Automatic rigging tool for centipedes and millipedes

Masters Thesis

of

Eduard Zell

19.Aug.2010

Supervisor: Jon Macey

NCCA-Bournemouth University

Contents

1 Introduction

The idea to create an automatic rigging tool for centipedes and millipedes was born shortly after the first presentation of the master projects. During a talk at the campus with Kier Eyles, who studies Computer Animation at the NCCA, the question was raised how a rig would look like for a centipede or millipede. This question resulted in the present master project.

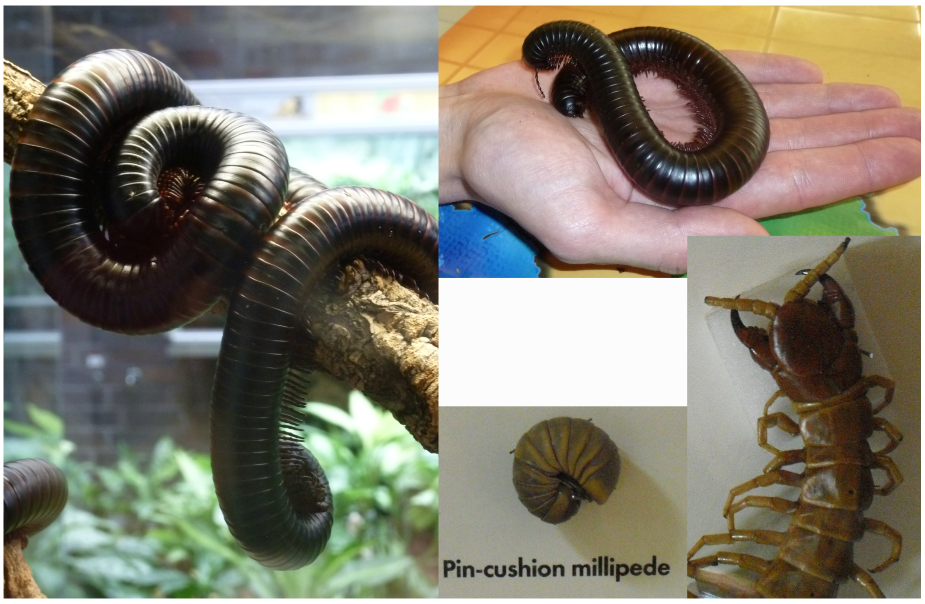

Figure 1: Examples for different centipedes and millipedes. Pictures taken at the Natural History Museum in London or the Colgne Zoo

Normally, a rigging project starts with the collection of reference material of anatomy and similar example rigs that others have already created before. The problem in this case was that the reference material was very limited. Most of the books about arthropods[1] focus either on bugs, spiders or crabs. Centipedes and millipedes are either not covered at all or only within some pages. Furthermore, most of the photo material found on the internet showed only a few features of the body. It has been even harder to find high quality videos to observe the motion of centipedes and millipedes. Finally, observation of real creatures was done in the Natural History Museum in London and the Cologne Zoo. This research improved significantly the understanding of anatomy and locomotion of centipedes and millipedes and helped to analyse photos or videos

[1]The phylum Arthropods includes all invertebrate animals with an exoskeleton. These are among others, insects, crabs, spiders or centipedes and millipedes [Cla73].

of these creatures more efficiently. After the analysis of the body, it became clear that with the number of similar segments that are needed to rig myriapoda,[2] it would be better to create the rig procedurally than stitch it together within a software package. This would reduce errors, increase the consistency of naming conventions and could be finally used as a general rigging tool for centipedes and millipedes. As the aim was to create a successful rigging tool, special attention has been paid to provide big flexibility to the animator. At the end, the animator should decide whether he wants to animate parts of the body by himself or to apply automatic or semi-automatic functions.

Unfortunately, there are nearly no documented rigging solutions for millipedes and centipedes, and even the existing documentations are too short to be useful. Furthermore, the rigs are mostly limited to walking cycles. This makes this work very unique and interesting, but as well more difficult. Nevertheless, two solutions have been found to simplify the animation process for the animator. The first solution demonstrates a compromise between full control of each body part and simplification. In this case, one master controller is created that defines the motion of several legs or spine segments at once. In the second solution a procedural walking millipede or centipede has been successfully implemented. Finally, both rigging systems have been combined in order to give the animator a high level of control over the rig but at the same time the comfort of automatically created animation.

This thesis summarises all the research that has been done to create a rigging tool for centipedes and millipedes. In the first sections, the focus will be on the anatomy and locomotion of millipedes and centipedes, which is very important to understand before rigging a creature. In the last sections, the focus will be on the solutions and problems encountered during the creation of the tool. As the tool has been created with Python and MEL in Autodesk Maya, it is assumed that the reader has a basic knowledge of the Software and some programming experience.

2 Anatomy of millipedes and centipedes

Millipedes (Diplopoda) and centipedes (Chilopoda) are part of the sub-phylum myriapoda, which is part of arthropods. As arthropods, biologists classify creatures like spiders, insects, scorpions or crabs and shrimps. The most visible characteristic of arthropods is a segmented body with an outer cuticle, the exoskeleton. This characteristic is very different to mammals, where the skeleton is completely covered with other organs which results in body deformations in

[2]Myriapoda is a subphylum that contains centipedes, millipedes and two other similar classes [Cla73].

case of movement. In opposite to mammals, bodies of arthropods are only deformed at joints where the cuticle remains soft and flexible. Other parts of the cuticle are hardened and inflexible [MO01, p.175]. Muscles of arthropods are hidden under the exoskeleton and are therefore invisible. In general, a joint of arthropods can be imagined as the intersection point of two non deformable tubes, one stitched into another. The surface between the tubes is soft and therefore deformable. Depending on the size of the inserted tube, the joint may be very flexible or completely rigged, also an offset of the inserted tube from the centre may change the degree of freedom in certain directions as described in [Cla73, p.36-39] or figure3.

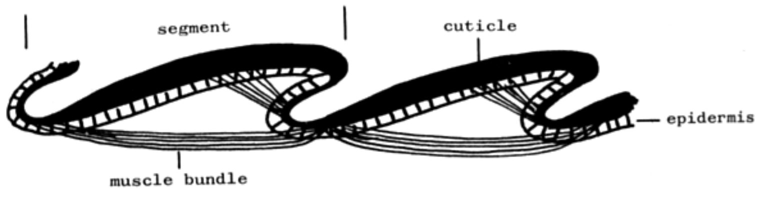

Figure 2: Cross section of an exoskeleton [WWE80, p.108]

Millipedes and centipedes consist both of numerous segments (more than 15) with one or two pairs of legs. All segments with legs tend to be similar for one creature, but can differ in length and form between the several species. The head of the millipedes and centipedes consists of two antennas, primitive eyes, mandible and at least one pair of maxillae[3]. As centipedes are predators in opposite to most millipedes, they have a second pair of maxillae as poison claws. Another difference between centipedes and millipedes is visible at the last body segment, where centipedes have long legs facing to the back and acting like antennas. As the name already implies, centipedes have less legs than millipedes in total and per segment. Centipedes have only one pair of legs per segment and between 15-177 segments [WWE80, p.159-161]. The number of segments for millipedes is similar, although in average higher than of centipedes, but because there are, excluding the first four segments, two pairs of legs per segment, the total number of legs is higher. The creature with the most legs (more than 600) is *Illacme plenipes.*

Further differences between centipedes and millipedes derive from their behaviour in nature. As centipedes are predators eating small insects and spiders, they are fast, flexible and flat. Their body is soft, except strong cuticles at the back and the lower side. Their legs are placed at the sides of the body. In opposite, millipedes digest plants or decaying material and live in soil. Their

