A Unified Spatial Representation for Navigation Systems

Michael Ramsey
Blue Fang Games, LLC
1601 Trapelo Road Suite 105, Waltham MA, 02451-7333
miker@masterempire.com

Abstract

The purpose of this paper is to outline the core components of a practical navigation system which uses a novel technique for spatial representation in a commercial entertainment product. This paper is based upon the navigation system developed for *The World of Zoo* (*WOZ*) by Blue Fang Games, LLC and published by THQ. WOZ placed the following requirement on our in game agents (which are animals, such as tigers and penguins): depending on the animals species they were required to locomote across land, water, exhibit the ability to climb and eventually to fly – all in a seamless manner. Animal locomotion in WOZ is driven by accumulating the root motion of multiple blended animations; this required a unique approach to the spatial representation of our environments. The system needed not only to take into account the defacto static environments that were created by the level designers, but also the dynamic structures that the animals use (depending on the players interactions at that particular moment). There was also the extra challenge of developing a system that was as straightforward as possible for level designers to work within. As Anthony J.D' Angelo so succinctly stated, "Don't reinvent the wheel. Just realign it." It is with this sage advice in mind that we reevaluated traditional navigable representations, in conjunction with how our animals *should* move through their environments. As important as the navigation framework was to the development of WOZ, the way the thought processes developed preceding the implementation is also of interest; as the re-understanding of what navigation is composed of (in virtually any environment) guided our decisions through the design and implementation stages.

Introduction

WOZ is an entertainment product for both the Wii and PC platforms. The fundamental experience for the player is to interact with a variety of different animals in a series physically credible exhibit. Players interact with the animals by not only providing the basic requirements – food and rest, but also by providing enrichment objects. A typical selection of objects that can enrich an animals experience are: balls, chewable stuffed animals and food dispensers. The player can also build structures in an animals exhibit, which not only provides interesting eye candy for the player, but it also provides points of interaction (e.g. flee, drink, graze, hide etc.) that depending on an animals mood and personality can be used.

WOZ provided us with the fundamental challenge of an environment that could be altered by the player at runtime in multiple spaces. WOZ has three distinct spaces – land, water and air (Reichenbach 1958). The shipping version of WOZ has support for land and water, whereas air will be implemented in the future (support for the air-space is really just further parameterization of the fundamental spatial construct that WOZ uses – which will be discussed below). This paper will outline the core components of the navigation system – particularly how the world is spatially created and utilized by the animals in WOZ.

Core Components and the Approach

Every navigation system has an atomic element that it's pathfinding system uses; for the majority of commercial games it is typically a system based on either a way point or a navigation mesh. Each approach has its benefits and drawbacks (Waveren 2001). A similar system to the one detailed here is the circle-based waypoint graph (Tozour 2003). WOZ's approach to navigation was unique in that we did not place an emphasis on the traditional 3-space point (Bergen 2004) in the world, as the atomic navigable element - instead we focused on the *representation of space* in the world. This fundamental realignment allowed us to pursue for an element that is in itself simple, but allowed us to construct a system for locomotion that is unified across the previously mentioned game spaces. While WOZ is typical in that it has it's fair share of goto position (x,y,z) and execute a specific action – such as eat; it is also atypical in that the motion and approach of the animal, is just as important as it's eventual end of route interactions. This fundamental understanding directed many of our decisions.

Our basic primitive for spatial representation is a sphere (in development we typically referred to it as a *navsphere)*.

Figure 1 shows a typical navsphere layout for one WOZ's exhibits. The navsphere is considered the atomic component, because when combined with other navspheres, we are able to form a navigable representation (a *navrep*) of our world. The navrep represents the interactable spatial extents of the environment at any given moment. The navrep not only informs the game side systems of where the animals *can go* in 3-space, but also, and perhaps more importantly, where they *cannot go*. The later being important for critical aspects of locomotion such as turning and inter-animal interactions.

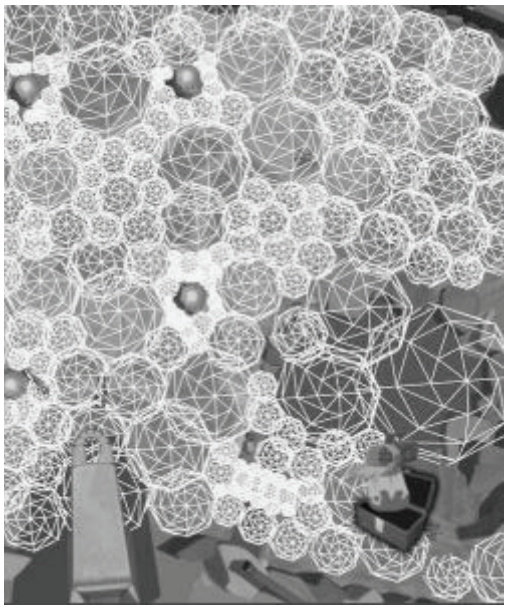

Figure 1: A subset of a Zebra's exhibit from WOZ showing a representative navsphere layout.

