How Do Physics Engines Work?

Erin Catto
Principal Engineer
Blizzard Entertainment

Google Says

https://en.wikipedia.org/wiki/Game_physics — might teach you game physics

https://www.reddit.com/r/GamePhysics/ — might entertain you https://www.reddit.com/r/GamePhysics/comments/7yic11/game_like_it_or_not_this_is_what_real_fighting/

https://www.reddit.com/r/GamePhysics/comments/8xqbds/unity_im_working_on_a_system_for_picking_up/

Game Physics

Makes the Best Bugs

Game Physics

Is a HUGE Topic

High level physics

- Physics engines
- Game engine integration
- Game design
- Tuning

Low level physics

- Collision queries
- Rigid body simulation
- Soft body simulation (cloth)
- Character movement
- Vehicle simulation
- And so on …

Physics engines

- Havok
- Box2D
- Bullet
- Nvidia PhysX
- Custom

Custom
Physics Engine

Game engine integration

UNREAL ENGINE

unity

löve

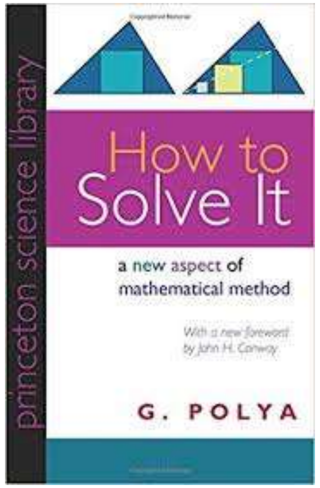
Custom Game Engine

Game Physics

Is a Difficult Topic

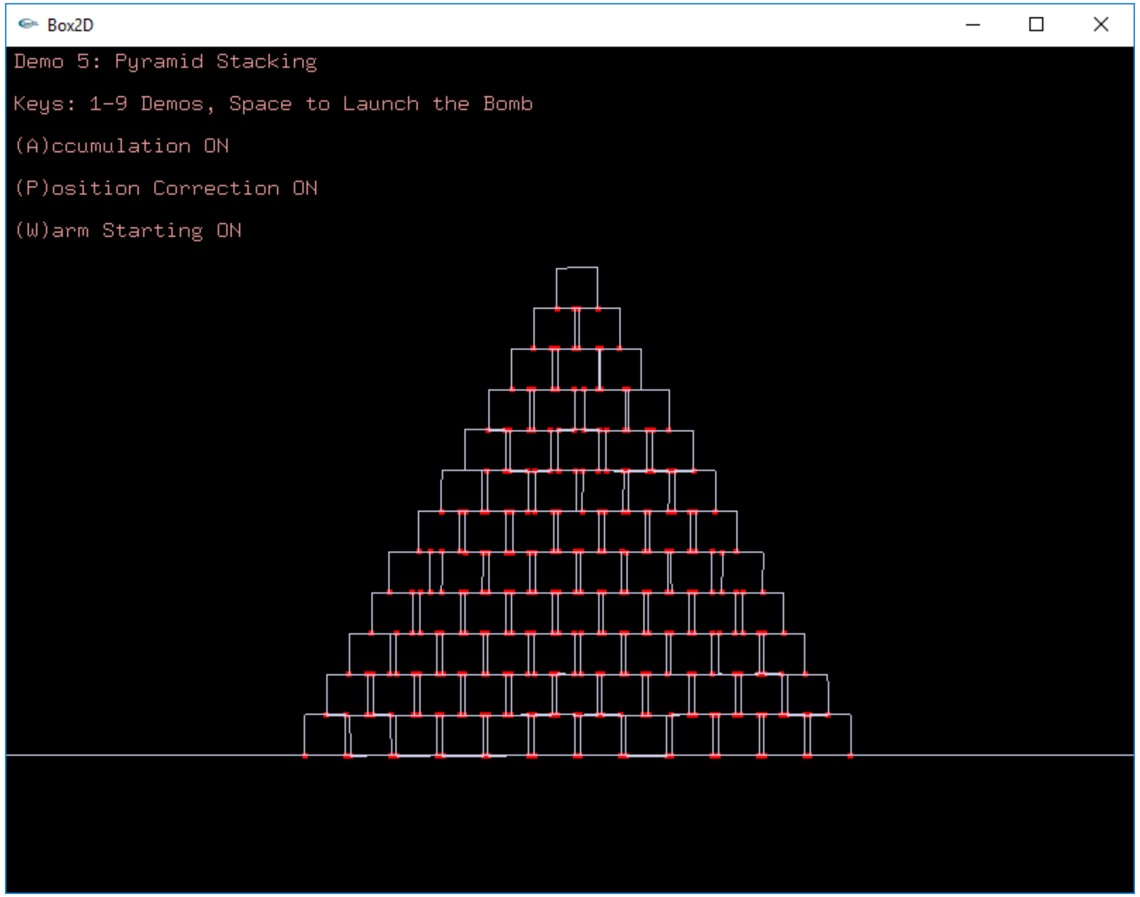
“If you can't solve a problem, then there is an easier problem you can solve: find it.”

