PATTERN MOVEMENT

Introduction

In this paper I'll explain how I went about implementing an AI system that finds a player via A* pathfinding, flocks enemies that come to close to one another and adds groups of enemies that patrol the map aka pattern movement.

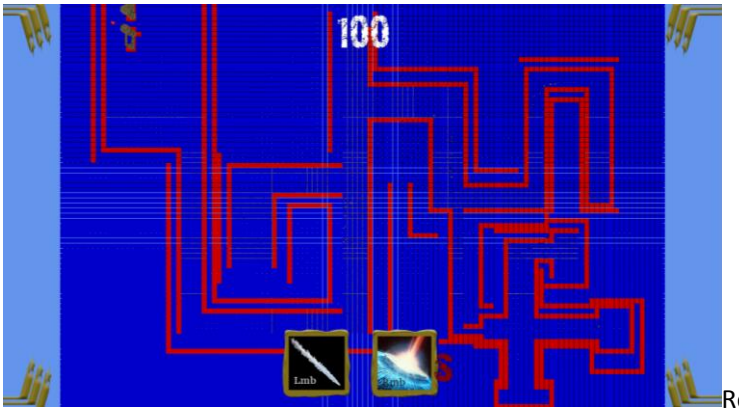
I implemented this inside the OverlordEngine which caused some difficulties that will also be explained.

A*

The first thing we start with is making sure our enemies can find our hero to attack, the way this will happen is when the player comes in "line of sight of the enemy" (The hero has a radius around him which will cause the hero to be seen when crossed) he'll go attack the hero.

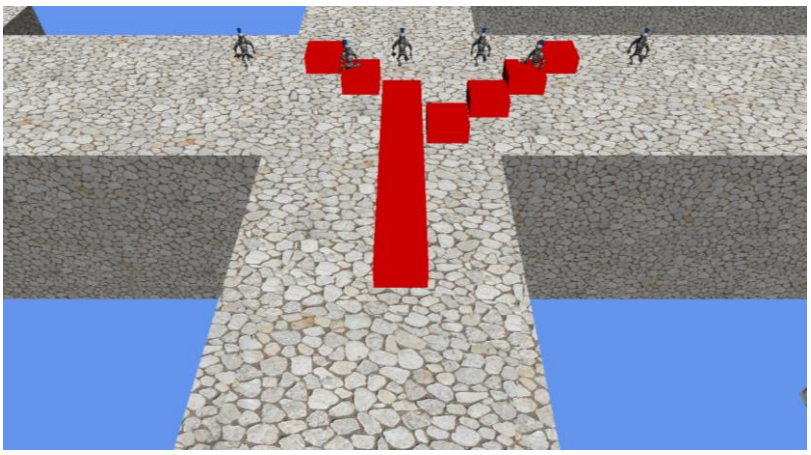
I will not explain the A* algorithm in this paper seeing as this is already documented in a lot of ways. I will however explain how I implemented the algorithm inside the engine.

The goal of A* is finding a path that doesn't cross unwalkable terain, so we'll have to detect whats walkable and what's not. This is done by placing triggers on every node(square) in the grid and if that trigger is triggered by our imobile terrain we set that node to "not walkable".



Red: not walkable node.

Now that we have our walkable level we let the algoritm do it's work by calculating the path between our 2 nodes. To make our enemy actually follow this path we needed to alter our character prefab by making the controls follow a given path. I did this by simply turning the controller in the direction of the next node in the path and moving it forward, when our enemy is inside the bounds of our targetted node we give the next targetted node as direction and so on till we arrive at our destination.

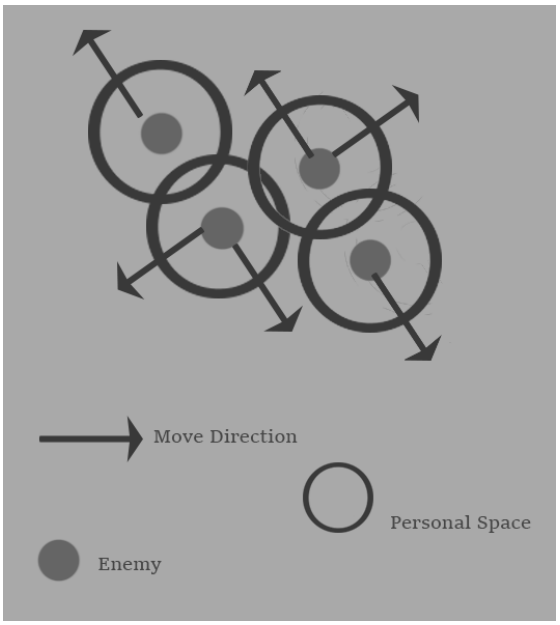


The path of the 2 enemys to the target.

