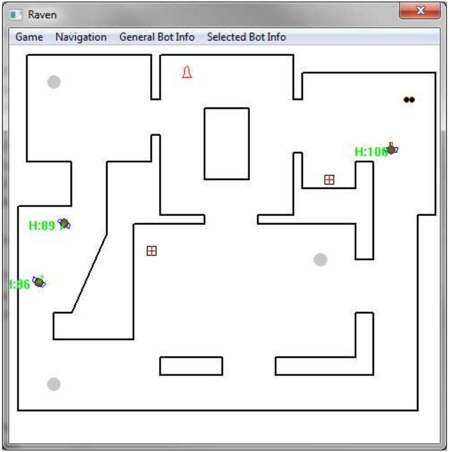
Raven Notes

CSCI 321

Based on *Programming Game AI by Example*, Buckland

May 27, 2015

Raven


Raven
Game
Navigation
General Bot Info
Selected Bot Info
H:100
H:89
H:36

Raven Game Class

- Map
- Bots
- Projectiles
- Path manager
- Grave markers

Raven Map Class

- Walls
- Trigger system
- Spawn points
- Doors
- Nav graph
- Space partition

Raven Weapons and Projectiles

- The Blaster
- The Shotgun
- The Rocket Launcher
- The Railgun

Trigger examples in games

- Step on a pressure plate
- Dead guard notifies other guards
- Shooting a gun activates a noise trigger
- Determine if a player has tried something three times
- Wounded enemies leave a trail of blood

Raven Triggers

- Respawning
- Givers
    - Weapon
    - Health
- Limited lifetime
    - Sound

AI design

- Weapon handling and movement independent
- Predict enemy's movement
- Choose appropriate weapon
- Select best weapon
- Aim slow weapons
- Select a single target from a group
- Perception
    - Visible
    - Noisy
- Perception memory
- Planning

Raven AI Overview

- Decision making
    - Attack
    - Find health
    - Chase target
- Movement
    - Steering
- Path planning

Perception: Bots too aware

Sensory omnipotence

- Eyes behind heads
- See you in the dark
- See you behind obstacles

Solution: better programming

Perception: Bots too unaware

Selective sensory nescience

- Set off a bomb behind them
- Leave a corpse seen by the next guard
- Forget about you once out of sight
- Forget about you if they turn their head

Solution: short-term memory

Weapon Handling

- Fuzzy logic for selection
- Aim using steering for slow weapons

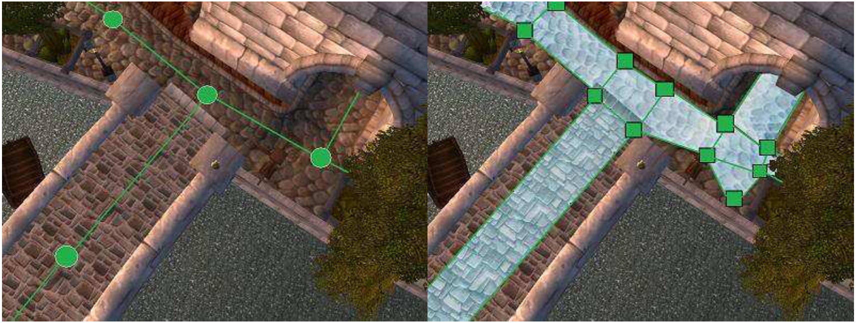
Weapon Handling Not Perfect

- Selection
- Aiming
- Fire rate
- Some bots *always* miss the first shot
- Lower skill when player's health is low

