MULTI-USER NON-LINEAR ADAPTIVE MAGNIFICATION FOR SATELLITE IMAGERY AND GRAPH NETWORKS

By

Ian Charles Ooi

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: COMPUTER SCIENCE

Examining Committee:

_____

Barbara Cutler, Thesis Adviser

_____

Charles Stewart, Member

_____

Shawn Lawson, Member

Rensselaer Polytechnic Institute
Troy, New York

July 2014
(For Graduation August 2014)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Animations of human characters are used heavily in video games, movies, and other fields. Especially with the increasing usage of complex environment traversal in both film and video games, many similar animations of athletic motions must be created with small changes to tune the motion to the particular situation, environment, and character. Creation of such animations is largely done by hand by artists using keyframing. In a keyframe animation, certain "key" parts of the animated sequence are specified, with the remaining frames filled in, or "tweened" using an automated interpolation method or manual frame addition. For 2D animation, this occurs as a series of images which are played back in order to produce the animation. In 3D, keyframe animations are performed on a 3D model.

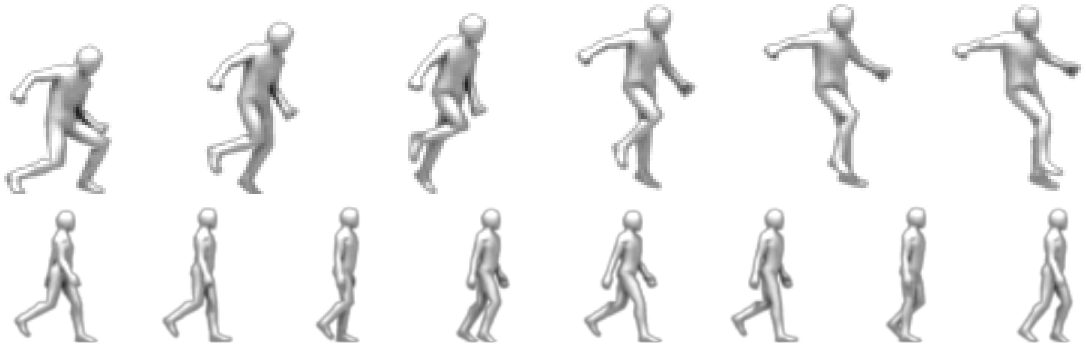

Figure 1.1: This example shows a 2D sprite sheet used to produce a jumping animation for a stick figure character. The frames in this case are laid out in a single image for demonstration purposes, progressing in order starting with frame 0, the frame farthest left in this sprite sheet.

3D models are described as a mesh, a collection of primitive polygons (i.e. quadrilaterals or triangles) which are stored as vertices. This mesh describes what is drawn, including any texture, color, and other material information. Along with the mesh, a skeleton, or rig, is stored. The rig describes a heirarchical structure of bones and accompanying joints. Each vertex is given a series of weights describing the impact each joint has on its transformation. This allows many vertices, and

therefore many polygons, to be transformed at once in organized groups, simplifying the problem of animating the model to a matter of transforming the skeleton in the desired manner. To animate this 3D model, an artist specifies keyframes of the animation by positioning the skeleton at different time steps. The stored keyframes, instead of an image, are the transformations of each joint at this frame or step of the animation, which a rendering or game engine can interpolate between to produce the final result.

