Optimization of Collision Handling based on Differential Thresholds of Human Perception

Shaila Abraham
Cachematrix
Denver, CO 80206

Min-Hyung Choi
Department of Computer Science and Engineering
University of Colorado Denver
Denver, CO 80217

Abstract - *An efficient collision handling mechanism is essential for most computer animation and to achieve realistic yet efficient animations we need to consider human perception in these animation systems. Current states of art collision handling algorithms do not consider human perception in depth, particularly in differentiating thresholds of various collision parameters. Yet the human perception plays a significant role in the quality of perceived animation when multiple collisions are presented in an animated scene. The responsiveness of the objects at the right time and the right behavior makes the animation system look natural. In this paper we describe an approach, which approximates objects by using an interruptible detection algorithm to proximately test for collisions between different objects. Based on the inputs from the user study made we are able to adjust the various parameters in our collision detection algorithm and were able to get considerable speed up.*

Keywords: *Collision handling, human perception, collision detection, computer animation, real-time animation*

1 Introduction

With the high demand for highly interactive systems all over the world, the developers of animation systems are forced to produce more realistic, real-time animations. By exploiting user interactions and human perception of animation we can make the animation system realistic. Collision handling places high computational load on graphics workstations. From the human perception we find that any dynamic or casual event is dependant on the way in which objects are touching or colliding and on the way that they behave post collision. In computer graphics, we see that handling the various events of collision is a core part of any animation system and consumes a large proportion of the computation time for each step of a simulation. Thus we see several researches are made in collision handling in computer animation, both from the computational and perceptual perspectives. The solutions to address these collision handling issues in computer graphics include but not limited to adding more computational power, add hardware accelerators, and develop algorithms that may be implemented on multiple processors.

However, these solutions make the issues to be postponed rather than eliminating them. People are highly sensitive to any physical event that occurs in their area of concern. There are some basic events that any human would not accept say for example one solid object cannot merge into another; on seeing and sensing a solid object we assume the properties of the solid based on the way in which they interact with each other. People judge whether objects are animate or inanimate based on the perception that the objects are moving of their own volition, or being moved by another object (referred to as the perception of causality). Much recent research, listed in reference [1], has shown that attention is one of the most important and interesting features of human perception, which can also be exploited for improving the efficiency and realism of graphical systems. Time critical approaches trade accuracy for speed, simplifying where necessary, to meet the demands of real-time rates throughout the simulation [3].

With the increase in the number of independently moving objects in the scene, the computational load also increases. We can reduce perceived inaccuracy in an animation by taking perceptual factors into account, these are done by estimating where on the screen a viewer is looking, either using analysis from user survey, or by attaching more importance to certain objects or regions in a scene. Collision handling plays a major role in any physically based animation where we can save computation time by optimizing the speed or accuracy. One of the major bottlenecks in any physically based animation includes collision handling.

We present a method in which we trade various parameters from the viewer's perspective. Section 2 presents the motivation to this work. In Section 3, we describe an architecture which performs collision detection based inputs from human's perception. We see some test scenarios in Section 4. In Section 5 we see about human threshold. In Section 6 we see the details of our experiments and we conclude in Section 7 with the approach to handle collision in an optimized manner based on human perception.

2 Motivation

Designing any computer animation, especially with collision handling and detection is a challenging task. Human

character animation draped with cloth is a good example that requires intricate and precise collision handling. When objects move collisions that occur must be detected and processed robustly in a timely manner. If the object is a virtual human being collision between its clothes and hair, and self-collisions between limbs and digits must be considered. We should also be handling any physical interaction between the virtual human and the objects surrounding it such as touching, hitting and throwing. These actions are usually triggered by collision. If there are more complex objects in the environment, the burden on the computational machine which powers the animation will be higher. This leads to a greater need for extremely rapid collision detection. Increasing performance of any computer animation is often driven by hardware developments, but significant opportunities exist to increase performance via algorithmic improvement. Yet, human does not have keen ability to detect small physical incorrectness or to discern if any micro action must be handled in a particular way to preserve suspension of disbelief.

