Motion Doodles for Quadrupeds:
How to Draw and Animate a Cat in 15 Seconds

CPSC 533b Course Project Report

Claus Beringer
University of British Columbia
Department of Computer Science
beringer@cs.ubc.ca

Karl Matthias Hamel
University of British Columbia
Department of Computer Science
kmhamel@cs.ubc.ca

April 25th, 2003

Abstract

This paper presents a technology that enables an untrained user to draw and animate a quadruped in tens of seconds. The system recognizes a quadruped drawn by a user and creates a skeleton from the sketch. Following a few simple rules the user is able to draw the movement she wants the quadruped to do. The system will parse the sketch and interpret it to let the quadruped walk, jump, trot, gallop or sit. To do all of this the system uses several techniques to understand the drawings, as well as key framing, controller based animation and inverse kinematics to animate the quadruped.

Keywords: Animation, Sketching

1 Introduction

This paper presents the results of a research project for the course "Algorithmic animation" taught by Michiel van de Panne at the University of British Columbia in 2003. The motion doodle system for human characters, i.e. bipeds, by Burke, Thorne and van de Panne [1] inspired us for this project and is the basis of our system. Similar to their system, ours allows the user to easily sketch and animate an articulated character in tens of seconds, whereas we focused on quadrupeds, such as cats and horses.

2 Motivation

The idea behind this system is to create a fast, easy to use, but still flexible and powerful technique to build animations.

Most of the current research tries to improve the technology of animation. New methods are found and we can animate and render things of a complexity we never thought possible. However only a small effort is put into finding ways to make it easier for users to create good animations.

Systems like the one we introduce in this paper can also fill the gap between simple storyboards and animations. Even today, when a complex animation has to be created, drawings are used to illustrate what is needed. This is done because we do not have animation systems which are capable of building a good and flexible animation in a short time and an easy way.

Most of the existing animation systems can only be used by trained users and it is even more unthinkable to let a child create a good animation. Children love to draw, it strengthens their imagination and thus it would be good to allow them to animate what they draw. However, to let this happen we need very easy to use animation tools.

Based on these ideas Burke, Thorne and van de Panne introduced a system [1] they called "Motion Doodles", which enables a user to draw a biped character and animate it in 10 to 20 seconds.

The system we introduce in this paper is based on their work. The next step after being able to animate bipeds is to animate different characters, especially quadrupeds. This step makes sense, since most mammals are quadrupeds. It is also very interesting to see if the different gaits and forms of motion the quadrupeds on our planet developed are transferable to a system like this.

Thus, we want to make it possible for a user to draw and then animate different quadrupeds in a fast and easy way.

3 Related Work

This project is based on the work of Burke, Thorne and van de Panne [1], which they presented at UBC's Imager-Lab-meeting. We reused several parts of their character recognition system, especially low-level methods, as well as the basic architecture of their approach.

In the field of computer vision there has been a lot of research in recognition of human drawings. However, the approach was very different and the aim was to understand human drawings in a very general way. A brief overview on this is given in [7] by Randall Davis.

In Computer Animation there is a lot of work on key framing techniques and controller based animation. A framework to combine and manage different controllers for physics based animation is described in *Composable Controllers for Physics-Based Character Animation* by Faloutsos, P. et al. [6].

For papers connected to our inverse kinematics solver we refer to Chris Welman's *Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation* [2], which describes the method we implemented.

4 Approach

In the current approach we concentrate on a single quadruped. As a first specification of a quadrupedal creature we chose the cat because it is fairly easy for an average user to imagine. There is however one problem in the development of a system for quadrupeds: humans don't intuitively know how animals are built, and unfortunately very often a number of things (e.g. fur) makes it hard to find out about the animal's skeleton. Since for humans it is much harder to imagine the physique of a (quadrupedal) animal compared to that of a human, we had to put extra effort in the development of a representation which is authentic compared to a real animal, but also applicable.

Thus, on the one hand we analyzed appropriate anatomy literature and photos of Eadweard Muybridge [9] of cats. On the other hand we also asked numerous people to draw cats – at the beginning without any constraints. Later we merged the gathered information and developed some loose constraints, such as the number of strokes allowed for drawing the cat. Finally, we came up with a fairly good representation that requires only a few basic constraints.

The developed system comprises two major parts, one for sketching the character and one for the motion sketching which takes also care of all computations for the animation.

