1. Introduction

Virtual Worlds [MK94] and their concept of cyber-real space interplay invoke such interactive digital narratives that promote new patterns of understanding. However, the "narrative" and "interactive" part, which refers to a set of events happening during a certain period of time and providing aesthetic, dramaturgical and emotional elements, objects and attitudes ([NM00], [TYK01]) is still an early topic of research. Mixing such aesthetic ambiences with interactive virtual character augmentations [CMM*03] and adding dramatic tension has developed very recently these narrative patterns into an exciting new edutainment medium [LHM03]. With the interplay of a modern real-time framework for integrated interactive virtual character simulation, we can enhance the experience with full virtual character simulations.

1.1 Overview

During real-time operation and having already prepared the VR content for the virtual simulation, our interactive virtual worlds allow the user to be immersed in them and witness storytelling experiences enacted by realistic virtual humans. Thus in Section 2 of this work we review the modeling processes of building such interactive virtual worlds. In Section 3 we present such a framework which is mandatory in order to handle the exponential complexity of virtual character drama that traditional rendering-centric VR systems cannot anymore handle. In Section 4 we describe our MR new illumination model for virtual characters. Finally in section 5 we present the detailed technologies for advanced real-time virtual human simulation, such as personality, emotion, facial, body, hair and clothes simulations.

Our two case studies in the VR site of ancient Aspendos as well as on the MR (Mixed Reality) site of ancient Pompeii together with the discussion and conclusions are discussed in section 6.

2. Modeling virtual worlds for real time (RT) interactive applications

The deployment of virtual reconstruction methodologies and techniques, aimed at the achievement of photorealistic results for an interactive 3D real time experience in a VR inhabited worlds, have been applied to the virtual restitution of the historical site of Aspendos. The necessary methodological and technical solutions required to realize the visualisation of the Roman theatre of Aspendos along with the inclusion of 3D fully animated Real Time virtual humans re-enacting an ancient Roman play, have been implemented in a real-time VR-AR Framework. Furthermore, specific modelling, illumination and real time rendering strategies, along with the design choices that were operated regarding both the preparation of the textured 3D scene with the animated virtual actors, and concerning the optimizations aimed at the creation of a viable model suited for the needs of a real time interactive visualization, have been carefully considered. For the realization of the 3D model of the Aspendos site, polygonal modelling techniques were generally preferred. Critical parts of the model, such as the Cavea, due to its extremely complex geometrical features, were given special attention to keep an acceptable trade-off between visual accuracy and the performance of the real time simulation. In order to model the Aspendos site virtual environment, accurate topographic data was provided in the form of a 2D elevation map of the area featuring elevation lines every 1 meter. After the repositioning of these splines in a 3D space, it was possible to build a high polygon 3D mesh (80.000 polygons) from which a greyscale elevation map has been extracted to procedurally generate a diffuse textures using distribution parameters such as relative position of the polygons, their orientation and slope.
To further increase the visual impact of the restituted environment, the addition of trees and vegetation has been included with the implementation of billboard techniques. Such billboards, representing a total of more than 30 different types of trees, have been prepared using as starting point real images of trees and bushes that have been edited to isolate their shape from their original background, and to add an alpha channel to allow their visualization with transparency effects.

In order to create the texture tiles to be mapped on the 3D geometry of the restituted theatre, the creation of a library of the main identified diffuse material textures present on the site has been made using as a base the on-site taken high resolution digital photographs: 150 different color corrected textures that can be tiled seamlessly on the objects of the 3D scene have therefore been prepared and assigned to the model’s surfaces.

To further improve the overall final visual impact of the 3D interactive real time simulation of the Aspendos site, the addition of diffuse and hard shadows cast by the sun plays a central role. At an earlier stage of the virtual restitution a full texture baking approach of a pre-computed radiosity solution has been used. This approach however implies the creation of one texture per object: the bigger is the object, the higher the resolution of the generated texture has to be in order to avoid visual discontinuities: therefore the risk to overload the video memory, either with excessively high resolution textures or by too many generated textures, is addressed. To overcome these restrictions the adoption of a light-map and diffuse-map real time multi-texturing approach has been adopted due to the fact that the diffuse textures can be tiled, and therefore their resolution becomes independent from the size of the object upon which they are mapped. Secondly the light maps can be downsized to very low resolutions without compromising the overall visual impact of the simulation, thus eliminating the video memory overload issue. In order to visually simulate a more convincing virtual illumination of the scene, the use of High Dynamic Range Image Based Lighting has also been implemented to create such lightmaps, as shown in the following figure.

However, even though a first illuminated model of the site featuring HDR IBL has been produced during earlier tests using as source for the illumination photographic light probes, the Aspendos virtual model had to be simulated under specific lighting conditions at specific dates and hours of the day. Thus, the use of virtually generated light probes, allowing an arbitrary positioning of the direct sunlight, by means of parameters such as location, date and time, has been implemented. In order to produce suitable virtual light probes, two approaches were used: the combination of several LDR panoramic images rendered at different exposures, and the direct computation of a synthetic HDR image. Having achieved acceptable results with either method, a virtual light probe featuring the lighting conditions at the site of Aspendos, as chosen by the ERATO consortium, at the 9th of June 2004, 15.00 pm, under an unclouded sky, has been produced and used to compute a global illumination solution (result shown in figure 7, left) for the texture baking process and the creation of the lightmaps to be used for the real time simulation.

The final computed illumination solution has then successfully been used to generate all the lightmaps that subsequently have been assigned to the self-illumination channel of their corresponding surfaces in order to modulate the luminosity of the tiled material textures for the real time simulation. Thus, the previous 3D model based on a full texture baking approach of the lighting simulation has been replaced by the new model employing a lightmap based approach. Such approach allows, in fact, the dissociation of the lighting information from the material textures, and consequently allows for an overall better visual quality of the real time rendered surfaces and a reduction of the total weight of all the textures loaded into memory to perform the simulation. In order to optimize the performance of the real time application, all the lightmaps have been also

© The Eurographics Association 2006.
down-sampled to a size of 256x256 pixels and have been blurred to remove possible compression artifacts due to their small size in comparison to the spatial extension of their assigned surfaces: the final visual quality achieved, in the form of screenshots taken while running the real time interactive simulation, is illustrated in the following figure.

Figure 5: Real time simulation without lighting information (left), Previous 3D real time full texture baked model (center), screenshots from the final 3D real time simulation of the new model using light-maps (right)

3. Platform choice for inhabited virtual worlds

3.1 MR-Life system design

Our MR (Mixed Reality) system is based on the VHD++ [PPM*03], component-based framework engine which allows quick prototyping of VR-AR applications featuring integrated real-time virtual character simulation technologies, depicted in Figure 6. The framework has borrowed extensive know-how from previous platforms such as presented by [SBT*99]. The key innovation is focused in the area of component-based framework that allows the plug-and-play of different heterogeneous human simulation technologies such as: Real-time character rendering in AR (supporting real-virtual occlusions), real-time camera tracking, facial simulation and speech, body animation with skinning, 3D sound, cloth simulation and behavioral scripting of actions.

The main design principle was to maximize the flexibility while keeping excellent real-time performance. The different components may be grouped into the two following main categories:

1. System kernel components responsible for the interactive real-time simulation initialization and execution.
2. Interaction components driving external VR devices and providing various GUIs allowing for interactive scenario authoring, triggering and control.

Finally the content to be created and used by the system was specified, which may be classified into the two following main categories: a) Static and b) Dynamic content building blocks such as models of the 3D scenes, virtual humans, objects, animations, behaviors, speech, sounds, python scripts, etc.

3.2. MR Framework operation for character simulation

The software architecture is composed of multiple software components called services, as their responsibilities are clearly defined. They have to take care of rendering of 3D simulation scenes and sound, processing inputs from the external VR devices, animation of the 3D models and in particular complex animation of virtual human models including skeleton animation and respective skin and cloth deformation. They are also responsible for maintenance of the consistent simulation and interactive scenario state that can be modified with python scripts at run-time. To keep good performance, the system utilized four threads. One thread is used to manage the updates of all the services that we need to compute, such as human animation, cloth simulation or voice (sound) management. A second thread is used for the 3D renderer, who obtains information from the current scene graph about the objects that must be drawn as well as the image received from the camera. It will change the model view matrix accordingly to the value provide by the tracker. The third thread has the responsibility of capturing and tracking images. The last thread is the python interpreter, which allows us to create scripts for manipulating our application at the system level, such as generating behaviors for the human actions (key-frame animation, voice, navigation).

The MR system presented in Figure 6 features immersive real-time interactive simulation supplied with proper information in course of the simulation. That is why content components are much diversified and thus their development is extremely laborious process involving long and complex data processing pipelines, multiple recording technologies, various design tools and custom made software. The various 3D models to be included in the virtual environments like virtual human or auxiliary objects have to be created manually by 3D designers. The creation of virtual humans require to record motion captured data for realistic skeletal animations as well as a database of real gestures for facial animations. Sound environments, including voice acting, need to be recorded in advance based on the story-board. For each particular scenario, dedicated system configuration data specifying system operational parameters, parameters of the physical environment and parameters of the VR devices used have to be defined as well as scripts defining atomic behaviors of simulation elements, in particular virtual humans. These scripts can modify any data in use by the current simulation in real-time. This allows us to continue running the simulation whilst some modifications are performed.
4. Real-time illumination models for interactive virtual inhabitants

Virtual Characters have already been synthesized with real actors in common non-real-time mixed reality worlds, as illustrated successfully by a number of cinematographic storytelling examples. These examples involve virtual creatures such as ‘Gollum’ in the “Lord of the Rings™” and ‘King Kong™’ both created by Weta Digital™, the virtual character ‘Jar-Jar Binks’ in “Star Wars I: The Phantom Menace™” created by Industrial Light and Magic™, ‘Stuart’ in “Stuart Little™” created by Sony ImageWorks™ and the virtual ‘Marilyn Monroe’ as appearing in the research film “Marilyn by the lake” by MIRALab, University of Geneva. All these ‘compositing’ examples involve non-real-time (offline) pre-rendered simulations and mostly are rendered and post-processed frame by frame in an ad-hoc manner by specialized digital artists or compositors as they are termed in the industrial domain of special effects (SFX).

The active SFX sector with applications in film, television and entertainment industry has exemplified such compositing effects in a constantly growing list of projects. Our main basic research efforts are confined in adding believable and realistically dynamic globally illuminated virtual humans for both the real-time VR [PLF*01] and AR [PSO*05], [PFT05] continuum.

