



STREET SMARTS

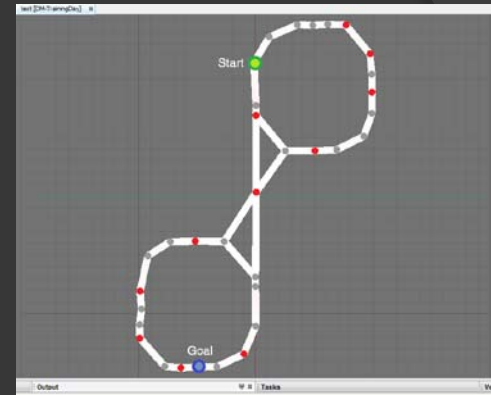
Better Pathfinding for Shooter Bots

Background

- Unreal Tournament 2004 – one of the most popular and critically acclaimed First Person Shooters. Gameplay revolves around closed arenas where human and computer-controlled opponents (bots) fight.
- Pogamut – A framework built and maintained by a team at Charles University, Prague, CZ. Utilizes Java and Unreal's built in scripting support to create an API for reprogramming the game's bots.

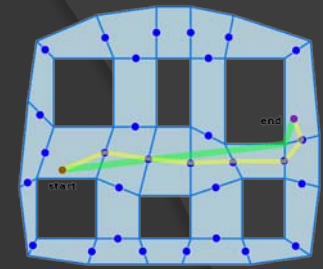


Project idea



- ⦿ A* is used for implementing path-finding around the map for the bot, assuming his medium and long term intelligence set a goal that is not immediately reachable. Still, under the current system, the bot “naively” runs between the given way-points, following the shortest path.
- ⦿ My proposal is to modify this system, so that A* does not only take under consideration physical space (i.e. shortest path), but such variables as cover, fire zones, bonuses, etc.

Why?