4.1 Character Sketching

The user sketches the quadruped in a 2D-side-view using 12 ellipsoidal figures, comprising the head, neck, torso, upper foreleg, lower foreleg, heel and toes on the foreleg, upper rear leg, lower rear leg, heel and toes on the rear leg, and finally the tail, where the sketch has to satisfy certain constraints. Additionally, the user might add some annotations to the body parts. From the sketch we infer the joints of the articulated character. In the next step we identify the different body parts, such as the torso or the tail, and then build the skeleton from this information. The sketched body parts are assigned to the corresponding skeleton parts. The skeleton data is then passed to the motion sketching / animation part of the system.

4.2 Motion Sketching

The user draws strokes (arcs and a loop), which are recognized and interpreted. The more or less abstract, but intuitive sketches specify the type of motion, the timing as well as the start and respective end points of the motion. For example, a long and high arc will be interpreted as a jump of the height and the length of the arc. The computation of the motion, i.e. the joint angles of the limbs, is based on a pool of controllers, one for each type of motion, such as a normal walking step or a leap. The controllers themselves work based on key frames and an inverse kinematics system.

5 Implementation

Our system includes two core components: the character sketching and the motion sketching. First, we will concentrate on the character sketching part. Since our system is based on the one of Burke et al. [1], and a number of lower level functions are taken from there, we describe some parts in less detail. For more detail on these parts the reader is referred to [1].

5.1 Character Sketching

The creation of the articulated character starts with the drawing by the user. She sketches 12 links, comprising the head, neck, torso, upper foreleg, lower foreleg, heel and toes of the foreleg, upper rear leg, lower rear leg, heel and toes of the rear leg, and the tail. Each link is defined by one continuous stroke and the order in which the user draws the links is not important. However, there are some constraints for the location and orientation. The cat must be drawn facing the right side of the screen, and the tail must be the most left link and the head the most right link. A "normal" standing pose is also mandatory. An example can be seen in Figure 1 (a). The abovementioned assumptions were necessary in order to ensure a correct recognition. In contrast to the human character in [1] for the quadruped several issues occur that make it necessary to constraint the sketch at this stage.

Once the character is sketched, the system automatically infers the joint locations and identifies the body parts. After having created the second foreleg and rear leg, the system finally builds the skeleton with the sketches assigned to the corresponding bones.

5.1.1 Inferring the Joints and the different Body Parts

When the user draws, each continuous stroke is recognized as one link. Due to the required constraints and the facts we know about the connectivity of the links, it is easy to identify the body parts and infer the skeleton.

Once a link is drawn the principal axes are computed (see Figure 1 (b)) and an oriented bounding box is fitted to the link (see Figure 1 (c)).

Once 12 links are drawn, the system determines the joints and with it the connections of the links. For the major axis' endpoints of each link the algorithm finds the closest link by computing the Euclidean distance to every point of the other links. When the closest link is identified the joint location is computed as the geometric intersection between the major axes of the two links. If the angle between the axes is less than 20°, i.e. they are almost parallel, the joint is computed as the midpoint of the line connecting the endpoints of the major axes of the two bounding boxes. Otherwise the joint would be too far away from a reasonable position.