This objective is two-fold, as it demands a) the complete set of technologies for VH simulation [PPM*03] as well as b) an illumination model that is flexible, easy to control and efficient to compute in relation to the other VH simulation technologies [MCS*04]. The main contribution of this effort is to propose a new MR [PSO*05] physically correct illumination model based on Precomputed Radiance Transfer (PRT) illumination model, in order that is applicable to the multi-segmented, deformable and animatable hierarchies that our animation (MIRAnim) system is controlling. The main issue that we resolved was to allow for the multiple segments that the virtual skeleton consists of (due to different joints, segments, cloth, hair meshes) to respond to a high dynamic range area light that can be captured from the real scene, similar to the method described in [PFT05]. Thus starting from the Rendering Equation approximation in diffuse PRT Methods [SKS02], [KLS05]:

\[ L_0(x) = \frac{K_s}{\pi} \int L_i(l) T^{DSM} dU \]

Where

\[ T^{DSM} = V(x, l) \cdot \max(n \cdot l, 0) \]

we define a new visibility function \( V(x, l) \) which checks visibility of the current vertex position against not only the current geometric mesh that resides but against a Proxy mesh \( M' \):

\[ M' = \sum_{i=0}^{n} S_i \]

Where \( S_i \) are all the Mesh Segments of the skeleton hierarchy of the virtual character. However, not all segments are desired to be able to cast or receive self-shadowing and other GI effects. Thus the construction of the proxy mesh \( M' \) is based according to the ‘type’ of the mesh segment, tagged as Receiver or Occluder during the artist mesh preparation loop.
**Receiver** can receive but cannot cast self shadowing to other objects of the Proxy Mesh

**Occluder** can cast but cannot receive self shadowing from other objects of the Proxy Mesh.

---

**Figure 8:** Image without (a, b) and with (c, d) Receiver-Occluder set processing for our dPRT method.

This is a disjoint set, i.e. an element cannot participate in both groups. If a 'Receiver' segment is found, it is excluded from the Proxy Mesh, thus being transparent to ray casting hits. If an 'Occluder' segment is found, then this segment is excluded from further dPRT processing and calculation proceeds to next segment. However, this segment is part of the Proxy Mesh and candidate for ray-casting hits by other segments. Figure 8 illustrates the VH result of DSM with and without the Receiver-Occluder specifications. The main artifacts appear between the Skull and Hair segments which is a typical case of VH arrangement, as separate meshes due to separate creation processes and materials involved. However, with the application of our Receiver-Occluder set of elements, this artifact is avoided, as shown in Figure 8.

We employ a Voxel Ray Tracer and Uniform Grid acceleration structure for offline ray-tracing for PRT self-shadowing tests. This uniform grid is being initialized in a first pass with this Proxy Mesh while tested in a second pass, similarly as before with rays originating from every vertex of every segment in the skeleton hierarchy. The main difference is that in the simple Diffuse Shadowed Transfer from Sloan et al. [SKS02], the ray tracer was initialized with containing only the parent mesh of the current vertex x for which rays arriving from directions l where tested.

Figure 9: **Tensor Distance Ray-Hit Relaxation Criteria for DSM Transfer and dPRT. Images shown with (a) and without (b) tensor distance threshold applied.**

It is usually manifested as wrong dark coloring in the vertices in the boundary edges of different skeleton segments, as these segments are modeled adjacent in the skin bind pose, but as they are separate segments, still maintain a distance between them, not visible to the eye but large enough for ray casting. Thus one relaxation criteria that we have been applying with positive results is the checking for the tensor distance between the ray origin and the ray-triangle intersection point, against a value e corresponding to the minimum distance between the different mesh segments.

E.g. for the H-Anim [H06] structure this corresponds to 1.71*10^{-4} m. Hence all false hit-point candidates returned by the ray tracer below this e value correspond to hits between the segment boundaries and thus ignored. Another issue for the shadow feeler visibility rays is that polygons adjacent to the current vertex ray origin are going to be coplanar to many rays originating from that vertex. Thus any ray tested against a polygon that included the ray origin will at best return a hit at that vertex origin. This also leads to incorrect shadowing so it is important that we exclude the polygons that contain our ray origin, from the M’ Proxy Mesh. Thus in the construction of M0 we create full Face, Edge, Vertex information adjacency in order to account for this special case. This second criteria has been already known to the ray-tracing community and similar approaches have been discussed. Since the final shading information is stored per vertex in the H-Anim virtual human hierarchy, it is independent of the underlying animation approach. Therefore our virtual human animation system is easily integrated within our MR framework to animate the skeleton of the virtual humans with shading according to the blending schedule that is defined in Section 5. The tutorial slides provide further analytic representation of the discussed algorithms and extensions to realize the above illumination model for virtual humans in interactive virtual and augmented worlds.
5. Simulating realistic virtual humans for real-time applications

5.1 Virtual Human Animation

There exist many techniques for animating virtual characters. Two very commonly used techniques are:

Key-framing: an animation is constructed from a set of key-frames (manually designed by an animator or generated automatically) by using interpolation techniques. Although this method results in very flexible animations, the realism of the animations is low, unless a lot of time is invested.

Pre-recorded animations: an animation is recorded using a motion capture/tracking system such as Vicon or MotionStar. The animation realism is high, but the resulting animation is usually not very flexible.

A method like Principal Component Analysis (PCA) can determine dependencies between variables in a data set. The result of PCA is a matrix (constructed of a set of eigenvectors) that converts a set of partially dependent variables into another set of variables that have a maximum independency. The PC variables are ordered corresponding to their occurrence in the dataset. Low PC indices indicate a high occurrence in the dataset; higher PC indices indicate a lower occurrence in the dataset. As such, PCA is also used to reduce the dimension of a set of variables, by removing the higher PC indices from the variable set. We will use the results of the PCA later on for synthesizing the dependent joint motions. For our analysis, we perform the PCA on a subset of H-Anim joints. In order to do that, we need to convert each frame of the animation sequences in the data set into an N-dimensional vector. For representing rotations, we use the exponential map representation [EMM04, Gra98]. In this representation, a rotation can be represented by a 3-dimensional vector $r$, as a rotation with angle $|r|$ around axis $r$.

The exponential map representation of a rotation is very useful for motion interpolation [PR97], because it allows to perform linear operations on rotations. In our case the linearity of the exponential map representation is crucial since the PCA only works in the linear domain. Any rotation matrix can be written in the exponential map representation, and any exponential map representation (modulo $2\pi$) is a rotation. Grassia [Gra98] provides an extensive overview of the advantages and disadvantages of various representations of rotations, including the exponential map.

Using the exponential map representation for a joint rotation, a posture consisting of $m$ joint rotations and a global root translation. In our case, one posture/key-frame is represented by 25 joint rotations and one root joint translation, resulting in a vector of dimension 78. We have applied a PCA on a large set of motion captured postures, resulting in a PC space of equal dimension.

Figure 10: Overview of the blending engine data structure

We will now present our animation engine, called MIRAnim. The main architecture of the animation engine is a multi-track approach, where several animation streams need to be blended into a final animation. The goal of our animation engine is to provide for a generic structure that allows for the implementation of different blending strategies. This is especially important, since our animations use different representations, depending on the application. Additionally, in the final system, we will need to perform blending operations on both body and face animations, which are two completely different animation formats that require different blending strategies. A large set of blending tools, for example time warping, splitting, fading, and so on are available. The advantage of using this generic approach is that once a blending tool has been defined, it can be used for any type of animation, regardless of its structure. In order to be able to use these blending tools, only an interface needs to be provided between the data structure used for blending, and the original animation structure. An overview of the blending engine is provided in Figure 10.

The basic structure used in the blending engine is the so-called BlendableObject interface. A blendable object is the representation of an animation that can be blended with other animations. The main functionality of a blendable object is that it returns a frame, given a timekey. A frame in the blending engine is called an AbstractFrame. An abstract frame consists of a number of elements, called AbstractFrameElement objects. Each of these elements is a list of floating point values. For example, in the case of body animations, an AbstractFrameElement could be a list of 4 floating points, representing a quaternion rotation, or a list of 3 floating points, representing a 3D translation. An abstract frame could then consist of a combination of abstract frame elements that are either translations or rotations. In the case of facial animation, the abstract frame element could be a list of 1 floating point, representing a FAP value in the MPEG-4 standard.

In addition to the basic data structures and tools used for blending animations, the blending engine also provides for a few extensions that allow to further parameterize the animation and blending process. For example, modifiers can be defined which act as a wrapper around blendable objects. Examples of such modifiers are time stretching, flipping, or looping of animations. Again, custom modifiers can be defined for different animation types. To give an example in the case of body animations: a modifier is available that performs a global transformation on the whole animation. Any sequence of modifiers can be used,
within each gesture tag, an animation ID is provided.

The final step in obtaining a mix of different blendable objects requires a structure that allows for activating and deactivating different animations according to some blending parameters. This structure is called a BlendingSchedule. Each blending action is associated with a timekey, which defines the time that the blending action should start.

The animation service is built around the blending engine, and it contains specific implementations for controlling both face and body animation. The service can blend several different types of motions, including real-time idle motions and key-frame animations. For each virtual human in the scene, an XML file contains the actions that are available to the human, as well as the parameters for blending these different animations. Internally, a vhdAnimationProperty is defined for each virtual human, that contains a blending schedule, and some options, such as a choice whether or not facial and/or body animation should be played or if the translation/orientation of the virtual human was defined on the global (world) coordinate system or local coordinate system. The service also includes an integrated player, that plays and blends scheduled animations in a separate thread for all the humans in the scene.

In the next section, we will discuss the interaction component that controls the animation engine.

5.2 Interaction

As a higher level control mechanism on top of the animation service, we have developed the interaction service. This service mainly handles the spoken interaction between a user and any virtual human in the scene. The core of the interaction service is a dialogue manager [EKM04], that responds to the user according to a predefined script. In order to provide for a more natural interaction, we have integrated the Microsoft Speech SDK (SAPI5.1) into our MR framework, so that Automatic Speech Recognition (ASR) as well as Text-to-Speech (TTS) is available.

From the response text generated by the dialogue manager, the facial speech animation is created automatically [EKM04] using the timing information obtained from the TTS system. Additionally, the responses generated by the dialogue manager contain XML tags, which define specific face and body gestures that should be displayed in synchrony with the speech animation. An example of such a tagged response is given as follows:

\[
\text{<begin\_gesture id="g1" anim="shake\_head"/>}
\]

Unfortunately, I have \text{<begin\_gesture id="g2" anim="raise\_eyebrows"/>}\text{no idea <end\_gesture id="g2"/> what you are talking about.}

\[
\text{<end\_gesture id="g1"/>}
\]

Within each gesture tag, an animation ID is provided.

When the gesture animation is created, these animations are loaded from a database of gestures. Next to the animation itself, the gesture database also contains information related to the scope of each animation (which part of the body it should play on) and the weights of the joints. Each animation is stretched so that it is in synchrony with the speech, and it is blended with the other running animations by the animation service. Because the same service is used for both scripted and interactive animation, it is possible to first play a script as shown in the previous section, and then dynamically switch to animation controlled by the interaction service. Since the animation playing itself is continuously handled by the animation service, the character animation is not disrupted when this switch is made (see Figure 15).