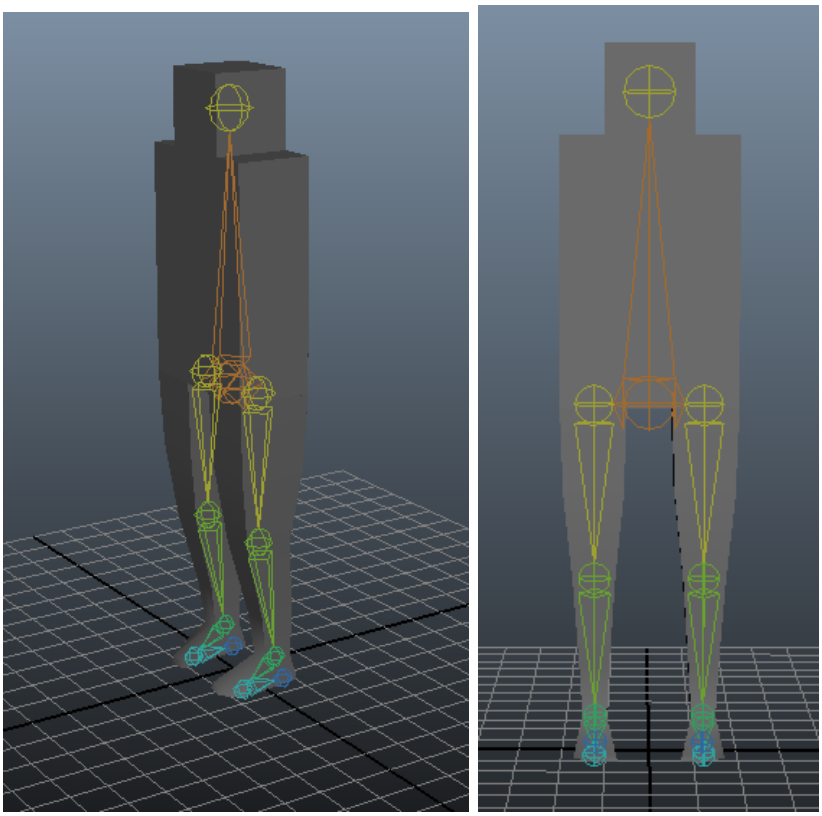

Figure 1.2: Above is an example of a character model in AutoDesk Maya. The character skin or mesh is shown in gray, with a rig shown in multicolor. While it is visualized as a series of spherical joints with connecting solids, the rig itself does not have a visual component in practice. The rig acts as a skeleton, deforming the mesh of the character model to make animation easier. Additional tools such as deformer groups and inverse kinematics handles can be used to further simplify the creation of animations for artists. While these tools simplify movement of the model, the artist still must position each joint for each frame of the animation, which is then stored for later playback.

Specifying these animation frames is work intensive, taking up significant time and resources to produce for a single character. Additionally, similar animations may need to be produced for slightly different scenarios, with only minor modifications required. These minor modifications can be to fit a different setting, such as a character jumping on Earth or on the moon, or can be for different characters, such as a large person moving in contrast with a small child. Though the movement itself may be similar, manual changes must be made, requiring artist time which could be spent generating new assets.

Recent work in animation generation seeks to automate this process, replacing the manual process with a procedural one. Physics based simulations can be used to produce controllers for the skeleton, determining joint positions and rotations for keyframes automatically. Not only does this reduce the effort involved in the creation process, but this also provides a basis for dynamic interaction between a character's animation and the environment, which is not possible with manual keyframe animations.

We present a controller that takes a skeleton as input, with additional parameters describing the character, which produces a sequence of poses for a keyframe animation. The additional character parameters describe the character's mass as well as the constraints placed on each joint to prevent unnatural rotations. Weight is specified per-joint to allow for calculation of the character's center of mass. Our controller works with the initial flex and takeoff stage of the jumping motion. The motion is calculated by modeling muscles as simple springs attached to the skeleton at 2 points. Spring constants for the muscles are determined by the user, which are applied in a linear spring calculation to determine the approximate change from rest length required to achieve a particular force. This change from rest length approximates the flexion required, which can be used to calculate a plausible amount of bend for the wind-up motion of a jump.

To control the motion and maintain plausibility, calculations are performed to determine the character's center of mass and supporting polygon. As the character should maintain balance while flexing its joints, the position of the character's joints are adjusted to keep the center of mass positioned over the supporting poly-

gon. Flexion proceeds with the character bending progressively, maintaining balance while moving to achieve the desired spring force in its leg muscles.

The next stage of the motion, the take-off where the character releases from the ground, uses the potential energy of the spring-muscles and accelerates the character's center of mass upward. Application of force works from the joints and muscles closest to the center of mass outwards through the skeleton. While not handled by our controller, the character would then proceed through the in-air portion of the jump, where the acceleration changes due to gravity as well as other forces before they finally land. We assume a simple trajectory for the in-air phase, though more complex motions with turns, flips, or interaction with the environment could be created. Other work has handled landing with a similar approach.

Our controller is made to be a module, able to be used with other controllers as part of a larger system. This allows each controller to do a smaller job well. Several such controllers can be connected to produce more complex animations or animation sequences, utilizing bounded starting and finishing conditions for the character. Additionally, certain cases during the duration cause the controller to stop early, for example a mid-air collision mid jump which would require a separate controller to handle this case.

1.1 Contributions

This thesis describes a controller which simulates a jumping motion on a character. The generated animation is created to be plausible in appearance, though it may not be a physically accurate representation. Specifically, we contribute a model for the windup and take-off phases of a jump and created a controller using this model in Unity3D.

Unity 3D is a game engine which we used to develop this system. It provides infrastructure for rendering, scene management, skeletal animation, asset import and management, lighting, and scripting. The system developed leverages the provided features through the Unity3D scripting interface. We developed scripts for calculation of muscle forces, as well as for applying the force to an imported model. Models were created in AutoDesk Maya and imported using Unity3D's asset im-

port as Unity3D game objects with attached Transform components which allow arbitrary transformations.