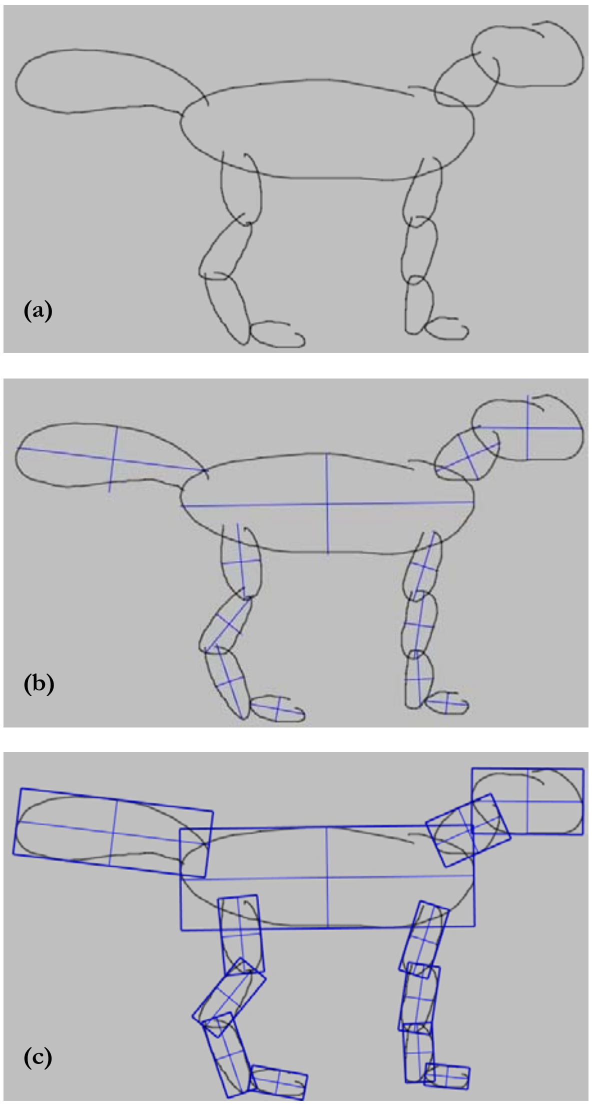

Figure1: Inferring the skeleton from the sketch
(a) sketched cat (**b)** principal axes
(c) bounding boxes

The neck joint and the tail joint first are situated on the torso bone. To achieve a more authentic joint position and motion behavior of the cat they are moved to the closest end point of the major axis of the corresponding link, i.e. neck and tail (see Figure 2 (a)).

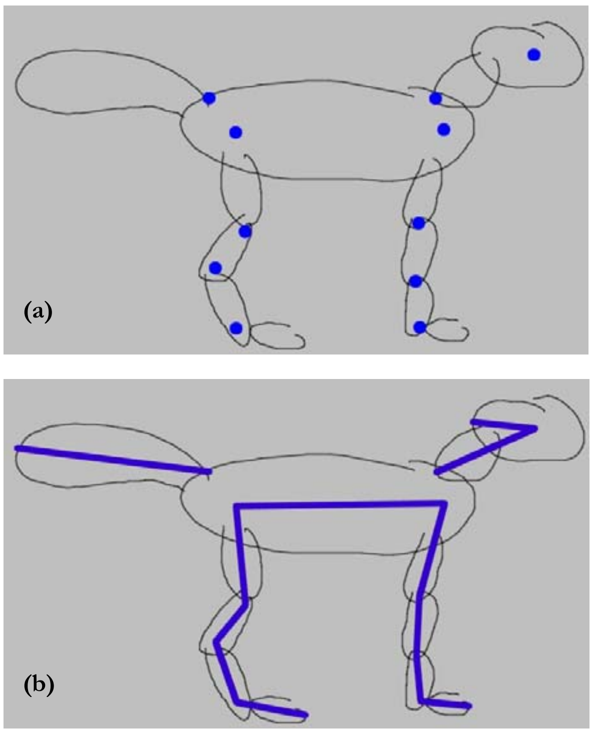

Figure 2: (a) joint locations (**b)** inferred skeleton

When all joints are determined and are assigned to the links, we identify the body parts. From the required constraints we can identify the most right link as the head and the most left link as the tail. The neck is the only link connected to the head. The torso is identified as the highest link, which is not one of the links head, neck or tail. The rear leg and the foreleg are distinguished through their relative location. And finally the different parts of the legs are identified by the implied connectivity (upper part-> lower part-> heel-> toes).

The pseudocode of the algorithm is shown in Figure 3. In some parts it is fairly similar to the one of Burke et al. [1].

5.1.2 Building the Skeleton

The bones of the representative skeleton are built by connecting the pertinent joints. Thus, for example, the upper fore leg bone will be built connecting the hip and the knee. However, there are exemptions. The bones for the body parts, which do not have two joints, are computed as the line from the one joint to the furthest endpoint of the sketched link.

When the skeleton is built, the torso and the tail are broken into parts, which introduces two new joints in the torso, between the lower and the mid spine, and between the mid and the upper spine, and three new joints in the tail. These joints are added at fixed fractions of the bone and allow for the bending of these parts of the articulated character.

