AI Report

Game Design
The game in which this AI was implemented is a simple 2d grid based top down shooter. Players can pick up health and different weapons from points on the map. The player can aim with the mouse and shoot by clicking. They can move by using the WASD keys. A cone is projected in the direction the player is aiming indicating the accuracy of a potential shot. This cone increases in size if the player turns, moves or shoots (encouraging players to shoot sparingly in order to maintain accurate shots). The game mode is simply a free for all death-match. The human player must fight against a number of bots.

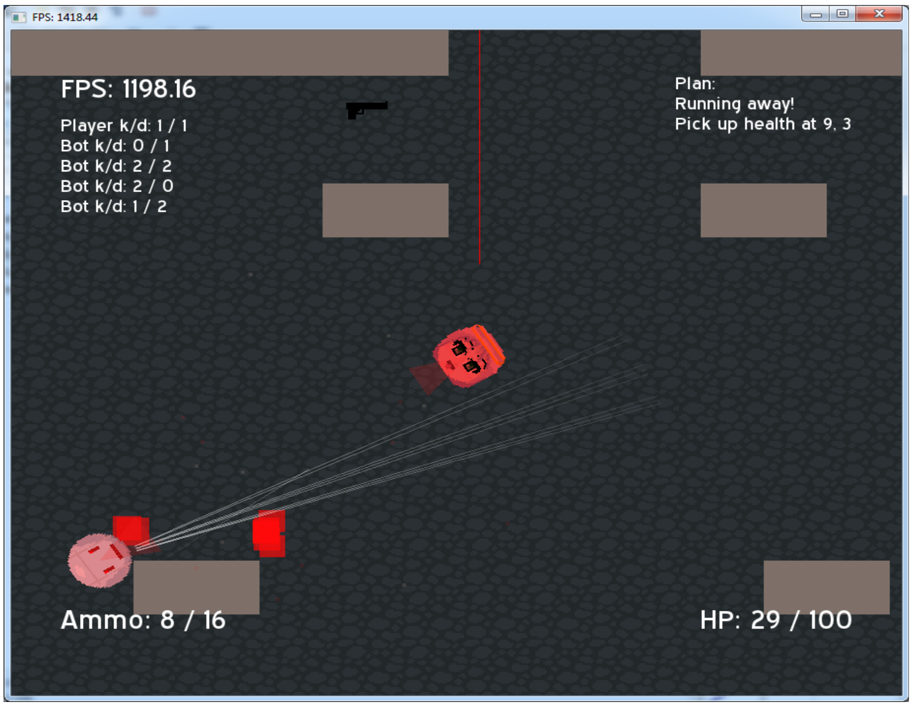
Maps can be created using the map editor "tiled" available at http://www.mapeditor.org/

AI Implementation
GOAP (Goal Oriented Action Planning)
An implementation of GOAP is used to allow the bots to plan for certain goals. A goal state is created, and a plan is made that will transform the current world state into the goal state. The plan is returned as a list of actions. The bot will then follow this plan until the current world state changes at which point the bot will formulate a new plan.

The algorithm itself uses A star path finding as a regressive search in order to find a suitable plan. Each node contains a possible action as well as a goal world state and a current world state. When the planner finds a node that has the same goal and current state, it has found a plan.

Each action contains prerequisite world state values and effect world state values. An action can only be carried out if the current state contains the prerequisites. For example, the Shoot action has a prerequisite that the player is holding a gun with ammo, and has the effect that the enemy dies.

The actions available to bots in the game are as follows:
- Shoot enemy
- Pick up weapon at (x, y)
- Pick up health at (x, y)
- Take cover at (x, y)
- Run away from enemy