Updating

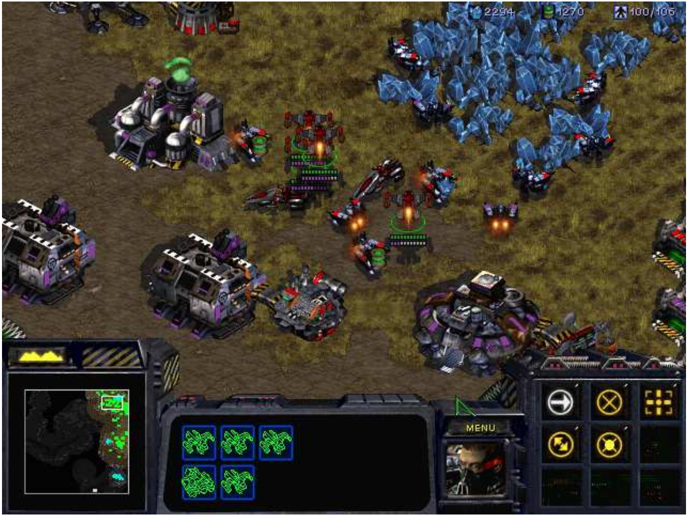
- Not everything every cycle:
    - Weapon selection
    - Visible opponent recognition
    - A^* path planning

Navigation Meshes

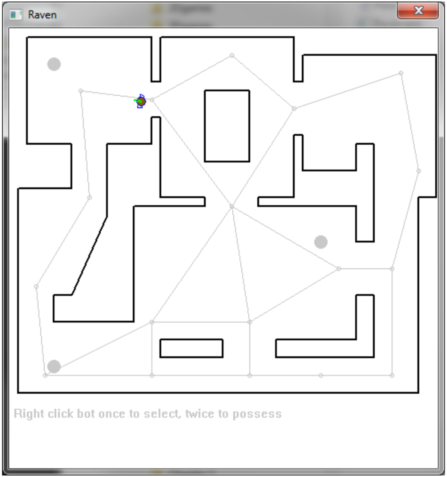


http://www.ai-blog.net/archives/000152.html

Tile Based Games



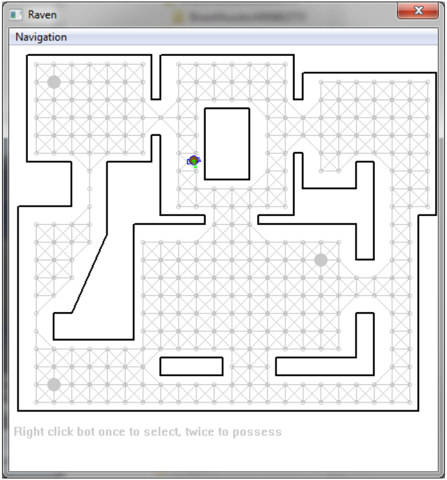
Sparse Graphs—Points of Visibility



Right click bot once to select, twice to possess

Use expanded geometry to generate automatically.

Fine Graph



Can be built automatically with flood fill.

Bot: move from x to y

- Algorithm:
    - Find A: closest graph node to x
    - Find B: closest graph node to y
    - Search for least cost from A to B
    - Move to A
    - Follow path
    - Move to B
- Problems:
    - Can be invisible points in coarse graph.
    - Coarse graph can result in poor paths.
    - Problems getting from A to x

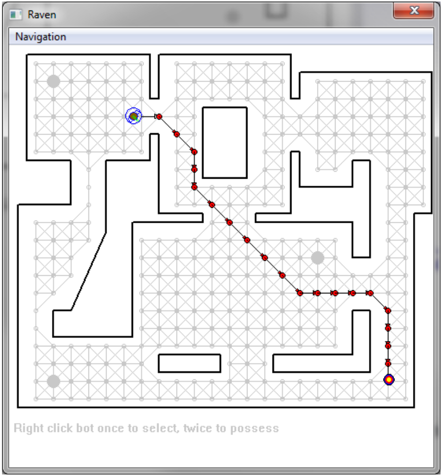
Spatial partitioning

- Need to find closest visible node.
- Partition space.
- Time goes from $O(n^2)$ in number of nodes to $O(d)$ in *density* of nodes, which is usually constant.

