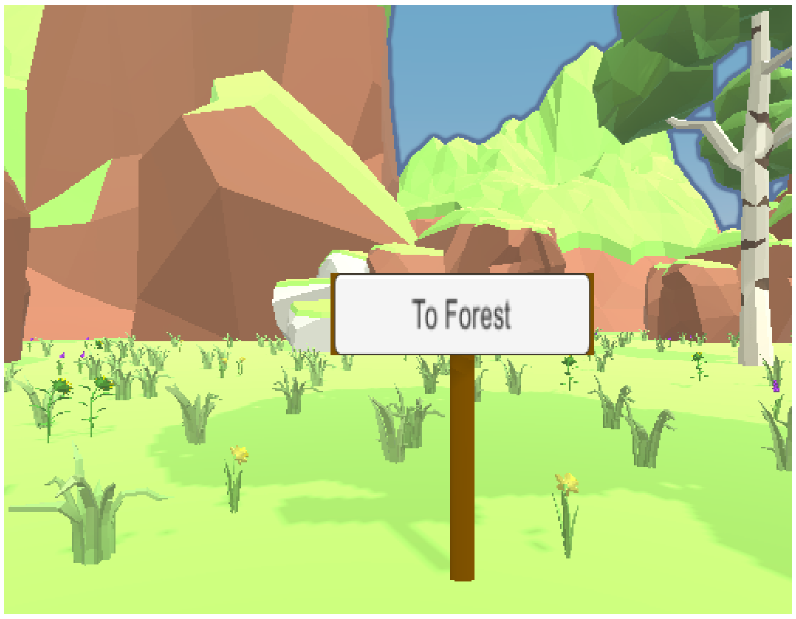
Elaine Lee, Hong Sen Du, Sandy Zhao

**One-paragraph overview:**
DuckDuckGoose is a VR game where users can interactively take care of their virtual pet duck. In the main area where the duck resides, users can pet the duck, play fetch with the duck using a ball, move around a duck house and pieces of furniture, and walk around with the duck following them. In addition, a hidden stat panel on the duck displays its current affection level and hunger level. Users can also feed the duck mushrooms. To forge mushrooms, users can teleport into a separate forest area and put collected mushrooms into their sack. Finally, users can carry the mushrooms back to the main area through the sack and feed the duck to keep it healthy and happy.

## Main Area

### Moving around the area



To move around in the main area, the user has two different options:

1. Physical walking with 1:1 mapping. The Main Camera tracks the headset's position and rotation, and will move according to the user's physical movements. This method minimizes VR sickness and feels more natural.
2. Joystick locomotion (left joystick). The user might not have access to a large physical space with no obstacles. With physical walking alone, they would not be able to navigate the large main and forest areas. To accommodate users playing in rooms of all sizes, joystick locomotion is also provided.
3. Scene transitions. To move to the forest scene, the user has to get close to a sign that says "To Forest." This sign has a UI button. When the button is pressed, Unity's Scene Manager will load the corresponding scene. We decided not to put the forest and main area into one scene because it would make the landscape way too big. Rather than

have the user waste time walking from the main area to the forest and vice versa, it would be a lot more efficient if this happened through scene changes.

## Moveable objects

Most of the objects in the landscape are immovable, since it is unrealistic for the user to move mountains and big trees. However, certain “man made” objects (umbrella, duck house, beach chairs) can be moved around the main area. The user can do so in two ways:

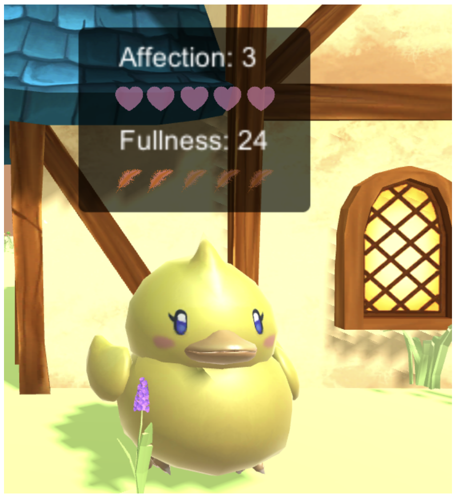
1. Grab and drop. By pressing the “select” button on their controllers, the user can pick up the item and move/rotate it around. To place the item again, they can simply let go of the button. Once the item is dropped, only translations in the xz-plane and rotations along the y-axis are preserved. In other words, the items will always right themselves and stay on the ground.
2. Controller buttons. The user can hover their virtual hand over the item or point at it with their virtual pointer. While hovering, the user can translate the item along the xz-plane using the right joystick. By pressing the A and B buttons on the right joystick, the user can rotate the item along the y-axis.

In both of these situations, we limited the range of movement for these objects to xz-translation and y-rotation. While allowing free-for-all transformations would have been easy, it is not realistic to have a house floating in midair or lying on its side.

