ACTIVE HAPTIC EXPLORATION

Philip Crabtree

BSc Computer Science & Cybernetics

ABSTRACT

Haptic technology allows the simulation of real objects and real world physics within a virtual world. Whilst this, in many fields, is a solution to many practical issues it also draws attention away from the possibility of simulating objects and environments that simply can not exist in the real world. This can extend from simulating behaviours such as inverse gravity right through to impossible objects such as a tardis. Haptics can also be used to investigate how the human mind creates links between object properties. i.e. A small object is often seen to be light where as a large object is heavy. Haptics allows these basic, natural human assumptions to be tested by creating objects with properties that go against instinct.

1. INTRODUCTION

The haptic interfaced used is the Perforce haptic interface designed by Cybernet Systems [1] and has full six degrees of freedom (6-DoF) which allows the user to manipulate an object in full three dimensional space including rotations around each axis. This device has 6-DoF in terms of both input and output enabling the user to not only manipulate objects in the virtual environment but also to feel how the environment reacts in terms of force and torque.

The position and rotation of the haptic arm joints are measured using quadrature encoders (attached to the motors controlling the position / rotation of each joint) which will transmit a phase-difference signal to one of two PCI interface Sensoray cards [2] via an amplifier. When controlling the position of the haptic arm (i.e. giving feedback to the user about the virtual environment) the above is done in reverse. That is, supplying a signal from the Sensoray cards to the motors such that they can position the joints of the arm to the desired position.

There are several, extremely noticeable differences between manipulating an object in the real world and manipulating an object in a virtual world through the means of a haptic interface. Perhaps the most confusing (in terms of usability) issue is that when reaching for an object in the real world one's hand will extend to the object. In the virtual world the haptic controller is either not in sight or only in peripheral vision which causes many users to have problems with depth perception [3]. The other major problem when controlling this particular haptic interface is due to the natural forces of friction and gravity that one has to overcome to physically move the arm where as in the real world there is no noticeable friction when extending ones arm to pick up an object. Both these problems can be combated in several ways which are discussed in later sections.

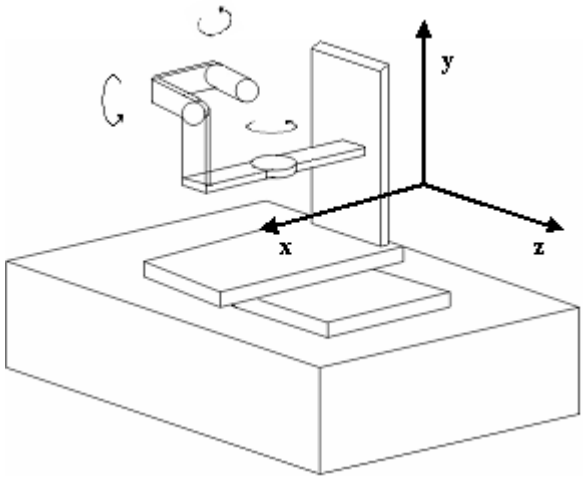

Figure 1. The Perforce haptic interface and its directions of movement

2. DESIGN CONSIDERATIONS

A haptic interface must be capable of accurately representing forces exerted onto a user's hand. The human nervous system can fairly easily detect vibrations lower than approximately 500Hz, hence the haptic device must transmit and receive updates as to its current and desired position at least 500 times each second. Any variance in this frequency will be detected by the user and perhaps miss-interpreted as a force exerted by a virtual object. The constant requirement for a fast frame rate suggests that a common operating system such as Windows or Linux is unsuitable, unless running on a high powered (preferably dual processor) machine due to their non-real-time behaviour. QNX (Quick Unix) is a real-time operating system which will allow any currently executing program (i.e. the haptic interface controller) to run on a higher priority than any other program which may be run as 'Idle System' processes. This means that a constantly high haptic frame-rate can be achieved as no other tasks will attempt to use the

processor. Perhaps one of the biggest issues with the use of QNX for the application is its lack of support for good graphic systems. OpenGL has been made available on some more recent versions of QNX but there is still a lack of support for graphic card drivers.

