Abstract

The project originally set out to produce an accurate simulation of kangaroo motion based on physics. Majority of computer animation of animals is produced by motion capture or manually created key frames. A model based solely on physics would be completely independent and would be able to adapt within its virtual environment without replying on user intervention.

The solution developed originally focused on developing a specific physics engine for modelling the kangaroo. However it soon became apparent that the problem was a lot more complicated than originally thought. As a result further research was conducted into existing physics based systems. It was at this point the Open Dynamics Engine (ODE) physics libraries became a vital part to ensure the success of the project.

By constructing a physics based model of a kangaroo, real life situations can be simulated. These simulations can then be modified by the end user to simulate environmental changes, as well as structural changes in the kangaroo, such as injury.

The kangaroo has a very unique method of travel, and little is understood as to why it manages to sustain such a high efficiency of locomotion. The literature survey discusses some of the observations made regarding the kangaroo motion, however to this date many of them still are unexplained.

Although the simulations achieved are far from satisfying the original objective to produce a hopping kangaroo, they did produce some useful results. The last simulation completed consisted of a fully constructed model of a simplified kangaroo skeleton, which slowly dragged itself across the ground of a virtual environment. This model is completely physics driven, and it is possible to find the forces exerted on each joint at any time in the simulation.

The most recent simulation still has some implementation problems which will need to be resolved in order for future progress to be made. However it is clear that once these problems are addressed, it will be possible to continue development to eventually satisfy the original brief.

Acknowledgements

The following people have been most helpful thought the development of the project:

Russell Smith – Author of the ODE libraries.

Nate Williams – Developer of robotics simulation packages using ODE.

Megan Fox – ODE mailing list, and fellow computer programmer.

Rudy J Lapeer, for his assistance and guidance throughout the project.

I would also like to thank the thousands of programmers whose postings on internet and in mailing lists that have been of great help.

Contents

1 Introduction

There are currently three main techniques used to computerise animal motion, the most popular of which is using key frames [1] [2]. Motion capture and dynamic modelling are less frequently used, although physics models tend to be more accurate, and more adaptable.

1.1 Key Framing

A key frame is a single still image within a scene setup that represents a significant point in the animation. An animation sequence will contain many frames only some of which will be key frames. The non-key frames are called tweened frames, which in computer animation are generated automatically through the process of inbetweening. The tweened frames are used to produce the animation that links one key frame to another, resulting in a smooth transition between frames [3].

Key framing is an animator's first choice because it is the only method that allows almost full control over what is being produced. This technique however is very time consuming as all the key frames need to be manually created, and often the tweened frames will need to be tweaked.

1.2 Motion Capture

Motion capture is a technique of digitally recording the movements of sensors placed upon a subject. These sensors are tracked in three dimensional space and their coordinates are streamed into a computer from which an animation can be created [4]. This technique is the most accurate way of recording animal motion, as the sensors are actually attached to the subject.

The disadvantage with this technique is that all the animation has to be captured initially, and then processed to produce the animation. As a result, the animation obtained is limited to what was originally captured. It is very difficult to modify the animation post production, and almost impossible to create new animation without using key framing. Despite these drawbacks, motion capture is still used frequently, and most often found in computer simulation games, in particular simulated sports such as football and rugby.

1.3 Physics based animation

Using kinematics and dynamics allows for a completely dynamic model to be produced. By calculating the desired angles for motion using inverse kinematics, the forces required to obtain and sustain these angles can then be calculated using inverse dynamics.

Unlike key framing and motion capture, physics based animation is dynamically calculated and produced on the fly. Although the models are animated on the fly they are complicated to create and often will require an embedded control system to maintain balance and deal with environmental changes.

1.4 The Project's Importance

Of the three previous methods, physics based models are the most interesting as they attempt to model the animal based on real forces and energy. These models also provide the greatest insight into how the animal moves and what would happen if the forces or energy were to change as a result of an injury for example.

A complete physics based model would allow an animal to travel through a virtual environment without the need for user input. The final model would need a control system in place in order to help the model maintain its motion if the current environment was to change, for example, a change in gradient.

After much thought it was decided that the animal to be modelled would be a kangaroo. This was because of a number of reasons. One of which is because the kangaroo moves in an almost symmetrical way which would require less geometric modelling. Additionally there has also been some work done into kangaroo locomotion because of its unexplained high level of locomotive efficiency. Lastly the project will continue the work carried out by Cameron Luke Greatorex, who successfully managed to produce an animated model of a kangaroo's motion cycle whilst hopping using 3D Studio Max and OpenGL.

1.5 Project Application

There are two main applications for physics based models: the gaming industry and the medical field. Currently, game developers are looking for ways to make their games more realistic, and provide more control of the characters within the game to the consumer. Nearly all the simulation games currently on the market rely on motion capture, as tricky manoeuvres can be hard to model or key frame. Based on current research modelling, a character in a computer game such as a footballer using physics would be a very complicated task and would not yield results any better than motion capture. However characters that required less complicated manoeuvres, such as running and jumping within a virtual environment could be implemented and would result in a more realistic representation on screen.

The medical application is considered more important than the entertainment application as it can provide useful information to people who work within the medical field. From a complete dynamic model of a kangaroo it will be possible to find out the forces and energy used at any point during its motion. This would be a significant step towards understanding how a kangaroo moves and how it can maintain its consistent speed at such a highly efficient rate. As well as understanding how it currently moves, it would also be possible to model what would happen if the energy and forces were different. From this it would be possible to simulate a wounded kangaroo and see how this would affect its motion.

Although this project is focusing on a kangaroo, the potential of creating a physics based model to any animal can be clearly seen. Accurately modelling humans will have significant benefits particularly in the medical field, where it will be possible to predict the effects of injuries and operations. Human modelling can be also used to study gait analysis and movement patterns of disabled people. This would provide a better understanding of the condition and what effect the condition would have on a person.

1.6 Project Objective

The key objective for this project is to model kangaroo motion based on force, energy and momentum rather than relying on techniques such as motion capture or manual key framing. This will produce a more accurate representation of the kangaroo's movement, and allow it to be dynamically animated rather than using a pre-defined motion. By using inverse kinematics the position of the kangaroo's joints can be calculated, and through the use of inverse dynamics the forces within the muscles can be calculated.

The model is required to travel between a given start and end point in a way accurate to that of a kangaroo. It should be possible at any time to view the forces exerted by each muscle, and adjust if required to alter the motion of the kangaroo.

1.7 Objective requirements

The objective itself has many requirements each of which are vital to make the project successful. It is important to address these requirements before the implementation stages; otherwise they could be overlooked and contradict the objective of the project.

The model of the kangaroo will need to be completely independent and able to maintain itself in a virtual environment (VE). Without this condition the model would depend on user input to control limbs, forces between those limbs and motion cycles.

There, as with all computer simulations will be small computational inaccuracies which can affect the realism of the model. These inaccuracies cannot be avoided, however compensation methods can be introduced to make the model more stable and stop the errors increasing.

The simulation is required to run in real time, and provide interactivity between the model, the VE and possibly the user. As a result there has to be a trade off between accuracy and performance. In this instance accuracy is very important; therefore the model will be rendered in a very simplistic form to reduce computational time.

The motion of the model is to be produced through the use of inverse kinematics and dynamics rather than preset motion sequences. This will enable an accurate representation to be constructed using energy and forces exerted by the kangaroo. From this it will be possible to develop a deeper understanding of how the kangaroo achieves its motion.

1.8 Project Outline

The report is broken down into key sections in order to make it more manageable and easier to follow. The remainder of the project will follow the outline detailed below.

Section 2 Current Research into Kangaroo Motion

This section draws upon existing research in the area encompassing kangaroo motion. Although little work on producing dynamic models has been done, there is a great deal of research into the locomotion of kangaroos, and how they manage to maintain such efficient means of travel.

Three research papers regarding the locomotion of red kangaroos are discussed in this section and their relevance to the project is commented on. These papers provide interesting observations that can be incorporated into the physics based simulation of the kangaroo.

Another paper in this section discusses how physics based models of a biped, a quadruped and a kangaroo were constructed. These models were then controlled by specifically written algorithms, which controlled the forces exerted on each joint.

Finally the physics library ODE is discussed, where its potential use for the project is identified. Although ODE is still in development stages it has been proven to be accurate and stable, for physics modelling.

Section 3 Project components

The project would be too hard to manage should it be attempted in one go. Instead the project has been broken down into five major tasks. Each one is designed to build upon the previous and gradually works towards a final solution. The final step of the project is to have a fully functional physics based kangaroo model, as described in the project objective (section 1.6).

Section 4 Design and Implementation of solutions

This section discusses how each task was designed and implemented. Screenshots of the simulations are also included and discussed, indicating their successful and unsuccessful aspects. As with most implementations, unexpected errors are observed, however during this stage these are noted and overcome during the next implementation.

Section 5 Testing

As well as the brief testing discussed in section 4, more detailed testing was carried out in this section of the report. After each simulation was run, mathematical comparisons where applicable were compared to that of the simulations results. For the simulations implemented in ODE it was not possible to produce mathematical comparisons at this stage in development. However it was possible to check for any simulation errors that could be visually observed. As found in the last simulation completed there were quite a few, and these will need to be addressed should the project be continued at a later date.

Section 6 Evaluation and Conclusion

This section evaluates the overall success of the project taking into account the original objectives and whether or not they were adhered to. The possibility of future work is also discussed focusing on solving the problems discovered form the testing part of the project.

2 Current Research into Kangaroo Motion

Although there has been no official release of a working system that models animal dynamics there has been a vast amount of research into the area. As discussed earlier there are already working models based on motion capture and key frames but these do not truly reflect the forces and energy involved in producing kangaroo motion. Physics based models are much harder to find, and are often unstable, and far from producing visually satisfying results.

2.1 Energetics and Biomechanics of Locomotion by Red Kangaroos.

The paper "Energetics and biomechanics of locomotion by red kangaroos" by Rodger Kram and Terence J. Dawson focuses more on the unsolved questions regarding the Red kangaroo's highly efficient locomotion [12]. Although this paper does not provide an explanation for the high efficiency it does successfully dismiss a number of theories surrounding the subject. Kram and Dawson conducted a series of experiments with red kangaroos, recording oxygen consumption, limb posture and estimated tendon stresses during hopping.

The key observation made during the experimentation was that for horizontal motion the rate of oxygen consumption did not increase with an increase in speed. This confirmed the original tests carried out by Dawson and Taylor [5] in 1973 suggesting that kangaroo oxygen consumption was linear with varying speed. This is shown in figure 2.1a where the dotted line represents Dawson and Taylor's original recordings and the open circles indicate Kram and Dawson's recent findings. With a line of best fit added, the oxygen consumption can be observed to only marginally increase with speed.

What was clearly noticed however was the variation of oxygen consumption with a varying gradient. This indicates a direct relation between oxygen consumption and speed on uneven terrain. This suggests that more work is required to overcome the increase in gradient, and the locomotive efficiency deteriorates as a clear correlation between speed and oxygen consumption can be observed. This is also indicated on figure 2.1a.

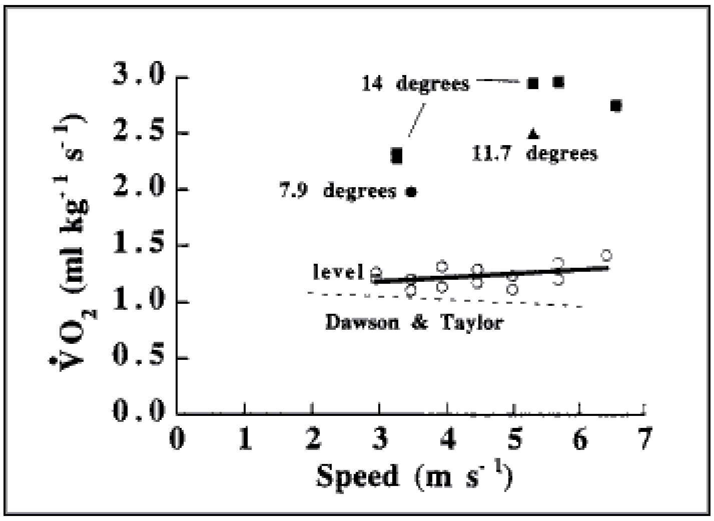


Figure 2.1a: During level hopping, the rate of oxygen consumption is essentially constant across speed, indicated by the open circles. With an increase in gradient the rate of oxygen consumption increases with an increase in speed, indicated by the square markers. The dotted line represents Dawson and Taylor's original findings in 1973 [5].