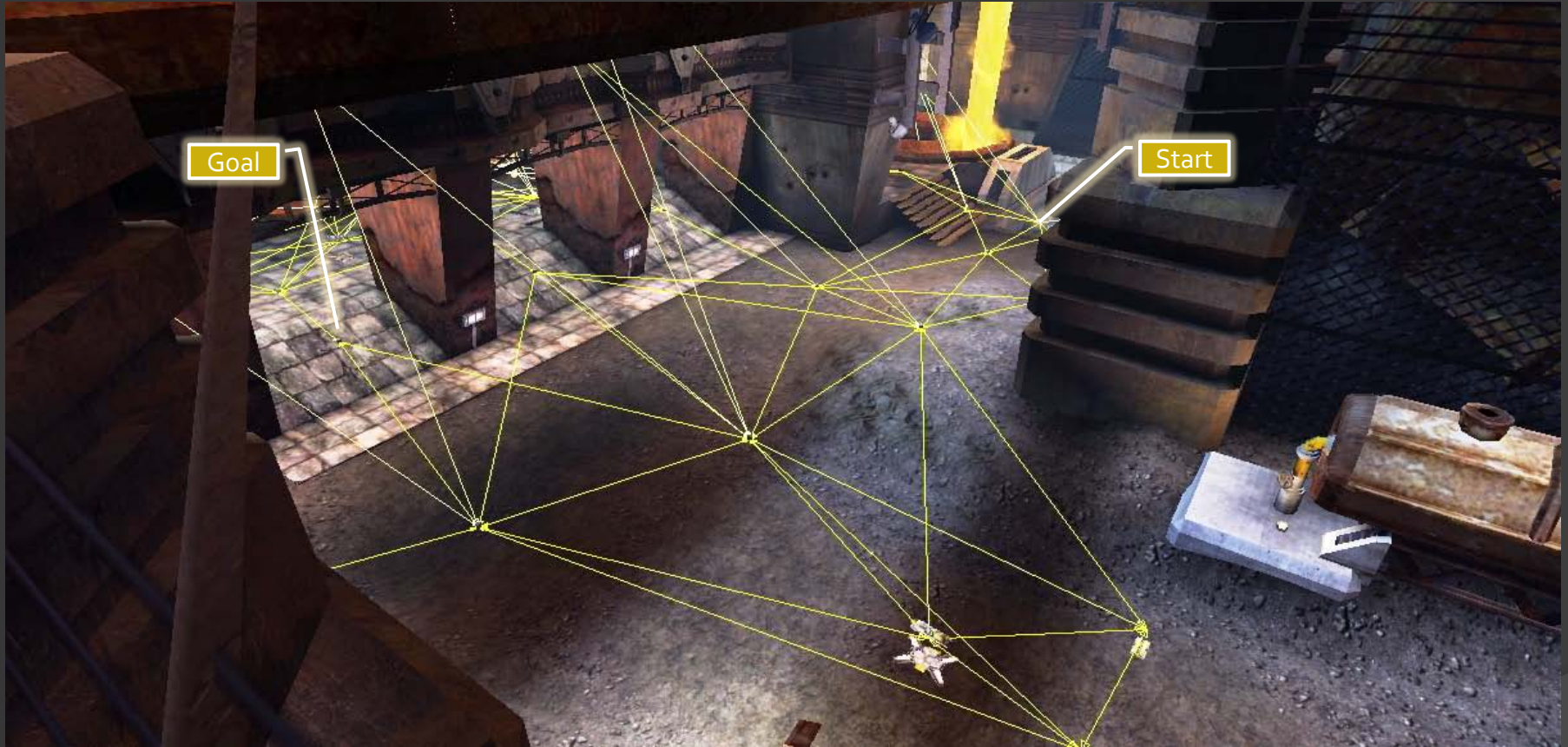


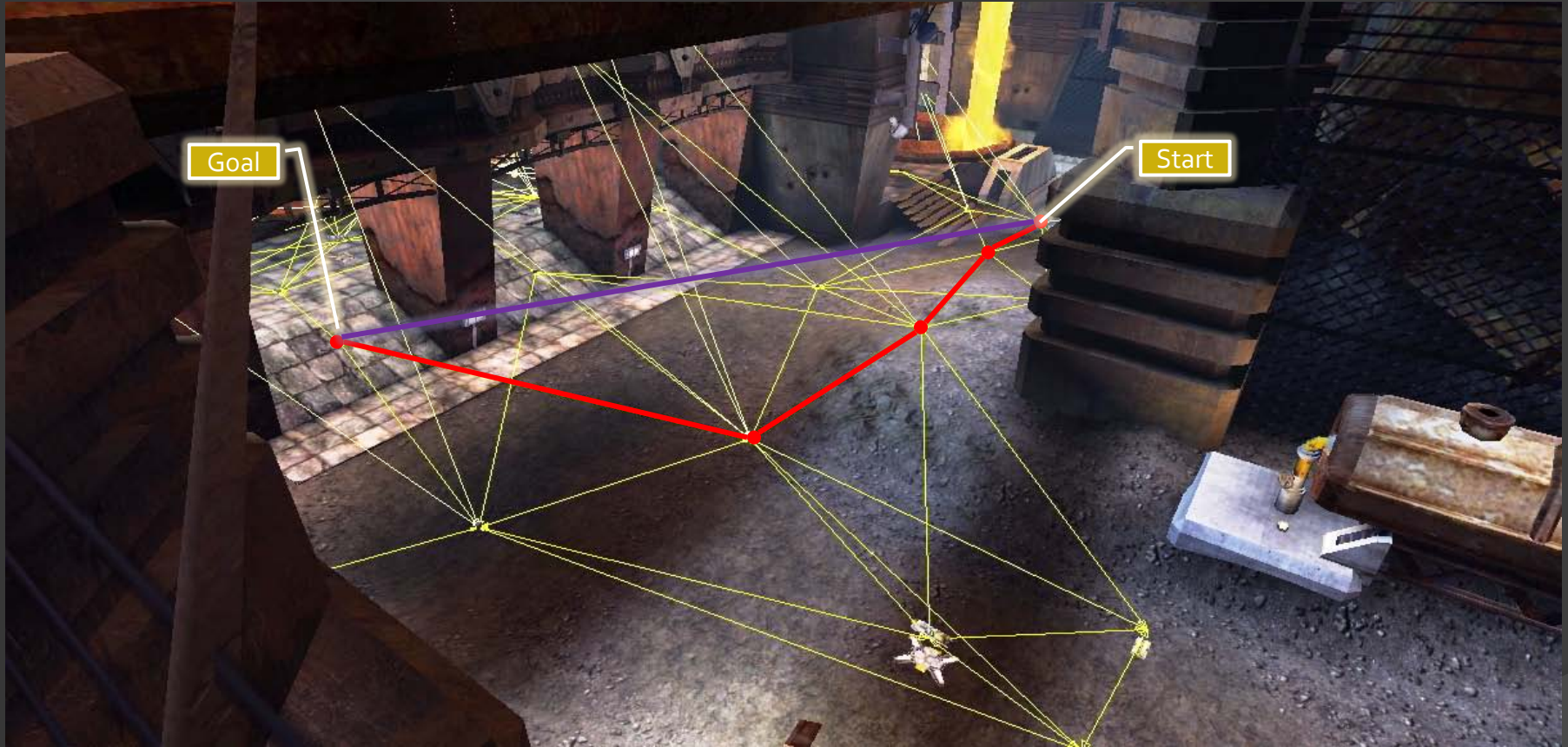
- **Implementable** – To actually work and improve on a modern, popular and commercial computer game. Only modifies the pathfinding module, thus allowing **all** bots to use the enhanced intelligence (Plug n' play).
- **The Genius of A*** - Using A* for only finding a shortest path is a waste of the algorithm's strength. We saw earlier in the course how A* and its variations are on the forefront of AI. Instead of using it to affect a small part of the AI, why not search through **tactical** space and make it worth the expense?
- **Efficient** – We're doing A* anyway, the effort in calculating the search space a bit differently is computationally negligent.

