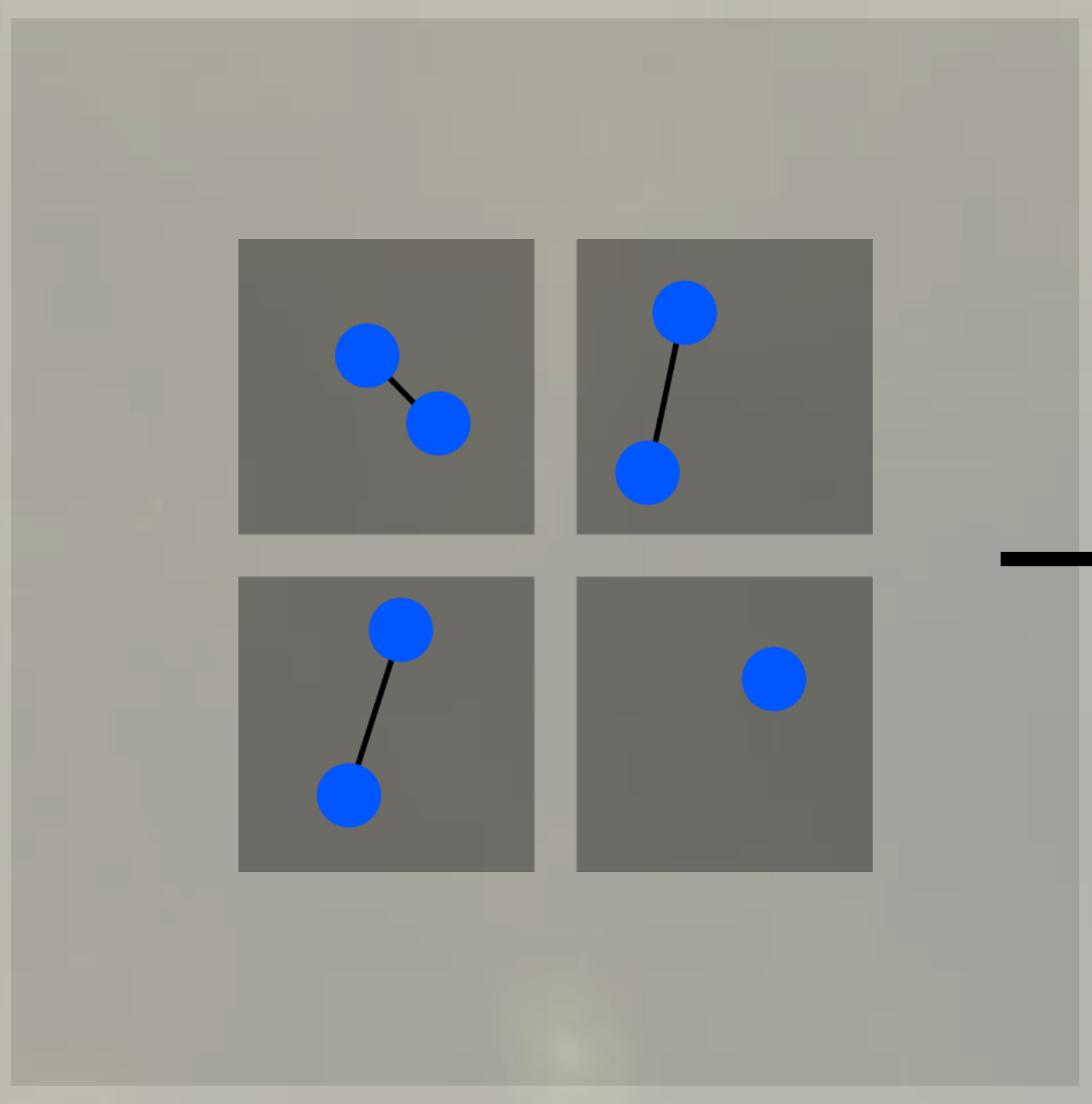
TECHNICAL

Brett Dickson Jason Flanders
Chelsey Salberg Noah Torrance
Steven Vignos

MAP GENERATION ALGORITHM

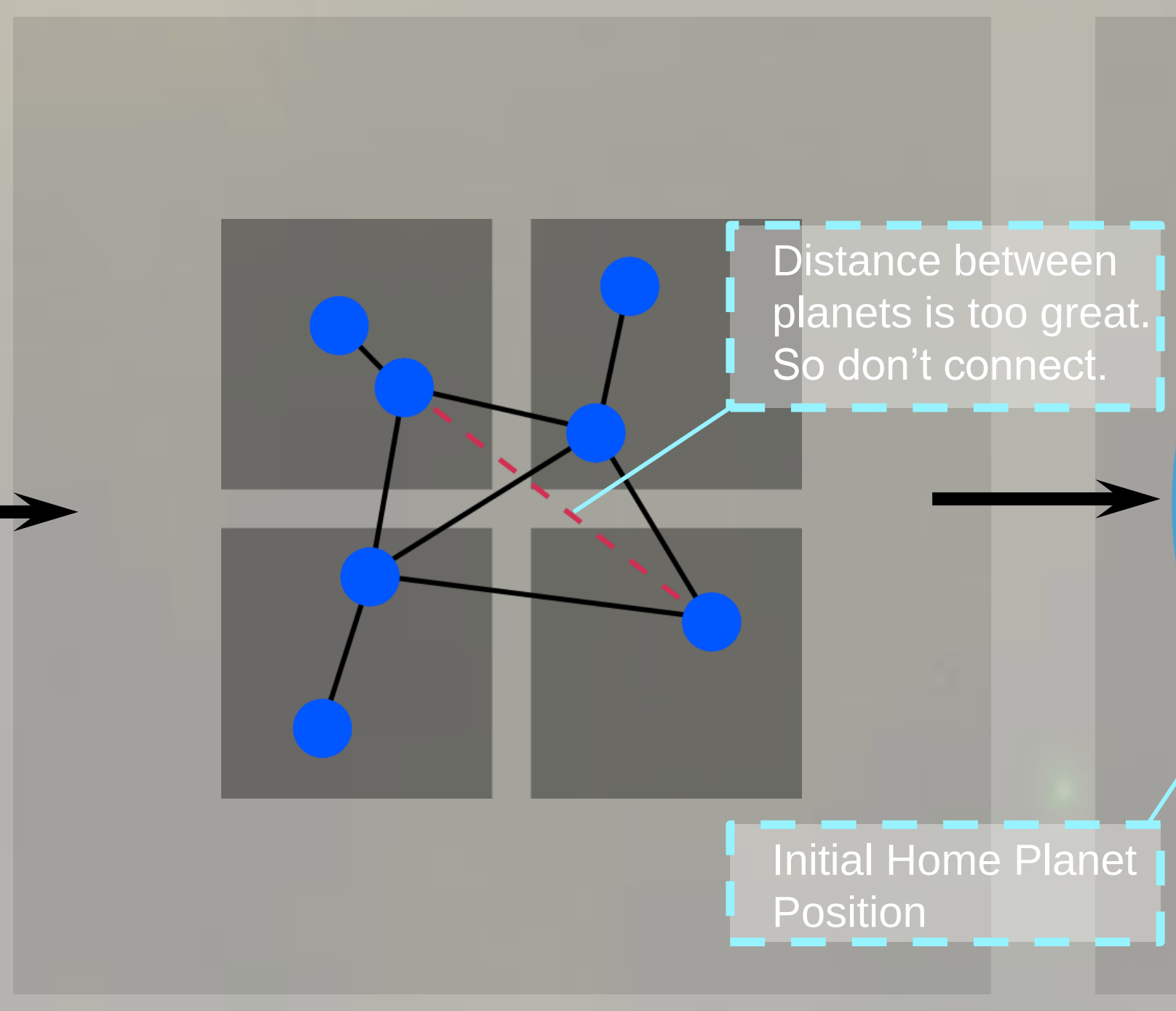
Place Sectors and Planets



Planets are placed in connected pairs in a square grid whose dimension is the square root of the ceiling of half the number of planets. The position of planets within grid spaces are randomly determined using the Poisson disk algorithm.