5.3 Body Animation and Motion Retargeting

The general principle for animating a virtual 3D character - whatever it looks like – is as follows. First of all the character is given an underlying skeleton with a hierarchical structure (Figure 1) which is the object that will drive the animation later on. Because of its hierarchical formulation (i.e. each limb is placed with respect to the limb onto the new dimension of the skeleton on which it is being applied, and the character will which it is attached), all the joints rotations and segments offsets must be computed, along with one unique root transformation used for placing the skeleton in the workspace. This approach can be seen as placing the character at its correct location in the 3D space and then adapt its pose for the current frame. It has many benefits such as the possibility to easily edit the skeleton pose in order to make it match a desired configuration by simply rotating the joints (because of the hierarchical formulation, rotating the shoulder joint will make the entire arm follow the shoulder), or the fact that a given animation can be applied on various skeletons without having to recompute the rotation of all the joints. The drawbacks - quite tightly linked to the benefits - are mainly that because there exist no explicit relationship between an animation and the sizes of a skeleton, a motion can be directly applied on only one skeleton and one must modify - or retarget - the motion before applying it on another skeleton. This problem can be illustrated as follow: 2 characters with different sizes and shapes (for instance one tall and on small), while walking, will not cover the same distance if they perform the exact same number of steps. But because the global location of a skeleton is recorded once and for all when the animation is created, the global location of the root joint of the animation will not match have its feet sliding on the ground, if not penetrating it or floating in the air.
Once the underlying skeleton is animated, the character is given a virtual skin: this is the skinning phase of the animation (Figure 2). A virtual skin is a 3D mesh attached to the skeleton in such a way that it follows the skeleton animation. The attachment process consists of defining a relationship between each vertex of the skin and a subset of bones from the skeleton so that the skin follows the motion of the skeleton in a sound and realistic manner. Even if the skinning attachment is done very carefully by skilled animators, it can also yield to various unpleasant visual artifacts which are usually corrected afterwards by hand. Basically, the same adaptation issues that were found for the skeleton will have their skin equivalent. For instance, if the skeleton animation was created with a slim morphology in mind, applying it to a fat character will make the body penetrate itself because the actual diameter of each limb is now much larger but the animation wasn’t adapted accordingly.

As stated in the previous section, motion retargeting is a tricky process which is highly required when dealing with body animations.

Various methods were proposed in order to perform this task efficiently, and among the literature a few cornerstone attempts will be outlined here. Popovic and Witkin [POP95] proposed to directly manipulate the motion curves (figure 3) in order to tune an already existing motion. They basically tweak the motion curves in the parameter space by applying interpolating functions (cardinal splines) between an existing motion and user-defined key frames.

Gleicher [GLEI98] proposed to directly deal with the motion itself rather than its constituting parameters using a spacetime optimization algorithm. His method aims at adapting an existing motion clip so that it matches a new skeleton, and constraints defined by the user (such as foot plant, or grasp…see figure 4). For this, he uses a constrained non-linear solver which enforces the requirements while preserving the original motion. Such solvers are many, and as the original motion doesn’t match the constraints, his one goes towards a points which satisfies the constraint while maintaining an objective function as low as possible. In order to distribute the modifications upon the motion over as many frames as possible, the corrections aren’t applied directly but rather through splines control points. Later on, Lee and Shin [LEE99] improved this approach by making use of a hybrid inverse IK algorithm and a hierarchy of splines in order to preserve the high frequencies features of the motion.

Tolani et Al. [TOL00] developed a hybrid numerical and analytical IK algorithm which can be applied to a 7 degree of freedom limb. Based on the observation that given a configuration of the elbow and/or wrist, it is possible to restrict the range within which the elbow joint can lie into. When all the possible ranges of possible values for this angle are found, the correct value is then taken using several possible means. For orientation constraints, 2 angle variables are still active after the joints limits simplification and again, an optimization (this time non-linear) must be performed on the two variables in order to find out the correct values for the angles. Choi and Ko [CHO00] proposed a numerical IK method based on inverse rate control.
in order to re-target a motion to a new character in real-time. The method uses inverse kinematics and inverse control rate for achieving the compliance with given target positions of end effectors. Unlike regular inverse kinematics, the system doesn’t estimate the joints configuration, but rather their velocity which is then integrated over a time step in order to get the real character configuration. Several geometric configuration can be enforced as one primary goal, and a secondary goal is introduced which is match the original motion as much as possible.

Shin et al. [SHI01] developed a hybrid system focusing on real-time performances in order to drive cartoons characters for TV shows. They implemented first a Kalman Filter in order to stabilize the motion data they acquire from the real-time capture system. Starting from this cleaned up motion they developed a hybrid system for re-targeting the motion in real time to a new character. Their system has 3 separate steps. First a global correction for the root joint is estimated using the intersection between spheres representing a range of motion. Second, a numerical optimization (conjugate gradient) is used in order to estimate the best body (without the limbs) configuration that matches the constraints and remains as close as possible from the original motion. Last, the limbs configuration is adapted using an analytic inverse kinematic method also proposed by them. They proposed an importance based approach in order to choose between re-targeting the motion so that it matches the surrounding environment or stick as much as possible to the original motion. For that, they defined an importance factor for each constraint (i.e. if an end effector must be close to a surrounding object) which is inversely proportional to the distance of the recorded end effector to the considered object.

Kovar et al. [KOV02] proposed a dedicated –rather simple- method in order to get rid of foot skating (figure 6). The method uses a set of foot plant constraints that are either extracted automatically, or specified by the user. These constraints are then satisfied by smoothly changing the location of the root joint and configuration of the legs.

The root joint adaptation is done by calculating the possible location of the root joint in order to satisfy the constraints using reach spheres. The limbs configuration adaptation is then performed using a variant of the inverse kinematics method proposed by Tolani et al. Because sometimes the target can cause the limb to jerk in some circumstances, the limb is stretched by a factor of a few percents.

6. Case studies

6.1 Simulating interactive virtual plays in ancient theatres

The ERATO INCO-MED EU Project (http://www.at.oersted.dtu.dk/~erato/) aims for the re-enacting of ancient theatrical plays in reconstructed ancient Theatres. Specifically we were involved in the reconstruction of the Hellenistic ancient Aspendos theatre in Minor Asia, Turkey and the simulation of virtual actors of that historical period, re-enacting parts of ancient plays (Figure 14). Our implemented basic scenegraph OpenGL renderer OpenScenegraph in our VHD++ framework provides both advanced particle system simulation and occlusion and view frustum culling, which have been both successfully integrated and tested in the ERATO platform. The aggressive view frustum culling allows the final scene of 300.000 polygons with 1GB of total textures size, which includes the full 3D model of the theatre, featuring radiosity processed lightmaps applied at material level to simulate global illumination trough a multi-texturing approach, and the 3D real time virtual actors illuminated through PRT(precomputed radiance transfer), to be rendered at ~20fps, according to the view frustum and part of the visible scene within the virtual camera frustum.

Figure 14: Screenshots from the final 3D interactive real time simulation featuring our adapted Precomputed Radiance Transfer illumination model for deformable characters and a diffuse map-lightmap approach for static surfaces

Regarding the reconstructed theatre, offline light-mapping rendering methods have been utilized that allowed for viewing the theatre under static area light of the Aspendos virtual Light probe. The main reason for the static lightmap solution was the fact that the reconstructed model exhibited a very high polygonal count with also a high number of required Texture memory, thus making any real-time algorithm not destined for such heavy scenes, difficult to apply. The following figure illustrates a final example with the VHs rendered via the key-fill algorithm for illumination registration in the virtual inhabited world, with the static lit via the same area light, mesh of the theatre of Aspendos.

6.2 Pompeii and the thermopolium of Vetutius Placidus trial

With the help of the Superintendence of Pompeii [ASP05], who provided us with all necessary archaeological and historical information, we have selected the ‘thermopolium’ (tavern) of Vetutius Placidus and we contacted our experiments there. The results are depicted in the fol-
lowing Figure 15 where the technologies employed for simulating and authoring our virtual humans where already described in [PSO*05]. Our latest research efforts provide real-time interaction in such inhabited MR worlds, as depicted in Figure 15. For this AR case study we have extended the system described in [PSO*05] and [PFT05] by allowing for interaction with a virtual human in MR. This aims to allow visitors of ancient Pompeii to be equipped with a mobile AR guide and experience real-time digital narratives of ancient virtual life coming to life within their natural ‘real’ Pompeian environment. The two crucial components for realizing these AR experiences are a) the feature based camera tracker and b) the MR framework for virtual life simulation and AR registration.

The work presented in this tutorial described the three new components in the previous framework: a) procedural animation, b) interaction and c) MR rendering based on PRT methods. As our original storytelling experience has been intended to revive the life in ancient Pompeii, Italy in an AR simulation manner, a real-size paper maquette of the walls of the Thermopolion of Vetutius Placidus has been recreated in the lab as depicted in Figure 15. The employed hardware platform was based on a P4 3.0 GHz mobile workstation, with a NVIDIA Geforce5600Go graphics card and Unibrain firewire web camera attached on an I-Glasses HMD. The resulting performance was approximately 15fps for 12,000 polygons, depending on the performance of the markerless feature-based camera tracker.

6.3 Conclusions

Nowadays, when laymen visit virtual worlds, generally, they cannot fully grasp the vibrant life that is integrated in these simulated worlds. This is particularly true with ruins such as of ancient sites such as Pompeii and Aspendos, where we would like to interact and understand the behaviors and social patterns of living people from ancient Roman times. With the extensions and algorithms that we propose for virtual human interaction, rendering and simulation coupled under a complete real-time framework for character simulation, we aim to provide new dramaturgical notions for such inhabited virtual worlds. Such notions could extend further research in VR and develop it as an exciting edutainment medium.

7. Acknowledgements

The authors wish to thank Alessandro Foni and Nedjma Cadi for their digital artistic contribution. The work presented has been supported by the EU IST FP6 programme, in frame of the EU IST ENACTIVE (http://www.enactivenetwork.org/) project and the EU IST FP5 programme in frame of the EU INCOMED ERATO (http://www.at.oersted.dtu.dk/~erato/) project.

References


[EKM04] A. Eggges, S. Kshirsagar, and N. Magnenat-Thalmann. Generic personality and emotion


[GLE98] M. Gleicher, Retargeting motion to new characters. SIGGRAPH 1998


[POP95] Z. Popovic, A. Witkin, Motion warping, SIGGRAPH1995


Real-time Inhabited Virtual Worlds and Interaction - haptic module

Antonio Frisoli and Massimo Bergamasco and Emanuele Ruffaldi

1PERCRO, Scuola Superiore Sant’Anna

1. Interacting in virtual worlds with haptic devices

1.1. Introduction to haptic interfaces

The rendering to the human operator of the sensation of physical interaction with a virtual environment (e.g. forces generated either by the weight or by the collision with simulated objects) or with a remote environment (e.g. in teleoperation systems) can be achieved by utilizing appropriate interfaces capable of generating adequate sensory stimuli. Such interfaces, called Haptic Interfaces (HIs), are force feedback devices that can exert a controlled force on the operator’s limb like if he would be in be touch with a real object.

