

Two different shaders were used in Unity, one for the objects and one for the ground. The object shader was used to assign the colour of the object, as well as generate the saturation and activation effect.

4.10.1 Object shader

The first part of the object shader was to assign the general colour of the material. To enable some diversity to the colour we made it possible for the colour to be picked at random (from a selected colour palette) based on the location on the map. Figure 4.9 shows how this was accomplished. As different materials on the same object could occupy the same x/z-coordiate (e.g. the roof of the shed) a seed was added to allow different colours on the same location. A boolean was also added to disable/enable the random colours as some materials, such as the treestump, were less suited for random colours. It should, however, be noted that the leaves of the trees had a different colour palette added to allow random colours for these.

The graph for the first effect, the transition from desaturated to saturated (henceforth referred to as state 0 and 1 respectively), can be seen in Figure 4.10. First a noisemap is used to allow some randomness in the transition between the states. The upper "Step" node applies a threshold to the noise map based on the *DeteriorationAmount* variable while the lower does the same but with the *EdgeWidth* variable added to the threshold amount. This creates to maps with a slight difference in size of the masks. With these two maps substracted from each other an edge is created between state 0 and 1. By multiplying the edge with a colour we can make sure it is easily differentiated from the base colour of the material. Lastly

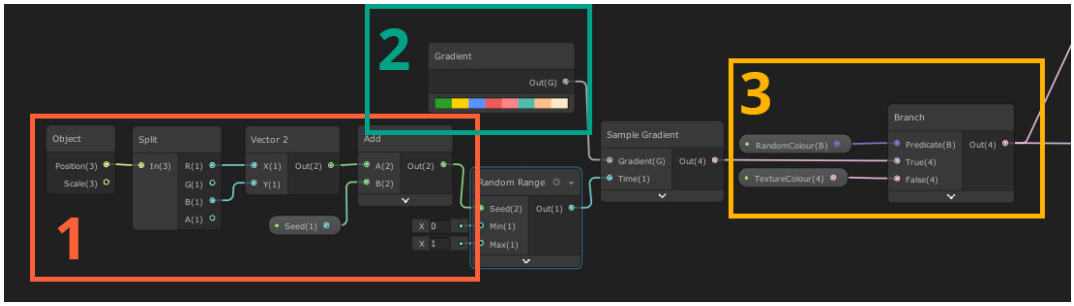


Figure 4.9: 1) generates a seed for the random selection of colour by creating a Vector2 based on the object's X- and Z-coordinate. 2) has the predefined colour palette. 3) applies the random colour if the boolean is set to true.

an inverse of the lower "Step" node is created as a mask for state 0. This inverted mask did not end being used, but was implemented with the intention of controlling the saturation amount and/or contrast in case we ended up using a texture, rather than a uniform colour.

The second effect each material has is the activation effect, which can be seen on Figure 4.11. This appears when the object is directly looked at. Similarly to the edge effect mentioned above, the main part of the effect is achieved by applying a "Step" node to a noise map, which is then multiplied by a colour. Whether the effect is visible or not depends on the *Activated* boolean variable. When set to *true* the activation effect is multiplied by zero, effectivly disabling it, and the saturation for the base colour is set to 3.

With all the effects generated the last part is to combine everything into the final output, shown on Figure 4.12. First the colour is multiplied with the mask for the saturated part. The saturation set by the *Activated* boolean is added and finally the desaturated parts are added for a final colourmap which is output in the *Albedo channel*. *2)* in the figure shows the unutilised nodes which could be used to fade details if a texture was used instead of a uniform colour. The edge- and activation effect are added together and output in *Emission channel*. As we added "golden" objects variables, the *Mettalic* and *Smoothness* variables were also created to allow for changes on these maps for individual objects.

4.10.2 Ground shader

As Unity's "Universal Render Pipeline", which was used for our project, does not have any way of applying blur via its shadergraph, a substance graph made in Substance Designer [6] was used. The graph simply split the incoming image into four channels (R,G,B,A) applied the blur and combined the channels again. The main purpose of the ground shader was to apply the biome mask generated with the Substance graph to the different biome maps. A simplified version of the

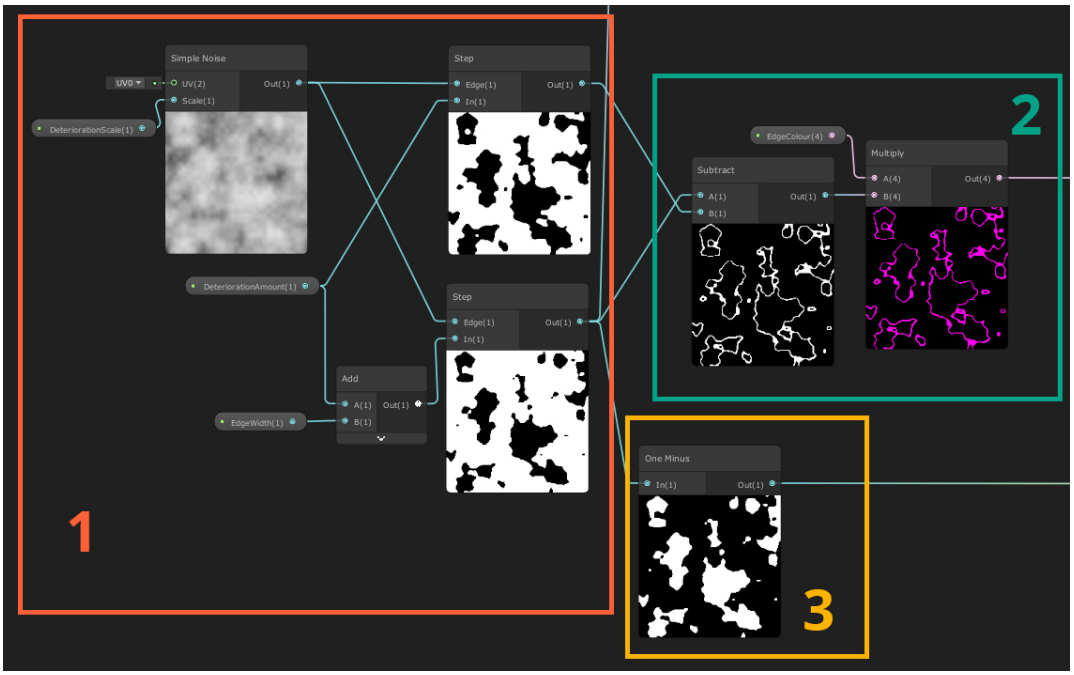


Figure 4.10: 1) creates two different maps based on how far into the saturation transition (in percentage) the material is. 2) subtracts the two maps creating the edge effect, which is multiplied by a colour. 3) inverts the smaller of the maps to create a mask for the desaturated part of the material

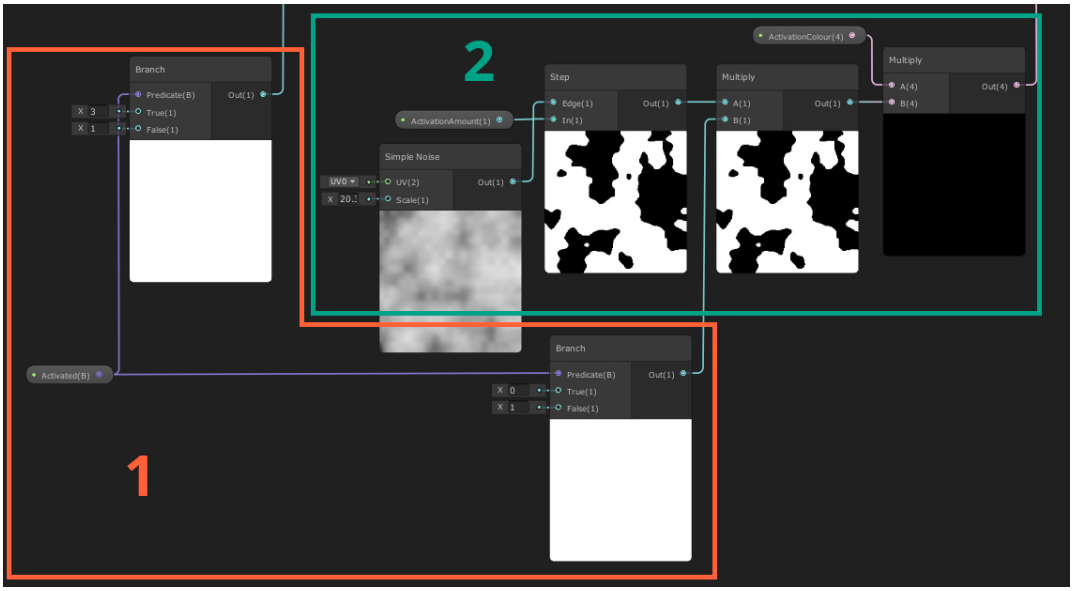


Figure 4.11: 1) has a boolean controlling whether the activation effect should appear and sets the saturation depending on whether the object is activated or not. 2) creates the activation effect that covers the material based on how far into the activation the material is

Figure 4.12: 1) combines the different colours and adds the proper saturation before outputting in the *Albedo* channel. 2) sets the saturation, contrast, and brightness of state 0. 3) is the final output, with custom values for the *Albedo*, *Emission*, *Metallic*, and *Smoothness* channels.

unity shader graph can be seen in Figure 4.13. The biome map has four channels in it: Red, Green, Blue, and Alpha. Each of these contain masking for each of the four biomes – Forest, Farm, Village, and Desert, respectively. The final map is generated by added each masked biome on top of the ground, leaving every area not containing a biome, textured with the default ground texture. Finally the saturation of the map is changed depending on whether it is within (saturated) or without (desaturated) the cone of vision.

