Computing Exact Arbitrary Clearance for Navigation Meshes

R. Oliva, A. Beacco & N. Pelechano
Universitat Politècnica de Catalunya

Character navigation in virtual environments is mostly handled by a combination of path planning with local movement algorithms. Clearance should be taken into consideration when choosing a path, but also when deciding the location of way points within portals. Previous work has considered clearance for path planning, but it is ignored when assigning way points within portals. Paths with clearance, do not necessarily guarantee that characters can walk through collision-free way points in portals. In this work we present a method for calculating clearance in navigation meshes (NavMeshes) consisting of convex cells of any type, as well as a novel method to calculate portals with clearance. We also introduce a new method to dynamically locate attractors over portals based on current trajectory, destination, and clearance.

### Introduction

- Clearance in NavMeshes is taken into account by most path planning algorithms, to guarantee that the character can walk between way points without colliding with the static environment.
- Previous work is either bounded to a specific amount of clearance, or only work with a specific type of NavMesh (e.g. triangular meshes, medial axis).
- Our work aims at dealing with an arbitrary amount of clearance, and also at assigning collision-free way points spread over the whole length of portals.
- We have integrated both our local movement algorithm and exact arbitrary clearance computation into NEOGEN [1].

### Clearance Value of a Cell

Given a cell $C$, an entry portal $P_1$, and an exit portal $P_3$, we classify the obstacle edges of the cell into edges to the left (leftString) and edges to the right (rightString) (respect to the crossing path from entry to exit portal). Cells do not need to be strictly convex [1].

The algorithm iterates over every notch (concave vertex) in leftString looking for the closest edge in rightString and calculating the clearance for the notch. The endpoints of each string are also treated as notches. The clearance value of the left string $cl_L$ is the minimum of the notches' clearance. Likewise, we calculate the clearance value of the right string $cl_R$. And finally, the clearance value of the described path is computed as follows:

$$cl(P_2, P_2) = \min(cl_L, cl_R)$$

### Computing Dynamic Way Points

As characters approach portals, way points are assigned dynamically either as their orthogonal projection over the portal or as the farthest away end point of the portal.

Local movement will steer characters towards different way points.

### Results & Conclusions

- We have presented general technique to compute paths with arbitrary clearance, as well as a method to pre-calculate portals with clearance and dynamic way points (average time = 4.5μs.) which provides natural looking trajectories.
- Tested 1000 path queries with $A^*$ in different scenarios and character sizes. Since clearance helps to prune the search, on average the results were:

<table>
<thead>
<tr>
<th>clearanace</th>
<th>0.5</th>
<th>1.0</th>
<th>1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>times faster</td>
<td>1.06</td>
<td>1.15</td>
<td>2.27</td>
</tr>
</tbody>
</table>

(Intel Core 2 Quad Q9300 @ 2.50GHz, 4GB of RAM).