WWU - Computer Science Department
CSCI 497M/597M Virtual Worlds
Final Report – Fall Quarter 2021

Natural Selection Simulation

Jeremy Hummel, Walker Herring

Abstract

Our objective is to create a 3D virtual ecosystem that simulates several types of organisms undergoing the process of natural selection. This project will later be included as one component of a larger virtual world. The natural selection simulation will consist of organisms which have a set of randomly determined traits, such as size and number of offspring, among others. These organisms will reproduce and pass on their traits, with slight alterations, to their offspring. There will be several types of organisms: plants, which cannot move and reproduce asexually; herbivores, which eat plants and require another herbivore to reproduce; and carnivores, which chase after and eat herbivores. In order to make the virtual ecosystem fit into a virtual world, networking will be implemented to allow users to connect to a server, move around the ecosystem, and spawn organisms of their own.

The Unity game engine will be used to render the 3D simulation. We will implement a genetic algorithm to simulate natural selection, and we will use the uMMO package to make the virtual ecosystem networked and persistent.

1. Introduction

Motivation

Firstly, our dynamic ecosystem will create a framework for AI that future virtual worlds projects could utilize. The organisms we are creating could be repurposed by others as AI agents that can dynamically learn to perform a task in the virtual world.

Dynamic AI agents like the organisms in our virtual ecosystem make a virtual world more interesting and immersive to users. Users are more interested in a virtual world if it changes around them, rather than remaining static. In many virtual worlds, a user who visits a given area will always see the same AI agents performing the same preset tasks. If users of a virtual world visit a virtual zoo and see the same static animations being played on a loop, they will be unlikely to come back. However, if those animals behaved in a way that was determined by a genetic algorithm, users would be much more interested in

observing their behavior. Users will be further engaged by the opportunity to create their own virtual organisms and observe how this affects the ecosystem.

This project could also be used in an educational context. A middle-school science teacher might utilize a natural selection simulation to demonstrate to students how the "survival of the fittest" principle allows organisms who are well-suited to the environment to thrive. Students could perform simple experiments by adding new organisms to the ecosystem and observing the results.

Challenges

The biggest challenge will be creating a genetic algorithm that allows meaningful changes to occur in the ecosystem. If the changes that can occur between generations are too minute, users will not be able to appreciate the dynamic nature of the ecosystem. On the other hand, if they are too dramatic, the populations of plants, herbivores, and carnivores will become imbalanced and everything in the virtual ecosystem will die off. Before we know if the genetic algorithm is capable of generating worthwhile results, we need to implement multiple types of organisms. For this reason we will not be able to test the main algorithm in our project until most of the work is completed.

Performance will also pose a challenge. The ecosystem must have enough organisms that a meaningful process of natural selection can take place. However, we will need to prevent a huge number of new game objects from being created, as client-server communications about the positions of all those objects would not be feasible.

Another challenge we will face is keeping the scope of the project reasonable. It will be tempting to add numerous traits to each organism before the genetic algorithm has been implemented. This could lead to a situation where we are not able to create and test a functional build. We will need to focus on building a functional natural selection simulation, rather than adding too much complexity to the organisms themselves. We can then go back to the organisms and add interesting traits and features when this is done.

2. Related Work

Key Technologies

Our key technologies will be Blender [1] for art assets and animation, Unity's own NavMesh[2] for pathfinding, DOTween[3] for tweening, Sensor Toolkit[4] for AI sight, uMMO[5] for a networked and persistent world, Doozy UI[6] for any in game UI, and MyBox[7] for some convenient Unity C# extensions and tools. Design patterns we will utilize are singletons for game and system managers, observers for systems that need to observe for changes in game and system states, factory design pattern and object pooling for large amounts of organisms, and state machines for the AI, system and world states. The reason for us choosing to use these tools and design patterns comes down to us being more familiar with these tools and design patterns over other options and they allow us to quickly prototype builds and test out concepts as we are working with a very small amount of time.

We will create our own meshes, surfaces, and animations for the moving organisms and user drones.. We will get meshes and surfaces for the plants from a free low-poly asset library we have selected from the unity asset store[8]. The Cartoon FX library[9] will be used to add particle effects when organisms reproduce and die.

Related Class Projects

- Resource Generation – While we perform initial tests on our virtual ecosystem, we will place organisms in the scene manually. However, we will coordinate with the resource generation group so that their algorithms can place an appropriate number of our organisms in a scene. This way, we will not need to create our own redundant resource generation algorithm to get our scene running.
- Terrain Generation – The terrain generation group will create randomly generated terrain meshes. We will coordinate with them to ensure our AI will work with their terrain system. During testing, we will use a flat scene with a flat NavMesh overlayed on it. However, we must eventually be able to overlay this NavMesh on the terrain mesh the terrain generation group will be generating instead.

3. Architecture

Requirements

- Within the scene there are:
    - Plant Organisms (Herring)
    - Herbivore Organisms (Hummel)
    - Carnivore Organisms
    - Water Pools
- All organisms grow from very small to full size over a period of time after they are born. (Herring)
- All organisms have traits called genes that are modified via a genetic algorithm. These traits are taken from the parent organism(s) and then randomly modified to determine their new value upon the organism's birth. Not every organism makes use of every gene, but they all contain a Genetics class. The following traits will be part of the Genetics class: (Hummel)
    - Sex
    - Size
    - Maximum movement speed
    - Maximum distance where the organism can see other organisms
    - Time to forget the location of a previously seen object
    - Metabolism (rate at which energy is used up)
    - Duration of initial growth stage
    - Desire to reproduce
    - Duration of pregnancy (mobile organisms) / time between reproductions (plants)

    - Chance that a reproduction successfully produces an offspring
    - Maximum number of offspring the organism can have during its life
    - Chance that the organism's genes will be mutated when it passes them on to its child.
- Organisms have an energy property. (Hummel)
    - When an organism has no energy, it dies.
    - Plants contain some amount of energy based on their size. This energy depletes over time until the plant dies.
    - Mobile organisms use a small amount of energy in their idle state, and more energy while in their fleeing, mating or persiung states. Each state defines a constant value which is multiplied by the organism's Metabolism to determine how much energy is subtracted each game tick.
- Mobile Organisms (herbivores and carnivores) can eat other organisms, destroying them and gaining energy equal to the consumed organism's energy. (Hummel)
- Mobile Organisms have a hydration property.
    - When an organism has zero hydration, it dies.
- Mobile Organisms can move towards water and drink, gaining hydration.
- Mobile Organisms can move towards food when they are hungry, and water when they are thirsty. (Hummel)
- Mobile Organisms can only locate a destination if it is within a cone in front of them. The length of this cone is determined by their sight distance trait. (Hummel)
- Mobile Organisms make use of a data structure to remember other organisms they have recently seen. When an Organism enters a Mobile Organism's vision, a reference to it is stored in that Mobile Organism's memory data structure.
- Herbivores can flee from carnivores.
    - If a carnivore looking for food is extremely close to an herbivore in any direction, the herbivore will flee.
    - If a carnivore looking for food is within an herbivore's vision cone, the herbivore will flee.
- Mobile Organisms need a mate of the opposite sex to reproduce. When they find a mate who also wishes to reproduce, a new mobile organism is created near the female organism after a gestation period. (Hummel)
- Plants reproduce by checking to see if there is an available space near them. If there is, a new plant is created. The duration between plant reproductions is determined by the same trait that governs a Mobile Organism's gestation period. (Herring)
- When any organism reproduces, the parent(s)' traits are randomly altered to determine the traits of the offspring. For mobile organisms, either the two parents' traits are averaged or traits are

taken entirely from the mother. The resulting traits are randomly altered to determine the offspring's traits. For plants, the single parent organism's traits are simply duplicated and then randomly altered to determine the offspring's traits. (Hummel)

