Physics 2 - Joints

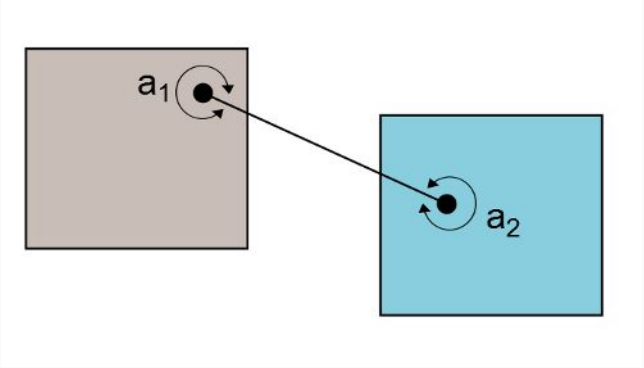
Ricard Pillosu - UPC

What is a Joint ?

- Joints are used to constrain bodies to the world or to each other.
- Each joint needs 2 bodies. One of them can be static.
- If both are static or kinematic, they are ignored.
- Those bodies won’t collide to each other by default (can be changed).
- If a body is destroyed, all joints attached are destroyed automatically!

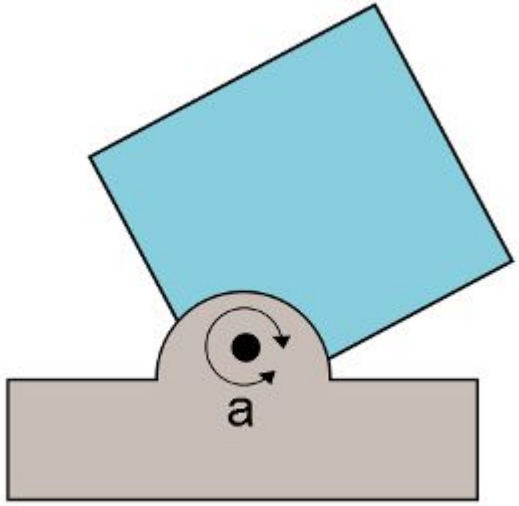
Distance Joint

- Keeps a constant distance between bodies
- Anchor points are at the center by default
- Can be made into a spring using:
  - Frequency in Hz (less than world step / 2)
  - Damping Ratio (0.0 … 1.0 where 1.0 is no damping)
- Live demo here



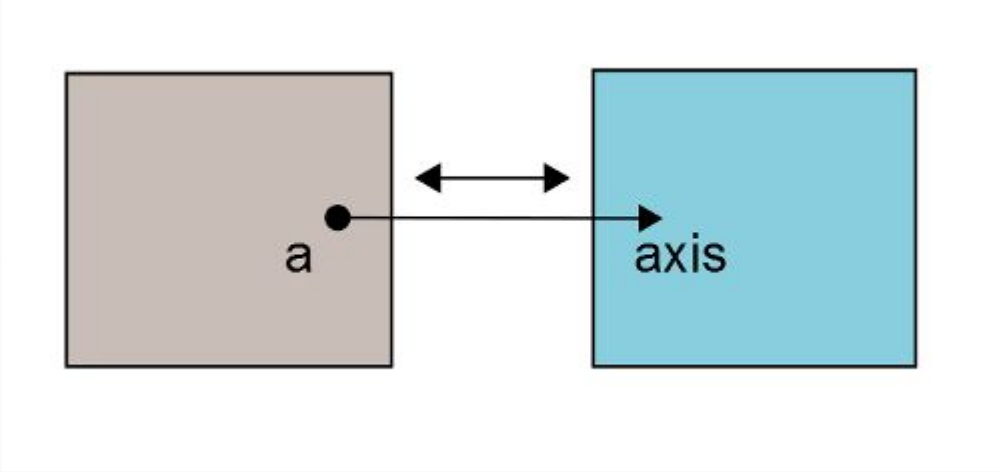
Revolute Joint

- Both bodies share the same anchor point
- Accepts min / max angle limits
- Can use motors to produce torque
- Live demo here and with motor here



