SLANT SIX GAMES

RESIDENT EVIL®
Operation Raccoon City

Situational Awareness: Terrain Reasoning for Tactical Shooter A.I

Resident Evil: Operation Raccoon City developer Slant Six Games publisher Capcom
GameWallpapers.com hosted by JTLnet.com

Situational Awareness

- The capability of an A.I. agent to reason is limited by the data its sensor's can acquire about the world
- The representation of the data we use to represent the world has a direct impact on the amount of reasoning we can do in our given budgets.
- Many design problems can be solved easily with a solid and rich world representation.

techniques:

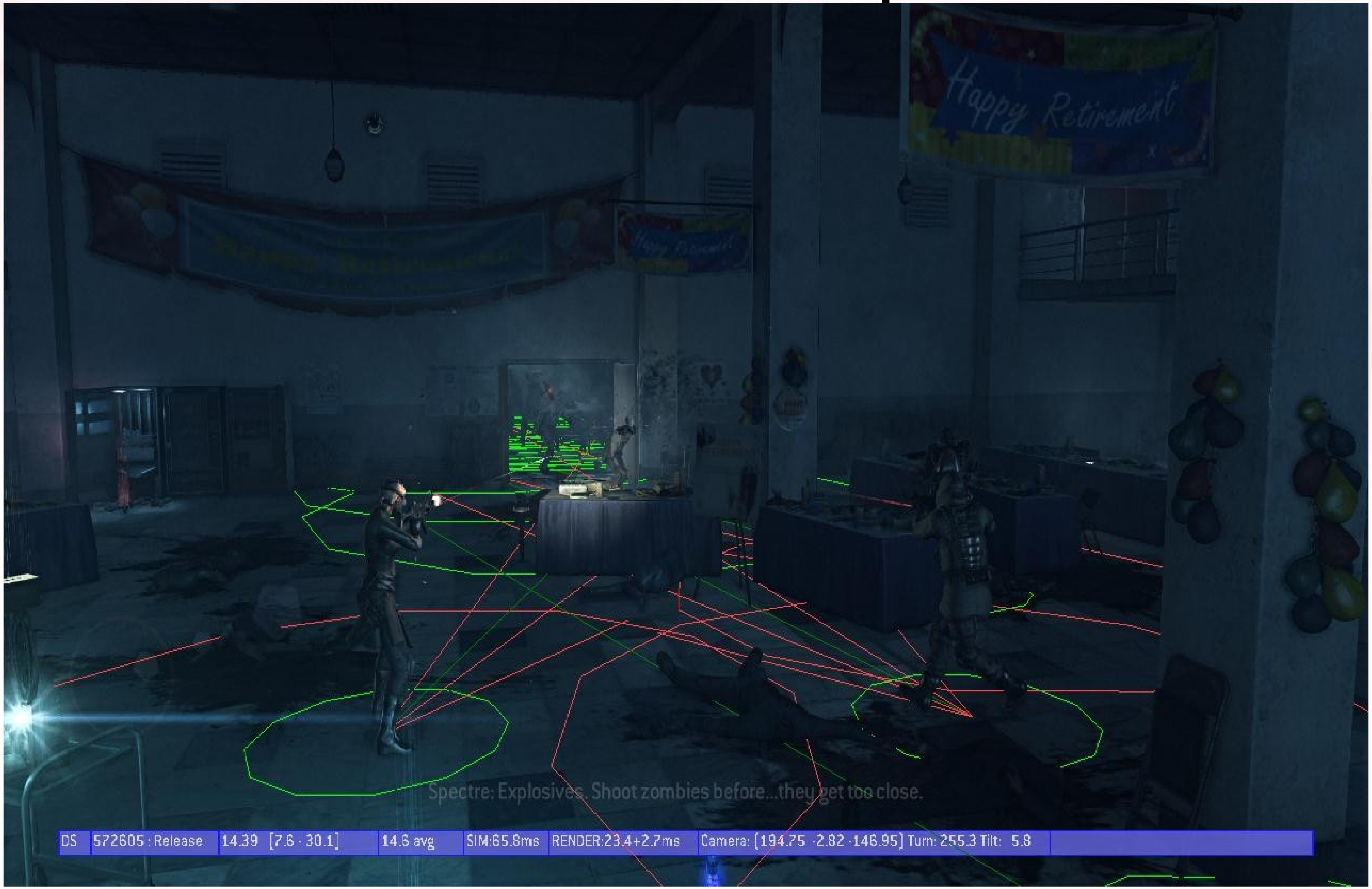
Goals

- Richer behaviours
- Increased flexibility in responding to changes to the world
- Dynamic, and more complex environments.

Pit falls - Portents of doom !

- Noisy Simulation - Unpredictability
- Behaviors that are not transparent to the designers and end users.
- Reduced designer control and Increased tuning complexity.

What data do we need to represent ?



Spectre: Explosives. Shoot zombies before...they get too close.