Force feedback is mainly based on the stimulation of human proprioceptive and kinaesthetic sensorial system, providing information respectively on the body posture, movements and applied tensions.

The Arm and Hand Exoskeleton [Ber96] represents one of the first attempts to transfer the experience in teleoperation master design to the field of haptics. In the first half of the 1990s, the demand of simpler haptic interfaces stimulated the development of new desktop haptic interfaces. In reason of their low cost, today they are with no doubt the most prevalent kind of force feedback interfaces in use.

Generally the user can grasp a handle or a stylus, through which the interaction force generated in the Virtual Environment are exchanged with the user. The contact forces are applied to the endpoint of the stylus, which represents the contact point available for interaction with the virtual environment, and can simulate hand-held tools, such as pens, scalpel, endoscopic surgery tools.

When a mechanism is devised for being used as an Haptic Interface [HCLR94], the dynamic performance of the mechanical structure is an important aspect [Yos90].

2. Haptic Devices at PERCRO

At PERCRO there is a consolidated tradition in designing and building haptic interfaces with different and specific characteristics. In particular in the following two basic systems are described, a desktop single contact point and a two-contact points haptic device.

2.1. The Haptic Master 3DOF device

The Haptic Master is 3 DOF device based on a innovative parallel kinematics which allows only translational motion of the coupler. The coupler is a fully parallel translating platform realized by connecting a base and a moving platform (coupler) via three independent legs. Each leg is characterized by an open serial kinematics composed of 2 links and presents 5 DOF: the two universal joints at the ends and an actuated revolute joint at the elbow provide the pure translation of the coupler.

The third joint of each leg is actuated by a permanent magnet DC motor, through a steel cable transmission system, which introduces some elasticity into the actuation system and determines the dynamic properties of the device. The transmission system acts both as a speed reducer and a mean for localizing the motor far apart from the actuated joint, close to the base, so that moving masses are reduced and consequently reflected inertia at the end-effector too. Ball bearings are used to reduce friction and improve the force-feedback fidelity.

A six component force sensor, ATI Industrial Automation Nano17, is mounted on the coupler below an interface pen-like tool which can be manipulated by the user. The force sensor has a force range of 550 N along x,y axes, i.e. in a plane parallel to the base, and of 570 N along the z axis, aligned with the vertical direction, with a resolution of 0.05 N. High resolution optical encoders are mounted on motor shafts and are used for measuring the current interface position. Since the coupler orientation and position are kin-
matically decoupled, the direct kinematics algorithm of the mechanism is simple and can be computed in real-time.

The control hardware is constitute by a single board computer with PentiumIII 866MHz processor, running a RTAI Linux real-time operative system. A ServoToGo data acquisition board is used for reading force and position signals from the haptic device and commanding the motor signals.

The maximum peak force that the system can exert at the center of the workspace is 18 N, the stiffness at the end effector ranges from 8 N/mm in the worst condition up to 56 N/mm in the middle of the workspace. The operational workspace is a cylinder with diameter 200mm and height 200 mm and the lowest dynamic mode off the system is located around 55 Hz.

### 2.2. The GRAB system: a two contact points haptic device

Manipulation by means of standard commercial haptic devices [RHD99], commonly permanently connected to the finger through a thimble, presents some drawbacks for the implementation of effective multipoint systems, such as lack of colocation of thimbles due to the mechanical interference during manipulative procedures, restriction of the common workspace, calibration of the relative position of devices when several ones are used for each contact point.

The haptic interface herein presented was designed to enhance specifically manipulation capabilities [BACS06]. The following specifications were addressed in the design:

- **contact points**: to allow haptic interaction with two or more contact points;
- **workplace**: to fit within a standard office desktop with minimum encumbrance;
- **workspace**: to cover a large workspace for two hands/fingers cooperative manipulation. The optimization of device encumbrance often reduces the amount of workspace available at the end effector. Even if specific virtual panning and zooming exploration procedures can be set up to explore large environments [CK05], these constraints highly reduce the manipulation capabilities of the device;
- **force range**: to display forces typical of manipulation by hands in unstructured environments;
- **isotropy**: to achieve a isotropic behavior both under kinematic and dynamic conditions. Isotropy display of forces and inertias is more important for multipoint displays, since differences of transparency between different fingers are easily perceived;
- **other performances**: to minimize residual friction, reflected inertia and increase mechanical force bandwidth; all these factors influence the transparency of the device.

#### 2.2.1. System design

The system is composed of two identical robotic arms with equivalent RRPRRR kinematics, shown in Figure 3, that provide the force-feedback for the two fingers. The user can operate the device by inserting his fingers in two thimbles placed on the end-effectors of both the arms, so that both single hand (thumb and index) and two hands (right and left indexes of two hands) interaction are possible. A set of rubber thimbles of different sizes allow any finger size to properly fit in the device.

Each arm has 6 degrees of freedom (dofs), of which the first three ones, required to track the position of the fingertip in the space, are actuated, while the last three ones, required to track its orientation, are passive. The first three dofs are kinematically equivalent to the combination of two orthogonal and incident rotational pairs and one prismatic pair, that drives a barrel along a third incident axis. The combination...
Antonio Frisoli & Massimo Bergamasco & Emanuele Ruffaldi / Real-time Inhabited Virtual Worlds and Interaction

Figure 3: Schematic model of the device

of these three dofs allow to position the end-effector over a large workspace. The last three rotational pairs form a spherical wrist that allow the rotation of a sizable thimble around a common center. Figure 4 shows the kinematics and the geometric notation adopted for the description of the device.

The first two degrees of freedom are actuated by means of a differential transmission composed of two capstans acting on a common driven pulley. The concurrent motion of these capstans produce a barrel rotation (link 3) along the horizontal axis $z_0$. The differential (opposite) motion of the capstans produce a barrel rotation along the vertical axis $y_0$.

This kind of transmission makes the operation of device symmetric with respect to two possible motions of the barrel (tilting and yawing), achieving a high kinematic isotropy along these directions.

According to this design the maximum continuous force in the worst case condition is $4N$, while the maximum peak force in the same position is $10N$. These forces can be applied in any direction. Typical maximum continuous force value (in the center of the workspace) is $7N$ in all directions.

2.2.2. Workspace

The dimensioning procedure also considered to guarantee a minimal workspace including a box of 300x400x600mm. The reachable workspace is given by the intersection of the two manipulator workspaces, as shown in figure 5. The geometry of the optimized workspace allows to integrate the two devices placed one in front of each other. This arrangement does not affect considerably the common reachable workspace allowing therefore the user to manipulate objects over a large space.

The mechanical interference of the structures is a further limitation occurring during the integration of several devices into the same workspace. In fact manipulation of virtual objects requires the user to freely move and rotate his own hand(s). In this system the degree of interference between the two structures is reduced, and does not occur during common manipulative procedures either with one hand or two hands.

As far as the reflected mass/inertia is concerned, the use of a differential transmission systems has allowed to make the properties of the mass matrix more regular, reducing consequently the effect of Coriolis and inertia disturbances during
fast motions. Specific design solution of the actuation and the transmission are arranged to fix the transmission cables to the links, reducing the pretensioning of cables and relative friction. No reduction gear was employed in the design, achieving therefore an almost zero backlash system. The actuation was made by low-inertia brushed DC servomotors, with an iron-less construction of the rotor. The kinematic solution allowed to ground the first two motors with larger bulk and thus reducing the amount of moving masses. The third motor, that provides the translational motion of the barrel, acts also as counterbalance of the barrel weight in the worst kinematic position, i.e. when the barrel is completely extended.

2.2.3. Static and dynamic performance
The whole structure was completely made in aluminum with some parts in carbon fiber. All links were designed for maximizing mechanical stiffness. In order to reduce the moving mass of the barrel while preserving torsional stiffness, four additional rods were added along each barrel.

A particular attention was given to the mechanical stiffness of the barrel along the y and z directions. Such stiffness depends on the extension of the barrel and may become critical when it is completely extended. In order to evaluate the compliance along these directions we evaluated the two main contributions:

- a linear contribution $c_{\text{lin}}$ due to the transmission system, concentrated at the level of rotational joints 1 and 2. This compliance is constant at level of joints, while the equivalent compliance at the end-effector is proportional to the square of barrel displacement ($L$);
- a flexional contribution $c_{\text{flex}}$, due to the mechanical rigidity of the barrel axis which bends under the action of a tangential force (in y,z planes) applied at its end-effector.

The barrel was considered equivalent to a cantivelered rod and therefore its compliance may be expressed as a cubic relationship between the applied force and the barrel displacement ($L$).

According to the above consideration the final compliance at the level of the end effector, can be expressed as

$$c_y = c_{\text{lin}} + c_{\text{flex}} = a_1 L^2 + a_2 L^3$$  \hspace{1cm} (1)

This compliance was experimentally determined by fixing the end-effector of the device to a rigid support and applying a command torques to the motors corresponding to increase forces in the direction $F_y$. The displacement error was measured using the information provided by the encoders for different values of the barrel displacement. The interpolating cubic curve was found as the best least-square fit with experimental data and is shown in figure 6. Experimental measured stiffness spans in the range from 1.5 to 13.5 N/mm.

The performance of the device along the barrel axis, since it is almost aligned with the grip direction, can greatly affect the control of grip during simulation. The stiffness of the device along the barrel axis was experimentally assessed, by connecting the end-effector to an ATI nano17 force sensor, and grounding the sensor to a rigid support. The experimental stiffness estimation was found as the least square linear correlation between measured forces and displacements along the barrel axis. A value of $K_x = 23.74$ N/mm ($R^2 = 0.994$) was found.

Also the dynamic performance of the device was experimentally assessed, by measuring the force response of the device along the barrel direction (x axis) in the same conditions as above. The input torque to the motor was a chirp input command. The force response is shown in figure 7, and confirms that the device can reach a large dynamic bandwidth along the x direction, between 70 and 80 Hz.

![Figure 6: Experimental mechanical compliance along y in the worst case condition](image)

![Figure 7: Dynamic force bandwidth of the device along x direction](image)
3. Control schemes for haptic interfaces

The dynamic relation between the contact force and the contact point velocity during a one point physical interaction between two bodies is called impedance. An ideal (in the sense of transparency) haptic device has to reproduce the same mechanical impedance of the virtual object the user is touching. Admittance and impedance control are the two principal classes of controllers used in haptic interfaces [FSA+04]. In the following we will indicate the mechanical impedance of the master haptic device, of user’s hand and the desired virtual impedance respectively with $Z_M$, $Z_H$ and $Z_D$. $F_Y$ are the forces voluntarily exerted by the operator.

