Game Physics
Assignment Sheet 5

Prof. Dr. Jan Bender, MSc. Andreas Longva

Contact: {bender, longva}@cs.rwth-aachen.de

SS 23
Deadline: **15/06/23**

Arguably, the most important functionality of a general-purpose physics engine is the ability to handle contacts (collisions, contacts at rest) between objects. For example, if we throw a box onto a floor, we expect the box to avoid falling through the floor, and moreover that it does so in a physically plausible way.

In this assignment sheet, we will first extend our Sequential Impulses implementation to account for *non-penetration constraints*, which prevent objects from penetrating each other. However, this is not sufficient for a physically plausible simulation: objects will slide forever along surfaces. In the next step, we will implement a simplified - but still realistic - friction model. Your presentation should be well-prepared and at most 10 minutes long.

You are strongly recommended to read the section on contact constraints in the *Crash course* document that accompanies this course.

Assignment 1 - Non-penetration constraints

Consider two objects that are exactly intersecting at a point $\mathbf{r}$, as illustrated in Figure 1. Let $\mathbf{r}_1$ and $\mathbf{r}_2$ be the local points corresponding to $\mathbf{r}$ in each body's local frame. In general, the *contact normal* $\mathbf{n}$ for a contact can be chosen in different ways. However, it usually corresponds to a surface normal, and especially for a vertex-face contact as in this case, it makes sense to use the surface normal of the face.

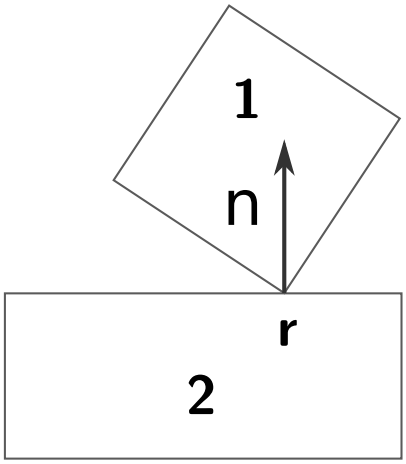


Figure 1: An illustration of two boxes in contact.

The non-penetration constraint between the two bodies can then be written

$$C_N(\mathbf{z}_1, \mathbf{z}_2) = \mathbf{n}^T(\mathbf{r}_1^w - \mathbf{r}_2^w) \geq 0.$$

The constraint measures the separation of the two points $\mathbf{r}_1^w$ and $\mathbf{r}_2^w$ *along the contact normal* $\mathbf{n}$.

a) Convince yourself that the above constraint would prevent the two bodies in Figure 1 from penetrating. What happens if a horizontal force is applied to Box 1?

b) Derive the constraint velocity map G_N for the non-penetration constraint.

Hint: How does C_N relate to the constraint function C_B for the ball joint between two bodies? What does this imply for the relationship between G_N and the constraint velocity map G_B for the ball joint?

c) Similarly, derive the changes in velocity of the two bodies $M^{-1}G^T\lambda$ given an impulse, and an analytic expression for the constraint matrix $S = GM^{-1}G^T$ (which in this case is a scalar).

The non-penetration constraint described above is alone not sufficient to model contacts between bodies. To see why, consider the following two scenarios:

- Imagine that $C_N = 10$ in Figure 1, i.e. that the two bodies are separated by a distance of 10. In this case, we can apply any attractive force represented by $\lambda_N < 0$ to pull the bodies together, as long as the constraint $C_N \geq 0$ still holds.
- Consider again the scenario in Figure 1. Assume that we apply a force $G_B^T \lambda_N$ with $\lambda_N > 0$ that maintains the constraint. Then any $\lambda_N^* \geq \lambda_N$ would also maintain the constraint, but pushing the bodies arbitrarily far away from each other.

The above examples both suggest that we must additionally constrain the constraint force itself. The first example demonstrates how the normal force may become negative: this is clearly unphysical, as we would expect normal forces to always be *repulsive*, in the sense that the bodies are always pushed apart, but never together. This implies that $\lambda_N \geq 0$.

The problem demonstrated by the second example is that the normal force becomes non-negative even as the bodies are separated. This leads us to conclude that if $C_N > 0$, then $\lambda_N = 0$. Similarly, if $\lambda_N > 0$, then $C_N = 0$. We can encode this in the *complementarity constraint* $\lambda_N \cdot C_N = 0$. The same constraint must hold at the velocity-level, i.e. $\lambda_N \cdot \dot{C} = 0$.

