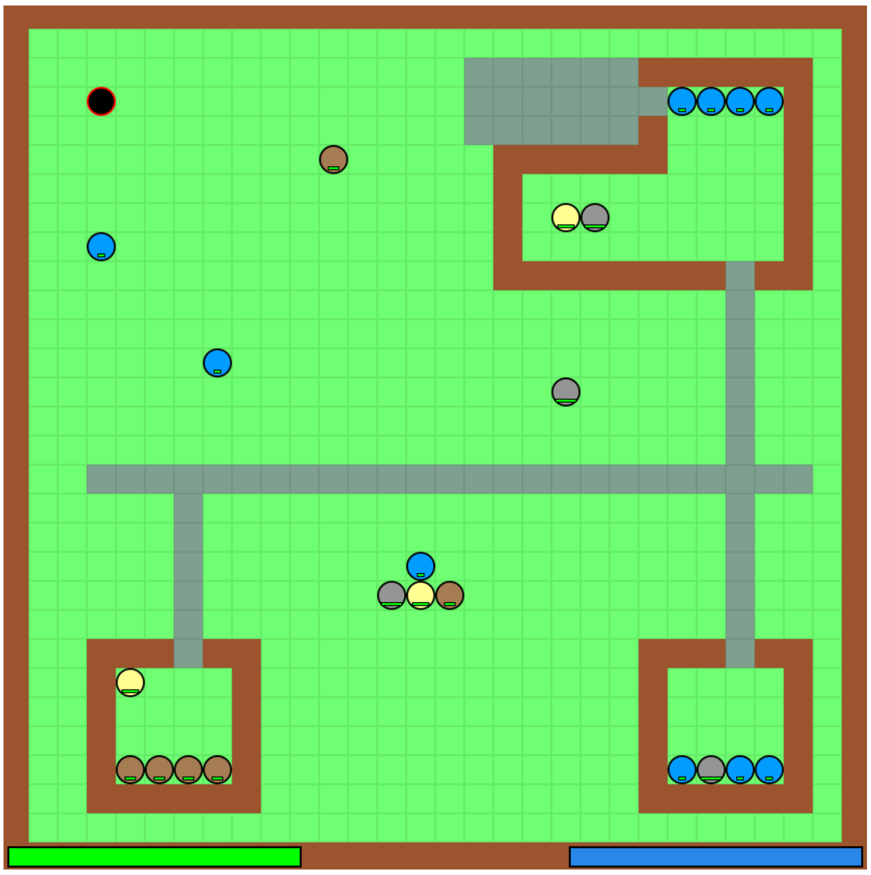
AI Game Programming

Assignment 2

By Robert Brede



Initial Specification

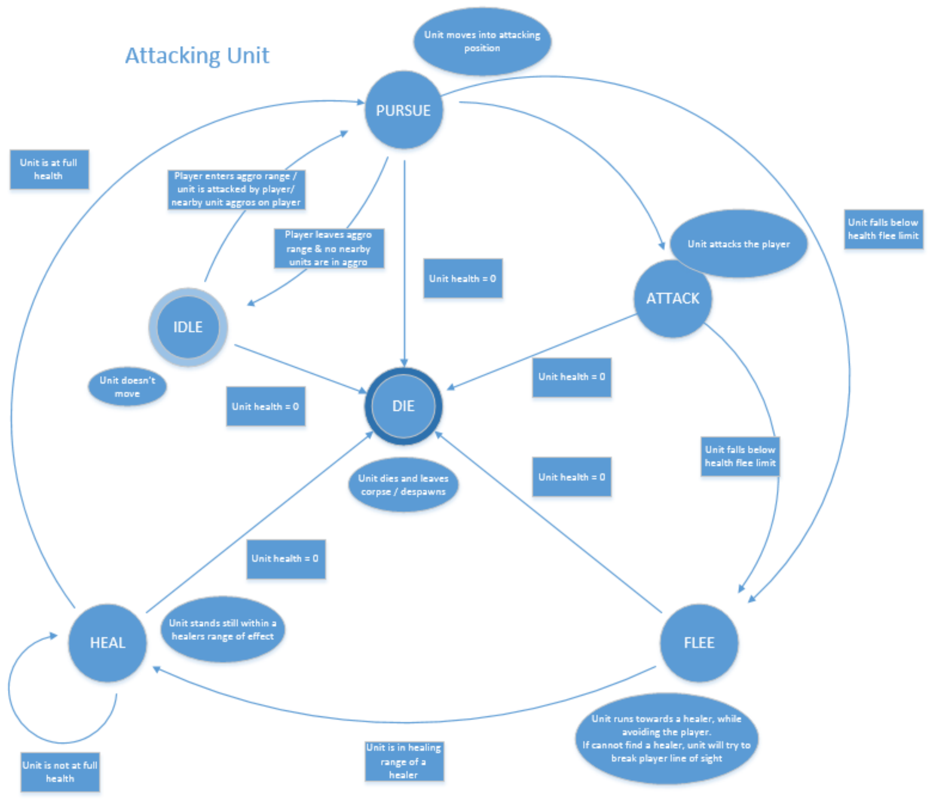
The initial specification of this project was to create a top-down grid based game, in which the player has to defeat varied types of enemies, each of which act differently. There would be 4 different enemy types: knight, archer, wizard and healer.

