RIGGING SKELETAL PERISSODACTYL AND ARTIODACTYL UNGULATE LIMBS USING ANALYTIC INVERSE KINEMATIC-BASED SOLUTIONS FOR A FEATURE FILM PRODUCTION ENVIRONMENT

A Thesis

by

WILLIAM LAWRENCE TELFORD JR

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2006

Major Subject: Visualization Sciences

RIGGING SKELETAL PERISSODACTYL AND ARTIODACTYL UNGULATE LIMBS USING ANALYTIC INVERSE KINEMATIC-BASED SOLUTIONS FOR A FEATURE FILM PRODUCTION ENVIRONMENT

A Thesis

by

WILLIAM LAWRENCE TELFORD JR

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee, Karen Hillier
Committee Members, Ergun Akleman
Donna Hajash
Head of Department, Mark Clayton

December 2006

Major Subject: Visualization Sciences

ABSTRACT

Rigging Skeletal Perissodactyl and Artiodactyl Ungulate Limbs Using Analytic Inverse Kinematic-based Solutions for a Feature Film Production Environment.

(December 2006)

William Lawrence Telford Jr, B.E.D., Texas A&M University

Chair of Advisory Committee: Prof. Karen Hillier

The goal of this thesis is to develop and construct a repeatable, scalable, and portable rigging solution for the skeletal limbs of ungulates, maximizing functionality while streamlining intuitive interface controls for a feature film production pipeline. The research presents a methodology for breaking down character reference materials commonly available to feature film productions like artwork, anatomical drawings, photographs, and client provided performance criteria. It then presents a modular methodology and approach for successfully evaluating and applying the character reference to the construction of skeletal limbs using ungulates as the primary example. Each limb is broken down into modules that more easily translate into the digital world. The methodology then further defines how to combine and apply digital rigging tools such as constraints and inverse and forward kinematic techniques in a layered and modular way in order to achieve a robust character rig. The resulting ungulate limb rig provides an efficient, intuitive, and robust solution capable of replicating the given performance criteria as well as an example of a scalable approach applicable to non-

ungulates. In application of the repeatable modular approach presented, huge efficiency gains have been realized in feature film production pipelines. Animation studios are under increasing pressure to create larger quantities of work, at higher quality, with shorter timetables, and smaller relative budgets. This methodology successfully meets those criteria.

ACKNOWLEDGEMENTS

I would like to thank my thesis committee for their continued patience, guidance, and support. I would also like to thank my family for their encouragement in all of my life's pursuits. Finally I would like to thank my wife for her love, sacrifice, and understanding as this research extended from school and on into my career.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

Page

1. INTRODUCTION

The study of character rigging is a relatively new discipline in the field of computer graphics. According to Aaron Sims and Michael Isner, it is the dream of assembling all the tools and mechanical systems of 3D and creating a new creature [1]. It has had a slow and steady evolution, growing out of a need for more complex motion while providing intuitive controls to execute increasingly more complex animation. Historically animators are responsible for their own character rigs. Over time it has become clear that while animation tends to be an artistic task requiring aesthetic and stylistic decision making, rigging is a more mechanical task requiring logic and good problem solving skills. It is difficult to find someone who possesses the skills to both animate and rig well. As such, visual effects companies and animation studios are identifying specialists to concentrate solely on character rigging.

Early character rigs depended on simple hierarchical structures of nodes. A hierarchy implies a parent-child relationship between all nodes [2]. They were animated with forward kinematics (FK). FK is a technique in which each node in the hierarchy inherits its transforms from the node that precedes it in a parent/child relationship. It requires the animator to explicitly define the orientations of all joints [2]. By employing this method, an animator can animate a character's leg by traversing down the hierarchy, first rotating its hip, which in turn moves the knee (its child), ankle, and any subsequent

---

This thesis follows the style and format of *IEEE Transactions on Visualization and Computer Graphics*.

joints. The animator then rotates the knee which moves the ankle and any subsequent joints. This technique is very successful for animation that requires free hanging motion like a swinging arm, but is extremely tedious for other tasks such as planting a footfall. Anytime a parent or subsequent ancestor of a node is moved, it requires that its children be counter-animated back to the ground. This creates a painstaking redundancy in the animator's workflow. A better solution is a technique called inverse kinematics (IK). Using IK a user can drive many joints through the placement of just one node [2].

There are many different kinds of IK solvers. One of the most basic is a two "bone" or three "joint" analytic IK solver. This system employs simple trigonometry and requires that you have three nodes (A, B, and C) at unique points in space and another node to act as a goal or target (Figure 1).

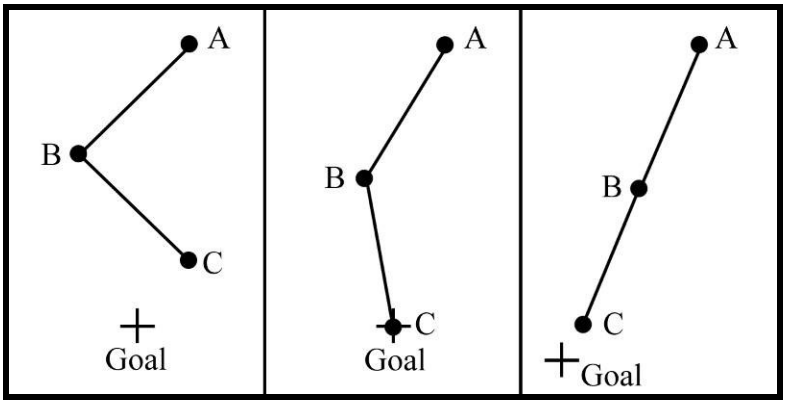

Figure 1: Simple IK structure. Parented nodes A, B, and C. Nodes A, B, and C achieving goal. Nodes A, B, and C attempting to reach goal.

Node C is parented to B, and B to A. This structure resembles an arm from shoulder to wrist. The goal exists outside the hierarchy. To simplify the system the transforms of nodes A, B, and C are limited to rotations only. The IK solver populates nodes A, B, and C with the appropriate rotations to allow C to reach the goal. If no solution is achievable, the solver comes as close as possible. Using this technique, an animator is able to place the location of a character's hip and the location of its foot, while the location of its knee is calculated for them. This greatly simplifies problems like placing footfalls.

The FK and IK methods described above are just two of the many building blocks available in most commercial and proprietary animation software packages used by visual effects companies and animation studios. At its root all characters are still just hierarchical structures of nodes. It is the rig's job to simplify the animator's interaction with those nodes allowing the animator to concentrate on performance and not technology. The complexity of the problem does not end there. Beyond an animator's interaction with a rig, it is usually necessary for the rig to provide the foundation for a deformation structure for geometry that represents the character's skin. This foundation is analogous to a character’s skeleton.

To build a successful rig it is not only important to have a strong understanding of the available toolsets, but to also have a strong understanding of a character's anatomy. This applies to characters that don't exist in real life as well. A character rigger must still define and adhere to a functionally appropriate anatomy. Simply a slight shift

in the position of any joint in a character can make or break the believability of the motion and deformations. It is possible, and usually necessary, to take liberties with the structure of a character, but how and why the character deviates from its true anatomy must be planned and understood. Usually these deviations require a compromise in either performance or deformation for the sake of simplicity. The spine of a character is a good example of this. Many times the spine of a 3D character is built upon a spline, allowing the use of four or five joints as opposed to the much higher anatomically correct number. This eliminates the need for controlling so many joints in the spine, but it does so at the cost of more accurate deformations. Quite often this is an acceptable compromise between accuracy and ease of use, as it greatly improves the animator's efficiency. How and where to make compromises is usually decided by animation supervisors and character-rigging supervisors after carefully weighing the cost benefit of such a decision.

To successfully rig for a feature production environment the impact of the rig on the entire production pipeline must be considered as well as the impact on animation. The choices made at each stage of the pipeline (modeling, rigging, animating, lighting, and effects) have an impact both up and down stream. The topology and form of a model is heavily dependent on rigging requirements. The default pose of a character is often dictated by what is most convenient for the rig and deformations. Many lighting problems can be traced back to a poorly deforming character, as certain deformation problems do not become evident until they are properly shadowed and shaded. In many

cases effects can interact heavily with a character, putting its own set of constraints on the rig.

2. PROBLEM STATEMENT

Feature film production pipelines require a robust character rig to achieve the level of realism required by a modern audience. The rigging solution must be implemented in both a computationally and time efficient manner. Prior work in this area concentrates on specific implementations within specific commercial software packages. The goal of this thesis is to develop and construct a repeatable, scalable, and portable rigging solution using the example of skeletal limbs of ungulates, maximizing functionality while streamlining intuitive interface controls for a feature film production pipeline. The methodology and philosophy presented is an approach that is applicable across multiple platforms, software packages, and countless rigging problems.

3. RIGGING PHILOSOPHY

In order to successfully rig a character, it is necessary to have a strong philosophy about how to approach the problem. Rigging is, at its most basic, a series of problems that need to be solved. It is appropriate to first approach each problem individually. By doing this a rig can be separated into a set of modules that serve as building blocks. These modules can then be assembled into larger and larger systems. By attacking the problem in a modular way, it is possible to reuse modules both within the same character as well as others and increase efficiency. For example, it is not uncommon to use identical modules on front and rear limbs. Multiple modules assembled into a system can then be viewed as an even larger module. This process is then repeated to develop a way of layering complexity into a character.

Layering helps keep a rig consistent, improves the efficiencies of both rigger and animator, and facilitates debugging. Reusing modules in the same rig assures that similar problems are solved and behave the same way in all areas. This consistency is invaluable to animators counting on the behavior of the rig. Without it an animator would have to spend more time learning how to use the rig. The consistency also helps the rigger as they build more complex systems that depend on these modules. If an error or instability is found in a module it can easily be corrected and propagated out to similar modules. Without this, the rigger could easily overlook a similar instability elsewhere in the rig. In addition, the consistency from module to module and character to character allows other

character riggers to participate in the construction and support of a character with little education as to the history of the rig.

