Moving Beyond Ragdolls:

Generating Versatile Human Behaviors by Combining Motion Capture and Controlled Physical Simulation

by Michael Mandel

Carnegie Mellon University / Apple Computer

mmandel@gmail.com

http://www.mmandel.com/

Guest Speaker: Victor Zordan

University of California Riverside

What should our characters be able to do?

- Lot’s of behaviors - leaping, grasping, moving, looking, attacking
- Exhibit personality - move “sneakily” or “aggressively”
- Awareness of environment - balance/posture adjustments
- Physical force-induced movements (jumping, falling, swinging)

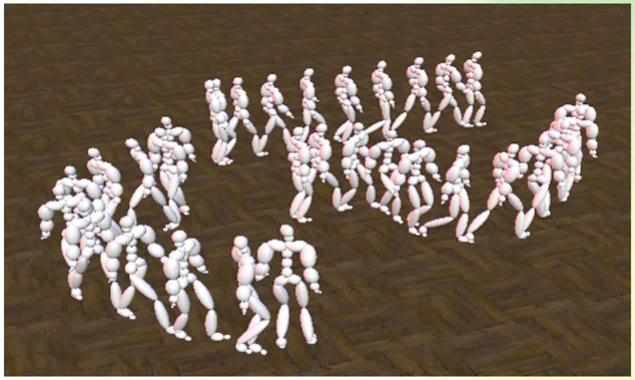
All this, and also be directable (by a player or NPC)...

Motivating Video: Nike "Presto's"

How can we create a character
like this for our games?

Motion Capture Data

+ Captures style and subtle nuances
- Predetermines characters abilities (inflexible)
- Can't interact well with environment



Motion Capture Data

+ Captures style and subtle nuances
- Predetermines characters abilities (inflexible)
- Can't interact well with environment

Physical Simulation

+ Interacts well with environment
- "Ragdoll" movement is lifeless
- Difficult to develop complex behaviors
- Difficult to interface with existing motion

Motion Capture Data?? ← Stylistic Realism

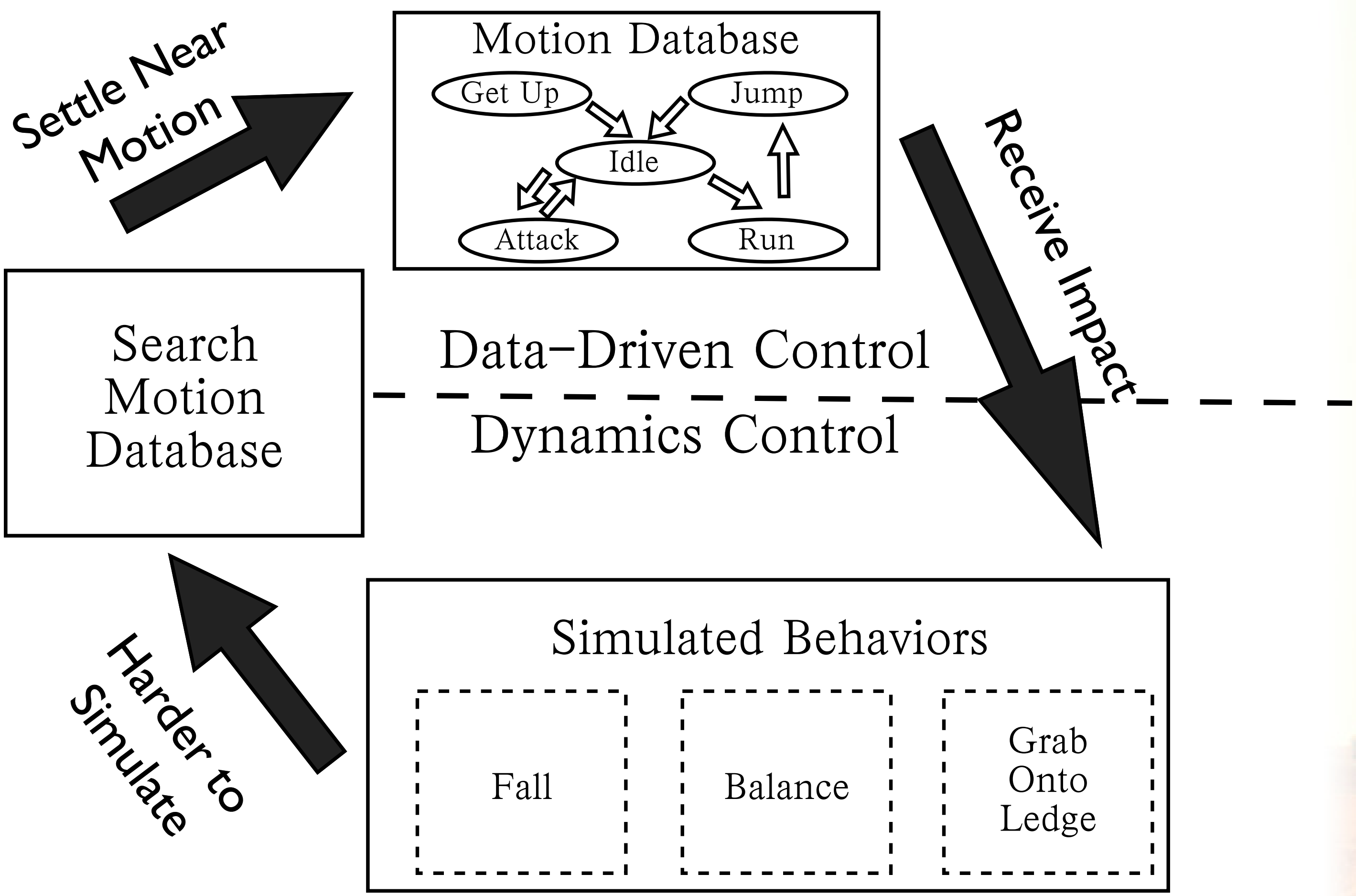
Physical Realism → **Physical Simulation??**

Proposed Method:

- Combine the best of both approaches
- Activate either one when most appropriate
- Add life to ragdolls using control systems
  - (only simulate behaviors that are **manageable**)

GameDevelopers Conference

High-Level Example



SAN FRANCISCO CA MAR 7-11 GDC 05

GameDevelopers Conference

Outline

1. Simulating Behaviors
   - Quick Ragdoll Intro (Michael)
   - Controllers (Victor)

2. Executing Transitions (Michael)

   Motion Data ⟷ Simulation

3. Another Take on Transitioning (Victor)

4. Conclusions (Michael)

SAN FRANCISCO CA MAR 7-11 GDC ›05

GameDevelopers Conference

Simulation Basics: Setting Up Ragdolls

Supply Your Dynamics Engine:

1. Set of primitives for each body part
2. Mass and inertial properties
3. 1, 2, or 3-DOF joints between parts
4. Joint limit constraints
5. External forces (gravity, etc.)

Dynamics Engine Supplies:

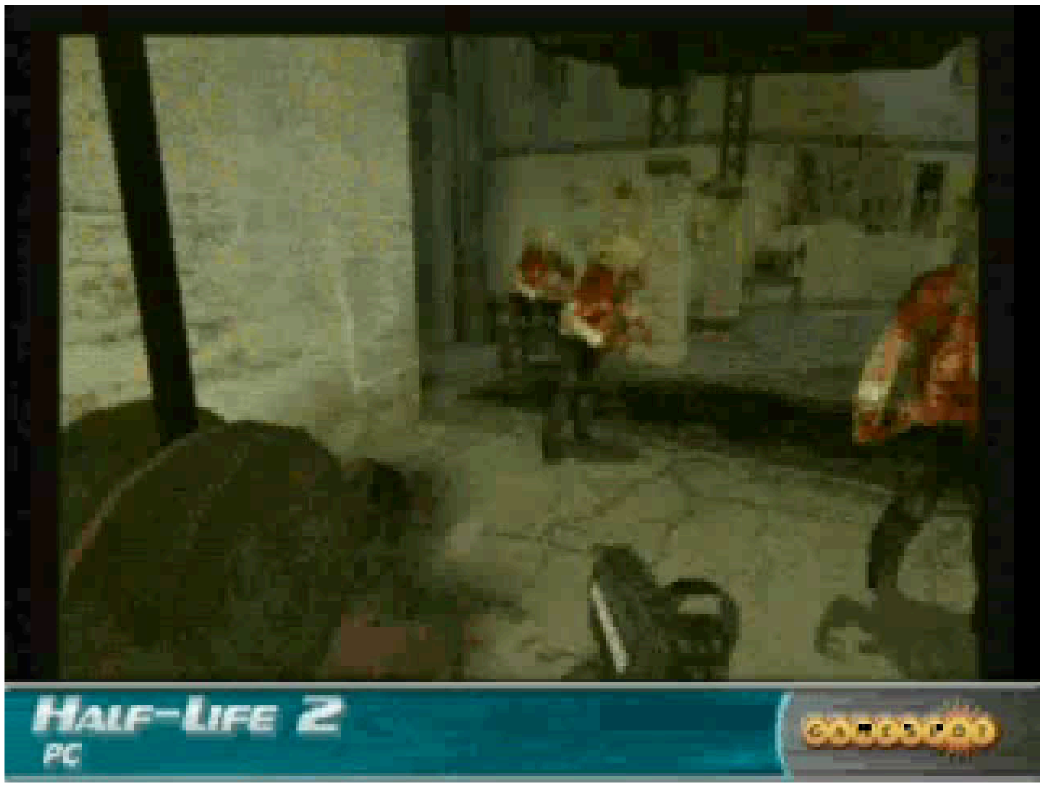
- Updated positions/orientations
- Collision resolution with world

Ultimately, you will drive your skeleton with the simulated primitives...

SAN FRANCISCO CA MAR 7-11 GDC 05

Ragdoll Simulation in Today's Games

Half Life 2



Works great in these situations...

