Rag doll physics

M. Hennix, P. Hugoson, G. Johansson, A. Lombardi*, T. Miljevic, A. Nilsson, M. Wassborn
Linköping University
ITN/Campus Norrköping
SE-601 74 Norrköping
Sweden
e-mail: annlo@itn.liu.se

Abstract

Physical modelling of real dynamic processes is a challenging task and it has acquired an increasing importance from an educational point of view. Students should become familiar with the different metodologies and learn to solve the related problems. In this case no specific textbooks are necessary but a good training tool is to make students face the problems by implementing a model of a real system. This paper presents the achievements of a project course where students are asked to model, simulate and animate a physical system. Physically based modelling is used and it has been very successfull as it is shown by the excellent results produced in animation.
A physically based animation of the human body is illustrated. The human body is looked upon as lifeless, with mass and inertia as only physical attributes. No attempt to simulate any muscular activity is made. The project can be divided into three major parts: the physical behaviour of a rigid body in space, the behaviour of the constraints and the collision handling. The animation is carried out in real time and interaction with the system is possible.

1 Introduction

Physically based animation [1, 4] is an area that has had a big development in the last decade. It is used in computer graphics and it can also be a very good tool in education, especially to make students understand the need of a good physical model in order to have a realistic animation [2, 3]. Using physics to determine the motion of objects adds realistic effects to animations. Physical laws are used to produce animations and communicate information about how different models interact in a simulated environment. It is possible to obtain real time animation of systems with good graphics rendering for the object under study. Nowadays engineers are more and more asked to build mathematical models of the studied object and to simulate the behaviour by the use of computer graphics. It is, therefore, important that students become familar not only with building mathematical models of physical systems but also with the idea of how models can be used to aid the graphical representation of the system. This is the aim of the *Modelling project* course given to the students of the *Media Technology* program at the Department of Science and Technology (ITN) of Linköping University.

2 Animation of physical systems

Many components are needed in a system for computer animation. It is necessary to create the model of the object to be animated, to simulate the model and finally to create an image and render a frame in the animation. The modelling part can be very demanding if also constraints and collision detection are taken into consideration.

2.1 Modelling

The construction of a mathematical model is based on the insights of how the system actually works. Experts of the field help in this first stage. The second step represents the task the engineer is asked to solve: transfer this knowledge into a useful model. A wide range of literature is available on this topic, the students of this project have already attended the *Modelling and simulation* course based on the book by Ljung and Glad [5] and are familiar with these concepts. The model built for the rag doll is based on physical laws. Some approximations are taken into account but neverthless the resulting animation has a behaviour very close to

*corresponding author

the real system.

2.2 Simulation

Equations for forces and torques are written down and the acceleration of the system is determined. In order to get the position of the system at any time, integration must be carried out. This involves well-known numerical problems [6], among which to determine the best integration method and the step size. The choice of the numerical method depends on many factors such as the complexity of the system and the accuracy requested by the application. When simulation is used in real time applications, the choice of the integration interval plays a relevant role and it must take into account the frame rate of the graphical implementation.

2.3 Animation

A graphical animation makes it possible to represent how a system changes over time. In this context, the term system denotes not only the object itself but the object together with its environment. This implies that animation can be used to represent how an object moves in time. The computer renders the scene and displays all changes in rapid succession creating in this way the illusion of motion. *Motion control* is the way how these changes are specified to the computer. There exist several different ways of achieving this control [7]; among these, *physically based modelling* has become very successful in recent years. This is the approach used in this project: physically based models are incorporated into the animation; the objects are given mass and inertia, then forces and torques are applied to achieve a certain motion.

3 Modelling project

In the modelling project students work in small groups and they decide together the system they are going to work with. The first step is, therefore, to analyze a physical system and try to understand the possible difficulties that might arise while the task is carried out. In the beginning some assumptions on the behaviour of the system are made in order to get a simplified model and the graphical representation of this simple model is implemented. For instance, if the aim is to animate a car, the students will start with the model of the dampers and a mass will be represented moving along a given profile of the road. Step by step some of the simplifying assumptions are removed and a more complex system is obtained. Most of the students choose to work with systems they are familiar with so that they can also use their own experience to evaluate the correctness of the model.

