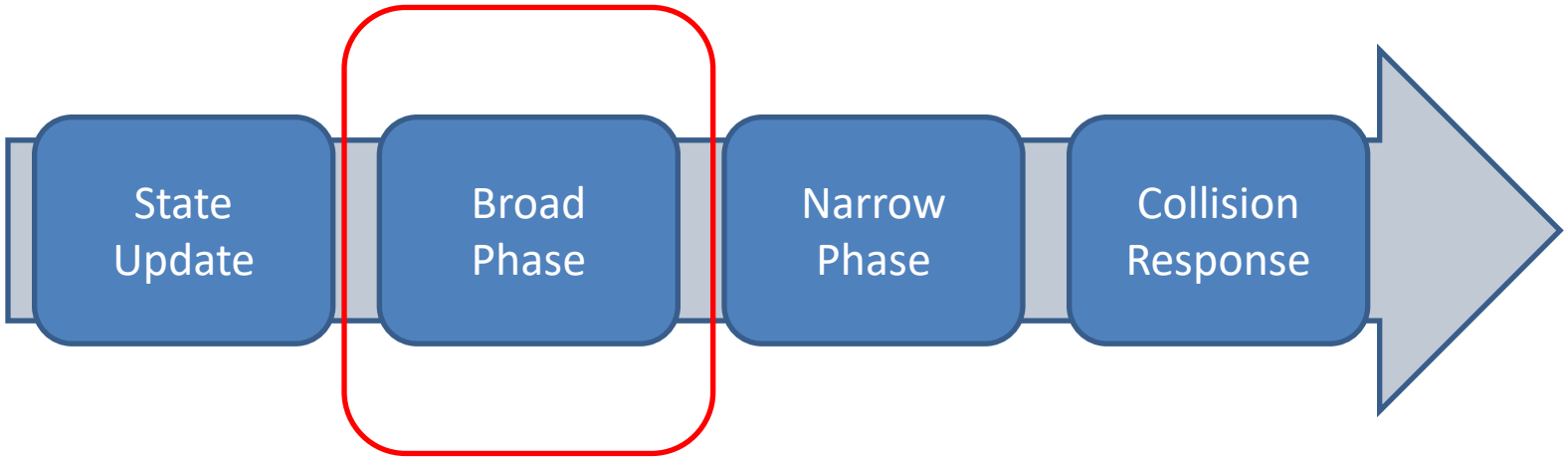
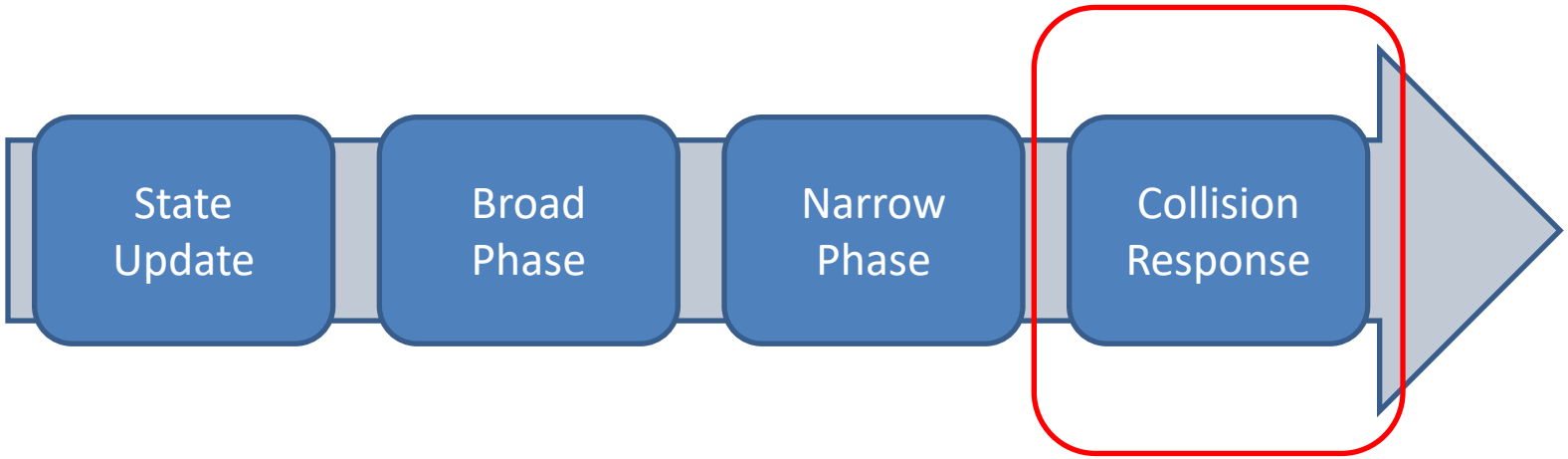
CGP 9