Ragdoll Simulation in Today's Games

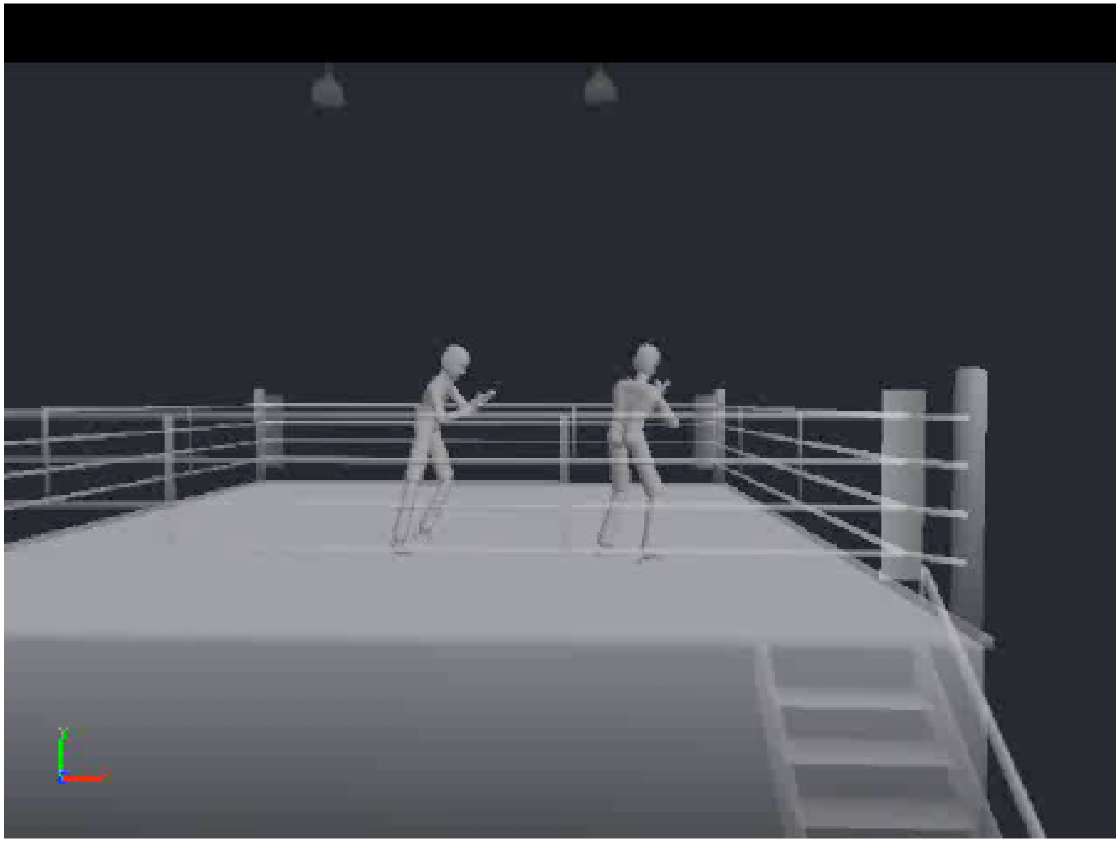
Fight Night 2004

Works great in these situations...

GameDevelopers Conference

Boxing Reactions WITH Controllers

modeling conscious reaction...



courtesy of Natural Motion's *Endorphin*...

http://www.naturalmotion.com

Outline

1. Simulating Behaviors
   - Quick Ragdoll Intro (Michael)
   - Controllers (Victor)

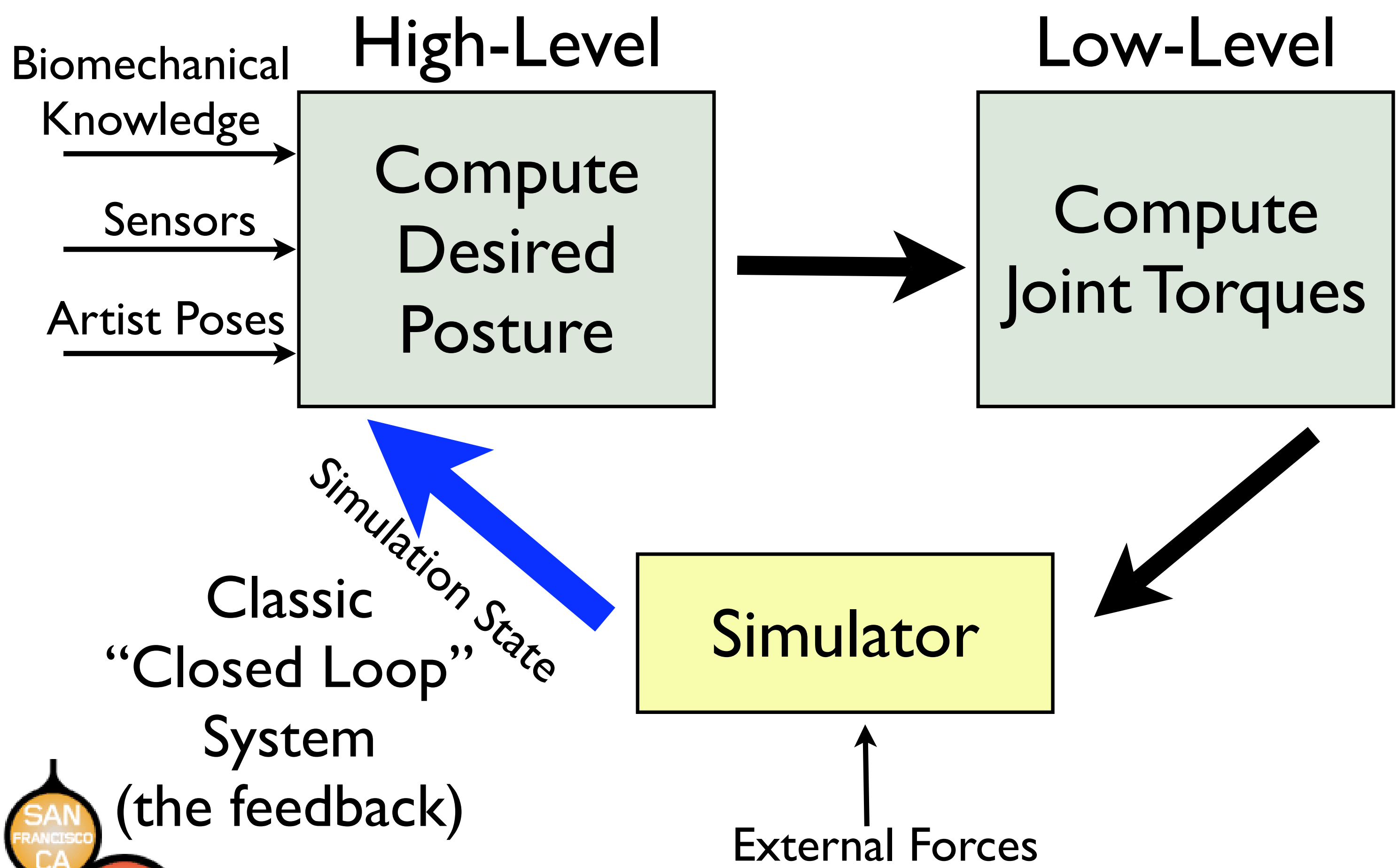
2. Executing Transitions (Michael)

   Motion Data ⟷ Simulation

3. Another Take on Transitioning (Victor)

4. Conclusions (Michael)

Basic Controller Structure



Practical Limitations

- Can be difficult to design (complex coordination of limbs)
- Results can look stiff and unrealistic
- Tip: let natural dynamics of the system control some of the body

More Ballistic

↓

Less DOFs to directly specify

Easier

Falling

Balancing

Running

Tai Chi

Harder

Know your limitations...

Types of Control

Basic Joint-torque Control

- Low-level control
- Sparse *Pose* control
  (May be specified by artist)
- Continuous control
  (Ex: Tracking mocap data)

Hierarchical Control

- Layered controllers
- Higher level controller determines correct desired value for low level
- Derived from sensor or state info
  - Support polygon, center of mass, body contacts, etc.

GameDevelopers Conference

Joint-torque Control

Proportional-Derivative (PD) Controller

- Actuate each joint towards desired target:

$$\tau = k_s(\theta_{des} - \theta) + k_d(-\dot{\theta})$$

θ_{des} is desired joint angle and θ is current angle

k_s and k_d are spring and damper gains

- Acts like a damped spring attached to joint (rest position at desired angle)

  - Alternatively, we could set θ_{des} to mocap (called motion tracking)

Choosing Controller Gains

- Gains are often hand tuned (tedious!)
- Reduce tuned parameters to a single spring and damper
- Scale gains by effective moment of inertia of the chain of bodies connected to each joint:

$$H = \sum_i (m_i r_i \times v_i + I_{CM_i} \omega_i)$$

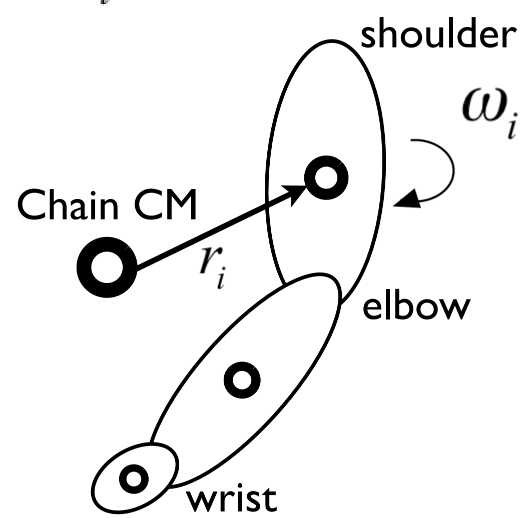
m_i - Mass of body *i*

r_i - Relative center of mass (CM)

v_i - Relative velocity of CM

I_{CM_i} - Inertia tensor of body

ω_i - Angular velocity of body



- More adaptive to natural dynamics of a behavior

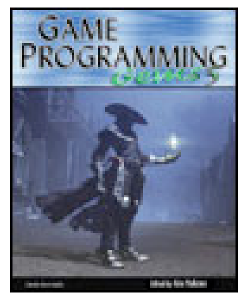
(see [Zordan '02] for more...)

