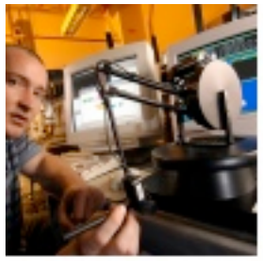
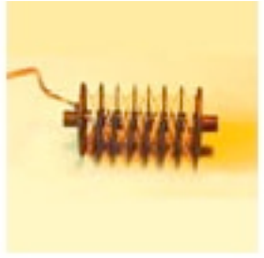
CS277 - Experimental Haptics
Lecture 9a

Course Projects, Past & Present







Course Project Requirements

- Final project is worth 40% (40 points)
  - Project proposal (5 points)
    - May 6th
  - Project milestone/presentation (5 points)
    - May 20th and May 22nd
  - Project demonstration & showcase (30 points)
    - May 30th

Project Proposal

- Should be (only) 1-2 pages long
- Give instructors an idea of what you intend to accomplish
- We will give you feedback with respect to suitability and scope
- Submit one proposal per team (one or two people) by May 6th

Project Milestone/Presentation

- In-class presentation
  - Project goals
  - Demo of intermediate results
- Fundamental aspects of your project should be in place and working reasonably well
- Helps us to detect problems early on and to help you with unanticipated challenges

Project Showcase

- Joint exhibition with ME327
  - Friday, May 30th, 1:30-3:00 PM
  - Building 550 atrium, grove area
- Distinguished guests, high-profile exposure!
- 1-page abstract = no comprehensive report
- Let us know well ahead of time if you have a schedule conflict

Course Project Ideas

- Video game, simulation, or advanced algorithm/technique
  - Physics/dynamics simulations
  - Collaborative or competitive networked haptics
  - Rich haptic environments
- Let's see some examples!

Haptic Airhockey

Denise Jones, Min Young Kim

The objective of this project is to build an air hockey game with haptics. The main components of the game are the dynamics simulation, the force control, and the sound control. This is a one player game, with artificial intelligence used for the movement of the opponent's paddle.

Fig 1. Start screen

In order to simulate the true feeling of air hockey, the surface of the table is soft, as though the paddle is on a layer of air, though the side walls are solid. In addition, there are three different scenarios for dealing with the collision between the puck and the player's paddle. If the paddle is stationary, the puck is simply reflected, and the force is proportional to the pucks change in velocity. If the difference between the puck velocity and the paddle velocity is very small, then it is assumed that the collisions between the paddle and puck will be inelastic. This is because for a small amount of time the puck and the paddle will move together at the same velocity as is the case during inelastic collisions. The force felt on the paddle is equal to the pucks mass times the acceleration applied to it. If both the paddle and puck are moving quickly toward each other and a collision occurs, then elastic collision equations are used to calculate the final velocity of the puck and the force felt on the paddle.

Fig 2. Attempted Cheating

As an added bonus, an anti-cheating effect was implemented. Since it is not allowed in air hockey to pick up and move the puck, when a user picks up the puck with the paddle, a spring force pulls the user's paddle with the puck back toward the table.

We used BASS sound library and several sound data for implementing unique sounds into the game. There are "clanks" when the puck hits the wall or a paddle. There is also a spring sound while the user is cheating. The frequency and volume of these sounds are affected by the force reaction of each of the events. There are also unique sounds when the user achieves a goal, the AI achieves a goal, or if the user wins or loses. Finally there is 3D background music while playing the game.

Lock Picking Simulation

David Johnson

Picking a lock is one of the few activities that doesn't use any sense other than touch. Everything that happens inside of the lock is hidden from the person's view and the only way to get information from the lock is through the forces you feel through the lock picks.

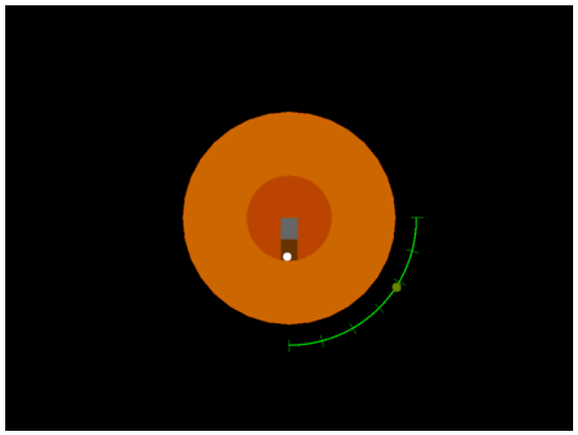


Fig 1. Front view of the lock

This program gives you the ability to see what is happening inside the lock as you pick it. When you get the pin to the right height it changes color as you feel the change in force. By giving visual cues that correspond with the forces, this program should be able to train a person to recognise the slight changes in force that are associated with setting the pin to the correct height. It also teaches you how the torque affects how the pins feel. Adjusting the torque can play an important role in how easy it is to identify the next pin to set and how easy it is to set the pin without lifting it too far.