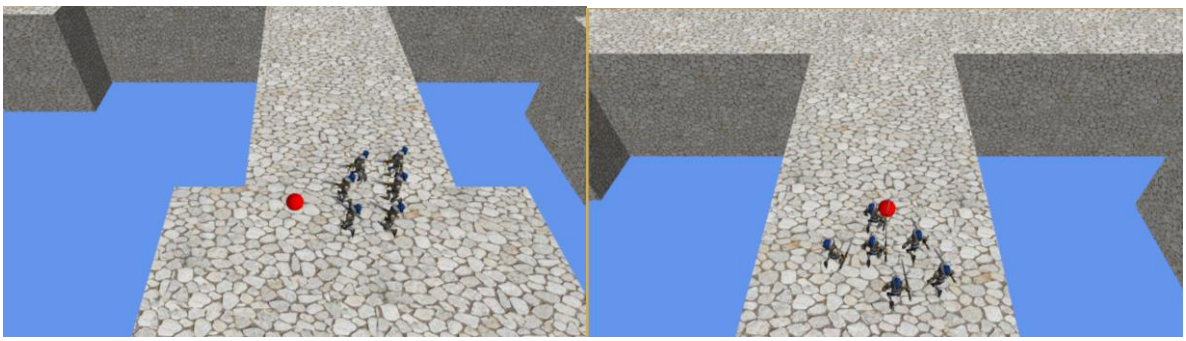
Flocking

This is when a group of enemies group at a certain location, if there's no flocking they will all walk inside each other or if they collide with each other they start doing unpredictable actions which we don't want. Flocking can be done in a lot of different ways it also depends on the way your AI's movement is handled.

Flocking in combination with A* was a bit tricky because we see every enemy as an individual even inside a patrol giving them each a path towards the goal this caused the use of a different way of flocking then we had previously seen. What I decided to do was see the enemies as a group with serious personal space issues, whenever they come to close to each other they look at the other and walk straight backwards till everyone has its personal space.



This caused our A* star to break because the space around each one won't fit them all inside our goal making them spin around each other, I fixed this by checking how many more steps they needed to take till reaching the goal and if this was small enough they were allowed to stop moving, of course after finding their personal space. As well we need to make sure the enemies don't walk into each other while in the patrol. To make sure of this we essentially do the exact same thing as before but with some conditions, there's a hospitality role where the one closest to the goal doesn't do de flocking actions but the other one goes out of his way.



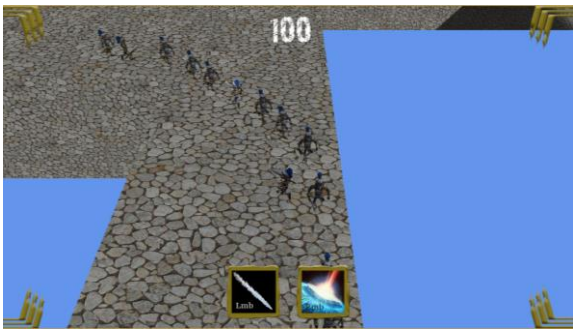
Patrolling

Or pattern movement is creating the illusion of intelligent behavior, because they seem like doing complex thought out manouvres. Because our movement is controlled by A* we need to do a tiled version of patrolling instead of the standard where every thick you command the object on where to go. In a tiled enviroment well give the enemy a some preset paths that follow each other causing the enemy to move this also makes it easy to interrupt the path when the hero's in range.

