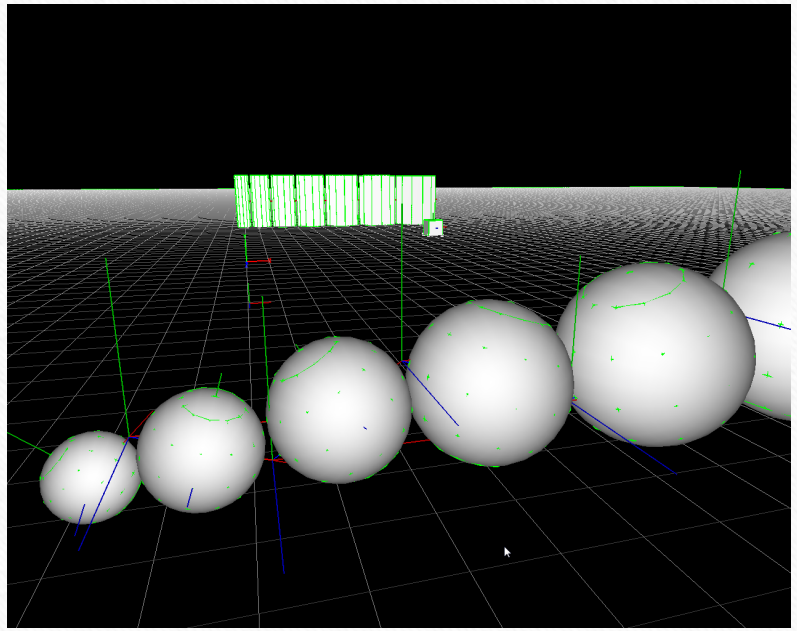
Physics II

CITM

Bullet Physics – Bullet Constraints

Some new code!

- You can find some new code that will allow you to:
  - RayCast and check for collisions with the ray.
  - Cast a ray from the “mouse” position.

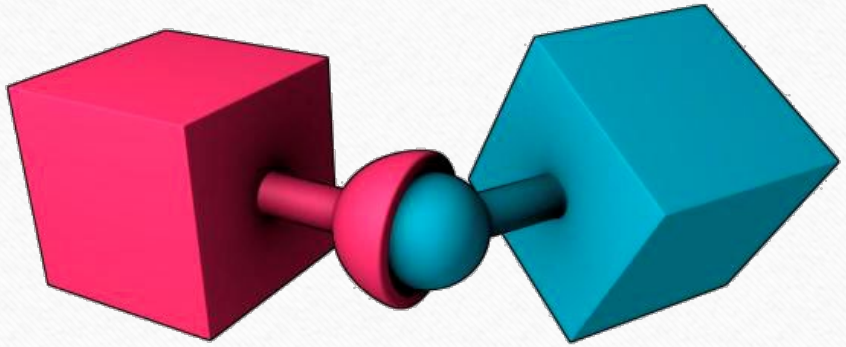


Bullet Constraints

- Point to Point (Distance joint)
- Hinge (Revolute joint)
- Slider (Prismatic joint)
- Cone Twist
- Six Degrees-of-freedom

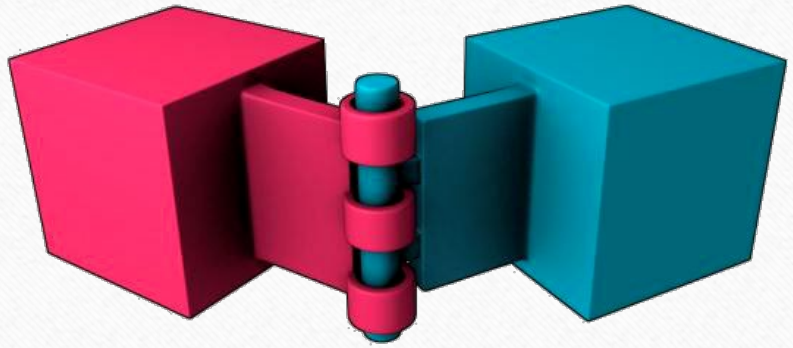
Point to point (P2P)

- The Point constraint (called a point-to-point constraint in the Bullet Physics library) limits the translation so that pivot points between the two rigid bodies match in world space. You can use the Point constraint to create effects, such as a chain-link, or to pin objects together.



