Real-Time Physics-Based Fight Characters

The Fundamental Workings and Principles – Reinventing the Wheel

Ben Kenwright[+]
www.xbdev.net

Abstract

In this paper, we present a practical physics-based character system for interactive and dynamic environments. It uses a number of straightforward, computationally efficient, and conditionally stable techniques to produce responsive, controllable, and interactive character avatars. We describe different physics-based simulation techniques to produce interactive animations and present a detailed description of pitfalls and limitations. For example, our system demonstrates the fundamental principles of balancing, joint torque calculations, and mass-properties that we combine in an application to show a controllable real-time character-character fight game. We also demonstrate the plausibility of our approach through numerous important simulations to illustrate the robustness and advantage of our system.

Keywords: Character Animation, Balancing, Inverse Kinematics, Inverted Pendulum, Responsive, Real-Time, Avatar

1. Introduction

Generating real-time physics-based characters for interactive environments is exciting and challenging **[FVDPT01][ML10] [KLK04][TGTL11]**. The challenges include not only making the characters respond realistically to unforeseen circumstances but also to have the characters use their joint torques intelligently to recover and behave in a natural life-like way. Typically, character systems use animation libraries or simplified physics-based models to produce realistic motions. Some of these solutions are propriety owned or are complex to implement and difficult to customize to new situations and character types (e.g., non-biped creatures). For example, Euphoria **[EUPH12]**, Havok **[HAVO12]**, and HumanIK **[HIK12]**, provide commercial middleware runtime solutions for creating believable, interactive character animations for games.

A real-time physics-based character system, at its heart, revolves around the fundamental problem of control. Since the laws of rigid body motion are well defined from classical mechanics **[KM12]**, we are able to construct articulated body characters that respond and move realistically due to forces and torques. The problem is determining joint torques for specific actions (e.g., jumping, punching) while including feedback for balancing and realism are all significant factors for making the physics-based character system a viable usable solution for virtual environments, such as games.

This paper describes a complete physics-based character system for controlling a wide variety of articulated avatars in real-time environments. The goal of this system is to produce physically realistic results using a general and practical combination of techniques. For example, we represent each character as an interconnected set of rigid body links (see **Figure 3-1**) that our system controls through forces to mimic real-world movements. Our system allows the user to customize and tune parameters to produce a wide range of diverse motions from a minimal amount of data.

We evaluate and demonstrate our system by implementing a real-time character-character fight game that allows us to show a variety of interactive and controllable actions, for example, punching, and kicking. In addition, we compare our system with a pure interpolated solution with no-interactive physics-based feedback. In conclusion, our results show how a dynamic character-system can dramatically increase the realism and emersion factor of a game while providing numerous added benefits (e.g., responding to unpredictable situations realistically).

1.1. Motivation

The motivation for this paper is aimed at moving virtual game characters away from traditional hard-coded inflexible key-frame based techniques to more intelligent physics-based solutions. Moreover, we focus on a practical and realistic real-time method that can be implemented and demonstrated using current hardware technology for interactive environments, such as games; whereby, virtual characters move realistically due to their joint torques and are influenced by force disturbances, such as gravity.

1.2. Contribution

The contribution of this paper is the demonstration and explanation of the creation of physics-based game characters. We use uncomplicated 2D examples in conjunction with practical real-world cases to show the reader how they would go about building a physics-based character animation system for real-time environments, such as games. Furthermore, we demonstrate the viability of our method by incorporating it into a real-time fight game demo. In summary:

- Practical physics-based character system
- Demonstrate a character-character fight game

2. Related Work

Physics-based biped characters with responsive characteristics have been investigated across numerous fields (e.g., computer graphics and robotics). We discuss and explain some of the important research in each field that has led to the creation of more dynamic and interactive character solutions.

A. Computer Graphics

To begin with, Komura **[KLK04]** simulated reactive motions for running and walking of human figures, while Zordan **[ZH02]** simulated characters that reacted automatically to impacts and smoothly returned to tracking. This work was later extended **[ZMCF05]** for combining existing human motion capture data to produce physics-based responsive motion segments that played for varying force disturbances (demonstrated using a martial art test bed).

+e-mail: bkenwright@xbdev.net

Meanwhile, Shiratori [SCCH09] developed a novel controller that could generate responsive balancing actions; while Tang [TPZZ06] modified interactive character motions for falling with realistic responses to unexpected forces; subsequently, McCann [MCP07] presented a method for blending various motion capture segments that emulated responsive character movements. Additionally, Arikan [AFOB05] did similar work on generating how people respond to being pushed around. While some controllers can generate responsive motions, they can be robot-like and reparative. However, Kenwright [KEN12b] extended Perlin's [PG96][PERL95] earlier work on combining coherent random noise with an intelligent physics-based model to produce more life-like and non-repetitive character motions.

