



AI for Animation and Combat

Based in part on material developed by
John McCloskey
Jeffrey Miller
Amish Prasad
Lars Linden
Van der Sterren, W.
Reed, C. and Geisler, B.
and Orkin, J.

(AI Game Programming Wisdom vols. 1 & 2)

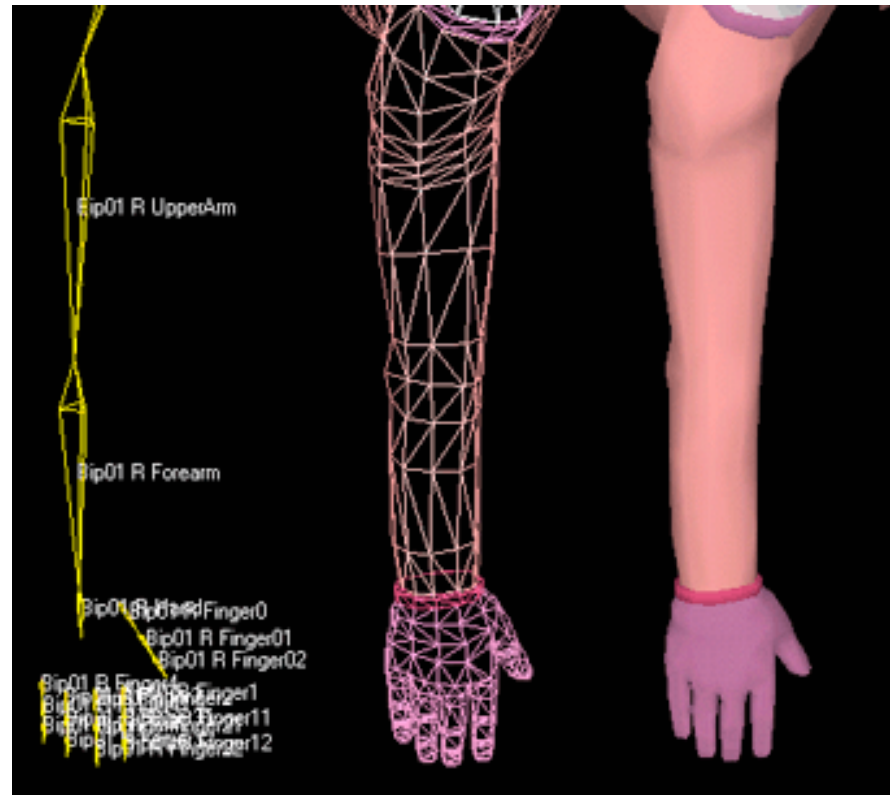


AI Components: Animation

- NPC **models** built by artists
 - Use tools such as "3D Studio Max" or "Maya"
- Models are are constructed from bones
- Bones are connected by articulated joints
- The skeletal system is covered by a mesh of textured polygons

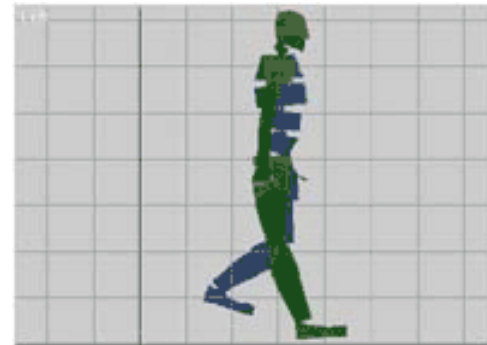
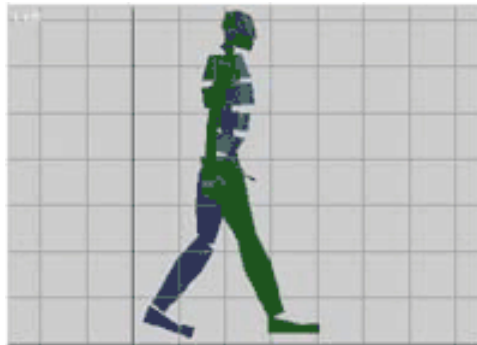
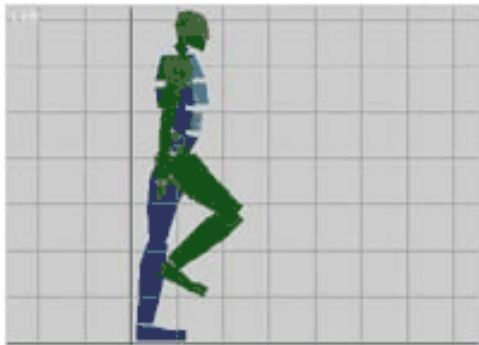
AI Components: Animation

- Example:



AI Components: Animation

- Animation sequences are generated by defining how joints should articulate through time
- Walking sequence:





AI Components: Animation

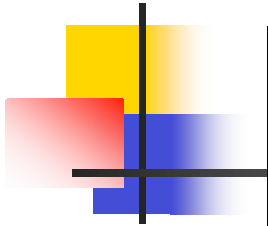
Animation sequences for a model are either:

- Hand generated by a computer animator
- Recorded from real human (or animal) movements and applied to a skeletal system (“motion capture”)

AI Components: Animation

- Motion Capture:





Tom Molet (EGCAS '96)



AI Components: Animation

Animation sequences tend to be:

- Motion primitives:
 - Run, Walk, Jump, Side-step, Climb
- Transitions
 - Start_Walk, Run_To_Jump, Jump_Land



AI Components: Animation

Some animation sequences only take control of part of the body:

- `wave_hello`
- `hand_signal_stop`
- `swing_ice_axe`



AI Components: Animation

- First step in A.I. is to select which animation sequence or sequences should be applied to a model
- Many influences:
 - Desired behavior chosen by decision system
 - What animation is currently playing
 - The current velocity and direction of the NPC
 - The terrain the NPC is standing on



AI Components: Animation

- Second step is to parameterize animations
 - Speed up or slow down animation
 - Slow walk, fast walk
 - Accelerate / decelerate stop and start of run
 - Slow run as approach sharp turn
 - Blend between animations
 - walk-to-run
 - 70% normal walk + 30% limp
 - Layer animations
 - Mix hand_wave on top of walk animation



AI Components: Animation

- Next might add selected Joint Control
 - Take control of particular joints
 - Either:
 - Ignore joint motion in pre-generated animation
 - Blend with pre-generated joint motion
 - Used for:
 - Head Turning
 - Looking at a particular object or location
 - Arm aiming
 - Point gun at a location



AI Components: Animation

- And finally, add inverse kinematics
 - Algorithmically determine the joint configuration required for an end-effector (hand or foot) to reach a particular location
 - Used for:
 - Keeping the feet on the ground on uneven terrain or when walking up stairs
 - Reaching hand out to open a door, pick up and object



Combat: Most Challenging

- Assessing the situation intelligently
 - Spatial reasoning
- Selecting and executing appropriate tactics
 - Camp, Joust, Circle of Death, Ambush, Flee and Ambush
- Perceptual modeling
- Weapons Combat



Combat: Spatial Reasoning

- 3D map geometry is difficult to parse
- Solution: Custom databases
 - Place hints throughout the world
 - Can be error-prone and inefficient
 - Do not handle dynamic obstacles



Perceptual Modeling

- Visual subsystem: seeing target
 - Distance to visual stimulus
 - Angle of stimulus relative to field of view
 - Line of sight calculations
- Auditory subsystem
 - Ensure that the AI can hear objects in the world
 - AI must interpret and prioritize sounds
- Tactile subsystem
 - Handles anything the AI can feel
 - Damage notifications and collision notifications



Weapon Combat

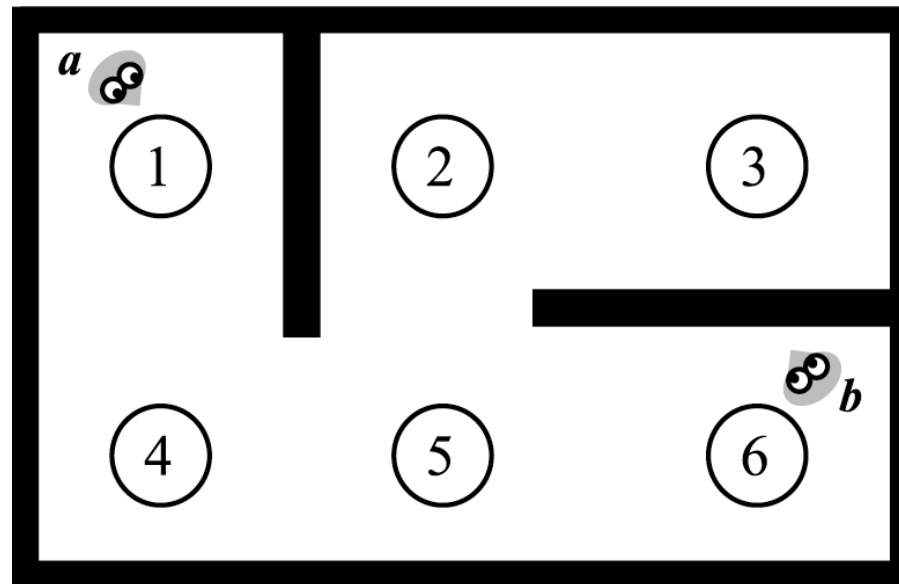
- Successful Attack Probability
 - Calculate value to represent the chance to hit, generate random number
 - If number is above to-hit value, try to miss target
- Factors:
 - AI skill, Range, Size, Relative Target Velocity, Visibility and Coverage
- Shoot and Miss
 - Pick a target coordinate outside the body
 - Place shot inside target's field of view




Tactical Analysis

- Level designers place waypoints in the environment for navigation
- Node graph contains information of connectivity between nodes for a map
- Waypoints can also be evaluated for their visibility
- Information can be used to make tactical decisions

Waypoint Analysis



 = Enemy



Waypoint Analysis

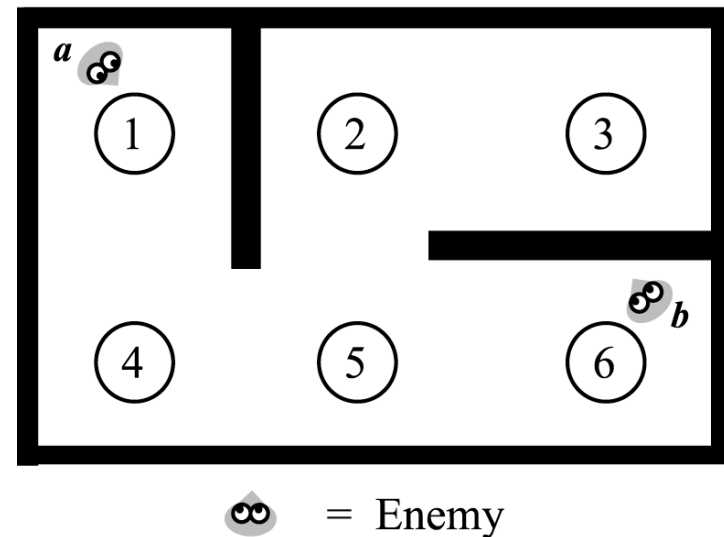
- Limited CPU time
- Decisions must be made quickly (as few CPU cycles as possible)
- Data must stored efficiently
- Store visibility data in a “bit-string” class