In the impedance control the actuator force $F_a$ is computed on the basis of the position $X$ of the end-effector through the desired impedance $Z_D$ mathematical relationship:

$$F_a = Z_D X$$  \hspace{1cm} (2)

This method is the most commonly employed in haptic interface systems [YYM95, AH99], since it does not require any additional force sensor and can be particularly effective if the impedance of the device can be considered negligible, as reported in [MS94].

The addition of a feedforward pre-computed torque or the feedback linearization of the dynamics are common techniques which can be used together with the open-loop force control, in order to compensate the dynamics of the device and improve the transparency of the device. However an exact knowledge of the dynamic model of the system is required to achieve an ideal cancellation of undesired effects.

Force sensors can be usefully employed for improving the transparency of the device, guaranteeing a sufficient system stability with a good quality of perceivable force.

$$< x | y > = \int_0^\infty (x(t),y(t))dt$$  \hspace{1cm} (4)

where $(x(t),y(t))$ is usual scalar product.

Similarly for discrete systems we can define:

$$< x | y > = \sum_i (x(i),y(i))$$  \hspace{1cm} (5)

The presence of the controller $G$ allows to cancel the dynamics of the mechanism, in order to render more transparent the force feedback.

3.1. Formal definition of passivity

Consider to have two variables that can be associated to the concept “effort” $y$ and “flow” $x$. Usually flow refers to a generalized vector of velocities and angular velocities, while effort refers to a generalized vector of forces and torques. The inner product of effort and flow is the instantaneous power input to the associated physical system, and can be formalized by introducing a scalar product between $x$ and $y$.
3.2. Definition of passivity

So a system with input x and output y is passive if

\[ < x|y >_{T} \geq \beta, \forall T \]  

(6)

where the notation means that the function is zero for \( t > T \), while is defined for \( t \leq T \).

A system is strictly passive if \( \exists \delta, \beta \)

\[ < x|y >_{T} \geq \delta ||x||^2 + \beta \]  

(7)

3.3. Equivalent formulation of passivity

Suppose that the function \( h(t) \) is associated to its transfer function \( H \) through the convolution integral,

\[ y(t) = \int_{0}^{\infty} h(t - \tau) u(\tau) d\tau = h * u \]  

(8)

and so in the frequency domain:

\[ \hat{y} = \hat{h} \hat{u} \]  

(9)

If we consider the Parseval theorem that states that, given the Fourier transform \( \hat{y}(j\omega) \) and \( \hat{h}(j\omega) \):

\[ \int_{0}^{\infty} x(t) y(t) dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{x}(j\omega) \hat{Y}(j\omega) d\omega \]  

(10)

where the asterisk * denotes the complex conjugate.

We have that the passivity of a system with transfer function \( h \), and input \( u_T \) should satisfy the following equality

\[ < u_T(t)|(h * u_T)(t) >= \int_{0}^{\infty} u_T(t)(h * u_T)(t) dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{u_T}(j\omega) \hat{h}(j\omega) \hat{u_T}(j\omega) d\omega \]  

(11)

that can be rewritten as

\[ < u_T(t)|(h * u_T)(t) >= \frac{1}{2\pi} \int_{-\infty}^{\infty} ||\hat{u_T}(j\omega)||^2 \hat{h}(j\omega) d\omega \]  

(12)

Then since \( \hat{h}(j\omega) + \hat{h}(-j\omega) = 2Re(\hat{h}(j\omega)) \), because \( \hat{h} \) is a transfer function with real coefficients, in order to be physically realizable, we can equivalently state

\[ < u_T(t)|(h * u_T)(t) >= \frac{1}{\pi} \int_{0}^{\infty} ||\hat{u_T}(j\omega)||^2 2Re(\hat{h}(j\omega)) d\omega \]  

(13)

or as it is also often found:

\[ < u_T(t)|(h * u_T)(t) >= \frac{1}{\pi} \int_{0}^{\infty} ||\hat{u_T}(j\omega)||^2 Re(\hat{h}(j\omega)) d\omega \]  

(14)

From (12) we can see how the passivity implies that we should have that:

\[ \forall \omega > 0 \ 2Re(\hat{h}(j\omega)) > 0 \]  

(15)

or equivalently the Nyquist plot of \( h \) is in the right half-plane.

For a MIMO system, considering the equivalent \( \hat{h}(j\omega) + \hat{h}(-j\omega) = 2Re(\hat{h}(j\omega)) \), we can express the passivity condition by imposing the following condition on the transfer function matrix \( H \):

\[ \forall \omega > 0 \ \hat{H}(j\omega) + \hat{H}^T(-j\omega) > 0 \]  

(16)

The Virtual Environment was classically modeled as a rigid impedance with damping factor \( b_c \) and stiffness \( k_z \):

\[
\begin{align*}
F_c &= - \left( b_c + \frac{k_z}{T} \right) V_c & \text{if } X_c \text{ is penetrated inside the wall} \\
F_c &= 0 & \text{during free motion}
\end{align*}
\]  

(17)

3.4. Classical approach to passivity

In [BC94] a classical approach for computing the limits of the stability for a class of sampled-data systems is presented.

The haptic interface behaves as a rigid body with mass \( m \) with some viscous friction \( b \), while the impedance exerted by the human operator is passive.

The operator can be replaced with an impedance \( Z_h \) [MIHB85] bounded by the maximum value of:

\[ Z_{h,\text{max}} = 300 \frac{N \text{sec}}{m} + \frac{1000 N}{s} \]  

(18)

If as in [BC94] the transfer function of the haptic interface system is reduced to a first-order system, composed of a mass \( m \) and a damper \( b \) on which both operator and actuator forces are acting (see figure 12), then the transfer function of the operator-HI system, observed at terminals of \( Z_c \) is given by:

\[ G(s) = \frac{1 - e^{-Ts}}{s^2} \frac{1}{sT + b + Z_h(s)} \]  

(19)

\[ \copyright \text{ The Eurographics Association 2006.} \]
where the Zero Holder has been classically modeled as

\[ ZOH(s) = 1 - e^{-Ts} \tag{20} \]

\[ \begin{align*}
G^*(j\omega) &= \frac{1}{T} \sum_{k=-\infty}^{\infty} G(s + jk\omega) = \\
&= \frac{T}{2} \left( e^{-j\omega T} - 1 \right) \frac{1}{1 - \cos(\omega T) m \omega + b + Z_d(j\omega)} \tag{21}
\end{align*} \]

where \( T \) is the sampling time. By imposing conditions for closed loop stability by the Small Gain Theorem to (21) for \( Z_c = k_c + \frac{b}{T^2} (1 - e^{-T}) \), it is found that the following expression should hold [CS94]:

\[ b > \frac{k_c T}{2} - b_c \cos(\omega T) \tag{22} \]

At less than half the Nyquist frequency (which may be 500 Hz or greater), the factor \( \cos(\omega T) \) is positive, and so we can assume as condition for stability as:

\[ \frac{b + b_c}{k_c} > \frac{1}{2} \tag{23} \]

### 3.5. Z-Width of the Haptic Display

Figure 13 reports experimental conditions for stability of an Haptic Interface system.

The implementation of the derivative was based on a low-pass filter, with pole at 60 Hz, and was numerically realized with backwards integration.

The upper limit on \( b \) which is shown in figure 13, refers to the amplification of noise due to the discrete-time differentiation of a quantized signal. The position sensors were high resolution encoders with 4000 pulses per round, with an angular resolution of 0.0016 rad.

† This is the case when euler method is used for the discrete-time numerical integration of dynamic equations.

![Range of admissible stiffness values at 1KHz](image)

**Figure 13:** Z-width plots for \( f_s = 1KHz \) and derivative with low-pass filtering

### 3.6. Impedance based closed-loop force control

When velocity or acceleration sensors are available, the cancellation of inertial terms can be obtained with a feedforward compensation scheme using either a pre-computed torque or inverse linearized dynamic model. In such a case the accuracy of the mathematical model can greatly condition the effectiveness of the implemented control and the computation time is affected by the complexity of the model.

The usage of force sensors, placed at the zone of connection with the operator, allows to achieve an analogous result, but requiring lower computation time. In the case of Haptic Interfaces, the closed loop control must be designed in order to guarantee a full range stability independently of user’s behavior.

If we consider a simplified model of the haptic device as given by (??), it is easy to see how a proportional action of the controller will not be sufficient to have a robustly stable feedback control law. At least an integral action is required in order to have a first order dynamics of the controller:

\[ \dot{F}_a = \left( G + \frac{K}{s} \right) \left( F_d - F_o \right) = C(s)\Delta F \tag{24} \]

where \( C(s) \) indicates the Laplace transfer function of the force controller.

Notwithstanding the accuracy that can be achieved with a PI controller, by adding in feedforward the open-loop value of desired force \( F_d \), the response of the controller is improved. In this way the closed loop control is used only to cancel modelling errors which cannot be compensated in open loop.
\[ F_a = F_d + (G + \frac{K_i}{s}) \Delta F \] (25)

### 3.7. Evaluation of transparency

In order to analyze the transparency of the device under a simple force control loop, suppose, for reducing the mass \( M \) of system (14) felt by the operator, to adopt the control law \( F_a = -GF_o \), where \( G \) is the control gain.

So from the dynamics equation of this system we find:

\[
M \ddot{x} = -F_a(1 + G) \Rightarrow \ddot{x} = -\frac{(1 + G)}{M} F_a
\] (26)

In this case by acting on the values of gain \( G \) it is possible to reduce the value of reflected inertia given up to \( \frac{M}{1 + G} \). For higher gains \( G \) it is possible to reduce the perceived mass up to zero, while for a value of \( G = -1 \) it is simulated the dynamics of an infinite mass.

When the elasticity of the system is not negligible, assumptions of (26) are no longer valid, but the model represented in Figure 14 should be adopted.

If the same control law is adopted \( F_a = -GF_o \), by introducing the notation \( \tau = \frac{M}{M_0} \), it can be shown that the controlled system admits an analogous representation in terms of physical equivalents, shown in Figure 14, where the new values of \( M_0 \) and \( k' \) are given by:

\[
\begin{align*}
M'_m &= \alpha M_m = M_m \frac{\tau - G}{1 + G} \\
k' &= \beta k = k(\frac{\tau - G}{1 + G})
\end{align*}
\] (27)

![Figure 14: Equivalent system of the two-mass model](image)

Note that now the endpoint mass \( M_i \) actuated by the environment is unchanged, and it is an “uncontrollable element”.

It is worth to notice that the total reflected mass of the system is unchanged with respect to the case of single rigid body, since:

\[
M_{tot} = M_m + M_l = \frac{M_m + M_l}{1 + G} = \frac{M}{1 + G}
\] (28)

Clearly large negative feedback are not admissible in this case, since \( G > \tau \) would result in a negative spring, mass and damper, then implying a violation of the coupled stability criterion and therefore the potential instability when coupled to a passive environment.