In the same way, a foot-placement model was presented by Singh [SKR11] for reconstructing a character's motions. The research focused on a simplified footstep model for simulating crowds by means of circular foot support region approximations in local pelvis space to generate foot position and orientation information. Similarly, Wu [WM08] presented a method for controlling animated characters by modifying their foot placement information so that it was physically correct. Furthermore, Kenwright [KDM11] demonstrated a responsive inverted pendulum model with an upper body postural feedback control system. This work was later built upon to provide a more controllable model using an approximate foot support area [KEN12b] similar to Singh's approach [SKR11].

B. Robotics

Appreciating some of the relevant work in the field of robotics that contributed to the development of responsive biped controllers, we outline a few interesting and important papers. Shih [SGL93] developed a straightforward model for enabling characters to respond to small disturbances; while Stephens [STEP07] and Pratt [PCDG06] developed controllers that could generate motions for recovering from a range of push disturbances.

3. System Overview

The overall system is composed of numerous small and simple pieces that when combined presents us with a powerful, flexible and controllable character-based system with interactive and dynamic properties. The main pieces of our system are:

- Character Structure (*i.e., Links, Angles*)
- Physical Properties (*i.e., Masses, Sizes*)
- Poses (*i.e., Standing, Punching*)
- Torques and Forces (*i.e., Integration*)

Furthermore, we discuss and examine various sub-details, such as handling uneven terrain, balancing, stability, strength, and joint limits.

We include feedback to ensure postural upright standing and walking are able to work on un-even terrain and dynamic situation such as walking up and down a slope. We present a number of visualizing techniques to help the developer identify issues and limitations through graphical feedback information. For example, inverted pendulum wheel (number of steps, step size and angle, small steps, big steps), angular limits for the inverse kinematic solver.

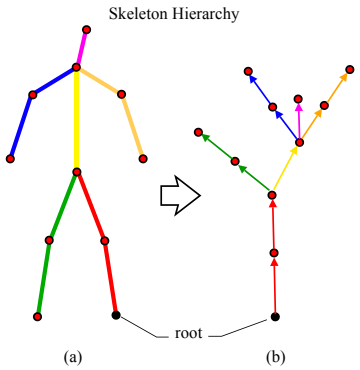

Figure 3-1: (a) Skeleton model, and (b) Tree Hierarchy.

Since our character's model was constructed from articulated rigid body links, it would automatically react appropriately to hits and punches. For example, having the character's posture respond appropriately to disturbances, such as being punched in the stomach, feet sliding due to being pushed, and mimicking a real-world human reaction.

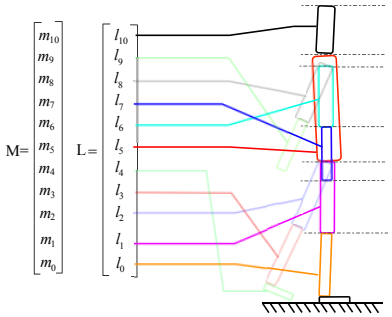

Figure 3-2: Illustrating the character's limb lengths and masses.

4. Character Structure

As shown in **Figure 4-1(a)** our character is a set of interconnected rigid links. We connect the links in a hierarchy formation with each joint having a child-parent relationship and the foot or pelvis as the root-base. We store each joint's orientation relative to its parent. For the simple 2D biped shown in **Figure 4-1(b)** we have 10 separate joint angles. The joint angles are placed in a single vector matrix (see **Figure 4-1(c)**) to make it easier to manage and manipulate.

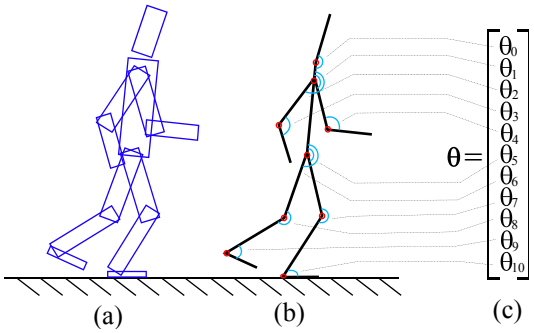

Figure 4-1: (a) Character modeled as a connection of rigid bodies. (b)-(c) The decomposition of the character joints' and their matrix representation.

