Thomas Lefley
Girton College
tl364

Director of Studies
Chris Hadley

Supervisor
Flora Ponjou-Tasse

Overseers
Dr. Neal Lathia
Dr. Markus Kuhn

Computer Science Part II Progress Report

Mesh to voxel transformation for optimised physics-based interactions

Thursday 29th January, 2015

1 Anticipated and achieved progress

The following is the original proposed time-line of work to be done up to the current point of progress, with analysis of each milestone.

Sprint 1 (24/10/14–7/11/14)[Completed]

Implement a mesh to voxel transformation algorithm 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.

Voxelisation of closed meshes has been achieved on the GPU as required. A HLSL DirectCompute shader implementing the Schwartz method for solid voxelisation, as detailed in the project proposal, was found and a C# driver written. Voxelisation of the 'Stanford Bunny' model, composed of 69,666 triangles, is voxelised in an average of 100ms.

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.

Due to the unordered datastructure used by Unity3D to store mesh vertices, a method for voxelising only a portion of a mesh has not yet been identified which does not increase complexity. As this optimisation is not part of the success criteria it has low priority and so has not yet been completed.

Sprint 3 (28/11/14–5/12/14)[Completed]

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

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

Voxelisation only occurs if the collision force of the two objects is greater than the predefined threshold of the object to be destroyed. The collision force is calculated based on the velocities and masses of the colliding objects.

Sprint 4 (5/12/14–16/1/15)[Completed]

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

A log of progress and thought processes has been maintained, including diagrams.

Sprint 5 (16/1/15–30/1/15)

Write the initial chapters of the dissertation. Write the progress report. Build the progress demonstration.

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

The progress report is complete and preparation for the demonstration ongoing. No work has yet been done on the written dissertation.

Sprint 6 (30/1/15–13/2/15)[Completed]

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.

Methods for identifying the point of impact in voxel space, as well as propagating the impact force within the solid have been implemented. The latter process is working as required however it will require further speed optimisation for the real-time application. Physical properties can be defined for objects and are used within the algorithms, their use can easily be changed at a later date if required to increase physical accuracy.

Sprint 7 (13/2/15–27/2/15)

Complete work on physically based destruction algorithms.

Milestones: Data confirming impact propagation through voxel bonds.

As above, visual data confirms that impact propagation is working, however further optimisation is required.

Review

While it would appear that I am ahead of schedule, having completed work a month before originally planned, I believe that the original allocation of time was skewed. My plan did not give much time for evaluation and writing, aspects which will likely take more effort than anticipated.

2 Current accomplishments

At this point in time, when two objects collide, either which are flagged as destructible will be voxelised, the impact force and point calculated and the force propagated through the voxelised objects. This force propagation is achieved by determining a series of points, normally distributed in a radius around the impact point, with the force of the collision as well as physical properties of the object determining the radius and number of points. Each voxel is then mapped to the point it is closest to and these groupings will form the fragments created by the collision.

This approach is based on a Voronoi Diagram which are commonly used for applications such as object destruction in computer graphics[1].

3 Difficulties faced

Voxelisation

Implementing solid voxelisation on the GPU took much longer than anticipated and I faced several difficulties. Originally I had planned to find an existing library and refactor it to suit my needs as the voxelisation of objects in not the focus of the project. However, no such library that could be easily integrated with Unity3D could be found.

My first attempt to solve the problem involved re-writing a CPU based JAVASCRIPT solution in HLSL. However this ultimately did not work as the CPU code relied heavily on the use of variable length loops and arrays, which are not available in HLSL due to code being unrolled at compilation time.

Eventually I found a DirectCompute shader accompanied by C++ driver code and was able to port the driver to C# for successful use in Unity3D. However, this whole process took until the beginning of January to complete, much longer than I anticipated.

Voxelisation of object subsections

Originally I had planned to optimise voxelisation such that it only occurred on a portion of the object to be destructed. This would be so that, for large objects with many vertices, unnecessary voxelisation of areas which would be be damaged would not have to occur, reducing voxelisation time.

However, Unity3D stores vertices in an unordered array, meaning determining which are within an area and should be passed on for voxelisation would require looping over all vertices in the CPU portion of the algorithm. This would produce a much higher time overhead than it would save in the GPU stage and so has not been implemented.

Should an efficient method of filtering the vertices as required be devised then this optimisation will be implemented. Such an implementation would involve processing the vertices into a tree structure to make neighbourhood searches fast. However, as it is not explicitly part of the success criteria it has been left for now.

[1]Sara C. Schvartzman, Miguel A. Otaduy, (2014), *Fracture Animation Based on High-Dimensional Voronoi Diagrams*, Available: http://www.gmrv.es/Publications/2014/SO14/I3D2014.pdf, Last accessed 29th Jan 2015.

4 Work to be completed

Marching cubes

An implementation of the Marching Cubes algorithm for creating a mesh from voxel data has been found which runs on the GPU. The outputs from the destruction algorithm must be meshed using this and this is the next step in the project. One difficulty I may face at this stage is if there is significant detail loss each time a model is voxelised and then re-meshed. Should this be the case, then I anticipate two alternatives.

The first would be using an approach whereby the fragment meshes are formed using a more traditional Voronoi approach for fracturing rigid meshes, with the internal voxel structure being used to increase the accuracy of the physical properties of each fragment, such as mass.

The second would involve completing a subsection voxelisation method such that the undamaged mesh is not altered. The decision will depend on the extent of the detail loss as well as the time complexity overhead imposed by each approach.

Parallelisation

Currently, all parts of the destruction pipeline run sequentially with the physics engine thread, causing stuttering. One optimisation to be completed is to move the pipeline to its own thread however this is not trivial due to Unity3D's lack of thread safety in some areas.

Evaluation

Scenes must be constructed and various performance metrics profiled for use in the evaluation of the success of the project. Unity3D does not have any native object destruction meaning that a third party implementation will have to be used or devised. Several exist which use a Voronoi approach or otherwise without the use of voxels, making them ideal for comparison[2][3].

Dissertation

The dissertation must be written. Work on this will begin after the progress demonstration with the aim of having it finished by 8th May at the very latest.

Real-time user modification of objects

This goal refers to the ability to 'paint' voxels onto an object in real-time to change its shape. As it is not explicitly part of the success criteria it will not take precedence over any optimisations required to make my solution for object destruction viable as defined by the evaluation criteria. As such this previously set milestone will unlikely be achieved.

[2]http://meshinator.blogspot.co.uk/
[3]http://www.dunnalex.com/