4 The physics

The main goal of the work described in this paper is to represent something that looks like a human body. A body built of boxes with ball joints connecting them together and making it stable at an acceptable frame rate. Important aspects are collision detection and to have interaction with the rag doll.
Considering every force F acting on the body (with mass m) as impulses it yields two kinds of motion: linear and rotational.
The linear motion is implemented by using the linear momentum p

$$p = mv \tag{1}$$

where v denotes the linear velocity of the body. The formula to update the linear momentum is given by

$$\dot{p} = F \tag{2}$$

from (1) and (2) the velocity is derived and therefore the position of the centre of mass of the body is obtained at any time step of the simulation.
Rotation is determined [8, 9] by considering the angular momentum L

$$L = I \cdot \omega \tag{3}$$

where I and ω represent, respectively, the inertia tensor and the angular velocity expressed in the global coordinate system. The inertia tensor I takes into account that the body can rotate around the three axes [8]

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{4}$$

When the body is rotated, the inertia tensor I will also be rotated in order to agree with the physical properties of the body. If the rotation can be described using a rotation matrix R, then the rotated inertia tensor is given by

$$I' = R\, I\, R^{-1} \tag{5}$$

The updating formula for the angular momentum is

$$\dot{L} = \tau \tag{6}$$

with τ denoting the torque produced by the force F acting on the body at the distance h to the centre of mass: $\tau = h \times F$.

5 Constraints

When dealing with constrained rigid bodies, additional methods are required in order to achieve correct physical motion. Forces acting on the body of the system affect all other bodies of the system through joints. A constraint equation must be solved during the simulation in order to obtain the additional forces - *constraint forces* - acting on the bodies.
When taking into account constraint forces, the second Newton's law can be written in a more general notation [10]

$$\ddot{q} = W(Q + \widehat{Q}) \tag{7}$$

where q is a vector of both positions and angles, W is a matrix containing the inverse of the mass and the inertia, Q denotes a vector of forces and torques acting on each body and $\widehat{Q}$ forces and torques produced by the constraints (see Appendix A for more details).
Only ball joints have been used in this work: the two bodies connected by a joint must always be in contact. This contact is obtained by forcing a point on the other body. These two points are called *anchor points*. The difference between these two points must be zero, i.e.

$$C = (p_1 + R_1 a_1) - (p_2 + R_2 a_2) = 0 \tag{8}$$

where a_i is a vector going from the centre of mass of body i to the joint, p_i describes the position of body i and R_i denotes a rotation matrix which rotates the vector a_i to the global frame as it is illustrated in Fig. 1.

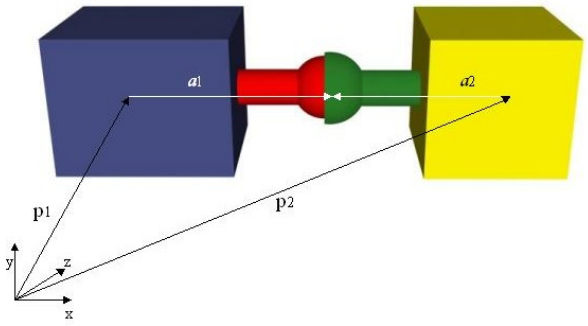

Figure 1: Keeping the bodies together with the constraint vectors

Hinges and universal joints have also been implemented to some extent even if the implementation is not completely working yet.
In order for the points to stick together they must move with the same velocity and acceleration. This implies

$$C = \dot{C} = \ddot{C} = 0 \tag{9}$$

By differentiating C, we obtain

$$\dot{C} = \frac{\partial C}{\partial q}\dot{q} \tag{10}$$

the matrix $\frac{\partial C}{\partial q}$ is the Jacobian and will be denoted as J in the following.
$\ddot{C}$ is obtained by differentiating $\dot{C}$

$$\ddot{C} = \dot{J}\dot{q} + J\ddot{q} = \dot{J}\dot{q} + JW(Q + \widehat{Q}) \tag{11}$$

