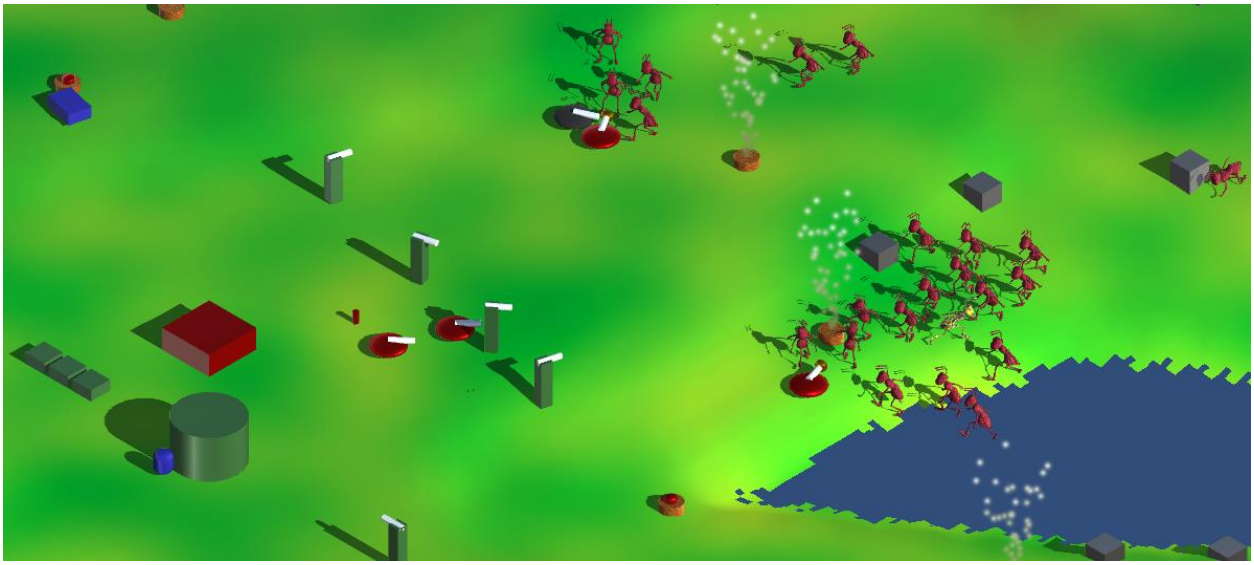
BUGMAN ASSAULT

Design Documentation

Griffin Shea



Game Concept

Bugman Assault is a real-time strategy game where the player takes on the role of the Field Marshal of the EUBDC (Earth United Bug Defence Coalition).

In the year 2222, the vicious Bugmen of Bugulon Prime landed on Earth and began slaughtering human populations. The EUBDC was assembled in order to organize a counteroffensive to these invading forces. The war has been long and brutal but the EUBDC has finally pushed the Bugmen across the stars all the way back to their home planet. All that remains to be done is to eliminate the hive on Bugulon Prime so the EUBDC has ordered their best General to organize the final assault.

The player therefore must command EUBDC forces to harvest resources on Bugulon Prime (including metal and gas), build factories, barracks, fortifications, manufacture tanks, and finally lead an attack to eliminate the enemy hive. However, the Bugmen are strong and numerous – if the player lets down their guard for too long and lets their stockpile become destroyed, the mission will be failed…

Introduction

This report will begin by enumerating how this project met technical requirements from the different milestones. Then, I will enumerate the main game objects in Bugman Assault and describe each including a list of components and CRI cards. Finally, there will be some finer

detail on different interactions between classes in the game. As an appendix, I have appended the READ_ME.txt file included with my source code submission.

Technical Requirements

First prototype – 3D rendering:

1. Game is rendered in an isometric view of an illuminated 3D world including shadows.
2. Each game object uses distinct shaded 3D model, though most of these models are simple shapes, and I have reused the same few materials across different objects.
    a. The World object uses a 3D model generated from a png image file
    b. The World model and the Bugmen skin materials utilize a Perlin noise shader.
3. The game objects are equipped with Unity Collider components and Rigidbody components to handle collision detection and response.
4. Use DEMO_MODE to spawn every type of game object at game start.
5. Player unit and building interactions are fully implemented.
6. User can scroll across the world using arrow keys or W, A, S, and D. User can zoom in and out using Z, and X.