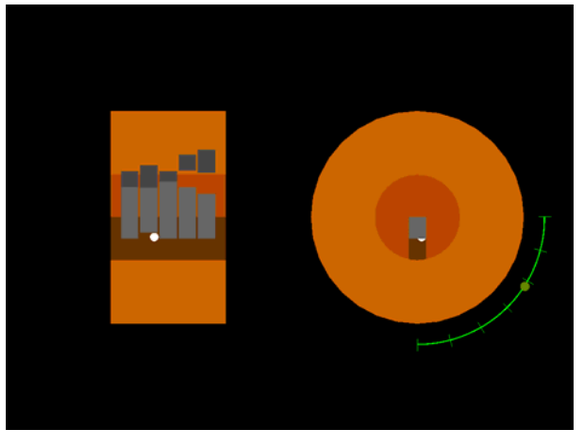


Fig 2. Cross section of the lock so user can see what is happening inside

How to Pick a Lock

You first use a tension tool to apply torque to the keyhole, like you would when you turn a key. As the lock tries to rotate it gets blocked by one of the pins. This pin becomes difficult to push up on because of the sideway pressure on it. Putting a lock pick inside the keyhole, you lift up on each of the pins you find the one that is more difficult to lift than the others. This is the pin that must be set to the correct height next.

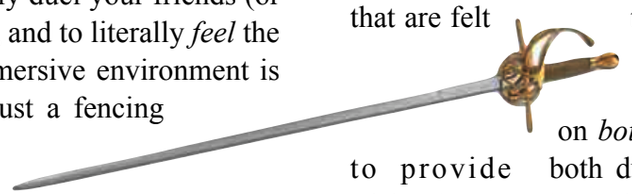
As you are lifting that pin, at some point you should feel the force from the pin suddenly decrease. This happens when the pin is at the correct height. If you stop lifting the pin at this point the pin will remain at this correct height. The lock will rotate very slightly and catch on another pin. If you repeat the same procedure for each pin and get all the pins to their correct heights the lock will be free to turn and will open.

MAXIMUM FENCER Deluxe

Sonny Chan & Robert Wilson

Maximum Experience

Prepare yourself for the ultimate virtual fencing experience! By taking advantage of the latest in haptic interface technologies, Maximum Fencing Deluxe delivers a truly unique experience that cannot be replicated using any traditional game interfaces. It allows you to virtually and safely duel your friends (or foes) over your local network, and to literally *feel* the wrath of their blade. The immersive environment is so real that it is no longer just a fencing game, but a *simulation*!



Maximum Haptics

Point proxies are extremely simple, useful, and intuitive. When it comes to modeling blades, though, they just don't cut it. Maximum Fencer Deluxe features a ray proxy approach similar to one described by Mitra and Niemeyer (Mitra & Niemeyer, *Presence* 16:4, 2007) that treats each blade as a line segment. The player is coupled to the virtual environment through a digital spring-damper between the device location and the "manipulation point" on the blade, where a fencer would normally control the blade with his thumb and finger. Contact between the blades, guards, and bodies is handled using velocity constraints. These regulate the velocities at the closest points between two objects so that penetration of objects cannot occur during an integration time step in the dynamics engine. As a result, the user feels contact forces complete with mechanical advantage. To complete the haptic immersion, physical properties of the blade such as mass, inertia, the first bending mode, and friction are modeled.

Maximum Immersion

The dueling action of Maximum Fencer Deluxe takes place within a fully immersive 3D environment. Carefully animated blades and human models allow the duelist to see all the details that are felt through the haptic interface. Full-speed haptics and dynamics simulations are run on *both* the host and client systems to provide both duelists with the ultimate feel of fencing. Finally, a full musical score and a set of finely tuned sound effects complete the illusion of really being there!

Maximum Strategy

To succeed at Maximum Fencer Deluxe the initiate must learn fundamental skills mirroring the sport of fencing. Parries, disengages, and advance attacks are all critical, and maintaining proper distance and timing remain central. With training, two duelists can execute advanced strategies such as feints and combination attacks.

Powered by: CHAI 3D, NOVINT, 3D ENGINE, OpenGL, fmod music & sound effects system

Haptic MasterMind

Santhi Elayaperumal, Matt Montgomery

Introduction

Haptic MasterMind is a "touchy-feely" twist on a classic strategy game. In traditional MasterMind, the player breaks a code based on colourful round pegs. From a choice of six pegs, any combination, including repetition, of four pegs make up the code. In this haptic version, balls of various textures replace the colourful pegs, and the player "feels" his or her way out of the code. The balls can be placed in one of four slots, each representing a place in the code. Based on scored guesses, the user attempts to break the code.

Scoring

The default "hard" scoring method is such:

A number with a black background tells how many balls have the correct haptic texture in the correct slot.

A number with a white background tells how many balls have the correct haptic texture but are in the incorrect slot.

Pressing "E" on the keyboard turns on "easy" scoring, which gives a score based on each slot position.

A green circle means "correct effect and correct position."
A yellow circle means "correct effect and wrong position."
A red "x" means "wrong effect," texture is not in the code.