CHAPTER 2
PREVIOUS WORK

2.1 Background

Producing athletic animations for human characters is difficult. One method, motion capture is used for production of realistic animations for human athletics and other motions, however it requires the collection of information for each motion and does not adapt to the virtual environment. Muscle-based approaches produce realistic motions which adapt to the environment, using a complex model of the musculo-skeletal structure. Geijtenbeek et al. use a rough, user created muscle routing on a skeleton to produce various gaits that are learned based on the velocity and environment. The muscle routing is optimized to remain within a region while providing optimal forces on the skeleton based on freedom of motion of the skeletal joints and the calculated optimal length of the muscle. This model is then used to compute sequences of muscle activations, modeling neural signals, which produce the final animations. This method is effective, producing good results in various levels of gravity on at least 10 different bipedal skeletons [1].

Figure 2.1: Example of a muscle routing on a skeleton from Geijtenbeek et al. [1].

Inverse kinematics approaches attempt to generate the motion based on a desired final position, determining the skeletal position by solving the system given constraints. Koga et. al use path planning, inverse kinematics, and forward simulation to generate animations of arm motions for robots and humans working cooperatively. They produce arm manipulations that avoid collisions and result in final positions and orientations for specified parts of the arm to produce motions such as a human putting on glasses and a robot arm and human cooperating to flip a chessboard [3].

Physical simulations utilize a rigid-body character with a user-defined skeleton to find optimal poses based on desired conditions. Ha et al. utilize such a

Figure 2.2: Breakdown of a hands-first falling approach from Ha et al. [2] and of a feet-first landing approach. Ha et al. use a rolling strategy to minimize stress on the body and produce a realistic fall.

scheme to generate landing motions for human characters based off linear velocity, global angular velocity, and angle of attack. The system chooses either a feet first or hands first landing strategy and moves into a roll to reduce stress on the body using principles from biomechanics and robotics. A sampling method is applied to determine successful conditions, producing bounding planes for the data. The movement is broken into stages of airborne and landing, in which the character re-positions for the designated landing strategy, and executes the landing strategy respectively. Each of these is separated into impact, roll, and get-up stages. Movement and joint positions are produced using PID servos [2]. Other work on producing such controllers was produced by Faloutsos et al. who described a method of composing such controllers by giving pre-conditions, post-conditions, and intermediate state requirements. The composed controllers are then chosen at each step based on the current pose and which controller is deemed most suitable [4]. Hodgins et al. created several controllers for running, vaulting, and bicycling, creating realistic motions and secondary motion using rigid bodies and spring-mass simulations [5]. Geijtenbeek and Pronost provide a detailed review of physics based simulations [6].

CHAPTER 3
ANIMATION

Jumping is the acceleration of a character's center of mass upward. This motion can be divided into several stages. First is the lead-up or wind-up stage in which the character flexes or gathers momentum to perform the jump. This takes the form of a slight crouch (TODO reference cat jumping paper or the background section. should this be in background?) which prepares the character to exert the necessary force against the ground.

Next comes the take-off stage. The character pushes against the floor with their feet, accelerating their center of mass to break contact with the floor.

3.1 Force calculation

3.1.1 Inverse Kinematic Solving

As the skeleton is a hierarchy assumed to be rooted at the hip, a problem arises with applying rotations to joints. To keep a character's feet rooted to the floor as is expected,

3.2 Center of Mass and Balance

3.3 Thrust

References

[1] T. Geijtenbeek, M. van de Panne, and A. F. van der Stappen, "Flexible muscle-based locomotion for bipedal creatures," *ACM Transactions on Graphics*, vol. 32, no. 6, 2013.

[2] S. Ha, Y. Ye, and C. K. Liu, "Falling and landing motion control for character animation," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 155:1–155:9, Nov. 2012. [Online]. Available: http://doi.acm.org/10.1145/2366145.2366174

[3] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe, "Planning motions with intentions," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '94. New York, NY, USA: ACM, 1994, pp. 395–408. [Online]. Available: http://doi.acm.org/10.1145/192161.192266

[4] 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*, ser. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 251–260. [Online]. Available: http://doi.acm.org/10.1145/383259.383287

[5] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien, "Animating human athletics," in *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '95. New York, NY, USA: ACM, 1995, pp. 71–78. [Online]. Available: http://doi.acm.org/10.1145/218380.218414

[6] T. Geijtenbeek and N. Pronost, "Interactive character animation using simulated physics: A state-of-the-art review," *Computer Graphics Forum*, vol. 31, no. 8, pp. 2492–2515, 2012. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2012.03189.x