FPS: 1198.16

Player k/d: 1 / 1
Bot k/d: 0 / 1
Bot k/d: 2 / 2
Bot k/d: 2 / 0
Bot k/d: 1 / 2

Plan:
Running away!
Pick up health at 9, 3

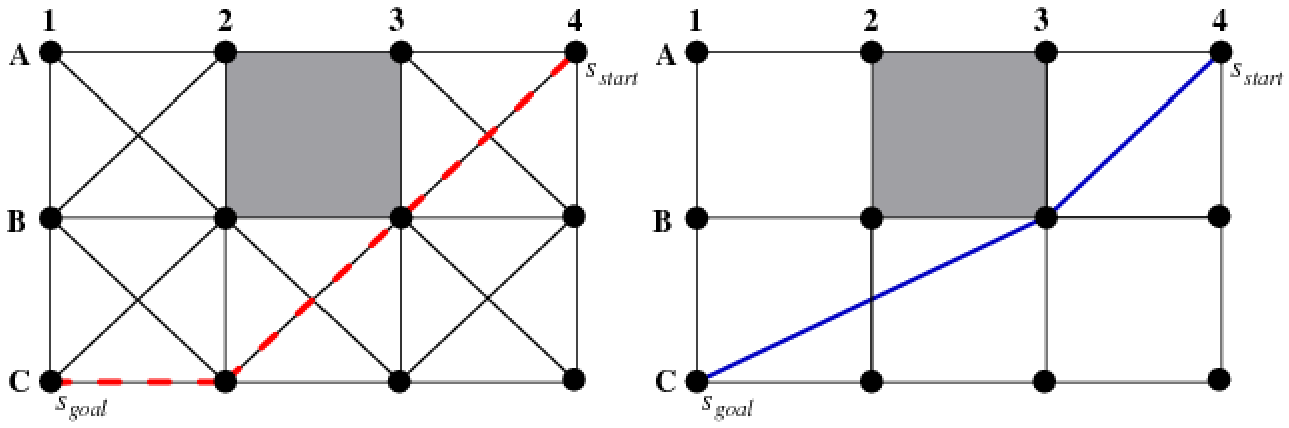
Ammo: 8 / 16

HP: 29 / 100

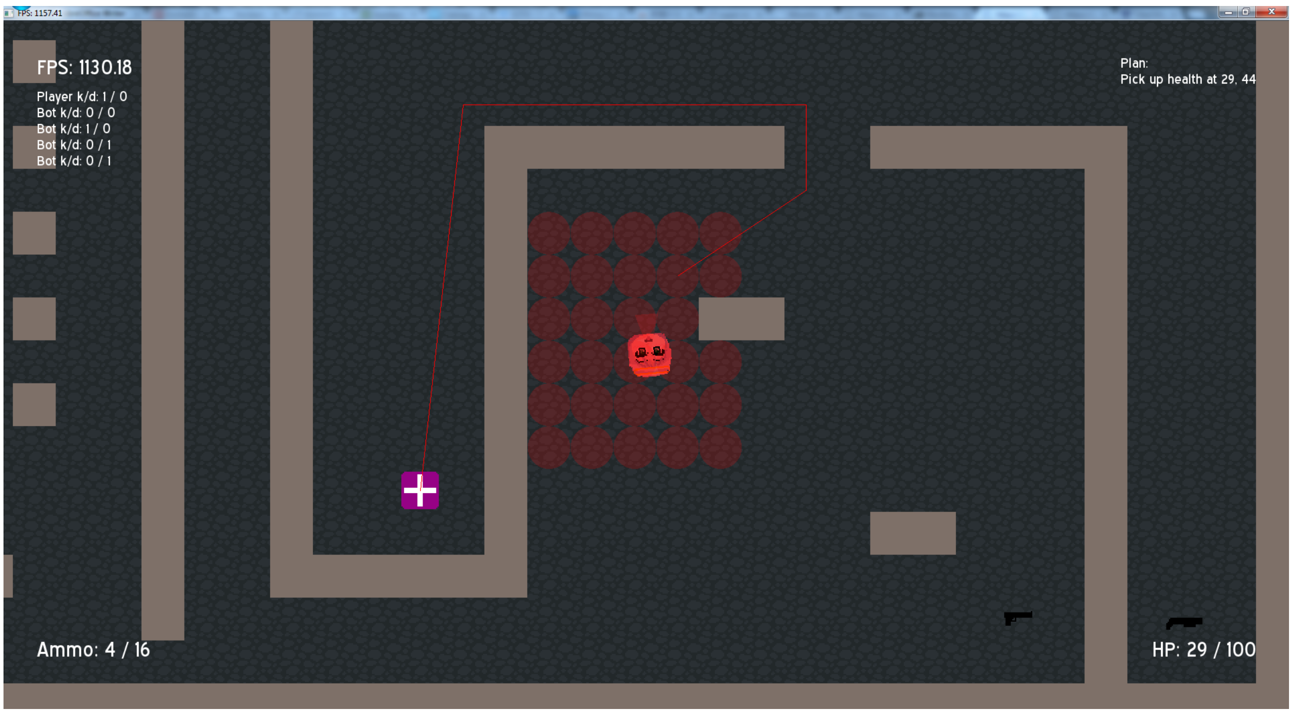
In the screenshot above, the current plan can be seen in the top right corner. As they are under fire, they are running away (since there is no cover), after they have completed that they will collect health.

The goal of the bot is changed depending on various values. When they are too close to each other, a goal of RUNNINGAWAY = TRUE is given in order for them to avoid being too close. Other times, ENEMYALIVE = FALSE and LOWHEALTH = FALSE are used as goals (LOWHEALTH = FALSE to avoid the bot being too suicidal).

Path Finding
Path navigation is done using an implementation of theta star. Theta star is a slight modification onto the A star algorithm that uses line of sight checks in order to allow it to find paths that do not necessarily stick to the two dimensional grid.