V_a

= visibility from node “a”

Waypoint Analysis

	Node					
	1	2	3	4	5	6
(a) $V_1 =$	1	0	0	1	0	0
$V_2 =$	0	1	1	0	1	0
$V_3 =$	0	1	1	0	0	0
$V_4 =$	1	0	0	1	1	1
$V_5 =$	0	1	0	1	1	1
(b) $V_6 =$	0	0	0	1	1	1





Waypoint Analysis

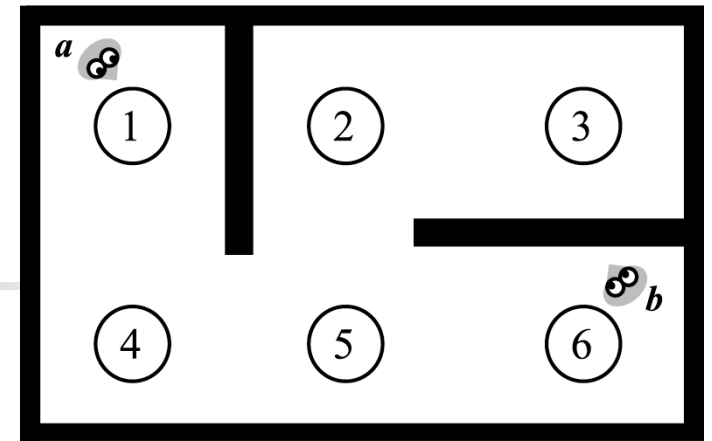
- Danger Nodes
 - Determined by “OR”ing the visibility of all enemy’s (k) nearest nodes


- Safe Nodes
 - Is its inverse

$$V := \bigcup_{j=0}^{j=k} V_j$$



Waypoint Analysis



 = Enemy

DANGER NODES:

$$V = V_a \cup V_b = 1 \ 0 \ 0 \ 1 \ 1 \ 1$$

Nodes 1, 4, 5 and 6 are dangerous

SAFE NODES:

$$\bar{V} = 0 \ 1 \ 1 \ 0 \ 0 \ 0$$

Nodes 2 and 3 are safe

Node

1 2 3 4 5 6

$$(a) V_1 = \begin{matrix} 1 & 0 & 0 & 1 & 0 & 0 \end{matrix}$$

$$V_2 = \begin{matrix} 0 & 1 & 1 & 0 & 1 & 0 \end{matrix}$$

$$V_3 = \begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 \end{matrix}$$

$$V_4 = \begin{matrix} 1 & 0 & 0 & 1 & 1 & 1 \end{matrix}$$

$$V_5 = \begin{matrix} 0 & 1 & 0 & 1 & 1 & 1 \end{matrix}$$

$$(b) V_6 = \begin{matrix} 0 & 0 & 0 & 1 & 1 & 1 \end{matrix}$$



Finding a Safe Attack Position

- While attacking a selected enemy, an NPC shouldn't expose itself to its other enemies
- A good attack position will:
 - Provide line-of-site (LOS) to the selected enemy
 - Provide cover from all other enemies



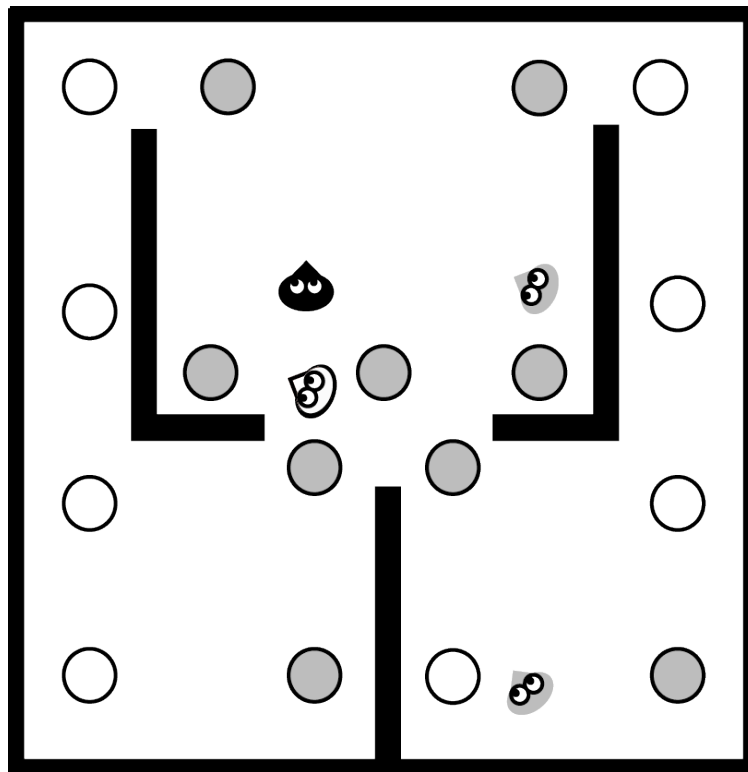
Finding a Safe Attack Position




- To find such locations, first find all nodes which have LOS to the selected enemy
- Call selected enemy "a"



V_a

Finding a Safe Attack Position



-  NPC
-  Selected Enemy
-  Other enemies

selected

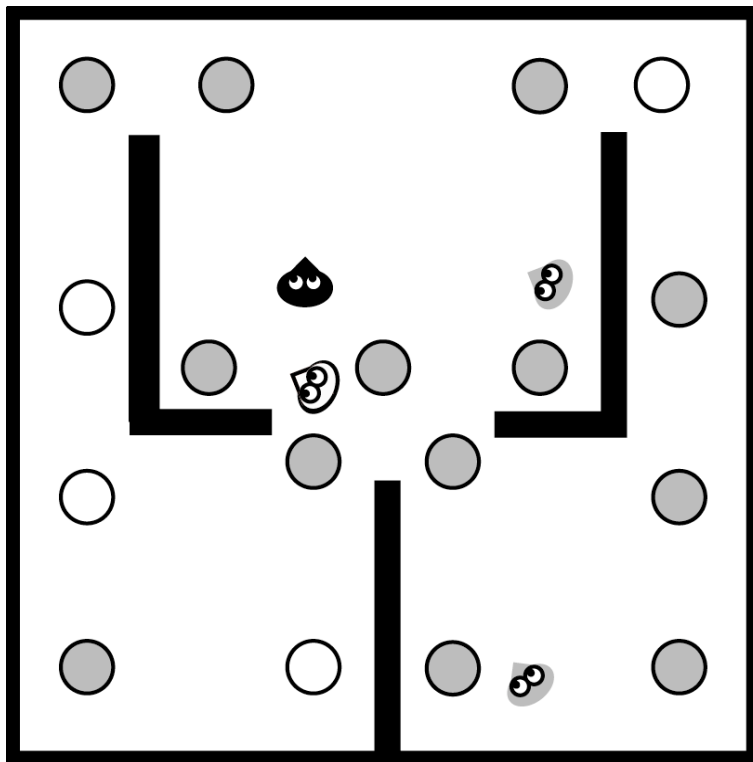





Finding a Safe Attack Position

- Next determine the set of nodes that are visible to all other enemies

$$V_{\bar{a}} = \bigcup_{j=0}^{j=k} V_j, j \neq a$$

Finding a Safe Attack Position



-  NPC
-  Selected Enemy
-  Other enemies

other

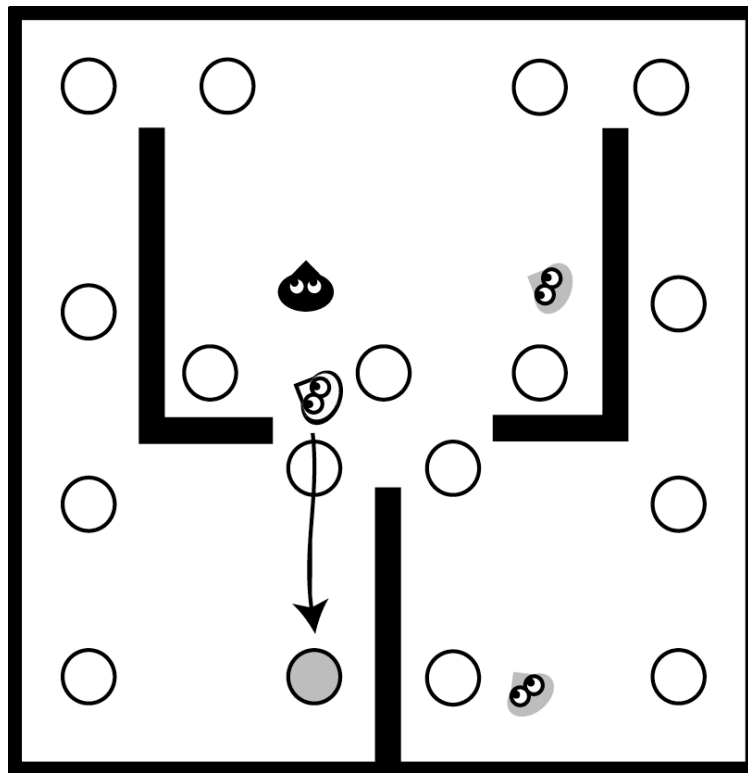





Finding a Safe Attack Position

- The set of good attack positions is the set of nodes with LOS to the enemy intersected with the inverse of the set of nodes with LOS to all other enemies

$$V'_a := V_a \cap \overline{V_{\bar{a}}}$$

Finding a Safe Attack Position



-  NPC
-  Selected Enemy
-  Other enemies



Ambush Points

- Unless cheating is employed, NPCs don't have full knowledge of the world
- May not know where all their enemies are located

So:

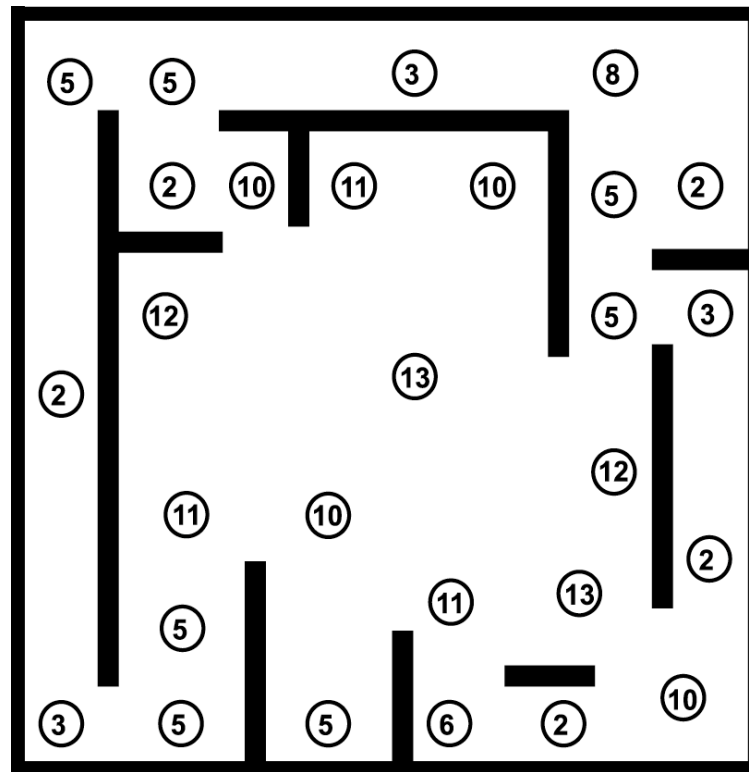
- Find a good location to wait in, for attack
- Not all positions are created equal