Dr Wojciech Palubicki

Where we were at…



Today…

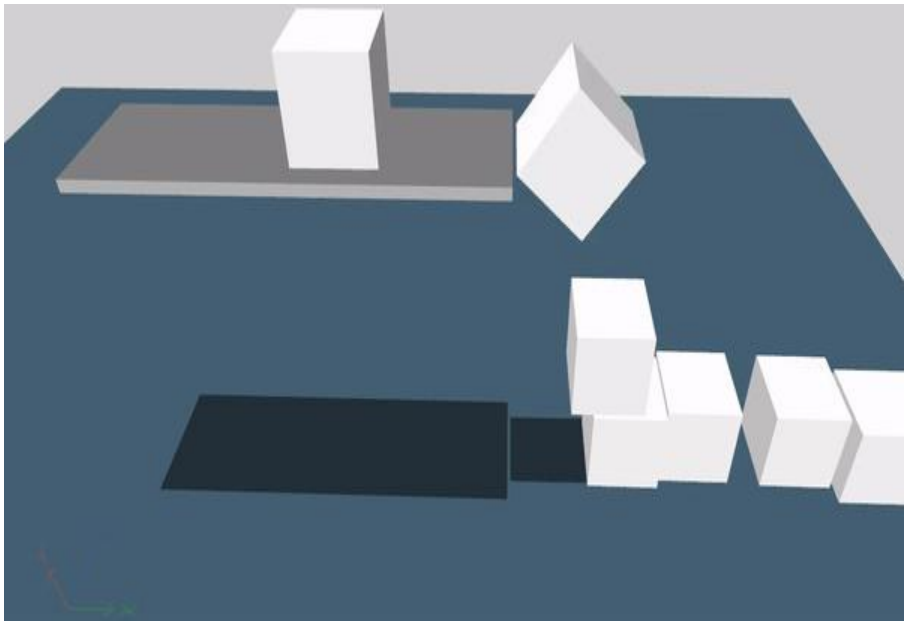


Basic Idea

- Limit motion of particles/ rigid bodies by introducing constraints.

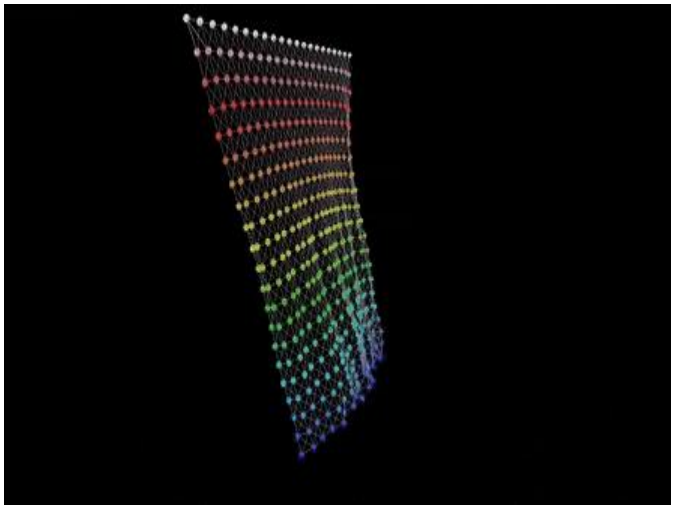
Constraint Types

- Contact and Friction



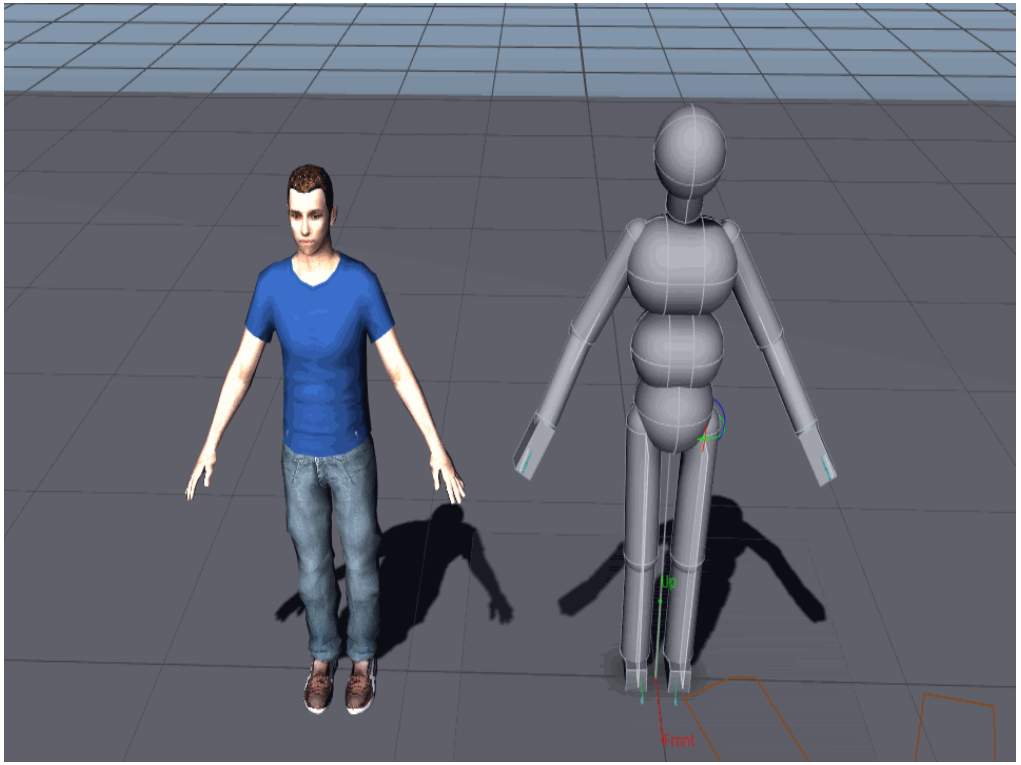
Constraint Types

- Distance: Cloth



Constraint Types

- Joint angle constraints of ragdolls



Collision Response

- Solving physical constraints in addition to the equations of motion
- Constraints are mostly contact constraints, but also
  - Friction constraints
  - Distance constraints
  - Joint angle constraints
  - …
- The ideal collision response system deals with all of these

Naive Take

- Apply the constraints to the objects in pairs
- Use the laws of conservation of motion for each collision; P_0 is the point of collision, x_A and x_B are the centers of mass of the objects, v^- is the pre-impact velocity of the objects, v^+ is the post-impact velocity of the objects