## Virtual hands and raycast pointers

In our game, we offer two options to manipulate and interact with objects: virtual hands and the traditional raycast pointers. We chose support for these two in order to make every action within the application feel more natural depending on the player’s intentions. For example, to interact with the duck, it is much more natural to only be able to interact using your hands. The same applies when foraging for mushrooms in order to mimic the physical movement of picking up/storing the mushroom and feeding the mushroom to the duck. However, in terms of object manipulation, we chose that interacting with these objects via raycast pointer made the most sense. Additionally, if the player wants to rotate the object, using virtual hands would be more awkward as the player would have to use both hands to slowly rotate the object around. As a result, a pointer allows the user to also manipulate objects from afar as well as rotate and translate the objects using the keybindings we have defined above in addition to using the virtual hands. To make sure some interaction methods do not collide, we do not allow the raycast pointer to interact with the duck by putting the duck on a different interaction level. Support for both of these options offers the fullest range of interaction for the user for a better experience.

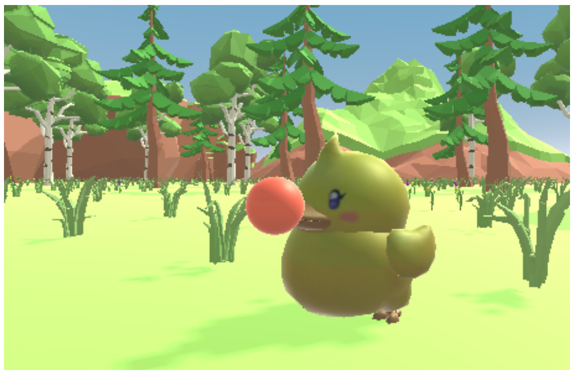
## Duck stats

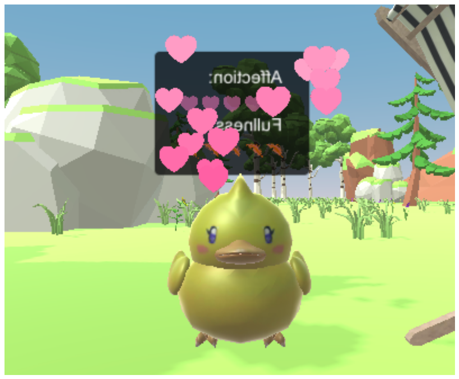


The duck tracks two different statistics throughout the game: affection and hunger. Both of these stats will steadily decrease over time, and the user has to keep feeding and interacting with the duck to prevent it from getting hungry and depressed. The stats display has several key features:

1. The stats appear on a panel with a dark background over the duck's head. We chose this location instead of a display somewhere static in the world in order to make it more easily viewable without having to look somewhere else. This allows the user to always be updated on the duck status in an area that is logical. We also made the panel always look at the player so no matter what orientation the player is relative to the duck, the user will always be able to see the stats.
2. The stats only appear above the duck's head when they change (happens when hunger/affection drops, or when the user interacts with the duck). We decided to keep the duck stats hidden most of the time because the goal of this game is to create a believable environment, and having a floating UI display takes away the believability. This also notifies the user that there has been a system status change.
3. In addition to an affection and hunger count, we also have a 5-star system for displaying affection and hunger. Even though numbers are helpful to understand how much affection/food the duck currently has, the star system lets the user easily visualize how happy/hungry it is relative to the ideal level.

## Interacting with the duck





As mentioned in the prior section, the duck will get hungry and sad without sufficient feeding and petting. The user can interact with the duck in several ways:

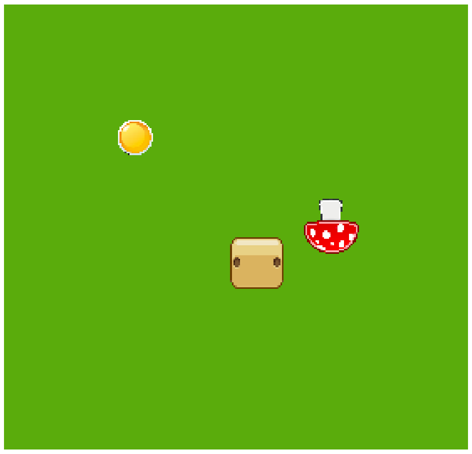
1. Walking around with the duck. Thanks to Unity's NavMeshAgent, the duck will follow the player wherever they move.
2. Petting the duck. The user simply has to hover their virtual hand (controller) over the duck. In response, the duck will release a bunch of heart particles, and the affection count will increase. The duck will also make a 'quack' sound to provide more feedback for the user that he/she is doing a great job.
3. Feeding the duck. The user can reach into their supply bag (on the left side of the body) and use the select button to grab a mushroom. Once the mushroom triggers the collider on the duck's beak, it will be "eaten" and the hunger scale will change accordingly.
4. Playing fetch. The user can pick up and throw a special orange ball. Once the ball leaves the user's hand, the duck's NavMeshAgent will start following it instead of the user. When the duck collides with the ball, it will pick it up and walk back to the user. Upon arriving at the user's location, it will drop the ball and start following the user around again as usual.
5. Growth and death. The duck's size changes according to its hunger scale. When the duck's hunger hits 0 or >50, the duck will explode and die. Just as a real duck should not be overfed or starved, we want to teach our users the same lesson with a virtual duck.

