Destroying LA for "2012"

Nafees Bin Zafar David Stephens Mårten Larsson Ryo Sakaguchi Michael Clive
Ramprasad Sampath Ken Museth Dennis Blakey Brian Gazdik Robby Thomas

Digital Domain



For the movie "2012", Digital Domain was tasked with destroying its home city of Los Angeles... digitally. The destruction effects would be seen up close, and there had to be a lot of it. The solution involved turning to geometric fracturing tools, and rigid body dynamics (RBD) simulations. However, this sequence was going to require two orders of magnitude more detail than we had accomplished before.

Destruction Toolset

Organic fracture shapes such as the rock formations in the fissure wall could be created using high resolution level set techniques (CrackTastic). However when fracturing buildings this approach tends to produce too many polygons in order to maintain sharp corners and seamless fracture lines. Hence we also developed a polygon-based fracture tool dubbed PolyBuster. A user controlled distribution of points defined 3D voronoi cells which served as cutting planes. The input geometry is then iteratively cut, and the newly opened faces are capped. Various rendertime displacements were employed to introduce additional geometric details on the exposed faces.

Managing upwards of 300,000 RBD objects stresses the limits of RBD solvers and animation systems. The available third party solutions would not meet our requirements. Several physics engine SDKs used in video games fit our performance and scalability needs. We chose to use the Bullet Physics engine because it was open source and had a very active user community. Our RBD toolset, called Drop, was implemented as a set of Surface OPerator (SOP) plugins to the Houdini animation software.

All the objects in a simulation are input as triangle meshes, and given a unique ID. Instead of introducing a custom data type to represent an RBD object, we generate a single point for each object with attributes for RBD control parameters such as mass and velocity. This allows artists to manipulate objects and simulation parameters using built in Houdini operators, and avoids the overhead of processing all the geometry for that object. The object geometry is fed directly into the final solver operation. The solver node outputs control points containing the object transforms and body state. The output transforms are then applied back to the input geometry or to high resolution render geometry.

Airport and Fissure Sequence

The main simulation components in the Airport and fissure sequence were the ground, roads, buildings, and street level elements like trees and cars. These vastly different types of objects had to behave in an art directed way. In controlling how an object moves we were essentially trying to represent elastic deformation behavior with rigid bodies. This kind of behavior can be approximated by the use of constraints. Bullet provided many different types of constraints such as point-to-point, and hinge types. Each constraint was represented in our system as a point with control attributes and IDs of the objects they constrained. By controlling placement of constraints and setting their intrinsic parameters, we could create many different material behaviors. For example, rigging a building involved fracturing the building geometry, then seeding point-to-point constraints between the pieces based on nearest neighbors. Additional constraints could be placed to control zones of collapse. A final set of constraints would bind the bottom of the object to the ground. Adding a little bit of noise to the constraint parameters provided very natural spatial variations for the destruction animation. We had to enhance the constraint system in Bullet by adding dynamic constraint parameter updates, including breaking constraints based on displacement or constraint restoration impulse limits.

Rigging vehicles and trees involved a specific geometric arrangement of constraints. We built several different vehicle rigs to represent different types of cars and buses. The car simulation results were then applied to the specific car models. Results of the tree simulations were applied to the tree geometry with a lattice based deformer.

While every building had a unique simulation, the fissure wall sections were more generic and less visually recognizable. Therefore we created a reusable library of simulations. Each library element was a long simulation (800+ frames) of a square patch of ground collapsing. The artists would lay down curves to indicate the fissure path in their shot, with each curve segment indicating which library element to load at that location.

Collapsing Downtown LA

The buildings in downtown LA had to collapse in a very specific way. Some of this control could be achieved with the constraints, but in certain cases we had to incorporate keyframed animation. The transition from keyframed animation to simulation was handled dynamically per object by thresholding collision impact magnitudes. Though we had to represent a wide range of object sizes, the effect of small objects on a large one is negligible. In order to optimize simulation runtimes we used a layering approach. The output of the large object simulations would be reimported into a simulation for smaller objects. We also added information about the maximum and total collision based impulses on the simulation output points, and generated information about constraint states. This data was then used to determine location and timing of the smaller objects. For example, broken constraints from the building simulations were used to trigger particle simulations to represent broken shards of glass. Each building required simulating 50k to 100k dynamic objects, and each frame required less than 2 minutes.

Conclusions

Our goal was to create an efficient set of tools, and a flexible artist workflow to address 90% of our needs. Utilizing a robust and optimized RBD system and reducing execution of the operator network in Houdini gave us very fast simulations. The point based data representation reduced data processing in Houdini and provided interactive workflow for the artists. It also allowed them to build a lot of custom art direction tools. Methods of optimizing geometry for the simulation will lead to faster artist throughput.

SIGGRAPH 2010, Los Angeles, California, July 25 – 29, 2010.
ISBN 978-1-4503-0210-4/10/0007