**Clash of Bears – Tech Brief**

**Enemy AI**

Abhilash Ravindranath - 26 Sept, 2023

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Description** | **Author** | **Date** |
| 0.1 | Initial revision | AR | 26-Sept-2023 |

|  |
| --- |
| **Table of Contents** |
| [Goals](#_77am1vxdp3za)  [Dependencies](#_ua71e8z15zik)  [Requirements](#_jm4ce76ganwt)  [Technical Approach](#_qstbaz7yf5oi)  [See Also](#_omxloiqvood5) |

# **Goals**

*("Why" -- explain why this feature is important and what it must achieve in order to be successful.)*

Key goals include:

1. Creating different states for the enemy “Player”
2. Resource collection
3. Building
4. Training units
5. Sending units to attack
6. Exploring map
7. Expanding territory

2) Collaborating with different feature owners to link feature scripts with AI states.

3) Create a challenging AI that mimics how a human player would play.

# **Dependencies**

*("Where" – Identify where your feature overlaps, depends, or supports other portions of the project [features, art, level design, etc]. Be as detailed as possible here, and provide a plan for how to handle these dependencies.)*

1. Resource Management:

Enemy builders will collect resources and bring them back to the base building (town hall). Builders will prioritize which resource to collect depending on a predefined ratio.

2. Building Management:

Enemy builders will construct buildings based on a predefined priority (per town hall):

1. Town Hall
2. Farms
3. Lumber mill
4. Blacksmith
5. Barracks
6. Towers (Defence)
7. Shipyard

3. Combat:

Enemy units will be trained, and will go into attack mode when certain prerequisites are met (Game time, units trained etc)

4. Exploration:

The first unit trained will go into explore mode. Units will keep exploring until the entire map is revealed.

5. Expansion:

When resources run out, or a town hall reaches a certain size, the enemy will build a new town hall in the next resource hub.

# 

# **Requirements**

*("What" -- clarify what must be delivered to successfully meet the goals, as well as any additional constraints and requirements as appropriate.)*

1. AI Manager Script with the following states (coroutines)

1. Idle: Do nothing. The AI will be in this state when there are enough builders, resources, and units.
2. Train units: AI will go to this state when there are fewer than necessary builders or combat units.
3. Build: AI will enter this state IF

i. There are sufficient resources

ii. The building is not already built in the town hall

1. Gathering resources: AI will enter this state, which will train builders if necessary or make builders do different tasks, if the resources do not meet the minimum threshold.
2. Explore: AI will train combat units, and will send them to explore the map until a fraction of it is revealed.
3. Attack: AI will train combat units, and will send them to attack the player in waves during random times during the game, and if sufficient units are trained,

**Technical Approach**

*("How" - what is your proposed strategy for meeting the requirements? Provide enough detail to evaluate the approach. Need not be a full-on technical design, but should be detailed enough to either start development or guide a detailed technical design when needed.)*

I will be the creative director for this game, and will assign feature owners certain requirements to reach each week. Then, each week we will link all the scripts and work on a combined scene, which we will use to playtest the game.

First phase – Create all minimum necessary scripts, assets etc to test if the prototype works as expected.

Second phase – Add additional scripting and assets to create the MVP.

Third phase – Time allowing, we will polish the game and add all nice to have features to the game.

Each phase should not take more than 2 weeks.

# **See Also**

*(Further reading and reference material as appropriate. Link external web sites, internal docs, etc. with a brief description to explain why it's relevant.)*

1. Case study - Warcraft 2 (Blizzard, 1995):

We will be modeling the game to replicate the mechanics and functionalities of Warcraft 2. One of the earliest but most robust entries into the RTS genre. We can replicate the in game economy, unit behaviors, resource harvesting times etc.

This will allow us to focus on programming instead of balancing the game.

2. Unity documentation (as required).

3. Asset store for miscellaneous art assets (as required).