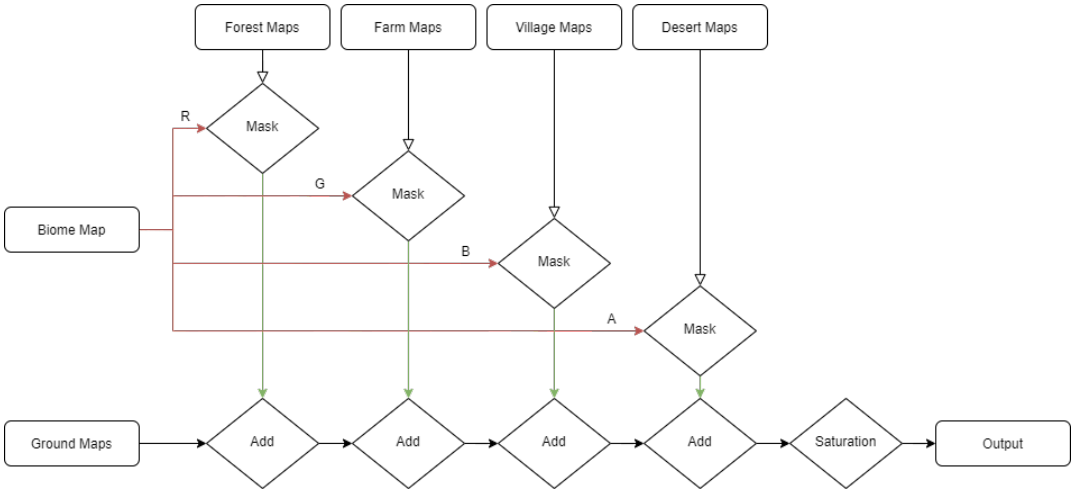


Figure 4.13: A simplified chart of the biome masking process.

5. Evaluation

In this section, we include the questionnaires used in this study.

5.1 Participant demographics

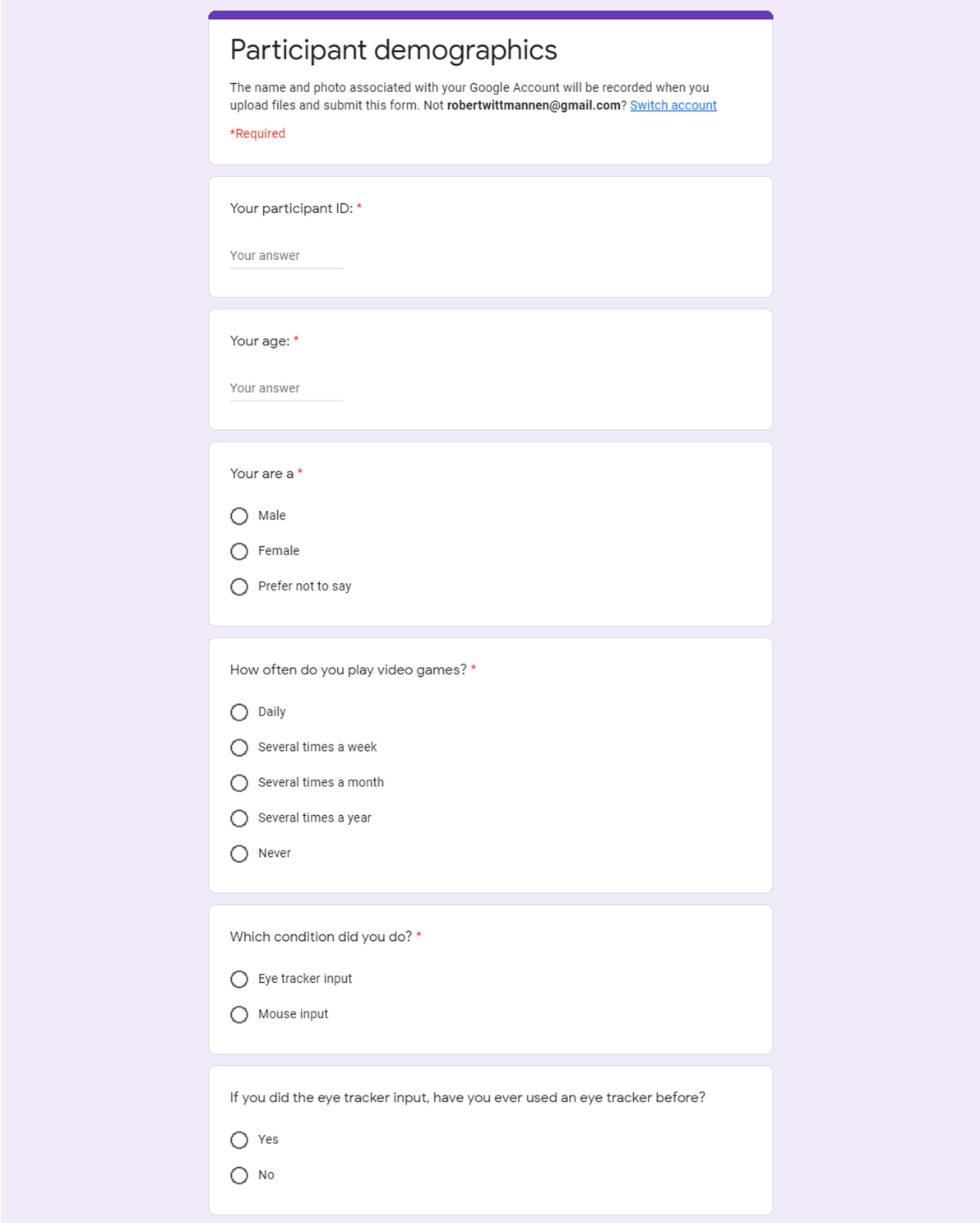
We asked the participants a few demographic questions such as age, gender, eye tracker experience. Figure 5.1 shows the questions.

5.2 Immersion Questionnaire

This questionnaire was developed by Jennet et al [3]. There were two versions developed, in this study we used the version with simpler wording. Six questions (6, 8-10, 18, 20) were negatively worded, meaning their score was inversed. Refer to Figure 5.2 for the questions.

5.3 User Engagement Scale Short Form

This questionnaire was developed by O'Brien et al [8]. There are 12 questions in total, 3 of which are marked inversely (5, 6 and 7). The questions can be seen on figure 5.3



Participant demographics

The name and photo associated with your Google Account will be recorded when you upload files and submit this form. Not **robertwittmannen@gmail.com**? Switch account

*Required

Your participant ID: *

Your answer

Your age: *

Your answer

Your are a *

- Male
- Female
- Prefer not to say

How often do you play video games? *

- Daily
- Several times a week
- Several times a month
- Several times a year
- Never

