INERTIALIZATION: HIGH-PERFORMANCE ANIMATION TRANSITIONS IN GEARS OF WAR

DAVID BOLLO
PRINCIPAL SOFTWARE ENGINEER
THE COALITION (MICROSOFT STUDIOS)

Animation Transitions

Animation Transitions

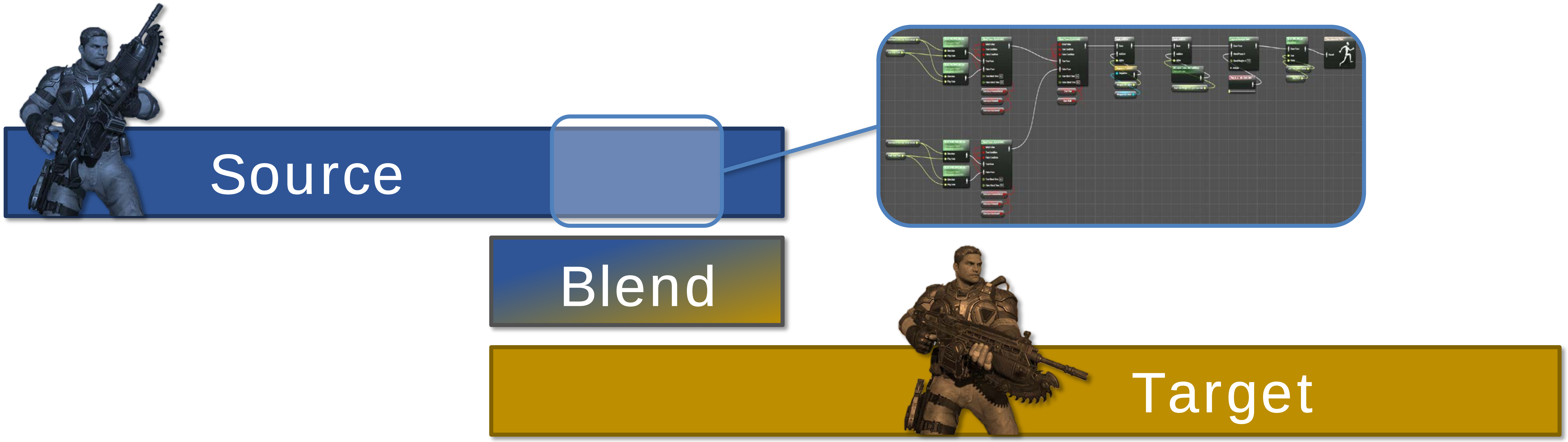
- Transition from one animation state to another
- Typically a cross-fade blend between poses

Animation Transitions

- Optimizations are often focused on the blend step
- Fast SLERP, optimizing cache and memory usage, etc

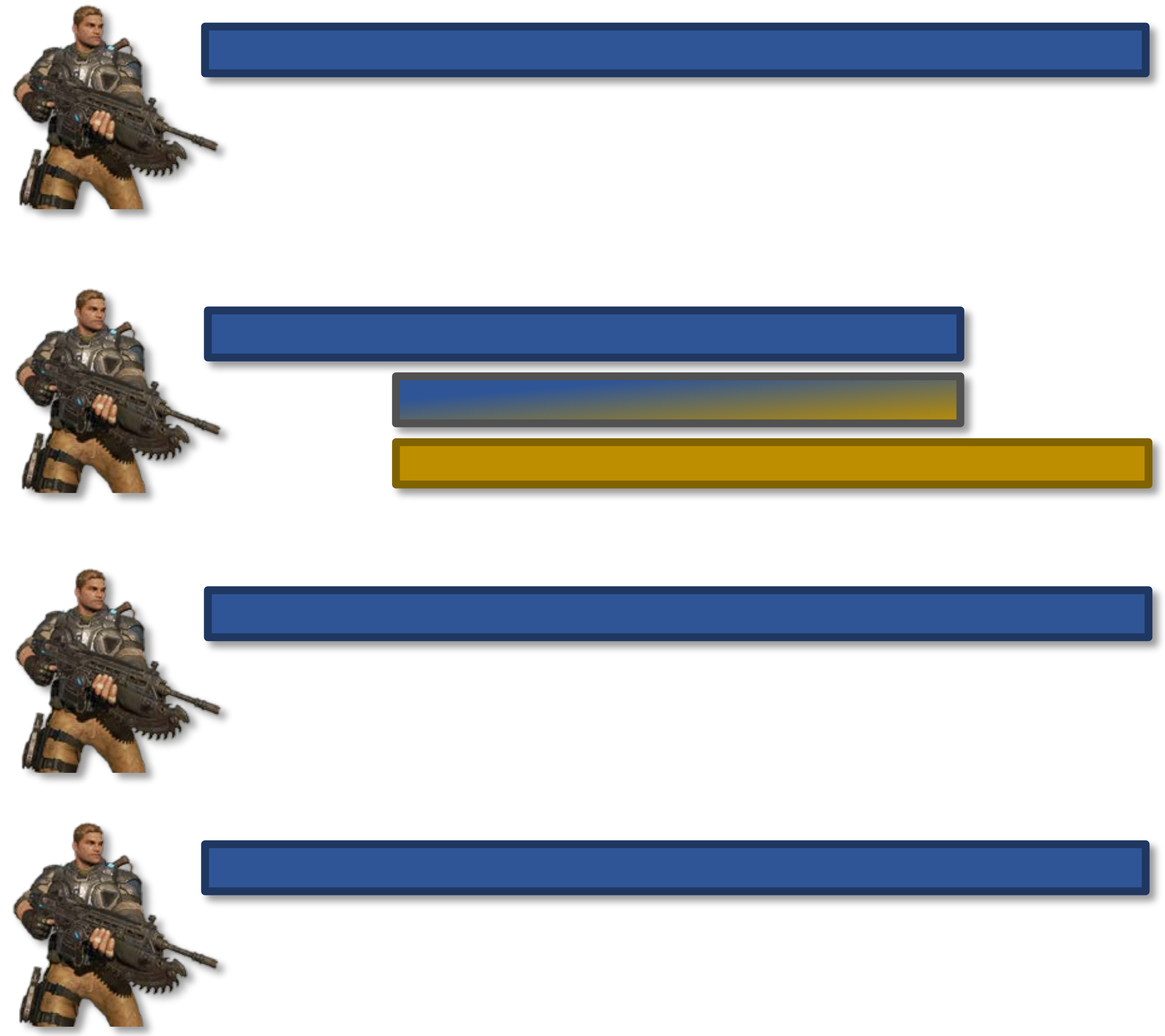
ANIMATION TRANSITIONS

- Biggest cost is evaluating both Source and Target
- Source/Target cost is much greater than blend cost



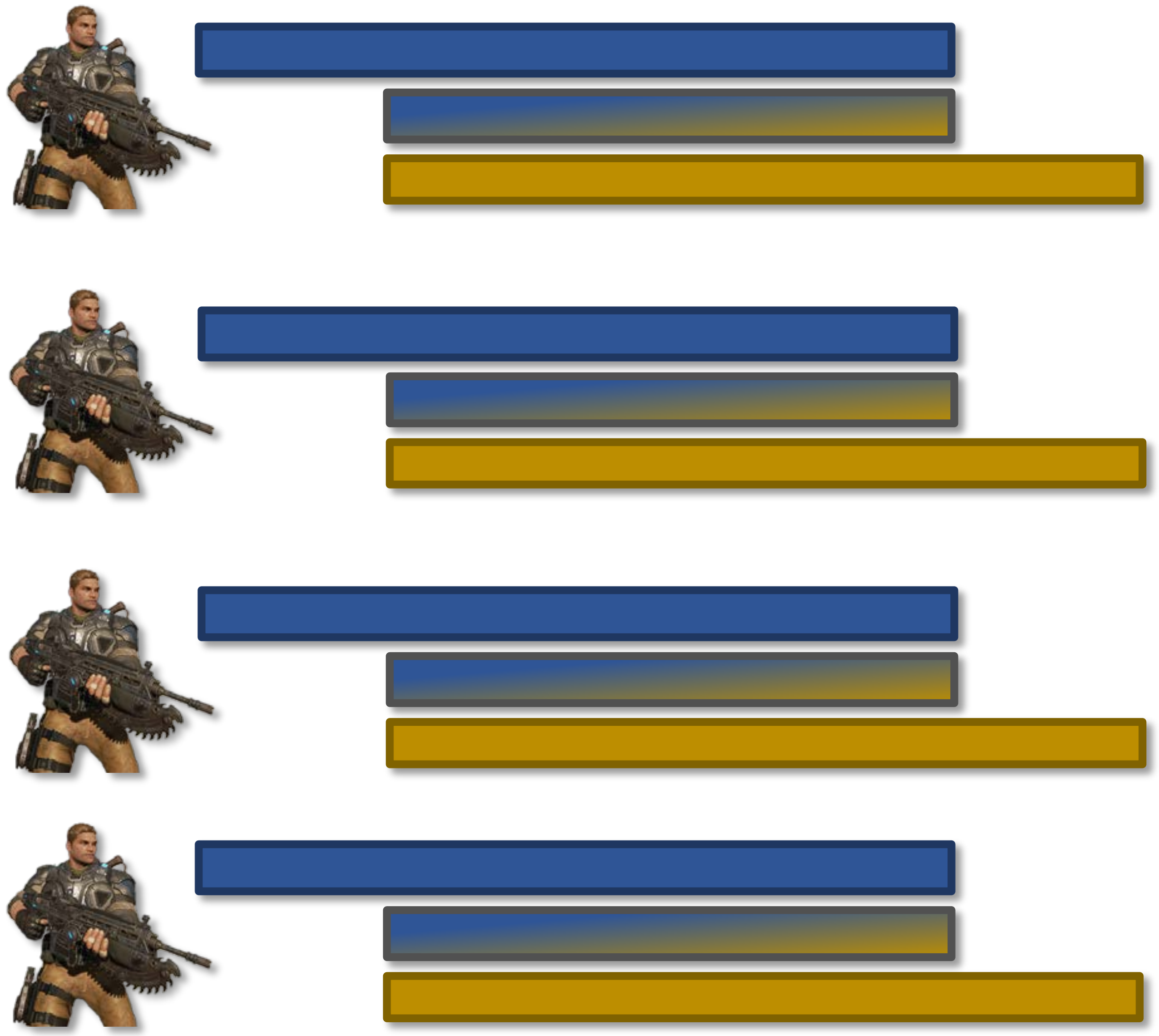
MULTIPLE CHARACTERS

- If we're lucky…
- Only a few active transitions at once



Multiple Characters

- But in the worst case…
- Everybody transitions at the same time
- Double the animation cost

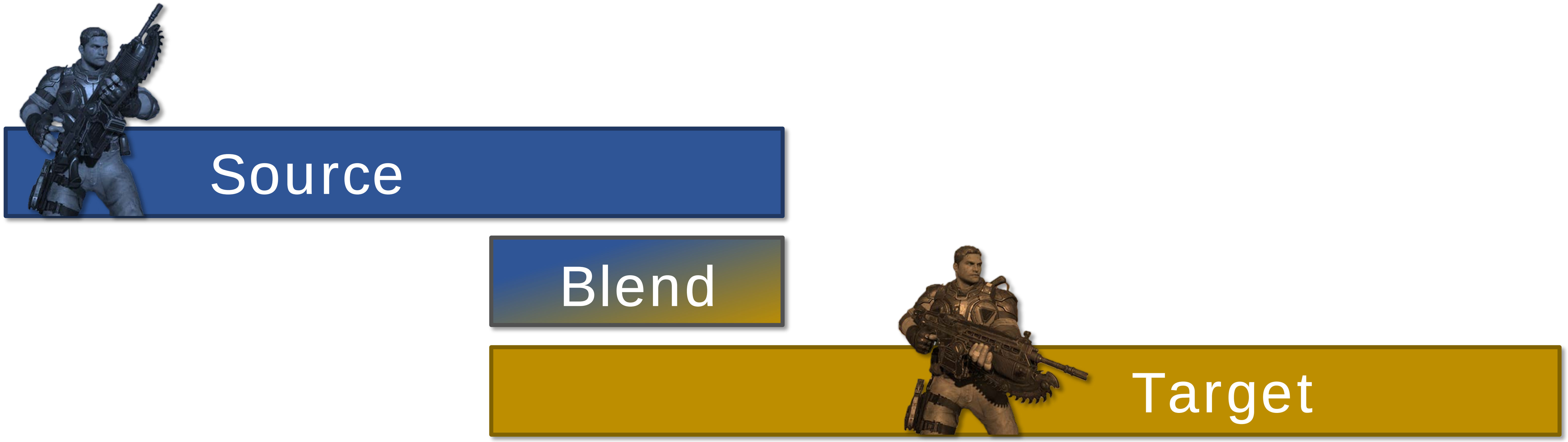


Can We Do Better?

- Intuition: Real humans don't "blend"
- (but they do have inertia)

CAN WE DO BETTER?