The joint angles updated on a frame-by-frame basis. Hence, we specific a set of variables for each link, which are:

- Link sizes and dimensions (constant)
- Joint limits (i.e., min/max threshold) (constant)
- Joint angles (e.g., **Figure 4-1(c)**)

The parameters can be customized to create a crowd of different character types (e.g., short, fat, thin) simply by modifying the array of constants (e.g., link lengths, limits). The default root of the

character hierarchy was the left foot (see **Figure 3-1**). However, on the fly, we could change the root to either the pelvis or the right foot based on the situation; for example, which foot was the support foot or if the character were not in contact with the terrain, we would make the pelvis the root.

4.1. Mass Properties

The default mass properties for our biped character were based on real-world statistical models of the human body by Clauser **[CMCY69]** and Dempster **[DG67]**.

For example, we assumed the default total body weight was approximately 60kg for the ratios in **Table 1**.

Head +Trunk	52.2%	Head	8%
Arm	3.2%	Body	28%
Forearm	2.1%	UpperArm	17.4%
Hand	0.8%	LowerArm	15.7%
Thigh	10.9%	Hand	10%
Shank	4.7%	UpperLeg	23%
Foot	1.8%	LowerLeg	25%
		Foot	3.7%

Table 1: An approximate human male's body weight proportions.

For example, we show how the individual limbs are constructed from rigid body rectangles to represent the individual components of our articulated character model in **Figure 4-2**.

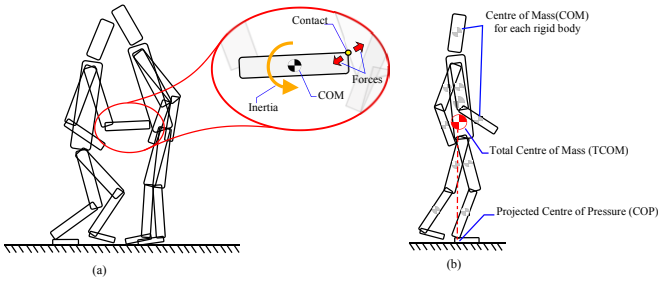

Figure 4-2: (a) Character interaction with each limb having its own mass, inertia properties, and contact information; (b) Sum of the individual rigid body limbs contribute to an overall set of information for balancing and stepping.

We calculated the total centre of mass position by summing the individual joint centre of mass positions and dividing them by their total mass as shown in Equation **4-1**.

$$COM_{total} = \frac{\sum_k m_k COM_k}{\sum_k m_k} \qquad \textbf{4-1}$$

where COM_{total} represents the position of the total centre of mass, m_k is the mass of link k, and COM_k is the position of the centre of mass of link k.

	Lengths	*Mass*
body	470	15710
head	190	7910
upper-leg	390	10050
lower-leg	550	3510
upper-arm	300	2180
lower-arm	430	1830
(with hand)		
foot	125	1110

(length in millimetres, mass in grams)

$$M = \begin{bmatrix} 7910 \\ 1830 \\ 2180 \\ 1830 \\ 2180 \\ 10050 \\ 1110 \\ 10050 \\ 3510 \\ 10050 \\ 3510 \end{bmatrix} \quad L = \begin{bmatrix} 470 \\ 430 \\ 300 \\ 430 \\ 300 \\ 470 \\ 125 \\ 550 \\ 390 \\ 390 \\ 550 \end{bmatrix}$$

Figure 4-3: Example default masses and lengths.

5. Key-Poses (Control-Poses)

We use a number of key-poses (also known as control-poses) so the player can make the avatar perform a wide range of actions. Each key-pose consists of an array of joint angles (e.g., see **Figure 14-2**). We calculate the joint torques for the rigid body skeleton so that it achieves the selected key-pose. We calculate the joint torques using a proportional derivate (PD) feedback controller (see later section for details).

The repertoire of action poses for our fight characters are: (see also **Figure 14-1** and **Figure 14-2**)

• Stand	• Duck
• Kick	• Jump
• Jab-Punch	• Step-Back
• Protect	• Step-Forward
• Upper-Punch	• Lean-Back
• Kick	• Lean-Forward

Alternatively, for a complicated character model with a greater number of degrees of freedom (DOF) (e.g., 3D), the repertoire of actions would ideally be taken from a database (e.g., motion capture animation library).