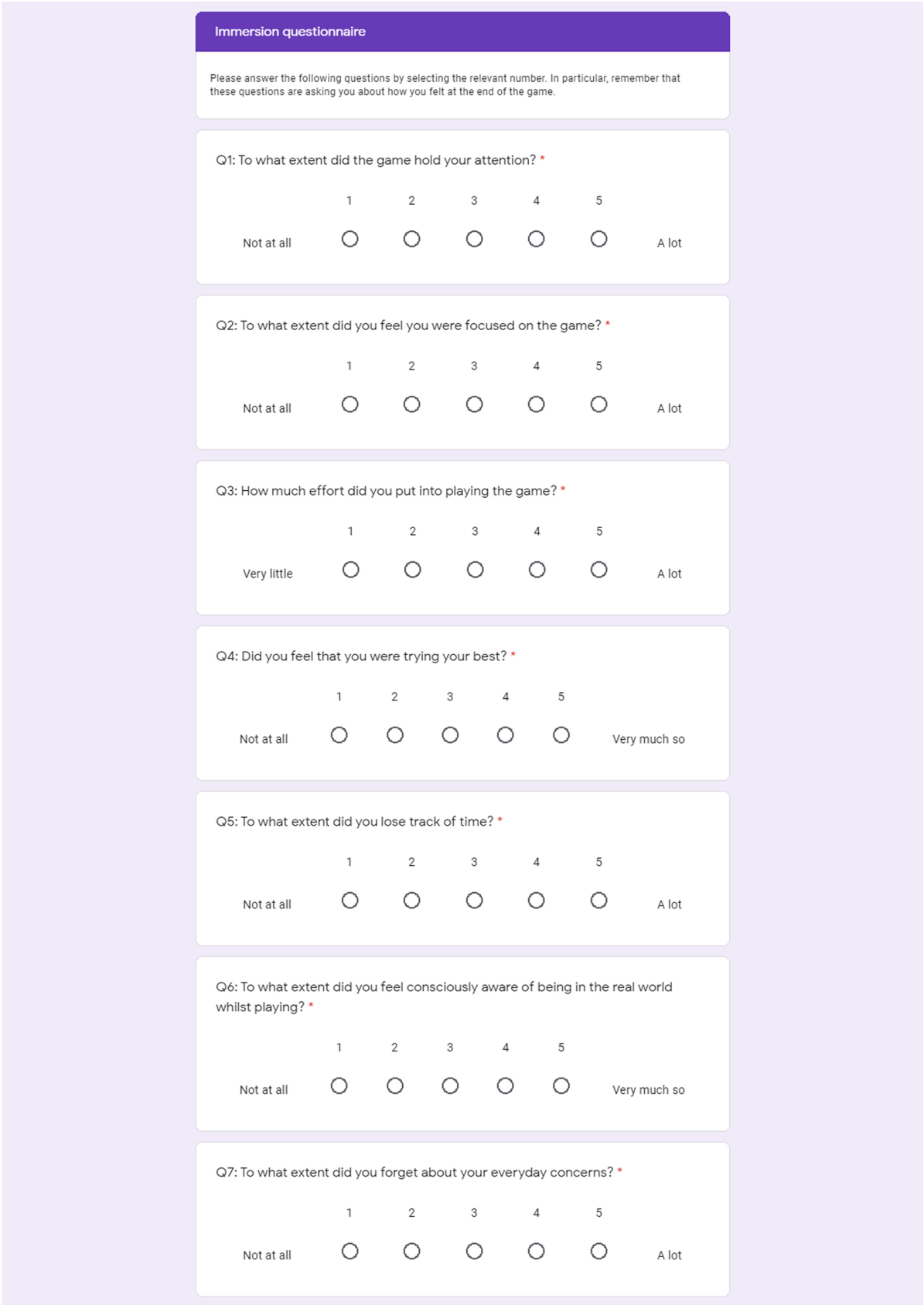
Which condition did you do? *

- Eye tracker input
- Mouse input

If you did the eye tracker input, have you ever used an eye tracker before?

- Yes
- No

Figure 5.1: Participant Demographics Questions



Immersion questionnaire

Please answer the following questions by selecting the relevant number. In particular, remember that these questions are asking you about how you felt at the end of the game.

Q1: To what extent did the game hold your attention? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	A lot

Q2: To what extent did you feel you were focused on the game? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	A lot

Q3: How much effort did you put into playing the game? *

	1	2	3	4	5	
Very little	○	○	○	○	○	A lot

Q4: Did you feel that you were trying your best? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Q5: To what extent did you lose track of time? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	A lot

Q6: To what extent did you feel consciously aware of being in the real world whilst playing? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Q7: To what extent did you forget about your everyday concerns? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	A lot

Q8: To what extent were you aware of yourself in your surroundings? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very aware

Q9: To what extent did you notice events taking place around you? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	A lot

Q10: Did you feel the urge at any point to stop playing and see what was happening around you? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Q11: To what extent did you feel that you were interacting with the game environment? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Q12: To what extent did you feel as though you were separated from your real-world environment? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Q13: To what extent did you feel that the game was something you were experiencing, rather than something you were just doing? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Q14: To what extent was your sense of being in the game environment stronger than your sense of being in the real world? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Q15: At any point did you find yourself become so involved that you were unaware you were even using controls? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Q16: To what extent did you feel as though you were moving through the game according to your own will? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Q17: To what extent did you find the game challenging? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very difficult

Q18: Were there any times during the game in which you just wanted to give up? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	A lot

Q19: To what extent did you feel motivated while playing? *

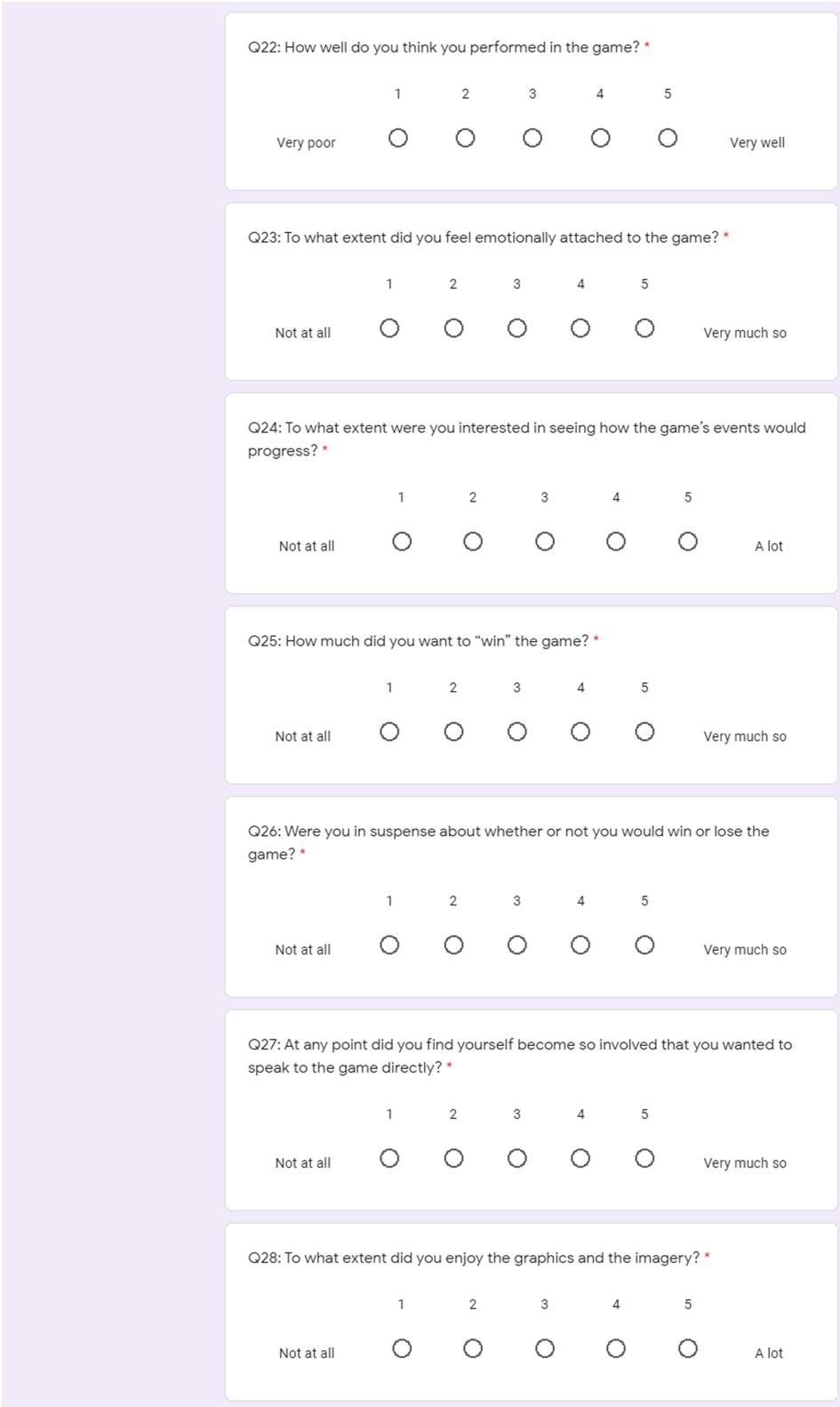
	1	2	3	4	5	
Not at all	○	○	○	○	○	A lot

Q20: To what extent did you find the game easy? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	Very much so

Q21: To what extent did you feel like you were making progress towards the end of the game? *

	1	2	3	4	5	
Not at all	○	○	○	○	○	A lot


Q22: How well do you think you performed in the game? *
1 2 3 4 5
Very poor
Very well
Q23: To what extent did you feel emotionally attached to the game? *
1 2 3 4 5
Not at all
Very much so
Q24: To what extent were you interested in seeing how the game's events would progress? *
1 2 3 4 5
Not at all
A lot
Q25: How much did you want to "win" the game? *
1 2 3 4 5
Not at all
Very much so
Q26: Were you in suspense about whether or not you would win or lose the game? *
1 2 3 4 5
Not at all
Very much so
Q27: At any point did you find yourself become so involved that you wanted to speak to the game directly? *
1 2 3 4 5
Not at all
Very much so
Q28: To what extent did you enjoy the graphics and the imagery? *
1 2 3 4 5
Not at all
A lot