Moreover is easy to show that in the best case of realization of a dynamic compensation with \( G = \tau \), we can achieve a perceived mass of \( M_i \), as can be verified by direct substitution into (28).

### 4. Grasping in Virtual Environments

Grasping is one of the basic haptic modes [DK02] and has key importance in most types of interactions between humans and the world surrounding them. Grasping an object allows us to identify some of its properties (geometry, material, surface textures) [KLCR99], change its physical state (position in space, internal structure) and use it for mediated interaction with other objects. The haptic mode is highly dependent on the application [Ka90, DK02]. The possibility of interacting with more than one point of contact seems to be fundamental for the manipulation of objects in virtual environments: for instance, tasks such as determining temperature or hardness require only one point of contact, while determining shape and size of an object are usually more efficiently conducted with more than one point of contact.

Humans un-consciously use suboptimal [GW84] algorithms for the prehension of objects when performing tasks with their hands. For instance, during a peg-in-hole task, they precisely adjust the relative position and interaction force between the peg and hole. Johansson and Westling conducted a series of experiments, relating tactile information to grip force when performing a lifting task [R384]. The ability to adjust grip force appears to be independent of the surface friction characteristics, but further studies from the same authors confirmed that this is not true for the case of objects with different curvature, and propose an active role of rotational friction for the stabilization of the grip [GJ98].

The nature of contact during slip provides important tactile cues regarding features on the surface as well as the nature of movement of the object, and can explain how humans take advantage of slip sensitivity when perceiving objects.

Even if almost all existing haptic interfaces provide a user interaction based on a single contact point, an increased number of contact points, not only allows to display a more natural haptic interaction [JM03, FBW04], but also improves the quality of interaction that users can perform in the environment. Multipoint haptics [BFSB04a, BSD04] refer to those devices that can simultaneously interact with the user through more than one contact point. These systems allow the representation of force and torque feedback during the simulation of dexterous manipulation and complex maneuvering of virtual objects and can improve the interaction in several applications, e.g. assembly and disassembly in virtual prototyping [MPT99a, MW03], medical palpation during simulated physical examination of patients [KNK*05] and many other ones. The existing devices at the state-of-the-art have several limitations for the simulation of grasping...
with more that one contact point, both in terms of workspace and mechanical interference among fingers. Furthermore, the limited amount of available devices for multifinger manipulation also has slowed down the development of haptic rendering modules for multiple contact points.

Non-direct contact haptic devices [SYYM04], such as encountered haptics, who externally track the position of the human fingers, represents a promising technology for the implementation of haptic devices with multiple contact points. Some preliminary prototypes [CFM+05] have been already developed, but however, the level of technology is still far from reaching the quality for a satisfactory multipoint interaction.

From a psychophysical point of view, the interactions with virtual objects during multipoint manipulation poses several interesting research questions at the level of sensorimotor coordination of movement in humans. This issue is of great interest in the context of virtual environments, where the simulation can be used also for assessing the performance of humans in the execution of a given task [YL03].

There is also an increasing request from applications of a better integration of the physical stimuli during manipulation in virtual environments. In the CREATE [DAM+03] project, children can learn the elements and the geometry of a Greek temple by reassembling it from its ruins. In the advanced visualization system the haptic rendering generates the physical perception of the contact and the ability of manipulating objects in a natural fashion, increasing the user’s attention and control during the execution of the task and the realism of the interaction. A snapshot of the application is shown in the left side of Figure 15, while in the right side it is shown the equivalent virtual manipulation analyzed in the present work.

The purpose of this work is to assess the capabilities of haptic systems for grasping and manipulation of virtual objects. A new haptic device for manipulative tasks with two contact points is used, designed to overcome most of the existing technological constraints, such as achievement of the colocation of several contact points, increase of force and common workspace capabilities.

The present work also investigates the relationship between the psycho-physical perception of prehension and the features of the physical model used for the force control of the device. Such a model is based on several parameters, that represent several physical behaviors such as gravity, friction and dynamics. An assessment phase, designed according to experiments already conducted in real environments, highlighted a good matching between the manipulation in virtual environments and reality, providing further indications for the design and the setup of physical manipulative virtual models.

5. Overview of a classical haptic rendering architecture

The VE engine implements a specific architecture for computing the force feedback during the exploration and manipulation of the objects in the virtual environment. This process should provide a stable force feedback according to the forces arising during explorative and manipulative tasks, such as inertia, weight, contact and friction forces.

5.1. General architecture

The complexity of the interaction in manipulative environments requires a multithreaded/multirate architecture. Diagram 16 describes how components are connected and grouped in different computing threads. There are two fundamental state information exchanged by the modules:

- the Device State, representing all the information coming from the device, e.g. the position of the contact points;
the Object State, characterized by the position and orientation of each object in the virtual space and their properties.

**Figure 16: The Haptic Rendering Architecture**

Stability of the haptic response requires that the rendering process (Haptic Thread) should be updated at high speed frequency (1KHz typical). The Haptic Renderer receives the Device State as input from the Device Controller and computes the corresponding contact forces as output. The haptic rendering is performed on the basis of the collision information of the haptic contact point with the objects, computed by the Collision thread, running at a much lower rate. The Collision Detection computes the set of primitives that are in a volume around the contact point, then used by the Haptic Renderer to compute the force feedback. The Collision Detection is actually implemented using an external module based on bounding volume hierarchy and running at about 100Hz.

The simulation of the dynamics of the objects is achieved through a Dynamic Simulator, that is carried out through a separate thread running at about 200 Hz, because its computation is potentially intensive, depending on the complexity of the environment and the kinematic constraints between the objects.

In the complete architecture, two geometrical representations are associated to each object: the first one is the haptic geometry used for the collision detection with the haptic contact points and the second one is the physics geometry that can be simpler. This approach allows to achieve a precise haptic interaction, while maintaining a smooth dynamic simulation.

The contact force exerted by the user are applied to the touched object through the Dynamic Simulator, allowing the direct manipulation of the object. It is worthwhile to analyze some stability considerations regarding the manipulation of virtual objects. Whenever an object is grasped in virtual environment by the user’s fingers, it is virtually tied to the contact points through a couple of springs representing the virtual stiffness of the environment. In this case, the stability limits of manipulation are determined by the typical resonance frequency of this kind of contact, that can be computed as:

$$\frac{1}{2\pi} \sqrt{\frac{2K_s}{M_t}}$$

where $K_s$ is the object stiffness in the virtual environment and $M_t$ represent the virtual object mass. For the cases considered in the present work, masses ranging from 0.1 ÷ 0.5 Kg and stiffness ranging from 500 ÷ 2000 N/m, the typical resonance frequencies varies between 5 ÷ 45Hz, well beyond the frequency of the dynamic simulation.

5.2. The linear friction model

The force information for controlling the objects is determined through the position of the contact points. The position of each contact point is measured directly by the haptic interface ($x_h$) and the relative feedback force ($\mathbf{F}$) is computed through a constraint-based proxy method with friction, based on [ZS95b,MH04,BDS03]. The haptic rendering algorithm computes the position of an additional point $\mathbf{x}_p$, called proxy, lying on the object geometry, and on the basis of current $\mathbf{x}_h$ position of the haptic interface and contact geometry ($\mathbf{C}$) the force is generated through a direct rendering method $\mathbf{F} = \mathbf{G}(\mathbf{x}_h, \mathbf{x}_p, \mathbf{C})$, using the coupling between the proxy $\mathbf{x}_p$ and the interface position $\mathbf{x}_h$. The proxy algorithm should take into account the movement of the object, and in particular it is necessary to update the last proxy position with the movement of the object: this is easily achieved by representing the proxy in object coordinates.

The standard proxy algorithm is modified with the linear friction algorithm using the friction cone model [MH04]: the movement of the proxy towards the goal position is prevented by the friction, and perceived by the user as a tangential force. The algorithm works by building a friction cone with the top at the haptic contact point, the base centered at the god point and an aperture depending on the friction coefficient. If the last proxy position is inside the cone the proxy is not moved and the user perceives a tangential force opposite to the moving direction, otherwise the proxy is placed at the boundary of the cone. The evaluation of the position of the proxy respect to the friction cone is equivalent to decomposing the contact force in the normal and tangential components and evaluating if the tangential is greater than the normal force multiplied by the static friction coefficient. The advantage of this algorithm with respect to others are that is only position based and it does not require a velocity estimation for the evaluation of the friction force.

The static friction is extended with the dynamic friction by using a proxy algorithm with two states, slip and not slip. During the not slip state the proxy is not moved if inside the static friction cone otherwise the state is changed to slip and the proxy moved to the border of the dynamic friction.
cone; when in slip mode the state is changed to not slip if the proxy is inside the dynamic cone otherwise the state is kept and the proxy moved to the border of the dynamic friction. The figure 17 shows the two friction cones and the transition diagram.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{friction_cones.png}
\caption{The linear friction cones}
\end{figure}

The algorithm can be numerically implemented through the following steps, performed while the finger is in contact with the object and the contact is not broken. At a generic \(k\)-th time sample:

\textbf{a: Computation of goal position.} The new goal position is computed as \(x_g = x_s\), where \(x_s\) is the surface point which minimizes the distance between the HI point \(x_h\) and the contact surface. \(x_s\) is assumed as the new goal value for \(x_p\).

We assume the following definitions:

\[
\begin{align*}
    r & = ||x_g(k) - x_p(k-1)|| \\
    d & = ||x_g(k) - x_h(k)||
\end{align*}
\]  

(29)

\textbf{b: Analysis of the friction condition.} In static conditions the new position of the proxy can be expressed as:

\[
x_p(k) = x_p(k-1) \quad \text{if} \quad \frac{|F_l(k)|}{\mu_s |P(k)|} = \frac{r}{\mu_s d} < 1
\]  

(30)

where \(|P| = k_l d\) is the force directed along the contact normal and \(k_l\) is the force directed tangential to the linear stiffness, used for calculating the elastic penetration force.

Otherwise, conditions of dynamic friction should be applied and the god-object, sliding over the surface, is moved on the boundary of the dynamic friction cone:

\[
x_p(k) = x_g(k) + r' \quad \text{if} \quad \mu_l d < r
\]  

(31)

with

\[
r' = \frac{x_p(k-1) - x_g(k)}{r} - \mu_l d(k)
\]  

(32)

\textbf{c: Computation of friction force.} A new force \(F(k) = k_l(x_p(k) - x_g(k))\) is computed using the new value of \(x_p\).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{object_grasping.png}
\caption{Example of object grasping}
\end{figure}

Force \(-F(k)\) is applied to the virtual object, while a force \(F(k)\) is also applied to the user.

\textbf{d: Computation of the new position of the object.} New velocity \((v, \omega)\) and position \((x, \theta)\) are computed for the virtual object.