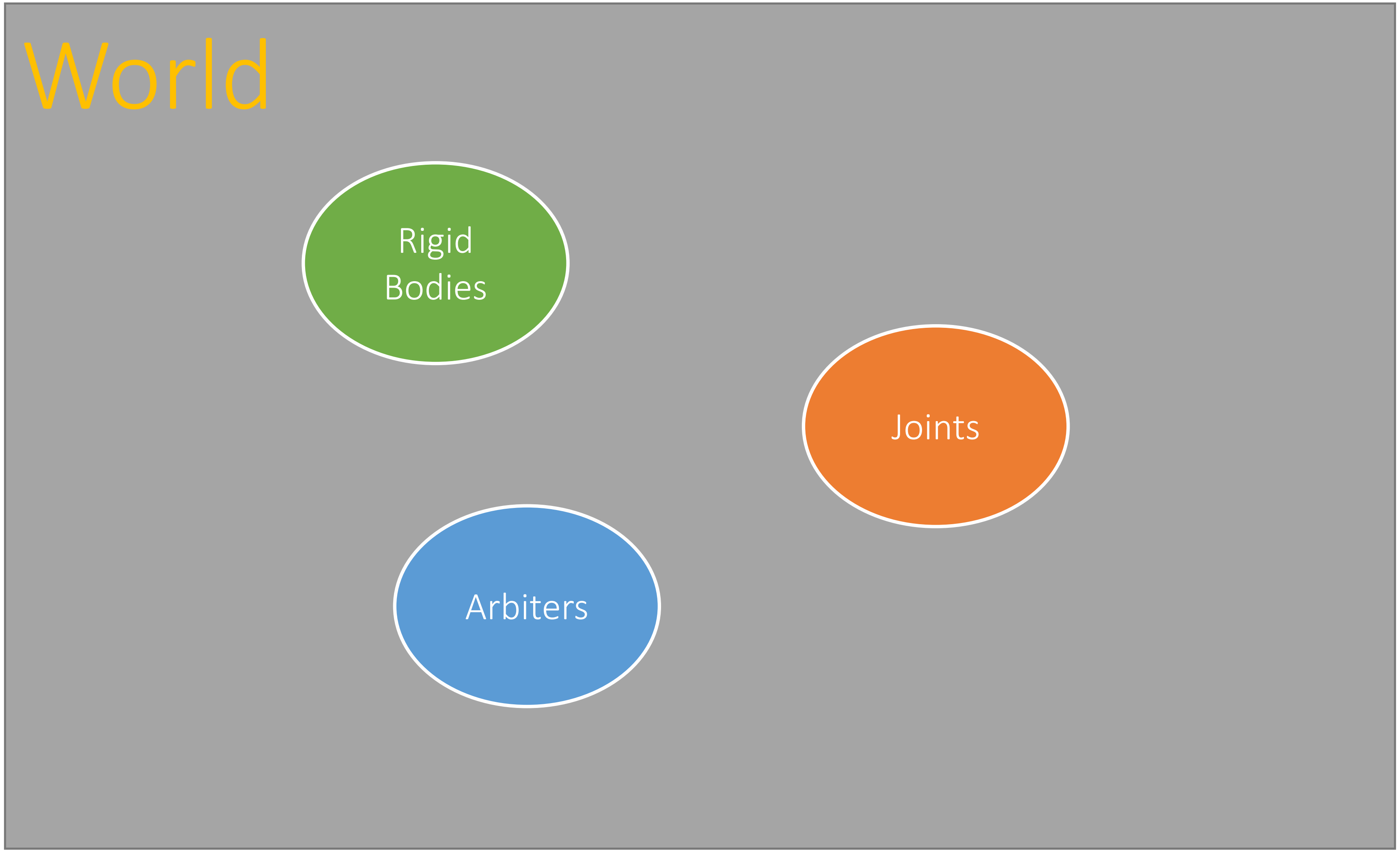
-George Pólya



Box2D Lite

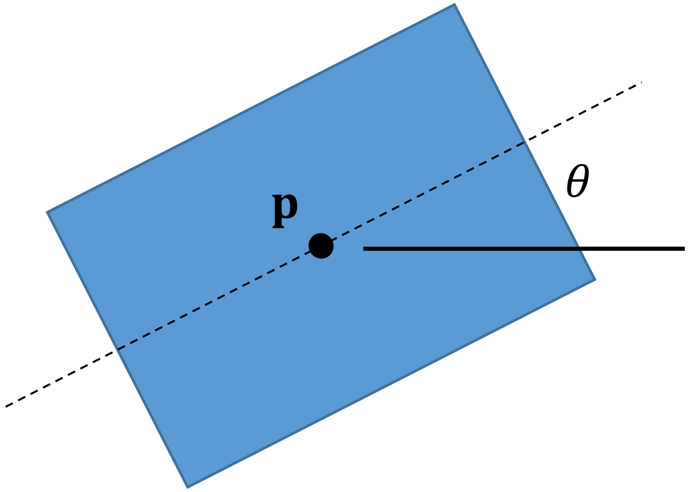
- Simple rigid body physics engine
- Created in 2006
- Presented at the GDC
- Links at the end



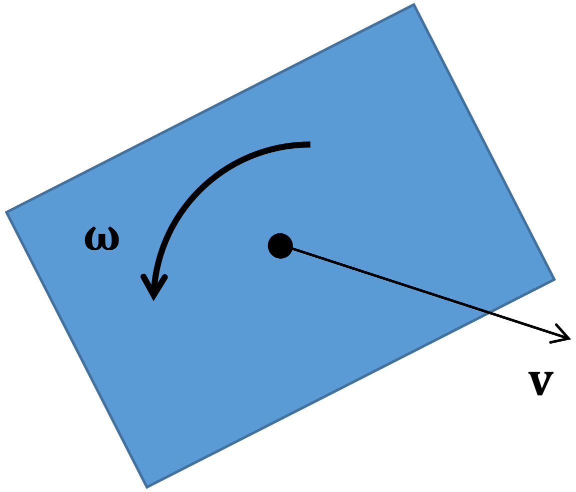

World
Rigid Bodies
Joints
Arbiters

```cpp
struct World
{
    std::vector<Body*> bodies;
    std::vector<Joint*> joints;
    std::map<ArbiterKey, Arbiter> arbiters;

    Vec2 gravity;
    int iterations;
};


struct ArbiterKey
{
    Body* body1;
    Body* body2;
};
```

Body state

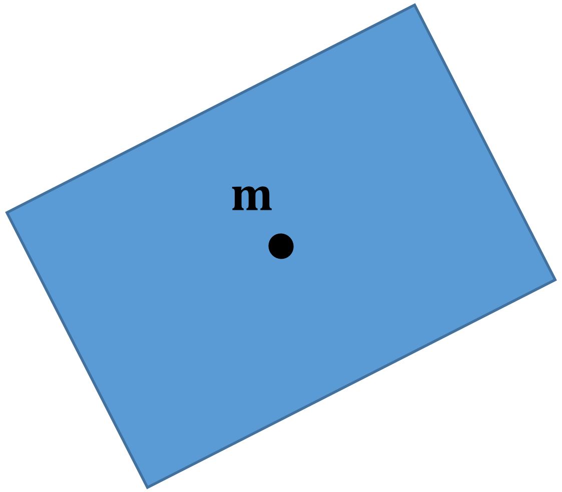


Position and Rotation

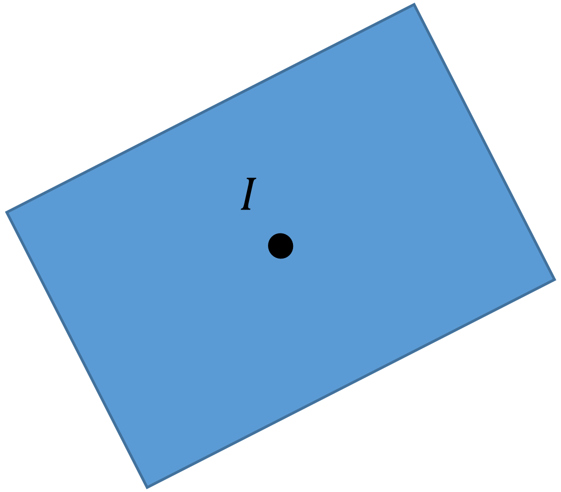


Linear and Angular Velocity

Mass properties



Mass



Inertia Tensor

In 3D the inertia tensor is a matrix

```
struct Body
{
    Vec2 position;
    float rotation;
    Vec2 velocity;
    float angularVelocity;

    Vec2 width;
    float friction;
    float mass, invMass;
    float I, invI;

    Vec2 force;
    float torque;
};
```