Haptic Ball Types

Of the six haptic balls to choose from, three are "squishy" and three are "stiff". Of the two types, there are three effects:

1.) Rubbery

2.) Magnetic

3.) Vibrating

The "squishy" objects are deformable, and were created using the GEL-Module dynamics libraries in CHAI3D. The deformable objects will change shape and surface effects when the cursor interacts with them. The stiff objects will not deform. To quickly see if a ball is deformable or stiff, the player can repeatedly tap the cursor on it.

Game Environment

Figure 1 shows an example of a game in play.

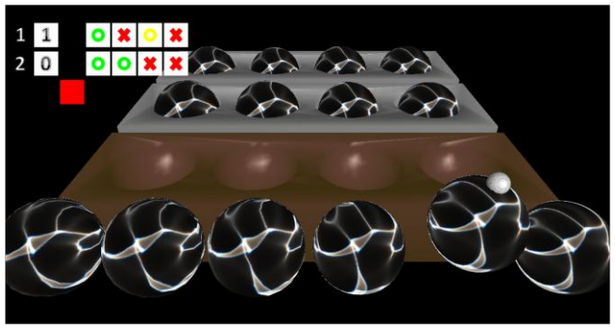


Figure 1. The game environment while in play. The active row is highlighted, and its score is followed by the red square. In "easy" scoring, both scoring methods are shown.

The six playable balls to choose from appear in front of the "active row". These balls will always have the same haptic effect based on their initial position. Pick up a ball by moving near it and pressing the CENTER button on the Novint Falcon.

Once over the desired slot in the active row, drop it into the tray by letting go of the CENTER button. To delete a ball in play, pull it out of the slot, and while holding onto the CENTER button, press "D".

When all four slots are full, press the RIGHT button on the Falcon to get scored. If the code has not been broken, a new tray will appear. It is possible to scroll back into the workspace to feel balls in the previously played rows. The row will then highlight and the corresponding score will be indicated by a red square.

The game environment shows the haptic effects and forces between the multiple objects in the screen. Force optimizing methods were implemented to only render forces on objects near the cursor, in order to keep up the haptic rendering rate. The game programming makes use of multiple structures which inherit cClasses within the CHAI3D library.

Haptic Toothbrush

Sammy Y. Long

Haptic Toothbrush offers the exciting experience brushing the virtual teeth with our favorite device Novint Falcon! The body and bristles of the tooth brush realistically turn and bend as they interact with the teeth. User can rotate the device as they like.

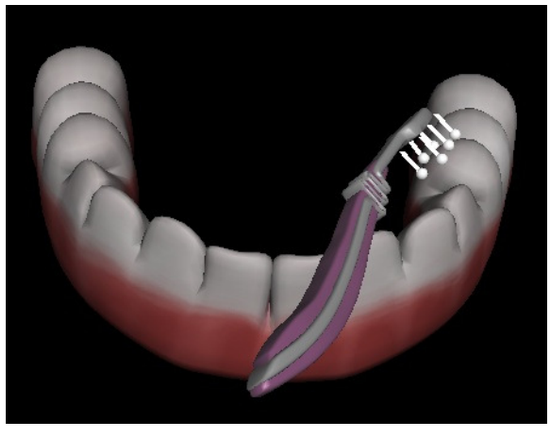


Fig 1 Haptic Toothbrush

Algorithms behind the Brush

1. *Multiple Interacting Points (god-object)*

I extended CHAI3D `cGeneric3dofPointer` to a toothbrush mesh with 6 3dofPointers attached which represent the end points of each bristle. God-object algorithm detects if the bristle penetrates the tooth, updates the proxy, and computes the contact force.

2. *Virtual Coupling Force & Torque*

A virtual coupling spring is added between the virtual brush (center of mass) and the real device position, with a damping component (Figure2). In each time step, the position of the brush is updated by Hooke's Law, given the two forces applied on it: the force from the bristles ($F_{bristle}$; in green; closer look in Fig3), and the force from the coupling spring (F_c; in black). Explicit Euler Integration is performed.

$$F_{brush} = F_{bristle} + F_c$$

The orientation of the toothbrush is updated, in every time step, from the sum of the two torque exerted on it: one resulted from $F_{bristle}$, and the other from the torsion spring.

$$T_{brush} = r \times F_{bristle} - k_R\theta - b_R\omega$$

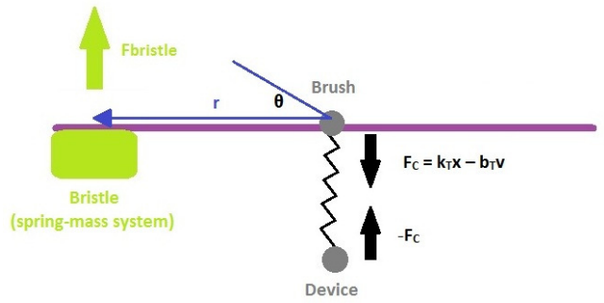


Fig 2 Virtual Coupling

3. *Spring-Mass Dynamic System for each Bristle*

Each bristle is modelled by a spring which connects its end point and its attachment point to the handle, named it "bristle spring" (Fig3). The attachment points are fixed relative to the brush handle. Each end point position is updated according to the two forces exerted on it: F_{attach}, the spring force from the attachment spring, and F_{avatar}, the spring force which models the contact (god-object).