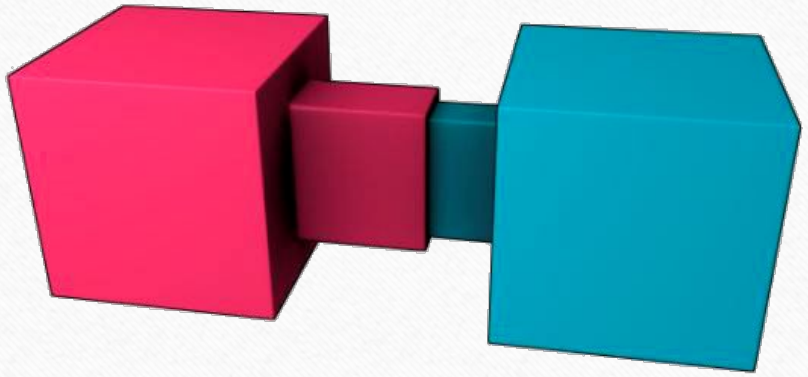
Hinge

- The Hinge constraint restricts the translation and two additional angular degrees of freedom, so the body can only rotate around one axis. The hinge axis is defined by the Z axis of the constraint. This constraint is useful for representing doors or wheels rotating around an axis. The user can specify limits and motor settings for the hinge.



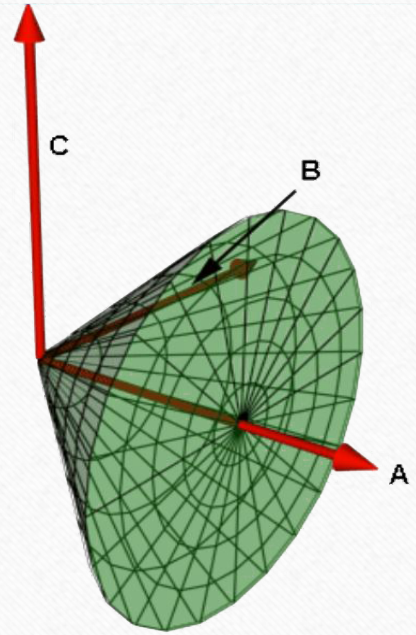
Slider

- The Slider constraint allows rigid bodies to rotate around one axis and translate along the same axis. The slide axis is defined by the Z axis of the constraint.



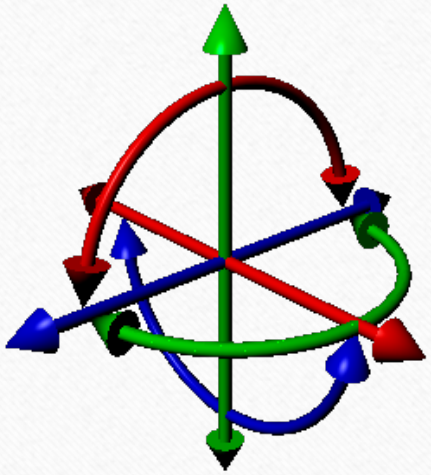
Cone twist

- For ragdolls, the Cone-Twist constraint is useful for limbs like the upper arm. It is a special point-to-point constraint that adds cone and twist axis limits. The X axis serves as twist axis.



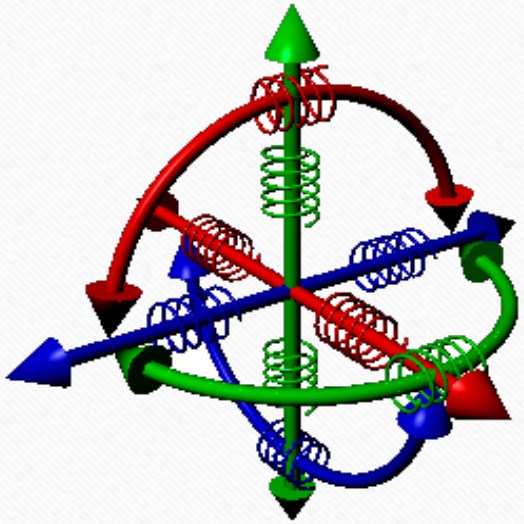
Six Degrees-of-freedom

- The Six Degrees-Of-Freedom (SixDOF) constraint can emulate a variety of standard constraints if each of the six Degrees of Freedom (DOF) is configured. The first 3 DOFs axis are linear axis, which represent the translation of rigid bodies, while the latter 3 DOFs axis represent the angular motion. Each axis can be locked, free, or limited. By default, all axes are unlocked.