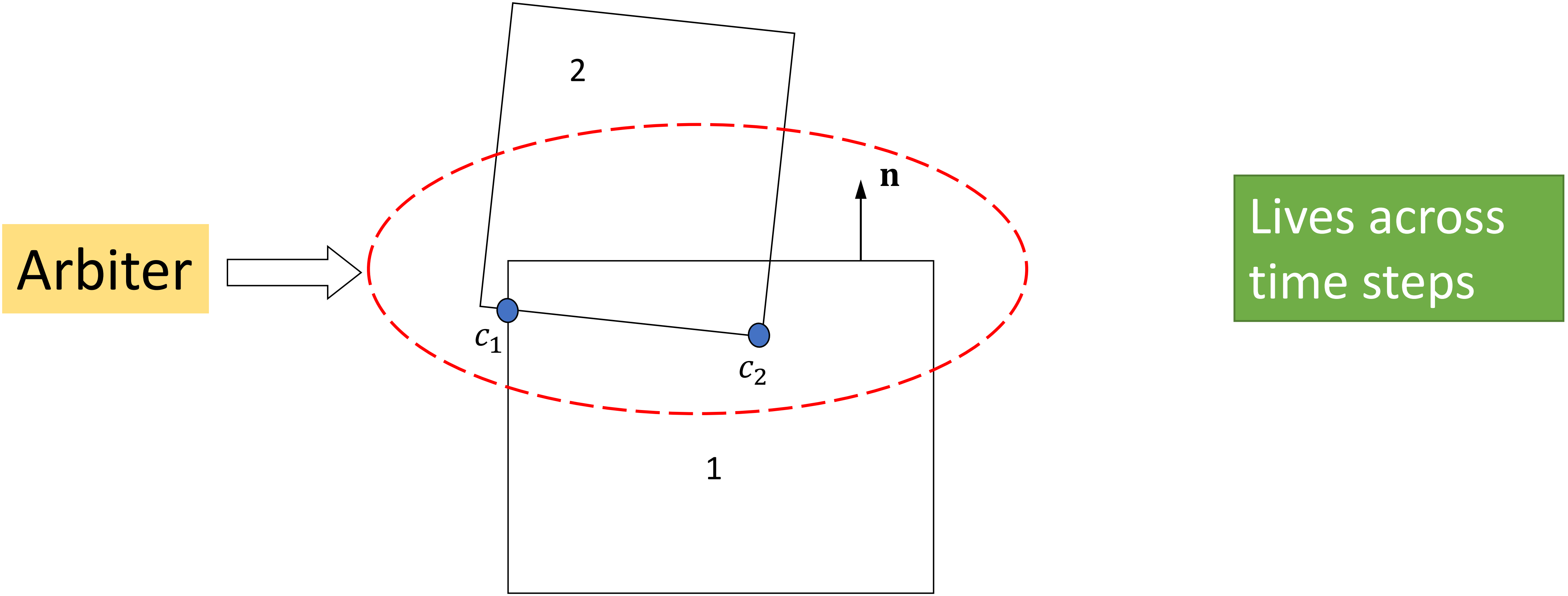
state

box properties

applied forces

static bodies:
invMass == 0
invI == 0

An Arbiter holds the contact points between two bodies



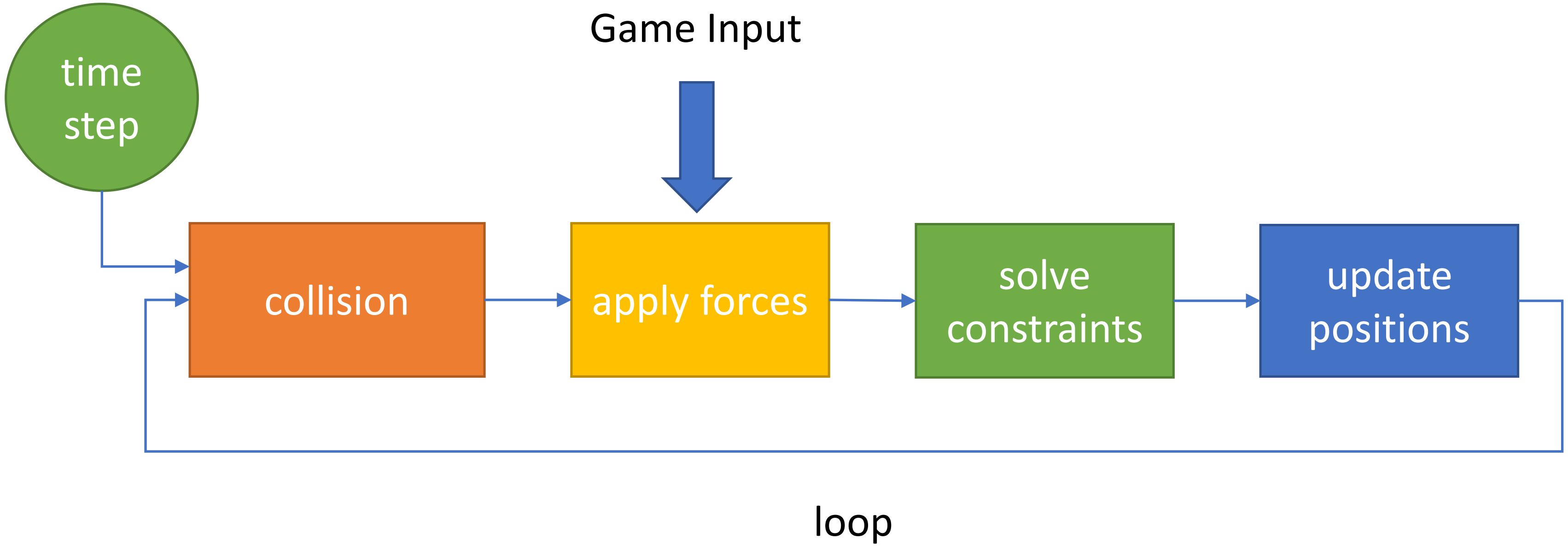
```
struct Arbiter
{
    Body* body1;
    Body* body2;

    float friction;

    Contact contacts[2];
    int numContacts;
};
```