In addition to the regular bones there are two fake bones created, connecting the neck with the upper spine, and the tail with the lower spine. They work as internal connections to complete the skeleton so that all bones are bonded. Finally, adding a second foreleg and rear leg completes the skeletal structure.

```
1.   For each sketched link
2.       Compute the principal axes and fit an oriented bounding
         box to the link
3.   For each link i
4.       For each major axis endpoint on link I, $P_i^1$ and $P_i^2$
5.           Go over all links j != i, find the closest point $P_j$
6.           If major axes of links are not parallel
7.               create joint $J_n$ at geometric intersection of
                 major axes of i and j
8.           else
9.               create joint $J_n$ at midpoint of line $P_iP_j$
10.  identify head, neck and tail applying the constraints
11.  move head joint and tail joint
12.  identify torso as highest which is not head, neck or tail
13.  build skeleton
14.      create second foreleg and rear leg parts
```

Figure 3: Pseudocode for recognition of skeletal structure

The current approach works fairly well even though it fails for weird drawings of cats, e.g. a X-legged cat, or if the pose is totally different from the required one. However, the algorithm gives reasonable results considering the assumptions that are made.

5.1.3 Adding Annotations

The first sketch of the quadruped is fairly simple with little detail. This is exactly what makes the system so easy and quick to use. Nevertheless, one might want to

add more details. In the present state, it would be nice, for example, to add some fur to the cat or design the head with ears etc. Our system allows that after the 12 links are drawn. The user can sketch further figures that are bound to the closest link and are animated as if they were parts of the link.
The possibility to add various kinds of annotations also enhances the applicability of our tool. So we take a step closer to replacing the manual storyboard used for animations and films.

5.2 Motion Sketching

The second part of the system is responsible for the motion of the character. This includes all the motion sketching and the computation while the resulting animation is afterwards displayed in the other part.

After having received the skeleton data, this part of the system tracks the mouse input of the user. Her sketches are identified and the needed motions are built. The resulting poses of the quadruped are created and send to the other part of the system, where the drawn body-parts are fitted back on the skeleton and displayed.

The movement the user wants has to be interpreted and the data extracted out of the drawing. Based on this information the correct motion has to be created. This is done by using key framing in combination with an inverse kinematics solver and constraints.

We will first describe the system design, then explain the controller technology and the key framing and finally talk about the inverse kinematics system.

5.2.1 System Design

The architecture of this system is, as mentioned before, based on the biped-system of Burke, Thorne and van de Panne [1]. We decided to reuse their basic structure but to change it, to make it more flexible and to decrease the coupling.

In Figure 4 we present an overview on the architecture.

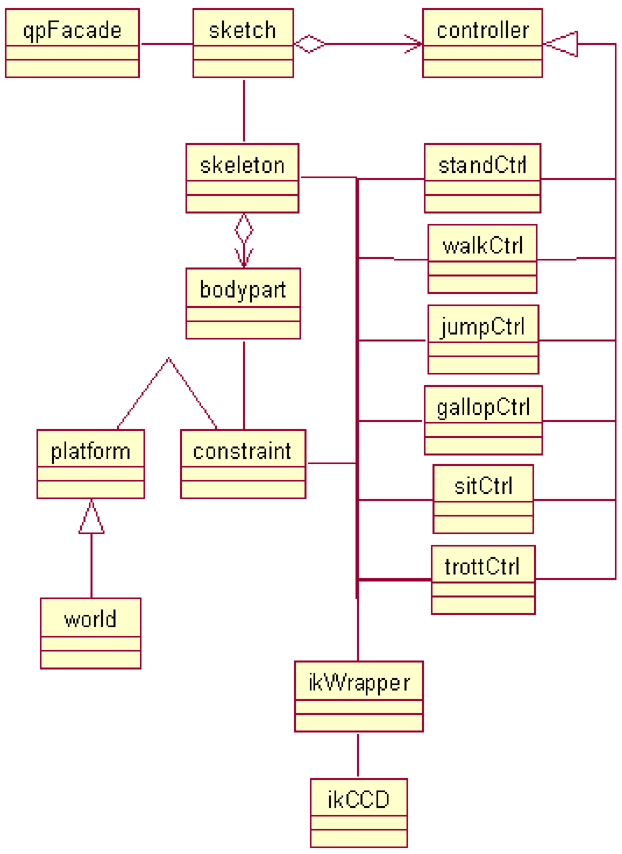

Figure 4: System Design

The `qpFacade` class encapsulates the whole part of the system from the character sketching part. All communication between the two parts go through this class. `Sketch` is responsible for the main functionality and coordinates the computation of the animation. We have several controllers, which take care of the different gaits or movements. They control the movements through key frames and constraints on the motion, which are defined by the `constraint` class, as well as through inverse kinematics, which is encapsulated in the `ikCCD` class. The quadruped itself is defined by the `skeleton` and the `bodypart` class.