- The number of organisms cannot exceed a set cap due to performance concerns. Organisms cannot reproduce when the cap has already been reached. (Herring)
- Some of the mobile organisms' traits are represented visually.
    - Carnivores have two sharp teeth, herbivores have no sharp teeth.
    - Speed: color of the body.
    - Vision Distance: Size of eyes.
- Users can create herbivores, carnivores, and plants at locations they choose in the virtual world.
    - A user can select which type of organism they want to create with the number keys or mouse wheel and click on the ground to place an organism there.
- Positions and states of all the organisms are stored on the server and can be viewed by clients
- Users are represented to other users as mechanical drones flying around the virtual space.
- Graphs display the number of plants, herbivores, and carnivores to each user
- Sound effects and custom music

Architecture or Design Space

Organisms are broken into the following class hierarchy:

- Organism
    - Mobile Organism
        - Carnivore
        - Herbivore
    - Sessile Organism
        - Plant

The Organism class:

- Genetics: a class containing all traits, represented as floats.
- Genetic algorithm: This algorithm determines how an offspring's traits are calculated from the parent(s)' traits.

  Genetic Algorithm Pseudo Code [10]:

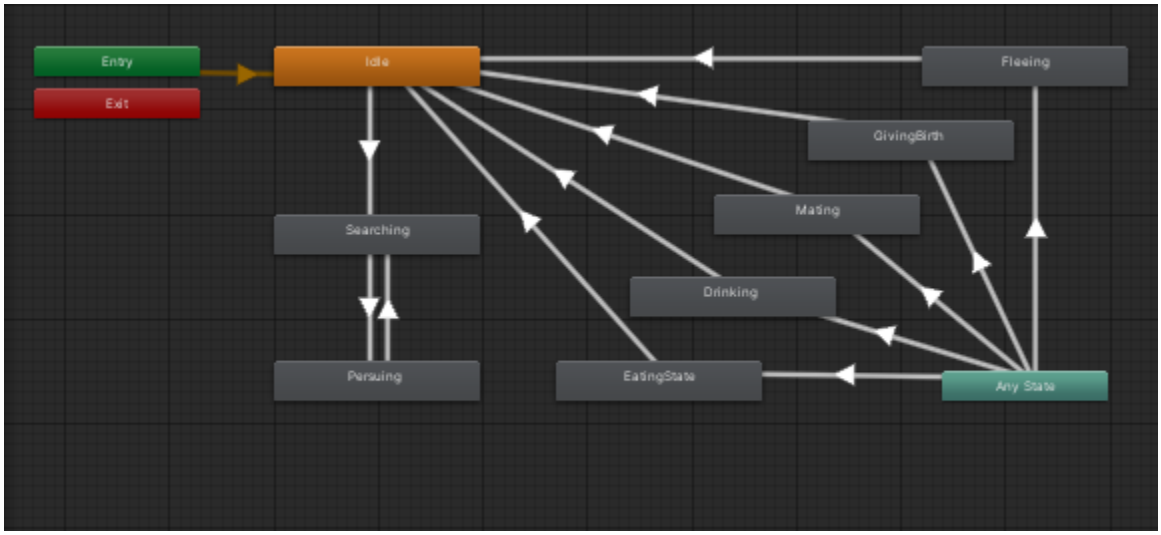
```
foreach (float gene in Genotype):
bool useFatherGenes = RandomBool;
bool mutate = Random.Range(0,
Mathf.Lerp(256, 1,
(motherGenes[Genotype.MutationChance] +
fatherGenes[Genotype.MutationChance]) / 2)) == 0);
float geneValue = useFatherGenes ?
fatherGenes[Genotype.MutationChance] :
```

```
motherGenes[Genotype.MutationChance];
if (mutate): geneValue += RandomBool ? geneValue : -geneValue;
gene = Mathf.Clamp(geneValue, 0f, 1f)
AddGeneToOffspringGenotype(gene);
```

- The method Born is called when an organism is created. This method calls the genetic algorithm and sets the organism's traits.
- Growth: The organism visually grows from a very small size to their maximum size using DOTween's DOScale method.
- Hydration and Energy Ticks: these which update hydration and energy, so that they are not updated every frame.
- Abstract Reproduce method: mobile and sessile organisms reproduce differently. They will each implement this method.
- Abstract ApplyTraits method: carnivores, herbivores, and plants visually display their traits differently. They will each implement this method.

The Mobile Organism class:

Mobile Organisms will use the following state machine.



- Pathfinding: Mobile Organisms have a NavMesh component which they utilize to find their way towards their destination. Their destination could be food, water, or a mate.
- Random Idle Movement: In the idle state, a destination near the organism is chosen randomly every few seconds.
- Memory: Mobile Organisms contain a LIFO data structure that represents their memory. Whenever they see an object that they can eat, a reference to it is entered into the data structure. When they are hungry, they pop the first item off of the data structure and try to find it. After some amount of time, determined by a genetic trait, the organism will delete the last thing in the data structure, "forgetting" it.

The Sessile Organism class:

- Reproduction: Sessile Organisms reproduce based on a timer. When the timer reaches zero, the sessile organism attempts to reproduce by doing a series of sphere casts around it to find an unoccupied space. If such a space is found, it instantiates a new plant.

Organism Population Management:

- Each type of organism has a population limit. If this limit is reached, new organisms will not be allowed to spawn in order to avoid crashing the simulation by instantiating a huge number of organisms.
- We utilize Object Pooling to further optimize performance. With this technique, as many instances as we will ever need for the ecosystem are created as soon as the scene runs. They are then activated as needed, rather than being instantiated every time an organism reproduces. This minimizes the time Unity spends on instance creation and garbage collection while the scene is running.

Networking:

We will utilize UMMO's networking code to create a database that stores the locations and states of organisms in our scene, as well as the locations of other players.
Our player object will be a modified version of UMMO's player object. We will change the mesh of the object, remove components we do not need.

4. Use Cases

Title: Manually Create Organism
Actor: User
Precondition: User has selected their desired organism type using the mouse wheel or number keys.
Trigger: User clicks the ground.
Basic flow:

1. User clicks the ground.
2. If the space they clicked is unoccupied, the organism they selected is created.

Alternative flow:

2A1. User clicks a space that is occupied.
2A2. An error notification is displayed.

Postcondition: A new organism has been placed in the environment, or the user has received an error notification.

Title: Mobile Organism Reproduction
Actors: Male mobile organism, female mobile organism
Precondition: Both organisms have enough energy to search for a mate, and have randomly decided they want to mate.
Trigger: The organisms enter each other's vision cones.
Basic flow:

1. Organisms register each other as they enter each other's vision cones.
2. Organisms move towards each other, pause for a moment, and use up some energy in the mating process.

3. The female organism begins its gestation period.
4. When the gestation period ends, a new organism is created with traits based on its parents.

Alternative flow:

4A1. The female organism dies before the gestation period ends.
4A2. The new organism it would have given birth to is never created.

Title: Plant Consumption
Actors: Herbivore, plant
Precondition: There is a plant in the scene for the herbivore to eat
Trigger: Herbivore's energy level falls below a threshold
Basic flow:

1. Herbivore's energy level falls below a threshold and it enters its Searching state.
2. The herbivore moves towards the plant it remembers seeing most recently.
3. The herbivore destroys that plant and gains energy.

Alternative flows:

3A1. The plant the herbivore remembers no longer exists.
3A2. The herbivore moves around randomly until a plant enters its vision cone.
3A3. The herbivore destroys that plant and gains energy.

