RAG-DOLL IMPLEMENTATION IN MAYA

MASTERS THESIS

SANJANA RAJENDRAN
N.C.C.A BOURNEMOUTH UNIVERSITY
September 5th, 2005

Contents

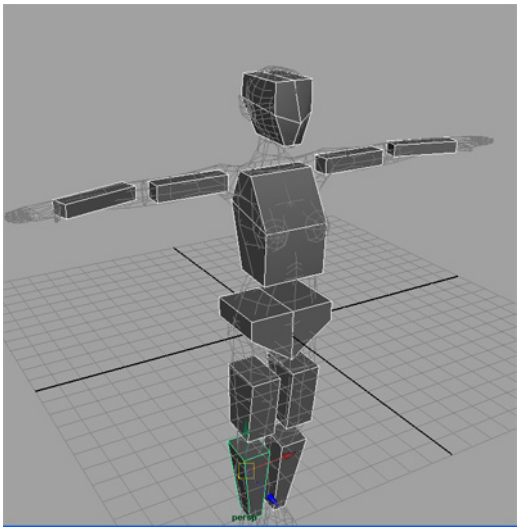
LIST OF FIGURES

1.INTRODUCTION

Dynamic Animation is a new area of Computer Animation that is being increasingly used in games and visual effects. It results in natural, realistic-looking motion of characters, which is difficult to achieve by traditional key framing. The human body is complex and its tough to anticipate and reproduce and reaction of human beings to forces and objects, by just key framing alone. So lately the CG community has started to look at the possibilities of using physics in character animations as well. This has resulted in something the game developers refer to as *rag doll behavior*.

Rag doll, as the name implies, is the ability to let the characters be influenced by the surrounding environment while apparently limp themselves, like rag dolls. The rag-doll by itself does not produce any motion, but under the influence of forces and fields, it reacts in a way a normal human would. rag doll behavior does not imply controlling the user controlled characters with physics, it merely concerns simulating physical effects on lifeless characters or lifeless limbs. Applying constraints to the rag-doll to specify how a normal human would or wouldn't move does this. These simulations can however be mixed with existing animations. The main area of research however is the subject of shifting the character control from the animations system to the physics system.

Figure 1.1:A basic rag-doll(2)

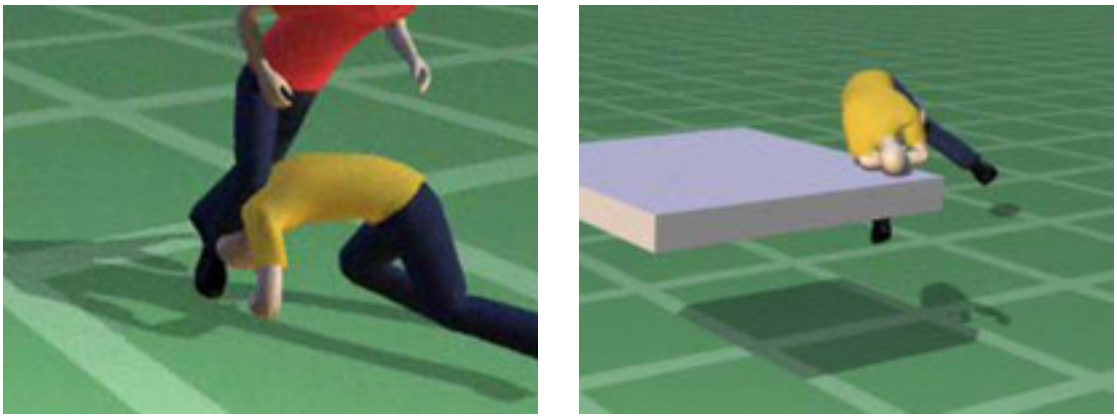


The chief advantage ragdolls offer over traditional animations is that they allow much more correct interaction with the surrounding environment. Where it would be absurd to try and hand-craft custom animations for all conceivable circumstances, ragdolls fill in and generate a mostly-correct interpretation of events on fly.However,the disadvantage is that since real-time physics simumlations are computationally intensive,very simple structures are used to approximate the characters and therefore small parts like fingers,toes etc often go unsimulated.

There are software like Endorphin which use Artificial-Intelligence driven behavior routines and laws of physics to determine how the characters would move, to dynamically animate human characters. Endorphin allows an animator to import

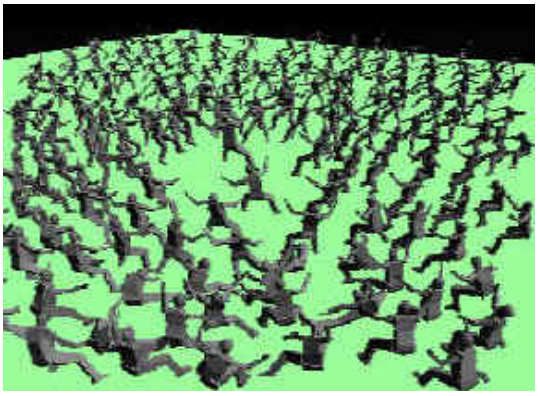
animation or motion capture data, after which, the animator can override behavior of that character at any moment and let AI behavior take over. Other packages like Massive also feature dynamic animations called 'SmartStunts'. With Smart Stunts it's possible for filmmakers to direct the motions and possible reactions they want for virtual actors ahead of time with their real stunt actors. Then, all of the reactions that follow in Massive are reactions that are completely in character. In addition to human stunts, Massive allows completely arbitrary, non-human-specific skeletons and the ability to animate a large number of things interacting realistically. There are also various ways of implementing ragdoll models, which are described later on.