5.2.2 Controller

5.2.2.1 Controller Management

Every controller in our system represents one gait or movement of our quadruped.

Each controller is able to identify if it can fullfill the motion the user wants. If the controller recognizes the user's sketch, no other controller will be started until the current controller comes to the point where it is unable to continue animating the motion the user wants. Until then this controller will control the motion of the quadruped. It will analyze the drawings the user made, extract and collect the needed data.

When a controller recognizes that it cannot continue working with the drawing of the user, it will finish its work and the `sketch` class will try to find another controller. In this case all the other available controllers will try to identify the next part of the user's drawing. If one of the controllers identifies the new part of the drawing it will go on analyzing.

All controllers that identify a part of the motion sketch will be stored in a queue and will afterwards animate the quadruped based on the data they extracted from their part of the user's input.

Since all the functionality regarding with one movement is encapsulated in one controller it is very easy to add new controller or remove others.

5.2.2.2 Sketch Identification and Analysis

A controller identifies the user's drawing based on a fairly simple algorithm. First we extract the direction of the mouse movement. The direction can be up, down, right, left or up-right, down-right, up-left, down-left. The identification of the movement is based on this information and the absolute mouse position translated into world coordinate system.

If the Y-coordinate of the mouse position (in world coordinate system) goes below the ground at certain points is very important. The height of the ground is used as a threshold to differentiate between a number of possible motions. Another important threshold is the so-called jump-line. We distinguish some of the movements based on whether the drawn sketch goes above this line or not. For example an arc-like-movement starting below the zero line, going up and down again to zero is a jump only if it goes over the jump-line. If it stays below that line it is one part of a gallop.

Tthe same information is handled to extract the needed data. In the case of a jump-like movement this can be the maximum height, the X-position where the maximum height is reached, the time the user needed to draw up to this point, the time it took to go back to zero, etc. All this data is stored in the controller until it is called to animate.

5.2.2.3 Animation

Based on the collected data the controller animates the skeleton. Most of the motions are based on key frames. However, some of them may be changed to a high percentage by inverse kinematics. All the motion is also adjusted by applying several constraints that affect the movements.

We will explain some of the techniques by using the jump-controller as a simplified example.

For a jump most of the motion during the flight is based on key frames, since we have no ground contact. Obviously it has to be adjusted to create the correct angles depending on the height to length ratio of the jump. The quadruped must start with a much higher slope if the jump is very high.

During the anticipation before the quadruped gets airborne we adjust the motion by using inverse kinematics. If the jump is higher the quadruped will crouch more.

When the quadruped hits the ground after the jump we also have to use inverse kinematics. The reason for this is that most quadrupeds put their back feet close to the point where the front feet are. Since we might have very different legs as they are based on the user's drawing we cannot rely on the key frame data. The follow-through is also implemented by using inverse kinematics.

When the front feet hit the ground we set a constraint to force them to stay at the same position without moving into the ground or sliding along it. When the back feet touch the ground we move the constraint from the front feet to the back feet. From now on the foreleg is only controlled by inverse kinematics.

The jump is a very good example of a movement using equal amounts of inverse kinematics and key frames. For a walk for example we have to use a much higher proportion of inverse kinematics. Actually only the tail, head, body and one leg at a time are controlled by key frames. All the rest is controlled by inverse kinematics.

5.2.2.4 Key frames

The key frames are stored as joint angles on a skeleton with 26 joints. We use four joints for every leg, three for the spine, four for the tail, one for the neck, one for the head and the root-joint.

The interpolation is done using the technique Burke, Thorne and van de Panne used [1]. This is a Catmul-Rom interpolation [4], which uses four joint angles as parameters. The previous angle, the current angle to interpolate from, the next angle to interpolate to and the following angle. It works very well and creates realistic interpolations without any problems.

5.2.2.5 Inverse Kinematics

For the inverse kinematics we use an implementation of the cyclic coordinate descent method. We chose this method because it is very fast and works very well.