[3]see also figure4

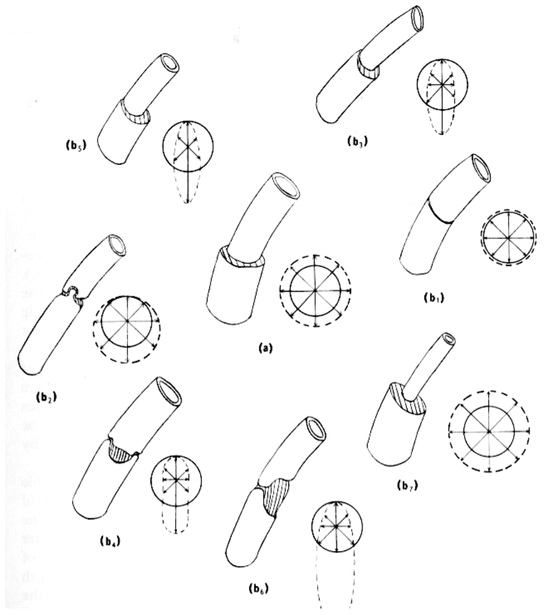

Figure 3: Main joint types of arthropods with different level of flexibility [Cla73, p.37]

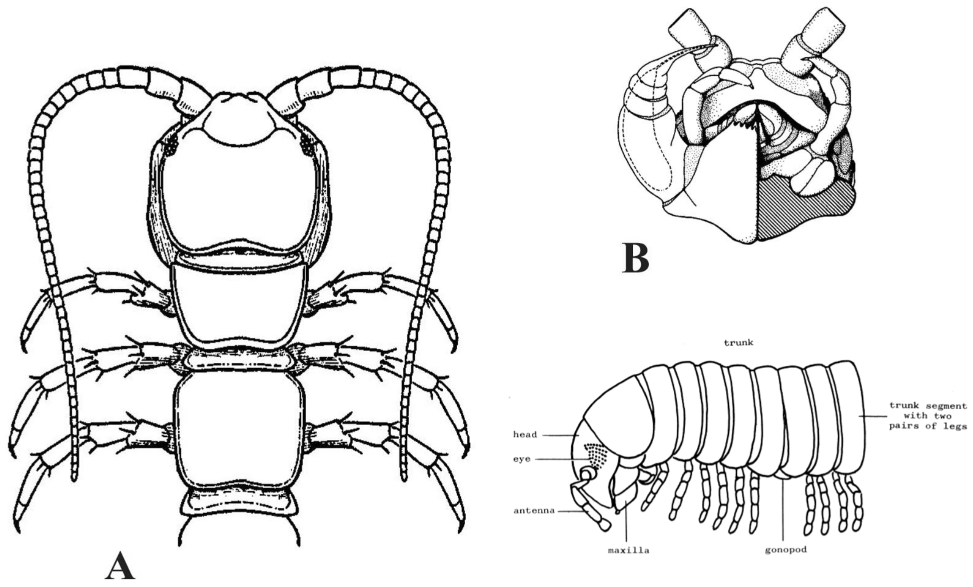

Figure 4: Head anatomy of a centipede and a millipede [Uni10],[WWE80, p61]

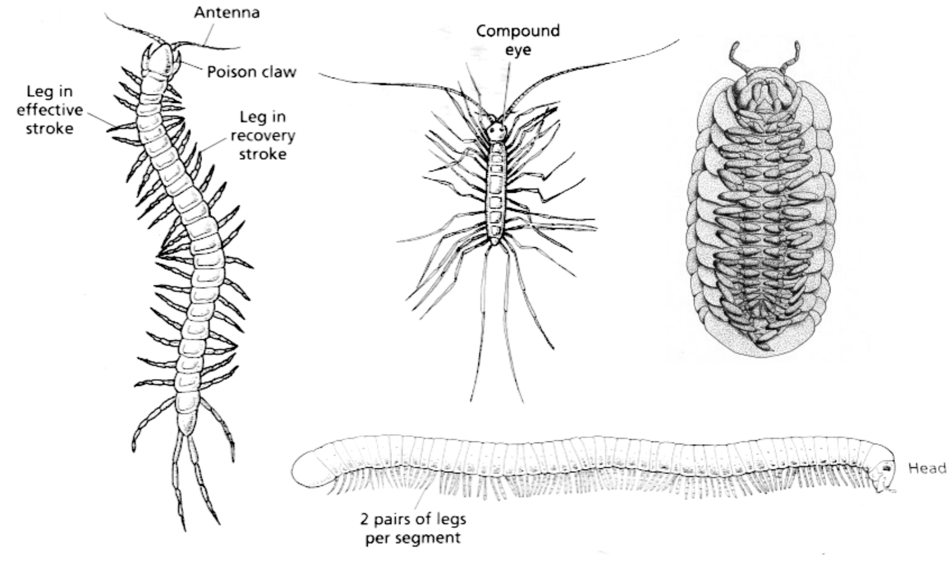

Figure 5: Examples for different centipedes and millipedes - **Top left:** Vietnamese Centipede - **Top center:** House centipede - **Top Right:** Pill millipede - **Bottom:** Giant millipede [MO01, p.218-219],[Jan05]

bodies are round, hard and the legs are placed at the lower side. Millipedes often coil up their body into spirals and some even into pills. In order to have such a flexible back but being at the same time well protected, the cuticles of each segment overlap each other when the back is straightened as it can be seen in figure 2.

The legs of myriapoda are constructed in a simple way in comparison to the legs of other insects. The joint between the coxa [4] and the body allows the back- and forward motion of the leg, the joints around the trochanter act like a knee joint allowing the leg to move up and down, but with a higher degree of freedom for motion in the up direction. The joints from the femur to the pretarsus are similar to the joints around the trochanter, but have a higher degree of freedom in the down direction [Cla73, p.40].

3 Locomotion of Millipedes and Centipedes

The motion of the centipedes and millipedes is very little discussed in literature in general and mostly with a main focus on legs. Therefore, most of the description of the motion in this thesis is based on personal observation of real millipedes in the zoo or from video examples [Nat07], [mem07]. For a better

[4] compare with figure 6

overview, the analysis of independently controlled parts of the body is covered separately.

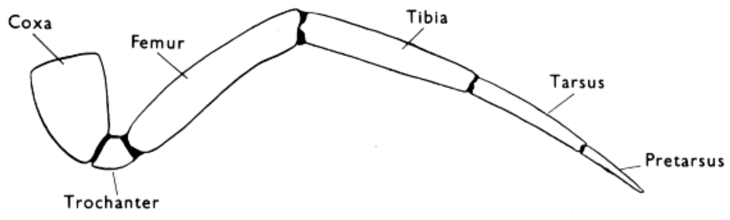

Figure 6: Leg anatomy of a millipede [Cla73, p.41]

Antennas

As mentioned before, millipedes and centipedes have antennas on the head of their body and centipedes tend to use their last pair of legs as well as antennas. But in opposite to many other insects, the antennas of myriapoda are nearly as thick as legs, and the myriapodas rely heavily on them to explore the environment. Therefore, the dominant motion of antennas is mainly controlled by the creature. Secondary motion can only be observed at centipedes because they move much faster than millipedes, and as they rarely actively control their back antennas, the secondary motion is best observed at that body part.

Mandible

The mandible of Myriapoda is unfortunately well hidden beneath the head and is very small, even for the big tropical species. For the human eye it is only well visible with a microscope. But as the mandible is mainly used while eating, it is nearly impossible to observe the mandible, because it might be covered with food or hidden behind the head or the maxillae.