The success of a character rig also depends heavily on its stability. It is important that all systems in the rig be as stable as possible. In using a modular approach a rigger can more easily insure the stability of a rig by verifying that the modules used are each stable as a unit. When a rig is unstable it can drastically affect the performance and efficiency of the character. Quite often instabilities can manifest themselves in ways that don't always seem repeatable. Instabilities can be as drastic as a joint flipping 180 degrees or as subtle as a joint that can't achieve its rest position by a fraction of a degree.

A rig's usability is also greatly affected by its efficiency. Many tools available to riggers come with a steep computational cost. Quite often this cost is calculated by determining how fast animation on the rig can be played back. The ideal frame rate for film is 24 frames per second (fps), or real time. It is not uncommon for rigs with expensive deformations to play back at 1 or 2 fps. The slower the rig the more difficult the rig is to animate. This expense must be balanced with the complexity of the rig. As animators work one frame at a time this usually doesn't inhibit their ability to pose the rig, but it does slow down their ability to evaluate motion. To accommodate this problem most animation packages have caching systems that allow for real time playback of complex rigs.

One common component found in complex character rigs is automation. Automation can be as simple as having one joint's rotational value follow another joint's rotational value or as complex as providing all rotational movement of a ball given only its

translation through space. Automation is an extremely powerful tool, but it can also be disruptive. It is very difficult to direct the performance of an automated object, as its motion is driven indirectly through another system. This does not necessitate the exclusion of automation from a character rig, but it does require that the animators have the ability to turn off the automation.

4. CHARACTER REFERENCE

Constructing a solid character rig requires thorough research and planning that entails gathering character design reference. Design reference can come in many forms including artwork, anatomical reference, video, and client descriptions. Any conflicting information must be reconciled to develop a complete character profile. For example, vertebrates have a small amount of flexibility in their spine. This allows for small amounts of squash and stretch. A computer generated (CG) character may require a more cartoon-like performance, necessitating larger amounts of squash and stretch.

A common issue when working with anthropomorphic characters is reconciling lip-synch with an animal whose facial musculature does not support such motions. Quite often additional structures and controls must be created that would not exist in the character reference to allow the desired performance.

4.1 Form

Many times reference for a character is given in the form of two dimensional (2D) artwork. It is also very common for this form of reference to have anatomical inconsistencies due to different drawing techniques such as foreshortening of the image. When constructing a character it is most useful to have 2D reference images with as little distortion as possible. Other cheats are completely aesthetically driven. One of the most famous examples of a cheat when translating a character for 2D to three dimensions (3D) is a problem referred to as the Mickey Mouse ears problem. Mickey Mouse's ears are represented as perfect circles at all times. When he is facing camera they are placed

evenly to the left and right side of his head. When he is shown in profile his ears actually appear one in front of the other. When viewing a 2D cartoon this placement is unobtrusive to the viewer, however translating this character into 3D presents a multitude of problems. First, for his ears to appear perfectly circular from all angles, they would have to be constructed as perfect spheres. This solution would be incorrect, as it is implied that his ears are flat discs. The other issue would become apparent as a camera moved around his head. As the camera moved from head on to his profile the ears would need to physically change their contact point with Mickey's head. This example does not show a flaw in his design, but rather an inconsistency in the behavior of artistic reference and true anatomical reference that needs to be reconciled before rigging can begin.

Not all characters present problems as complex as the Mickey Mouse ears problem. Many times a character is meant to replicate its living counterpart as much as possible. When it is possible, it is necessary to study true anatomical reference. This can consist of anatomical drawings (which can suffer from drawing cheats as well), nature photographs and films, and an age-old technique used by artist such as Leonardo Da Vinci, animal dissections. It is even possible and necessary to gather anatomical reference for creatures that might not truly exist. In mythology there are many creatures that mankind has dreamed up which do not exist; however that does not mean that anatomical reference does not. For example, Pegasus was part horse part bird. He has six limbs: two wings and four legs. No existing mammal has six limbs. If it became necessary to construct Pegasus in 3D one could still use anatomical references to build him. The same modular approach described earlier can be applied to natural systems as

well. The majority of Pegasus's structure and anatomy could be derived from reference of horses while the wing reference could be derived from eagles or other large birds. Chuck Jones states in his book that what he looks for is not how another creature is different, but how he is the same [3]. The only area not addressed by these references would be how to join the wings and forelimbs into the torso in close proximity to each other. In those cases the last resort is to make an educated guess to interpret the anatomy.

4.2 Function

Once the anatomical structure and skeletal requirements of the character are understood, it is necessary to evaluate the performance criteria of the character. What is its range of motion? Does a quadruped need to behave as a biped? Does the quadruped need to be able to pronate and supinate its forelimbs in performing as a biped? Performance criteria can dictate that a joint, that in the natural world would usually be limited, have those limits relaxed or removed all together in the CG version to provide a broader range of motion. If a character's performance requires it to break its leg so that his knee can bend backwards, then the rig needs to be capable of achieving this in an animator controllable way.

5. TOOLS

Once a character's design, performance criteria, and anatomy are known, the appropriate tools can be chosen. Most non-deformation related tools can be categorized as a transform. Transform refers to the way in which an object's position or orientation can be affected by the user (animator).

5.1 Kinematics

Outside of a single object placed in space, the next most common way to place objects when rigging is through a hierarchical approach. Objects are arranged in a parented hierarchy in which each node can have only one parent, but unlimited numbers of children. An additional limitation is that objects cannot be parented in a cyclic fashion. For example you may not parent object A to B and B to A. In a hierarchical structure each object inherits the motion of its parent object.

5.1.1 Forward Kinematics

In order to animate in such a system it is necessary to start with the root node, or the uppermost parent in the structure, and then traverse down the hierarchy to its children. This is repeated until all nodes have the necessary animation. Unfortunately any performance changes that affect a node can necessitate the re-animation, or counter-animation, of its children in order to restore their original position in space. This method is referred to as forward kinematics or FK.

5.1.2 Inverse Kinematics

To avoid re-animation or counter animation another technique is used, inverse kinematics or IK. IK allows a user to place and orient a controller and the computer calculates the joint angles for the hierarchy in order to reach the goal [4]. IK algorithms can be divided into two major types, analytic solvers and iterative solvers (Figure 2).

An analytic IK solver uses the cosine rule to calculate joint angles. In a 3-node hierarchy the entire system can be viewed as a triangle with 2 sides that are unchanging in length. When placing the goal for the system, its distance from the root node in the hierarchy can be calculated, giving the third side of the triangle. With the length of all three sides of the triangle known, all angles can be calculated. This does not provide a unique solution, as the entire system can still be rotated about the axis formed by the root and the goal providing an infinite number of solutions. One more target must be created to stabilize the solution. This target, along with the root of the hierarchy and the IK goal, define the plane that the solution exists within to provide a unique solution. This is the more mathematically efficient methodology of the two.

Iterative IK solvers are less efficient as they can only approximate a solution. The user specifies a set number of iterations. The algorithm approaches a solution, error checking itself with each iteration, until the solution is within a specified tolerance (Figure 3). This method becomes less efficient the higher the number of iterations and the smaller the error tolerance is set.

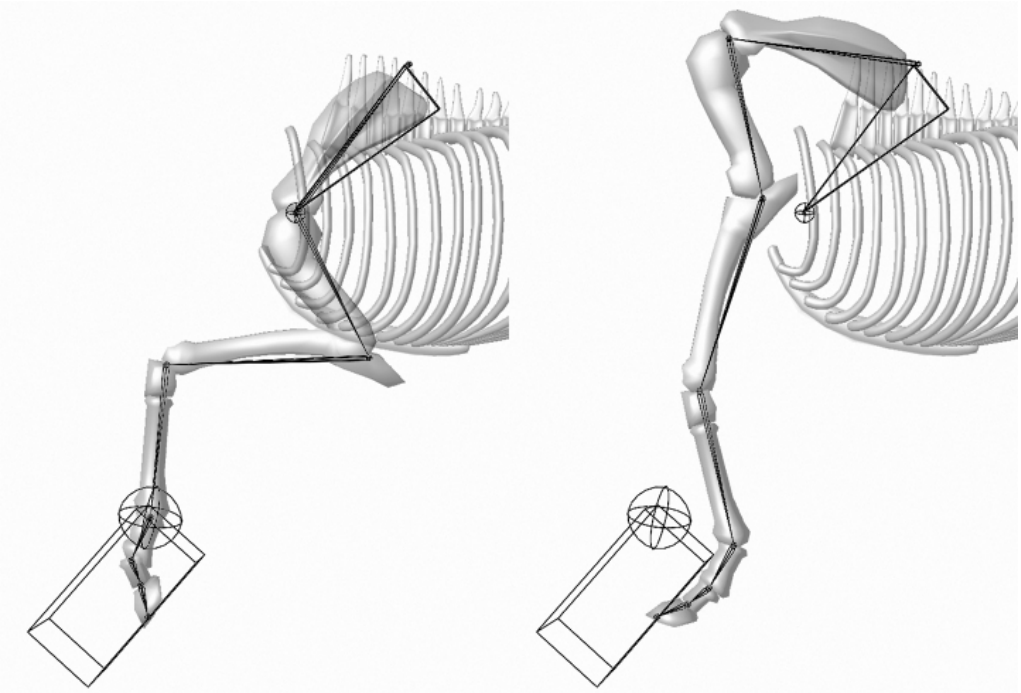

Figure 2: An example of a proposed rigging solution versus a single iterative IK solution.