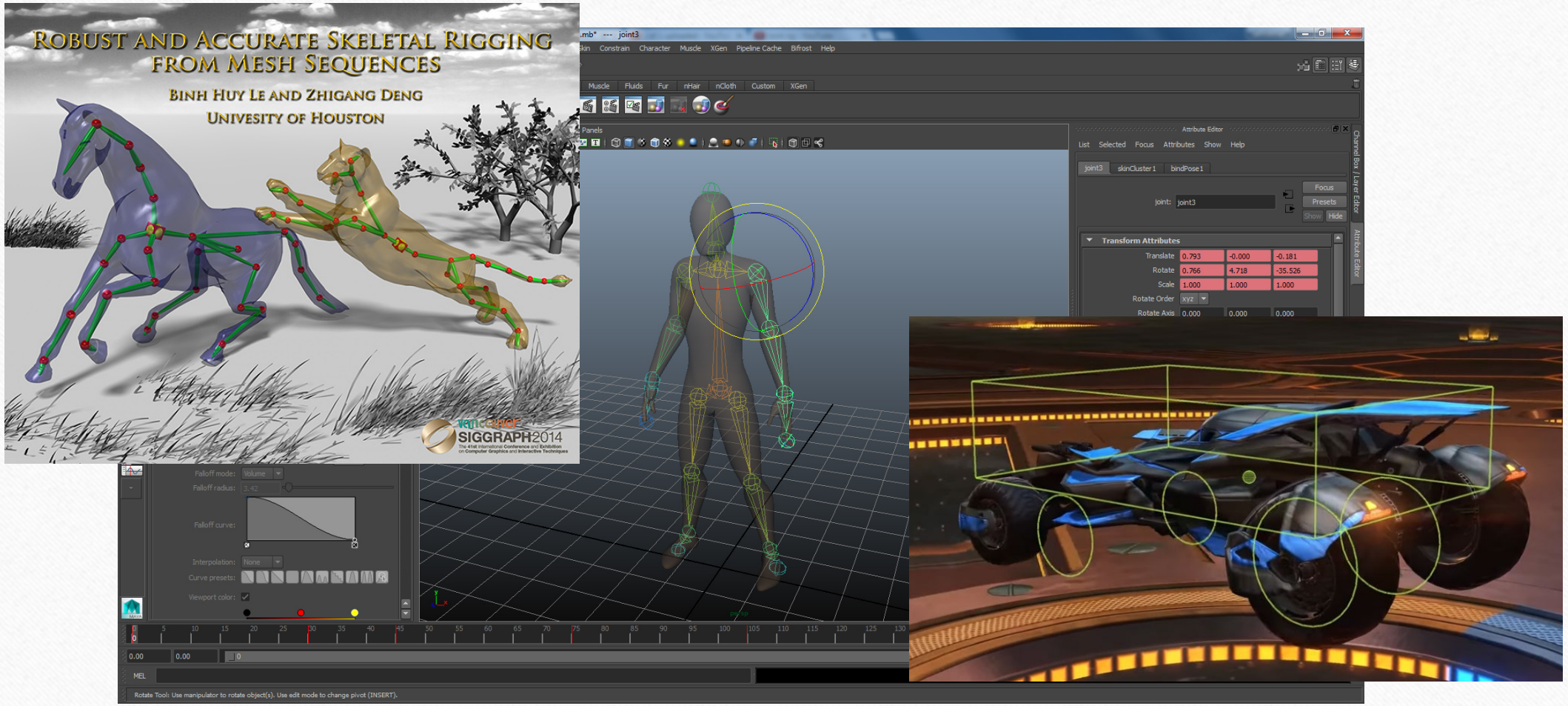


And other variants

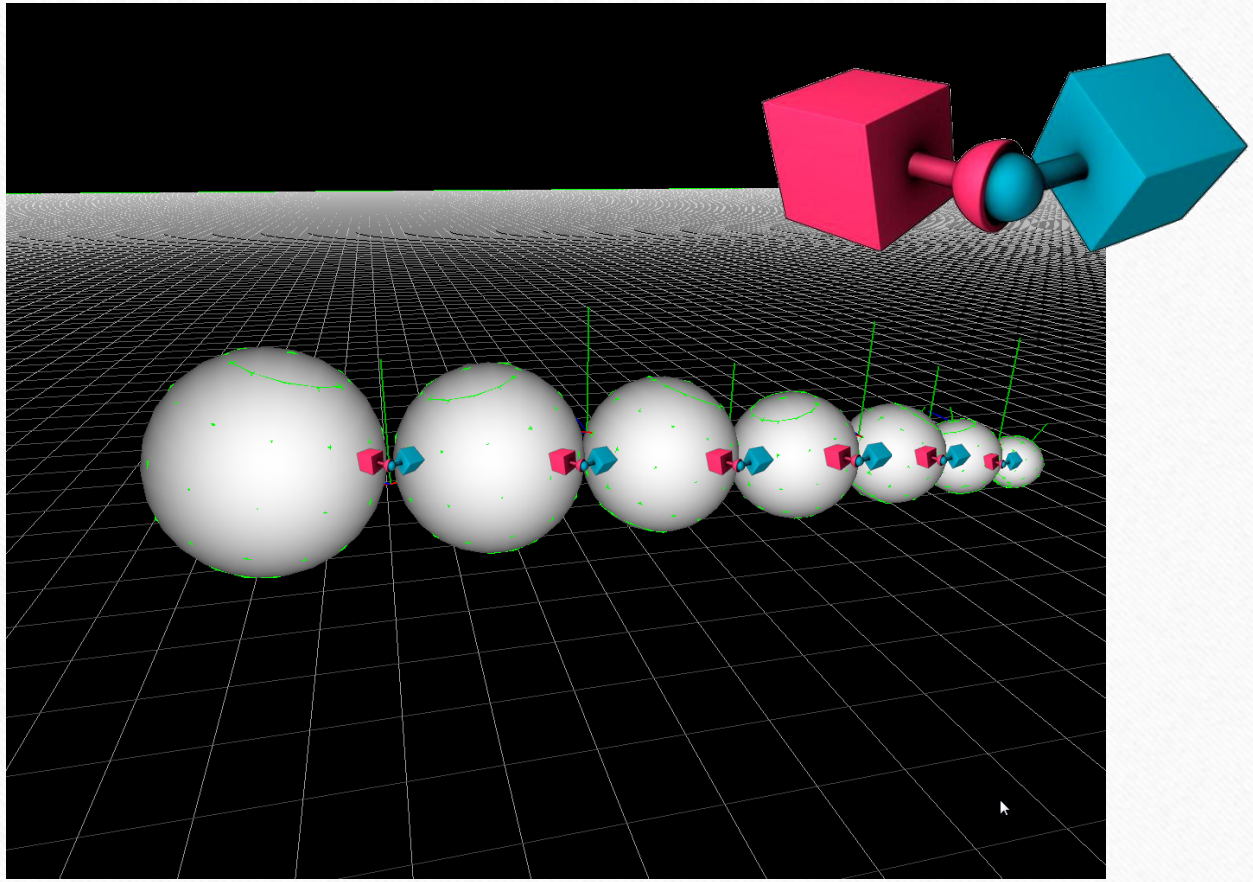
- Spring variants
- Modified variants (Limiting, concatenating)
- Motor constraints