Figure 1.2:'SmartStunts' in Massive(3)



The inspiration behind the project arose on seeing a web page, which featured the use of Softimage and Behavior to animate a group of characters responding to a blast wave. The project used state machines in Behavior to use AI driven routines to animate the characters dynamically. It was then decided to implement an identical scenario in Maya, where a character could react to a blast wave dynamically. This led to the use of rag-dolls to turn the character dynamic. The challenge lay in creating a mel script which could automate the process for the user and convert a human skeleton into a physically-simulated ragdoll. This script also sets the explosion fields according to the user's requirements to dynamically animate the character. Another challenge was also simulating the effect of the blast wave. Various fields were tested on the character and finally it was decided to use the gravity field itself, in different directions. Also to give it the momentum and effect of a blast wave, an object simulating the blast wave was made to collide with the Rag-doll.

Figure 1.3:Dynamic animation of a blast wave(4)



2.PREVIOUS WORK:

The traditional method of creating rag-dolls involves creating rigid body object for each of the main bones and connecting them by hinge constraints to simulate a skeleton. One of the drawbacks of this method was that some kind of additional constraining had to be added as the rag-doll proved too flexible and inhuman-like.

2.1.OTHER PSEUDO RAGDOLL CREATION METHODS

Some of the other methods of creating 'pseudo rag-dolls' that have been used are as follows(5):

- Verlet integration: used by *Hitman: Codename 47* and popularized by Thomas Jakobsen, this technique models each character bone as a point connected to an arbitrary number of other points via simple constraints. Verlet constraints are much simpler and faster to solve than most of those in a fully modelled rigid body system, resulting in much less CPU consumption for characters.Verlet integration is also popularly used as a velocity-less Cloth Simulation technique.This technique is most popularly used in the games industry to achieve the ragdoll effect.

- Inverse kinematics post-processing: used in *Halo*, this technique relies on playing a pre-set death animation and then using inverse kinematics to force the character into a possible position after the animation has completed. This means that, during an animation, a character could wind up clipping through world geometry, but after he has come to rest, all of his bones will be in valid space.

- Blended ragdoll: this technique works by playing a pre-made animation, but constraining the output of that animation to what a physical system would allow. This helps alleviate the ragdoll feeling of characters suddenly going limp, but offers correct environmental interaction as well. This requires both animation processing and physics processing, thus making it even slower than traditional ragdoll alone

The former was finally chosen due to its computational effectiveness.

David Basalla, a fellow Msc Computer Animation course mate, had previously implemented a similar rag-doll model in Maya. One problem faced by the model was that due to adequate constraints, the resulting motion was sometimes a bit too flexible. So one of the challenges was adding more physical constraints including those for the elbows, feet and hands to prevent the body from getting too flexible in unrealistic directions .The constraints act as a 'check' and perform the same role as that of an elbow or an knee in the human body.

Also subtle differences in the proxy objects were made to improve the performance of the rag-doll. For example, the rag-doll was constructed so that the different cubes for the different bones were of different size, true to a real body. Also the human body retains its balance due to a certain balance in the masses of the different parts. For example, if the head were heavier compared to the body, we would all topple over. Since, the torso and the pelvis are the heavier parts, and the hands, legs and had lighter, the center of gravity remains within the pelvis that enable us to function normally. So masses were assigned to different parts of the body according to its size and weight, to give more of a sense of balance.

3.TECHNICAL BACKGROUND:

Mel (Maya Embedded Language) was chosen to implement the project as it not only had a complete range of commands but also had a wide range of user-interface creation commands. Mel is a relatively simple programming language, similar to C, controls all of Maya's action right from creating an object to selecting something. In fact, Maya's dialog windows itself are written in Mel. Typical cases where Mel is used are in cases, which involve a repetitive use of some action or procedure. A simple script could be written for the same and the task is accomplished by the click of a button. Therefore, every action performed by an user is a Mel command and the result of the same is stored in the script editor. Once the script has been written and compiled, it can be stored on the window shelf enabling easy access for the user each time the script needs to be run.

Most of the research in Dynamic Animation has been in creating different types of dynamic systems, saving computational time and also including mocap and extending that into dynamic animation.

The sigraph paper "A Particle Dynamics System" by Zoran Ka ci¥c-Alesi Marcus Nordenstam David Bullock of Industrial Light and Magic provided an insightful look into the other methods of creating pseudo ragdolls. In this method, they create each bone as a Mass-Spring model, consisting of numerous particles connected by springs and use an integration technique to detect the movement of the character. According to the paper