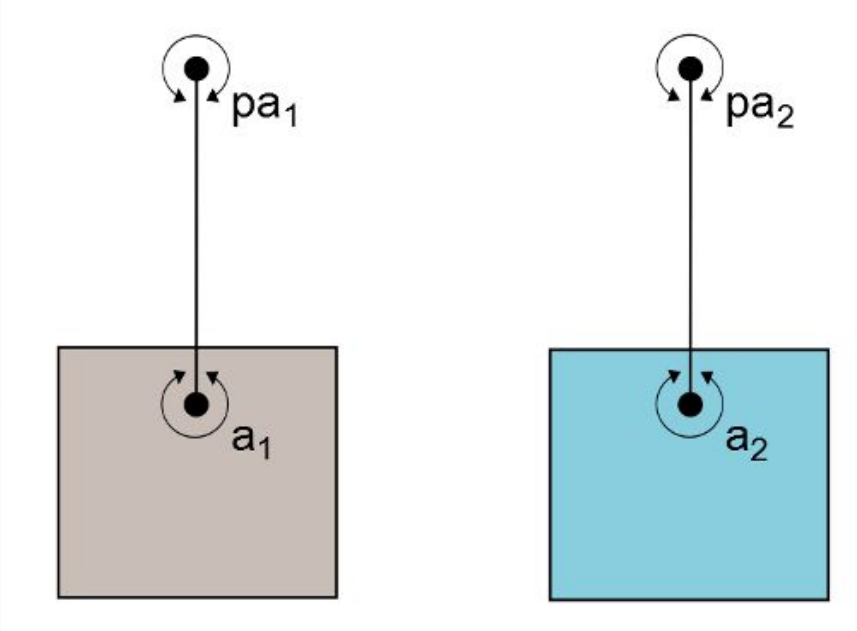
Prismatic Joint (slider joint)

- Same as revolute joint but with translation
- Allows on degree of freedom
- Accepts motors
- Could simulate pistons
- Live test here

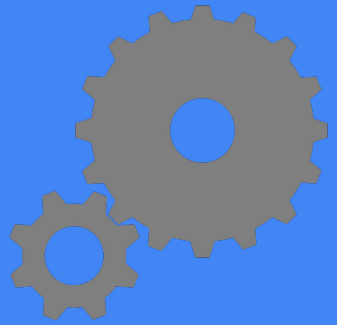


Pulley Joint

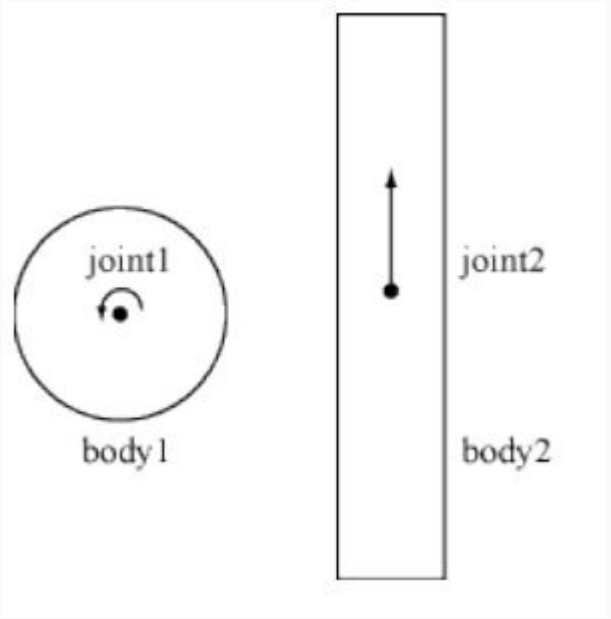
- Simulate a two bodies with a single pulley
- As one body goes up, the other goes down
- Accepts ratios



Gear Joint

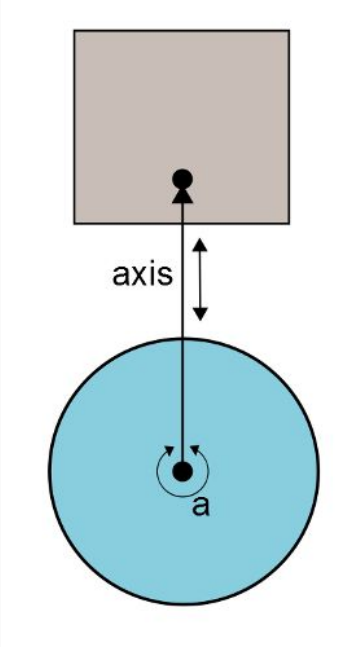


- Simulate a gear :)
- They use other revolute or prismatic joint
- Accept motors
- Dangerous destruction process!
- Live demo here



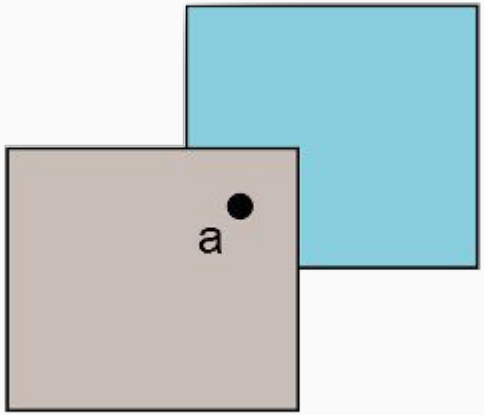
Wheel Joint (previously Line joint)

- Used to simulate car wheels
- Prismatic joint but allows rotation
- Allows rotation and one degree of freedom
- Accepts spring options to simulate suspension
- Accepts a motor