In order for a user to get a sense of presence within a world there must be some form of visual representation of the virtual environment and a representation of the users hand position with respect to that virtual world. In order to do this accurately, with a high enough frame rate as to realistically represent the virtual world, the graphic system will be run separately on a second computer linked to the haptic interface system via a standard Ethernet networking cable. This will not only allow for a higher quality of graphic representations but it will also allow the haptic QNX system, discussed above, to focus solely on the task of controlling the haptic arm without the need to process costly graphics. The graphics system will, in general, only read data that is sent across the network concerning world object positions (including the position of the end effecter of the haptic arm). This data is then fed into the rendering system to produce the visual representation of the virtual world through the use of OpenGL. As mentioned above, the haptic interface will have a minimum requirement of 500FPS (frames per second). The graphical system needs approximately only $1/10^{th}$ of this frame rate as the human eye can generally only detect frequencies no higher than 85Hz. Therefore updates concerning the positions of objects within the virtual world need only be transmitted approximately once for every ten haptic frames. Whilst this will drastically reduce the load on both processors it is still approximately 50 updates per second. The exact size of each update (in terms of bits) will alter with the varying application but it is to be expected that a minimum of six floating point values (representing the end effecters position and rotation) will be transmitted at each update. For this reason it has been decided that UDP will be a more appropriate solution for broadcasting data points than using TCP/IP on the basis that UDP transmission and reception is faster than that experienced with other forms of network communication. UDP does have its shortcomings. One of these is that data packets are not checked or verified and so they can easily become lost or corrupted. This, in practice, causes very few issues as the data is updated on such regular intervals and hence any lost or corrupt data is immediately corrected by the next update.

It was mentioned previously how the problem of special awareness can occur when using a device that is not in the same space as the visual representation. Perhaps the best way to overcome this problem is to project the virtual world image on to a reflective yet slightly transparent surface, like a two-way mirror, such that the haptic device end effecter (placed behind or below the surface) appears to be in the virtual world. This however is not practical with the haptic device used and is more suited to a device such as the Phantom [4]. The second method of reducing the issue is to use a pair of stereoscopic three-dimensional glasses. This requires the virtual world to be represented in stereo although this can be achieved using OpenGL with relative ease. The enhanced depth will now allow the user to judge distances in relation to the z-dimension with greater accuracy. In addition to the application of stereo images, shadows will be added to the world which will further increase the user's accuracy as shadows can provide a reference on a flat plane (i.e. the ground) as to the distance between two given objects (e.g. the virtual object the user is manipulating and a ball).

The other problem discussed earlier is that of the natural forces experienced when manipulating the haptic arm due to the device's weight. Spring coils have been placed on the relevant joints to combat gravity although they do little to ease the user's movement. A force torque (FT) sensor is capable of measuring the force and torque placed upon the device. The force torque sensor used on this device is the 'Mini40' model manufactured by ATI Industrial Automation, capable of measuring in 6-DoF [5]. The FT sensor is placed in the exact centre of rotation of the haptic device when in its rest position and is attached to the haptic arm on one side and the users 'tool' on the other. The sensor will give 6 values (one relating to each degree of freedom) which is input through the Sensoray cards to be interpreted by the QNX system at each frame. As very subtle pressure can be detected by the FT sensor this can be read and amplified in code with the result of giving the arm a desired position away from the current position, hence pulling the arm along in the direction the user wishes to move. This, when set-up correctly, gives the arm a weightless feeling and thus provides a vastly increased sense of immersion.

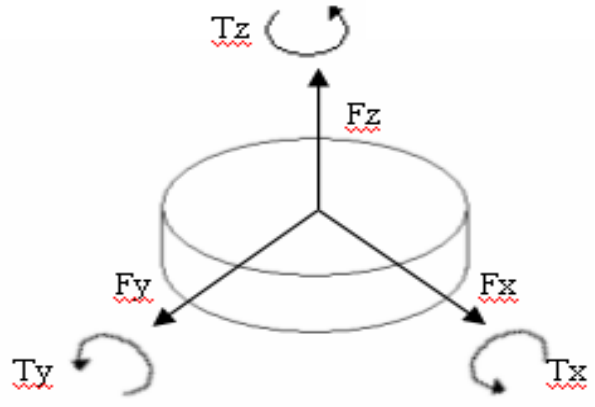

Figure 2. The ATI-ia Mini40 Force Torque sensor and the directions of applied force and torque.