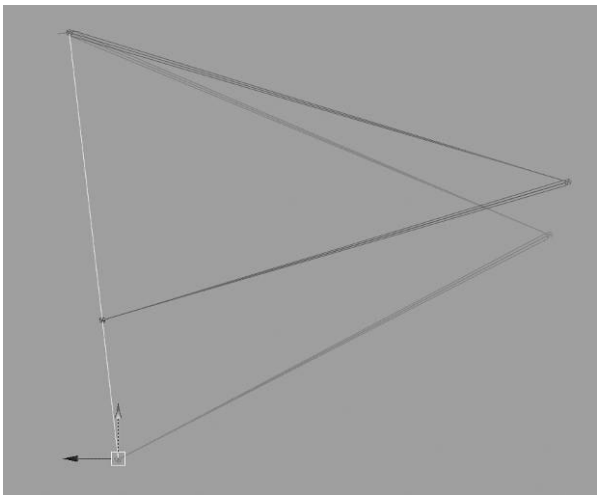

Figure 3: Comparison of an analytic IK solver versus an iterative solver with a single iteration and a high tolerance.

This method is often utilized for hierarchical systems with more than three nodes, as it requires more effort for the rigger to create an analytic solution. Some commercial packages utilize only an iterative solver, despite the computational expense, due to its versatility.

5.2 Transforms

Most animation packages provide for an FK and IK hierarchical rigging approach. It is also helpful to understand what other types of transforms are available. Common transforms include but are not limited to translational constraints, rotational constraints, aim constraints, geometry constraints, analytic surface or volume constraints, and projection constraints. Different animation packages may label transforms differently, but most are either readily available or easily implemented using existing tools.

A translation constraint allows one object or node to replace its own position in space with the position of another object. Many times versions of this tool have options in place that allow you to more freely define the space in which the object is constrained. The most common translation constraint is the ability to replace one object's world space position with the world space position of another (Figure 4).

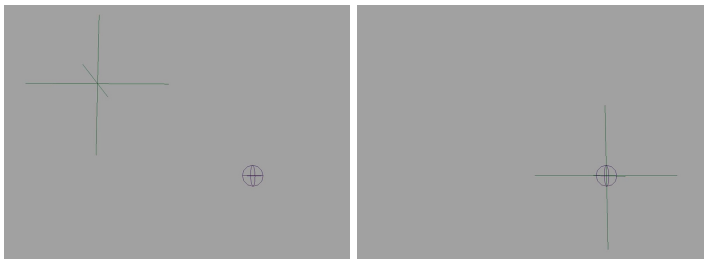

Figure 4: A translation constraint applied in world space.

Other permutations of this constraint are replacing one's local position (its position relative to its parent in the hierarchy) with the world position of another, replacing one's world position with the local position of another, or even replacing one's local position with the local position of another. These relationships can get very complex, sometimes allowing objects to be represented in spaces other than their own parents.

A rotational constraint is similar to a translation constraint. Instead of affecting the position of an object, it affects an object's orientation instead. Many of the same

options are available to this constraint. Rotations can be more complex than translations when interpolating their effect on and off. It is necessary to determine what interpolation method be used, as the results can be drastically different. Two choices of rotational interpolation are Euler angles and quaternion. Interpolating quaternions usually provides a more predictable behavior than interpolating Euler angles as the latter can appear to change direction during interpolation, but it comes at a higher computational cost.

In its most basic form, an aim constraint allows one object to point an axis at another object. Unfortunately, using it in this form is a very common source of mathematical instability in a character rig. Pointing a single axis of an object at another does not provide a singular solution to the orientation of the object. With one axis pointed at another object it's still possible for the object to spin about that axis providing an infinite number of solutions. To resolve this, most implementations of an aim constraint provide the ability to add a second and even third target to the constraint. Adding a second constraint will stabilize the object through most situations; however as the targets for both axes converge, the system becomes unstable (Figure 5).

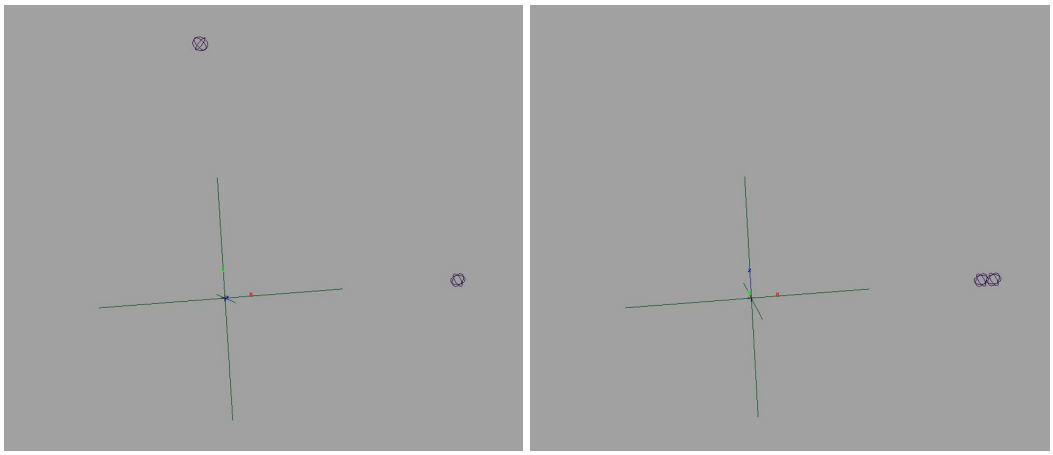

Figure 5: A stable and unstable aim constraint. In the second image both x and y axes are trying to point in the same direction creating instability.

When the two targets share the same point in space, or are coincident, the spin is no longer stable. This is why sometimes a third target for the third axis of rotation is necessary. Unfortunately it is always possible to pose this system in a way that is unstable. With a good plan and rigging philosophy this can mostly be avoided. As with all rotational solutions, interpolation of this constraint's affect is subject to the interpolation method chosen.

In complex rigs it is sometimes necessary to constrain an object's position to a piece of geometry or its orientation to a surface's normal and tangent. There are multiple ways of achieving this. One is a geometry constraint that maps an object to the closest point on geometry's surface (Figure 6). This process can be heavy for polygonal or sub-d geometry surfaces, as computationally expensive operations to find an intersection on a surface are necessary. The more dense the surface, the more expensive the constraint

becomes. To avoid the heavy calculation of geometry constraints, it is possible to constrain an object to a surface or volume that has an analytic representation. Nurbs surfaces, ellipsoids, spheres, and cubes are examples of surfaces and volumes that can be represented analytically, it is a less expensive operation to identify a point on the surface or contained within the volume. It is common to replicate a piece of geometry, like a polygonal surface, with an analytic surface for purposes of calculating a constraint cheaply.

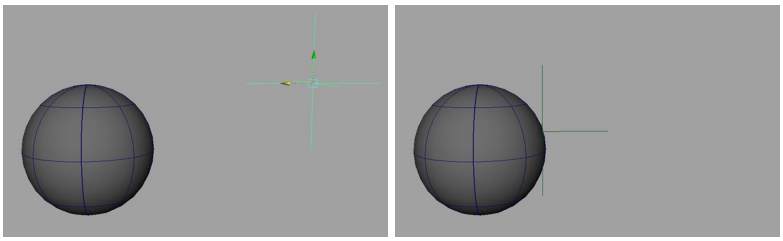

Figure 6: An object being constrained to a nurbs sphere.

Another constraint that is easily implemented if not readily available involves projection calculations. Using simple mathematical projection algorithms it is possible to project an object using a planar, spherical, or cylindrical projection onto a point, segment, line, plane, or surface. Projection, surface, geometry, and volume constraints can all provide very complex ranges of motion or limits to the motion.

5.3 Expressions and Scripts

Outside of transforms, a robust character rig can also require the use of formulas and scripts. Maya, a popular 3D animation package, refers to the ability to use formulas to control a rig as expressions. At its most basic it allows one value in a rig to drive another. For example, a formula might be created to set the Y rotation of object A to be the same value as the Z rotation of object B. In more complex forms, a formula may be written that lets multiple values drive other values through a more complex algebraic formula. These formulas can be powerful as they can be evaluated as the animator interacts directly with the rig. Due to their interactive nature, formulas may not be able to handle all complex calculations. Small scripts can be created to handle even more robust tasks. Scripts can be used to populate values temporarily, leaving the ability for the animator to modify the output after calculation unlike formulas or expressions. For example, a script might be used to copy animation from one character to another without keeping a live connection between the two. After the animation is copied, the animator still has the ability to animate the two characters independently without affecting the other.

6. STRUCTURE

The structure is where the rig interprets the anatomy. It is possible to represent every anatomically accurate bone in 3D but it comes at a cost. The more bones that are introduced into the structure the more objects have to be controlled. Conversely, the fewer bones represented into the structure the less accurate the skeleton becomes. A balance between the two must be struck, allowing few enough bones to be manageable and enough bones to convey the appropriate performance and anatomical structure.

6.1 Natural Anatomy

In stark contrast to the age of rigging and computer graphics is the study of animal anatomy. In 350 BC Aristotle had to consider the parts of animals and why each part is as it is and why they possess them [5]. Just as he did, to successfully rig a character the same questions must be asked. This research focuses on the anatomy of ungulate limbs. An ungulate is defined as any hoofed mammal. Ungulates are divided into two major categories perissodactyla and artiodactyla, which refers to an ungulate with an odd number of toes or an even number of toes respectively. Paleontology has not determined to what degree the two are related to each other, but their anatomy is similar enough to be represented by a single digital paradigm [6]. Horses, cattle, deer, and pigs are examples of ungulates. An ungulates method of locomotion is referred to as unguligrade or hoof-walking as opposed to digitigrade or toe walking and plantigrade or sole walking.

For the sake of this research the ungulates limb anatomy is divided into several structures. On the forelimb it is divided into the shoulder, the elbow, the knee, the fetlock, and the hoof. On the hindlimb it is divided into the hip, the stifle, the hock, the fetlock, and the hoof (Figure 7).