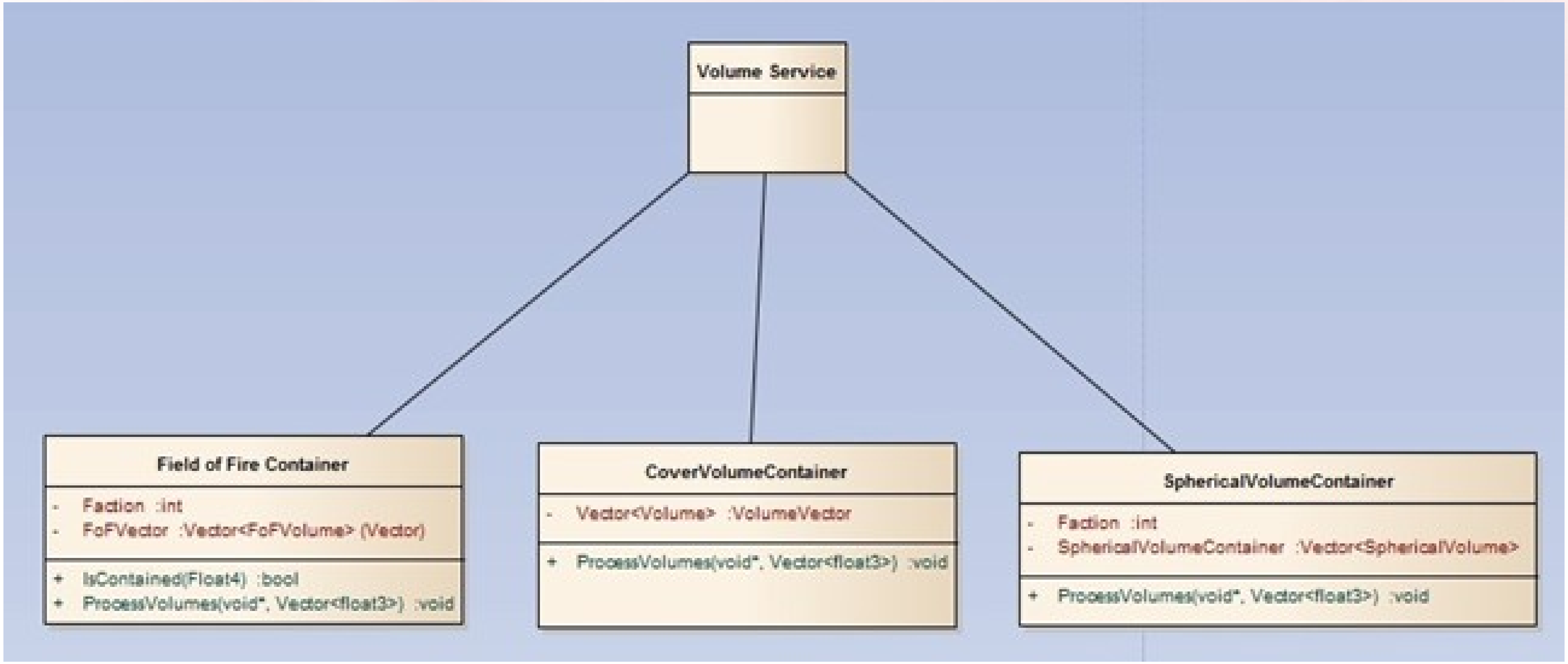
```
DS | 572605 : Release | 14.39  [7.6 - 30.1] | 14.6 avg | SIM:65.8ms | RENDER:23.4+2.7ms | Camera: (194.75 -2.82 -146.95) Turn: 255.3 Tilt:  5.8
```

What data is it useful to represent ?

- Fields of Fire (including Melee)
- Potential Cover points / Zones
- Points of Interaction for smart objects
- Hazard avoidance volumes, static and dynamic.
- Design Annotations
- World Events: synchronized events with listeners
- Progression representation for linear games, try encoding this into your hierarchical node graph
- Where agents can move and the movement of our friends and enemies. We'll come back to this.

How to represent the data we have chosen

- Having a fast, bespoke Volume service is key.



Volume Service

Field of Fire Container

```
-  Faction :int
-  FoFVector :Vector<FoFVolume> (Vector)
+  IsContained(Float4) :bool
+  ProcessVolumes(void*, Vector<float3>) :void
```

CoverVolumeContainer

```
-  Vector<Volume> :VolumeVector
+  ProcessVolumes(void*, Vector<float3>) :void
```

SphericalVolumeContainer

```
-  Faction :int
-  SphericalVolumeContainer :Vector<SphericalVolume>
+  ProcessVolumes(void*, Vector<float3>) :void
```

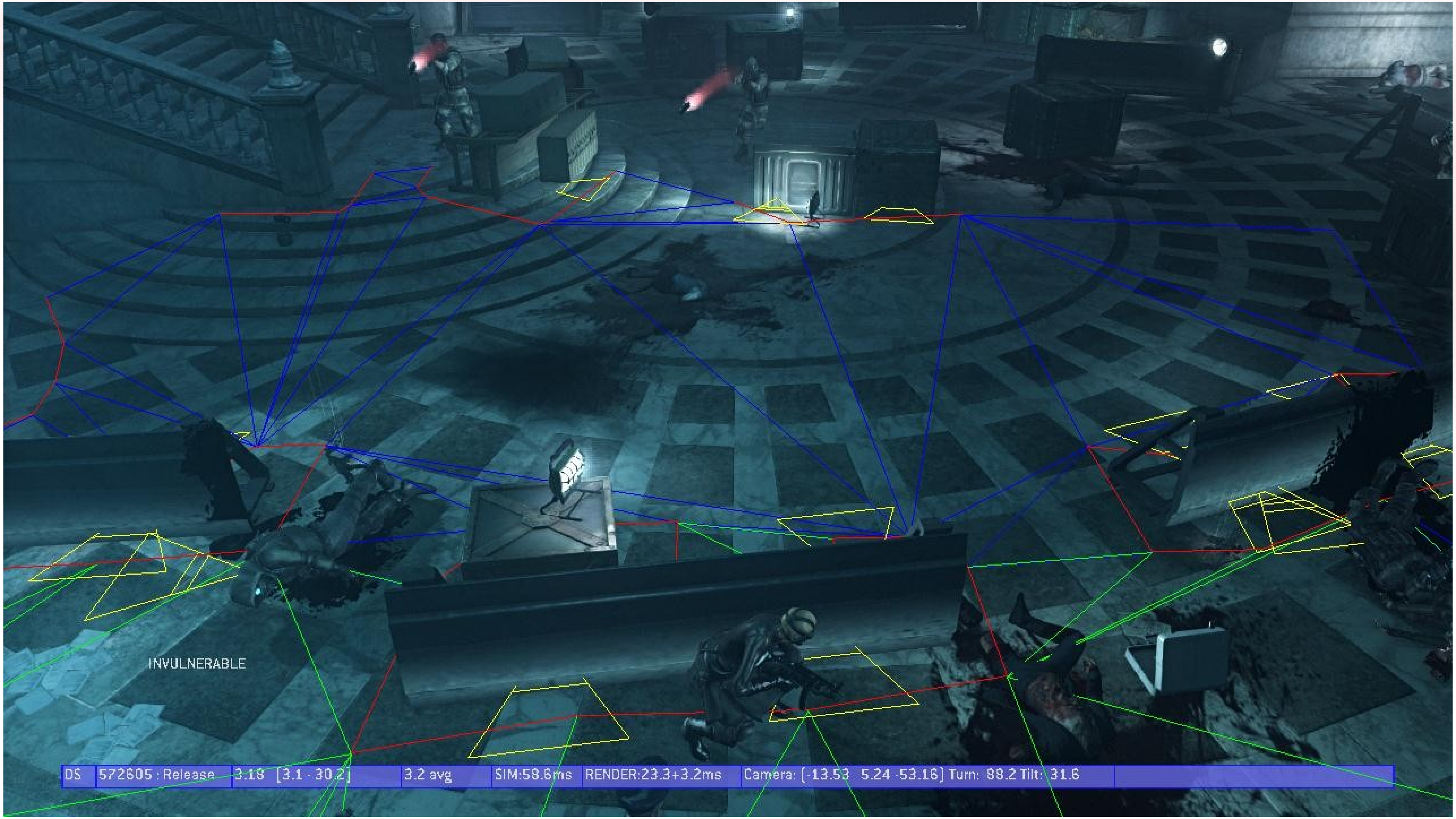
What about our traverse-able area representation ?