The transducer consists of three beams (symmetrically placed) which, when a force is applied, flex. Each beam is attached to a strain gauge which alters its resistance value depending upon the force applied. The actual force and torque components of the applied strain are calculated in software when required.

3. APPLICATIONS

The applications were designed to demonstrate either situations or environments that can not exist in real life or as a psychological test to see how a human relates size and weight [6]. The latter of these applications is known as Visual-Weight dominance and can take several forms.

3.1. Visual-weight dominance

The user is presented with a cup shaped representation of the haptic arms end effecter and two different coloured balls on screen. One ball is slightly larger than the other but both have an equal mass. The user is asked to pick up each ball in turn (by scooping with the controlled cup) and announce which one they believe is the heaviest. This application also exhibits some basic physical laws such as gravity. To implant these laws there were several options. One option was to use an open source physics engine such as ODE [7]. The advantages and disadvantages of ODE were weighed up and it was found that whilst a highly accurate physical simulation was produced, complete with collision detection, the processing time for each frame became unacceptable. This was found to be the case with nearly all open source physics engines and as a result a purpose built physics engine was produced. The purpose built engine is not as accurate as an engine such as ODE would have been but it has very little impact on the haptic frame rate as only necessary calculations are performed.

3.2. Visual-weight dominance (springs)

This follows the same concept as the above test but instead of balls the user is presented with two equally sized springs. One spring appears to be stiff and the other appears to be fairly springy. The user now controls a sphere which can be placed above the springs and used to compress them. Naturally the user will put more initial pressure on the spring that appears stiffer although the properties will have been altered such that the spring that appears springy is harder to compress than the spring that looks stiffer. The only element of a physics engine that is required for this application is the collision detection routines as the force experienced by the user can be simulated for increased system performance. These routines are identical to those created for the first visual-weight application.

3.3. Cloth

In the real world when a human is presented with a piece of cloth or rubber we assume that, unless the material is torn, there is no way to pass through. This application demonstrates that in a completely virtual environment these things are possible. The user is presented with a controllable sphere and a sheet of cloth. When pushing the sphere against the cloth the user will feel an appropriate force pushing against them. However, if the user continues to push harder then eventually they will break through the cloth and get to the other side. In some respects this can perhaps be seen as an exaggeration of the modelling of water surface tension. Note that for the purposes of this application the cloth is simulating a thin sheet of rubber or elastic. This was chosen because, compared to other cloth materials, the force excerpted on the user is considerable and hence more suitable for a haptic application. A description of how the deformation of a surface, such as cloth, can be achieved can be found in section 3.5.

3.4. Tardis

Again, this demonstration aims to show how haptic interfaces can be used to represent object properties that can not exist in the real world. The user has a controllable sphere which is initially placed inside a larger, wire-frame sphere (the tardis) with a small section from the top removed to represent an opening. When inside the tardis the user is able to freely move around until collision with the edge of the tardis occurs. If the user comes out of the tardis they are still able to move freely but they will notice that their movements are considerably smaller compared to those experienced when within the tardis. Again the user will collide with the outside tardis walls if they get to close. The effect of the tardis is achieved by simply scaling down movements made whilst inside the tardis. i.e. 1 unit of movement by the arm in real space is scaled to be equal to, for example, 0.75 units in virtual space thus allowing the user to move the arm further before hitting the edge of the tardis giving the impression that the tardis has more volume that it appears to possess.

3.5. Object deformation

Object deformation is best demonstrated in the cloth simulation application [8]. To deform any given object there are two available strategies, one is to have a set of ‘control points’ common to both the haptic system and the graphical system. These objects might represent vertex points or similar. Another method is to dynamically create movable points on a shape when a collision is made between the users controlled object and the object to be deformed. The cloth simulation application follows the first of these two methods. A simple plane with movable control points is not enough to accurately simulate cloth. Each control point should be assigned a mass value representing the material being simulated. Assigning a mass allows each point to be affected by gravity and furthermore, allows for simulation of other external forces [9]. To further represent a real cloth, each control point is linked to several other control points by ‘springs’ (i.e. the cloth is a mass-spring system). If the very central control point is pushed away from its resting position then its surrounding control points will react and move accordingly, as will their surrounding control points.