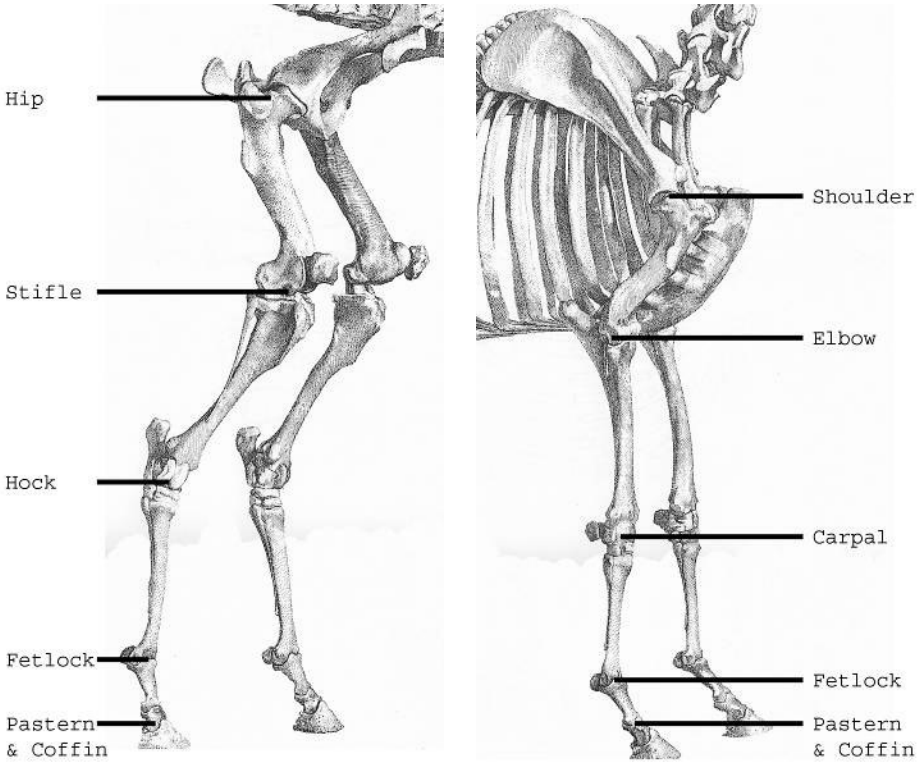

Figure 7: Horse skeleton. (Adapted from Ref. [7]).

6.1.1 Forelimb

In the forelimb the shoulder structure controls both the rotation of the scapula, the translation of the scapula, and the rotation of the shoulder. It is the most complex structure in ungulate limbs to replicate and it produce the most complex motion. The scapula structure produces such a complex motion due to the degrees of freedom found in the rotation and translation of the scapula. Unlike humans there is no bone that connects an ungulate's shoulder structure to its trunk. Humans have a clavicle which connects the shoulder to the sternum. In place of such a bone the ungulate has a complex system of muscles and tendons that hold the shoulder flush to the animal's rib cage. Straps of muscle connect from the scapula down to the animal's chest called the thoracic sling or pectoral sling, acting like a basket that the chest hangs in (Figure 8). This downward pressure on the muscle pulls the tips of the scapula outward. To counter this motion another set of muscles and tendons connect from the scapula to the spine.

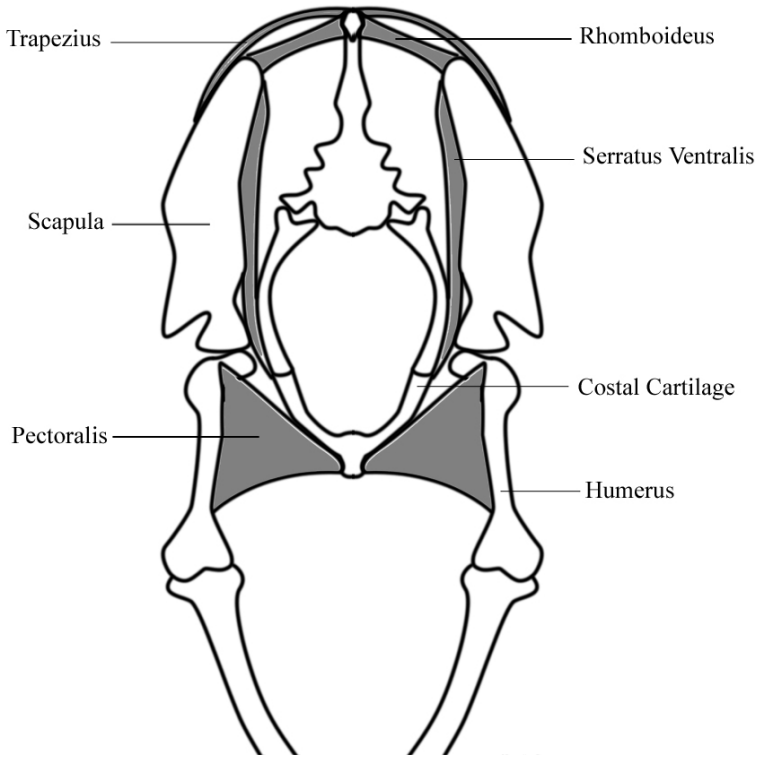

Figure 8: Pectoral sling.

The motion of the scapula is handled primarily by the trapezius and pectoralis muscles. By contracting and relaxing these muscles in different sequences the animal is able to slide its entire shoulder structure fore and aft and up and down along it rib structure as well as rotate the scapula in three degrees of freedom about arbitrary pivot points in the structure. The shoulder joint is a ball and socket joint allowing three degrees of rotational freedom, with limited lateral motion. These rotations are primarily controlled by deltoid and tricep muscles.

The next structure on the ungulates forelimb is the elbow structure. This joint is a simple hinge joint, allowing primarily only one degree of rotation freedom. There is of course, as in any real joint, a certain negligible amount of play and give in the other axis of rotation as well as negligible amounts of translation. The rotation of the elbow is handled primarily by the bicep and tricep muscles.

The knee structure, or carpal joint, has two primary degrees of freedom, or is biaxial. It only has a negligible amount of axial rotation. The rotations of this joint are primarily controlled by the radialis muscle.

The fetlock, or the metacarpophalangeal joint, has one primary degree of freedom. The rotations of the fetlock are controlled by a system of muscles, tendons and ligaments including the superficialis and interosseus medius muscle. These tendons have three primary functions: flexion and extension, support and stabilization, and shock absorption [8].

The hoof structure consists of the pastern joint and the pedal joint analogous to the knuckle joints in a human's fingers. Each joint has one primary rotational degree of rotation with very little movement and is controlled by a system of tendons and ligaments.

6.1.2 Hindlimb

In the hindlimb, the hip has three degrees of freedom as it is a ball and socket joint. The primary rotation for locomotion is controlled by the gluteus superficialis muscle and the tensor fascie latae muscle.

The stifle structure has one primary rotational degree of freedom. Its rotation is handled by the biceps femoris muscle and the quadriceps muscles.

The hock structure has one primary rotational degree of freedom. The rotation of this joint is handled by the gastrocnemius muscle and the extensor digitorum longus muscle. The hock and stifle flex are reciprocal joints, which means that they flex and extend in unison [9].

The fetlock structure, or the metatarsophalangeal joint, has one primary degree of freedom. The rotations of the fetlock are controlled by a system of muscles, tendons, and ligaments including the interosseus medius muscle.

The hoof structure, like the forelimb, consists of the pastern and pedal joint. Each of these joints has one primary rotational degree of freedom and is controlled by a system of tendons and ligaments.

6.2 Digital Anatomy

The digital anatomy is where the rig interprets the natural anatomy. Certain liberties are taken given different performance criteria, but the end product is still derived from real world examples. For the rigging of ungulate limbs the natural anatomy is broken up into functional modules for the sake of digital replication. These digital modules can then be assembled to provide a functional structure. The digital structures for the forelimb are the shoulder, the elbow, the knee, the fetlock, and the hoof. The hindlimb is divided into the hip, then stifle, the hock, the fetlock, and the hoof (Figure 9). If the modules are defined efficiently, certain modules may be reused. For this research the fore and hind

fetlock structures are identical. This allows for a level of efficiency and consistency in the rig.

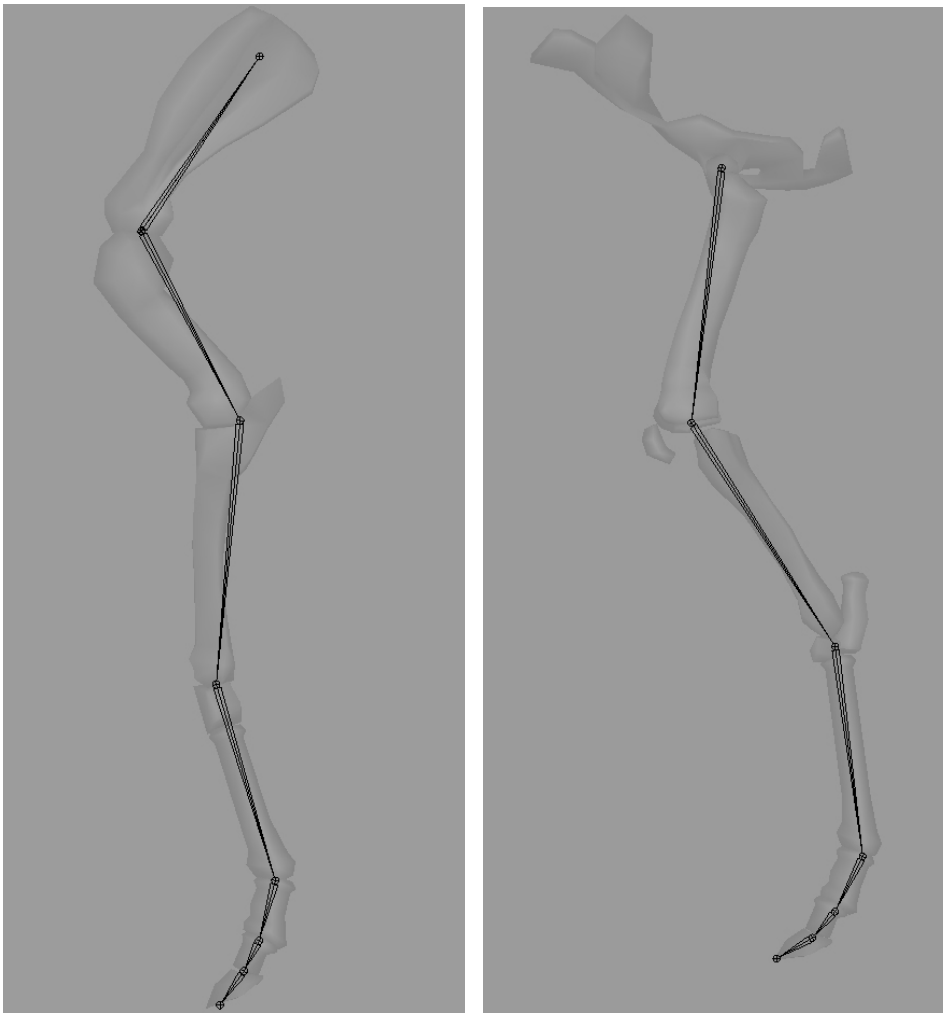

Figure 9: Digital ungulate limb joints.