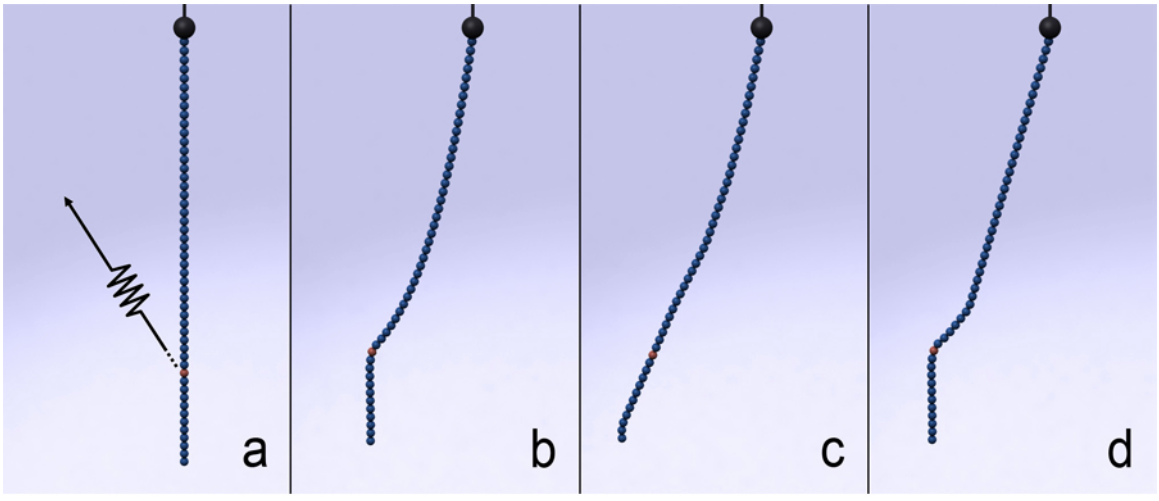
"Our system is built on a spring-mass model. At the lowest level, the solver sees a world consisting of a potentially large number of point masses, or centers of mass for rigid bodies, connected with springs in an arbitrary configuration. While this may not be the most accurate model of the world, it is simple and it works surprisingly well for many applications. At the application level, the dynamics system is a module in a general 3D modeling and animation system. At this level the world, or the scene, as we often call it, consists of many complex

polygonal or B-Spline surface models animated using a combination of key frame animation, procedural methods, deformations, simulation, and motion capture. The user has complete control over every attribute that describes the objects in the scene, their motion, and interaction ñ there is a graphical user interface for interactive control and a scripting language that can achieve the same programmatically".(1)

The paper quotes very good results using the Verlet Integration technique, which is widely used in Molecular and Dynamic animation.

The paper "Adaptive Dynamics of Articulated Bodies" by Stephane Redon Nico Galoppo Ming C. Lin, Department of Computer Science, University of North Carolina, published in this year's sigraph also offers an interesting observation. This techniques saves computational time by not simulating all the joints at all time and by choosing only certain joints that contribute the most to the current state of the articulated body .The user can choose the number of joints he wants to simulate depending on the accuracy desired, distance from the camera etc. So in cases where there are a large number of characters very far away from the character, the joints in the fingers etc can avoid getting simulated, thus saving computational time.

The next figure shows an example of the adaptive selection of the set of active joints: **(a)** one of the links of a 50-link pendulum is attached to a point in the environment through a spring; **(b)** equilibrium state when 50 joints are active; **(c)** equilibrium state when 5 joints are active, without adaptive determination of the active joints (breadth-first selection); **(d)** equilibrium state when 5 joints are active, with adaptive determination of the active joints by the adaptive dynamics algorithm. (6)



Adaptive determination of the active region

The following figure shows a complex scene, where 200 human characters, represented by 17,800 rigid bodies and 19,000 degrees of freedom, are suddenly pushed away from the camera due to applied forces. The adaptive dynamics algorithm allows an animator to progressively reduce the

number of simulated joints in the characters as their distance to the camera increases, A in order to improve the efficiency of the dynamics simulation.(7)

Adaptive dynamics of articulated characters

4.SOLUTION

The goal of the project was to convert a joint skeleton into a ragdoll and allow the user to dynamically set a blast wave to simulate an explosion scenario. The focus was more on realistic, human-like motion of the ragdoll rather than extending it for a crowd scene, which was not possible due to time constraints.

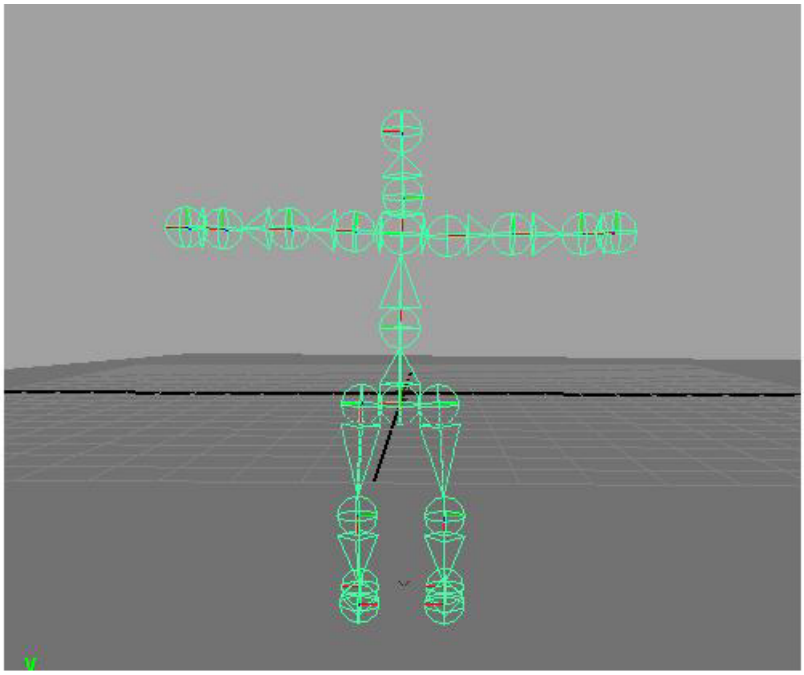
There were two specific parts to the problem. They were:

- Creating the dynamic skeleton
- Creating the effect of the blast wave.