- IDEA: Eliminate blended transitions!
- Fix the discontinuities as a post-process

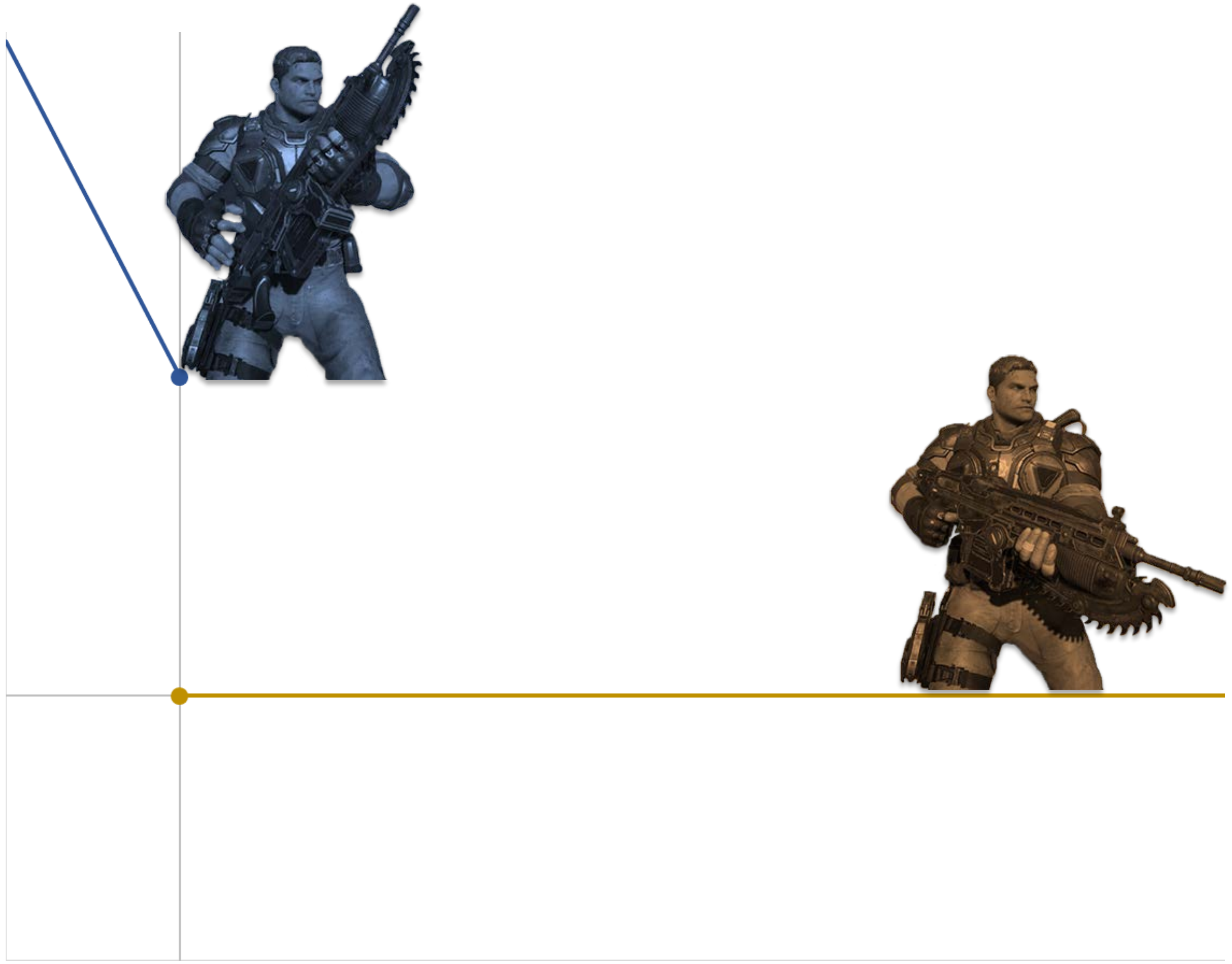


Can We Do Better?

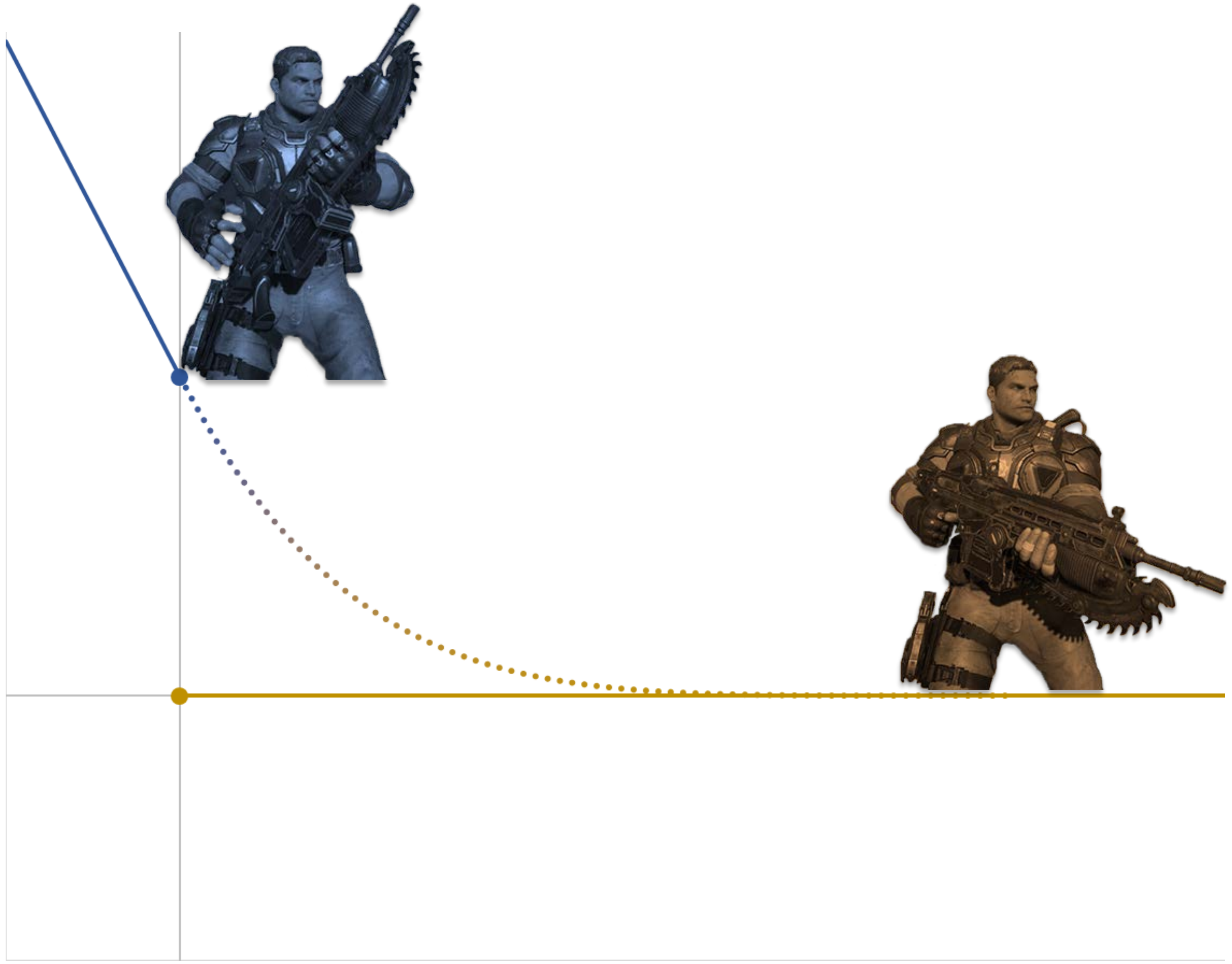
- IDEA: Eliminate blended transitions!
- Fix the discontinuities as a post-process

Transitions as a Post-Process

Transitions as a Post-Process



Transitions as a Post-Process



Goals

- **Respect** the original animation
  - No changes when not transitioning
- **Believable** and aesthetically pleasing
  - Smooth and momentum-preserving
- **Stay on-model**
  - No bad / unnatural poses
  - No overshoot

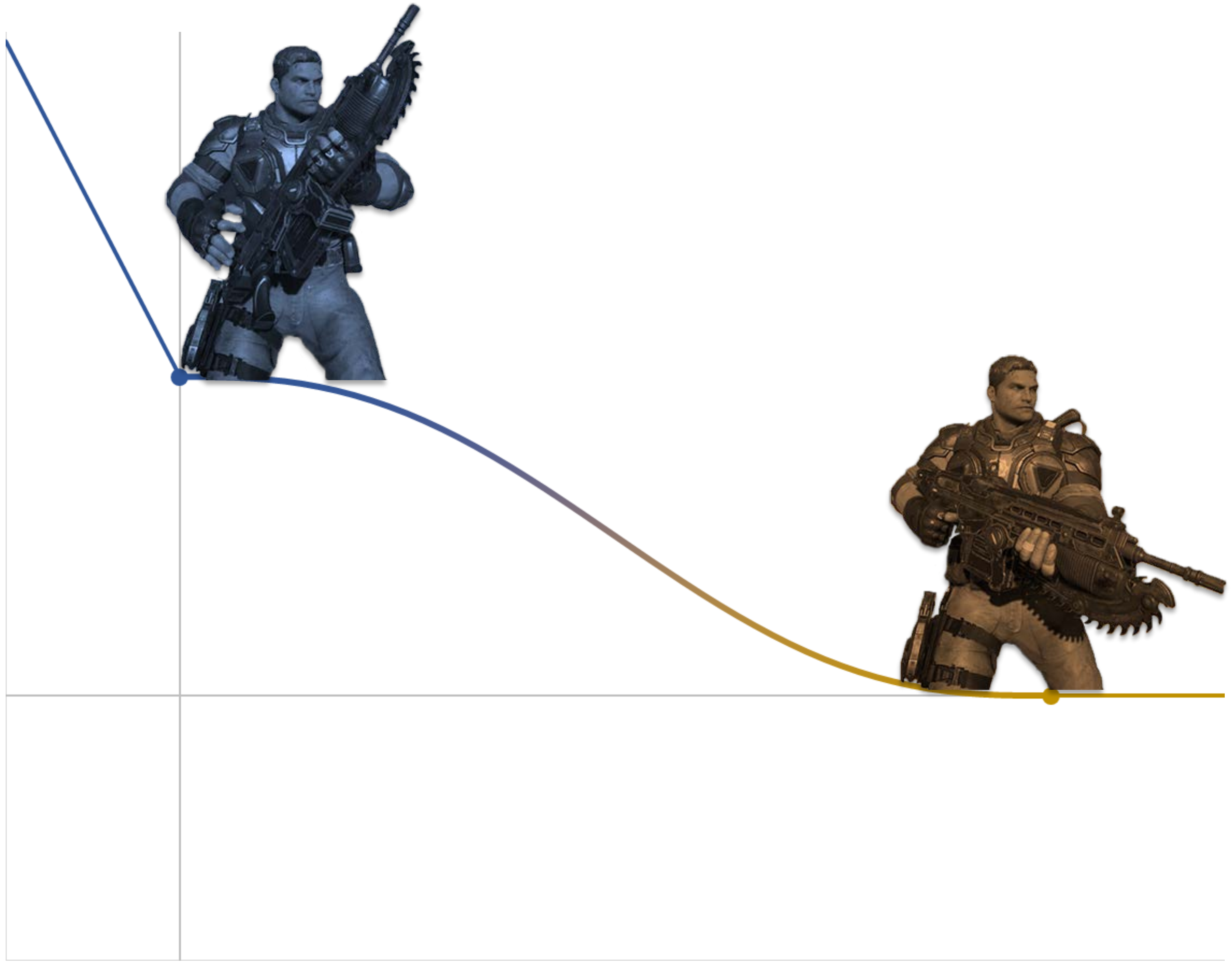
Idea #0: Filter Discontinuous Pose

- Apply a filter to the output pose stream
- Difficult to tune
- Introduces lag
- Deviates from input even when not transitioning

Idea #1: Blend From Pose

- Capture pose difference between Source and Target
- Ease out the difference over time

Idea #1: Blend From Pose

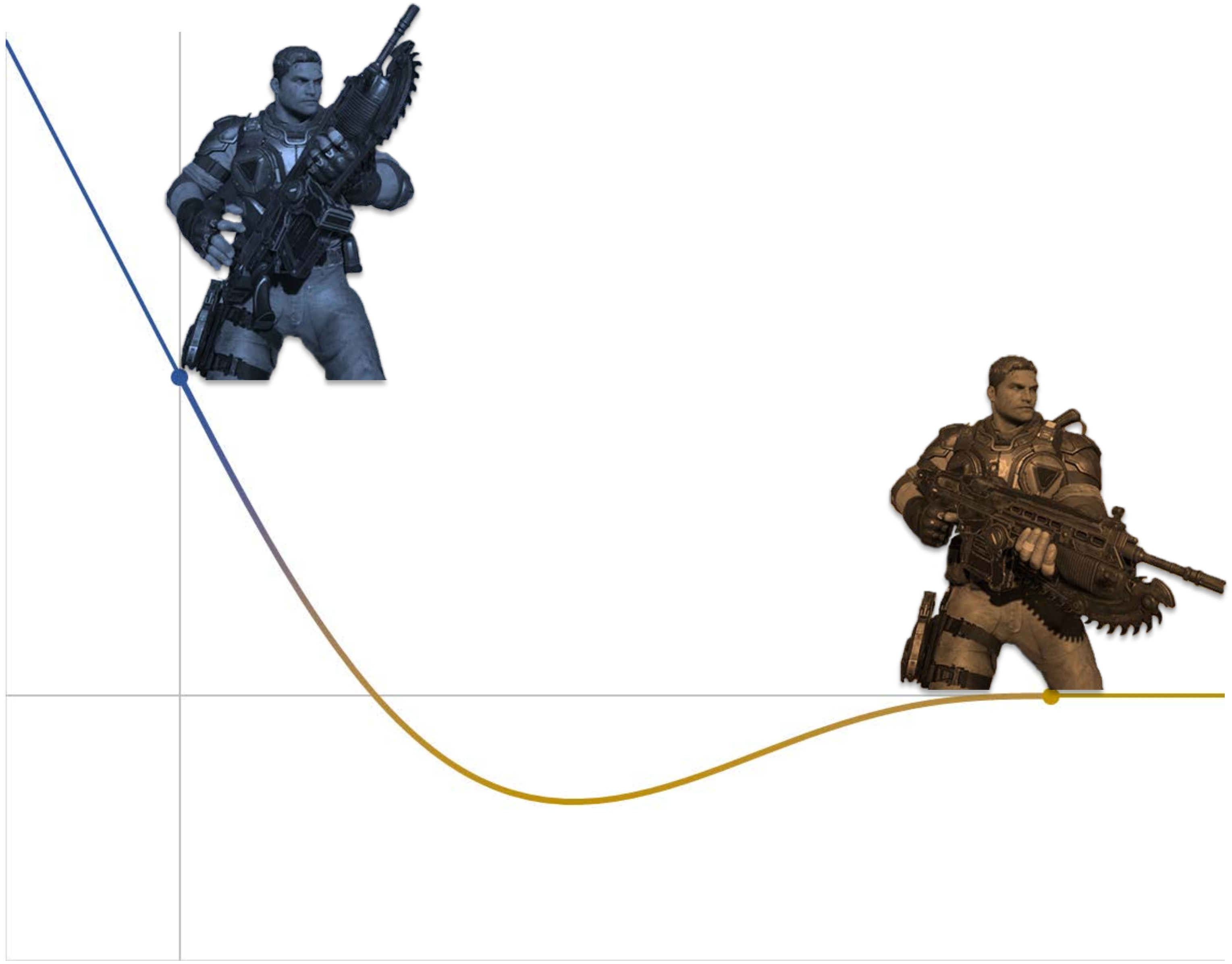


Idea #2: Match Velocity

- Capture pose difference between Source and Target
- Ease out the difference over time
- Remember Source velocity (via finite differences)
- Match initial velocity
- Quintic polynomial [Flash and Hogan 1985]