6. Foot

The foot is an important part of characters stepping movement and believability. However, the feet were added as a pure cosmetic afterthought. To ensure the feet reacted and moved realistically we used a spring-damper connection (as shown in **Figure 6-1**). Hence, the orientation angles' for the feet was calculated at run-time, and the array of joint angles had no influence. Moreover, as the feet are lifted and placed upon the ground, they move and respond appropriately to mimic real-world characters stepping.

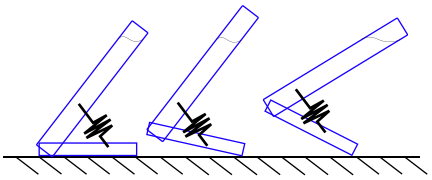

Figure 6-1: Foot ground interaction is emulated with a spring-damper connection.

6.1. Interpolation (Linear, Bezier Curve, Quaternion)

Creating life-like stepping motions makes the character more believable and life-like, while helping to maintain the connection between the player and the avatar. Artefacts, such as slipping and floating can break this connection and produce unnatural and bizarre movements. Hence, we generate foot placement information for balancing and locomotion and combine them with with Bezier splines to create smooth natural-looking stepping transitions.

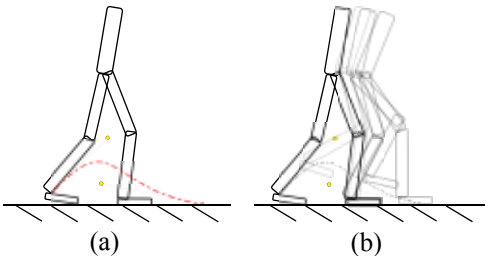

Figure 6-2: Foot stepping trajectories are calculated using Bezier curves so they appear more natural, (a) trajectory paths for a simple forward step, and (b) blended motions following the trajectory.

7. Proportional Derivative (PD) Controller

The proportional derivative controller **[TLT11]** (also known as an angular spring) is the method; we use to generate joint torques. We have the current pose, and the desired pose. This enables us to

calculate the error. The error is used to calculate the joint torque. The error is combined with a spring constant and a damping constant to control the stiffness and strength of the joint.

In conjunction with the set of constants that specify limb sizes and limits, we also specify a set of joint spring and damping coefficients. The PD coefficients determine how responsive and fast a character is to achieving a specific pose. For example, very low spring constant with a high damping constant would produce a very slow and stiff character. The conventional PD controller used in the majority of numerical simulations is given in Equation 7-1.

$$\tau = (q_c - q_d)k_p - \dot{\iota} \qquad \textbf{7-1}$$

where q_c, q_d are the current and desired orientation, $\dot{\iota}$ is the current angular velocity, and k_p, k_d are the proportional and derivative gain constants respectively.

We can create an assortment of character movements from a single model by specifying different coefficient gain constants. However, as shown in **Figure 7-1** they can produce stiff, oscillatory, jittery, and even highly undesirable unstable movements.

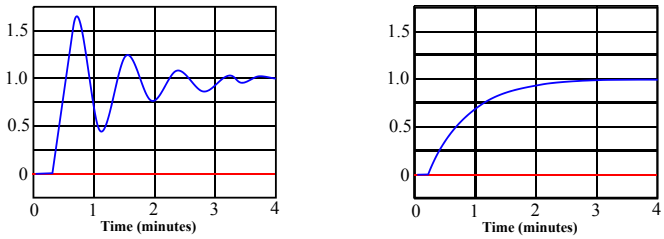

Figure 7-1: (a) under-damped, and (b) over-damped proportional derivative convergence response.

For simple systems with one or two links (e.g., **Figure 7-2**) we can easily calculate the damping coefficients to produce a desired response (e.g., critically damped). However, for more complicated system structures with multiple interconnected joints, we need to use an iterative approach in conjunction with human intervention to find the coefficients. While we ideally desire a critically damped system in practice, it is rarely obtainable. Hence, for our complex character structure, we leaned towards a more over-damped solution since it provided the most stable result while producing the most natural-looking character movements.

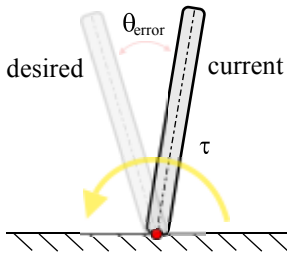

Figure 7-2: Simple PD example to provide the angular responses for Figure 7-1.