- $f =$

- $r_A = P_0 - x_A, r_B = P_0 - x_B$
- $v_A^+ = v_A^- + fN_0/m_A$
- $\omega_A^+ = \omega_A^- + + I_A^- (r_A \times (fN_0))$
- Push away from interpenetration as long as interpenetration exists

Naive Take

- Apply the constraints to the objects in pairs
- Use the laws of conservation of motion for each collision; P_0 is the point of collision, x_A and x_B are the centers of mass of the objects, v^- is the pre-impact velocity of the objects, v^+ is the post-impact velocity of the objects
- $$f = \frac{-(1+\varepsilon)\left(N_0 \cdot (v_A^- - v_B^-)\right) + (\omega_A^- \cdot (r_A \times N_0) - \omega_B^- \cdot (r_B \times N_0))}{1/m_A + 1/mB + (r_A \times N_0)^T I_A^- (r_A \times N_0) + (r_B \times N_0)^T I_A^- (r_B \times N_0)}$$
- $r_A = P_0 - x_A, r_B = P_0 - x_B$
- $v_A^+ = v_A^- + fN_0/m_A$
- $\omega_A^+ = \omega_A^- + + I_A^- (r_A \times (fN_0))$
- Push away from interpenetration as long as interpenetration exists

Naive Take

- Jitters a lot, and does not support stacking
- May be acceptable in very sparse scenarios (space/flight simulator)

Naive Take 2

- Apply the constraints to the objects in pairs
- Apply again during the same tick
- Average/combine the various impulses
- Push apart objects so they do not penetrate

Naive Take 2

- Apply the constraints to the objects in pairs
- Apply again during the same tick
- Average/combine the various impulses
- Push apart objects so they do not penetrate
- Constraints are still broken
- Slow

Unconstrained Kinematics

- A rigid body is characterized by
- $\dot{x} = v$
- $\dot{q} = 1/2\omega q$

Unconstrained Kinematics

- For a system of N bodies, we can define the system derivative as

$$V = \begin{bmatrix} v_1 \\ \omega_1 \\ \vdots \\ v_n \\ \omega_n \end{bmatrix}$$

Constraints

- Our system allows *pairwise* constraints between bodies
- The k-th constraint, between bodies i and j , has the form

$$C_k\,(x_i, q_i, x_j, q_j) \;=\; 0$$

- The vector C holds all the constraints. $C \;=\; 0$, or $C(x) \;=\; 0$, is a function of the state vector, so by the chain rule

$$\dot{C} = \; JV \;=\; 0$$

Constraint forces

- Each constraint causes a reaction force f_c and a reaction torque τ_c
- The vector of all reaction forces is

$$F_c = \begin{bmatrix} f_{c_1} \\ \tau_{c_1} \\ \vdots \\ f_{c_n} \\ \tau_{c_n} \end{bmatrix}$$

Constraint forces

We know that $\dot{C} = JV = 0$

- $JV = \begin{bmatrix} J_1 \cdot V \\ \vdots \\ J_m \cdot V \end{bmatrix} = 0$

Constraint forces

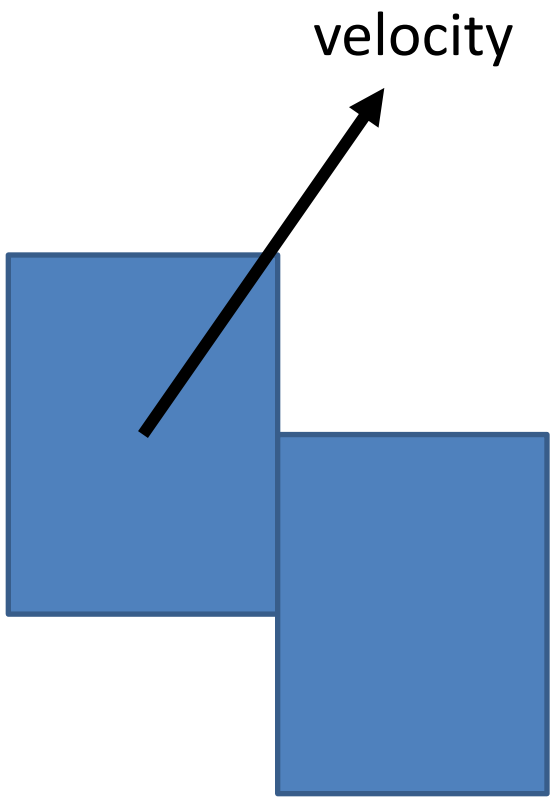
We know that $\dot{C} = JV = 0$

- $JV = \begin{bmatrix} J_1 \cdot V \\ \vdots \\ J_m \cdot V \end{bmatrix} = 0$
- This means that V is orthogonal to each row of J