The main motivation for this work comes from the success of similar approaches used in stating that human perception is important for collision detection, where knowledge of the focus of humans that allows resolution and computational power to be concentrated on areas or objects in a display that are most important. We can make some savings in the computationally expensive area of collision detection and physical response determination. For example, considering low level of detail, proxy or fake objects could be used for collisions between objects that are not currently being attended or focused on by the viewer, or we can ignore response to the objects that are less likely to be noticed. We also want to determine if there is something about such dynamical events themselves i.e., their velocity or orientation or collision accuracy that attracts attention in some way. Based on the above assumptions we can say that we should be able to develop a model that does prediction for collision based on human perception.

3 Software Architecture

In this paper we explain briefly how collision detection for objects in 3D Space was implemented. Our approach started with having many objects colliding continuously. Then we did an in depth study of various parameters used in collision detection followed by the user study to gain knowledge about the user's perspective of various collision parameters. Based on the study we adjusted the collision handling algorithm to attain good speed up. We are using standard algorithms to compute intersection between ray and plane [1].

3.1 Model Geometry

Spheres were chosen as model objects due to their no orientation, no view point changes, and no deformation. The purpose of using a very simple geometry is to isolate the effect of individual human perception category of gap/penetration, velocity, and angle. For sphere, it's easy to predict the direction of movement after collision. While for other general rigid body objects, especially when they are occluded or spins with high angular velocity, the human ability to predict the behavior is drastically diminished and therefore human test subjects are having hard time to testify if a presented animation is right or wrong. This will lower the purity of perceptual threshold test on each category, resulting in general degradation. It would produce vague, unidentifiable source or perceptual errors. For example, if objects are sharply edged, human typically don't have ability to discern the consequential motion, so even very unlikely behaviors could be considered acceptable. To prevent this and to get direct relation between human perception and probable object behavior, a sphere was chosen as a model object. Also everyone has pretty good estimate of bounce off behavior of spheres.

3.2 Sphere-Sphere Collision Detection

A sphere is represented using its center and its radius. Determining if two spheres collide is easy. By finding the distance between the two centers we can determine if they intersect, if the distance is less than the sum of their two radiuses. The problem lies in determining if 2 moving spheres collide where 2 sphere move during a time step from one point to another. Their paths cross in-between but this is not enough to prove that an intersection occurred (they could not pass at a different time) nor can the collision point be determined. When complex shapes are used or when these equations are not available or can not be solved, a different method has to be used. The start points, endpoints, time step, velocity (direction of the sphere + speed) of the sphere and a method of how to compute intersections of static spheres is pretty straight forward. To compute the intersection, the time step has to be sliced up into smaller pieces. Then we move the spheres according to that sliced time step using its velocity, and check for collisions. If at any point collision is found (which means the spheres have already penetrated each other) then we take the previous position as the intersection point (we could start interpolating between these points to find the exact intersection position, but that is mostly not required).

If the time steps are smaller, the slices will increase with the accuracy of collision detection. Consider an example with time step 1 and with 3 slices In this case we would be checking the two balls for collision at the following time intervals 0, 0.33, 0.66, 1 and so on.

3.3 Collision Response

To determine how to respond after hitting static objects like bounding planes is as important as finding the collision point itself. Using the algorithms and functions described, the exact collision point, the normal at the collision point and the

time within a time step in which the collision occurs can be found. To determine how to respond to a collision, laws of physics have to be applied. When an object collides with the surface its direction changes that is it bounces off. The angle of the new direction (or reflection vector) with the normal at the collision point is the same as the original direction vector. When a sphere hits another sphere getting the collision response is much more difficult. Ordinary differential equations of particle dynamics have to be solved and the final velocities are determined.