Maxillae

Similar to the mandible, maxillae are mainly used to hold the food while eating or to hunt. Finding video material or observing such movements in the zoo became a big challenge. Finally, the movement has been derived from studying the anatomy in combination with some videos found on the internet. The anatomy of the maxillae are similar to the anatomy of legs, with the only distinction that they the maxillae are used as tongs and the motion is comparable to a human hand where a thumb and a finger would grab for an object. This results in very flexible tongs for Myriapoda.

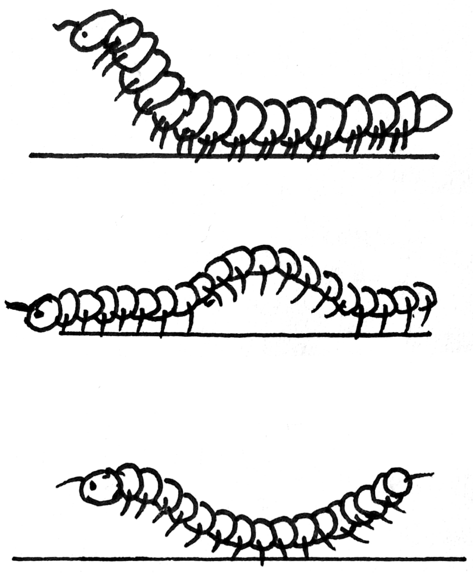

Figure 7: Different possible spine deformations of a millipede/centipede. **Left:** Lifting front body - **Center:** Lifting of the body center - **Right:** Lifting of the end segments

Spine

The movements of the spine in combination with the legs are the key for correct animation of centipedes and millipedes. Their spine joints are flexible in each direction and each segment can be controlled separately, but as each spine segment is very short compared to the total size of the creature, the spine appears more like a chain. During the observation of millipedes, the following typical behaviour has been discovered. First of all, there is no significant up and down movement of the spine if the millipede is walking. In opposite to mammals that tend to move their spine in vertical direction, millipedes have more legs, which provide them with more stability while walking. As they move for- or backward, the majority of the legs stays at the ground, and therefore, any up or down movement of the spine would cause that more legs are unnecessarily moved and energy would be wasted. But there is another interesting fact when millipedes or centipedes are moving forward. Millipedes normally dig in soil where they create small tunnels of the diameter of their body, and therefore, each segment of the spine follows the motion path of the head. This has the effect that millipedes always seem to walk along an invisible line. The slow walk of centipedes is similar. Their body segments also follow the same motion

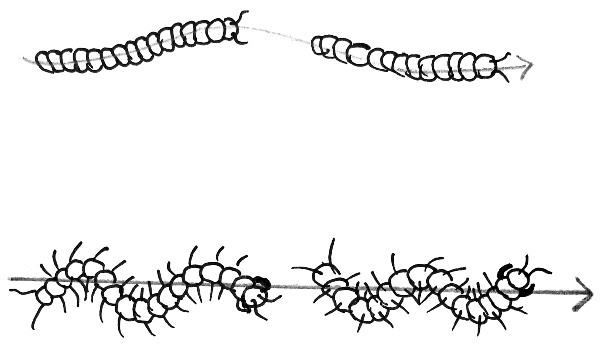
Figure 8: Motion path comparison between a walking millipede and a running centipede

path, but when centipedes are running or swimming, they tend to move like snakes. In such a case, their spine looks more like a sinus function and less like a curve following a path. Besides walking, millipedes explore their environment moving the spine to look around. They lift their front body and move it side by side. A similar motion happens if a millipede wants to change its direction. In this case, it lifts the front body only as much as needed to lift the front legs from the ground and moves the spine sidewards. Finally, it continues to walk into the new direction. Besides its front body it can also lift central segments of the spine or the back. This behaviour can be seen in the video of National Geographic [Nat07] or in figure 8, where the centipede lifts the central segments of its body to carry babies or where it lifts the back during the fight against the mouse.

Legs

The movement of the legs is actually the most fascinating part of myriapoda, because the legs look like a wave while walking[5]. This wave is the result of each pair of legs being in phase, but slightly out of phase to the previous pair. To avoid stumbling about its legs, each pair is never moved further than the previous one. The motion of each leg can be subdivided into two parts. During the transfer stage the leg is lifted and moved forward. In opposite to that, the leg stays attached to the ground and pushes the body forward during the propulsive stage. There are different theories that try to simulate the motion of each leg. The first was mentioned by Sinclair, who reports that the legs move in a set of five [Sin85]. Another model is mentioned in [Sat04]. It is based

[5]see therefore the enclosed video

on the assumption that the motion of each leg can be described with cycloid functions. But the authors of the same paper criticises that this assumption is wrong and suggest their model based on a circle of reference. Similar to the previous model, Sathirapongsasuti describes the movement of a millipede with the help of a circle, but takes only one part of the circle to simulate the motion of the leg. The problem with all approaches is that the described solution is only valid for a constant forward speed and only if the millipede is moving straight forward, which makes it suitable for computer graphics only to a very limited point. Although the “circle of reference” seems to be more suitable for the simulation of a leg, it is at the same time more complex and relies for a changing of speed on heavier computation. In opposite to that, the method of cycloid functions simulates as well an arc as steps for the millipede, but is faster to compute and experiment with it is easier. Therefore, it seems to be the method of choice to implement. It is also to doubt whether an inexperienced viewer would notice the difference between the two approaches.

4 Millipedes and centipedes in computer graphics

Insects are widely present as virtual characters in film and television, in many like “A Bug’s Life”, “Ants”, “Joe’s Apartment” they even became main characters. This fact motivated the research about centipedes and millipedes shown in recent movies, hoping to find existing rigging examples for these creatures. Unfortunately, only four film examples have been found, but only one can be considered as realistic. The first example with millipedes is “A Bug’s Life”, where cartoon millipedes are shown towards the end of the film. As these characters are only shown for a couple of seconds, they are designed in a very simple manner and no information has been found about their design or set-up. Besides the two millipedes, there are two pill bugs in the film and the bonus-DVD contains some sketches of their bodies, which may serve as a good modelling reference for spine segments and joints and gave at the beginning of this project some helpful information about the general design of insects and how they can be stylised for an animation film.

Another example for centipedes has been found in “King Kong” by Peter Jackson, where Weta Digital created a giant centipede that attacks Ann Darrow, who is played by Naomi Watts. This example is the best example for a realistically computer animated centipede. The body and the motion are based on real centipedes and only the shape of the mouth seems to be designed by an artist. Other examples of Myriapoda in film are the Arthropleura in

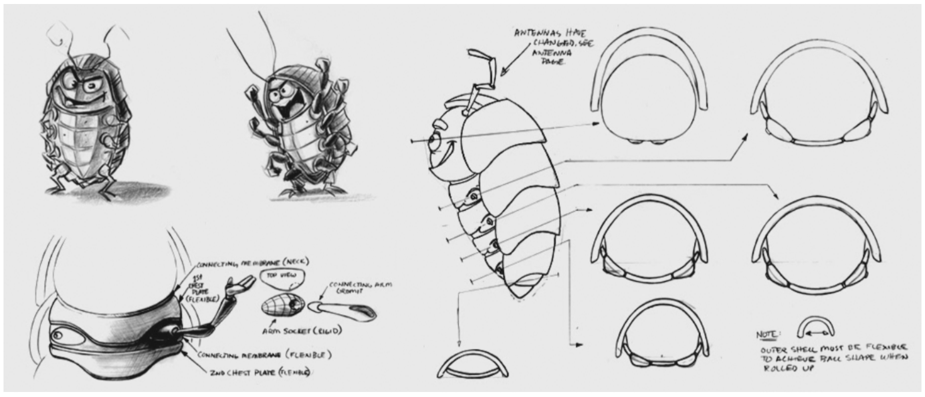
CONNECTING MEMBRANE (NECK)
1st CHEST PLATE (FLEXIBLE)
TOP VIEW
CONNECTING ARM GROMMET
ARM SOCKET (RIGID)
CONNECTING MEMBRANE (FLEXIBLE)
2ND CHEST PLATE (FLEXIBLE)
ANTENNAS HAVE CHANGED, SEE ANTENNA PAGE
NOTE:
OUTER SHELL MUST BE FLEXIBLE TO ACHIEVE BALL SHAPE WHEN ROLLED UP