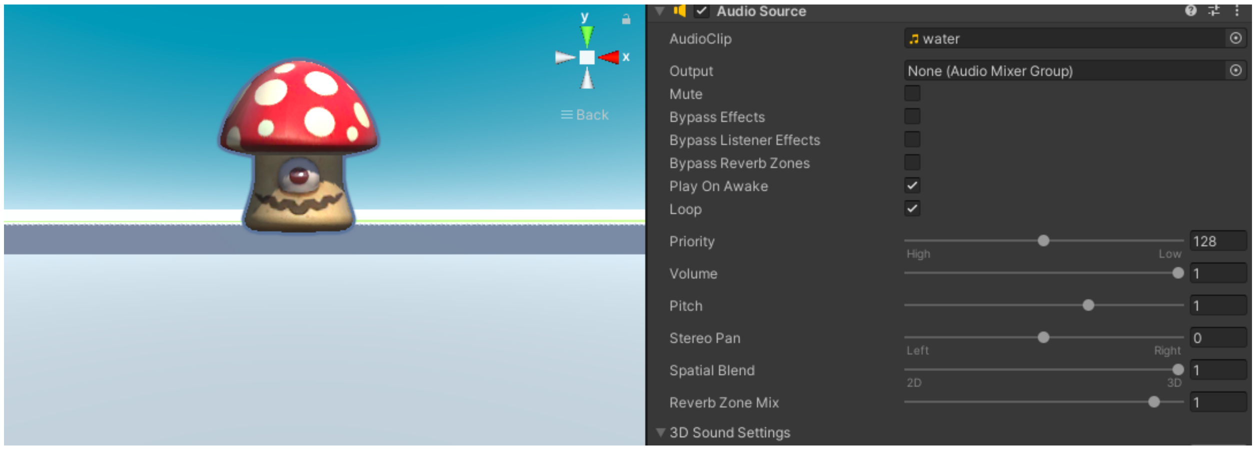
# Forest

## Moving around the area

Similar to the Main Area, the user has two different options to move around to provide maximum flexibility.

## Wayfinding and navigation





Audio Source

| | |
|---|---|
| AudioClip | water |
| Output | None (Audio Mixer Group) |
| Mute | |
| Bypass Effects | |
| Bypass Listener Effects | |
| Bypass Reverb Zones | |
| Play On Awake | ✓ |
| Loop | ✓ |
| Priority (High – Low) | 128 |
| Volume | 1 |
| Pitch | 1 |
| Stereo Pan (Left – Right) | 0 |
| Spatial Blend (2D – 3D) | 1 |
| Reverb Zone Mix | 1 |

3D Sound Settings

The forest area is very big and has lots of procedurally generated trees, so it may be difficult for the user to navigate to their destinations. To make things a little easier, we implemented two different wayfinding features:

1. Minimap. The map is attached to the user's left controller and can be toggled on or off using the X button. This map shows a birdseye view of the user's surroundings, making it easier for them to see the "Main Area" sign or any mushroom monsters. This map ignores the non-interactable trees and flowers, only showing the grabbable mushroom monsters and scene transition signs. We decided to limit the amount of icons on the maps to prevent overloading the user with too much information.
2. Binaural cues. When the user comes within range of a mushroom monster, an attached AudioSource starts playing sound cues. As the user gets closer, the volume gets louder.

Because the AudioSource has been configured to 3D, it will also give the left and right ear different feedback depending on its relative position to the user. For example, if the user is to the left of the mushroom, they will hear more sound in their right ear because their right ear is closer to the mushroom.

## Moveable objects: Mushroom Monsters

In the forest, users are able to pick mushrooms through selection and manipulation. Once a user locates a mushroom through wayfinding techniques described above, they can then uses either hands to pick up a mushroom and put it in the sac located at the left side of the user. Although we use stereo audio clues as one of the options for wayfinding, all audio clips are turned off once the mushroom is being picked up.

**Rationale behind using mushroom monster:**
Although we have a lot of purple and yellow mushrooms that look more edible in the forest scene, we decided to use these objects as forest decorations instead of manipulable objects for maximum user enjoyment. We have combined these mushroom prefabs with rocks, grass, flowers, and trees to create a realistic and vivid forest scene. Thus, as a result, we would like to use a different mushroom object for users to pick up and interact with. Mushroom monsters provide fun animations and are also easily distinguishable among other immobile mushrooms. This maximizes the experience of wayfinding and the joy of spotting a mushroom object to interact with.

**Continued interaction with mushroom monsters:**

1. Once users put a mushroom monster into the sack, they can also continue interacting with it by getting it out of the bag and placing it on the ground. Again, this is to increase the variabilities of manipulation techniques and keep consistency with the Main Area, where users would be able to pull mushroom monsters out of the sac and feed the duck.
2. Another way for continued interaction with the mushroom monsters is through instantaneously spawned mushroom monsters. Each time a user picks a mushroom monster from the ground, a new mushroom monster is randomly spawned into the forest scene. At each given time, there are five mushrooms in the forest, each at least far apart enough from each other that the stereo audio clue is triggered exclusively by one mushroom monster at a time. Such design allows users to continuously find and pick up new mushrooms without seeing too many of them at a given time.