connectivity

combined friction

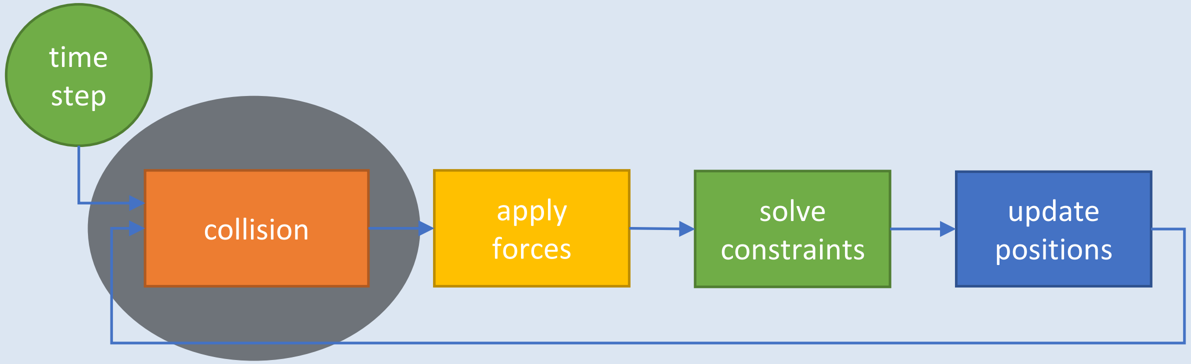
run-time data

Simulation Loop



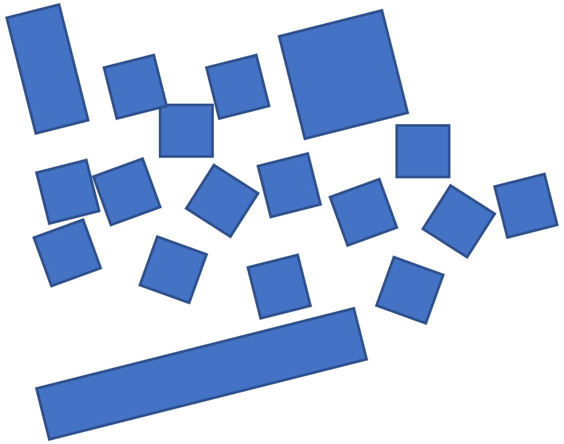
Stage 1

Collision

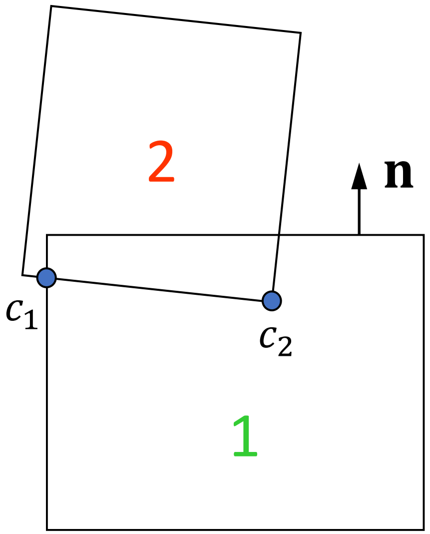


Collision Phases

broad-phase



narrow-phase



Broad-phase

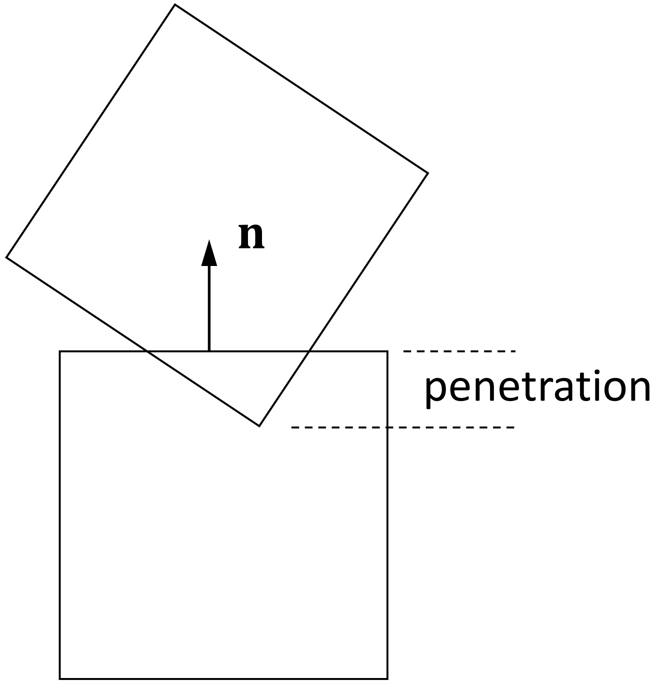
- Finds pairs of overlapping boxes
- Creates Arbiter for new pairs
- Updates existing Arbiters

WARNING: Box2D Lite uses a horribly slow O(N^2) broad-phase. Use an AABB tree, grid, etc. to speed up a real engine.

```cpp
void BroadPhase()
{
	for (int i = 0; i < (int)bodies.size(); ++i)
	{
		Body* bi = bodies[i];

		for (int j = i + 1; j < (int)bodies.size(); ++j)
		{
			Body* bj = bodies[j];

			if (bi->invMass == 0.0f && bj->invMass == 0.0f)
				continue;

			Arbiter newArb(bi, bj); //<<<<<<<<<<<<< This performs the narrow phase collision
			ArbiterKey key(bi, bj);

			if (newArb.numContacts > 0)
			{
				ArbIter iter = arbiters.find(key);
				if (iter == arbiters.end())
				{
					arbiters.insert(ArbPair(key, newArb));
				}
				else
				{
					iter->second.Update(newArb.contacts, newArb.numContacts);
				}
			}
			else
			{
				arbiters.erase(key);
			}
		}
	}
}
```

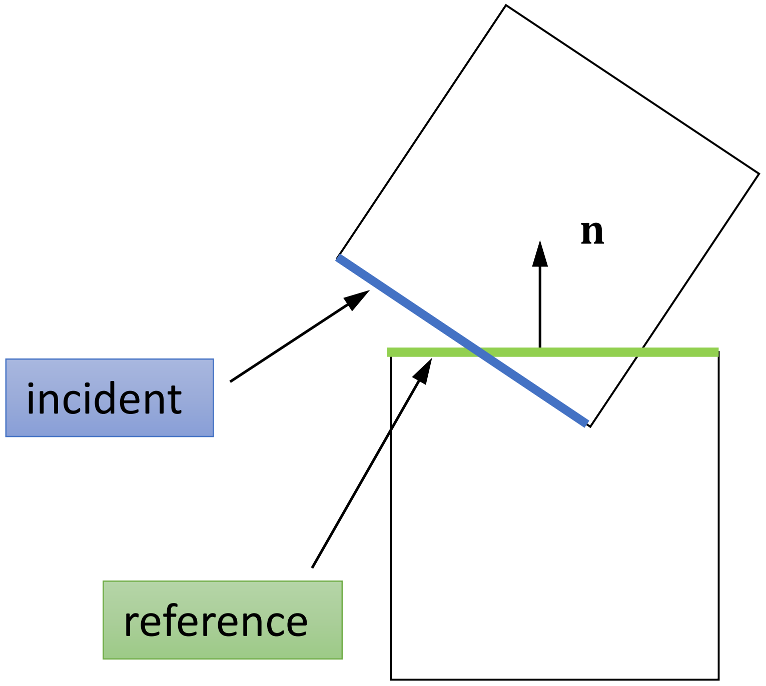
Narrow-phase
Box versus Box Collision

Find the normal vector with minimum penetration



Box-Box Clipping Setup

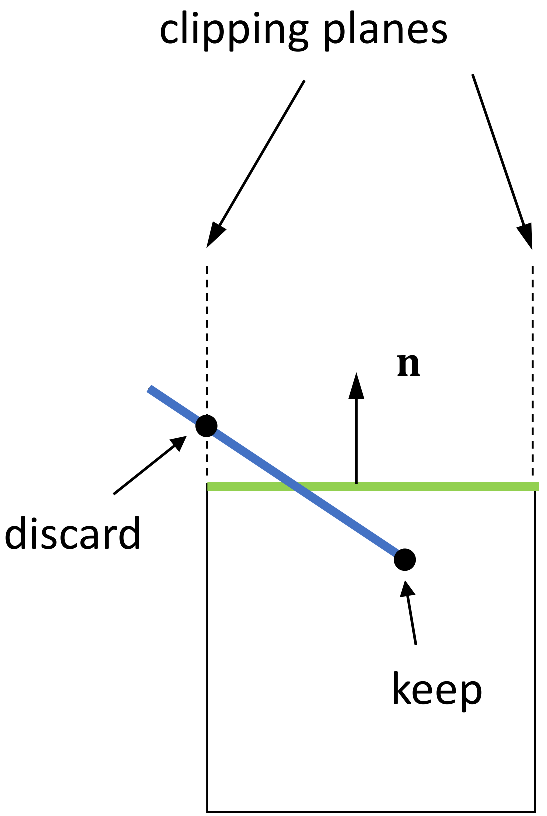
- Identify reference face
- Identify incident face



