TRUFFLE HUNTERS

Final report for the master practical course

Intro

Explore the real world like never before with our interactive map, uncovering hidden treasures of magic truffles in your surroundings

Tatev Tsokolakyan
Natalie Adam
Nuo Xu

1. Background Story

We are a truffle trading company, and we have a job for you!

No matter how ridiculous the following words sound, please listen. Our clients are inhabitants of another dimension of the Earth. Even if you live in it every day, you can't see it or interfere with it.

This dimension is called the truffle-sphere. Many spore clusters grow in it, but they are invisible to the human eye, which we call Ether Truffles. Since the truffle-sphere shares the same spatial and weather conditions as our world, the growth of the Ether Truffle also relates to those factors that influence the fungus growth in our world.

Our clients, the inhabitants of the truffle-sphere, rely on those Ether Truffles to sustain their daily lives. However, they cannot collect truffles by themselves. In the past, our clients designed some collecting robots to help them collect truffles. However, something weird has been happening recently. Collecting robots stopped moving and collecting. A global famine is now spreading in the truffle-sphere. If there are no sufficient quantities of Ether Truffles collected, the population of the truffle-sphere will soon die out.

Our company now plans to hire humans to complete the harvesting operation. The business is still exploratory and requires only a license to search and pick Ether Truffles with the relevant equipment. Once the marketplace is created, you can sell them to us on the marketplace.

Come and join us!

2. Project Overview and Environment Setup

We utilize Node.js to develop our backend server, employing Express.js to manage HTTP requests and handle CRUD operations efficiently. MongoDB serves as our database, maintaining two tables: truffle information and user data. To interact with MongoDB, we employe Mongoose, a powerful object modeling tool designed for Node.js. Our server is hosted on an AWS T3.micro instance, chosen for its cost-effectiveness and suitability for applications with low to moderate traffic. To ensure consistent accessibility, we use Elastic IPs, providing a static IPv4 address for the server.

We choose Unity 2021.3.33f1 or higher version for the game development part. We utilize Mapbox for the map texture and location-based game development. As for the AR part, we use Unity ARFoundations and XR Interaction toolkit. Finally, we utilize shader and ONNX deep-learning effects to transform the player's surroundings with different visual effects.

We finished our truffle design last semester, and this semester, we modeled all our truffle prefabs in Blender using a low poly art style. Currently, there are four different types of truffles available. The type of truffle will become more important in the future when we implement the benchmark for trading truffles. There will be rarer types of truffles, which will trade at a higher rate. At the moment, one type of truffle is poisonous. Picking it up will trigger special visual effects as a metaphor for the player's vision being impaired.

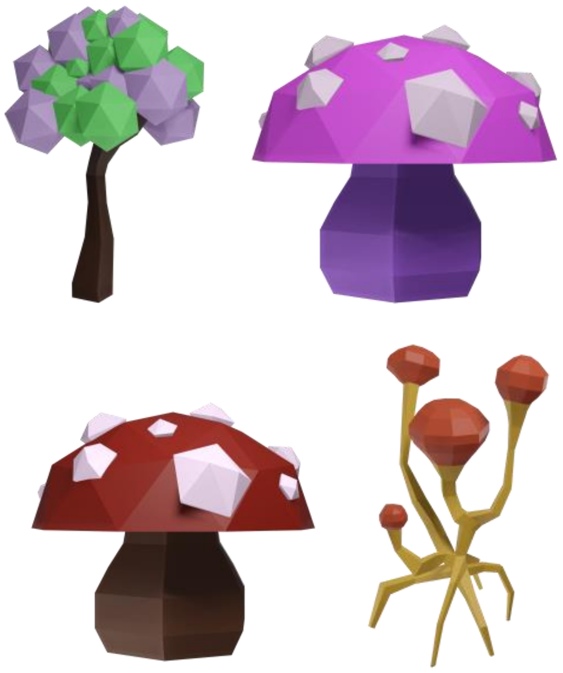


Fig 1. Types of truffles in “Truffle Hunters”