For this research it is important to have a flexible and functional rig. As such, many structures are modal. A balance must be struck between too many modes and not enough control, as additional modes can add to confusion in usability. When possible the

following structures have both an IK and FK mode. Other structures have the ability to have automated transforms toggled on and off.

6.2.1 Forelimb

The shoulder structure has to mimic the same complex motion that the natural structure exhibits. It is possible to give animators unlimited control of the shoulder and scapula joints, but it would put an undue burden on their creativity without providing any limits or constraints to the motion. All arcs and motion would need to be verified by the animator, allowing for the possibility of unrealistic motion. This is not desirable in a production environment. The steps in designing the digital structure for the shoulder are identical to the steps needed to design the entire character. As in all aspects of the process, it is about layering the complexity of a character, working on the problem at different granularities. The performance criterion of the shoulder needs to be addressed first. How should it move? How much control should the animator have over different aspects of its motion? What motion should be automated and what motion should be explicitly driven by the animator? These questions are addressed in a production environment by discussion between supervisors of both animation and rigging.

How should it move? The first scapula/shoulder motion to address is the slide. The rig needs the ability to mimic the ungulates ability to slide the entire shoulder structure up and down and fore and aft along the musculature of the ribs. This is accomplished utilizing a surface representative of the shape of the rib cage (Figure 10).

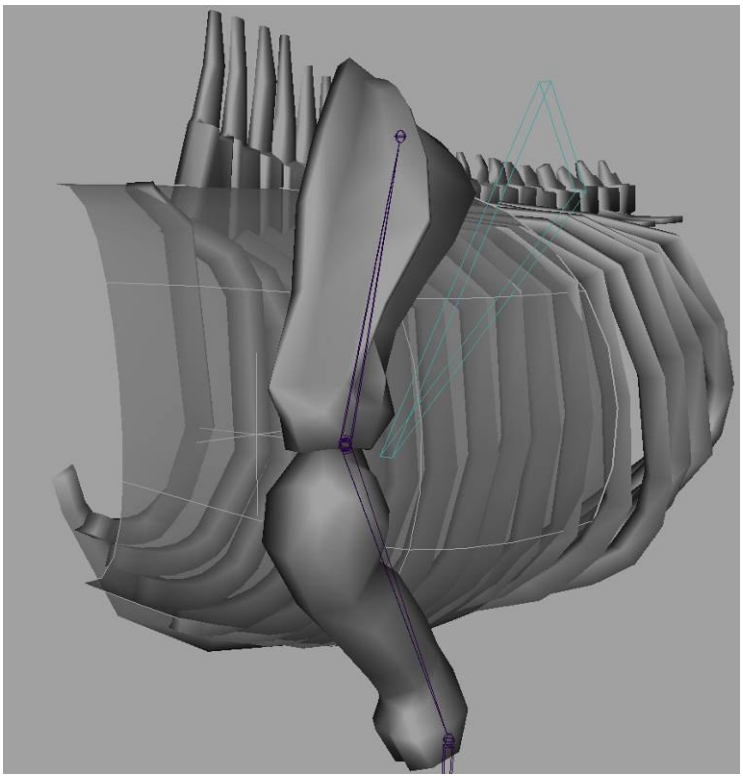

Figure 10: Digital scapula structure.

By constraining the system's translations onto the surface, the user is allowed a wide range of motion without risking separating the structure from the body. Several methods exist for constraining to the surface. The most efficient is indexing into the parametric space of the surface. The parametric space of the surface is defined by u and v coordinates as opposed to Cartesian coordinates [2]. The shoulder also requires that the scapula can be rotated along the axis perpendicular to the ungulates torso in almost a full 90+ degree sweep (Figure 11).

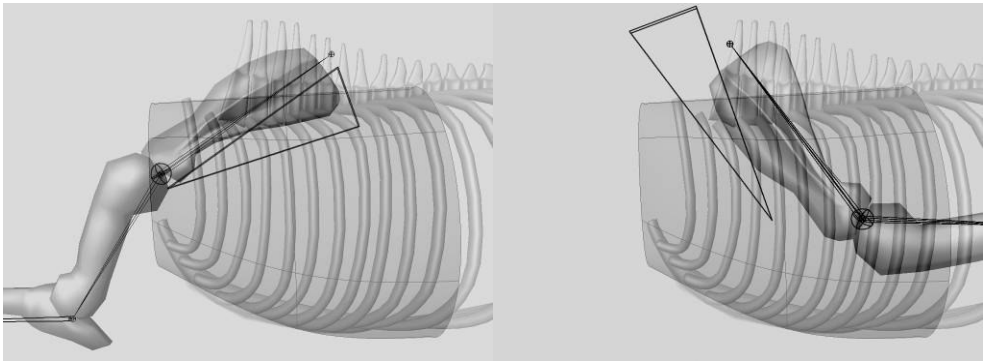

Figure 11: Rotational range of motion with modified pivot for scapula.

In addition, the scapula needs to have the ability to tilt a few degrees along the axis of the spine and be able to tilt fore and aft. In nature the rotation needs to happen about arbitrary pivot points, but by also providing the animator with pre-specified pivot point and translation controls, the animator can mimic the motion of an arbitrary pivot point, by translating while rotating. Several things must be considered when choosing a pivot point. The first issue to consider is if a pivot point is placed anywhere other than central to the joint between the scapula and humerus it is necessary for the humerus to inherit the motion of the scapula, or be a child of the scapula as opposed to being its sibling or parent. This is so the scapula is not able to pull away from the humerus when rotating. It is also necessary that both bones inherit the same translation so that one bone may not be translated away from the other. The second issue to consider when placing the pivot point is what motion that pivot point will allow. If the pivot is placed at the joint it allows the scapula and humerus to simply rotate about the joint. If the pivot is place further up the scapula in a configuration where the shoulder inherits its motion, the

rotation of the scapula will cause a rotation in the scapula and humerus as well as a translation of the humerus. Using an IK system or simple target control on the humerus any rotation created by this control would be cancelled out on the humerus and only manifest itself as a rotation in the scapula when the pivot is in the first configuration. In the second configuration both would rotate, but the humerus would slide in an arc along the ungulates body as well. This extra motion is good in layering complexity into the animation with minimal user controls. However, when an animator is refining animation and wishes to re-align the scapula, he may not do so without disturbing the position of the humerus. This is a trade off that must be resolved prior to completing this structure. This researcher chooses to place the pivot point at the shoulder for maximum controllability in a production environment. An additional control is added to the structure as a child of the humerus and parent of the scapula that pivots about the shoulder joint. The sole purpose of this control is to provide a rotational offset to the position of the scapula without disturbing the position of the humerus or successive joints.

The overall motion of the humerus must still be addressed. For this structure, no unique control for humerus translation is given, just rotation. In this configuration it is known that the humerus inherits all translations from the control of the scapula control, however there is still freedom to further control its rotation. The humerus is designed with a modal functionality. In the FK mode the humerus inherits all rotation and translation from it parent, giving the user the ability to further rotate the humerus about

its pivot point using the provide x, y, and z rotation controls (Figure 12). The other mode is a pseudo IK mode

Figure 12: Shoulder FK structure.

In many arm structures it is common to have the humerus under the control of an IK structure with its root at the shoulder and its goal at the knee. In such structures it is acceptable to animate by placing the knee using the goal and allowing the IK to solve the angles for the rest of the arm. This method does not provide accurate control for an ungulate as ungulates' knee joints are not near the end of their limbs. Ungulates walk on only the tips of their digits covered with hooves [10]. Controlling them from the knee does not provide enough control over their contact with the ground. They must be controlled with a goal at the bottom of their hoof (Figure 13).

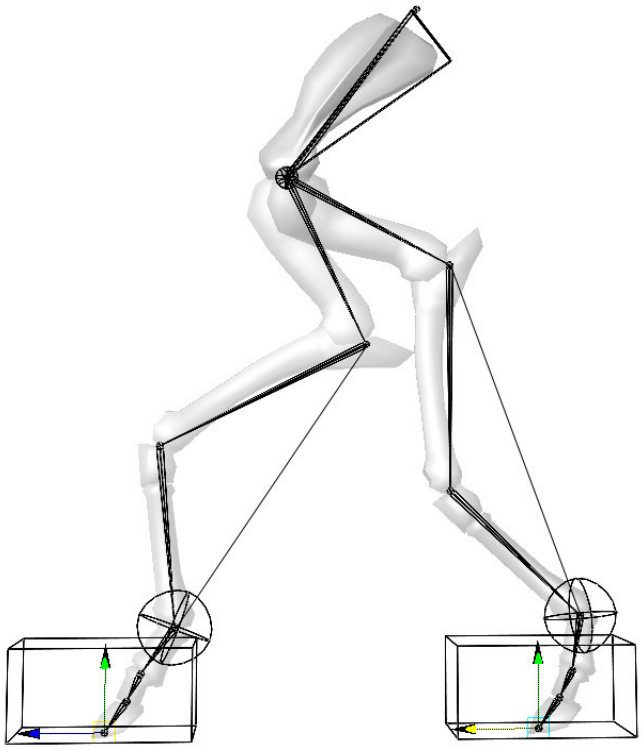

Figure 13: IK structure rooted at elbow with its first goal at the fetlock.

