Thomas Lefley
Girton College
tl364

Computer Science Part II Project Proposal

Mesh to voxel transformation for optimised physics-based interactions

24/10/2014

1 Introduction and Description of the Work

1.1 Motivation

Physics simulation is one of the key components of a modern game engine. The more accurate the simulation is, the more immersive any games built with the engine are likely to be. If the world around the player does not react as expected then it is difficult to become immersed in the game environment.

1.2 Problem

Most game engines simulate the world using a mesh and rigid body representation. This limits how accurate physical reactions, such as collisions, can be regarding object damage. If two rigid bodies collide then their vector components can be resolved based on their speeds and angles of incidence amongst other properties. However computing the damage to their bodies is less trivial, the meshes can be dented and predetermined joints broken but this leads to objects appearing to lose mass. Damage can look predetermined and independent of the actual collision. As meshes are hollow representations of objects, any calculations involving mass are difficult.

1.3 Solution

Representing objects volumetrically allows for more accurate object destruction. Any fragments which would be severed from an object should consist of a portion of that object's mass. If the object is represented as voxels then this can be simulated by breaking voxels from the parent object. Computing the physical properties of the fragments dynamically can then be done based on the number and orientation of voxels they contain[1].

Permanently representing every object volumetrically would become inefficient in scenes with many objects. Instead objects will be represented with meshes and converted to volumes in real-time on collision[8]

For example, if a bullet is fired at a brick wall, both the bullet and wall will be rigid meshes when the bullet is shot. On collision, both the wall and bullet will be voxelised. Assuming the bullet consists of only one voxel, no destruction calculations will be performed on it. Using the force of impact and physical parameters of the wall, it will be calculated which voxels should break off. These voxels will then be transformed back into rigid meshes and the correct movement vectors will be applied. The wall's mesh will be reformed based on the new volumetric structure. This is illustrated in figure 1.3.

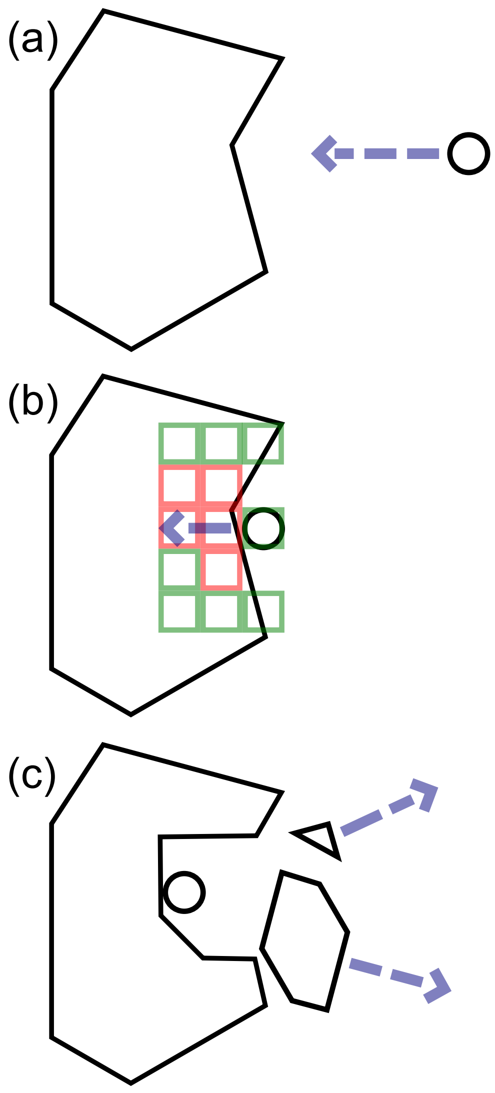

Figure 1: A simple rigid mesh projectile is fired towards another rigid mesh (a). On collision a portion of each body is voxelised (b) depending on the force of the projectile and physical parameters of the objects. It is determined which voxels will detach, shown in red. Meshes are reformed around all separate bodies and physical forces are applied (c).

2 Background

2.1 Mesh/Voxel Conversion

Binary voxelisation of polygonal objects is a memory efficient method for defining occupied space and can be achieved in real-time on modern GPUs[6, 5, 2]. I will implement solid voxelisation as opposed to surface volexisation as I require the entire volume of an object to be modelled for my volumetric destruction.

Marching cubes exists as an algorithm for constructing 3D surfaces from volumetric data[7]. the implementation of this algorithm I will be using runs in real-time on modern GPUs[4].