If there are an odd number of planets, one of the grid spaces will have only one planet.

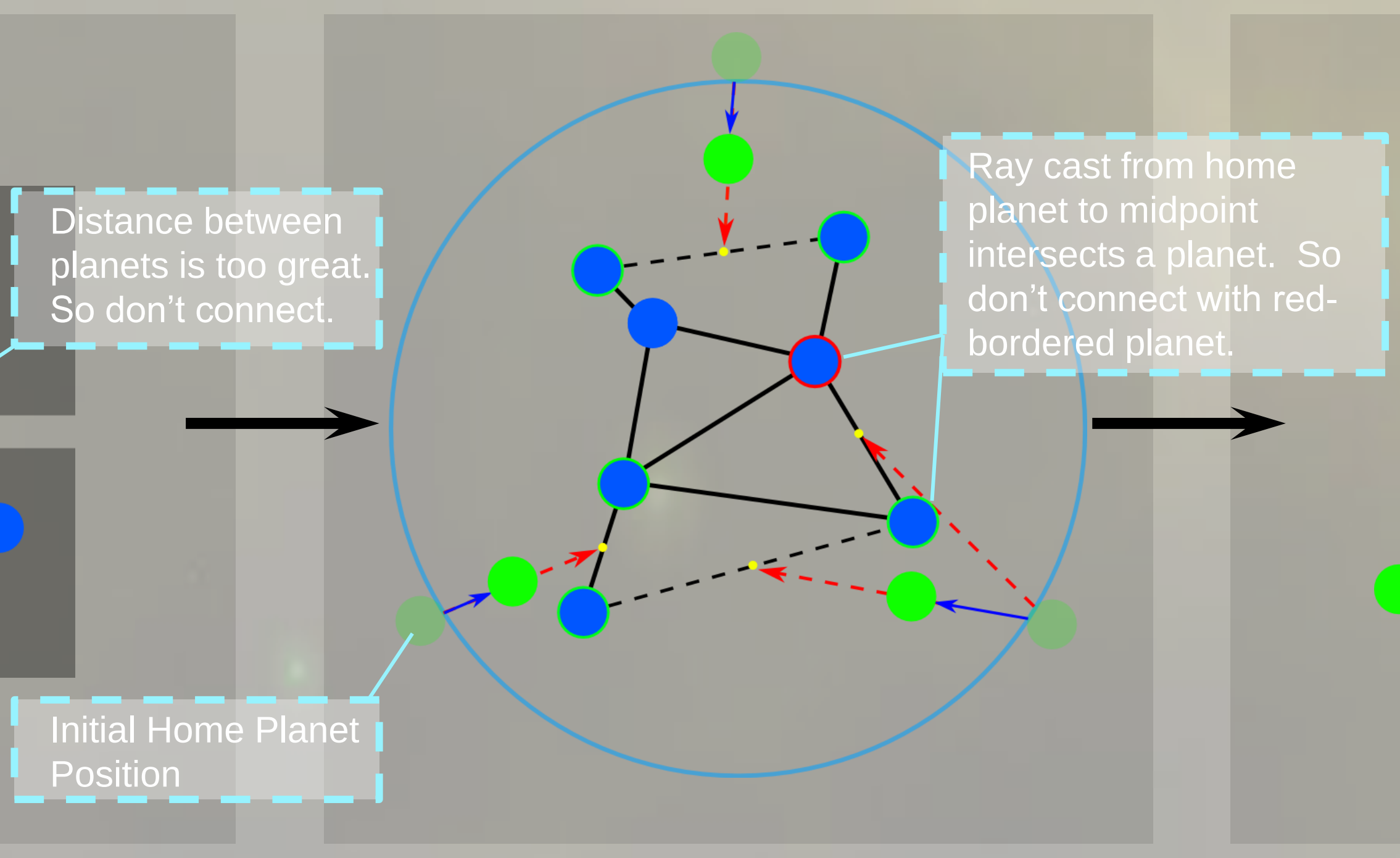
Connect Sectors



The closest planets between 2 adjacent grid spaces are connected with an edge.

Diagonally adjacent grid spaces are only connected if the 2 closest planets between them are within a set distance proportional to the size of the map.