4.1.CREATING THE DYNAMIC BODY

The body was created from 'proxy objects', which were created according to the size of each part of the human body. The proxy objects are then aligned to the skeleton by using Maya's 'orient' command. The body was created and tested on a single skeleton and is specific to that skeleton. One of the problems was trying to make a script flexible enough for different skeletons, but due to time constraints, and the difficulty in achieving this, a single skeleton was chosen and all the tests done on that.

Figure 4.1:The test skeleton used

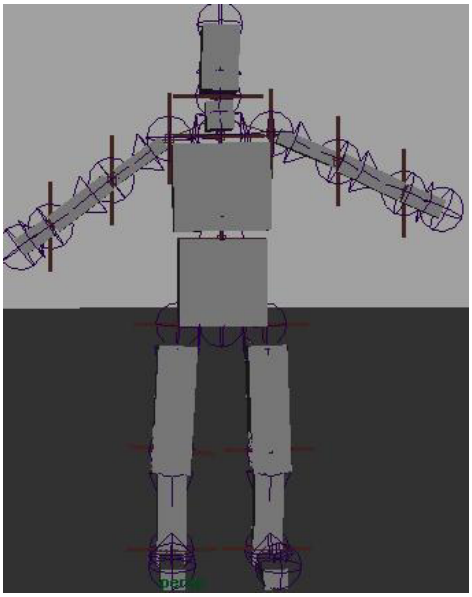


The main procedure in creating the body was traverse (string $joint, string $group name). This procedure traversed through each joint in the skeleton and did the following:

- Checked if the joint was the root, if so, it turned on a root flag.
- If not, calculated the position of the joint and the position of the next joint.
- It then calculated the mid point between the 2 joints.
- The proxy object cube was then created according to the joint and scaled according to the distance between the two and placed at the midpoint of the joints and oriented according to the joint.
- Any extrusions for the knees or elbows were done depending on where it was necessary
- A parent constraint was then done for the joint and the corresponding proxy object using Maya's parent Constraint command. This command is very convenient as it has a flag called weight that can be 1 or 0 depending on whether the constraint is on or off. The constraint is initially set to 0 so that the skeleton follows the proxy objects. Since the proxy objects are active rigid bodies, there react to forces and fields and the skeleton in turn moves with the proxy objects, dragging the character mesh along with it, if any.

- A counter was increased and the function was called again and the same procedure was repeated for the next joint .

Figure 4.2:The complete ragdoll



To traverse among all the joints in the skeleton and add distinct attributes, the joints were then grouped using a group () procedure, which uses Maya's group command to create an empty group and add the attributes Width, Thickness, Length and State. The width, thickness and length were provided to adjust the scale of the proxy object according to the size of the character. Also an attribute called State was added so that the user can easily shift between the normal animated state and the rag-doll state.

The proxy objects then needed to be connected by Hinge constraints. This was done by the procedure create hinges (string $jointname, string $polyname, string $rootname), where $jointname was the name of the specific joint, $polyname was the name of the block, and $rootname was the name of the root joint. These constraints act like the joints connecting the bones in the human body. The then needed to be rotated into place so that they allowed the bone to bend in the right direction. They constraints act similar to a door hinge. Sometimes there was a problem with the constraints of the arm not being oriented to the correct angle of 90 degrees, but strangely enough the rotations of the arm seemed to work fine.

It is then required to make the skeleton move according to the proxy skeleton. So a parent constraint command is used to constraint the skeleton to the proxy objects so that it overrides key frames and follows the proxy objects. There command also comes with a 'weight' attribute, which can be turned on or off to enable or disable the parent constraint. This attribute is then connected to the State attribute added while creating the group of joints, for the user to change between ragdoll and normal behavior.

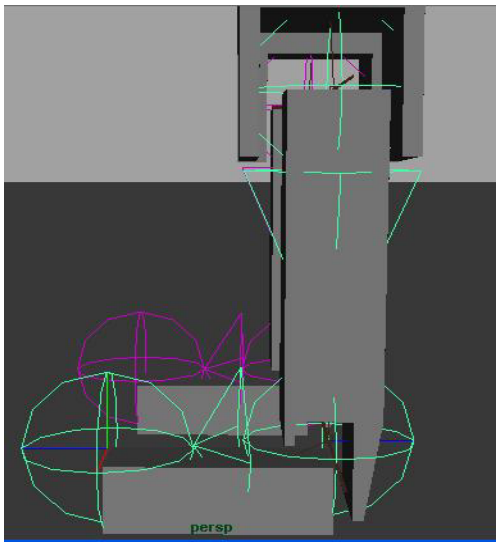
Also, while scaling the blocks, it was required to know how wide the torso and pelvis needed to be. This was calculated dynamically by calculating the distance between the two thigh joints. The torso was made slightly bigger than this distance and the pelvis was made slightly smaller than the torso to give balance to the ragdoll. This was accomplished using the get

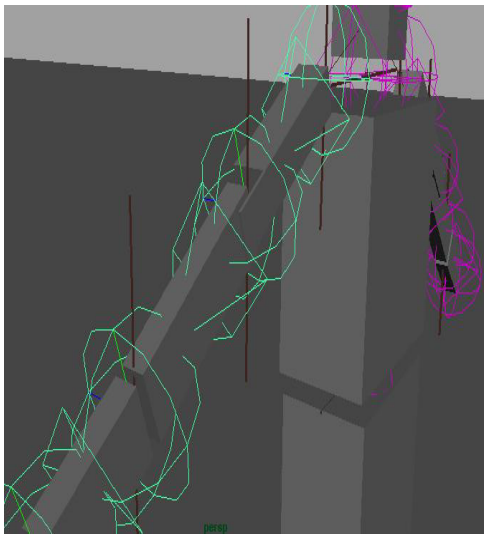
Distance () function, which determines which the thigh joints were, and obtained their world space coordinates and then used distance formula to calculate the distance between the two and returns the distance back to the main function.

