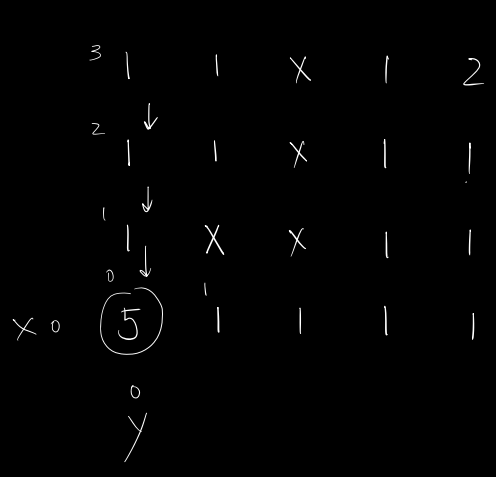
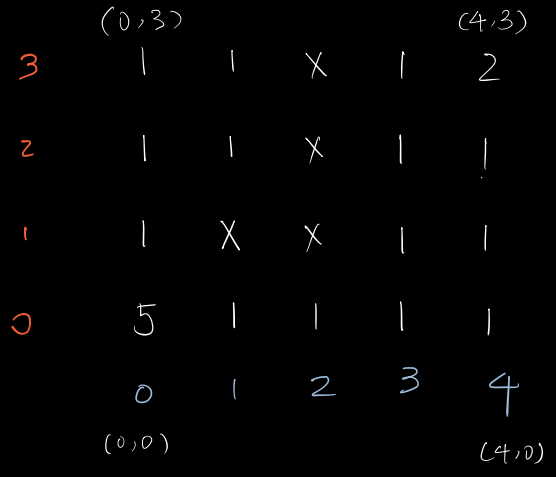
Set up VS 2019 AI HW

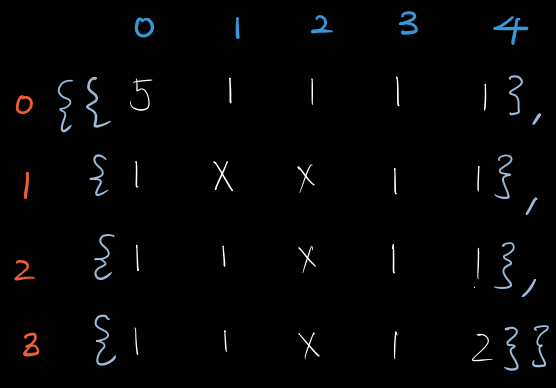
Finish AI Reading

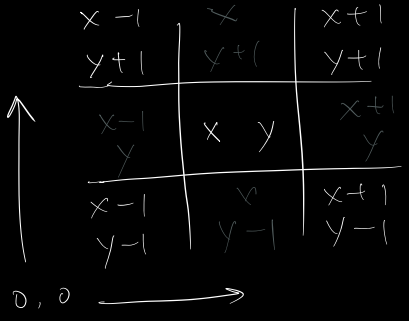
Find 2 other papers

Presentation outline









Precomputed Pathfinding for large world MMO servers

navigation mesh autogeneration

Component based hierarchical look up table

a group of connected polygons — at lowest layer
sublayer components at other layers

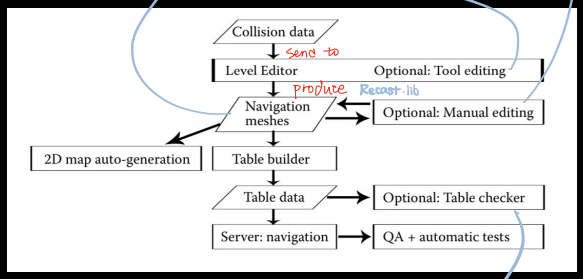
Navigation system

- simulated on servers, each server runs several maps' paths (<4 km² each)
- Too large, A* not fast enough — use precomputed path lookup table
  - paths must be found automatically
- ensure consistency between player / NPC available path
  - navigation mesh
  - hierarchical look up table to reduce memory/time cost
- The generation progress must try to minimize the number of polygons & match collision data as much as possible.
  - shrink mesh at the edge to match player radius so NPC could also move freely.
- allow falling meshes (knock-back) — connect those into navigation meshes so that NPC will be in valid state even be pushed