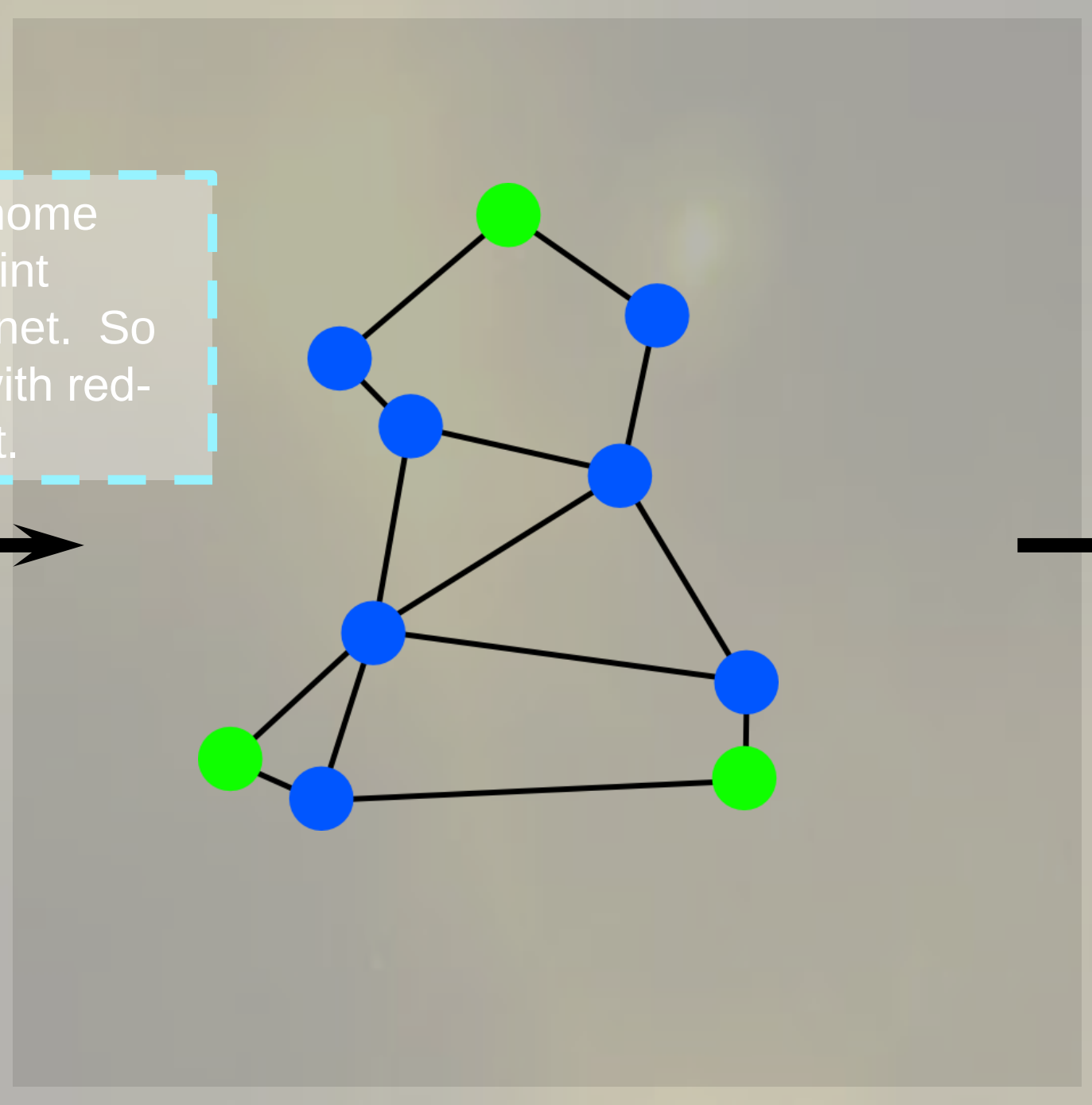
Place Home Planets


Ray cast from home planet to midpoint intersects a planet. So don't connect with red-bordered planet.
Initial Home Planet Position

Home planets are placed circularly around the map.

For each home planet, 2 of the closest planets to the home planet is chosen. If the edge to the midpoint between the 2 planets intersects a planet, then the farther of the 2 is replaced with the next closest planet. The home planet is then moved closer towards the midpoint proportionally to the size of the map.

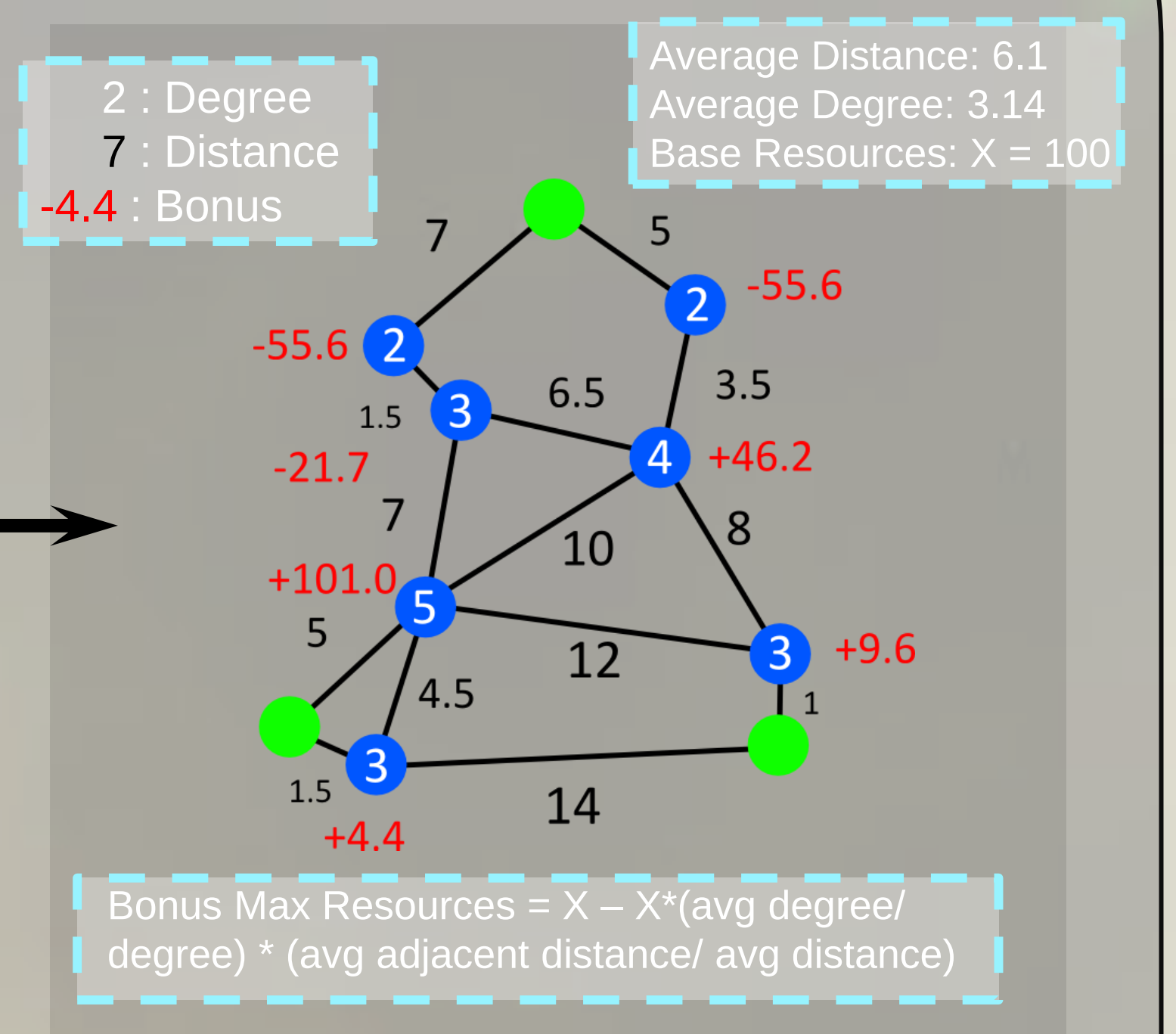
Connect Home Planets



The home planets are connected to the planets chosen in the previous step.

