Real-Time tearing and fracturing

Myriam Beauvais*
ID 260760034
COMP-559, Fundamentals of Computer Animation

Figure 1: *Star Wars: The Force Unleashed game content and screenshots courtesy of LucasArts, a division of Lucasfilm Entertainment Company Ltd. ©2009 Lucasfilm Entertainment Company Ltd. or Lucasfilm Ltd. All rights reserved. Image taken from Parker and O'Brien, 2009*

1 Abstract

2 Introduction

A lot of different objects, when under some persistent and or increasing forces, will end up getting distorted and breaking apart. Because virtual objects are often put in such situations in video games, movies or simulations, addressing this topic is very pertinent and interesting in order to recreate realistic sequences.

The project proposed is the implementation of an interactive application allowing for deformation, tearing and fracturing of meshes in real-time, according to the method described in [Parker and O'Brien 2009]. Optionally (in function of the advancement and progress rate), to improve the deformation of a mesh the technique described by [Rivers and James 2007] will also be implemented.

3 Related Work

4 Technical details

- ~~Implementation of a basic fps camera, and of a picker (modified for using forces)~~
- Creation of the scene:
  - Create particles as gameobjects with sphere primitives, having bound the Rigid Body script as component to enable PhysX (Unity's physics engine) calculations of collisions.
  - bind particles with joints. Used HingeJoints (describe various tests done). Add reflexion about use of spring.cs (and actually use it if have time to)
  - Create a mesh having vertices at the particles' position.
  - reupdate the mesh according to the particles new positions (indices stay the same
  - Although the scene is in 3D, all calculations and implementations take into account that we are only dealing with planes.
- Fracturing...

4.1 Additional concepts and tools needed

Element needed	Solution
3D Viewer	Unity project
Interface	Unity's UI tool
Linearized Backward Euler Integrator	Self-Implemented
Collision Detection	Unity's physic engine

4.2 Implementation

The methods and algorithms described were implemented in Unity3D with the use of custom scripts. The project's setup is a scene composed of three (3) *GameObject*s; a *Camera*, a *Light* and a *Scene Manager*. To allow interaction with the objects, a basic First Person Camera behaviour was implemented and added to the *Camera object*. The *Scene Manager* object has two *Script Components*. The first one describes the creation of the test scene, with an object that can be fractured (called *FracMesh*) and the second one allows for picking and dragging objects in the scene using an external force.

4.3 Scene Creation

The test scene has only one object which can be fractured and a plane. The breakable object is forced to be a plane itslef for simplification purpose, and it's size can be fixed by a parameter available to the user. The decision of limiting the shape of the object to a plane allows to treat it as a simple 2D mesh and implement [O'Brien and Hodgins 1999]'s fracturing algorithm using triangles instead of tetrahedrons.

*e-mail:myriam.beauvais@mail.mcgill.ca

5 Results

6 Conclusions

References

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proceedings of ACM SIGGRAPH 1999*, ACM Press/Addison-Wesley Publishing Co., 137–146.

PARKER, E. G., AND O'BRIEN, J. F. 2009. Real-time deformation and fracture in a game environment. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 156–166.

RIVERS, A. R., AND JAMES, D. L. 2007. Fastlsm: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph. 26*, 3 (July).