Live Demo

Created With:

OPEN DYNAMICS ENGINE™

http://ode.org

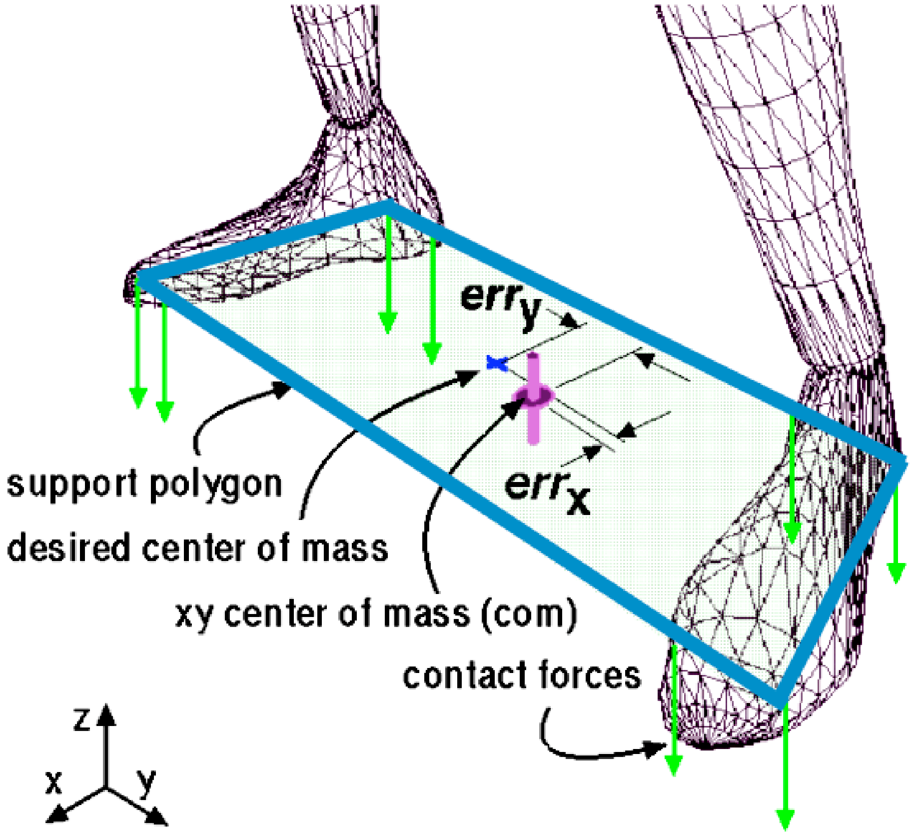


Full Source
in Game Gems 5

Download simplified demo source at:
http://www.mmandel.com/gdc

GameDevelopers Conference

Mid-Level Control: Standing Balance



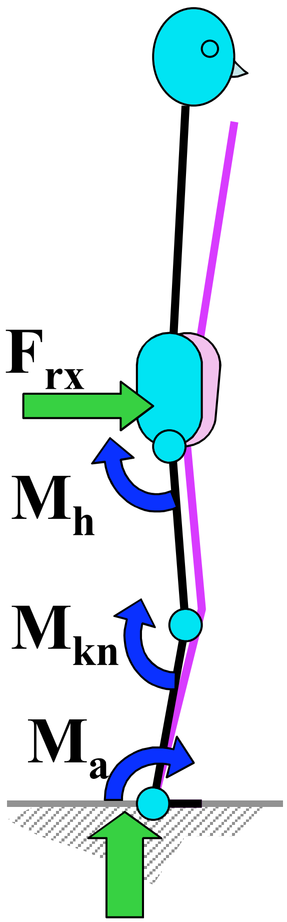
Controller's Goal:

Keep the simulation's center of mass (COM) safely inside the support polygon made by the feet

To Accomplish Goal:

Pick a desired COM and minimize errors by making corrections in the leg actuation

Mid-Level Control: Standing Balance



Balancing force to control COM computed from the balance error:

$$F_{r(x,y)} = k_r\,(err) - b_r\,(\dot{err})$$

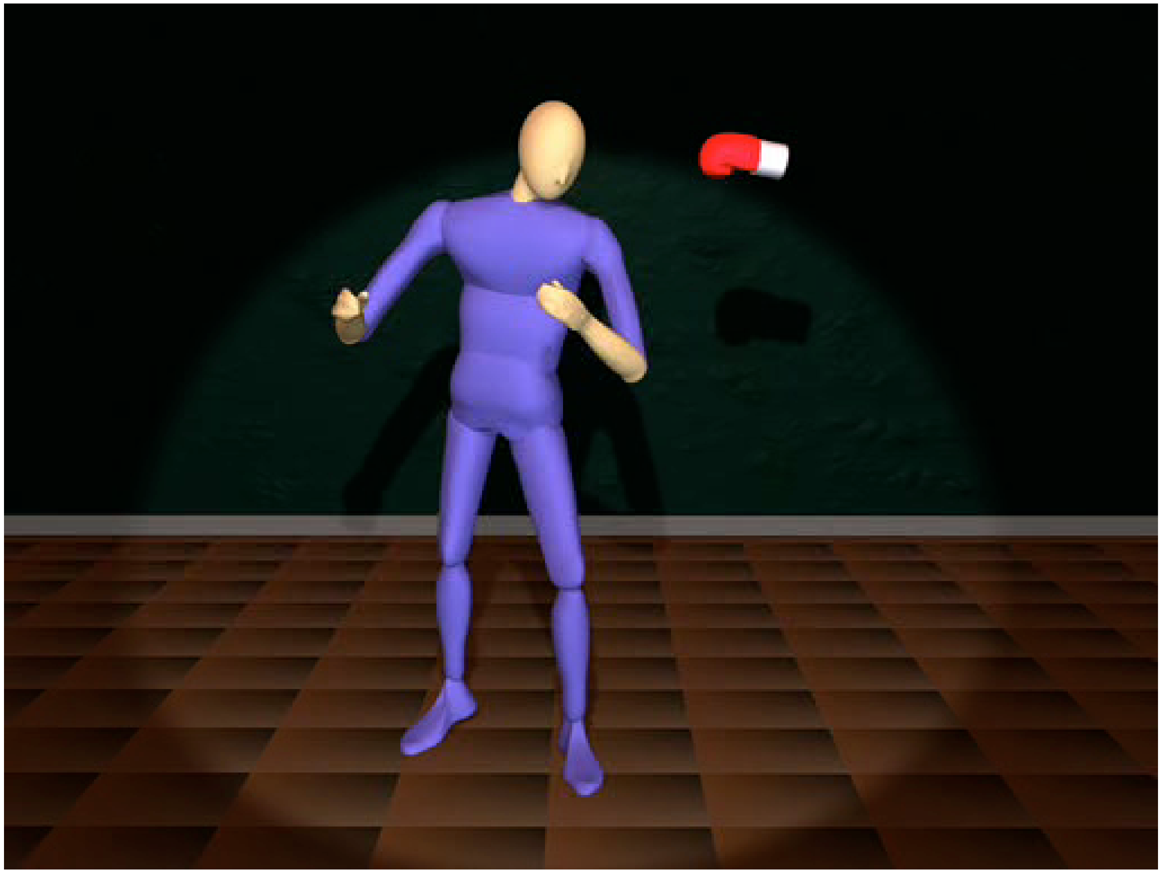
Convert force to torques:

$$M_{(h\text{->}a)} = F_r \times X_{(h\text{->}a)}$$

$$\tau_{balance} = {}^{J}T_0\ {}^{0}M_{(h\text{->}a)}$$

GameDevelopers Conference

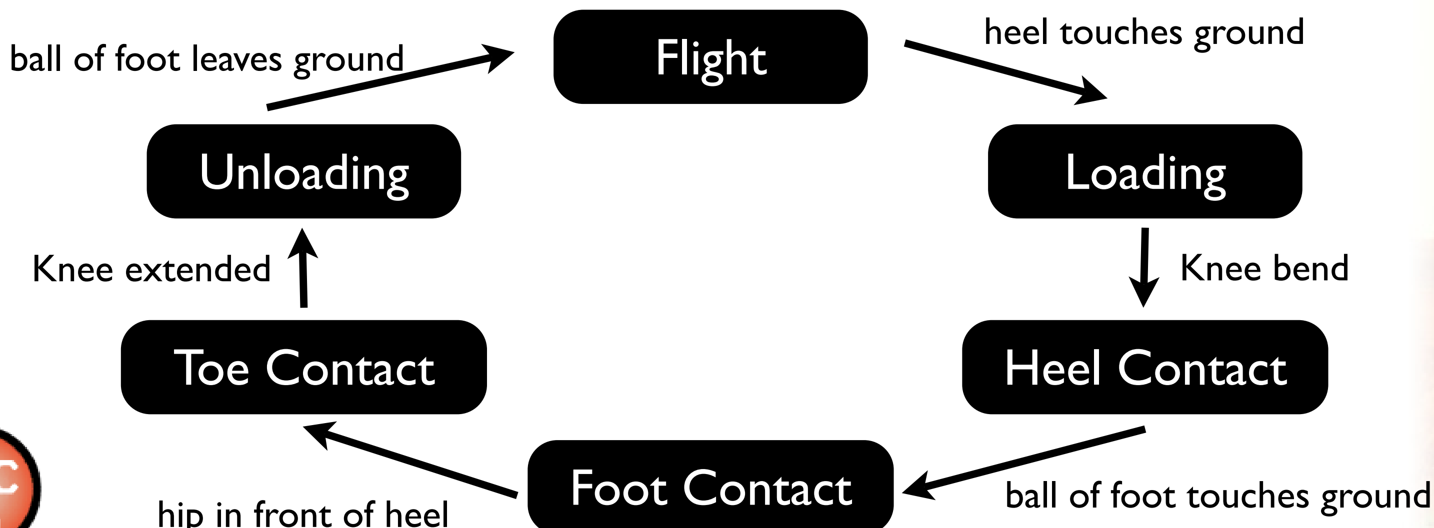
Combine with basic tracking to allow reacting to contact while standing



Breaking Down Behaviors

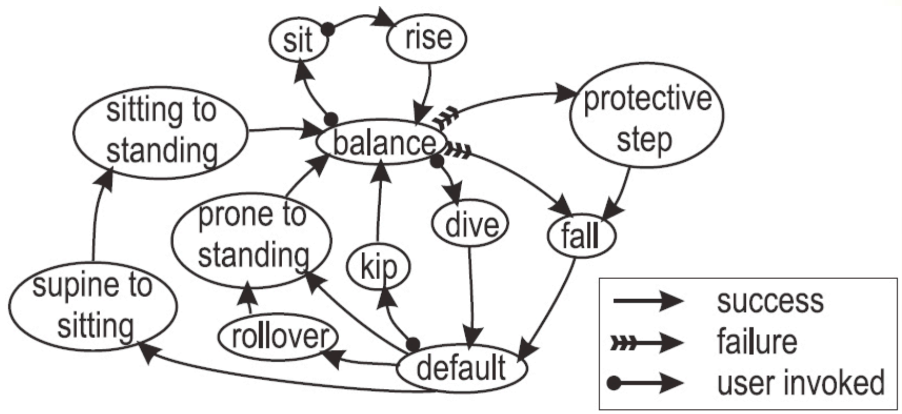
- Finite State Machines are a common representation for motor control states
- Time or event based transitions

Example: Running (see [Hodgins '95])



Complex Behaviors From Simple Controllers

[Faloutsos et. al '01]



- Build basic behaviors
  - sit, stand, fall (pose controllers)
- Classify transitions between behaviors based on conditions
- Supervisor controller swaps between them when conditions met

Simulation References:

[Faloutsos et al., Composable Controllers for Physically-based Character Animation. SIGGRAPH '01]

[Hodgins et al., Animating Human Athletics. SIGGRAPH '95]

[Laszlo et al., Interactive Control for Physically-based Animation. SIGGRAPH '00]

[Mandel, Adding Life to Ragdoll Simulation Using Feedback Control Systems. Game Programming Gems 5]

[Smith, The Open Dynamics Engine. Available at http://ode.org/.]