Second prototype – animation and physics:

1. (REMOVED FROM FINAL SUBMISSION) Hermite splines with ease-in and ease-out are used to move units.
2. Bugmen Warrior and Scout objects feature skeletal animation of Cartoon Ant model shared by user yogi_484 on free3d.com (https://free3d.com/3d-model/cartoon-ant-43724.html) using Unity Animation Controller activated by scripts. Seven animations procured from Mixamo are featured in total.
3. Physics are controlled by Unity's built-in physics system, Rigidbody, and Collider components. Steering forces are applied to player and enemy units by scripts, the NavMeshAgent component also applies forces to move units along calculated paths.
4. Obstacle avoidance is provided by NavMeshObstacle components.

Third prototype – A.I.:

1. Pathfinding of player and enemy units is provided by Unity's NavMesh system, using NavMeshAgent and NavMeshObstacle components.
2. All player units incorporate some form of semi-automatic actions.
    a. When the player selects a Collector and targets a MetalSupply, the Collector moves to the MetalSupply, picks up resources, then moves back towards the Stockpile to deliver the resources, and repeats this series of actions until the MetalSupply has been depleted.

    b. Builders can queue build orders so that once the current build order is completed, the Builder will automatically move towards the next build location and begin that task.
    c. The player can target enemies with a selected Tank and the Tank will path find towards its target, then chase it and attack it until it is destroyed. Tanks will also automatically begin to chase and shoot at enemies which come close enough. The Tank uses a finite state machine for its A.I. Turrets include the same capacities, though the logic is simplified, as they are immobile.
3. There are three components to enemy A.I.:
    a. Scouts use a finite state machine with three states: dead, explore, and report. They begin in the explore stage where they wander randomly steering until they encounter a player's building or unit, then they turn on their report state and run towards the Hive. Once health reaches 0, Scout goes to a dead state where it does nothing.
    b. Warriors use a finite state machine with five states: dead, wander, idle, charge, and attack. Warriors spawn from the Hive in the wander state, and randomly steer crawling for a randomized amount of time, after which they go to an idle state, where they do nothing. If a player's building or unit comes within a Warrior's aggression range, it will move into an attack state, and chase the building/unit, dealing damage once it has come near enough. The Hive may activate the Warrior's charge state if it is idle: in the charge state the Warrior will run towards the destination given by the Hive. Once health reaches 0, Warrior goes to a dead state where it does nothing.
    c. The Hive A.I. spawns units every five seconds, and over the course of five minutes, these spawned enemies will increasingly be oriented towards the player's stockpile (at game start, they are spawned in completely random directions). Every fifth Bugman spawned is a Scout, the rest are Warriors. Once the Hive receives a report of a player's unit or building location from a Scout, it will broadcast a signal to all idle Warriors to charge that location.
4. Meaningful play is implemented in the game by disabling the DEMO_MODE boolean on the World object. The player spawns with a single Barracks and a single Builder in addition to their Stockpile, which begins with set amounts of consumable resources. All game mechanics are implemented. Please follow the walkthrough and guidelines in the README.txt included with the source code to successfully complete the game. The game ends when either the Stockpile or the Hive health reaches 0: running health totals are displayed in the top right of the screen.
5. The player receives feedback for actions in a few ways:
    a. Top left of screen shows player's current resource counts.

b. Bottom right shows selected unit type and current health.
c. Bottom left shows selected unit controls for orders.
d. MetalSupplies shrink in size as they become depleted.
e. GasSupplies stop spewing gas once a harvester is build on top of them.
f. Builder orders show green “scaffolds” for future orders, and a blue “scaffold” for the current order which it is building.
g. Units and buildings that are destroyed change material.
h. Bugmen which have died go into a death animation and then lie still before disappearing.
i. Player can see where is being targeted with a red sphere cursor that moves along the map with the mouse.

Additional Features:

1. Factories can be used by the player to create new units, the Hive creates new enemy units on a timer. The Builder can be used by the player to create new buildings.

Game Objects

Buildings:

All buildings contain in addition to Transform, MeshFilter, MeshRenderer, and Collider components:

- A Health component to count health and handle what happens when a building is destroyed.
- A NavMeshObstacle component so that units can pathfind their way around these buildings on the NavMesh.

