Methods of Creating Destructible Assets for Video Games

Ryan M. Hoss, *Vicious Cycle Software, Inc.* and Todd B. Emma, *East Tennessee State University*

***Abstract*— Many of today's big budget, "triple-A" video game titles take cues from the special effects and storytelling styles of Hollywood blockbusters. However, the major factor that sets the video game industry apart is the interactive nature of its products. While augmented reality, motion control, and stereoscopic 3D are changing the way the player interfaces with a game, other advances are being made with the content of the games themselves. Developers using lighting systems with ambient occlusion, lifelike terrain with foliage, and high-level shading networks are able to bring an unparalleled amount of realism to their products. In addition to these advancements, destructible assets are also being used in many different ways across multiple platforms and genres. Here the authors look at several ways these ideas are used currently and go into detail on a method of implementation.**

***Index Terms*—CG FX, Games, Kismet, Particles, Real-Time**

I. Introduction

VIDEO games are one of the leading forms of entertainment in the world continue to emulate their closest neighbor, the film industry. Many of today's big budget, "triple-A" video game titles take cues from the special effects and storytelling styles of Hollywood blockbusters. However, the major factor that sets the video game industry apart is the interactive nature of its products. While augmented reality, motion control, and stereoscopic 3D are changing the way the player interfaces with a game, other advances are being made with the content of the games themselves. Developers using lighting systems with ambient occlusion, lifelike terrain and foliage, and high-level shading networks are able to bring an unparalleled amount of realism to their products. In addition to these advancements, destructible assets are also being used in many different ways across multiple platforms and genres.

II. Methods

The methods used for creating the destructible objects for these two games were very different because of the way in which the objects were utilized in gameplay. The video game *Earth Defense Force: Insect* Armageddon made use of 3DS Max macroscripts and baked animations to produce relatively inexpensive buildings. After the macroscripts broke up a modeled building into pieces, the baked animations were created by keyframe animating the building pieces and skinning them into a single object that could be exported into the game engine. These buildings could be placed anywhere in the game world and be destroyed at any time. Developers of the *Red Faction* video game series created a proprietary toolset called Geo-Mod that allowed the designers to control how the destructible objects affected the game world.

DOI: 10.5176/2010-3043_2.3.182

III. Content Creation

One of the most common and most scalable ways of creating a destructible object is to create a "pristine" mesh (the static, undamaged version of the model) and then swap it out with a destruction mesh. A destruction mesh is the fractured mesh with an ambition that is either animation or a dynamic simulation. The following example details the creation of a destructible version of one of the most universal video game assets, the wooden crate.

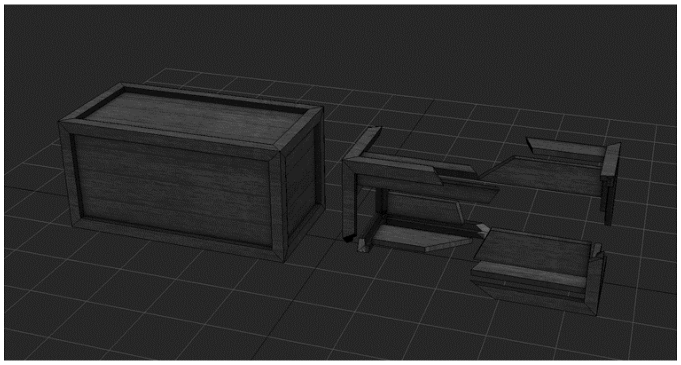


Fig. 1. The crate's pristine mesh (left) and its destruction mesh (right).

Beginning in 3DS Max, the pristine mesh was modeled and textured. That mesh was then used as a base to create the destruction mesh. The existing textures and geometry were transformed into a framework of broken wooden pieces that closely matched the original model. While retaining accuracy in the model is important, it is also a good idea to create interesting fractures with unique shapes and sizes for use when creating the dynamic simulation.