- Triangle Mesh, Node Graph, Grid, other ?
- Fixed size
- Don't respond well to dynamic events
- Expensive to compute complex heuristics that use world data.
- If performing volume queries it adds expense and complexity to path smoothing.

Benefits of a Navigation Mesh

- Commonly used and well understood algorithm
- Gives us a basis for some good analytic reasoning, i.e fast ray casts, and collision queries.
- Allows us to return 'corridors' to navigate within to easily facilitate variable width units and complex steering behaviors.

A Standard Navigation Mesh example



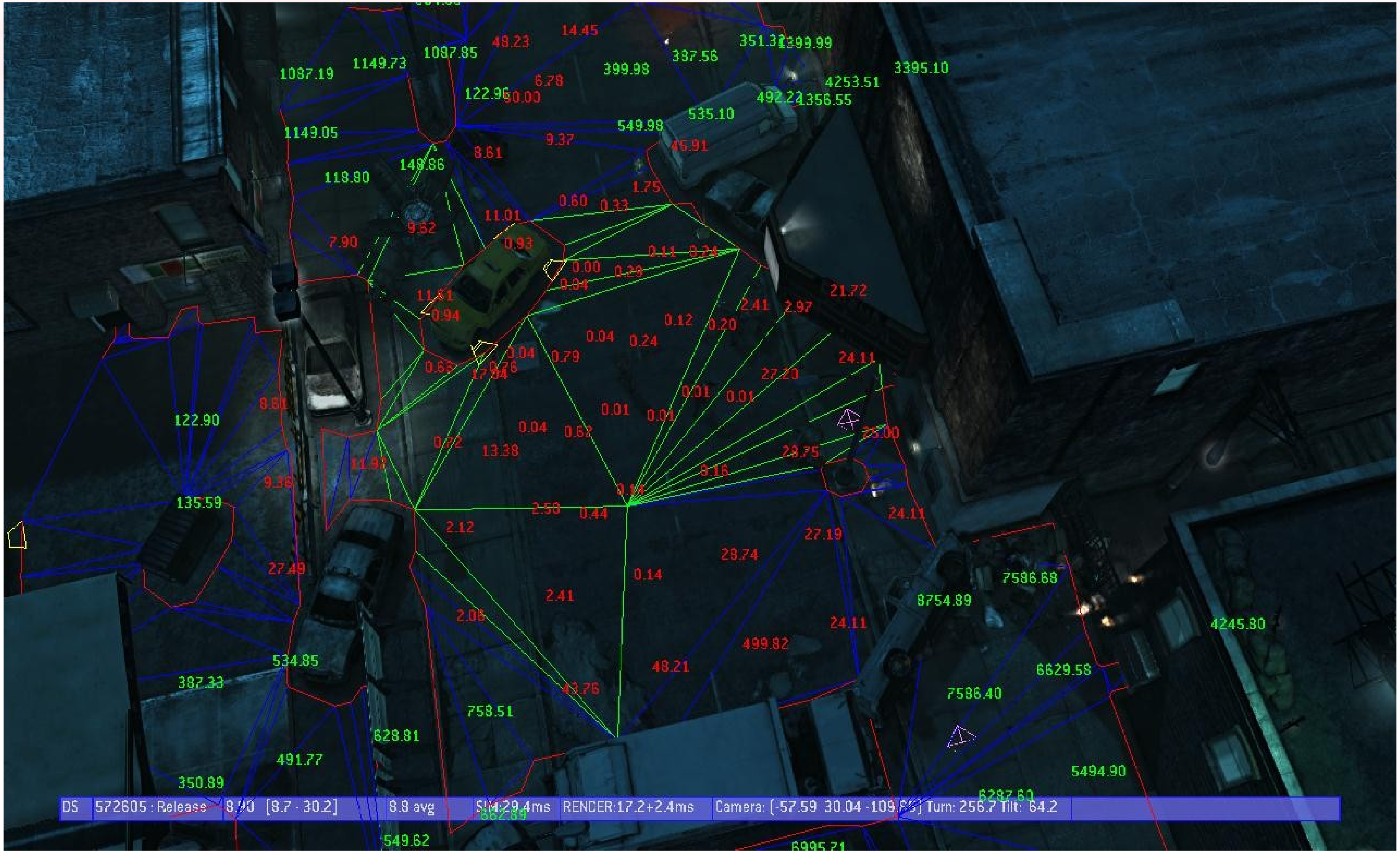
INVULNERABLE

```
DS | 572605 : Release | 3.18  [3.1 - 30.2] | 3.2 avg | SIM:58.6ms | RENDER:23.3+3.2ms | Camera: [-13.53  5.24 -53.16] Turn: 88.2 Tilt: 31.6
```