The above solution is a very good approximation of a real piece of cloth and this can be increased further by the addition of more control points within the same area. The most obvious difference is that a straight line is drawn between control points rather than a curved surface that is expected. NURBS (non uniform rational

B-splines) allow for three-dimensional curved surfaces [10] to be represented and are the most popular choice for the representation of organic life-forms in high-end graphical systems. The fundamental problem with using NURB surfaces for the applications listed is that performing collision detection in real-time is a very slow process (generally based on iterative ray tracing [11]) and hence results in a very slow, unacceptable frame rate. Methods exist that do deal with the real-time implementation of NURB surface collision [12] but when these methods are combined with the delays experienced when transmitting data across the network (due to the shear amount of data that must be communicated at any one time) the result is still unacceptable.

4. PHYSICS AND COLLISION DETECTION

As mentioned previously, some of the designed applications require a custom built physics engine. All of the applications require some level of collision detection and to fully understand the physics engine one must first appreciate how the collision detection routines are executed.

4.1. Collision detection

In most cases the virtual object representation of the end effecters current position is a sphere. Primarily this is because the speed at which sphere collisions can be executed. To explain the principles of object collision a simple two-dimensional sphere-sphere collision example will be used.

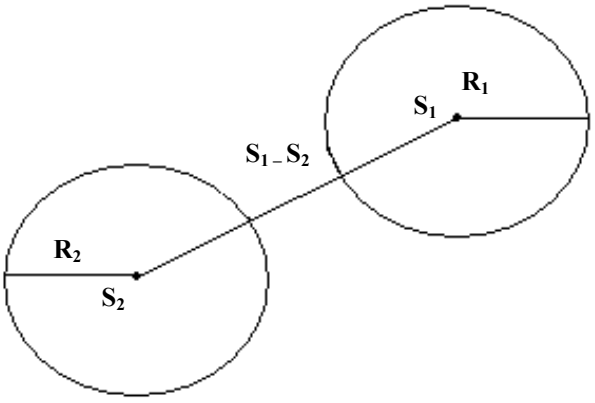

Figure 3. Sphere-sphere collision

Figure 3 demonstrates how the distance between two spheres is calculated. The sum of the R_1 + R_2 is subtracted from the calculated distance and a collision occurs if the total value is equal to or less than zero. i.e. a collision occurs when:

$$(S_1 - S_2) - (R_1 - R_2) <= 0 \qquad (1)$$

Where:

$$(S_1 - S_2)^2 = (S_{1x} - S_{2x})^2 + (S_{1y} - S_{2y})^2 + (S_{1z} - S_{2z})^2 \qquad (2)$$

This is as far as the collision detection engine goes. All resultant forces are calculated in the physics engine (note that the physics engine calls the collision detection routines). Note that the above principles can be extended to nearly all other objects only with the added complexity of finding the distance from the objects centre to an edge. This is especially complex in deformable surfaces of both polygon mesh form and NURBS surfaces [13]

4.2. Physics

The physics engine will, first of all, calculate the new positions of any objects that require moving (e.g. objects that are falling due to gravity or moving due to a previous collision). The collision engine will then be called to check if any collisions have occurred between any objects in their new positions. Supposing a collision has occurred, the next step will be to calculate the resultant force. If the collision occurred between an object and the haptic interface then the resultant force is added to whatever force may currently exist (due to the force torque sensor or otherwise) to represent how the collision might feel.

