

Game Architecture

Rigid Body Dynamics

Giving things weight.

Today's Agenda

- Reality check
- Linear dynamics
- Angular dynamics
- Collision response

Do We Need This?

- Using a physics simulation has pros and cons.
 - Pros
 - Realism
 - Emergent gameplay
 - Unpredictability and “free” animation.
 - Cons
 - Limited control or influence
 - Complexity
- The decision to use physics has an impact on every discipline.
 - **The goal is always to create an excellent game.**
 - If a physics system makes the game more excellent, go for it.

Physics

- It's a huge subject.
- Do what engineers do best: simplify.
 - Our only focus is a small branch of physics known as dynamics.
 - Dynamics covers how objects move over time.
 - We narrow that even further to rigid body dynamics.
 - Rigid bodies are infinitely hard, non-deformable solids.
 - Their shape does not change.
- We also limit ourselves to classical Newtonian mechanics.

Foundations

- Units
 - Use MKS. Just do it™.
- Linear and angular dynamics are independent!
 - It may seem crazy, but it's true. We can calculate them independently. Phew.
- Center of mass
 - Linear dynamics operates on an object as if its full mass is concentrated at a single point.
 - Center of mass can be calculated by taking a weighted average of all portions of the object.
 - For an object of uniform density, the center of mass is simply the centroid.
 - For example, a uniform density sphere's center of mass is its exact center point.

Linear Dynamics

- Position, velocity, and acceleration.
- Vector calculus is just scalar calculus on each vector element.
- Velocity, \mathbf{v} , is the derivative of position, \mathbf{r} :

$$\frac{d\mathbf{r}}{dt} = \mathbf{v} = \dot{\mathbf{r}}$$

- Acceleration, \mathbf{a} , is the derivative of velocity:

$$\frac{d^2\mathbf{r}}{dt^2} = \ddot{\mathbf{r}} = \frac{d\dot{\mathbf{r}}}{dt} = \frac{d\mathbf{v}}{dt} = \dot{\mathbf{v}} = \mathbf{a}$$

Forces

- A **force** is anything that causes a mass to accelerate or decelerate.
 - It has a magnitude and direction, so it's represented by a vector, \mathbf{F} .
 - It's the derivative of momentum, \mathbf{p} :

$$\mathbf{F} = \frac{d\mathbf{p}}{dt} = \frac{d(m\mathbf{v})}{dt} = m\mathbf{a}$$

- The total force applied to an object is the sum of all forces applied to it:

$$\mathbf{F}_T = \sum_i^N \mathbf{F}_i$$

Solving Equations of Motion

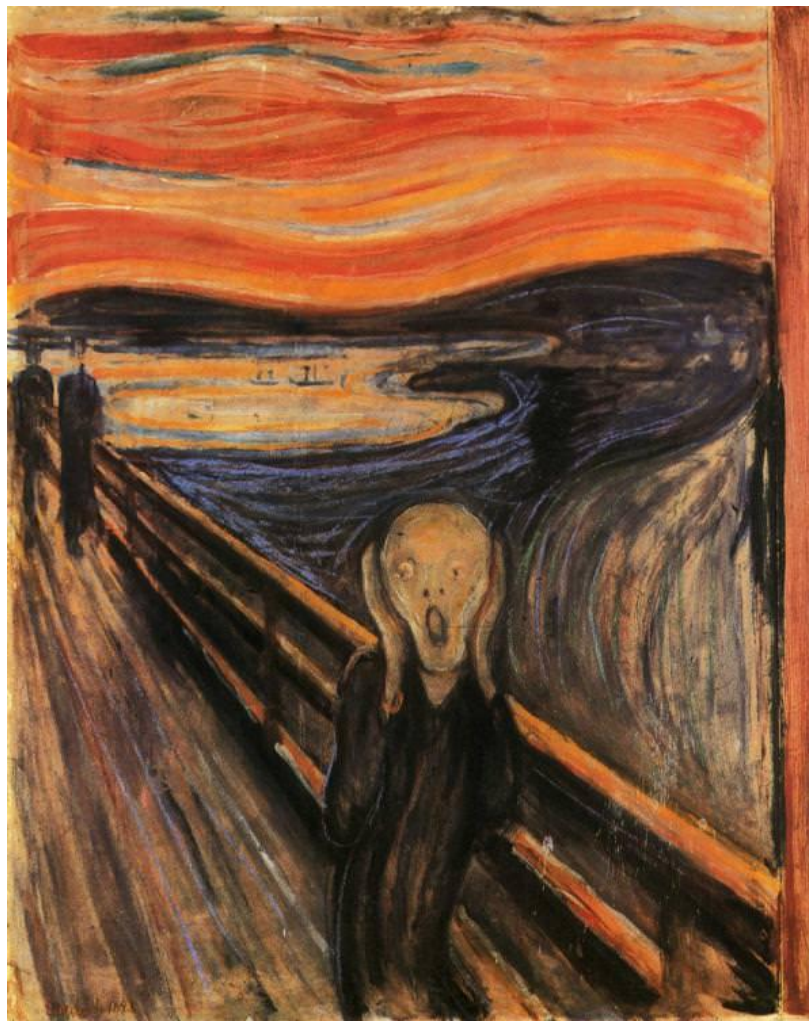
- A force can be constant or it can be a function of time or other quantities.

$$\mathbf{F}(t, \mathbf{r}(t), \mathbf{v}(t), \dots) = m\mathbf{a}(t)$$

- We need to solve an equation that takes both position and its derivative:

$$\ddot{\mathbf{r}}(t) = \frac{\mathbf{F}(t, \mathbf{r}(t), \dot{\mathbf{r}}(t))}{m}$$

- So we need to solve an ordinary differential equation.



Integration

- Instead, we can integrate velocity over time to change position.
- Integrating acceleration over time changes velocity.
- So twice integrating acceleration over time changes position.
- Each frame we...
 - Integrate acceleration over delta time to get velocity.
 - Integrate velocity over delta time to get position.
 - Thus the object moves.

Analytical Solution

We can try an analytic solution, much like the one you'll recognize from high school physics class.

But games are interactive. Forces are going to change over time, making predicting their behavior impossible.

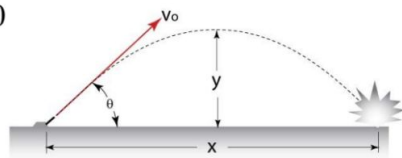
$$\mathbf{a}(t) = [0, g, 0]$$

$$g = -9.8m/s^2$$

$$\ddot{y}(t) = g$$

$$\dot{y}(t) = \int \ddot{y}(t) = gt + v_0$$

$$y(t) = \int \dot{y}(t) = \frac{1}{2}gt^2 + v_0t + y_0$$



Explicit Euler

- Given acceleration per second, \mathbf{a} , and delta time, Δt :

$$\mathbf{v}_n = \mathbf{v}_0 + \mathbf{a}_n \Delta t$$

$$\mathbf{r}_n = \mathbf{r}_0 + \mathbf{v}_n \Delta t$$

- Simple enough, right?