Figure 9: Sketches of a pill bug from Pixar's "A Bug's Life"[Les99]

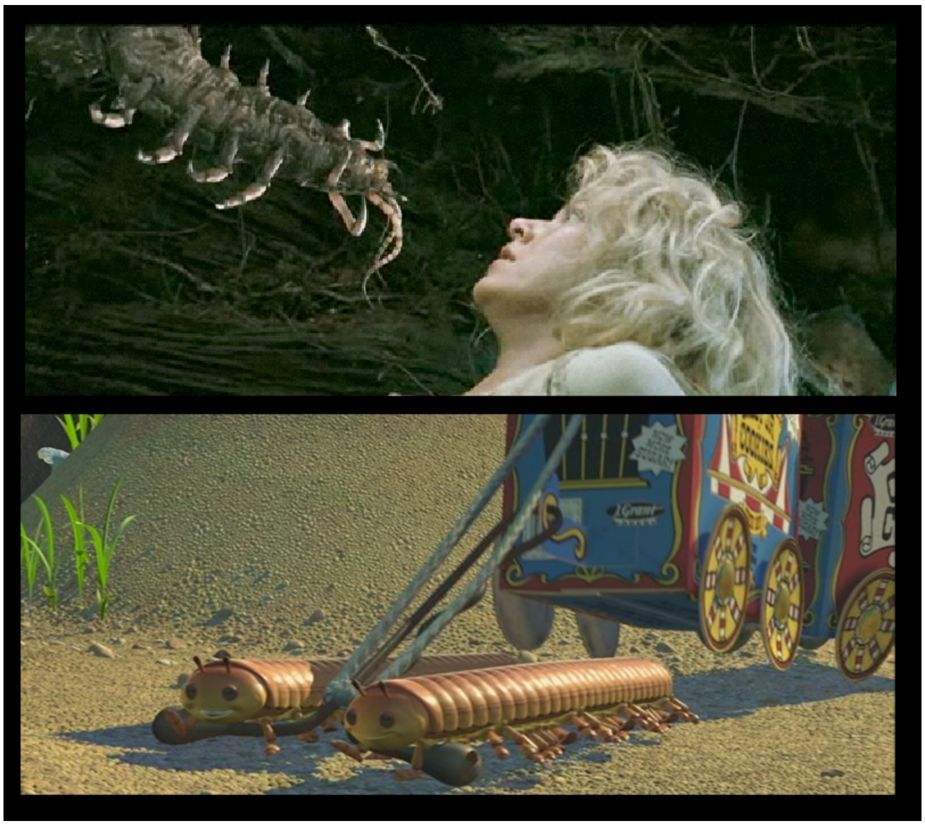

Figure 10: **Top:** Screenshot from King Kong with a centipede [Jac05], **Bottom:** Screenshot of a millipede from "A Bug's Life"[Les99]

“Prehistoric Park – The Bug House” [BBC06] or an unreal giant centipede in “Primeval-Episode 1.2” [Hai07]. But the last examples are less suitable as reference, because the creatures are only shown for seconds and interrupted by many cuts.

Unfortunately, no description about the rigging of any of these creatures has been found, but instead two other millipedes’ rigs. The millipede rig of Tapani [Tap08a] is completely created as an ICE system with dynamics in Softimage|XSI, but as he indicates in an internet forum, the rig is not suitable for production and might be “hacky” [Tap08b]. The main idea of this rig was to experiment with ICE, and therefore, it is understandable that it only provides little control for the animator. In his solutions the creature walks along a path and legs are moved automatically with the help of dynamics. Another solution to rig a millipede is given by Nelson [Nel08]. Although the description is very short and the videos are not displayed correctly in the browser, it seems that he implemented a basic rig with the help of expressions that time-delay the movement of the first segment.

5 Implementation

5.1 Software and Scripting languages

Some attention should be paid to the software system and the scripting languages used for this project, before a detailed description of the rigging system is given. For this project, Autodesk Maya has been chosen because of personal preference and its popularity within the 3D-industry. But many other software packages like Softimage|XSI, Houdini or Blender are suitable as well for the implementation of this rigging system. In fact, any software that fulfils the following requirements can be considered as suitable:

- is extensible by a scripting or programming language to create rigs procedurally
- supports custom user interfaces
- provides expressions or other similar tools to create custom dependencies or animation
- provides inverse kinematics comparable to the “Rotation Plane Solver” and the “Spline Solver” in Maya
- offers the possibility to create a path animation for a curve that will deform the curve according to the path

- provides a solution to measure the arc length of the curve and modify the speed of the path animation

Of course it is also possible to create the described rigging system in a package that only provides a subset of the named features. But in such a case more effort is needed because the missing feature has to be implemented for the software as well.

The built-in scripting language in Maya is Mel, but since version 8 of Maya, Python can be used as well for scripting. Unfortunately, the current Python implementation in Maya does not take advantage of the object oriented features of Python, because all Python functions from the "cmds" module are the same as the already existing MEL functions. A legitimated excuse for this fact could be that the transition from MEL to Python should be as simple as possible, but there is another shortcoming of Maya according to Python. Expressions in Maya still have their own scripting language which is very close to MEL [Gou03], and although MEL functions can be called without problems in an expression, Python functions can only be called with the MEL function python that calls a Python function that is passed as a string [Aut09].

```
//Example code for python in Maya expressions
float $var1=2;
float $var2=3;
float $ret=`python("module.add("+(string)$var1+","+(string)$var2+")")`;
```

Nevertheless, Python has been chosen as the scripting language for this project because many other packages like Houdini, Blender and Softimage|XSI support Python. Therefore, the final rigging system can be easier transferred to other packages. If it is possible to create the same rig in different software packages, it becomes also easier to transfer the animation between the two packages, and especially mixed pipelines will benefit from this feature. Another advantage of Python is the great number of available libraries that extend the core language for example with vector and matrix mathematics. Comparing the advantages and disadvantages between MEL and Python for this project it becomes clear, that although with MEL the interaction between expressions and scripting might become easier, it is limited to the use within Maya.

Other general advantages of Python in Maya that do not have any significant importance to this project, are the access to the C++ libraries and the use of PyMel. PyMel is an open-source project initiated by the studios Luma Pictures, Attitude Studio and ImageMovers Digital [Lum10a] to reorganise the Maya Python module with the aim "to operate in a more succinct and intuitive way" [Lum10b]. The only drawback of PyMel is that it is a separate package that is not shipped with Maya and therefore needs to be installed additionally by the user. Furthermore, PyMel does not provide any significant improvements for the development of the rigging system. That is why the default Python module for Maya has been chosen to implement the rigging tool.

5.2 Overview of the rigging system

As already mentioned before, the basic idea of this project was to create a flexible rigging tool that can be used for any centipede or millipede. To achieve this requirement, the rigging tool has either to adapt to the user's need or to the anatomical differences between centipedes and millipedes. These differences are:

- variable number of spine segments
- different number of legs per segment
- additional pair of maxillae (poison claws) of the centipede
- additional legs at the end segments used as antennas for centipedes

But it should be also considered that not all centipedes or millipedes might be designed anatomically correct. The number of leg joints may therefore be reduced to two, or fantasy millipedes might be created that mix the anatomical characteristics of centipedes and millipedes. Additionally, the animator should be supported with controls to simplify the animation of hundreds of legs or of all spine segments. It would be even better if procedural walking could be generated and the animator only had to define the walking path.

