# Isopod Damage System: Technical Design Document

## Overview

This document describes the damage system to be updated for the next Isopod Labs game.

## Goals

Our existing scheme is capable of the following:

* Delays using Timer entities.
* Spawning of various effects, such as sounds, particle emitters.
* Spawning of pre-authored pieces of debris.
* Ability to leave behind a damaged version of the original model.
* Multiple stages of damage, including recursive ones which regenerate the original object.

We wish to add the following features:

* Faster traversal of entities using function pointers (rather than static linked list)
* Damage-triggered animations
* Allow the damage scripting to be defined using the IsoEditor – in a similar fashion to the animation hierarchies.
* Ability for constraints from the physics system to break down under set stresses. In fact, add a new constraint that combines 3-axis position with rotational constraints, and is designed to progressively reduce the number of rotational axes, until the pieces are hanging on ‘by a thread’.
* Continuous damage for, e.g. items on fire
* Shader-based progressive damage states.
* Advanced fire and smoke effects.
* Procedural mesh generation for simple shapes that break into smaller pieces.
* Mesh hierarchies that split complex meshes into multiple smaller meshes on damage.
* Z-buffer tricks for displaying holes in complex meshes.

## Pipeline

Damage setups can be authored in 3DS Max using the IsoEntity modifier. The base model applies a DamageableObject entity, and specifies the hitpoints. The damage hierarchy is specified by attaching a child node named ‘damage’. Entities can be placed on the damage node, or children of the damage node, to specify additional models to be created, or effects spawned.

These entities can also be specified using the IsoDummy node type, which makes entities such as timers and particle systems visible as representative models.

In addition, we shall extend the ix editor to allow creating damage specifications as a secondary stage after exporting the base models from 3DS Max. It is anticipated that the module used for animation editing will also be usable for the damage system with little or no modification. See the Animation System TDD for details.

Additional entities will be defined to realize the expanded abilities of the new system. Such as:

* BreakableConstraint to create progressively reducing constraints
* ApplyAnimation to include an animation hierarchy to be applied to a node.
* DamageChild to be able to trigger child meshes to further break based on a collision id.

## Runtime Component