FIT3094 - Assignment 1 - Documentation

Pavle Spasic 29691931

Outline of the Algorithms

Each agent begins its life with selecting a Food to target, this is done by going through and finding the food that is the furthest away whilst still with a maximum distance.

The distance to a food is simply the size of a vector from the agents current location and where the food is.
After getting a reference to the best food for the Agent to go too, simple math is used get the location of the grid, and a reference to the grid is returned using the GetActorOfClass to get the GridNode from the Level Generator.

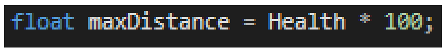
Once the Agent has a Food that it will go for, the agent that find a path to that grid spot through the A* algorithm. It begins by using the same as in the Food finding function to use the world position to obtain the grid position.
When the algorithm checks if the current node is the GoalNode, if it isn't and isn't the start node, it will check if the CurrentNode has a health pack, and will promptly avoid that tile if it does not to grab any extra Food on the way to the Goal Node.
Continuing with the A* algorithm, neighbours going in vertical and horizontal positions were added to be checked, only neighbours which were not in the closed list, the ignore list and those which were of type Open and Swamp could be checked. The Ignore List Is used when the agent comes across another agent and that one has priority over some nodes for collision avoidance. The rest of the GeneratePath Function is standard A* Algorithm.

Similar to the lab tasks but still different, nodes were set as a parent node so that a path could be made going backwards from the goal. The Connect Path function creates an array going from the goal to the start, and then adds them to the Path array for the movement.

Collisions with other agents are avoided by using a collision sphere that reaches approximately outside 1 grid spot away from them. Each tick the agent checks if there are any overlapping agents. If there are, it checks if they are going to the same spot on the grid, and it also notes which of the 2 have the smallest path.
If 2 agents are going into the same spot, the Agent with the smaller path adds some nodes where the other agent wants to go to the IgnoreNodes array and regenerates a path to the goal node.

Reasoning

Having an agent choose the longest food without dying allows for more food to be up at any time. If the agent can survive for an extra 20 seconds, another Food actor will spawn in the Map.



```
float maxDistance = Health * 100;
```

The max is the health times 100 as the distance between to nodes on the grid is 100, with the move speed also being 100, the agents came to move 1 spot on the grid before losing health. This Max distance will make sure that the agent doesn't go for food outside of its health range.

The A* Algorithm was chosen due to its ability to have weighted paths and its efficiency. Upon inspecting the code the GridNodes had a GetTravelCost function which acted as the H value for the function. This function also exposed that Swamps would be a another traversable GridType.

Avoiding Health packs during the A* algorithm in that point didn't require much extra work and simply fitted in. Avoiding them is key to not waste any health on the way to the goal node as then the Agents would overflow on Health.

Problems Encountered

I Struggled starting with the Agent code as i didn't understand how to get the grid positions for the Agents, although, with looking at the maths that was already there, I could switch some stuff around and then use GetActorOfClass as the LevelGenerator variables are Public.

The initial algorithm searched for its first food in the BeginPlay() function. A Problem i encountered was that the GetAllActorsOfClass function wasn't returning any food. I found moving the call to the food finding function to fix this problem as the food had not yet been spawned during the original calling.

I encountered a problem with overflow memory and using to much in one of my original renditions of the code. I assume the problem was that connect path was using a dynamic sized array that was constantly being resized when I removed Grid Nodes when i added them to the Path. The code was rewritten to remove the constant resizing of the Arrays.

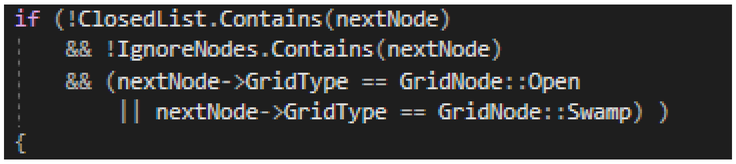
The original method that i chose to find food was to go for the furthest food if the agents were above 25% health, although i noticed that many times the agents were dying mid trips and never made it to the health packs. This was changed to have a estimated maximum distance that they could travel with their health.

An older version had the agents constantly loop looking for the same food and not going anywhere, this was fixed with the food finding function revamped.

I used a collision sphere to detect collisions with other Agents, it was not centered to begin with, although with some playing around i made it able to line up with the visible component.

Current Bugs

Sometimes the Agents can find a food and find a path but still not have a path. They get stuck doing nothing. The editor lags, unsure of cause.



```
if (!ClosedList.Contains(nextNode)
    && !IgnoreNodes.Contains(nextNode)
    && (nextNode->GridType == GridNode::Open
        || nextNode->GridType == GridNode::Swamp) )
{
```

In this section of the code, the editor can randomly crash saying that the 3rd line is a null reference, once again not sure how this is occurring.

If 2 agents get stuck in a tight area, the agent which has to recalculate constantly might end up in a wall somehow and sometimes gets stuck.