Unlike the ball joint, which is represented by an equality constraint, the non-penetration constraint is an *inequality* constraint (with additional constraints on the constraint force). Whereas we for the ball joint derived the Sequential Impulses method from a global perspective of all involved bodies and constraints, the resulting implications are more involved for inequality constraints, as we would have to introduce somewhat complicated concepts such as Linear Complementarity Problems (LCP). Hence we will instead only describe the necessary alterations to the algorithm to be able to deal with inequality constraints at a local level, that is contact-by-contact.

Similar to how we iterated over all ball joints in the previous exercise, we can now iterate over all *constraint manifolds*, which Bullet represents in the class `btPersistentManifold`. The manifolds are updated when calling `performDiscreteCollisionDetection()`. Each manifold contains information about interpenetrations between two rigid bodies. Each manifold may contain several contact points, represented by the class `btManifoldPoint`. The code in Figure 2 demonstrates how to fetch the list of current manifold points. How to actually iterate over manifolds and the associated rigid bodies is demonstrated by the code in Figure 3.

```
1  std::vector<btPersistentManifold *> fetchManifolds(btDynamicsWorld & world)
2  {
3      const auto num_manifolds = world.getDispatcher()->getNumManifolds();
4      std::vector<btPersistentManifold *> manifolds;
5      for (int i = 0; i < num_manifolds; ++i) {
6          manifolds.push_back(world.getDispatcher()->getManifoldByIndexInternal(i));
7      }
8      return manifolds;
9  }
```

Figure 2: Example for how to obtain the available manifolds from the dynamics world.

Algorithm 1 Example pseudo code for a possible implementation of a single step with Sequential Impulses for joints and contacts.

```
Perform collision detection
Set joint and contact target velocities
while not converged and i < max iter do
    Apply correctional impulses for joints
    Apply correctional impulses for contacts
end while
```

Contact resolution can be integrated with the existing ball joint implementation by applying correctional impulses before or after the ball joints. See Algorithm 1. As with the ball joints, one iterates over each contact and applies a corrective impulse for each contact.

```
for (size_t i = 0; i < num_manifolds; ++i) {
  auto manifold = manifolds[i];

  // The const_cast is not nice, but apparently necessary
  auto bodyA = btRigidBody::upcast(const_cast<btCollisionObject *>(manifold->getBody0()));
  auto bodyB = btRigidBody::upcast(const_cast<btCollisionObject *>(manifold->getBody1()));

  const auto is_rigidA = bodyA != nullptr;
  const auto is_rigidB = bodyB != nullptr;
  if (!is_rigidA || !is_rigidB) {
    // Only handle contact if both collision objects are rigid bodies
    // (note: even static bodies should have a rigid body)
    continue;
  }

  const auto num_contacts = manifold->getNumContacts();
  for (int c = 0; c < num_contacts; ++c) {
    auto & contact = manifold->getContactPoint(c);
    const auto n = contact.m_normalWorldOnB;
    const auto r_a = contact.m_localPointA;
    const auto r_b = contact.m_localPointB;

    // Compute and apply delta_lambda etc.
  }
}
```

Figure 3: Example for how to iterate through available manifolds.

d) A simple way to deal with the inequality constraints is the following: First compute the correctional impulse $\Delta\tilde{\lambda}$ that would yield zero constraint velocity. That is, solve $S_N\tilde{\lambda} = -G_N\mathbf{u}$. Then, apply the impulse only if $\tilde{\lambda} > 0$.

   Integrate frictionless contact resolution with your Sequential Impulses solver so that it enforces non-penetration constraints in this manner. Test your implementation on several examples, such as a sphere falling on a plane/box, or stacking of boxes.

   Hint: Contact between a single sphere and a plane or large box typically only produces a single contact point, which is a good starting point for testing your implementation.

e) So far, we do not have any mechanism for constraint stabilization, and so we will inevitably experience drift at the position level. Moreover, fast-moving objects that have visible penetration will not be restored to a penetration-less state.

   Analogous to the constraint stabilization for ball joints, we may define a *target velocity* that accounts for the violation of the position-level constraint function. Implement this type of constraint stabilization in your solver.