There's a problem...

- Let's say we have a car at rest accelerate at 10 m/s^2 .
- Where is the car after 10 seconds?

Analytic

$$\mathbf{r} = \frac{1}{2}\mathbf{a}t^2 + v_0t$$

$$\mathbf{r} = \frac{1}{2}(10)(10)^2 + 0$$

$$\mathbf{r} = 500$$

Explicit

t=0	v=0	r=0
t=1	v=10	r=10
t=2	v=20	r=30
t=3	v=30	r=60
t=4	v=40	r=100
t=5	v=50	r=150
t=6	v=60	r=210
t=7	v=70	r=280
t=8	v=80	r=360
t=9	v=90	r=450
t=10	v=100	r=550

Wait, what? How?

- We've discretized the change in velocity over time.
 - At each timestep, we apply the full change in velocity.
- In the real world, the velocity was changing through the timestep.
 - So at time 1.5, velocity was 15.
 - Our calculation assumed it was 20.
- This leads to an overestimation of the distance the car travels.
- One second timesteps are obviously an exaggeration.

Fourth-Order Runge-Kutta

$$\mathbf{v} = f(t, \mathbf{r}) \quad \mathbf{r}(t_0) = \mathbf{r}_0$$

$$t_{n+1} = t_n + \Delta t$$

$$k_1 = f(t_n, \mathbf{r}_n)$$

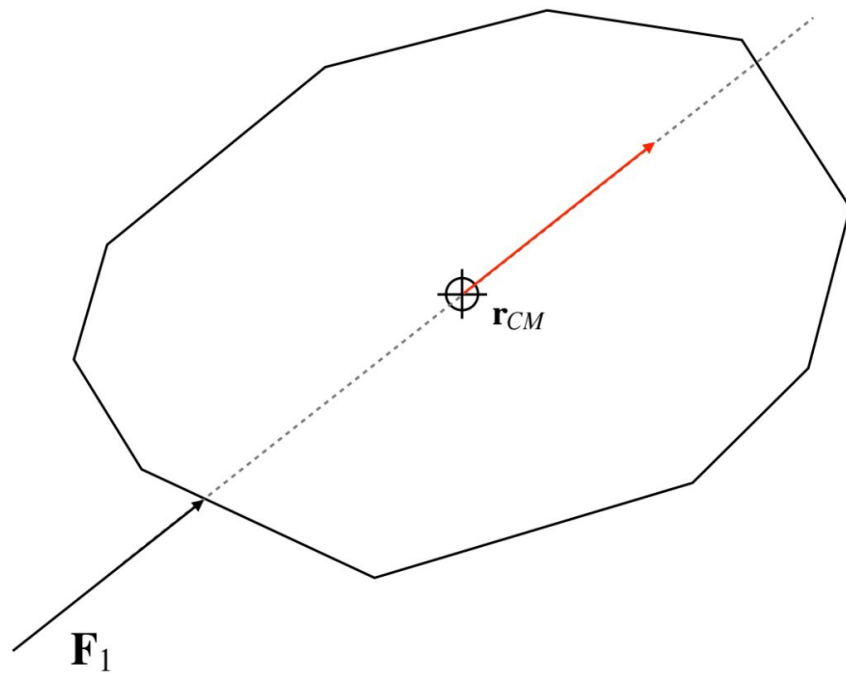
$$k_2 = f\left(t_n + \frac{\Delta t}{2}, \mathbf{r}_n + \frac{\Delta t k_1}{2}\right)$$

$$k_3 = f\left(t_n + \frac{\Delta t}{2}, \mathbf{r}_n + \frac{\Delta t k_2}{2}\right)$$

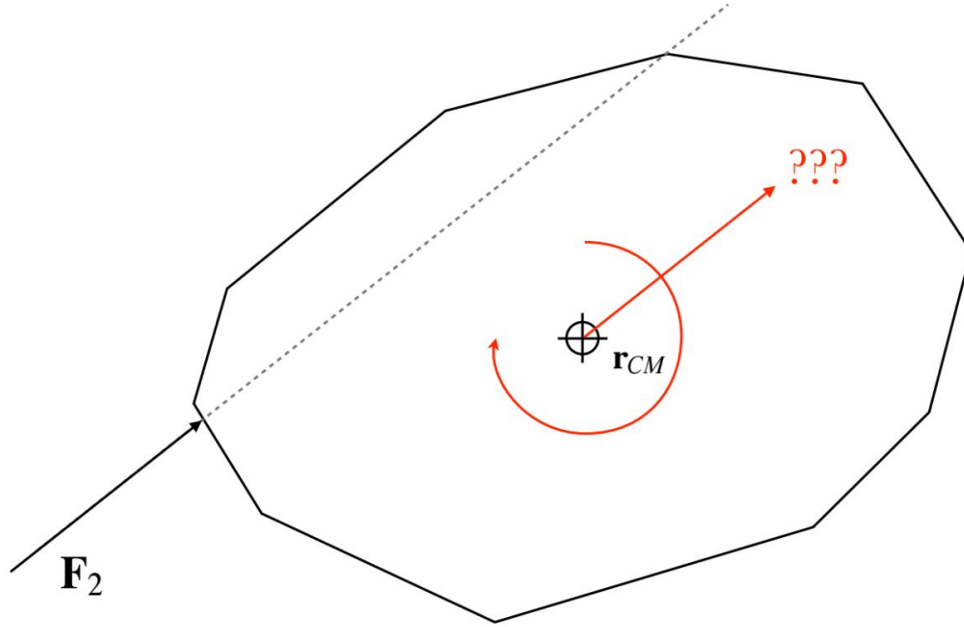
$$k_4 = f(t_n + \Delta t, \mathbf{r}_n + \Delta t k_3)$$

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

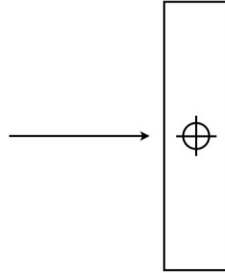
Angular Dynamics



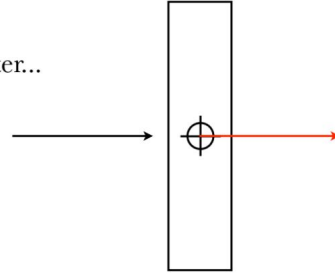
Angular Dynamics



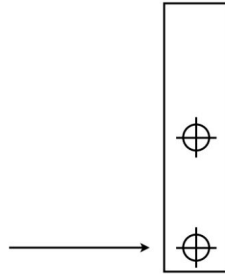
Angular Dynamics



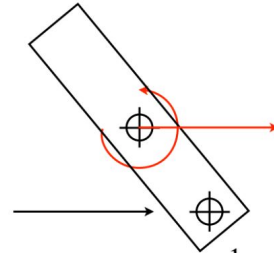
10 seconds later...



$$\text{Energy} = \frac{1}{2}mv^2$$



10 seconds later...



$$\text{Energy} = \frac{1}{2}mv^2 + \frac{1}{2}\mathbf{L} \cdot \boldsymbol{\omega}$$

Angular Dynamics

- Recall with linear dynamics, force applies to the center of mass.
 - Creates translation.
- **Torque** applies to offset from center of mass.
 - Creates rotation.
- Torques can be summed just as forces were.