3B1. The herbivore finds a plant on its way to the plant it remembers.
3B2. The herbivore destroys that plant and gains energy.

Postcondition: The herbivore has gained some energy and a plant has been destroyed.

Title: Carnivorous Consumption
Actors: Herbivore, Carnivore
Precondition: There is an herbivore and a carnivore in the scene.
Trigger: Carnivore's energy level falls below a threshold
Basic flow:

1. Carnivore's energy level falls below a threshold and it enters its Searching state.
2. Carnivore moves towards the herbivore it remembers seeing most recently.
3. The carnivore begins to chase that herbivore.
4. The carnivore catches up to the herbivore, destroys it, and gains energy.

Alternative flows:

4A1. The carnivore runs out of energy while chasing the herbivore.
4A2. The carnivore dies and the herbivore remains alive.

4B1. The herbivore is fast enough to move out of the carnivore's vision cone as it flees.
4B2. The carnivore returns to its Searching state and looks for new prey.

5. Integration Plan

Terrain Generation

The plan for integration with the Terrain Generation Project is as follows: if the terrain generation is successfully networked and tested to be working with NavMesh then we will simply merge the scripts and

required assets into our project. After merging we will do some testing to ensure everything works together, if not we will work with the Terrain Generation team to fix the issues that arise. We have communicated with the Terrain Generation group to confirm that they plan to support NavMeshes.

Universal Context Menu
In order to integrate with the Universal Context Menu, we will need to slightly modify our player interaction to disable our context menu when the user wants to bring up the Universal Context Menu on an organism. We will work with the Context Menu team to migrate and merge their scripts and assets into our Eco Sim project. Once merged successfully we will test to ensure everything is functional and works with networking. If issues arise we will work with the Context Menu team to fix issues as they arise.

6. Conclusions

Summary
We created a 3D virtual ecosystem using Unity that simulates several types of organisms undergoing the process of natural selection. This natural selection simulation consists of organisms that have a set of randomly determined traits, such as size, number of offspring, how far they can see, how good an organism's hearing is, etc. These organisms reproduce and pass on their traits, with slight alterations, to their offspring. There are several types of organisms: plants, which cannot move and reproduce asexually; herbivores, which eat plants and require another herbivore to reproduce; and carnivores, which chase after and eat herbivores and require another carnivore to reproduce. Our virtual ecosystem is networked using uMMO[5] and Mirror[11]. Players can see one another, fly around the environment freely, and spawn organisms.

Future Work
Another team picking up this project could first implement the thirst system and the visual display of traits. These were the aspects of our design we were unable to fully complete. Beyond that, there is enormous potential to expand the project by including other types of organisms. Here are a few ideas:

- Parasites that attach to mobile organisms and drain their energy. These parasites could inherit from our sessile organism class.
- Herbivores that burrow into the ground and disappear for a short time. A pursuing predator with a short memory would forget about the burrowed herbivore and wander away.
- Fruit-bearing plants that are not killed when another organism eats their fruit.
- Poisonous plants that inflict a status effect on any herbivore that consume them, causing the herbivore’s health to deplete over a short period of time. The strength of the poison could be determined by a gene. Organisms could add poisonous plants to a permanent memory that they pass on to offspring.
- Flying organisms that are difficult for non-flying predators to catch.
- Aquatic organisms.

Additionally, organism behaviors could be improved. For instance, herbivores should run away from predators in the same direction the predator is moving, instead of fleeing in a random direction.
Finally, users could be given more tools to analyse the virtual ecosystem. Currently, the only tool is a simple set of bar graphs. Line graphs that track population over time would allow for more in-depth testing of the ecosystem

Project Value

The main thing we gained from this project was an understanding of virtual world networking. We quickly realized the importance of the client serving as simply a viewer and input handler, because if the organisms displayed in the client have any active behaviors, they quickly become out of sync with the ones on the server.

Besides this, the project gave us an appreciation of the way a simulation like ours can exhibit interesting behaviors we did not explicitly design or expect. For instance, in order to prevent organisms from wandering out of bounds, we only allowed them to wander a certain distance away from the last organism in their memory. The unexpected part came when a group of herbivores wandered off as a group. They only remembered each other, so they did not wander back to the original group they came from. We realized we had accidentally allowed the organisms to form distinct herds. These herds take different evolutionary paths as organisms exclusively mate with others from their herd.

Viewing the simulation we built in action demonstrated how dynamic, evolving systems increase immersion in a virtual world. Users of a virtual world will appreciate non-player characters whose behavior and current state are determined dynamically, rather than being set in stone by designers. If gives the user the feeling that they are discovering something unique, rather than something identical to what thousands or millions of others have already found.

Milestone 2 Progress

As of Milestone 2, we have implemented a user-controlled camera, plant organisms, and herbivore organisms. Not including the networking part of our project, we are about 70% complete, because the only organism type we are missing is the carnivore. Most of the carnivore's behavior is inherited or very similar to the herbivore, so it will not take as much time to get carnivores working as it did for herbivores. With the networking component taken into account, our project is about 30% complete.

Currently, plant organisms grow and reproduce as detailed in the Requirements. They try to find an open space nearby and reproduce if they are successful. The new plant then grows to full size.

All organisms inherit a Genetics class as a member. This genetics class contains all the genes that we will be simulating. The genes are easy to modify from the Unity inspector for testing purposes, and they are modified randomly upon reproduction.

We have implemented pathfinding using a NavMesh baked into the terrain. Herbivores can successfully find their way towards their chosen goal, which can currently be food or a mate.

The herbivores can reproduce and produce a new herbivore which is smaller in size. It will then grow to its maximum size.

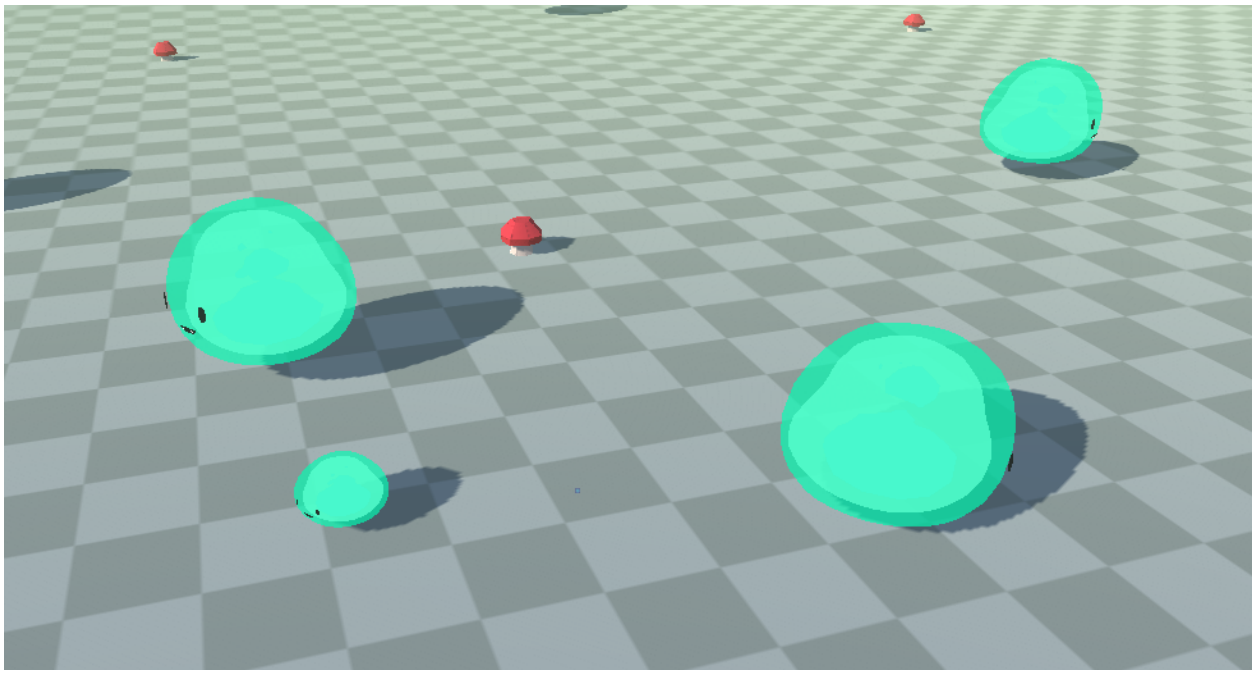


Figure 1. These two larger herbivores in the front of the scene have just reproduced. The smaller herbivore is their offspring.