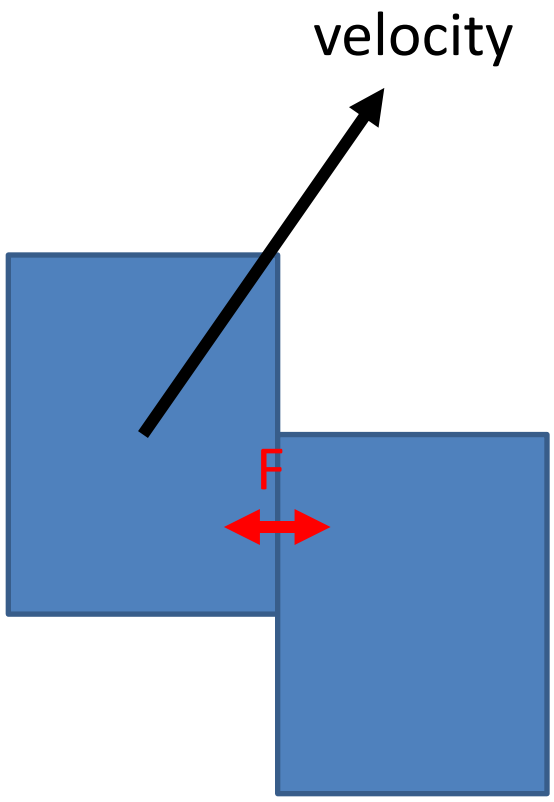
Constraint forces

- Constraint forces perform no work, so $F_c \cdot V = 0$

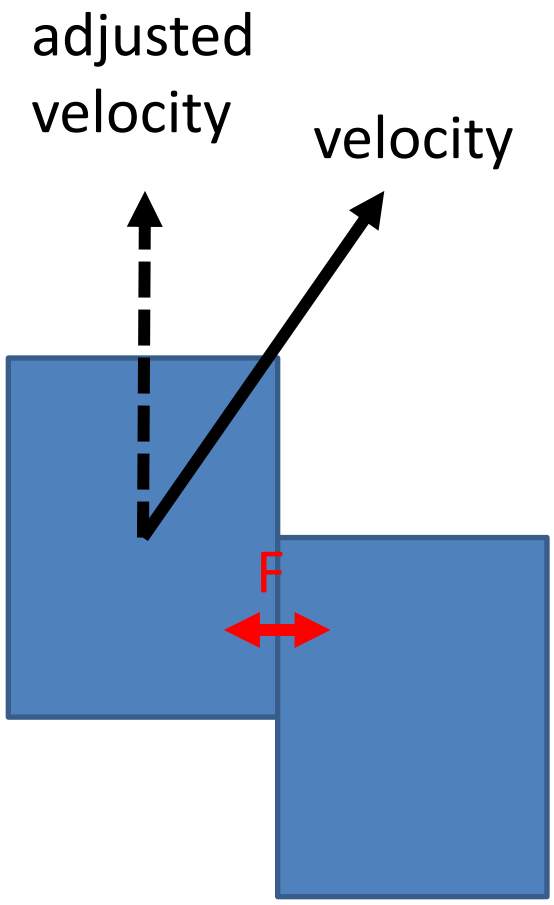
Contact Constraint



Contact Constraint

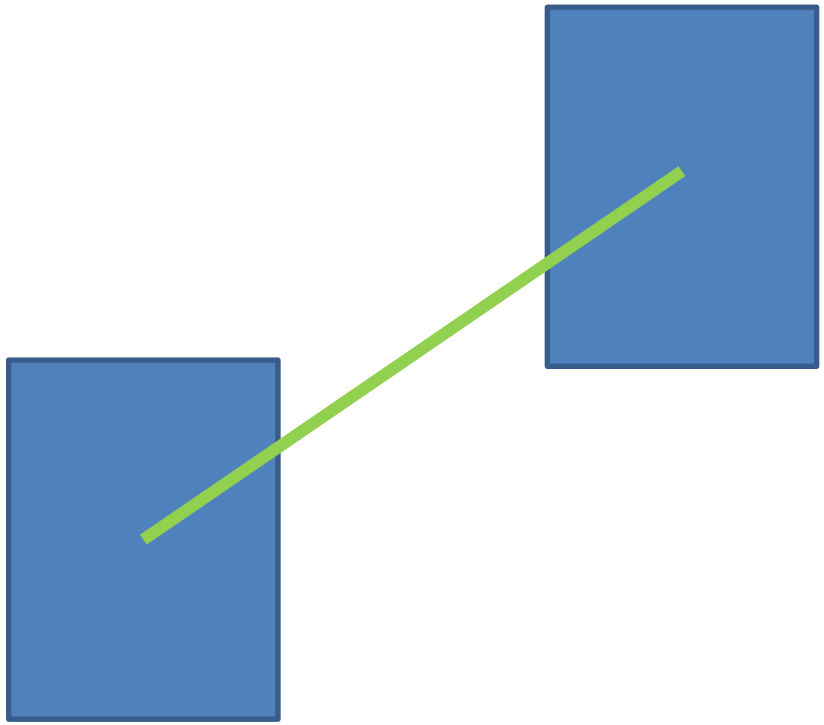


Contact Constraint

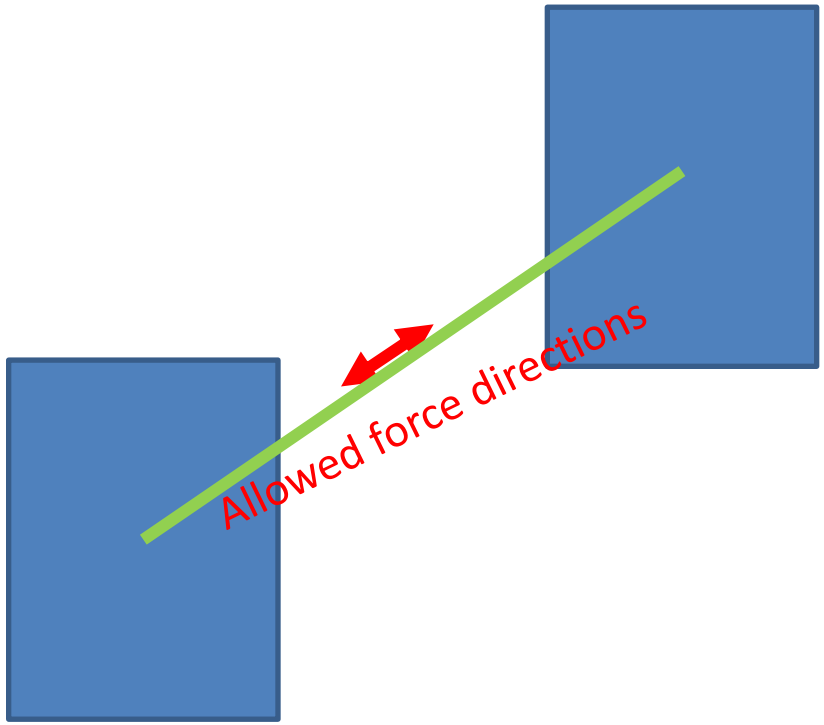


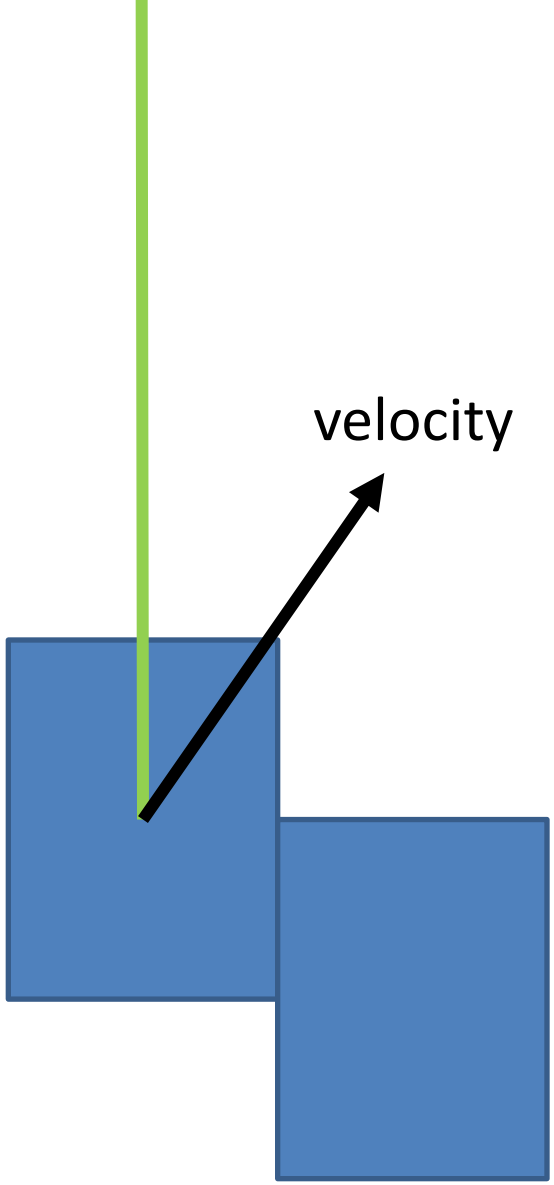
$$F_c \cdot V = 0$$

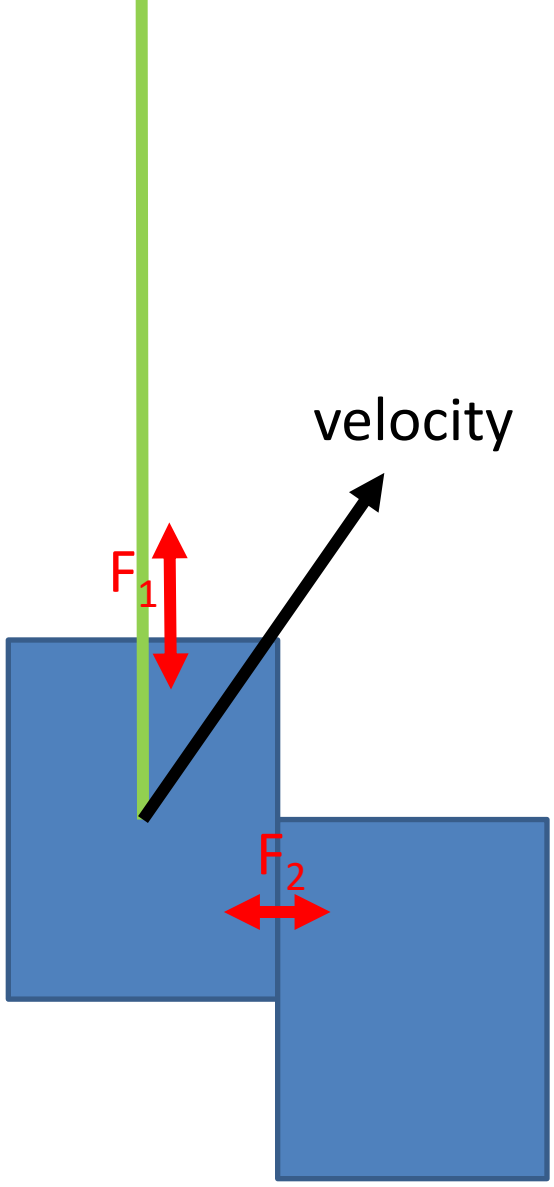
Distance Constraint

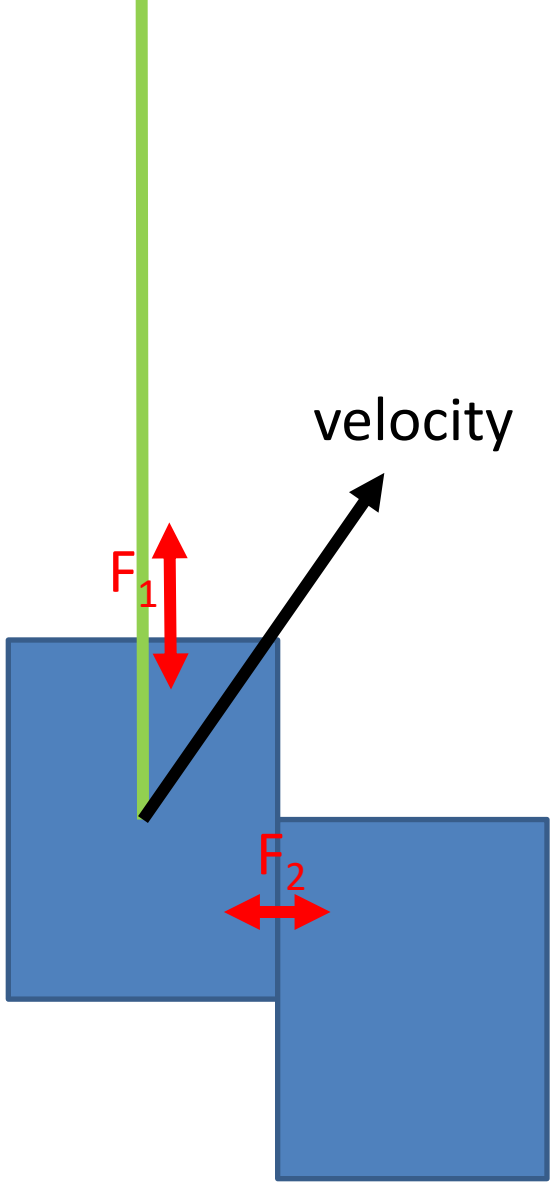


Distance Constraint




velocity


velocity
F1
F2



blend forces:

$$F_1\lambda_1 + F_2\lambda_2$$

Constraint forces

- We can use $F_c = J^T \lambda$ for some vector λ of undetermined (Lagrangian) force multipliers

$$F_c \cdot V = J^T \lambda \cdot V = \left(\sum_i J_i \lambda_i \right) \cdot V = \sum_i J_i \cdot V \lambda_i$$

$$= \sum_i 0 \cdot \lambda_i = 0$$

Constraint forces

- We compute the matrix J of constraints from the collision system
- We then solve for λ, compute F_c, and finally obtain V

Distance constraints

- The simplest constraint is a distance constraint
- Two points of two bodies must remain at a given distance
- $C\ (x_i, q_i, x_j, q_j)\ =\ \frac{1}{2}(|\mathrm{p_j} - \mathrm{p_i}|^2\ -\ L^2)\ =\ 0$
- If we derive this, we get $(d = |p_j\ -\ p_i|)$

$$\dot{C}(x_i, q_i, x_j, q_j) = d \cdot (v_j + \underbrace{\omega_j \times r_j - v_i}_{\text{a row of J}} - \underbrace{\omega_i \times}_{\text{some columns of V}} r_i)$$