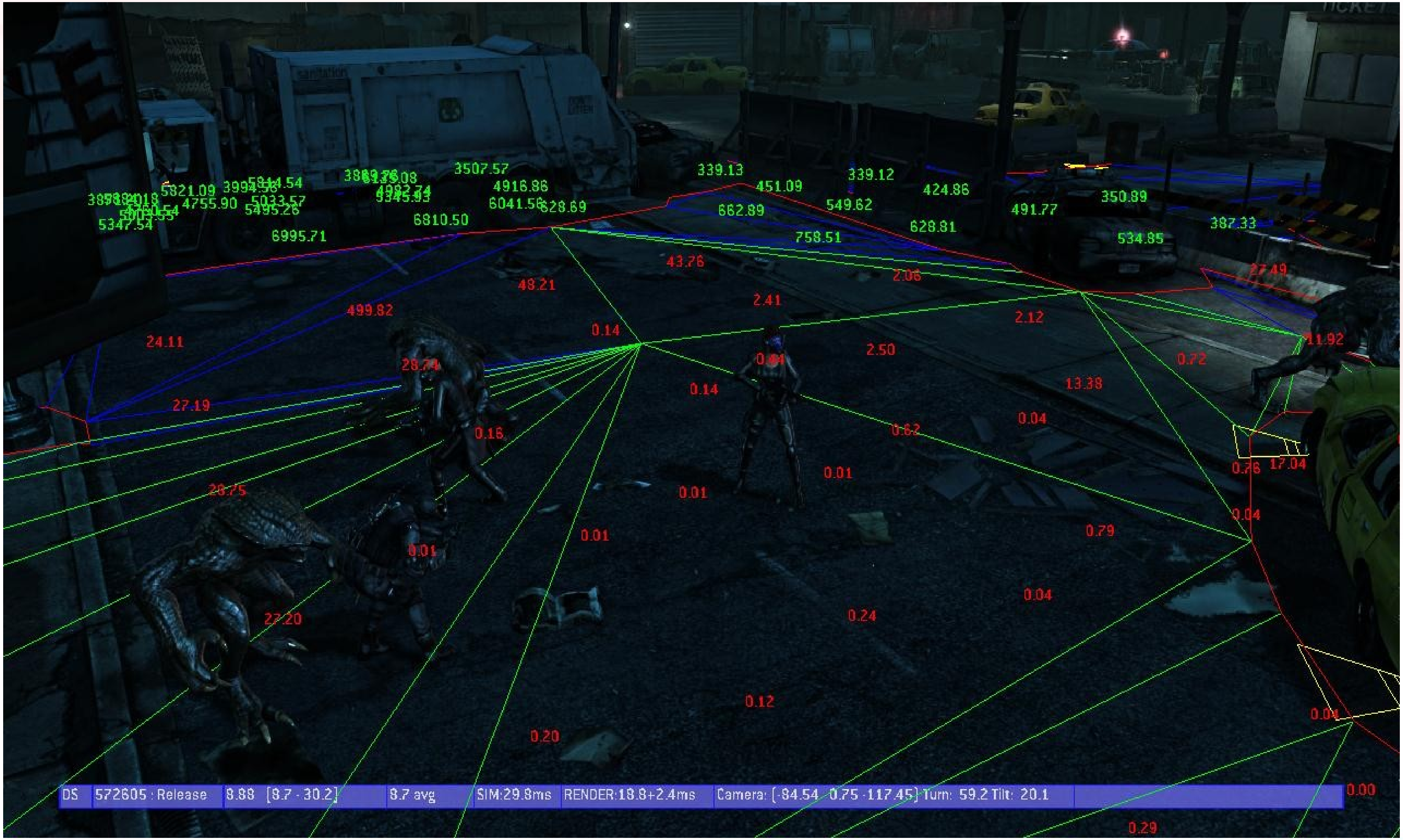
Terrain Reasoning : Overview of position scoring

- Reasonably common and well documented practice.
- Given a point and radius we collect all triangles in that radius
- We then rasterize the triangles to produce a list of points
- Points are then scored according to any number of desirability criteria, I.e in cover, in friendly field of fire, in enemy field of fire, distance to target, distance to move, has LOS to target etc, using a series of graphs, filters and boolean operations.
- highest scoring point is selected.

Position scoring in practice



```
DS  572605 : Release  8.90  [8.7 - 30.2]  8.8 avg  SIM:29.4ms  RENDER:17.2+2.4ms  Camera: [-57.59  30.04 -109.63]  Turn: 256.7  Tilt: 64.2
```



```
DS  572605 : Release  8.88  [8.7 - 30.2]  8.7 avg  SIM:29.8ms  RENDER:18.8+2.4ms  Camera: [-84.54  0.75 -117.45] Turn: 59.2 Tilt: 20.1
```

The Problem

- There is a disconnect between the information used to calculate our desired target position and the information we consider when choosing a path to get to it.