A general rule of animal locomotion is that animals tend to prefer to travel at the rate that uses the minimal energy consumption per unit distance. This rule has been applied to many animals by Hoyt and Taylor [6] and is proven to be accurate. It has been observed however that kangaroos do not follow this rule and tend to prefer a speed within the region of 3.9ms-1 to 6ms-1 despite being capable of maintaining speeds of up to 14ms-1. Since it has been proven that the oxygen consumption does not increase with speed, the optimal speed should simply be the maximum.

It would appear that the main reason for the kangaroo not travelling at its maximum speed is related to self preservation. While hopping, the kangaroo exerts a high level of stress on the ankle extensor tendon, very close to the ultimate strength of a mammalian tendon which is 100MPa. Whilst hopping at only 3.9ms^{-1} the peak tendon stress was calculated to be 45.4MPa which is approximately 45% of the ultimate strength. When more results were taken at increasing speeds it was noticed that the tendon stress increased as speed increased. At around 6ms^{-1} to 8ms^{-1} the tendon stress was dangerously close to the ultimate strength of a tendon. This would most definitely explain why kangaroos prefer slower running speeds, as whilst maintaining high speeds they run the risk of damaging the tendon.

It was originally believed that kangaroos obtained the highly efficient locomotion from the highly elastic tendons. Although there is no doubt that elastic tendons help conserve energy during each hop cycle, recorded to be up to 41% per hop, but this does not vary with speed, nor is it a unique characteristic of kangaroos. There has been evidence produced by Dimery et al. [7] [8] that suggests that quadrupeds have even more specialised tendons, and in fact store and recover elastic energy more effectively then kangaroos. If this was the key contributing attribute responsible for highly efficient locomotion, the quadrupeds' locomotion should be more efficient than that of a kangaroo; however this is not the case.

Kram and Dawson observed that with an increase in speed, the ground contact per hop decreased. This would suggest that the increase in speed was not due to an increase in force, but instead, that the force was merely generated quicker. This theory was also confirmed by observing that as the gradient increased, the oxygen consumption also increased indicating more work being done. This confirmed that an increase in work done required an increase in oxygen consumption. From this observation it can be deduced that travelling at faster speeds does not require more work, as the oxygen consumptions remains essentially constant throughout varying speeds.

Although the paper by Kram and Dawson does not focus on dynamically modelling the kangaroo, it does cover some vital areas surrounding the motion of kangaroos. If an accurate dynamic model of a kangaroo is to be constructed it would have to conform to the observations made in this paper.

2.2 Work Capacity of the red kangaroo heart

Dawson furthered the research on kangaroo locomotion by investigating the work capacity of the red kangaroo heart [9]. It was discovered that kangaroos have a much higher aerobic capacity than initially believed, comparable to that of a thoroughbred horse. After measurements of mitochondrial volumes per kilogram of body mass, it was observed that red kangaroo hearts are capable of delivering much larger amounts of oxygen per unit time to working muscles compared to less locomotive efficient species. This reinforces the theory of kangaroo muscles generating forces quicker at increased speeds, as opposed to generating greater forces.

2.3 Running springs: speed and animal size

The paper "Running Springs: Speed and Animal Size" [10] investigated into more detail the observation that as speed increased, the ground contact between each stride decreased. This was due to the elastic properties of the Achilles tendon, which is able to conserve up to 41% of the mechanical energy required for each stride.

A simplified model was designed consisting of only a single leg spring and a mass; this was used to represent the compression and extension of the limbs during each stride. This spring represents the connection between the foot and centre of gravity of the kangaroo. This is a drastic simplification, disregarding the kangaroo's weight distribution as well as the role its tail and upper body plays in the motion. However it still is used well to explain how the angle of approach varies with speed for each stride. The model is depicted in figure 2.3a where it is represented in three stages of motion, beginning, middle and the end of the stance phase.

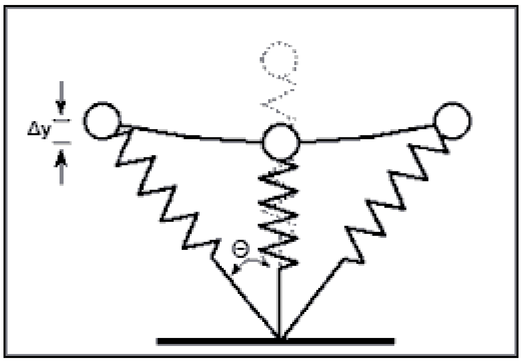


Figure 2.3a: The model used to represent trotting and hopping. Where the circle represents the centre of gravity of the animal, and the spring represents the connection between the foot and the centre of gravity. The notation Δy represents the change in centre of gravity throughout the stride. The notation Θ represents the approaching angle perpendicular to the ground. The spring undergoes compression at the middle point, and the dotted spring indicates its uncompressed position.

From the model constructed it can be seen that the centre of gravity only varies very slightly in the vertical direction with each stride. Kangaroos are known to have a very consistent centre of gravity throughout motion, and this model would explain it. As the kangaroo lands, the spring is compressed, converting kinetic energy to stored elastic energy. The centre of gravity then moves forward and the spring contracts keeping the centre of gravity at the same height. Once the model is vertical the spring expands releasing the stored elastic energy.

The angle Θ as shown in figure 2.3a was noticed to vary with a change in speed when measured for the kangaroo during hopping. Figure 2.3b shows the results obtained from this experiment and the pattern can be clearly observed. This would explain how the kangaroo was able to maintain increasing speeds without the need for additional force or energy. The noticeable relationship between angle and speed is not unique to that of a kangaroo, although the angle does play a more significant role compared to that of the other animals on which the tests were carried out (horse, dog and goat).

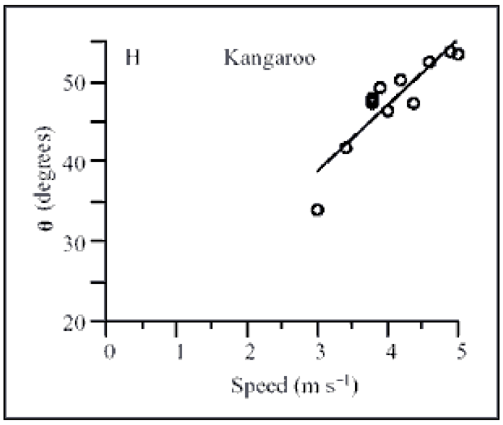


Figure 2.3b: The relationship between approach angle and current speed. An increase in angle results in an increase in speed.

This paper has also contributed another defining characteristic of kangaroo motion in that the faster it travels the greater the angle of approach is. When constructing a dynamically driven kangaroo model it will be interesting to record how this affects its motion, as it should be able to be used to vary the speed at which the kangaroo travels.

2.4 Animation of Dynamic Legged Locomotion

The paper "Animation of Dynamic legged locomotion" by Marc H. Railbert and Jessica K. Hodgins [13] reports on the development of control algorithms to animate dynamic legged locomotion. All motions are generated by numerically integrating equations of motion originally derived from physics models. By introducing user specified variables such as speed, height and gait into these equations, the user can control specific aspects of the motion. Notice that although the user is specifying certain values, these do not control the forces or energy operating within the system. These specified values are considered more to be conditions that the motion must meet, rather than initial conditions.

Three models were used to study the legged locomotion, a biped, a quadruped and a kangaroo; these are depicted in figure 2.4a.

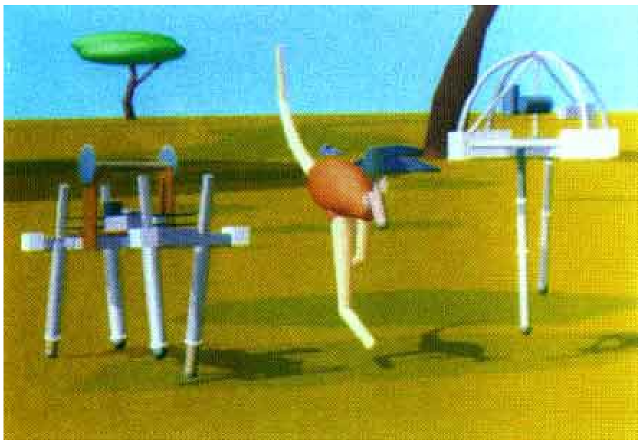


Figure 2.4a: The systems developed
By Railbert and Hodgins.

The original research was conducted on the biped and the quadruped as they had robotic versions of these to use as a foundation for the virtual simulations. However the interest soon turned to the kangaroo, interested by its unique running style and the ability to assume a symmetric model. Instead of adopting the approach of two identical legs and arms, the decision to simply flatten the model was taken. After minor simplifications of the model ,this left the underlying structure of three tail segments, three leg segments and a segment for each the head, body and arm. This simplified model would make the algorithms simpler, yet still portray the kangaroo's original motion in a two dimensional form.

As previously mentioned one of the most important requirements of a dynamic model is its ability to be independent. Once the original attributes of the motion are specified by the user, the system should then run on its own, not requiring any user intervention or correction. Figure 2.4b shows the general process used for the animation, where the user provides information about the desired animated behaviour such as speed or gait. From the moment the animation is started, the control algorithms take over and ensure the motion is maintained, by adjusting the forces exerted on each joint.

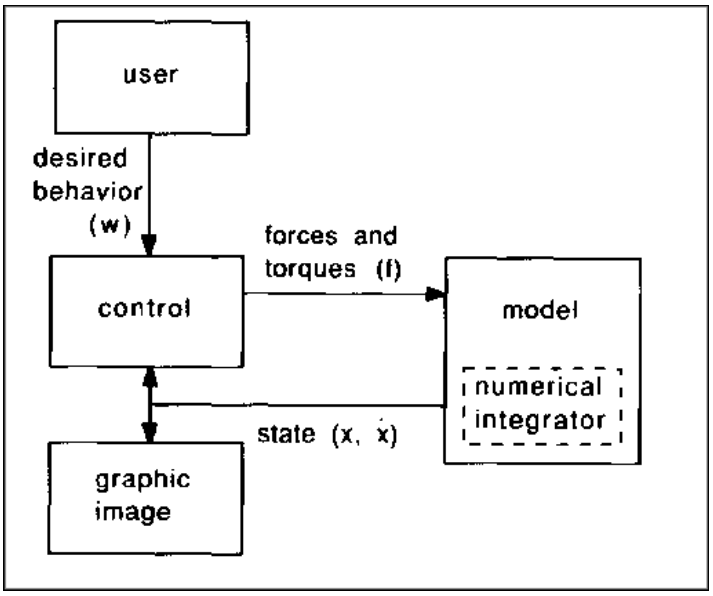


Figure 2.4b: The block diagram of the animation process used by Railbert and Hodgins [13]. The 'model' block consists of the algorithms that calculate what forces need to be applied to the model. The 'control' block applies the forces and torques to the system.

Within the paper there is a very significant section on the elasticity of the muscles within the kangaroo, and how this is responsible for maintaining its energetic efficiency during locomotion. When the animal collides with the ground, muscles within the legs stretch converting some of the kinetic energy to elastic energy. The elastic energy is then released when the animal takes its next stride, and the cycle continues. This form of energy transfer can make up a significant proportion of the total running energy, which can be between 20% and 41% [23].

As the model constructed assumed a two dimensional problem, it made modelling the kangaroo a lot simpler. As be can seen from figure 2.4c the kangaroo structure contains only nine sections with seven joints connecting them. This is clearly a big simplification of a real kangaroo's structure however it is more than adequate for a representation of a kangaroo's motion.

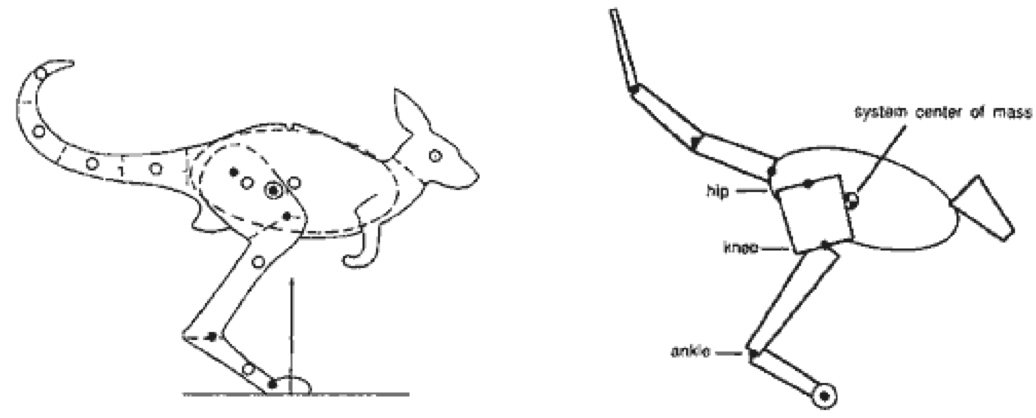


Figure 2.4c: The simplifications used to produce the model of the kangaroo

The results yielded by the experimental tests carried out by Railbert and Hodgins were very impressive, closely matching those of a kangaroo. This suggests that although their model required specific adjustment for each given set of requirements as set by a user, it was still successful in matching the motion of a kangaroo.