A* reminder

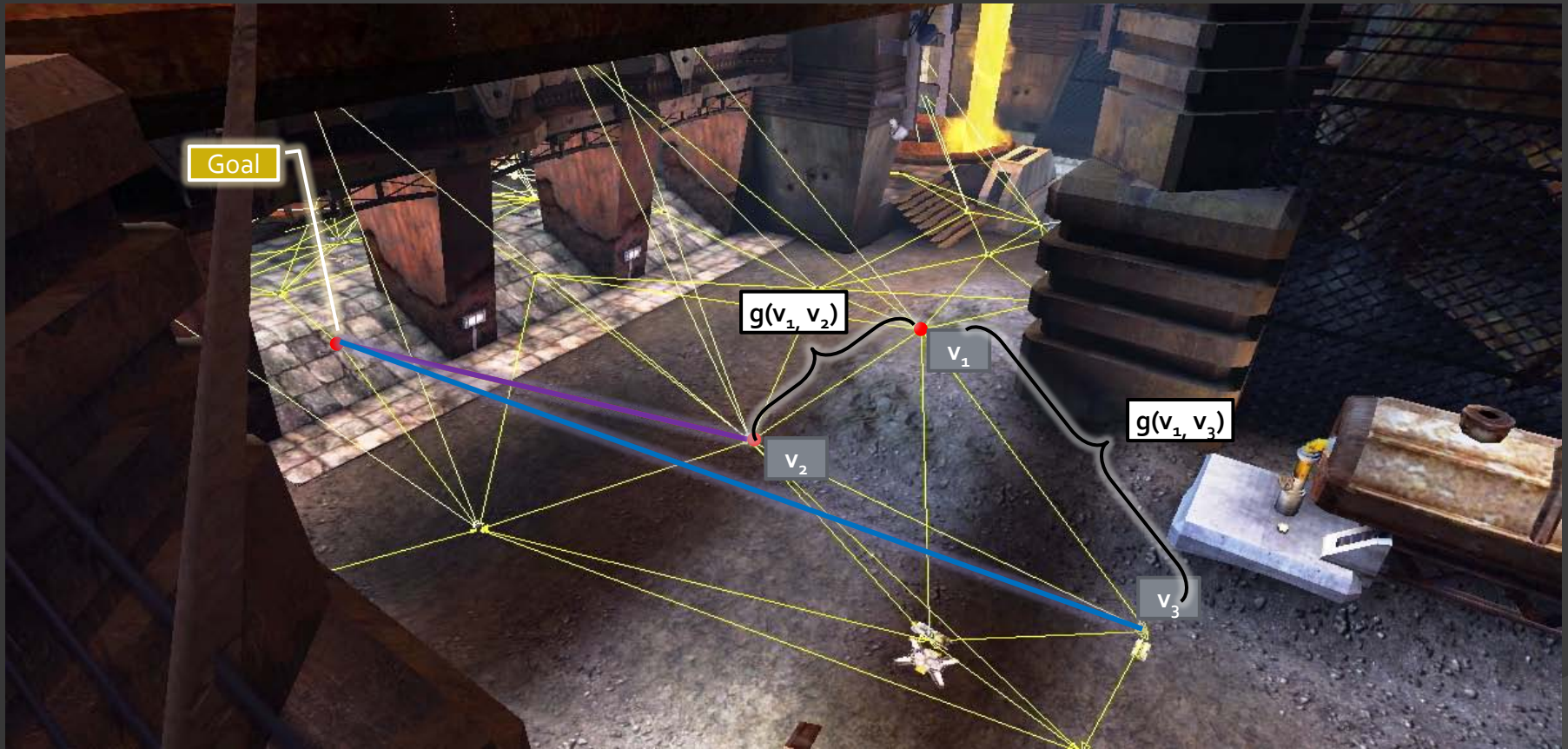
- ⦿ In A*, the next vertex chosen for expansion is the one that minimizes the evaluation function: $f(v) = g(s \rightarrow v) + h(v \rightarrow r)$
- ⦿ Where s is the starting vertex and r is our goal.
- ⦿ g – sums the weights between s and v . In the original A*, these weights equate to Cartesian distance.
- ⦿ h – Best guess at how much it will take to travel from v to r . The original: also “as the crow flies” Cartesian distance. No path can be shorter than this, ensuring an optimal result.