Static Waypoint Evaluation

- To find a good set up position:
 - Establish the exposure of all waypoints in a map
 - Process can be done off line, before game is even started

Static Waypoint Evaluation

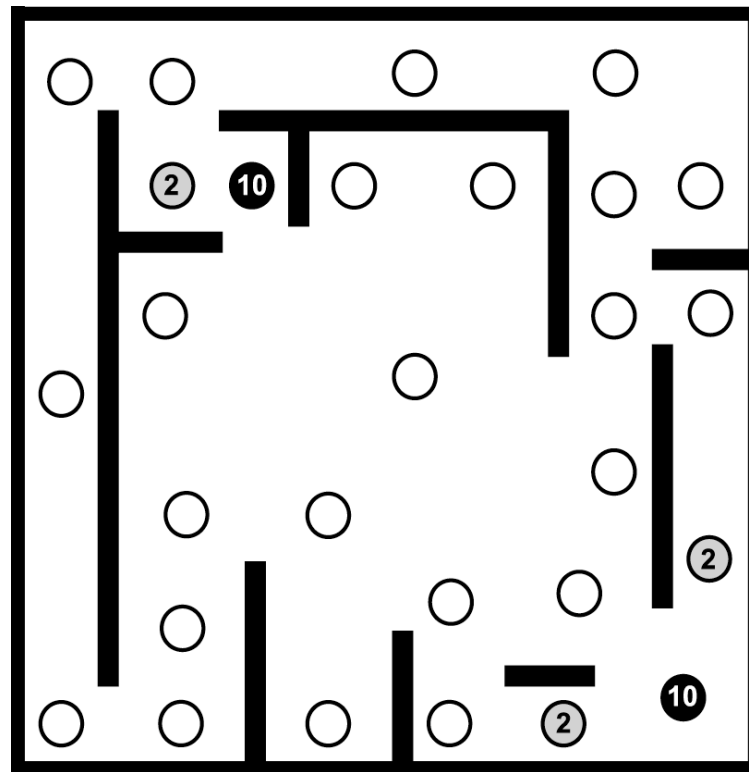




Static Waypoint Evaluation

- A good location is one which:
 - Has high exposure (visibility)
 - Easy to locate enemies
 - Easy to establish LOS to attack an enemy
 - Has areas of low exposure nearby
 - Can hide easily
 - Can run for cover easily

Static Waypoint Evaluation





Pinch Points

- Observation of human players reveals that experienced players anticipate the actions of their opponents
 - For example, if an enemy enters a room with only a single exit an experienced player will wait just outside the exit setting up an ambush
- Such “pinch points” can be pre-calculated by analyzing the node graph



Pinch Points

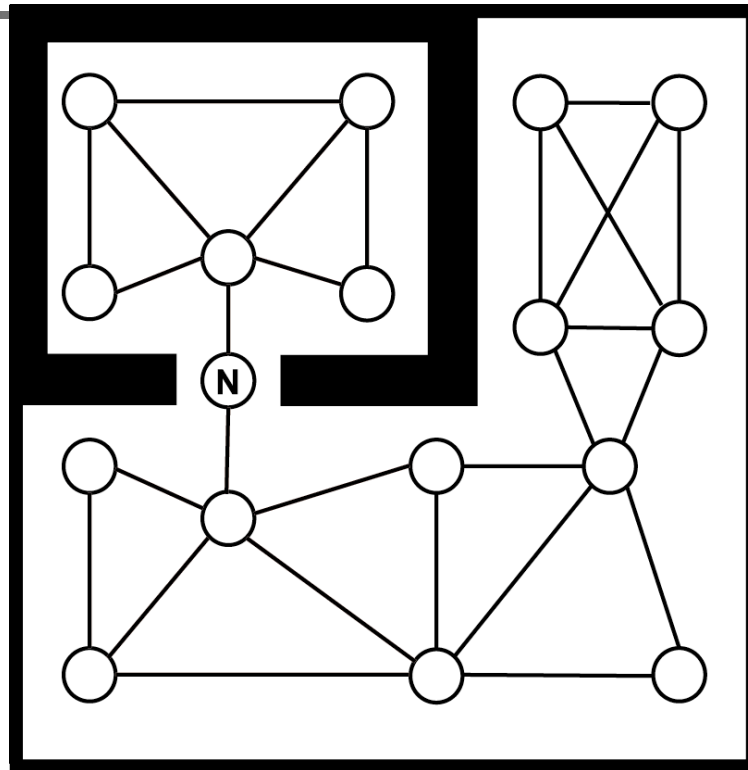
To find pinch points:

For each node, **N** in the node graph with only two neighbors:

- Temporarily eliminate node, **N**, from the graph, call its neighbors as **A & B**
- If both **A & B** are connected to large regions, **N** is not a pinch point, try another **N**
- Else attempt to find a path between **A & B**
- If path exists, **N** is not a pinch point, try another **N**
- Else call the node connected to the larger region, **O** (for outside)
- And the node connected to the smaller region, **I** (for inside)

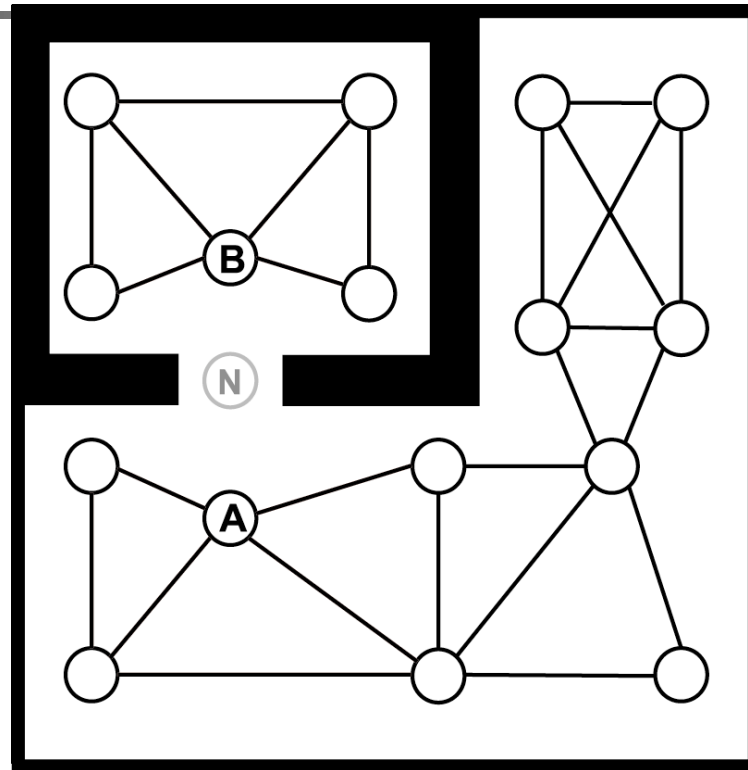
Let's do that again step-by-step:

Pinch Points



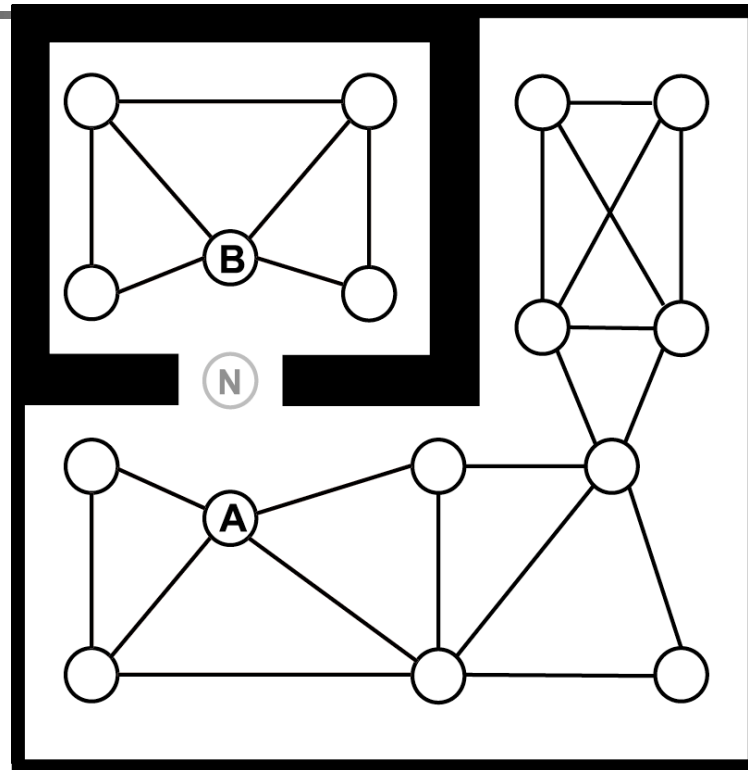
- For each node, **N** in the node graph with only two neighbors:

Pinch Points



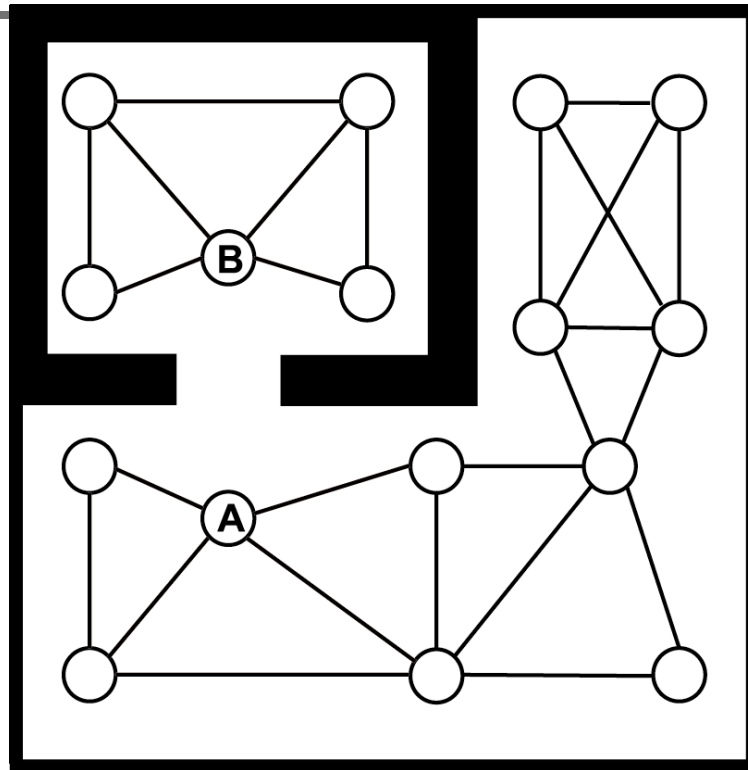
- Temporarily eliminate node, **N**, from the graph, call its neighbors **A** & **B**

