Role Playing Game (RPG) Development for Artificial Intelligence Testbed

Katherine Lima[1] Department of Computer Science, Swarthmore College
Advised by. Aline Normoyle Department of Computer Science, Swarthmore College

Abstract

Over the summer I worked to create a custom RPG game that will be the basis for testing AI-controlled Bots as video games are great decision making environments. Our approach is to develop a player versus player, capture the flag game that would allow the user to set up matches, use custom abilities and battle other players. With this game, we plan to test companion AI Bots who can be members of a team and communicate with other players via chat. We used an open source medieval fantasy role player game called Flare to build our pvp game Capture (Figure 1 and 9).

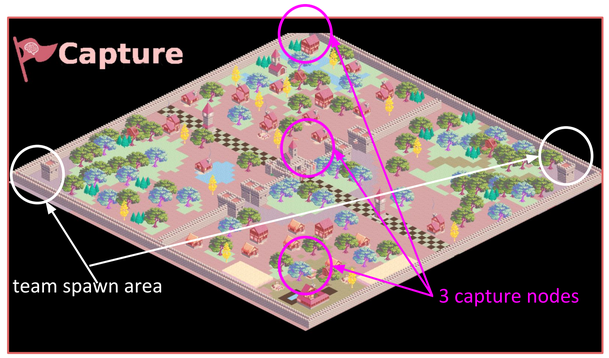


Figure 1. Our custom capture the flag game map.

Capture vs Flare

Flare contains many of the basic features that we would need for the game, such as fighting abilities, weapons, Health/Mana points and ability to die and respawn, graphics, user interface, animation system and path-planning.

However, to turn Flare into the testbed we need, there needed to be some reconfiguration. Flare is a single-player quest-based adventure game, that needs to be transformed into a competitive multiplayer game.

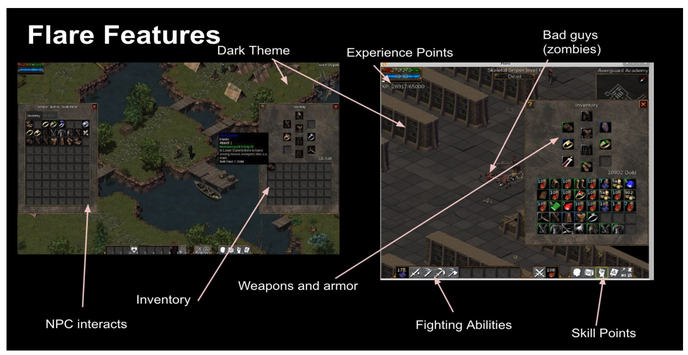


Figure 2. Features of the Flare Engine game

Capture Game Editing

A major part of creating our flare game was map editing! Here we have examples of our original flare-map layout and methods of creating custom maps for our game.

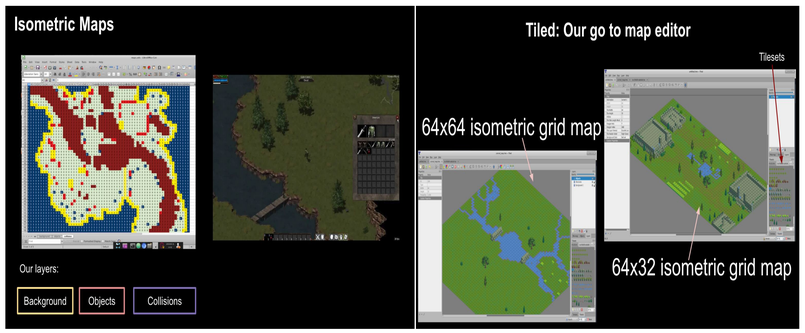
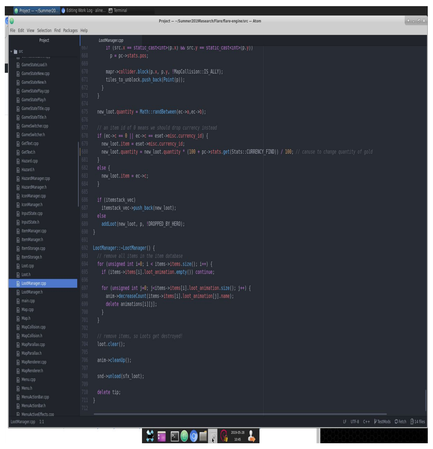