4.2.PHYSICAL CONSTRAINTS

A major issue was constraints needed as even at this stage the ragdoll was too flexible and unrealistic and would just topple over given any force. So physical constraints were applied for a number of parts of the body by extruding out surfaces. For example a kneepad and elbow were extruded out to prevent the bones from bending backwards. Also it was found that some kind of constraining was needed for the foot too, as when the character was thrown in an explosion, the foot often bent backwards. So a portion of the ankle was extruded down to prevent the bending of the foot backwards.

Figure 4.3:Physical constraints in the leg and arms





4.3.CREATING THE BLAST WAVE

The primary effect of any explosion is the blast wave that causes a discontinuous effect in pressure, which in turn causes all the destruction that we normally associate with an explosion. So the challenge lay in creating the effect of a blast-wave that would make the rag-doll behave convincingly like it was caught in an explosion.

The obvious choice was a combination of fields, so the rag-doll was tested with radial and Newton's fields. But the resulting motion proved too dramatic and therefore other fields were tested. Finally it was found that by using the Gravity field directions other than -y, the effect of a blast wave could more or less be approximated. However, the effect was still of a person being blown away by a strong gust and not of a blast wave which has a stronger and more sudden impact. So a simple spherical object was made to collide with the rag-doll to provide the impact and momentum that a blast wave has.

A simple user interface was created which allowed the user to set the strength and location of the explosion and also the start animation frame for dynamic animation. The y coordinate of the location was maintained as zero to ensure the explosion occurs on the ground. A ratio was then computed using the strength of the explosion and distance from the character. The idea was that the closer the character was to the explosion, greater would be the impact of the explosion on it. Therefore, the gravity fields were also increased according and the spherical object depicting the blast wave was also scaled to increase the impact of collision. Similarly, the more the strength of the explosion greater is the impact on the character. Thus it was determined that the gravity values should be a combination of strength and distance values. The values x and z directions of the gravity field were thus calculated using a formula

Ratio=strength/distance

Where,

Strength=strength of the explosion as specified by the user, and
Distance=distance of the character from the explosion calculated by Distance formula.

The x and z directions of the gravity field were then calculated using the formula,

Dir x=x/(x+z), and dir z=z/(x+z), so that as the gravity along direction x increases, the gravity along direction z decreases.

A problem was found at this junction whenever the character was not at the origin. Whenever the character was moved away from the origin, the character was pulled towards the location of the explosion. To avoid this problem, gravity fields were calculated as though the person was always at the origin and the characters coordinates along the x and z directions were subtracted from the location of the blast. So the coordinates of the character become

New coordinates of character=coordinates of the blast-old coordinates of the character

This ensured the calculation of the values of the magnitude of the x and z values of the gravity field. The next step was the calculation of the direction, whether the values were positive or negative. For this, a simple algorithm was developed; to determine which coordinate the values belong in and determined the signs accordingly.

The algorithm was as follows:

- It was checked if the values fit in the (x,z) or (-x,z) or (x,-z) or (-x,-z) coordinates.
- If the value lay in the (x,z) coordinate, it was checked if the character lies in the origin. If not, the coordinates of the character were subtracted from the location of the blast to get the new location.
- In the (x,z) coordinate, x and z were simple calculated using the above mentioned formulas. Since both are positive, and the character is imagined to be at the origin, the

effect of the blast wave should be along the opposite direction .So both the signs were reversed.

- In the (-x,-z) coordinate again, the direction would be position and therefore the signs are not changed.
- In the (-x,z) and (x,-z) coordinates the positive or negative sign depends solely on which value is greater. So it is checked if x or z is greater. If x < z then it is checked which magnitude is greater using the abs function, which returns the absolute value. If the magnitude of z is greater, the signs of the z and x values are reversed.
- The opposite happens in the other case where z is lesser. This time again the magnitudes are compared and the signs are reversed only when the magnitude of x is greater than the magnitude of z.