Pinch Points



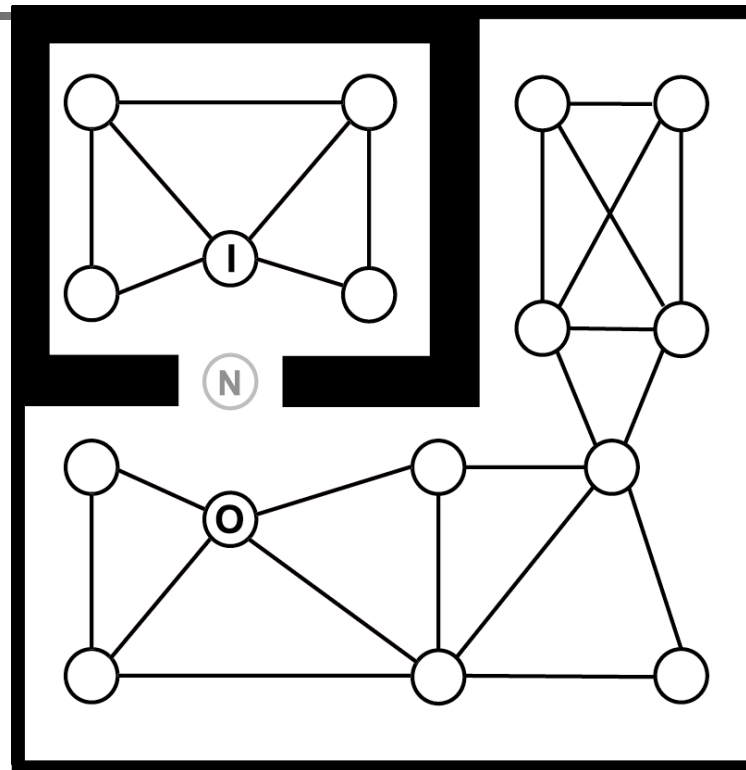
- If both **A** & **B** are connected to large regions, **N** is not a pinch point, try another **N**

Pinch Points



- Else attempt to find a path between **A** & **B**, if exists try another **N**

Pinch Points



- Else call the node connected to the larger region, **O** (for outside)
- And the node connected to the smaller region, **I** (for inside)



Pinch Points

Once a pinch point has been located a good ambush location is one which:

- Has a line of sight to the waypoint outside the pinch location "O"
- Can't be seen from the pinch location "N"

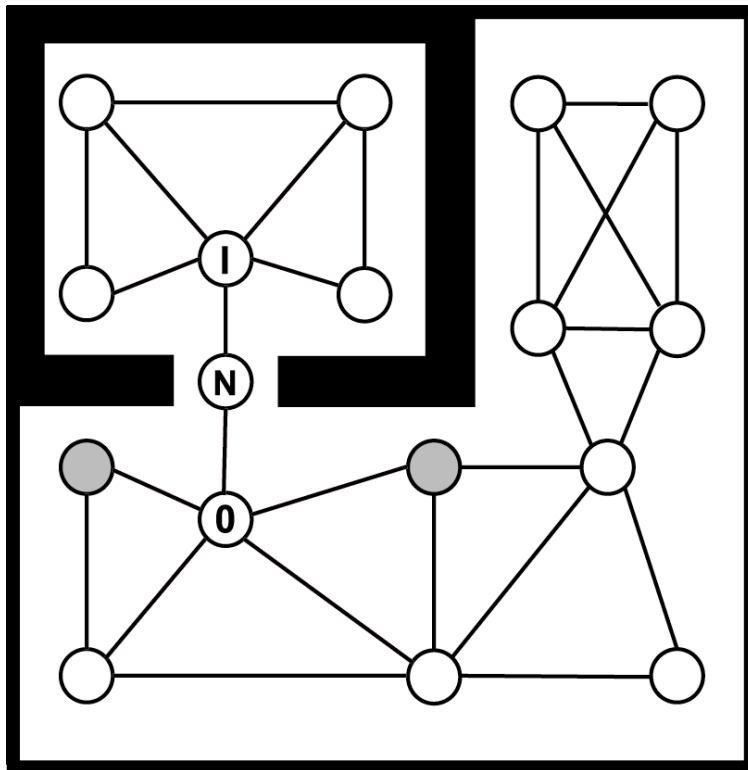


Pinch Points

- Nodes that have a line of sight to pinch location "O" V_O
- Can't be seen from the pinch location "N" \bar{V}_N
- Good ambush locations is their intersection:

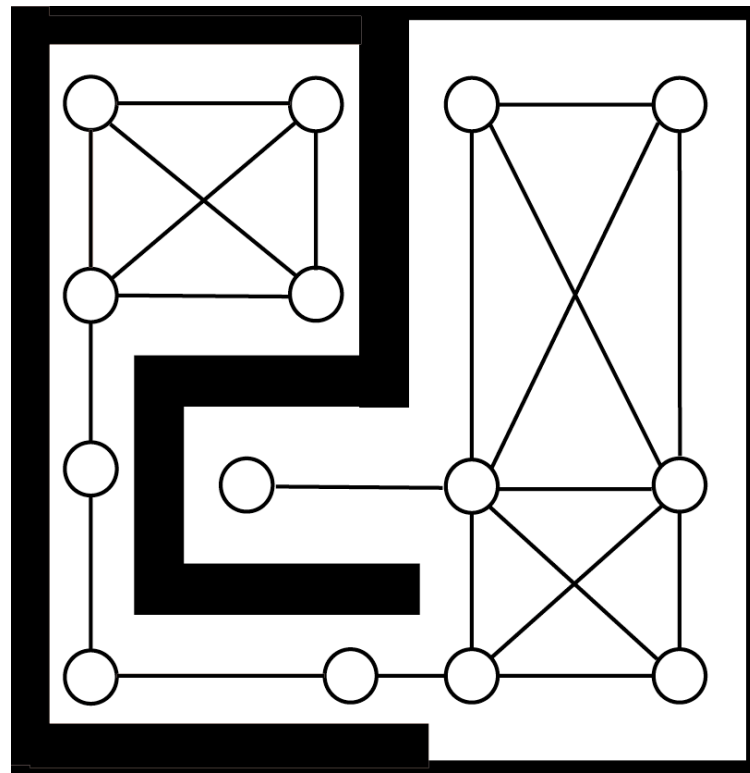
$$V_P = V_O \cap \bar{V}_N$$

Pinch Points



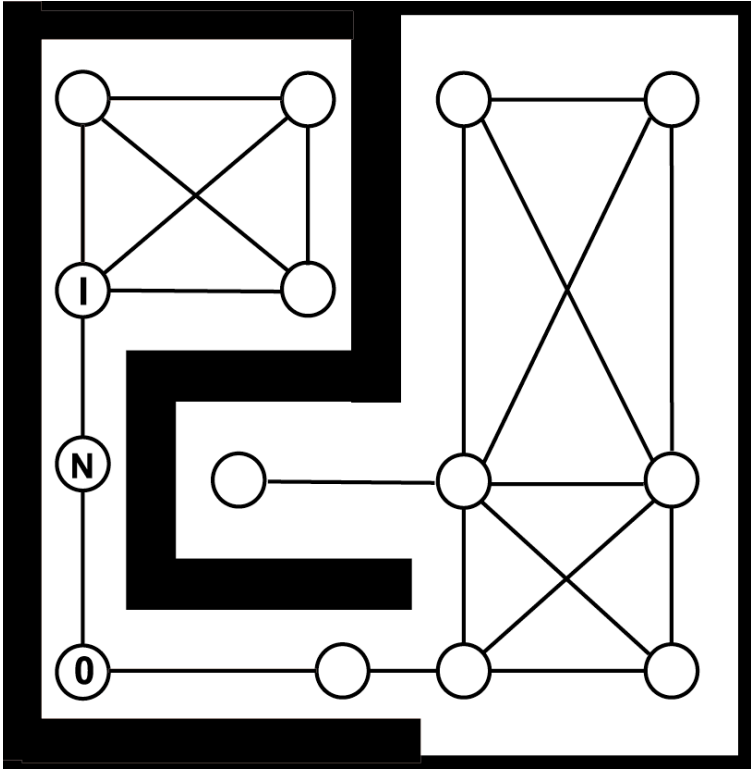
Pinch Points

Another Example:



Pinch Points

Result:





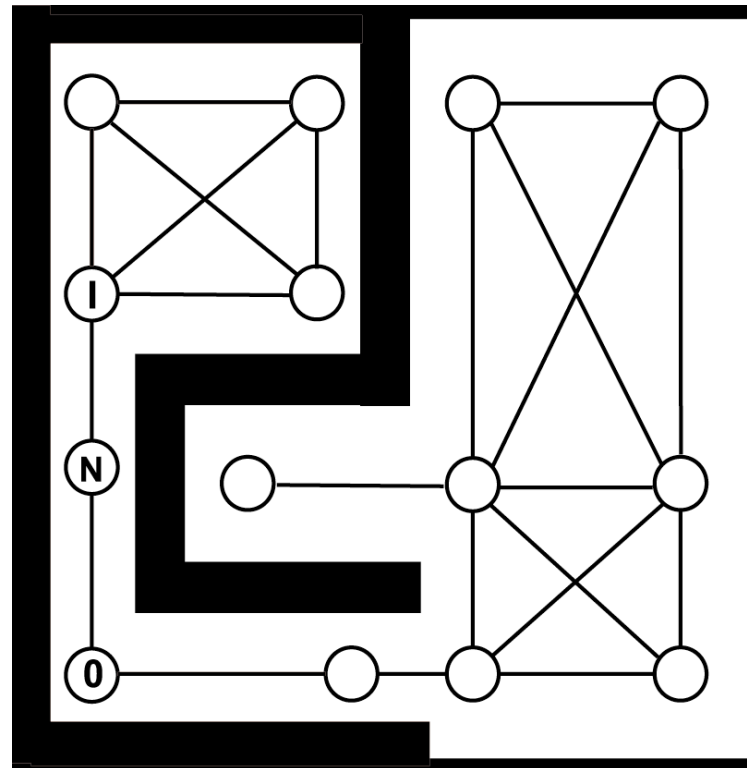
Pinch Points

Slightly altered version to find pinch points at the end of hallways:

For each node, **N** in the node graph with only two neighbors:

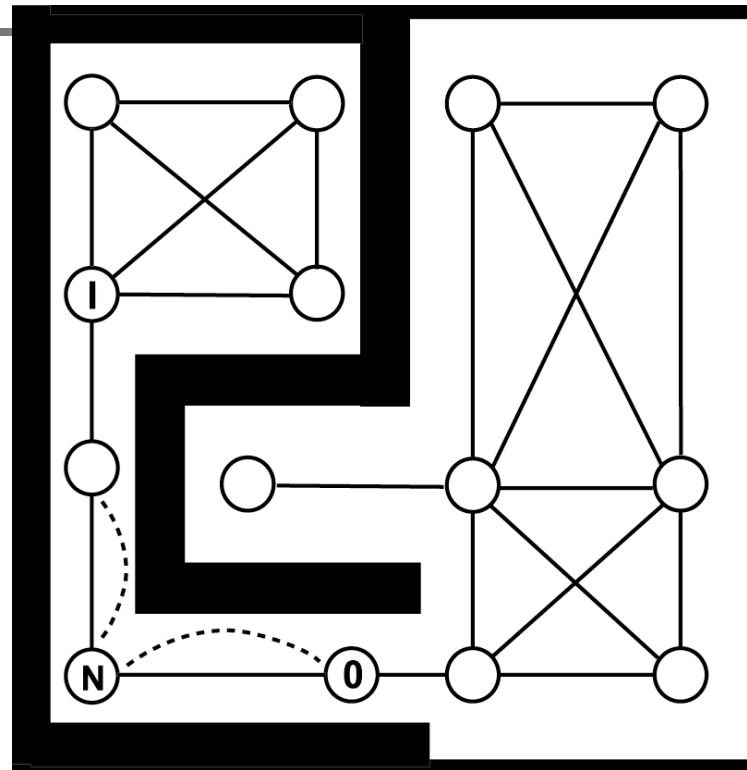
- Temporarily eliminate node, N, from the graph, call its neighbors as A & B
- If both A & B are connected to large regions, N is not a pinch point, try another N
- Attempt to find a path between A& B
- If path exists, N is not a pinch point, try another N
- Call the node connected to the larger region, O (for outside)
- Call the node connected to the smaller region, I (for inside)
- **If O has only one other neighbor in addition to N**
 - **Move N to O**
 - **Move O to the other neighbor of the old O**
 - **Repeat until O has more than one neighbors**