The result is a map in which the planets do not overlap, edges do not intersect planets except at their end points. The resulting map can be described as a strongly connected graph where the degree of the vertices (the planets) is at least 1 and at most 8.

Assign Bonus Resources



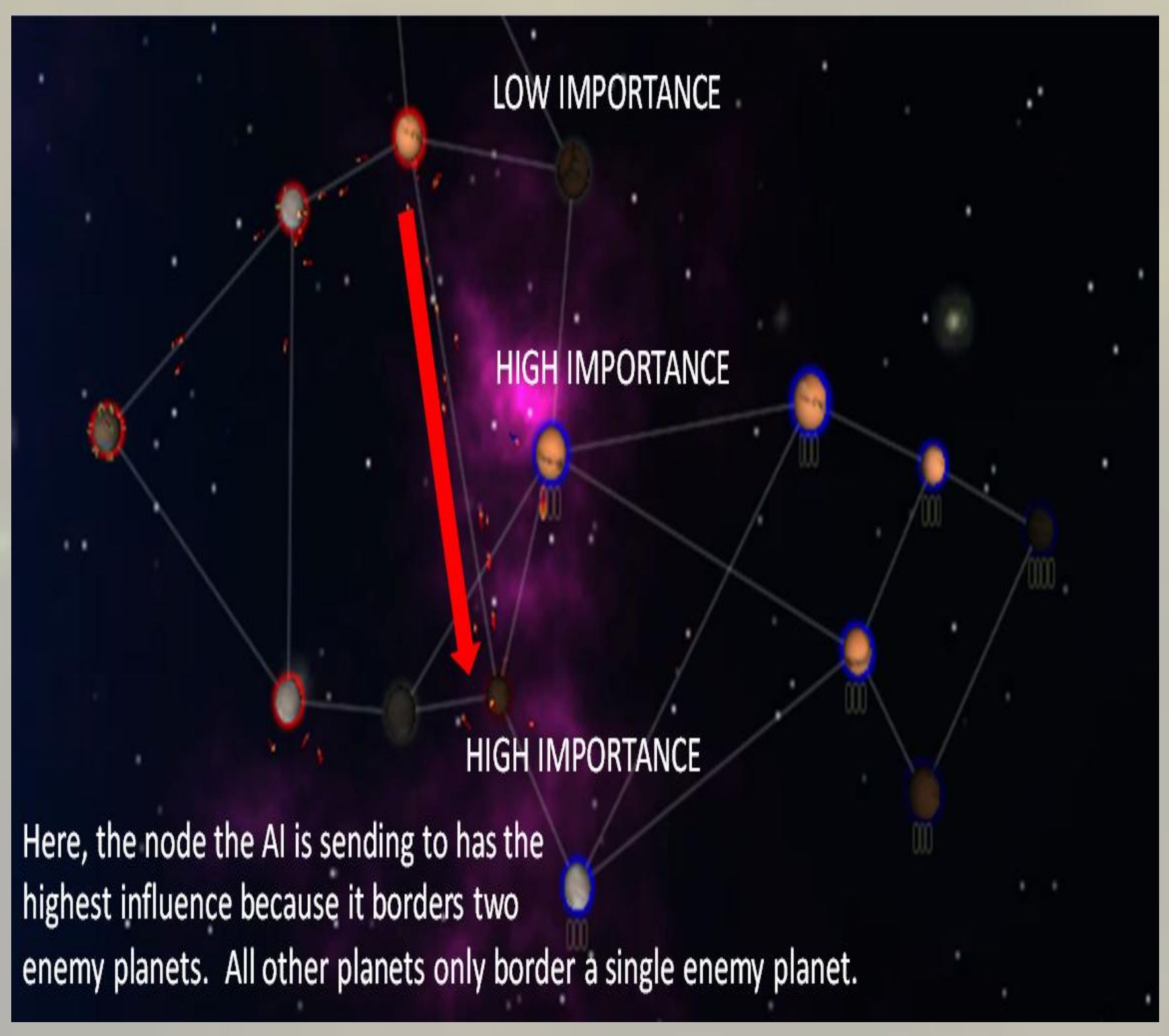
A maximum resource value is assigned to each planet. A bonus to the maximum resource value is calculated after placing and connecting the planets.

The bonus value assigned to a planet is proportional to the number of planets it connects to, and proportional to the average distance to the planets it connects to.

Home planets have no map-based bonuses.

ARTIFICIAL INTELLIGIENCE: INFLUENCE MAPS

Influence mapping is a model for artificial intelligence systems to make accurate decisions based on the current state of the perceived world. Here, influence maps track the state of game by considering all the planets and how they relate to the AI's controlled planets. There are many different factors that contribute to influence values, such as connectivity of the examined planet and even further, who controls said surrounding planets.