Angular Dynamics in 2D

- In 2D, angular dynamics is almost identical to linear dynamics.
- Each angular quantity has a linear equivalent.
 - Torque τ is a rotational force.
 - Angular momentum \mathbf{L} vs. linear momentum \mathbf{p} .
 - Angular acceleration α vs. linear acceleration \mathbf{a} .
 - Angular velocity ω vs. linear velocity \mathbf{v} .
 - Orientation θ vs. position \mathbf{r} .
 - Moment of inertia \mathbf{I} vs. mass m .

Angular Dynamics in 2D

- Objects are treated as thin sheets lying on the xy plane.
 - The axis of rotation is the z axis.
- Orientation is described by a scalar, θ .
- Angular speed is in radians per second (rad/s), and is the derivative of orientation with respect to time.

$$\omega(t) = \frac{d\theta(t)}{dt} = \dot{\theta}(t)$$

- And angular acceleration is in radians per second squared (rad/s²).

$$\alpha(t) = \frac{d\omega(t)}{dt} = \dot{\omega}(t) = \ddot{\theta}(t)$$

Moment of Inertia in 2D

- Just as mass describes an object's resistance to change in linear velocity, the **moment of inertia** describes an object's resistance to change in angular speed.
 - The closer mass is to the axis of rotation, the easier the object is to rotate around that axis.
- Calculating the moment of inertia is beyond the scope of this lecture.

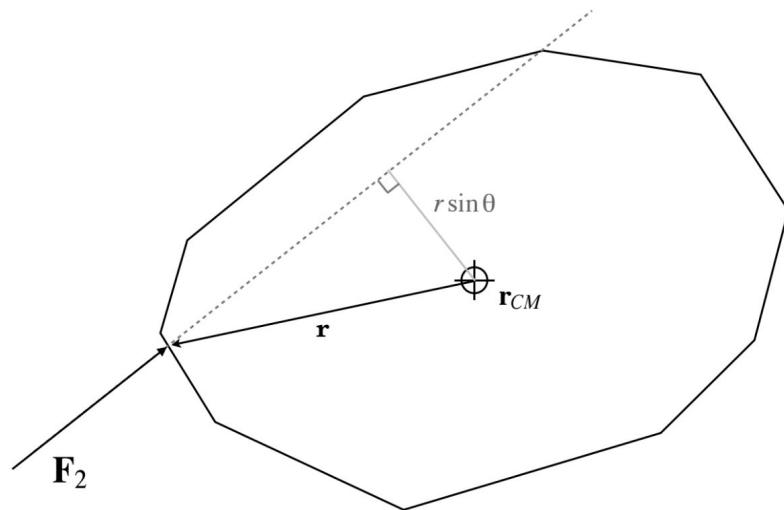
Calculating Torque

- Make a vector, \mathbf{r} , from the center of mass to the point of contact.
- Torque is the cross product of \mathbf{r} and \mathbf{F} .
 - Force applied to the center of mass yields a cross product of 0.

$$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{F}$$

- Torque is related to angular acceleration and the moment of inertia similarly to mass and linear acceleration.

$$\tau_z = I\alpha(t) = I\dot{\omega}(t) = I\ddot{\theta}(t)$$



Solving Angular Equations in 2D

- Just like linear dynamics, we have some ordinary differential equations to solve:

$$\tau(t) = I\dot{\omega}(t) \qquad \omega(t) = \dot{\theta}(t)$$

- And again just like linear, we can solve them with integration:

$$\omega_n = \omega_0 + \frac{\tau_n}{I}\Delta t$$

$$\theta_n = \theta_0 + \omega_n\Delta t$$

- Don't actually use explicit Euler though.

Angular Dynamics in 3D

- Ok, that 2D stuff is cool and all, but aren't we working in 3D?
- Yes!
- The concepts are similar.
- The math is more complex.
- Brace yourselves.

The Inertia Tensor

- The 3D equivalent of the 2D moment of inertia.
- A 3D body will have different distributions of mass around each axis.
 - Therefore, it will have different moments of inertia for each axis.
- The **inertia tensor** is a 3 x 3 matrix, **I**:

$$\mathbf{I} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}$$

- The diagonal are the moments of inertia around each principal axis.
- The 0s are known as the products of inertia.
 - They are 0 when the body is symmetrical around each axis.

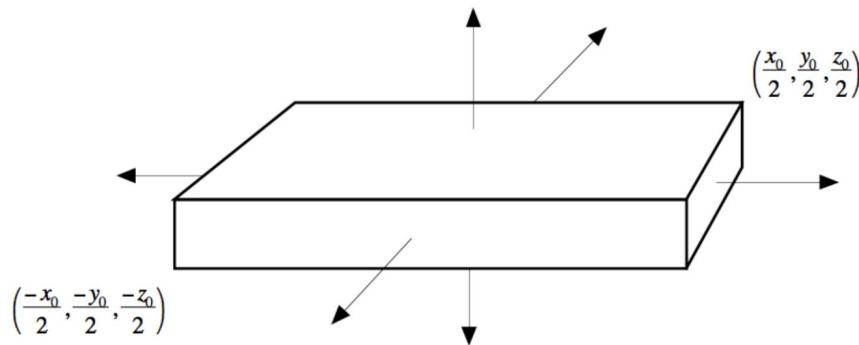
Calculating the Inertia Tensor

- At a given time t , let r_i' be the displacement of the i^{th} particle from center of mass $x(t)$: $r_i' = r_i(t) - x(t)$.
- We can define the inertia tensor in terms of r_i' :

$$\mathbf{I}(t) = \sum \begin{bmatrix} m_i(r_{iy}'^2 + r_{iz}'^2) & -m_i r_{ix}' r_{iy}' & -m_i r_{ix}' r_{iz}' \\ -m_i r_{iy}' r_{ix}' & m_i(r_{ix}'^2 + r_{iz}'^2) & -m_i r_{iy}' r_{iz}' \\ -m_i r_{iz}' r_{ix}' & -m_i r_{iz}' r_{iy}' & m_i(r_{ix}'^2 + r_{iy}'^2) \end{bmatrix}$$

