Abstraction and Refinement in Games with Dynamic Weighted Terrain

Nathan R. Sturtevant
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
nathanst@ualberta.ca

Devon Sigurdson, Bjorn Taylor, Tim Gibson
Improbable Canada Inc.
Edmonton, AB, Canada
{devon, bjorntaylor, timgibson}@improbable.io

Abstract

This abstract looks at one version of the pathfinding problem in games and discusses how it motived our recent work at the AIIDE 2019 conference.

Pathfinding in Games

In 2019 Improbable Inc's Canada studio began to collaborate with the Moving AI Lab at the University of Alberta. Our first task was to look at the problem of pathfinding in games to see if we could open up new design possibilities for character pathfinding in Improbable's games. The first result of this collaboration was published at AIIDE 2019 (Sturtevant et al. 2019); in this abstract we give a high-level overview of the motivations for that work and the general approach that was adopted for the initial technologies built into the game.

We begin by looking at some of the properties of pathfinding in games that makes this problem unique from other applications, such as navigation for cars on roads. However, the games industry itself is quite broad, so this cannot taken to be representative of all games. Given this caveat, some unique features of pathfinding in games include:

- **Free space traversal**: Characters are able to move through free space and are not restricted to a finite number of traversable edges. For instance, the underlying terrain in the Unreal game engine[1] is represented by a NavMesh (Tozour 2002), which uses the Recast toolset[2] to build a polygonal representation of free space in the world.
- **Dynamic terrain**: Characters are free to modify the terrain. This could be by building buildings, bridges, or by harvesting forests to turn them into grassland, but could also be a result of destruction by enemies or environmental forces. The terrain representation must be able to be changed at runtime to reflect these changes. Thus, any techniques used to speed up pathfinding must be amenable to re-computation at runtime.
- **Different terrain costs**: While all terrain may be traversable, different terrain types can have different costs. It is preferred that characters follow roads, when they exist, and use other terrain types when appropriate. Thus, a pathfinding system should be able to reason about terrain and return appropriate paths.
- **Per-character terrain costs**: While human characters will prefer to travel on roads, other types of creatures might prefer to stay off the roads to avoid being seen and travel within forests.
- **Dynamic terrain costs**: Terrain costs can change during gameplay either from casting a spell, or from other dynamic events such as weather. While taking a boat across a lake in normal weather may be fine, characters may dynamically avoid water during inclement weather or when fighting a creature with lightning abilities. Taken together with per-character terrain costs, this suggests that changes in terrain cost should not require significant re-computation.
- **Fast computation**: Pathfinding requests typically need to be satisfied in 1ms or less, meaning that optimality (in terms of path length) is often sacrificed in return for faster results.
- **No clear definition of optimality**: Many pathfinding applications prefer optimal paths with respect to metrics such as distance or time. In games there are often additional artistic constraints on movement that make it difficult to even define an optimal path, so traditional optimality metrics are only an approximation of path quality.

Approach: Abstraction and Refinement

The general approach that we took uses *abstraction* to build a high-level representation of the world. Paths found in the high-level representation can then be *refined* to paths in the real world. Even optimal approaches such as contraction hierarchies (Geisberger et al. 2008) use abstraction and refinement, but we use a class of approaches that have been well-studied (Holte et al. 1996; Botea, Müller, and Schaeffer 2004; Sturtevant and Buro 2005; Bulitko et al. 2007; Sturtevant and Jansen 2007; Harabor and Botea 2008; Pelechano and Fuentes 2016) and that produce suboptimal

[1]https://www.unrealengine.com

[2]https://github.com/recastnavigation

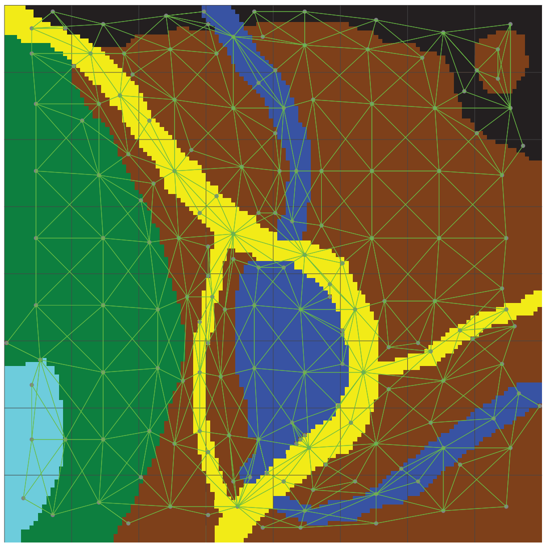

Figure 1: The dynamic terrain abstraction on a larger map with five different terrain types.

results. These approaches have also been used in the games industry (Sturtevant 2007; Alain 2018).