$F_{bristle}$, the total force exerted by the bristle system used in the previous steps, is the sum of all individual spring force from each bristle.

$$F_{bristle} = \text{sum}\,(F_{attach}[i]),\ 0 \le i < 6$$

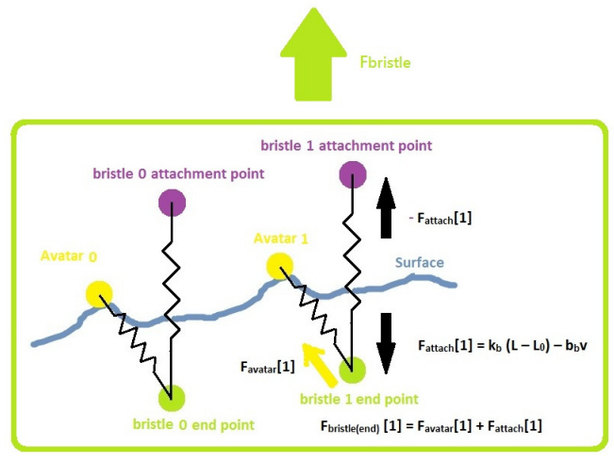


Fig 3 the Bristle Spring-Mass System

Reference:

[1] CHAI3D
[2] DAB: Interactive Haptic Painting with 3D Virtual Brushes. Bill et al. Department of Computer Science, University of North Carolina Chapel Hill, NC

Haptic Water Polo

David Pursell

Haptic water polo is a game that tries to emulate some of the fundamental individual aspects of water polo, specifically carrying and shooting a ball. In water polo, there are goals on each side of the arena, and the primary objective is to throw the ball into the opposing goal. One unique aspect of the game is that no players except the goalkeepers may touch the ball with both hands simultaneously, so all catching and shooting must be done with a single hand. This makes the game a prime candidate for haptic representation, since these actions can be intuitively conveyed through a single controller.

The most interesting part of this simulation is how to control the ball. The two main goals of this project were that the player must feel as if he is holding a weight in his hand when his avatar is holding the ball, and when the ball is shot it must travel more or less in the direction that the player wanted.

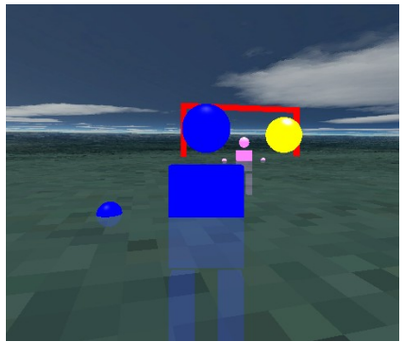


Fig 1. Holding the ball in shooting position

The first goal, feeling the weight of the ball, is simple if the player is holding the ball still. However, fakes, an integral part of water polo, are performed by bringing the arm partially forward as if shooting, but then neglecting to release the ball and bringing the arm back again. When the user performs a fake, the program calculates a discrete approximation of the ball acceleration to apply a counteractive force, so that the player will feel like the ball is continuing to pull his arm forward as he is trying to halt the fake.

The second goal, accuracy while shooting, is achieved by assuming an exact coupling between hand position and ball position. This means that instead of simulating momentum and forces due to the hand pushing on the ball, the program instead assumes that ball velocity is equal to the user's hand velocity. While not quite physically accurate because it doesn't account for things like skin pliancy and finger deformation, this assumption allows for a reasonably accurate and controllable ball velocity

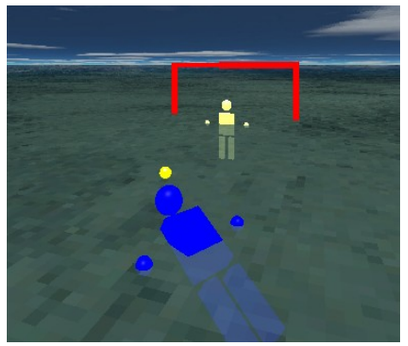


Fig 2. Swimming towards the ball

One additional goal for this project was to limit the user input to the haptic device itself. This becomes a challenge due to the variety of actions that need to be taken, for instance, the user needs to control both which direction the avatar swims as well as how the avatar's arm moves while shooting the ball. The solution implemented here is to divide the controller's workspace into halves. When the user is in the bottom half of the workspace, the avatar is swimming; in the top half, the avatar is vertical in the water (called “eggbeatering” in water polo). This leads to an intuitive control scheme – when the user wants to lift the ball out of the water, he just raises his hand – that uses only the haptic device for input.

Realistic Slicing Sensations using Haptics

Alex Quach

The goal of my project is to accurately model the forces involved when slicing through various deformable objects using a knife. The feeling of slicing through an orange, a piece of meat, a cucumber, or a tomato is unique to the sliced object, and my project attempts to capture the various sensations that come from slicing through each of these objects. I came up with a parameterized model for expressing the deformability, surface stiffness, and other properties of the layers of each object. For example, the orange's peel would have a different feeling than the fruit itself.