T. Flash and N. Hogan. 1985. The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model. *Journal of Neuroscience* 5, 7 (July 1985), 1688 – 1703

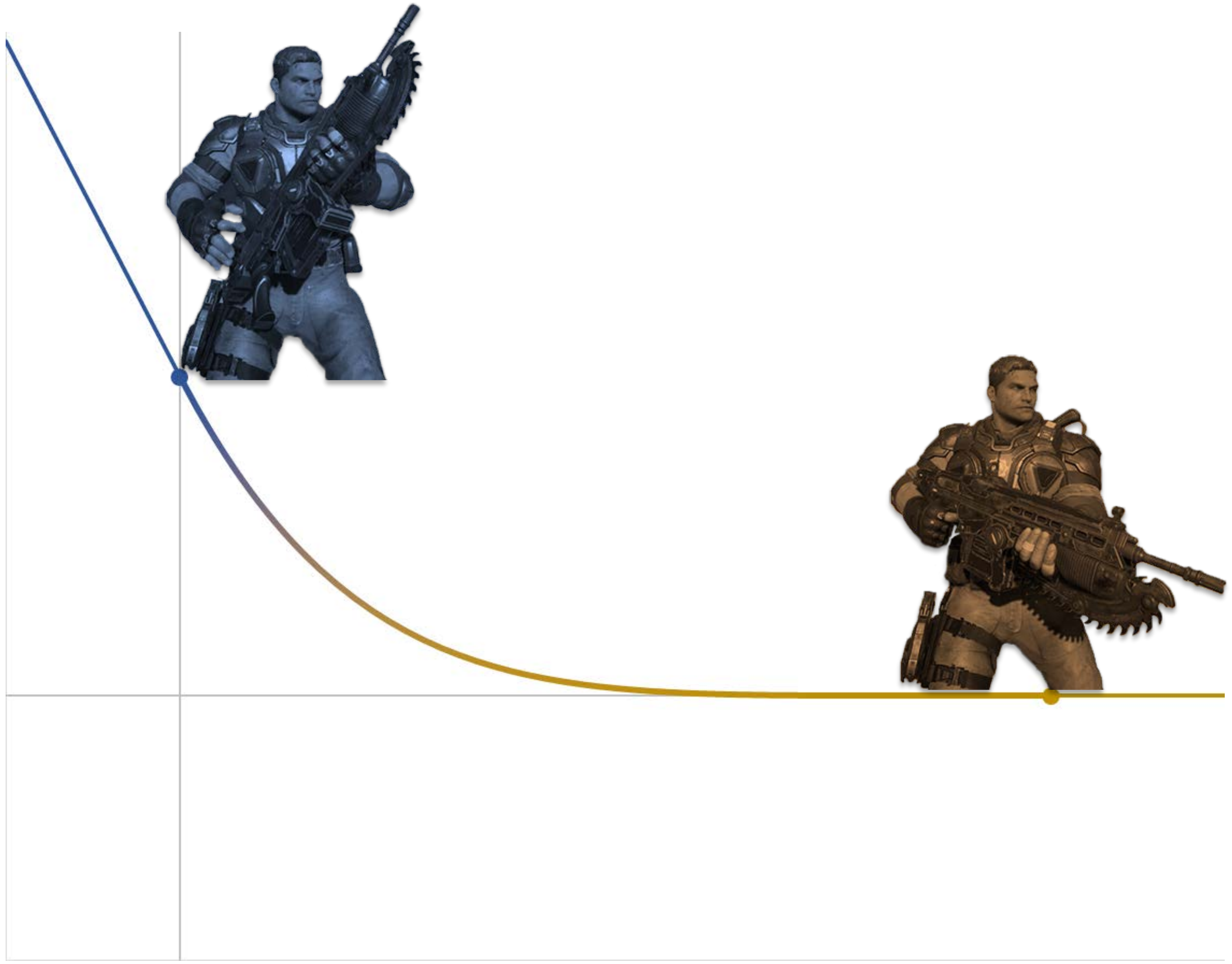
Idea #2: Match Velocity



Idea #3: Limit Overshoot

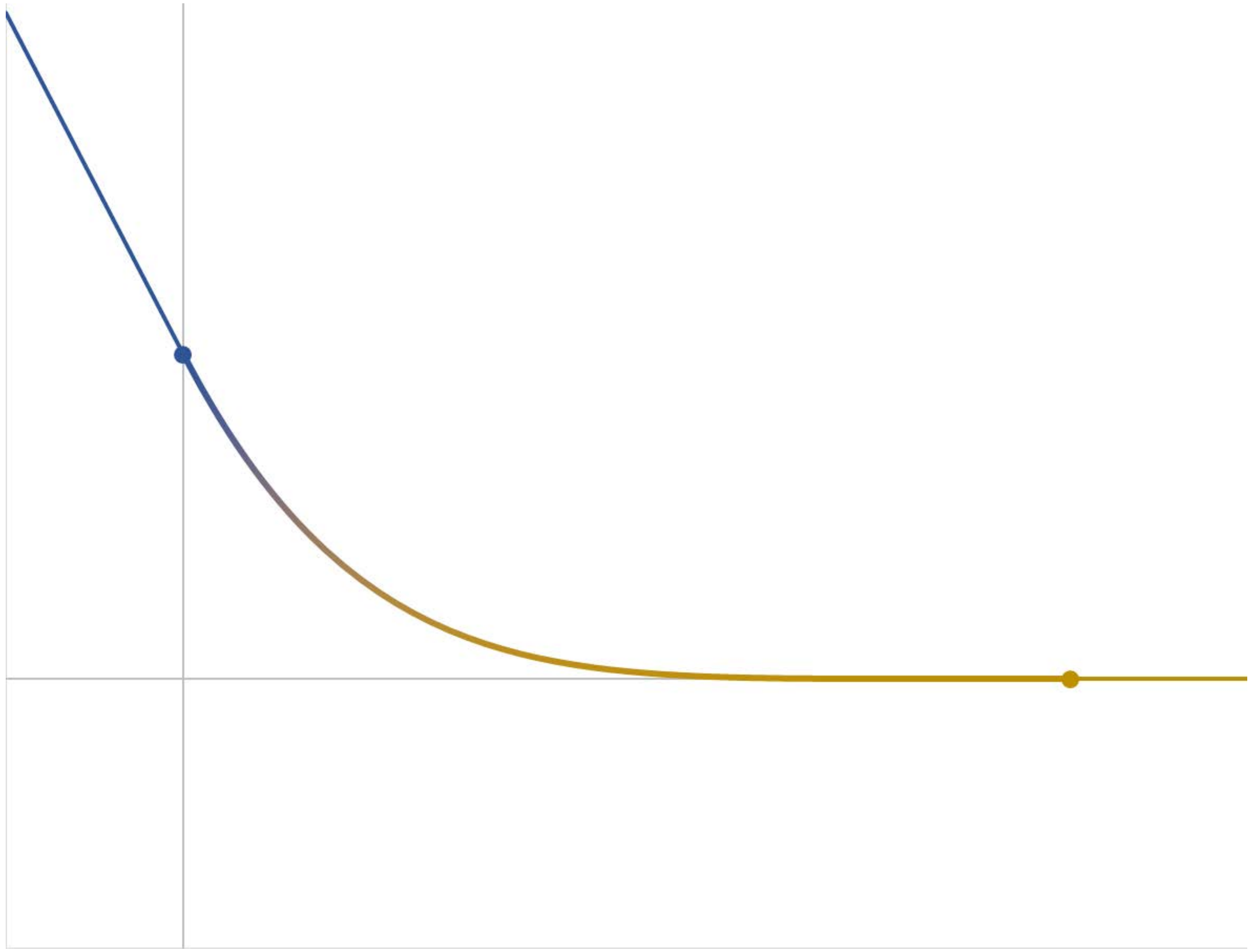
- Capture pose difference between Source and Target
- Ease out the difference over time
- Remember Source velocity (via finite differences)
- Match initial velocity
- Limit overshoot by controlling initial acceleration
- Choose a_0 to give us zero jerk at t_1