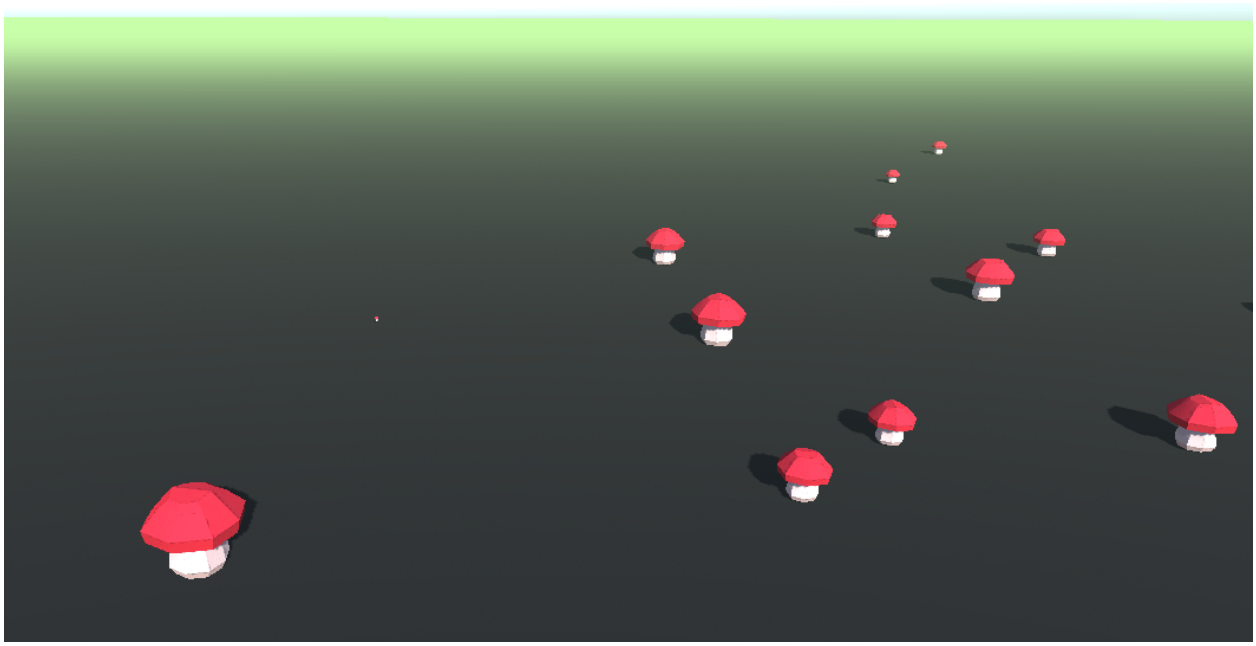


Figure 2. If there is an unoccupied space near them, these plants can reproduce. A tiny newly-created plant is barely visible in the middle-left portion of the image.

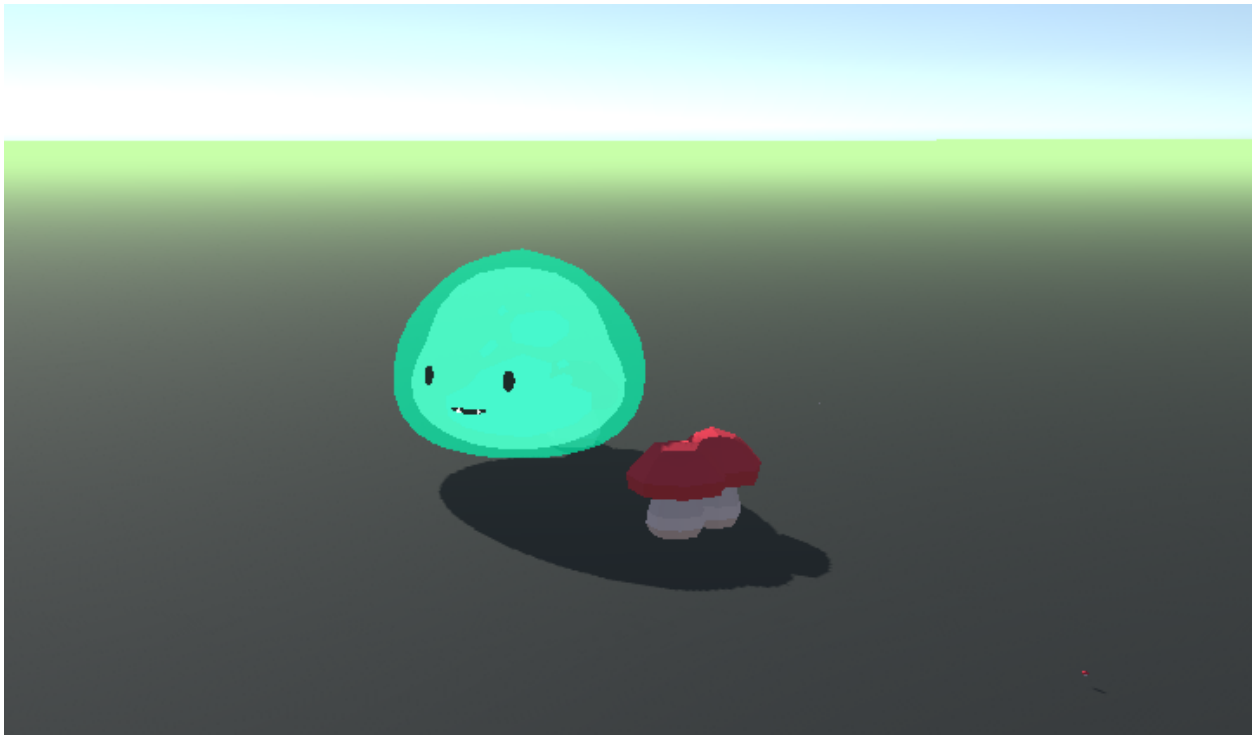


Figure 3. This herbivore just consumed a plant and gained energy.