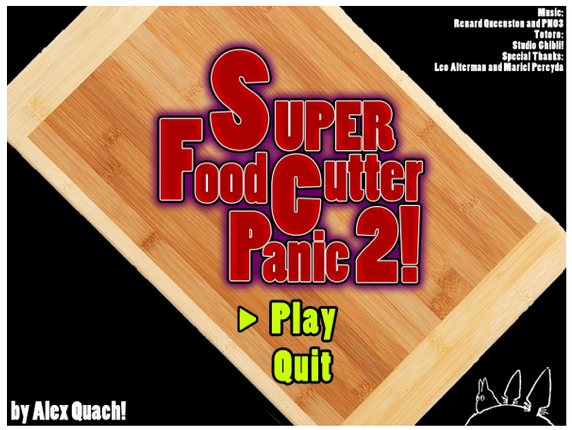


Fig 1. Title screen.

In Super Food Cutter Panic 2, the player must cleanly cut through the given object without mangling it. The player begins with an easy object like tofu, before moving on to more difficult objects like oranges and eggs. The player must carefully use the haptic device to carefully bisect the object, sawing back and forth as necessary. The knife probably will not be serrated, so it may take some vigorous sawing to get through some of the objects.

Fig 2. Gameplay.

Getting the objects to be simulated well took some effort, but addressing stability and the slicing of the polygonal model was extremely difficult. Slicing is notoriously difficult to do using standard polygonal models. Voxels can help, but require completely different algorithms that I didn't have time to implement. As for the forces, I realized quickly that the haptics device could not generate forces large enough to give an adequate cutting sensation, and that the haptics device cannot detect force. Because cutting revolves more around the force applied, I had to make major concessions in my theoretical cutting model to make it work. Furthermore, I ran into stability issues, because I was often being caught in small areas with large deltas in forces applied. Since the knife is wedged in the food and therefore has normal forces from every direction, it's extremely easy for the device to become unstable. A lot of my effort went into attempting to keep the device stable while ensuring a good experience. Although I didn't get as realistic an experience as I wanted, I learned a great deal about the problems that haptics researchers often encounter, and got a lot of experience with constructing a video game.

Crosscut Saw Simulation

John Jessen

Overview

This project attempts to realistically simulate the feeling of sawing through a tree with a crosscut handsaw. Forces against the user's hand are implemented using the chai 3D haptic rendering library and a Novint Falcon haptic device. Rendering of the tree structure was implemented using OpenGL and C++. Sound was added using SFML.

Implementation: Haptics

To simulate the forces applied to the saw when sawing through the tree, a set of points representing the tip of each blade on the saw were defined relative to the position of the haptic tool in the graphical environment. To allow for fast collision detection between the saw points and the tree, the tree is rendered and defined by a set of cubic voxels arranged into the shape of a cylinder. When it is determined that at least one of the points of the saw has penetrated the surface of the tree, the projected point of the haptic tool position is locked into place until a significant enough force is applied to the haptic device to destroy the intersected voxels.

Figure 1. **Collision detection and rendering of the tree were implemented using voxels and point quads.**

This force is determined by the distance of the tool from the projected point. To determine when the user has applied enough force to destroy the voxels intersected by the saw, the force applied by the user is divided by the number of voxels intersected by the points of the saw. If the resulting value is larger than some predefined voxel strength threshold, then the intersected voxels are deleted and the projected point is moved in the direction of the actual tool position.

Implementation: Graphics

Using voxels also allows for easy rendering of the deformation of the tree. For each voxel present in the environment, a point quad is rendered, where the color of the quad depends on the distance of the quad from the center of the tree. As voxels are destroyed, fewer points are rendered, creating the illusion of actually sawing away parts of the tree. A simple particle system was also used to show sawdust emanating from a lateral cut on the tree.

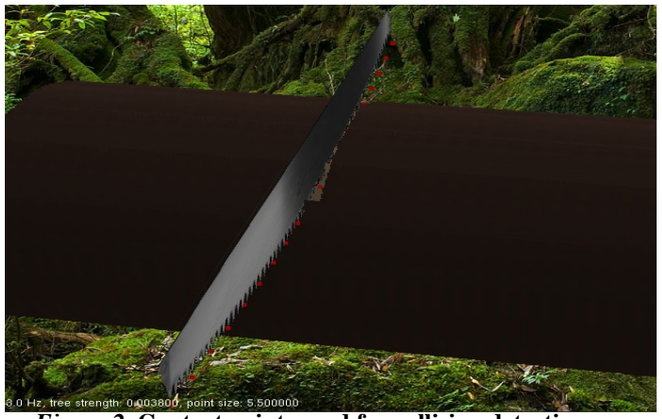


Figure 2. **Contact points used for collision detection are presented in red.**

Implementation: Sound

In addition to simulating the visual and haptic aspects of sawing, two sound tracks of a person sawing forward and backward through a tree were used. While the approach used in this project is somewhat lacking in realism, a portion of the forward or backward sound tracks were played depending on the location the saw gets stuck in the tree and the distance by which the saw moves after destroying a set of voxels. To vary the sawing sound, the pitch and volume of each incremental sawing sound are randomly tweaked from the source sound file.

