Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-20)

Geometry of Hiding: Generating Visibility Manifolds

Adrian Koretski, Clark Verbrugge
School of Computer Science McGill University
Montréal, Québec, Canada
adrian.koretski@mail.mcgill.ca, clump@cs.mcgill.ca

Abstract

Understanding what a character observes in a game as they move through the space is useful for game analysis and design. In this work we describe a tool that generates *visibility manifolds* given a game terrain and game agent path. Our tool accommodates multiple kinds of game agent motion and considers the impact of visual occlusion in order to efficiently generate a closed 3D mesh representation of the area seen over time. The resulting shape demonstrates unintuitive properties of game agent observations, and an efficient Unity implementation allows the constructed shape to be used in interactive game design.[1]

Keywords: visibility polygon, stealth, geometry

Introduction

In this work, we describe a tool allowing us to generate *visibility manifolds,* meshes based on game agent paths that represent the evolution of an agent's field of view over time. Such meshes are useful for analysis of in-game properties by designers, such as for exploring safe player routes in stealth games and understanding level coverage by guards. These manifolds can also be used to analyse AI perception as they represent general visibility data rather than a single model. The tool generates individual 2D representations of an agent's field of view, or *visibility polygons*, at discrete timesteps using a simplified angular sweep algorithm (Asano 1985). It then uses properties of the polygon vertices to establish connections between time-ordered, consecutive polygons to generate a final 3D mesh. The application itself is implemented in the Unity game engine which allows easy integration with other game environments. To generate manifolds, obstacle and guard path game-objects are tagged as "obstacles" and "paths" respectively. A master game-object is then added to the scene where it performs all the calculations for the user.

[1]Demonstration materials can be found here: https://github.com/AdrianKoretski/AIIDE-2020-demo

Related Work

Our design is influenced by efforts in the medical community to create a solution when piecing together MRI and CT scan slices into 3D representations. These solutions are often geared towards generating smooth 3D representations by satisfying constraints and minimising functions (Zou et al. 2015). A large number of the solutions also focus on connecting branching shaped, non parallel slices and use limited or fuzzy data (Bermano, Vaxman, and Gotsman 2011; Boissonnat and Memari 2007). Although interesting, these solutions are less applicable to game contexts, as we have full and precise slice data, and require polygonal (non-smooth) surfaces for easy computation of intersection. Our approach shares some similarities with work by Barequet and Vaxman (2007). Their approach consists of extracting vertex properties from the slices and using them to establish connections. We differ in that we use meta properties based on the slice generation whereas they use geometric properties.Prior work in game contexts has also been done, based on stacking individual visibility polygons (Tremblay 2016). Lack of good connectivity between layers, however, induces discretization artefacts, requiring expensive, fine-granularity in timesteps to improve precision.

Methods

In order to generate visibility manifolds, individual visibility polygons are first created and connected to one another. To generate the individual polygons, discrete steps and data in the game agent path are used. The polygons are then augmented with characteristics applied to their vertices referred to as nodes, and are then used to connect the visibility polygons to one another. Finally, the connections are triangulated to create the final mesh.

Game Agent Path: The game agents path is broken down into discrete steps, referred to as *pips*. Each pip is comprised of a position and orientation which are used to generate the visibility polygon, in addition to a timestamp, used to place the polygon at the correct manifold height. The distance between pips defines the granularity of the manifold in time.

Visibility Polygon: The construction of the visibility polygons is based on the sweep algorithm, modified to re-

strict the maximum range and angle of view. Once constructed, a node is placed at each vertex of the polygon.

Nodes: Nodes connect to one another to form the polygon and carry extra data with them. This data corresponds to unique identifiers and types determining the characteristics associated to the original vertex they derive from. Node characteristics may correspond to obstacle vertices, points of occlusion or result from the boundaries of the visibility polygon. Two nodes are deemed similar if they share a type and identifier but may differ in time and position.

Polygon Similarity: Similarity is used to determine whether two polygons can be trivially connected. Two polygons are deemed to be similar if both have the similar nodes in the same order around the circumference of their respective polygons. Non similar polygons occur due to changes in field view brought on by occlusion, and can cause artifacts when connected over large distances. In this event, the point where the change occurs is found by recursive bisection over time. Once found, additional polygons are generated on either side, separated by a small threshold set by the user, referred to as the non similarity threshold.