must be done at the very end of the work

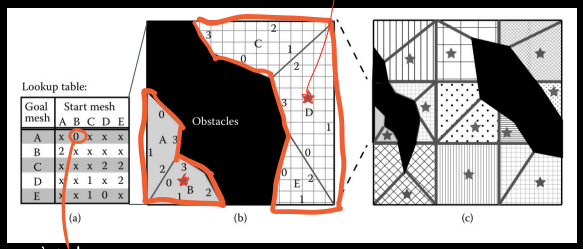
seed point — remove all non-connected polygons

mark special area implement door logic



The navigation process

examine collision data & algorithm



which direction to go

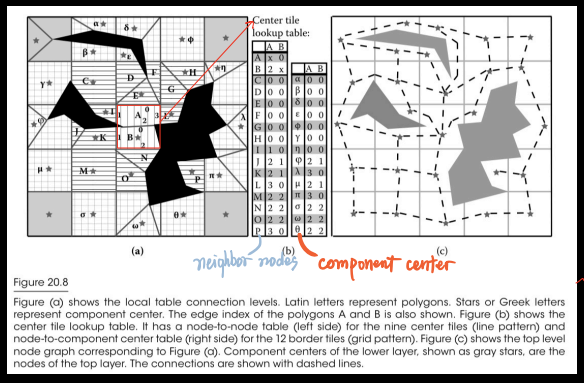
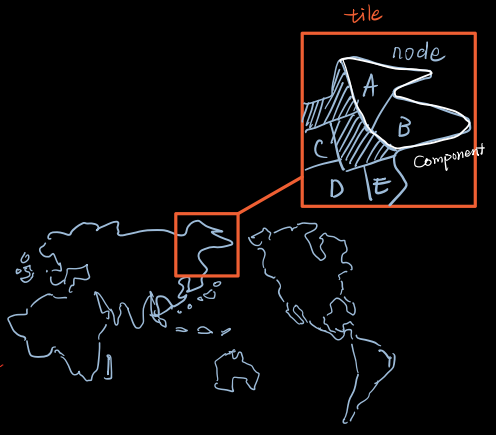


Figure 20.8

Figure (a) shows the local table connection levels. Latin letters represent polygons. Stars or Greek letters represent component center. The edge index of the polygons A and B is also shown. Figure (b) shows the center tile lookup table. It has a node-to-node table (left side) for the nine center tiles (line pattern) and node-to-component center table (right side) for the 12 border tiles (grid pattern). Figure (c) shows the top level node graph corresponding to Figure (a). Component centers of the lower layer, shown as gray stars, are the nodes of the top layer. The connections are shown with dashed lines.



reducing look-up table size

by optimizing falling meshes

① falling mesh only point to 1 ground pos

② dest on falling mesh directly re-direct to

③ do not store falling mesh info (20% save)

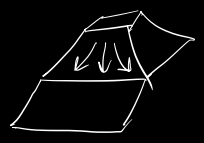


Table-building Progress

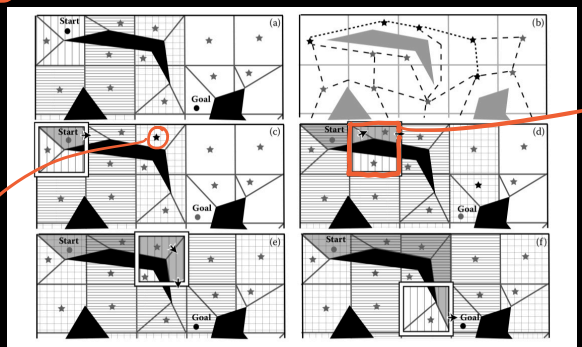


Figure 20.9

Example of pathfinding using precomputed data. Figure (a) shows the query start and goal point. Figure (b) shows the top level with the top level path. Figures (c,d,e,f) show the path planning process for the current tile (vertical line pattern) in each pathfinding iteration, as well as the edges chosen (black arrow). The resulting mesh path is shown in gray.