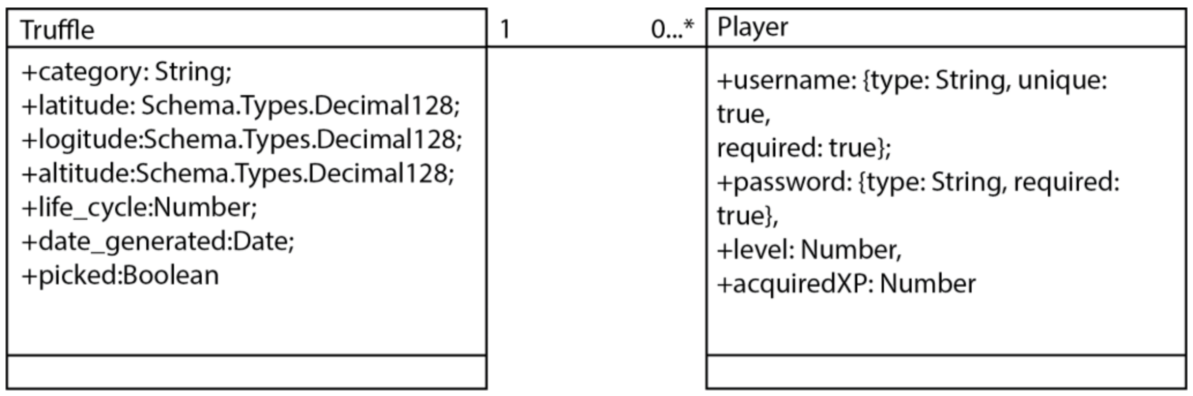
3. Backend Development

a. Database Model

We use MongoDB to store data on the server. MongoDB is a document-oriented database utilizing JSON-like documents with optional schemas. Our data models contain two schemas:

- Truffle model: stores information about truffles such as specific location, category, life-cycle, etc.
- Player model: stores the player's username, password, current level, and acquiredXP.

The following diagram shows the relationships between the truffle model and the player model. The connecting line between the boxes shows that Truffle and Player are related. The numbers close to the truffle model show that a Player can collect zero or more truffles, while the numbers on the other end of the line next to the player show that a truffle can be collected by zero or one player.



Truffle
+category: String;
+latitude: Schema.Types.Decimal128;
+logitude:Schema.Types.Decimal128;
+altitude:Schema.Types.Decimal128;
+life_cycle:Number;
+date_generated:Date;
+picked:Boolean
1
0...*
Player
+username: {type: String, unique: true,
required: true};
+password: {type: String, required: true},
+level: Number,
+acquiredXP: Number

Fig 2. Diagram for our data model

Once we finished the data model design, we generated 1000 random truffle locations around Munich and moved on to implement CRUD operations via routers.

b. Routes and Controllers

Express.js is one of the most popular Node web frameworks. Express's router package helps us to separate routers and their controllers (callback functions for corresponding routers) into different files and forward requests to correct controllers for further data manipulation. The following diagrams represent the basic flow of handling HTTP requests and responses in the server.