The pseudo code is as follows:

Case 1:

```
If(Location x and Location z >0)
{
Dirx=-(dirx)
Dirz=-(dirz)
}
```

Case 2:

```
If((locationx>0 && locationz<0) or (locationz>0&& location x<0))
{
        If(locationx<0)
        {
                if(magnitude of z > magnitude of x)
                {
                Dirx=-(dirx)
                Dirz=-(dirz)
                }
        }

        else
        {
                        if(magnitude of x > magnitude of z)
                {
                Dirx=-(dirx)
                Dirz=-(dirz)
                }
        }
}
```

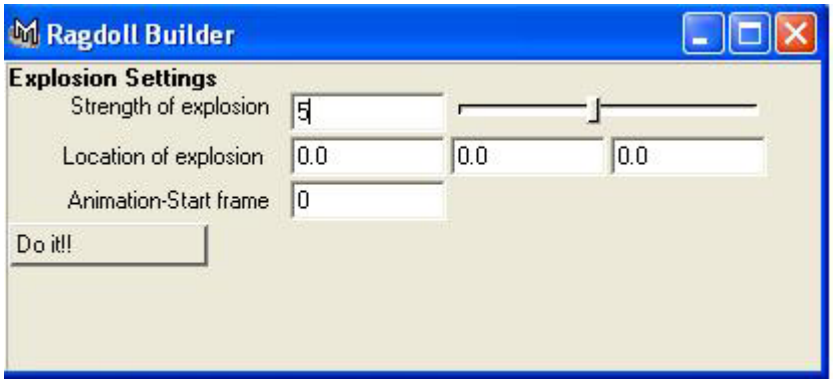
The values of direction x and direction z were then key framed at the start frame providing by user. The state attribute of the ragdoll was also key framed to change from normal to

dynamic mode at the start frame. This enabled the user to animate the character by hand and convert it to ragdoll whenever desired.

4.4.USER INTERFACE

The user interface was kept simple and user-friendly. It simply requires the user to select the root of the joint skeleton, and have a gravity field created called 'gravity' to run the script. On running the script a window opens which allows the user to enter the strength of the explosion, location of the explosion and start frame. There is also a simple procedure called button action (), which transfers the values from the user interface to the program for calculation.

Figure 4.4:The-user-interface window



4.5.PROBLEMS FACED

One of the main problems faced was that sometimes the scene became too much computationally intensive and thus caused Maya to freeze or close. Playing around with the rigid solver settings generally solved this. Another major problem was the placement of the proxy objects. Sometimes the objects would get too close and would cause rigid body interpenetration, again causing Maya to freeze. Testing and re-positioning the blocks exactly so that the physical constraints served their purpose but at the same time weren't too closed to the other blocks to cause collisions sorted out this problem.

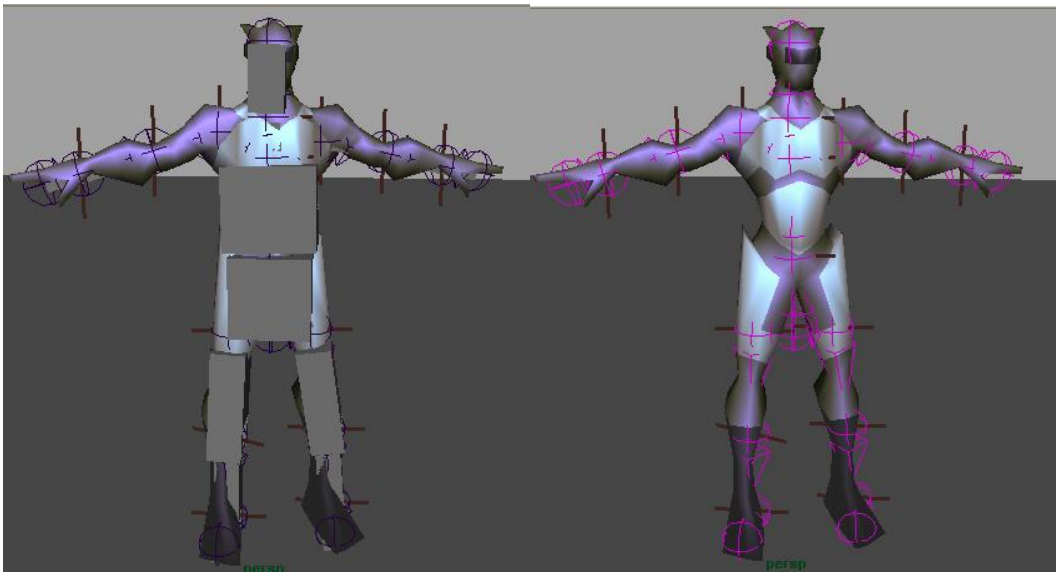
Another difficultly was in making the detection of the joints dynamic which involved multiple conditions in certain cases. These have been done in certain cases, but in certain other cases, due to time constraints, the joints have been referred to by the name. This confines the flexibility of the program to a large extent to the specific skeleton.

Overall, the rag-doll produces realistic and naturalistic motion under the influence of a blast wave. The code could have been more dynamic and could possibly provided for other types of skeletons as well.

Finally, the ragdoll was tested on a character mesh. The skeleton was sized according to the character and a smooth binding was done. The root was then selected and the script runs to

create the ragdoll. The proxy objects of the ragdoll can then be resized according to the character during the width and height sliders. This then converts the character mesh into the rag doll as the mesh is bound to the skeleton and follows the skeleton. The skeleton in turn is parent constrained to the proxy skeleton. So the character mesh too follows the proxy skeleton.

Figure 4.4:Adding a character mesh



4.6.PROCEDURES USED

1. showMyWindow()-creates the user interface window
2. buttonaction (string $start,string $location,string $strength)-passes on the values of strength,location and start animation frame from the window to the program
3. createGroup(string $root)-calls the group function to create the group
4. group()-creates a non-empty group and sets new attributes of width,height and state.
5. blast(float $locationx[],int $strength1,int $startframe)-sets the gravity fields to create the explosion effect according to the strength and location of the explosion.
6. createhinges(string $jointname,string $polyname,string $rootname)-takes two proxy objects and creates a hinge constraint and orients them accordingly.
7. traverse(string $joint ,string $groupname)-This is the main function.This traverses through the group,creates the proxy objects,creates the hinge constraints and parent constraints them to the joint.
8. getDistance(string $rootname)-returns the distance between the 2 thigh joints.