User Input

Pressing 3 and 4 will increase and decrease the point quad size. Pressing 1 and 2 will increase and decrease the strength of each voxel in the tree, making the tree harder to saw. The cuts made to the tree can be reset by pressing R, and the points used for collision detection between the saw and the voxels can be turned on and off by pressing P.

References:

"An amputation simulator with bone sawing haptic interaction", M.S. Hsieh, M.D. Tsai, Y.D. Yeh, Oct. 2005

"Visuohaptic Simulation of Bone Surgery for Training and Evaluation", D. Morris, C. Sewell, F. Barbagli, K. Salisbury, N. Blevins, S. Girod, Nov. 2006

RHYTHM GAME WITH HAPTIC DEVICE

Chintan Hossain, Yanzhu Du

Introduction

Rhythm games are a popular genre of games which simulate playing an instrument. Notes drop down into view and the player must play the right notes at the right time. The more accurately the notes are played, the higher the score. Almost all rhythm games available currently require a dedicated controller to play. Haptic devices are well suited to replace dedicated controllers in rhythm games, because a haptic device can simulate tactile interactions with a musical instrument.

Gameplay

There are four drums within the workspace of the haptic device, which are rendered as rigid objects. Every time the avatar hits the top of one of the drums, a drum sound is played. The target notes for each of the drums drop into view. When a drum is hit, the player gains score according to the timing accuracy of the drum hit.

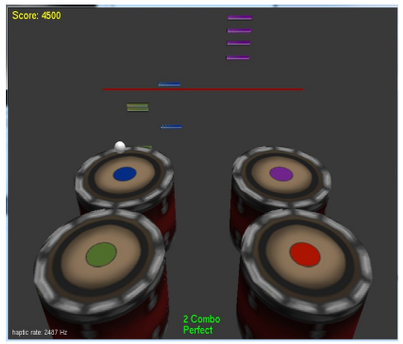


Fig 1. Screenshot of the Haptic Drum game

The game can be played with multiple devices, turning it into a 2-handed game. One avatar is displayed for each haptic device, and each device can hit any drum within the workspace. Using one device for each hand makes the game more fun, and allows more complex and challenging beats to be played.

Event Based Haptics

The game makes use of event based haptics to make the drumheads feel stiffer and more realistic. When a drum head is hit, the program generates a momentum impulse perpendicular to the drum head. This causes user to feel a sharp kick upon hitting the drum head, which makes it feel like a rigid object. The momentum impulse is spread out over several haptic rendering time steps due to the maximum force limit of the device.

Fig 2. A momentum pulse, illustrated in blue, is applied upon contact with a drumhead. This kicks the avatar away, making the drum feel stiffer.

Haptic Pottery

Rifat R Joyee, Narendran Thiagarajan

Haptic pottery is a single person game for pottery enthusiasts to get "hands-on" with the art of pot making. Players use the falcon to haptically interact with the clay on the pottery wheel. A variety of tools allow different types of interaction with the clay to make different types of products. The sound effects add more realism to the experience. The haptic pottery has a free form mode and a game mode - where the players are asked to remake a pot from a given picture.

Fig 1. Starting point: Clay on the wheel

User Experience

Pottery is made by deforming clay into objects of a required shape. Here we simulate pottery by starting with cylindrical mount of clay. The player uses the falcon to touch the surface of clay and shape it similar to real pottery. There are various tools they can use to curve out different shapes. We have implemented three different tools - *cylinder, sphere* and *point tool*. The point tool makes very narrow groove in the pottery clay, the cylindrical tool curves out larger area easily and the sphere tool is similar to the cylinder but with a round object.

Game Mechanics

The game starts with a target shape and challenges the player to achieve that using the toolkit. The score measures how close the user is to the target. As the user levels up, the shapes get harder and the wheel spins faster. Different tools can be selected by pressing *1 to 4*. Use *'a', 's'* and *'d'* to select the level.

Fig 2. Finished pot

Techniques

The technique we use to render the pottery is Cylindrical Element Method (by Han, et al.). This techniques is based on the fact that clay pots on a turning wheel are symmetric about a central vertical axis. During collision detection, the position of the tool at any point is known. Also the since the cylinders are stacked vertically there by greatly decreasing the number of cylinders to check collision with. If collision is detected 3 different forces act on the tool - Stiffness, Viscosity and Friction.

Texture and Sounds

To make the graphic experience richer we used texture on the pottery wheel, wet clay and background. We also use SDL sound library to simulate the electric motor under the wheel and the background environment. The ambient sound is always played for a pleasant user experience. There are different sounds for the rotating wheel and the tool shaping the clay.

Staff Fighting Simulator

Aubrey Shapero, Dillon McCoy

This program uses two Novint Falcons on one computer. The devices are set up back-to-back, and users grip them from the sides. This simulates the sensation of grabbing a staff with two hands. The user can feel the forces and torques according to the actual physics of holding a rigid stick in two locations.

Fig 1. Two Falcons back-to-back

Our program also implements a new feature that we call "dynamic gripping." There are many programs that use proxies other than plain spheres, but few programs, if any, allow the user to change the interaction point on the non-spherical proxy. Changing the interaction point on a non-spherical proxy is non-intuitive and strange, assuming there is one interaction point. However, a staff proxy is the simplest non-spherical proxy in which changing the interaction points along the proxy is intuitive and quite natural.