For this reason the structure built starts the IK chain at the elbow instead of the shoulder, leaving the humerus out of the IK chain all together. There is an overlapping structure referred to below as knee lock that integrates the control of the humerus into a different IK solution. That IK structure is used for locking the knee straight for certain sections of an ungulate walk cycle.

The digital knee structure has three modes, FK, IK, and knee lock. FK is the most simple. X, Y, and Z rotation controls are provided to rotate the knee and its children on top of any transformations it has inherited from it parents.

The IK structure for the knee is rooted at the elbow and terminates at the fetlock. As the goal for the IK is moved the fetlock joint follows its position and orientation, and the knee's rotation is calculated for the user. A pole target for the direction of the bend is placed out in front of the knee to provide a unique solution to the IK solver.

The most complex structure created to control the knee is the knee lock functionality. Through portions of an ungulates walk cycle, the animal will plant its hoof and lock its knee straight as it pushes its body forward (Figure 14).

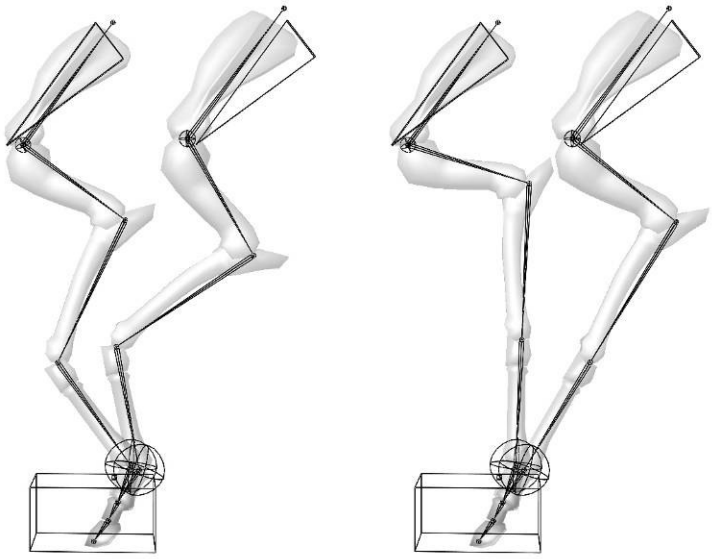

Figure 14: Two poses shown with and without kneelock.

Doing foot plants is a task best handled by IK as it eliminates the need for counter animation. Fixing a rotation in FK is as simple as not animating the rotational

control, but given a fixed goal (a hoof plant) and a moving root (the ungulates body moving forward) it is very difficult to counter animate the shoulder rotation to keep the knee from bending or from pulling the hoof off of its goal. This re-introduces the problem of counter animation. The desired position of the knee is perfectly straight, therefore certain liberties can be taken with the skeletal structure. In this case a false limb was created. The false limb contains fewer joints than the real limb. It has a duplicate shoulder joint, a duplicate elbow joint, and a duplicate fetlock joint. There is no duplicate knee joint. An IK chain is then constructed that is rooted at the false shoulder and terminates at the false fetlock. The goal for this IK chain can be the same goal as the first IK chain as well as the same pole target. Now when the leg is animated on a follow through the false leg shows the correct orientation for the humerus, radius, and metacarpal bones. By constraining the orientation of the true shoulder to the false shoulder, the real skeletal structure will maintain a locked elbow position. The constraint of the shoulder orientation is then driven by a control labeled knee lock, allowing its value to be keyed *on* and *off* as desired.

The next digital structure is the fetlock. The fetlock is a modal structure providing both IK and FK options. As on the knee the fetlock is not given its own set of translation controls, only rotations (Figure 15).

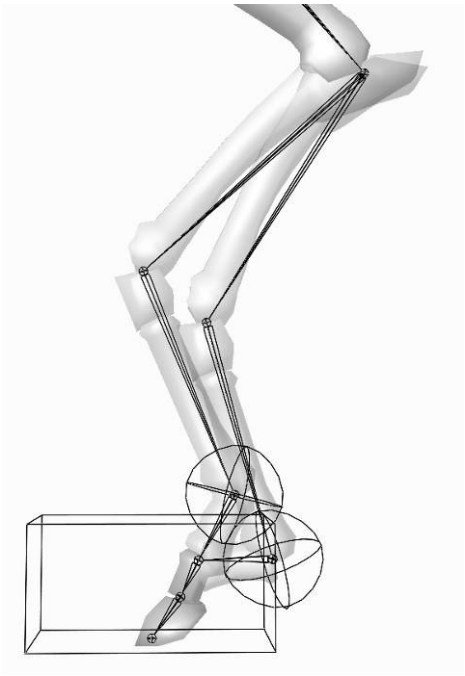

Figure 15: Result of rotating fetlock control.

The IK solver for this limb terminates at the fetlock, making it possible to either inherit the orientation of the goal or ignore it. The IK will still solve the other joints properly. It is also possible to indirectly control the goal by providing it with a parent. This allows the user to work with a control offset from the original goal. For the ungulates limb the IK goal is ultimately controlled by a node that is positioned at the front most tip of the hoof. By utilizing this control an animator is able to place the control on the ground, thereby planting the hoof, and then rotate the control, allowing the ungulate to rotate its leg about the tip of its hoof. This FK and IK control structure do not provide all the control needed to fully animate this structure in the limb. When an ungulate applies weight to the limb there is varying amounts of give in the fetlock and

successive hoof joints that act as a shock absorber. It is necessary to create a structure that can accommodate this motion while still allowing the IK to solve.

Two solutions exist in this methodology that allow for this motion. One is a simple FK-like rotation that takes place around the first joint of the hoof. The other is a more automated give that can be set and driven on by a percentage. The first is the most straightforward implementation. An extra node with rotations on it is created sharing the same pivot as the first joint in the hoof. The goal of the IK structure is then parented to this node. The extra node is then parented into the control at the tip of the hoof. Now rotations on the extra node allow the fetlock and its parent joints to arc about the first hoof joint. The next system automates this. The methodology is to rotate the extra node for the user so that a relationship is created between the proximal phalanx and the radius. As the radius changes orientation so would the phalanx. Unfortunately any structures that derive the motion of the phalanx from the motion of the radius would create a cycle because the motion of the radius is ultimately derived from the rotation of the phalanx through the IK solver. In more simple terms the phalanx depends on the radius therefore the radius cannot depend on the phalanx without creating a cycle.

Another structure that solves in parallel to the original IK must be created to avoid this cycle. This is done by aiming the extra node at yet another node created behind the limb. Now as the new node is translated the proximal phalanx aligns itself with it. This in itself does not solve the problem. It has merely changed the control from a rotational control to a translational control. A system must be constructed to drive the translations of the new node.

A simple way to create a relationship between the proximal phalanx and the radius is by constructing a quadrilateral between the radius, the metacarpal, and a false reverse radius and reverse metacarpal (Figure 16).

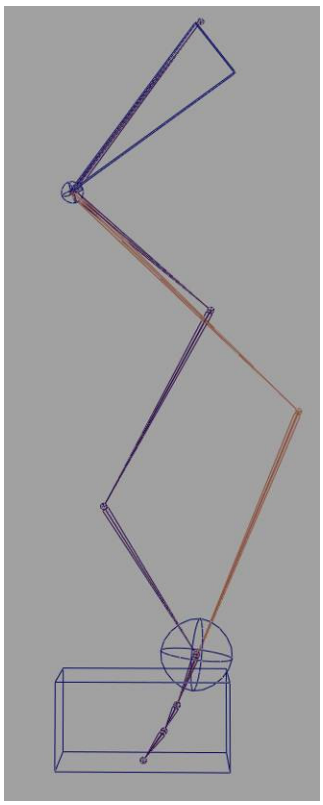

Figure 16: Reverse leg used to auto place fetlock rotation.

Two sides of the quadrilateral already exist in the base skeleton. The existing IK structure terminates at the elbow; therefore a cycle can be avoided by only creating dependencies on nodes that exist above the elbow in the hierarchy. The first node above

the elbow is the shoulder. The second two sides of the quadrilateral can be created as its children. They are created by first creating a false elbow joint, followed by a false knee joint and a false first hoof joint. The false knee joint is placed behind the true knee in line with the aim of the proximal phalanx. An IK solver is then created that roots at the false elbow and terminates at the false first hoof joint. The goal of the new solver is parented to the node that controls the give in the fetlock. The node where the phalanx points can now be translationally constrained to the new false knee. Subsequently when then IK handle for the whole leg is moved the fetlock automatically adjusts its give (Figure 17).