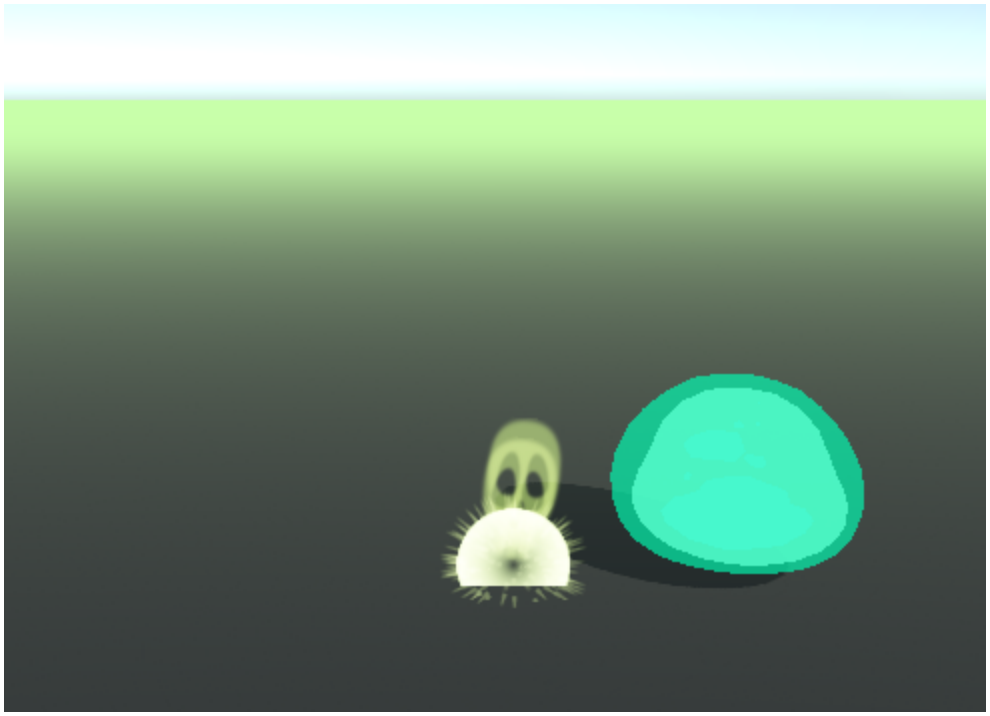


Figure 4. These particles indicate that an herbivore has just run out of energy and died because it was not able to find a plant to eat. The herbivore on the right is still alive.

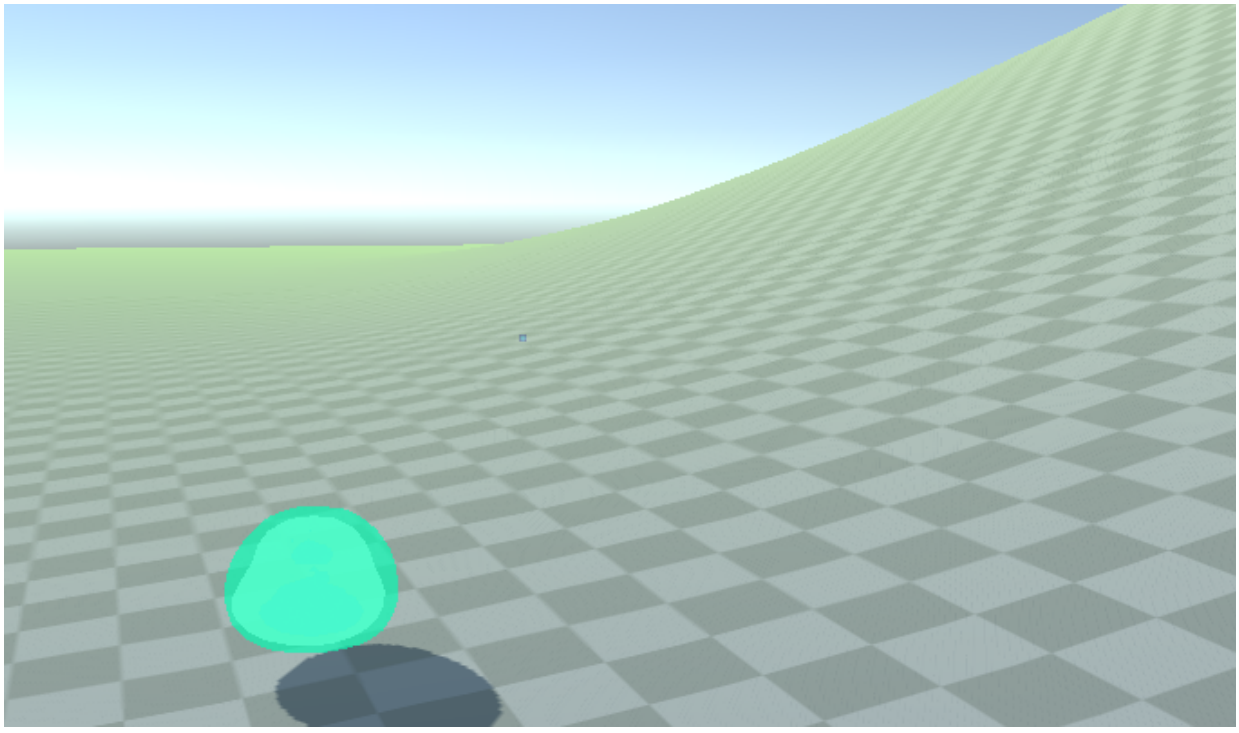


Figure 5. The herbivores can use a NavMesh to move up and down terrain like this hill.

Milestone 3 Progress

As of Milestone 3, we have implemented player networking, networked herbivore organisms, and redesigned all of the UMMO UI to fit into our Sci-Fi theme. Now that players are networked, the project is significantly closer to completion. The remaining tasks are networking sessile organisms, syncing genetics across all clients, and creating carnivorous organisms. These tasks will not take much time, as networking sessile organisms will mostly make use of existing logic, and carnivorous organisms inherit most of their behaviors from the existing organism class. The only change we must make to carnivorous organisms is the type of organism they search for and consume and syncing genetics is making use of functionality already provided by uMMO.

Currently, plant organisms can grow and reproduce as detailed in the Requirements. They try to find an open space nearby and reproduce if they are successful. The new plant then grows to full size. As stated above, plants are not yet networked but this will not take long to implement because we can make use of existing logic.

All organisms inherit a Genetics class as a member. This genetics class contains all the genes that we will be simulating and they are modified according to our genetic algorithm upon reproduction. Genes are not yet replicating across clients.

We have implemented pathfinding using a NavMesh baked into the terrain. Herbivores can successfully find their way towards their chosen goal, which can currently be food or a mate. Herbivores are fully networked.

The herbivores can reproduce and create a new herbivore which is smaller in size and has genetics determined by our genetic algorithm. It will then grow to its maximum size over a period of time determined by its genetics. Herbivore reproduction is fully networked.
We have player interactions now, they can spawn organisms, inspect an organism's traits, and communicate with one another via a chat window. The first players will be spawned into an empty world and will be charged with repopulating it using these interactions. We have a menu that shows a population graph implemented, however it only works locally and needs to be synced with the server still. Players themselves are completely networked and integrated with uMMOs player class.

Due to our huge leap in progress this project is more than 70% complete. The only things left are syncing genetics over the network, duplicating and modifying the herbivore class to become a carnivore, networking plants, and integrating the Terrain Generation and Universal Context Menu Projects into our simulation.

Figure 6. The login menu offers the option to host a server. A server must be running in order for clients to connect.