The paper by Railbert and Hodgins has shown some great insight into the project, and has clearly explained some of the underlying principles of kangaroo motion. As the model uses complicated algorithms which are specifically tailored to their model structures, they do not represent a kangaroo's structure closely enough. The dimension and weight distribution values are that of their structure, rather than that of a real kangaroo. Although these values do bare a close resemblance they are far from scientific.

2.5 Dynamics and Inverse Dynamics

Dynamics and Inverse Dynamics play an important role within the articulated structure of the kangaroo, as without forces or energy the structure will merely collapse.

2.5.1 Forward Dynamics

Forward dynamics is used when a specified force or torque is applied to a body to produce acceleration. From knowing the force exerted by a particular muscle or spring, and the angle at which is it applied, it is possible to determine the resultant motion produced with respect to time.

2.5.2 Inverse Dynamics

Inverse dynamics is more applicable to this project. The first requirement for inverse dynamics is to have a contact force. This force is then inversely propagated along the kinematic chain to determine the torques required to achieve a certain angle. This is much more realistic, for example when a limb is moved; the brain identifies a target and contracts/expands the required muscles accordingly. When using inverse dynamics it is important to include constraints that make sure that breaks in the model, or joints bending back on themselves, do not occur. As a result it is likely that some poses will not be possible considering the restraints, which is perfectly acceptable.

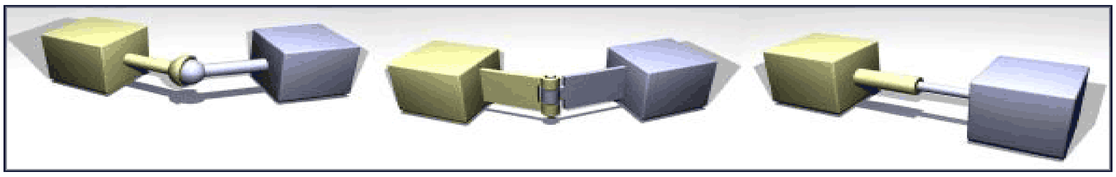
2.6 The Open Dynamics Engine (ODE) Version 0.5

ODE [11] is a free, open source dynamics library for simulating articulated rigid body dynamics. It has been developed by Russell Smith with help from several contributors over the years. ODE also has an inbuilt collision detection system called OPCODE which could prove very useful for kangaroo development, especially when calculating ground collision forces. The current collision primitives are sphere, box, capped cylinder, plane, ray, and triangular mesh, although only boxes will be required for the project.

As mentioned before, ODE's primary function is to simulate articulated rigid body structures, which are created when rigid bodies of various shapes are connected together with various types of joints. ODE is designed to be used in interactive or real-time simulation, which means it is able to deal with environmental changes on the fly, as well as user interaction.

Although ODE is still being developed it has been assured that it is incredibly stable, and that what has been developed is both accurate and reliable. There is still lacking support for complex volumes and the environment can easily be overloaded if there are too many joints in the scene. That aside, the features required for the project are fully implemented and have been stable since version 0.4.

The rigid bodies are connected together via various types of joints, each behaving in a specific way. The major joints implemented in ODE are shown in figure 2.6a.



Ball Joint Hinge Joint Slider Joint

Figure 2.6a The major joints available in ODE. Although ODE has support for other joints, they are merely combinations of the above. Each joint is capable of having forces applied to it, as well as flexibility constraints.

The ODE library is designed to be used with OpenGL and C/C++ and there are a few examples provided. The binary available from the ODE homepage [12] also included a graphic interface which allows the user to navigate the virtual world, whilst running the simulation.

ODE does prove to have the dynamics engine required to create a virtual kangaroo within a real physics environment. It will just take a lot of research to discover how to make use of its potential.

2.7 Simulation of animal motion in a virtual environment [14]

The research carried out by Cameron Luke Greatorex offers a good insight into how kangaroo motion can be modelled. A model of a simplified kangaroo skeleton was constructed in 3D Studio Max. Video motion capture of a kangaroo was used to animate the skeleton using key frames, which is shown in figure 2.7a.

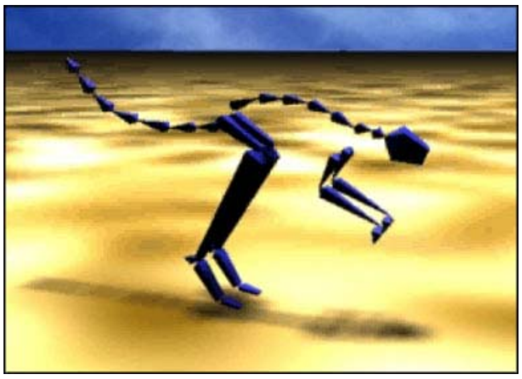


Figure 2.7a: Screen capture of the video produced using 3D Studio Max.

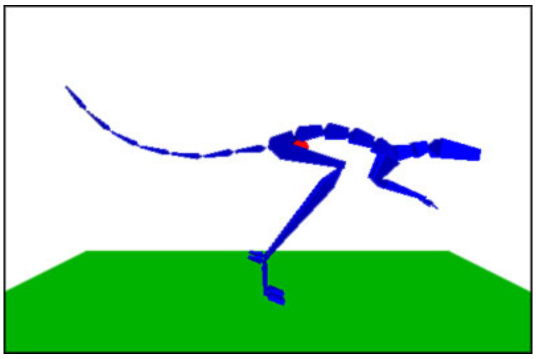


Figure 2.7b: Screen capture of OpenGL motion simulation. The red circle indicates the centre of gravity, and is calculated automatically.

The 3D Studio max model was then recreated in OpenGL (figure 2.7b) to allow for energy and force calculations. The OpenGL simulation was used to record the path of the centre of gravity throughout the kangaroo's motion. The simulation was also used to perform external energy calculations.

The motion produced from both the 3D Studio Max and OpenGL implementation will be very beneficial to the project. Any simulations produced in the project can be compared to that of the motion produced by the simulations implemented by Greatorex.

3 Project components

The aim of this project is to try to understand how the chosen animal moves, and what forces are responsible. More importantly, to use these forces to produce a true to life representation of the animal's motion. With this model it will be possible to animate the animal.

As discovered from the literature survey, there are many unanswered questions surrounding the motion of kangaroos, and how they manage to travel great distances with little energy cost. There has been many theories suggested, however they all are dismissed as soon as a more recent paper is published which conducts further testing.

Several observations remained consistent throughout the research; these should be incorporated into the model.

When the kangaroo travels at faster speeds, there is not an increase in oxygen consumption. It can be deduced that the work done at faster speeds is no greater than that of slower speeds, therefore the overall energy use at varying speeds remains constant. However this was only observed on horizontal terrain. As soon as an increased gradient was introduced it was recorded that energy consumption increased with an increased speed [12].

Kangaroos have exceptional elastic energy storage within their limbs. This feature conserves up to 41% of the total energy required for each stride. The amount of energy conserved will vary depending on gradient, and will decrease as the gradient gets steeper [7,8,11,12].

Kangaroos generate the same muscular force at all speeds but do so more rapidly at faster hopping speeds [9,12]. The speed is controlled with the change of the angle of approach between the foot and the kangaroo's centre of gravity. The greater the angle of approach, the faster the kangaroo will travel; this is illustrated in figure 2.3b [10].

With the above observations it will be important to make sure that the model constructed portrays these observations, and hopefully some new ones. The structure of the solution presented by Railbert and Hodgins [13] will be very similar to that of this project, specifically isolating the user from the internal calculations of the simulation.

As there currently is no open source available for the dynamic modelling of kangaroos in VE's it means that a new line of work will need to be undertaken. The project's overall objective is to have a dynamically functioning model of kangaroo motion through a VE. This objective is quite large and it would be much more manageable to break the objective into smaller objectives. It seamed logical to work from a very simplistic model of just a bouncing ball and gradually increasing the complexity of the model until the main objective is reached.

Sub-objectives are to produce simulations of the following:

- Vertical based force movement
- Projectile based force movement
- A two segment bone structure
- A three segment bone structure
- Animal motion based on dynamics

By following this structure would mean that if the project were to not be completed it would be clear to see what point had been reached and how work could be resumed.

The relevance of each sub-objective is detailed below.

3.1 Vertical based force movement

This will be based on a pogo stick movement, however at this stage assuming no directional movement other than in the vertical. This will be assumed to be elastic initially, until inelastic collisions involving energy loss through heat and sound are introduced.

- Research vertical based force movement.
- Research spring energy and elastic and non-elastic collisions.
- Develop pogo stick simulation for elastic collisions.
- Develop pogo stick simulation for non-elastic collisions.

The logical approach when developing a new simulation is to start from the very basics and gradually increasing the complexity. A simple bouncing ball was chosen to be the start point for the model, only travelling in the vertical plane. This would require some research into the equations of motion to ensure this ran at real time, and was accurate. Once this step was completed non-elastic collisions would be introduced as the majority of collisions are non-elastic. In the kangaroos motion a lot of energy would be lost though heat, so this would need to be incorporated into the model.

3.2 Projectile based force movement

By dealing with an object accelerated at a given angle it will be possible to calculate motion paths, taking into account elastic and inelastic collisions. Once this is achieved it will be possible to modify the vertical based force movement to produce a pogo stick based simulation that can move in all directions.

- Research projectiles.
- Develop Projectile simulation.
- Apply directional based movement to pogo simulation.

Once a vertical simulation has been developed the next logical step would be to add horizontal motion. This would be introduced in the form of using models for projectiles, and applying this to the bouncing ball. This should allow the user to specify the initial launch angle and velocity, and then the simulation should take care of the motion path modelling. Once this has been achieved it should be relatively simple to introduce non-elastic collisions into the simulation.

3.3 A two segment bone structure

By initially assuming that there is no balance to take into consideration and that the motion is entirely vertical, a two segment bone structure will be created. This will be able to move vertically at first, and then other dimensions will be introduced. To make the movement more accurate, the model will be modified to include the shift of the centre of gravity which actually causes the directional movement.

- Research two segment bone structures.
- Research structure dynamics, force storage, and elasticity.
- Develop a model of a two segment bone structure using kinematics.
- Introduce dynamics into the model and produce a simulator.
- Introduce additional dimensions.
- Modify simulation to include centre of gravity, and use to produce directional change.

This section will be the biggest challenge, as to go from a bouncing ball to an articulated structure is quite a step. Required will be a great deal of research conducted into centripetal forces and acceleration. This is considered to be the biggest mile stone, as once it has been worked out how to complete a two segment bone structure it should not be too tricky to add another segment. By adding forces to the model it should be possible to produce a simple hinge like motion, controlled by varying forces.

3.4 A three segment bone structure

The same order of development as the two segment bone structure. This essentially should produce an operational hopping leg like structure.

- Research three segment bone structures.
- Develop a model of a three segment bone structure using kinematics.
- Introduce dynamics into the model and produce a simulator.

By completing a three sectioned structure it should be possible to simulate the movement of a kangaroo leg. The simulation will be limited to two dimensional space, as the kangaroo's motion is assumed to be symmetrical and only forward and backward balance will be considered.

3.5 Animal motion based on dynamics

Additional segments will be added one by one; once a complete structure is achieved it will be possible to move the model to user specified coordinates.

- Continue adding segments to build up to a complete structure.

Gradually building up the structure of the kangaroo will enable each segment to be individually tested. Once a full kangaroo structure has been completed, movement constraints can be added, this will stop joints bending back on themselves.

The next step is to add motion to the model in the form of a forward hop. Whilst tackling this step, it will be important to include the observations noted in section 3.0, and see how this affects the motion of the kangaroo.

4 Design and Implementation of solutions

Each simulation was coded in C/C++ using the OpenGL graphics library, to allow full control over the implementation. The features unique to the kangaroo do not become apparent in the simulations until simulation 5, as the previous simulations are rather generic, and could be applied to any animal.