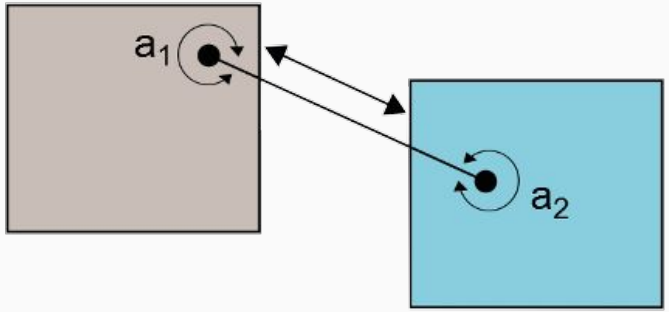
Weld Joint

- Constraints all relative motion
- Similar to having several shapes per body
- But accepts spring parameters!
- Not recommended for exploding objects
- Live demo here



Rope Joint

- Like distance joint but has no min distance
- Useful to prevent stretching
- Live demo here



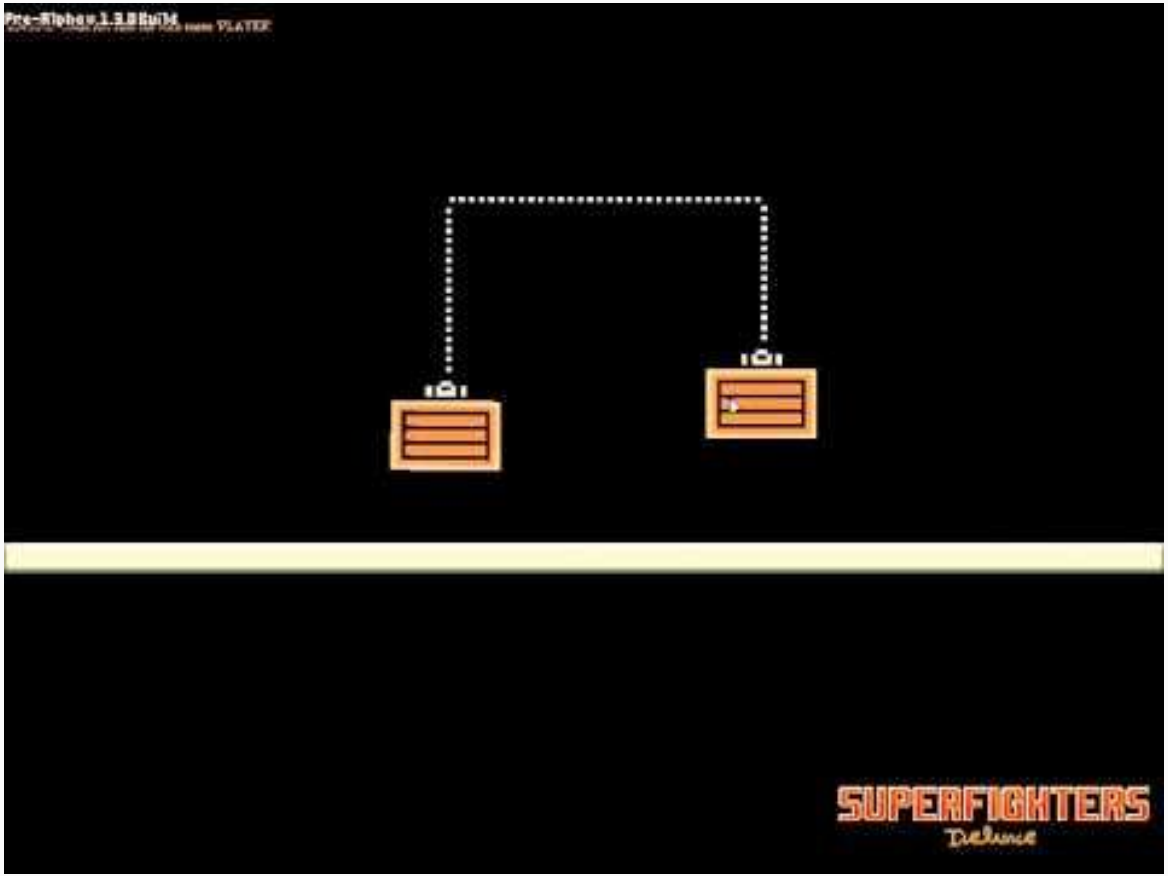
Friction Joint

- Attempts to drive the relative motion between the bodies to zero
- The maximum force and torque members are used to limit the rate at which the motion is driven to zero
- Live demo here