Box-Box Clipping

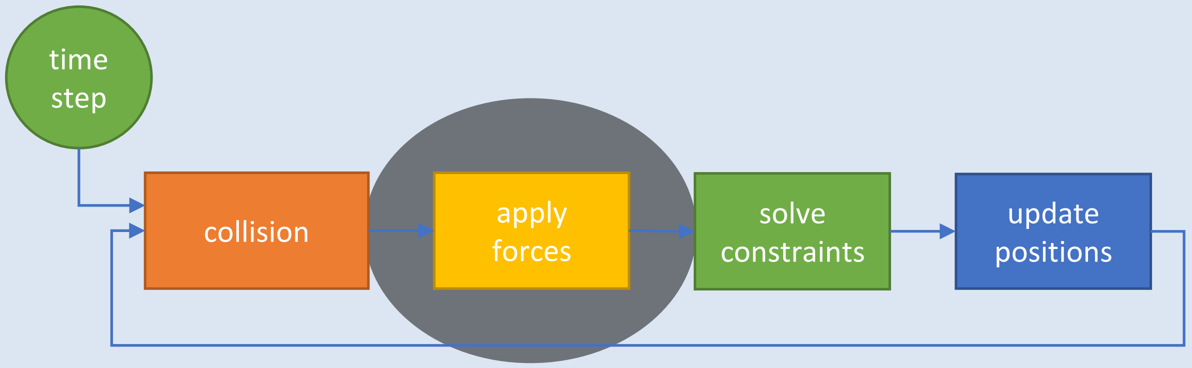
- Clip incident face against side planes

- Discard points above reference face



Stage 2

Apply Forces

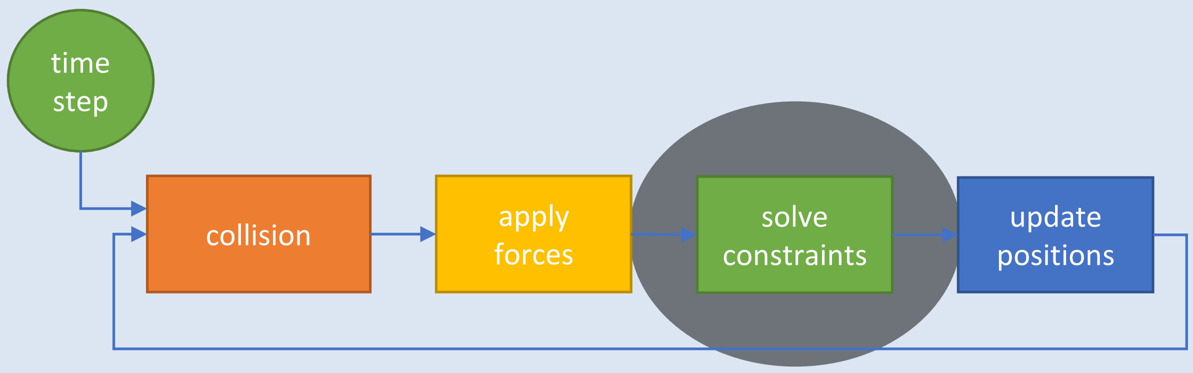


```
for (int i = 0; i < (int)bodies.size(); ++i)
{
    Body* b = bodies[i];

    if (b->invMass == 0.0f)
        continue;

    b->velocity += timeStep * (gravity + b->invMass * b->force);
    b->angularVelocity += timeStep * b->invI * b->torque;
}
```

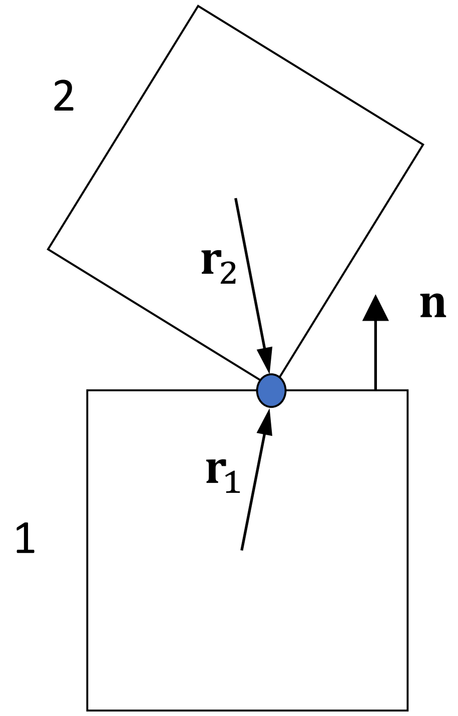
Newton's 2nd Law

Stage 3

Solve Constraints



Relative Velocity



Relative velocity at contact point:

$$\Delta\mathbf{v} = \mathbf{v}_2 + \boldsymbol{\omega}_2 \times \mathbf{r}_2 - \mathbf{v}_1 - \boldsymbol{\omega}_1 \times \mathbf{r}_1$$