Common solution

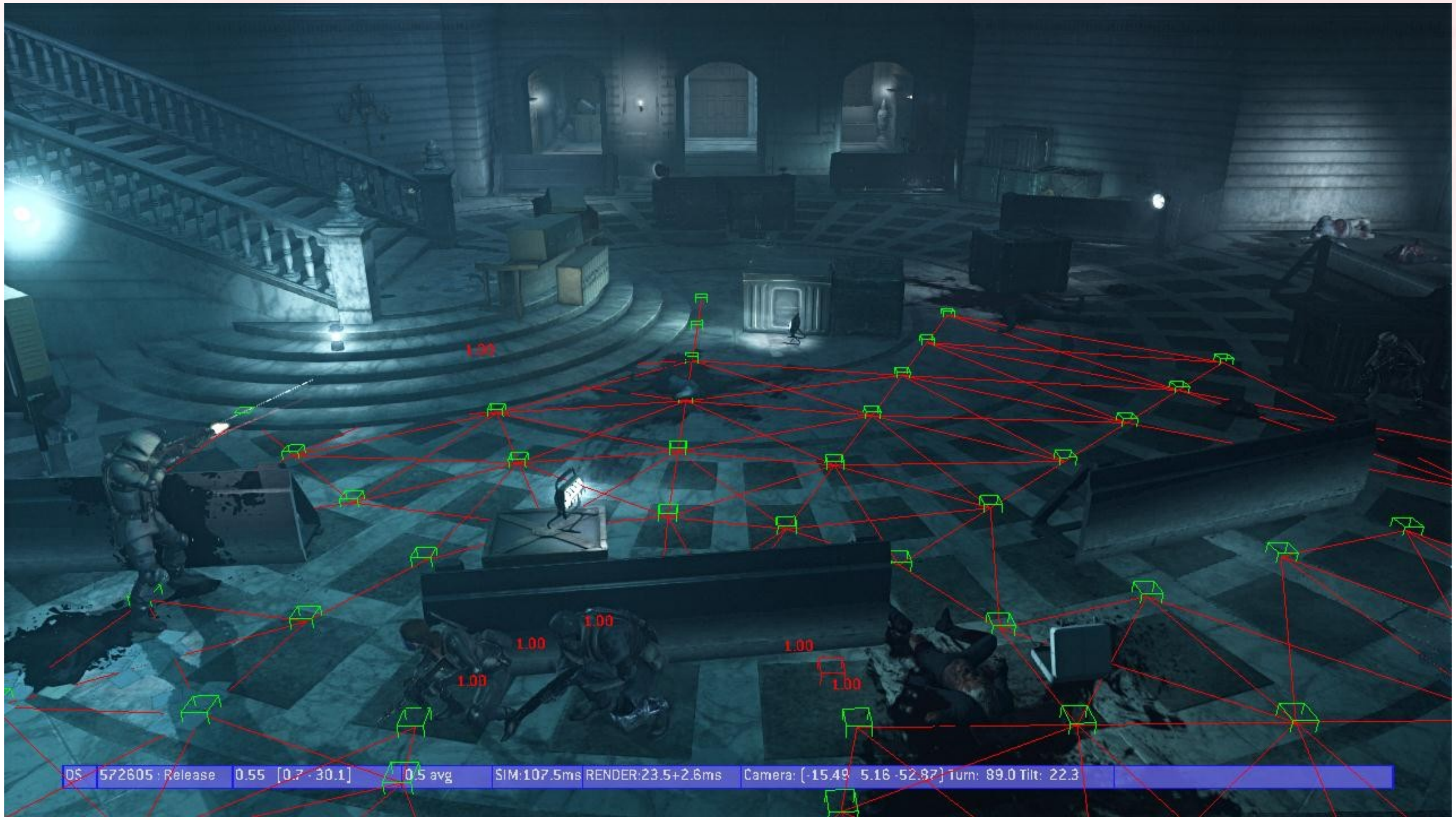
- Use an influence map
- Let steering worry about it
- Or use a much higher resolution world representation

Is this ideal ?

- Leads to 'bumper car' steering or movement that does not look 'deliberate'
- Lots of re-pathing is common
- Limits the reasoning we can do on our choice of destination
- High resolution solutions slow down pathing (even with HPA*), bloat memory and still provide scope for edge cases and contain a lot of wasted data.
- Makes pathing much more non-deterministic and limits our ability to reliably broadcast paths.