3.4 Euler's Equation & Explosions

To simulate realistic movement with collisions, determining the collision point and computing the response is not enough. Movement based upon physical laws also has to be simulated. The most widely used method for doing this is using a simple first order Euler's method. As indicated all the computations are going to be performed using time steps. This means that the whole simulation is advanced in certain time steps during which all the movement, collision and response tests are performed. As an example we can advanced a simulation 2 sec. on each frame. Based on Euler equations, the velocity and position at each new time step is computed as follows:

Velocity_New = Velocity_Old + Acceleration*TimeStep

Position_New = Position_Old + Velocity_New*TimeStep

Now the objects are moved and tested against collision using this new velocity. The acceleration for each object is determined by accumulating the forces which are acted upon it and divide by its mass according to this equation: Force = mass * acceleration.

Every time a collision takes place an explosion is triggered at the collision point. A nice way to model explosions is to alpha blend two polygons which are perpendicular to each other and have as the center the point of interest (here intersection point). The polygons are scaled and disappear over time. The disappearing is done by changing the alpha values of the vertices from 1 to 0, over time. To be correct we had to sort the polygons from back to front according to their eye point distance, but disabling the Depth buffer writes (not reads) also does the trick. Notice that we limit our number of explosions to maximum 200 per frame, if additional explosions occur and the buffer is full, the explosion is discarded.

4 Test Scenarios

The following are the various test scenarios considered in our experiments.

4.1 Collision Gap Accuracy Test

Newton's third law of motion states that when two bodies collide they exert equal and opposite forces on one another, we can clearly see that human perception greatly relies on the dynamic properties of the world like mass of the objects, direction in which collision happens and the like. For example, a collision of 2 moving objects on collision will cause both the objects to move but there will be definitely change in direction of movement. The human brain appears to have very low-level processes that allow people to distinguish between animate and inanimate objects. Do we need each and every object in an animated environment to exactly collide? Or are we ok on some level of accuracy of collision? Let's say that as a user I'm ok with two objects if they overlap each other slightly. Using this test scenario for an animation environment we would be able to adjust the collision algorithm so that we could use the computational resources for other purposes.

4.2 Collision Velocity and Direction Test

This test scenario is used to determine how fast the objects movement should be so that users are good on the animation. We provide various adjustable parameters like variation in after collision speed, variation in initial velocity. We are also using factors such as size and speed of objects to choose the levels of detail at which to render objects in a scene before and after collision. Based on user's input like "I liked them better when they were moving with maximum speed" or, contradictorily: "I liked them better when they were side by side and moved with normal speed"; "They looked more bouncy when they were moving at medium speed"; we were able to change the collision algorithm to obtain the best collision effect.

4.3 After Collision Change in Angle Test

This test scenario is used to determine if the users are ok with a slight deviation in the collision angle. By default let's say the angle of deviation of objects is zero. Are the users ok with a 20 degree deviation of the colliding object after collision? Or the users are more particular on the degree of deviation of the colliding object saying it should be always zero. For this test we provide the users with series of varying sliders. We also note the direction of the deviation after collision. The users can also do combination of all the above test scenarios.

5 Human Perception Threshold

The collision effects can be exploited in a real-time application by tracking the user's fixation position. When the viewer is looking directly at a collision, it would be given a higher priority than a collision occurring at a slight different location, which itself would receive a higher priority than other collisions presented more peripherally. To achieve more

realistic animations we need to conduct many user surveys to get the human perception of animation. Several user surveys were carried out for the various collision handling scenarios like accuracy of separation between the colliding objects, velocity of the moving objects, angle of deviation after collision.

5.1 Accuracy

In this case objects that do not touch each other but leave a gap, decreases with increasing strangeness of the collision point. 2 Spheres Red and White were used. Collisions were presented at various accuracy levels ranging from -100% to 100%. There were many replications at each accuracy level, each in a different screen location. The order of presentation was also randomized. The users have the ability to pause or stop the collision at any time. The pause collision button can be turned on to view the exact accuracy. To verify the accuracy level we added accuracy detail radio button which gives us the exact accuracy when the collision gap goes over 0%. The spheres appeared and moved towards each other for 1 second, and then moved apart for 1 second after touching each other, leaving a small gap of 3 millimeters (mm) when the accuracy is set to 20%, or a larger gap of 2.8 centimeters (cm) when the accuracy is set to 100%. Based on the user survey we are able to see that people are more particular in accuracy.