   NB! Remember that the constraint is only violated if $C_N < 0$.

f) The previous approach, in which we simply check whether the corrective impulse is positive, works for the simplest scenarios, but it is in general problematic for more complex scenarios.

   Set up a scenario in which boxes are stacked in such a way that the boxes do not align properly. See Figure 4. Does your solver manage to maintain the stack? If not, can you give a reason for why?

To deal with the aforementioned issues, we must note that the correct handling of the inequality constraint is to bound the *total applied impulse* to be non-negative, not the individual "delta-corrections". That is, for a single contact, we must enforce that at iteration k,

$$\tilde{\lambda}_N = \Delta\tilde{\lambda}_N^1 + \Delta\tilde{\lambda}_N^2 + \cdots + \Delta\tilde{\lambda}_N^k = \sum_{j=1}^{k} \Delta\tilde{\lambda}_N^j \geq 0.$$

Since at iteration k, we have already applied all preceding impulses, we may only adapt the currently computed delta

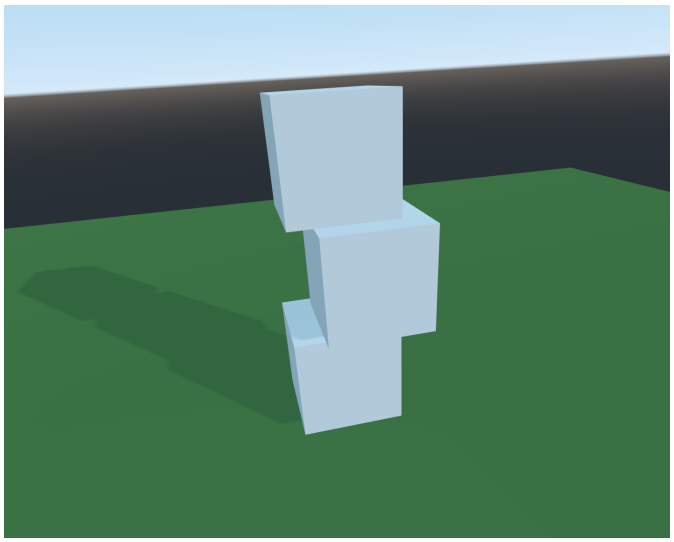


Figure 4: A set of boxes stacked in an unsymmetric manner.

impulse $\Delta\tilde{\lambda}_N^k$ so that the inequality is satisfied. That is, we must ensure that

$$\Delta\tilde{\lambda}_N^k + \sum_{j=1}^{k-1} \Delta\tilde{\lambda}_N^j \geq 0.$$

To implement this, we need to keep track of the current accumulated impulse at the current time step. Bullet has a variable `m_appliedImpulse`, part of the `btManifoldPoint` class, which you can use for this purpose. **NB!** The accumulated impulse must be reset to zero at the beginning of each time step.

g) Explain how to modify the applied impulse so that it satisfies the above inequality. Make the modification to your implementation. Does your algorithm work better for stacks of boxes?

h) Set up some test scenes and try to evaluate the strengths and weaknesses of the algorithm.

Coulomb friction

Non-penetration constraints are only one part of the puzzle; for realistic behavior of bodies in contact, we must also incorporate a plausible friction model.

In the next assignment, we will implement a simplified variant of *The Coulomb friction model*. The first assumption of the Coulomb model is that the friction force is always perpendicular to the normal force (which enforces non-penetration). If the normal force is perpendicular to the surface of a body, this means that the friction force must lie in the plane tangent to the surface, and so we often also talk about *tangential forces*. In particular, we can pick basis unit vectors $\mathbf{t}_1, \mathbf{t}_2 \in \mathbb{R}^3$ for this tangent plane by requiring that both $\mathbf{t}_1$ and $\mathbf{t}_2$ are perpendicular to the normal vector $\mathbf{n}$. Typically we also pick an orthonormal basis, so we let $\mathbf{t}_1$ and $\mathbf{t}_2$ also be orthogonal. We may now express our normal force $\mathbf{F}_N \in \mathbb{R}^3$ and friction force $\mathbf{F}_T \in \mathbb{R}^3$ as

$$\mathbf{F}_N = \lambda_N \mathbf{n}, \qquad \mathbf{F}_T = \alpha_1 \mathbf{t}_1 + \alpha_2 \mathbf{t}_2.$$

Note that we define the normal and friction forces as the forces that *act on the first body, from the second*. From Newton's third law, it follows that the force acting on the second from the first is $-\mathbf{F}_N$ for the normal force, and $-\mathbf{F}_T$ for the friction force.