Considering (9), Eq. (11) results in

$$JW\widehat{Q} = -\dot{J}\dot{q} - JWQ \tag{12}$$

The constraint forces are only used to limit the forces and are not supposed to do any work: the constraints should not add energy to the system. To ensure this, Witkin [10] introduces the concept of *virtual work* which gives

$$\widehat{Q} = J^T\lambda \tag{13}$$

where λ is a vector of Lagrange multipliers designed to make sure that the energy balance in the system does not change. Inserting (13) into (12) it yelds

$$JWJ^T\lambda = -\dot{J}\dot{q} - JWQ \tag{14}$$

The final step is to introduce a feedback in order to compensate for numerical drift. The final constraint equation becomes

$$JWJ^T\lambda = -\dot{J}\dot{q} - JWQ - k_s C - k_d\dot{C} \tag{15}$$

where k_s and k_d are spring and damper coefficients. They are chosen either high or low depending on the desired feedback.
In the right hand side of (15), the four variables C, $\dot{C}$, J, $\dot{J}$ appear. The computation of C and $\dot{C}$ is quite straightforward since they represent, respectively, positions and velocities. The computation of J and $\dot{J}$ is reported in Appendix B.
Eq. (15) has the form $A\lambda = B$ with the matrix A being positive-definite and symmetric and therefore it can be solved with Cholesky factorization. This is a very important result for simulation because it gives a significant improvement in terms of speed. For large systems, it means to go from $O(N^3)$ iterations to $O(N^2)$.

6 Collision detection

Joint constraints are not enough to give a natural appearance during simulation. This is due to the fact that the body parts free to move can collide with each

other and exchange energy. The word *collide* is used although nothing can actually collide in virtual space: the problem is to determine if any point in one body part is inside another part of the body. The naive approach for the problem is to check if every point on every body part is inside all other body parts. This is very heavy from a computational point of view and it is not possible to implement with reasonable frame rate.
Collision detection and response [11] can be divided into three parts. The first consists in identifying which objects are close enough that a collision might happen. It is called the *broad phase* as it checks the entire simulation for collision and therefore has a broad prespective. The second phase focuses on confirming the actual collision and finds the exact point of impact. It is called *narrow phase* as it has a narrower approach with respect to the previous one. The third phase calculates the behaviour of the bodies after the confirmed collision and hence it is called *collision response*.

6.1 The broad phase

An object is defined in the three dimensional space as a group of points that are sorted either into triangles or quadrates to form planes. Depending on which order the points are defined, the planes have a normal that shows in which direction the plane exists. If an object has a complex shape, many triangles and points are needed in order to define it.
To handle the broad phase, it is necessary to check if a set of points in one object is close to the set of another object. Since a more complex object consists of more triangles and points it is hard to get an idea of what space this object occupies. This problem is solved by using a method that uses easily defined shapes called bounding objects. The method consists in taking an advanced set of points (a non primitive object) and putting it inside an easily defined object like a box or sphere. In this case, the simulation consists of a body built by boxes so the choice was easy. The axis aligned bounding box method has been implemented. It consists in projecting the maximum and minimum value on each global axis that the body part utilises. The smallest value is indexed B while the largest value is indexed E, as in beginning and end (see Fig. 2).
The B and E values for every body part and axis are stored in three lists that are sorted after values. The advantage of this structure is that only bodies having B and E values close to each other can possibly collide when the coordinates of their bounding boxes overlap. The sorted character of the list also gives information

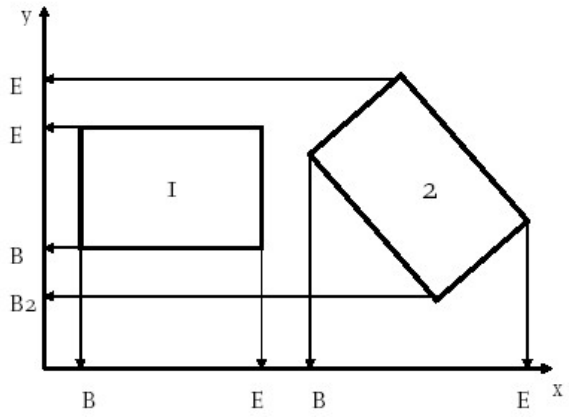

Figure 2: Principle of the axis aligned bounding box method

that something has happened between two bounding boxes when a change in the list is nedeed. When the broad phase has found two colliding body parts it puts them in a list that is sent to the narrow phase.

