Unreal Engine* 4 Optimizations for 4.23 and Beyond Part 1: Chaos Destruction and Niagara Particle Systems

By: Scott La Fetra and Errin Miller

Introduction
At the Game Developers Conference (GDC) 2019, Epic revealed their new physics system, Chaos, and the destruction capabilities it brings to the Unreal Engine* 4 (UE4). Working with Epic, Intel game engineers have added Intel® Implicit SPMD Program Compiler (Intel® ISPC), that allows all current and future Chaos systems to take advantage of every CPU core available.

In this article series, we will go over a collection of optimizations and things to look for when including Chaos in your future UE4 projects. Part 1 focuses on the new Chaos Destruction Systems included in UE 4.23.

Enabling Plugins
In order to use the techniques mentioned in this article a UE 4.23+ project is needed and all appropriate plugins enabled for the new Chaos systems. The plugins window can be opened by selecting Edit and then Plugins in the editor menu toolbar. Once open, bring up all Chaos plugins by typing chaos in the upper right search bar. Check Enabled boxes for Chaos Niagara, Chaos Solver, and ChaosEditor. The Chaos Cloth plugin is selected in the figure below, but is not required for these tutorials.
Creating a Chaos Destruction System
The first step to creating a Chaos Destruction System (Chaos DS) is to select the static mesh desired within your map level in-editor. We will use the asteroid model pictured below for this, but the process is the same with any static mesh you create.
Once selected, we can turn our static mesh into a geometry collection. This is done by opening the new Fracture Editor tab in the Editor Modes panel and clicking **Generate Geometry Collection Mesh**, shown circled in red in Figure 3. After the geometry collection is created, the static mesh selected is automatically removed from the level and replaced with the new destructible meshes.

Geometry collections are how Chaos DSs are represented in our scene. For this article we created a geometry collection from a single mesh, but you can select multiple meshes in-editor before clicking Generate Geometry Collection Mesh to generate one Chaos DS from all the selected meshes combined.
With the Chaos DS created we are now able to fracture it. Within the Facture Editor window are the Fracture settings for the destructive system. There are a number of options when fracturing a mesh, but for this article we will use the Uniform Voronoi.

![Fracture Editor Mode tab showing Uniform Voronoi configuration.](image)

When setting the random range for the minimum and maximum number of chunks to be generated, we can see a preview of how the mesh will be fractured in the editor viewport.
Finally, once we are satisfied with how the mesh will be split, fracture the mesh by selecting **Fracture** at the bottom of the Fracture Editor panel.
Figure 7: Expanded view of a fractured mesh.

The basic Chaos DS created is destructible, but now we can increase the visual fidelity of the destructible with other Chaos features. By clustering a Chaos DS we can improve the performance of the system before it is broken, and get a more realistic visual when it is broken. While clustering does not have a large effect on graphical load, it greatly reduces the physics complexity, and might be something to consider if your scene is already highly CPU-intensive.

To cluster a Chaos DS we will use the auto cluster tool that the engine provides. First, we select all the chunks in our geometry collection by clicking **Select All**, shown circled in Figure 8.
Figure 8: Fracture Editor Modes tab with Select All icon indicated.

With all the pieces selected, click **Auto Cluster** and create a new level for the Chaos DS system. To switch between different levels, use the Level select menu, as shown in Figure 9.
This divides the Chaos DS into a handful of larger sections, each containing the small chunks originally created with the first fracture. These higher level chunks let you better control the overall destructibility as well as providing destructibility of the destructible itself; for instance, a concrete pillar breaking apart, and the chunks of that pillar falling and breaking into even smaller chunks when hitting the floor.

Figure 10: Fracture Level 1 mesh view.
Figure 11: Fracture Level 1 expanded mesh view.

Notices

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© 2019 Intel Corporation

Tweet: Intel game engineers added Intel® Implicit SPMD Program Compiler to allow Unreal Engine 4 Chaos systems to take advantage of every CPU core available.

Summary: Intel game engineers added Intel® Implicit SPMD Program Compiler to allow Unreal Engine 4 Chaos systems to take advantage of every CPU core available.