Calculating the Inertia Tensor

- Consider a rectangular block of uniform density with its center of mass at its exact center.
- The sums in the previous equation are integrals over the volume of the block.



$$\rho(x,y,z) = 1$$

$$M = x_0 y_0 z_0$$

Calculating the Inertia Tensor

$$\begin{aligned}
 I_{xx} &= \int_{-\frac{x_0}{2}}^{\frac{x_0}{2}} \int_{-\frac{y_0}{2}}^{\frac{y_0}{2}} \int_{-\frac{z_0}{2}}^{\frac{z_0}{2}} \rho(x, y, z) (y^2 + z^2) dx dy dz \\
 &= \int_{-\frac{x_0}{2}}^{\frac{x_0}{2}} \int_{-\frac{y_0}{2}}^{\frac{y_0}{2}} \int_{-\frac{z_0}{2}}^{\frac{z_0}{2}} (y^2 + z^2) dx dy dz \\
 &= \int_{-\frac{x_0}{2}}^{\frac{x_0}{2}} \int_{-\frac{y_0}{2}}^{\frac{y_0}{2}} y^2 z + \frac{z^3}{3} \bigg|_{z=-\frac{z_0}{2}}^{z=\frac{z_0}{2}} dx dy \\
 &= \int_{-\frac{x_0}{2}}^{\frac{x_0}{2}} \int_{-\frac{y_0}{2}}^{\frac{y_0}{2}} y^2 z_0 + \frac{z_0^3}{12} dx dy \\
 &= \int_{-\frac{x_0}{2}}^{\frac{x_0}{2}} \frac{y_0^3}{3} z_0 + \frac{z_0^3}{12} y_0 \bigg|_{y=-\frac{y_0}{2}}^{y=\frac{y_0}{2}} dx \\
 &= \int_{-\frac{x_0}{2}}^{\frac{x_0}{2}} \frac{y_0^3 z_0}{12} + \frac{z_0^3 y_0}{12} dx \\
 &= \frac{y_0^3 z_0}{12} + \frac{z_0^3 y_0}{12} \bigg|_{x=-\frac{x_0}{2}}^{x=\frac{x_0}{2}} \\
 &= \frac{y_0^3 z_0 x_0}{12} + \frac{z_0^3 y_0 x_0}{12} \\
 &= \frac{x_0 y_0 z_0}{12} (y_0^2 + z_0^2) \\
 &= \frac{M}{12} (y_0^2 + z_0^2) \\
 I_{yy} &= \frac{M}{12} (x_0^2 + z_0^2) \\
 I_{zz} &= \frac{M}{12} (x_0^2 + y_0^2)
 \end{aligned}$$

Calculating the Inertia Tensor

$$I_{\text{body}} = \frac{M}{12} \begin{vmatrix} y_0^2 + z_0^2 & 0 & 0 \\ 0 & x_0^2 + z_0^2 & 0 \\ 0 & 0 & x_0^2 + y_0^2 \end{vmatrix}$$

Forget everything I just said.

There's an easier way.

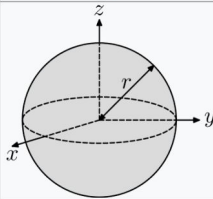
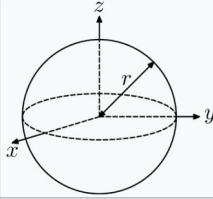
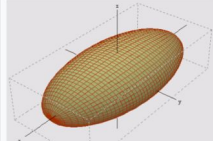
List of 3D inertia tensors [\[edit \]](#)

This list of [moment of inertia tensors](#) is given for [principal axes](#) of each object.

To obtain the scalar moments of inertia I above, the tensor moment of inertia \mathbf{I} is projected along some axis defined by a [unit vector](#) \mathbf{n} according to the formula:

$$\mathbf{n} \cdot \mathbf{I} \cdot \mathbf{n} \equiv n_i I_{ij} n_j ,$$

where the dots indicate [tensor contraction](#) and we have used the [Einstein summation convention](#). In the above table, \mathbf{n} would be the unit [Cartesian basis](#) \mathbf{e}_x , \mathbf{e}_y , \mathbf{e}_z to obtain I_x , I_y , I_z respectively.

Description	Figure	Moment of inertia tensor
Solid sphere of radius r and mass m		$I = \begin{bmatrix} \frac{2}{5}mr^2 & 0 & 0 \\ 0 & \frac{2}{5}mr^2 & 0 \\ 0 & 0 & \frac{2}{5}mr^2 \end{bmatrix}$
Hollow sphere of radius r and mass m		$I = \begin{bmatrix} \frac{2}{3}mr^2 & 0 & 0 \\ 0 & \frac{2}{3}mr^2 & 0 \\ 0 & 0 & \frac{2}{3}mr^2 \end{bmatrix}$
Solid ellipsoid of semi-axes a , b , c and mass m		$I = \begin{bmatrix} \frac{1}{5}m(b^2 + c^2) & 0 & 0 \\ 0 & \frac{1}{5}m(a^2 + c^2) & 0 \\ 0 & 0 & \frac{1}{5}m(a^2 + b^2) \end{bmatrix}$

Just do this.

Orientation

- In 2D we only needed a scalar value.
- For 3D, we need three value, one for each axis.
 - But we know that Euler angles are prone to issues like Gimbal lock.
- So, we'll use quaternions.
 - Recall, quaternion consists of unit axis of rotation, \mathbf{u} , scaled by sine of the half angle, θ , and a scalar equal to cosine of the half angle.

$$q = [q_x, q_y, q_z, q_w] = [\mathbf{q}, q_w] = [\mathbf{u} \sin(\frac{\theta}{2}), \cos(\frac{\theta}{2})]$$

Angular Velocity

- We can represent angular velocity as a unit vector (the axis of rotation), \mathbf{u} , scaled by the two dimensional angular velocity, ω_u :

$$\boldsymbol{\omega}(t) = \omega_u(t)\mathbf{u}(t) = \dot{\theta}_u(t)\mathbf{u}(t)$$

- In 2D if no torques acted on a body, angular speed was constant.
- In 3D, this is not true.
 - The axis of rotation can change without external torques.
- In other words, **angular velocity is not conserved**.

But wait!

Angular *momentum* IS conserved.

Angular Momentum

- Analogous to linear momentum.

$$\mathbf{L}(t) = \mathbf{I}\omega(t)$$

- We use angular momentum as the primary quantity in the angular dynamics simulation.
 - Angular velocity is calculated after angular momentum.
 - We'll see how shortly.