IV. Physical Forces

Once the destruction mesh was created, the Reactor plug-in was used to simulate the wooden pieces as if the crate had been smashed by an attack or ability. The Reactor plug-in is a 3DS Max-specific plug-in that simulates physical forces on a mesh and creates keyframed animations based on the simulation. How the object will be used in the game is one of the most important aspects to take into consideration when creating baked animations. Because the animation will behave the same way every time, it needs to be visually interesting and make sense with the gameplay. When a player destroys this crate, the resulting destruction animation should not only look believable, but feel satisfying for the player to interact with. The animation should also read well up close and at a distance to ensure that the player can tell what is happening depending on how they are playing the game.

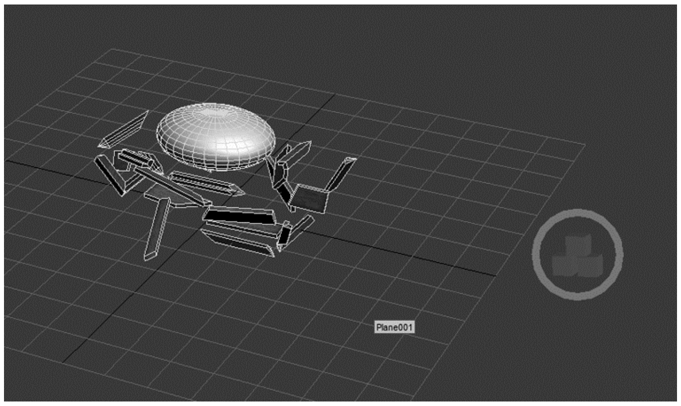


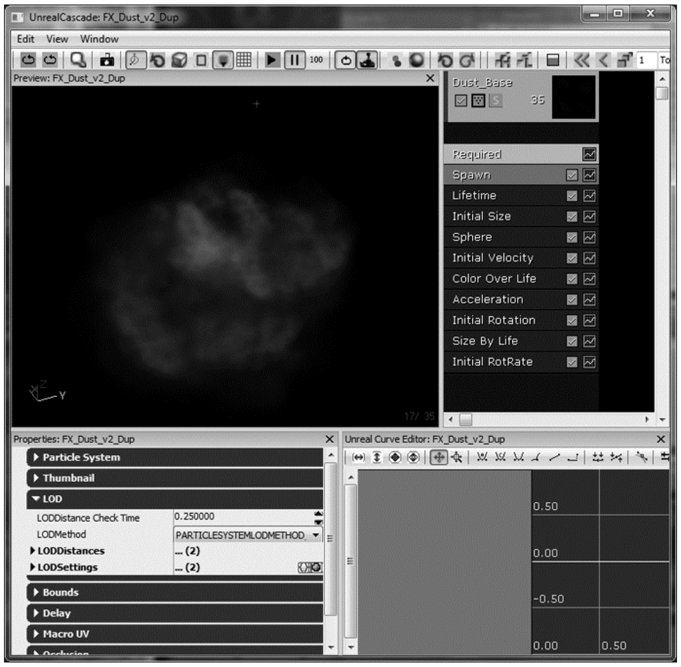
Fig. 2. The pieces of the destruction mesh are simulated with the Reactor plug-in and baked into keyframed animations.

V. Baking

After the animations from the Reactor simulation have been baked, they can be exported into a game engine. In this example, the Unreal Development Kit was used to develop a scene that tested the functionality of the destructible asset. In addition to the animations, the pristine mesh and destruction mesh have been imported into the engine.

VI. Bringing it into the Game Engine

A dust particle system was also created in Unreal Cascade to enhance the final sequence. Cascade is the particle editor that comes with the Unreal video game engine. Whereas the baked animation will be the same every time, particle emitters are an easy way to add variation to the overall destruction effect. The texture used in the effect is a set of four slightly different images of smoke that the particle system chooses at random. Modules in this system then control and randomize properties such as the lifetime, size, shape, color, acceleration, and rotation of each particle. A distortion material is also used on each particle, further randomizing the effect.



