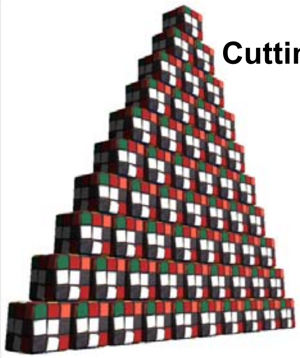


Cutting Edge Rigid Body Dynamics

Ádám Moravánszky
NovodeX AG

Overview

- rigid body dynamics in computer + video **games**
- simulation structure overview
- a selection of rarely mentioned issues and optimizations
- cutting corners and approximations
- NovodeX

Where does Physics Fit in?

layers of an interactive simulation:

- Behavior, AI — long way to go
- Physics — could be better
- Graphics, Sound — works great

state of the art:
- math + basic algos well known for a while now
- no “best” algorithm, implementation difficult
- TODO: features, robustness, performance, scalability

Isn’t Rigid Body Dynamics a solved problem?

why classic robotics research is only a start:

	Robotics	Games
Problem size	~1 robot	virtual world
Configuration	derive motion eqs for one robot	very dynamic
Mechanisms	robot created so that motion eqs are simple	anything, ev. very redundant
Constraints	primarily equality (joints)	primarily inequality (contacts)
Accuracy	simulation	visually OK

Isn't Rigid Body Dynamics a solved problem?

- a recent very relevant research paper:

S. Redon, et al. *Gauss' least constraints principle and rigid body simulations*. may 2002.

- four game middleware companies use four conceptually very different simulation approaches

- I want to knock over a house made of individual bricks in real time...

State of the Art: Boxes

Trespasser
1998, Dreamworks interactive

- universally applicable
- stress test for technology: who can make the tallest stack?
- good friction model is important

State of the Art: Cars

Carmageddon TDR 2000
2000, Torus Games

- boxes, plus:
  – suspension, steering, tires, aerodynamics, engine, gearbox, damage model

State of the Art: Cars

Carmageddon 2: Carpocalypse
1998, Stainless Software

- conflict between fun factor and realism
- design controllers that override real physics

State of the Art: Bodies

Unreal Tournament 2003
2002, Epic Games

- boxes, plus:
  - joints, complex limits, joint friction

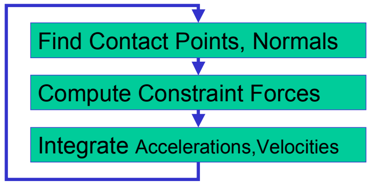
State of the Art: Bodies

Hitman 2
2002, IO Interactive

- only dead bodies for now: balancing is hard!
- augment live characters w. dynamics
  - clothes
  - secondary movement

Simulation Structure

let's write a simulator:

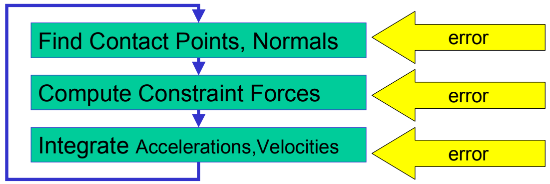


primary goal: maintain nonpenetration constraints between a group of rigid bodies

(TIP: start in 2D)

Sources of Error

why it won't work the first time:



an error introduced anywhere may prevent nonpenetration constraints from being satisfied

Collision Detection

Find Contact Points, Normals

– most Collision detection research so far has dealt with:
  - determining if bodies intersect or not
  - penetration depth of convex bodies
  - distance between bodies

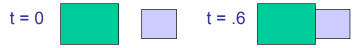
– we really need:
  - contact points and normals between eventually penetrating nonconvex bodies
  - lightweight data structures

Why Deal with Penetrations?

