Summary of the dissertation Development of three AI techniques for 2D platform games

Martin Persson

January 12, 2006

D-level Dissertation (20p) 2005:11

The techniques developed in the dissertation ”Artificial intelligence in 2D platform games” will be summarized in this article. It will explain the purpose and the way each technique, described in part II of the dissertation, works in a very basic way.

The purpose of the dissertation itself was to develop three AI techniques that never were properly implemented when 2D platform games were the biggest genre in computer games, the term computer games is used because from a programmers point of view all computers and video game consoles are computers.

1 Line of sight

Line of sight is the technique that makes the enemies see the player. In old games the enemies saw the player all the time. The purpose is to make the enemies see the player on the same conditions as the player sees the enemies.

The line of sight technique works by checking if there are no objects in between the enemy AI agent and the player. It draws a straight line between the AI agent and the player. If there are no objects intercepting the line the enemy can see the player. So if the player is hiding between a large block the enemy cannot see the player. Because the player can only see what is displayed on the screen the player have a limit to the visual field. In the same way the enemy AI agents have maximum lent of the straight line. Only the enemies that have the player within that limit do the line of sight check for performance purposes. 40 enemies can have the player within their limits without the game lagging on a 1 GHz computer. Since the player have an area and is not a single dot it is possible to draw several lines between the enemy's eyes and the player one of several lines is possible to draw. A line is picked at random which solved the problem and gave the enemies a less mechanical feeling since they were not perfect, meaning that they did not necessarily notice the player the first frame that the player is within the enemy's line of sight.

2 Image recognition of the level

Image recognition of the level can be compared to image recognition in mainstream AI but in games the image that the AI agent is supposed to analyze is the level.

This technique works by checking for collision, with the blocks in the game, to locate the objects that the AI agent can interact with. The most basic technique is to check if there is a collision with any object, which the enemy AI agent can stand on, right in front of its feet. If there is a collision then it means that the enemy AI agent can continue without falling down but if there is no collision it means that the enemy will fall down if it continues forward without doing something. If the enemy has reached a gorge then a larger check is done to see if it is an object within the distance that it can jump. In this simple way of collision detection with the terrain blocks the AI agents can visually analyze the terrain just like the player uses his eyes.

The technique is only the idea of how it can be done but this technique have no real limit to what kind of information can be gained with it.

3 Pathfinding

Enemies need to know what path to take on the map. If they want to get from one point to one other they cannot just follow a straight line and hope that there is no wall in between the points that make the line. So a technique that helps them find their way is needed.

Pathfinding works by placing nodes on the map. These nodes are connected to each other so when an AI agent stands at one node it gains information from the node to know in what direction the next node is. All nodes have a list containing every other node on the map and which one of its connected nodes have the shortest path to the destination node. Compared to conventional pathfinding, shortest path algorithms or algorithm using heuristics, this technique is faster but it has restrictions to how many nodes the map can have. On the other hand the architecture of 2D platform games only requires a very small amount of nodes.