5.2 Velocity

The objects that had more speed or velocity are compared to objects with minimum or normal speed. 2 Spheres Red and White were used. Collisions were presented at various post collision ball speed levels ranging from 0% to 200%. Default setting is at 100% of ball speed after collision. The users have the ability to pause or stop the collision at any time. The show direction button displays the pre and post collision velocities and direction. To verify the optimal velocity level we added show direction button which gives us the exact direction of the balls pre and post collision. The spheres appeared and moved towards each other for 1 second, and then moved apart for 1 second after touching each other, the after collision speed is doubled when the percentage is made to be 200% while it is reduced to half when we set it to 50%. The ball speed can be changed for one of the balls or for both.

5.3 Bounce off Angle

The objects that had 90^0 deviations are compared to objects with 0^0 or with objects that deviated 45^0. 2 Spheres Red and White were used. Collisions were presented at various post collision ball angle deviations ranging from -90^0 to 90^0. Default setting is at 0^0 of deviation post collision. The users have the ability to pause or stop the collision at any time. The show direction button displays the pre and post collision velocities and direction. To verify the bounce off angle we added show direction button which gives us the exact direction of the balls pre and post collision. The spheres appeared and moved towards each other for 1 second, and then moved apart for 1 second after touching each other, the after collision angle is made based on the selection.

6 Experiments

Collision events can be classified into different types listed below:

1. Objects approaching each other
2. Objects touching each other and then reversing
3. Objects showing repulsion effect.
4. Objects penetrating

By varying the accuracy percentage we are able to simulate the above collision events.

6.1 Collision Event

In our experiments, three types of collisions may occur (see figure 1): - "True" collisions, where entities touch, the collision is detected, and fully accurate collision response occurs. We may consider this as being the control situation for experimental purposes. (Figure 1)

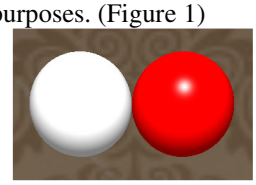

Figure 1: True Collision – 0% Collision Gap Accuracy

- Interpenetrations, where the entities also touch, but the collision are not detected or are ignored by the application. The entities are therefore allowed to continue on their previous path, even though it causes them to merge into each other to a greater or lesser degree. (Figure 2)

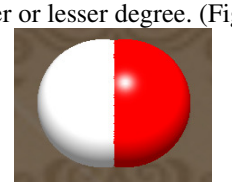

Figure 2: Interpenetrations – Negative Collision Gap Accuracy

- Repulsions, where the entities are close to each other but have not actually touched. (Figure 3)

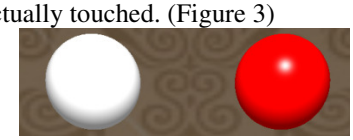

Figure 3: Repulsions – 100% Collision Gap Accuracy

The application decides to take the chance that they are actually touching, and accepts this situation as a true collision. This may then cause the entities to change their paths, causing a repulsion effect. The application can control which type of collision anomaly will occur most often, if not always. It could be argued that a detailed user study conducted led us to

our approach into which type of anomaly is most preferred by the viewer. However, based on our user study we were able to see that most of the users preferred true collisions as opposed to other collision events.

6.2 Effect of Velocity

In our experiments, two types of velocity changes are being considered:

1. Initial velocity Pre Collision for the two balls.
2. Post Collision Speed of One ball or two balls.

Our first experiment was to keep the velocity by default that is the initial velocity is 100%. The figure below shows the direction of the objects here both the balls have the same initial velocity but in the opposite direction. (Figure 4)

Figure 4: Initial Velocity: 100% for both objects

Now let's see the case when the initial velocity is increased to 200% the direction of the velocity is represented by the line. Here it's obvious that the velocity is higher. This is shown in (Figure 5).