Idea #3: Limit Overshoot

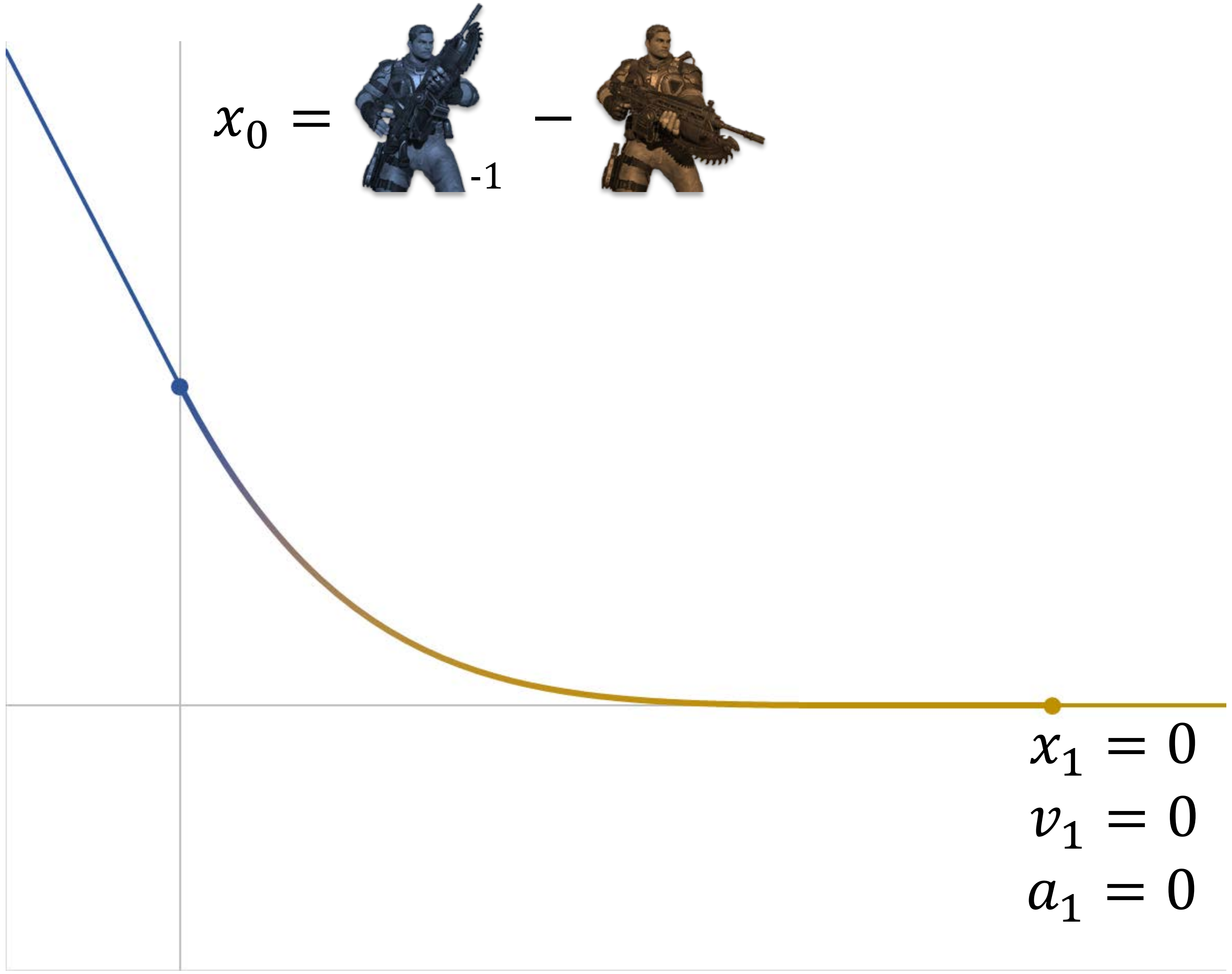


Inertialization

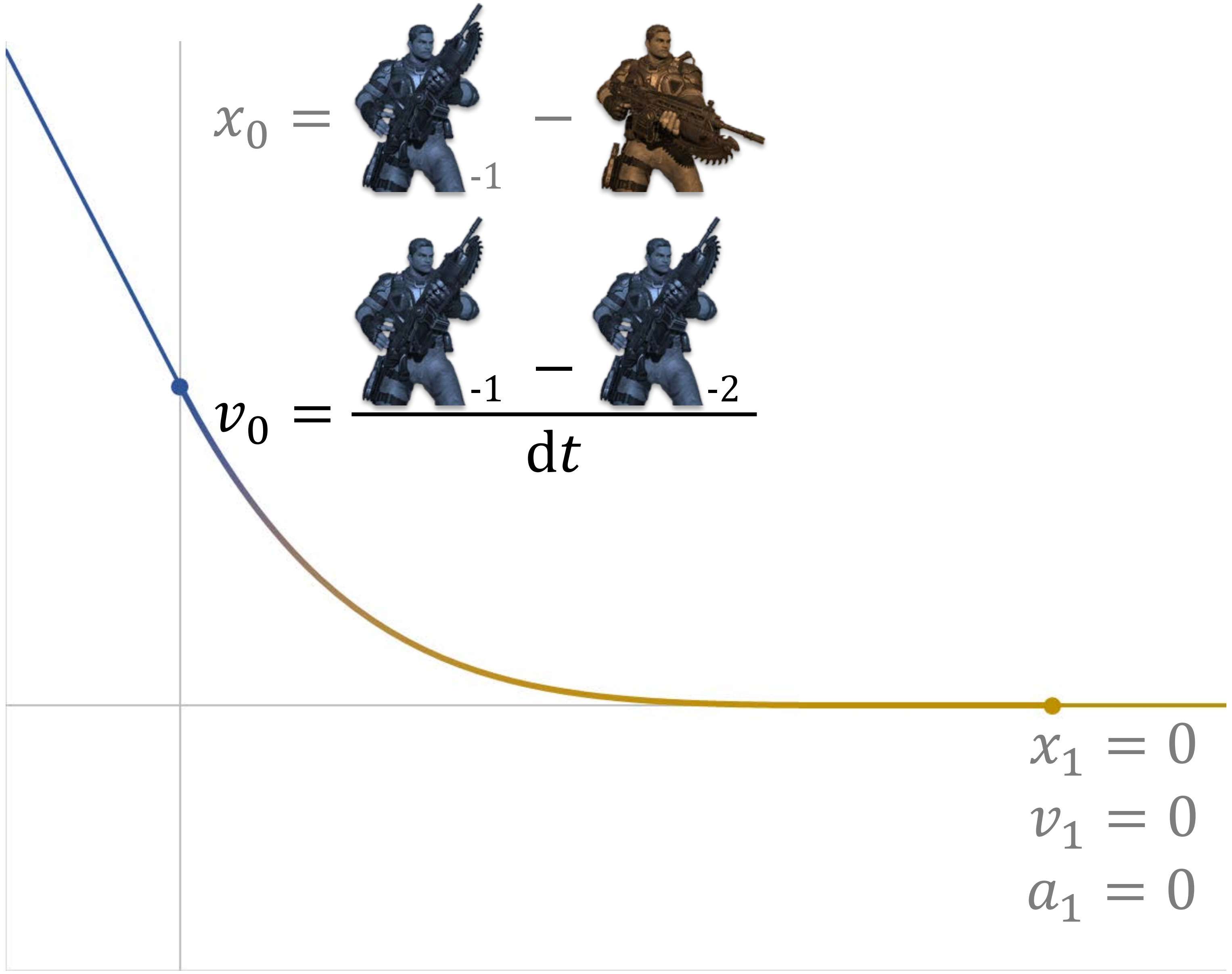
INERTIALIZATION



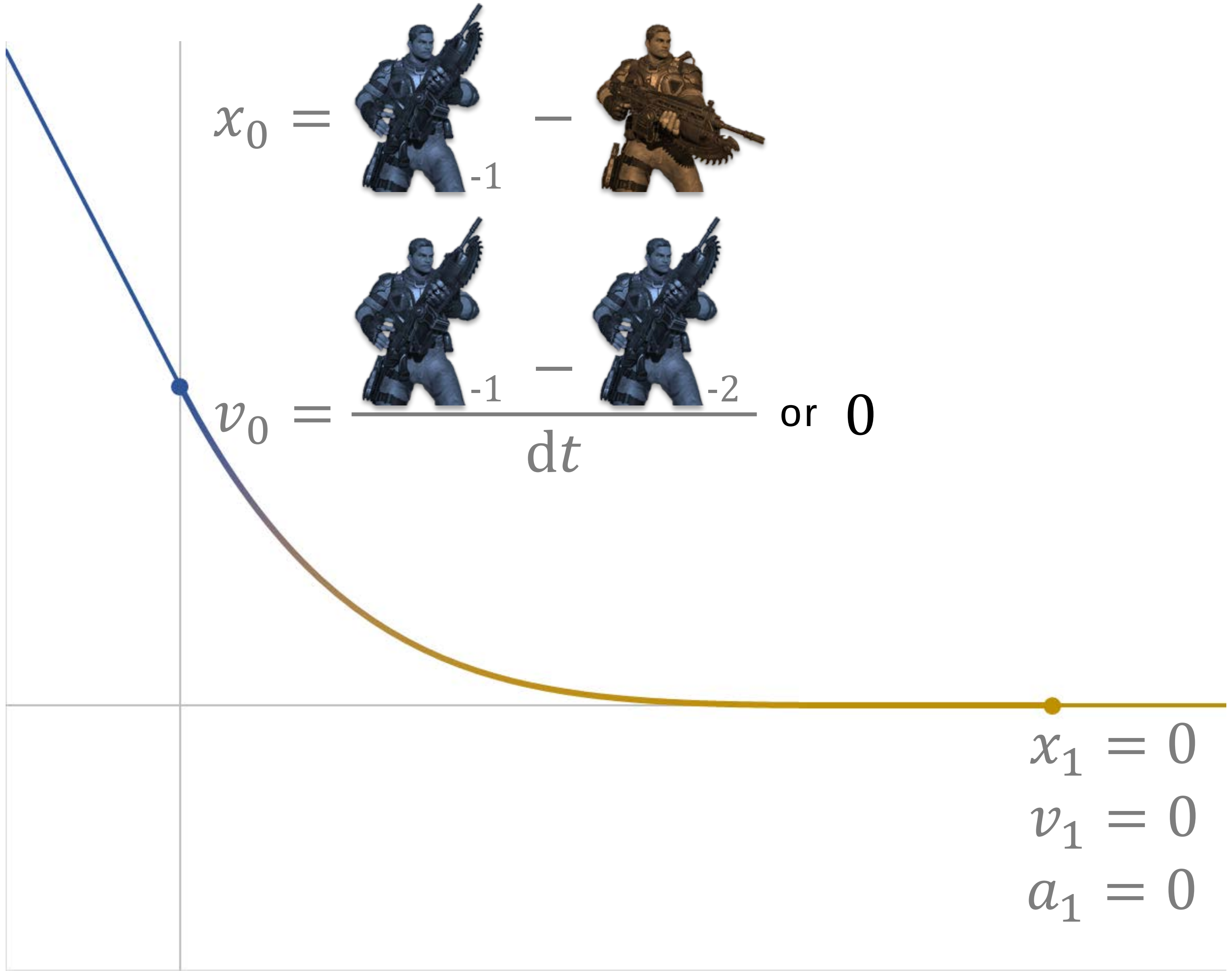
Inertialization – Initial Values



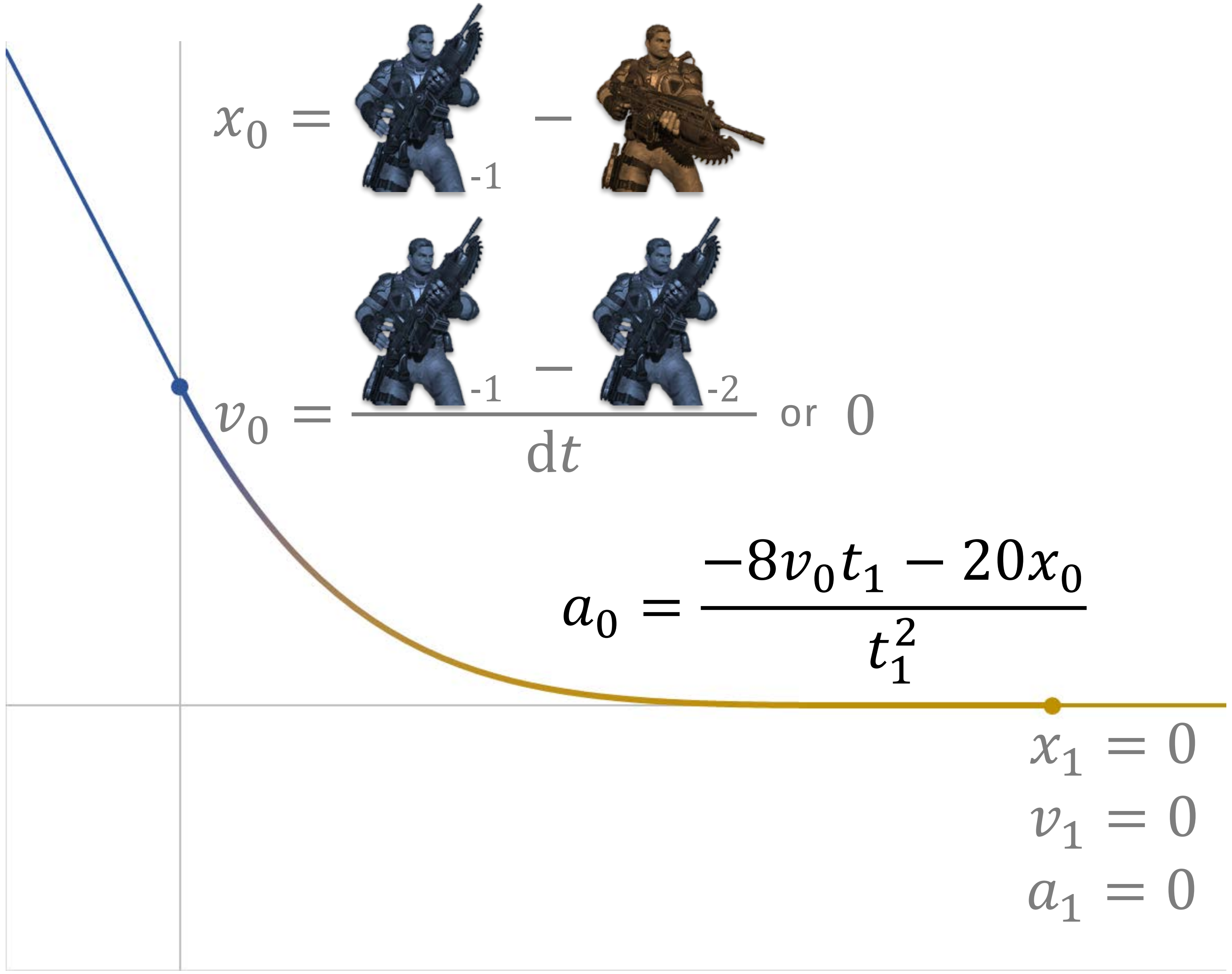
Inertialization – Initial Velocity



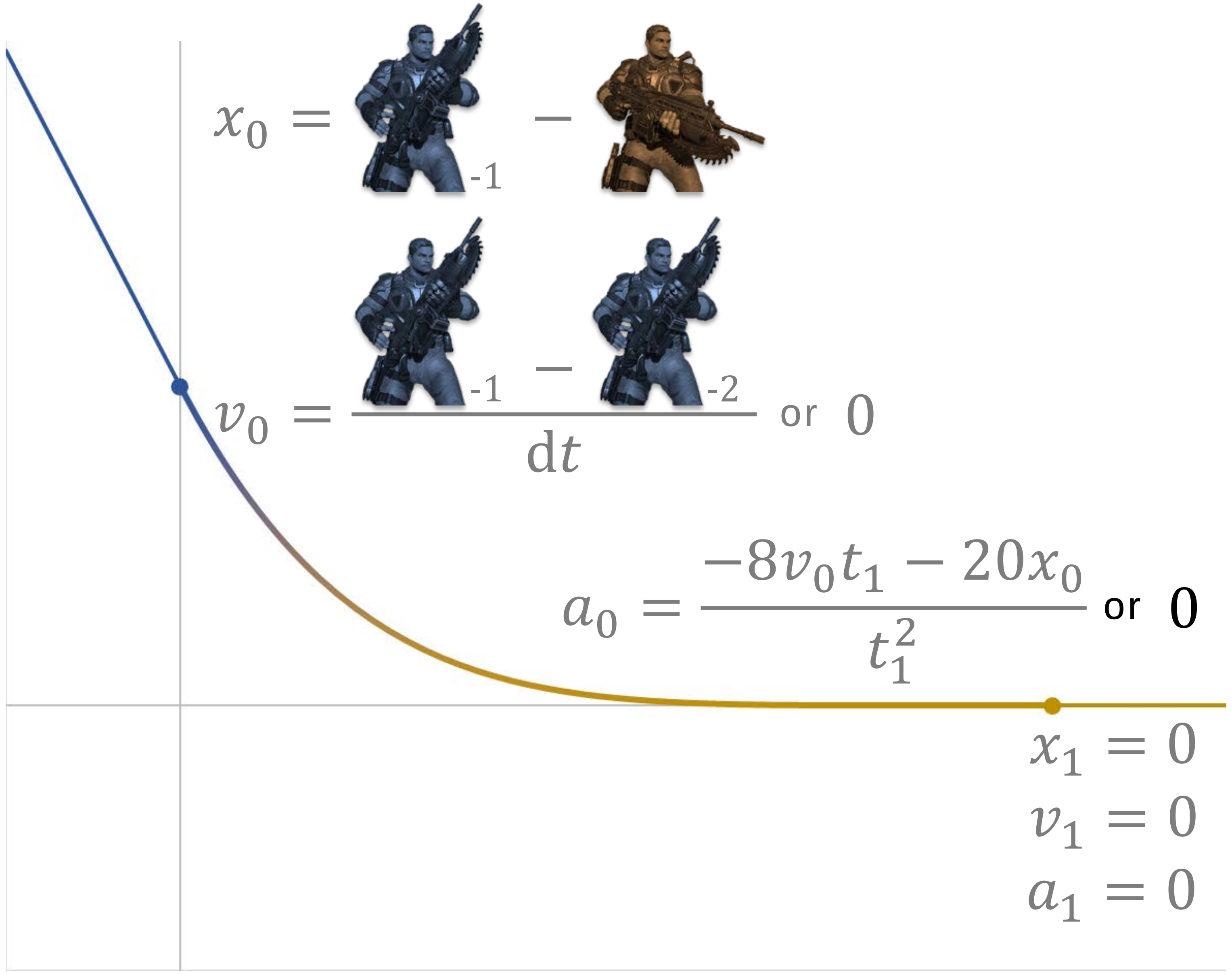
Inertialization – Initial Velocity



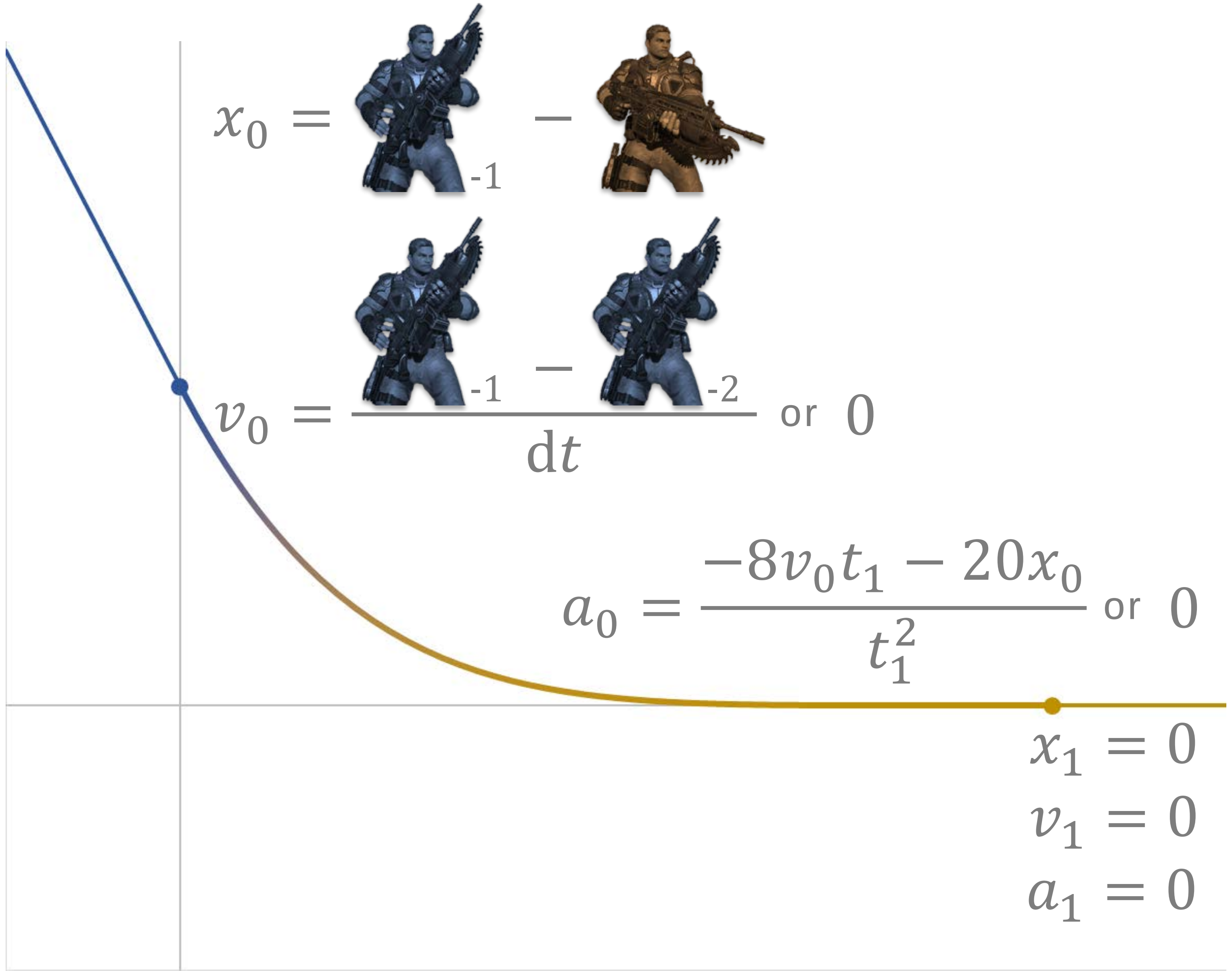