[Zordan et al., Motion Capture-Driven Simulations That Hit and React. Symposium on Computer Animation '02]

GameDevelopers Conference

Outline

1. Simulating Behaviors
   - Quick Ragdoll Intro (Michael)
   - Controllers (Victor)

2. Executing Transitions (Michael)

   Motion Data ⟷ Simulation

3. Another Take on Transitioning (Victor)

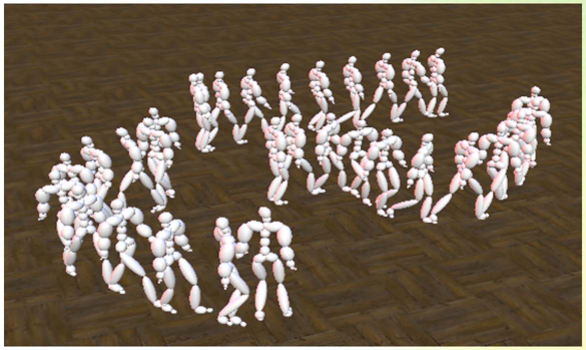
4. Conclusions (Michael)

SAN FRANCISCO CA
MAR 7-11
GDC 05

Executing Transitions

State space of data-driven technique:

- Any pose present in the motion database



State space of dynamics-based technique:

- Set of poses allowable by joint limit constraints
- MUCH larger because it:
  - can produce motion difficult to animate or capture
  - includes large set of unnatural poses

Clearly, some <u>correspondence</u> must be made to allow smooth transitions between the two

Transitioning Between Techniques

When to transition?
How to transition?

Motion Data → Simulation

- Easy. Just initialize simulation with pose and velocities extracted from motion data.

Simulation → Motion Data

- Much harder. No way to predict the ending pose of the simulation...

Correspondence Steps:

1. Identify closest frames of motion
   (Choose most appropriate frame)
2. Drive simulation towards best match

Simulation → Motion Data

1. Identify closest frames of motion
2. Drive simulation towards best match

Problem: Find nearest matches in the motion database to the current simulated motion.

Simple Approach:

1. Data Reduction/Representation

- Automatic keyframe extraction on relevant motion
- Data Representation
  - Joint Positions

2. Process into Spatial Data Structure

- kd-tree works well

3. Search Structure at Runtime

- Query pose comes from simulation
- Pose as nearest neighbor search problem
  - Choose motion most relevant to in-game situation

GameDevelopers Conference

Simulation → Motion Data

1. Identify closest frames of motion
2. Drive simulation towards best match

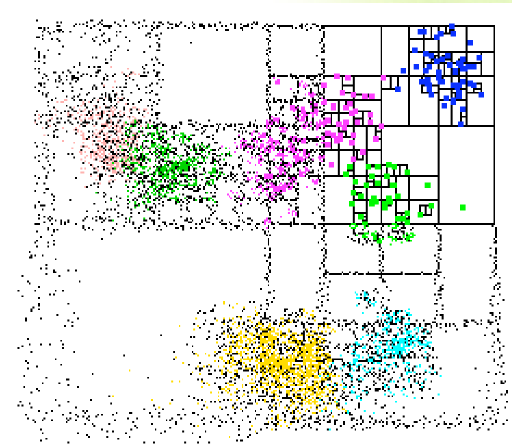
Data Representation: Joint Positions

- Need representation that allows numerical comparison of body posture
- Joint angles not as discriminating as joint positions

Original Joint Positions Aligned Positions

- Ignore root translation and align about vertical axis
- May also want to include joint velocities

A Note on Searching Efficiently...



Approximate Nearest Neighbor (ANN) Search

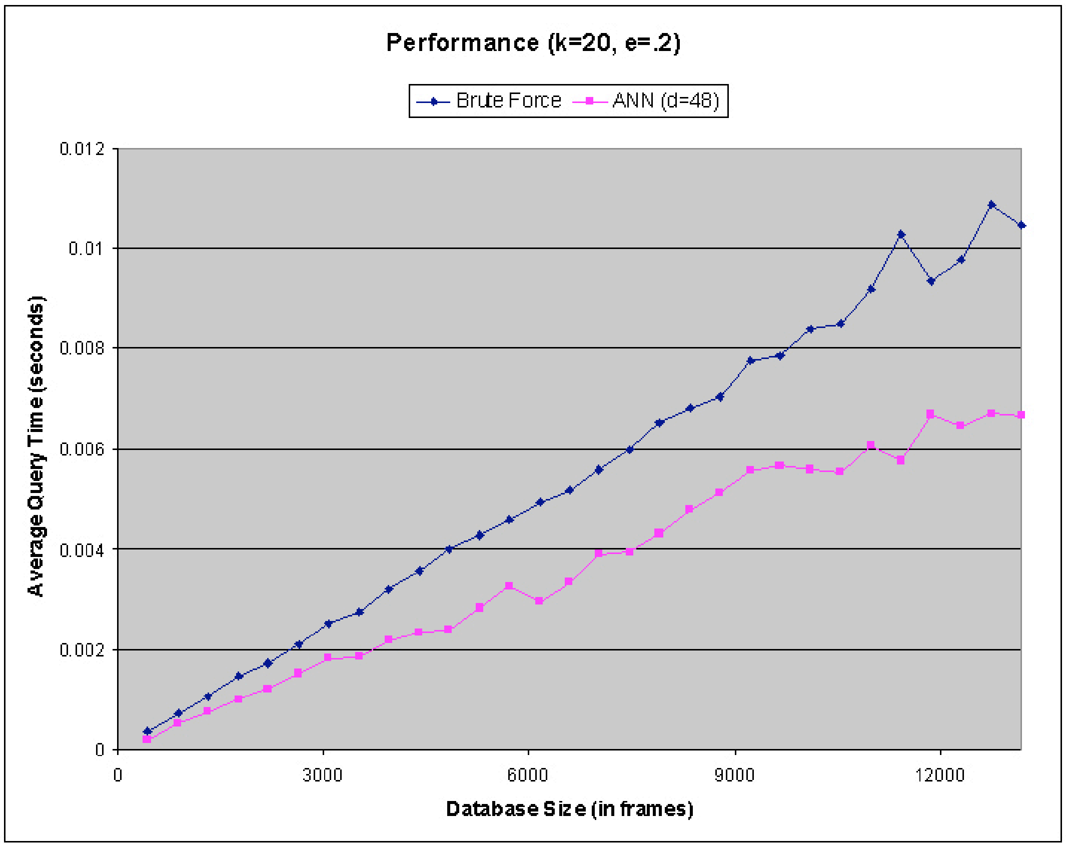
S. Ayra and D. M. Mount. *Approximate nearest neighbor queries in fixed dimensions*. 1993.

- Results guaranteed to be within a factor of $(1 + \epsilon)$ of actual nearest neighbors
- $O(\log^3 n)$ *expected* run time and $O(n \log n)$ space requirement
  - Much better in practice than KNN as dimensionality of points increases
- Balanced box decomposition tree (*bbd-tree*) fits input data tighter
  - Metric trees and spill trees can do even better...
  - Locality Sensitive Hashing (LSH) is also an alternative (see [Liu et. al 2004] and [Gionis et. al 1998])

Free code available at:
http://www.cs.umd.edu/~mount/ANN/

Sample Performance Numbers (using ANN)

- Motions consist of sneaking, running, attacking, idling, etc.



- Averaged over 150 trials
- d=48 for each frame
- epsilon=.2 for ANN

(Between 1/100th and 1/1000th of a second per query)

Speeding it up?

- Human motion is inherently coordinated
  - i.e. frames matching left elbow more likely to match right elbow
- Nearest neighbor algorithms suffer from:

↓ Exponential decrease in performance as the ↑ Dimensionality of data increases

Possible Solution:

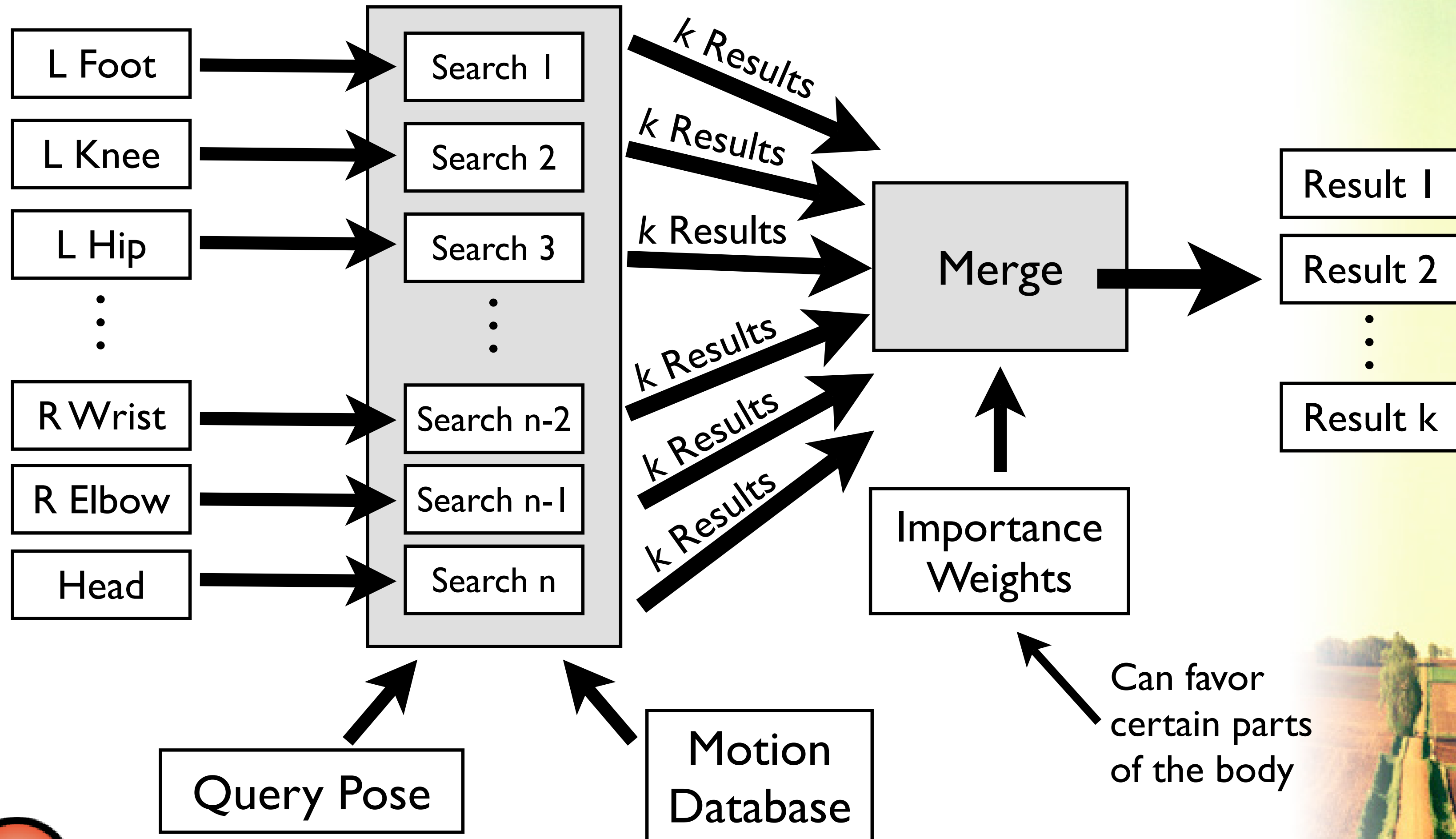
Decouple the joints and search them separately
(keeps dimensionality low),
then combine the results

