Eric Gladysz

Flotilla – Research Report

Problem Statement: “How can we add vagueness/uncertainty to fog of war in RTS mini-maps?”

Significance: RTS game have had one main concept of “stealth” for most of its history: fog of war. The general idea is that a play will only know of non-allied entities (units, buildings, etc.) if they are near an allied one. This large collection of radii made from every allied entity creates the area of visibility of what can be seen. In this system, “hiding” from an enemy is merely keeping one’s units out of an enemy’s visibility area. This is not a very flexible system on its own. The intention was to create a mini-map closer in behavior to radar maps, focusing more on chance detection, signal noise, and consideration for volume.

Development: To avoid the potential for cheating, it was decided early on that the server would generate the radar detection and send it to the clients as contextless “points” that represented a position in the world that meant “something is there”. A comparison was immediately made to rasterization in a render pipeline, and the crude first concept involved rendering the world in low resolution on the server side and use the resulting image as the map data to distort/send. This idea was quickly discarded as impractical, since the server would need to render the world, modify the resulting image manually, and then send that image over the network quickly, which was a computationally slow procedure and one that is not easy to develop quickly. Instead, a 2D world was used on the server which would occasionally “sample” random positions to see if they collide with any object. Points that did so would then be collected and sent over the network to clients, which would then render them for more than one frame. Server samples were limited to a certain number per tick per client, and only successful collisions were sent to clients, so there was a well-defined worst-case data usage of “every sample is a hit” which scaled linearly to the number of clients connected. Points were given a detection “strength” to give the client an idea of how “certain” a detection was (for this project, it was simply the radius from the point to the nearest edge of the collided object). The client rendered stronger points as larger and rendered them for longer times, gradually decreasing that strength as time went on, until the point lost all strength and was discarded.

Results/Conclusion: Sending only chance collisions with entities in the world and letting those points linger on the client side allows decent visual feedback on the client side without any extreme network usage. In the context of a mini-map, the small space and relatively smaller entities would be well represented with the variable-size points on a slightly less partitioned rendering space. This project uses floating point numbers for positioning, but an RTS could easily use integers for its map grid. It could also combine sampled points into higher strength points on a scaled space with fewer positions, allowing better packing of data on the network. Random modifications to the strength values of points (to simulate signal noise) can also easily be added. Not yet rectified is the transition between the radar-based system here and the “regular” view of entities in the regular fog-of-war system, though the visibility area could easily be ignored when choosing where to sample points, and the entity’s actual position can be converted to a “last-seen” radar point if visibility is lost for whatever reason. There is further potential in this system that I intend to further explore going forward.