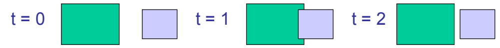
Find Contact Points, Normals

- avoid expensive rollbacks

- avoid simulation slowdown with continuous collision detection



- simulation doesn't fail when user starts / puts it in a slightly nondisjoint state
- or when forced into a nondisjoint state due to simulation error

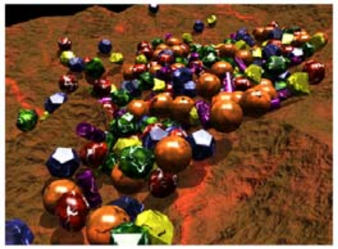


Novodex game development technology

Benefits of a Constant Time Step

Find Contact Points, Normals

- forget about subdividing time step whenever a physics event occurs
- physics events occur way too frequently
- but their effects are mostly negligible



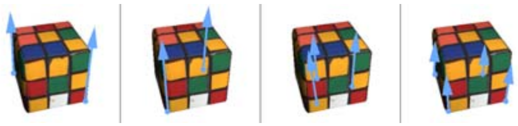
(c) MERL

Novodex game development technology

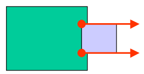
Finding Good Contacts

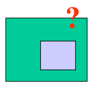
Find Contact Points, Normals

- as few contact points as possible should be able to transmit all significant interactions between a pair of bodies



- solution is not unique, and a good solution is increasingly difficult with high penetration depth





Novodex game development technology

Baraff's Contact Constraint

Compute Constraint Forces

$a_i \geq 0$ nonpenetration constraint

However, this does not uniquely determine the normal force. Additional constraints:

$f_i \geq 0$ contact force is repulsive

$f_i a_i = 0$ constraint force is workless

$(\Leftrightarrow \text{ if } (f_i) \text{then } a_i = 0; \text{ if } (a_i) \text{then } f_i = 0;)$

Stack n contacts' variables. Matrix form:

$$a = \begin{bmatrix} a_0 \\ a_1 \\ .. \\ a_n \end{bmatrix} \quad f = \begin{bmatrix} f_0 \\ f_1 \\ .. \\ f_n \end{bmatrix}$$

Novodex game development technology

Baraff's Contact Constraint

Compute Constraint Forces

$$a = \begin{bmatrix} a_0 \\ a_1 \\ .. \\ a_n \end{bmatrix} \quad f = \begin{bmatrix} f_0 \\ f_1 \\ .. \\ f_n \end{bmatrix}$$

bodies' accelerations are a linear function of the forces applied:

$a = F(f) = Af + b$

n contact problem becomes:

$f \geq 0 \quad Af + b \geq 0 \quad (Af + b)^T f = 0$

This is the definition of the Linear Complementarity Problem (LCP).

– Note: even though contacts are in 3D, we only care about motion along contact normal, so each contact provides a single row in **A**.

Novodex game development technology

LCP

Compute Constraint Forces

LCP fun facts:

- complexity between linear programming (LP) and quadratic programming (QP)
- used for economics, simulation, optimization

- NP complete in general
- fortunately our matrix **A** is PSD
- this is an example of several special cases which are not NP complete
- here solution is found after solving a short sequence of linear equality systems of size n x n

Novodex game development technology

Iterative vs. Pivoting Solvers

Compute Constraint Forces

- LCP can be solved with either:
- pivoting algos (like Gauss elimination)
  - they change the matrix
  - do not provide useful intermediate result
  - may exploit sparsity well
- iterative algos (like Conjugate Gradients)
  - only need read access to matrix
  - can stop early for approximate solution
  - faster for large matrices
  - can be warm started

Novodex game development technology

Equivalence to LCP

Compute Constraint Forces

- if you are computing contact forces to satisfy nonpenetration constraints in any way, you have written a certain kind of LCP solver
- even if you are using simple penalty methods
- because if any of below don't hold, you don't have realistic motion:
  - **f** >= 0 **a** >= 0 **fa** = 0
- if your sim is only approximate, then the LCP solution is approximate
  - for example penalty methods are usually 'bouncy'
  - (= the contacts are not quite workless)
  - So |fa| < eps

Equlvalence to LCP

Compute Constraint Forces

- does this mean we can't write a better contact force solver than what is in the LCP textbooks?
  - no:
    - matrix **A** does not have to be explicit n x n
    - **a** and **f** do not have to be stored explicitly either
    - you can work in a different space
    - you can approximate in a wide variety of ways
    - you can always come up with a transform of your inputs / outputs to a classic LCP formulation
    - if you introduce more complex constraints, for the sake of realism, you may end up with a QP or NCP problem; the LCP is a special case.

