Simulating Visual Geometry

Seminar: Current Topics in Physically-Based Animation

Patrick Bayer

December 28, 2016

Contents

1 Introduction

- Ultimate goal in virtual worlds: What you see is what you simulate
- Unfortunately not so easy to fulfill because of performance restrictions in real time environments $\rightarrow$ two or three different representations for simulated objects (visual mesh, simulation mesh, collection of convex shapes for collision handling)
- Possible mimatch of collision behaviour and visual appearance

- New method with a single representation for both, simulation and collision handling, and an almost identical visualization mesh
- Objects created as triangel or quad meshes that are not prepared for physically-based animation $\rightarrow$ embed the visual mesh in a simulation mesh or attach it to a more general structure
- Use of a simulation mesh close to the visual mesh/directly derived from the visual mesh
- Convex shapes as primitives
- General graph to connect the primitives
- Primitives are treated as oriented particles and represented with convex polyhedra
- Oriented particle method for simulation

2 Related Work

- Combining the rigid body formulation with a deformable model
- Fracture and tearing algorithms

2.1 Simulation Models

- General, non-conforming unstructured mesh
- Unified solver based on position based dynamics
- Oriented particle method
- Shape matching

2.2 Embedding of a Visual Mesh

- Embed visual mesh in a tetrahedal mesh and a regular grid
- Blend the transformations defined in nearby particles by their position and orientation

2.3 Fracturing and Tearing

- Use of pre-fractured models
- Pre-defined fracture patterns at the impact location
- Representing objects and fracture patterns with convex decompositions

3 The Method

3.1 Physical Mesh Creation

- Input: Visual mesh with triangle and quad faces potentially intersecting each other
- Physical properties that cannot be derived from the geometry are defined by the user
- Create volumetric convex primitives if needed
- Average positions of the vertices are used for simulation

3.2 The Simulation Method

- Oriented particle approach
- Shape matching
- Standard collision detection

3.2.1 Oriented Particle Method

- Oriented particles store rotation, spin and the usual linear attributes (postion, velocity, ...)
- overall look and feel of objects > accurate reproduction of their small scale behaviour

3.2.2 Shape Matching

3.3 Deforming Primitives

- Oriented particle method only modifies the location and orientation of primitves → potential gaps and collision artifacts
- Primitives have to stay convex in order to perform collision handling and fracturing
- Deform primitives in using the local affine deformation matrices

3.4 The Visualization Mesh

- Vertices of different primitives are joined for visualization using global IDs
- Compute the resulting visual positions with help of the average positions of joined vertices

3.5 Plastic Deformation

- Treat the entire object as a rigid body if there are rigid parts
- Rearrange the primitives only when they are plastically deformed
- Deform objects if the relative normal velocity is above a threshold defined in the material
- Deformation offset = relative normal velocity * time step

3.6 Subdivision, Fracturing and Tearing

Fracture

- Connected set of convex polyhedra as fracture pattern
- Align fracture pattern with the impact location
- Compute the cuts of all the primitves of the objects against all fracture cells → new independent objects
- Modification: Keep all the resulting pieces in the same object
- Always apply fracture patterns in the undeformed configuration

Tearing

- Use a threshold for the distance between connected primitives in order to check whether the connection is broken
- Use only a subset of the links including the links introduced by applying a fracture pattern

4 Results

5 Conclusion and Future Work

References