\textbf{e: Update of the proxy position.} The current value of \(x_p\) is corrected to take into account the displacement of the virtual object:

\[
x_p = x_p + x_c
\]  

(33)

The algorithm is repeated from step a.

5.3. The grasping model

The above architecture is easily extensible to multiple contact of points: the Device Controller invokes the haptic rendering for each contact point, and the Collision Detection component performs a collision test for each contact point. From the point of view of the dynamic simulation the two contact points are invisible and they affect the simulation by applying forces on the grasped object.

The grasping of objects in this system is based on the friction and on the dynamic simulation of the objects. The grasping force exerted by the user over the object produces a friction tangential force for each of the contact points that allows to raise and manipulate the object. This model is well integrated in the dynamic simulator because when the object comes in contact with other objects the user receives a force feedback as a change of depth caused by the movement of the object. The figure 18 shows a simple example of object grasping in this system, with the display of the two contact points and the contact forces shown as the yellow vectors.

In the design phase of the grasping there are some parameters that are correlated and should be correctly evaluated.
for allowing a precise grasp of the object: the static friction coefficient, the mass of the object and the stiffness of the object. In the evaluation part of this paper the effect of these different parameters were analyzed on a group of subjects.

Figure 19 shows the behavior of the system during the grasping of an object, with a diagram for each contact point. Three phases can be identified: the first is the lifting phase when the user starts to grasp the object, the second is the holding phase and finally there is the releasing phase when the object starts to slide. In the diagrams the green and red lines represent the position of the contact point and the object along the Y axis (aligned along the gravity vector). During the lifting and holding phases the two positions have a constant difference that depends on the grasping point on the object, then in the releasing phase they diverge because the object is falling. The black line shows the status of the friction that is zero when there is no contact, one during the non-slip state and two in the slip state: it is clear that during the lifting and holding phases the proxy is in non-slip state because it is firm between the fingers and, when the object is released, initially it changes to slip mode and then the contact is lost. Finally the blue line represents the grasping force that has an increasing and varying behavior during the lifting phase, but it is almost constant during the holding phase.

6. Evaluation of grasping load and slip force in pick and place operations

In order to assess the efficiency of the device and the rendering for pick and place operations, a specific evaluation was performed to compare the available data on human grasping of real objects [GW84, JGB73] with the virtual case. The influence of weight on static grip has been experimentally studied by [GW84], where safety margins for grasping for prevention of slipping are analyzed. The safety margin is defined as the difference between the grip force and the slip force, that is the minimum grip force required for preventing slipping.

6.1. Subjects and general procedures

Three healthy right-handed men, aged between 27 and 35 yrs, served as subjects for the study. The subjects sat on a height-adjustable chair. In this position the subject might hold with his right hand the two thimbles connected to the haptic interfaces, and respectively wear them on the thumb and right index of his hand. A wide visualization screen was placed in front of the screen and a desktop, where during the experiment the subject was invited to place its elbow. A sequence of 27 objects was presented twice to each subject, for a total of 54 runs performed by each subject. All the objects in the randomized sequence were cubes with the same geometry, with pseudorandom changes in the weight \( m \) (0.1, 0.2, 0.4 Kg), in the friction coefficient, both static \( \mu_s \) (0.4, 0.8, 1.2) and dynamic \( \mu_d \) (0.3, 0.6, 1.1), and in the stiffness \( k(0.5,1,2 \ N/mm) \). In each randomized sequence all the possible combinations of weight, friction and stiffness, without repetition, were presented to the subject. The values of \( \mu_d \) were univocally associated to \( \mu_s \). The experiments were conducted with only one grasping condition, with the object hold between index and thumb tip of the same hand. Values for friction coefficients were assumed by [KBF*97] where experimental values of linear friction are reported between index tip and different materials, equal respectively to 0.42, 0.61 and 1.67 for rayon, suede and sandpaper.

6.2. Methods

The experiment consisted of a series of test runs. At the start of the experiment, one object with the shape of a cube was visualized at the center of scene.

Each subject was asked to grasp the object by index and thumb fingers, and to get acquainted with the weight of the object, by lifting it up and letting it falling down by continuously decreasing the gripping force.

After the necessary time to get acquainted with the object, the subject was asked to hold the object stationery in the air for 10 seconds with the minimum grasp force that he considered necessary, with the elbow leaning on the plane. When the subject was holding the object in the fixed position, both grasp \( F_p \) and friction \( F_t \) forces (respectively normal and tangential to the object surface) and positions of finger tips, object and proxies were recorded for each contact point. Statistical analysis was performed in SPSS 13.0.

6.3. Results

A significant correlation was found between the values of the gripping force \( F_p \) and stiffness, weight and friction values, as shown in table 1. The value of grip force \( F_p \) was found

© The Eurographics Association 2006.
to be significantly positively correlated with mass and stiffness, while negatively with friction value. Table 1 reports the correlation coefficients obtained with a Spearmann non parametric test and significance level \( p < 0.001 \).

**Table 1**: Correlation table (** \( p < 0.001 \)).

<table>
<thead>
<tr>
<th>Correlation measure</th>
<th>Grip force ( F_g )</th>
<th>Friction ( \mu_s )</th>
<th>Mass ( m )</th>
<th>Stiffness ( k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation</td>
<td>( -0.346^{**} )</td>
<td>( 0.391^{**} )</td>
<td>( 0.333^{**} )</td>
<td></td>
</tr>
</tbody>
</table>

Figure 20 presents several bar plots comparing the grasping force \( F_g \) in different conditions according to change in friction (top-bottom) and mass (left-right). Different colors are used for clustered bars representing the effect of stiffness in each test run.

In figure 21 it is shown the change of grip force \( F_g \) vs. weight, with superimposed the error bars with confidence interval of 95\%, for 9 different conditions given by different combination of friction (bottom-top) and stiffness (left-right).

### 6.4. Discussion

From the analysis of the results, it can be seen as greater gripper forces are required for holding heavier weights and stiffer objects, while lower gripper forces are required for higher friction values. This confirms the empirical laws that have been already found in the case of manipulation of real objects with bare fingers. In [GW84] it was found that the relative safety margin, defined as the safety margin in percent of the grip force, was about constant during lifting with increase of weights, was almost constant with change of weight. The calculation of the safety margin in the case of virtual manipulation allows to make an interesting comparison. As it is shown in the logarithmical plot in Figure 22 in the case of virtual manipulation, the safety margin tends to be reduced with increasing weight of the lifted mass.

This can be explained by the larger dispersion of grip forces observed for lower mass values. In fact, due to the absence of local sensation of slip, it was more difficult to discriminate the weight of lighter objects. Moreover lighter objects required a smaller resolution in the control of force \( \Delta F \), that is limited by the position resolution of the device \( \Delta X \), according to the law \( \Delta F = k \Delta X \), where \( k \) is the simulated contact stiffness. This is confirmed by the finding that better safety margins are obtained for lower values of the stiffness, as it is evident from the plot of figure 23, where grip forces are plotted vs. slip forces. While the minimum required grip
force is represented by the diagonal line, experimental data can be clustered in three main groups according to the value of contact stiffness during the simulation.

![Figure 23: Grip vs. slip force, showing the relative safety margin during grasping of virtual objects](image)

The evaluation of the performance during grasping and weight lifting, has shown that the simulation produces outcomes that are similar to experimental findings on real objects. We are currently testing the behaviour of the system when the contact friction is also implemented. Future activities will consider including evaluation with respect to angular motions and the comparison of additional feedback stimuli (audio and tactile).

7. Point based rendering

7.1. Soft contact and rotational friction

Traditionally, haptic rendering algorithms can be divided into two main categories. Three degrees of freedom algorithms, such as the god-object and the proxy [ZS95a, RKK97] have been extremely popular due to the overwhelming majority of three degrees of freedom devices available on the haptic market and due to their simplicity and computational efficiency. Six degrees of freedom devices [WMT99, OL03] simulate a more general type of contact, the one between two generic objects, but tend to be more computationally costly and harder to use given the still limited availability of satisfactory six degrees of freedom haptic devices.

In [BFSB04b] the authors of this paper proposed a simple extension to the god-object algorithm allowing to simulate a soft-finger contact, i.e. a point-contact with rotational friction constraints. The advantage of this algorithm is that it lets users interact more realistically with a virtual object using two points of contact. In this case, in fact, users are capable of applying form closure to an object, something that they would not be able to do using two point contacts such as the ones simulated by using the god-object and/or proxy algorithms. This extra capability, however, comes at no real additional computational cost, given that collision detection is still point based.

One of the limitations of the algorithm proposed in [BFSB04b] was to consider the effect of rotational friction and linear friction as two completely uncoupled. This is not the case in reality, as proven for instance by Howe and Cutkosky in [HC96]. In order to take into account the mutual effect of linear and rotation friction on each other, in this section we present a soft-finger god-object algorithm based on the concept of limit curve, which was used in [HC96] to study robotic sliding manipulation [BFSB04a, FBR*07].

7.2. Tangential forces and torques during the control of grip

During the manipulation of objects our fingertips are generally subjected to tangential torques about the axis normal to the grasp surface in addition to linear forces tangential to the grasp surface. Experimental researches conducted by [JP98, KBF97] sensorizing human fingertips during object manipulation have shown as friction tangential forces and torques are linear dependent on the value of the exerted grip force. Our experimental observations [BFSB04b] confirmed how this relationship could be assumed linear in the range of operating forces during human grip. In particular Johansson et al. [KBF97] reported accurate values of linear and rotational friction coefficients, assuming that the following expressions hold:

$$ F_{T} = \mu \Gamma P $$

$$ M_{T} = \mu \Gamma P $$

More interestingly, from the analysis conducted on different surface materials (rayon, suede, sandpaper), he found that the ratio $\mu / \mu_l$ has a value that is almost independant of the material, as it is reported in table 2. In order to be more general, the relationship between $M_{T}$ and $P$ can be described also in terms of an analytical function $\Gamma$ according to the model used to describe the elastic properties of the fingerpad, so that expression (35) becomes $M_{T} = \Gamma(P)$.

This result can be easily explained by analyzing the

<table>
<thead>
<tr>
<th>Surface material</th>
<th>$\mu_l$</th>
<th>$\mu_r$</th>
<th>$\mu_l/\mu_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rayon</td>
<td>0.42 ± 0.07</td>
<td>3.05 ± 0.57</td>
<td>7.39 ± 0.91</td>
</tr>
<tr>
<td>Suede</td>
<td>0.61 ± 0.1</td>
<td>3.84 ± 0.74</td>
<td>6.37 ± 0.88</td>
</tr>
<tr>
<td>Sandpaper</td>
<td>1.67 ± 0.24</td>
<td>10.11 ± 1.5</td>
<td>6.20 ± 1.54</td>
</tr>
</tbody>
</table>

Table 2: Experimental values of means ±SD of $\mu_l$, $\mu_r$ and their ratio measured at the index fingertip for different materials.
physics underlying the generation of the friction. In fact the resultant linear and friction forces can be found by integrating the pressure and tangential forces distribution over the fingerpad area during the contact.