Pinch Points



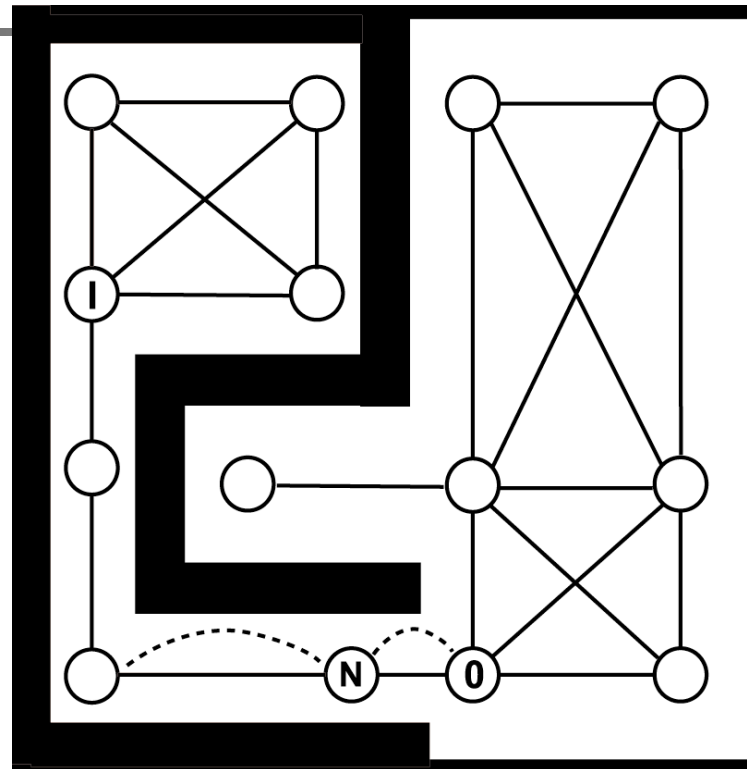
- If O only has one other neighbor in addition to N

Pinch Points



- Move N to O, Move O to other neighbor of old O
- Repeat till O has more than one neighbors

Pinch Points



- Move N to O, move O to other neighbor of old O
- Repeat till O has only one neighbors



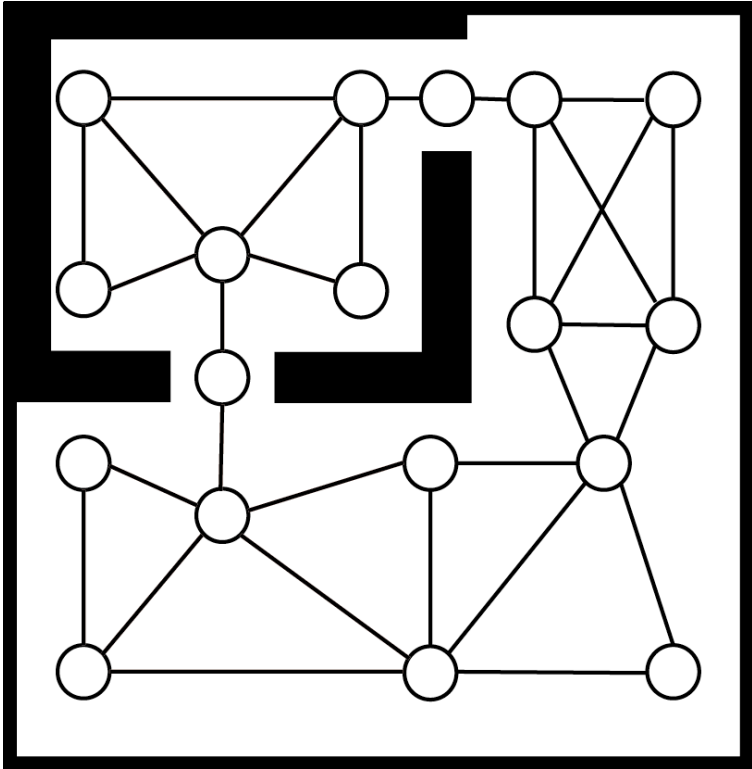
Pinch Points

- Calculate good ambush locations:

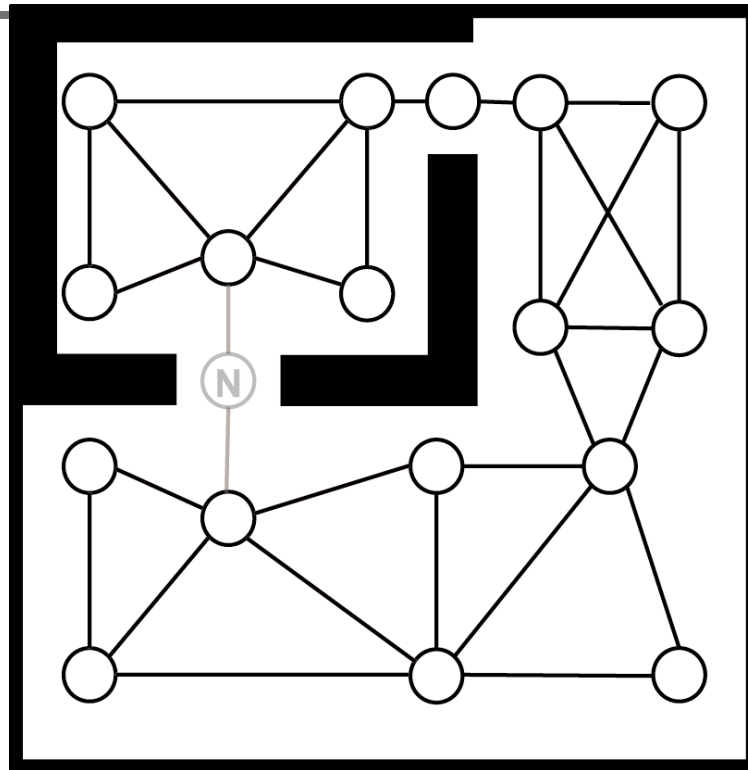
$$V_P = V_O \cap \bar{V}_N$$

Pinch Points

Final Example:

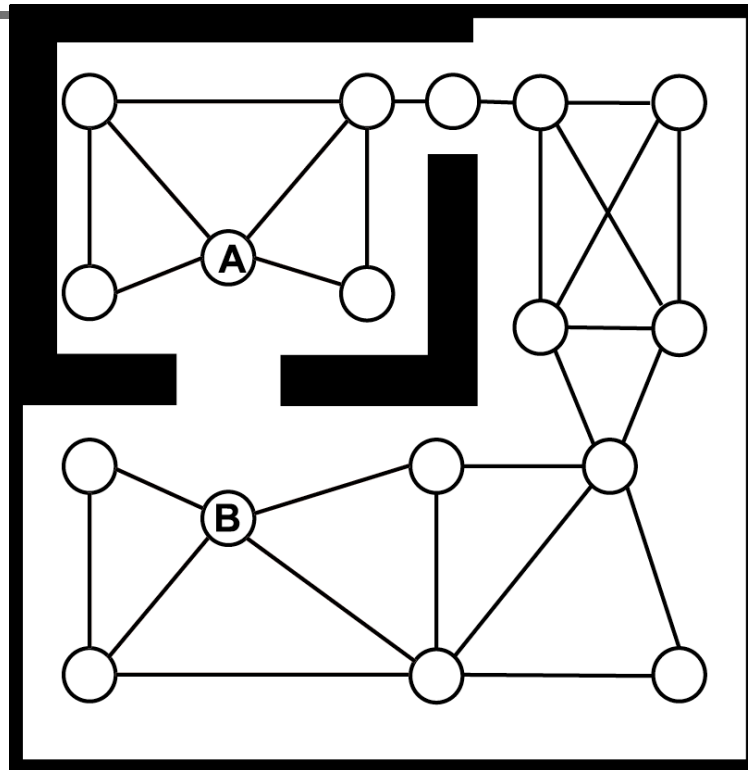


Pinch Points



- For each node, **N** in the node graph with only two neighbors

Pinch Points



- **Attempt to find a path between A& B**
- **If path exists, N is not a pinch point, try another N**



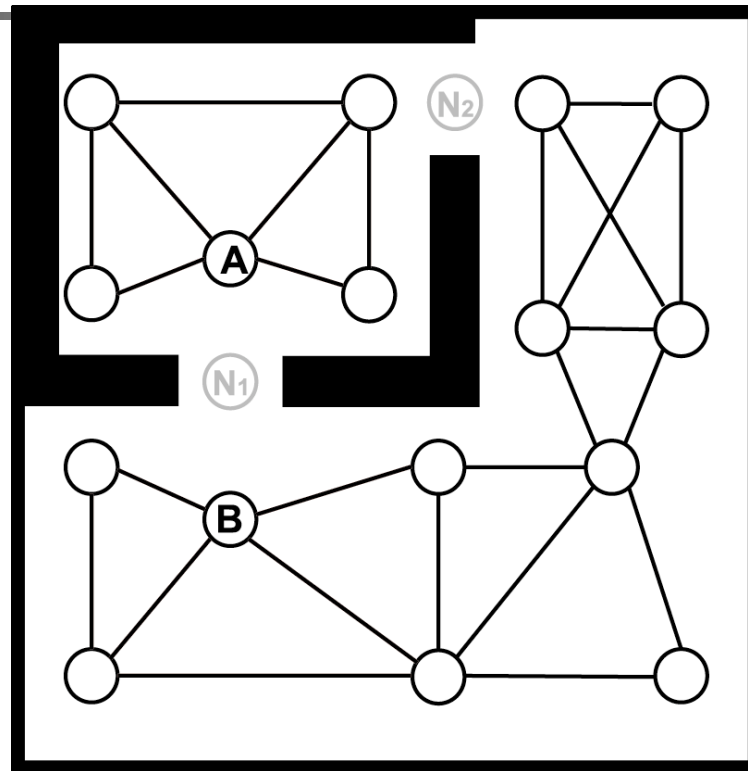
Pinch Points

If NPCs organize into squads regions with multiple pinch points can be employed:

For each node, N_1 in the node graph with only two neighbors:

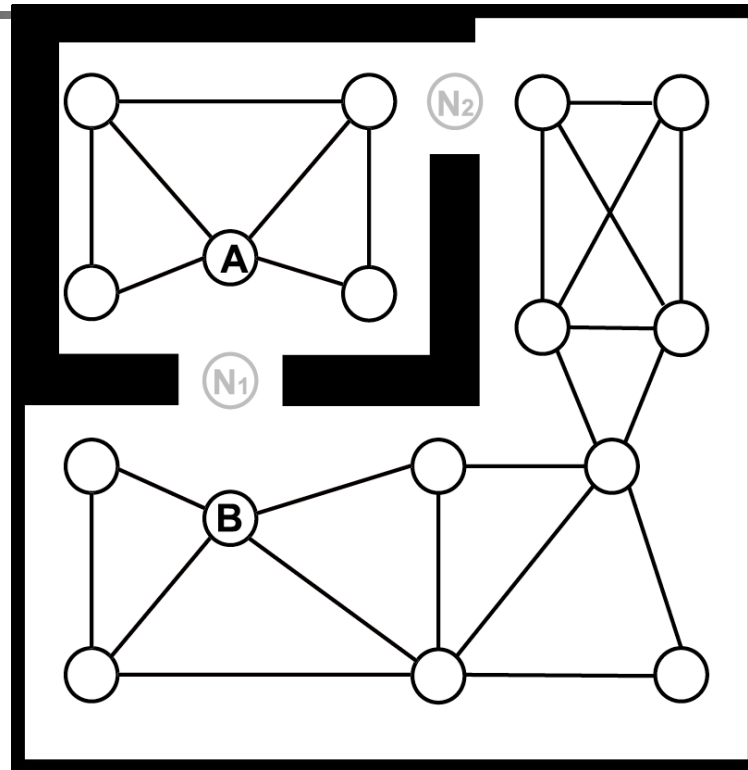
- **Temporarily eliminate node, N_1 , from the graph, call its neighbors as A & B**
- **If A & B are connected to large regions, N_1 is not a pinch point, try another N**
- **Attempt to find a path between A&B**
- **While generating the path if a node with only two neighbors is found,**
 - **Temporarily eliminate it and call it N_2 .**
 - **Attempt to find a path between A&B**
 - **If path exists, not a pinch point, try another N_1**
- **Call the nodes connected to the smaller regions, I_1 and I_2 (for inside)**
- **Call the nodes connected to the larger regions, O_1 and O_2 (for outside)**

Pinch Points



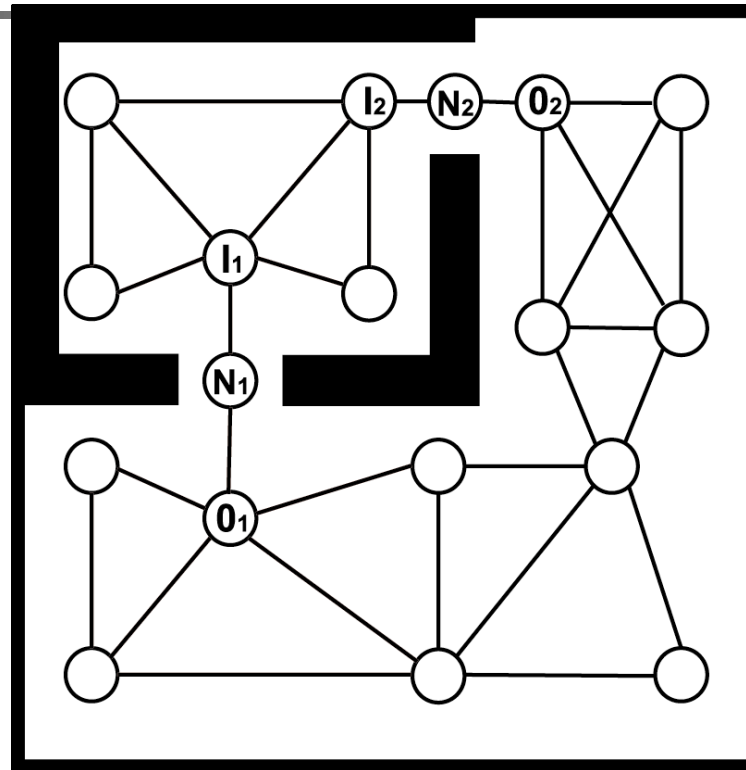
- While generating the path if a node with only two neighbors is found
 - Temporarily eliminate it and call it N_2

Pinch Points



- Attempt to find a path between A & B
- If path exists N_1 is not a pinch point, try another N_1

Pinch Points



- Call the nodes connected to the smaller regions, I_1 and I_2 (for inside)
- Call the nodes connected to the larger regions, O_1 and O_2 (for outside)



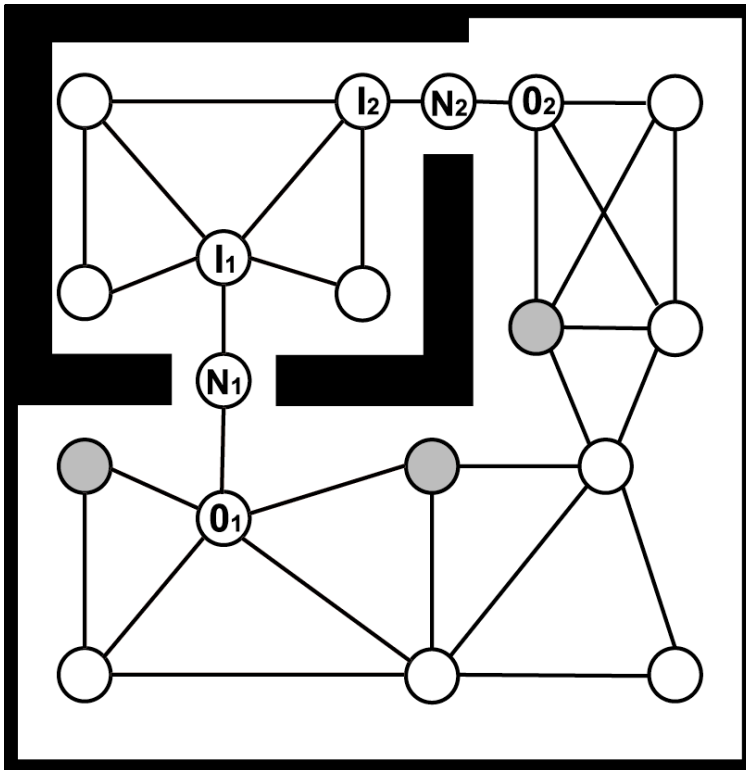
Pinch Points

- Calculate good ambush locations:

$$V_{P_1} = V_{O_1} \cap \bar{V}_{N_1} \cap \bar{V}_{N_2}$$

$$V_{P_2} = V_{O_2} \cap \bar{V}_{N_1} \cap \bar{V}_{N_2}$$

Pinch Points





Tactical Analysis: Review

- Using the node graph to evaluate map locations:
 - Finding safe and dangerous locations
 - Finding places from which to attack
 - Finding location to set up sniper positions
 - Finding pinch points

Embedded Environment Information



- The world looks boring when NPCs are just standing around doing nothing until engaged by the player
- *No One Lives Forever 2* uses embedded environment information to keep the NPCs busy (pioneered by *The Sims*?)
- Embedded environment information: objects tell NPCs how to interact with them, NPCs react to objects as they get close
- Problem: unconstrained behavior resulted in unfavorable ones, e.g., an NPC wandering all over the level looking for something to do, tasks are done out of order, etc.



Constraining NPC Behavior

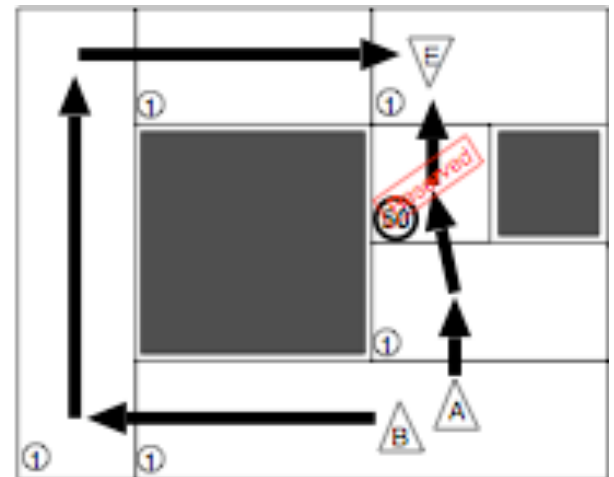
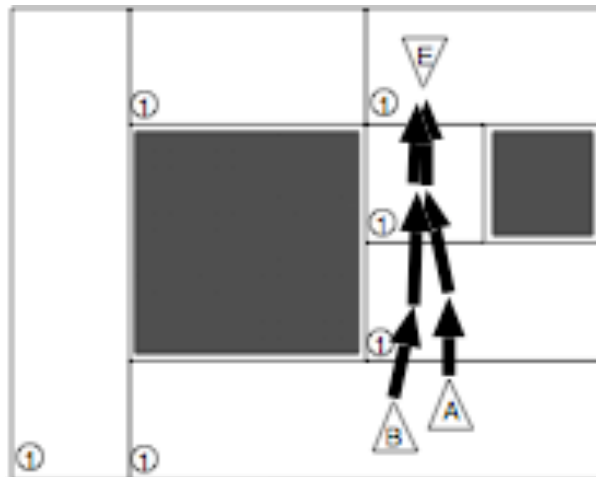
Added constraints:

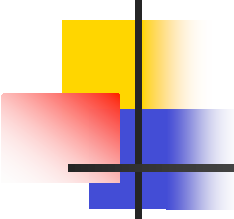
- **ownership**: an NPC owns a set of objects. NPCs do not try to acquire and interact with objects own by others.
- **dependency** between objects: cannon must be loaded before they can be fired.
- **responsibility**: objects tagged with class of NPC, NPCs tethered to regions
- **prioritization**: aggressive behavior over investigative behavior, over relaxed behavior; specific over general behavior.

