

Structure-Aware Procedural Destruction

submitted by

Thomas AE. Smith

for the degree of Doctor of Engineering

of the

University of Bath

The Centre for Digital Entertainment

May 2014

COPYRIGHT

Attention is drawn to the fact that copyright of this thesis rests with its author. This copy of the thesis has been supplied on the condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the prior written consent of the author.

This thesis may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation.

Signature of Author

Thomas AE. Smith

Summary

In this proposal and literature review a number of possible approaches to developing a procedural destruction system are investigated, and an approach developed to guide practical research and development in this area.

Chapter 1

Introduction

Paragraph on video games introducing context and the drive for increased fidelity, enabled by ever-improving hardware. Also mention variety.

Providing high-fidelity destruction requires investment. Some approaches to rendering damage applied to in-game assets scale better than others. There are a number of different axes on which investment may scale - by the level of fidelity provided by the system, or by the number of assets which may be damaged. In an environment where a variety of assets may have a wide variety of different types of damage inflicted, attempting to realistically render the results of these damage events results in a combinatorial explosion that quickly becomes unmanageable both in terms of processing it in real time and when attempting to supply the assets that support rendering damaged structures.

A number of approaches have been taken in order to attempt to automate the process both of determining what sort of damage should be rendered, and also procedurally generating the assets (meshes, textures, VFX) necessary in order to faithfully render this damage. To date, the majority of such procedural systems focus on a per-material or per-object approach that responds to a single kind of impact damage and merely varies the magnitude of the effect applied in order to respond to differences in damage kind — i.e. the difference between a single stray bullet impact and a nearby high-yield explosive detonation.

Where larger scale destruction effects are necessary, these are typically handled by a combination of small-scale procedurally destructible elements, and custom designer-written scripts that make use of higher level knowledge about the physical structure of the object being destroyed and the ‘intent’ of the destruction - whether it is simply intended to provide a visual effect, or will have some ludic ramifications on the players’ abilities or accessible areas within the game.

Some genres of game — notably space- or naval-combat sims, feature large physical structures that can be subject to [idiosyncratic] recognisable forms of destruction — for example, hull deformation as a result of impact, or separation of smaller components such as conning towers. The types of damage that each portion of such a ship may be subject to are dependant not only on the kind of damage applied, but also on the physical structure of the area affected and the supporting physical structures surrounding it. No existing procedural destruction systems apply knowledge about the structure of the affected object when generating possible damage outcomes, and so in this document a review of related research is presented, and a promising avenue for further investigation is suggested.

1.1 Document Overview

Chapter 2

Literature Review

In order to inform the development of the project, it is worth considering research in related areas. First, a range of existing approaches to procedural destruction systems within commercial games are considered — accompanied by academic overviews of particular techniques where available. Then, the concept of declarative solvers is introduced, with particular reference to existing projects and approaches that might provide promising initial directions. Finally, there is a brief overview of some of the other techniques that may be necessary in order to bind the abstract output of a solver system to a concretized visual representation within a game.

2.1 Existing Implementations

Any particular procedural destruction system is distinguished by the implementation chosen for two primary considerations: the [decision-making] approach, and the rendering solution. The [something] is often defined by the ludic environment of the system — design decisions will be made at the gameplay level about the particular model of damage most suited to the desired in-game experience. A suitable way of communicating this information to a player via visual (rarely, audible) representation can then be developed.

In order to provide context for the description of the proposed system, it is instructive to investigate the range of methods that have previously been used to model and represent damage and destruction within games to date. A number of the approaches described below are mutually incompatible, however there are a small number of subsets that are often used in combination with each other, and which may indicate useful techniques that could be applicable in the present domain. The sections below are arranged roughly in order of increasing implementation complexity, and therefore also

chronologically in order of representation fidelity.

2.1.1 No Destruction

In early and/or simple games, it is commonly the case that all game objects are simply indestructible, which requires no in-game damage metric or alternative visual rendering.

2.1.2 Scripted Destruction

The simplest possible form of destruction is simple presence/absence of an object in response to a particular in-game event — typically collision with a projectile fired by the player. More advanced implementations may replace the object with a visual effect such as an explosion.

2.1.3 Triggered Destruction

The overall ‘health’ or ‘structural integrity’ of an in-game metric, no visual representation. Possible death animation

2.1.4 Art swap

- various thresholds, identical each time. Fidelity scales with artist investment

2.1.5 Voxel-based

approaches

2.1.6 Destructible materials

2.1.7 Scripted systems

Previous research on destruction of comparatively small objects [vG11].

2.2 Declarative Solvers

LOG-Ideah Truth maintenance systems

2.3 Relevant Rendering Techniques

Chapter 3

Proposed Approach

Paragraph on overview of approach, to provide context for

3.1 Research goals

Working System, to demonstrate: ASP liveness Reification Loose Coupling

3.2 The Problem of Scale

Show off how much better this approach is M

Chapter 4

Evaluation

4.1 Performance

integration with a real-time game means that the timely performance of the system is as serious concern. It may become appropriate to perform compile time optimisations, pre-calculation and pre-caching of possible solutions in order to ensure that during a match the system is able to provide appropriate damage solutions in real time. Writing in 2008, Boenn et. al. [BBDV*08] suggest that while some of the faster ASP solvers at the time were responsive enough to provide an interactive experience while generating melodies, they did not at the time provide real-time performance. There appears to be no more contemporary research that indicates significant recent speed increases, and so this is likely to remain an active research area for the project. Evaluation of the performance of the system within the runtime context of the game environment is most likely to be performed via frame-rate comparisons throughout development. As the project should maintain a loose coupling with the game itself, it should remain relatively easy to selectively disable it for the purpose of comparison tests across similar in-game scenarios. A range of game events may be investigated both with and without the procedural destruction system enabled, and then logs of in-game metrics including frame-rate may be analysing in order to determine the impact of the processing load required by the implemented solution.

4.2 User Acceptance Testing - Developers

4.3 User Acceptance Testing - Players

As the game that the system will initially be developed within is being released on an Early Access platform, it will be possible to perform A/B testing and receive user feedback on varied incarnations of the system during development. This may range from informal feedback via community channels to focused user surveys designed to elicit specific impressions of all aspects of the game — including the procedural destruction system. In particular, it would be useful to evaluate the impact of the system on user enjoyment and effectiveness, specifically possible increased satisfaction when applying damage to ‘enemy’ ships and structures, and increased passive awareness of the status of their own and other visible ships for tactical decision purposes.

Chapter 5

Timeline

Due to the commercial nature of the project it is likely that a completed version of the system will be needed within the next year and a half explain the early access system incremental development start with minimal working system within six months develop further guided by feedback after system completion, further support / development may be necessary

Chapter 6

Conclusion

is ASP viable in real-time for solving the combinatorial explosion problem can we render asp solutions in a convincing manner is the system design sufficiently environment-agnostic

Bibliography

- [BBDV*08] BOENN G., BRAIN M., DE VOS M., ET AL.: Automatic composition of melodic and harmonic music by answer set programming. In *Logic Programming*. Springer, 2008, pp. 160–174.
- [vG11] VAN GESTEL J.: *Procedural destruction of objects for computer games*. PhD thesis, Department of Mediamatics Faculty of EEMCS, Delft University of Technology, 2011.