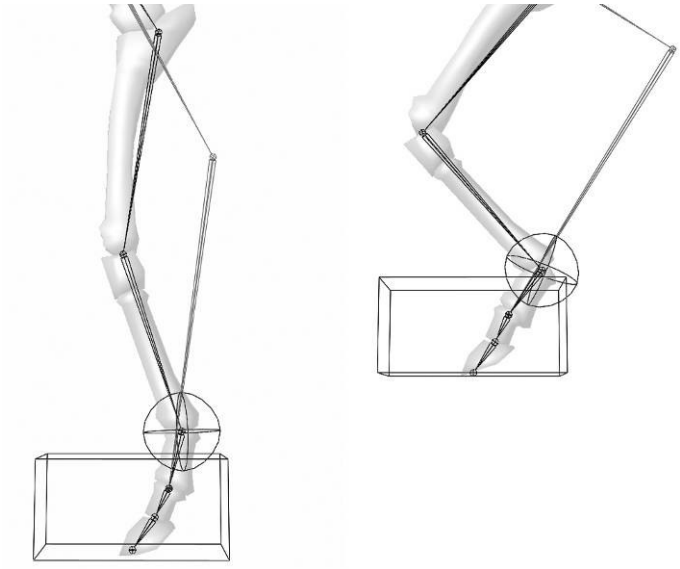

Figure 17: Demonstration of fetlock motion when IK handle is moved.

A rotational constraint between the manual give and the automatic give is then created. As the user dials the constraint on and off it controls what percentage of fetlock give is automated, while still providing a manual offset. This same structure can be reused to control the fetlock of the hindlimb.

The hoof has a very limited range of motion. The main extent of the needed motion is to curl when running and give when bearing weight (Figure 18). The digital structures created are referred to as curl and squash. The curl is a simple structure that requires a single animatable parameter. The single degree of freedom of all hoof joints is then depended on the parameter using a simple formula. The squash structure is more complex.

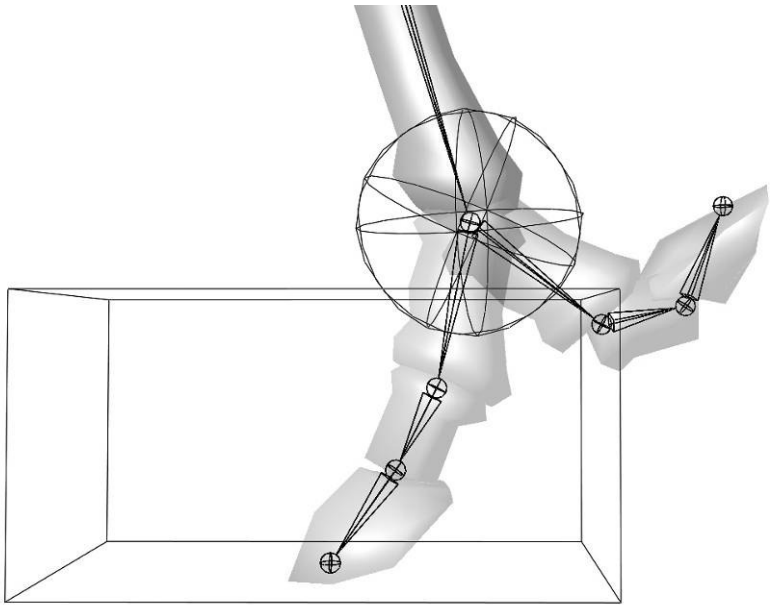

Figure 18: Curl control.

The squash control, like the curl, can be a single parameter, but the expression is more involved. The basic concept is a control that allows the first joint of the hoof (pastern joint) to lower and rotate at the same time so that the tip of the hoof does not penetrate the ground plane (Figure 19).

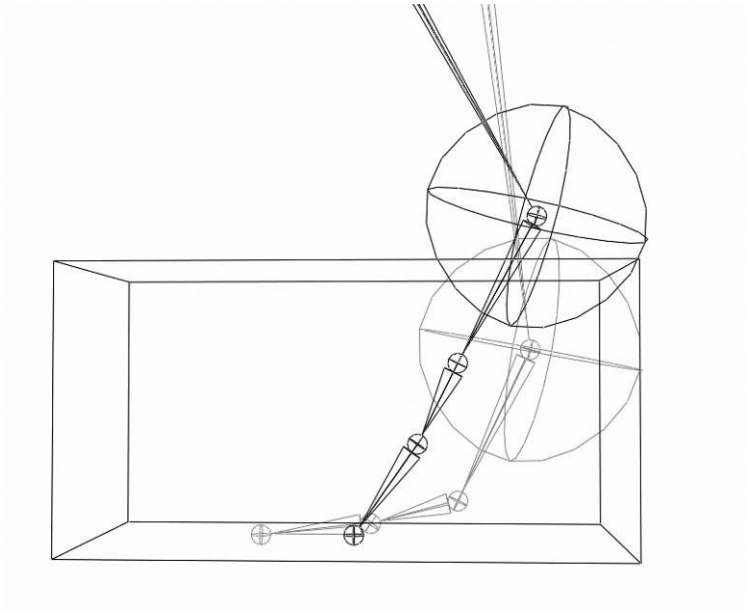

Figure 19: Squash control.

This, like an IK structure, can be calculated using simple trigonometry. The distance from the pastern joint to the tip of the hoof is a fixed distance. Given the distance between the pastern joint and the ground plane, the angle required to keep the

tip of the hoof on the ground can be calculated. The squash parameter adjusts the height of the pastern and the formula continuously re-evaluates the necessary angle.

The squash structure is identical on both perissodactyl and artiodactyl ungulates. One addition is made for ungulates with more than one toe. Same as the toe squash; it is desirable that each toe spread a specified amount. For multiple toes the squash structure is duplicated for each toe. Each structure is then parented to a rotational control that rotates each toe away from each other. A formula can then be written to allow the squash parameter to drive this spread rotation.

6.2.2 Hindlimb

The hindlimb of the ungulate does not require nearly as complex a system as the forelimb. The structure used for the fetlock and hoof are identical. The complexity of the forelimb comes primarily from the complexity of the scapula structure and the tendency to lock the knee straight when walking.

The digital hip structure is the hierarchical root of the hindlimb. As a joint, the hip needs to move fore and aft, but also requires subtle rotation in the other two degrees of freedom. In FK the hip requires three degrees of rotation. When designing the IK structure for the limb, the overall motion of the leg must be considered. When ungulates walk or run the bones of their leg maintain a distinct relationship. The hock and stifle joints are reciprocal joints which forces the femur and the metatarsal bone stay close to parallel to one another through the entire range of motion (Figure 20). As such it is necessary that the IK structure helps maintain this relationship.

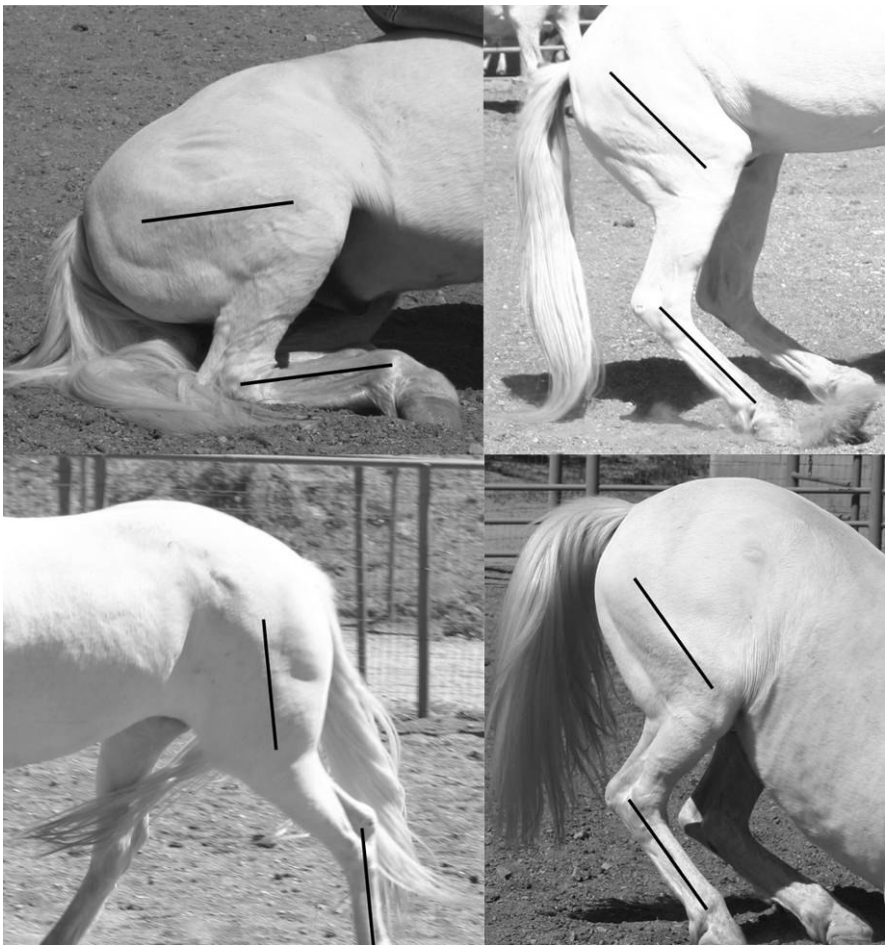

Figure 20: Parallel bones created by reciprocal motion in stifle and hock joints.

In order to accomplish this, the hip, stifle, and hock must be controlled by the same IK system. This is accomplished through a false reverse leg consisting of a hip, false reverse stifle, and fetlock (Figure 21). The structure is constructed so that the bone between the false reverse stifle and the fetlock is aligned with the metatarsal bone. The length of the two bones in the reverse leg need to total the length of the hindlimb from the hip to the fetlock so that they align at full extension. The distribution of this length

can be altered to provide additional control. The reverse leg is rooted at the hip with its IK goal at the fetlock while the hindlimb has two IKs. The first IK is rooted at the hip with its goal at the hock. The second IK is rooted at the hock with its goal located at the fetlock. The two IK goals for the hind limb can be parented to the bone of the reverse leg created between the reverse stifle, and the fetlock joint.

Figure 21: Reverse leg structure used to generate reciprocal motion in stifle and hock joints.

Now any translation of the reverse legs goal results in movement of the hindlimb, keeping a fixed angular relationship between the femur and the metatarsal bone. Then by creating controls to alter the length relationship between the two reverse bones the animator has the ability to adjust the rotational distribution between the stifle and the fetlock (Figure 22).

