Zerg Rush Hour: Simulating Swarms for *StarCraft 2* Cinematics

Matt Cordner* Bill La Barge†
Blizzard Entertainment

1 Introduction

In *Starcraft 2: Heart of the Swarm*'s introductory cinematic, tens of thousands of alien creatures known as the Zerg descend upon and infest a human city. To animate them all, a procedural particle-based crowd system was created to simulate their movement and behavior. Artist-friendly controls direct the swarm to move, attack, climb, and leap using volume primitives in a lightweight implementation, providing fast turnaround for modifying and executing new iterations.

2 Fast Simulations Using Particles

The Zerglings were required to move through the city in an insect-like swarm, crawling over each other and around obstacles in their path. They needed to avoid certain areas, navigate around corners, attack targets, and react to explosions. Fast simulations and intuitive controls were also desired.

We began with a simple Houdini particle setup and created a single VEX node to implement most of the swarming behavior. The VEX code relied on point cloud methods, volumes, and a *zstate* variable to determine each individual Zerg creature's movement capabilities and desired actions. Some examples are whether the creature was climbing, scrambling over other Zerglings, attacking, being crushed, blocked, overcrowded, or had enough support to propel forward, and in which direction.

Once a target velocity vector was determined, a simple algorithm adapted from Kück's bubbles simulation was applied. In this method, the total force acting on the bubble is a summation of repulsive and attractive forces, friction, and gravity.

$$\mathbf{V}_i = \frac{1}{k_v + k_{of} + k_{air}} \left(k_v \overline{\mathbf{V}_i} + k_{of} \overline{\mathbf{V}_i^o} + \mathbf{F}_i^{ra} + \mathbf{F}^g\right) \qquad (1)$$

The properties of bubbles fit the swarm profile excellently: the Zerg moved fluidly, staying close to one another without much interpenetration, and could climb over obstacles and each other rather easily. The simulation would query the state of the previous frame by treating it as a point cloud to find each Zerg's nearest neighbors, their *zstates* and velocities. The target velocity was worked into the above equation by applying it as a frictional force, to simulate the Zergling propelling itself forward by pushing off of whatever it is currently supported by (e.g., ground, obstacles, other Zerglings).

3 Using Volumes for Crowd Simulations

Volumes are ideally suited for particle simulations, as they have a low overhead for accessing data such as velocity vectors and signed distance fields. In our case of animating the swarm, volumes were used for the same reasons. Instead of defining a list of objects that the swarm would interact with and iterating over those elements to find their relative locations and distances to each individual creature, we employed volumes to represent them all.

For the general direction of movement, we created a velocity field that defines the path of the swarm through streets and around buildings. For objects that the Zerglings could climb over, a signed distance field was sufficient for level ground, ramps, rubble, tanks, and other larger creatures such as the Nydus Worm that breaches the plaza. A fast sample of these volumes would return the distance to the object, gradient and the normal of that element. With that information, we could determine what the Zergling is encountering and whether or not it should climb over, go around or attack it. "Avoidance volumes" were created to have the smaller Zerglings avoid being stomped on by the much larger Ultralisk creatures. Another set of volumes were turned on and off in concert with many of the explosions, which scatter the creatures nearest to them in all directions.

Figure 1: The Zerg swarm fills a city street.

When we reached the final shot, we noticed that Zerglings were running through the ranks of enemy marine formations. We again used volumes to define "kill zones," to identify where Zerglings were coming under fire from the marines' rifles and the intensity of that fire. Each Zergling inside the kill zone would check a random function see if it should be killed and play a death animation.

4 Rendering The Swarm

Each particle from the simulation represented a single creature. For visualization in Houdini, we instanced cached animations of a running Zerg creature onto each particle. To render final frames, the particle data was written to RIB files and then parsed to insert the appropriate geometry procedural calls and shaders. Loading geometry at render time through the use of a procedural kept our RIB data small, and provided flexibility to utilize different LOD models.

5 Conclusion

For animating a swarm of insect-like Zerg, we were able to create a very fast, large-scale simulation system that demonstrated complex behavior but was still easy to manipulate and augment by the artist. Our use of volumes was instrumental in gathering the data necessary to determine the creatures' actions and method of interaction with the scene.

References

H. KÜCK, C. VOGELGSANG., AND G. GREINER, 2002. Simulation and Rendering of Liquid Foams. Graphics Interface (GI).

*mcordner@blizzard.com
†blabarge@blizzard.com

SIGGRAPH 2013, July 21 – 25, 2013, Anaheim, California.

ACM 978-1-4503-2261-4/13/07