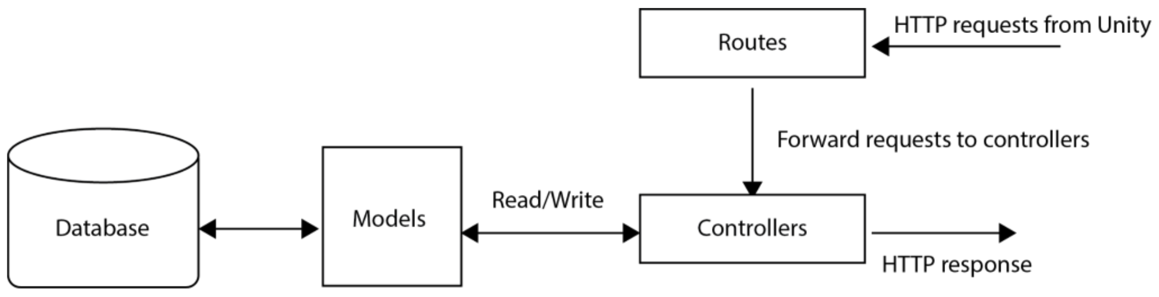


Fig 3.Diagram of handling HTTP requests and responses

We implemented "/login", "/register", and "/updateuser" to create a login session, register a new account and update the user's level and acquired points. Further, we created the corresponding controllers in... the login_controller.js file. The login function will first check if the username exists in the database, then check if the password is correct, and finally return codes and messages for login results. The register function will create a new user entry in the user table with initially 0 acquiredXP and level 1, while the update_user function will update the acquiredXP and level in the database.

Like manipulating truffle data, we implemented "/api/truffle", which will return a list of all truffle information, whereas "/api/truffle/:longitude/:latitude" will return truffles close to the player's current location. The "api/truffle/update" will update the corresponding truffle information, marking it as picked.

c. Server Client Talking

After finishing the server backend setup, we moved on to implementing network communication on the client side. UnityWebRequest provides a modular system for composing HTTP requests and handling HTTP responses. When the player opens the application, our game will load the login/register page, and the user can choose to log in or create a new account. The login.cs script will POST a form containing a username and password to the login or register router. Once the user successfully logs in or creates a new account, the GameManager object stores their information throughout the whole game session. Then, the Map scene will be loaded, and the ServerTalker.cs will POST the user's current location to the server to fetch nearby truffle locations and start spawning pinpoints on the map. Once the user collects some truffle, their acquired points and level will increase. The UpdateUser.cs will send another form with the user's updated acquiredXP and level to the server to update the corresponding data in the database. All these three server-client talking functions are implemented in coroutine and through UnityWebRequest.

4. Location-Based Games

Our game client contains three scenes: *Login*, *Map*, and *Capture*. The *Login* scene contains a UI form, which submits the username and password to the server. Upon successful validation by the server via the *FormSubmitter* script, the game transitions to the *Map* scene.

The *Map* scene integrates the *Mapbox API* to provide map functionality, including location tracking and event spawning. The player's 3D character is spawned at their GPS coordinates, enhancing immersion. Using the *ServerTalker* script, player location data is sent to the server, triggering the retrieval of truffle locations stored in a *JSON* file. These locations are seamlessly integrated into the map using the *SpawnOnMap* script, visually represented by 3D pins. If the player collides with one of these pins while exploring the map, the *TrufflePlane* scene is loaded, facilitated by a box collider added to the 3D pin prefab. Additionally, a small button in the *Map* scene allows users to view their accumulated points, with potential plans for a leaderboard feature.

In the *TrufflePlane* scene, a random selection of truffles appears for collection. Upon gathering all truffles, the player's XP points get updated on the server. Subsequently, the *Map* scene reloads, enabling the player to explore another truffle location. Users can exit the *TrufflePlane* scene at any time by tapping the close button.