Dynamically Generated Node graph

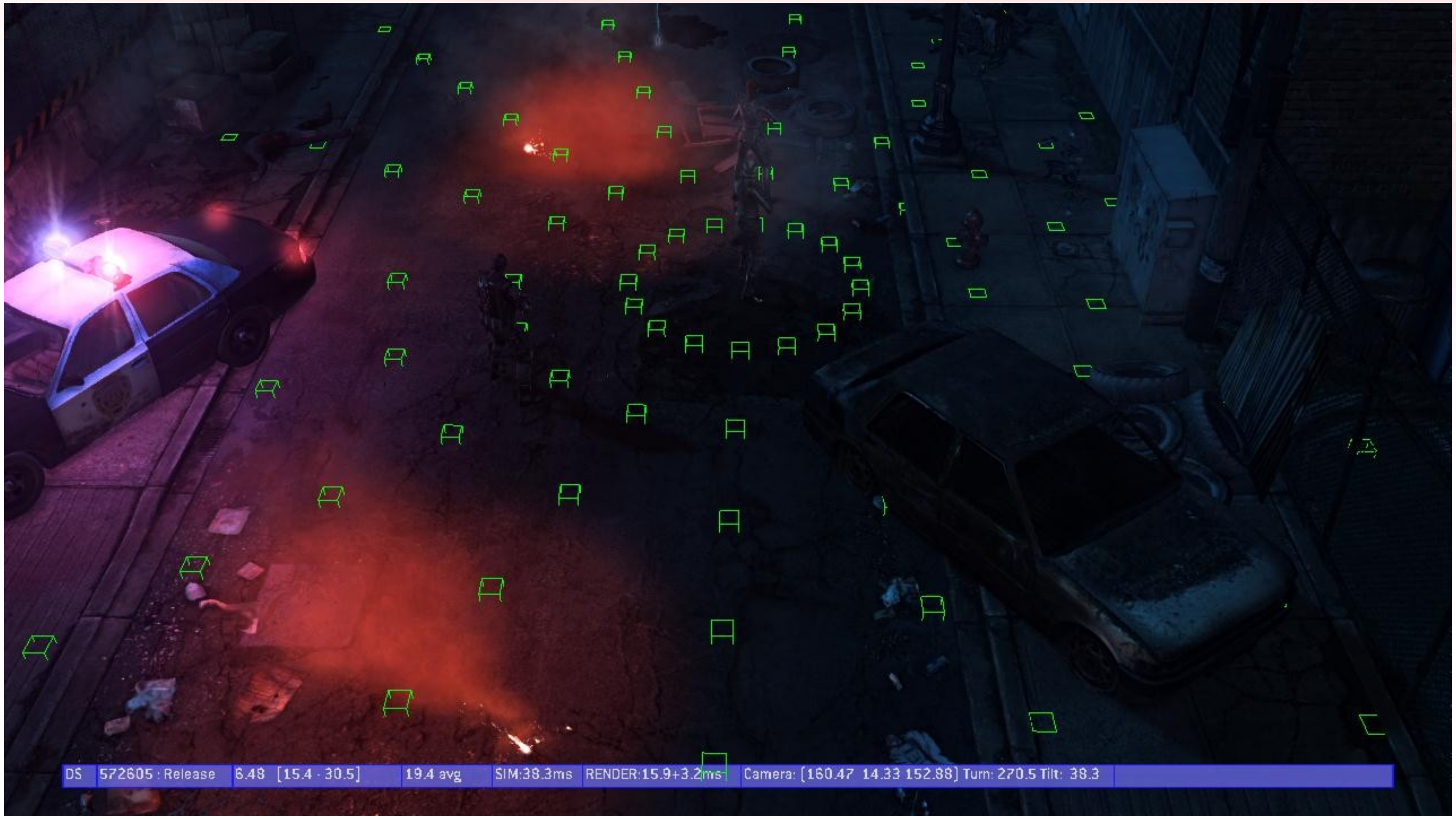
- We generate a new dynamic node network on top of the nav mesh and use that for both position scoring and navigation



```
DS  572605 : Release  0.55  [0.7 - 30.1]  0.5 avg  SIM:107.5ms RENDER:23.5+2.6ms  Camera: [-15.49  5.16 -52.87] Turn: 89.0 Tilt: 22.3
```

Algorithm 1 : Node generation

Generate a circular map of nodes at the point of interest (player position, squad position etc)

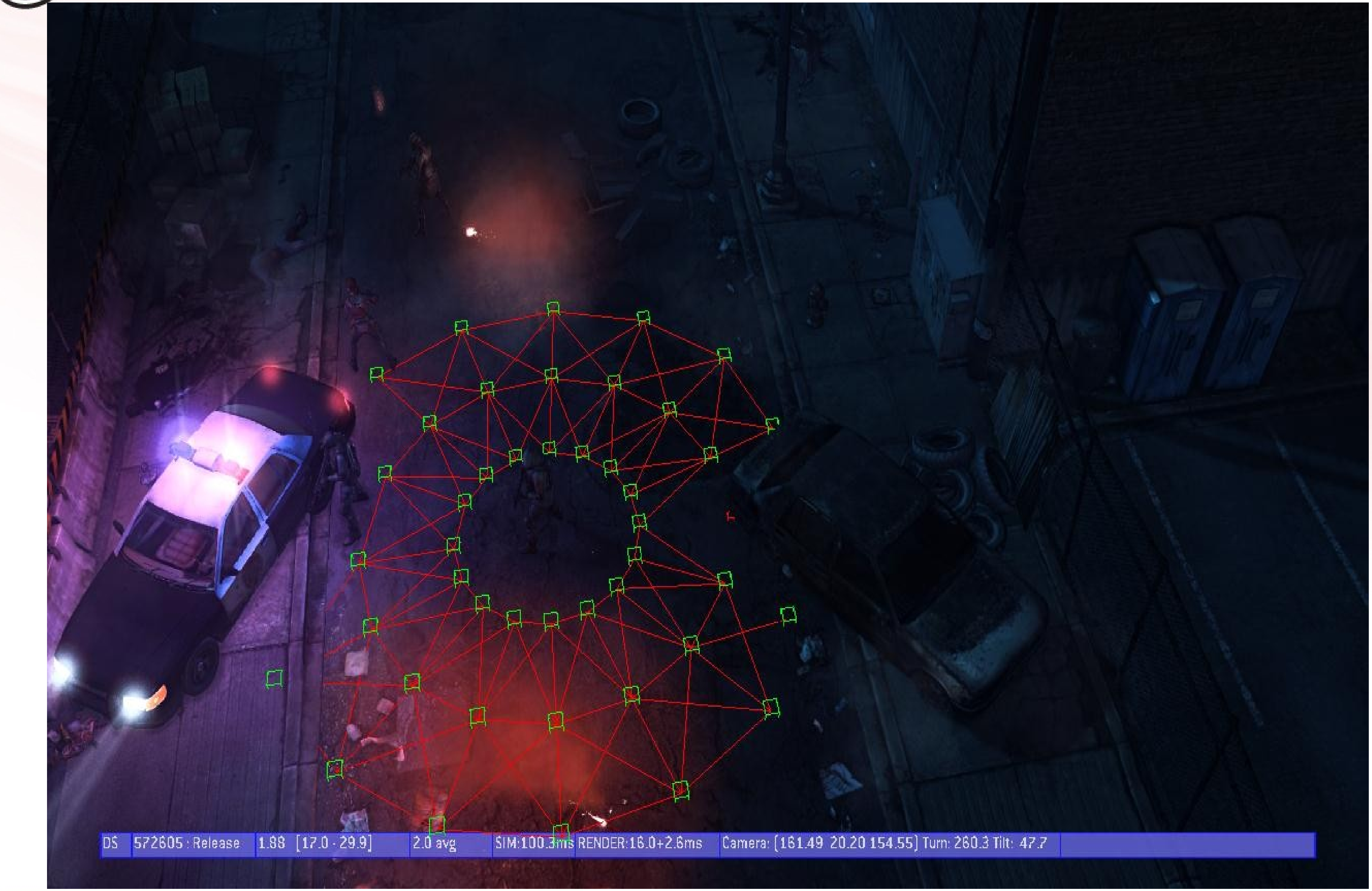


```
DS 572605 : Release 6.48 [15.4 - 30.5] 19.4 avg SIM:38.3ms RENDER:15.9+3.2ms Camera: [160.47 14.33 152.88] Turn: 270.5 Tilt: 38.3
```