Inertialization – Acceleration



Inertialization – Acceleration



INERTIALIZATION – X(T)



$$A = -\frac{a_0t_1^2 + 6v_0t_1 + 12x_0}{2t_1^5}$$

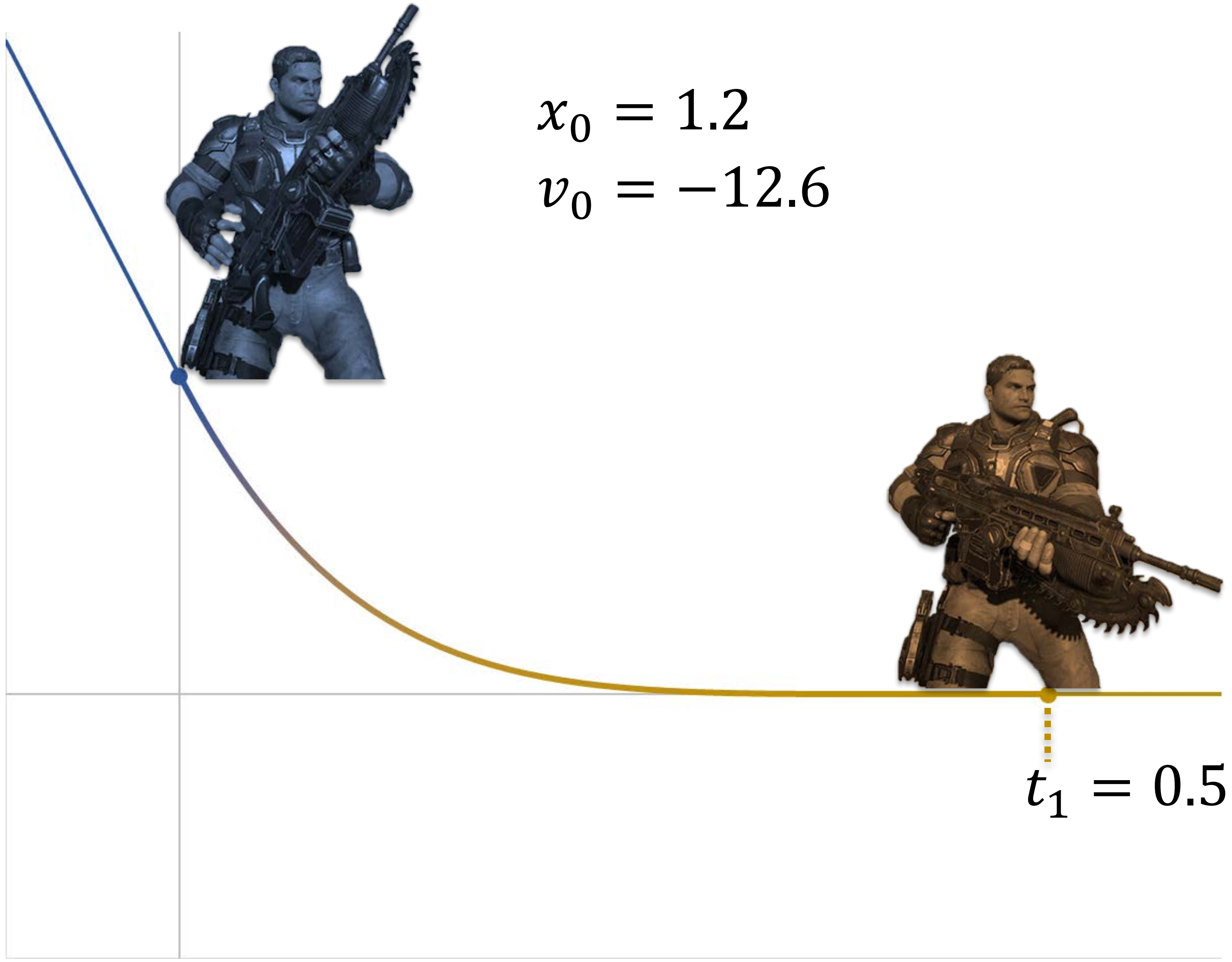
$$B = \frac{3a_0t_1^2 + 16v_0t_1 + 30x_0}{2t_1^4}$$

$$C = -\frac{3a_0t_1^2 + 12v_0t_1 + 20x_0}{2t_1^3}$$

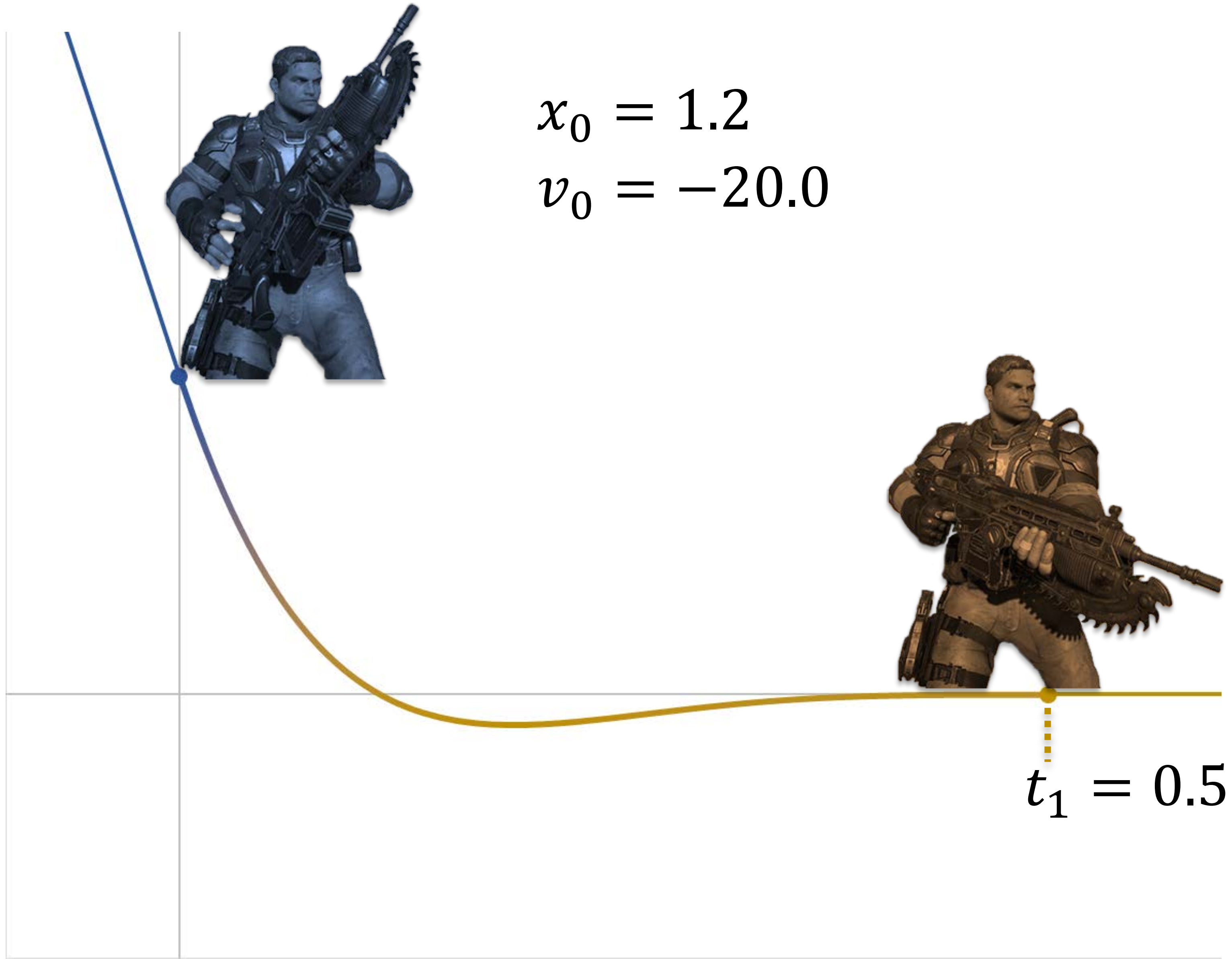
$$x_t = At^5 + Bt^4 + Ct^3 + \frac{a_0}{2}t^2 + v_0t + x_0$$