Torque in 3D

- Torque is still the cross product of the force and the vector from the center of mass to point of contact.
 - Hooray!
- But, we write it in terms of angular momentum, since it's our new primary quantity.

$$\tau = \mathbf{I}\alpha(t) = \mathbf{I}\frac{d\omega(t)}{dt} = \frac{d}{dt}(\mathbf{I}\omega(t)) = \frac{d\mathbf{L}(t)}{dt}$$

Solving Angular Equations in 3D

- Intuitively, we'd solve the equations the same way we solved linear motion and 2D angular motion.
- So, we have our ordinary differential equations.

$$\tau(t) = \mathbf{I}\dot{\omega}(t), \quad \omega(t) = \dot{\theta}(t)$$

- And we solve them by integrating:

$$\omega(t_2) = \omega(t_1) + \mathbf{I}^{-1}\tau(t_1)\Delta t$$

$$\theta(t_2) = \theta(t_1) + \omega(t_1)\Delta t$$

Solving Angular Equations in 3D

- We construct an angular velocity quaternion:

$$\omega = [\omega_x, \omega_y, \omega_z, 0]$$

- Then we can relate orientation quaternion to the angular velocity quaternion:

$$\frac{dq(t)}{dt} = \dot{q}(t) = \frac{1}{2}\omega(t)q(t)$$

Solving Angular Equations in 3D

- Finally we have our ODEs for angular momentum.

$$\tau(t) = \dot{\mathbf{L}}(t) \qquad \omega(t) = \mathbf{I}^{-1}\mathbf{L}(t)$$

$$\omega(t) = [\omega(t), 0] \qquad \frac{1}{2}\omega(t)q(t) = \dot{q}(t)$$

- And integration method.

$$\mathbf{L}(t_2) = \mathbf{L}(t_1) + \tau(t_1)\Delta t$$

$$\omega(t_2) = [\mathbf{I}^{-1}\mathbf{L}(t_2), 0]$$

$$q(t_2) = q(t_1) + \frac{1}{2}\omega(t_1)q(t_1)\Delta t$$

Angular Dynamics Summary

- That was a lot. Let's recap.
 - Angular velocity is not conserved, but angular momentum is.
 - At each timestep, we:
 - Calculate the torques on an object, and sum them.
 - Add our net torque to the angular momentum.
 - Multiply the resulting angular momentum by the inverse inertia tensor matrix.
 - Stick the result into our angular velocity quaternion, with 0 as the w component.
 - Integrate orientation using the formula described previously.

Collision Response

- All of this simulation is great, but what happens when two things collide?

Energy

- Collision response involves the conservation of **energy**.
 - Energy may be potential, V , or kinetic, T .
 - The total energy, $E = V + T$, of an isolated system is constant.
- Kinetic energy from linear motion can be defined in terms of mass and velocity, or momentum and velocity vectors:

$$T_{linear} = \frac{1}{2}mv^2 \qquad T_{linear} = \frac{1}{2}\mathbf{p} \cdot \mathbf{v}$$

- And kinetic energy from rotational motion:

$$T_{angular} = \frac{1}{2}\mathbf{L} \cdot \boldsymbol{\omega}$$

Impulse

- Simplifications.
 - Collision force acts of infinitesimally small time, creating an impulse. Velocities change instantaneously.
 - There's no friction at the point of contact.
 - Submolecular interactions and effects can be approximated with a coefficient of restitution, ε .
- All analysis based around the idea that linear momentum is conserved:

$$\mathbf{p}_1 + \mathbf{p}_2 = \mathbf{p}'_1 + \mathbf{p}'_2$$
$$m_1 \mathbf{v}_1 + m_2 \mathbf{v}_2 = m_1 \mathbf{v}'_1 + m_2 \mathbf{v}'_2$$

- Kinetic energy is also conserved, with some lost due to heat and sound:

$$\frac{1}{2}m_1 v_1^2 + \frac{1}{2}m_2 v_2^2 = \frac{1}{2}m_1 v_1'^2 + \frac{1}{2}m_2 v_2'^2 + T_{lost}$$

Impulse

- An impulse causes an instantaneous change in velocity.

$$\hat{\mathbf{p}} = \Delta \mathbf{p} = m \Delta \mathbf{v}$$

- With no friction, the impulse vector is normal to the collision surface.

$$\hat{\mathbf{p}} = \hat{p} \mathbf{n}$$

- If one body experiences the impulse \mathbf{p} , then the other body must experience an equal but opposite impulse.
- So we can write our equations as follows...

Impulse

$$\mathbf{p}'_1 = \mathbf{p}_1 + \hat{\mathbf{p}}$$

$$\mathbf{p}'_2 = \mathbf{p}_2 - \hat{\mathbf{p}}$$

$$m_1 \mathbf{v}'_1 = m_1 \mathbf{v}_1 + \hat{\mathbf{p}}$$

$$m_2 \mathbf{v}'_2 = m_2 \mathbf{v}_2 - \hat{\mathbf{p}}$$

$$\mathbf{v}'_1 = \mathbf{v}_1 + \frac{\hat{p}}{m_1} \mathbf{n}$$

$$\mathbf{v}'_2 = \mathbf{v}_2 - \frac{\hat{p}}{m_2} \mathbf{n}$$

- The coefficient of restitution relates the velocities before and after:

$$(\mathbf{v}'_2 - \mathbf{v}'_1) = \epsilon(\mathbf{v}_2 - \mathbf{v}_1)$$

Impulse

- We solve the equations assuming no rotation¹:

$$\hat{\mathbf{p}} = \hat{p}\mathbf{n} = \frac{(\epsilon + 1)(\mathbf{v}_2 \cdot \mathbf{n} - \mathbf{v}_1 \cdot \mathbf{n})}{\frac{1}{m_1} + \frac{1}{m_2}}\mathbf{n}$$

- If one body is immovable:

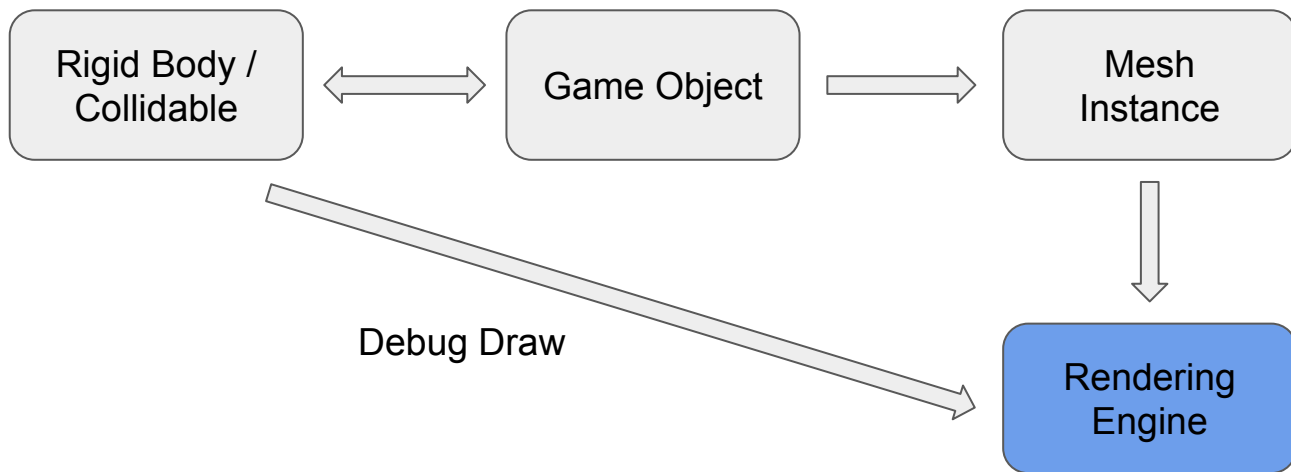
$$\hat{\mathbf{p}} = -(\epsilon + 1)m_1(\mathbf{v}_1 \cdot \mathbf{n})\mathbf{n}$$

- But wait... what about rotation?
 - Things get messier.
 - Calculate velocities of points of contact, rather than centers of mass.
 - Calculate rotational impulse.
 - Beyond the scope of this lecture.
 - Links to resources at the end of these slides.

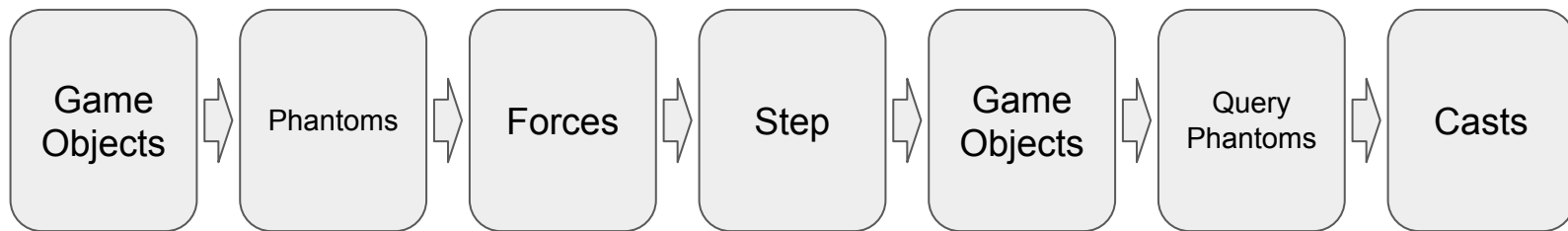
Also not covered...

- Concepts
 - Penalty forces
 - Friction
 - Constraints
- Optimization
 - Rest and sleeping
 - Simulation islands