4.1 Design of Vertical based force movement

A falling two dimensional ball has been modelled using the equations of motion. The ball dropped from a specified height, accelerates towards the ground at 9.81ms^{-2} (acceleration due to earth's gravity). When the ball's height is equal to 0 (ground) a collision has occurred and the new velocity of the ball is calculated. The coefficient of restitution between the ball and the surface will determine what velocity the ball leaves the ground at. In an ideal world the collision would be completely elastic, in which case the coefficient of restitution would be 1, resulting in the ball maintaining all its energy, just changing in direction. In reality this is not the case, energy is lost through heat or sound for example, and the ball will lose energy after each collision and eventually come to rest.

The following equations are used to calculate the ball's position at key stages throughout its motion.

$$s = s0 - \left(ut + \frac{at^2}{2} \right)$$

Equation 4.1i

$$v = u + at$$

Equation 4.1ii

$$v = eu$$

Equation 4.1iii

Where s = distance, u = initial velocity, t = time, a = acceleration, v = final velocity, e = coefficient of restitution and s0 = start height.

Using equation 4.1i as the ball is starting at a specified height its initial velocity is 0, therefore the effect that the initial velocity has can be disregarded since it will be 0. By substituting u as 0 gives equation 4.1iv.

$$s = s0 - \frac{at^2}{2}$$

Equation 4.1iv

$$s = ut + \frac{at^2}{2}$$

Equation 4.1v

From equation 4.1iv the distance the ball has fallen can be calculated by simply substituting in values of t. This was embedded in the main program loop, and synchronised with the system clock. This enabled the ball to be plotted and moved at real time. The ball had hit the ground when s was equal to zero. Due to the precision of a PC the likelihood of s equalling exactly zero were slim, so it was decided to produce the collision response when s became negative.

Equation 4.1v was used to model the ball's upward motion, as it now has an initial velocity. The initial velocity after the collision is calculated using equation 4.1iii and is used within equation 4.1v.

The ball's velocity on its upward journey will eventually equal 0 due to gravity, which determines the new start height of the ball as this is the highest point the ball will reach. From here the motion of the ball is calculated again using the same method as earlier just with a new start height. This cycle continues until the start height is equal to the ground, which completes the ball's motion.

4.2 Design of Projectile based force movement

Projectiles can be modelled using parabolas, and depending on when the exerted force is exhausted, the propelled object will experience an acceleration towards the ground. There are two phases of motion for a projectile: the propelled phase and the falling phase. The path that a projectile takes can be modelled using Newton's laws of motion resulting in a parabolic curve.

The curves describing all these objects in motion share the basic $y = -x^2$ shape. This shape is known as a parabola because the change in the height is happening in a different way to the change in horizontal distance [4].

This was added into the existing model of the bouncing ball to add directional movement. After a little research this turned out to be a lot simpler than initially expected. All that was required was to add the horizontal component of the motion. This done, a completed motion of a directional changing bouncing ball was complete, that ran in real time, accurate to real physics laws.

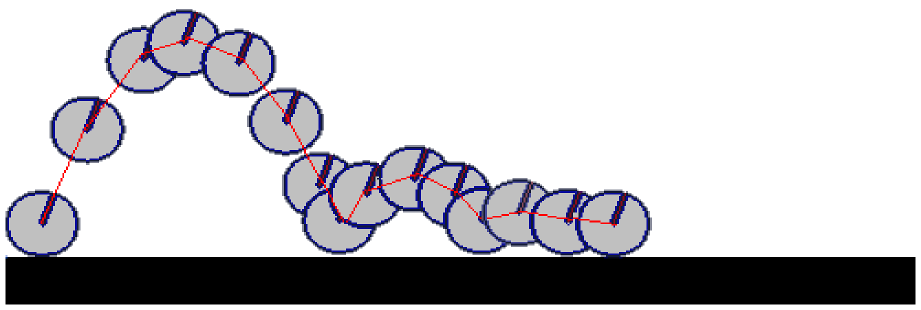


Figure 4.2a – Selective screenshots from the OpenGL implementation of projectile based bouncing ball, with a non-elastic collision.

Figure 4.2a shows a bouncing ball projected at an angle of 70 degrees from the horizontal, with an initial velocity of $35ms^{-1}$. The implementation allows the user to select an angle of projection and vary the initial velocity. Currently a non-elastic collision is simulated by varying the coefficient of restitution manually.

Once the initial conditions of the simulation are set, the user can start the simulation. From this point on, the user has no input into the simulation and the motion of the ball is calculated automatically in real time.

4.3 A two segment bone structure

Some additional research was carried out at this point regarding angular motion, and centripetal forces. The books "Mechanics and properties of matter" [16] and "Newtonian mechanics" [17] were studied in order to obtain a better understanding of how the physics could be used for the simulation.

However it appeared that several major components of a physics simulation system had been overlooked, and would need to be addressed in order to make the simulation work.

The most important one was a collision detection and response system. Without this the model would simply fall through the ground. Until now, the lowest point on the bouncing ball had been checked against the horizontal ground plane. However the simulation would require an axis oriented collision system, which would allow each segment to be checked against one another as well as the ground.

After careful consideration it was concluded that it would be worth further investigating the functionality of the ODE physics engine [18]. ODE was developed by Russell Smith as what started out as a personal project. As a result, there is little documentation detailing how to implement it. As the libraries are particularly new compared to mature libraries such as OpenGL for example, there are very few reference websites.

From reading the ODE documentation [11] it was clear that with ODE's inbuilt physics engine and collision detection library (OPCODE), ODE would be the logical system to use to develop the remainder of the project.

After studying some examples it was decided to skip over the two segment bone structure for two reasons. The first of which was that development time for the project was running out, and it was already behind schedule. The second was that it became apparent that creating a two segment bone structure would be no less complicated than creating a three segment structure. A three segment bone structure would be much more useful to complete, and would be able to simulate a hopping motion much like that of a kangaroo, unlike a two segment structure.

4.4 A three segment bone structure

This simulation was the first that would take advantage of the ODE libraries. As a result, it would require its own sequence of sub-objectives. The first of which will be a simulation of a single bodied object simply falling under the effects of gravity. Gradually increasing the complexity of the simulation will make development more manageable.

4.4.1 ODE Typical application format

ODE comes packaged with its own graphics interface which uses OpenGL to create a virtual world in which all simulations are run. The interface also provides world navigation using a mouse, as well as the ability to change specific simulation parameters, figure 4.4.1a.

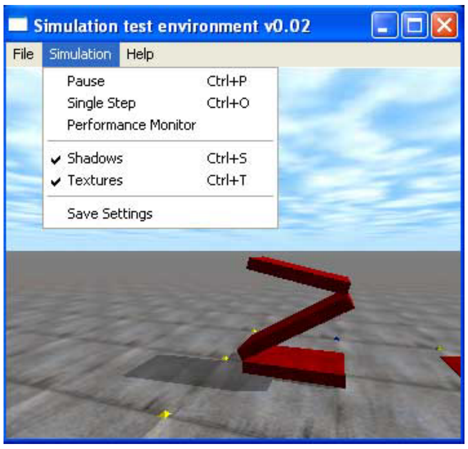


Figure 4.4.1a: Ode simulation interface.

A typical ODE application takes the following form:

```
static void nearCallback (void *data, dGeomID o1, dGeomID o2)
static void start()
static void command (int cmd)
static void simLoop (int pause)
int main (int argc, char **argv)
```

Although ODE is compatible with most languages and graphics libraries it was decided it would be sensible to stick with the languages the examples were written in.

Each of the key sections above are described and explained below.

The method `nearCallback` is responsible for collision detection between two objects. Using OPCODE, every time a collision between two objects occurs this method is called, passing in each of their ids. Depending on the parameters defined within the method, the type of the collision response is determined.

The `start` method is responsible for defining the initial viewpoint for the simulation interface.

The `command` method listens for keystrokes and can perform various actions when keys are pressed and released.

`SimLoop` is the thread of the application. This method runs continuously unless the simulation is stopped. Within this method the collision call back method `nearCallback` is called, as well as the more important call `dWorldStep`. The call `dWorldStep` is an ODE specific call that takes a step forward in the timeline of the simulation. So if for example the program was set to run the method simLoop every 0.1ms, the time step would be instructed to increment through the timeline of the simulation at that rate also in order to remain in real time. As well as being responsible for stepping through the simulation, this method also includes all the graphics calls for displaying the objects in the VE.

Within the `main` method all the objects and joints are created and their parameters are set. A world space needs to be created first, and once each object has been created it needs to be added to the world space. Each object has a dBodyID which is responsible for the graphical representation of the object, and a dGeomID which is used for the physics interface, referring to dimensions, weight distribution and density.

4.4.2 A Three Segment Structure That Holds an Angle

This section discusses the most significant parts of the development used to produce the three segment structure as shown in figure 4.4.2.

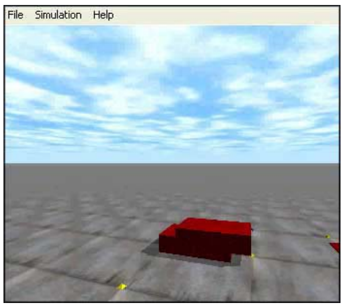


Figure 4.4.2a(a)

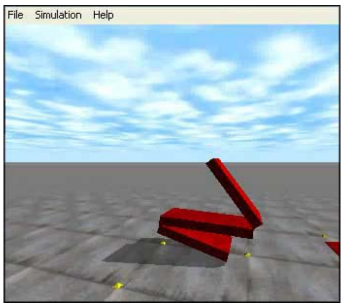


Figure 4.4.2a(b)

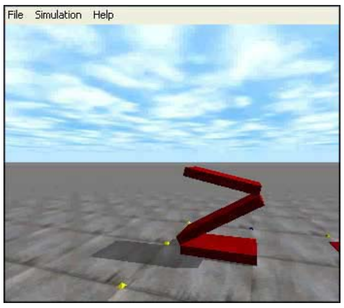


Figure 4.4.2a(c)

Three Screenshots from the three segment structure that successfully holds a user specified angle, despite gravitational influences.

```
static dReal toRadians(dReal degree)
static dReal toDegrees(dReal radians)
```

Two additional methods were written to convert between radians and degrees. This is because ODE uses radians, however in most cases it will be required to read in and display the angles in degrees.

The actual construction of the bodies and joints takes place in the main method, as they only need to be created once. Once created and added to the virtual world, various forces can be applied to them.

Each section of the structure is created and has a dBodyID and a dGeomID associated with it. First the body needs to be created and then added to the virtual world. Once created, the dimensions and the initial position of the dBodyID are defined, as well as its mass. An accurate mass that reflected the volume was calculated based on a user specified density, using the equation:

```
mass = volume x density.
```

A new dGeomID is created for the dBodyID, and is defined as a box of equal dimensions to that of the dBodyID. This defines the boundaries of the dBodyID and is used for collision detection, ODE supports many other shapes but for this simulation only boxes are used. The dGeomID needs to be linked to the dBodyID so that they move throughout the virtual world together.

This procedure was followed three times for each of the segments, and their positions were set so that they would appear to be placed on top of one another, as shown in figure 4.4.2a(a). The structure in its current state is merely a stack of three boxes places upon one another, which are in no way attached.

For this simulation it was decided that a hinge1 type joint would be used to connect one box to the next, as illustrated in figure 4.4.2b.

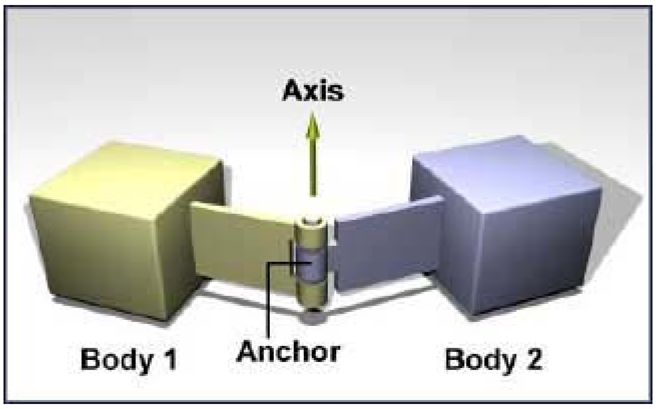


Figure 4.4.2b: A hinge 1 type joint, as implemented in ODE.