- We split this into a row for J and a part of V:

$$\dot{C}(x_i, q_i, x_j, q_j) = [-d^T - (r_i \times d)^T\ d^T (r_j \times d)^T]$$

Distance constraints

- Note that a row of J also contains many zeroes $(6 \times (N_{bodies} - 2))$
- The only columns that are not zeroed are those corresponding to the bodies i and j
- $\dot{C}(x_i, q_i, x_j, q_j) =$
  $[\cdots 0 \; -d^T \; -(r_i \times d)^T \; 0 \cdots 0 \; d^T \; (r_j \times d)^T \cdots 0]V$
  (a row of J)

Contact constraints

- The contact constraint measures the object separation; it is negative in case of overlap
- $C(x_i, q_i, x_j, q_j) = (x_j + r_j - xi - r_i) \cdot n_i = 0$
- $\dot{C}(x_i, q_i, x_j, q_j) =$

$(v_j + \omega_j \times r_j - v_i - \omega_i \times r_i) \cdot n_i + (x_j + r_j - x_i - r_i) \cdot \omega_i \times n_i$

- We assume that both penetration and angular velocity are small, so we ignore the second term

Contact constraints

- We can now separate $\dot{C}$ into J and V:
- $\dot{C}(x_i, q_i, x_j, q_j) =$
  $(v_j + \omega_j \times r_j - v_i - \omega_i \times r_i) \cdot ni =$
  $[-n_i^T - (r_i \times n_i)^T \, n_i^T \, (r_j \times n_i)^T] \, [v_i \, \omega_i \, v_j \, \omega_j]^T$

Contact constraints

- Notice that the force between bodies in contact can push them apart, but not pull them together
- This means that $0 \leq \lambda_k \leq +\infty$, where k is the constraint index for a contact constraint