There would be 2 main types of enemy, attacker and healer. The knight, archer and wizard would be attacks, and the healer would be the only healer. The attacking classes would aim to kill the player, by attacking the player until their health becomes 0, while the healers would aim to keep the attackers alive, by restoring their health.

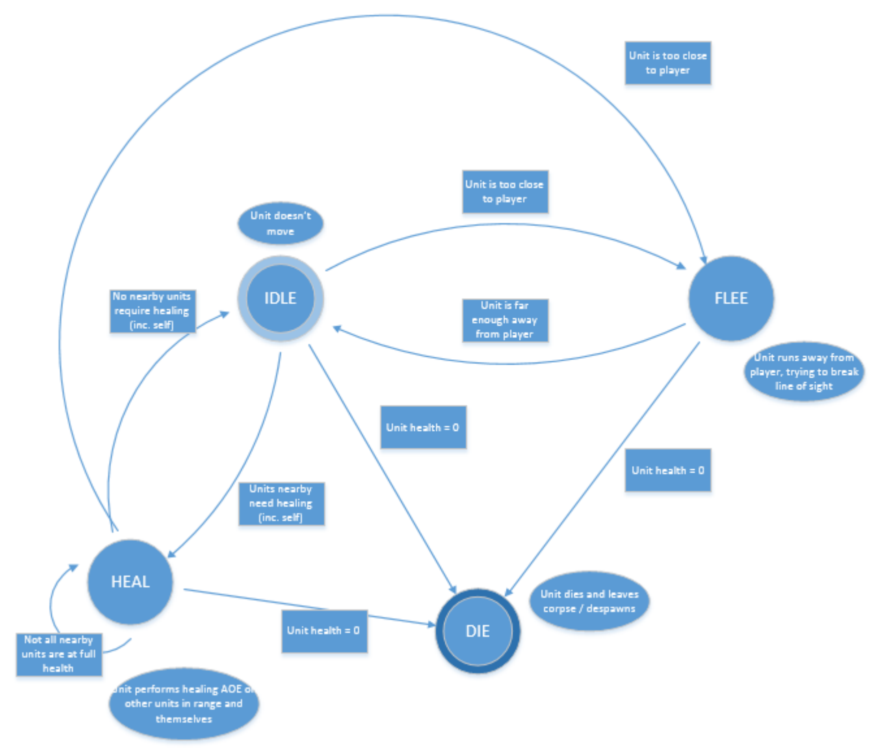
The attackers would start in an idle state and wait for the player to enter aggro range, or for another attacker in aggro range to become aggravated. Once aggravated, they would move to a pursue state, where they would move towards the player, until the player is in attack range. The knight would have to be next to the player to attack, whereas the archer and wizard could attack the player from further away. This would be balanced by giving archers and wizards less health. They would then move to an attack state, and attack the player, until the player has no health left. If at any point the attacker's health fell below 25% of its max health, it would move to a flee state, and move to the closest healer. Once in range of the healer, they would move to a wait state, where they would remain until their health had been fully restored. Once their health had been restored, they would move back to a pursue state. During any state, if their health fell to or below 0 health, they would move to a dead state, and remain there until they are deleted.

The healers would start in an idle state and remain stationary, waiting for an enemy to stand in their healing range while not on full health. They would the move to a healing state, and heal all enemies within their healing range, until all enemies had be fully healed. If at any point, the player moved too close to the healer, the healer would move to a flee state, and move away from the player, until they are far enough away from the player. It will then move back to the idle state. During any state, if their health fell to or below 0 health, they would move to a dead state, and remain there until they are deleted.

The player would have the ability to move and attack the enemies, with a variety of different attacks. The player would have health and mana. The mana would be used when certain attacks were performed, to limit the use of stronger attacks, and the player would die when their health reached 0. They would regenerate some mana over time, so mana consuming attacks can be used multiple times. The player would win when all enemies had been killed.


Attacking Unit
Unit moves into attacking position
PURSUE
Unit is at full health
Player enters aggro range / unit is attacked by player/ nearby unit aggros on player
Player leaves aggro range & no nearby units are in aggro
Unit health = 0
Unit attacks the player
Unit falls below health flee limit
ATTACK
IDLE
Unit doesn't move
Unit health = 0
DIE
Unit health = 0
Unit falls below health flee limit
Unit dies and leaves corpse / despawns
Unit health = 0
Unit health = 0
HEAL
Unit stands still within a healers range of effect
FLEE
Unit is not at full health
Unit is in healing range of a healer
Unit runs towards a healer, while avoiding the player.
If cannot find a healer, unit will try to break player line of sight

Healing Unit


Unit is too close to player
Unit is too close to player
Unit doesn't move
IDLE
FLEE
No nearby units require healing (inc. self)
Unit is far enough away from player
Unit runs away from player, trying to break line of sight
Unit health = 0
Unit health = 0
Units nearby need healing (inc. self)
HEAL
DIE
Unit dies and leaves corpse / despawns
Not all nearby units are at full health
Unit health = 0
Unit performs healing AOE on other units in range and themselves

Final Overview

The final has been set up to be turn based, so the player can move and then attack, and then each enemy will update their states and perform their state actions (i.e. Move, attack player, wait for healing etc.). This also gives the player more time to think about their actions, which gives the game a strategic aspect.