The length of the red line – $g(s,r)$
The length of the purple line – $h(s,r)$



Standing on v_1 , which node is better for me?
 v_2 , because: its g is shorter
its h is shorter, i.e. it's closer to the goal.

Our new g

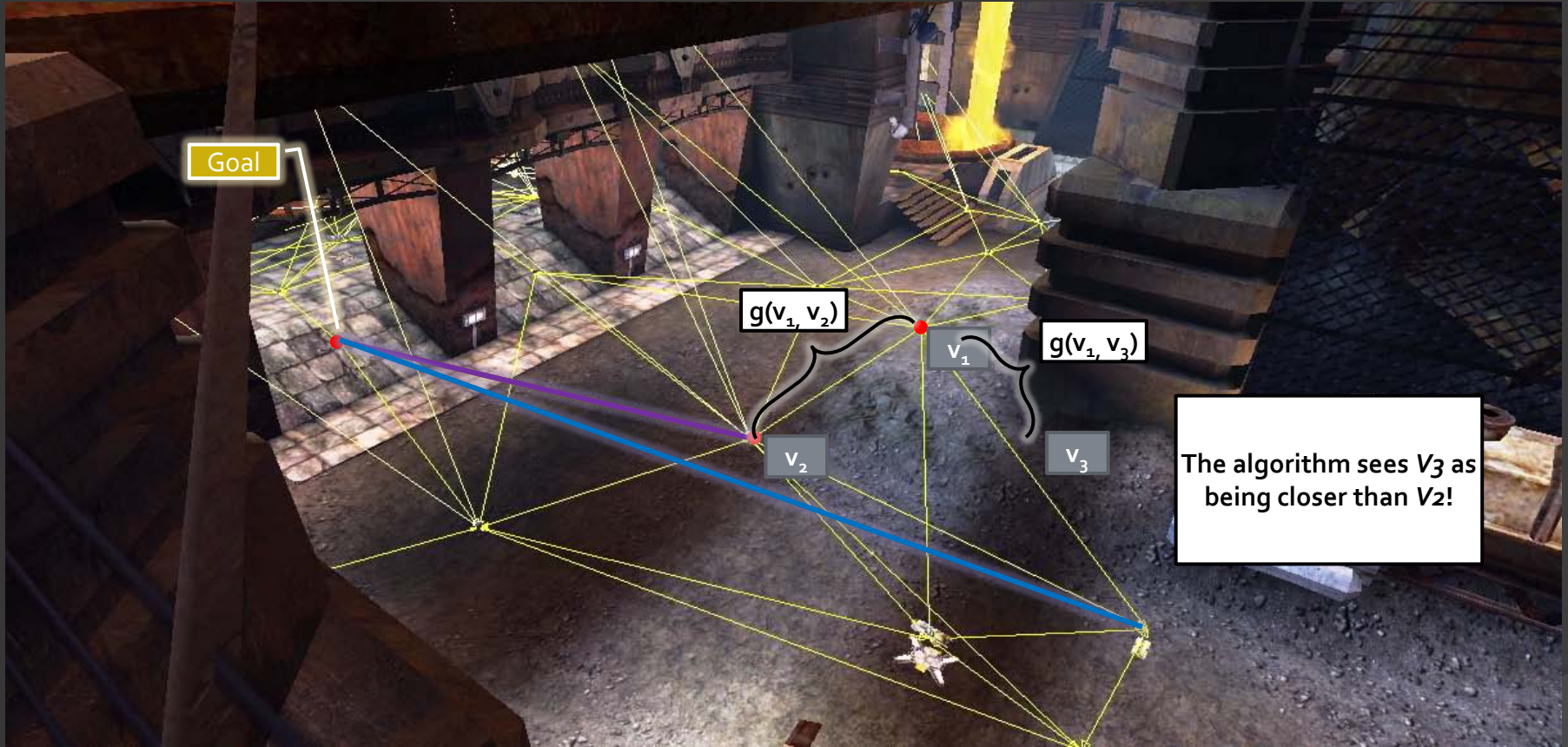
- Weighted sum of various considerations.