Contact constraints

- Due to numerical errors or issues with discrete steps penetration might happen anyway
- We allow the velocity to be modulated by a *pushing factor* which is proportional to the penetration
- This means that for contact constraints

  $J_i V = -\beta C_i$, for $\beta \leq \frac{1}{\Delta t}$

Friction constraints

- Friction constraints are very similar to contact constraints
- Friction happens along the tangent plane, so we have two constraints (one for u_i = T and one for u_j = B)
- $\dot{C}(x_i, q_i, x_j, q_j) =$
  $(v_j + \omega_j \times r_j - vi - \omega_i \times r_i) \cdot u_i =$
  $[-u_i^T - (r_i \times u_i)^T \, u_i^T \, (r_j \times u_i)^T] \, [v_i \, \omega_i \, v_j \, \omega_j]^T$
- $\dot{C}(x_i, q_i, x_j, q_j) =$
  $(v_j + \omega_j \times r_j - vi - \omega_i \times r_i) \cdot u_i =$

Friction constraints

- We must also bound the friction value (this is an approximation) to take the friction coefficient into account
- $-\mu m_c g \leq \lambda_{u1} \leq \ \mu m_c g$ and $-\mu m_c g \ \leq \lambda_{u2} \ \leq \mu m_c g$, where m_c is the mass assigned to the contact point

Equations of motion

- We now integrate our constraint system with the equations of motion
- We know the Newton - Euler equations of motion are
- $m\dot{v} = F = f_c + f_{ext}$
- $I\dot{\omega} = \tau = \tau_c + \tau_{ext}$

