CM50175
Research Project Preparation

Thomas Smith

Centre for Digital Entertainment
University of Bath

April 20, 2014

Born Ready Games

- Guildford-based ‘indie’ games studio
- Sucessful ‘Strike Suit’ franchise
  - Two games across five platforms
  - Space-based fighter combat at large scale
  - ‘Play a key role in a larger story’
- In-house art team, publishing, PR

Born Ready Games — Strike Suit Zero

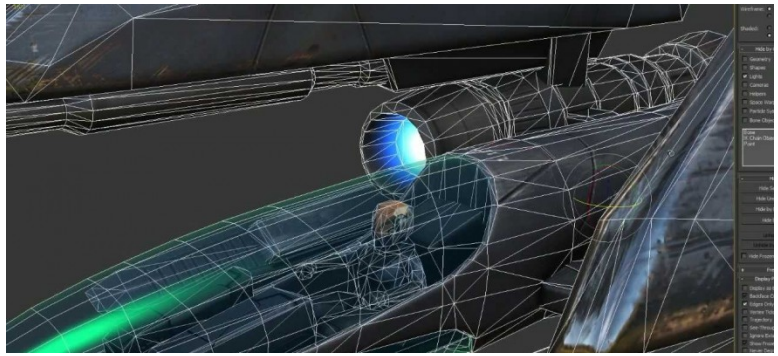
Strike Suit and capital ships

Born Ready Games — Strike Suit Zero

AXE fighter craft and
Thule research station

Art Pipeline

- High production overheads per asset
- Duplication of effort
  - Pristine mesh and textures
  - Damaged mesh and textures
  - Destroyed mesh and textures
- Static damage-based system for swapping meshes
- Procedural impulse vectors for detroyed meshes
- Larger ships have independent weapon hardpoints
- Relatively low variety of ship models in current game



Procedural Destruction

"The aim of the project is to explore and implement a declarative approach to the modelling of structure, so that reasoning about the effects of damage can take place over a knowledge-based representation from which a rendering can be synthesized automatically. The representation evolves over time in response to the damage inflicted, but could also be subject to other forms of failure arising from other environmental events."

— J. A. Padget

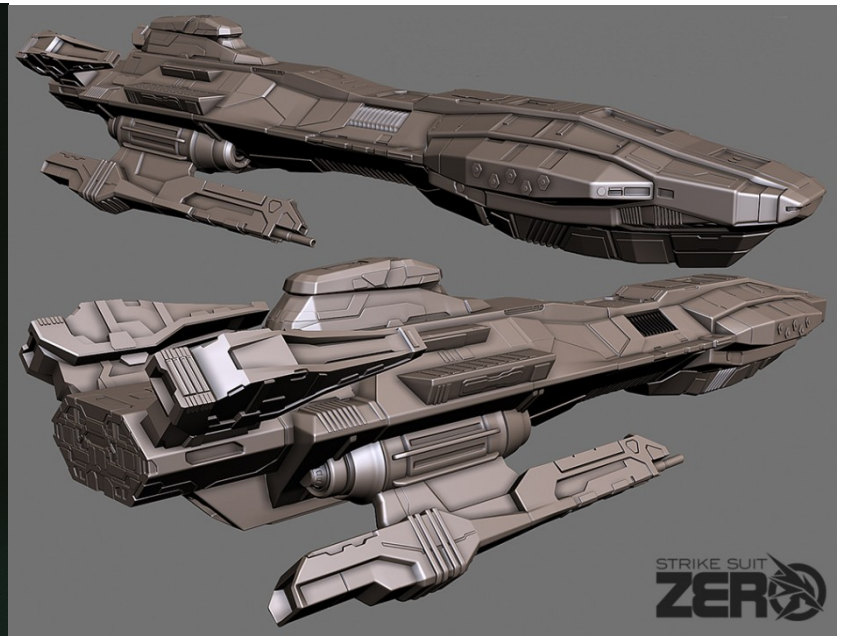
Procedural Destruction

> *"The aim of the project is to explore and implement a declarative approach to the modelling of structure, so that reasoning about the effects of damage can take place over a knowledge-based representation from which a rendering can be* **synthesized automatically**. *The representation evolves over time in response to the damage inflicted, but could also be subject to other forms of failure arising from other environmental events."*
>
> — J. A. Padget

- Research and build a procedural destruction system
- Generate custom damaged and destroyed appearances
  - Structure-dependent deformation or separation
  - Realistic response to damage type — laser, missile, etc.
- Real-time performance necessary

Context — Sample Capital Ships

U.N.E carrier ‘The Arcadia’



MUGE-class cruiser

Context — Size Comparison

AXE fighter craft vs. MUGE-class cruiser

Existing Approaches

Art Swap

One simple possible approach is to replace the rendered asset with pre-prepared progressively more damaged versions

- Current implementation — in-engine support
- Popular solution, used by many other games
- Low runtime cost — damage threshold checks

However:

- High production costs due to multiple asset versions
- Visually identical for every instance
- Fidelity scales with artist investment

Material-based destruction

Another option is to define the collapse behaviour of individual materials — similar to familiar systems that define the render behaviour

- Increasingly popular approach for varied destruction
- Small number of middleware solutions
  - **Havok**: pattern-based solid mesh subdivision
  - **PhysX**: experimental GPU-based rigid body destruction
  - Various other engine-specific custom solutions

However:

- High computational load — any quad might fracture
- Only works for single-material (homogeneous) items
- Does not account for structural information

Structure-aware solvers

Not currently used in industry — builds on work using constraints-based solvers for procedural level generation, and academic physics-based destruction research

- Possible destruction states are enumerable
- Structure and material behaviour are constraints
- Actual damage experienced is also a constraint
- Suitable damage effect can be generated and rendered

However:

- Potential high processing load — pre-selection needed
- Structure information must be provided

Proposed Approach

Answer Set Programming (ASP)

ASP is a declarative programming approach where *constraints* define bounds on the answer set, and a *solver* is used to generate valid instances of the set. To use ASP for procedural destruction, certain constraints must be defined:

- The possible types of damage inflicted
- The structure and materials of each ship
- The response of each structure/material to damage
- The actual damage sustained (during runtime)

In addition, for each possible damage effect generated, a rendering solution must be provided:

- Decals or mesh deformation for light damage
- Precomputed mesh subdivision for heavy damage

Integrated Tools

Benefits