Our particular implementation is a variant of existing approaches optimized for the problem definition above. In particular, it has the following features:

- **Local changes**: When a local change occurs in the world only a local update is needed in the abstraction.
- **Explicit representation of terrain types**: Unlike other approaches, terrain types are explicitly represented in the abstraction. When the terrain type changes, the abstraction must be updated, but when the cost of terrain changes it does not.
- **Paths respect terrain costs**: It is possible to use Weighted A* (Pohl 1970) or other recent suboptimal algorithms (Chen and Sturtevant 2019; Chen et al. 2019) to speed up pathfinding. But, our experiments show that Weighted A* generally achieves its speedup by ignoring terrain costs. Thus, significantly higher quality paths can be found by explicitly reasoning about terrain costs in the abstraction.
- **Small memory overhead**: The cost of storing the abstraction is significantly smaller than the cost of the full map.
- **Flexible underlying representation**: We implemented the abstraction approach both using grids and NavMeshes as the low-level representation. The approach works with both, but benefits from the structure of how NavMeshes are built in Recast.
- **Faster pathfinding**: The speedup achieved depends on the type of terrain and the underlying representation, but provides speedup in either cases.

The full approach is called a dynamic terrain abstraction. An example of the abstraction can be found in Figure 1. Planning is first performed in the abstract graph, and then is refined to a walkable path within the world. Additional details can be found in the full paper (Sturtevant et al. 2019).

Future Work

Although we are happy with the current implementation of this work, there is still room for improvement. The current approach, for instance, doesn't directly handle different sized agents; a different NavMesh is used for different agent sizes, although some approaches are able to handle this constraint (Kallmann 2010). The current approach also does not do any pre-computation to speed a single pathfinding request. For highly dynamic maps this could be too expensive, but past work has explored applying such techniques at the abstract level of the graph (Sturtevant and Geisberger 2010), and other work has looked at re-using pre-computed data after the map changes (Bono et al. 2019). Finally, this work does not address the issue of multiple-agents planning through space together. Current search-based approaches to this problem are not feasible for most games (Stern et al. 2019) in comparison to locally reactive approaches (Van Den Berg et al. 2011; Karamouzas, Skinner, and Guy 2014). As the development of the game continues, we will continue to work with them to address challenges that arise.

References

Alain, B. 2018. Hierarchical dynamic pathfinding for large voxel worlds. *Game Developers Conference*.

Bono, M.; Gerevini, A. E.; Harabor, D. D.; and Stuckey, P. J. 2019. Path planning with cpd heuristics. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 1199–1205. AAAI Press.

Botea, A.; Müller, M.; and Schaeffer, J. 2004. Near optimal hierarchical path-finding. *Journal of Game Development* 1(1):7–28.

Bulitko, V.; Sturtevant, N.; Lu, J.; and Yau, T. 2007. Graph abstraction in real-time heuristic search. *Journal of Artificial Intelligence Research (JAIR)* 30:51–100.

Chen, J., and Sturtevant, N. R. 2019. Conditions for avoiding node re-expansions in bounded suboptimal search. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Chen, J.; Sturtevant, N. R.; Doyle, W.; and Ruml, W. 2019. Revisiting suboptimal search. *Symposium on Combinatorial Search (SoCS)* 18–25.

Geisberger, R.; Sanders, P.; Schultes, D.; and Delling, D. 2008. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In McGeoch, C. C., ed., *Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08)*, volume 5038 of *Lecture Notes in Computer Science*, 319–333. Springer.

Harabor, D., and Botea, A. 2008. Hierarchical path planning for multi-size agents in heterogeneous environments. In *2008 IEEE Symposium On Computational Intelligence and Games*, 258–265. IEEE.

Holte, R. C.; Mkadmi, T.; Zimmer, R. M.; and MacDonald, A. J. 1996. Speeding up problem solving by abstraction: A graph oriented approach. *Artificial Intelligence* 85(1-2):321–361.

Kallmann, M. 2010. Shortest paths with arbitrary clearance from navigation meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 159–168. Eurographics Association.

Karamouzas, I.; Skinner, B.; and Guy, S. J. 2014. Universal power law governing pedestrian interactions. *Physical review letters* 113(23):238701.

Pelechano, N., and Fuentes, C. 2016. Hierarchical pathfinding for navigation meshes (hna*). *Computers & Graphics* 59:68–78.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artificial intelligence* 1(3-4):193–204.

Stern, R.; Sturtevant, N. R.; Atzmon, D.; Walker, T.; Li, J.; Cohen, L.; Ma, H.; Kumar, T. K. S.; Felner, A.; and Koenig, S. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. *Symposium on Combinatorial Search (SoCS)* 151–158.

Sturtevant, N., and Buro, M. 2005. Partial pathfinding using map abstraction and refinement. In *National Conference on Artificial Intelligence (AAAI)*, 47–52.

Sturtevant, N., and Geisberger, R. 2010. A comparison of high-level approaches for speeding up pathfinding. 76–82.

Sturtevant, N., and Jansen, R. 2007. An analysis of map-based abstraction and refinement. *Symposium on Abstraction, Reformulation and Approximation (SARA)* 344–358.

Sturtevant, N. R.; Sigurdson, D.; Taylor, B.; and Gibson, T. 2019. Pathfinding and abstraction with dynamic terrain costs. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

Sturtevant, N. R. 2007. Memory-efficient abstractions for pathfinding. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 31–36.

Tozour, P. 2002. Building a near-optimal navigation mesh. In *AI Game Programming Wisdom. (S. Rabin, ed.)*, 171–185.

Van Den Berg, J.; Guy, S. J.; Lin, M.; and Manocha, D. 2011. Reciprocal n-body collision avoidance. In *Robotics research*. Springer. 3–19.