HOWTO Physics Sim Architecture

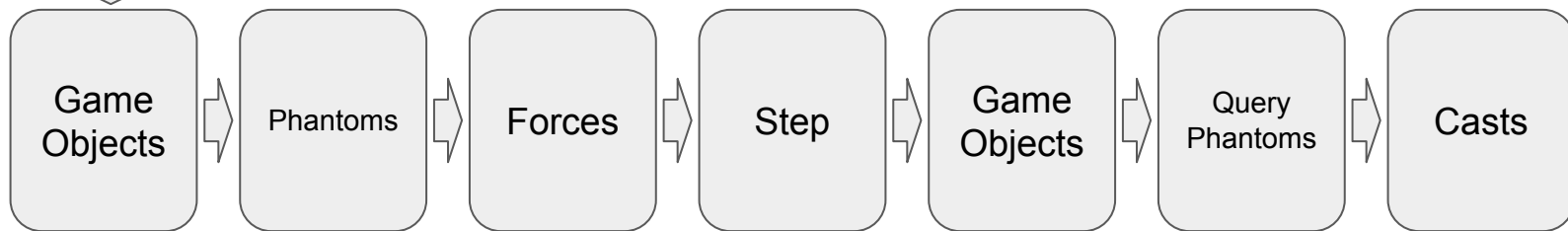


HOWTO Physics Sim Update

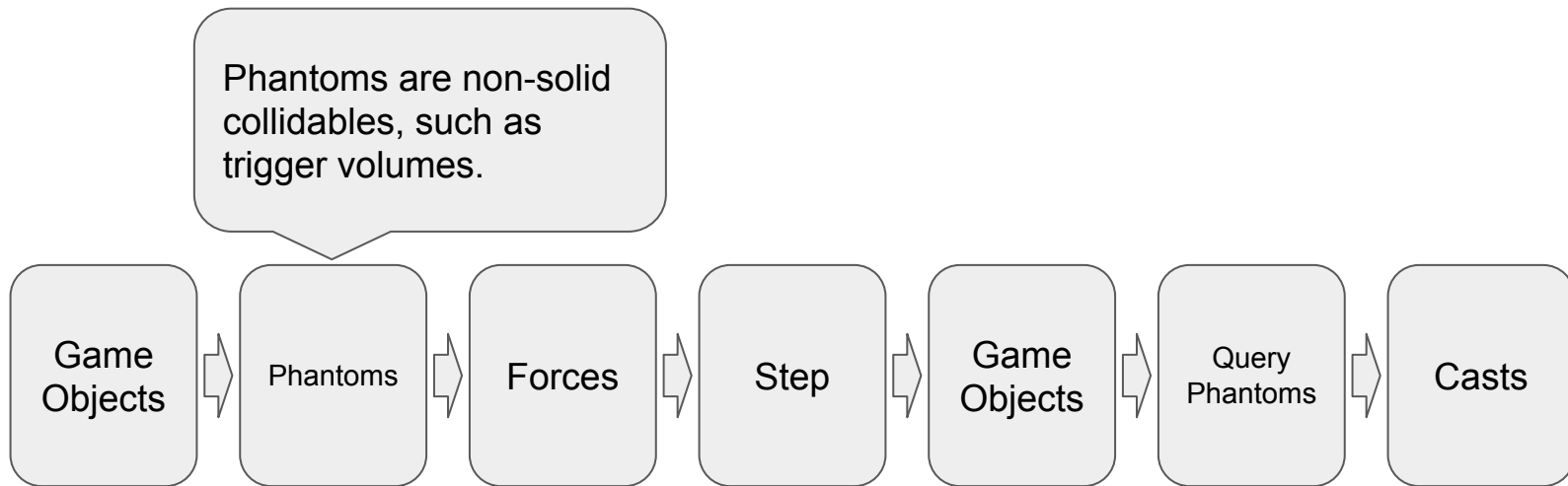


HOWTO Physics Sim Update

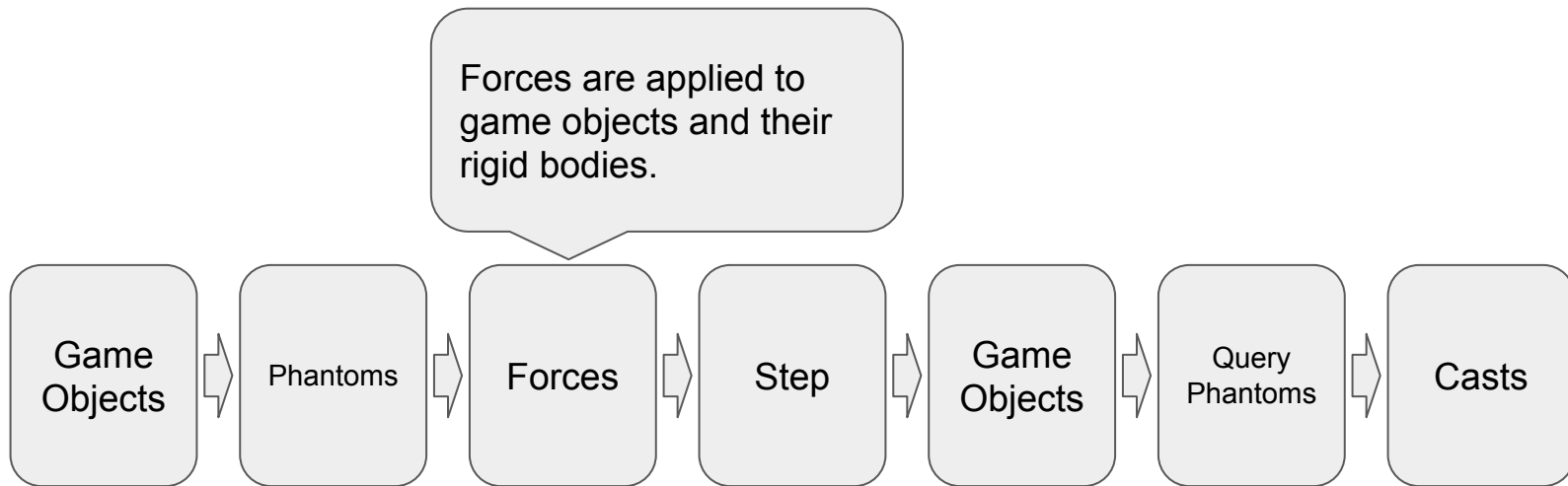
Transforms of game objects are copied to the transforms of their rigid bodies.