A new joint was first created within the virtual world and assigned a dJointID. The joint is then attached to the two bodies that are required to be connected. As the hinge 1 type joint only has one hinge axis this needs to be defined as the hinge anchor point. This is usually the centre of the edge that both the objects share. The axis on which the hinge pivots is then defined, as shown in figure 4.4.2b. There are many additional parameters that can be defined however only the LoStop and HiStop parameters were defined. These parameters take the maximum and minimum angle in radians that the hinge is able to maintain. This is very useful for restricting the range of motion of the hinge.

A collision space is added to the world, and the dGeomID's are added in order to assign the dBodyID's their collision boundaries in the virtual world. The simulation method is then run, which runs continually looped until the application is stopped or closed.

The physics environment that ODE creates unfortunately does not deal with angles directly; angles can be read but not set. This is a true reflection of real life in that when an angle is desired, a force is applied until that angle is reached.

For this stage in the simulation it was required to produce an angle of 30 degrees on each joint and maintain that angle despite gravitational effects.

The ODE libraries allow the current angle of a joint to be retrieved dynamically throughout the simulation; it is also possible to apply a torque to a hinge. The hinge 1 joint also has two other important parameters regarding force and velocity values. It is possible to set the desired velocity of the joint and the maximum force that can be used to maintain this velocity. If the velocity was set to 0 this would cause the joint to try to maintain its current angle using the specified maximum force. Should the force required to sustain the angle exceed the specified maximum force the structure will collapse.

Using these conditions, a very simple algorithm was devised that increases the angles within the structure until they both are at 30 degrees. From this point on the angles are maintained using a specified maximum force, despite gravitational acceleration.

```
if joint angle is less than 30 degrees then:

     apply more torque to joint.

else

     Set desired velocity to 0.
     Set maximum force to sustain velocity.
```

This was used on both joints within the structure and successfully produced the animation sequence show in figure 4.4.2a. This is by no means the most efficient or accurate way of obtaining and maintaining an angle, but up until this point it was the only known way of doing it.

4.4.3 A hopping three section structure (backwards)

Shown in figure 4.4.3a is the animation sequence produced for this section, the motion was controlled using sin wave to vary the force applied to the joint.

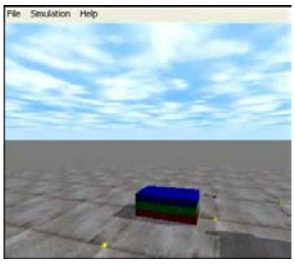


Figure 4.4.3a(a)

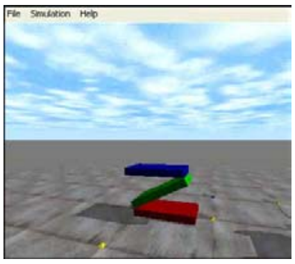


Figure 4.4.3a (b)

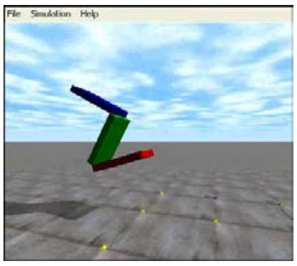


Figure 4.4.3a (c)

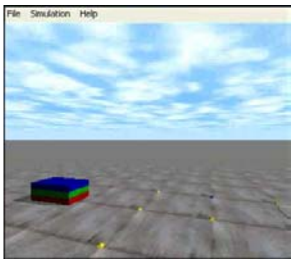


Figure 4.4.3a (d)

Four Screenshots from the three section structure hopping backwards. The structure hops the opposite way to that of a kangaroo, this is due to the position of the centre of gravity of the structure.

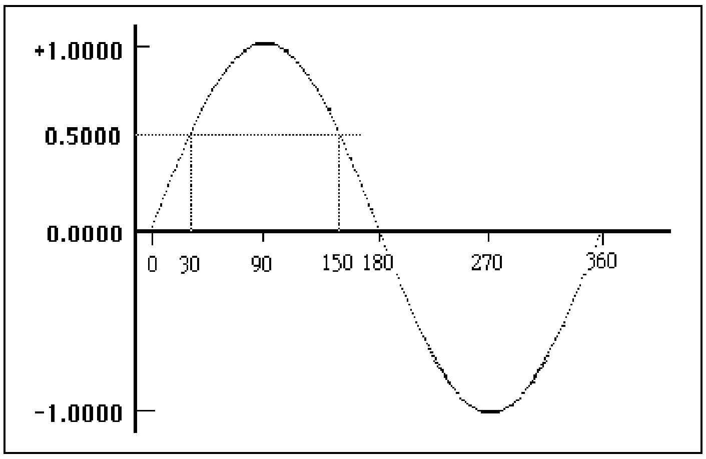


Figure 4.43b: The Sine curve between 0 and 360 degrees.

An algorithm was written that using the sine curve (figure 4.43b), applies a force that varies between the maximum and the minimum force.

```
sinIncrement += 0.5;

if sineIncrement == 360 then sinIncrement = 0.0;

jointForce = forceMax x sin (toRadians (sinIncrement));
```

The above algorithm causes the force applied to oscillate, and as a result produces an opening and closing of the joint. The joints were set to have a maximum angle of 60 degrees and a minimum angle of 0 degrees. The minimum constraint stops the structure collapsing into itself. The maximum constraint was used to produce the Z like structure and by applying a force quickly resulted in the structure leaping into the air.

4.4.4 A hopping three section structure (forwards)

It was discovered that the reason the previous structure did not move in the correct direction was due to the incorrect position of its centre of gravity. When an animal wants to move in a particular direction it needs to temporally shift its centre of gravity off balance in the direction of the required motion. It was observed from a kangaroo video recording that a kangaroo leans initially forward, off balance, just before it starts hopping [20].

To adjust the centre of gravity two separate approaches were used. The first simply extended the uppermost segment of the structure shifting the centre of gravity further forward, figure 4.4.4a. The second required an additional section to be added to the structure, which would represent the body of the animal. This section was also extended forward to create an off balance centre of gravity.

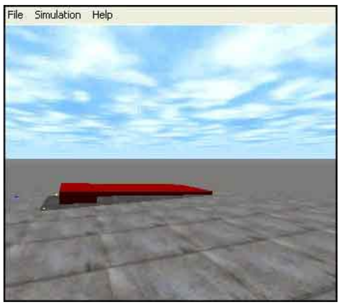


Figure 4.4.4a (a)

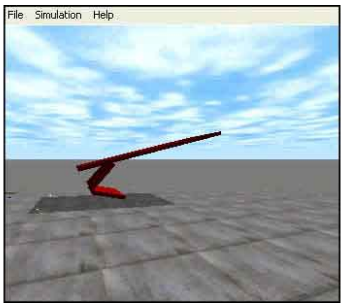


Figure 4.4.4a (b)

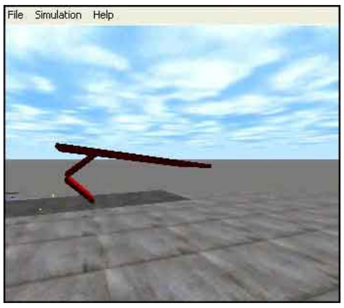


Figure 4.4.4a (c)

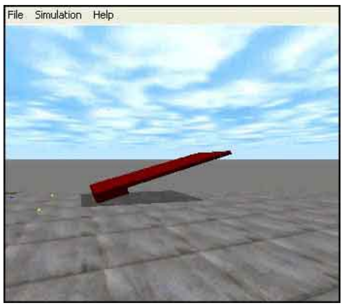


Figure 4.4.4a (d)

Illustrating how the direction of motion was corrected by offsetting the centre of gravity. This however did reduce the number of segments used to generate the motion *(Simulation 3).*

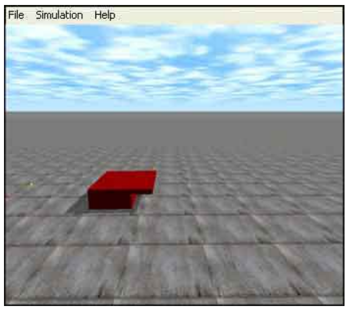


Figure 4.4.4b (a)

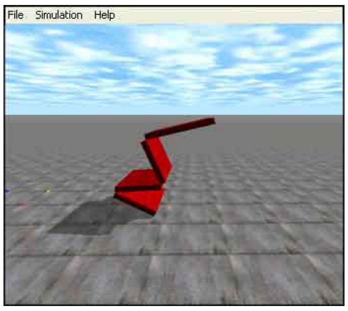


Figure 4.4.4b (b)

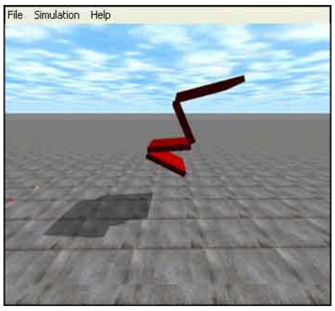


Figure 4.4.4b (c)

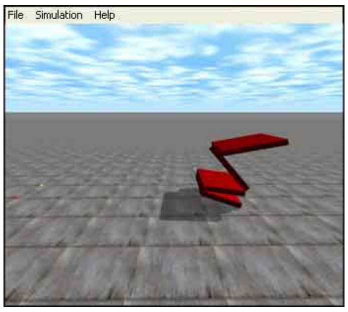


Figure 4.4.4b (d)

In this simulation, a fourth segment was added to act as the body of the structure, which was offset in the direction of required motion. This did not need to be as long as the extended segment in simulation 3 because it was given a greater mass.

The first approach as shown in figure 4.4.4a(a – d) produced the desired results, and the structure hopped in the correct direction. However this sacrificed a segment that should really be part of the leg structure, instead it was being used to offset the centre of gravity.

The second approach (figure 4.4.4b) yielded better results, as it still contained the full leg structure as well as an additional section for the centre of gravity offset. The algorithm for providing the motion was slightly modified to include a slight pause at the minimum force value. This was to allow the model to land correctly before trying to jump again, and added a slight pause in between each jump.

Both the simulations' motions are controlled by simply oscillating forces. As a result, if a change in the environment were to occur, the model in most cases would just fall over. The values used for the forces were simply obtained via trial and error, and whichever produced the best results were used. Once the final model is constructed real force and energy values will be incorporated in the system, and the kangaroo characteristics observed in previous research will be incorporated [5 - 10].

4.5 Kangaroo motion based on dynamics

After further research it was discovered that applying torques directly to joints was the incorrect way to create motion about a joint. Instead it was advised that an angular motor should be added between the two bodies; this allows the relative angular velocities of two bodies to be controlled [11].

A new angular motor is created within the virtual world and attached between the two bodies that share a hinge 1 type joint already. The mode of the motor needs to be defined as a `dAMotorUser` type, which allows the most flexibility of use. The motor needs to have three axes, although two of them will be fixed. If the motor was defined as only having one axis, the other two would have freedom of movement, which is not required for this model. Each of the three axes needs to be defined, and is represented by the `aNum` parameter. The axes in this case are arbitrary and each perpendicular to one another, however ode allows for any axis to be defined. Although ODE does not allow angles to be set, it allows angles to be retrieved. As a result the initial angle that these motors hold needs to be defined, in most cases it is 0 degrees and considered to be the its angle of rest.

The algorithm used to move the joints to the required angles was updated to reflect the addition of angular motors. Previously, the joints had been controlled by using the hi and low stops of the hinge 1 joint, however it was discovered this is inaccurate and can lead to unstable behaviour.

```
for(all motor joints)
{
      dReal v0 = desired angle - current angle

      if (v0 > 0.001) v0 = 0.01;
      if (v0 < -0.001) v0 = -0.01;

      v0 *= 1.5;

      set motor desired velocity as v0;
      set motor maximum force available;
}
```

The above algorithm was obtained from the ODE mailing list [19] and modified for the simulation. It works by calculating whether the current angle is less than or greater than the desired angle. Depending on this value a positive or negative velocity is set and is applied to the angular motor connecting the two bodies. A maximum force available to sustain the velocity is also defined. If the angle required is equal to the current angle then the velocity is set to zero and the maximum force value holds it in its current position.

This algorithm is more accurate than the previous because it allows for dynamic changes of the angles between joints. If for the example the angle between two bodies was to change because of an external influence such as gravity or another object, the angle is re-obtained. The previous algorithm did not account for this and instead the structure collapsed.

At this stage in the project it was decided that some major structural changes would be made to the implementation in order to make it a lot more manageable. When constructing structures with ODE it is simpler to place all the sections in line with one another. Then using the angular motors, the joints can be rotated to produce the required shape of the structure.