Connecting Polygons: Each pair of consecutive polygons are connected to one another via their nodes. Similar nodes are connected to one another. Remaining unconnected nodes are then connected based on proximity. These connections are formed with the overall structure in mind to ensure no connections cross one another.

Triangulation: To finish forming the mesh, additional connections are formed between each layer pair to triangulate and produce the final mesh. Given that each node connects to at least one node in the subsequent layer, inter-layer shapes consist of either triangles or quadrilaterals, which are easily split into triangles.

Results

The resulting mesh quality depends on two parameters: the granularity of the step sizes in the path and the size of the non similarity threshold. We observe how changing these parameters affects the output and how accurate said output is to an ideal manifold.

Step Size: For step size analysis, we use simplified examples where the game agent performs a simple rotation with no obstacles. Manifolds generated from more complex settings can appear odd and unintuitive, thus making them difficult to briefly analyse and explain. We see that for very coarse step size, the triangulation results in a poor approximation leading to large areas of missing geometry. Scaling down the step size causes the mesh to become smoother and limits the size of the missing geometry. These results, along with an ideal manifold, can be seen in Figure 1.

Non Similarity Threshold: For the non similarity threshold, we start with a very large threshold and reduce it while observing how the shape of the manifold changes. The experiment consists of the same motion as previously with an additional obstacle. Large values, illustrated in the left image in Figure 2, lead to inaccuracies where some parts are incorrectly perceived as non visible (above triangle 4 and 5) and others are incorrectly perceived as visible (triangle 1). By reducing the threshold size, we observe how these inaccuracies shrink to acceptable ranges. Taking this to the extreme (threshold of 1 millisecond) we see that we get a nearly perfect manifold.

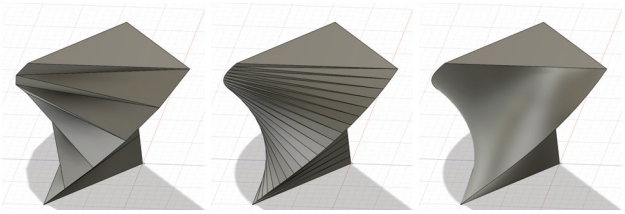

Figure 1: Illustration of step sizes. On the left, 2 sec / 18 degree step size versus center, 0.5 sec / 4.5 degree. On the right we have an ideal manifold.

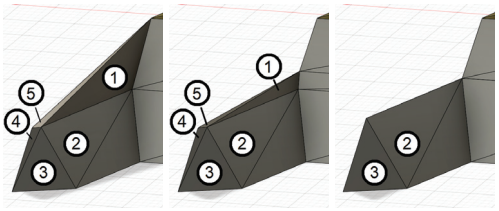

Figure 2: Illustration of connecting non similar polygons (top and bottom layers) using different thresholds. On the left, a large threshold generates an additional polygon, while the middle generates two closely positioned polygons straddling the point of change. On the right we have an ideal connection.

Conclusions & Future Work

Our tool is capable of real-time generation of visibility manifold meshes that approximate ideal visibility manifolds. Using small values for the non similarity threshold results in near ideal manifold construction. A drawback is the potential for excessively small or thin triangles, although these can also be removed through post-processing mesh simplification. Step sizes can, one the other hand, be adapted to individual areas of the terrain. Our tool generates accurate mesh data with sufficiently small step sizes even when visibility changes rapidly. Areas with slow changing visibility can use larger step sizes as this will minimally impact accuracy. Resulting inaccuracies are caused by naive triangulations of non-planar quads. This can be reduced in future work by using more detailed triangulations model.

References

Asano, T. 1985. Efficient algorithms for finding the visibility polygons for a polygonal region with holes. *Transactions of IECE of Japan* E-68:557–559.

Barequet, G., and Vaxman, A. 2007. Nonlinear interpola-

tion between slices. *ACM Symposium on Solid and Physical Modeling 2007* 97–107.

Bermano, A.; Vaxman, A.; and Gotsman, C. 2011. Online reconstruction of 3d objects from arbitrary cross-sections. *ACM Trans. Graph.* 30(5).

Boissonnat, J.-D., and Memari, P. 2007. Shape reconstruction from unorganized cross-sections. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, 89–98. Goslar, DEU: Eurographics Association.

Tremblay, J. 2016. *Computing Techniques for Game Design*. Ph.D. Dissertation, McGill University.

Zou, M.; Holloway, M.; Carr, N.; and Ju, T. 2015. Topology-constrained surface reconstruction from cross-sections. *ACM Trans. Graph.* 34(4).