Figure 3. Flare 2D grid map layout with 3 different layers and two custom tiled maps made to demonstrate how 2D isometric maps work.



```
# Avatar Attributes Definitions
# Values should be integer

stat_points_per_level=5
max_points_per_stat=75

stat=hp,8000
stat_per_level=hp,50
stat_per_primary=physical,hp,20

stat=hp_regen,50
stat_per_level=hp_regen,50
stat_per_primary=physical,hp_regen,20

stat=mp,200
stat_per_level=mp,5
stat_per_primary=mental,mp,2

stat=mp_regen,5
stat_per_level=mp_regen,5
stat_per_primary=mental,mp_regen,2

stat=accuracy,85
stat_per_primary=offense,accuracy,1
stat_per_level=accuracy,3

stat=avoidance,15
stat_per_primary=defense,avoidance,1
stat_per_level=avoidance,3

stat=crit,1
stat_per_level=crit,1

stat=dmg_melee_min,0
stat=dmg_melee_max,0
stat_per_primary=physical,dmg_melee_min,1
stat_per_primary=physical,dmg_melee_max,1

stat=dmg_ment_min,0
stat=dmg_ment_max,0
stat_per_primary=mental,dmg_ment_min,1
stat_per_primary=mental,dmg_ment_max,1
```

Figure 4-5. Flare codebase, an example of customizing the games loot rate, in figure 5 we see an example of the game's stat text files that were an easy way to change components of the game such as abilities, level, and HP.

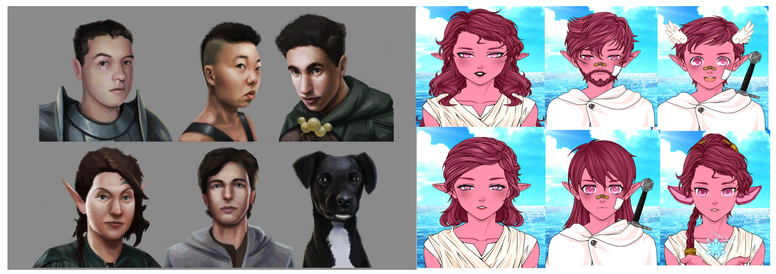


Figure 6. Flare Engine Portraits vs custom Capture Portraits. Our vision for the game, of course had to implement brighter and more colorful groups of possible portraits for players.

Multiplayer Networking

The future goal of this work will be to solidify networking and game logic that will allow us to fully perform our Capture game at full capacity. This environment will later be used to develop and test chatbots for the purpose of research. Figure 1 and 8 are examples of what our game looked like by the end of the summer in comparison to figure 2, the original flare game.

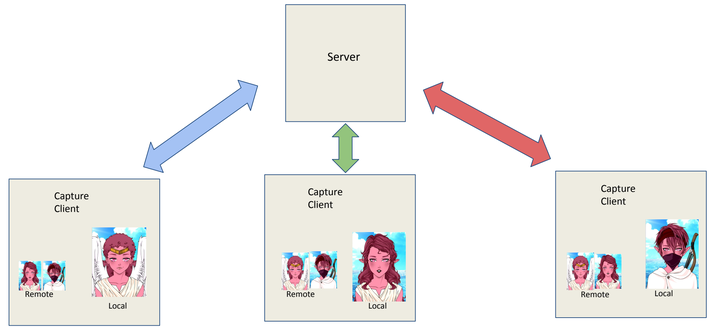


Figure 7. Network Clients communicate via a server application

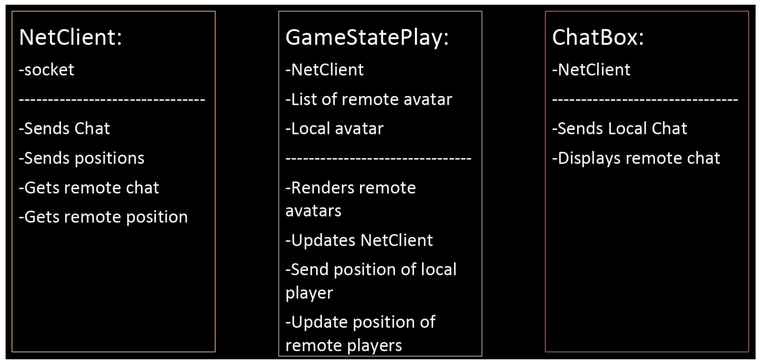


Figure 8. Network implementation. The Netclient manages a socket for communication with the server. Game state updates the each client with the network information from external clients. Chatbox displays the netclient chats in a chat window.

Figure 9. Example of two Capture clients communication where the chat of one client is sent and relayed to another client hosted by a server.