4.7.STRUCTURE

showMyWindow()

⇓

buttonaction()

⇓

blast()⇒createGroup()⇒group()

⇓

createhinges ⇒ traverse()⇐getDistance()

4.8.TESTS:

A number of tests were conducted with the rag-doll itself and also after binding the rag-doll to a character mesh. The dynamic simulation worked quite well and could almost play back in real time. The major force would always be a gravity field but this could be combined with other force fields as well, such as radials and drag fields. Sometimes during testing, the fields would be too strong and the simulation would go wrong and result in a disappearing character. With a character mesh sometimes there were strange deformations because of the lack of proper painting of weights.

Figure 4.5:Rendered images of ragdoll animation

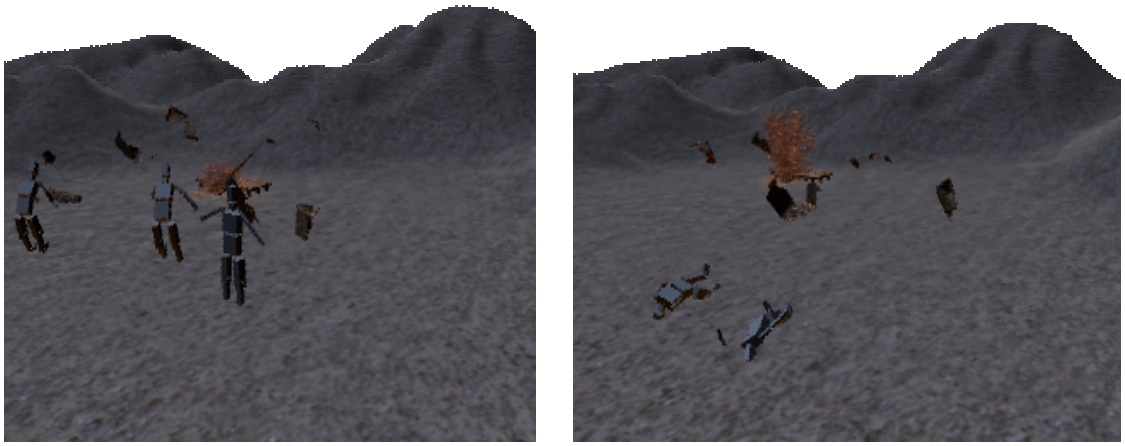
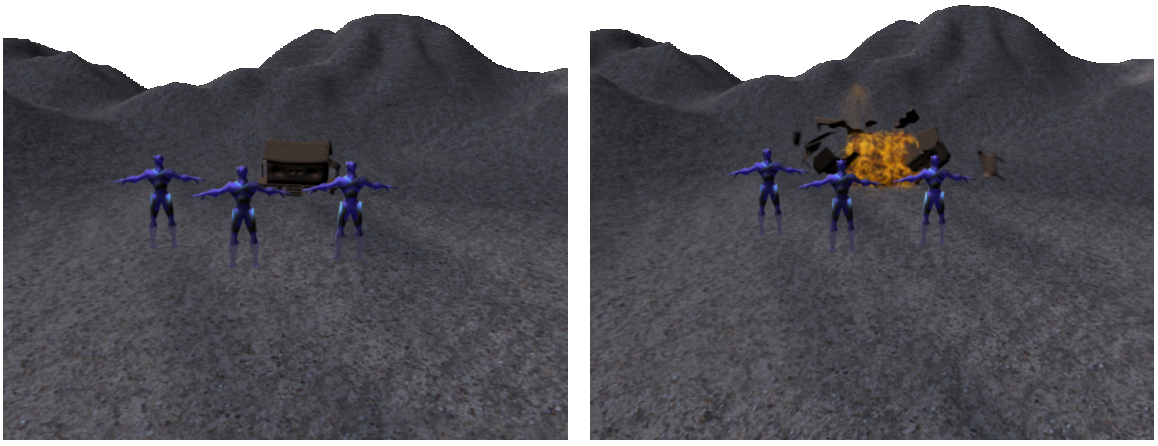
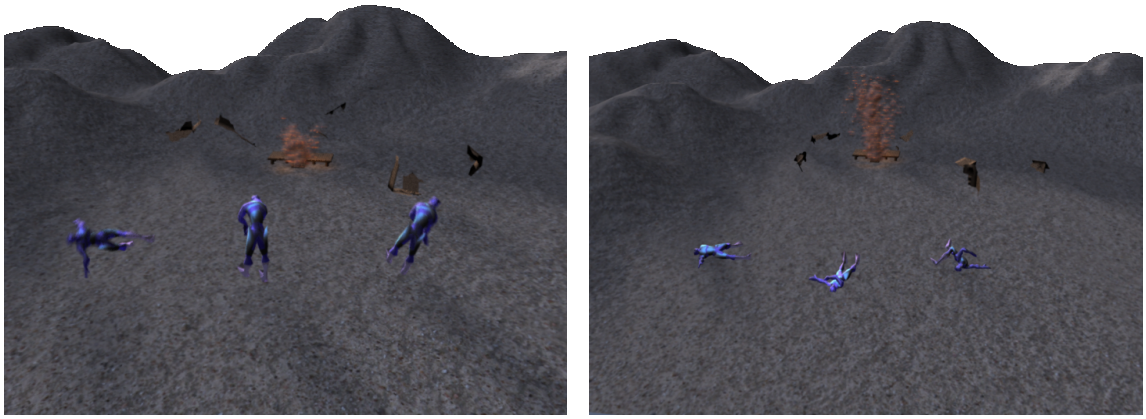