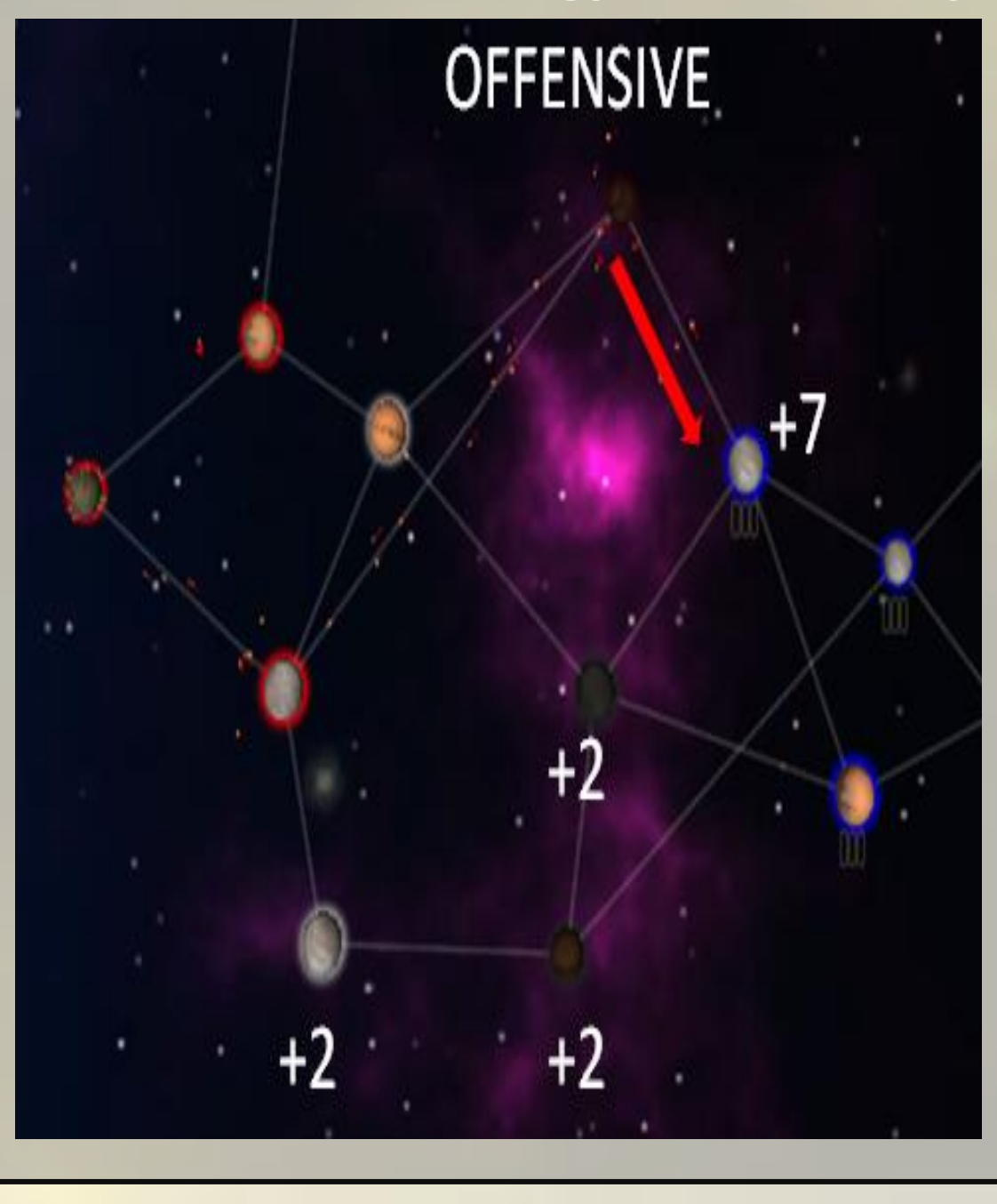


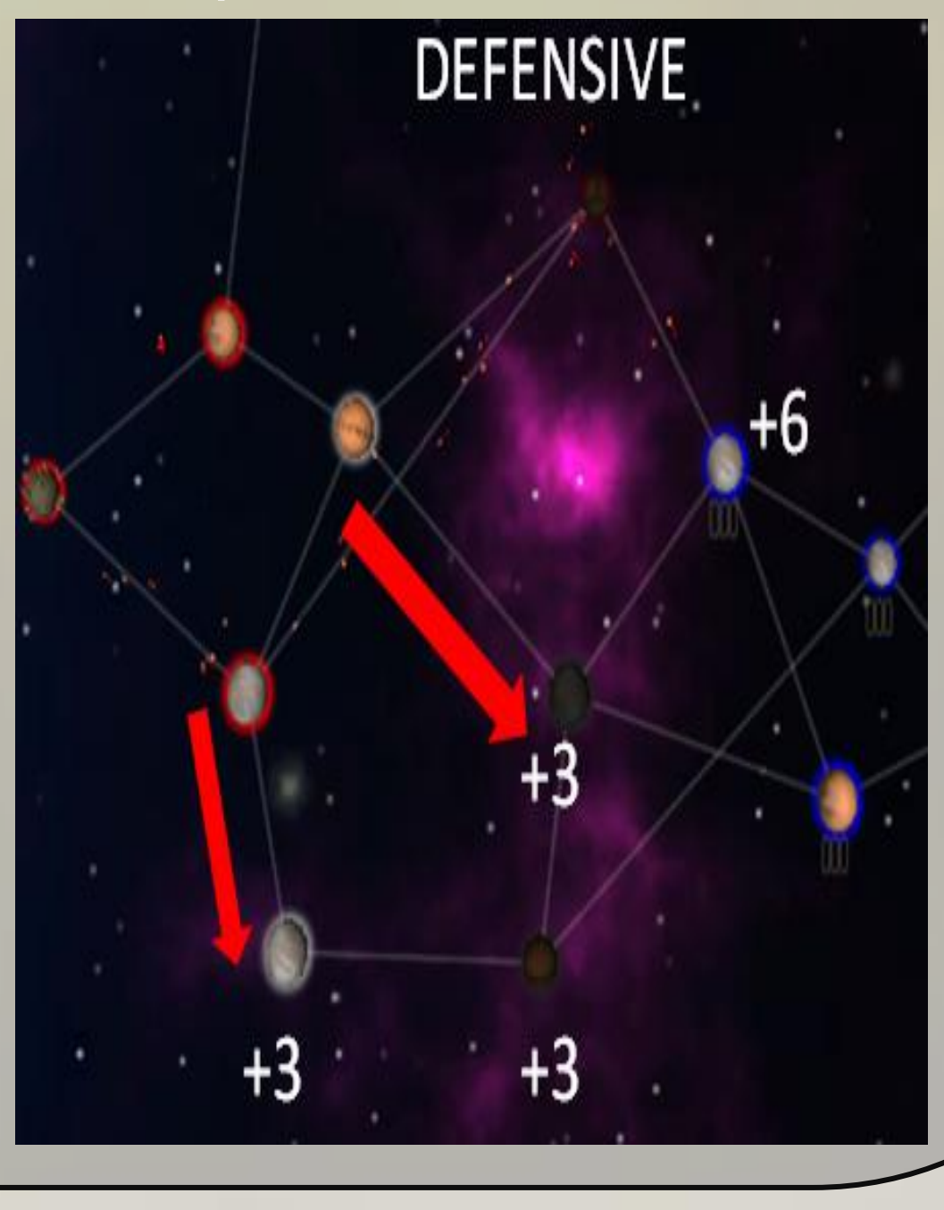
Here, the node the AI is sending to has the highest influence because it borders two enemy planets. All other planets only border a single enemy planet.