Using this principle, a bone class was written that would be used to create each segment of the structure. The first bone was placed manually, however every bone that would link to that has its position calculated automatically based on the previous bone's position and dimensions.

A joint class was also written that took the dBodyID's of two bodies it was to connect and the dimensions of the first body in the joint. This was needed in order to be able to successfully position the hinge's axis.

An image of a kangaroo's skeleton was obtained and this was used as a basis for the kangaroo structure. The skeleton is depicted in figure 4.5a, some simplifications were made in order to reduce computational load. The kangaroo model was to only be two dimensionally modelled and bones that played no significant part in the motion were removed, such as the rib cage and the fingers.

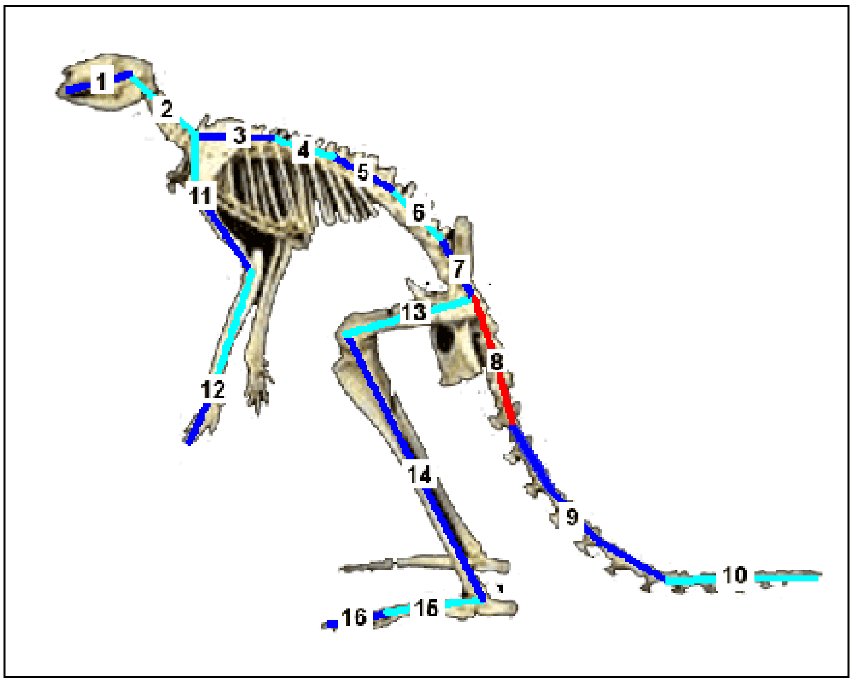


Figure 4.5a: The kangaroo skeleton obtained from Skulls Unlimited [21] was used to create the structure in the simulation.

Each joint object had a control variable linked to it; this was used to set the desired angle of that particular joint. The angles in this image were measured using the protractor tool in Adobe Photoshop and were applied to the model in the simulation. The simulation then used a simple loop to iterate through each joint and using the motor algorithm each joint was correctly positioned.

The centre of gravity of the kangaroo plays a significant role in its motion, determining its balance, which in turn will affect its motion. It was decided that it would be useful to have a representation of the centre of gravity within the simulation. This is useful for working out when the model was off balance, and where majority of its weight distribution was. To calculate the centre of gravity the following formula is used.

```
Centre of Gravity = ∑ (Mass * Distance) / Total Mass
```

By taking moments about a fixed point it is possible to calculate the position of the centre of gravity along each axis. As each bone has its own coordinate and mass value stored, it is easy to iterate though them and obtain the centre of gravity.

The algorithm used for calculating the centre of gravity is described below.

```
for each bone
{
      xTotalCOG += bone X coordinate * bone mass;
      yTotalCOG += bone Y coordinate * bone mass;
      zTotalCOG += bone Z coordinate * bone mass;

      totalMass += bone mass;
}

xCoord = xTotalCOG/totalMass;
yCoord = yTotalCOG/totalMass;
zCoord = zTotalCOG/totalMass;
```

The simulation is updated approximately every 1/10th of a second, as a result this is the rate at which the centre of gravity is dynamically calculated. The algorithm uses the coordinate (0,0,0) as the point of reference for calculating the centre of gravity.

Written into the simulation is the ability to manually adjust each joint's angle, which was to allow the manual tweaking of angles for particular poses and stages in the kangaroo's motion. By pressing the "," or "." keys you can cycle through each joint and adjust the angle using the "+" or "-" keys. It is then possible to print out all the angles of all the joints to a text file. This allows the user to hard code angles into the simulation to be used for the kangaroo's motion.

The collision detection code was also modified so that it ignored bone to bone collisions. This was done by checking whether the dGeomIDs of the colliding bodies were bones or not. This was required as the anchor point of the hinge was moved to be central along the Z axis, unlike the previous examples. If the dGeomID's were both found to be that of bones in the structure then the collision was ignored.

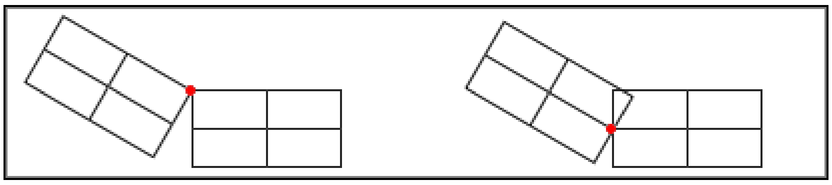


Figure 4.5b(a) Figure 4.5b(b)

Previous hinge anchor position (a), updated hinge anchor position (b)

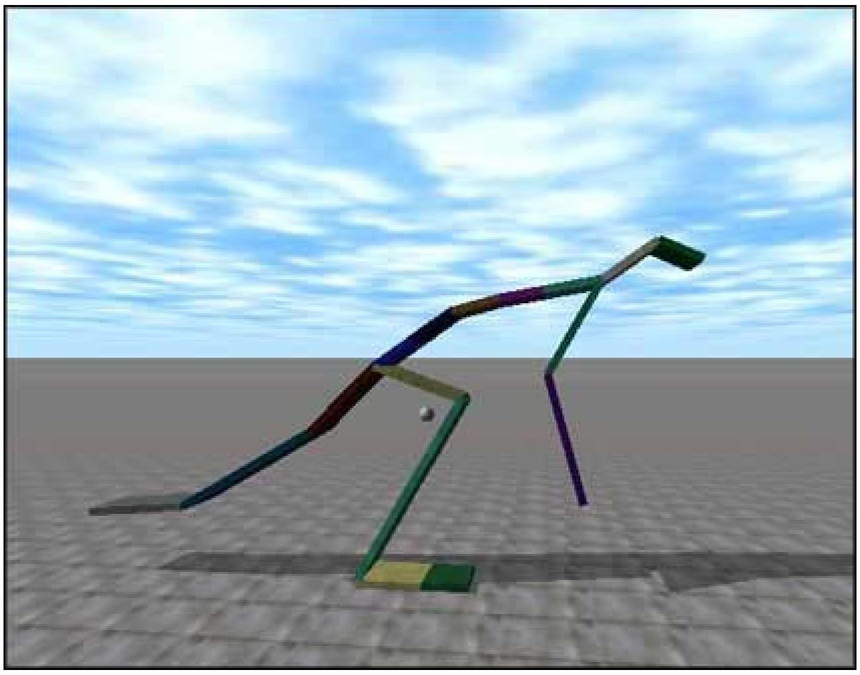


Figure 4.5c: The first fully constructed skeleton of the kangaroo with all bones held in place with angular motors, and the centre of gravity dynamically calculated and represented by the white sphere.

The ODE library has no units of measurement, which means a simulation of any scale can be created. The units although having no scale themselves are consistent with one another; as long as the developer is consistent with their values, the simulation will be accurate. The kangaroo simulation uses the scale of 1 ODE unit is equal to 10 real world units.

The kangaroo structure shown in figure 4.5c originally starts out as a horizontal row of sections (shown in figure 4.5d). Gravity is initially disabled whilst the structure is positioned. Once the bones are all in the correct position gravity is enabled, this causes the structure to fall to rest on the ground plane. This method of construction was chosen because if the structure is created from flat then the model was not upright, and required a lot of manual joint adjustment in order to position the model. Unfortunately as the whole environment is based on real physics it is not possible to just rotate an object without applying accurate forces.

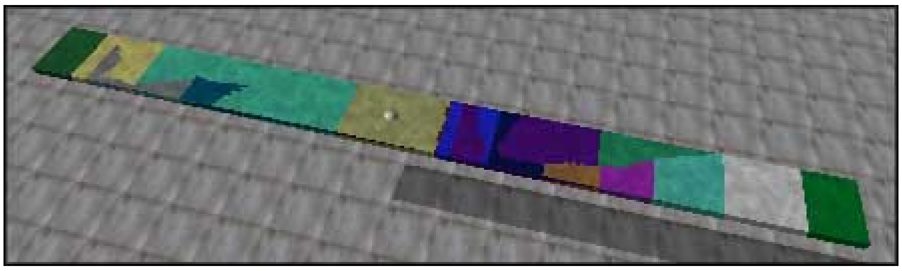


Figure 4.5d: The kangaroo structure before all the segments have been moved into place by angular motors to form the structure shown in figure 4.5c.

Four more poses were added to the simulation. These were taken from previous research into kangaroo motion [14]. The poses were manually constructed using the joint control system written into the program, and the angles of the finished pose where then exported to a text file.

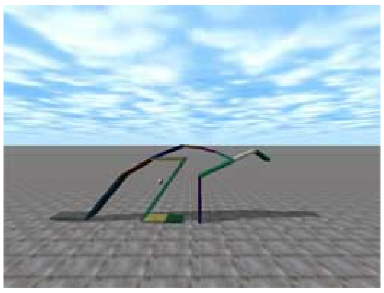


Figure 4.5d (a)

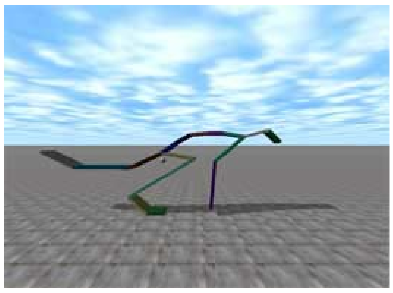


Figure 4.5d (b)

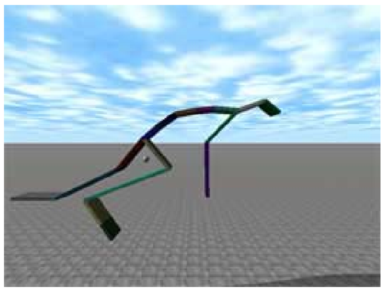


Figure 4.5d (c)

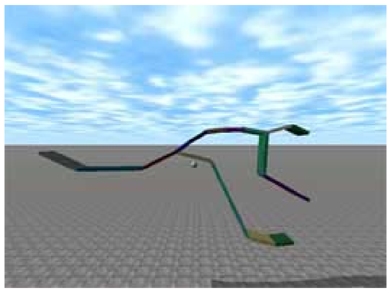


Figure 4.5d (d)

The 4 poses constructed in the ODE simulation. Pose (a) is a static pose and the centre of gravity is such that the model remains stable and balanced. Pose (b) is just as the kangaroo is about to jump and as a result the centre of gravity is such that the model is off-balance and about to fall forward. As the model tips forward, pose (c) is invoked and the kangaroo springs forward by extending its hind leg down and backward. During mid flight the kangaroo prepares to land by adopting the pose shown in (d). Unfortunately at this point due to the rigidness of the model it falls over. This is because the model has not been told to provide a level of give in its joints, to soften the landing.

The transitions between the poses (a) through to (d) shown in figure 4.5d were calculated and produced automatically. This is because the angles of the joint's are obtained using real torques and forces rather than using translation methods. As the joints are moved into their required positions it sometimes produces a rather un-natural order of positioning. This is because in some cases the adjustment of one joint causes another to be modified. The algorithm for moving each joint will take affect again, and eventually all the angles meet their requirements.

Currently, the force available to maintain each joint is set to be infinite, which as a result makes the structure very rigid. In a real life situation it is very rare that the animal will use the maximum amount of the force available to position its joints.

This is more evident when an animal lands after jumping; its legs act as shock absorbers and cushion the landing [7,8,10]. Although the animal will often have enough force in its muscles to maintain a rigid structure it will relax the leg muscles slightly. This is done to avoid injury because such a shock on a tense muscle could cause injury.