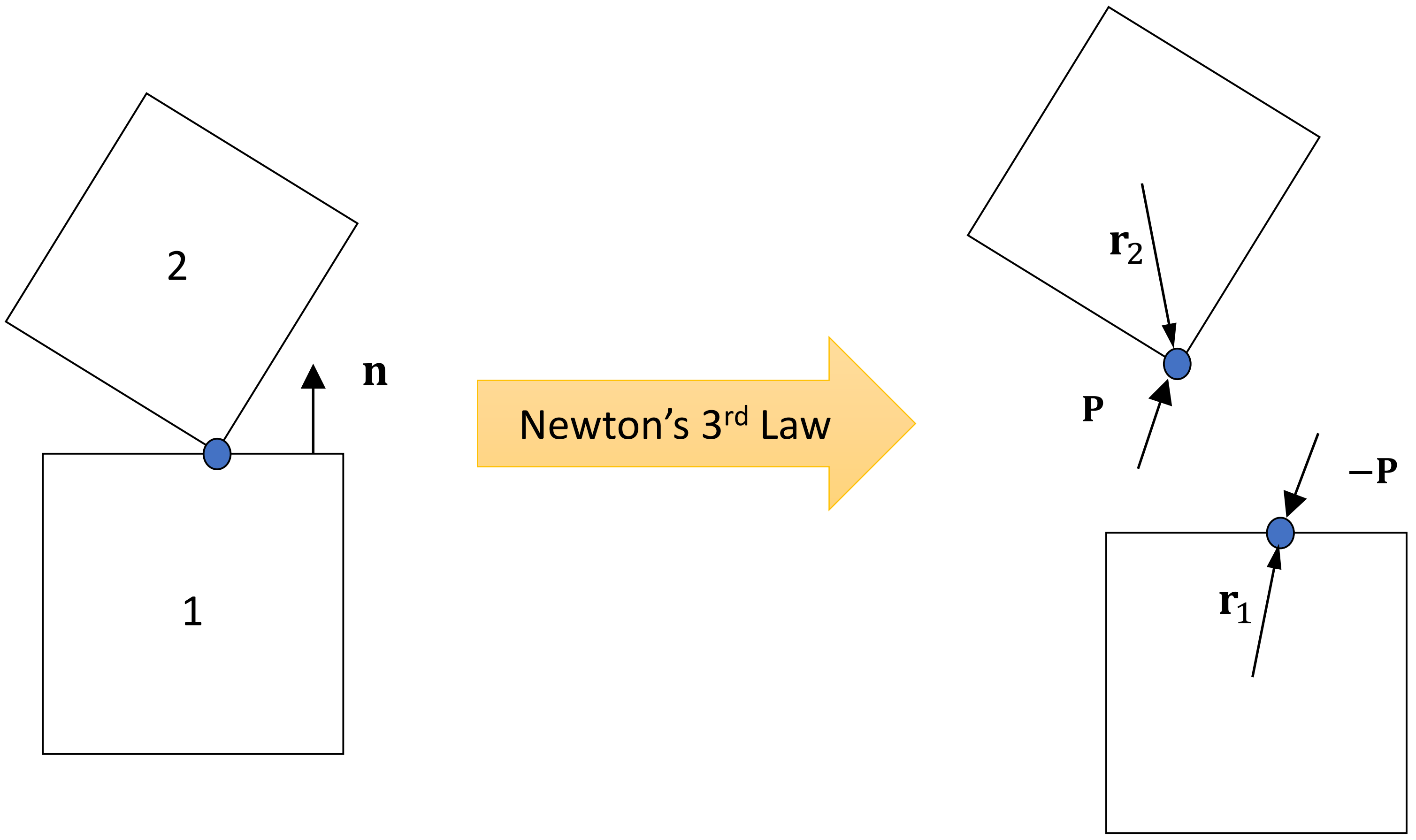
Velocity along unit normal vector:

$$v_n = \Delta\mathbf{v} \cdot \mathbf{n}$$

Non-penetration constraint:

$$v_n \geq 0$$

Idea: apply an impulse



Want to find an impulse P that makes v_n non-negative

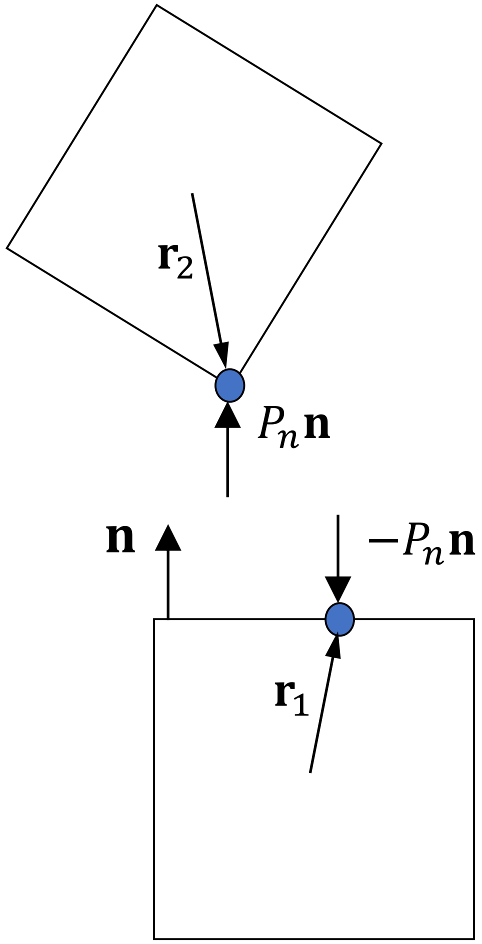
Simplify the impulse

We know the direction of the normal impulse. We only need its magnitude.

$\mathbf{P} = P_n \mathbf{n}$

The impulse can push, but not pull

$P_n \geq 0$



The impulse changes the velocity instantly

Newton's 2nd Law (again)

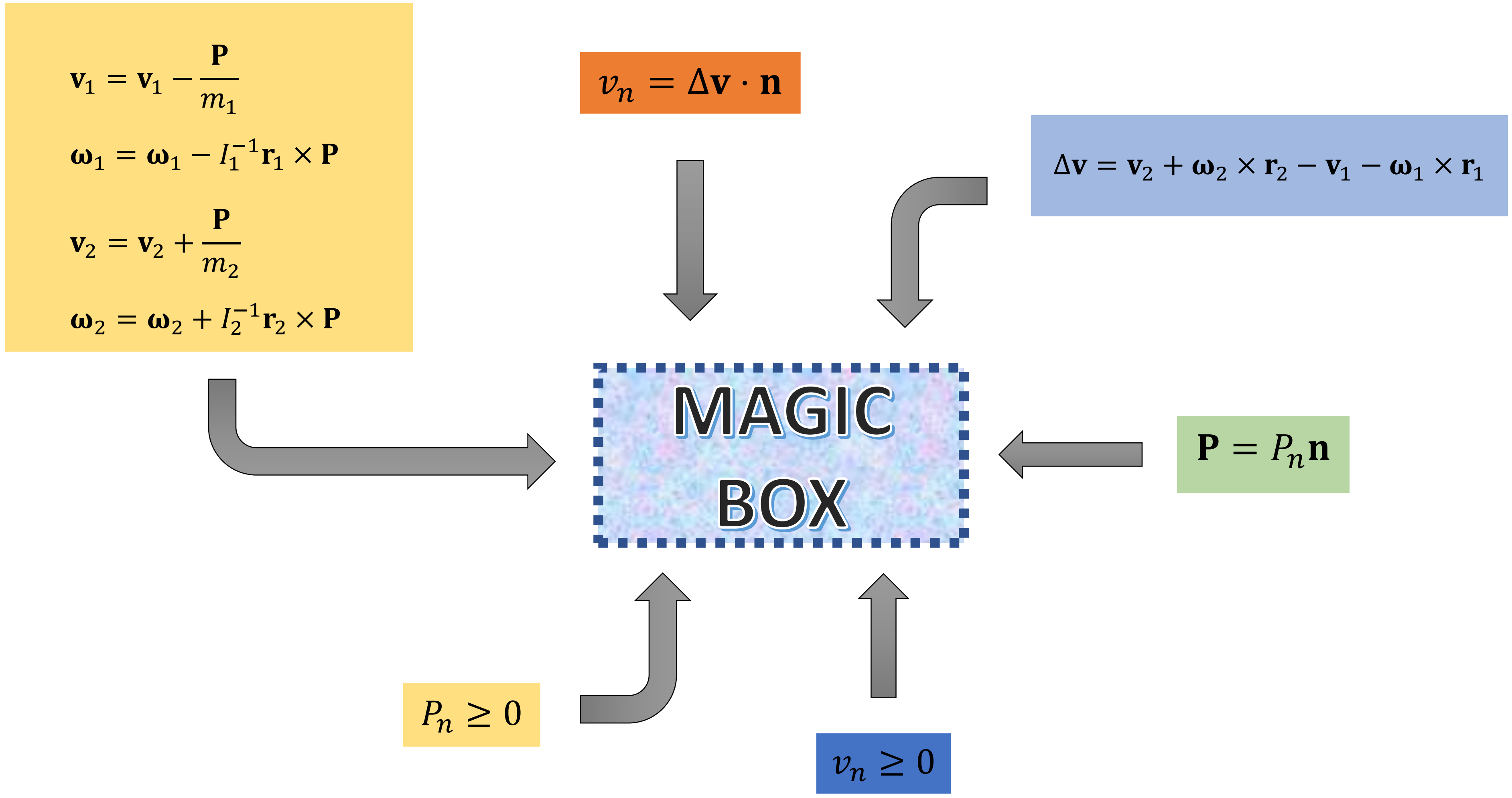
Use this to solve for the impulse.