$$g(v_1 \rightarrow v_2) = \sum_{i=0}^n [w_i \cdot m_i(v_1 \rightarrow v_2)]$$

$m_i \in [0,1]$ The tactical considerations.	$w_i \in [0,1]$ Their matching weights.
Distance between the two points	Normalized (max distance ⁻¹)
Items lying on the path between the two points (Health, ammo, weapons, adrenaline)	Depends on Bot's status. i.e if he is at 100 health (max is 200), then the health weight is 100/200 = 0.5
Players between the two points, the weapons they carry, what team are they on	Depends on Bot's status. Average of health and current weapon/ammo.
Cover value of point b	<i>Not implemented due to time constraints.</i>

In code:

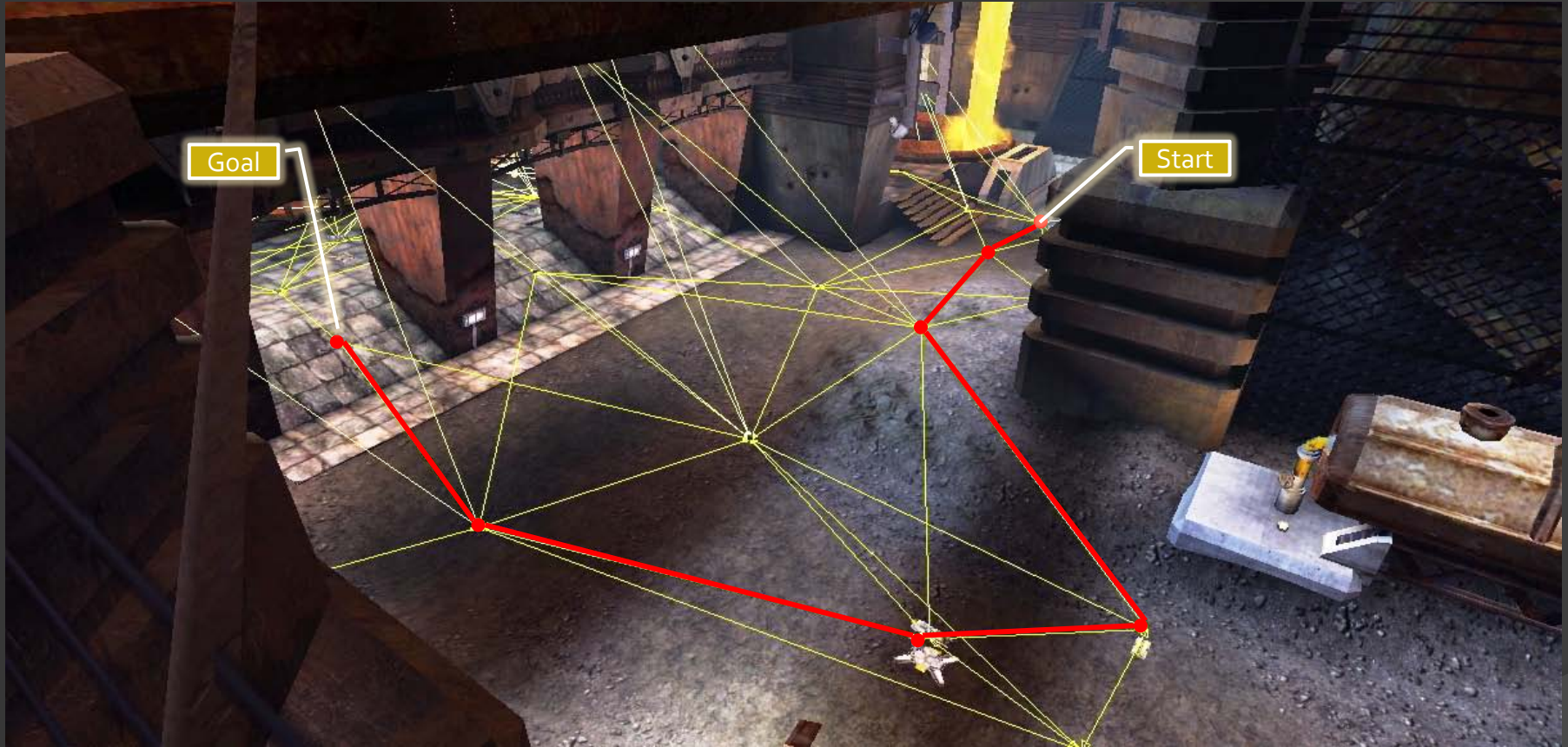
```
def g(a, b):  
    gvars = [distance(a, b) * weights["dist"],  
             items(a, b),  
             cover(a, b) * weights["cover"],  
             players(a, b) * weights["players"]]  
    return sum(gvars)
```