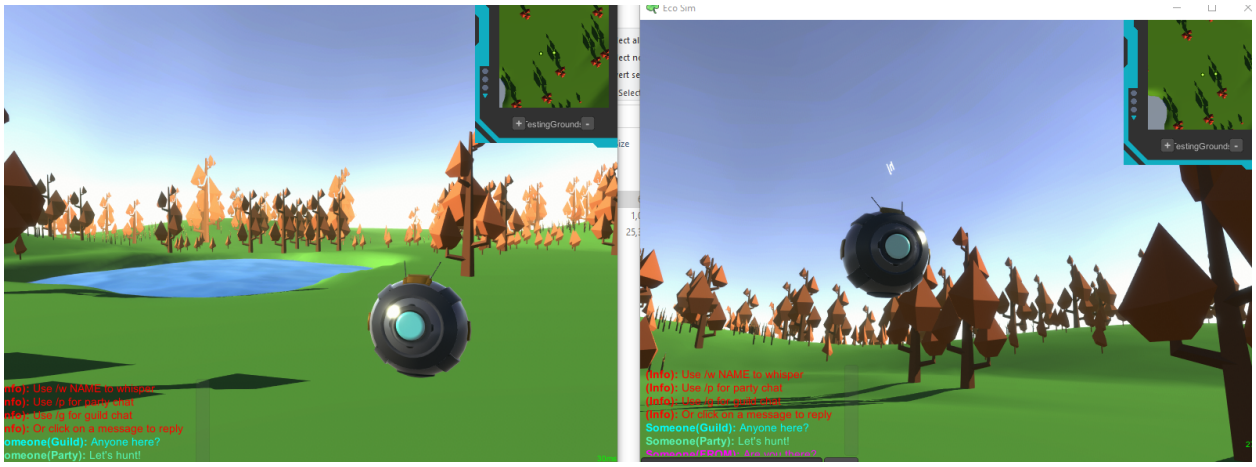


Figure 7. Multiple clients can connect to the server and see each other moving around in the world.

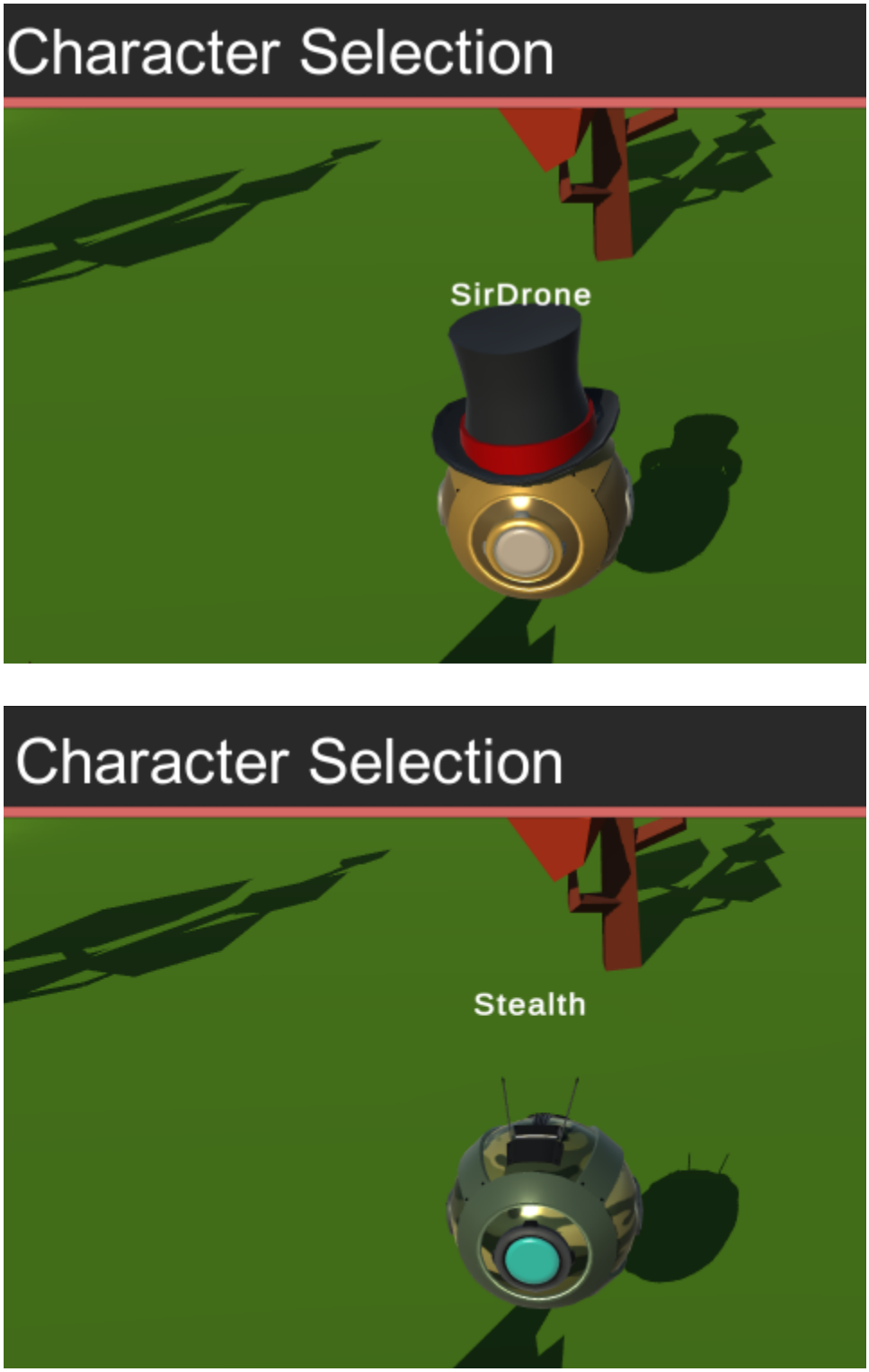


Figure 8. In order to distinguish between users, we created more drone assets. It is now possible to pick from 3 different drone types.

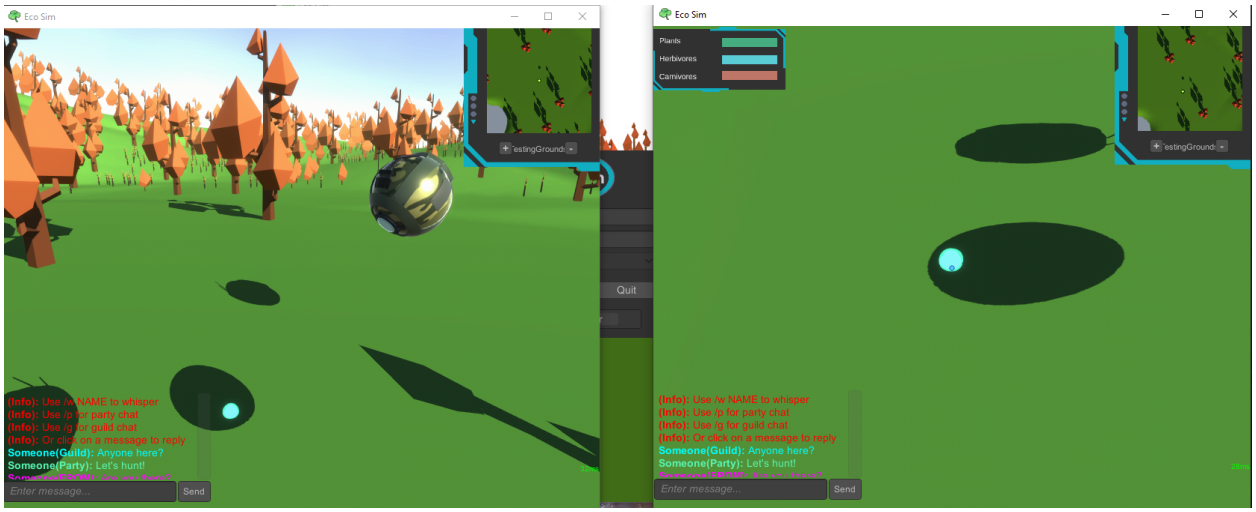


Figure 9. The client on the right has created an organism. The client on the left can see the new organism and the other client's drone avatar.

Figure 10. The client on the left can inspect the organism that the other client created.