Stockpile:

The Stockpile is the player’s main base of operations which holds resources and without which, humanity will be unable launch their final assault on the Bugmen. When the Stockpile’s health reaches 0, the game will be lost. The Stockpile keeps track of the player’s current count of metal, gas, as well as total and used manpower. The player can select the Stockpile and press DELETE to surrender. It looks like a big green cylinder.

Factory:

The Factory is where the player can produce new units to collect, build, and fight. The player can select the Factory with the left mouse and then use 1, 2, and 3 to add new orders to create Builders, Collectors, and Tanks. The orders are added to a queue only if enough metal, gas, and manpower were successfully procured from the stockpile, and a build timer is used to stagger production of units. X can be used to pop build orders from the end of the queue to refund

spent resources. Right click will move the Factory's "flag," which is the location produced units will move to once they leave the production line. The Factory looks like a big red box.

Turret:

Turrets are an essential tool for the player to ensure success in wiping out the Bugmen Hive because early on they will not have sufficient access to gas and manpower in order to train enough tanks to defend their buildings and units form assaults. Turrets automatically attack bugmen that come within their range, but the player can also manually target them using right click while a Turret is selected. Turrets look like tall skinny boxes with a turret on top.

Barracks:

Barracks are needed for the player to increase the number of units they can maintain through manpower. Each Barracks building increases the player's max manpower by ten, and each unit the player produces will increase the player's used manpower, which can never exceed the max. Barracks look like small green cubes.

Units:

All units have in addition to Transform, MeshFilter, MeshRenderer, and Collider components:

- A Health component to count health and handle what happens when a building is destroyed.
- A Rigidbody component to handle collision resolution and enforce constraints.
- A NavMeshAgent component used to move the unit along calculated paths in Unity's NavMesh system.
- An Orderable component and a MoveOrder component, which are used to implement the player's ability to select, control, and set move orders with right click for units.

Collector:

Collectors are used to collect metal from MetalSupplies dotted randomly across the world map. A player can select a Collector and use right click to select a MetalSupply and the Collector will automatically move back and forth carrying metal to the Stockpile. Collectors are the little blue capsules.

Builder:

Builders are used by the player to construct buildings: Factories, Turrets, and Barracks. Each building costs a certain amount of metal which is deducted from the stockpile when the player adds a build order using 8 (barracks), 9 (turret), or 0 (factory) plus the right mouse button on the terrain. Additionally, builders can build gas harvesters on naked GasSupply spouts dotted across the map using right click. Builders look like a blue box.

Tank:

Tanks are the essential offensive unit the player needs to use to destroy the enemy hive. They shoot at enemies which come within their range and can be manually targeted by the player to pursue a specific enemy. Tank A.I. logic utilizes a finite state machine with four states: dead,

idle, chasing, and pursuing. In the dead state, the Tank does nothing. In the Idle state, the Tank stays alert for enemies to enter its aggression range, and if this happens, it will move into the chasing state. In the chasing state, the Tank will steer towards the currently targeted enemy and shoot at it when it is within attack range. If the player manually selects a target for the Tank using right click, it will enter pursue state and use NavMesh navigation to move towards the enemy unit until it is within aggression range, where it changes into chasing state. The Tank looks like a squashed red sphere with a turret on top.

Enemies:

All units have in addition to Transform, MeshFilter, MeshRenderer, and Collider components:

- A Health component to count health and handle what happens when a building is destroyed.
- A Targetable component to allow the player to right click it.

Hive:

The Hive is the Bugmen's base of operations and its destruction is the player's prime objective. Once its health reaches zero, all living Bugmen will be die and the game will be won. The Hive spawns a new enemy Bugman every five seconds and every fifth one is a Scout, the rest are Warriors. If a Scout returns to the Hive with a player's building or unit location, the Hive will broadcast this location to all idle Warriors and initiate a charge.