A central modeling assumption in the Coulomb friction law is that the friction force must be bounded proportional in magnitude to the normal force. That is, we may write

$$\|\mathbf{F}_T\| \leq \mu \|\mathbf{F}_N\|. \tag{1}$$

for some coefficient $\mu \geq 0$, the *coefficient of friction*. Typically $\mu \in [0, 1]$. Moreover, it may depend on the relative velocity of the two objects. The above inequality (1) is often termed the *Coulomb friction cone*, since any admissible $\mathbf{F}_T$ is contained in a cone with its apex at the origin. This is illustrated in Figure 5a. Denote by $\mathbf{v}_T \in \mathbb{R}^3$ the relative velocity *in the tangent plane* at the point of contact, i.e. the relative velocity $\dot{C}_B$ projected onto the tangent plane. Coulomb's law of friction that distinguishes between two cases:

- Static friction: If $\mathbf{v}_T = 0$, then $\|\mathbf{F}_T\| \leq \mu \|\mathbf{F}_N\|$.
- Kinetic friction: If $\mathbf{v}_T \neq 0$, then$\|\mathbf{F}_T\| = \mu \|\mathbf{F}_N\|$, and $\mathbf{F}_T = -\mu \|\mathbf{F}_N\| \frac{\mathbf{v}_T}{\|\mathbf{v}_T\|}$. That is, the friction force attains its maximum magnitude, and its direction is the negative of that of the relative tangent velocity.

By writing

$$\boldsymbol{\lambda}_T := \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix},$$

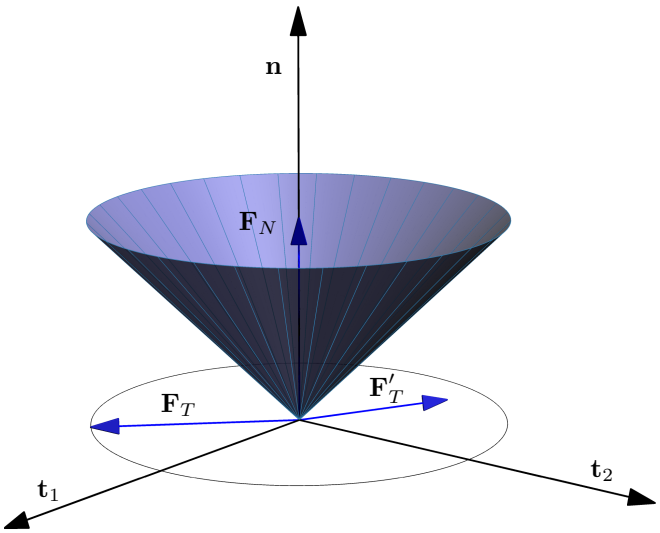
we can reformulate the friction cone (1) in terms of the variables λ_N and $\boldsymbol{\lambda}_T$ by noting that

$$\begin{aligned} \|\mathbf{F}_T\|^2 &= \|\alpha_1 \mathbf{t}_1 + \alpha_2 \mathbf{t}_2\|^2 = \alpha_1^2 \overbrace{\mathbf{t}_1 \cdot \mathbf{t}_1}^{1} + 2\alpha_1 \alpha_2 \overbrace{\mathbf{t}_1 \cdot \mathbf{t}_2}^{0} + \alpha_2^2 \overbrace{\mathbf{t}_2 \cdot \mathbf{t}_2}^{1} = \alpha_1^2 + \alpha_2^2 = \|\boldsymbol{\lambda}_T\|^2, \\ \|\mathbf{F}_N\|^2 &= \|\lambda_N \mathbf{n}\|^2 = \lambda_N^2 \mathbf{n} \cdot \mathbf{n} = \lambda_N^2. \end{aligned}$$

