THE FLAME IN THE FLOOD
“Spatial Awareness”

(A Theme, not a technique, or technology, or algorithm)
Spatial Abilities in Halo 2

- **Static Pathfinding**
  - Navigation mesh (ground)
  - Waypoint network (airborne)
  - Raw pathfinding
  - Path-smoothing
  - Hint integration (jumping, hoisting, climbing)
  - Static scenery-based hints
  - Static scenery carved out of environment mesh

- **Static feature extraction**
  - Ledges and wall-bases
  - Thresholds
  - Corners
  - Local environment classification

- **Object features**
  - Inherent properties (size, mass)
  - Oriented spatial features
  - Object behaviors (mount-to-uncover, destroy cover)

- **Dynamic Pathfinding**
  - Perturbation of path by dynamic obstacles
  - "Meta-search" / Thresholds / Error stages
  - Obstacle-traversal behaviors
    - Vaulting, hoisting, leaping, mounting, smashing, destroying

- **Path-following**
  - Steering on foot (with exotic movement modes)
  - Steering a vehicle (e.g. ghost, warthog, banshee)

- **Interaction with behavior**
  - What does behavior need to know about the way its requests are being implemented?
  - How can pathfinding impact behavior?

- **Body configuration**
  - Flying, landing, perching
  - Cornering, bunkering, peeking

- **Spatial analysis**
  - Firing position selection
  - Destination evaluation based on line-of-sight, range-to-target, etc.

- **“Local spatial behaviors”**
  - Line-tracing (e.g. for diving off cliffs)
  - Not facing into walls
  - Crouch in front of each other
  - Don’t walk into the player’s line of fire
  - Curing isolation
  - Detecting blocked shots

- **Reference frames**
  - The viral nature of the reference frame

- **Cognitive model / Object persistence**
  - Honest perception
  - Simple partial awareness model

- **Search**
  - Simple by design
  - Group search

- **Spatial conceptualization**
  - DESIGNER-PROVIDED
  - Zones, Areas (areas), Firing positions (locations)
The Most Fundamental of Questions

Where do I stand right now?

- Depends on a *huge* amount of context.
  - Internal: goals, intentions, behaviors, etc.
  - External: target position, actions, obstacles, etc.
- Extremely player-facing / gameplay relevant
- **Should** be in the hands of the designers.
Position Selection

1. **Gather** potential positions
2. **Score** each position with F(x)
3. **Choose** the best one
4. **Go there**
Representation

Point cloud (Halo 2) + Navigation Mesh

Navigation Graph (Killzone)
(Image from Killzone’s AI: Dynamic Procedural Combat Tactics, by R. Straatman, W. Van Der Sterren, A. Beij, GDC 2005)
Representation

Regular Grid (Third Eye Crime)
Representation

Regular Grid (Third Eye Crime)
Gather Step

Some approaches:

- Points defined and assigned by designers
  - Halo
- Spatial query (points within radius or box)
- Dijkstra’s algorithm to find accessible positions
  - Third Eye Crime
Dijkstra’s Algorithm

Find
- Accessible points
- Path distances
- Reconstruct paths
Dijkstra’s Algorithm

Find

- Accessible points
- Path distances
- Reconstruct paths
Dijkstra’s Algorithm

Find
- Accessible points
- Path distances
- Reconstruct paths
Dijkstra’s Algorithm

Note: Difference between reps w/ Connectivity info and those without.

Without:
- Point clouds

With:
- Navmesh verts
- Regular grids
Position Scoring

F(x)

Range(x)
Line of sight(x)
Threat(x)
Distance(x)

"Spatial Function"

42

(An “Apples-to-Oranges” problem)
Spatial Function

Input Types Include:

- Range(x) = range from x to target
- Path_distance(x) = path distance from NPC to x
- LOS(x) = line of sight from x to target (1.0 = 100% clear)
- D(x) = distance of x to any occupied space
- etc.

RE-use Dijkstra’s from gather phase
Spatial Function Inputs

range

LOS
Spatial Function

- Simplest form
  \[ F(x) = k_1 A(x) + k_2 B(x) + k_3 C(x) + \ldots \]

- With remapping:
  \[ F(x) = f_1(A(x)) + f_2(B(x)) + f_3(C(x)) + \ldots \]
Remapping

“flee”

“charge”

“maintain distance”
Remapping

“find”

“cover”
Spatial Function

- Simplest form
  \[ F(x) = k_1A(x) + k_2B(x) + k_3C(x) + \ldots \]
- With remapping:
  \[ F(x) = f_1(A) + f_2(B) + f_3(C) + \ldots \]
- Ideally, use a flexible syntax:
  \[ F(x) = k(f_1(A) - f_2(B)) / (f_3(C) * f_4(C)) \ldots \]
- Our own idiosyncratic form:
  \[ F(x) = (((f_1(A) + f_2(B)) + f_3(C)) * f_4(D)) + f_5(E) \ldots \]
Implementation

Layers
- Input source
  - range
  - los
  - path-distance
  - etc.
- Combination method
  - Additive
  - Multiplicative
- Remapping Function
  - output = F(input)
- Global modifications
  - Blur factor
  - Normalization
\[ \text{BLUR}() \]
Position Selection + Pathfinding

The criteria for choosing points is not the same as the criteria for getting there

(e.g. “choose a spot with clear LOS but try and stay covered while you travel there”)
Observation #1

Input functions are expensive
  - LOS, path-distance, obstacle-distance, etc.

BUT remapping / combining/sharing is relatively cheap

So once we’ve computed the input layers, we can likely afford to run multiple spatial functions
Observation #2

For Spatial Reps w/ Connectivity:

SINCE we probably have expensive spatial input already computed on grid cells / navgraph vertices

And SINCE Dijkstra/A* can accommodate penalty functions

We can use a SEPARATE spatial function to specify a Dijkstra/A* penalty function
  • specify both where to go, and how to get there
However...

All paths were built into the gather-phase Dijkstra

Demo Solution: Use Dijkstra for gather but NOT for final path creation
  - Once position selected, run A* from scratch to that destination using penalty function
  - Expensive...

... And still wrong!
- The path-distance input was provided by Dijkstra.
- Not accurate if penalty function is distorting path
Where to Stand vs. How to Get There

*Flame in the Flood* Solution

Pass 1: Run penalty function
- Gather phase via axis-aligned bounding box (AABB)

Pass 2: Run position scoring function
- Gather via Dijkstra
- Pass 1 provides Dijkstra penalty
All Behavior is Spatial

Spatial functions can be used for more than just position evaluation

- A* penalty
- path speed
- aim on/off
- target bias
- weapon choice
- ...

Knife

Pistol

Rifle
Conclusions

Get positioning into the hands of the designers.

Then it’s their problem.
Thanks!

Questions?