Scout:

The Scout's job is to explore the world map and search for the location of player's units or buildings. Once a location is discovered the Scout will return to the Hive to report it. The Scout contains the following additional components:

- A Rigidbody component to handle collision resolution and enforce constraints.
- A NavMeshAgent component used to move the unit along calculated paths in Unity's NavMesh system.
- An Animator component to animate the Bugman model

Warrior:

The Warrior's job is to attack the player's buildings and units, defending the hive. Warriors use a finite state machine with five states: dead, wander, idle, charge, and attack. Warriors spawn from the Hive in the wander state, and randomly steer crawling for a randomized amount of time, after which they go to an idle state, where they do nothing. If a player's building or unit comes within a Warrior's aggression range, it will move into an attack state, and chase the building/unit, dealing damage once it has come near enough. The Hive may activate the Warrior's charge state if it is idle: in the charge state the Warrior will run towards the destination given by the Hive. Once health reaches 0, Warrior goes to a dead state where it does nothing. The Warrior contains the following additional components:

- A Rigidbody component to handle collision resolution and enforce constraints.

- A NavMeshAgent component used to move the unit along calculated paths in Unity's NavMesh system.
- An Animator component to animate the Bugman model

CRI (Class-Responsibilities-Interactions) Cards

(I have modified the CRC format to replace *Collaborators* with *Interactions* in order to also show what other objects are involved and how so. "**Class** → method()" means that the method is activated by **Class**. "method() → **Class"** means that the method makes a call to **Class**. This will make more sense when you see the interaction chains section below.)

Notes:

- General stands for the player's UI interactions which activate unit and building abilities.
- When a class's method is not activated by another class, it is activated by logic in that class's update function.

CLASS(TYPE)	
RESPONSIBILITIES	INTERACTIONS

World	
pathGraph objects[]	**[Factory, Hive, Builder]** → addObject() **[Building, Unit, Enemy]** → removeObject() **[Unit, Enemy]** → planPath() **Collector** → removeResource()

Stockpile(Building)	
health energy metal manpower	**Warrior** → takeDamage() → **World** **[Factory, Builder]** → removeResource() **[Factory, Collector, Builder]** → addResource() **General** → surrender() → **World**

Factory(Building)	
health buildTimer buildQueue[]	**Warrior** → takeDamage() → **World** createUnit() → **World** **General** → addBuildOrder() → **Stockpile** **General** → cancelBuildOrder() → **Stockpile**

Turret(Building)	
health range power	**Warrior** → takeDamage() → **World** shootTarget() → **Enemy** **General** → setTarget()

target	

Collector(Unit)	
health heldResource destination	**Warrior** → takeDamage() → **World** pickupResource() → **World** deliverResource() → **Stockpile** **General** → setDestination() → **World**

Builder(Unit)	
health orderTimer orderQueue[]	**Enemy** → takeDamage() → **World** constructBuilding() → **World** **General** → addOrder() → **[Stockpile, World]** **General** → cancelOrder() → **Stockpile** **General** → setDestination() → **World**

Tank(Unit)	
health range power destination	**Warrior** → takeDamage() → **World** shootTarget() → **Enemy** **General** → setDestination() → **World**

Hive(Enemy)	
health buildTimer warriors[] scouts[]	**[Turret, Tank]** → takeDamage() → **World** **Scout** → orderAssault() → **Warrior** createUnit() → **World**

Scout(Enemy)	
health hive destination target	**[Turret, Tank]** → takeDamage() → **World** explore() → **World** reportTarget() → **Hive**

Warrior(Enemy)	
health range power destination	**[Turret, Tank]** → takeDamage() → **World** **Hive** → setDestination() bite() → **[Unit, Building]**

Interaction Chains

General → Stockpile.surrender() → World.removeObject()
If the general selects the stockpile and selects the surrender option, the stockpile will be removed from the world as if it were destroyed and the game will end.

General → Factory.addBuildOrder() → Stockpile.removeResource()
General → Factory.cancelBuildOrder() → Stockpile.addResource()
The general can select a factory and use the add and cancel build order options to add orders to the build queue. Each new build order will remove resources from the stockpile which can be added back if the order is cancelled. Stockpile.removeResource() should return a boolean to tell the factory whether there were enough resources to add the order.

Factory.createUnit() → World.addObject()
Hive.createUnit() → World.addObject()
When the build timer on a factory reaches zero, the top order on the build queue will be created as a unit object and added to the world. The factory then begins working on the next order in the queue. The hive operates similarly, only it has no queue and produces scouts and warrior in a 1 to 4 ratio. Newly created warriors will be added to the warriors list in the hive.