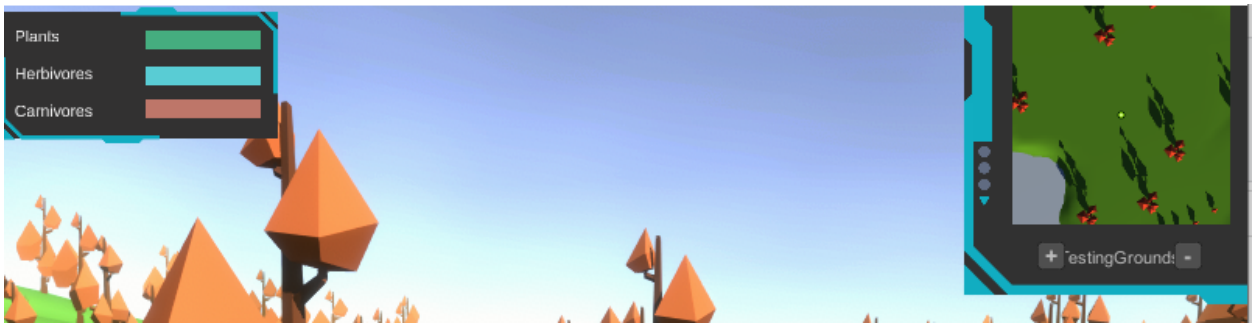


Figure 11. We have modified and extended the UMMORPG UI with custom sci-fi themed UI elements. In the future, the graphs on the upper left will display the population counts of each type of organism.

Final Milestone Progress

We have implemented about 99% of our intended features, we have complex organism behaviors, different types of organisms, networking is fully functional, players can spawn organisms and observe their behaviors, we have a networked graph that tracks the population, and we have integrated the Universal Context Menu into our project. What we added from the last milestone is predators, organisms now have hearing, a population bar graph and we have integrated the universal context menu. We didn't accomplish organisms having hydration and being able to find and drink water and we didn't get to implement all the planned visual traits for organisms. The bonus features we got working were organisms being able to hear other organisms, they have a hearing trait in their genetics and will use their hearing to either hunt down prey, if the organism is a predator, or run away from threats, if the organism is prey and we also implemented audio, so there is music, ambient noises, and interaction noises when a player spawns an organism.

Figure 12. The server host's view. The server is where the organism behavior logic happens, while the client simply displays their position.

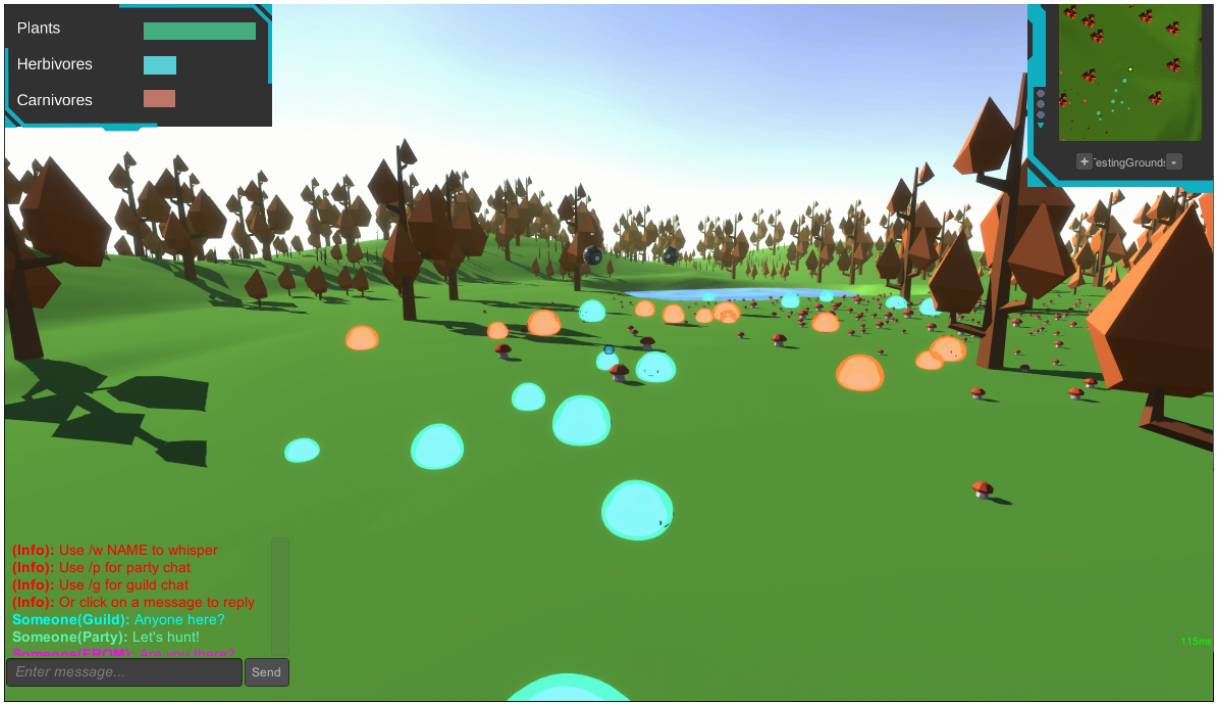


Figure 13. A large group of organisms. Some were spawned manually, while many others were born via our reproduction system. The graphs display the population size for each organism type.

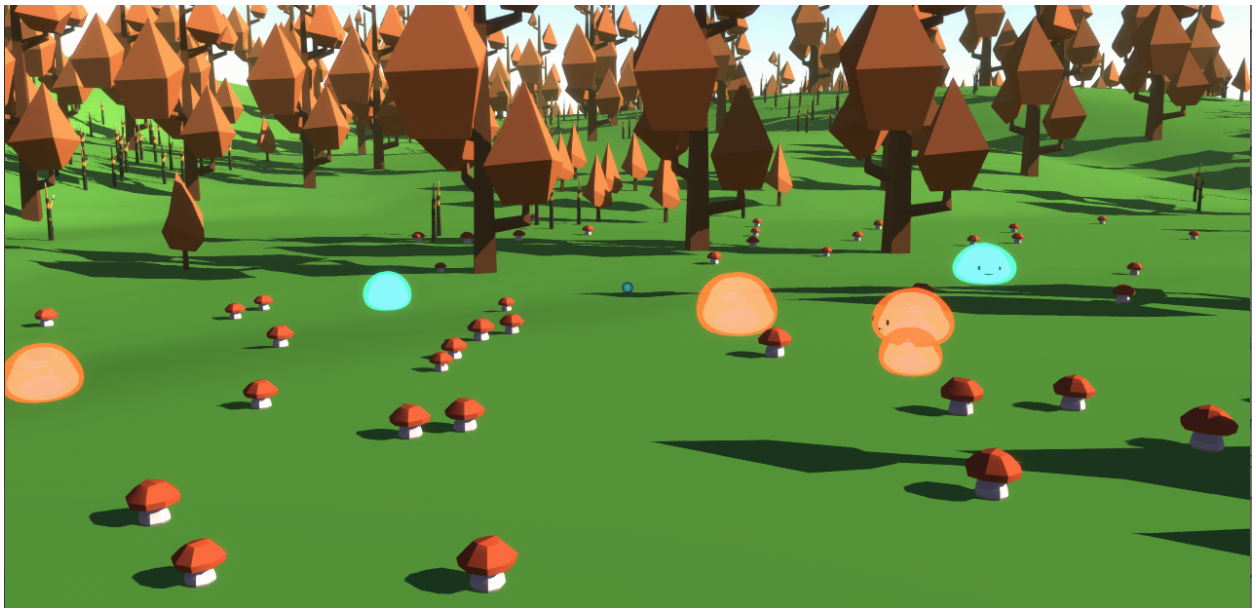


Figure 14. Orange carnivores and blue herbivores move around the environment. In the background, plants reproduce on the slope of a hill and their offspring align to the terrain.