LODDistance Check Time 0.250000
LODMethod PARTICLESYSTEMLODMETHOD_
LODDistances ... (2)
LODSettings ... (2)

Fig. 3. Screenshot of the dust particle system in Unreal Cascade.

VII. Scripting

The Kismet scripting system is used to put these pieces together into a single destructible asset. Kismet is the visually-based scripting language built into the Unreal video game engine. Kismet is used here to create an in-game sequence allowing the player to destroy the asset. Once the pristine mesh (placed in the scene as an Interpactor) takes a certain amount of damage from a game weapon, a series of events are triggered: the pristine mesh is destroyed, the destruction mesh is spawned and its animation is played (using Unreal Matinee), the dust particle system is spawned, and a “break” sound effect is triggered.

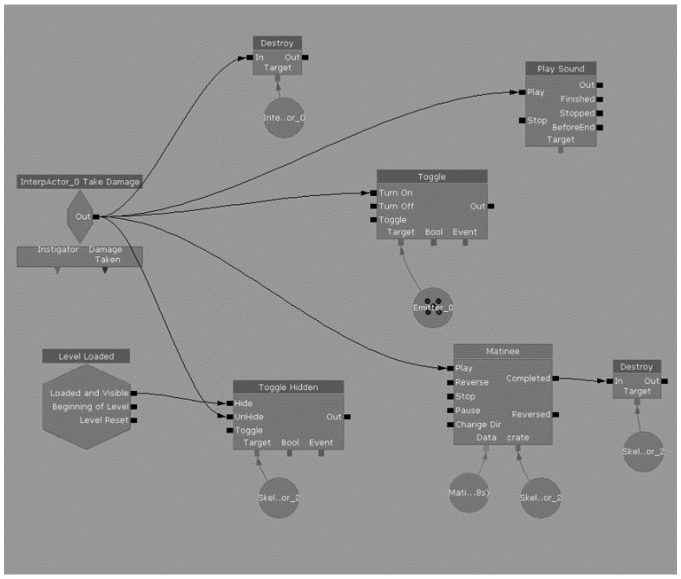


Fig. 4. The Unreal Kismet sequence that controls the functionality of the destructible crate.

VIII. Summary

This method of creating a destructible asset is inexpensive in comparison to other methods such as dynamically based custom coded asset destruction. The destruction mesh in this example uses baked animations instead of in-engine physics plug-ins. The pieces that compose the mesh have also been skinned into a single animated object connected to a root bone created in the 3D modeling and animation package. Individually keyframed pieces use more draw calls on the GPU, whereas the skinned animated object created in this example uses less draw calls and is therefore optimized for use in a game environment.