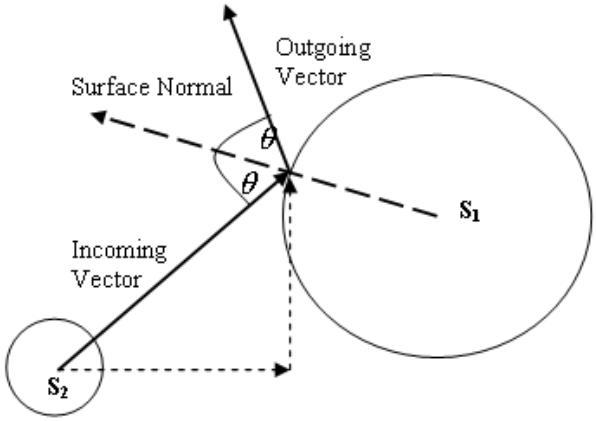

Figure 4. Resultant vector after collision

Figure 4 shows how the resultant force after a collision is calculated. For simplicity the figure only illustrates in two dimensions but adding a third dimension is a straight forward process. Again, sphere-sphere collision is used as the most simplistic implementation of these rules. The surface normal is a vector 90 degrees away from the tangent to the surface drawn at the collision point between the two objects. For a sphere, this can be expressed as an extension of the vector from the sphere centre point to the collision point as is illustrated in Figure 4. The outgoing vector is simply a reflection of the incoming vector through the surface normal. Of course in the real world (and as is modelled in the virtual world applications) some of the initial force of the incoming vector is absorbed by the collision (depending upon the elasticity of both objects) as well as being enhanced or reduced by external forces such as gravity. The force experienced by the second object (S_1 in this case) can be calculated by looking at the individual x, y, z and, if applicable, the rotational components of the incoming vector.

5. SYSTEM CONTROL

The whole control system acts as one large loop. The first stage in the loop is to ascertain the current position of the arm in real space and hence map this to the 'home position' in the virtual world. To get the current position the quadrature encoders are polled and the returned data is converted to floating point values. Forward kinematics equations are then performed in-order to ascertain the exact positions and rotations of each joint and as a result the end effecter can be represented correctly within the virtual world (note that the returned values are scaled to fit within the world co-ordinates). Information from the force torque sensor is now collected and is passed through a calibration matrix specific to each transducer. The resultant matrix will contain information as to how the user is applying force (or torque) to the haptic interface which can then be used to calculate a desired position for the arm. Collision detection between virtual world objects can now be performed. If a collision does occur between an object and the representation of the haptic interface end effecter then the resultant force can be calculated and passed back out to the motors resulting in the arm moving to the desired position (note that this desired position must take into account the desired position calculated previously). This desired position is achieved by using inverse kinematics and is essentially the opposite of forward kinematics. At this time, if requested, the positions of the joints and any other objects that exist within the virtual world are sent across the network to the graphics machine such that they can be displayed.

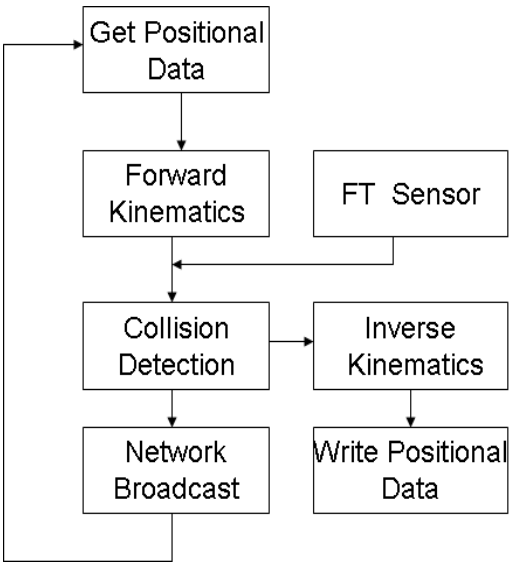

Figure 5. The control system

6. FUTURE WORK

One problem that has become clear during testing is the performance of the force-torque sensor. Whilst it certainly does effectively reduce the weight of the arm and hence allow for easier manipulation, it is apparent that, on frequent occasions, there is some confusion as to which way the user is trying to move the arm. For example, if the user was solely pushing along the y-axis it is possible that a signal will bleed through from the force torque sensor concerning a different axis and try, for example, to pull the arm across the x-axis. This is most likely occurring due to error in the calibration data received with the sensor. Further tests will be carried out to explore this behaviour in more depth and devise a solution to correct erroneous signals.

Whilst only few tests concerning visual-weight dominance have been carried out it has been commonly noted that there is some confusion when initially lifting one of the balls because or the additional force that must be overcome to actually lift the arm. This often results in a sudden, sharp upwards movement giving the ball in question such an impulse that it is thrown from the users 'cup'. This however, should be solved when the force torque sensor is introduced as movements to the arm will be made significantly easier.