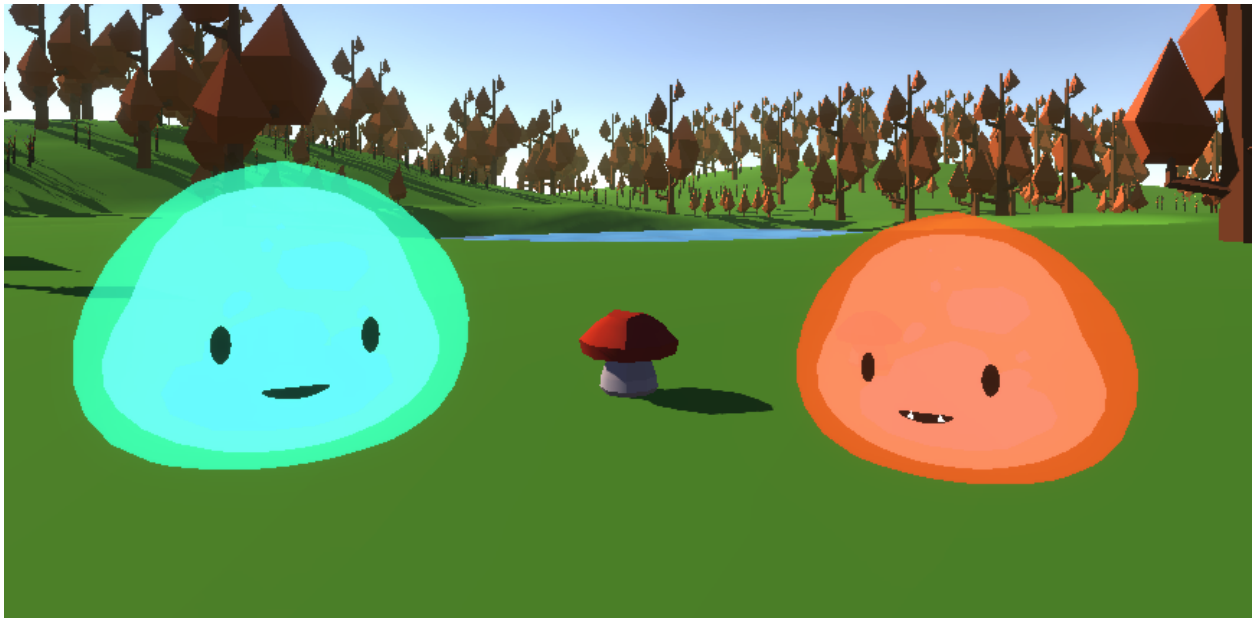


Figure 15. Left to right: herbivore, plant, carnivore. Beides its color, the carnivore is distinguished by its sharp fangs.

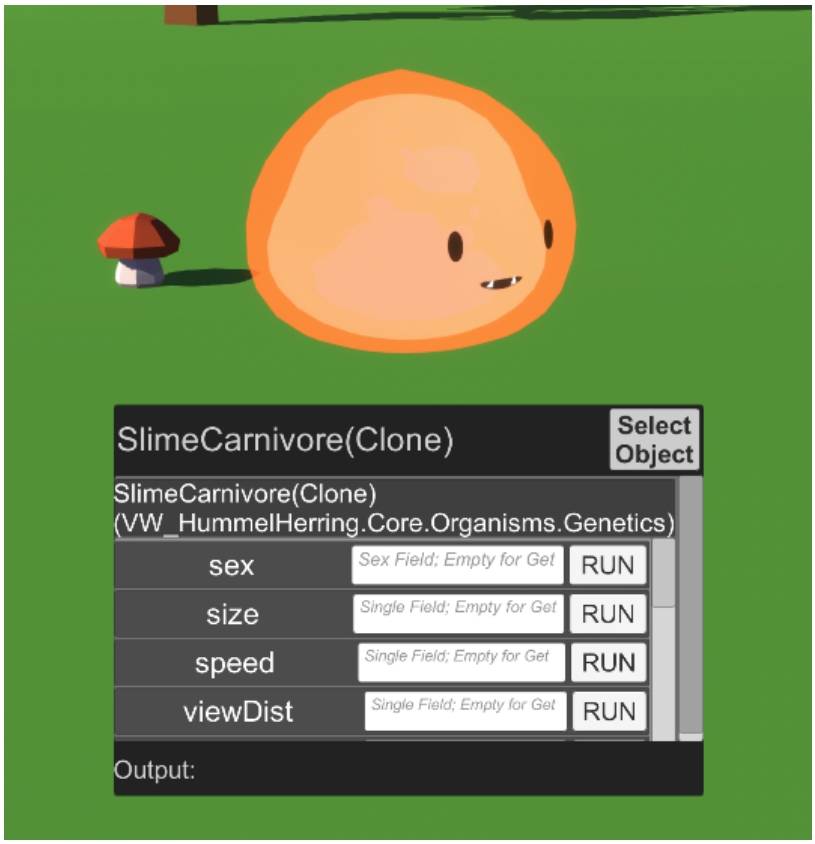


Figure 16. The universal context menu can be used to view organisms' properties.

Bios

- **Jeremy Hummel** – Hummel is a senior student in the Computer Science Department at Western Washington University. He has completed relevant courses and is expecting to graduate with a BS degree in Spring 2021. He was responsible for building the genetic algorithm, creating mobile organism behaviors, modeling, texturing and animating player and slime models, and integrating UMMO's networking framework into the project.
- **Walker Herring**– Herring is a senior, masters student in the Computer Science Department at Western Washington University. He has completed relevant courses and is expecting to graduate with a BS degree in Spring 2021. He was responsible for plant behaviors, user interaction, and integrating UMMO's networking framework into the project.
- **Dr. Wesley Deneke, Mentor** – Deneke is a professor in the Computer Science and Department. He leads student research projects that are currently focusing on how to simulate Human Workflows using 3D virtual worlds.

References

[1] Blender, https://www.blender.org/
[2] Unity Navmesh, https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html
[3] DOTween, http://dotween.demigiant.com/
[4] Sensor Toolkit,
https://assetstore.unity.com/packages/tools/ai/sensor-toolkit-88036?aid=1100l355n&gclid=CjwKCAiAjeSABhAPEiwAqfxURfgrYqBc3BN7BfjJIFNZb4MblZ2kq0Gf12GrVuekC6HLm3LzIlSfkxoCEtwQAvD_BwE&pubref=UnityAssets%2ADyn09%2A1723478829%2A67594162255%2A336275757769%2Ag%2A%2A%2Ab%2Ac%2Agclid%3DCjwKCAiAjeSABhAPEiwAqfxURfgrYqBc3BN7BfjJIFNZb4MblZ2kq0Gf12GrVuekC6HLm3LzIlSfkxoCEtwQAvD_BwE&utm_source=aff
[5] uMMO, https://assetstore.unity.com/packages/tools/game-toolkits/ummo-13867
[6] Doozy UI,
https://assetstore.unity.com/packages/tools/gui/doozyui-complete-ui-management-system-138361
[7] MyBox, https://github.com/Deadcows/MyBox
[8] Simplistic Low Poly Nature
https://assetstore.unity.com/packages/3d/environments/simplistic-low-poly-nature-93894
[9] Cartoon FX Remaster
https://assetstore.unity.com/packages/vfx/particles/cartoon-fx-remaster-4010
[10] ClodAlejandro's Genetic Algorithm
https://github.com/ChlodAlejandro/eco/blob/master/Assets/Scripts/Creature/GeneSet.cs
[11] Mirror Networking
https://mirror-networking.gitbook.io/docs/?q=