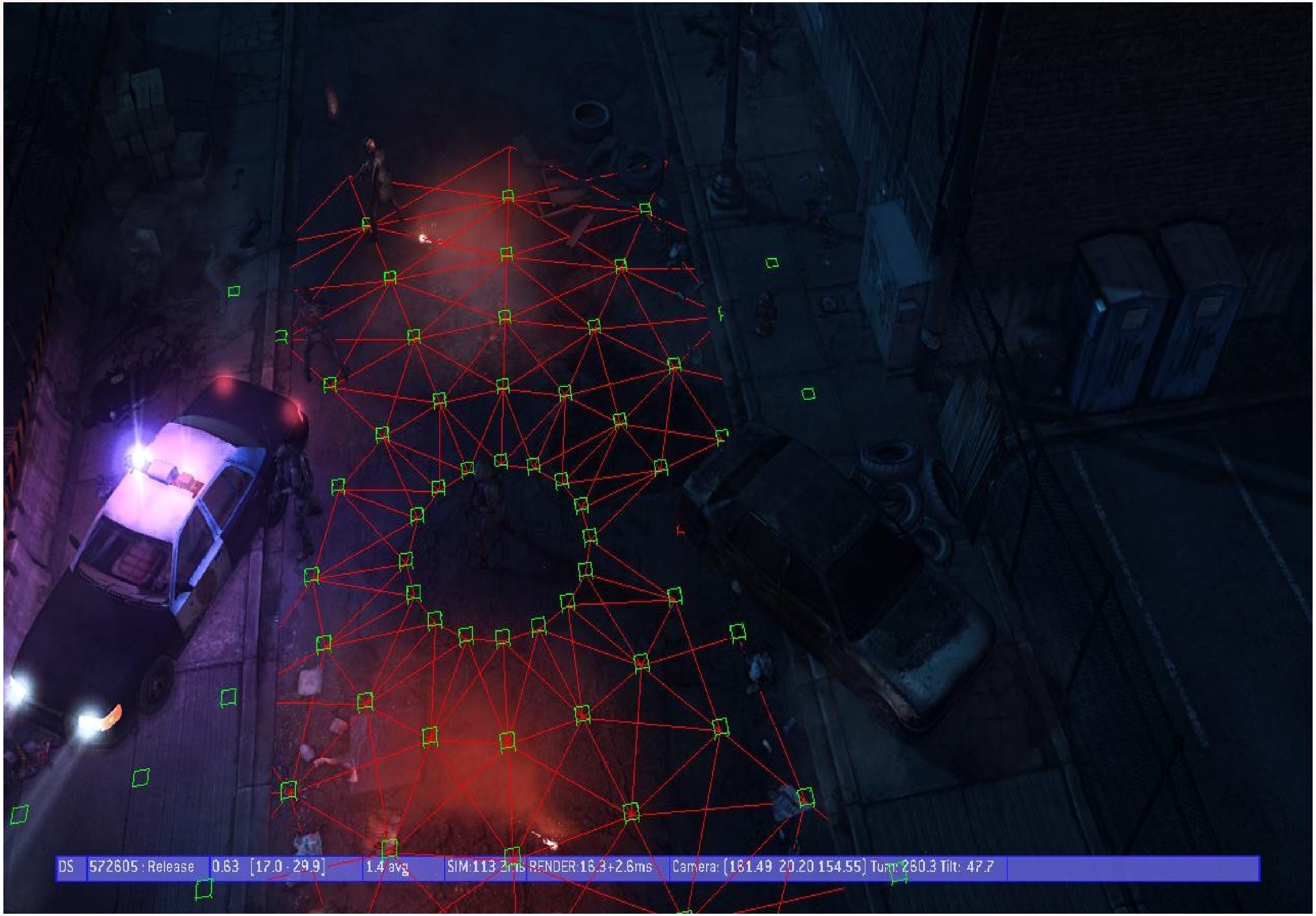
Algorithm 2 : Graph connection

Connect the network ->

- Each node to its neighbor on its own Ring, its previous and next ring
- If a connection fails by C% then clamp the node to this new position (very helpful for snapping nodes to cover)
- Clean up nodes with zero connections