The cloth simulation application also has areas that require work. Currently two models exist. One uses the normal technique of performing all calculations within the QNX system and simply passing the control point data over the network which is interpreted to give a visual representation of the cloth. This however proved to be too much of a strain on the network when performing at the minimum desired frame rate and as such was not suitable. For this reason the whole cloth simulation was transferred to be processed by the graphics machine resulting in a much higher frame rate. The cloth in the QNX system was simply replaced by an infinite plane running down the y-axis to represent the resting position of the cloth. The force experienced when the user crosses this plane appears to be due to the cloth pushing back but is however, just a simple calculation based on how far across the plane the user has gone. Future work in this area will investigate how it might be possible to transmit only new (i.e. updated) data or small sections of the overall data at any one time and hence reduce the strain on the network.

7. CONCLUSION

Whilst the system is still not fully implemented initial tests have proved to be, on the most part, successful and data has been gathered as to how the system may be improved as discussed in section 6. It does, however, seem clear that once the force torque sensor is successfully implemented some of the problems will be automatically overcome.

It is clear that whilst the aim of the project is to investigate environments and objects that can not exist within the real world, it is necessary to include some basic real-world rules such as collision response, gravity and, in most cases, relative movement. These rules need not be exactly as is experienced on Earth but for

practical use the behaviour should be predictable by the user. For example, the effect of gravity can be reduced or even inverted to fit the application area and the user will still be able to relate what they see and experience in the virtual world to the real world.

The level of technology discussed in this paper is generally aimed at the industrial user such as the military and National Health Service who have more of a need to create physically accurate environments rather than the types of applications discussed. However, this is not to say they have no use for such applications. The cloth simulator for example, may well be adapted to represent the forces experienced when inserting a needle into a patient's skin or the forces required to cut flesh. Whilst the kind of technology discussed in this paper is not readily available for the home user today it does appear to be an area of growth which companies such as video game designers are becoming more interested in. When this level of technology is readily available then it can be assumed games may incorporate features such as the tardis application and other forms of impossible environments or objects.

8. REFERENCES

[1] Perforce Haptic interface, Cybernet Systems, http://cybernet.com

[2] 626 Analogue and Digital IO PCI interface cards, Sensoray, http://www.Sensoray.com/html/626data.htm

[3] Laroussi Bouguila, Masahiro Ishii and Makoto Sato. Effect of Coupling Haptics and Stereopsis on Depth Perception in Virtual Environment. Precision and Intelligence Laboratory, Tokyo Institute of Technology

[4] Phantom Haptic Interface, SensAble Technologies, http://www.sensable.com/

[5] ATI Mini40 6-axis Force Torque Sensor. http://www.ATI-ia.com & Document #. 9610-05-1017-06 October 2003

[6] Mark O Ernst, Martin S Banks. Humans Integrate Visual and Haptic Information in a Statistically Optimal Fashion. University of California.

[7] Open Dynamics Engine (ODE). Russell Smith. http://ode.org

[8] Robert Bargmann. Real-Time Cloth Simulation. SSC

[9] David Baraff, Andrew Witkin. Large Steps in Cloth Simulation. Robotics Institute, Carnegie Mellon University.

[10] Demetri Trezoppulos, Hong Qin. Dynamic NURBS with Geometric Constraints for Interactive Sculpting. Department of Computer Science, University of Toronto, ACM Transacions on Graphics, 13(2), April, 1994, 103-136

[11] William Martin, Elaine Cohen, Russell Fish, Peter Shirley. Practical Ray Tracing of Trimmed NURBS Surfaces. Journal of Graphical Tools, Volume 5, No. 1, 2000p. 27-52

[12] Dean Macri. Using NURB surfaces in Real-time Applications., Gamasutra, November 17th 1999. http://www.gamasutra.com/features/19991117/macri_01.htm [requires membership]

[13] Rynson W.H. Lau, Mo Luk, Oliver Chan, Frederick W.B. Li. A Collision Detection Framework for Deformable Objects. City University of Hong Kong.

All web addresses referred to in this paper were verified on 17th February 2005.