7.1. Toy Example

Exploring how the different masses, lengths, gain, and damping coefficients affect a complex articulated structure is crucial. Since the variables are responsible for the responsiveness and hence the feel of the character's movement. As shown in **Figure 7-3**, we experimented with rigid body link configurations to get a feel for coefficient values and acceptable joint and force tolerances.

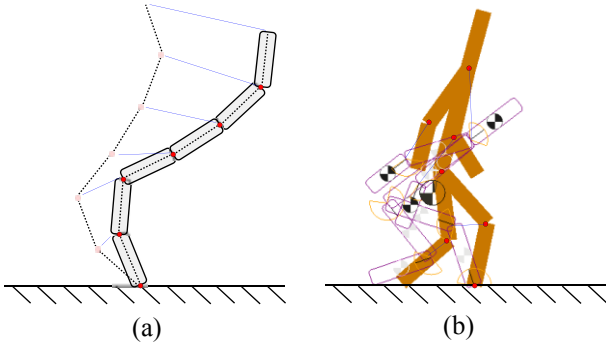

Figure 7-3: (a) Single linked chain inverse kinematic (IK) solution that we dragged around with the mouse cursor to experiment with the PD controller coefficients, and (b) character rigid bodies' following the fixed key poses.

8. Semi-Explicit Integration

We used a straightforward and efficient integration method known as the semi-explicit Euler method (also called symplectic Euler) **[HLW02]** shown in Equation 8-1 and Equation 8-2.

Linear:

$$v_{t+1} = v_t + F\frac{dt}{m}$$
$$x_{t+1} = x_t + v_{t+1}dt \qquad \textbf{8-1}$$

where F is force, m is mass, v is velocity, x is position, t, $t+1$ is the current and next frame, and dt is the time-step.

Angular:

$$\omega_{t+1} = \omega_t + \tau\frac{dt}{I_t}$$
$$q_{t+1} = q_t + dt\frac{\omega_{t+1}}{2}q_t \qquad \textbf{8-2}$$

where ω is the angular velocity, q is the orientation as a unit-quaternion, I is the inertial, dt is the time-step, and t and $t+1$ indicate the current and next frame. We ran the simulation at a fixed-frame time-step of 0.01s.

8.1. Constraint Enforcement

The revolute joint constraints were enforced between bodies implicitly using a practical rigid body solver **[KM12]**. Since our simulations were carried out in 2D, the computational cost of generating and solving the constraint-based equations was minimal. (While we opted for a custom constraint solver, alternatively both open source and proprietary constraint based methods are available, for example Open Dynamics Engine **[ODE12]** and Newton Dynamics Engine **[NDE12]**)

9. Implementation Details

We implementation the simulation experiments in C# on a 2.6Ghz Intel Pentium CPU computer. We ran the simulation with a fixed-size integration update of 0.01s (i.e., 100 Hz). To ensure the physics simulation remained stable with the fixed frame time-step, we clamped forces and torques to acceptable limits. This had the affect of penalty forces not being strong enough to push objects apart and allowed a small amount of penetration. We saw this as an acceptable trade off and argued that the visual artefact mimics real a world soft-body deformation. The editor and the character-character fight simulation easily ran at interactive frame rates so it was easy to customize and tweak motions and get instant feedback.

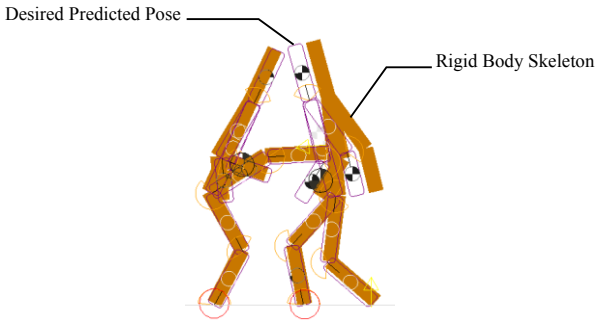

Figure 9-1: Desired and current pose using kinematic and articulated rigid body skeleton.

9.1. Collision Detection

Collision detection is a central component of any physics-based simulation. It allows rigid bodies to interact and engage one another and their environment. Without it, objects would behave like ghosts and pass through everything. Hence, to keep our system as simple and as computationally fast as possible we used simple geometric shapes (e.g., spheres, capsules, squares) to detect collisions (i.e., contact points and penetration depths). We used this collision detection information to apply separating penalty forces to push objects apart.

9.2. Balancing

We use a collection of stored key-frame poses (e.g., kick, punch as shown in **Figure 14-2**) to provide a controllable customisable collection of interactive animations that the user can switch between to make the avatar move. Furthermore, we include an inverted pendulum (IP) feedback model **[KKK*01]** to add intelligent balancing stepping information into the upright motions so they appeared more plausible and life-like (see **Figure 9-2**).

