19

Real-Time Character Dismemberment

Aurelio Reis
id Software

Modern games utilize a number of tricks to convey a realistic and intriguing world. In only a few years, realistic physics and destructible environments have garnered widespread industry adoption, yet despite this, few games attempt to model heavy damage on game characters. One of the main reasons for this is the complexity of decomposing the topology of a 3D mesh dynamically with real-time performance.

In this gem, we introduce an efficient general-use technique for character dismemberment that can be easily incorporated into an existing skeletal animation system. This implementation is perfectly suitable for games, real-time applications like military simulations, and other "serious games" where the accuracy of the damage modeling does not need to be precise (as in medical simulations).

While many games don't have the subject matter appropriate for dismemberment, when presented in a cartoonish way it's possible to approach gore and dismemberment such that it is relevant to the gameplay experience as opposed to being used purely for shock value or gimmick. Movies such as *Evil Dead 2* and *Kill Bill* have approached gore in a comedic way that rivets audiences. As zombie and monster games gain popularity, it's likely that they will have the most to gain from character dismemberment.

19.1 What is Character Damage Modeling?

A character in this context is roughly defined as any animated figure in the shape of a creature using a bipedal skeleton and keyframed animations for its motion. In the game world, a character is represented as an entity that is capable of moving around using some scripted or autonomous behaviors with animations that coincide with this motion. As the character moves throughout the world, it may come in contact with a number of interactive elements that can affect the character in one way or another. In a first-person shooter, for instance, a character may move about, be influenced in some way by the environment (like when riding an elevator), and may shoot or be shot by other characters (player or non-player).

When a character is killed, the results are usually presented using some kind of pre-canned animation sequence with special effects like blood and gore strewn about. In most modern games, it is common practice to use "ragdoll" physics once the character is completely dead to add an additional sense of realism. In addition, some games go so far as to create bloody chunks and body parts, so-called "gibs" (for giblets). Combined, these elements result in character damage modeling that makes the game world more believable to the player. Games, after all, are about consequences, and realistic character deaths add to that realism.

While most games are able to get away with this degree of damage modeling, some games require more precise detail. Additional elements like projected decals and character scarring adds a lot, but the most dramatic change would be in the shape of the character itself. The effects of severe blunt force or impact trauma on a humanoid is enough to dismember most major limbs with ease, and it is this particular subset of damage modeling on which this gem focuses.

19.2 Methods of Mutilation

There are a number of ways to approach character dismemberment. The easiest and most straightforward solution would be to explicitly model the character as an articulated collection of body parts. Whenever a limb's bounding volume is hit, everything below that body part in the hierarchy can be manually detached. This technique, while incredibly simple, can be difficult to tune because model seams become prominent at the body part boundaries. Aesthetically, the quality is quite poor, although it can be hidden by clever modeling tricks (e.g., placing a gorget at the character's neck seam). In addition, this technique would require a large number of body parts to work well, which means additional draw calls that can result in reduced performance.

The extreme opposite of this would be to use computational geometry. Constructive solid geometry (CSG) boolean operations can be used to remove chunks of a character by dynamically splitting and removing polygons from the mesh. The uniform character mesh is a great advantage, but unfortunately this technique requires heavy CPU processing both in pre-transforming the animated mesh to the proper pose and in the required geometric operations. This technique can also result in a highly triangulated mesh with an unpredictable number of polygons which can become both a memory and performance concern.

Another possible method would be similar to the one used in the *Soldier of Fortune* series of games made by Raven Software. In their approach, they used what they called "gore zones" to represent up to 26 areas that can be toggled off on a given animated biped. The character models were created in such a way that every gore zone was completely capped, sealed and internally textured. This allowed them the flexibility to represent internal cavities such as the skull where the brain could become detached due to heavy trauma. The obvious downsides are a