Merging and Unmerging Rigid Bodies

Otman Benchekroun, Eulalie Coevoet, Paul Kry

When two objects collide in the real world, the collision results in an abrupt change of velocity for both objects that leads to them flying apart or moving together. Modeling this for computer simulations can be resource intensive, even if we assume these objects to be rigid bodies (non-deformable). With many rigid bodies in a scene, it is required to check if each body collides with every other body, and if so, to determine the collision location. The contact forces that exist in each of the found collisions need to be determined, which can also be a taxing process [1, 2] Contact resolution is commonly handled by a Gauss Seidel solver [2] wherein we solve for the contact forces by iterating through all found collision points *N* times until our solution converges. We are interested in dealing with scenes that contain large numbers of contacting bodies, making the vanilla solution simulation too slow for real-time simulation. Many popular solutions exist to optimize these collision detection/resolution steps such as pruning unnecessary contacts [3] or avoiding unnecessary collision detection steps by partitioning space into a boundary volume hierarchy [4]. We go a different route, one that starts with the idea that in a giant stack of contacting rigid bodies, if these bodies have no relative velocity relative to each other, the entire stack could be considered as one single rigid body. This approach would allow us to skip the entire contact detection/resolution steps entirely. The first part of our research focuses on how we should identify groups of rigid bodies that can be treated as one. We propose that after two bodies have been in contact for a certain number of time steps, and their relative velocities are monotonically decreasing and below a user set threshold, these bodies can be merged into an entity that contains both sub-bodies and regulates both of their movements. We name these merged entities rigid collections. In addition, a rigid collection shouldn't stay merged forever; the second part of our research focuses on when an entity should unmerge and how exactly it should do so. If a new force were imposed on the body, we'd then want the bodies in the stack to separate appropriately. This introduces the problem of finding an appropriate criterion for unmerging rigid collections. We implement a naïve method first, wherein we check the sum of all forces acting on each sub-body in the collection and compare that sum to a threshold to unmerge. We've found any method involving a threshold will introduce undesired inaccuracies at some point. A better proposal for unmerging would be by remembering all contacts present in a collection, and to do a single Gauss Seidel iteration time-step to get a grasp on how contact forces are changing; if the new forces lie outside of the frictional tolerance range, that body should be split from the collection. The ability to create worlds with thousands of interacting rigid bodies and to simulate them in real time would be hugely beneficial to the video game industry, and we believe perfecting the merging-unmerging algorithm could be a big step in making this happen.

This project was performed under the supervision of Paul Kry with the collaboration of a post doctoral researcher Eulalie Coevoet. Both helped come up with ideas on merging-unmerging criteria while all the implementation, code and testing were completed by Otman Benchekroun on top of a basic provided physics codebase by Paul Kry.

References:

[1] Eran Gundelman, 2003, Non-Convex Rigid Bodies with Stacking

[2] Kenny Erleben, 2007, Velocity-Based Shock Propagation for Multibody Dynamics Animation

[3] Ming C. Lin, 1995, Fast Algorithms For Penetration and Contact Determination Between Non-Convex Polyhedral Models

[4] Gino Van De Bergen, 2004, Collision Detection in Interactive 3D Environments, Chapter 5