Equations of motion

- We can define a single matrix for all the bodies

$$M = \begin{pmatrix} m_1 E_{3x3} & 0 & \cdots & 0 & 0 \\ 0 & I_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & m_n E_{3x3} & 0 \\ 0 & \cdots & 0 & 0 & I_n \end{pmatrix}$$

- E_{3x3} is the identity matrix

Equations of motion

- We can easily invert this matrix

$$M^{-1} = \begin{pmatrix} (m_1 E_{3x3})^{-1} & 0 & \cdots & 0 & 0 \\ 0 & I_1^{-1} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & (m_n E_{3x3})^{-1} & 0 \\ 0 & \cdots & 0 & 0 & I_n^{-1} \end{pmatrix}$$

Equations of motion

- We define a single vector for all the external forces

- $$F_{ext} = \begin{bmatrix} f_{ext_1} \\ \tau_{ext_1} \\ \vdots \\ f_{ext_n} \\ \tau_{ext_n} \end{bmatrix}$$

Equations of motion

- Since we know that $F_C = J^T\lambda$, we can rewrite the equations of motion for n bodies as

$$\begin{cases} M\dot{V} = J^T\lambda + F_{ext} \\ JV = \varepsilon \end{cases}$$

- ε is the vector of force offsets which allows contact forces to perform work

Equations of motion

- Since we know that $F_C = J^T\lambda$, we can rewrite the equations of motion for n bodies as

$$\begin{cases} M\dot{V} = J^T\lambda + F_{ext} \\ JV = \varepsilon \end{cases}$$

- ε is the vector of force offsets which allows contact forces to perform work
- We have too many unknowns: V, $\dot{V}$, and λ

Equations of motion

- We approximate $\dot{V} \approx \frac{V_2 - V_1}{\Delta t}$

Equations of motion

- We approximate $\dot{V} \approx \frac{V_2 - V_1}{\Delta t}$

- We replace $\dot{V}$ $\begin{cases} M \frac{V_2 - V_1}{\Delta t} = J^T \lambda + F e_{xt} \\ J V_2 = \varepsilon \end{cases}$

Equations of motion

- We approximate $\dot{V} \approx \frac{V_2 - V_1}{\Delta t}$

- We replace $\dot{V}$ $\begin{cases} M\frac{V_2 - V_1}{\Delta t} = J^T\lambda + Fe_{xt} \\ JV_2 = \varepsilon \end{cases}$

- We solve for

V_2 $\begin{cases} V_2 = \Delta t M^{-1}(J^T\lambda + Fe_{xt}) + V_1 \\ V_2 = J^T\varepsilon \end{cases}$

Equations of motion

- We can now finish solving for λ
- $J^T\varepsilon = \Delta t M^{-1}(J^T\lambda + Fe_{xt}) + V_1$
- $J^T\varepsilon - V_1 - \Delta t M^{-1}F_{ext} = \Delta t M^{-1}(J^T\lambda)$
- $\varepsilon - JV_1 - \Delta t J M^{-1}F_{ext} = \Delta t J M^{-1}J^T\lambda$

Equations of motion

- Once λ is computed, we can determine F_c , F, and then V_2
- A regular integration step is then performed with the new velocities V_2

Iterative Solution

- The equation can be restated in simpler form:

- $\underbrace{\frac{\varepsilon}{\Delta t} - \frac{JV_1}{\Delta t} - JM^{-1}F_{ext}}_{b} = \underbrace{JM^{-1}J^T}_{A}\underbrace{\lambda}_{x}$

- $Ax = b$ for some A, b
- These systems can be solved iteratively with a method such as Projected Gauss-Seidel (PGS)

Projected Gauss-Seidel

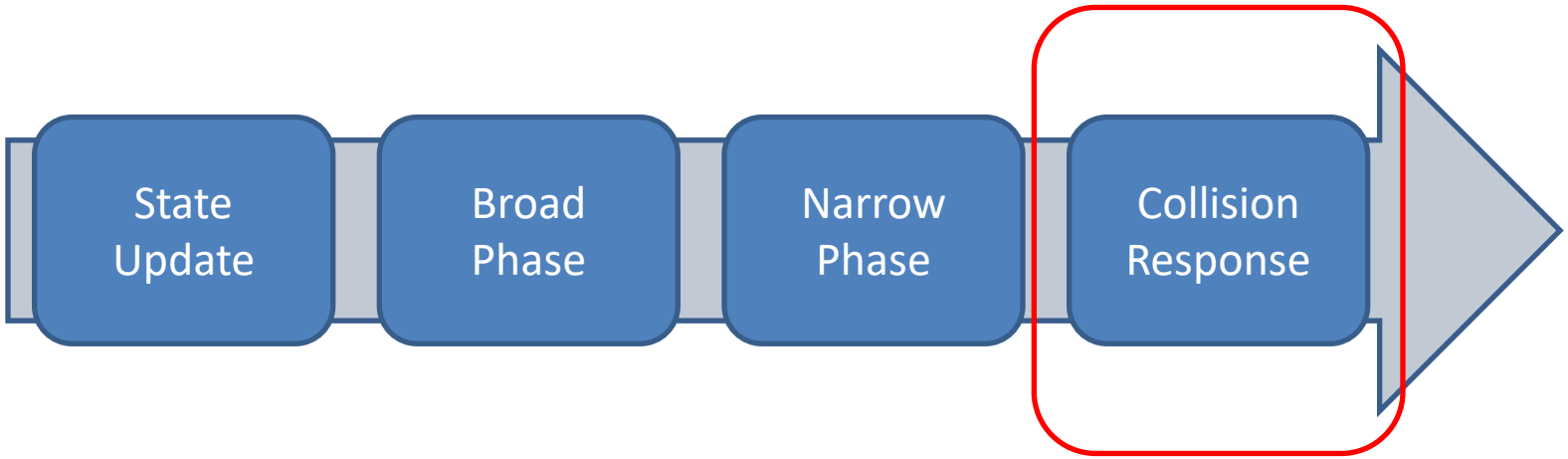
- `while not converged`
  - $\Delta x_i = (b_i - \sum_j a_{ij} x_j)/A_{ii}$
  - $x_i = \texttt{clamp}(x_i + \Delta x_i, min_i, max_i)$