Standing on v_1 , which node is better for me?

v_3 !, because:

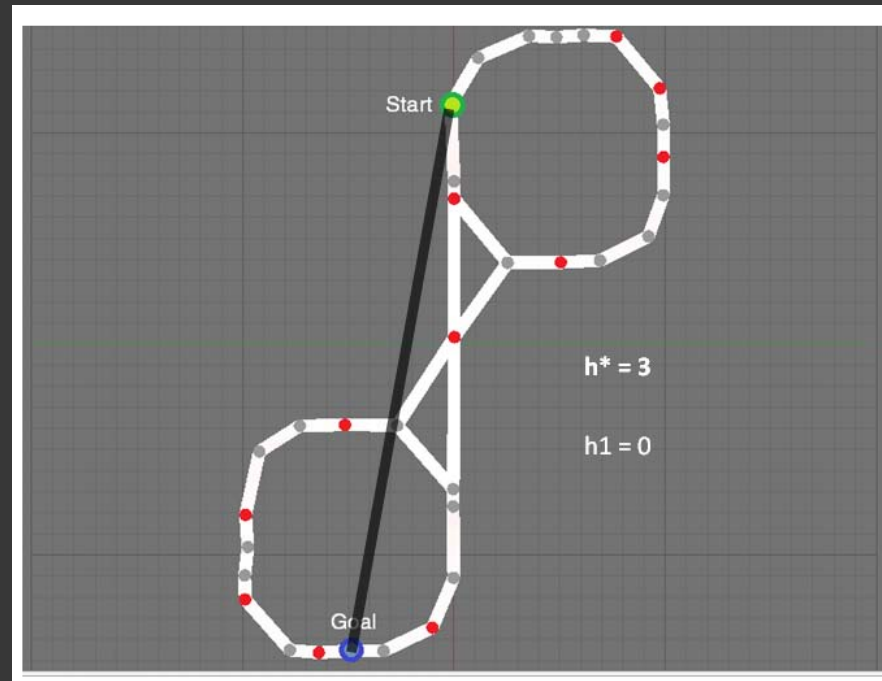
its g is shorter - cause of the item, and the gun near it
its h is probably longer, but remember $f = g + h$,
and h isn't so bad to deter the bot.



