Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

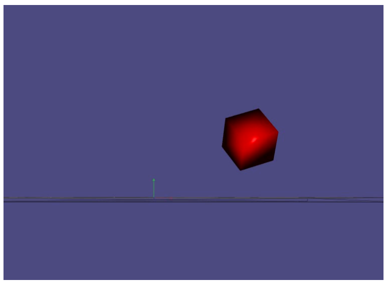
computer graphics laboratory
Prof. Markus Gross

Visual Computing
Exercise 11: Rigid Body Dynamics

Hand-out: 13. December 2022

Goals

- Understand the physical basics of rigid body dynamics
- Learn to implement a simple rigid body simulation



Resources

The lecture slides and exercise slides are accessible via the course web page http://graphics.ethz.ch (Teaching → Visual Computing)

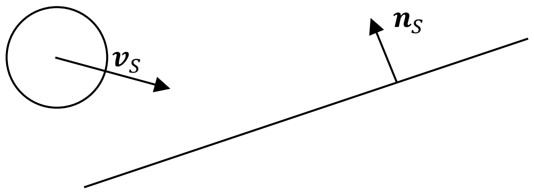
For an introduction on rigid body simulation, refer to the tutorial by David Baraff on rigid body simulation at http://graphics.cs.cmu.edu/courses/15-869-F08/lec/14/notesg.pdf .

Exercise 1) Rigid Body Dynamics Basics

In this exercise you have to answer some questions about the physical basics of rigid body dynamics.

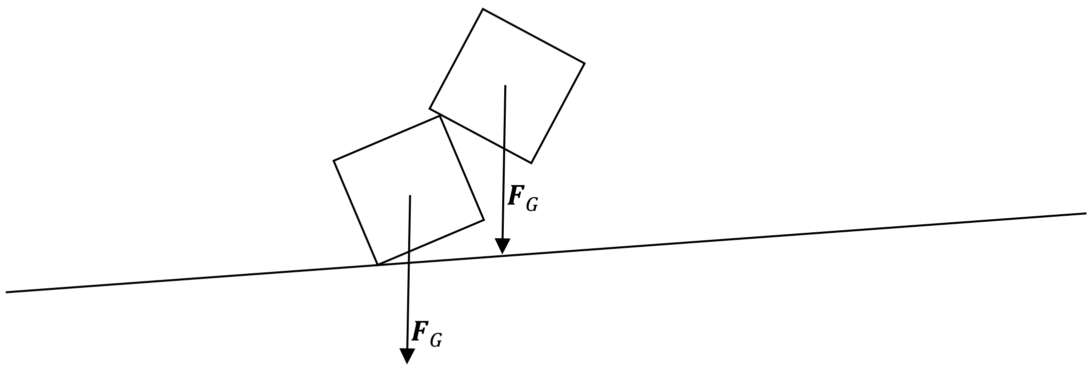
a) Simple collision

A sphere is moving towards a fixed plane. Calculate the linear momentum before and after the collision with the plane. Assume elastic collisions (momenta are conserved) and no external forces. The sphere has a mass of $m_S = 100kg$ and a velocity of $\boldsymbol{v}_S = \begin{pmatrix} 5 \\ -1 \\ 0 \end{pmatrix} \frac{m}{s}$ the plane's normal is $\boldsymbol{n}_P = \frac{1}{5}\begin{pmatrix} -3 \\ 4 \\ 0 \end{pmatrix}$.



b) Force and Torque

The picture below shows two cubes that are in contact. Both cubes have exactly the same mass $\boldsymbol{m}_C = 1kg$ and the velocity of both cubes is $\boldsymbol{v}_C = \boldsymbol{0}$. $\boldsymbol{F}_G$ denotes the gravitational force. Fill in any missing forces and torques that act on the bodies. Make sure that the direction and magnitudes of the forces are approximately correct. For the torques you may use curved arrows to indicate the direction.



c) Handling collisions

There are two common ways to update the states of rigid bodies after collisions. Either you adjust the state of the bodies after collisions by updating forces that act on the bodies or by updating momenta of the involved bodies. What are the advantages or disadvantages of each method?

d) Quaternions for rotations

When dealing with rotations, we rely on quaternions. What are the advantages of using quaternions compared to rotation matrices? Are there any rotations that cannot be represented by rotation matrices that can be represented by quaternions or vice versa?

Exercise 2) Implement a simple scene with rigid body dynamics

In this exercise you are going to implement the simulation of a simple scene, in which a cube is thrown onto a plane, with rigid body dynamics.

a) Implement a solver for the ODE of rigid body dynamics

To update the state of all objects an ODE has to be solved. A solver for an ODE can be implemented in a function

```
void RigidBodySim::advance()
```

This function will update the states of all the rigid objects for each time step, where the linear momentum, and angular momentum will be first integrated with the explicit Euler method, based on which, the linear velocity, angular velocity, center of mass, and rotation will be updated subsequently. Specifically, you should first get familiar with the simulation loop we will discuss in the course. Then, take a closer look at the class `Simulation`. One of the most important attributes of it is `m_objects`, which contains all the rigid objects described using class `RigidObject`, where the related helper functions are already prepared. After reading these classes, you are able to understand the class `RigidBodySim` derived from `Simulation`, and fill the missing code in the function `RigidBodySim::advance` according to my instructions. In this task, you need to write three small parts. The first one is to integrate velocities using the helper functions `RigidObject::setLinearMomentum` and `RigidObject::setAngularMomentum`. The second one is to integrate mass center using the function `RigidObject::setPosition`. The last one is to integrate rotation information with matrix-based and quaternion-based methods.

b) Implement collision detection and handling

At the beginning of simulation, collision detection and handling are required to avoid any possible penetration. In this task, you need to understand a very simplistic collision detection between the cube and the plane in `CollisionHandler::handleCollision()`. To simplify the collision detection, only consider collisions between the vertices of the simulated rigid body and the plane. Then provide the missing parts to judge whether the collision is happening and update the momentum of the cube after the collision. There is a very good UI you can play with by changing the simulation parameters, such as `eps`.