We implemented it based on a publication by Welman [2] but adjusted it to our needs to make it faster. The technique works as follows:

Based on the current end-effector position P_C and the desired position P_D as well as the current joint q we want to minimize an error value in the end-effector-position, which can be described as a sum of a position error E_P and an orientation error E_O:

$$E(q) = E_P(q) + E_O(q)$$

as we are not interested to set the orientation of the end-effector we only minimize the position error :

$$E_P(q) = \left\| P_D - P_C \right\|^2$$

This technique simply starts at the joint closest to the end-effector and tries to move it as close as possible to the desired position. This is repeated for every joint of the body-part. The whole process has to be repeated a few times to create good results.

We use maximal and minimal values for each of the joints to create realistic movements. To make the motions more animal-like we also introduced a stiffness-factor of 0.6, which turned out to be a good tradeoff between moving the end-effector to the desired position while avoiding bending only the joints close to the end-effector.

6 Results

In the current version the tool is able to animate quadrupeds that move like cats. The user can draw a cat. The system then recognizes the quadruped and extracts a skeleton, which can be used to animate.

The user then can draw the motion she wants. Currently the system can animate the following motions: stand, walk, jump, gallop, sit and trot.

Please see figures 5 and 6 for examples on how each of the motions has to be drawn. A walk (Figure 5 (a)) is simply an up and down movement

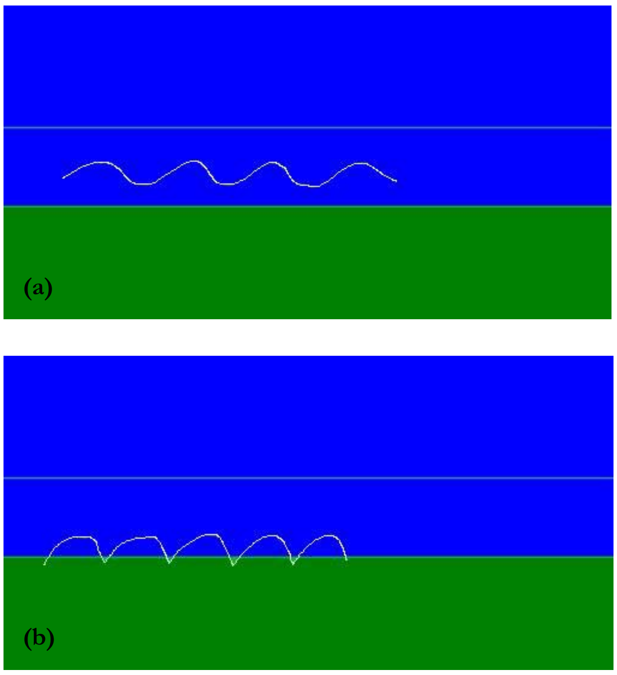

Figure 5: motion doodles **(a)** walk **(b)** trot

between the two thresholds (ground and jmp-line). A trot (Figure 5 (b)) is a bit similar but the sketch has to touch the ground in the lower points. During the trot the feet of the cat will be placed on the points where the ground-threshold is touched.

The cat will sit down if the user draws a loop like movement (Figure 6 (a)). A gallop (Figure 6 (b)) is similar to a trot but the distance between each of the lower points is much larger. A jump is an arc which goes above the jump-line (Figure 6 (c)).

The user has to draw a series of one or more motions with the mouse or a tablet. The system will recognize what the user wants the cat to do and it will put the controllers, which can handle certain parts of the wanted motion, in a queue. Starting from the first controller in that queue one controller after the other will be called to animate its part.

We are still working on connecting the character sketching and motion sketching parts in a better way. Both systems are working very well but the connection still needs some work. We might also need some adjustments when animating very weird