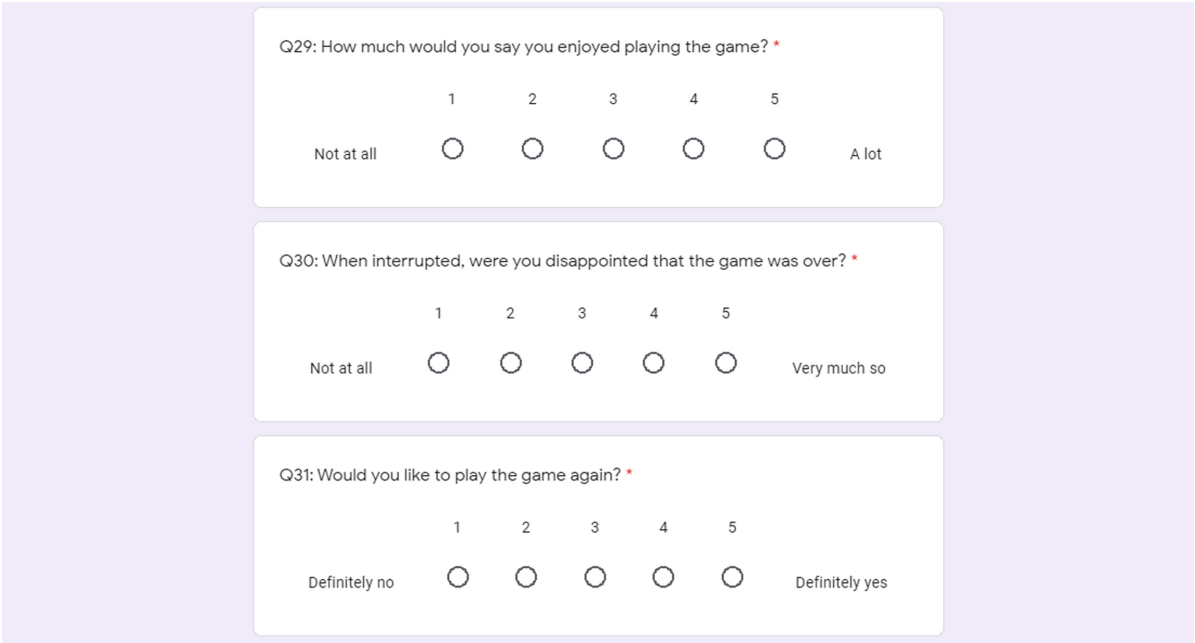
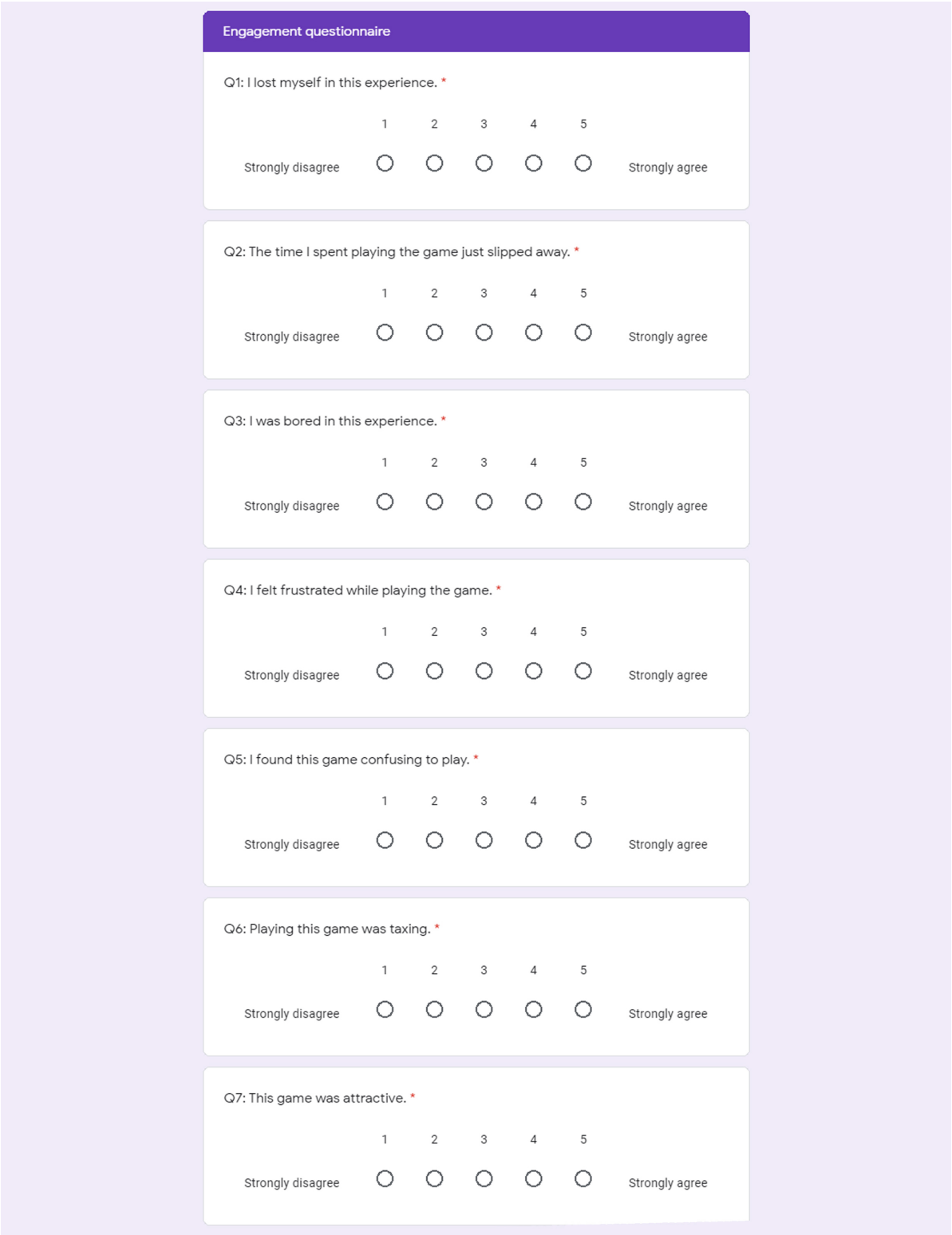


Figure 5.2: The Immersion Questionnaire



Engagement questionnaire

Q1: I lost myself in this experience. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Q2: The time I spent playing the game just slipped away. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Q3: I was bored in this experience. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Q4: I felt frustrated while playing the game. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Q5: I found this game confusing to play. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Q6: Playing this game was taxing. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Q7: This game was attractive. *