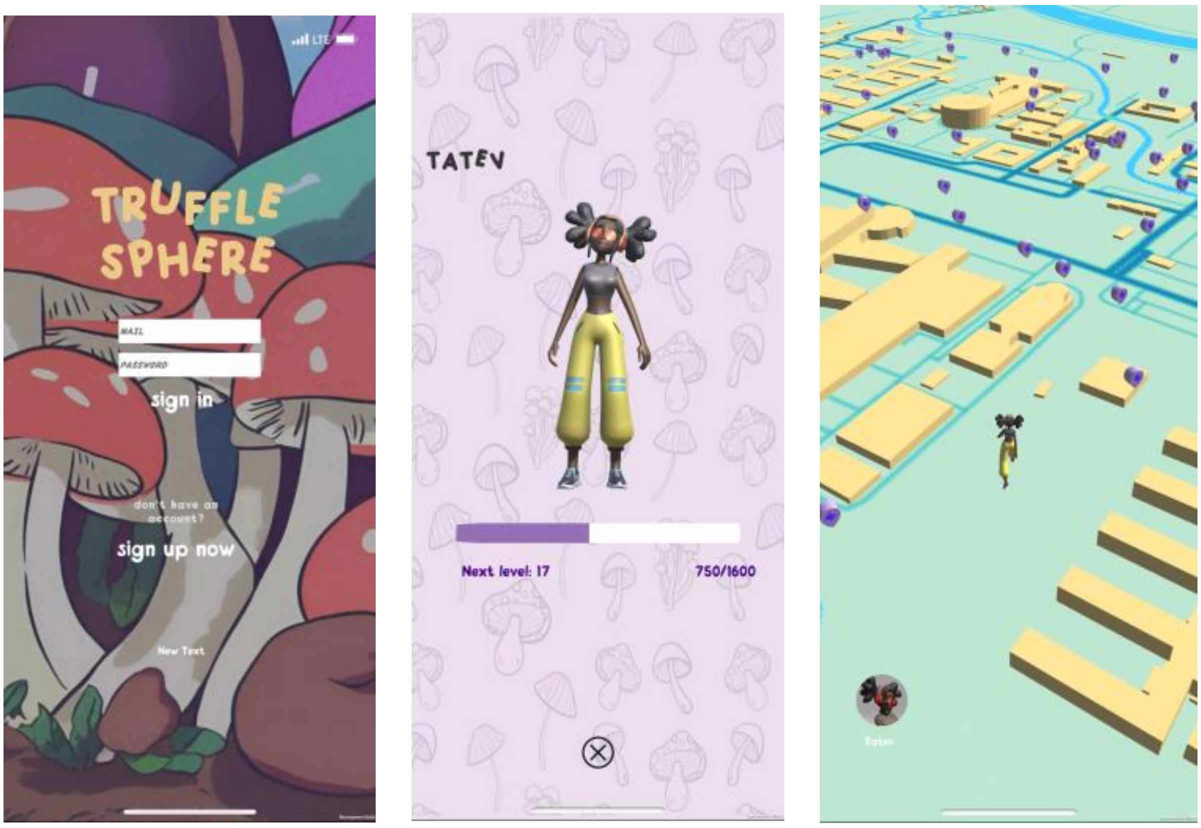


Fig 4-6: Left: Log in page; Middle: player's achievements; Right: Map

5. Truffle Spawning

We narrow down the area truffles can spawn in by fitting a box collider around the GPS location of the truffle provided by the server. When a player enters the area of the box collider, they can switch to the camera view on their phone and use their camera to search for truffles in the real world. To place our truffles in the real environment, Unity provides the AR Foundation package that offers various AR features to create AR apps. We use AR Foundation in combination with the Google ARCore XR Plug-in and the Apple ARKit XR Plug-in (Unity Technologies 2022).

a. Trackables

When a player discovers a truffle, we want to spawn the truffle in the real world. However, we want the truffle to stick to a surface and not float around. Therefore, we need knowledge about the environment. AR Foundations includes “Trackables” that can detect and track geometry in the real world by adding the corresponding Trackable Manager to the AR Session Origin GameObject (Unity Technologies 2022).

We compared three different methods to detect surfaces that seemed promising for our purposes:

- Meshing: Generates a mesh based on the scanned environment. However, despite good results, we decided against using meshing since it is currently only supported on iOS (Unity Technologies 2022).
- Point Clouds: Creates sets of feature points based on notable features in the environment (Unity Technologies 2022).
- Plane Detection: Creates *GameObjects* for each tracked flat surface in the environment. There is an option to detect horizontal surfaces, vertical surfaces, or both (Unity Technologies 2022).