Figure 5: Initial Velocity 200% for both objects

Similarly we did experiments on varying the after collision speed from 0% to 200% for one ball and also for 2 balls. Based on the user survey we are able to see that people are ok with a speed variation of 0% to 20% on an average.

6.3 Effect of Post Collision Angle

In this experiment, we study the effect of post collision angle change on collision detection. We could change the deviation angle from -90^0 to 90^0. (Figure 6) shows the collision effect with 0^0 angle of deviation

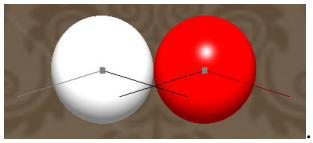

Figure 6: 0^0 Angle of Deviation after Collision

In the above figure the black lines represent the pre collision velocity and direction while the post collision is shown in grey and red colors. Now let's see the effect of changing the post collision angle from 0^0 to 90^0. The after collision angle is represented in the diagram below (Figure 7). Here we see that the new direction post collision is deviated 90^0.

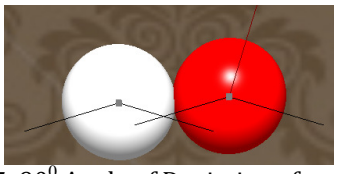

Figure 7: 90^0 Angle of Deviation after Collision

Based on the above experiments being held in our user study we were able to determine the tolerable limit for the post angle of deviation.

6.4 Location

In our experiments we also determine the useful field of view (UFV) for the perception of collision anomalies, i.e. to test what effect the location (e.g. above, below, above left etc...) of a collision has. In [Nies et al. 1998] they found that a 75 per cent confidence region for a visual search task was quite elliptic, with performance being better in the horizontal regions rather than in the vertical. Other effects were also evident, such as better detection to the right than to the left. They did find that there were big differences between subjects in the size and shape of the useful field of view. Hence, we will need to test collisions occurring in different regions, determined by the 9 cardinal directions, i.e. left, right, up, down, up-left, up-right, down-left, and down-right. The above experiments were conducted by dividing the screen into multiple sections 2, 4, 16 and studying the collision effect in specific areas. In the diagram below (Figure 8) we divide the screen into 4 sections with various accuracies of 0%, 25%, 50% and 100% accuracy.

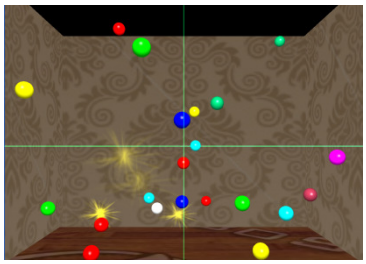

Figure 8: Screen with varying accuracy in 4 Section having 0%, 25%, 50%, and 100%

Based on our user survey we changed our algorithm by dividing the screen into sections and varying the accuracy at different sections, so that the accuracy in the corner of the room is < 100% while at the center of the screen we had 100%.

6.5 Results

It was essential to evaluate the techniques proposed, in order to justify and guide the later psychophysical studies. This set of experiments tested collision processing performance at the highest possible level of accuracy, i.e. processing every collision with 100% accuracy. In order to reach an overall target frame rate of at least 10 frames per

second, the collision handing should take up only a part of the 100 milliseconds available per frame. Therefore, in the absence of any load balancing schemes, we can estimate that at most 50 milliseconds should be spent on collision handling per frame.

We can see that collision handling times are unacceptably high for 50, 70 and 100 objects (Figure 9). We can see from this data that with large numbers of objects, our frame times will be too slow and variable for real-time performance.