enter new tile node-to-node finding start

subgoal

the furthest comp center that still in local look-up center (within 2 moves)

II level hierarchy

↓ further optimization

III level hierarchy

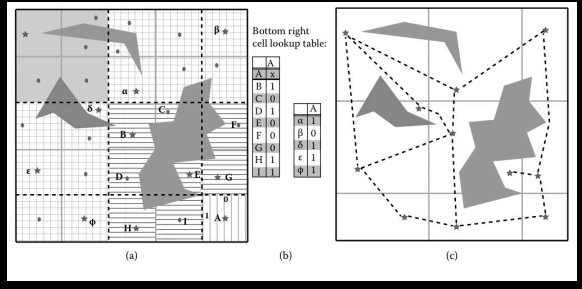
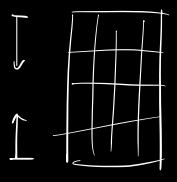


Figure 20.10

Figure (a) shows the second layer tile (dashed line) with dots for nodes and stars for component centers. The edge index of Node A is also shown. Figure (b) shows the bottom right tile lookup table. It has a node-to-node table (left side) for the center tiles (line pattern) and node-to-component center table (right side) for the border tiles (grid pattern). Figure (c) shows the top level node graph corresponding to Figure (a). Component centers of the lower layer, shown as gray stars, are the nodes of the top layer. The connections are shown with dashed lines.

component center

becomes upper level nodes



multi-level shrinks nodes / comps hence tables hence mem overheads

reversed pathfinding to avoid conflicting path

smoothed path

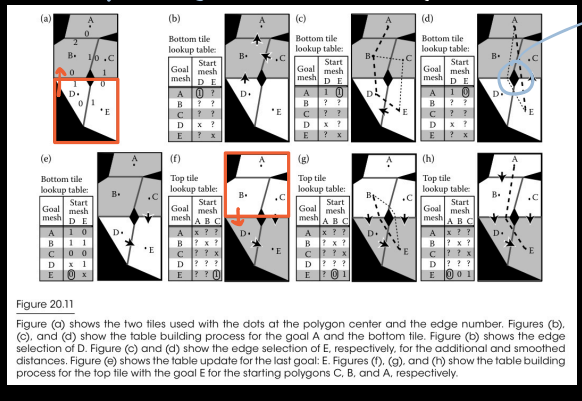
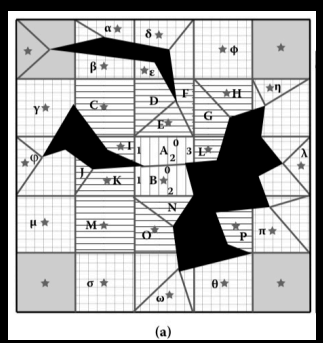


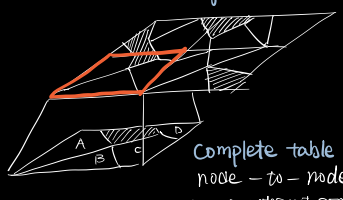
Figure 20.11

Figure (a) shows the two tiles used with the dots at the polygon center and the edge number. Figures (b), (c), and (d) show the table building process for the goal A and the bottom tile. Figure (b) shows the edge selection of D. Figure (c) and (d) show the edge selection of E, respectively, for the additional and smoothed distances. Figure (e) shows the table update for the last goal: E. Figures (f), (g), and (h) show the table building process for the top tile with the goal E for the starting polygons C, B, and A, respectively.



only patterned tiles, not nodes

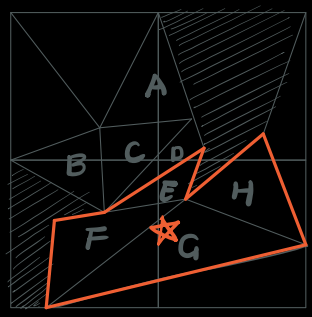
neighbors table



Complete table node-to-node (comp center not computed)

Listing 20.1. Pseudocode of the table building process, showing the main steps of the algorithm.

