

Hunter of Monsters

Instructor: Oliver van Kaick

Gabriel Martell, 101191857

Kelvin Jeon, 101087383

Alex Davidson, 101149335

Nicholas Veselskiy, 101192011

○ Theme and Setting:

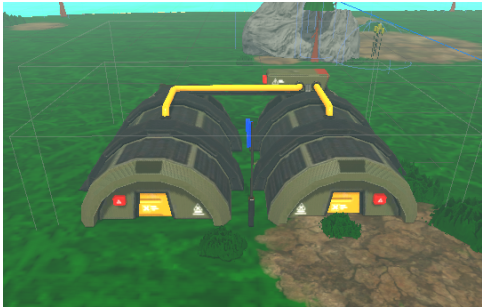


- To reiterate from our first proposal, the game will feature a setting similar to that of Monster Hunter where the world is populated with “monsters” and many of the technological elements are akin to an era where modern technology has not completely overtaken everything. Firearms *exist*, but the use of melee weapons is still prominent. In addition, the wildlife is abundant and nature is blooming. As for how the game is played, the player is expected to accrue resources and build up enough forces to tackle the oncoming wave of “monsters”.



○ Controls:

Left Click + Drag	Select Units
Right Click (After Unit Selection)	Default Action, if no target then move to the given position - If target is a resource then GRU units will harvest - If target is friendly then repair/heal - If target is an enemy then attack - Right-click adds patrol points if patrol mode is toggled on
Right Click (Building Selection)	Cancel build selection
B, C, T	One Key Click: - Highlight of building blueprint, determine placement Second Key Click: - Confirm placement, and order GRU units to build <i>Base</i> , <i>Camp</i> and <i>Trap</i> respective to the key pressed.
H, G	Produce Hunter or GRU units respectively.
P (After Unit Selection)	One Key Click; Toggle On: - See Right Click Controls Second Key Click; Toggle Off: - If patrol points are appended to a unit, have the unit patrol those selected points
ESC	Exit the game.

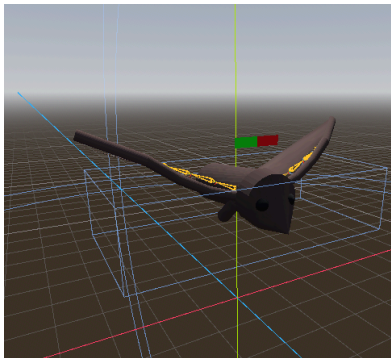
○ Game Objects and Units:

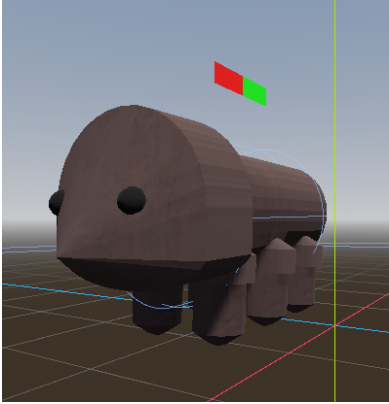
Player Units:

Type and Name	Definition	Picture
(Static) Base	Allows the player to build Grus.	
(Static) Traps	Allows the player to slow down/stun enemy units	
(Static) Training Camp	Allows the player to build military units.	

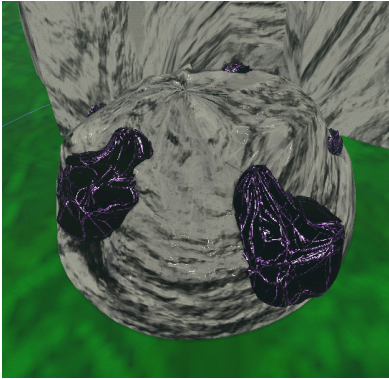
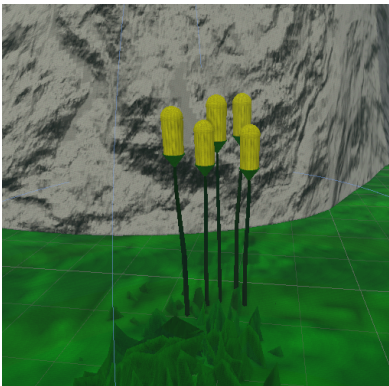
(Dynamic) Hunter	A military unit that can attack enemy units in range. (Humanoid)	
(Dynamic) GRU	Builder/Gatherer unit that can build buildings and gather resources.	

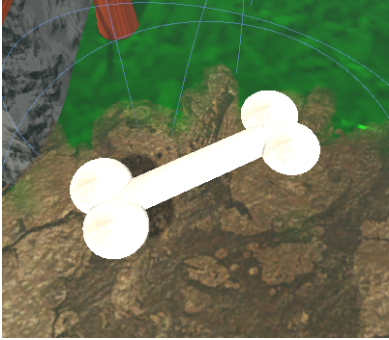
Enemy Units:

Type and Name	Definition	Picture
(Dynamic) Flying Enemy	A flying enemy monster unit that purely focuses on buildings.	

(Dynamic) Ground Enemy	A ground enemy monster unit that focuses on both fighting and destroying buildings.	
------------------------	---	---

Environment:

Type and Name	Definition	Picture
(Static) Zeny Ore	A basic resource; infinite.	
(Static) Food	A basic resource; infinite.	