Figure 9: Collision handling with 70 bouncing balls

The parameters used for this experiment are: Time: Time spent on collision handling and # Of Collisions: Number of collisions. The mean results for all measured parameters are shown in (Table 1). The experiment is run for 10 seconds and the results are gathered. Based on these results its clear that as the number of objects in the screen increase the time spent on collision handling increases. (Figure 10) provides a graphical overview of these data.

Table 1: Results from collision with no interruption

	20	30	50	70	100
Time (ms)	1	16	252	300	646
Collisions #	2	24	92	67	84

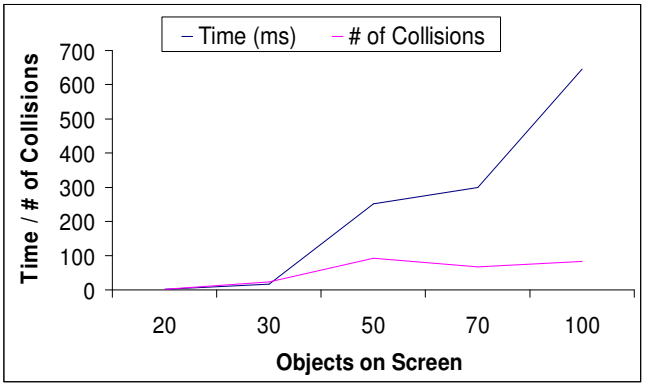

Figure10: Collision handling time with no interruption

The user survey shows us that viewers have the following tolerable level for the accuracy, velocity and angle. We took the mean of them all and we get accuracy tolerance = 1% that people are more concerned about accuracy; velocity tolerance = 5%; and angle tolerance = 20%. Taking into consideration of these values we did the analysis for 20, 30, 50, 70 and 100 objects with varying accuracy levels in the screen and our results are tabulated in the table below (Table 2). The experiment is run for 10 seconds and the results are gathered. Often binary search is used to detect exact collision time to precisely enforce contact conditions and repulsion. Finding of exact collision time becomes very complex when multiple collisions occur at a given time interval, and proper collision order sorting is also very time consuming. In the section below we see how the various criteria's promote the performance improvement of any animation system.

6.5.1 Collision Gap and Penetration

Collision gap adjustment, both gap and penetration allows us to eliminate costly backtracking to previous fraction of exact collision time search and collision order sorting operations. The ability to vary these criteria helps us to avoid spending more time on exact collision for every object. When we find the tolerable limit for collision gap or penetration we are able to apply the limit in the algorithm so that we could save the computational resources and lower the number of collisions.

6.5.2 Change of Angle and Velocity

By altering the bounce-off angles and velocity, we can intentionally avoid collisions that's about to occur. With some quick and approximate trajectory checks and collision time prediction, we can adjust the velocity and bounce off angles of post colliding objects to avoid eminent future collisions. This process is essentially altering the original behavior of simulated objects but since the alternation is within the differential threshold of human perceptual limit, they are unnoticeable and quite acceptable to users. Obviously we don't use an extensive search algorithm to avoid complicated secondary and tertiary level collisions, since our search algorithms is pretty simple one dimensional, i.e. looking into the direction of motion and see if there's any other ball in the direction of motion in a near field. Having a more sophisticated collision avoidance algorithm may further reduce the possible collisions globally but at the same time it takes more computation for the overhead, resulting in not much gain in overall throughputs.

The main advantage of this algorithm includes gain in computational power with the reduction in the number of collisions by adapting these limits. When the accuracy of collision is varied most collisions are ignored and therefore the numbers of collisions are reduced. But the animation is realistic since the actual collisions happen in the center of the screen. The behavior of the animation is different when the collisions in the peripherals are ignored. But since these surveys are based on human perception we find the collisions to be more realistic by considering collisions only in the center of the screen and ignoring the collisions at the peripherals. (Figure 11) provides a graphical overview of these data. From the two graphs we see a gain of about 7 times in the collision handling time spent when we adjusted our collision handling algorithm.