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

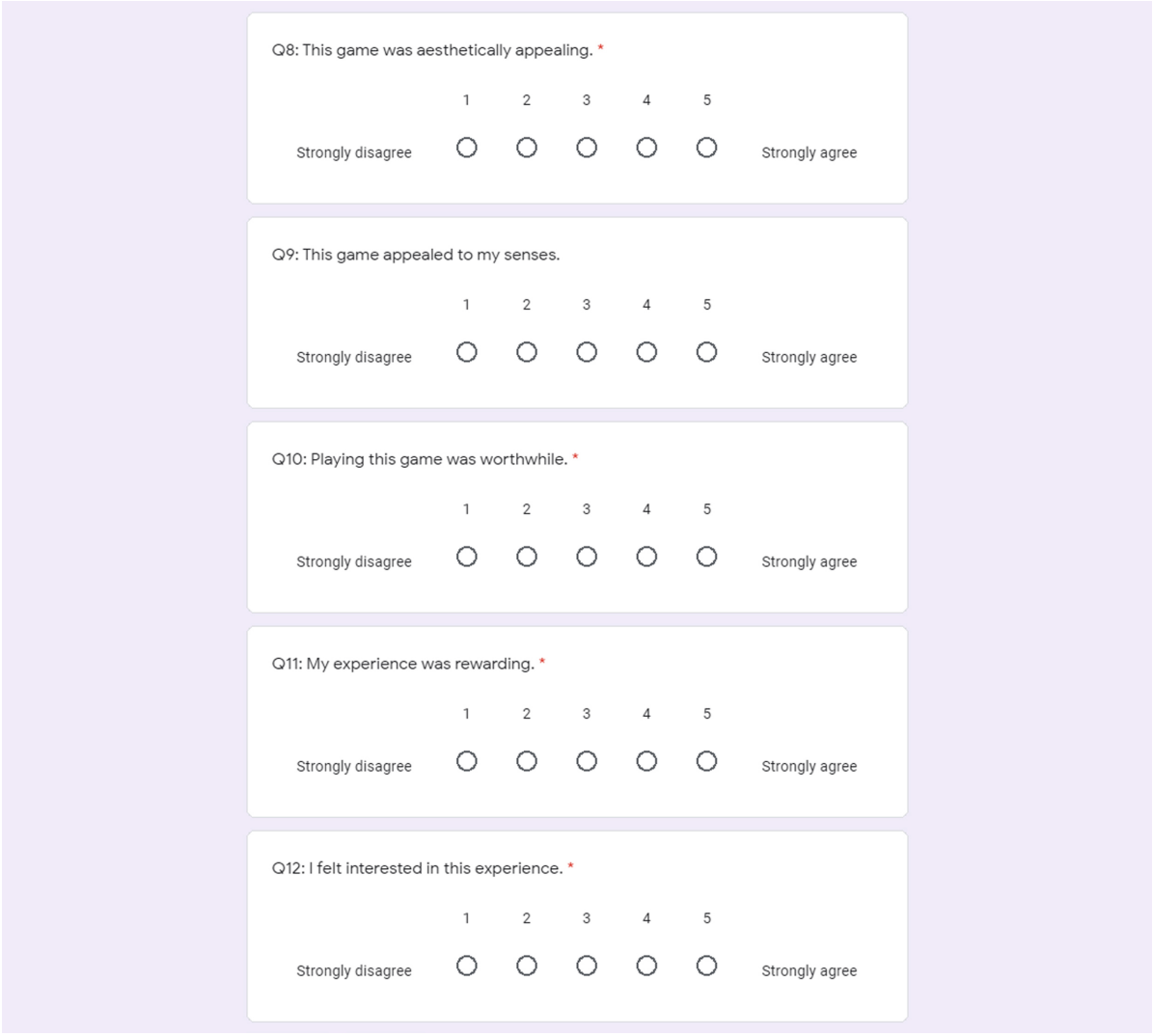


Figure 5.3: The User Engagement Scale Short Form

6. Statistics

This chapter includes the data captured by our forms.

6.1 Complete data

Table 6.1 shows the complete data from all test participants. "Distance" shows the distance traveled in game in meters. "TimeSecs" and "TimeMins" stand for Time in Seconds and Time in Minutes and represent the time participants spent playing the game. This time does not include eye tracker setup or filling out the questionnaires. "DarkDwell" shows the average time spent on looking directly at the dark objects. "Immersion" and "Engagement" columns show the calculated scores for those two measurements.

ID	Age	Gender	Input	Distance	TimeSecs	TimeMins	DarkDwell	Immersion	Engagement
1	25	M	ETI	4365	1711	29	15,99	95	3,5
2	23	M	ETI	1757	553	9	8,75	83	3,1
3	25	F	ETI	320	312	5	4,40	55	2,0
4	33	M	ETI	1151	365	6	7,75	70	2,3
5	26	F	ETI	3771	1380	23	13,17	71	2,4
6	23	F	ETI	863	301	5	2,62	85	2,5
7	24	F	ETI	7120	2075	35	14,27	107	3,6
8	28	M	ETI	6334	2051	34	20,79	88	2,6
9	24	M	ETI	631	241	4	2,14	92	2,9
10	28	F	ETI	2019	603	10	8,21	107	2,7
11	25	M	TI	15546	3785	63	21,15	77	3,0
12	30	M	TI	4101	897	15	3,75	75	2,4
13	25	M	TI	4525	1046	17	8,55	62	2,4
14	27	M	TI	11874	2706	45	21,33	62	2,3
15	23	M	TI	2114	501	8	6,33	72	2,8
16	31	M	TI	21080	5353	89	21,00	104	3,0
17	25	M	TI	3377	893	15	5,97	82	2,9
18	24	M	TI	6409	1409	23	9,56	127	3,7
19	25	M	TI	4947	1129	19	11,34	95	3,7
20	23	M	TI	2615	718	12	7,87	78	2,7

Table 6.1: Table showing the all the values for the measured variables

6.2 Immersion Questionnaire results

These are all the responses given by participants in the Immersion Questionnaire. The grey rows where the question number is followed by an asterisk in paren-

thesses stand for the questions where scores were marked inversely. The final two rows mark the calculated score and the averages of the scores. Note that the averages were not used in any way in the study.

Participant ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Q1	3	2	2	2	3	3	4	4	3	4	4	3	1	3	1	5	4	4	4	3
Q2	3	4	2	5	3	4	5	4	4	5	4	2	1	2	1	5	4	5	4	5
Q3	4	4	1	4	4	3	5	2	4	5	4	4	5	2	2	5	3	5	5	2
Q4	5	5	5	4	3	4	5	3	4	5	5	4	5	4	4	5	3	5	5	4
Q5	2	3	5	1	2	2	4	1	2	1	2	3	1	1	1	1	2	4	2	2
Q6 (*)	2	3	1	1	2	4	3	1	3	2	2	4	1	1	1	1	2	4	3	1
Q7	1	1	1	1	1	3	4	2	2	4	2	3	1	1	3	3	1	4	2	4
Q8 (*)	4	2	1	2	1	4	3	2	3	3	2	2	1	1	4	1	2	1	4	2
Q9 (*)	4	2	1	2	1	5	2	3	4	3	2	4	1	1	3	3	2	1	4	2
Q10 (*)	2	3	5	5	2	4	4	5	5	5	2	2	3	3	3	5	5	2	2	5
Q11	4	3	1	3	2	3	4	4	2	4	3	2	4	2	4	5	3	4	3	4
Q12	3	2	1	1	1	4	2	2	2	2	2	2	1	1	1	3	2	4	1	2
Q13	3	4	1	2	2	3	4	4	2	2	1	1	1	1	4	2	2	5	4	2
Q14	2	3	1	1	2	3	1	1	3	2	2	1	1	1	1	2	1	4	3	2
Q15	1	1	1	1	1	1	2	1	3	5	4	1	1	1	1	5	2	5	3	1
Q16	3	4	5	5	3	3	4	5	3	5	5	2	4	5	5	5	3	5	2	5
Q17	4	2	1	1	3	5	2	2	4	4	2	1	5	2	1	5	1	4	2	1
Q18 (*)	2	2	1	1	3	1	2	5	2	1	1	1	1	2	2	1	4	2	3	2
Q19	4	2	1	3	3	2	3	3	3	3	1	3	1	3	2	5	3	4	4	2
Q20 (*)	3	2	1	1	4	5	2	3	3	5	2	1	4	3	3	5	2	1	2	1
Q21	2	2	1	1	1	1	4	1	3	3	1	1	1	2	3	1	4	5	2	3
Q22	2	3	1	5	2	1	3	1	4	2	3	4	3	2	2	3	4	5	5	3
Q23	3	3	1	2	2	1	3	4	2	5	2	1	1	1	1	1	1	4	2	1
Q24	4	5	2	1	3	2	5	5	4	5	3	5	1	3	4	5	4	5	2	2
Q25	4	3	2	5	3	3	5	1	2	3	5	5	3	4	4	5	4	5	5	2
Q26	4	1	1	5	2	1	4	1	3	2	2	2	2	2	1	5	4	5	2	1
Q27	2	1	5	1	2	1	4	5	2	5	1	4	1	1	1	1	1	5	1	1
Q28	5	3	1	3	4	4	5	4	3	4	4	3	4	2	4	5	3	5	5	3
Q29	3	3	1	1	2	2	4	3	3	3	2	2	1	1	1	4	3	5	3	2
Q30	3	2	1	1	2	1	2	1	2	1	1	1	1	1	3	1	1	5	3	1
Q31	4	3	1	3	2	2	3	5	3	4	1	1	1	2	1	1	2	5	3	1
Score	95	83	55	74	71	85	107	88	92	107	77	75	62	61	72	104	82	127	95	72
Average	3,06	2,68	1,77	2,39	2,29	2,74	3,45	2,84	2,97	3,45	2,48	2,42	2,00	1,97	2,32	3,35	2,65	4,10	3,06	2,32

Figure 6.1: Participant responses on the Immersion Questionnaire

6.3 User Engagement Scale Short Form results

These are all the responses given by participants in the User Engagement Scale Short Form. The grey rows where the question number is followed by an asterisk in parentheses stand for the questions where scores were marked inversely.