In this program, there are default positions for the hand proxies, but the user can change his/her hand positions by gripping the devices described as follows: Gripping either hand will make that hand the "fixed" hand, and the other the "free" hand. When in this configuration, the position on the staff for the fixed hand gets fixed, and the staff proxy must go through the position of where the free hand is, but the free hand is allowed to slide along staff.

Additionally, the user will experience a number of different forces. First, there are the low frequency forces based solely on the position of the two devices versus the position of the proxy staff. Using

$$F_1 + F_2 = F_{total} \quad \text{and}$$

$$F_1 d_1 + F_2 d_2 = F_{total}\, d_{total}$$

allows us to calculate what low-frequency force each hand should feel, where F_{total} is the total force being felt, F_1 , F_2 are the left and right hand forces respectively, and d_1,d_2 are the distances of the left and right hands to the left end of the proxy staff, respectively.

The next force applied is the impulse response force, which is an implementation of event-based haptics in which the program applies a decaying sinusoid wave train of force on top of the low frequency force following the same equations presented before.

The last applied collision forces are the vibration forces, in which we model the vibration modes of an unsupported beam. The force felt at the left hand (and right hand similarly) for each mode is

$$F_L = A * |V| * \sin(2\pi f) * G_L * E_{impact}$$

where A is scalar, |V| is the speed at the point of the impact, and G_L and E_{impact} are the "gain" and the "excitation." G_L and E_{impact} are amplitudes of the unsupported beam equation at the points of the left hand and the collision point, respectively, for that mode.

Users also feel the weight of the staff and a longitudinal force towards the hand proxies if both hands are supposed to be fixed.

Fig 2. The pink spheres show the hand positions of the user when fighting the computer.

Finally, in the actual game, the player will have to block a pre-programmed strike and strike the computer avatar.

Haptic Jenga

Evan Jeng

Jenga is a popular game of strategy and manual dexterity. The game is played with 54 wooden blocks, each measuring (1.5 x 2.5 x 7.5 cm). To setup the game, the blocks are stacked into a tower with three blocks placed adjacently per level. The orientation of the blocks alternate every level, for example, if blocks in a given level lie north-south, then the next level will have the blocks lying east-west.

Once the tower is set up, players take turn to remove a single block from the tower and placing the block at the top of the tower. Only one hand may be used at a time during play. The game ends when any block falls from the tower, other than the block being knocked out to move to the top.

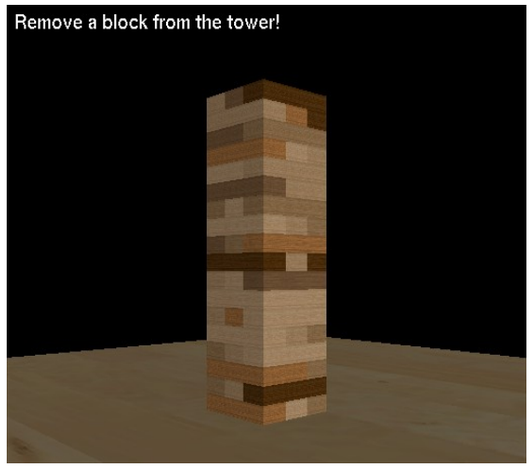


Fig 1. Initial game setup

Our game keeps track of the current phase within the turn (i.e. remove or replace) and instructs the player to act accordingly. During the remove block phase, the player can either push a block out of the tower, or press the user button on the Novint Falcon to enable grasping and pulling the block out. As in reality, a larger frictional force is experienced when operating on blocks near the bottom of the stack due to the weight of the blocks above. During the replace block phase, the game lets the player select from three potential spaces at the top of the tower, in which pressing the user button confirms the selection. If at any time, more than one block falls from the tower, the game comes to an end. However, the loser is permitted to vent his frustration by knocking around and throwing blocks from the collapsed pile before starting a new game.

Fig 2. The game ends when the tower collapses.

In addition to wood, different types of block materials (steel, glass, rubber) are selectable for an alternate gaming experience.

Camera controls are simple and intuitive through either keyboard control or moving the cursor off-screen in the desired direction.

The game was implemented in C++ using chai3d libraries and the Open Dynamics Engine for physics.

HAPTICUDA

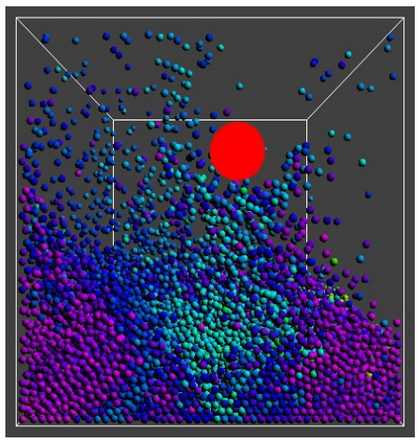
Nan Jiang

The high refresh rate of haptic rendering can be a computational challenge. Relying on the CPU alone limits what a user can experience on a haptic device. In recent years, games have been using the GPU not only for graphics rendering but also for complex physics simulations. In this project we bring haptics onto the GPU as well. By adding haptics extensions into many existing physics simulations, a user can experience a wide array of complex environments.