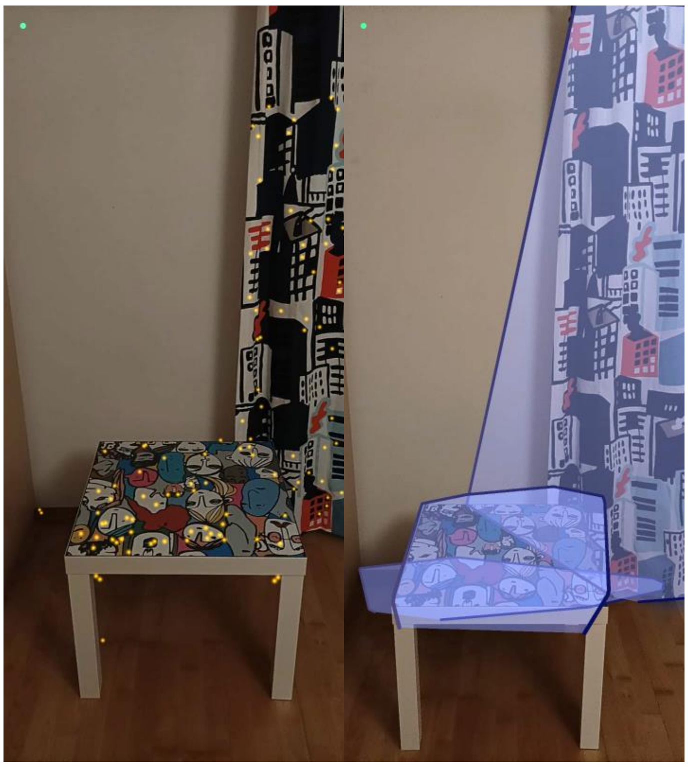


Fig 7-8. Left: Point Clouds; Right: Plane Detection

Now that we've determined the surfaces in the real world, we want to spawn truffles accordingly. We use the AR Raycast Manager to constantly shoot rays from the player's camera into the scene. When a ray hits a trackable, we spawn a random truffle at the location of the intersection (Unity Technologies 2022). For future work, we want to adjust the type of truffle based on the weather at the player's location.

Ultimately, we've chosen Plane Detection over Point Clouds since the results are more reliable, probably because hitting a plane with a ray is easier than hitting a single point.

Besides the spawning location of truffles, the server also determines the number of truffles that should spawn at a given location. To not spawn all truffles at the first intersection of a ray with a surface, we added the variable *"distanceTruffle,"* which defines the minimum distance between all spawned truffles in a scene. This variable also makes sure that truffles don't overlap when spawning.

b. Object Occlusion

When an object in the environment moves in front of a spawned truffle, the truffle should disappear (partially) behind that object. To implement object occlusion, we need to determine the depth information of the objects in the scene. The AR Occlusion Manager enables the background rendering pass and renders the truffles according to the information in the depth buffer of all the objects in the scene (Unity Technologies 2022).