GameDevelopers
Conference

Simulation → Motion Data

1. Identify closest frames of motion
2. Drive simulation towards best match

Speeding it up: Search Each Joint Position Separately



n 3-DOF searches is faster than one n-DOF search…

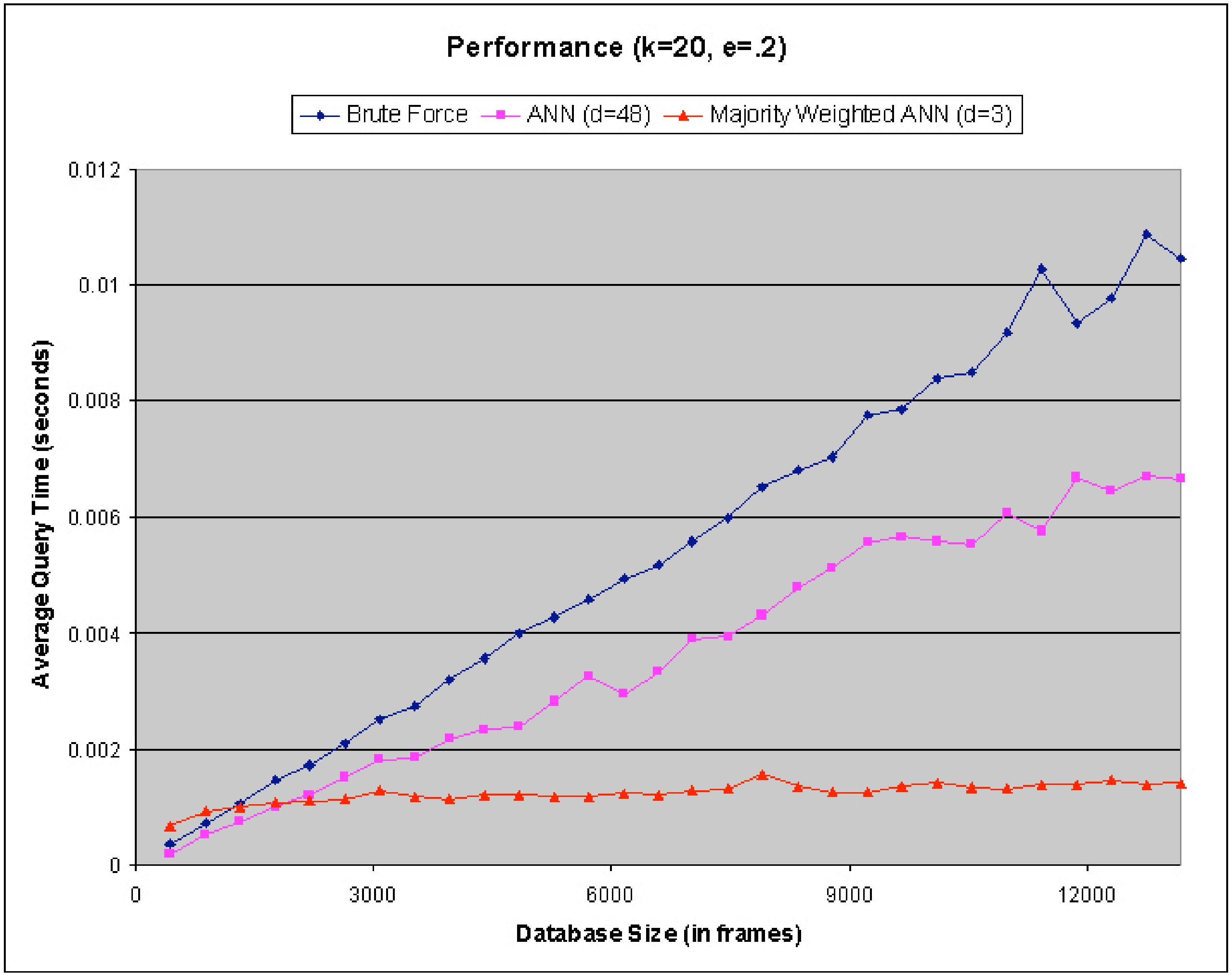
SAN FRANCISCO CA
MAR 7-11
GDC 05

GameDevelopers Conference

Simulation → Motion Data

1. Identify closest frames of motion
2. Drive simulation towards best match

Performance Improvement: Great!



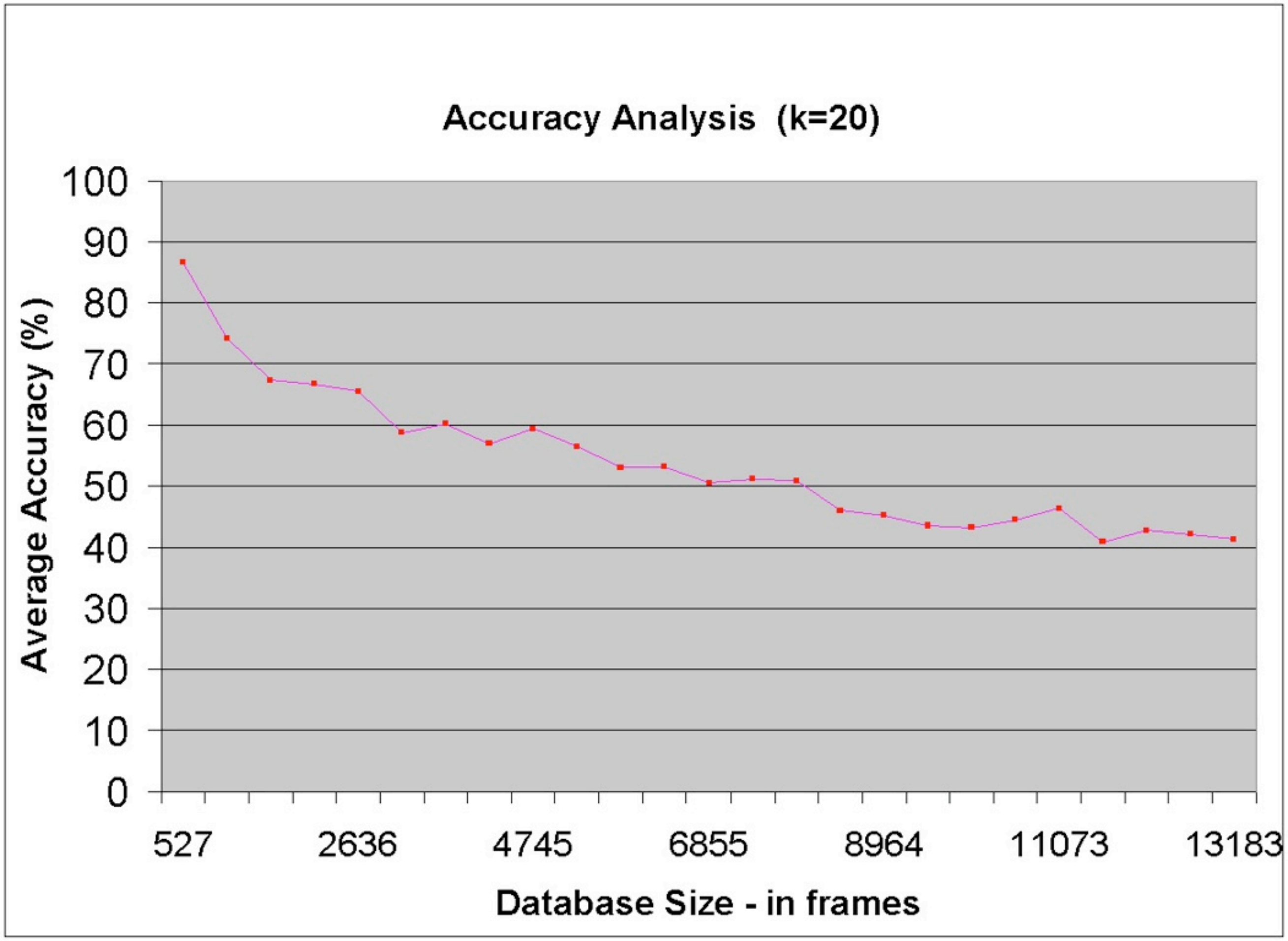
SAN FRANCISCO CA MAR 7-11 GDC 05

GameDevelopers Conference

Simulation → Motion Data

1. Identify closest frames of motion
2. Drive simulation towards best match

Accuracy: Not as great...