Obviously, such a rigging system is very sophisticated, especially if all possible exceptions in terms of anatomy are implemented from the beginning. Furthermore, this rigging tool seems to be very unique because no documentation has been found of comparable examples. That is why it seems to be better to solve all the technical problems like the procedural walking at the beginning and then add all anatomical exceptions to the rig later.

5.3 Naming conventions

If a rig is passed to the animators, it has to work efficiently, to be free of bugs and to be easy to use. One key to usability are naming conventions like "'L"' or "'R"' to indicate the side, and “Leg”, “Controller” or “Millipede” to indicate the part of the body, its function or the name of the creature. Mostly, a developer will implement a rigging system with default names, but these might become problematic when animators do not understand the names because of cryptic abbreviations like “LGrpCtrAnt” representing “Left_Group_Controller_Antenna”, or if they have three interacting creatures and all of them have the same names for the legs. Other reasons to change the names might be that users are not English native speakers and therefore “I” and “D” might be more appropriate to indicate the side than “L” or “R”.[6] To increase usability, the rigging tool lets the user decide how the created creature should be named, whether he wants to use abbreviations to keep the names short or full names to keep it more comprehensible, or whether he wants to name it in English or any other language. In order to guarantee that all the names in the rigging system are unique and therefore can be selected by their names, the following naming convention

[6]“I” and “D” are abbreviations of the Spanish words “izquierda” and “derecha” and their translation means “left” and “right”.

has been used. The names for all created nodes start with their object type (e.g. Controller, Group), followed by the body part (e.g. Head, Leg), which is sometimes supplemented by the segment number. Finally, the name of the creature itself is added.

5.4 Positioning the joints

After the definition of the names, the user has to define the position of the joints. As many other creatures, millipedes and centipedes are symmetrical, moreover, they also have a repeatable pattern at their spine and legs. Obviously, it might take a while until all the joints of a 100-leg centipede are positioned correctly. To simplify this procedure, the locators that will later be replaced by joints, are created step by step. In the first step only the head, the root and the end segments are created. The user should now position the locators according to his individual creature. The position of the head and the end segment is then used to distribute equally all the spine segments. Of course it cannot be assumed that the computed position of all spine joints is right, and therefore, the user still has the possibility to adapt the position of the locators to his individual character. A similar algorithm is also applied to the legs of the creature. According to the side specified by the user, the first and the last leg is created. After positioning the first and the last leg, all the other legs are created with an equal distance between each other. The derivation of the position for the in-between legs and spine segments guarantees at least that the position of the joints is close to the real position of the character. Although this solution is not ideal, it is definitely better than guessing the position of hundreds of legs.

Furthermore, the user has to define the position of the joints for the antenna and the maxilla(e). If this is done, one side of the creature is finished and all the locators can be mirrored to the other side in order to take advantage of the creature's symmetry. Of course, the names of the mirrored locators have to be modified accordingly. Even at this late stage the user still has the chance to modify the position of every locator, and as there is no hierarchical relationship between the locators, moving one locator in space does not affect the position of any other locator. This was the main reason to define all joint positions first with locators instead of using joints that are mostly created in hierarchical structures.

```
Pseudo-code for locator creation:

create root, head and end locator at default position
-----------
 wait
-----------
user defined position of the head and end locator
as well as the number of spine segments
compute the distance between head and end locator
compute equal distance between every spine segment
for all spine segments
```

```
create spine segment and place at computed position
-----------
 wait
-----------
user input for the number of leg segments and the side
create first and last leg at first and last spine
wait
user defined position of all joints for the first and
the last leg as well as the number of legs
for each segment
get position of the segment of the first and last leg
compute distance
for all legs
place leg segments at computed position
-----------
 wait
-----------
user defined number of antenna segments
create default antenna
-----------
 wait
-----------
user defined number of maxilla segments
create default maxilla
-----------
 wait
-----------
user positioned all locators
create all leg, antenna and maxilla locators on the other side
```

5.5 Creating the rig

Until now the rigging system relied a lot on user input. For this last step, the user only has to select the features he wants for his rig and then the rig will be created. Nevertheless, this part is the core of the rigging system and became more and more complex during the progress of this project. After a while it became clear, that in order to achieve all the requirements, more than one rigging system would be needed. The following chapters describe how several independent rigs are designed, how they interact with each other and how they help the animator.

The Binding Rig

The binding rig is a basic chain of joints. It starts with the first spine joint which is the parent to the head, and the next spine joint which becomes the parent

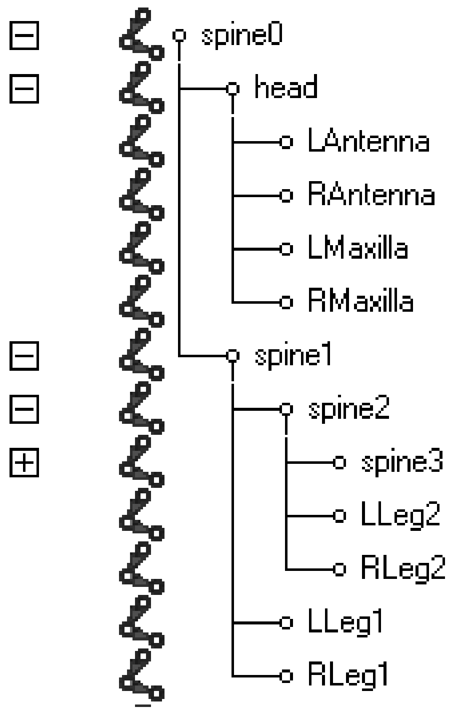

Figure 11: Simplified Maya tree to indicate the internal joint hierarchy

of the following spine joint. This results in a long chain of bones for the spine. Furthermore, every spine joint is as well parent for the according pair of legs. The maxilla and the antennas are parented to the head joint. The complete hierarchy is illustrated by a simple example in figure 11 As the name of the rig indicates, this rig is bound to the mesh of the character and all the animation is derived from other independent rigs. The separation between a binding rig and a rig to animate becomes very handy if more features are added to the rig during the production. New rigging features added during production many times require as well the replacement of the rig as the weights for skinning. In opposite to that, a binding rig can save time as the feature would be added to one of the independent rigs and only the resulting motion would be passed over to the binding rig [Rit06]. Furthermore, binding rigs can reduce complexity due to the possibility to blend between the motions of several independent rigs. This has been the main reason to implement the binding rig and give the animator the possibility to blend between the master and the automatic walking rig that will be discussed more in depth in the following sections.

The Master Rig

Originally, the master rig should be a simple rig that provides the animator with several functions to animate each joint individually. But after a while it became clear, that the demand to animate one leg separately many times will not be given. Instead, it is more likely that the animator will animate several legs at the same time, as the motion of one leg mostly affects as well the motion of the other legs of a centipede or a millipede. Therefore, it seemed to be more efficient to create a controller that enables to animate some master legs and then interpolate all the other legs automatically. To achieve that, another rig system has been created on top to the normal rig. The difference between the first and the second rig that will be called the master rig is only the number of legs and spine segments. The user has the possibility to define the number of master legs and spine segments and to control thereby the level of abstraction for the master rig. In the current implementation the position of the legs is linearly interpolated by expressions between the positions of the two closest master legs. As expressions are evaluated every time when the master rig has been moved, it is important to precompute as much as possible in order to save computation time. The number of legs is always constant for the rig, and that is why the weighting values for the legs are constant and can be precomputed during the rig creation.