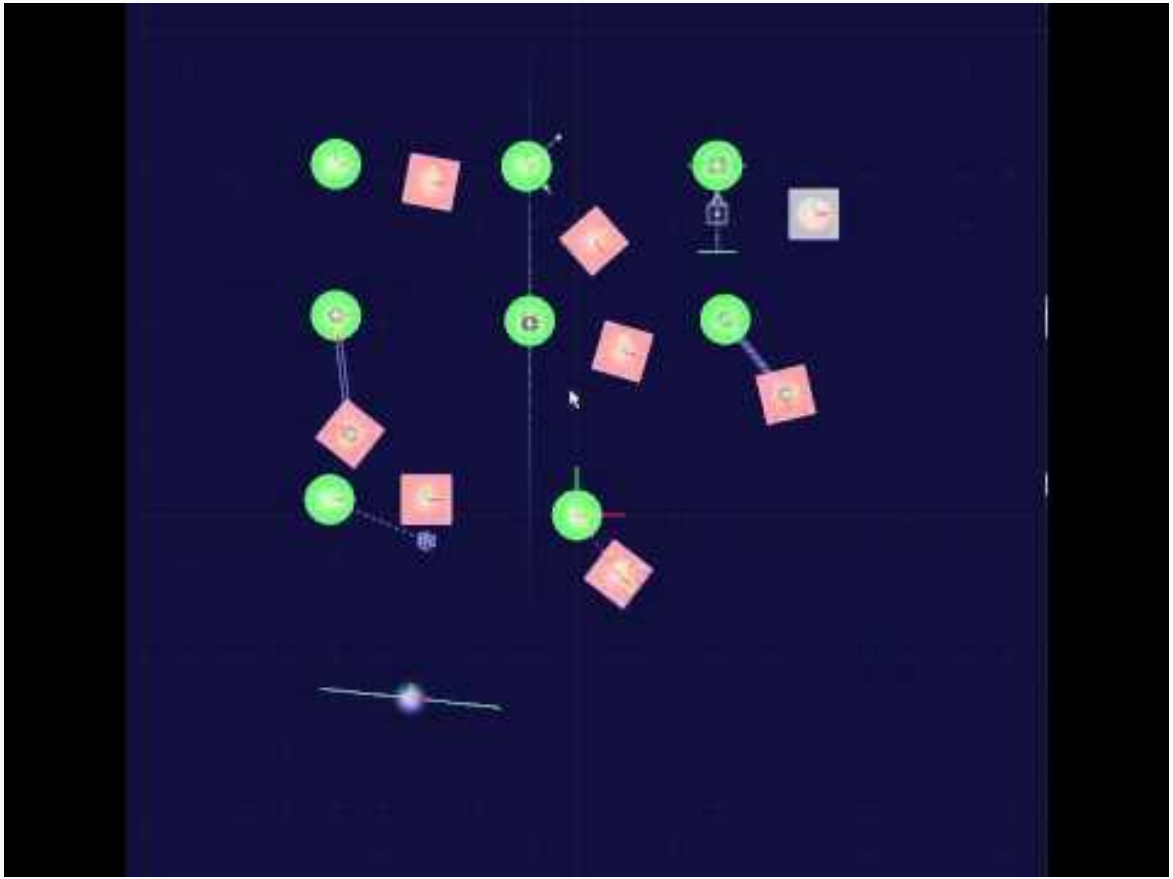
Motor Joint

- Simply pushes a *bodyA* to *bodyB* position and rotation
- Could simulate a magnet
- We can define maxForce and maxTorque allowed
- Live demo here

Try to identify the joints used in this video



Try to identify the joints used in this video



Mouse Joint

- Pulls objects to a position (normally for debugging purposes)
- Let’s code it (first check solution.exe):
  - We will use the large loop to draw all shapes to test all bodies
  - If we find we clicked on one, we create a mouse joint
  - While mouse is pressed, we update the joint target position
  - Remove the joint when the player stops pressing mouse button

TODO 1

“If mouse button 1 is pressed, test if the current body contains mouse position”

- Remember to convert mouse position from pixels to meters
- If you find one body you can skip the rest of the checks
- In the end you have a pointer to a body that was clicked (or NULL)

TODO 2

“TODO 2: If a body was selected, create a mouse joint”

```
b2MouseJointDef def;
def.bodyA = ground;
def.bodyB = body_clicked;
def.target = mouse_position;
def.dampingRatio = 0.5f;
def.frequencyHz = 2.0f;
def.maxForce = 100.0f * body_clicked->GetMass();

mouse_joint = (b2MouseJoint*) world->CreateJoint(&def);
```

TODO 3

“If the player keeps pressing the mouse button, update target position and draw a red line between both anchor points”

- You can use *App->render->DrawLine* method
- Remember to convert from pixels to meters!

TODO 4

“If the player releases the mouse button, destroy the joint”

- Remember to reset mouse_joint pointer to NULL!

Homework

- Experiment substituting the mouse joint with a distance joint
- Try having the big circle in the middle to rotate:
  - Which joint will you use ?
  - Do we need to use a motor ?