Participant ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Q1	3	1	5	1	2	3	4	1	2	2	2	2	1	1	1	4	2	5	3	1
Q2	4	2	1	2	2	2	4	3	3	2	2	3	2	1	1	1	2	3	3	2
Q3	2	3	5	4	4	3	1	1	4	3	5	4	5	4	5	2	3	4	2	4
Q4 (*)	2	2	5	2	1	1	2	3	2	1	1	2	1	1	3	1	4	1	3	5
Q5 (*)	4	4	1	1	1	1	3	1	4	1	4	3	1	3	2	1	3	2	5	2
Q6 (*)	4	3	1	5	3	4	4	3	3	2	5	5	3	4	5	3	5	1	4	5
Q7	4	4	1	1	3	3	5	3	3	3	3	2	4	2	4	5	2	5	5	3
Q8	4	4	1	5	4	4	5	4	4	4	4	2	4	3	4	5	3	5	5	3
Q9	4	4	1	2	2	3	4	2	2	4	4	1	4	2	4	3	1	5	4	2
Q10	3	4	1	2	2	2	3	3	3	4	1	2	1	3	1	4	3	5	3	2
Q11	4	2	1	1	2	2	4	2	2	2	3	1	1	2	1	2	4	3	3	1
Q12	4	4	1	1	3	2	4	5	3	4	2	2	2	2	2	5	3	5	4	2
Score	3,5	3,1	2,0	2,3	2,4	2,5	3,6	2,6	2,9	2,7	3,0	2,4	2,4	2,3	2,8	3,0	2,9	3,7	3,7	2,7

Figure 6.2: Participant responses on the User Engagement Scale Short Form

6.4 Boxplots

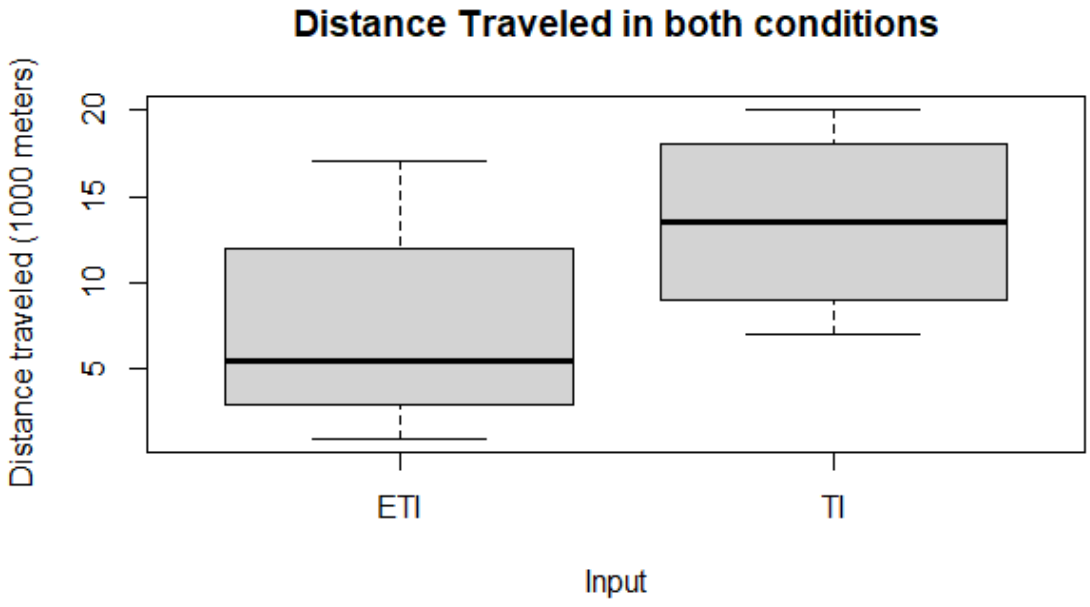


Figure 6.3: Boxplots showing the Distance travelled by players

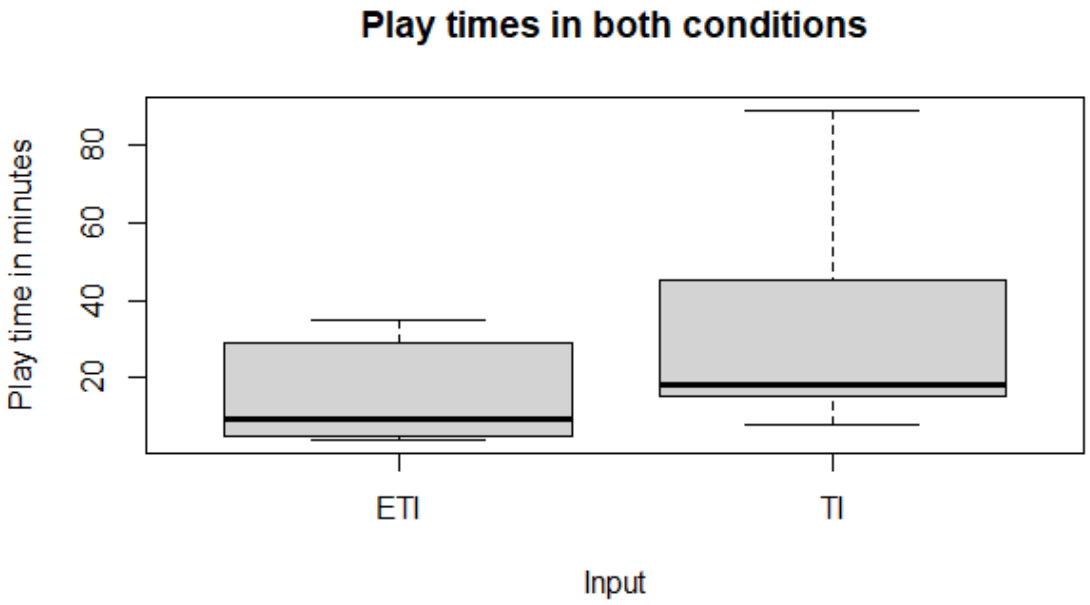


Figure 6.4: Boxplots showing the Time played (in minutes)