General → Builder.addOrder() → [Stockpile.removeResource(), World.planPath()]
General → Builder.cancelOrder()→ Stockpile.addResource()
Builder.constructBuilding() → World.addObject()
The general can select a builder and use options to add and cancel orders. When an order is added, resources are removed from the stockpile and a path is planned from the builders current location to the location of the new build order. The builder begins moving to the location of the build order on the top of the queue and begins a timer once it reaches that location. When the timer reaches zero, the building is added to the world. Cancelling a build order will notify the stockpile and add the resources that were taken. Once again, Stockpile.removeResource() should return a Boolean value representing whether the order could be successfully added or not.

General → Collector.setDestination() → World.planPath()
Collector.pickupResource() → World.removeResource()
Collector.deliverResource() → Stockpile.addResource()
The general can select a collector and set its destination by selecting a resource spawner on in the world. Once a destination is selected, the collector begins moving on a path to that

location. Once the collector reaches the resource, it will gather the resource, and begin moving back to the stockpile to deliver it. Once a resource spawner in the world has been completely depleted, it will be removed.

General → Turret.setTarget()
The general can select a turret and set the target to an enemy unit within its range.

General → Unit.setDestination() → World.planPath()
The general can select a unit and use an option to select a destination for that unit to move to following a path planned by the world. If the unit is a tank, the general can select an enemy unit as a destination instead. If the unit is a collector, the general can select a resource spawner in the world as a destination instead.

[Tank.shootTarget(), Turret.shootTarget()] → Enemy.takeDamage() → World.removeObject()
Warrior.bite() → [Unit.takeDamage(), Building.takeDamage()] → World.removeObject()
Tanks and turrets will shoot at the first enemy to enter into their weapon range unless they have a target, if it is a turret, or an enemy selected as a destination, if it is a tank. Shooting an enemy will subtract the tank/turret's power from the enemy's health. If the enemy's health reaches zero, it will be removed from the world. Similarly, warriors will try to bite the first unit or building that enters into their aggression range. The warrior will subtract its power from the unit/building's health, and if this health reaches zero, the unit/building will be removed from the world.

Scout.explore() → World.planPath()
Scout.reportTarget() → Hive.orderAssault() → Warrior.setDestination() → World.planPath()
Scouts will wander around the map somewhat randomly, though attempting to discover buildings and units controlled by the general. Once a scout reaches one of these targets, it turns around and plans a path to hive to report its enemy's location. When the scout reaches the hive, it transfers its target to the hive so the hive can broadcast that location to all of its warriors, setting each of their destinations and planning paths to the target. The warriors then go towards that location in full force to attack and destroy the destination, and anything else that enters their aggression range, until the target is destroyed, or the warrior itself is.

Appendix A - READ_ME.txt:

USE THE DEMO_MODE TOGGLE ON THE WORLD OBJECT (IN UNITY PROJECT EDITOR) TO SWAP BETWEEN DEMO MODE (ALL UNITS AND BUILDINGS INITIALLY SPAWNED) AND GAME MODE (PLAYER STARTS WITH STOCKPILE, ONE BARRACKS, AND ONE BUILDER) (THIS IS THE WAY THE GAME WAS MEANT TO BE PLAYED).

OBJECTS:

MetalSupply = grey cube

GasSupply = orange cylinders with gas speying out

Stockpile = fat green cylinder

Factory = big red box

Turret = skinny green tower

Barracks = little green box

Collector = blue capsule

Builder = flat blue box

Tank = red flattened sphere with turret

Hive = big orange sphere

Warrior = red bugman

Scout = yellow bugman

Walkthrough:

1. select builder and use it to build a factory.

2. select the factory and use it to build two collectors.

3. select the collectors and use them to collect from metal supplies on the map.

4. use the builder to build harvesters on gas supplies and barracks, gas and manpower are needed to make more units.

Additional guidelines/tips:

5. remember to retarget collectors which have depleated metal supplies on the map.

6. the builder can also create turrets to defend your buildings and units, and additional factories.

7. use the factory to make more builders, collectors, and tanks.

8. use tanks to target enemies and to eventually go on the offensive and assault the bugmen hive to win the game.

9. when a scout finds a player's unit or building, it will run back to the hive to report the location and then the warriors will begin an assault. you can use tanks to hunt these scouts down before they reach the hive or you can let them detect you far from your base in order to bait an assault toward that location.

10. you lose the game if the warriors destroy your stockpile.