Figure 4.6:Rendered images with character mesh





4.9.KNOWN BUGS:

The entire programming and testing was done with the skeleton facing the z axis so the script works only if the skeleton faces either the positive or negative z axis.
When the location of the blast is given the same x and z value such as +5 and –5,a divide by zero error occurs.
Sometimes, due to an unknown reason, when the script is run for the first time, an error message 'group1. Scalex not found' is displayed in the script editor and only one proxy object is created. This can be fixed by simply rerunning the script again and it normally works fine.

Another problem, which sometimes arises when the character mesh is used, is that the arms of the character cling together when the character falls as a result of the blast wave. A possible reason for this could be unmirrored painting of weights for the character and could be fixed by mirroring the weights.

5.CONCLUSION

The initial goal of the project was met. By using the Mel script it is possible to turn a skeleton into a ragdoll, which responds to physical forces. But setting the explosion parameters it is possible to simulate the effect of an explosion. A small animation depicting an explosion surrounded by some characters were done.

The resulting simulation of the rag doll simulation in an explosion scene is realistic and life-life. As mentioned earlier, it was initially planned to extend for a crowd scene, but as the time ran out, the focus was shifted more towards the realism of the simulation for one character .So the objective was more or less achieved. Of course, there are always room for improvements and enhancements. One of the first things that could be done is extending the same tool to work for a crowd scene. The challenges involved in this would be to make it computationally feasible and yet work well. The tool could also be extended to suit more than

one type of skeleton. One way of doing this would be to develop an interface that allows the user to choose each pair of joints individually where the bone is to be created and then let the tool perform calculations to create the proxy object.

The explosion settings have been calculated only for explosions on the ground due to time constraints. This could be extended for explosions in air also. In this case, the y axis would also have to be taken into consideration, and the gravity along the –y axis increased according to the strength and distance of the explosion.

Another issue is that the tool currently uses key framing to tell the rag-doll to switch from animated mode to dynamic mode. This could be turned automatic using Maya's contact data, which can sense if the rag doll is in contact with another object. But another problem arises which is that the rag doll needs to become dynamic at least one frame before it comes in contact with the object. So a bounding box is needed around the rag doll so collisions can be detected earlier.

Other computationally cheaper techniques such as Verlet Integration, used widely in games, as mentioned earlier could be looked into and perhaps certain features of the same be adopted. Also, Adaptive Dynamic Simulation as mentioned in "Adaptive Dynamics of Articulated Bodies" would certainly be a way to get better computation times and improve performance. If a Spring-Mass model were to be used, an interface could be developed where the user could specify how much percentile of the joints he wants to simulate depending on which joints influence the simulation most. Or, the number of joints could automatically decrease or increase depending on the distance from the camera.

6.REFERENCES

- (5)-Author: Wikipedia, the free encyclopedia,"Ragdoll Physics", Available from: http://en.wikipedia.org/wiki/Ragdoll_physics

  Accessed on: 1st September 2005

- Author: Thomas Jakobsen," Advanced Character Physics", Available from:

  http://www.gpgstudy.com/gpgiki/GDC%202001%3A%20Advanced%20Character%20Physics

  Accessed on: 1st September 2005

- Author: Zordan, V. B., Majkowska, A., Chiu, B., Fast, M**.** Riverside Graphics Lab, University of California, Riverside," Dynamic Response for Motion Capture Animation", Available from : http://www.cs.ucr.edu/rgl/projects/mocsim/mocsim.html

  Accessed on: 1st September 2005

- (4)-"Behavioral Dynamics", Available from:

http://web.archive.org/web/20040609173959/http://www.ykoga.com/

Accessed on: 1st September 2005

- (6),(7)-Author: Stephane Redon, Nico Galoppo and Ming C. Lin, In ACM Transactions on Graphics (SIGGRAPH 2005), 24(3),"Adaptive Dynamics of Articulated bodies", Available from:

  http://cs.unc.edu/%7Eredon/AD/

  Accessed on: 1st September 2005

- (2)-"Dynamic Rigging Tutorial", August 2003,Available from : http://www.goldenxp.com/tutorials/ragdoll/page1.htm

  Accessed on: 1st September 2005

- Author: Andy Nichols,Lecture by *Endorphin*
  Torsten Reil, CEO, Natural Motion, Available from:

  http://cs.unc.edu/%7Eredon/AD/

  Accessed on: 1st September 2005

- (1)-Author: Zoran Ka ci¥c-Alesi Marcus Nordenstam David Bullock of Industrial Light and Magic,Industrial Light and Magic , Eurographics/SIGGRAPH Symposium on Computer Animation (2003) D. Breen, M. Lin (Editors)

  Availlable from:
  http://www.uni-weimar.de/~caw/papers/p7-kacic-alesic.pdf#search='a%20particle%20dynamics%20system%20Industrial%20Light%20and%20Magic' Accessed on: 1st September 2005
  Accessed on: 1st September 2005

- (3)Available from :http://www.massivesoftware.com/
  Accessed on: 1st September 2005