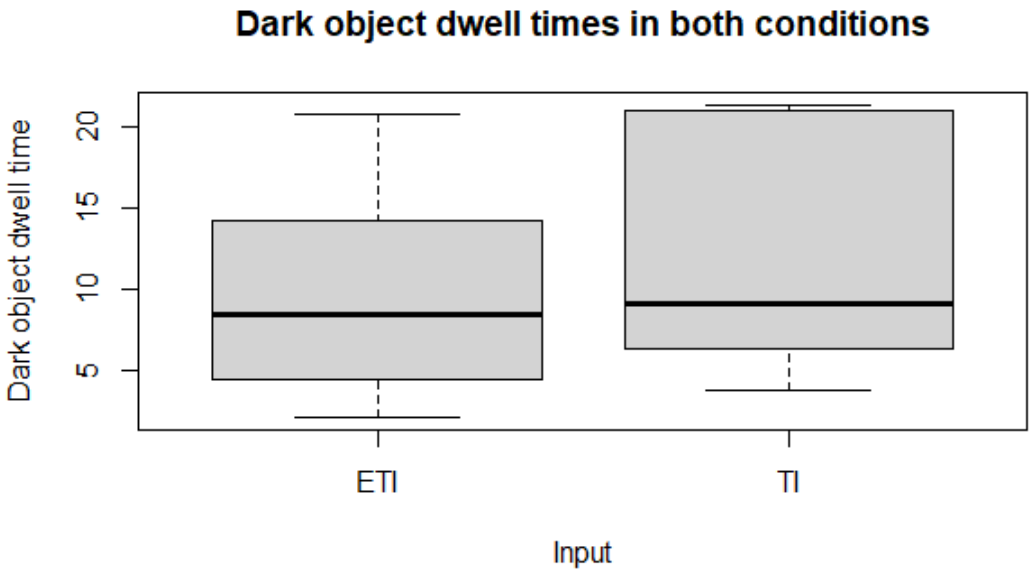


Figure 6.5: Boxplots showing the Dark Dwell Times of players

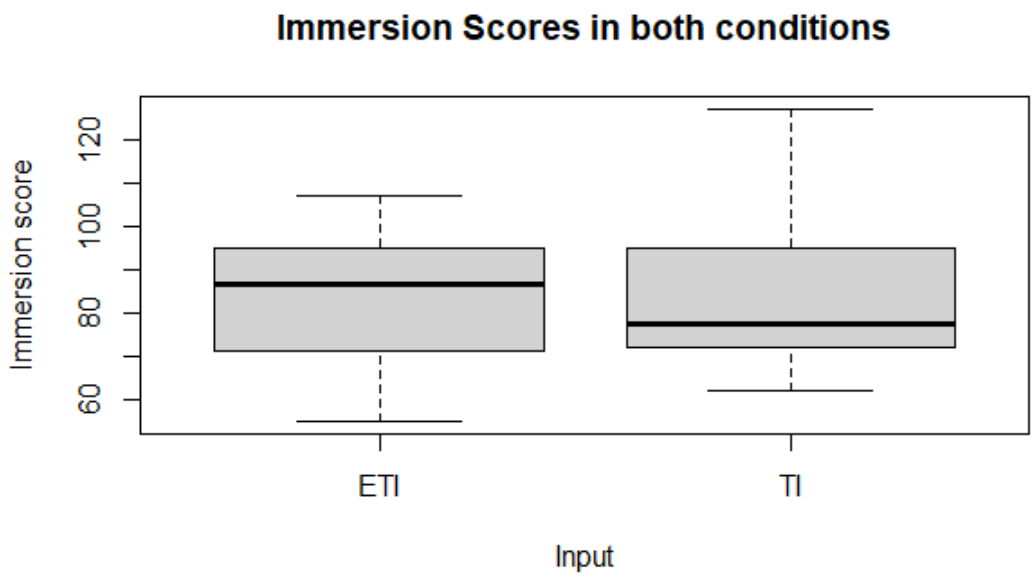


Figure 6.6: Boxplots showing the Immersion scores of players

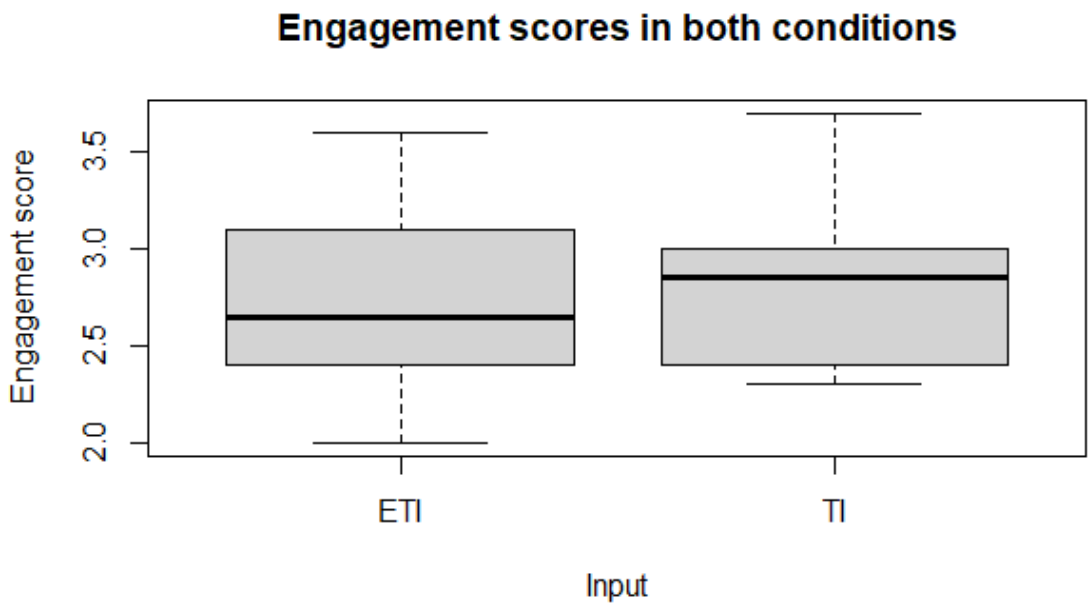


Figure 6.7: Boxplots showing the Engagement scores of players

6.5 Statistical test results

Below are the results of the statistical tests carried out on our data.

```
Levene's Test for Homogeneity of Variance (center = mean)
      Df F value Pr(>F)
group  1  0.2274 0.6392
      18
```

Figure 6.8: Levene's test results on Dark Dwell Times

```
Levene's Test for Homogeneity of Variance (center = mean)
      Df F value  Pr(>F)
group  1  7.1766 0.01532 *
      18
```

Figure 6.9: Levene's test results on Distance

```
Levene's Test for Homogeneity of Variance (center = mean)
      Df F value Pr(>F)
group  1  0.1027 0.7524
      18
```

Figure 6.10: Levene's test results on Engagement

```
Levene's Test for Homogeneity of Variance (center = mean)
      Df F value Pr(>F)
group  1  0.2846 0.6002
      18
```

Figure 6.11: Levene's test results on Immersion

```
Levene's Test for Homogeneity of Variance (center = mean)
      Df F value  Pr(>F)
group  1  3.9156 0.06335 .
      18
```

Figure 6.12: Levene's test results on Play Time in Minutes

```
        Wilcoxon rank sum exact test

data:  MED10ETI$DarkDwell and MED10TI$DarkDwell
W = 41, p-value = 0.5288
alternative hypothesis: true location shift is not equal to 0
```

Figure 6.13: Mann-Whitney test results on Dark Dwell Times

```
        Wilcoxon rank sum exact test

data:  MED10ETI$Distance and MED10TI$Distance
W = 20, p-value = 0.02323
alternative hypothesis: true location shift is not equal to 0
```

Figure 6.14: Mann-Whitney test results on Distance

```
        Wilcoxon rank sum exact test

data:  MED10ETI$TimeSecs and MED10TI$TimeSecs
W = 29, p-value = 0.123
alternative hypothesis: true location shift is not equal to 0
```

Figure 6.15: Mann-Whitney test results on Play Time in Minutes. There were ties, so we calculated for Time in Seconds as well.

```
        Wilcoxon rank sum exact test

data:  MED10ETI$TimeSecs and MED10TI$TimeSecs
W = 29, p-value = 0.123
alternative hypothesis: true location shift is not equal to 0
```

Figure 6.16: Mann-Whitney test results on Play Time in Seconds

```
        Shapiro-Wilk normality test

data:  MED10ETI$DarkDwell
W = 0.94903, p-value = 0.657
```

Figure 6.17: Shapiro-Wilk test on Dark Dwell Times in ETI

```
        Shapiro-Wilk normality test

data:  MED10ETI$Distance
W = 0.87875, p-value = 0.1262
```

Figure 6.18: Shapiro-Wilk test on Distance in ETI

```
        Shapiro-Wilk normality test

data:  MED10ETI$Engagement
W = 0.95586, p-value = 0.7378
```

Figure 6.19: Shapiro-Wilk test on Engagement in ETI

```
        Shapiro-Wilk normality test

data:  MED10ETI$Immersion
W = 0.95511, p-value = 0.729
```

Figure 6.20: Shapiro-Wilk test on Immersion in ETI

```
        Shapiro-Wilk normality test

data:  MED10ETI$TimeMins
W = 0.81205, p-value = 0.02031
```

Figure 6.21: Shapiro-Wilk test on Play Time in Minutes in ETI

```
        Shapiro-Wilk normality test

data:  MED10TI$DarkDwell
W = 0.82646, p-value = 0.03033
```

Figure 6.22: Shapiro-Wilk test on Dark Dwell Times in TI

```
        Shapiro-Wilk normality test

data:  MED10TI$Distance
W = 0.8136, p-value = 0.0212
```

Figure 6.23: Shapiro-Wilk test on Distance in TI

```
        Shapiro-Wilk normality test

data:  MED10TI$Engagement
W = 0.88768, p-value = 0.1597
```

Figure 6.24: Shapiro-Wilk test on Engagement in TI

```
        Shapiro-Wilk normality test

data:  MED10TI$Immersion
W = 0.88859, p-value = 0.1635
```

Figure 6.25: Shapiro-Wilk test on Immersion in TI

```
        Shapiro-Wilk normality test

data:  MED1OTI$TimeMins
W = 0.78335, p-value = 0.009093
```

Figure 6.26: Shapiro-Wilk test on Play Time in Minutes in TI

```
        Welch Two Sample t-test

data:  MED1OTI$Immersion and MED1OETI$Immersion
t = -0.23081, df = 17.331, p-value = 0.8202
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -19.24217  15.44217
sample estimates:
mean of x mean of y
     83.4      85.3
```

Figure 6.27: Independent t-test on Immersion

```
        Welch Two Sample t-test

data:  MED1OTI$Engagement and MED1OETI$Engagement
t = 0.57421, df = 17.968, p-value = 0.5729
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3457032  0.6057032
sample estimates:
mean of x mean of y
     2.89      2.76
```

Figure 6.28: Independent t-test on Engagement

Bibliography

[1] J. Smith and T.C. Graham. "Use of eye movements for video game control". In: 2006, p. 20. DOI: 10.1145/1178823.1178847.

[2] Niclas Ejdemyr. *Eye Tracking as an Additional Input Method in Video Games : Using Player Gaze to Improve Player Immersion and Performance*. eng. 2016. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:bth-11824 (visited on 02/05/2020).

[3] Erika Jönsson and Erika Jönsson. *If looks could kill - An evaluation of eye tracking in computer games*. Tech. rep. 2005.

[4] *Pixologic : ZBrush - The all-in-one-digital sculpting solution*. en. Library Catalog: pixologic.com. URL: http://pixologic.com (visited on 05/12/2020).