The AI makes a decision based on nodes with the highest influence values. Naturally, since a player cannot send units to a planet that is not directly connected to a planet that that player owns, the AI does not take into account these planets. In this way, the AI is not omniscient.

AI TEMPLATES

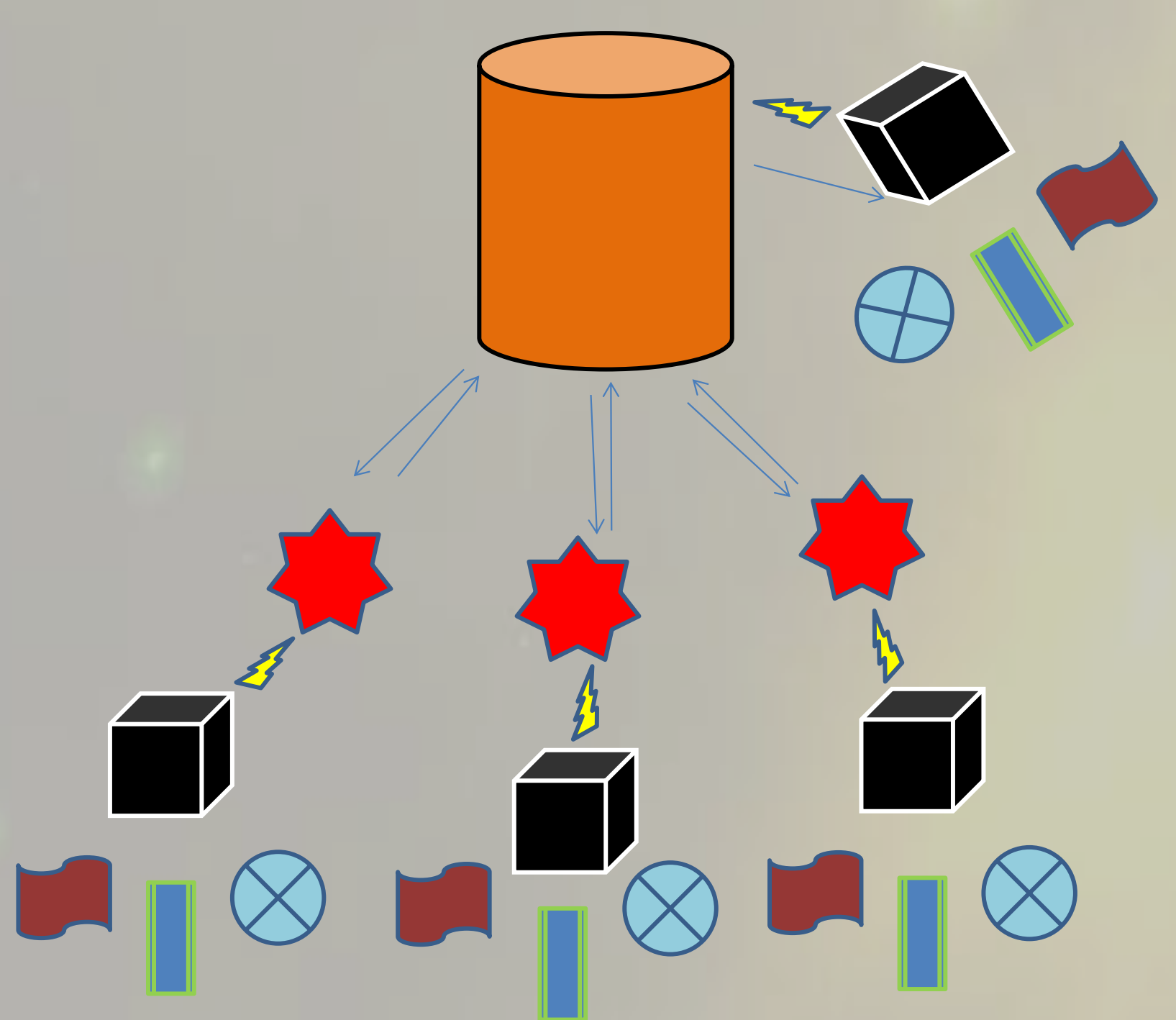
The AI also can be instructed via the pregame lobby to behave in an offensive, defensive, or rapid expansion fashion. This can be achieved through setting the influence values differently based on which behavior the user chooses. For example, the influence value for a reachable enemy planet on a defensive template is +3 whereas a reachable enemy on an offensive template is +7. So, when the AI makes a decision, it is more likely that it will send units (and thus attacking) the enemy when its template is offensive.





NETWORKING

Remote Procedure Calls (RPC's) are used by all clients including the client local to the host server. They are used to synchronize the scene for all players by firing events to the server when clients create game objects or change elements.



Authoritative Server Structure

Each client communicates directly to the server, and the server broadcasts to all clients.

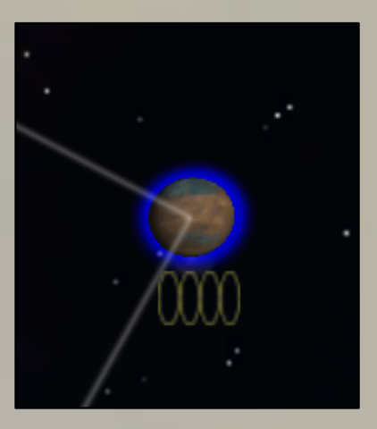
Clients are responsible for drawing and moving certain objects within their own scene, maintaining the local GUI, and responding to player input by firing remote events to the server.

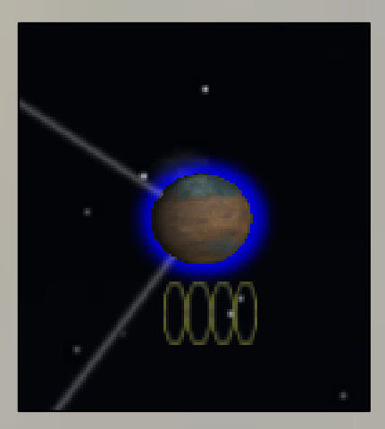
The server is responsible for handling all other game events.

FOG OF WAR SYSTEM

Linear interpolation is used to fade in/out objects from/to black as they enter/exit the fog of war.