Table 2: Results of Collision with user study input

	20	30	50	70	100
Time (ms)	0	30	31	30	94
Collisions #	1	12	13	9	8

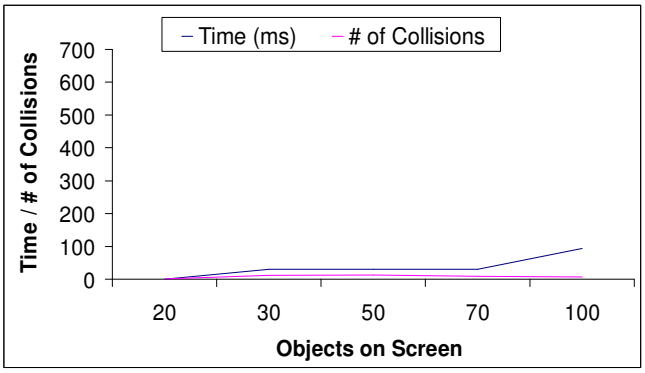

Figure 11: Collision handling time with user study input

7 Conclusions

In this paper we study and provide a solution to handle efficient collisions between objects by exploiting human perceptual threshold. We noticed that there is significant role of human perception by adjusting the various parameters to achieve speed up. We have developed the perceptually-adaptive collision detection algorithm that essentially alters the true correct object behaviors within the perceptual limit to minimize the number of collisions. The algorithm parameters are adjusted based on human subject test to get better speed up. The main algorithm is described in Section 3 and 4, along with a new model of human visual perception of collisions. In Section 5 and 6 we showed the feasibility of using this model as the basis for perceptual scheduling of collisions in a real-time animation of large numbers of homogeneous objects. It has been demonstrated that by using a ray based collision algorithm, perceived inaccuracy can be significantly improved. As described in Section 3 and 4, we validated this model psychophysically. This work is not only relevant for the problem of Collision Detection, but also for other applications where the processing of fine detail leads to a computational bottleneck. Our model could easily be adapted to accomplish perceptually-sensitive real-time shadows, shading, and other such techniques. Our psychophysical experiments can provide a starting point for other computer graphics practitioners who wish to enhance the realism of their real-time animations. For future work we would continue to work on multi level search, and global collision number optimization. We would also work on more in-depth studies for generic objects including deformable objects as our viable next steps. We think we have also contributed to the fields of Human Computer Interaction for interactive computer systems. The work in this paper can also be used in conjunction with analysis in functionality of human brain.

8 References

[1] Carol O'Sullivan. ; Collisions and Attention, ACM Transactions on Applied Perception, 2(3), pp309- 321, (2005)

[2]http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=30

[3] John Dingliana and Carol O'Sullivan; Graceful Degradation of Collision handling in physically based animation, Computer Graphics Forum (Eurographics 2000), 19(3, pp239 - 247, (2000)

[4] Palmer, I.J.; and Grimsdale, R.L.; Collision detection for animation using sphere-trees. Computer Graphics Forum, v14 i2. 105-116

[5] C. Lauterbach; S. Yoon; D. Tuft; D. Minorca; RT-DEFORM: interactive ray tracing of dynamic scenes using BVHs, in: IEEE Symposium on Interactive Ray Tracing, 2006, pp. 39-46

[6] Carol O'Sullivan and John Dingliana; Collisions and Perception, ACM Transactions on Graphics. Vol. 20, No. 3. July 2001.

[7] Christer Ericson.; Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology), Morgan Kaufmann Publishers Inc., San Francisco, CA, 2004

[8] C. Lauterbach; S. Yoon; D. Tuft; D. Minorca; RTDEFORM: interactive ray tracing of dynamic scenes using
BVHs, in: IEEE Symposium on Interactive Ray Tracing, 2006, pp. 39-46

[9] Carol O'Sullivan and Richard Lee; Collisions and Attention, ACM SIGGRAPH Symposium on Applied perception in graphics and visualization (APGV'04), pp165, (2004)