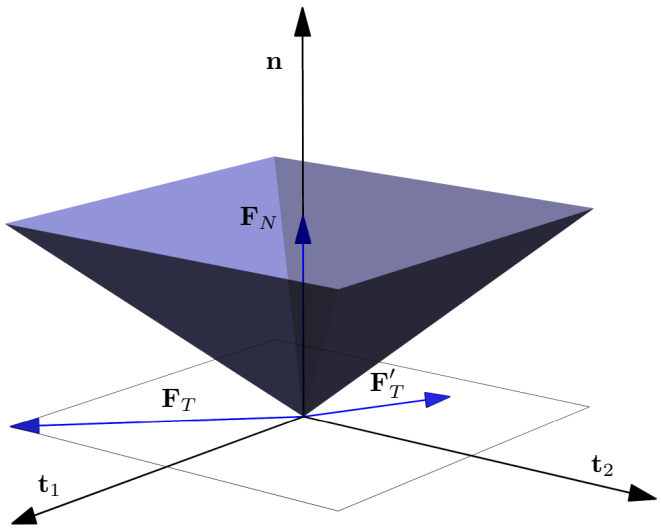
Since we assume that $\lambda_N \geq 0$, we may reformulate the friction cone (1) as

$$\|\boldsymbol{\lambda}_T\| \leq \mu \lambda_N. \tag{2}$$

Enforcing the exact friction cone is difficult. We will later see how we can simplify it by a so-called *linearized* friction cone.



(a) An illustration of the friction cone for $\mu = 1$. The tangential forces $\mathbf{F}_T$ and $\mathbf{F}'_T$ displayed here are only examples of admissible tangential forces. Any vector within the circle projected onto the plane spanned by $\mathbf{t}_1$ and $\mathbf{t}_2$ is a permissible tangential force.



(b) An illustration of a possible linearization of the friction cone, with $\mu = 1$. The tangential forces $\mathbf{F}_T$ and $\mathbf{F}'_T$ displayed here are only examples of admissible tangential forces. Any vector within the square projected onto the plane spanned by $\mathbf{t}_1$ and $\mathbf{t}_2$ is a permissible tangential force.

Figure 5: Exact and linearized (pyramidal) friction cones.

There is no position-level constraint associated with Coulomb friction. We can also not fit the friction model directly into the framework of velocity-level constraints that we have developed so far. However, with minor modifications we can accommodate a simplified friction model in our Sequential Impulses method in a way similar to how we handled non-penetration constraints. To do this, we need a velocity map G_T, which maps (generalized) velocities of the two bodies involved into velocities along the two directions $\mathbf{t}_1$ and $\mathbf{t}_2$. This is thankfully straightforward: we simply project

the relative velocity of the two bodies in the contact point, which is simply the constraint velocity of a ball joint constraint $\dot{C}_B$ at that point, onto each axis $\mathbf{t}_1$ and $\mathbf{t}_2$:

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} := G_T \mathbf{u} = \begin{pmatrix} \mathbf{t}_1 \cdot \dot{C}_B \\ \mathbf{t}_2 \cdot \dot{C}_B \end{pmatrix} = \begin{pmatrix} \mathbf{t}_1^T G_B \mathbf{u} \\ \mathbf{t}_2^T G_B \mathbf{u} \end{pmatrix} = \begin{pmatrix} \mathbf{t}_1^T \\ \mathbf{t}_2^T \end{pmatrix} G_B \mathbf{u}.$$

Here, w_1 corresponds to the relative velocity along the $\mathbf{t}_1$ axis, and w_2 corresponds to the relative velocity along the $\mathbf{t}_2$ axis, and G_B is the constraint velocity map associated with the ball joint. Note that this is consistent with the constraint velocity map for the normal force, which projects the relative velocity onto the normal direction.

Assignment 2 - Sequential Impulses with friction

a) Recall that associated with any force acting in a point $\mathbf{p}$ is a torque $\boldsymbol{\tau} = \mathbf{r} \times \mathbf{F}$, where $\mathbf{r} = \mathbf{p} - \mathbf{X}$ is the *arm* and $\mathbf{F}$ is the force.

Show that

$$\mathbf{f}_T := \begin{pmatrix} \mathbf{F}_{T,i} \\ \boldsymbol{\tau}_{T,i} \\ \mathbf{F}_{T,j} \\ \boldsymbol{\tau}_{T,j} \end{pmatrix} = G_T^T \boldsymbol{\lambda}_T,$$

where $\mathbf{F}_{T,i} = \alpha_1 \mathbf{t}_1 + \alpha_2 \mathbf{t}_2$ is the friction force on body i, $\boldsymbol{\tau}_{T,i}$ is the friction torque on body i and $\mathbf{f}_T$ is the generalized friction force.