In order to have more control over the motion of the master rig, each master leg has a separate bone chain for FK and IK animation[7], which is linearly blended with an IK-FK switch. Luckily, the default motion created by an FK controller is very close to the natural motion of a centipede or a millipede leg. The joint at the coxa[8] is moving back- and forward, the joints around the trochanter are bended during the up motion while the joints around the femur are bended during the down motion.

To add as well an IK-FK switch to the spine, a different approach has been chosen. First of all, an IK spline solver has been added to the spine chain of the master and the normal rig. Both chains share the same spline and therefore the same deformation. To control the shape of the spline, all knots are individually clustered[9] and IK controllers are created at the position of the cluster. Additionally, another joint chain is created that starts from the closest spine joint to the root position[10] and has a joint at all knot positions of the spline. If the joints are parented to the spline knots, the bone chain results as an FK controller for the spine. Again, an IK-FK switch controls whether the knot position of the spline

[7]FK and IK are common abbreviations for forward kinematics and inverse kinematics

[8]see figure 6

[9]Clusters are deformers which enables to group parts of an object like knots, vertices, etc. and to modify the individually [Aut09]

[10]see figure 12

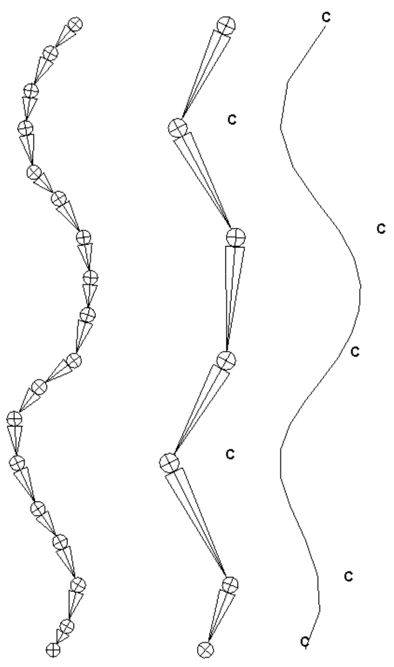

Figure 12: An Example for the construction of the spine; **Left:** Spine of the normal rig - **Center:** Spine of the master rig - **Right:** NURBS spline with clusters to control the position of the spline knots

is inherited from the IK controllers or from the FK joint chain. In opposite to the legs and the spine segments, the motion of any antenna or maxilla does not affect the motion of any other body part. Therefore, no simplification could be achieved for these parts of the body. To avoid confusion of the animator, it has been decided that all controllers should be moved to the master rig and the normal rig should only inherit the motion from the master rig. Therefore, all the joint chains for the antennas and the maxillae are copied to the master rig. Similar to the legs of the master rig, maxillae have as well a separate joint chain for the IK and FK animation which is blended by an IK-FK switch. The set-up of the antennas is comparable to the set-up of the spine. Each antenna has an IK spline solver with controllers that are parented to the knots of the spline.

5.6 Automatic Walking Rig

First attempt using custom Catmul-Rom splines

The automatic walking rig has been by far the most challenging part of the rigging system and it took two different approaches to create this rig. But before the animation of the legs could be driven by expressions, a solution needed to be found that transfers the applied path animation to the spine of a

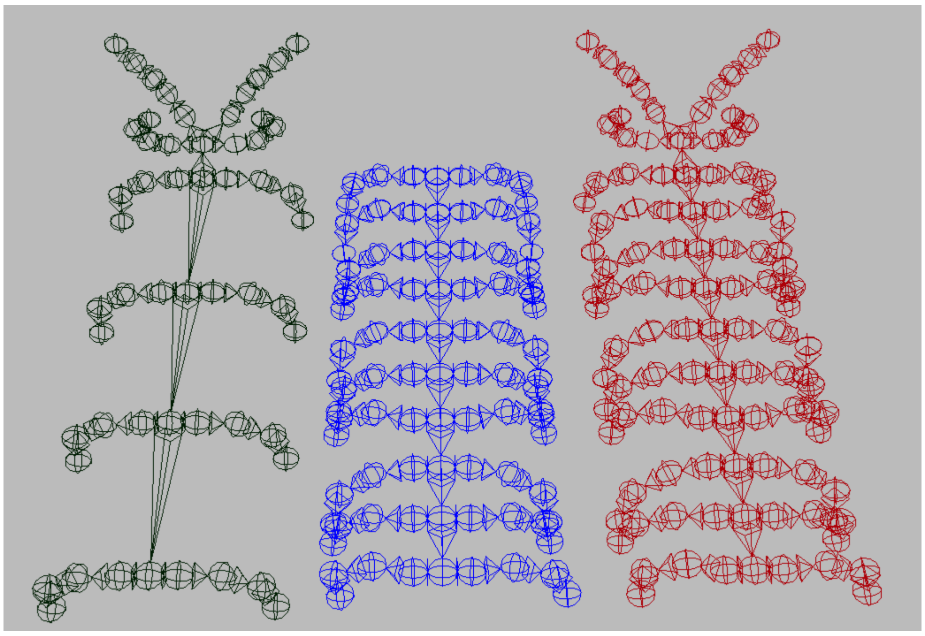

Figure 13: Examples for the joint structure; **Left:** Master rig - **Center:** Automatic walking rig - **Right:** Binding rig

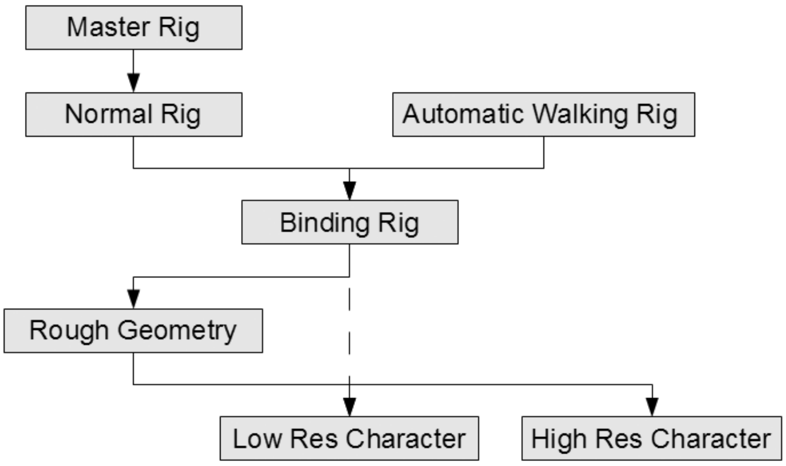

Figure 14: Hierarchy of the rigging system in combination with bounded geometry

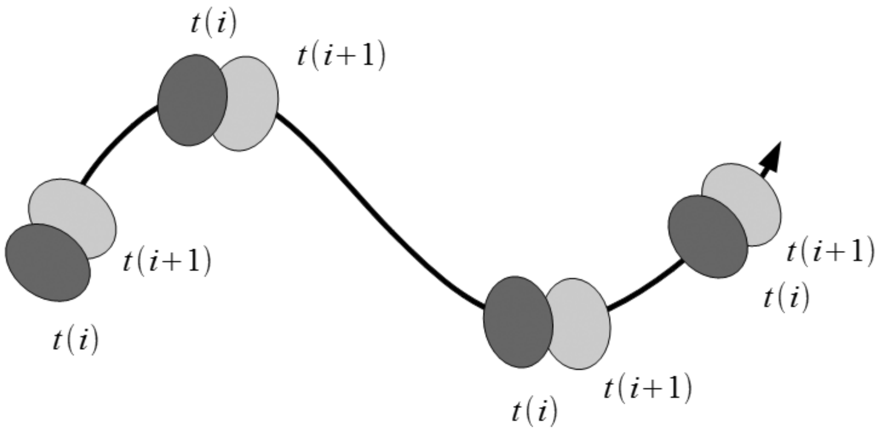

Figure 15: Illustration of the assumption that the motion path can be reconstructed from the spine segments - **t(i)** indicates the current frame and **slshape**t(i+1) the next frame

