Handling of plasticity in physics engine

June 16, 2016

1 Introduction

The application of modern computer game techniques enables the description of complex dynamic systems such as military vehicles with a high level of detail while still solving the equations in real-time. Film production and war games, in particular, is a key area that have benefited from simulation technology. In practice, games are often accomplished using an open-source platform such like ODE - Smith (2001-2007), Bullet Physics - Coumans (2003-2016) and Box2D - Catto (2007-2015).

Computational methods used in physics engines are divided to modules that handle collision detection and contact description and modules that handle solution of equations in real-time. Equations need to be solved can further be subdivided to be associated to motion, constraints and collisions. Velocity-based formulation is typically used in constraint based rigid body simulation frameworks as collisions cannot be handled easily in acceleration based formulations. Friction is typically taken into account and mechanical joints are handled by constraint equations. Detailed description of various components can be found in e.g. Erleben (2005).

Plasticity is not typically taken into account in gaming solutions. Breaking of various objects typically takes place based on collision or impulse. Nevertheless, breaking of steel or reinforced concrete structures using this approach is not appropriate making a simulation to look unrealistic. Theory for handling of plasticity has been presented already in Terzopoulos and Fleischer (1998). Müller et al. (2004) and Müller et al. (2005) present a method for modeling and animating of elastic and plastic objects in real-time using point based animation. This approach is not been widely used in simulation applications. On major issue is collision handling of deformable objects.

This study will introduce an approach to account plastic deformation in game applications. In the introduced method, the plastic deformation takes place if force or moment exceeds given limit, deformation absorbs energy and joint breaks if plastic capacity is exceeced. The approach is based on using joint motors to model plasticity. (Erleben, 2005, p. 90) suggests similar method for modelling friction in joints. Adjacent objects are connected by motors. Motor power production limits are estimated based on plastic section modulus. Joint breaking is accounted by summing plastic deformation and comparing it to predefined material based limit. Elastic part of deformation is modelled by employing spring description which is based on modification of existing constraint in Bullet Physics.

Approach presented in this work can be used in gaming industry to provide more realistic simulations without significant extra work. For gaming purposes presented method works best in scenarios where connected parts are relatively heavy. This allows normal integration timestep to be used without stability issues. This kind of metodology also opens large area of combining old structural analysis methods to modern simulation frameworks.

References

Catto, E. (2007-2015). *Box2D A 2D Physics Engine for Games*. url: box2d.org.

Coumans, E. (2003-2016). *Bullet Physics Library*. url: bulletphysics.org.

Erleben, K. (2005). *Stable, Robust, and Versatile Multibody Dynamics Animation*. Ph.D. thesis. University of Copenhagen.

Müller, M., Heidelberger, B., Teschner, M., and Gross, M. (2005). Meshless deformations based on shape matching. In: *ACM Transactions on Graphics (TOG)*, vol. 24, 3, pp. 471–478. url: http://www.beosil.com/download/MeshlessDeformations_SIG05.pdf.

Müller, M., et al. (2004). Point based animation of elastic, plastic and melting objects. In: *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 141–151. url: http://lgg.epfl.ch/publications/2004/mueller_2004_PBA.pdf.

Smith, R. (2001-2007). *Open Dynamics Engine*. url: ode.org.

Terzopoulos, D. and Fleischer, K. (1998). Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture. *Computer Graphics*, pp. 269–278. doi:10.1145/54852.378522, url: web.cs.ucla.edu/~dt/papers/siggraph88/siggraph88.pdf.