**Pathfinding System for RTS Project**

Jay Patel - <Clash Of Bears>

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

# **Goals:**

**Why:**

This feature is the most crucial as it aims to facilitate intuitive movement of player and enemy units, enhancing player control and immersion within the real-time strategy (RTS) gameplay.

**Explanation:**

In an RTS game, pathfinding allows game entities to navigate obstacles, find optimal routes, and interact with the game world realistically. It also ensures smooth navigation, promoting strategic decisions and tactical engagements.

**Key Goals:**

* Intuitive unit movement through player-directed actions.
* Efficient dynamic obstacle handling for adapting to changing game conditions.
* Seamless calculation of optimal paths for multiple entities simultaneously.
* Error handling to manage unexpected situations during navigation.

# **Dependencies:**

1. Building Management System:

* Requires information about valid areas for constructing buildings withing the game.
* Requires information about building locations for path calculations.

1. Dynamic UI System:

* Facilitate visual representation of available building locations based on NavMesh data for an enhanced user experience.
* Integrate with the NavMesh system to provide real-time information about paths and obstacles to the player during unit movement.

1. NPC AI System:

* Critical for the NPC Ais to ensure intelligent movement, tactical decision, and efficient path planning.

1. Animation and Particle Effects:

* Animation and particle effects can be triggered based on the NavMesh-calculated paths to provide visual feedback on unit movement and enhance the gaming experience.

1. Level Design:

* Level design should consider NavMesh-generated paths to ensure that the game environment is conducive to efficient unit movement and strategic gameplay

# **Requirements**

* Navigation System:
  + Implementation of a navigation graph representing walkable areas and obstacles. (Handled by NavMesh)
* Pathfinding Algorithm:
  + Integration of a suitable pathfinding algorithm (e.g., A\* algorithm) for efficient path calculation based on the navigation graph. (Handled by NavMesh)
* NavMesh Baking:
  + Generate and bake NavMesh data for the game environment, defining walkable areas and potential obstacles for pathfinding.
* Integration with Entity AI:
  + Integrate NavMesh pathfinding into the entity AI system to facilitate intelligent movement and decision-making for bear units.
* Path Constraints and Dynamic Updates:
  + Implement logic to impose specific constraints on paths and update the NavMesh dynamically to handle moving obstacles or changes in the environment.
* Error Handling:
  + Implement error-handling mechanisms to manage unexpected situations and ensure a smooth pathfinding experience.

**Technical Approach:**

* Utilize Unity's built-in NavMesh system, which is a robust and efficient pathfinding solution.
  + Set up NavMesh areas in Unity to define different types of terrain or movement costs.
  + Use NavMesh Agents to allow units to find optimal paths in real time.
  + Define NavMesh area types (Walkable, Not Walkable)
  + Utilize NavMesh.SamplePosition to check if a location is valid for building placement.
  + Ensure synchronization and real-time updates between the NavMesh system and data management components.
  + Use NavMeshAgent.SetDestination() to set NPC destination and navigate the NPC towards the target location using NavMesh.
  + Collaborate with level designers to optimize NavMesh data based on level layout and objectives.