Overshoot Revisited

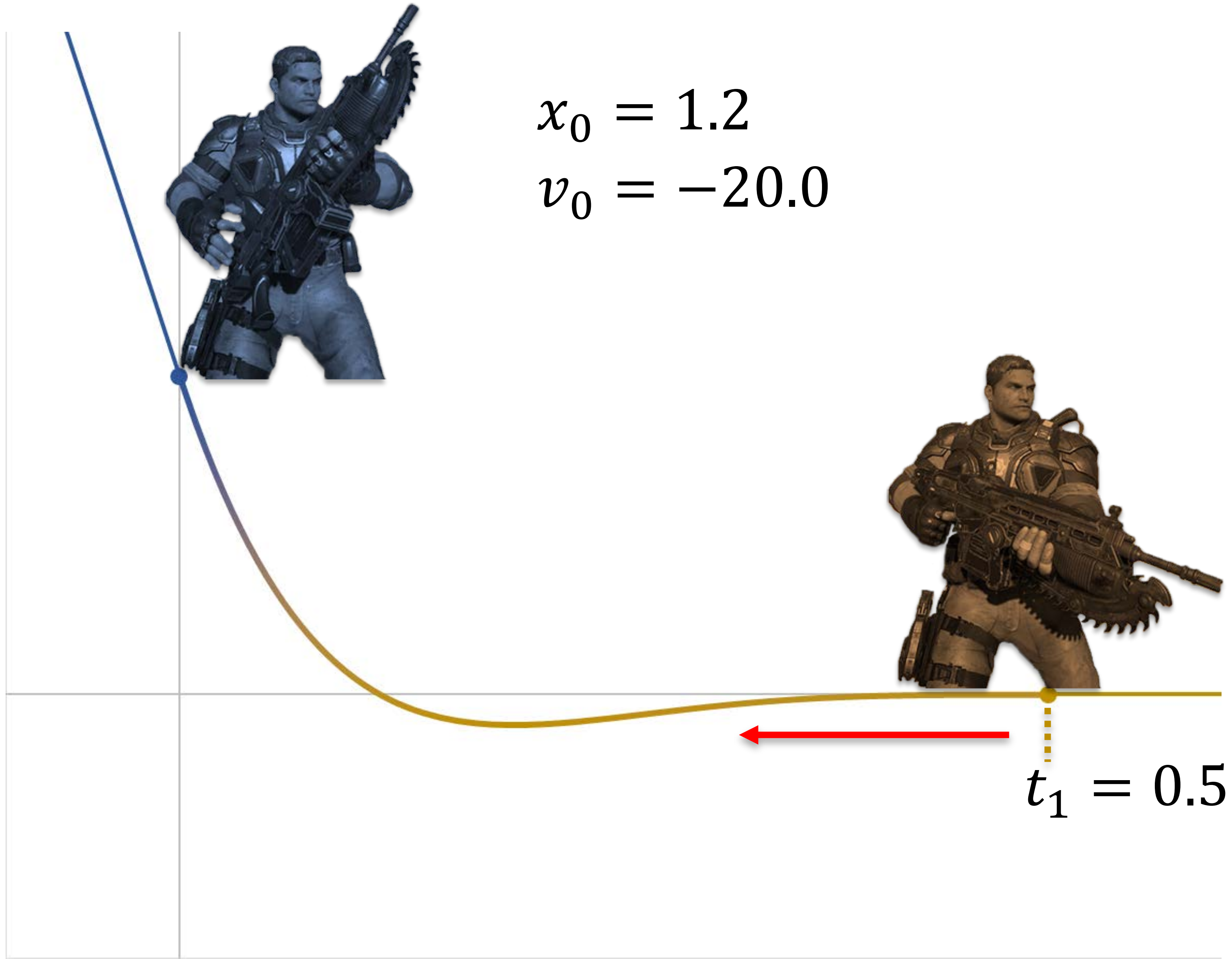
Overshoot Revisited



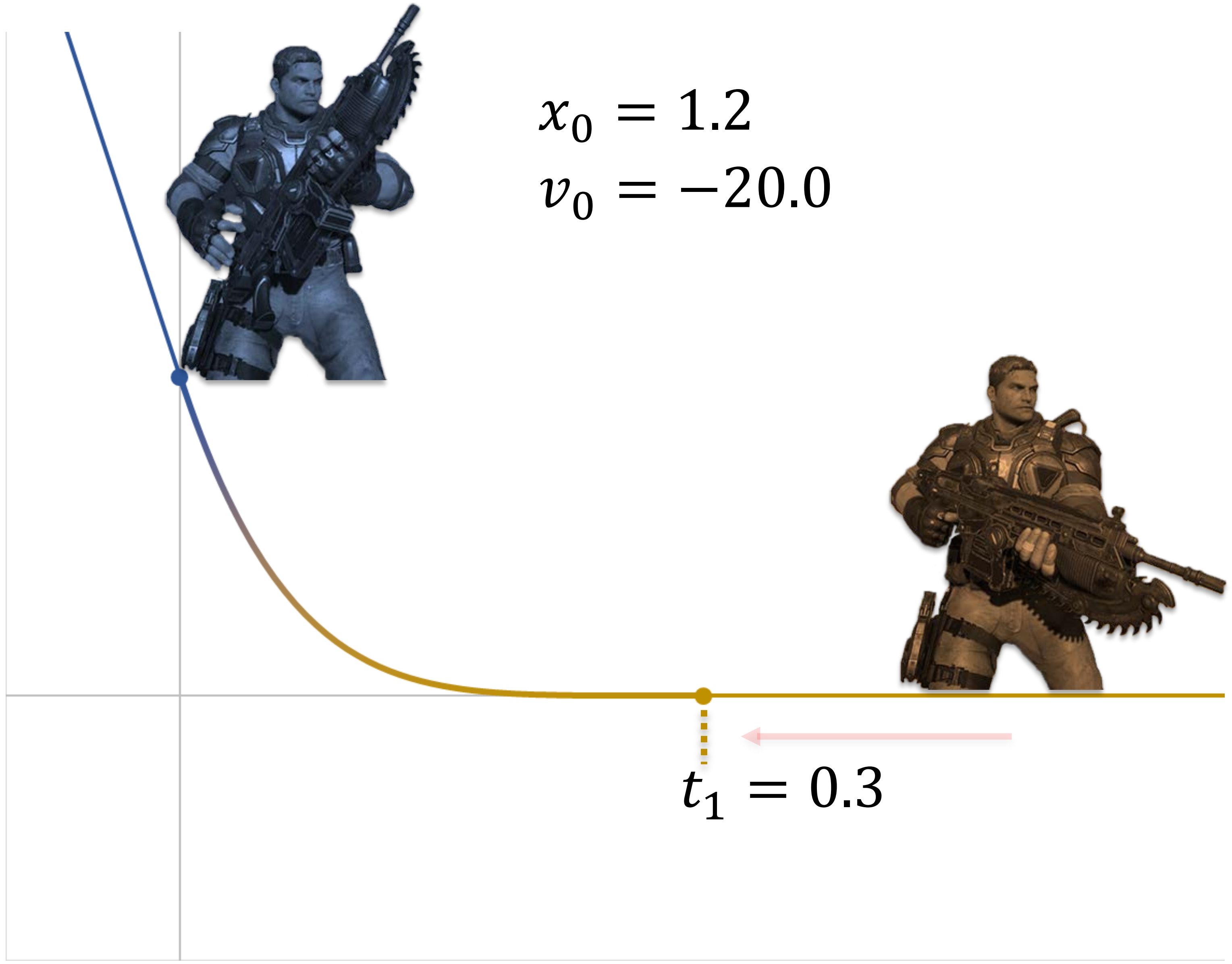
Overshoot Revisited



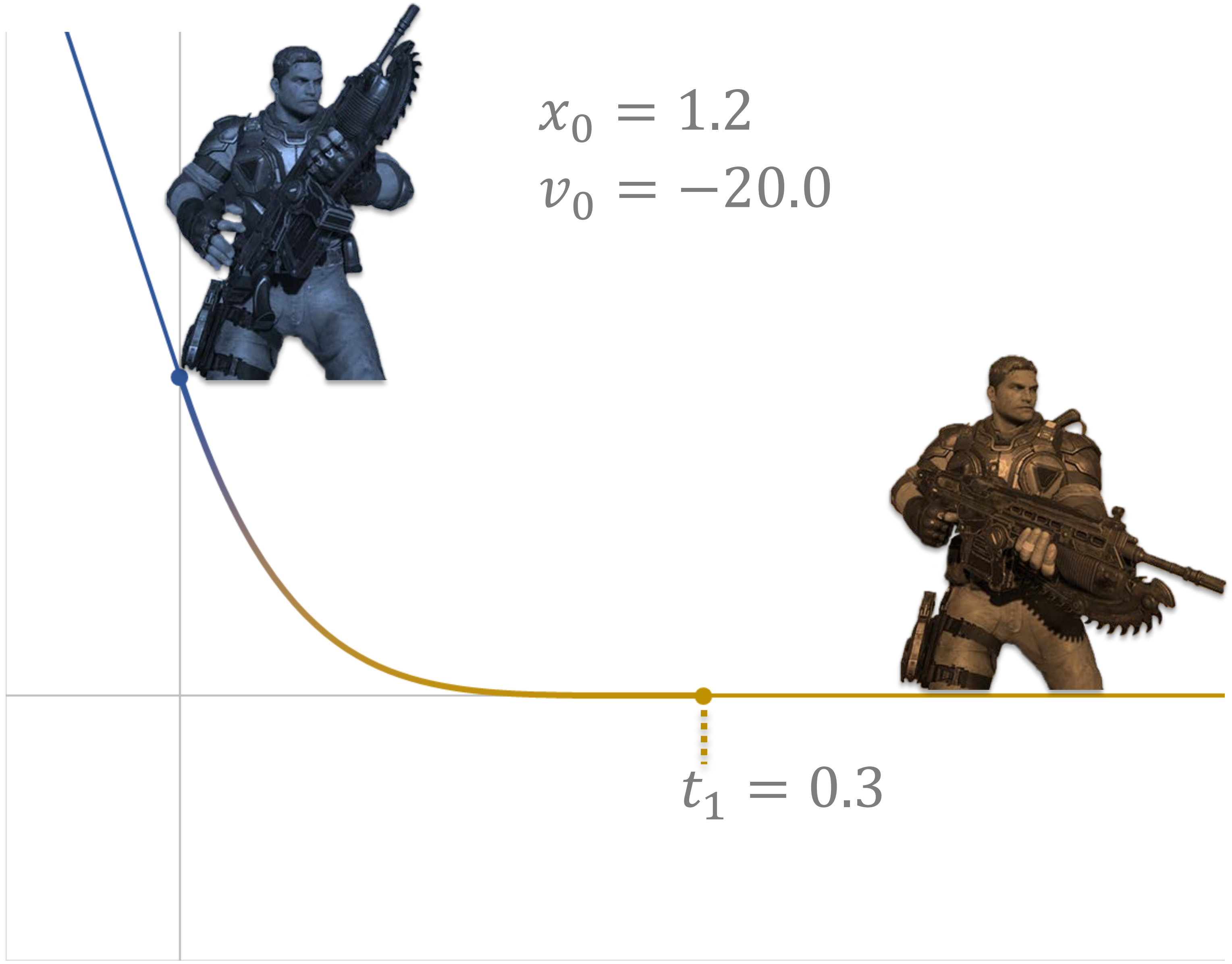
Idea #4: Clamp Transition Time



Idea #4: Clamp Transition Time

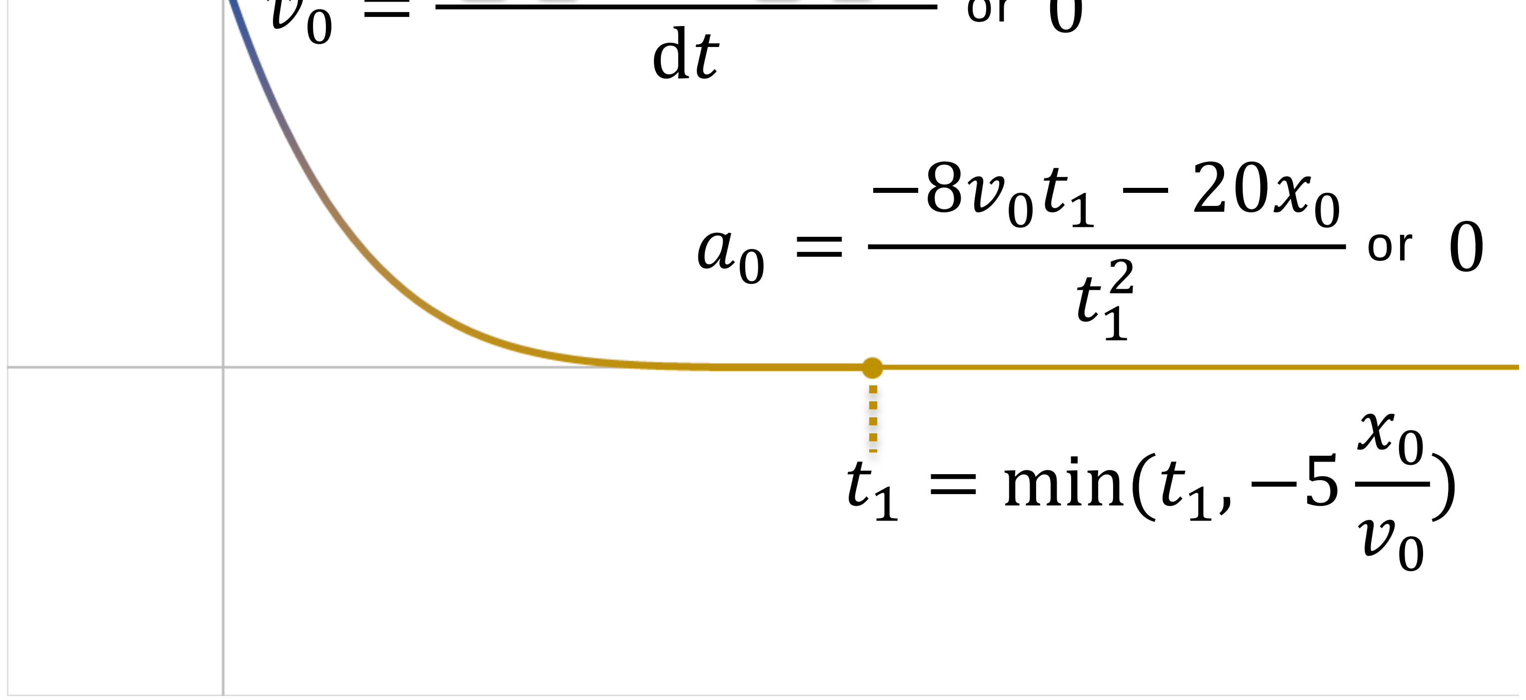


Idea #4: Clamp Transition Time



$$t_1 = \min(t_1, -5\frac{x_0}{v_0})$$

Inertialization on One Slide



$x_0 =$  -1 $-$ 

$v_0 =$ ( -1 $-$  -2) / $\mathrm{d}t$ or 0

$$a_0 = \frac{-8v_0 t_1 - 20x_0}{t_1^2} \text{ or } 0$$

$$t_1 = \min\left(t_1, -5\frac{x_0}{v_0}\right)$$

$$A = -\frac{a_0 t_1^2 + 6v_0 t_1 + 12x_0}{2t_1^5}$$

$$B = \frac{3a_0 t_1^2 + 16v_0 t_1 + 30x_0}{2t_1^4}$$

$$C = -\frac{3a_0 t_1^2 + 12v_0 t_1 + 20x_0}{2t_1^3}$$

$$x_t = At^5 + Bt^4 + Ct^3 + \frac{a_0}{2}t^2 + v_0 t + x_0$$

Vectors and Quaternions

Inertializing Vectors

- Obvious choice:
  - Inertialize x,y,z independently
  - Visual artifacts if vx_0,vy_0,vz_0 are too dissimilar (because of transition time clamping)
- Instead:
  - Decompose vector into direction and magnitude
  - Inertialize the magnitude

INERTIALIZING VECTORS

$\vec{x}_0 =$  $_{-1}$ $-$ 

$\vec{x}_{-1} =$  $_{-2}$ $-$ 

INERTIALIZING VECTORS

$\vec{x}_0 =$  $_{-1}$ $-$ 

$\vec{x}_{-1} =$  $_{-2}$ $-$ 

$x_0 = |\vec{x}_0|$

Inertializing Vectors

$\vec{x}_0 =$  -1 $-$ 

$$x_0 = |\vec{x}_0|$$

$\vec{x}_{-1} =$  -2 $-$ 

$$x_{-1} = \vec{x}_{-1} \cdot \frac{\vec{x}_0}{x_0}$$

INERTIALIZING VECTORS

$\vec{x}_0 =$  $_{-1} \; -$ 

$$x_0 = |\vec{x}_0|$$

$$v_0 = \frac{x_0 - x_{-1}}{\Delta t}$$

$\vec{x}_{-1} =$  $_{-2} \; -$ 

$$x_{-1} = \vec{x}_{-1} \cdot \frac{\vec{x}_0}{x_0}$$

Inertializing Vectors

$\vec{x}_0 =$  $_{-1}$ $-$ 

$x_0 = |\vec{x}_0|$

$$v_0 = \frac{x_0 - x_{-1}}{\Delta t}$$

$\vec{x}_{-1} =$  $_{-2}$ $-$ 

$$x_{-1} = \vec{x}_{-1} \cdot \frac{\vec{x}_0}{x_0}$$

$\vec{x}_\mathrm{t} = x_\mathrm{t} \dfrac{\vec{x}_0}{x_0} +$  $_\mathrm{t}$

Inertializing Quaternions