6.2 The narrow phase

When the narrow phase starts it takes the lists received from the broad phase and evaluates potential collisions. This is done by checking every point on the body inside the other body and vice versa. When the point that has enetered the other body part is found, the point of impact is determined from which the collision response will apply its change.
There is another kind of collision that needs to be taken into account namely the edge collision that is not found with the usual point inside body check. It is implemented by a different method that needs to be applied if the first method does not find anything. Our implementation does not go back to process the real point of impact but we rely on the fact that the evaluation is done so often that any penetration between the body parts during the time step is so small that it will not be an important factor. If the method finds more than one point or an edge collision it will calculate an average point that will give a good approximation.

6.3 Collision response

When the point of collision has been determined, the collision response computes the change in speed that occurs in a totally inelastic collision. The method uses the point of impact together with the plane or edge on which the collision occured and its normal in order to determine the new velocities of the colliding body

parts taking into account how much energy is lost during the collision. The method is able to handle collision with more than one body part by summing up all the new velocities from all the collisions and then applying the resulting velocity to the body.

7 Numerical method

At any time step of the simulation, the linear and angular momentum produced by forces and torques are computed and the velocities of the bodies are updated using (1) and (3). The new positions of the bodies are then computed using a numerical method [6]. Two methods have been implemented. *Euler*'s method is explicit in the sense that it uses only information at time t to obtain a solution at time $t+1$

$$y_{t+1} = y_t + hf(x_t, y_t) \tag{16}$$

Euler's method has rather limited stability region. Higher-order stability can be achieved by averaging explicit and implicit Euler's methods so that the *trapezoid* rule is obtained

$$y_{t+1} = y_t + \frac{h}{2}[f(x_t, y_t) + f(x_{t+1}, y_{t+1})] \tag{17}$$

In this method y_{t+1} appears implicitly on the right hand side and it is approximated by using explicit Euler's method given in (16).
Euler's explicit method has proved to be stable despite its worse precision so it is currently preferred in the animation as it is faster.
A variable time step is implemented. It is set to 0.01 by default and it is stable up to approximately 0.03, depending on the values chosen for K_s and K_d.

8 The program

The code is written and compiled using Microsoft Visual C++. A screenshot from the application is shown in Figure 3.
The available inputs and the system response can be seen on the window and changed during the animation as it is shown in Figure 4.

9 Conclusion

In this paper the graphical animation of a rag doll is illustrated. The goal of the work is to represent something that looks like a human body. A physical based model is first built, then it is implemented taking into account physical constraints and collision detection. The animation is carried out in real time making it possible to have interaction with the system during the simulation. The code is written in C++.

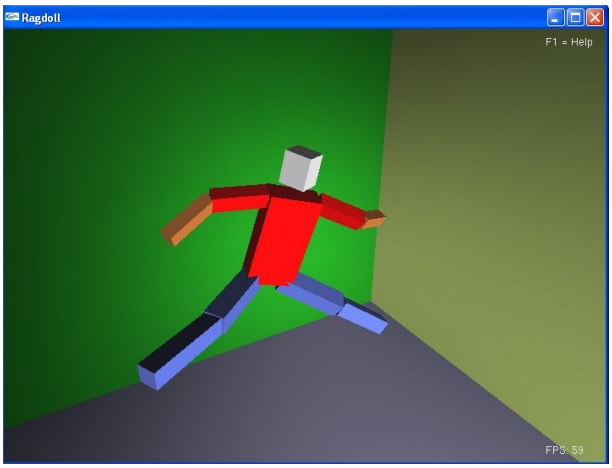

Figure 3: Screenshot from the program

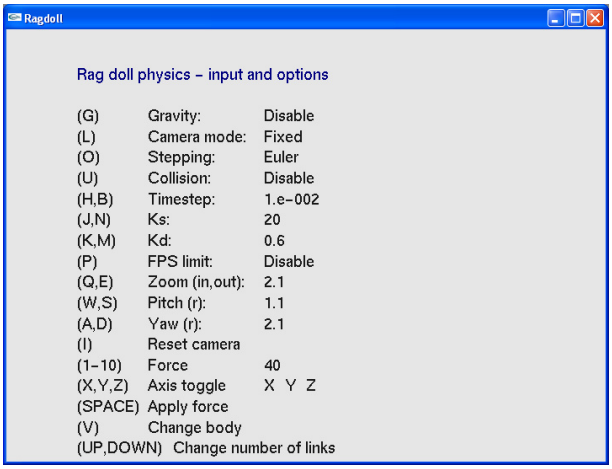

Figure 4: All available inputs

References

[1] P.E. Wellwstead. *Physical System Modelling*. Academic Press, New York, 1979.