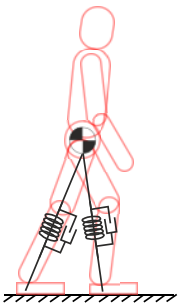

Figure 9-2: Illustration of the spring loaded inverted pendulum for stepping information.

The support region is the area of ground that is in contact with the character. We usually represent this support region by the character's foot (or feet). For slow or static motions, the overall centre of mass position of the character has to be above this support region to remain upright and balanced. Hence, we analyse the current balancing situation by examining the position of the overall mass of the rigid body skeleton and projecting it onto the support region for the foot (or feet). If the overall mass is within this support region, we assume the character is balancing; however, if the overall mass goes outside the support region, we assume that the character needs to take a corrective step to remain balanced. We extend this elementary model for dynamic situations by providing a more accurate balancing approximation. This improved balancing approximation uses the overall linear velocity of all the links to predict where the overall centre of mass position will be at the next time-frame period.

The inverted pendulum (IP) model allowed us to predict the character's foot placement position to counteract force disturbances that would cause the character losing balance and falling over. We accomplished this by modulating the fixed key-pose animation angles with an inverse kinematic (IK) solution to produce less repetitive and more natural stepping motions (see **Figure 14-1**).

10. Results

In this section, we describe some of the results from our physics-based character fight simulation. **Figure 14-1** and **Figure 14-2** show screenshots of different poses and their corresponding angle values. Screenshots of some character-character interactions are shown in **Figure 10-2**.

The implementation of our physics-based character system is not a small endeavour. While the basic system was constructed in a few days, it took several weeks to implement a usable system with various features (e.g., scripts, editor, IK solver). However, a fully-fledged physics-based animation system is worth the investment since it can create a diverse range of motions and character types with little work (i.e., modification of coefficients in the scripts such as strength, mass, and dimensions to create different walk types, responsiveness).

In practice, stiff over-damped joint PD controllers presented a more rigid and controllable character. Since the simulator ran at interactive speeds, it was easy enough for the animator to modify coefficients on the fly and get quick feedback on the resulting effects. Once the animator obtained their desired animation effect, they could save the coefficients to a script and reload them on demand in the final game.

We did not incorporate any stability analysis into our system. While in the majority of cases this was not a problem, but for excessively abrupt forces and torques, it could cause instability problems. To remedy this, and have the system recover gracefully, it was a choice between having the time-step decrease or clamp and dampen the forces and velocities. However, to maintain a real-time frame-rate we opted for the latter, so sporadic forces were cut off while damping ensured the system constantly lost energy and converged on a static solution (i.e., did not oscillate forever).

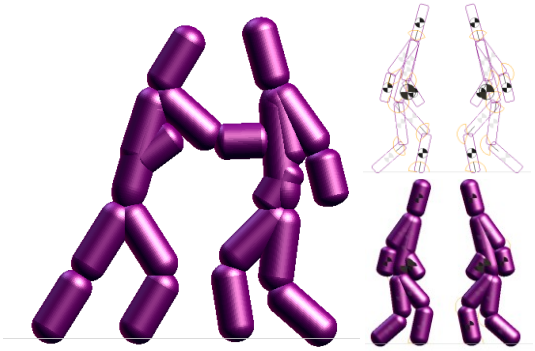

Figure 10-1: Rigid skeleton links as capsules showing stance pose and punch contact.

10.1. Performance

The frame-rate remained at a consistent 60fps, with the integration, interpolation, balancing logic, and the calculation of joint torques for two characters being calculated on average from ~0.01ms to ~0.05ms.

Figure 10-2: Character-character interaction (e.g., pushing, hitting, kicking).

11. Limitations

While we have implemented a robust and flexible physics-based character system, it does have a number of limitations. There can

be cases when unnatural and seemingly impossible poses occur due to the best guess selection of the current situation and the desired action (e.g., character has been knocked over and needs to get up). Furthermore, it can require arduous hand tuning of the numerous parameters to make the character's overall movements meet a desired style. We focused on upright biped characters that used a very simple data set for key poses. In addition, while the general system can accommodate diverse creature types it would require additional animation poses and customisability of behaviours, such as balancing and fight tactics.