millipede or centipede. Unfortunately, first attempts to create a bone chain that deforms according to the path animation with built-in functions in Maya have failed, and therefore, the following concept has been developed. The basic idea of this concept is the assumption that the distance which each spine segment is covering from frame to frame, is very short in comparison to the total length of the creature. Furthermore, it is known that every segment of the creature always follows the motion path of its previous segment. Consequentially it means that if an interpolation spline is created through the joint of the spine and then each joint is moved further a certain distance along the spline, the resulting motion should be very close to the motion along a path. Of course it might be that the spine bones change their length during the animation, but this change of the length has been considered in theory as very small[11]. Furthermore, it has been the only chance to realise such a motion, because all attempts to create a similar motion with the basic tools of Maya failed up to this point. In the case that this approach could have been confirmed in practices, it also would have simplified path animation. In opposite to current approaches, the animator could only use key-frame animation instead of switching between path animation and key-frame animation.

As Maya only provides NURBS splines that belong to approximation splines, a custom class that provides the functionality of the Catmull-Rom splines has been developed in Python. Catmul-Rom splines have been chosen, because it is possible to construct it only from the given interpolation points and no

[11]see figure 15

additional information is needed, which has been considered as a big advantage to many other interpolation splines. The matrix representation of the Catmul-Rom spline is given in equation 1 [Par08]

$$P(u) = U^T M B$$
$$U^T = \begin{bmatrix} u^3 \; u^2 \; u \end{bmatrix}$$
$$M = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{1}$$
$$B = \begin{bmatrix} P_i \\ P_{i+1} \\ P_i^{‘} \\ P_{i+1}^{‘} \end{bmatrix}$$

In order to be able to compute P(u) as well between the first and the second interior point, the initial tangent has been formed by equation

$$P\_1 = P_1 - (P_2 - P_1) \tag{2}$$

and accordingly the end tangent

$$P_i + 2 = P_i - (P_i - 1 + P_i) \tag{3}$$

As the arc length often cannot be computed analytically for a curve, a forward differencing algorithm has been implemented as suggested in [Parent]. In order to compute the arc length from P(0) to P(0.23), this approach would first compute the position of several in-between points and then approximate the arc length by adding the distance between the points. Firstly, the distance is computed between $P(0)$ and $P(0.2)$

$$\begin{aligned} d(0.2) \quad &= P(0.05) - P(0.0) \\ &+P(0.1) - P(0.05) \\ &+P(0.15) - P(0.1) \\ &+P(0.2) - P(0.15) \end{aligned} \tag{4}$$

and then linearly interpolated with the distance between $P(0)$ and $P(0.25)$

After implementing the algorithms for Catmul-Rom splines and the arc length, first test have been done in Maya to see whether the theoretical assumptions could be confirmed. A chain of joints that represent the spine of a millipede has been created. The first spine joint has been animated by key-frame animation, while all the other joints should be controlled by an expression. This expression takes at first the current position of all joints and the previous position of the first joint. Then it creates a Catmull-Rom spline from all points and computes the distance that the first joint has moved between the last and the current frame. In the next step, every expression driven point is moved by the same distance along the Catmul-Rom spline.

Unfortunately, the theoretical assumptions for this approach could not be confirmed. The first test showed that the bone length is changing significantly after some frames, furthermore the frame rate dropped to less than real-time. As this approach seemed to be at that time the only way to implement an automatic walking rig, a lot of effort has been put to guarantee that all the spline functions are bug free. Although extensive testing has uncovered some computation errors, the significant length change of the bone could not be solved at this point. This drawback motivated again the research on the default functionalities of Maya for path animation and whether there is a solution to deform a bone chain according to a motion path. Luckily, the following solution has been discovered.

Second attempt with default path animation in Maya

The second attempt to create an automatically walking millipede is mainly based on default Maya functionalities that are extended at certain points by expressions. At first, an IK spline solver with a new curve is created for the spine of the millipede, then the curve is attached to a motion path. Until now the creature would only follow the path but not deform its spine. In order to achieve that, the "flow" function of Maya is used. This will create a lattice around the curve that will be deformed according to the shape of the motion path. Of course the number of divisions for the lattice should be at least the same as the number of spline knots that control the spine chain. With the applied flow function the deformation of the motion path is now transferred to the creature's spine.

In addition to the spine deformation, a procedural animation needs to be added to the legs. Although Sathirapongsasuti derives several equations for the leg motion of millipedes, they are unfortunately based on assumptions that are not fulfilled in the present rigging system. In his work the millipede is moving with a constant speed along a path, but as the animator controls the speed of the creature, this assumption is not given. Luckily, the starting and ending point of the animation is known and the distance along the path from the current position to the starting position can be computed for any frame. In terms of

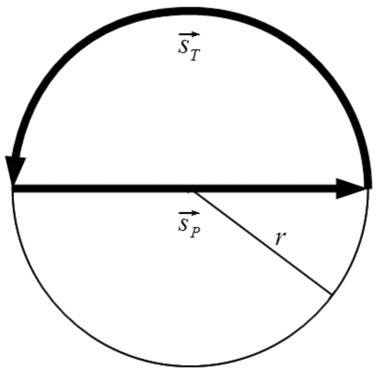

Figure 16: Simplified walking cycle of a leg

Maya this means to create an “arcLengthDimension” node for the spline that defines the motion path, and then connect the current position of the creature with the position of the arcLenghtDemension node. As the arcLenghtDemension node returns the length of the spline between the starting point and the current point, it is now possible to access the distance along the path.

In order to calculate the motion of the legs and to save computation time, a simplified concept of Sathirapongsasuti has been used. It is mainly base on the observations of Sathirapongsasuti but ignores that each leg moves along a circle of reference. It is assumed that the leg moves along a circle and can be divided in a propulsive and a transfer stage. During the transfer stage the leg is lifted and moved forwards. In opposite to that, the leg is moved backwards during the propulsive stage while the body is pushed forwards. Therefore, it can be assumed that the local transformation in x direction is equal during the propulsive and the transfer stage in terms of the length, but contrariwise in terms of the direction. The transformation in the y direction is during the propulsive stage 0 because the tip of the leg stays attached to the ground, and during the transfer stage it follows the circular motion.

With the reduction to a circular motion x and y can be computed as following:

$$x = -r \cdot \cos \frac{s - \phi \cdot d}{r} \qquad y = -r \cdot \sin \frac{s - \phi \cdot d}{r} \tag{5}$$

S is representing the moved distance from the starting point, d the distance of the leg from the first leg, ϕ the phase, r the radius of the circle and the minus is an offset of 180° that guarantees that the first pair of legs starts with the transfer stage. In order to give the animator more control over the walk, he can modify phi to change the phase between the legs. Furthermore, he has control over r that changes the step size, which should be modified when legs start to intersect each other. Additionally, a constant has been added to the computation of y in order to be able to adjust the step height. In the implementation of the rigging system in Maya these formulas have been used to manipulate the position of the IK handles of the legs.