$$\mathbf{v}_1 = \mathbf{v}_1 - \frac{\mathbf{P}}{m_1}$$

$$\boldsymbol{\omega}_1 = \boldsymbol{\omega}_1 - I_1^{-1}\mathbf{r}_1 \times \mathbf{P}$$

$$\mathbf{v}_2 = \mathbf{v}_2 + \frac{\mathbf{P}}{m_2}$$

$$\boldsymbol{\omega}_2 = \boldsymbol{\omega}_2 + I_2^{-1}\mathbf{r}_2 \times \mathbf{P}$$



$\mathbf{v}_1 = \mathbf{v}_1 - \frac{\mathbf{P}}{m_1}$
$\boldsymbol{\omega}_1 = \boldsymbol{\omega}_1 - I_1^{-1}\mathbf{r}_1 \times \mathbf{P}$
$\mathbf{v}_2 = \mathbf{v}_2 + \frac{\mathbf{P}}{m_2}$
$\boldsymbol{\omega}_2 = \boldsymbol{\omega}_2 + I_2^{-1}\mathbf{r}_2 \times \mathbf{P}$
$v_n = \Delta\mathbf{v} \cdot \mathbf{n}$
$\Delta\mathbf{v} = \mathbf{v}_2 + \boldsymbol{\omega}_2 \times \mathbf{r}_2 - \mathbf{v}_1 - \boldsymbol{\omega}_1 \times \mathbf{r}_1$
MAGIC BOX
$\mathbf{P} = P_n\mathbf{n}$
$P_n \geq 0$
$v_n \geq 0$

Solution

Answer:

$$P_n = \max(-m_n v_n, 0)$$

The *effective mass*:

$$\frac{1}{m_n} = \frac{1}{m_1} + \frac{1}{m_2} + [I_1^{-1}(\mathbf{r}_1 \times \mathbf{n}) \times \mathbf{r}_1 + I_2^{-1}(\mathbf{r}_2 \times \mathbf{n}) \times \mathbf{r}_2] \cdot \mathbf{n}$$

Velocity update

$$\mathbf{P} = P_n \mathbf{n}$$

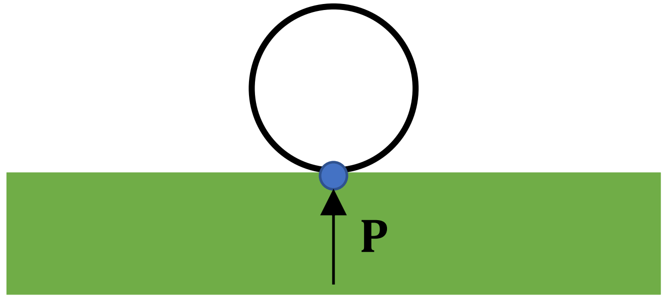
$$\mathbf{v}_1 = \mathbf{v}_1 - \frac{\mathbf{P}}{m_1}$$

$$\boldsymbol{\omega}_1 = \boldsymbol{\omega}_1 - I_1^{-1}\mathbf{r}_1 \times \mathbf{P}$$

$$\mathbf{v}_2 = \mathbf{v}_2 + \frac{\mathbf{P}}{m_2}$$

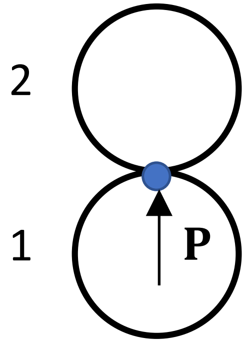
$$\boldsymbol{\omega}_2 = \boldsymbol{\omega}_2 + I_2^{-1}\mathbf{r}_2 \times \mathbf{P}$$