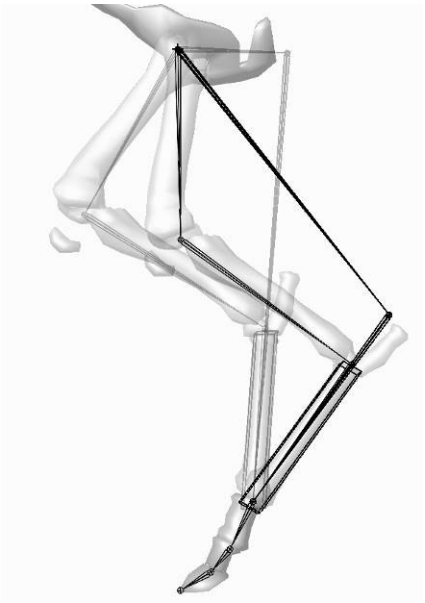

Figure 22: Changing angular distribution in stifle and hock by altering length proportions in the reverse leg.

For further control, an FK like structure is created to give the user the ability to animate a rotational offset to alter the angle of the metatarsal bone supplementing the movement provided by the IK structure. To do this a control is created at the location of

the reverse leg fetlock parented to the second bone of the reverse leg with a full set of rotations. The IKs of the hindlimb can then be re-parented to the new control. When animating the IK controls, the leg responds the same, but now any rotation placed on the new control creates a rotational offset of the metatarsal bone.

The hindlimb is completed with the addition of identical structures to the forelimb for the fetlock to the hoof.

7. INTERFACE

Ultimately each character rig will have a user base, usually a group of animators. It is common for character rigs to have controls that number in the thousands, supporting multiple modes of operation. Once the functionality of the rig is in place it must be presented in a user friendly way. This means reducing control sets to provide the most functionality with the fewest and most intuitive set of controls. The solution to this usually lies in layering the control sets, presenting the broadest controls first, but exposing layer upon layer of additional control for finer tweaking of motions.

At its most basic level, users are presented a set of animatable parameters for control of the character rig. There are several methods for populating and accessing these controls. In order to populate a control the user must choose which control to populate. This selection can usually be done by typing a command to access the control by name, through some window or interface that will choose sets of controls based on functionality, or using a mouse to pick and manipulate an icon. The last method must be integrated into the design of the rig itself in order to place icons in 3D space in an intuitive way. An icon can be manipulated in most animation packages using a visual controller that allows the user to alter an attribute in 3D called a manipulator [2]. The rigger must determine what the most intuitive icon for each particular control is. Icons can be as basic as two dimensional primitives (squares, circles, and triangles) and as complex as animating three dimensional shapes. Regardless of the complexity of the icon, it is necessary to consider its purpose.

Well designed icons provide an indication to the user of the function of the control. A well designed rig will incorporate visual cues like size, shape, color, drawing order, and opacity into the icon design (Figure 23).

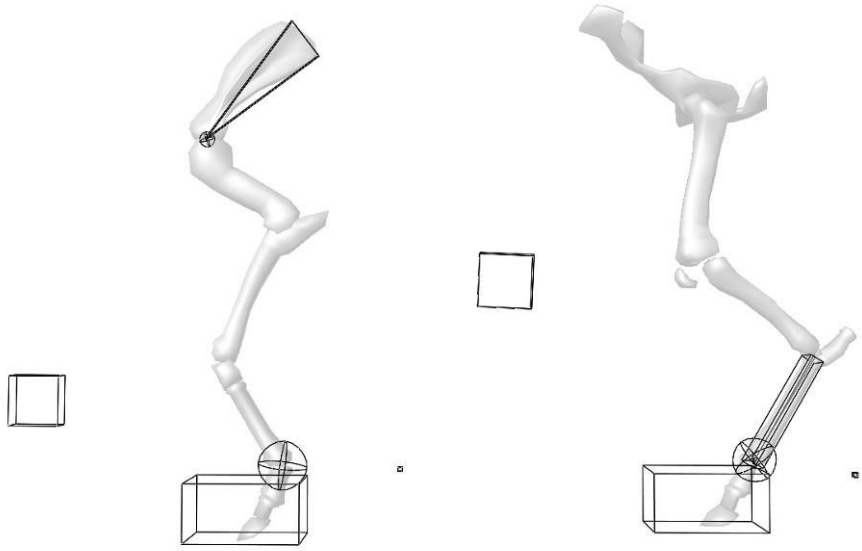

Figure 23: 11 icons used to control ungulate limb rig.

This allows users to quickly access needed controls without interrupting their workflow. The icon for scapula control might be designed to be a triangle that shares the orientation and size of the character's scapula. Icons can be color coded to provide additional information to the user. One possibility is a set of red, green, and blue icons to represent right, center, and left animation controls respectively. When an animator is

working in an orthographic side view, such a system allows them the ability to access the correct control without switching camera views or resorting to trial and error. A complex production rig can have hundreds of controls, often in very close proximity to one another. If all controls were the same size, shape, and color animators would spend much of their time searching for the correct icon.

Each icon exposes a set of animatable parameter to the user. It is the rigger's job to arrange and manage these parameters in a usable way. It is possible in one extreme to take every animatable parameter of the rig and place them on a single icon. While this makes all controls accessible to the user, it would not provide a very clean or intuitive interface. On the other extreme, every parameter could be placed on its own icon. This would be just as perplexing to the user. It is also possible to run controls for the hindlimb from icons for the forelimb. Appropriately grouping parameters is just as important. A balance must be struck so that each icon represents a reasonable grouping of related parameters.

8. CONCLUSIONS

All areas of the production pipeline benefit if the appropriate amount of research and development (R&D) and time is devoted to build a comprehensive rig. A well-designed rig can help identify and resolve modeling problems, as well as facilitate the construction of accurate deformations while reducing the excess man-hours and confusion of creating deformation based band-aids to correct rigging flaws. It can also reduce the number of animation iterations while improving the overall motion, performance, and efficiency of the character, in addition to avoiding surface-based lighting problems.

The preceding methodology is a strong foundation for the construction of any character rig needed in computer graphics. The modular and layering approaches can be utilized for the construction of far more complex systems. As with any animation, a well designed rig does not guarantee good animation, just as good animation is not an indication of a well designed rig. The use of a well designed rig does allow for greater efficiency and control for animators of all skill levels.

Commercially available animation packages as well as proprietary software provide users with varying levels of access and functionality. Not all systems presented are available across all software. It is sometimes necessary to adapt to the toolsets at hand. The underlying principals remain consistent.

Rigging by nature is a problem-solving endeavor, and as such, limitations do apply. Not every component of every rig can be constructed modularly and re-used.

Unique problems will present themselves and require unique solutions. However, the methodology presented allows for a process and philosophy, that when followed will produce a robust and feature rich foundation for both characters and a feature film production environment. Using this modular philosophy to layer a character rig together will drastically improve both rigger and animator efficiency.

In application of the presented methodology huge efficiency gains have been realized in feature film production pipelines. Animation studios are under increasing pressure to create larger quantities of work, at higher quality, with shorter timetables, and smaller relative budgets. This methodology has successfully helped meet those criteria.

9. FUTURE WORK

While this thesis outlines a successful methodology for the movement of bones, more work is required to successfully translate that motion into a believable deformation. Many more complex structures must be layered on top of this base motion to achieve believable skin motion.

Another opportunity for future research is to auto-populate animation controls based off a more limited animation. This would be beneficial in the scapula motion where there is not a one-to-one relationship between leg position and scapula orientation. In a future scenario an animator might animate the translations of the animal's shoulder and limb, but a post process could populate the scapula's rotations based on a pre-defined criteria.

It is also reasonable to expand work into interface design that allows the user faster access and organization to the vast amounts of controls that can be presented to the user. The systems can provide quick selection of controls, graphical representations of the character, and even ways to organize and present prior animation libraries or scripts for greater animation control.

Lastly, expanding the toolsets available to the rigger allows for faster and more robust solutions. Significant rigging and runtime can be saved by breaking down the rigged structures into smaller components that can be internalized in the software so it is not necessary to rebuild them from the beginning every time.

REFERENCES

[1] M. Isner and A. Sims. *The Official Softimage XSI 4 Guide to Character Creation*. Boston: Muska & Lipman, 2004, pp. 79.

[2] D. Gould. *Complete Maya Programming: An Extensive Guide to MEL and the C++ API*. San Francisco: Morgan Kaufman, 2003, pp. 473-476.

[3] C. Jones. *Chuck Amuck: The Life and Times of an Animated Cartoonist*. New York: Farrar, Straus and Giroux, 1989, pp. 260.

[4] R. Parent. *Computer Animation: Algorithms and Techniques*. San Francisco: Morgan Kaufmann, 2002, pp. 192.

[5] Aristotle. (350 BC). *On the Gait of Animals.* [On-line]. Available: www.netlibrary.com [Jan. 23, 2005].

[6] F. Walther. *Communication and Expression in Hoofed Mammals. Animal Communication*. Bloomington, Indiana: Indiana UP, 1984, pp. 1.

[7] G. Stubbs. *The Anatomy of the Horse*. New York: Dover Publications, 1976, pp. 87.

[8] L. Schultz. *Howell Equine Handbook of Tendon and Ligament Injuries*. New York: John Wiley & Sons, 2004, pp. 18.

[9] S. Harris. *Horse Gates, Balance and Movement*. New York: Howell, 1993, pp. 10.

[10] M. Allaby (1999). *A Dictionary of Zoology.* [On-line]. Available: www.netlibrary.com [Jan. 23, 2005].

VITA

William Lawrence Telford Jr

B.E.D., Texas A&M University, December 1997

M.S. Visualization Sciences, Texas A&M University, December 2006

Visualization Laboratory

C418 Langford Center

Texas A&M University

3137 TAMU

College Station, Texas 77843-3137