Game AI Final Project Report

Ian Buckley, Chris Donlan, Lanssie Ma

July 27, 2016

Abstract

The Squad Companions project was completed as the capstone project for the course. The primary objectives of implementing player navigation, effective supporting AI, dynamic formations, and contextual player-AI interaction. Navigation was addressed by modifying the core files. Behavior trees were the primary means of implementing the companion AI with the contextual player interaction. Ultimately, the project was a success, and the grand objective of supporting the player was achieved.

1 Introduction

We have completed the Squad Companions project. The Squad Companions game involves the player and 3 AI companion agents assaulting an enemy base while fending off enemy agents and attacks. The 3 agents include two Hero class agents and one Healer class. The completion of the project involved addressing a number of objectives. Modification of the core file was necessary to enable player navigation. Defining a dynamic formation was critical in supporting the player's attempts to win the game. Emphasis was placed on ensuring that the companion AI was independently effective so that players with different abilities can win. Lastly, using a simple context based interaction between the player and the companions was central in giving the player control over her squad.

2 Approach

The following sections will highlight the approach used to achieve these objectives.

Player Navigation The core files originally supported only simultaneous moving and shooting by the player, which is frustrating and unusable for the player. Modification to the core files addressed this by implementing WASD movement for the player. Furthermore, the player is able to shoot in the direction of the cursor, and dodge in the direction of the directional key press. As seen in many recent games, it is still desirable to have the character move to a location indicated by a mouse click, so not only can the player move with WASD, she can right-click to move to a desired point while at any time, a directional key press can interrupt and override this movement.

Implementing these controls introduced issues with obstacle collisions in which the player was able to move through obstacles. This was corrected by ensuring that the four corners of the player character sprite were not able to enter obstacles.

Dynamic Formations A triangle shaped formation of the companions was defined relative to the player's position. The two companions face outwards to block bullets from striking the player and the healer. The healer stands between the player and the heroes. Because the formation is defined relative to the hero's position, when the hero moves, the companions move in formation. Companions are able to shoot and perform area-of-effect attacks while in formation, but are not allowed to dodge. To further support the player while under formation, the formation is dynamically updated to face the nearest enemy, which serves the dual purpose of protecting the player and improving companion accuracy. Furthermore, the tight formation allows the healer to quickly heal teammates. Additionally, the formation rotates to protect the player from the nearest enemy.

Effective Companion AI Behavior trees were used to enable the companions to effectively assault the enemy base and minions while supporting the player. The primitive behaviors used by the Hero class companions were Chase Enemy, Kill Enemy, Retreat to Healer, Retreat to Base, and Maintain Formation.

With each of these behaviors, opportunistic shooting, dodging, and area-of-effect attacks were enabled by including a function call for each in the primitive behaviors. For the Healer class companion, an addition Heal Teammate primitive was introduced. Behavior trees were found to be very effective in supporting the player's objectives.

For seamless, low-level interaction, a cover system was implemented so that the companions move in formation to the closest cover to the player. Cover was defined with respect to the stationary enemy shooters (towers and base), and safe cover locations were defined from the occluded areas behind obstacles. using this cover system, the player is able to move through the obstacle field without having to designate which cover the companions should go to. To highlight the cover system (and a use of hierarchy in order to solve a performance bottleneck), a more in depth discussion of the cover system is included.

Formation And Cover The formation "snaps" to cover nodes calculated during game start-up when cover nodes are within 100 (subject to change) units of the player character. The final list of cover nodes is large (sometimes very large, depending on settings). In order to avoid any problems with speed reductions during the sorting, the snap is effected by a hierarchical sort.

In order to calculate the nodes, first the shadows are calculated from the stationary shooters using a polar coordinate transformation. Each stationary shooter has its own particular set of shadows. Each shadow is stored as a tuple containing the maximum "r" value, and the min and max theta values. Next, a grid of game points is generated (at moderately large step sizes to keep everything reasonable) and each point is checked relative to each of the stationary shooter's shadow parameters. If a point passes all of the tests, it is accepted as a cover node attached to the particular obstacle casting the shadow.

After that, the hierarchical sort is set up: a list of shadow centroids is calculated, and each centroid references a list of cover nodes for each particular obstacle. In order to calculate the nearest cover nodes, the game sorts the centroids, then the corresponding list of cover nodes, and sends the rearmost companion to the third nearest node. The other companions' destinations are then calculated relative to the selected cover node. As soon as the hero moves far enough away, the formation will re-position itself either around the hero, or in the next cover area.

Healer Capabilities The healer operates under a different spec for behavior in regards to dealing with it's lower health and special ability. The healer's default operations include the standard formation and snapping to cover, while searching for companions to heal as long as they are in a close enough range to heal and have low enough health. Since the healer has very little health, we added a cap on the distance the healer can travel to heal it's team to avoid being killed itself. When the bark is heard, the healer looks for the Hero to heal as long as it's in a close enough distance, and has low enough health (50%). After healing, it will return back to position. If the bark is called and the healer does not deem the Hero in need to being healed, he will stay in his current position. The structure of the code is the Bark and Teammate Daemon both evaluate the same state variable for the Healer, and both daemons lead to the same find and heal methods. These methods both evaluate according to the player the Healer must heal, which is either the Hero or the Companion.

Contextual Companion Interaction To support the player's ability to control the squad, a key was designated as the bark key. By pressing the key, the context was determined based on a number of variables including formation state, player health, and distance to the player. For the Hero class companions, barking was used to toggle between maintaining a formation and assaulting the enemy. For the Healer class companion, player health and the distance to the player were used to determine whether the healer would heal the player. Bark daemons were used in the behavior trees to prune sub-trees to quickly switch between behaviors.

3 Discussion

The approach was found to be very effective in supporting the player's objective to destroy the enemy base. The most significant challenges were improving and modifying the core files to enable navigation by the player, identifying the coordinate transformation necessary to transform Cartesian and polar coordinates into the coordinates of the game world, and identifying the best way to implement barking. Despite these challenges, all of the objectives defined in the proposal and the project description were successfully addressed. A cover system was generated, contextual interaction was implemented, and the companion AI was created with the player's objectives in mind.

4 Conclusion

This project serves as a laudable capstone for the course. All of the main topics addressed in the homework assignments were featured, and the addition of the player tied the Game AI theme of improving the player experience into the course. Future improvements to the game could include procedurally generated maps, variable difficulty implemented by adding stronger and smarted enemies, and complex barks. Limitations imposed by PyGame have been addressed, and future iterations of the class should make use of these improvements.