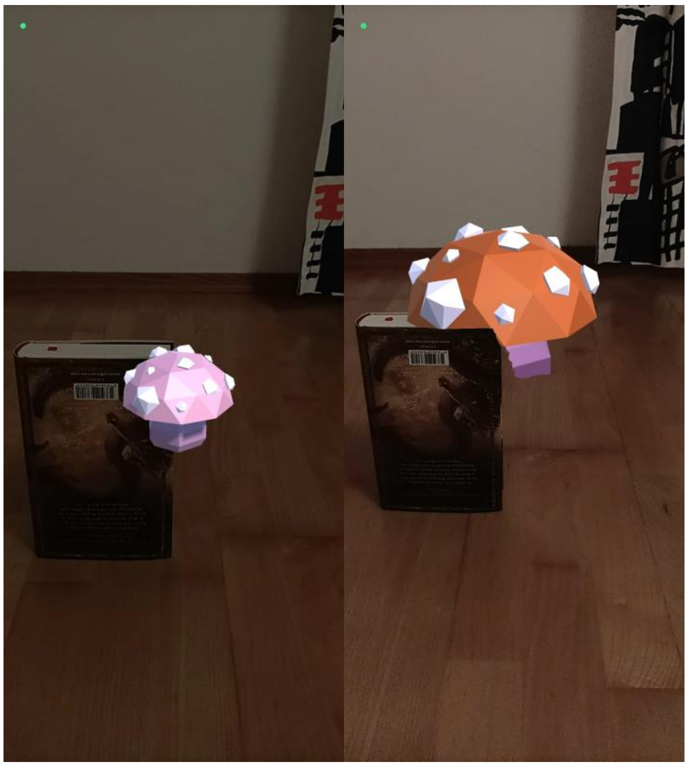


Fig 9-10. Left: No occlusion; Right: AR Occlusion Manager

6. Truffle Collection and Visual Effects Implementation

A player can collect a truffle by clicking on it. The truffle disappears from the player's view with a particle explosion. To prevent a new truffle from being spawned immediately at the vacant position again, we add the truffle position to a list of positions where no new truffle is allowed to spawn, and a cooldown for the truffle starts. Only once the cooldown of the truffle reaches the defined "spawnCoolDownDuration" does the truffle fully disappear from the scene, and a new truffle can spawn at that location. The collected truffle's points are directly added to the player's profile, updating it on the server.

We implemented two approaches: real-time style transfer model inference and a shader pipeline of motion-reactive point clouds. However, due to the limitations of graphics capabilities on older Android devices, the style transfer effect is only available on iOS platforms.

a. Style Transfer

We utilize Unity Barracuda, a deep-learning inference library, to implement style transfer to the capture scene. This effect involves applying the visual style of one image (in our case, a pre-trained mosaic style) to another image (the game scene captured by the camera), making game more immersive. We encountered several challenges when implementing the post-processing effect with Unity Barracuda. The most prominent one was to integrate the style transfer effect with the game environment while maintaining performance and visual coherence.

First, we created a script named "StyleTransfer" and attached it to the main camera game object. This script initializes a worker type for the *Barracuda inference engine*. Within the script, we utilize the OnRenderImage() method instead of the Update() method to call the StylizeImage() method. This decision allows access to the *RenderTexture* for the game camera as well as the RenderTexture for the target display. We invoke StylizeImage() if the boolean variable stylizeImage is set to true, ensuring that the effect is applied only when desired. Before running the style transfer inference, we preprocess the input image. The style transfer models expect RGB channel values to be in the range [0, 255], whereas color values in Unity are typically represented in the range [0, 1]. Therefore, we scale the RGB channel values of the InputImage by 255 to ensure compatibility with the model.

Besides style transfer, we also explored Pix2Pix this semester. Pix2Pix learns a mapping from input images to output images and achieves a more intriguing and psychedelic effect. We pre-trained a conditional Generative Adversarial Network (cGAN) using a dataset of space images. This cGAN was trained to transform pre-processed blurry black-and-white images into space-themed visuals. After training the model using TensorFlow, we converted it to the ONNX format, compatible with Unity Barracuda. However, due to constraints on mobile device resources, we opted for a more lightweight style transfer model with fewer parameters.

b. Particle System

We also employ a simple shader pipeline to dynamically transform the environment into moving particles based on the surrounding's motion. To achieve this goal, we first extract the movement vector by saving it to a texture, utilizing optical flow between consecutive frames to capture motion information. This texture serves as a crucial input for our subsequent shader operations. We then utilize a combination of vertex and geometry shaders to render points in the space and apply movement based on the optical flow texture. The vertex shader processes each vertex of the image, while the geometry shader enables the generation of additional geometry, allowing for the creation of point clouds. As a result, we achieve a visually captivating effect that dynamically reacts to the motion within the game scene.

The shader approach is computationally cheaper compared to the deep-learning approach. Therefore, we ensure efficient performance and a seamless gameplay experience across various devices.