In the final game, there attackers and healers, although all attackers have to be next to the player to attack, so the archer and wizard are simply weaker versions of the knight. The attackers states are as planned, but the healers do not move, so are stationary. Attackers will also stay in an idle state if they cannot path to the player or a healer (as attackers cannot walk through each other, it can block all access to the player or a healer). It attempts to find a path, but if it cannot it will remain in an idle state, until a path can be found. If they cannot find a path to the player, and they are on less than 50% health, they will path towards a healer, rather than sit in idle and not move.

The healers will heal all agents around them every turn, and cannot move. They are therefore placed in locations far away from each other, and hidden from the player. They are set up to be able to move to the flee state, but the flee state itself has not been implemented.

State Manager

The agent's states are controlled by a state manager, which holds the current state the agent is in, and runs the update for each state. When the state update is called, it makes any changes in state at the start, so the correct action is performed each turn, rather than 1 turn later. If no change needs to be made, the state's corresponding action is performed. State changes are made by each state's update returning a pointer to a new state, or itself. If a pointer to itself is returned, it indicates that a state change was not necessary. If a pointer to a new state is returned, the state manager calls the end function of the previous state, and then the start function of the new state. This means audio/visual prompts can be run/made when agents change state, or preparations can be made for the update of that state (e.g. finding a path to the player).

States

Idle
Each attacker starts in the idle state. In this state, the main action is to stand still and do nothing. If the agent has < 50% health, and can find a path to a healer, they will move to the flee state. If the agent has > 50% health, and can find a path to the player, then they will move to the pursue state. If the agent has < 50% health, but cannot find a path to a healer, it will remain stationary. If the agent has > 50% health and cannot find a path to the player, it will also remain stationary and end its turn.

Pursue
In this state, the main action is to move towards the player. In the state start, it will generate a path to the player. (If at any point a path cannot be generated, the pathing algorithm will return a path to itself, as a flag to show there is no valid path) If the agent's health is < 25% they will move to the flee state. If the agent is next to the player (1 node adjacent to the player, including diagonally), they will move to the attack state. If the player has moved, or if something has invalidated the path (e.g. another agent has moved onto the path, therefore blocking it), a new path to the player will be generated. If no valid path to the player can be generated and the agent's health is < 50%, it will move to the flee state. If no valid path can be generated, and the agent's health is > 50%, they will move to the idle state. If the agent has generated a valid path to the player, it will move 1 node along the path, and then end its turn.

Attack
In this state, the main action is to deal damage to the player. If the agent has < 25% health, it will move to the flee state. If the player is outside the attacking range (of 1 node), the agent will move to the pursuing state. Otherwise the agent will deal 2 damage to the player and end its turn.

Flee
In this state, the main action is to move towards the nearest healer. In that state start, it will generate paths to each of the healers, and select the shortest one to move along. It will also store the nearest healer in a attribute called target healer. If no path to a healer could be found, there will not be a target healer. If there is a target healer, and the agent is next to that healer, it will move to the wait state. If there is no path to a healer, it will move to an idle state. If the agent has generated a valid path to a healer, it will move 1 node along that path, and end its turn.

Wait
In this state, the main action is to remain stationary until health is fully restored. If the agents has full health, it will move to the pursue state. Otherwise it will remain stationary and end its turn.

Die
This is the final state for an agent. During any other state, if the health of the agent falls to or below 0, it will move to the die state. The agent will remain in this state until it is deleted.

Player

The player has the ability to move and attack each turn. The player can move 1 node in any direction each turn. The player can then choose 1 of 5 attacks to perform. The normal attack deals damage 1 node away from the player in any direction. The swipe attack deals damage 1 node away from the player in any direction, and deals damage to the nodes which are adjacent to both the player and the attack targeted node. The line attack deals damage to each node up to 3 nodes away from the player, in 1 direction. The AOE attack does damage to each node adjacent to the player. The heal self attack restores some of the players health. Each of the attacks (apart from the normal attack) use up some of the player's mana. The player regenerates a small portion of the mana each turn. Both the move and the attack can be skipped.

Evaluation

The final program is missing some features of the initial specification. The ranged attack would require the agents to path towards the player, until they gained a line of sight and were close enough in range to the player to attack. The healer would require an algorithm to find the path which would take the agent the furthest from the player, until the player was out of the flee range. A Star epsilon path finding could also be implemented, so the agents who are fleeing attempt to avoid the player when pathing to a healer.

The final program being implemented as a turn based game allows the player to play more strategically, and have more time to think about each move. It also allows the player to see each of the agents move individually. The health bars allow the player to target weaker enemies, but the attacks of both player and enemy have no animations, meaning that it can be difficult to tell when the player has been attacked.