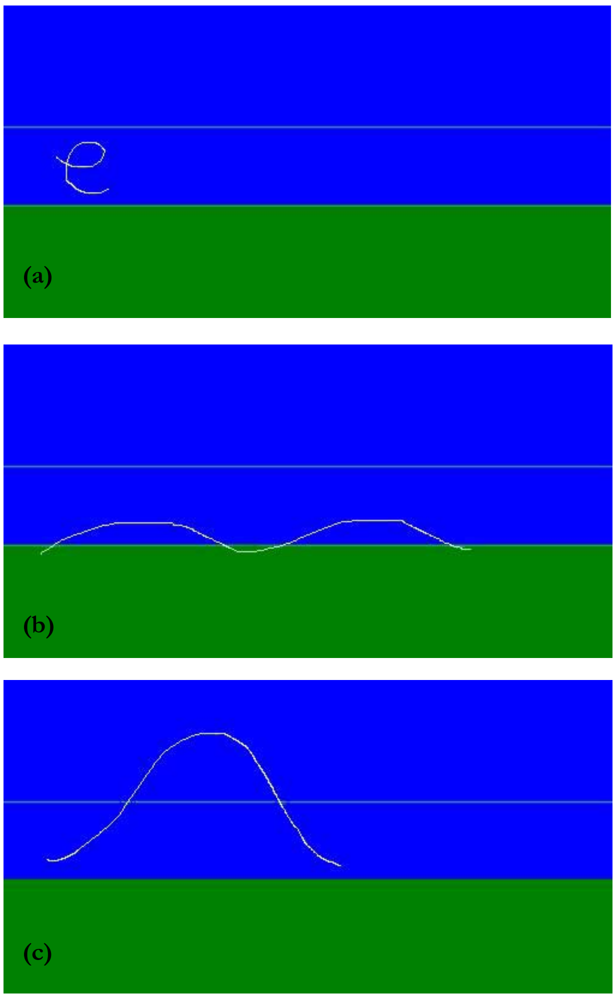

Figure 6: motion doodles
(a) sit down **(b)** gallop **(c)** jump

quadrupeds, e.g. with very long front legs and very short back legs.

We are also working on bending the tail in a more realistic manner as the user can draw it as one ellipse, while we have 4 joints representing it. For the animation a good-looking tail is very important and so we will put some effort in this.

7 Conclusion & Future Work

We have presented a system, which enables the user to sketch and animate a cat in tens of seconds without any training. It is very easy to use and allows one to produce reasonable animations in a very small amount of time. However, in the current state the communication between the character sketching and the motion sketching part is not yet finished. We will have to make some adjustments to finish the work on the cat. In addition the sketching of the character is still fairly constrained. We would like to have a more sophisticated skeleton recognition algorithm to allow for more valid character drawings.

A major future extension will be to allow for a number of different quadrupeds with their gaits. In the framework of some follow-up research work we will introduce a horse with its different gaits, such as a trot or a gallop. We hope to reuse the character sketching part with only some minor changes. The major effort will be in developing new controllers for the horse, whereas it is possible to exchange controllers with little effort due to the architecture of the animation part of the system. This first extension for horses will show that our system is fairly easy to extend for other quadrupedal creatures.

Given different creatures with different gaits it would be very interesting to have the possibility to "mix" the creatures in order to build new creatures, such as 30%-cat-and-70%-horse-creature.

8 Acknowledgements

We want like to thank Dave Burke, Matthew Thorne and Michiel van de Panne for their help and their kindness to allow our project to be based on their work. This project would otherwise not have been possible in the amount of time available.

9 References

[1] Burke, D., Thorne, M., van de Panne, M., *Motion Doodles: A Sketching Interface for Character Animation*. not published

[2] Welman, C., *Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation*. http://fas.sfu.ca/pub/cs/theses/1993/ChrisWelmanMSc.ps.gz B.Sc, SFU, 1989

[3] Thorne, M., *CPSC 533B Course Project - An Inverse Kinematics System*. http://www.cs.ubc.ca/~mthorne/cpsc533project.html

[4] Catmull, E., Rom, R. *A class of local interpolating splines*. Computer Aided Geometric Design, R. E. Barnhill and R. F. Reisenfeld, Eds. Academic Press, New York, 1974, pp. 317–326

[5] Dunlop, R., *Catmull-Rom Splines*., last edited 5/21/2002 http://www.mvps.org/directx/articles/catmull/

[6] Faloutsos, P., van de Panne, M., Terzopoulos, D., *Composable Controllers for Physics-Based Character Animation*. Siggraph 2001

[7] Davis, R., *Position Statement and Overview: Sketch Recognition at MIT*. Submitted to 2002 AAAI Spring Symposium on Sketch Recognition

[8] Schneider, P, *Phoenix: An Interactive Curve Design System Based on the Automatic Fitting of Hand-Sketched Curves*. MS Thesis University of Washington, 1988

[9] Muybridge, E., *Animales in Motion* Edited by Brown, L. S., Dover Publications, Inc. New York