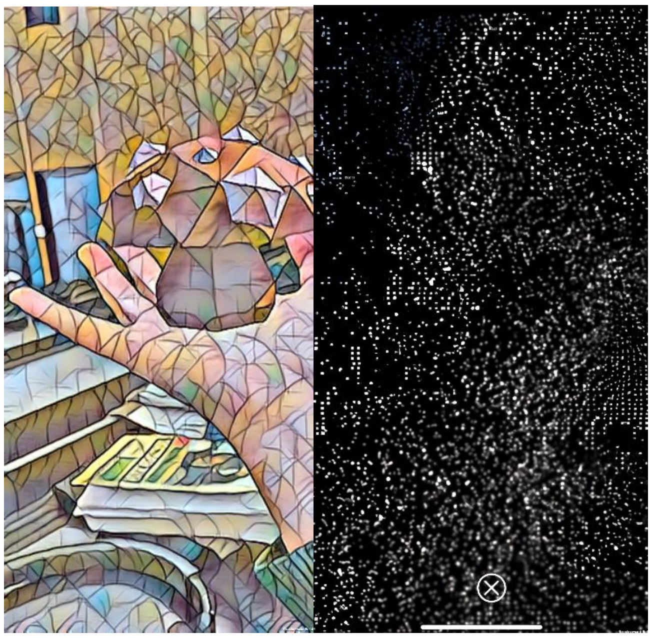


Fig 11-12. Left: Style Transfer Effect; Right: Particle System Effect

c. Make the Hunting More Enjoyable

One unique feature of our game involves a post-processing effect applied to the main camera after collecting a certain type of truffle (poisonous). The mosaic visual effects will only be deactivated after the player collects another non-poisonous truffle, while the altered visual style makes locating non-poisonous truffles more difficult. This little trick design makes the gameplay more enthralling and challenging to the player.

Another engaging feature is taking a picture. Players can directly save frames of the gameplay to their device's photo gallery. Taking pictures of rarer truffles or player's interaction with truffles will encourage players to share memorable moments on different social media platforms.

7. Future Work

- Truffle generating system: The initial idea was to maintain a truffle generating system which can generate truffles with different values based on local weather data, such as temperature, humidity, sunlight strongness, etc. For example, when it is raining today, players can infer where more valuable truffles appear tomorrow.
- Audio AR: Another interesting yet unexplored idea is utilizing audio AR for navigation. We would like to combine the idea of audio beacon (a certain rhythm that will be played when players head in a certain direction) from Soundscape into our navigation part in the future.
- Trade system: After players collect truffles, they wound not want to let them stay as some points or achievements in their personal pages. In the future, players can trade truffles for coins to buy advanced truffle collection tools.
- Truffle collecting tools: We also would like to model more collecting tools, like hooks, detectors, robots etc. Players will need more advanced tools for rare truffles and they need to collect more truffles to trade to get more advanced tools.

References

Unity Technologies. (2022). "Unity AR Foundation Manual." Retrieved 12. March 2024, from https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.1/manual/index.html.

MozDevNet. “Express tutorial part 3: Using a database (with mongoose)” Retrieved 25. March 2024, from https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose

MozDevNet. “Express tutorial part 4: Routes and controllers” Retrieved 25. March 2024, from https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes

Unity Technologies. “Unity Documentation version 2022.3 Scripting API: UnityWebRequest” Retrieved 25. March 2024, from https://docs.unity3d.com/ScriptReference/Networking.UnityWebRequest.html

Pix2pix: Image-to-image translation with a conditional Gan : Tensorflow Core. (n.d.). Retrieved 25. March 2024 from, https://www.tensorflow.org/tutorials/generative/pix2pix

Microsoft. (n.d.). Microsoft/Soundscape: An IOS application/service that AIDS navigation through spatialized audio. Retrieved 25. March 2024 from https://github.com/microsoft/soundscape