[2] P. Sweeney, A. Norton, R. Bacon, D. Haumann, and G. Turk. Modelling physical objects for simulation. In *Proceedings of the 1991 Winter Simulation Conference*, pages 1187–1193, 1991.

[3] R. Lelouche. Use and misuse of graphic animations and simulations in educational systems. In *Proceedings of the International Workshop on Advanced Learning Technologies*, 2000.

[4] F.E. Cellier. *Continuous System Modelling*. Springer-Verlag, New York, 1991.

[5] L. Ljung and T. Glad. *Modeling of dynamic systems*. Prentice hall, 1994.

[6] E. Hairer, S.P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations*. Springer-Verlag, New York, 1987.

[7] P. Fishwick and H.A. Porr. Using discrete event modeling for effective computer animation control. In *Proceedings of the 1991 Winter Simulation Conference*, 1991.

[8] D. Baraff. Rigid body dynamics I. http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/baraff/www/pbm/rigid1.pdf.

[9] H. Benson. *University Physics*. John Wiley & Sons, 1996.

[10] A. Witkin. Physically based modelling: Principles and practice. constrained dynamics. http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/baraff/www/pbm/constarints.pdf.

[11] D. Baraff. Rigid body dynamics II. http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/baraff/www/pbm/rigid2.pdf.

A Appendix

Referring to (7), the matrices in the case of two bodies are the following:

$$q = [p_{1x} \quad p_{1y} \quad p_{1z} \quad \theta_{1x} \quad \theta_{1y} \quad \theta_{1z} \quad p_{2x} \quad p_{2y} \quad p_{2z} \quad \theta_{2x} \quad \theta_{2y} \quad \theta_{2z}]^T \tag{18}$$

$$Q = [F_{1x} \quad F_{1y} \quad F_{1z} \quad T_{1x} \quad T_{1y} \quad T_{1z} \quad F_{2x} \quad F_{2y} \quad F_{2z} \quad T_{2x} \quad T_{2y} \quad T_{2z}]^T \tag{19}$$

W is a square matrix with dimension 12 and it can be represented with the following blocks

$$W = \begin{bmatrix} M_1^{-1} & & & & & \\ & I_1^{-1} & & 0 & & \\ & & M_2^{-1} & & & \\ & 0 & & & & \\ & & & & I_2^{-1} & \end{bmatrix} \tag{20}$$

each block on the diagonal has dimension 3; the blocks for the first body are

$$M_1^{-1} = \begin{bmatrix} \frac{1}{m_1} & 0 & 0 \\ 0 & \frac{1}{m_1} & 0 \\ 0 & 0 & \frac{1}{m_1} \end{bmatrix} \quad I_1^{-1} = \begin{bmatrix} \frac{1}{I_{1x}} & 0 & 0 \\ 0 & \frac{1}{I_{1y}} & 0 \\ 0 & 0 & \frac{1}{I_{1z}} \end{bmatrix} \tag{21}$$

the same notations are valid for the second body.

B Appendix

For ball joints, J and $\dot{J}$ can be computed by differentiating (8).

$$\begin{aligned} \dot{C} &= v_1 + (\omega_1 \times R_1)a_1 - v_2 - (\omega_2 \times R_2)a_2 = \\ &= v_1 + \hat{\omega}_1 R_1 a_1 - v_2 - \hat{\omega}_2 R_2 a_2 = \\ &= v_1 - \hat{a}_1' \omega_1 - v_2 + \hat{a}_1' \omega_2 \end{aligned} \tag{22}$$

where we have used the following results from algebra: $\omega \times R = \hat{\omega} R$, $a' = Ra$ and $\hat{a}b = -\hat{b}a$. The matrix $\hat{a}$ has the form

$$\hat{a} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{23}$$

Since $\dot{C} = J\dot{q}$, J is a matrix containing the coefficients of the linear and angular velocities

$$J = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -a_{1z}' & a_{1y}' \\ a_{1z}' & 0 & -a_{1x}' \\ -a_{1y}' & a_{1z}' & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & a_{2z}' & -a_{2y}' \\ -a_{2z}' & 0 & a_{2x}' \\ a_{2y}' & -a_{2z}' & 0 \end{bmatrix}^T \tag{24}$$

The same technique can be used to compute $\dot{J}$.