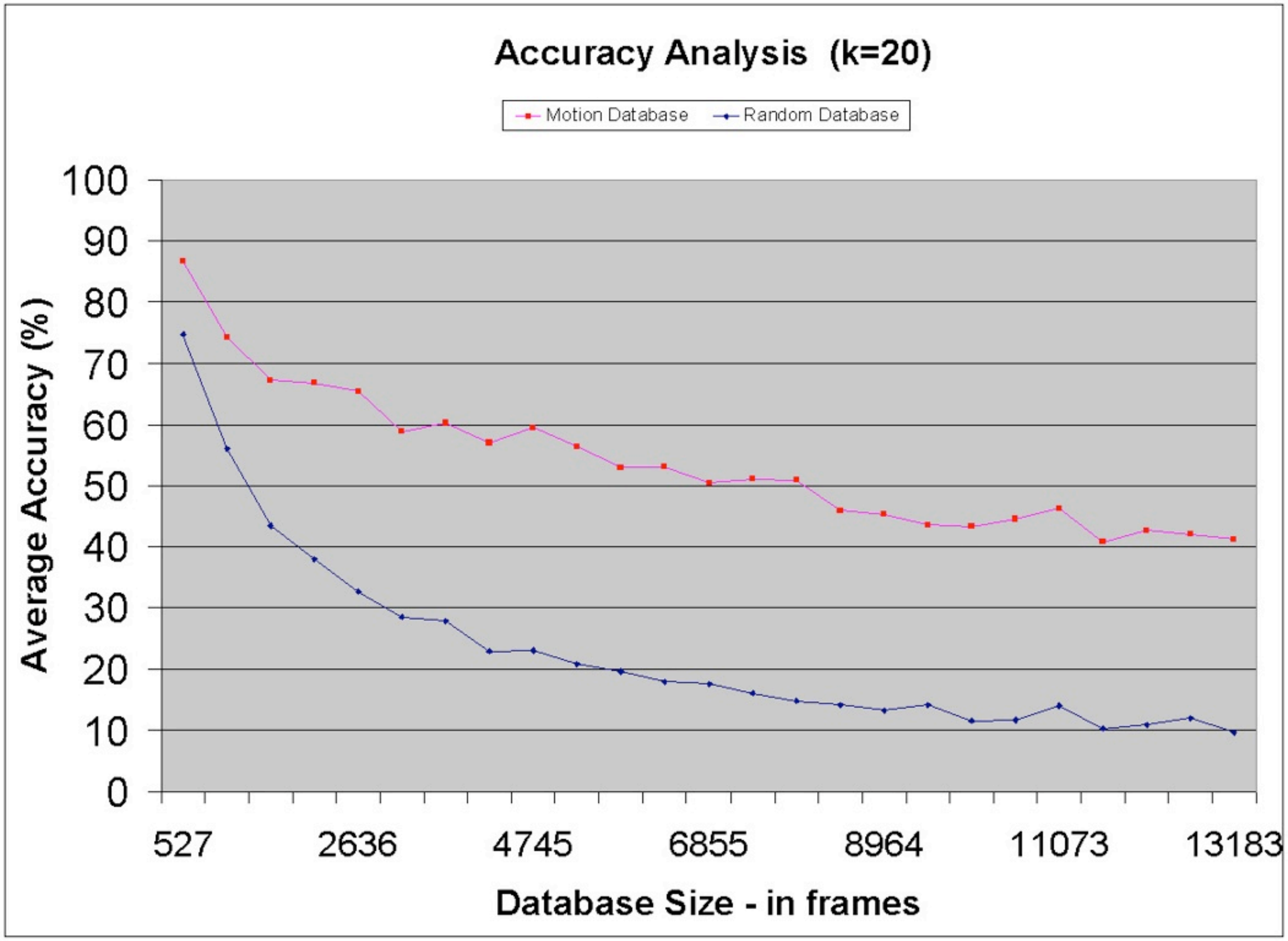


GameDevelopers Conference

Simulation → Motion Data

1. Identify closest frames of motion
2. Drive simulation towards best match

Accuracy: Sanity Check



Successfully utilizes natural correlations in motion data

Speeding it up: Tradeoffs...

- Pair more joints together to increase accuracy
  - Tradeoff performance for increased accuracy
  - Pair least correlated joints for best results...

- Can adjust performance levels for different situations
  - Less visible characters use faster, but lower quality results
  - Could also just play with epsilon parameter to increase performance

- Just because you don't find the best match, doesn't mean you don't get a good match
  - Wash out the error when you drive simulation toward match

Where are we now?

What's Missing?

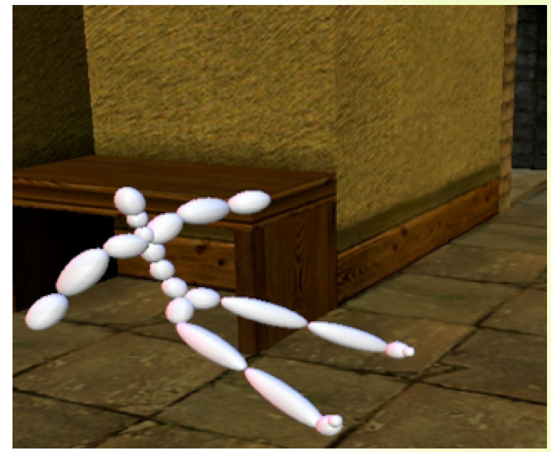
1. The fall lacks life
2. Transition has blending artifacts

Data-Driven: Walk → Dynamics: Simulated Fall → Data-Driven: Get Up → Idle

Fixing the Transition...

Problem: At the time a transition is requested, the simulation is NOT likely to be in a posture contained in the motion database

(It IS likely, however, to be interacting closely with the environment)



How can we get the simulation to settle near the best matching motion data?

Can we maintain physical constraints between the body and the environment?

Fixing the Transition...

Solution: Settle Controller

Actuate joints using a special PD controller to settle the simulation near selected motion data

- Pose controller uses search result as target joint angles
- A physically grounded alternative to blending
  - Avoids object interpenetrations and foot sliding...
- Complex situations might be handled by more specialized controllers
- Can always finish it off with blending if you get stuck...

Adding Life to the Falling Motion

One Possibility: A Simple Pose Controller

- Look at initial conditions of an impact and choose initial desired reaction from a database of example poses
- May update desired pose as simulation evolves - still totally data-driven (and artist directed)

This can work well, but might not be as dynamic as we'd like.

Another Solution:
A Continuous Controller

Adding Life to the Falling Motion

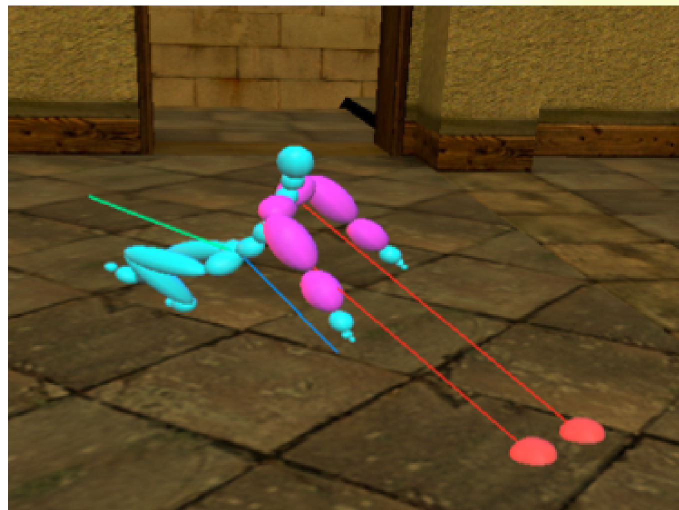
Goal:

Reasonably approximate what humans do during a *full* loss of balance (biomechanically inspired)

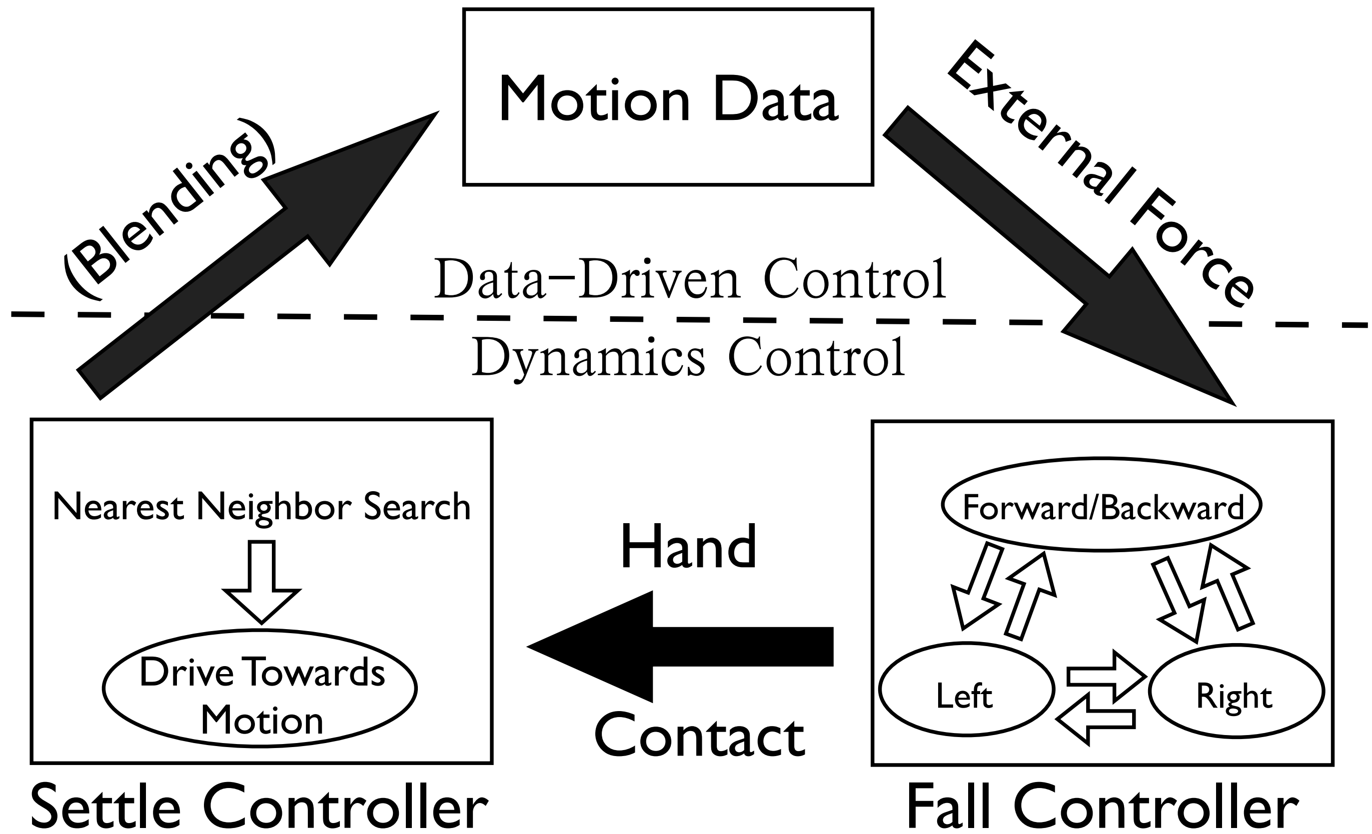
highly effective motor control strategies → hard to model

Possible Approach:

- Track predicted shoulder landing locations with arms
- Direction the body falls determines which arms do tracking
  - Can change as simulation evolves
  - Properly tune body gains...