- Similar construction to vectors:
  - Decompose quaternion into axis and angle
  - Inertialize the angle

INERTIALIZING QUATERNIONS

$q_0 =$  $_{-1}$ $*$  $^{-1}$

$q_{-1} =$  $_{-2}$ $*$  $^{-1}$

INERTIALIZING QUATERNIONS

$q_0 =$  $^{-1}$ $*$  $^{-1}$

$q_{-1} =$  $^{-2}$ $*$  $^{-1}$

$\vec{x}_0 = Axis(q_0) \qquad x_0 = Angle(q_0)$

INERTIALIZING QUATERNIONS

$q_0 =$  -1 $*$  -1

$\vec{x}_0 = Axis(q_0) \qquad x_0 = Angle(q_0)$

$q_{-1} =$  -2 $*$  -1

$$x_{-1} = 2\tan^{-1}\frac{\overrightarrow{q_{\mathrm{xyz}}}\cdot\vec{x}_0}{q_{\mathrm{w}}}$$

Twist of q_{-1} around $\vec{x}_0$

K. Shoemake. 1994.
Fiber Bundle Twist Reduction
Graphics Gems IV, 230 – 236

Inertializing Quaternions

$q_0 =$  -1 $*$  -1

$\vec{x}_0 = Axis(q_0) \qquad x_0 = Angle(q_0)$

$$v_0 = \frac{x_0 - x_{-1}}{\Delta t}$$

$q_{-1} =$  -2 $*$  -1

$$x_{-1} = 2\tan^{-1}\frac{\overrightarrow{q_{\mathrm{xyz}}} \cdot \vec{x}_0}{q_{\mathrm{w}}}$$

Twist of q_{-1} around $\vec{x}_0$

K. Shoemake. 1994.
Fiber Bundle Twist Reduction
Graphics Gems IV, 230 – 236

INERTIALIZING QUATERNIONS

$q_0 =$  $^{-1}$ $*$  $^{-1}$

$\vec{x}_0 = Axis(q_0) \qquad x_0 = Angle(q_0)$

$$v_0 = \frac{x_0 - x_{-1}}{\Delta t}$$

$q_{\mathrm{t}} = \left\{ \frac{Axis:\ \ \vec{x}_0}{Angle:\ x_{\mathrm{t}}} \right\} *$  $_{\mathrm{t}}$

$q_{-1} =$  $^{-2}$ $*$  $^{-1}$

$$x_{-1} = 2\tan^{-1}\frac{\overrightarrow{q_{\mathrm{xyz}}} \cdot \vec{x}_0}{q_{\mathrm{w}}}$$

Twist of q_{-1} around $\vec{x}_0$

K. Shoemake. 1994.
Fiber Bundle Twist Reduction
Graphics Gems IV, 230 – 236

Blending vs Inertialization

Blending vs Inertialization

Blending

- Evaluate both Source & Target during transition
- Variable anim frame cost

Inertialization

- Only evaluate Target during transition
- Fixed anim frame cost

BLENDING VS INERTIALIZATION

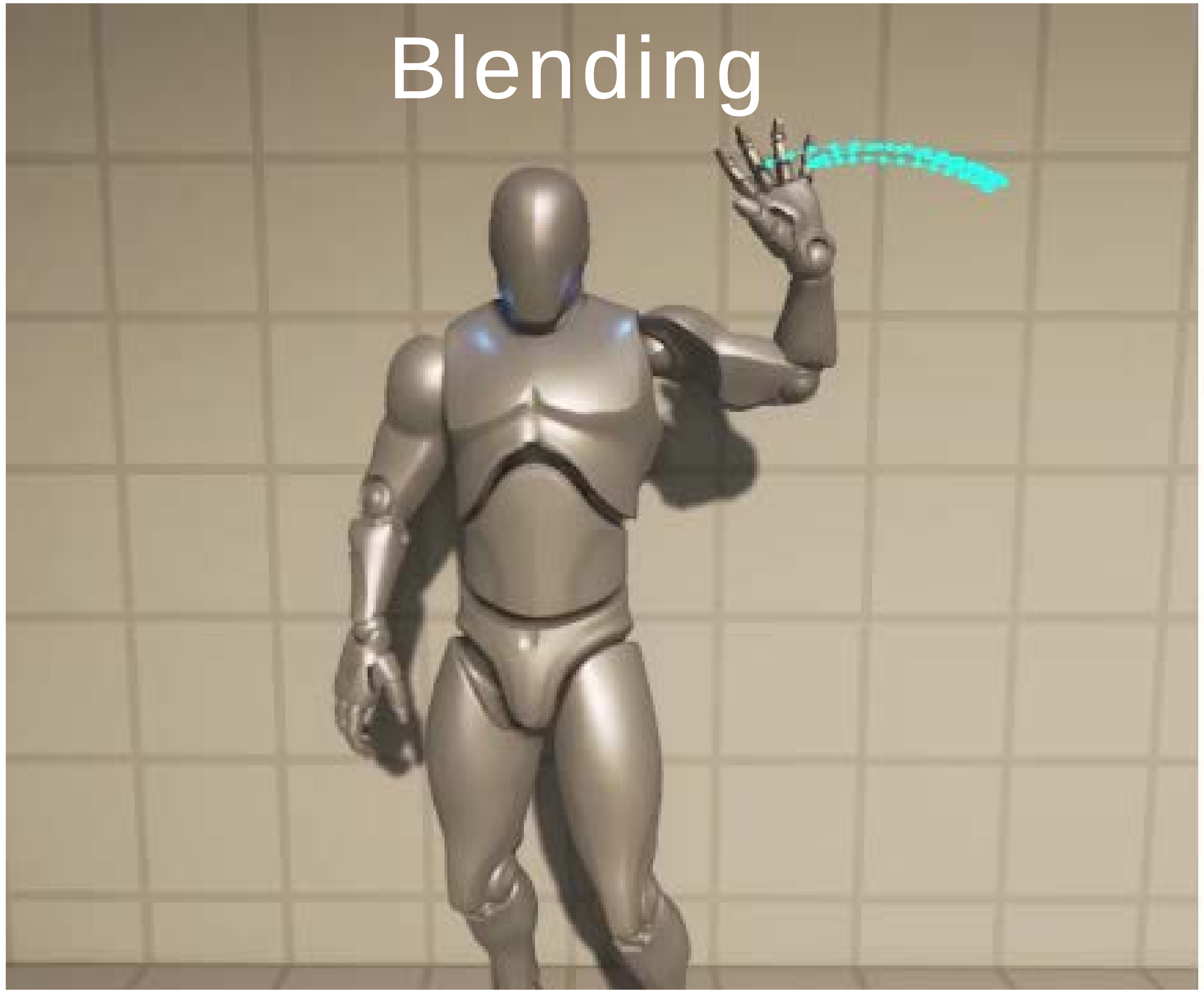
Blending

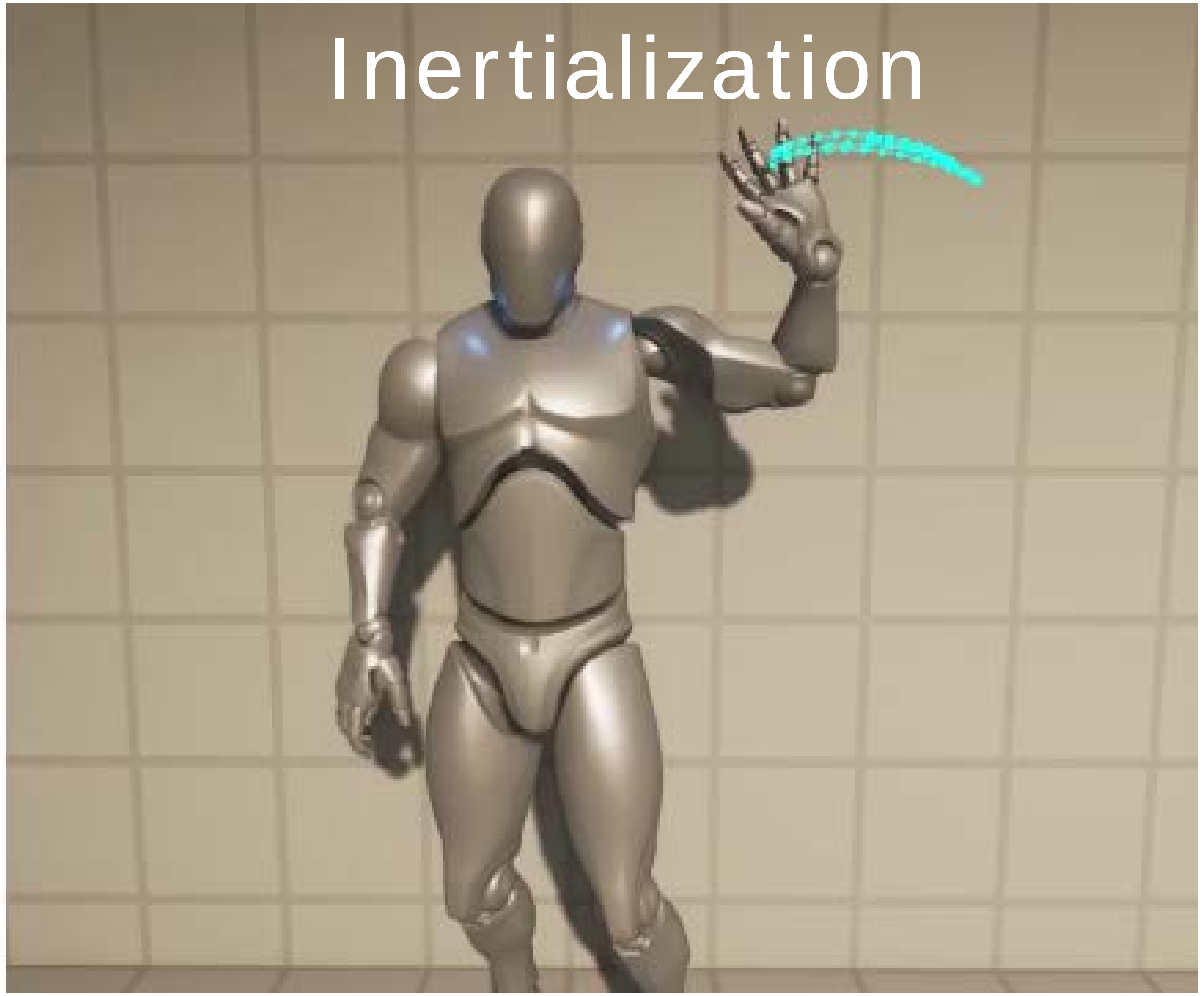
- Manage multiple sets of state during transitions
- Adds complexity