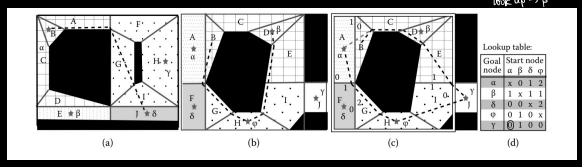
```
ComputeConnectivity();
FallingMeshSetup();
BuildMeshTable();
for(int i = 0; i<max_level; ++i) {
    ComponentComputation(i);
    SplitProblematicComponents(i);
    ComputeComponentCenter(i);
    if (i+1 != max_level) {
        AddHierarchy(i);
        BuildHierarchicalTable(i);
    }
}
for(int i = max_level-1; i>0; --i) {
    RemoveInvalidSubLink(i);
}
```

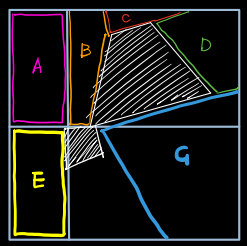


Subnode constraints

manually exclude to avoid confliction

infinite loop β → α → look up → β





global path β → φ

local path B(β) → A(α) → H(φ)

doesn't form valid path in global perspective

Inconsistencies

improper division result in detour

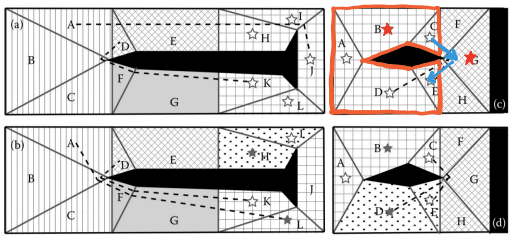
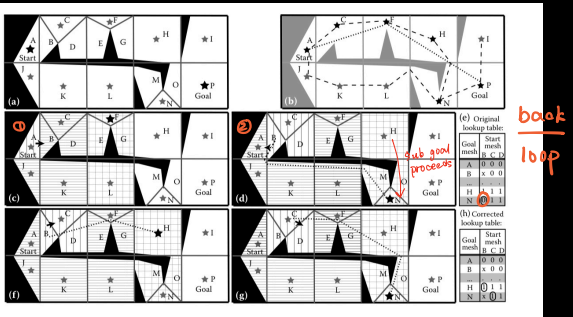


Figure 20.13

This figure shows how grid tile components are split. Areas with the same pattern correspond to the same component, while empty stars indicate potential component centers. Figures (a) and (b) show the opposite path inconsistency, while figures (c) and (d) show the convexity problem. Figures (a) and (c) show the components before and figures (b) and (d) show them after the split.



back loop

Figure 20.14

Invalid sublink. The pattern shows the table access method (line for node-node, grid for node-component). Dotted lines represent the shortest distances. Figure (a) shows the problem arising when the start point is within A, and the goal within P. Figure (b) shows the top layer nodes with links shown as a dashed line, the shortest path with black stars, and the smoothed distance is shown as a dotted line. Figure (c) shows how the subgoal F is used to go from A to B. Figure (d) shows that if the subgoal N is used from B, there is an inconsistency. Figure (f) shows that the inconsistency is solved if the subgoal H is used instead. Figure (g) shows that the path from C to N is valid. Figures (e) and (h) show the lookup table used in figures (d) and (e), respectively, before and after the sublink removal.