The physics simulations are offloaded to the GPU using NVIDIA CUDA. Many particle and grid based physics algorithms exist that take advantage of the computation power of the GPU. We have included three simulation demos.

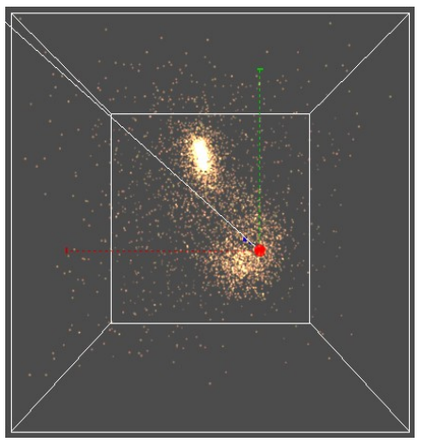
The Ball Pit

This simulation is based on an existing demo from the CUDA SDK. The GPU performs the dynamic collision of 32K particles. The haptic device controls the motion and senses the impact on a large ball. Though the force feedback is simply the sum of colliding forces, you easily feel the granularity of each particle.



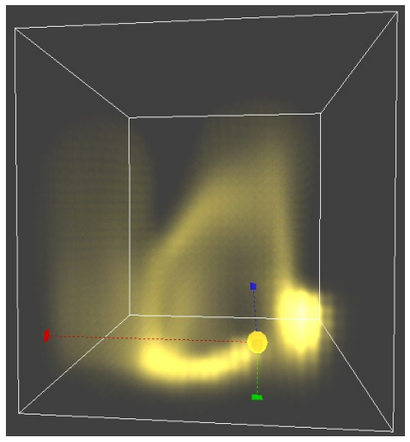
Mass vs. Anti-mass

Another simulation based on a CUDA SDK demo. Forces between particles are generated through gravitational potential fields. The GPU calculates the pair interaction between 8K particles and the haptic cursor.



Ink Drop

This simulation is built from scratch based on existing stable fluid algorithms. An invisible fluid responds to the movement of the haptic cursor. Ink can also be injected to visually track the behaviour of the fluid. The haptic feedback is based on pressure and velocity of the fluid surrounding the cursor. System stability can be problem when the fluid both respond and acts on the haptic device. As ink, velocity, and pressure are processed through 262K grid elements, making a mess has never been this easy.



Fruit Ninja Real

Yu-Ta Lu

Fruit Ninja Real is a game in which some fruits are shot into the air and then fall, and the player's task is to cut through the fruits while they are in the air before falling out of the screen to get high scores. The game idea is the same as the iPhone game Fruit Ninja. However, in Fruit Ninja Real, the game experience is augmented with the usage of the haptic device - Falcon. The player is now operating Falcon to control the cutter in the virtual 3D world, and when the cutter slices through the fruits, the corresponding force will be passed back to the player as if he/she is cutting fruits in the air in real world.

In the game, the fruits follow physical rules. This means their motion will be affected by the force applied to it. They rotate if the player is not applying force right toward its center, and they are pushed away if the player is not slicing them hard enough to cut through them.

Fig 1. Fruit Ninja Real Game Play

Method

I modeled the fruits using a multilayer approach. Imagine we want to model a watermelon. A watermelon is composed of two main layers: the peel (green) part and the meat (red) part. These two layers have different properties. The peel layer is stiff while the meat layer is softer. Moreover, imagine that you are using a fruit knife to cut the watermelon lightly from its surface and then gradually applying more force. At the beginning, the knife will be stuck on the surface when the force is low. Then, as you put more force on the knife, at a certain moment the knife suddenly can cut into the peel. This property is similar to friction. When the knife is inside the fruit, there is static friction between the knife's side face and the fruit's texture preventing the knife from moving further, and when the knife is applying the force higher than the maximal static friction the friction becomes dynamic friction which remains constant. Other than the two friction properties, a layer also needs the stiffness property to display how hard the layer is. In all, there are three main properties: maximal static friction, dynamic friction, and stiffness.

Besides the layers' properties, to make the experience even more real, when the final force applied on the tool is computed, the same force in the opposite direction will be applied to the fruit, resulting in physically real motion composed of rotation and translation.

Discussion

When implementing, lots of efforts were put on tweaking the three parameters of the fruit. Although in theory the dynamic and static friction properties make a difference, but it turned out that the difference is not easily sensable, probably because the friction force direction is parallel to the tool force direction rather than perpendicular to it.

One further idea I think interesting but didn't have enough time to implement is modeling the texture of the surface of the peel. For example, while a watermelon has very smooth surface, a pineapple's surface can have some bumps, and a kiwi fruit's surface is hairy (which sounds very hard to model). This may add more reality to the fruit model.

Summary

- Start thinking about course projects now
  - Find a partner - it will make life easier
  - Bring ideas for discussion on Thursday
- Be creative, innovative, and artistic
- Remember to make use of haptics!!!