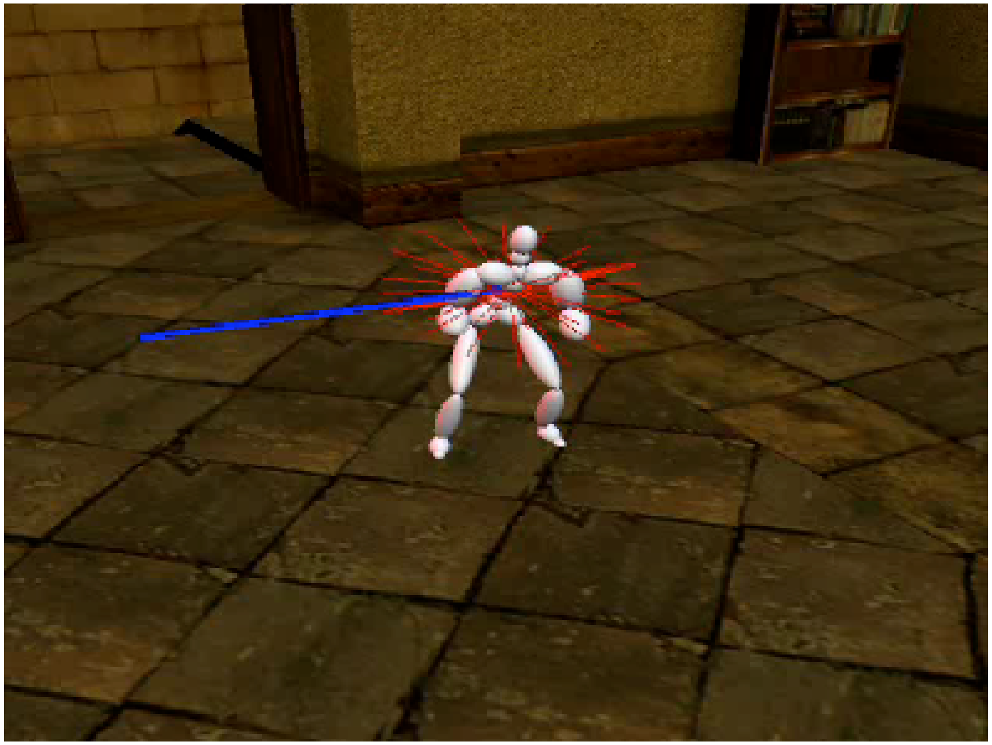


Prototype System Overview



1. Motion Data Control
2. Transition to Simulation
3. Simulated Falling Behavior
4. Transition to Motion Data
5. Motion Data Control

Results



Data-Driven	Dynamics	Data-Driven	
Idle →	Simulated Fall →	Get Up →	Idle

GameDevelopers Conference

Results: Extending the Fall Controller...

Data-Driven	Dynamics	Data-Driven
Idle →	Simulated Fall and Roll	→ Get Up

SAN FRANCISCO CA
MAR 7-11
GDC 05

Outline

1. Simulating Behaviors
   - Quick Ragdoll Intro (Michael)
   - Controllers (Victor)

2. Executing Transitions (Michael)

   Motion Data ⟷ Simulation

3. Another Take on Transitioning (Victor)

4. Conclusions (Michael)

GameDevelopers Conference

How do we make physically-based transitions while taking advantage of mocap?

Basic Idea:

1. Start from mocap
2. Move to simulation when interaction takes place
3. Perform graph-like search
4. Return to mocap as soon as possible!
   (i.e. BEFORE hitting ground or straying too far from mocap)

The Problem:

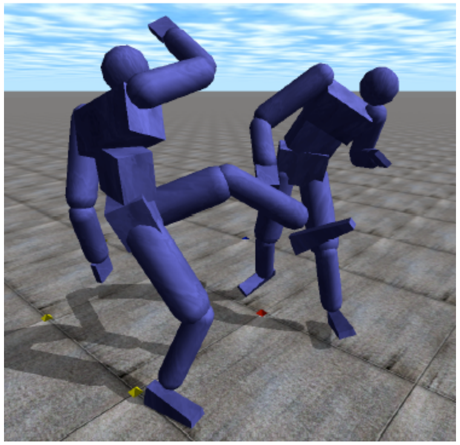
How do you transition from simulation to mocap elegantly?

SAN FRANCISCO CA MAR 7-11 GDC 05

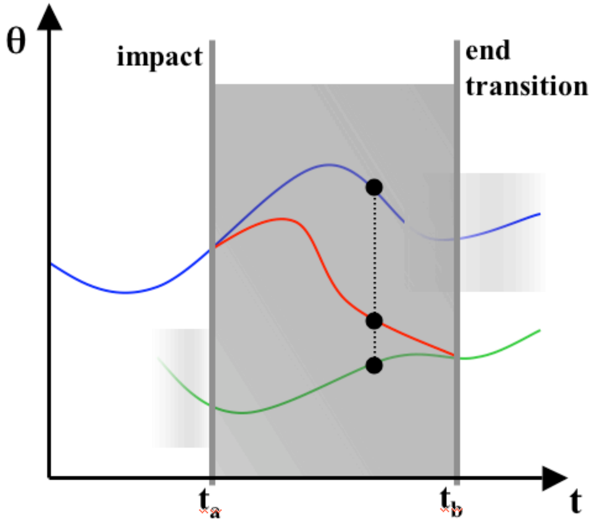
GameDevelopers Conference

Physically-Based Transitions Following Impacts, With Motion Capture

[Zordan et. al '04]



- Apply impact forces to sim
- Search using *window*-ing to find clip post interaction
  (see [Kovar et. al '02])
- Actively track the motion clip as it transitions, to get the posture in place with joint torques
  - Add global positions using forces to position character



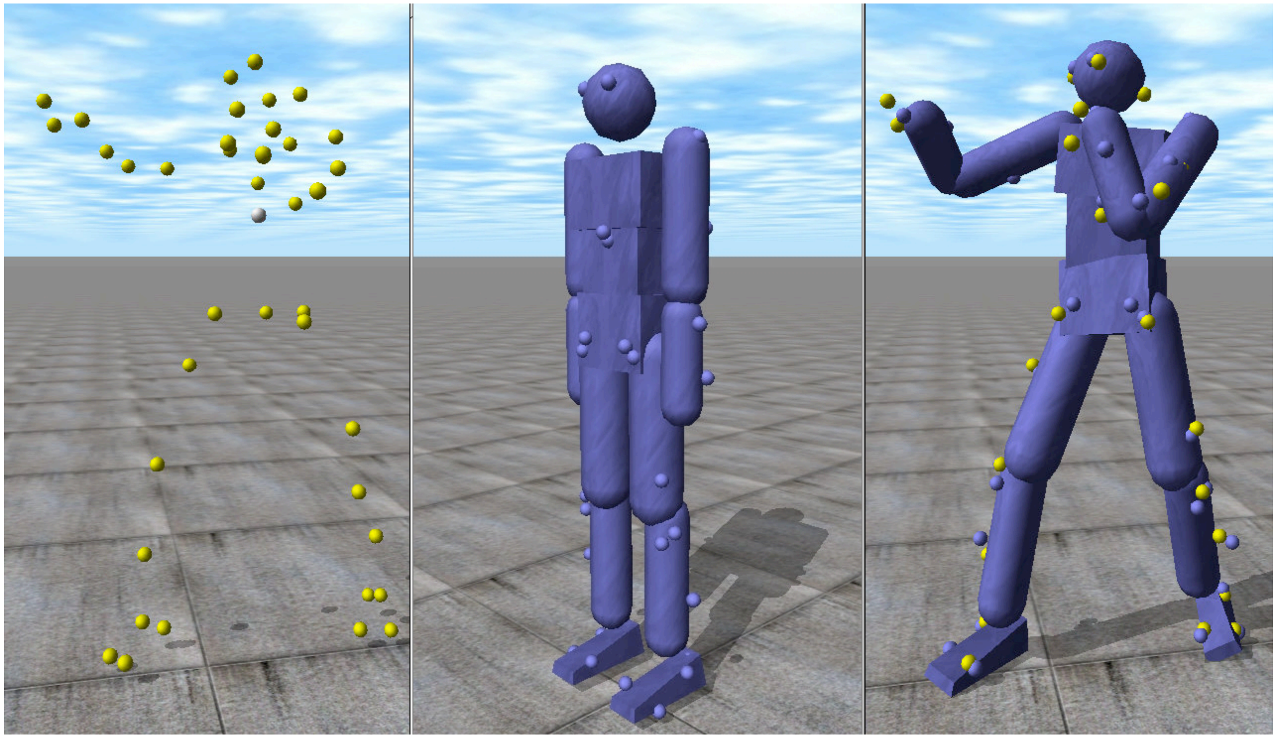
SAN FRANCISCO CA
MAR 7-11
GDC 05

Physically-Based Transitions

Motivated from using a sim. to map data (Zordan & Horst '03)

Use same approach here to create "docking" forces

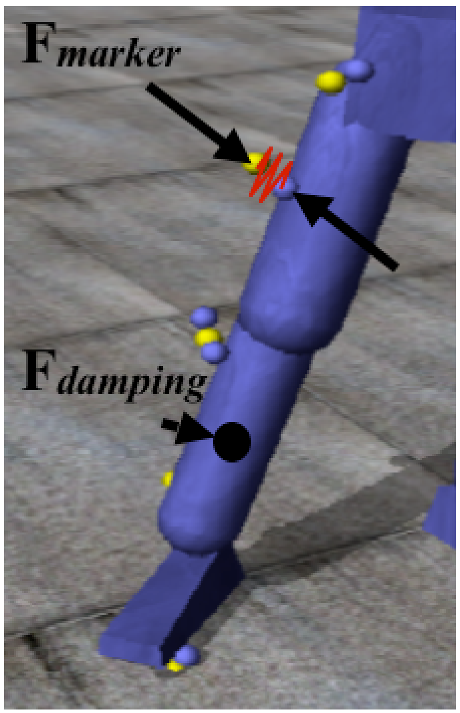
Optical Data + Simulation Posture



GameDevelopers Conference

Physically-Based Transitions

- Forces pull (or dock) character into place
- Starting from virtual 'landmarks,' we guide the simulated bodies using *intuitive* forces