Effective mass: example 1



$$m_n = m$$

Effective mass: example 2

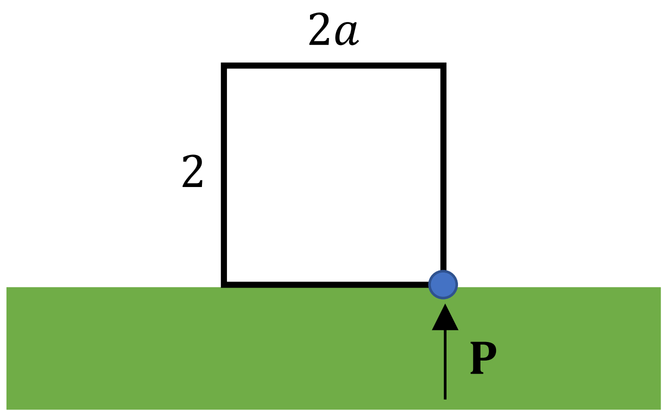


$$\frac{1}{m_n} = \frac{1}{m_1} + \frac{1}{m_2}$$

Case: $m_1 = m_2 = m$

$$m_n = \frac{1}{2}m$$

Effective mass: example 3

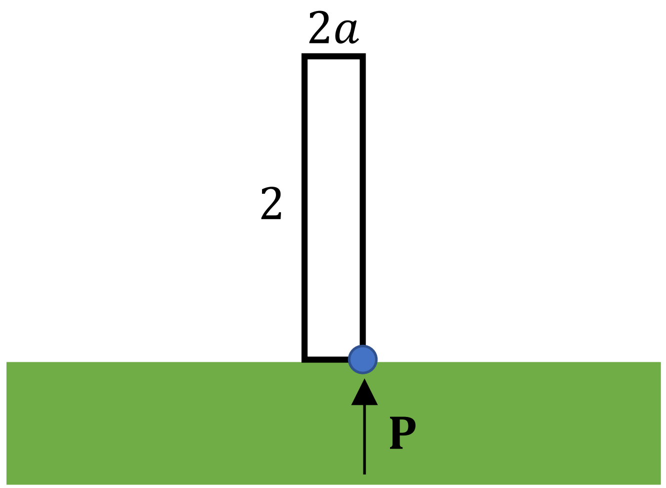


$$m_n = \frac{a^2+1}{4a^2+1} m$$

Case: $a = 1$

$$m_n = \frac{2}{5} m$$

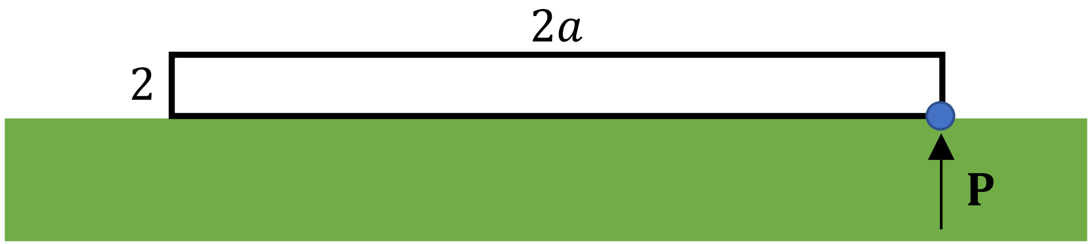
Effective mass: example 3



$$m_n = \frac{a^2 + 1}{4a^2 + 1} m$$

$$\lim_{a \to 0} m_n = m$$

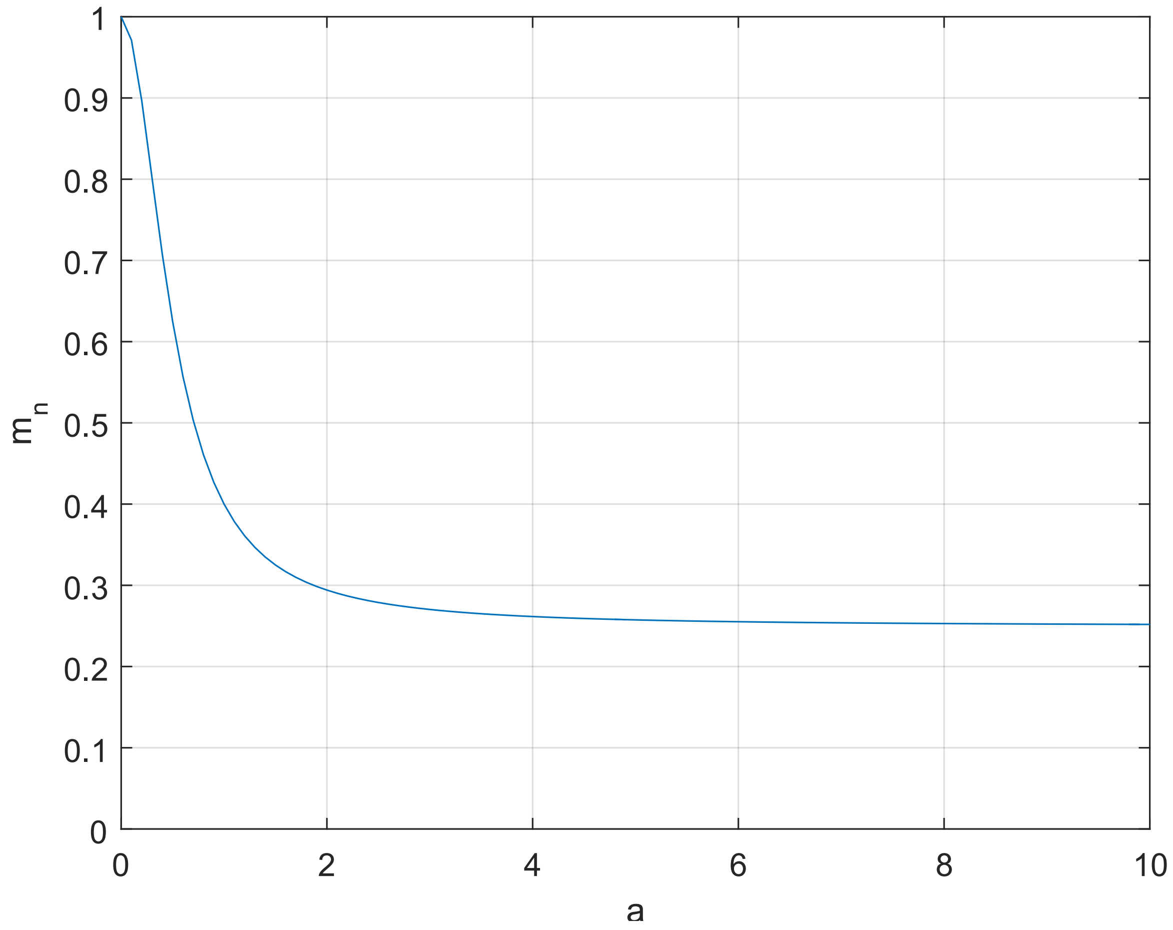
Effective Mass: example 3



$$m_n = \frac{a^2 + 1}{4a^2 + 1} m$$

$$\lim_{a \to \infty} m_n = \frac{1}{4} m$$

Effective mass versus box width



Case: $m = 1$

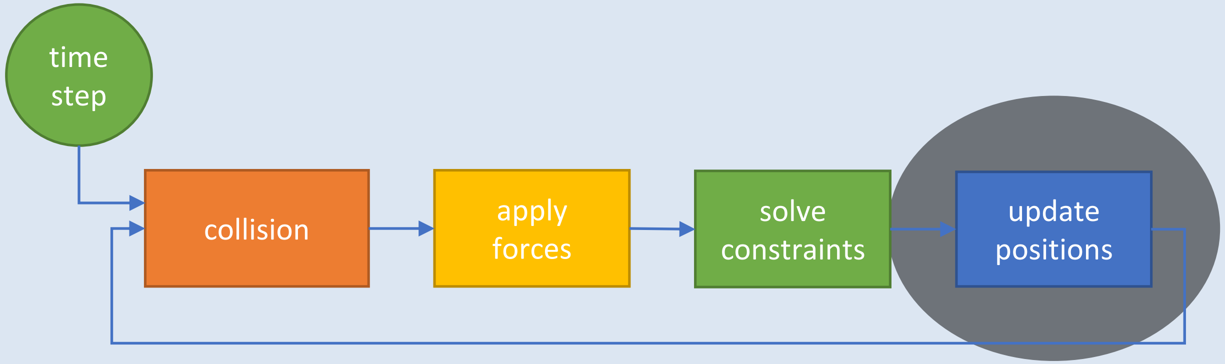
$$m_n = \frac{a^2 + 1}{4a^2 + 1}$$

More details left out

- Friction
- Overlap removal
- Joints
- Solver convergence

Stage 4

Update Positions



```
for (int i = 0; i < (int)bodies.size(); ++i)
{
    Body* b = bodies[i];

    b->position += timeStep * b->velocity;
    b->rotation += timeStep * b->angularVelocity;

    b->force.Set(0.0f, 0.0f);
    b->torque = 0.0f;
}
```

now loop

Next steps

- Download Box2D Lite
  - https://github.com/erincatto/box2d-lite
- Read
  - Docs folder at https://github.com/erincatto/box2d-lite
  - https://box2d.org/downloads/
- Tinker: Add circles. Add a better broad-phase. Make a game!
- Ask me: @erin_catto