[5] *Maya Software | Computer Animation & Modeling Software | Autodesk*. en-US. Library Catalog: www.autodesk.com. URL: https://www.autodesk.com/products/maya/overview (visited on 05/12/2020).

[6] *Substance | The leading software solution for 3D digital materials*. en-US. Library Catalog: www.substance3d.com. URL: https://www.substance3d.com/ (visited on 05/12/2020).

[7] *Unite Austin 2017 - Game Architecture with Scriptable Objects*. URL: https://www.youtube.com/watch?v=raQ3iHhE_Kk (visited on 05/27/2020).

[8] Heather L. O'Brien, Paul Cairns, and Mark Hall. "A practical approach to measuring user engagement with the refined user engagement scale (UES) and new UES short form". en. In: *International Journal of Human-Computer Studies* 112 (Apr. 2018), pp. 28–39. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2018.01.004. URL: http://www.sciencedirect.com/science/article/pii/S1071581918300041 (visited on 04/21/2020).

A. Appendix

Below you can find the technical documentation that was included with the game files. The PDF titled "User Guide for MTA 201040 Testing" shows the instructions on how to set the Tobii eye tracker up. For those who did the TI condition the PDF only included the part from "About the game" and onwards.

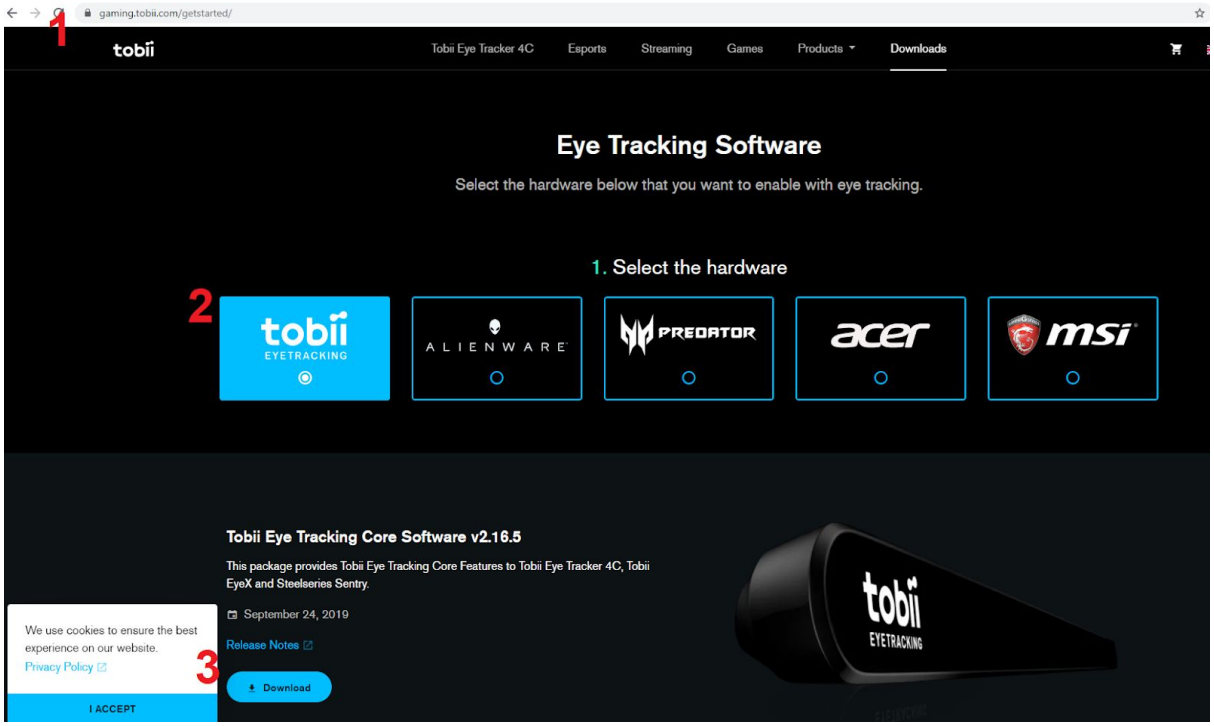
User Guide for MTA201040 Testing

Connecting and Positioning the Tobii 4C Eye tracker

1. Place the eye tracker under your screen so that the Tobii logo is on the left, facing up
2. Connect the eye tracker into a USB port
3. Wait until the device is connected and the driver is installed

Download the Tobii Eye Tracking Core Software v2.16.5

1. Click on the following URL: https://gaming.tobii.com/getstarted/
2. Find and select the Tobii eyetracking logo on the left
3. Click the download button

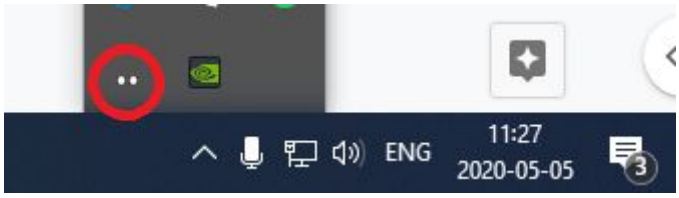


4. Install the application and follow the setup.

Please note that the eye tracker has to be calibrated for each user!

Adding more users

1. With the eye tracker connected and the software running, click on the software logo on the taskbar



2. On the bottom right, click on the username

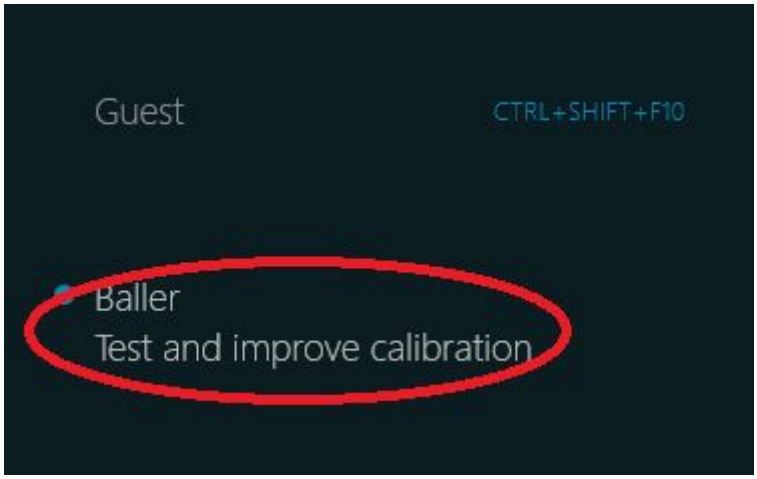
3. When the new panel appears on the left, click on Create New Profile…

4. Follow the setup instructions.

Recalibrating existing users

1. With the eye tracker connected and the software running, click on the software logo on the taskbar
2. On the bottom right, click on the username

3. Select your username and click Test and improve calibration



4. Follow the setup instructions.

You are now ready to run the test!

About the game

The game is about exploring a desaturated environment. By looking at the objects around you, you saturate them, and gain the temporary ability to spawn more. The game has no time limit, you are encouraged to explore and may leave at any point. After you're done, the game will present you with an overhead shot of the world.

Please select eye tracking in the game menu before starting!

Retrieving the collected data and screenshot

1. After completing the test navigate to the folder where the game is
2. Open the **MED10Project_Data** folder, and inside, navigate to the **Logs** folder.
3. You will find a .json file with the formatting **<input type>_<username>.json** and a .png file with the formatting **<input type>_<username>_screenshot.png**
4. You will be asked to upload these files to the appropriate fields in the questionnaire.

If you have any questions, you are welcome to contact us.

Link to the questionnaire

B. Appendix B

Below is an example of a consent form that was sent out to the participant prior to testing.

Participant consent form
Eye tracking in an emergent narrative
Group MTA201040

We are group MTA201040, from the 10th semester of Medialogy at Aalborg University.

For our thesis project, we have developed a game to compare immersion and engagement with eye traker input and traditional input.

We are kindly asking you to play through our game once and then fill out a questionnaire afterwards.

You have the rights to:

- Refuse to answer any question you find inappropriate
- Redact information at any time
- Stop the test at any point

All data will be anonymized and kept in a secure place for up to a year after which it will be deleted. The data may be used in scientific publications, reports and presentations.

I confirm that I have read and understand the information sheet for the above study. I have had the opportunity to consider the information, ask questions and have had these answered satisfactorily.

Please tick the boxes

	YES	NO
1. I confirm that I have read and understand the information sheet for the above study. I have had the opportunity to consider the information, ask questions and have had these answered satisfactorily.	☐	☐
2. I understand that my participation is voluntary and that I am free to withdraw without giving any reason.	☐	☐
3. I understand that all personal information will be anonymised and treated confidentially.	☐	☐
4. I agree to take part in the above study.	☐	☐

Name: _____ Signature: _____

Date: _____

Test facilitated by:

Mózes Adorján Mikó Contact: mmiko15@student.aau.dk