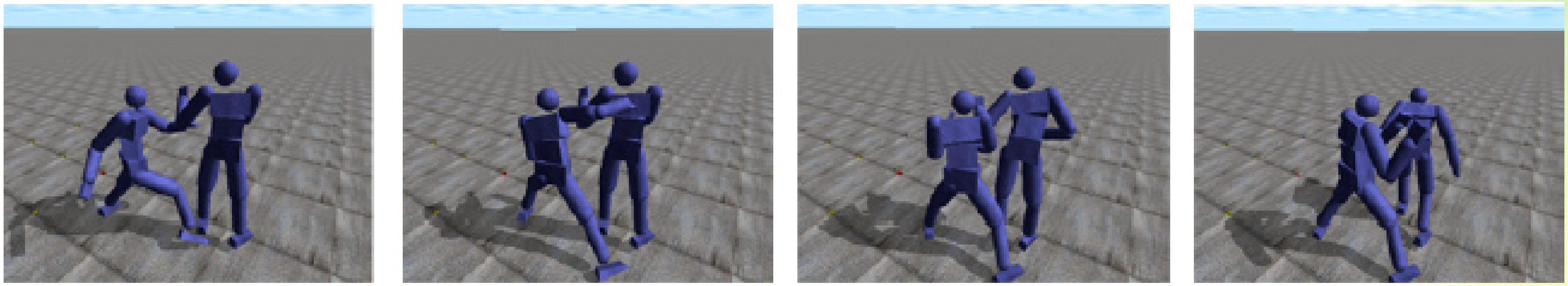
Springs pull the simulation to the marker data

$$F_{marker} = -k_f X_{error}$$

Body forces damp motion

$$F_{damping} = -b_f V_{body}$$

Physically-Based Transitions



Internal torques mimics human reaction

External forces minimize error while not breaking the physical engine

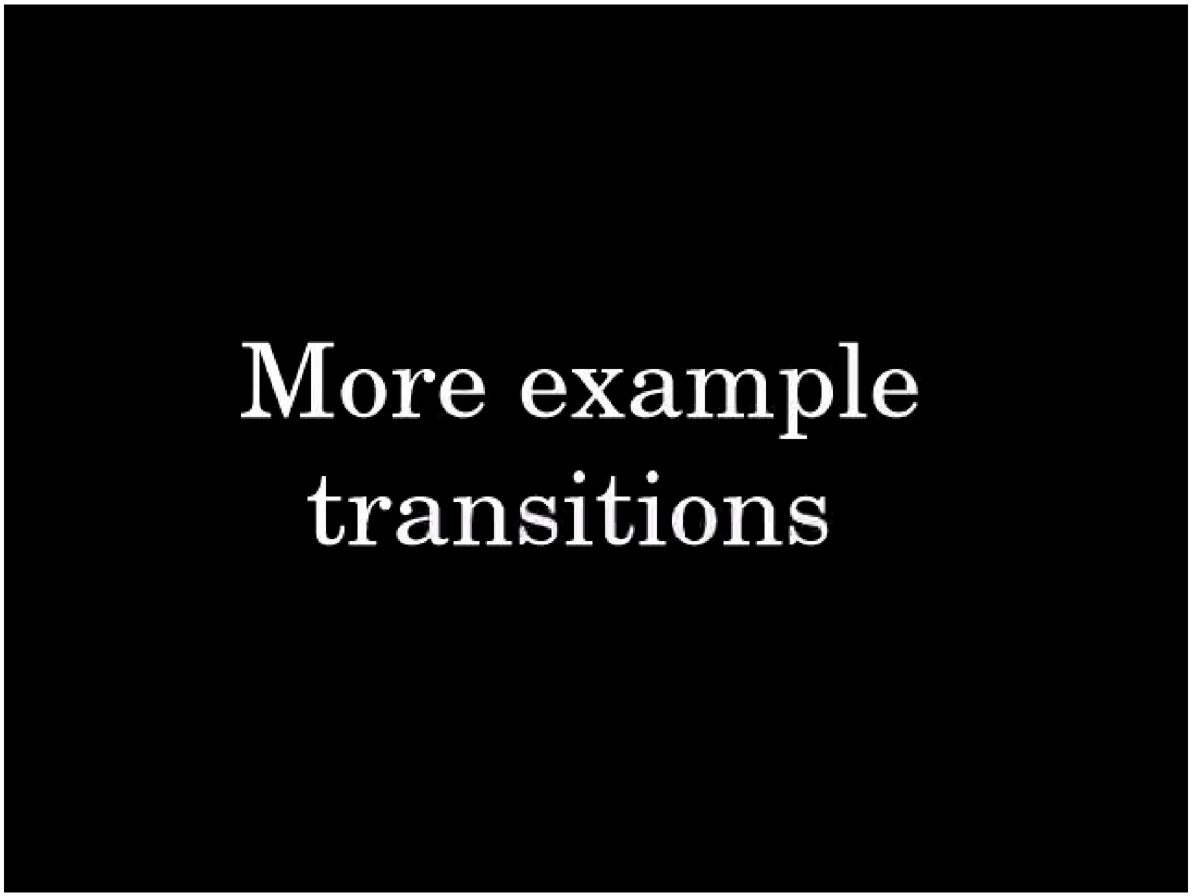
- This method combines mocap *"pose-clip"* while the interaction forces are still taking place...

- Doesn't guaruntee a perfect match at the end, but we manage this with blending!

Animation Examples

GameDevelopers Conference

Animation Examples



More Recent Results

Outline

1. Simulating Behaviors
   - Quick Ragdoll Intro (Michael)
   - Controllers (Victor)

2. Executing Transitions (Michael)

   Motion Data ⟷ Simulation

3. Another Take on Transitioning (Victor)

4. Conclusions (Michael)

Making it Practical...

Games need to <u>guarantee</u> robustness

- Games can sacrifice physical realism for robustness/ speed - know when using simulation is appropriate!
- Start simple - pose controllers with artist predefined reactions
- Specify only the DOFs necessary
  - Let the natural dynamics of the system guide the behavior
- Fake things like balance control
  - Make the ground "stickier"
  - External balancing forces to keep the body upright
  - Consider simulating only some of the body

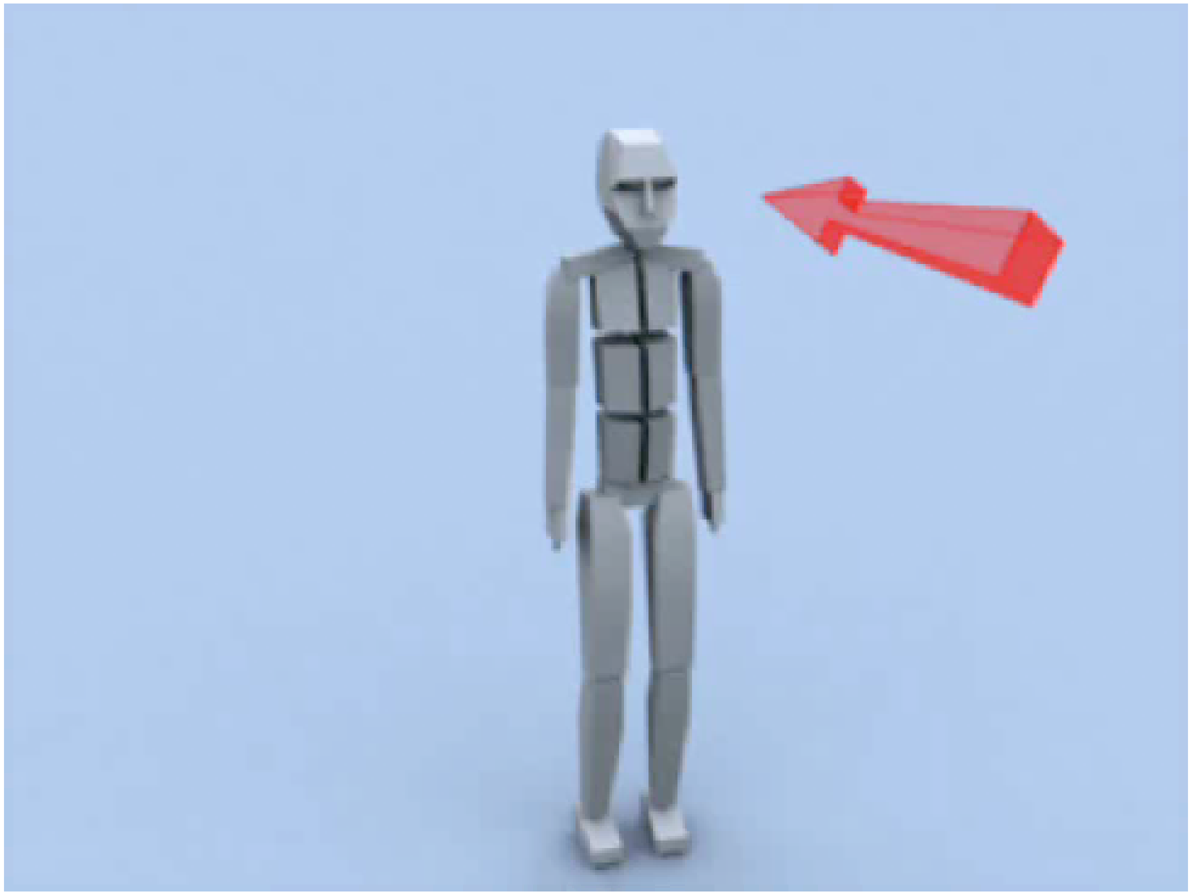
From Research...

Reil and Massey '01, Oxford University

From Research to Robustness...

Natural Motion's *Endorphin*

http://www.naturalmotion.com/



Hybrid System Discussion

- Hybrid system supporting roundtrip transitions between motion data and simulation
  - Choose best approach for current in-game situation
  - Easy to add to your existing skeletal and ragdoll systems

- Support future goal of simulating everything with ability to fall back on pre-recorded motion
  - Bottom-up approach allowing incremental additions to simulated behavioral repertoire

Special Thanks

Carnegie Mellon Motion Capture Lab

(free mocap data available at http://mocap.cs.cmu.edu)

Jessica Hodgins

Victor Zordan

Moshe Mahler

Rich Carson and Iikka Keranen

Questions / Comments?