The completed “smart” path

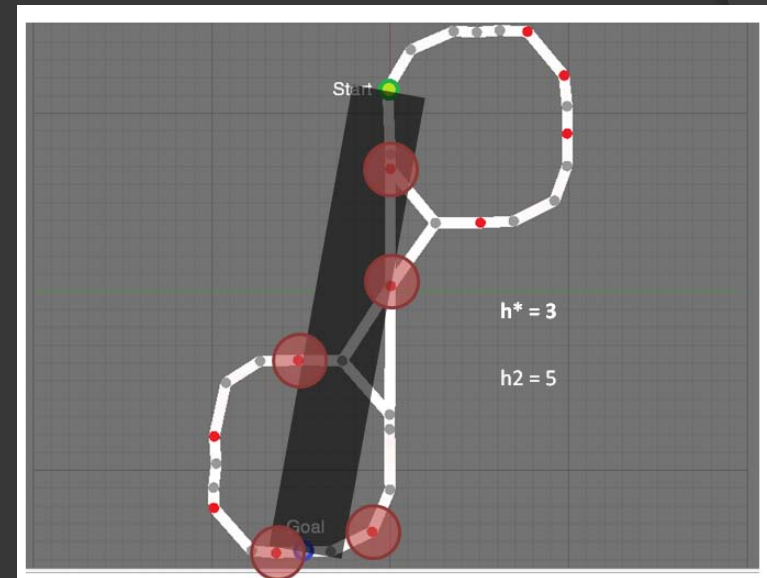
Heuristic problem:

- ⦿ We have a good heuristic for distance, but how can we make an educated guess towards how many features we might pass on our way?



Solution:

- What if the line was thicker? What if each feature had a large “radius”?
- Chosen because it is computationally cheap, and requires no pre-calculation of map features.
- Other options examined included:
 - Probability (average spread of features * distance to travel)
 - Potential fields (send a particle with velocity \mathbf{v} towards goal, features have “gravity”, see how far his total path was.



Empirical results

① 1 on 1: enhanced bot vs regular

- 20 games played, up to 25 kills.

	Regular	Enhanced	
Average Lifetime	16.5 seconds	20.2 seconds	+18%
Average Kills	17	25	+32%

⑤ 5 on 5 team game: enhanced bots vs regulars

- 5 games played, up to 100 kills

	Regular	Enhanced	
Average Lifetime	11.4 seconds	12.6 seconds	+9%
Average Kills	90	100	+11%

Video!



<http://www.vimeo.com/18221321>

Conclusions

- On the whole, the enhancement works. Improves bot's "killer" efficiency by ~16%, and with no bugs or significant resource use.
- Problems:
 - Cover not implemented due to time constraints.
 - Heuristic could be better. Maybe use Temporal Difference Learning.
- Further work: Perhaps replacing the entire's bot AI with a search-tree method. Or at least having the navigation graph effect behavior (i.e. if good place to crouch – send "crouch!" to higher level of AI).

If you want to try it...

- ◎ Me: benjycook@hotmail.com. I can host servers with the bots if you wish to see them in action.
- ◎ Source, this presentation, and more available at <https://code.google.com/p/smart-path-unreal-2004/>
- ◎ Requires NetBeans, Pogamut and UT 2004.
- ◎ International competition in building human-like bots: <http://www.botprize.org/>

References

Smed, J. and Hakonen, H.

Algorithms and Networking for Computer Games

(2006) John Wiley & Sons, Ltd, Chichester, UK, pg. 96-113.

Amit's A* Pages

<http://theory.stanford.edu/~amitp/GameProgramming/>

Don't follow the shortest path!

May 26, 2008 at 4:49 pm · Christer Ericson

<http://realtimecollisiondetection.net/blog/?p=56>

Pogamut Framework for UT Bots

<https://artemis.ms.mff.cuni.cz/pogamut>

Pearl, Judea

Heuristics: Intelligent Search Strategies for Computer Problem Solving.

(1983). New York, Addison Wesley. *Formal Properties of Heuristic Methods, Abstract Models for Quantitative Performance Analysis, Searching with Nonadmissible Heuristics.*

Planet Unreal Weapons Guide

<http://planetunreal.gamespy.com/View.php?view=UT2004GameInfo.Detail&id=26&game=4>

Beyond Unreal weapons Guide

http://liandri.beyondunreal.com/Unreal_Tournament_2004#Weapons