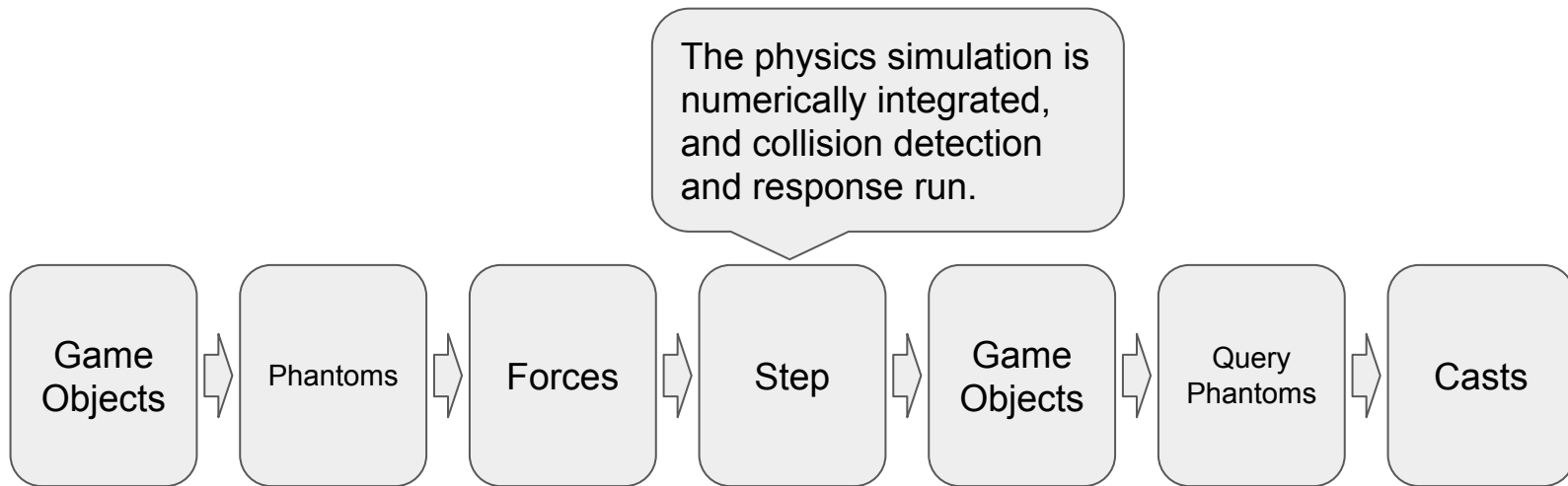
HOWTO Physics Sim Update



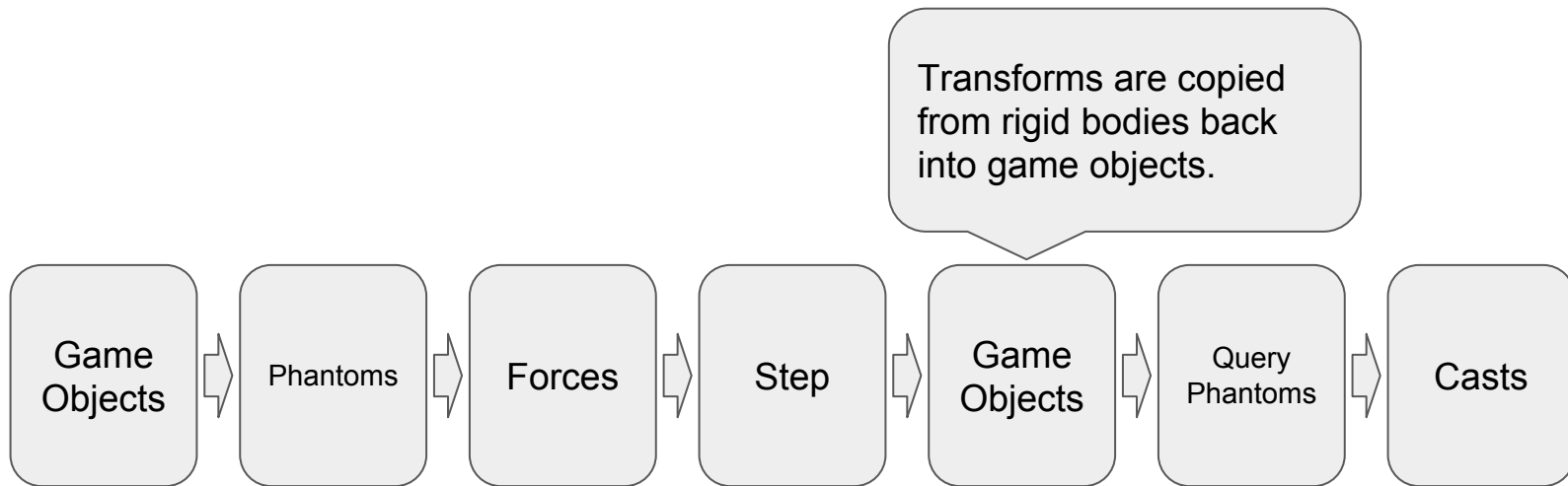
HOWTO Physics Sim Update



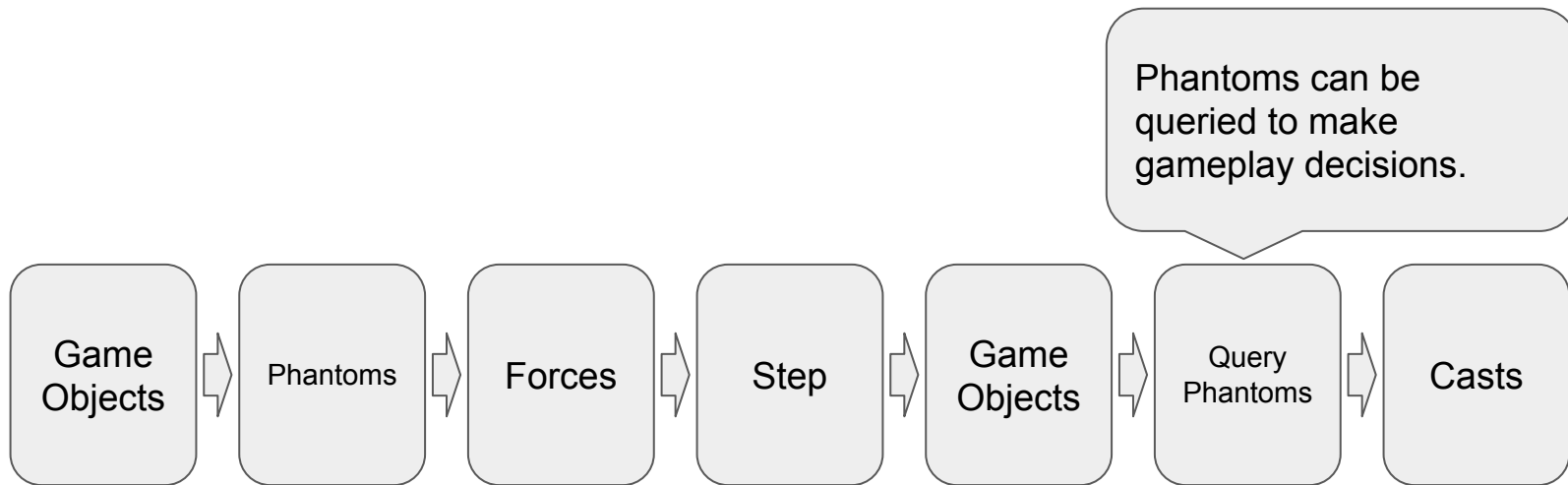
HOWTO Physics Sim Update



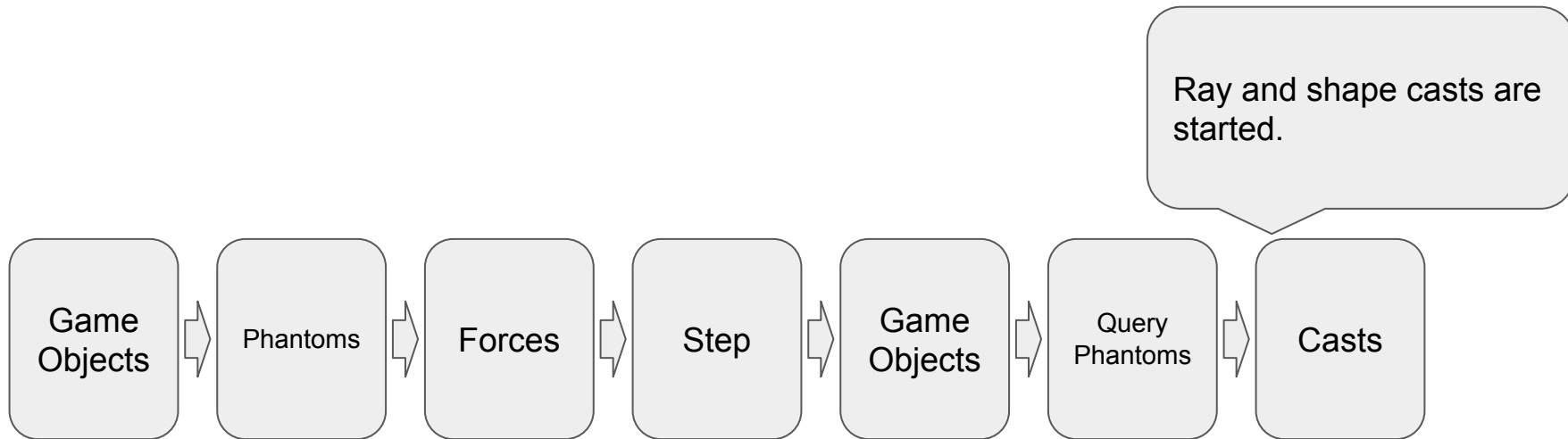
HOWTO Physics Sim Update



HOWTO Physics Sim Update



HOWTO Physics Sim Update



Multithreading

- Several strategies exist for multithreading a physics simulation:
 - A **dedicated thread** can be used to simulate physics.
 - Physics calculations can **fork and join** from the main thread.
 - Physics calculations can be executed as **jobs** on dedicated job threads.
- In all cases, typical thread-safety practices must be employed.

Want to learn more?

- Excellent series of articles by Chris Hecker.
 - http://chrishecker.com/Rigid_body_dynamics
- Browse source for a real physics engine!
 - <http://bulletphysics.org/wordpress/>
 - <https://developer.nvidia.com/physx-source-github>
- Gaffer on Games
 - <http://gafferongames.com/game-physics/>

End of Lecture

- Homework 5 is due Monday, 3/6, at 11:59pm EST.