4.5.1 Simple Kangaroo Motion.

Although some attempts of applying motion to the kangaroo had been made it was clear that it was not going to be easy to obtain a complete hopping model straight away. Instead as with other aspects of this project the motion was to be built from the ground up, working from a basic and simple simulation gradually increasing the complexity.

It was decided that the first addition of motion to the simulation would be in the form of a small stepping like motion. Where only the leg moves to essentially drag the model across the virtual world, the tail and arm would be used to balance the model during this early stage of development.

To obtain the movement of the structure, the oscillations of the sine wave would be used. The previous simulations used the sine wave to vary the level of torque applied to the joint, however since those simulations it has been discovered that torque should not be applied directly to a joint; instead its movement should be controlled by an angular motor. These angular motors can only be controlled by specifying a desired velocity and then using an algorithm in order to obtain specific angles between joints.

To produce this very simple motion sequence, two sine curves will be used to control the desired angles on two joints. The two joints that are used to control the motion are shown in figure 4.5.1a.

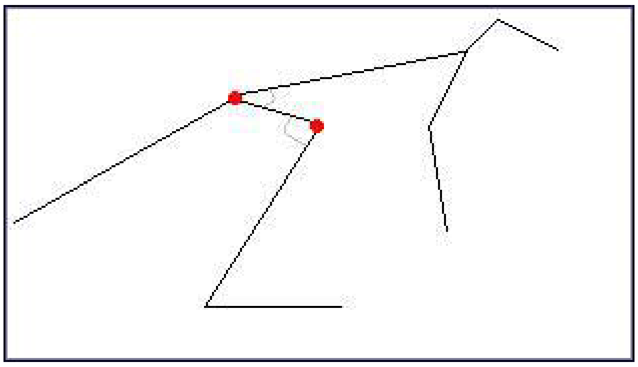


Figure 4.5.1a: The basic kangaroo structure, the highlighted joints are those that will be used to control and produce the motion. The other joints will remain at a constant angle, however this can change if the force required to maintain the angle exceeds the maximum force allocated to the angular motor.

Unlike the simulation detailed in section 4.4.3 the angles of the joints can be controlled more accurately. The previous simulation had the force oscillate from a positive force, through 0, to a negative force. This resulted in the joint completely closing once every cycle of the sine wave. For the basic walking simulation this is not required, instead the angle of the joint must oscillate through a specific range, for example between 110 and 160 degrees.

The angles that were used for the joints highlighted in figure 4.5.1a were obtained by manually adjusting the joints through the program. Once the motion range of the joints had been obtained, an algorithm was constructed in order to oscillate through that range of angles.

The algorithm produced for both the joints is shown below. Although both joints vary at the same rate, the middleLegStep was initially defined at 270 degrees, whereas the topLegStep was defined at 0 degrees. This was done so that when the upper joint was at its smallest angle the middle joint's angle would begin to increase, pushing down on the ground.

```
topLegStep += 0.25;
if(topLegStep > 359.0)
{
      topLegStep = 0;
}
jointCtl[11] = -145 -70*sin(topLegStep);

MiddleLegStep += 0.25;
if(MiddleLegStep > 359.0)
{
      MiddleLegStep = 0;
}
jointCtl[12] = 140 -60*sin(MiddleLegStep);
```

The change in angle of the joint needs to be recorded initially, in order to be able to construct the formula used to move the joint to each extent of the motion. For the upper leg it was discovered that an angle of -215 to -75 would be sufficient. The angle range is 140 degrees, and the sine curve is used to create an oscillation from -70 degrees to +70 degrees. This is then offset by subtracting -145 from the final value resulting in an oscillating measurement, between -215 and -75. This is then stored into the control value for that joint, and used to position it.

The same technique was applied to the lower leg but instead a range of 80 to 200 degrees was used.

As shown in figure 4.5.1b the motion produced is still a long way from a complete hopping structure, however it is a step closer to the final product. The kangaroo structure has no centre of gravity awareness at this stage, and as a result it is difficult to produce a fully balanced model. To overcome this, the kangaroo's tail and arm were lowered to act as stabilisers to stop the model falling forward or backwards.

Currently the motion sequence produced only lifts the kangaroo off its 'stabilisers' for a very shot amount of time, only enough for the slightest movement forward.

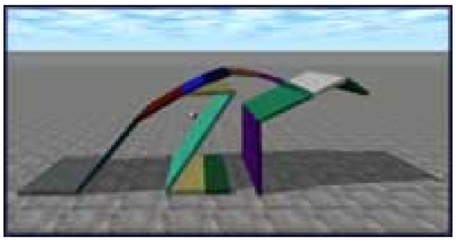


Figure 4.5.1b (a)

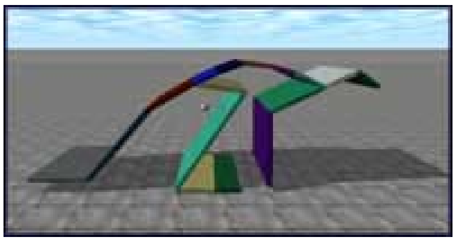


Figure 4.5.1b (b)

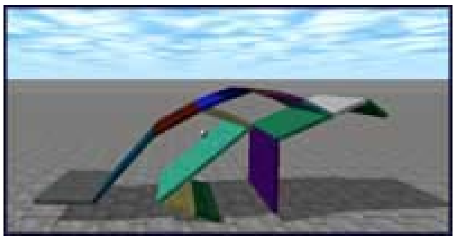


Figure 4.5.1b (c)

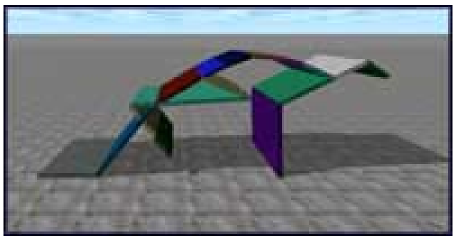


Figure 4.5.1b (d)

The simple kangaroo motion achieved using only sine curves to vary the desired angle on two of the kangaroo's joints. The tail and arm act as stabilisers to stop the model falling forward or backwards. This however is only a temporary measure until a means of balance is implemented.

Although the model gradually creeps forward it is far from a hopping simulation, however this step has brought to light a few more problems. The most evident of which is the reaction between the structure and the ground. It appears to be too solid, and too elastic. As soon as the structure makes contact with the ground the contact is very unrealistic, and frequently results in the model launching into the air uncontrollably.

5 Testing

Each simulation was run individually at each stage of the project's development. This made it possible to spot problems in the development stages before the simulations become to complex.

5.1 Testing the bouncing ball simulations.

The simulations for the bouncing ball models were tested both visually and mathematically. This allowed an easy comparison of results, to ensure that the simulation was not losing accuracy. It was found that the simulation of the vertically bouncing ball lost a very small amount of accuracy after every collision; this was due to the way collisions with the ground were detected. The collisions were checked by comparing the lowest point of the ball to that of the highest point of the ground. When the ball's lowest point was less than that of the ground's highest point a collision had occurred. Unfortunately because computers cannot be infinitely accurate there will be occasions when between time steps the collision occurred. This illustrated in figure 5.1a, where the time step containing the actual collision is missed.

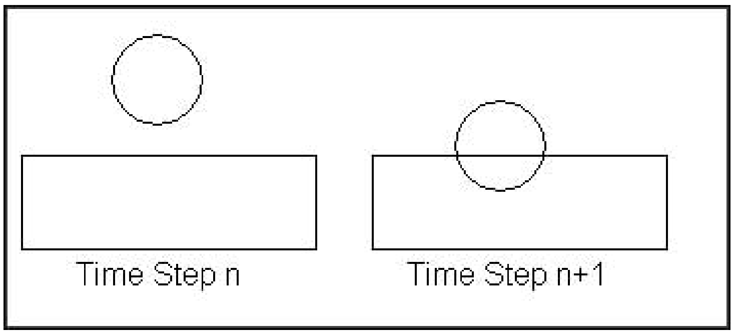


Figure 5.1a: Somewhere between the time step n and n+1 the collision has occurred, however as the simulation update rate is determined by that of the CPU clock, the increments can sometimes be too large.

The ball was dropped from a test height of 100m, with gravity set to that of earth's (-9.81ms^{-1}) and the collision was set to be elastic. With the collision set to be elastic the ball should bounce back to the start height of 100m, and any variation in the maximum height would be due to computer inaccuracy. It was calculated that an average distance of 0.000001m was lost after each collision. This value was negligible and would not effect the simulation significantly enough to justify writing a more accurate method.

The current method as previously described checks if the ball's lowest point is equal to or less than the grounds highest point. When a collision is recorded, the velocity of the ball is recalculated at its current position.

There were some simple ways to combat the problem of calculating exact collisions, however it was decided that none of them actually solved the problem, they instead provided merely ways of disguising the errors [22].

When testing the projectile based model, the same inaccuracy was observed as that of the vertical based simulation. The accuracy loss was deemed negligible and did not affect the workings of the simulation.

5.2 Testing ODE based simulations

Testing the ODE based simulations could not be handled in the same way that the bouncing ball simulations were. Because the ODE libraries handle a lot of the physics calculations it is impossible to do mathematical based comparisons to that of a paper solution.

As the ODE libraries have been tried and tested through many stages in development, it is assumed that any problems or instabilities observed are due to the program implementation, rather than the libraries.

5.2.1 Testing the two segment bone structure

This stage of development was omitted from the project after it was concluded that it did not contribute anything to the development of the final solution. For more details regarding why it was omitted please see section 4.3 of the report.

5.2.2 Testing the three segment bone structure

This section of the development was broken down into three smaller sections, each of which builds upon the previous.

The three section structure that holds an angle was tested by running the simulation and observing whether the structure successfully maintained the angle. The simulation was run and the structure was erected and maintained, although a slight vibration was noticed on the structure.

This vibration was due to the force being constantly supplied despite the angle having been met. Originally intended was to have the force move the joint to the correct angle, and then the joint can sustain a zero velocity at that position. However it was noticed from testing that the force needed to be disabled once the angle had been reached, and only re-enabled should the angle of the joint change.

The second stage of the three segment structure was to add some motion; this was done using sine waves as discussed in section 4.4.3. Once the forces were set to oscillate the first thing that was observed was that the structure was hopping the wrong way. To try to combat this initially, the angles at which the joints would oscillate between were reduced; however this just reduced the rate of movement, rather than the direction.

For the time being the simulation was tested despite travelling in the wrong direction to ensure it was stable, and that the simulation was consistent.

It was later discovered that the reason the structure was travelling in the wrong direction was due to the mass distribution of the structure. This was addressed in the hopping three segment structure that travelled in the correct direction.

The first attempt of adjusting the three segment structure was to extend the length of the uppermost segment. This would move the centre of gravity forward; a great deal of testing was required to get this adjustment to work correctly. As having it too long resulted in the structure falling forwards, and failing to lift itself off the ground, whilst if too short, and the structure would hop backwards.

As now a forward hopping structure was completed, an extra segment was added to simulate the main body of the structure, leaving three segments as the leg. This was extended a little forward, however to move the centre of gravity forward it was given an increased mass compared to that of the other segments.

These simulations, although complicated to implement, were relatively simple to test, as there was no set motion to adhere to. However the interesting results will come from the kangaroo shaped model, where a more accurate approach to the simulation is required.

5.2.3 Testing the Kangaroo structure

While testing the structure that was based around a real kangaroo skeleton several problems were encountered. Most of these problems existed in the previous simulations however they were not nearly as evident, and often as a result overlooked.

Briefly discussed in section 4.5 was the problem that when the simulated kangaroo landed, there was no acting suspension within the limbs. This made the model appear very rigid and often led to it producing inaccurate motion (figure 5.2.3a). The reason for this has been established, however an appropriate solution is as yet to be implemented. Currently an angle is obtained through the use of an angular motor with a specified maximum force. The problem arises because this maximum force is currently constant, which means that no matter what stage in the motion, the angular motor will always try to sustain the angle with the maximum force available. In a real situation the joint would cushion the fall, acting as a shock absorber. Although it may well be possible to fall from a height and maintain a locked joint structure, it is avoided to reduce potential injury. The kangaroo benefits from this aspect of "cushioning" and up to 41% of the energy from each hop is elastically stored for the next hop [5-8].