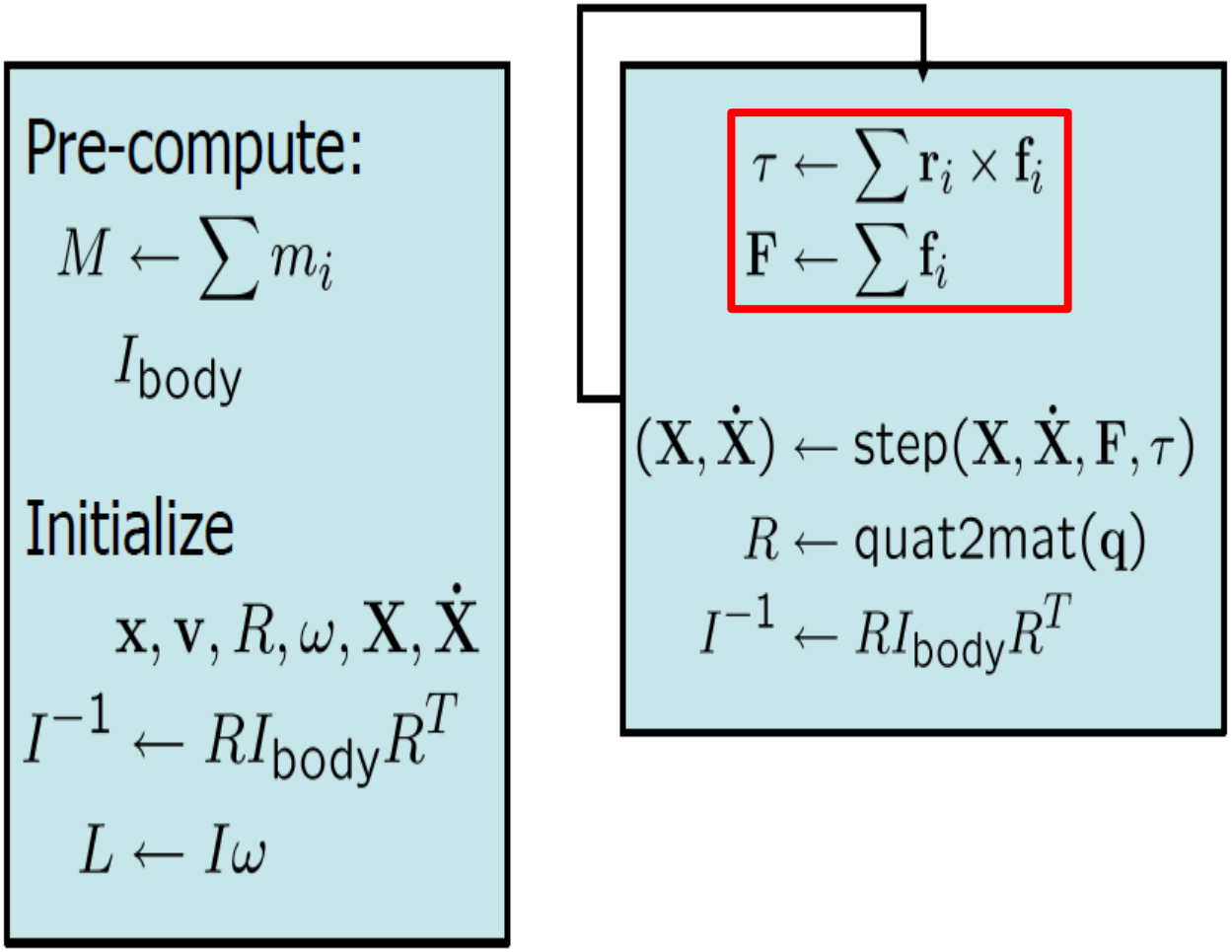
Iterative Solution

- A is going to be very **sparse**
  - You may optimize the summation $\Delta x_i = (b_i - \sum_j a_{ij} x_j)/A_{ii}$ a lot by ignoring the zero entries of A

Contact caching (warm start)

- PGS is faster the closer the initial x vector is to the solution
- If we store previous contact points and their λ_i values, PGS converges sooner


State
Update
Broad
Phase
Narrow
Phase
Collision
Response


Pre-compute:
$M \leftarrow \sum m_i$
I_{body}
Initialize
$\mathbf{x}, \mathbf{v}, R, \omega, \mathbf{X}, \dot{\mathbf{X}}$
$I^{-1} \leftarrow R I_{\text{body}} R^T$
$L \leftarrow I\omega$
$\tau \leftarrow \sum \mathbf{r}_i \times \mathbf{f}_i$
$\mathbf{F} \leftarrow \sum \mathbf{f}_i$
$(\mathbf{X}, \dot{\mathbf{X}}) \leftarrow \text{step}(\mathbf{X}, \dot{\mathbf{X}}, \mathbf{F}, \tau)$
$R \leftarrow \text{quat2mat}(\mathbf{q})$
$I^{-1} \leftarrow R I_{\text{body}} R^T$

Physics Libraries

- PhysX, *Box2D*, *Chipmunk Physics* and *Bullet Physics*