Hint: Recall Newton's third law and compute the friction forces and torques in two ways: first by considering the torque corresponding to the friction force on each body, and second by evaluating the transpose of the constraint velocity map.

We will begin with a simple friction model in which we only consider friction along one axis. We choose this axis to be $\mathbf{t}_1 = \frac{\mathbf{v}_T}{\|\mathbf{v}_T\|}$, so that the friction force is parallel to the relative tangential velocity. In that case, the (reformulated) friction cone constraint (2) becomes a simple scalar constraint:

$$-\mu\lambda_N \leq \alpha_1 \leq \mu\lambda_N, \tag{3}$$

and the friction force is given by $\mathbf{F}_T = \alpha_1 \mathbf{t}_1$.

In order to integrate friction handling with the Sequential Impulses method, we first apply all normal impulses, then apply all tangential impulses for each iteration of the constraint solver. This is illustrated as pseudocode in Algorithm 2.

Algorithm 2 Pseudo code for a possible time step with frictional contacts.

```
Perform collision detection, initialize joints, contacts
Reset applied impulses α1 (and α2) and compute tangent vectors t1 (and t2) for each contact
while not converged and i < max iter do
    Apply correctional impulses for joints
    Apply correctional normal impulses
    Apply correctional friction impulses
end while
```

b) The way we determine and apply correctional tangential impulses is in practice very similar to the implementation of the normal impulses. First, solve for the impulse $\Delta\alpha_1$ along $\mathbf{t}_1$ that cancels velocity in the direction of $\mathbf{t}_1$, i.e. such that $S_1 \Delta\alpha_1 = -G_1 \mathbf{u}$, where $G_1 = \mathbf{t}_1^T G_B$ is the first row of G_T defined above, and $S_1 = G_1 M^{-1} G_1^T$. Then adapt $\Delta\alpha_1$ such that α_1 (the *accumulated* friction impulse) satisfies (3).

In `Bullet` terms, $\mathbf{t}_1$ can be stored in the variable `btManifoldPoint::m_lateralFrictionDir1`, and the accumulated impulse α_1 can be accumulated in the variable `btManifoldPoint::m_appliedImpulseLateral1`. For the constant μ, you can use the value of `btManifoldPoint::m_combinedFriction`, which is set by Bullet.

Derive analytic expressions for S_1 and the velocity update $M^{-1} G_1^T \alpha_1$. Implement the proposed method, and test

your implementation with examples involving simple sliding friction, as well as static scenes such as a stack of boxes. With the current implementation, you should also be able to roll a ball on a flat plane.

c) Whereas our previous simplified friction model generally does a good job of simulating dynamic friction, it does not in general handle *static* friction very well. Can you explain why?

To better handle static friction, we need a friction model that takes into account more than one direction. The simplest possible choice is the *pyramidal* friction cone, or the *box friction model*, in which we approximate the friction cone by two linear constraints, one for each direction $\mathbf{t}_1$ and $\mathbf{t}_2$:

$$-\mu\lambda_N \leq \alpha_1 \leq \mu\lambda_N, \tag{4}$$

$$-\mu\lambda_N \leq \alpha_2 \leq \mu\lambda_N. \tag{5}$$

This type of linearization of the friction cone is illustrated in Figure 5b. Because $\mathbf{t}_1$ and $\mathbf{t}_2$ are orthogonal by choice, the two variables α_1 and α_2 can be independently solved for.

d) Extend your implementation from the previous assignment to also account for friction in the second direction $\mathbf{t}_2$. The variables $\mathbf{t}_2$ and α_2 can be stored in `btManifoldPoint::m_lateralFrictionDir2` and `btManifoldPoint::m_appliedImpulseLateral2`, respectively.

Test your implementation with scenarios for which static friction is necessary. As an example, imagine throwing a ball at a stationary stack of boxes.

e) Once your contact resolution works well for simple problems, try to benchmark your contact handling resolution by constructing tough scenarios. For example, how many boxes can you stack reliably without the boxes falling down? Or what happens if you have interaction between light and heavy objects?

f) Finally, try to run your maze game with your new contact resolution method. Does it work as well as the built-in physics engine or with the Bullet physics engine?