2d Platformer AI

Abstract

A 2d platformer game was created. Then, an AI was created to be able to play that game, which means being able to find a way to the goal while interacting with the environment.

Game

One part of the project was developing a prototype for a 2d platformer in Unity, which contains typical elements of the genre, like different movements and interactions.

Movements:	Walk, Jump, Double Jump, Stomp, Bounce
Interactions:	Doors, Monsters, Teleporters, Spikes, Platforms

Also, the game supports that an AI can easily press virtual keys as input and has a way to let the AI read certain variables of an object to roughly have the same pre-conditions as a human would have.

AI

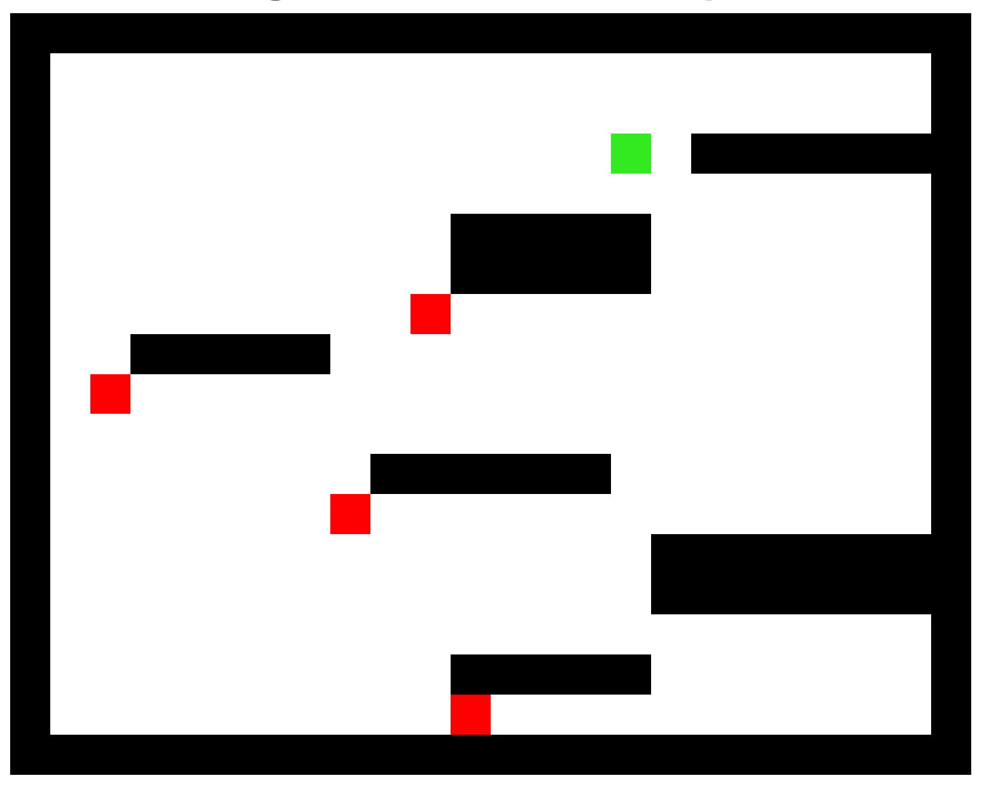
The AI chooses a goal, calculates a path to it and then follows that path.
To make things easier, it knows all the rules of the game to be able to model the future. Additionally it saves the collected level information to have an internal map of the world.

Goal planning

The AI has to reach the goal point to finish the game. Since it does not know at the beginning where this goal is, it takes unexplored points as a dummy goal until it eventually sees the correct one. A safety mechanism prevents the AI from trying to reach an unreachable point and therefore not exploring more of the map.

Pathfinding

The most important part is to calculate how to reach the selected goal. Common algorithms like a navmesh are not applicable here, because the objects not only have a position, but also velocity and internal values, which determine how they can move and interact with each other. This makes it necessary, that the AI quantises the objects to states and uses those as nodes for a node-based pathfinding algorithm like A*.
Improvements are made by not using a direct distance as heuristic, but rather calculating a heuristic upfront with a grid-based pathfinding to avoid obvious dead-ends.
Another improvement is to dynamically generate waypoints. Here, a waypoint is a sub-goal created in such a place that compared to a given starting point, it is always behind an obstacle. So if that point is reached, the obstacle has been passed. If a waypoint is reached, that waypoint is used as a new starting point for creating new waypoints.
Finding paths to the waypoints near the overall goal is preferred to those waypoint further away.



As an example on the right:
Green is the starting point and red are the calculated waypoints.

Pathfollowing

Although the AI has a very good model of the world, it is not perfect. Therefore, the movements of the AI have to be adjusted while it follows the calculated path. This can be as easy as moving one step more or less than calculated, but if a severe inequality between model and world occurs, some parts of the path have to be re-calculated.
The adjustments cannot be made as soon as a mistake occurs, because that way the AI would be stalled until the corrected way is calculated. Therefore, there is always a delay between detecting a mistake and adjusting to it.
The faster the AI can calculate the adjustment, the shorter the delay has to be; in other words that means, that the longer the AI takes for the calculations, the more the world gets out of sync with the internal model.

Next steps

As being part of my bachelor thesis, the AI will still be improved:
Instead of interacting with other objects mostly by accident, a mechanism to plan for those interactions will be added.
Also, the project will be used to test and compare different pathfinding algorithms in different scenarios.

Tobias Wolf - tw067