Since a navsphere represents spatial information about the environment, it thus inherently allows us to define spatial information relative to that space in the world. We attribute spatial biases to a navsphere (internally to make the concept more palatable to level designers we use the term *pathing bias*). Typical spatial biases attributed to a shape were animal size, navsphere type – such as, water, land or air shape, as well as other game related attributes.

Navspheres are placed during the level development process (which is done through 3DS-Max). With only a few guiding principles such as not placing navspheres in solid geometry or not placing navspheres within each other - the process is relatively straight foward when working in conjunction with the WOZs built in visualization tools (the figures in this paper show some of the visualization tools developed).

For the navigation system to route efficiently over the spatial representation of the world, there is the need to build a basic connectivity graph for the environment. Figure 2 contains a visual representation of the connectivity graph that is generated from Figure 1's navsphere layout. An initial algorithm for the construction of a rudimentary connectivity graph is detailed in Algorithm 1.

```
Loop over every navsphere in the scene (A)
{
 A = currentOuterLoop navsphere
 Loop over every navsphere in the scene (B)
 {
    B = currentInnerLoopNavSphere
    if A is not the same shape as B
    {
     if Shapes A and B overlap
     {
       Establish a bidirectional pathway between navsphere
       A and navsphere B and store link in shape
     }
    }
 }
}
```

Algorithm 1: A straight forward algorithm to generate navsphere connectivity.

The connectivity graph algorithm looks for navsphere overlap. When a navsphere overlaps with any other navsphere a bidirectional link is established; thereby allowing movement out of and into the respective navspheres. While the above algorithm is a good way to get started, it is generally not leverage able in large scenes with a large number of dynamic objects. A straight forward to implement world partitioning system is required to minimize the cost of generating the connectivity graph. By implementing a loose quad-tree (Samet 2006) partitioning system, connectivity generation is significantly reduced and could be used in a shipping product.

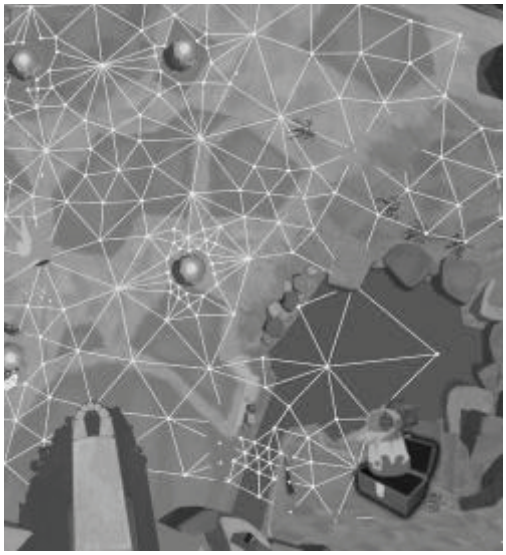

Figure 2: A generated connectivity graph from the navsphere layout in Figure 1.

Routing

Routing over the generated connectivity graph is done using the A* algorithm (Stout 2000). However, it is only

used in it's most direct form – to generate a *possible route* from point A to point Z; not an *exact* path. WOZ has a multitude of other systems in place to generate an animals appropriate motion. We use the generated route to store only the *desired* sense of progression through an environment towards the goal location; as the actual motion of the animal may cause it to deviate (which it routinely does) from the exact route generated by A*. This is very similar to how motion occurs in the real world – animals and humans don't follow an exact course, but they deviate according to not only their understanding of the world, but also by the physicality of available space (Lefebvre 1997). Some of the techniques that are in place to facilitate this locomotion are: route collapsing for direct paths, specific routes that are generated based upon an animals facing and destination, and terrain biases.

Handling a Dynamic World

One of the exciting aspects of WOZ is the ability to alter the environment in a nondeterministic manner. This is achieved by constructing interactable structures for the animals and by allowing the player to alter the basic environment at runtime. WOZ is a real time game, working in a constrained environment - so techniques needed to be implemented that are not only memory and CPU efficient but also allowed for the non-deterministic alteration of the environment. The concept that came about was the *cull shape.* The cull shape has initially been implemented as an extension of the navsphere. Figure 3 illustrates the use of cull shape in a penguin environment. Cull shapes are used to cut-out aspects of the environment that a dynamic object is intersecting with and then replace with it any potential new navspheres. This cut and replace process worked out well as it allowed our structures to be buildable at multiple levels, e.g. we can cycle through the construction of a series of ice floes on a lake in real-time without any significant computation.