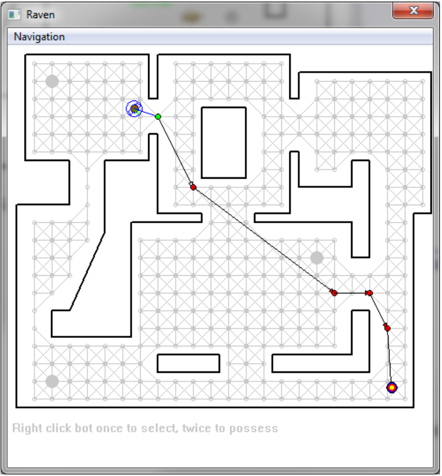
Path finding to an item type

- A^* good if we know the destination.
- Finding the shortest path to the nearest ammo (on shortest path)?
- Can use Euclidean distance and then A^*.
- Dijkstra's algorithm better if there are *many* items.

Path smoothing

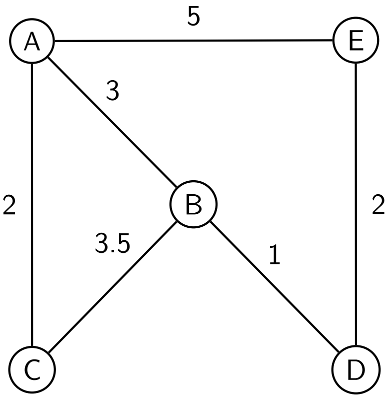


Right click bot once to select, twice to possess



Right click bot once to select, twice to possess

Precompute all shortest paths



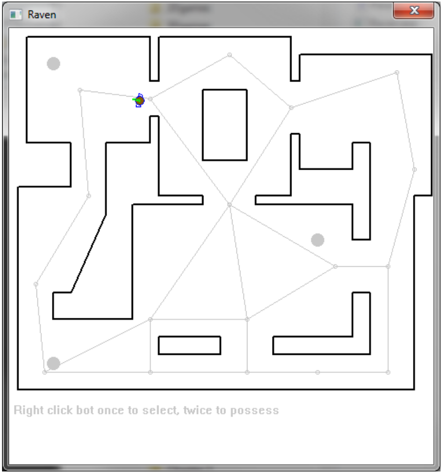
	A	B	C	D	E
A	A	B	C	B	E
B	A	B	C	D	D
C	A	B	C	B	B
D	B	B	B	D	E
E	A	D	D	D	E

Can be augmented with total path costs.

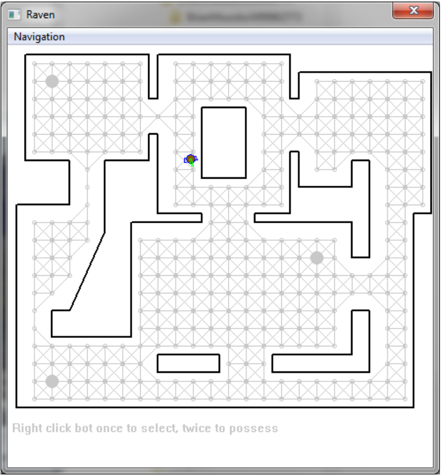
Time Sliced Path Search

- Don't do searches all at once.
- Break them up into slices.
- Bots must avoid twiddling thumbs:
    - **Seek** in meantime.
        - Must use path smoothing.
    - Use modified A^* to return partial results.

Hierarchical path finding

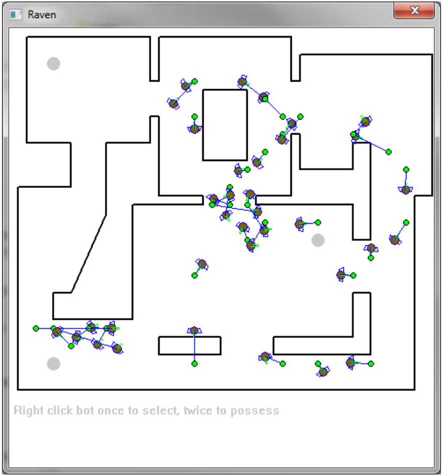


Right click bot once to select, twice to possess



Right click bot once to select, twice to possess

Bots getting stuck



Right click bot once to select, twice to possess