**Camera Systems for RTS Project**

Baqar Ali Zaidi - <Clash Of Bears>

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

# **Goals:**

**Why:**

The isometric camera system and fog of war feature are crucial for imelementing and enhancing gameplay in our real-time strategy (RTS) game.

**Explanation:**

In an RTS game, the camera system is pivotal for player interaction and engagement. The isometric camera provides a strategic view, reminiscent of popular games like World of Warcraft 2, which players are familiar with and find immersive. The fog of war feature adds depth to gameplay by limiting player knowledge of unexplored areas, encouraging exploration, and introducing strategic elements.

**Key Goals:**

Implement a top-down isometric camera system similar to World of Warcraft 2 to provide players with a familiar and engaging perspective.

Explanation: This goal focuses on creating a camera system that emulates the iconic isometric perspective of WoW 2. The camera should offer smooth controls and a view angle that allows players to strategize effectively.

Create a dynamic fog of war system that enhances strategic depth by limiting player knowledge of unexplored areas.

Explanation: Fog of war adds depth to the game by hiding unexplored terrain and enemy movements. Achieving this dynamic fog system will challenge players to make tactical decisions based on limited information.

Ensure smooth and responsive camera controls for an immersive gaming experience.

Explanation: Smooth and responsive camera controls are essential for player enjoyment. Without responsive controls, the game can feel frustrating and less immersive.

# **Dependencies:**

1. Pathfinding Systems:

Overlap: Pathfinding systems help units navigate the game world. The camera's view affects how paths are calculated, and fog of war can impact unit movement choices by limiting visibility.

Supports: The camera and fog of war features depend on pathfinding to ensure units move intelligently and to provide accurate vision-based fog updates.

2. Resource Management Systems:

Overlap: Resource management impacts gameplay and player decisions. The camera's UI might display resource quantities, and resource availability can affect what buildings or units players can produce.

Supports: The camera's UI may need to display resource information, and fog of war might hide resource locations that players must uncover.

3. Building Management Systems:

Overlap: Building construction and placement can be influenced by the camera's perspective. Fog of war may hide areas where players can build structures.

Supports: The camera system affects how players view and interact with buildings, while fog of war impacts which areas are suitable for building.

4. Dynamic UI Systems:

Overlap: The camera system often includes user interface (UI) elements for player interaction. UI elements like minimaps and fog of war indicators are directly related to these features.

Supports: Dynamic UI systems are essential for displaying fog of war information and providing camera control options to players.

5. Data Management Systems:

Overlap: Data management systems handle game data, including unit stats, building properties, and fog of war states. The camera and fog of war systems rely on accurate data representation.

Supports: Data management ensures that the camera and fog of war systems have access to the necessary information for rendering and gameplay.

6. NPC AI Systems:

Overlap: NPCs (Non-Player Characters) use the same camera perspective as players. The fog of war can affect NPC behavior, as they may make decisions based on what they can see.

Supports: The camera view affects how NPCs perceive the game world, and fog of war can influence their strategic decisions and interactions with players.

8. Combat Systems:

Overlap: Combat systems involve unit interactions, attacks, and defense. The camera perspective affects how combat animations and effects are displayed.

Supports: Fog of war can hide enemy units, influencing combat encounters and forcing players to make decisions based on limited information.

9. Animations and Particle Effects:

Overlap: Animations and particle effects are visually represented in the game world. The camera's view determines how these effects are displayed.

Supports: Fog of war can obscure animations and effects, impacting the visibility of combat or other in-game actions.

10. Audio:

Overlap: Audio cues and sound effects contribute to the game's immersion. The camera's view can determine which audio sources are audible.

Supports: Fog of war can limit what players hear, affecting their awareness of off-screen events.

11. Level Design:

Overlap: Level design includes map layout, terrain, and the placement of objects. The camera system's perspective influences how the game world is presented.

Supports: Fog of war impacts how level designers create maps and how players explore and strategize within them.

# 

# **Requirements**

**Isometric Camera System:**

An orthographic camera that provides a top-down perspective.

Smooth panning and zooming controls using keyboard (WASD) and mouse scroll wheel, akin to WoW 2.

Optional: Camera rotation controls for added flexibility.

Boundary constraints to prevent the camera from moving outside the playable area.

Dynamic camera behavior that adapts to the game's pace and action.

**Fog of War:**

A grid-based system covering the entire game map.

Units and buildings possess vision ranges.

Continuous updating of fog of war based on unit movement and exploration.

Realistic line-of-sight calculations for advanced fog of war (optional).

An interface element, such as a mini-map overlay, to display explored and visible areas.

Optimized performance for large game maps.

**Technical Approach:**

**Isometric Camera System:**

* Create an orthographic camera in Unity.
* Explanation: In Unity, configure the camera to use orthographic projection to achieve the top-down isometric view.
* Implement input handling for panning (WASD) and zooming (mouse scroll wheel).
* Explanation: Develop input scripts that detect WASD key presses for panning and mouse scroll wheel input for zooming.
* Develop a camera script to manage boundary constraints and camera behavior.
* Explanation: Write a custom camera script that enforces boundary constraints and responds dynamically to in-game events.
* Optionally, implement camera rotation controls if required by game design.
* Explanation: If the game design calls for it, add controls to allow players to rotate the camera.
* Rigorously test camera controls for responsiveness and smooth movement.
* Explanation: Thoroughly test the camera controls to ensure they are responsive and provide a smooth gaming experience.

**Fog of War:**

* Establish a grid-based system to represent the fog of war.
* Explanation: Create a data structure that covers the entire game map, with each cell representing a state of the fog of war.
* Assign vision attributes to units and buildings.
* Explanation: Define attributes for units and buildings that determine their vision range and contribute to the fog of war calculations.
* Optionally, incorporate line-of-sight calculations for advanced fog of war.
* Explanation: If advanced fog of war is desired, implement algorithms that consider line of sight, obstacles, and terrain.
* Utilize shaders or masking techniques for rendering the fog.
* Explanation: Use graphical techniques to render the fog over unexplored and explored areas, providing a visual representation to players.
* Integrate a user interface element to display fog of war information.
* Explanation: Design and implement a user interface element that conveys fog of war information to players, such as a mini-map overlay.
* Optimize fog of war updates for performance on varying map sizes.