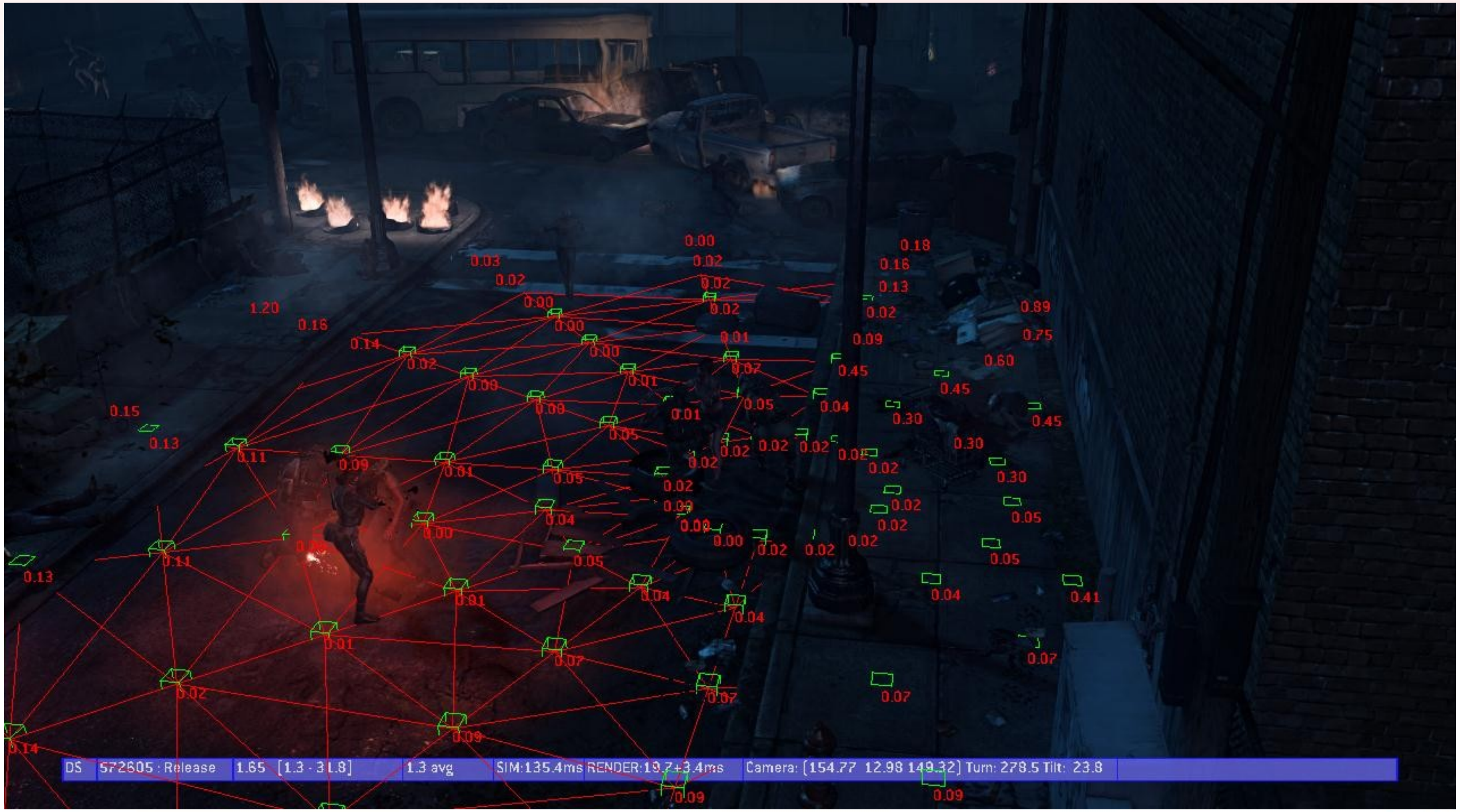
```
DS  572605 : Release  1.88  [17.0 - 29.9]  2.0 avg  SIM:100.3ms RENDER:16.0+2.6ms  Camera: [161.49  20.20 154.55] Turn: 260.3 Tilt:  47.7
```



```
DS  572605 : Release  0.63  [17.0 - 29.9]  1.4 avg  SIM:113.2ms RENDER:16.3+2.6ms  Camera: (161.49 20.20 154.55) Turn: 280.3 Tilt: 47.7
```

Algorithm 3 : Scoring

We can now score the points against any criteria desired from our world data.



```
DS   572605 : Release   1.65  [1.3 - 31.8]   1.3 avg   SIM:135.4ms RENDER:19.7+3.4ms   Camera: [154.77  12.98 149.32] Turn: 278.5 Tilt:  23.8
```

Optional Extras

- Insert design annotation or 'tactical positions' into the network
- Clamp nodes to dynamic volumes.
- Leashing area's or Preferred zones

Isn't that going to be very slow ?

- Potentially - So how do we get around that ?
- We localize it to the player('s) action areas or each squads locale - You could think of it as a lower level of your path finding hierarchy.
- Lower the resolution of the network when possible
- We can easily evolve it over a number of frames.
- Optimize your NavMesh raycast
- Keep data structures of the node network tight and organized.
- Spacial partitioning structure for NavMesh.
  - 32 x 32 Node graph over 50M radius -0.77ms on PS3 PPU.

Specializing your NavMesh raycast.

- General Specialization to early out clauses for position off NavMesh
- Cache the face index of each node in its own structure and pass that back to the raycast function to avoid expensive look up's on data that can be cached from previous inquiries
- Pre-cache the 'face neighborhood'
- Exploit the face winding order to provide hints for any additional lookups.

Navigating across the dynamic network

- Simply overload our heuristic function and neighbor expansion access function in A*
- Heuristic cost for crossing a node is Distance (d) + f(Score)
- Path finding from and too the network is trivial
- Smoothing becomes much simpler – tho still more complex than simple distance based pathing however smoothing is not always needed, the spherical design produces quiet smooth paths.

Benefits.

- Memory efficient
- Fast to compute paths and to score points
- Unifies position picking and path finding
- Also can act as a path finding LOD - Agents (like Zombies) that do no need complex reasoning can use the base Nav mesh.

Avoiding Pitfalls.

- Take the world representation as a snap shot in time.
- Commit to actions -> convey decision making through audio / animations / Archetypes
- Example : Crossing the players field of fire.
- Additionally the use of smoothing and rate of change integrator can be used if we have the space to store history.

Any Questions ?

Connect

Twitter : @WintermuteBC

Email : nbamford@slantsixgames.com