Embedded Info, Animation, Sound

Embedded info can be used in path finding:

- Node graph is an example embedded environment information
- embed in path how to jump over crevices or how to open door (*Soldiers of Fortune 2*)
- illusion of coordination by “reserving” a path



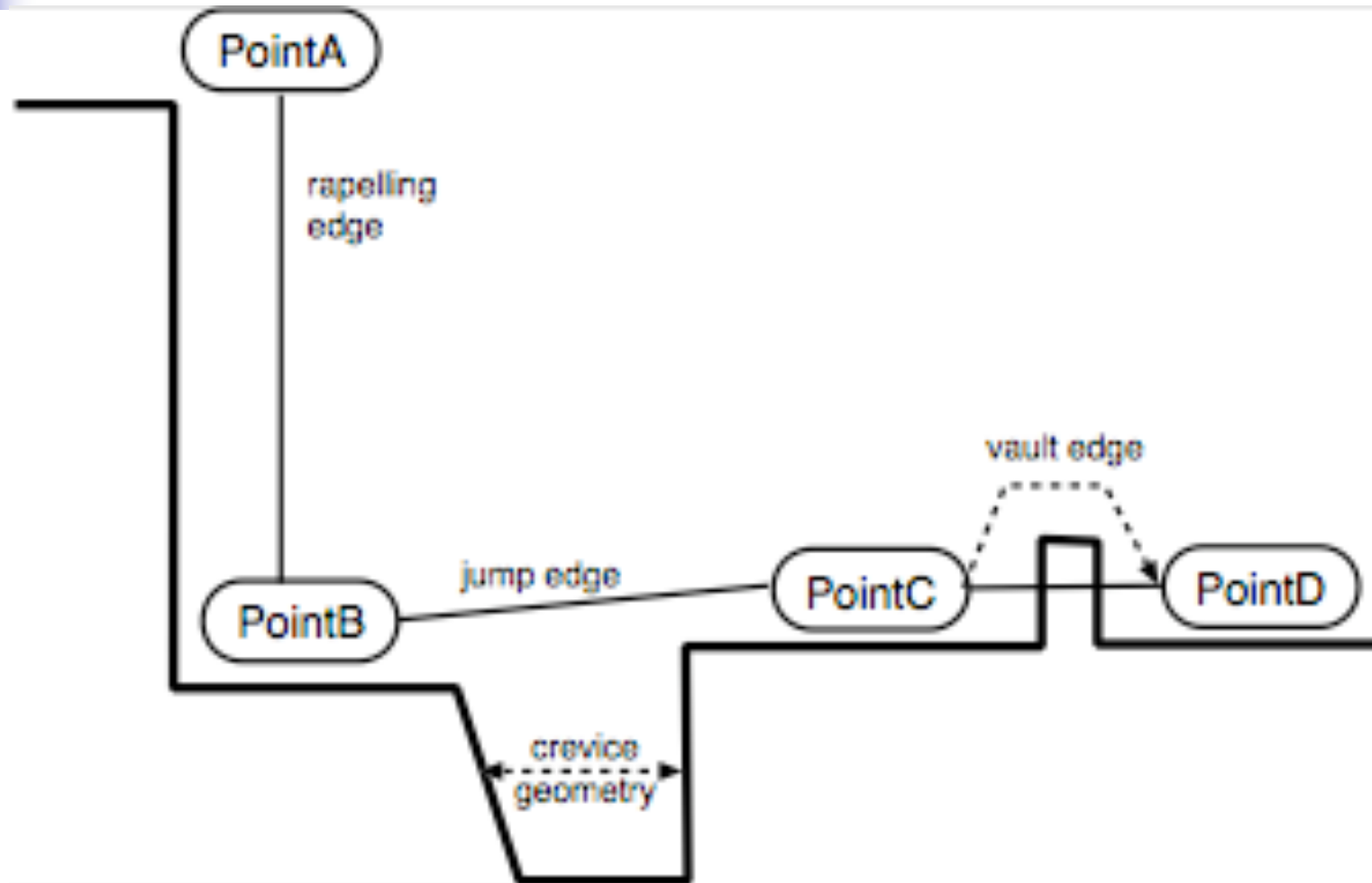


Embedded Info, Animation, Sound

Team AI behavior can be used with animation and sound to add realism:

- request for cover or order split up
- choose a different posture or animation from team members
- jumping, vaulting, rappelling based on embedded info

Embedding Animation





Emergent Behavior and Squad AI

- Emergent behavior:
 - behavior not hard-coded as FSM or decision tree
 - global behavior emerges from *interaction* of elements with simple local behavior
 - more than the sum of individual behavior
- Pros: more scalable, less predictable, not scripted
- Cons: unintended behavior, local rules hard to define

- Example: flocking



Squad/Team AI

Centralized:

- leader receives info, issues order
- complex maneuvers can be planned and coordinated
- cannot easily accommodate the strengths and needs of individual members

Distributed:

- squad maneuvers as emergent behavior
- simple extensions to individual AI
- robust against members being “taken out”
- weak at maneuvers requiring tight coordination
- examples: *Half-Life*, *No One Lives Forever 2*



Distributed Squad AI

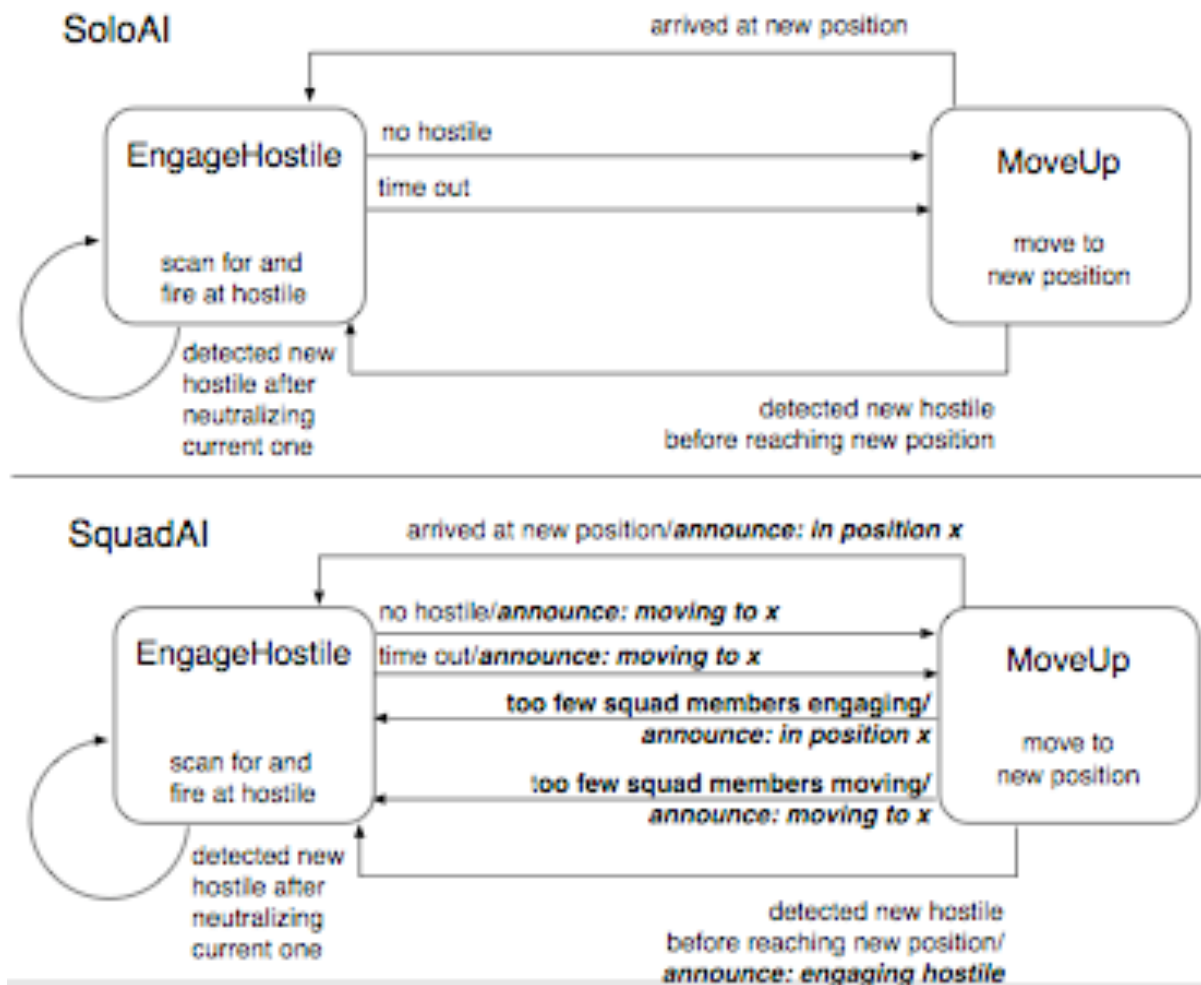
Each member publishes:

- **state:** conscious, unconscious
- **intention:** "I'm moving to position (x, y, z) "
- **observation:** "hostile at (x, y, z) "

Action selection by:

- situation of team mates
- threats to team
- own states

Fire&Move: Solo vs. Squad





Example: Squad Assault

Local behavior: fire&move FSM, considering:

- **Avoidance:** prevent blocking other team member's line-of-fire
- **Cohesion:** maintain team cohesion, stay within audible range, maintain line-of-sight to other members
- **Separation:** spread-out, prevent becoming a bunched-up target
- stay close to cover
- stay away from enemy's line-of-fire
- take weapon capabilities into account
- fire&move FSM



Squad Assault

Each member knows its own:

- state
- position
- claimed destination position
- line-of-fire

and enemy's:

- last known position and state
- predicted position and state
- line-of-fire
- weapon damage radius
- squad members engaging this hostile
- squad members able to observe this hostile



Squad Assault

Individualizing behavior:

- **riflemen** move quick and often, to close in on the hostile
- **machine gunners** are slowed by load, deliver support fire from a rather static position and need cover
- **snipers** engage enemy from a distance, need clear line-of-fire and good cover
- wounded or reloading members need cover



Problems with Emergent Behavior

Conflicts in path and position selections, members bumping into each other

Solution: prioritize member based on

- time to destination (shorter first)
- urgency (need for cover)
- strength of weapon (blast away)

No good position within range of search

Solution: randomly increase range of search, added benefits: flanking behavior, more varied behavior



Problems with Emergent Behavior

It's not always obvious what local behavior is needed to achieve certain global behavior, e.g., how to design squad ambush or covering pull-back?

Corollary: there could be some *unfavorable*, unintended behavior



Centralized Squad Attack

Different styles:

- **authoritarian:** orders must be obeyed
 - pro: allows for individual suicide mission for “the greater good”
 - con: local details could be overlooked by squad leader
- **coaching:** each member assigned tasks that they execute to their best ability (or not)
 - squad member may not have the larger picture to know what's best for the squad
- **mixed:**
 - authoritarian with early negative feedback
 - annotate coaching tasks with value
 - rule-of-engagement: switches between the two based on situation

AI in First-Person Shooter Games



Quake III Arena

- Released in 1999 by id Software
- Designed to be a multiplayer only game
- The player battles computer-controlled opponents, or bots
- Bots developed by Jan Paul van Waveren





Quake III Bot AI

- FSM based – Uses a stack for short-term goals
- Use Fuzzy Logic for some decision making
 - Collecting weapons and armor
 - Choosing a weapon for combat
- Fuzzy Relations were selected using Genetic Algorithms
- Each bot has a data file containing weapon preferences and behavior-controlling variables