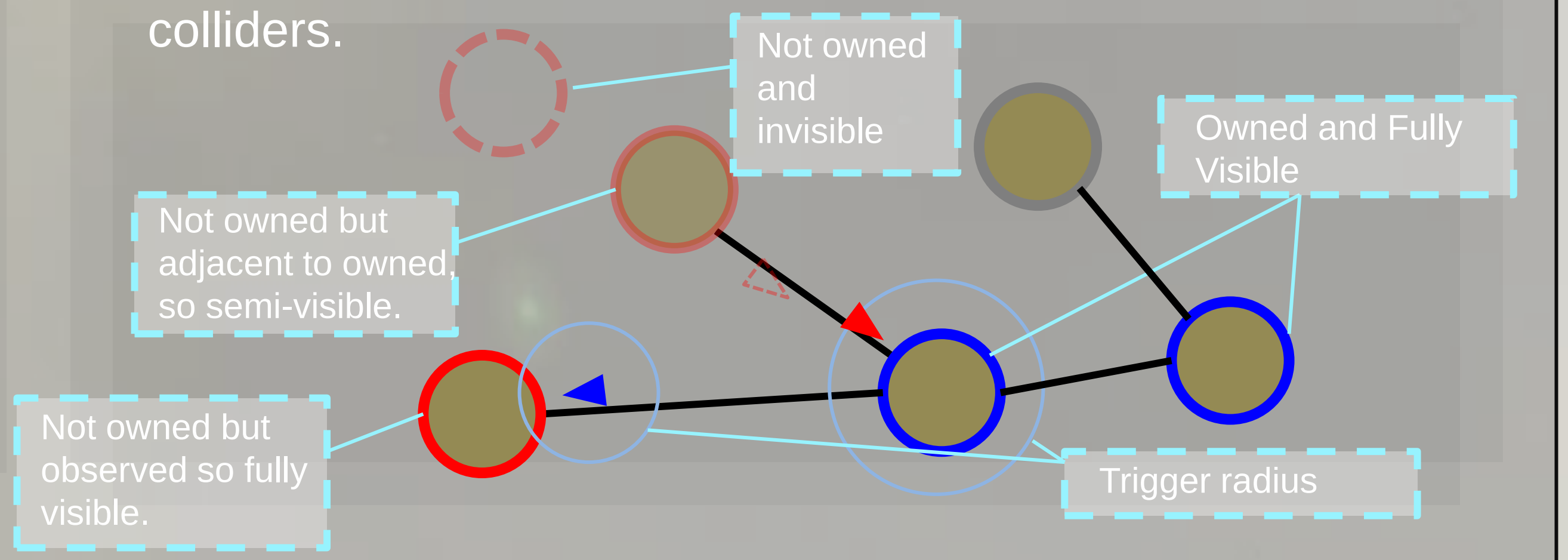
A custom shader was written to allow object transparency while keeping occlusion data about objects in the scene.


Standard Shader

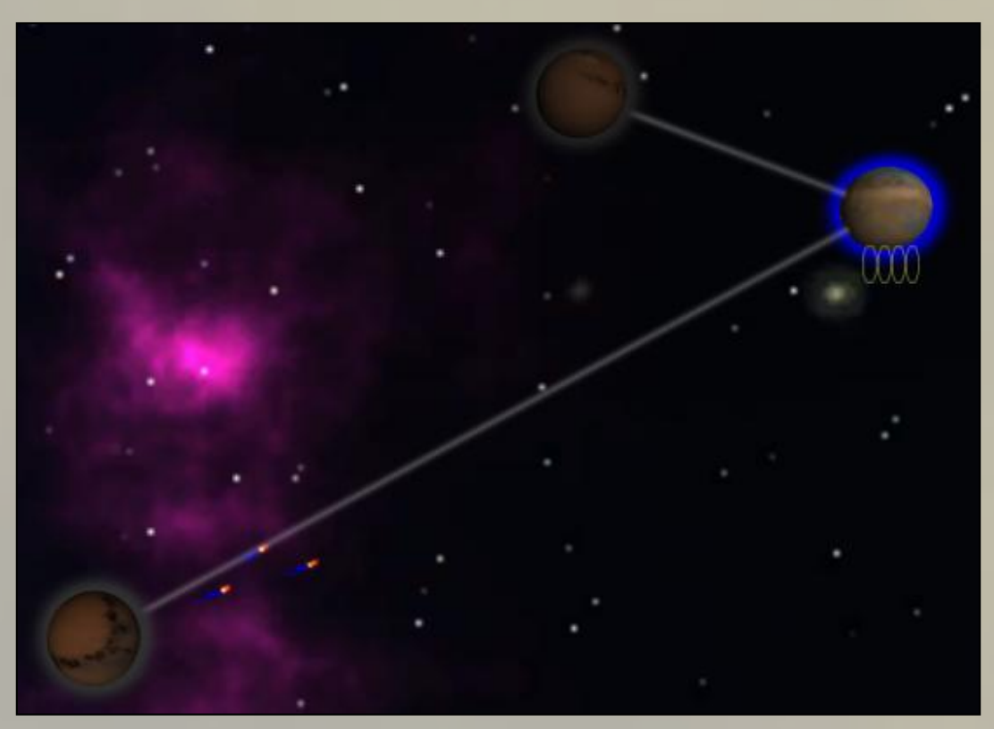

Custom Shader

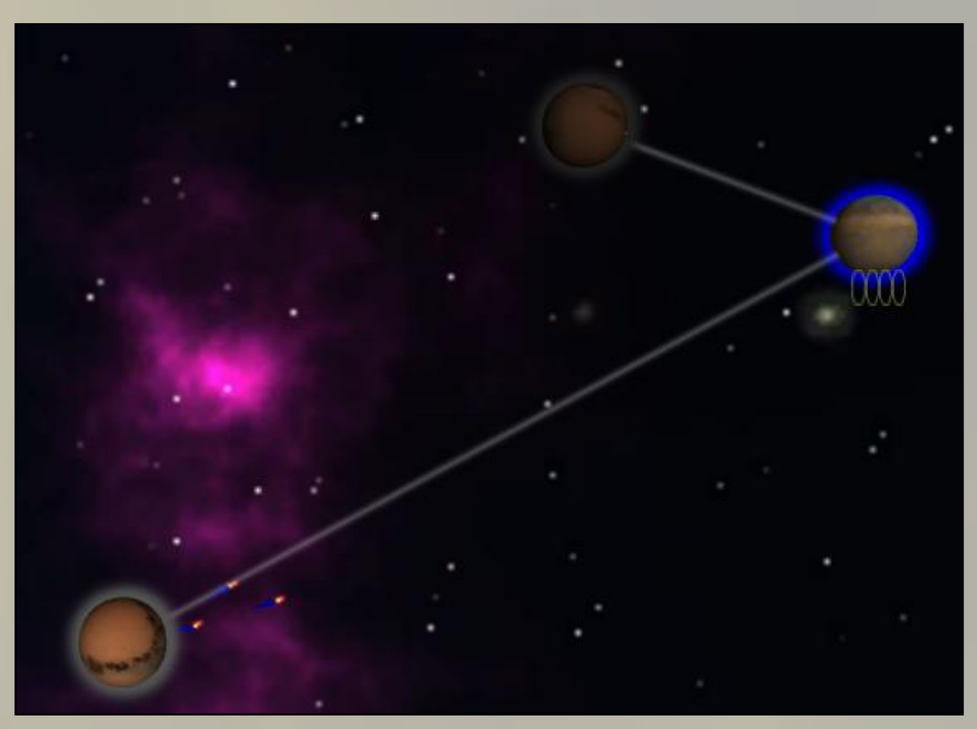
The Fog of War system uses Unity's layer Culling Mask system to only render certain layers. Visible and invisible layers are defined so that unknown and unobserved planets or fleets can be prevented from being rendered. Semi-visibility is achieved through color adjustment