Example: Configuration Space

Compute Constraint Forces

- **A** matrix, while PSD, is in contact space:
  - O(n^2) storage for n contacts
  - not always sparse
  - ill conditioned
- it is possible to reformulate into a configuration space problem, where **f** is not expressed explicitly, and energy minimization constraint is on bodies' accelerations.
  - matrix **B**: O(n * m) storage (m = no. bodies)
  - always sparse
  - much better conditioning

Friction

Compute Constraint Forces

- physics fact: friction forces can influence normal forces and vice-versa
- ignore effect of friction on normal forces and solve sequentially for best performance
- but they have to be solved for simultaneously for best results

Joints

Compute Constraint Forces

- the joint forces of an articulated system are equality constraints:

$$a_i = 0$$
$$a = Af + b = 0$$

- solve for **f** with any linear system solver
- but special properties of A (PSD, symmetric, sparse, etc.) make a carefully chosen solver superior

MLCP

Compute Constraint Forces

- an articulated system with contact constraints results in both equality and complementarity constraints to be solved for simultaneously
- Mixed Linear Complementarity Problem

- first m rows of **A** do not have constraint on corresp terms of **f**, and =0, instead of >0
- pivoting or iterative LCP solvers can be generalized to solve MLCPs

Joint Limits and Actuators

Compute Constraint Forces

- joint limits can be modelled as contacts
- limits and contacts can be made 'soft' by adding appropriate multipliers to the constraint equation
- actuators can also be formulated as equality or inequality constraints on velocity, and thus fit into the LCP scheme too

Force vs. Impulse

Compute Constraint Forces

- instead of computing contact forces, we may compute contact impulses
- Advantages:
  - reduced integration error: Impulses integrated only 1x, while forces 2x until they influence pose
  - More control: It is OK to directly set the acceleration of objects without preventing constraints from being satisfied.
- Disadvantage:
  - accelerations not neccesarily continuous. (Not a problem in practice.)

- all algorithms work both with forces or impulses

Integration

Integrate Accelerations,Velocities

- classical way to cope with integration error is to choose higher order integrator
- must integration and contact force determination be separate?
  - if the algorithm computing the contact forces knows about they type of integration scheme used, it can anticipate its error, and compensate for it.
  - this way even fast Euler integration works great
  - Big disadvantage: external effects not formulated as constraints have severe integration error

Work at NovodeX

Questions?

- If you are already knew all this and are interested in an exciting job or internship, contact me:
  adam.moravanszky@novodex.com
  or Matthias Müller.

Friction Cones

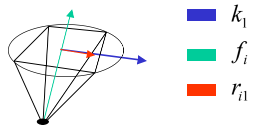
Compute Constraint Forces

- express friction as an LCP constraint:

$\mu f_i \geq r_{ic}$ Coulomb friction law

$r_{ic} \geq 0$ friction force in direction of

$r_{ic} \cdot (k_c \cdot v_t) = 0$ sliding velocity



- Note: v,f and r are interdependent, so implementing this needs a slight generalization of LCP solver

Hybrid Animation

- dynamics needs to be able to coexist with 'canned' animation, and kinematically controlled motion.
- Example: non-physical automatic door closing on box
- mostly domain specific solutions:
  - break box
  - apply an arbitrary force to the box, and stall the animation of the door while box moves away.
  - etc..

Integration

Integrate Accelerations,Velocities

- thus two common scenarios:
- 1
  - try to pack all effects (friction, actuators, limits, spring and damper elements, external forces etc.) into LCP solver as some sort of constraint
  - solve whole system together
  - don't worry much about integration
- 2
  - implement most effects as external forces on system
  - make sure you have a good integrator!

Example: Penalty Method

Compute Constraint Forces

- no matrix is stored
- **f** is a function of interpenetration **p**
- but: **p**'' = **a** so **f** = F(**a**) still...
- after **f** is determined:
  - apply forces to bodies
  - integrate forward in time
  - get new **p**
- we encoded this 'response' of the system as matrix **A**
- a good penalty method converges to a solution over time as an iterative LCP solver does