Bot Network

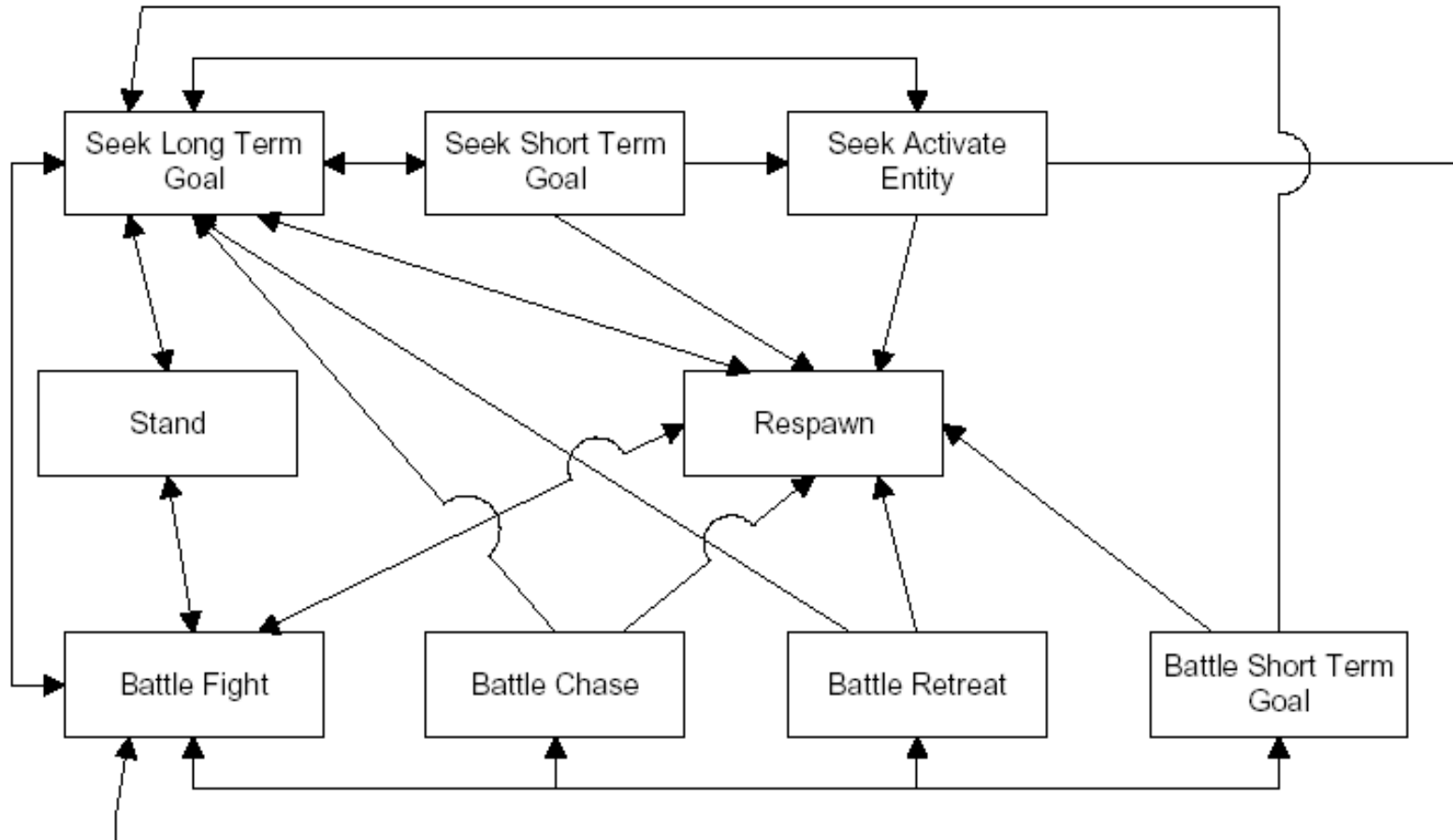


Figure 15.1: AI network.

Quake III Bot Navigation

- AAS (Area Awareness System)
 - Level is subdivided into convex hulls that contain no obstacles
 - Connections between areas are formed



Figure 18.2: jump reachability



Figure 18.3: jump pad reachability



Bot Chatting

- Deathmatch
 - Not much more than a fun extra
- Team-Play
 - Bots can follow orders to defend, attack, escort
 - Bots will take 'Team Leader' position if player doesn't
 - Team Leader delegates tasks to bots and players



Bot Input

- Bots simulate human input
 - 90 degree FOV
 - fog and the invisibility powerup impact vision
- Bots use sound to detect enemies

Half-Life

- Released by Valve Software in 1998
- Built using the Quake/Quake 2 engines
- AI uses a “schedule driven state machine”





Story-Based Game

- Half-Life is a plot-driven game, so the AI must further the story
- NPC's aid player throughout game, but are rarely essential
- Scripted sequences immerse the player in the story and create sense of importance



Scripting

- Scenes are built inside levels using triggers and movement nodes
- Examples
 - Security guards or scientists give player information about his goals
 - Battles between aliens and Marines
 - Scientist panics and runs into tripmines

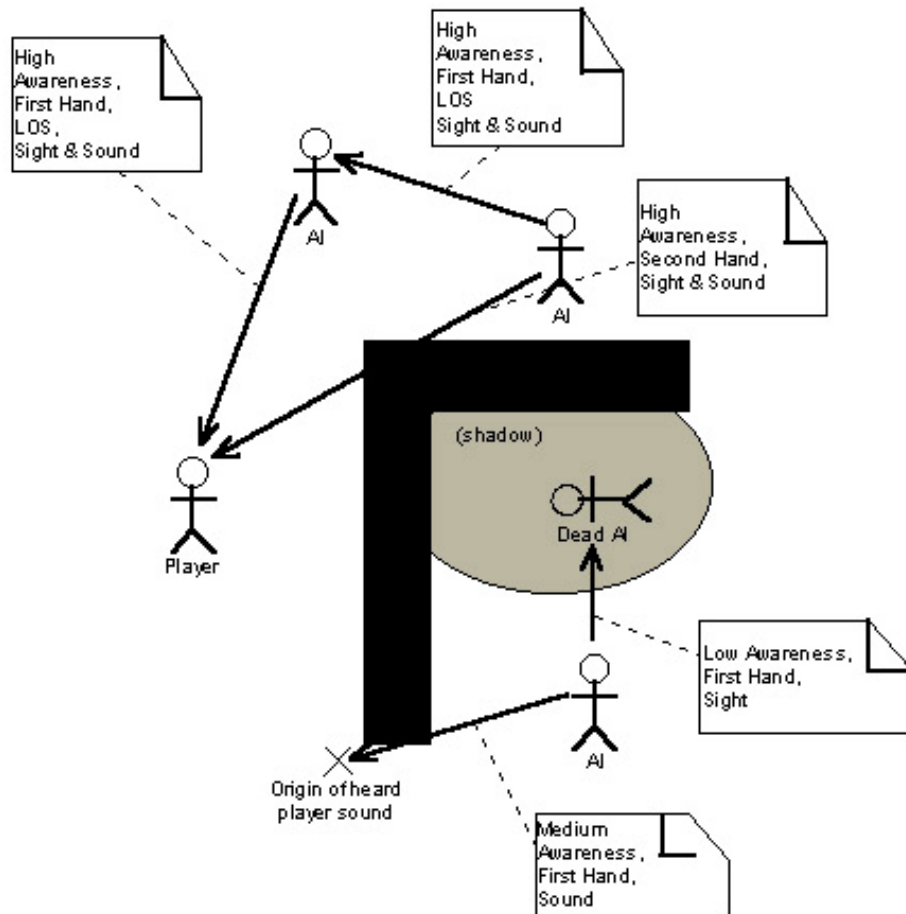


Thief



- Developer - Looking Glass
- Publisher - Eidos interactive
- Revolutionary "Dark Engine"
- Based on stealth
- Released November 11th, 1998

“Dark Engine”



Lightly scripted game
Specifically single-player
Multi-state sense system
Decision state machines
Centers around the
system's output

The logo for Tom Clancy's Ghost Recon. It features the text "Tom Clancy's" in a white serif font at the top. Below it, the words "GHOST RECON" are written in a large, bold, white, blocky font with a metallic, 3D effect. The letter "O" in "GHOST" has a small circular detail. The letter "O" in "RECON" is replaced by a green reticle symbol, which is a circle with a crosshair and a small "TF" in the center. The entire logo is set against a black background.

Tom Clancy's GHOST RECON

- Published - Ubi Soft Entertainment
- Greg Stelmack, lead engineer
- Development - Red Storm Entertainment
- Realistic combat battlefield game
- Released November 13th, 2001



Ghost Recon AI Technique

- A lot of scripting for individual missions
- Enemy and team units use FSM's
- Modifiable hierarchical commands
- Local navigation and pathfinding (causes some hang-ups small environmental details)

Ghost Recon Unit Control

- Control five other teammates
- Tactical overlay map
- Set team engagement strategy
- Units respond to other unit actions





Ghost Recon Gameplay Focus

- Realistic military features
- Stealth and avoidance add new aspect to AI
- Both enemies and friendlies must have heightened senses of awareness
- “Gameplay rules all.” - Greg Stelmack



- Epic Games – Unreal Engine
- Steve Polge, lead programmer at epic
- Digital Extremes – Gameplay depth and design
- Very fast multiplayer FPS
- Large emphasis on team play

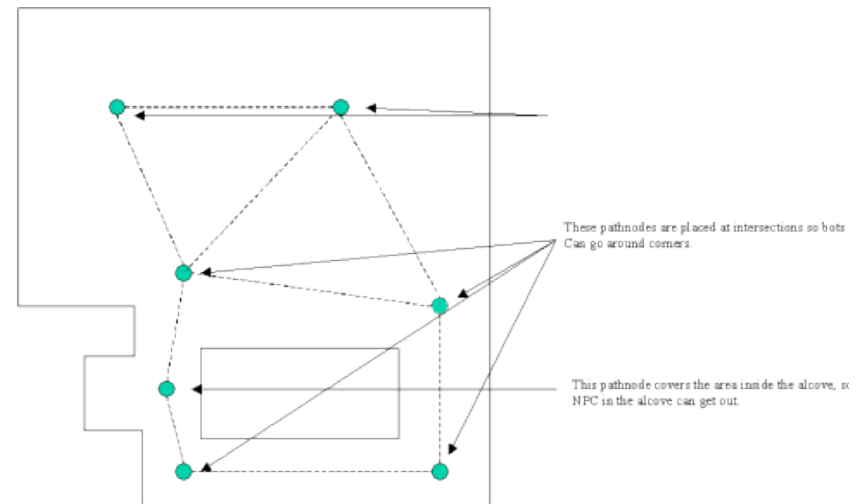
Unreal Scripting

- UnrealScript much like Java/C++
- Scripting used to control specific Bot actions
- Every respawned Bot checks script flag



Unreal Pathfinding

- Based upon the common pathnode technique for navigation
- Uses a pre-computed data structure for guiding movement
- Complex algorithm-controlled assemblage of linked lists, Navigation Points, and Binary Space Partitioning (BSP) collision data





Unreal Bot Combat

- AI uses states heavily
- Several triggers that determine Bot's actions
- "Type" of Bot determines fighting style
- Accuracy and speed factor into Bot's difficulty level
- Fun factor heavily influences Bot strategy



Unreal Team Play

- Incorporates several team oriented games:
 - Team Deathmatch
 - Capture the flag
 - Bombing run
 - Double domination
- Hierarchical AI system
- Player controlled team
- Bots have numerous types of flocking patterns
- Team bots are mediocre, while enemy bots are excellent



References

- http://www.pcgamer.com/eyewitness/eyewitness_2002-09-18.html
- <http://udn.epicgames.com/>
- <http://www.unrealtournament2003.com/>
- <http://www.eidosinteractive.com/gss/legacy/thief/>
- <http://ai.eecs.umich.edu/people/laird/gamesresearch.html/>
- ftp://ftp.kbs.twi.tudelft.nl/pub/docs/MSc/all/Waveren_Jean-Paul_van/thesis.pdf
- http://www.gamasutra.com/features/19991210/birdwell_pfv.htm