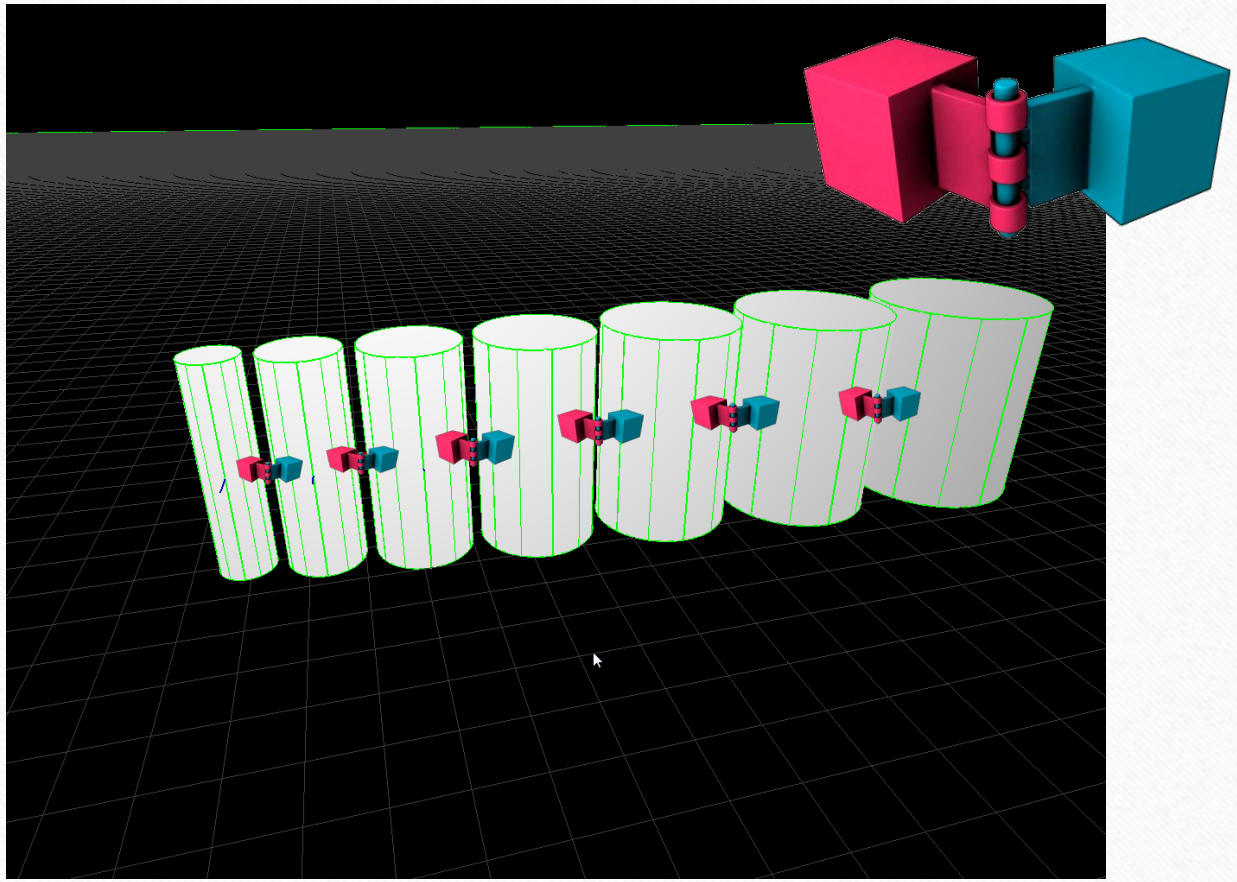
Complex joint systems



Our first goal



Our second goal



TODO 1

Implement the code to add a Point to Point constraint (btPoint2PointConstraint)

void AddConstraintP2P(const Primitive& bodyA, const Primitive& bodyB, ...);

- Implement a method to link two PhysBodies or Two primitives with a P2P.
- You'll need to create a btPoint2PointConstraint:

```
btPoint2PointConstraint(btRigidBody& rbA, btRigidBody& rbB, const btVector3& pivotInA, const btVector3& pivotInB);
```

- And then, add it to the world with: world->addConstraint( ).
- Remember to delete the "new" constraints in the CleanUp( ).

TODO 2

Link all the spheres with your P2P constraints

- Here you have some code spawning a few Spheres. Link them together with your newly created P2P constraints!

- Both pivots are in local space for each of the bodies.

TODO 3

Implement the code to add a Hinge constraint (btHingeConstraint)

void AddConstraintHinge(const Primitive & bodyA, const Primitive & bodyB, ...);

- Implement a method to link two PhysBodies or Two primitives with a Hinge.
- You'll need to create a btHingeConstraint:

```
btHingeConstraint(btRigidBody& rbA,btRigidBody& rbB, const btVector3& pivotInA, const btVector3& pivotInB,
btVector3& axisInA, btVector3& axisInB);
```

- And then, add it to the world with: world->addConstraint( ).
- Remember to delete the "new" constraints in the CleanUp( ).

TODO 4

Link some other spheres with your Hinge constraint

- Experiment with different axis. They should behave like a bike chain.
- Remember! Axis and anchor points are in the LOCAL SPACE of the rigidBodies.

Extras

- Test limiting the constraints!
- Add a motor to some of them, see how they react.
- Try controlling the motor with the keyboard.

- Advanced: Create a “mouse joint”, that allows us to grab and move bodies with the mouse.