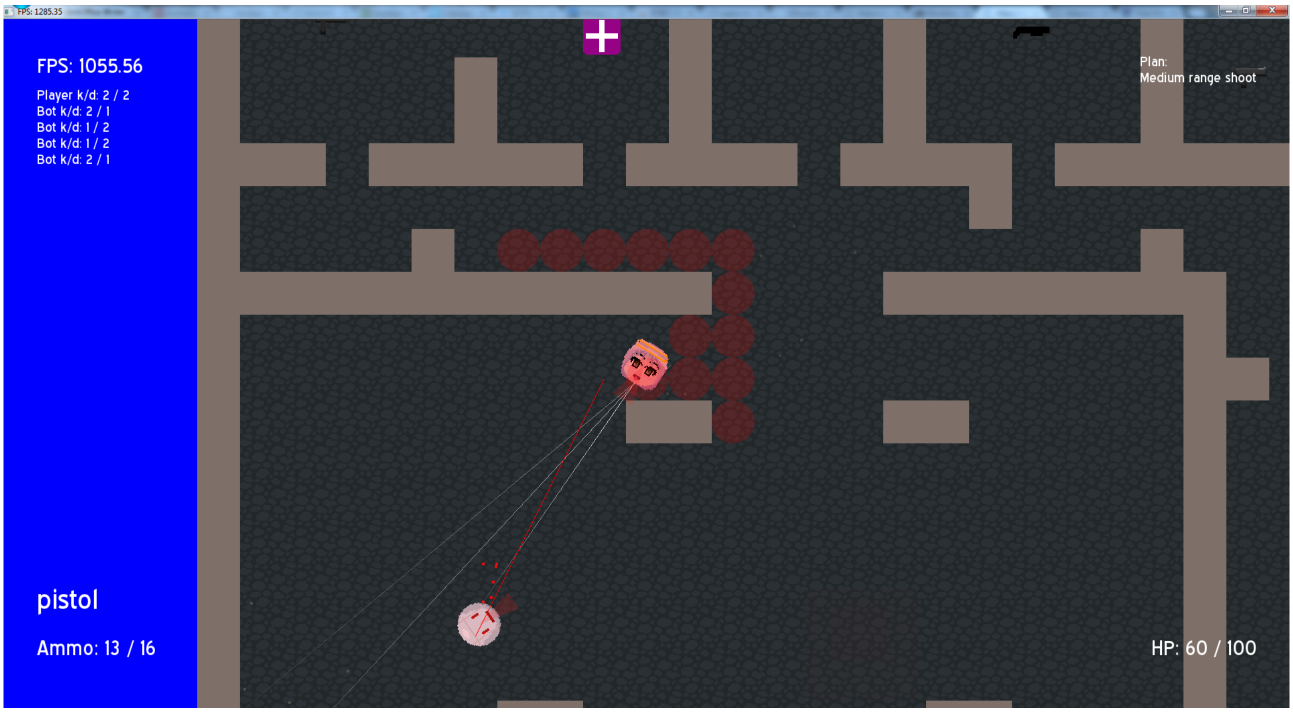


In the above image, A star path finding is on the left and theta star is on the right. As can be seen, theta star does not stay strictly on the grid, actually resulting in a shorter path. The added advantage to using this algorithm is that the movement shown by the bots is much more natural looking.



Here the a player is seen pathing towards a health pack.

Taking cover
The world state keeps track of points on the map that are potential cover. Rays are casted from the position of each player on the map to each point within a radius from the bot. If all of the rays cannot reach, then that position is added to the world state as a cover point.



Above, you can the red circles indicates cover spots. In these locations, the player is safe from being hit by bullets from other players in their current location.

During planning, actions to move to these cover points are added and can be incorporated into bots plans. The prerequisite for this action is that the player is under fire and the effect is that the player is not under fire.

The result of this is that bots will attempt to go to cover locations to take cover from enemy fire, shooting from cover.

Attempted features
I originally wanted to make use of a HTN planner (faster and more flexible than STRIPS) however, I could not find an implementation written in C++ and I do not know the field well enough to implement one myself. I decided to choose STRIPS instead, but I again could not find a C++ implementation. STRIPS is simpler so I made my own implementation.

I also wanted to look into threat analysis of other players in order for bots to choose the most appropriate enemy to engage, however due to a lack of time and how well a naïve "shoot at the closet" implementation worked I decided to leave it as it was.

Area of effect weapons were also planned. Weapons such as rocket launchers and grenades. However due to lack of time I decided to leave them out. Area of effect weapons would have required the AI to respond in a drastically different way.

Importance of AI
Before the take cover action was added, bots would walk directly towards the player and shoot. Fights are much more interesting when bots duck in and out of cover.

This game does not have multi-player support therefore, it is necessary to have AI to control bots for the player to play with. If this wasn't the case, the player would be left in a world with nothing to do other than pick up weapons and shoot walls.