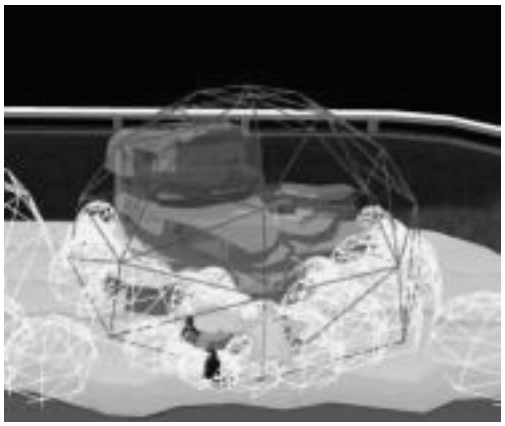

Figure 3: An example asset for WOZ that shows how an attached cullshape (red sphere) is used to invalidate surrounding navspheres for navigation.

The Core Navsphere API

The core API that is required is remarkably straightforward (the support code for game-side interaction was more lengthy then what was required for implementation of the core navigation system). Since the entire world is spatially represented as spheres we required only the following functions:

- A function to determine if a sphere overlaps with another sphere.
- A function to calculate separation between two spheres.
- A function to determine if a three dimensional point is within a sphere.

Some of the utility functions that we found useful (which we specifically added for route smoothing) were the ability to calculate the exact points of overlap between two spheres. This allowed us to bias the animals motion over multiple points along the route without using the origins of the navspheres. If the origins of the navspheres were exclusively used as targets, the animals could move in unnatural patterns. Also the ability to use tangential information generated from overlapping navspheres was initially useful in biasing motion away from objects in the environment.

Conclusions and Future Work

WOZ has a solid foundation of navigable primitives to build upon, but there are definitely aspects that could be investigated further.

Interaction with objects in the world are not spatially consistent. In the current implementation, when an animal is the same navsphere as an object, the majority of the interactions are driven by the standard "goto position(x,y,z)"; and while it is efficient, it does not lend itself to maintaining a harmonious interaction with an object when other agents are involved. A more prudent approach is to define objects with a simple spatial primitive (a recent test using a simple encapsulation sphere yielded positive results) or perhaps if need-be, convex hulls can be used to remove parts of the underlying navrep. A simple area test could be used to essentially guide animals to a spatial defined region within a navsphere. This would also allow for the management of interactions with an object in a potentially more natural and controlled manner.

Another aspect that could improved is that entities should have their internal connectivity graphs precomputed at tool-time. This would have saved us a few ms of CPU cost.

Some resources have been spent in investigating methods of automating the construction of the navrep, however, numerous challenges surface because of the dynamic nature of WOZ's environments. (Waveren 2007) discusses a method to automate the construction of an area based system, however, it is predicated on a rigidly constructed environment that has been partitioned already

via the binary space partitioning (BSP) algorithm (Ericson 2005). Some of the WOZ specific challenges are related to stitching the dynamic objects into the scene, and the level designer specified build hierarchies.

One of the main goals for our navigation system was to have it developed around a spatially accurate representation of our games environments, and by using the components outlined in this paper, we have helped achieve that goal in WOZ.

Acknowledgments

I wish to acknowledge the following individuals (in alphabetical order) for not only their contributive efforts to development of the WOZ AI system (as this paper was just a slice of that system in it's entirety) but also as fellow AI developers: Bruce Blumberg, Steve Gargolinski, Ralph Hebb, and Natalia Murray.

References

Bergen, Gino Van Den. 2004. *Collision Detection in Interactive 3D Environments.* San Francisco, CA: Morgan Kaufmann.

Ericson, Christer. 2005. *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology).* San Francisco, CA.*:* Morgan Kaufmann

Gibson, James J. 1986. *The Ecological Approach to Visual Perception.* Hillsadale, New Jersey: LEA.

Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics SSC4 4 (2)*: 100-107.

Lefebvre, Henri. 1997. *The Production of Space*. Malden, Mass.: BlackWell Publishing.

Reichenbach, Hans. 1958. *The Philosophy of Space and Time.* Mineola, NY.: Dover

Samet, Hanan. 2006. *Foundations of Multidimensional and Metric Data Structures.* San Francisco, CA: Morgan Kaufmann.

Stout, Bryan. 2000. The Basics of A* for Path Planning. *Game Programming Gems*:254-263.

Tozour, P. 2003. Search Space Representations, *AI Game Programming Wisdom 2*: 94.

Waveren, Jean Paul Van. 2001. The Quake 3 Arena Bot, Master's thesis. University of Technology Delft.

Week, Jeffrey. 2001. *The Shaping of Space.* New York, NY.: Marcel Dekker, Inc.