We generated uncomplicated trajectory paths for stepping motions while the majority of the information came from static key poses. However, the characters can at times appear static and robot-like. One possible way to reduce this problem and to improve a character's believability is to inject coherent random movement into the physics-based model to make the characters appear more life-like and unique **[KEN12a]**.

12. Conclusion

In this paper, we have demonstrated an approach for creating a physics-based character system for interactive virtual environments such as games. For example, we implemented a biped fighting game to show the potential of our approach. While our game and character simulations were 2D and possessed only a few degrees of freedom with a limited repertoire of actions, it demonstrated the ingenious potential and possibility of using an intelligent dynamic based system instead of interpolating hard-coded key-framed animation libraries.

While the focus of this paper was a practical physics-based character system that could be used in a highly dynamic and interactive environment, such as a fighting game, there are still a number of attractive and exciting avenues for future work. For example, expanding the available action repertoire for the fight game (e.g., special moves), more intelligent AI, non-biped characters (e.g., horses), 3D, terrain interaction, picking up objects (e.g., struggling to pick up an axe or sword).

13. References

[AFOB05] O. Arikan, D. a. Forsyth, and J. F. O'Brien, "Pushing people around," Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '05, no. July, p. 59, 2005.

[CMCY69] Clauser C. E., McConville J. T., Young J. W. August (1969). WEIGHT, VOLUME, AND CENTER OF MASS OF SEGMENTS OF THE HUMAN BODY. Aerospace Medical Research Laboratory, Ohio.

[DG67] Dempster W. T., Gaughran. G. R. L. American Journal of Anatomy, Vol. 120, No. 1. (1967), pp. 33-54

[KLK04] T. Komura, H. Leung, and J. Kuffner, "Animating reactive motions for biped locomotion," Proceedings of the ACM symposium on Virtual reality software and technology - VRST '04, p. 32, 2004.

[ZH02] V. B. Zordan and J. K. Hodgins, "Motion capture-driven simulations that hit and react," in Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '02, p. 89, 2002

[ZMCF05] V. B. Zordan, A. Majkowska, B. Chiu, and M. Fast, "Dynamic response for motion capture animation," ACM Transactions on Graphics, vol. 24, no. 3, p. 697, Jul. 2005.

[SCCH09] T. Shiratori, B. Coley, R. Cham, and J. K. Hodgins, "Simulating balance recovery responses to trips based on biomechanical principles," Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation - SCA '09, p. 37, 2009.

[TPZZ06] B. Tang, Z. Pan, L. Zheng, and M. Zhang, "Interactive generation of falling motions," Computer Animation and Virtual Worlds, vol. 17, no. 3-4, pp. 271-279, Jul. 2006.

[MCP07] J. McCann and N. Pollard, "Responsive characters from motion fragments," ACM Transactions on Graphics, vol. 26, no. 3, p. 6, Jul. 2007.

[SKR11] S. Singh, M. Kapadia, and G. Reinman, "Footstep navigation for dynamic crowds," Animation and Virtual, 2011.

[WM08] C. Wu and J. Medina, "Simple steps for simply stepping," Advances in Visual Computing, 2008.

[SGL93] C. L. Shih, W. A. Gruver, and T. T. Lee, "Inverse Kinematics and Inverse Dynamics for Control of a Biped Walking Machine," Journal of Robotic Systems, vol. 10, no. 4, pp. 531-555, 1993.

[STEP07] B. Stephens, "Humanoid push recovery," 2007 7th IEEE-RAS International Conference on Humanoid Robots, pp. 589-595, Nov. 2007.

[PCDG06] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in Humanoid Robots, 2006 6th IEEE-RAS International Conference on, 2006, pp. 200–207.

[KDM11] B. Kenwright, R. Davison, G. Morgan, "Dynamic Balancing and Walking for Real-Time 3D Characters," Motion in Games, 2011.

[KEN12a] B. Kenwright, "Generating Responsive Life-Like Biped Characters," The Third Workshop on Procedural Content Generation in Games (PCG 2012), May 29th, p. 1-8, 2012

[KEN12b] B. Kenwright, "Responsive Biped Character Stepping: When Push Comes To Shove," Cyberworlds, September 2012.

[PG96] K. Perlin and A. Goldberg, "Improv: A system for scripting interactive actors in virtual worlds," in Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, 1996, pp. 205–216.

[PERL95] K. Perlin, "Real time responsive animation with personality," Visualization and Computer Graphics, IEEE Transactions on, vol. 1, no. 1, pp. 5–15, Mar. 1995.