(Static) Bones	A basic resource; infinite.	
----------------	-----------------------------	---

○ Unit Actions:

Type	Contains
Dynamic Player (All)	“idle”, “move”, “repair”, “patrol”, “attack”.
Dynamic Player (GRU)	“gather”, “build”
Static Player (Base and Camp)	“create” GRU, “train” Hunter, “destroy” itself
Static Player (Trap)	“destroy” itself, “damage”
Dynamic Enemy (All)	“idle”, “move”, “attack”

○ Game Architecture:

The game architecture has had many changes, as well as some aspects that stayed true from the initial proposal. To reiterate all functionality,

- For all dynamic units, player and enemy, derived from the dynamic unit parent class, “*BaseUnit.gd*” has multiple operation states. This includes “AttackMode”, “MoveMode”, “RepairMode”, “PatrolMode”, “Other” and “None”. In addition to this, dynamic units keep track of their respective faction, as well as a death signal. Some attributes this parent class contains are health, speed, attack power, acceleration, interaction speed and rotation speed. This class also keeps track of whether the current unit is moving, attacking, whether it's selected by the user as well as the set patrol points.
- For all static units, the parent class “*BuildingGeneral.gd*” has no operation states - it contains its health, faction, and unit creation queue. The child scripts, such as “*Base.gd*” and “*TrainingCamp.gd*”, focus on unit generation - keeping track of its creation cost and time-to-create.

- For harvestable resources, the “*Gru.gd*” script recognizes the resource it is currently harvesting, which is then timed and relayed to the “*Player.gd*” which contains the resource information.
- For input, **all** input is checked for in the “*CameraScript.gd*” process method - therefore treating this script as the input manager discussed in the initial proposal.

○ **Prototype Milestones vs Final Milestones:**

Animation: The final milestone of animation was achieved by creating each model and its respective armature in Blender. In Blender, different animations for different actions were created appropriately for each unit - and then exported into the product. Aside from polishing general animations, there is nothing that is missing from the initial milestone goals. There were no animations in the prototype as functionality was prioritized over this.

Pathfinding: Pathfinding was used using GODOT’s in-house Navigation3D agent and region baking. There haven’t been any major differences in this milestone from the initial prototype to this final deliverable - other than some general fixes. The region is re-baked every time a new building is made, and the region is already confined on the island by an invisible barrier blocking any further pathing outside of the intended area.

Semi-Automatic Actions: Units only need to be directed to perform a task once and then they will perform their given task without any intervention from the player. If a unit is directed to attack an enemy they will move to the target and attack them without any more player direction. The same applies to directing a unit to harvest or build. The actions from the prototype milestone to the final deliverable did not receive any major changes. Aside from building now including another key click to confirm, the rest were minor bug fixes and polishes.

Enemy AI: Our game uses a rule-based AI which uses the following sets of rules: First, if there are no friendly units nearby, move towards the center of the map. If there are friendly units nearby, attack them. The enemy AI will prioritize attacking friendly buildings as opposed to friendly units. The enemy AI during the prototype milestone, although attacked, would always traverse to some provided position (say, $\text{vec3}(0,0,0)$) if no other enemy faction-related unit was in the aggro range. For the final deliverable, the enemies instead know where your buildings are and will go towards them for demolition.

Meaningful Play: Players are tasked with surviving a set number of enemy waves with increasing difficulty. To accomplish this task the player is required to gather resources and build units to defend against the increasing waves of enemies. For this final deliverable, there are more waves for the player to defend against - whereas the prototype had three waves only. In addition, the screen presents a victory screen once these waves survive, or if all buildings and units are dead, a game over screen is portrayed instead.

Feedback to Player: The UI is clear and straight to the point. Acknowledgement of health on units, selection box when LClick + Dragging and units selected icon, and lastly, resource count on the HUD with simple and effective visuals. As for the prototype version and the final

deliverable, there wasn't much change other than fixing some casted shadows. In addition, while the prototype milestone also had this UI, the final milestone added more polish to keep the style consistent.

○ Additional Implementations:

Building Feedback: When deciding to build, you are prompted with the building blueprint which gives a better visual of where your constructed building will be. When in collision with terrain or another building, a red-spherical indicator will notify the player that the building cannot be constructed there.

Creation of New Units: Units can be created by acquiring resources throughout the map. To create a unit the player must have either a camp or base constructed depending on the type of unit they would like to construct.

Additional Units with Animations: In addition to the one humanoid unit (the GRU) which uses animations, we also gave skeletal animations to both of our enemy units, as well as some unique animations to the player-controlled hunter unit.

Additional Screens Based on Game State: The game has two dynamic screens, one game-over screen for when all player units and buildings are destroyed, as well as a win screen for if the player can survive all seven waves.

○ Project Sources:

Shaders:

[Material Shader](#)

[Water Shader](#)

Textures:

[Bone Texture](#)

[Dirt Patch](#)

Models:

[Tesla Coil](#)

[Base](#)

[Barracks](#)

All .obj files of terrain and resources on the map scene were made by Gabriel Martell

Normal Maps:

[Leaves](#)

[Gem](#)

[Bark](#)

[Grass](#)

[Rock](#)

[Corn](#)

Map Normal (Gabriel Martell)