2.2 Related Work

Nießner et al present a method for real-time collision detection of patch based objects using mesh to voxel transformations [8]. While their work is on collision detection, it does demonstrate that real-time mesh to voxel conversion is viable for use in resolving physics interactions.

Miguel Cepero is developing a procedural voxel engine Voxel Farm[3]. His work includes volumetric destruction of objects in a manner similar to this project's aims[1].

3 Methodology

3.1 Starting Point

I will be working using the Unity3D physics and game engine, with which I have prior experience. The engine will provide general features such as rendering, rigid mesh physics and game object management which I will expand upon to achieve my more specific goals. As Unity3D is a widely used commercial engine, its use will also allow for comparable evaluation against scenes implemented using only the base features.

I will be using an existing implementation of the marching cubes algorithm on the GPU for voxel to mesh transformations[4].

3.2 Substance and Structure of the Project

This project will involve implementing mesh to voxel transformations efficiently at runtime based on existing methods[6, 5, 2], as well as the calculation of correct physical responses amongst the individual voxels of the volumetric object. The nature of these physical responses will depend on the physical parameters of the parent object. These parameters and how they impact the physical destruction will have to be defined.

Voxelisation will also be required to be lazy for efficiency. Objects should only be voxelised if the triggering collision is above a certain force threshold which will be defined by the object's physical parameters. If the impact force is not above this then it can be assumed that no damage will be done and there is no need to voxelise. Whole objects should not be voxelised, only a large enough portion to bound the damage that will be done.

The project has the following main sections:

1. Real-time voxelization of polygonal meshes on the GPU based on one of the methods cited above.
2. Destruction of volumetric objects based on their physical parameters. By propagating collision information from each voxel to its orthogonal neighbours from the point of impact, it can be calculated which voxels should be dissociated from the parent object and become independent objects.
3. Real-time user modification of objects by adding or removing voxels.
4. Real-time mesh formation from volume data on the GPU using a plugin for Unity3D[4].
5. Writing the dissertation.

3.3 Further goals

If time allows, two further areas will be explored:

1. **Object deformation** As well as modelling object destruction while volumetric, object deformation could also be simulated. By representing each voxel as being connected to its neighbours by a rigid bond, the orientation of these bonds can be changed to deform an object based on its malleability, brittleness, density, etc. .

2. **Structural integrity.** The main focus of the project is physics caused by interaction such as two objects colliding. A further goal would be modelling the internal physics of resting objects. Take for example a wooden beam fixed at one end with a weight at the other, if this weight is too heavy, the beam should break at the correct point. This simulation is trickier as the cue for when to voxelize and resolve is not as obvious.

4 Evaluation

4.1 Success Criteria

The following should be achieved:

- Implement and demonstrate realtime conversion between mesh and volumetric objects.
- Implement and demonstrate object destruction based on collision information and physical parameters.

4.2 Evaluation Criteria

The following are evaluation criteria:

- A scene with a reasonable number of collisions should run with a framerate above 30FPS and no less than half that of an identical scene using only traditional rigid body physics.
  - Comparable scenes built using the base Unity3D rigid mesh physics engine will be used for efficiency evaluation. Metrics to be compared are physics realism, GPU VRAM usage and render thread time.
  - My own computer will provide a high specification test case, the MCS machines found in the Intel Lab at the William Gates Building will represent a mid range specification.
- Conversion between mesh and volumetric objects should not be noticeable.
  - There should be no obvious slowdown or framerate drop when a conversion occurs.
  - The delta time between voxelisation and mesh reformation should be below 0.5s.

4.3 Further Goals

The following are stretch goals to be achieved if time allows:

- Volumetric deformation
- Structural integrity.

Further detail can be found in section 3.3

4.4 Demonstration

At the progress meeting I will demonstrate a stuttered scene in which a bullet will be fired at a brick wall. The scene will be paused at the point of impact for explanation and highlighting of these processes involved:

1. A portion of the wall and bullet will be voxelised as determined by the force of impact, and the volumetric representations shown as in the video demonstration from Nießner et al[9].
2. As little progress will have been made on volumetric destruction at this point in the project, instead all the voxels will be removed.
3. The wall's mesh will then be rebuilt from the updated volumetric representation.

A second scene will also be demonstrated in which a ball is dropped from a low height onto another brick wall. It will be shown that no voxelisation takes place as the collision force is not great enough.

5 Resources

I will be working with a free license of Unity3D installed on my own computer with the following specifications:

- 3.40GHz i5-4670k
- 8GB RAM
- 400GB remaining SSD storage
- 750GB remaining HDD storage
- AMD Radeon HD 7950 GPU
  - 3GB GDDR5 Memory

I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure.