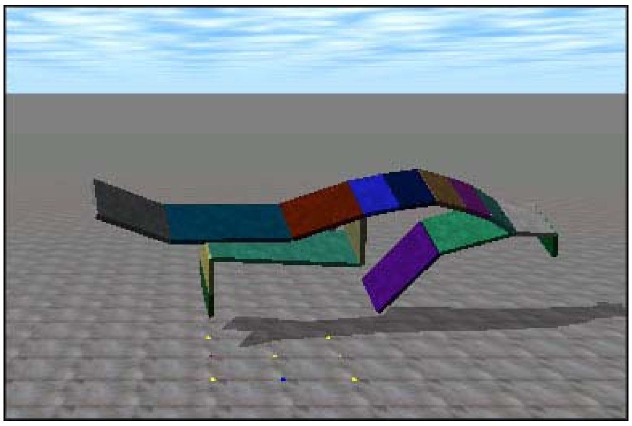


Figure 5.2.3a: The structure is very rigid, and often falls over.

An attempt was made to try and reduce the rigidness of the structure by setting a variable maximum force for each joint, by having it changed during significant parts of the kangaroo's motion. This approach unfortunately yielded unsatisfactory results, and often resulted in the structure collapsing under the strain of its own weight.

A collision between the kangaroo structure and the ground was very hard and often did not produce the motion that would have been observed in a real collision. This is because ODE uses hard contacts, which means a special non-penetration constraint is used whenever two bodies collide [11].

The second algorithm written to move the joints to their desired positions (covered in section 4.5) has a small bug. It appears that the ODE environment uses a degree angle range of -180 to +180, rather than the typical 0 to 360. As a result it has been observed that when crossing the point where -180 degrees equals +180 degrees the algorithm fails. The algorithm is designed to move to the angle required with the shortest amount of time steps. As a result when moving from an angle of +170 to -170 where the difference is 20 degrees, the algorithm calculates the difference as 340 degrees and takes the longer route to satisfy the angle. This causes significant problems when producing the kangaroo's motion, and in order to continue to produce an accurate representation of the motion this problem will need to be addressed.

When the basic motion simulation was implemented as described in section 4.5.1 it was noticed that the joints often did not move between their full extents. Using the sine curve to oscillate the angles of the joints did not produce the results that were expected. The angle that was being oscillated was not the angle of the joints; instead it was the desired angle of the joint. On some occasions the angular motor is unable to move the joint quick enough to the desired angle before the angle is changed to the next angle in the sequence. To overcome this problem the velocity of the angular motor was increased so that it would reach the angles quicker; however this caused instability problems once the angle was met, often causing the joint to resonate.

It would be possible to implement a way of making the angular motors operate at a non-constant velocity, unlike their present configuration. If the angular motors were configured to accelerate and decelerate, this could make the motion appear more realistic. It would need to accelerate initially then decelerate as the desired angle is approached, making the motion appear a lot smoother.

After much consideration it has been concluded that to solve some of the previous problems a controller would be needed for the model. This would be able to calculate joint velocities, angles and forces dependant on the models current configuration, and what motion it was performing.

Already discussed are the current problems in the simulation with regards to the structure and its configuration. However there are some problems which are related to the stability of the ODE libraries, rather than that of the model itself.

After testing the simulation on four different computers it was observed that the simulations ran at different speeds. This would simply be due to the limiting CPU speed of the testing computer. Simulations written using ODE use a timeStep variable to determine the rate at which the simulation is updated. To make the simulation update at a fixed rate regardless of computer speed the time taken to perform each update was recorded. This value was then used as the timeStep, and varied from pc to pc. After some discussions with some developers from the ODE mailing list it was later discovered that this still does not produce a consistent simulation across computers of varying speeds. It was advised that using a varying timeStep value makes the simulation very unstable, and can often cause problems when the timeStep becomes too small or too large depending on CPU activity on the computer.

A new algorithm has been developed that runs consistently across computers of varying speeds, however this has only been implemented into the forward hopping three section structure. This is because once the timeStep is fixed and set to update at a frequency of 10 Hz, the inaccuracies in the simulations become more evident. The algorithm that runs the simulation at consistent speeds is detailed below.

```
static void simLoop
{
      clockInc = (clock()-clockLast)/CLOCKS_PER_SEC;

      // Make sure time is accrued properly for the phys step count

      totalTime += clockInc;

      // Then advance the world up to present time

      while(totalTime > stepValue)
      {
            // SIMULATION OPERATIONS

            dSpaceCollide (space,0,&nearCallback);

            dWorldStep (world,stepValue);

            totalTime -= stepValue;
      }
```

```
        // DRAW OBJECTS TO WORLD

        clockLast = clock();
}
```

If the simulation simLoop method is run less frequent than 10 times a second the algorithm ensures that the simulation is kept up to date. For each simLoop the worldStep and spaceCollide typically will be called only once, however if the simulation is running slow they are called more frequently. If for example the simulation is running three times to slow, the while loop will loop three times until the simulation has caught up to real time.

As now the simulation is running in real time, it is now needed to use real world values, rather than the ODE default units. The first thing that needed to be corrected to real values was gravity as currently this was set to -0.00981 ms^{-1} in accordance with the scale of the current model, however as now the system is running with world values it needs to be adjusted to -9.81 ms^{-1}. The weight of each of the segments was given a density of water, and the weight was calculated based on each segments dimensions.

The last observation made from testing was the instability of ODE when handling multiple collisions simultaneously when either a complete surface has collided with another parallel surface or the structure has fallen onto its side. When either of these situations occurs the model appears to adopt a life of its own and launch into the air uncontrollably (figure 5.2.3b). After some discussion with the developer of ODE, Russell Smith, the reason this occurs is because the simulation is too simple. To overcome this problem it has been advised to make a large number of dummy joints in order to slow the simulation down and increase its complexity.

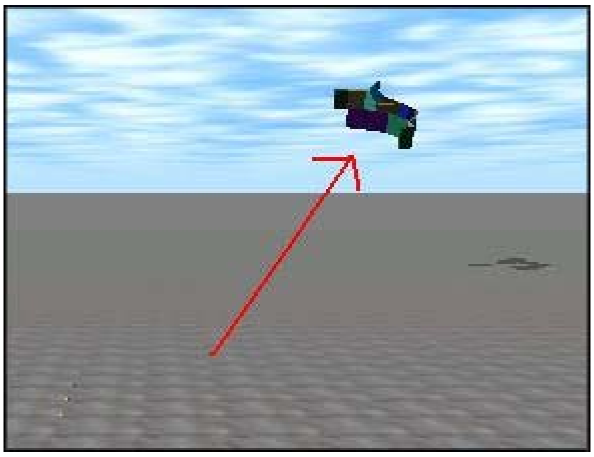


Figure 5.2.3b: If the structure fell onto its side the simulation became unstable. This resulted in the structure launching into the air uncontrollably.

6 Evaluation and Conclusion

As originally discussed in the provisional proposal the objective to produce a dynamic model of a kangaroo based solely on physics is a big task. It was never expected that a complete model of a kangaroo would be produced, but more of a significant step towards the solution.

The project has satisfied the original requirements of producing physics driven animation which uses neither key frames nor motion capture. Although the first two simulations were written directly in OpenGL and C++ they were still vital to the understanding of how to develop a physics driven VE. However once the break through in understanding how to implement ODE driven environments it was clear that all future simulations should take advantage of the ODE libraries.

The original work plan discussed the construction of structures with varying numbers of segments. This aspect of the work plan was followed accurately, and as a result has provided some very useful simulations that will be of great use should someone choose to continue the research.

The project did not reach the stage where a fully functioning kangaroo structure had been constructed. As a result, a lot of the information obtained through the literature survey was not implemented into the simulation. The information however will be required once a stable kangaroo model has been implemented.

Although the research surrounding kangaroo motion and producing physics based models of kangaroos is far from complete, the developments discussed in this report are a significant step towards accomplishing these goals.

If someone were to continue the research it would be clear to see what had been achieved and why from this report. As a result it will be possible to resume the research, and continue along the same lines of development.

The most significant success in the project was the implementation of segmented structures developed using the ODE interface. Although it took a while to learn how to use the ODE libraries, once this had been accomplished physics based simulations started to take shape.

Currently, the desired angles for the motion will never be correctly reached because of a lack of a controller. A controller minimises the error between the actual and desired angle by adjusting the input variables (force and corresponding velocity). The lack of force-driven (and corresponding acceleration; i.e. Newton’s second law) segments makes the movement unrealistic.

Because of the absence of a controller, the model will not be able to balance, unless some support is used (e.g. front arm and tail). However, this will make the simulation look unrealistic

To continue work in this field it is vital that the problems identified in the testing section of this report are addressed. The next step would be to develop a feedback control system that could be used to maintain balance, as well as position the joints more accurately. Once a control system is added it will be possible to incorporate a more complicated motion sequence, which will eventually lead to a complete hopping kangaroo model driven entirely by physics.

References

[1] - Pullen K, Bregler C, Motion Capture Assisted Animation: Texturing and Synthesis, Stanford University. Reference paper located at: http://graphics.stanford.edu/~pullen/papers375.pdf [Last accessed 26/04/2005].

[2] - University of Wisconsin, Animation lecture, CS559 Fall 2004. Lecture slides located at: http://www.cs.wisc.edu/~schenney/courses/cs559-f2002/lectures/cs559-28.ppt [Last accessed 26/04/2005].

[3] – About.com, 'Key Frame' definition. Located at: http://animation.about.com/od/glossaryofterms/g/keyframe_def.htm [Last accessed 26/04/2005].

[4] – Answers.com, "Motion Capture" definition. Located at: http://www.answers.com/topic/motion-capture [Last accessed 26/04/2005].

[5] - Dawson TJ, Taylor CR. Energetic cost of locomotion in kangaroos, Nature (London) 1973;246:313-4

[6] - Hoyt DF, Taylor CR. Gait and energetics of locomotion in horse. Nature (London) 1981;292:239-40

[7] – Dimery NJ, Alexander RM, Elastic properties of the hind foot of the donkey, (Equus asinus), J Zool (London) 1985;207:9-20

[8] – Dimery NJ, Ker RF, Alexander RM, Elastic properties of the feet of deer (Cervidae), J Zool (London) 1986;208:161-9

[9] – Dawson TJ and Bolliger E, Work capacity of the red kangaroo heart, School of Biological Science, University of New South Wales, Australia. Last accessed 26/04/2005. Reference paper.

[10] – Farley CT (1), Glasheen J (2), McMahon TA (3), Running Springs: Speed and Animal Size, (1) Museum of Comparative Zoology, Concord Field Station (2) Department of Orgasmic and Evolutionary Biology, Harvard University (3) Division of Applied Sciences, Harvard University, 1993;185:71-86.

[11] – Smith R, Open Dynamic Engine, V0.5 User Guide, May 2004. User guide located at: http://www.ode.org/ode-latest-userguide.html [Last accessed 26/04/2005].

[12] – Kram R (1) and Dawson TJ (20 , Energetics and biomechanics of locomotion by red kangaroos, (1) Department of Integrative Biology, University of California, Berkeley, (2) School of Biological Sciences, University of New South Wales, Sydney, 1997;120:41-49.

[13] - Macr H. Raibert (1) and Jesssica K. Hodgins (2), Animation of dynamic legged locomotion , (1) MIT leg Laboratory (2) IBM Watson Research Center. Pages 349 – 358, Computer Graphics, Volume 25, Number 4, Julv 1991.

[14] - Greatorex CL, Simulation of animal motion in a virtual environment, Research Project 2004. University of East Anglia, Norwich, Norfolk, England.

[15] - Modelling projectiles, located at: http://www.makingthemodernworld.org.uk/learning_modules/maths/04.TU.02/?section=1 [Last accessed 26/04/2005].

[16] – Tyler F, Mechanics and properties of matter, Published by London: Edward Arnold, 1961.

[17] – French A.P, Newtonian mechanics. Published by W. W. Norton & Company, New York, (March 1, 1971). ISBN 0393099709.

[18] – ODE Physics engine, www.ode.org [Last accessed 26/04/2005].

[19] – ODE Mailing list, www.ode.org/mailingList [Last accessed 26/04/2005].

[20] – Gotfootage.com, kangaroo motion located at: http://www.gotfootage.com/gf/store/node/viewer/type/quick/qs/Austral*/pg/3 [Last accessed 26/04/2005].

[21] – Skulls Unlimited, Kangaroo skeleton located at: http://www.skullsunlimited.com/graphics/kangeroojv.jpg [Last accessed 26/04/2005].

[22] – John Vince, Virtual Reality Systems / John Vince. Published by Harlow: Addison – Wesley, c1995. ISBN 0201876876.

[23] – Alexander, R, McN. 1988, Elastic Mechanisms in Animal Movement (Cambridge University Press: New York).