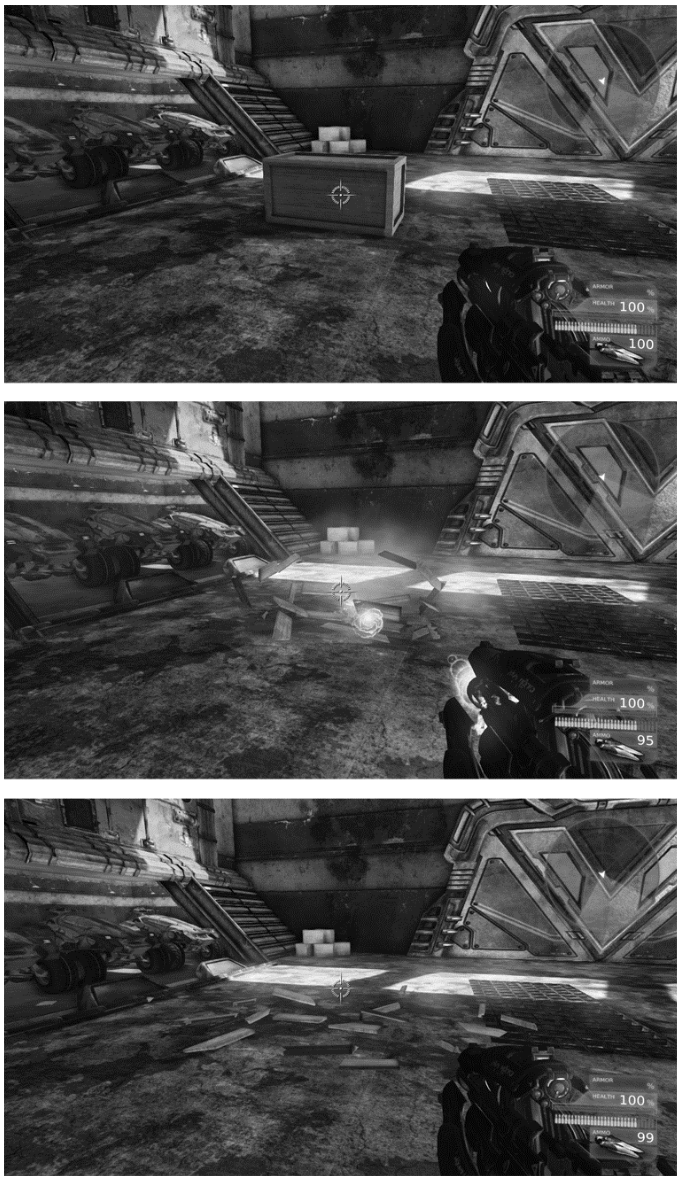


Fig. 5. The final destructible crate, set up in the Unreal Development Kit.

References

[1] Caron, F. (2009). Gaming expected to be a $68 billion business by 2012. Retrieved from: http://arstechnica.com/gaming/news/2008/06/gaming-expected-to-be-a-68-billion-business-by-2012.ars

[2] Geo-mod/physics f.a.q.. (2000). Retrieved from http://web.archive.org/web/20100330132736/http://redfaction.volitionwatch.com/faq/geomodfaq.shtml

[3] Havok destruction. (2011). Retrieved from http://www.havok.com/sites/default/files/pdf/Havok_Destruction_2011.pdf

[4] McClelland, M. C. (2011, October 17). Interview by R.M. Hoss [Personal Interview]. Methods of destruction in earth defense force: insect armageddon.

[5] Vfx optimization: Getting results - draw calls - render thread. (2010). Retrieved from http://udn.epicgames.com/Three/VFXOptimizationResults.html

Ryan M. Hoss was born in Greeneville, Tennessee in 1987. After discovering his passion for creating video games in high school, he attended East Tennessee State University just as their Digital Media program began forming a gaming concentration. He acquired his B.S. in Digital Media with a concentration in digital interaction from ETSU in 2010. He is currently pursuing a master's degree there.

He currently works as a Special Effects Artist at Vicious Cycle Software in Morrisville, North Carolina. Since he joined the company in 2011, he has contributed to multiplatform video games such as *Ben 10: Galactic Racing*, *Madagascar 3: The Video Game*, and *Ben 10: Omniverse*. He is currently working on an unannounced project.

Mr. Hoss is a member of the IGDA, ECA, and ACM SIGGRAPH organizations. In 2012 he was a roundtable presenter at the East Coast Game Conference, where he hosted a question and answer session about the role of special effects in the game industry. He also continues to assist ETSU's gaming program by providing portfolio critiques to students and lectures on industry topics and techniques.

Todd Emma, Assistant Professor Gaming, 2D, Interactive. Todd spent several years in Fort Lauderdale, FL where he operated his design studio "Voodoo Designs" and taught at the Art Institute of Fort Lauderdale. He later became the Director of Marketing and Web Design at Steeda Auto Sports while continuing to teach part time. Missing the world of gaming and 3D, Todd went back to teaching full time at the Art Institute in the Game Design department as the Game Engine Technology Expert. Working with the leading 3D game engine and 3D modeling software, he successfully led graduating classes into successful gaming positions across the country. After a short stay at South East Missouri State University as a multi-media instructor, he is now working with the faculty at ETSU in Game Design.