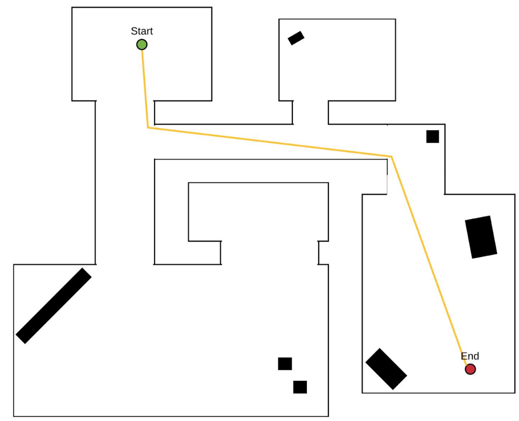
COMP3600 Final Project - Milestone 1
Jackson Kilrain-Mottram (u6940136)

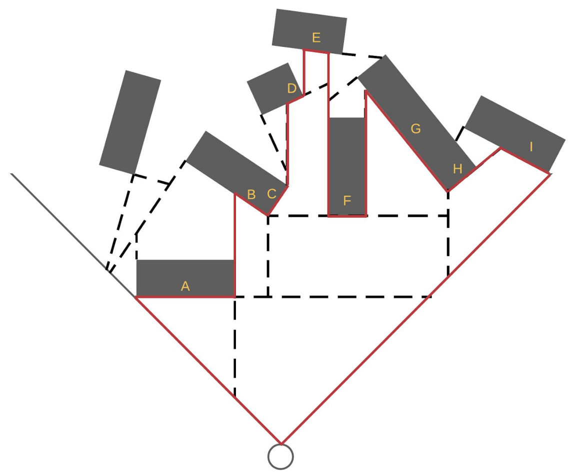
Overview

The idea is to use C++ (Linux) to create an algorithm that finds the minimum amount of polygons required to be rendered in order to go from the starting point to the finish point.



Here A* is used to determine the shortest path, with an additional heuristic applied to the continuously generated BSP tree for the polygons that need be rendered as the player progresses. Using the A* algorithm on this level design produces a path similar to the following (note the use of low-fidelity point-to-point deviation for the path).

BSP Tree (convex hulls)



BSP trees are used to find minimal sub-polygons of the render space to render, minimising the amount of arbitrary polygons to render at once. Here the calculation is limited to differential geometry, only re-evaluating the convexity of the polygons (to find the minimal amount of convex sub-polygons to render) when a particular polygon hash does not appear in the current list of minimal polygons. In the example, we can see there are 9 polygons (labelled from A to I) required to render the scene at that frame.

Hashing

Polygon objects contain two things: coordinates of corners and a texture reference. In order to reduce the overhead of duplicate data when storing polygons in a list during BSP evaluation, a hashing algorithm will be used to generate a hash based on the polygon object. The hash method will be polynomial rolling hashing on a string formatted as such: `“<texture name><raw coordinate values>”`

A* path finding

Utilising A* within the search will require some modification in order to have a heuristic that can evaluate the convexity of a BSP tree. As such the implementation will be dependent on an iteratively enumerated BSP tree based on player FOV. Here A* will compare polygon hashes with existing hashes, total sub-polygon count from the convex mesh of the BSP tree and the regular distance heuristic based on current location relevant to the start and end.