[FVDPT01] P. Faloutsos, M. van de Panne, and D. Terzopoulos, "Composable controllers for physics-based character animation," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, no. 1, pp. 251–260.

[ML10] I. Mordatch and M. D. Lasa, "Robust physics-based locomotion using low-dimensional planning," ACM Transactions on Graphics, 2010.

[TLT11] Tan J., Liu K., Turk G. "Stable proportional derivative controllers," Computer Graphics and Applications

[TGTL11] Tan J., Gu Y., Turk G., Liu C. K. "Articulated Swimming Creatures," ACM Transactions on Graphics, Vol. 30, No. 4, Article 58, Publication date: July 2011.

[HLW02] Hairer E., Lubich C., Wanner G. "Geometric Numerical Integration, Structure Preserving Algorithms for Ordinary Differential Equations," Springer, 2002.

[KKK*01] Kajitas S., Kanehiro F., Kaneko K., Yokoi K., Hirukawa H. "The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation." In Proceedings of the IEEE/RSJ International, Conference on Intelligent Robots and Systems, (2001), pp. 239–246

[KM12] Ben Kenwright, Graham Morgan (2012), "Practical Introduction to Rigid Body Linear Complementary Problem (LCP) Constraint Solvers", In Algorithmic and Architectural Gaming Design. pp. 159-205, 2012

[ODE12] Open Dynamics Engine, http://www.ode.org/ (accessed September 05, 2012).

[NGD12] Newton Game Dynamics, http://www.newtondynamics.com/ (accessed September 05, 2012).

[HIK12] HumanIK, Autodesk Gameware, http://gameware.autodesk.com/humanIK, 2012

[EUPH12] Euphoria, www.naturalmotion.com, 2012

[HAVO12] Havok, www.havok.com, 2012

14. Appendix

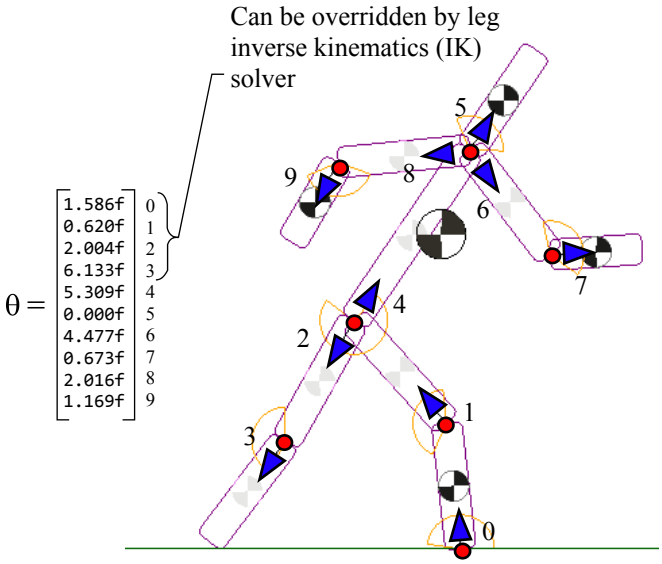

Figure 14-1: Simulation screenshot showing each element's mass and the local angle offsets from the parent. (Note the angles are not set in stone as they can be override and modulated with other key poses, such as an inverse kinematic solution to provide more natural results).

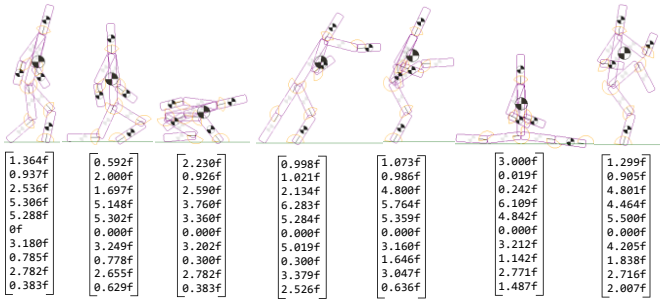

Figure 14-2: Some of the key poses for the fight character simulation (i.e., stand, crouch, duck, punch, kick, splits, block).

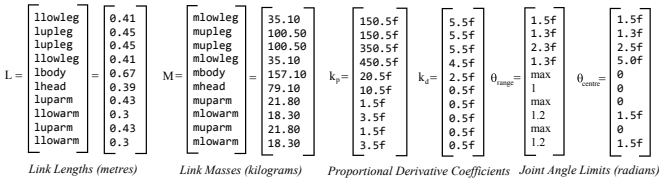

Figure 14-3: Default biped character variable values.