All source code will be be backed up and version controlled using Git with GitHub. Documentation, as well as any further resources, will be created and stored in a Microsoft OneDrive cloud folder with regular backups to Google Drive as well as my own hard disk drive.

6 Timeline

Sprint 1 (24/10/14–7/11/14)

Implement a mesh to voxel transformation algorithm from section 2.1 and become familiar with the voxel to mesh transformation plugin.

Milestones: Some example code, which will probably not be used in the final project, and some example scenes demonstrating conversion.

Sprint 2 (7/11/14–21/11/14)

Refactor the mesh to voxel algorithm to only voxelise a portion of an object.

Milestones: Working voxelisation of predefined portions of an object.

Sprint 3 (28/11/14–5/12/14)

Hook the voxelisation algorithm to collision cues above a predefined threshold.

Milestone: Examples of conversion occurring and not occurring as required on cue.

Sprint 4 (5/12/14–16/1/15)

Review code and progress over the Christmas break. Ensure everything is well documented, including source and evaluation of current progress.

Sprint 5 (16/1/15–30/1/15)

Write the initial chapters of the dissertation. Write the progress report. Build the progress demonstration as detailed in section 4.4

Milestones: Preparation chapter of Dissertation complete. Submission of progress report. Demonstration ready.

Sprint 6 (30/1/15–13/2/15)

Begin to work on physically based destruction algorithms for the splitting apart of volumetric objects, including transferral of force to any objects formed from dissociated voxels. Define the parameters which will impact physical reactions.

Milestones: Ability to define physical properties for game objects.

Sprint 7 (13/2/15–27/2/15)

Complete work on physically based destruction algorithms.

Milestones: Data confirming impact propagation through voxel bonds.

Sprint 8 (27/2/15–13/3/15)

Integrate the voxel to mesh plugin with the result of volumetric destruction.

Milestones: Able to see object split apart as the result of an impact of the correct magnitude.

Sprint 9 (13/3/15–27/3/15)

Implement real-time user modification of objects.

Milestones: Example scene where playing is able to modify objects via physical and non-physical means.

Sprint 10 (27/3/15–10/4/15)

Code review and code freeze. All success criteria should be implemented by this point. No new features will be written during this period, only bug fixes and optimisations.
Further dissertation progress will be made.

Sprint 11 (10/4/15–24/4/15)

Create comparable scenes using base Unity3D physics for profiling.
Profile against these scenes as well as on computers with varying specifications.

Sprint 12 (24/4/15–8/5/15)

Finish the dissertation, preparing graphs for insertion. Review the whole project, check the dissertation, and spend a final few days on whatever is in greatest need of attention.

Final Sprint (8/5/15–15/5/15)

Final proofreading and submission of dissertation.

Milestone: Submission of Dissertation.

References

[1] Miguel Cepero, (2014), *Appetite for Destruction*,
Available: http://procworld.blogspot.co.uk/2014/08/appetite-for-destruction.html,
Last accessed 22th Oct 2014.

[2] Miguel Cepero, (2011), *OpenCL Voxelization*,
Available: http://procworld.blogspot.co.uk/2011/04/opencl-voxelization.html,
Last accessed 22th Oct 2014.

[3] Miguel Cepero, (2014), *Procedural World*,
Available: http://procworld.blogspot.co.uk/,
Last accessed 22th Oct 2014.

[4] Scrawk, (2014), *Marching Cubes on the GPU in Unity*,
Available: http://scrawkblog.com/2014/10/16/marching-cubes-on-the-gpu-in-unity/,
Last accessed 22th Oct 2014.

[5] *Fast Parallel Surface and Solid Voxelization on GPUs*,
M. Schwarz, H. Seidel, *ACM Transactions on Graphics*, Dec 2010, pp. 179:1–179:9.

[6] *A Low Cost Antialiased Space Filled Voxelization Of Polygonal Objects* ,
S. Thon, G. Gesquire, R. Raffin, *Graphicon*, 2004.

[7] *Marching Cubes: A High Resolution 3D Surface Construction Algorithm* ,
W. Lorensen, H. Cline, *SIGGRAPH Comput. Graph.*, July 1987, pp. 163–169.

[8] *Real-time Collision Detection for Dynamic Hardware Tessellated Objects* ,
M. Nießner et al, *Eurographics*, 2013.

[9] M. Nießner et al, (2013), *Real-time Collision Detection for Dynamic Hardware Tessellated Objects*, [Online video],
Available: http://graphics.stanford.edu/∼niessner/papers/2013/
2collision/niessner2013collision.mp4,
Last accessed 23th Oct 2014.