6 Program structure

This project was not only challenging because of the complexity of the topic, it became even more challenging because only little experience existed before with the scripting languages Python or MEL and no experience about scripting for Maya. The consequence was that especially at the beginning most of the code has been developed just for special use cases. But in the course of time it became clear that parts of the code could be changed into general functions and called at different parts of the system. Therefore, the internal coding structure of the rigging tool has been restructured several times during the project and still needs some restructuring to increase productivity for further development. Although Python is an object oriented language, this feature has not been used for this project because only very little data is saved and most of the data is used at several parts of the program. For this specific project, it seemed that the advantage of a better structure created by an object oriented approach could also be achieved if all the functions were distributed to different Python modules. Using these modules as containers for specific functions is comparable to the use of objects that save no data at all. Finally, all modules have been bundled in one Python package that currently consists of the following modules:

__init__
The default module for each package that provides the information of the other modules.

MillipedeRig
The MillipedeRig contains all rig related functions. This can be general functions that only create a simple joint chain or very specific functions that create the master rig.

MillipedeLoc
The MillipedeLoc module contains all functions that create any locators that will later be replaced by joints.

UserInterface

This module contains all the functions needed to create the user interface. It also saves the information how Maya has to call functions of the other module of the rigging system in order to communicate between the rigging system and the interface.

RigData

This module saves all the naming conventions and provides a rich set of functions to get proper names of several parts of the rig.

Utility

This module extends some functionalities of Maya by combining several functions of Maya. For example, it provides functions that copy the position of one object to another.

CatmullRomSpline

The CatmullRomSpline module supports the creation of Catmul-Rom splines and provides functions to compute the arc length of the curve or to detect points along the spline that are positioned at a certain distance from another point of the spline. But as described earlier, this module could not been used for this rigging system.

Wire Controller

This module is a custom Python version of a more elaborated Mel script from [Sol03].

euclid

This module has been taken from [Hol10] and provides vector and matrices mathematics in Python.

7 Conclusion

This master project has been a big challenge because it combines many to the author up to this point unknown areas of expertise like the anatomy of centipedes and millipedes, Catmull-Rom splines, scripting for Maya and the scripting language Python. During this project, detailed knowledge has been obtained of the anatomy and locomotion of centipedes and millipedes, which is documented in this thesis. Although strong effort has been made to find detailed description of the locomotion of millipedes, only few sources have been found and it might be that the description of the interaction between the legs and the spine has not been documented before. Furthermore, individual solutions have been found for crucial rigging problems in the context of this project. For example, an automatic walking rig has been developed, whose motion is very close to the natural motion of millipedes and centipedes, or a special rig has been designed to simplify the animation of numerous legs. Unfortunately, time has been lost during the first unsuccessful implementation of the automatic walking rig. But this is comprehensible as this project could not rely on any other comparable work. At the same time, it has been a nice learning experience about the implementation details of a path animation system and interpolation splines. Although the current implementation of the rigging system does not support all anatomical characteristics of millipedes and centipedes, it is so far the most flexible rigging system known to the author, and the remaining options for the rig should be implemented within a short amount of time.

Like in the case of many other programming projects there is always something that could be improved. Possible improvements of the rigging system would be a user interface that is created with the rig and provides better control of all switches that either blend between different rigs or between IK and FK. Furthermore, feedback should be collected from animators, and if there are any mayor concerns according to the usability or the functionality of the rig, their suggestions should be implemented as well. Additionally, this rigging system could be extended with only little effort to a general rigging system for worms, snakes or many arthropods, like spiders, insects or shrimps. As the tool is designed to create several legs and spine segments procedurally, only some new rules according to the number of legs would be needed in order to create rigs for other creatures. For example, to create a worm rig, it would be sufficient to remove all extremities from the current system. In order to create a rig for shrimps or lobster, tongs would be added to the maxillae and the number of legs would be created independently from the number of spine segments. Overall, it seems that this project does not only provide an interesting solution to rig centipedes or millipedes, but as well many interesting approaches that could be transferred to rig other creatures.

References

[Autodesk, 2009] Autodesk (2009). Maya documentation.

[BBC, 2006] BBC (2006). Prehistoric park - bugs house. http://www.youtube.com/watch?v=Wl7xZrtxb_I&feature=related.

[Clarke, 1973] Clarke, K. U. (1973). *The biology of the Arthropoda.* Contemporary biology. Edward Arnold, London.

[Gould, 2003] Gould, D. A. D. (2003). *Complete Maya programming: An extensive guide to MEL and the C++ API.* Morgan Kaufmann series in computer graphics and geometric modeling. Academic Press, San Diego, CA.

[Haines, 2007] Haines, Tim, H. A. (2007). Primeval-episode1.2 (giant spiders and centipede). http://www.youtube.com/watch?v=MRomApaygfU.

[Holkner, 2010] Holkner, A. (2010). pyeuclid. http://code.google.com/p/pyeuclid/.

[Jackson, 2005] Jackson, P. (2005). King kong.

[Janssen, 2005] Janssen, Ralf, P. N.-M. D. W. G. (2005). A review of the correlation of tergites, sternites, and leg pairs in diplopods. http://www.frontiersinzoology.com/content/3/1/2.

[Lesseter, 1999] Lesseter, John, S. A. (1999). A bug's life: collector's edition.

[Luma Pictures, 2010a] Luma Pictures (2010a). Pymel. http://code.google.com/p/pymel/.

[Luma Pictures, 2010b] Luma Pictures (2010b). Pymel - documentation. http://pymel.googlecode.com/svn/docs/index.html.

[memutic, 2007] memutic (2007). Scolopendra, the dangerous centipede of thailand. http://www.youtube.com/watch?v=C8HSkAr7v5g.

[Moore and Overhill, 2001] Moore, J. and Overhill, R. (2001). *An introduction to the invertebrates.* Studies in biology. Cambridge University Press, Cambridge, New York.

[National Geographic, 2007] National Geographic (2007). Centipede vs. grasshopper mouse. http://www.youtube.com/watch?v=AcYW31N96b4.

[Nelson, 2008] Nelson, J. R. (2008). Millipede rig. http://www.jonathanrnelson.com/doodles.htm.

[Parent, 2008] Parent, R. (2008). *Computer animation: Algorithms and techniques*. The Morgan Kaufmann series in computer graphics. Elsevier Morgan Kaufmann, Amsterdam, 2. ed. edition.

[Ritchie, 2006] Ritchie, Kiaran, C. J.-B. K. (2006). *The art of rigging: Volume I ; a definitive guide to character technical direction with Alias Maya*. Toolkit.

[Sathirapongsasuti, 2004] Sathirapongsasuti, Jarupon, P. N.-W. P. (2004). Walking with a millipede: A mathematical explanation of millipede's walk. `http://files.thaiday.com/news/science/Walking_With_A_Millipede.pdf`.

[Sinclair, 1985] Sinclair, F. (1985). Myriapoda. In *The Cambridge natural history*, volume 5. Cambridge University Press, Cambridge.

[Solsona, 2003] Solsona, Javier, L. L. (2003). rig101 wire controllers. `http://student.vfs.com/~m07goosh/rigging101/freestuff.htm`.

[Tapani, 2008a] Tapani, A. (2008a). Milliped rig. `http://anttontapani.com/index.php?option=com_content&view=article&id=7&Itemid=8`.

[Tapani, 2008b] Tapani, A. (2008b). Post on millipede rig in area. `http://area.autodesk.com/forum/autodesk-softimage/ice---interactive-creative-environment/rigging-a-millipede/`.

[University of Aberdeen, 2010] University of Aberdeen (2010). Centipedes. `http://www.abdn.ac.uk/rhynie/centipede.htm`.

[Webb et al., 1980] Webb, J. E., Wallwork, J. A., and Elgood, J. H. (1980). *Guide to invertebrate animals*. Macmillan, London, 2. ed. edition.