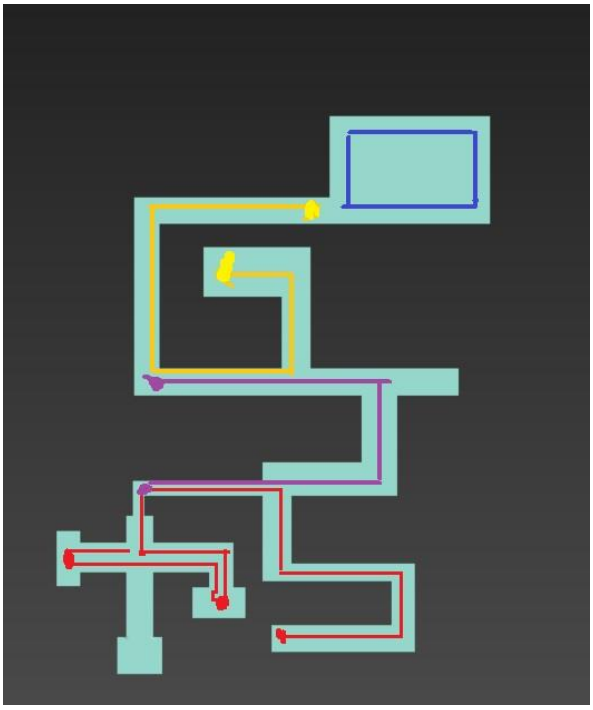
The way I did this is by calculating the path's per patrol group during initialize to avoid performance isues.

```
m_PathPatrol2[0] = FindPath(FindNodeFromWorldPos(XMFLOAT3(1239, 0, 1057)), FindNodeFromWorldPos(XMFLOAT3(1711, 0, 1533)), m_GridAstar);
m_PathPatrol2[1] = FindPath(FindNodeFromWorldPos(XMFLOAT3(623, 0, 1523)), FindNodeFromWorldPos(XMFLOAT3(1239, 0, 1057)), m_GridAstar);
m_PathPatrol2[2] = FindPath(FindNodeFromWorldPos(XMFLOAT3(1033, 0, 2046)), FindNodeFromWorldPos(XMFLOAT3(623, 0, 1523)), m_GridAstar);
m_PathPatrol2[3] = FindPath(FindNodeFromWorldPos(XMFLOAT3(1711, 0, 1533)), FindNodeFromWorldPos(XMFLOAT3(1033, 0, 2046)), m_GridAstar);

m_VecPatrolPaths.push_back(m_PathPatrol2);

for (std::list<Enemy*>::iterator it = m_VecPatrol2.begin(); it != m_VecPatrol2.end(); ++it)
{

    (*it)->GetTransform()->Translate(1711, 0, 1533);
    (*it)->SetPath(m_PathPatrol2[0]);
    m_PatrolPath2Step = 0;
}

m_VecPatrols.push_back(m_VecPatrol2);
m_VecPatrolPathSteps.push_back(m_PatrolPath2Step);
```

The calculated paths will be given to every enemy inside the patrol and then let the flocking do it's work which will cause them to form a formation that walks the path.



Ones the patrol has reached it's destination we give them the next path. When the hero comes into range the enemy's will leave the patrol and start going after the hero untill they die.



Here is our level with 4 different patrol groups who each have there own path.

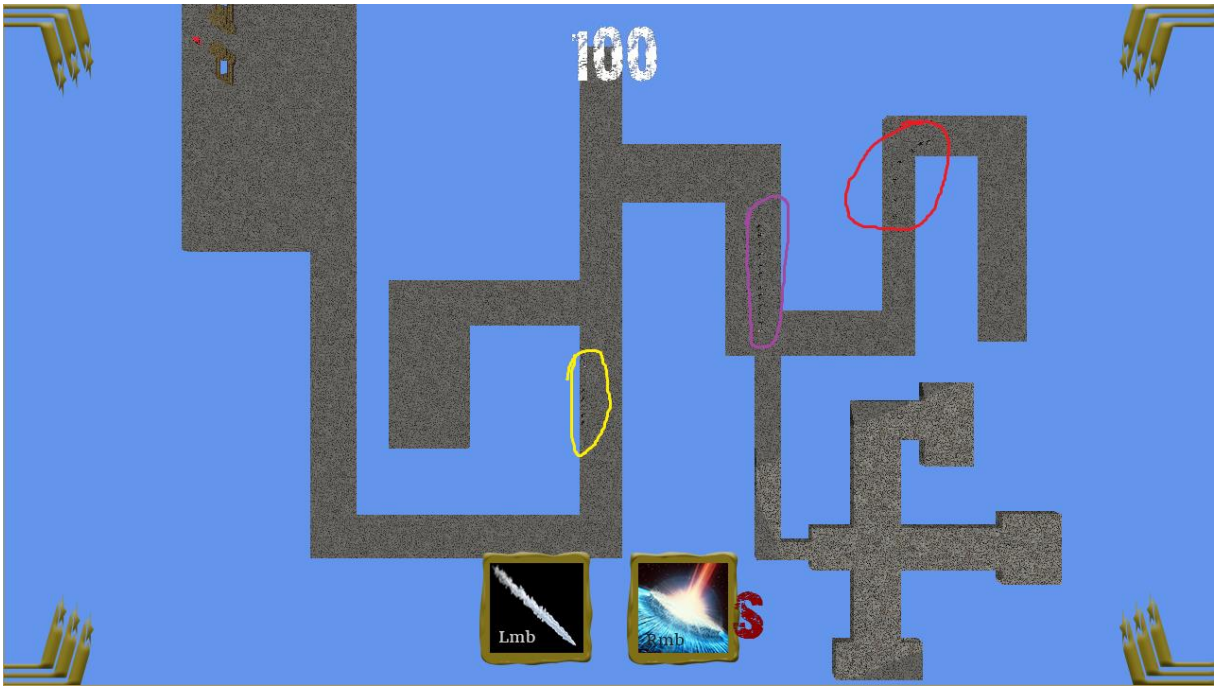
Each path will need to be set up differently because of A*:

For yellow and purple we'll only need to calculate 1 path between this being the path between the start and end point. Ones we reach the end point we just reverse our path and the enemy will be off again.

For red: we'll calculate 3 different paths because we give that patrol more end points. This gives us the ability to put some randomness inside of the patrol cause you can have the group decide which path to take based on variables or by RNG.

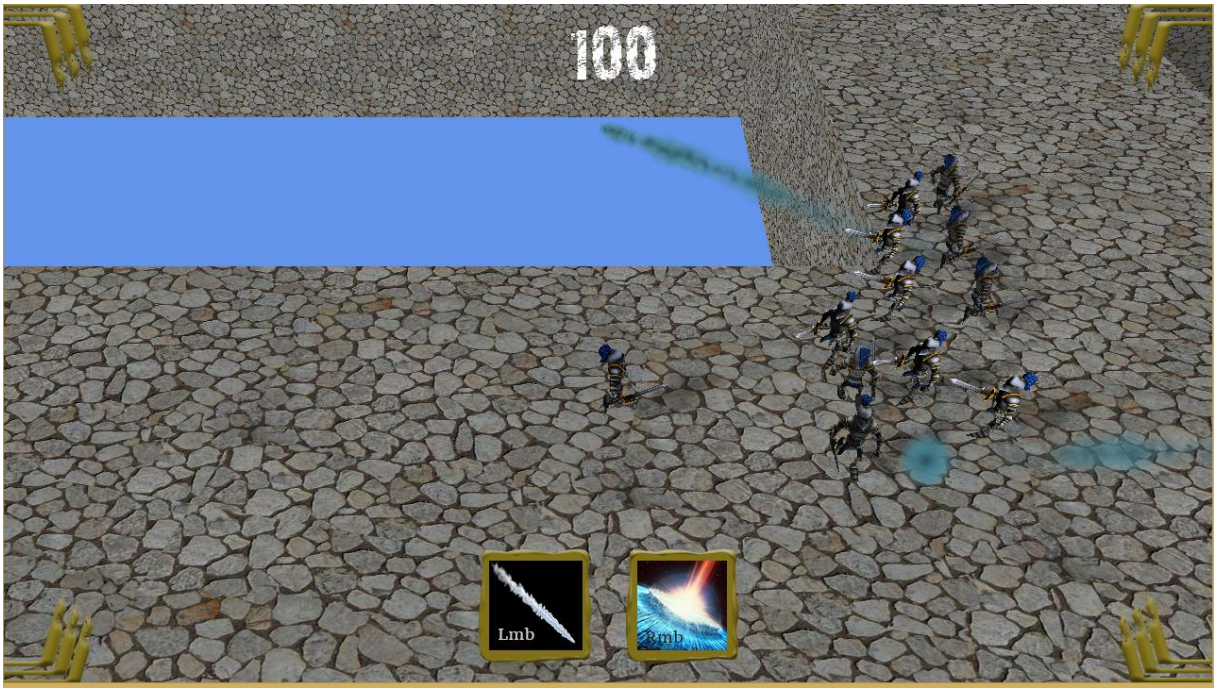
For blue: eventhough the path is a simple rectangel we'll need to calculate the 4 different paths(the sides) because of the way our A* implementation works (we allow diagonal movement).

This way we can add as much patrol groups as we wan't and give them whatever path we want.



Conclusion

Putting this all together brings us to a simple game where the character's goal is to beat the the maze while stumbling on patrols. These patrols can either be defeated or evaded. Defeating them is quite straightforward, ones they see you they'll come for you en you'll have to defeat them using your abilities. To avoid them you'll need to stay out of there range and let them pass till you see a window of opportunity and hope you don't run in to them again.



Resources

AI for Game Developers by David M Bourg, Glenn Seemann
Specifically chapter 3. Pattern Movement and chapter 4.Flocking

https://www.youtube.com/watch?v=nhiFx28e7JY

https://www.youtube.com/watch?v=Bdu5_Q5QI2Q