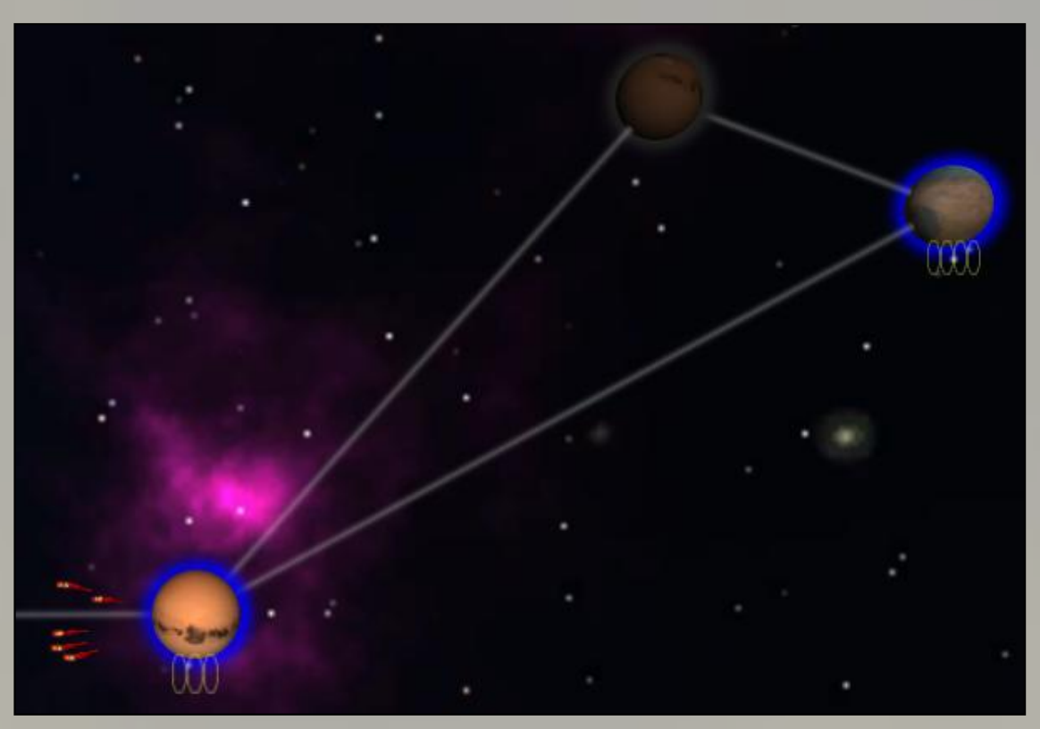
Player owned planets and fleets have triggers that increase visibility of objects that intersect with the trigger colliders.

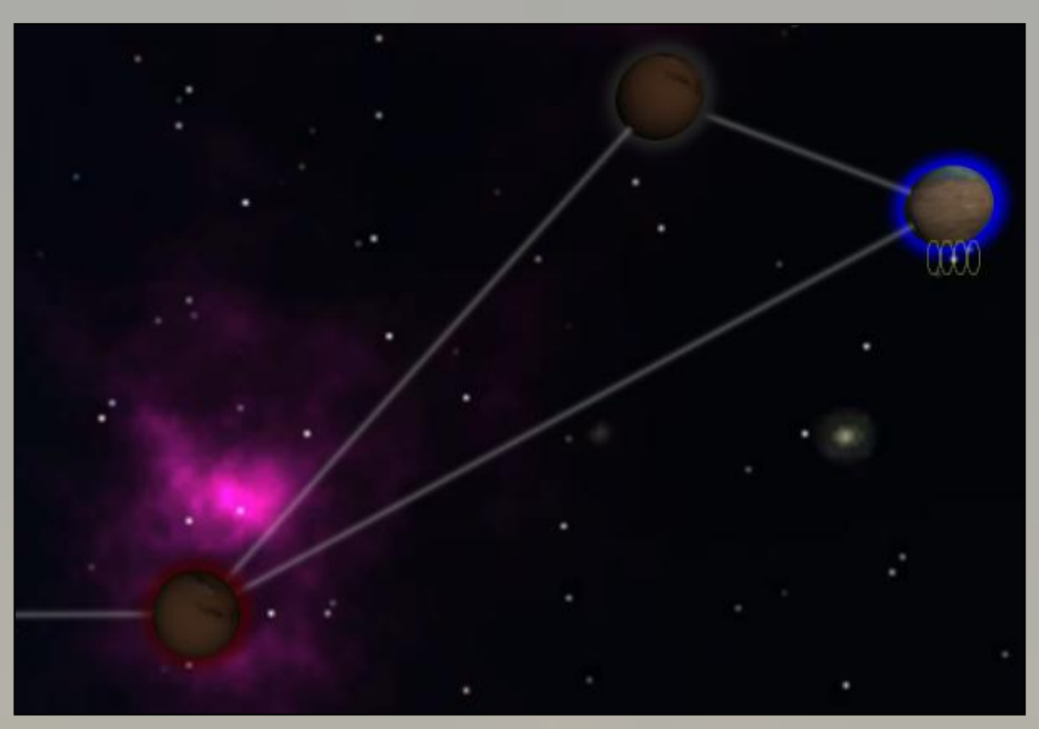

Not owned and invisible
Owned and Fully Visible
Not owned but adjacent to owned, so semi-visible.
Not owned but observed so fully visible.
Trigger radius

Fog of War effect in effect.


Semi-Visible and Fully Visible


Observed by Fleet


Captured Planet Now Observes


Lost Planet to Enemy