Inertialization

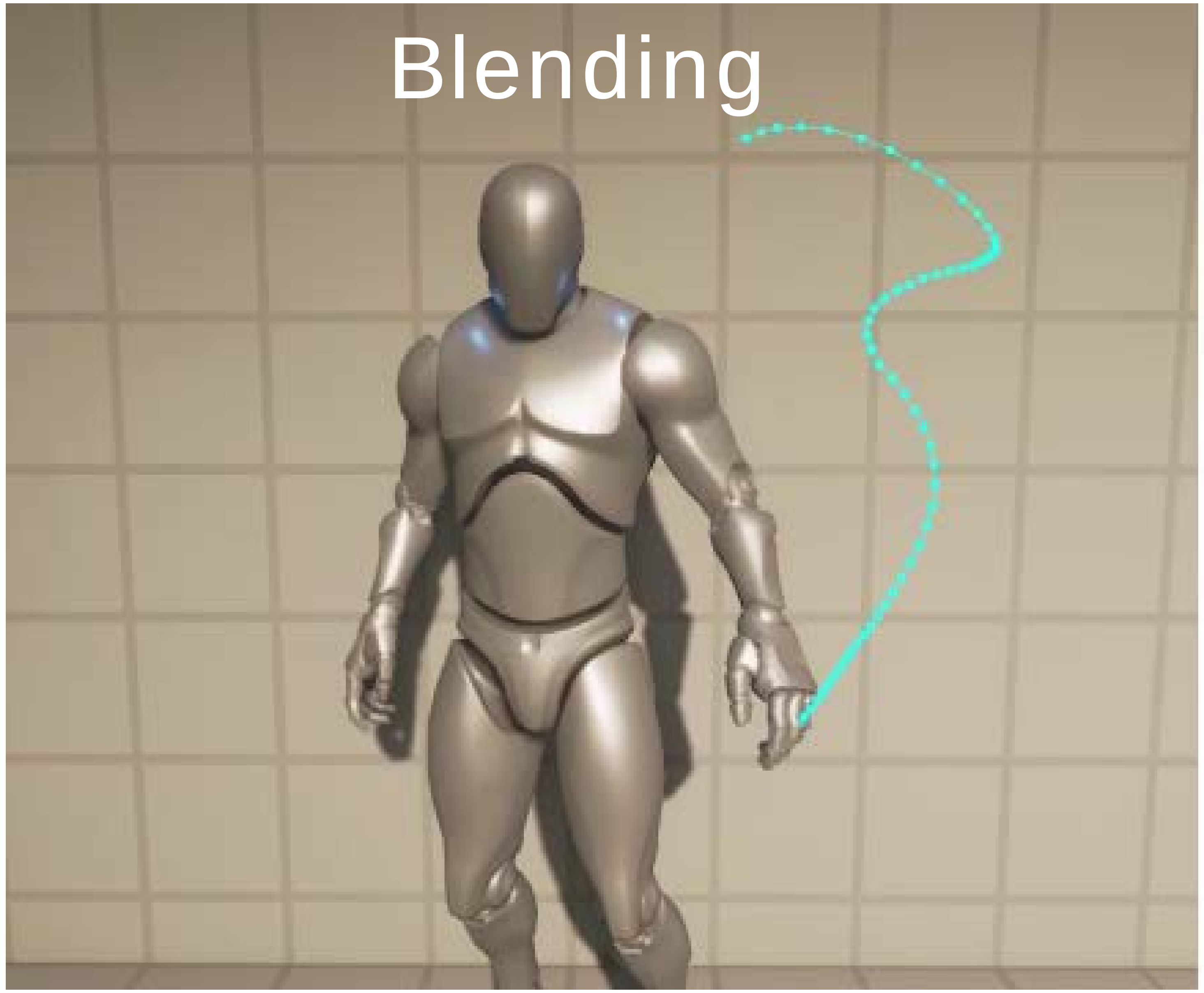
- Only maintain one set of state during transitions
- Fire and forget

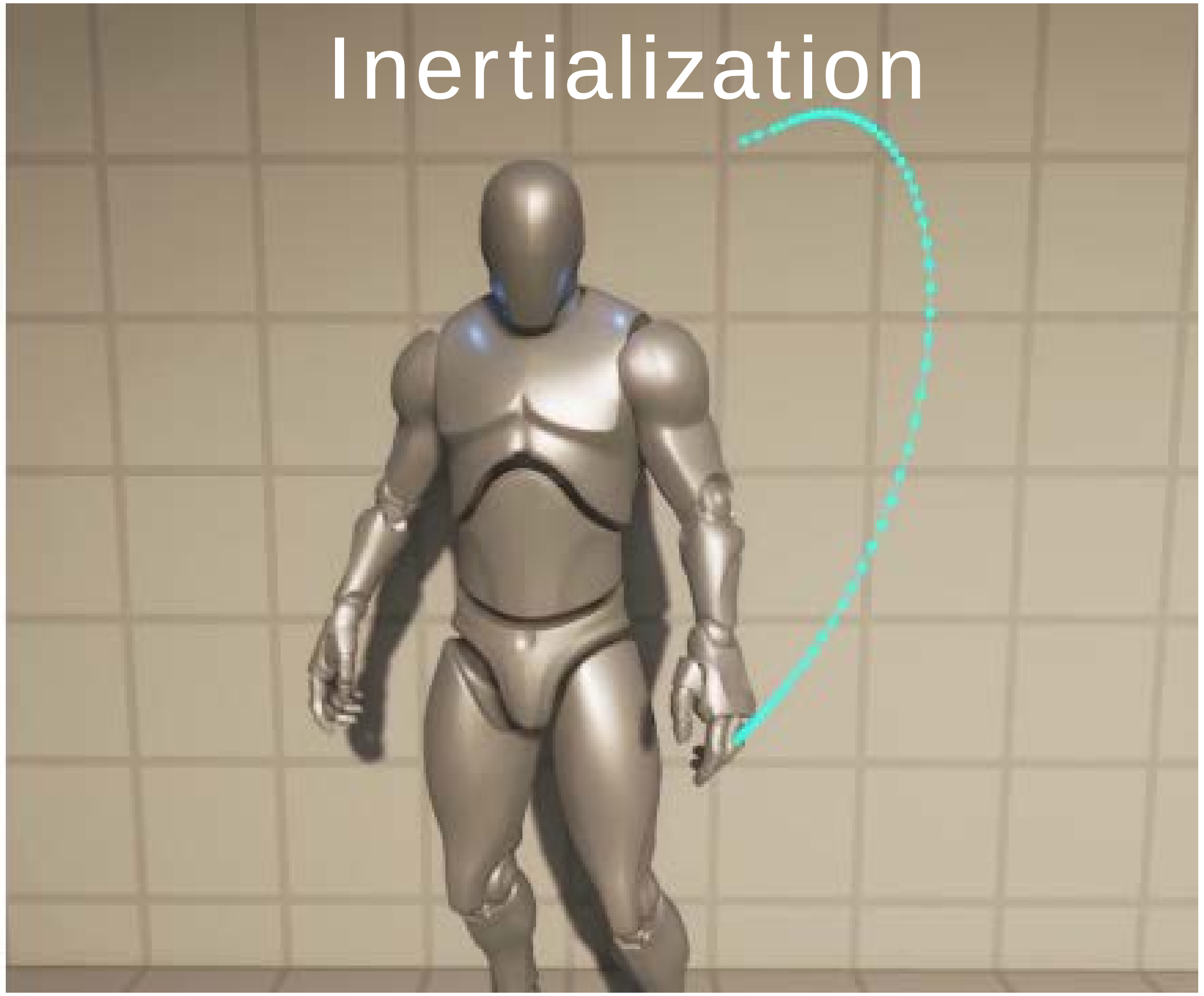
BLENDING VS INERTIALIZATION





Blending vs Inertialization





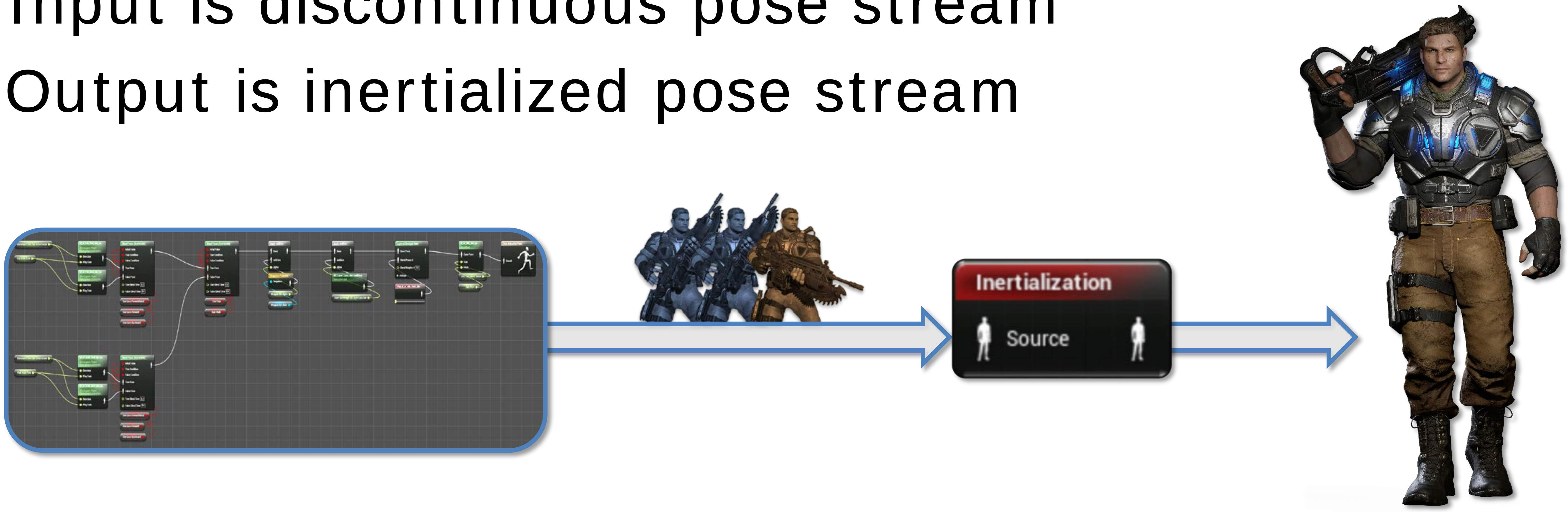
Inertialization in a Game Engine

Inertialization in a Game Engine

- Inertialization Node / Filter
- Animation System Hooks
- Code Hooks

Inertialization Node

- Evaluated after the main animation graph
- Input is discontinuous pose stream
- Output is inertialized pose stream



Inertialization Node

- When a new inertialization is requested:
  - Compute and store x_0, v_0 for all joints
  - Store t_1 and set $t = 0$
- Every frame:
  - Update t with delta time
  - Evaluate and apply x(t) for all joints
  - Store the OUTPUT pose in the pose history buffer

Animation System Hooks

- Add “inertialization” as a new blend curve type
- When a blend is requested with “inertialization” type:
  - Inertialize with the supplied blend time
  - Zero the blend time to bypass regular blending

Code Hooks

- Expose “Request Inertialization” to code
- Eliminate other types of discontinuities
- And other tricks...

TIPS AND TRICKS

Image Source: Evan Amos via Wikimedia Commons

Smoothing Other Discontinuities

- Gears of War 3:
  - Snap character rotation when switching to sprint
- Gears of War 4:
  - Snap character rotation when switching to sprint
  - Inertialize away the discontinuity

Locomotion Filtering

- Gears controls are very responsive (twitchy)
- Filter inputs to locomotion blend spaces
- If filtered values are too far from actual values…
  - Snap to actual values
  - Inertialize
- Fluid pose even with twitchy inputs

Fire & Forget – Motion Warping

- Don’t need to maintain warp point data across transitions
- Only 1 active warp at a time
- Simplifies bookkeeping
- Simplifies replication

S. Dickinson. Motion Warping in ‘Gears of War 4’: Doing More with Less. GDC 2017

Thank You

David Bollo
dbollo@microsoft.com