Let us assume that the fingerpad is modelled as a sphere, the contact area is assumed to be a circle of radius \( a \), and the pressure distribution is assumed to be axial-symmetric. Under the effect of contact force \( P \), a distribution of pressure \( p(r) \) is generated over the contact area, such that:

\[
P = \int_0^a p(r) 2\pi r dr
\]

Under static conditions, friction forces depend on the friction coefficient \( \mu \). In such case \( p \) produces on an infinitesimal area \( dA \) a tangential traction generating a friction moment \( M \) given by:

\[
M = \int_0^a \mu p(r) 2\pi r^2 dr
\]

If we assume that the contact is occurring under limit Hertzian conditions, it can be found that:

\[
\frac{\mu_r}{\mu_l} = \frac{M}{\mu_l P} = \frac{3\pi}{16} a(P)
\]

Under the hypothesis of uniform distribution of pressure we can find instead:

\[
\frac{\mu_r}{\mu_l} = \frac{2}{3} a
\]

So the ratio of rotational \( \mu_r \) to linear friction \( \mu_l \) is a constant parameter depending on the contact geometry (subject’s fingerpad), and can be considered almost independent of the material. Its value is proportional to the radius of the contact area of the fingerpad. A numerical algorithm for the realistic simulation of frictional properties should consider this ratio as a constant independent of the task. So in the remaining analysis, we will adopt the values of linear and rotational friction coefficients derived from table 2.

### 7.3. The extended soft-finger god-object algorithm

In the following we will present two algorithms that can be used to simulate the haptic interaction between a set of human fingertips and a virtual object. The first algorithm proposed is simpler to understand and can be easily added on top of pre-existing state of the art haptic rendering algorithms supporting point-contact interaction. The second algorithm is based on more complex mathematical foundations, making it more difficult and harder to implement on top of pre-existing algorithms, but is capable of simulating the interaction between linear and rotational friction effects of the human fingertips in a more realistic way. It is important to note that both algorithms are independent of the type of model chosen to simulate rotational friction between a human fingertip and an object. For a review of possible models see [BFSB04b].

#### 7.4. The proxy algorithm with uncoupled friction

A 4 DOF god-object can be used to simulate a soft finger contact. Three of such degrees of freedom describe the position that the point of contact would ideally have when touching a virtual object, as for the standard god-object algorithm with linear friction [BDS03]. A fourth variable is added to describe the relative angular motion between the two soft finger avatars and a virtual object. It is important to note that the two parts of the algorithm are disconnected, i.e. they do not influence each other in any way.

![Figure 24: The classical friction cone for simulation of linear friction](image)

When a soft finger avatar comes into contact with a virtual object \( x_p \) is set to the current value of the angle describing the rotation of the soft finger avatar \( \alpha_0 \). The position of the haptic interface is described by the position of the HI point \( x_h \). The following steps are then performed until contact is not broken.

At a generic \( k \)-th time sample:

**a: Computation of goal position.** The new goal position for the god-object is computed as \( x_g = x_s \), where \( x_s \) is the surface point which minimizes the distance between the HI point \( x_h \) and the contact surface. The new angular position of the user’s fingers is calculated as \( \alpha_g = \alpha_s - \alpha_0 \), where \( \alpha_s \) is the angular rotation measured by the haptic device. \( x_g \) and \( \alpha_g \) are assumed as the new goal values respectively for \( x_p \) and \( \alpha \).
We assume the following definitions:

\[
\begin{align*}
 r &= ||x'_g(k) - x_p(k-1)|| \\
 \rho &= |\alpha_g(k) - \alpha_p(k-1)| \\
 d &= ||x'_g(k) - x_h(k)||
\end{align*}
\]

(40)

b: Analysis of the friction condition. In static conditions the new position of the god-object can be expressed as:

\[
\begin{align*}
 x_p(k) &= x_p(k-1) \quad \text{if} \quad \frac{P(k)}{\mu P(k)} = \frac{L}{\mu d} < 1 \\
 \alpha_p(k) &= \alpha_p(k-1) \quad \text{if} \quad \frac{M(k)}{\Gamma(P(k))} = \frac{k_p}{d} \mu d < 1
\end{align*}
\]

(41)

where \( |P| = k_d \) is the force directed along the contact normal and \( \Gamma(P) \) depends on the model chosen for the rotational friction and \( k_l \) and \( k_r \) are the haptic servo-loop gains, equivalent to a linear and rotational stiffness, used for calculating the elastic penetration force and torque. If a linear approximation is used for the function \( \Gamma(P(k)) = \mu P(k) \), then the second condition can be rewritten as:

\[
k_p \rho < k_r \mu d
\]

(42)

Otherwise, conditions of dynamic friction should be applied and the god-object, sliding over the surface, is moved on the boundary of the dynamic friction cone:

\[
\begin{align*}
 x_p(k) &= x_p(k-1) + r' \\
 \alpha_p(k) &= \alpha_p(k-1) + \rho'
\end{align*}
\]

(43)

with

\[
\begin{align*}
 r' &= \frac{x'_p(k-1) - x'_g(k)}{\mu d(k)} \\
 \rho' &= \frac{\alpha_p(k-1) - \alpha_g(k) \Gamma(P(k))}{k_r}
\end{align*}
\]

(44)

In case of a linear approximation for the \( \Gamma \) function, the equivalent condition is reduced to:

\[
\rho'(k) = \frac{\alpha_p(k-1) - \alpha_g(k)}{\mu d(k)} k_l
\]

(45)

c: Computation of friction force and torque. A new torque \( M(k) = k_l (\alpha_p(k) - \alpha_g(k)) \) and a new force \( F(k) = k_l (x'_p(k) - x'_g(k)) \) are computed using the new value of \( \alpha_p(k) \) and \( x'_p(k) \). Torque \( -M(k) v_n \) and force \( -F(k) \) are applied to the virtual object (where \( v_n \) represents a unit vector with direction along the contact normal).

A force \( F(k) \) and a torque \( M(k) v_n \) are also applied to the user (if the device used is capable of actuating such a wrench).

d: Computation of the new position of the object. New velocity \( (\dot{v}, \omega) \) and position \( (x, \theta) \) is computed for the virtual object. Angle \( \omega_c \) representing how much the object has rotated about axis \( \omega \) is computed as

\[
\omega_c = |\omega \cdot v_n| \Delta_T
\]

(46)

where \( \Delta_T \) is the servo-loop sampling time.

e: Update of god-object position. The current value of \( \alpha_p \) and \( x_p \) are corrected to take into account the displacement of the virtual object:

\[
\begin{align*}
 x_p &= x_p + xc \\
 \alpha_p &= \alpha_p + \omega_c
\end{align*}
\]

(47)

and then repeat from step a.

7.5. Analysis of joint application of linear and rotational friction

Considering the effects of linear and rotational friction as completely uncoupled is an approximation of what happens in reality. The limit curve method introduced by [HC96] is reviewed in this paper to develop an algorithm which takes into account the interplay of these two separate effects.

Figure 25: Kinematics of the sliding of the fingerpad over the surface

Refer to Figure 25, where a human fingerpad is represented. When contact with an object arises the distribution of pressure \( p \) between fingerpad and object can be assumed to be symmetric. According to the formulation in [HC96], we will refer to a reference system with origin at the center of symmetry of the pressure distribution \( p \) and will consider that the sliding of the fingerpad over the surface can be described as the combination of a translation and rotation around the Center of Rotation (COR) \( C \), shown in Figure 25. Another assumption following [HC96] is that the friction is independent of sliding speed, so that the velocity can be represented aligned along the unit vector:

\[
\hat{v}(x, y) = \frac{[-d_y, d_x]}{||d(x, y)||}
\]

(48)

The friction force per unit of surface is so given by the following:
\[ \mathbf{q} = \mu \mathbf{p} \mathbf{v} \, d\mathbf{A} \] (49)

where we have assumed that \( \mu_h = \mu_d = \mu \).

For axisymmetric contacts, if we assume that the x-axis is intersecting the COR, it is possible to express (48) as a function of the distance \( c \) of the COR from the origin of coordinates \( O \). For this choice of coordinates, we have also that the resultant force \( F_t \) is equal to zero, so that we can limit our computations to \( M \) and \( F_r = F_t \), where \( F_t \) indicates the resultant of the tangential forces. They are given by the following:

\[
\begin{align*}
F_t &= -\int_{0}^{2\pi} \int_{0}^{R} \mu p(r) \left( \frac{r \cos(\theta) - c}{\sqrt{r^2 + c^2 - 2rc \cos(\theta)}} \right) \, dr \, d\theta \\
M &= -\int_{0}^{2\pi} \int_{0}^{R} \mu p(r) \left( \frac{r - c \cos(\theta)}{\sqrt{r^2 + c^2 - 2rc \cos(\theta)}} \right) \, dr \, d\theta
\end{align*}
\] (50)

### 7.6. Computation of the curve limit

Computation of (50) requires a previous knowledge of the pressure distribution on the fingerpad, that is dependent on the mathematical model, adopted for describing the elastic response of the finger. For a given value of applied normal pressure \( P \), a limit curve will describe the analytical relationship between \( F_t \) and \( M \).

If we adopt as coordinates a set of dimensionless coordinates \( F/F_r \) and \( M/M_r \), only one limit curve can be plotted for each of the mathematical models proposed in [BFSB04b], as shown in Figure 27, where:

\[
\begin{align*}
F_r &= \mu_l P \\
M_r &= \mu_l P
\end{align*}
\] (51)

When the point \([F_r/M_r]\) is inside the unitary circle, no sliding can happen between the two bodies. When such condition is not satisfied, both linear and rotational sliding can occur. The maximum friction moment \( M_r \) is reached in the condition of purely rotational friction, i.e. a rotation around a fixed point, where the grip is taking place.

The maximum linear friction force \( F_r \) is reached in the absence of rotational friction, and should satisfy the Coulomb friction law \( F_r = \mu_l P \).

It can be seen how the models do not present particular differences, and that the limit curve can be approximated by circles with unitary radius in dimensionless coordinates, as shown in Figure 27.

So in the set of coordinates \( F_r, M_r \), the limit curves can be approximated by a family of ellipses. Figure 27 shows a set of limit curves for different values of the applied normal force \( P \), in the case of Hertzian distribution. The limit curves for \( P = \text{const} \) (\( p \)-curves) are ellipses with semiaxes given by the maximum friction linear force \( F_r \) and momentum \( M_r \) according to the applied normal force \( P \).

When the gripping force is increased \( P \), the limit curve is shifted toward ellipses with greater semiaxes. For a given gripping force \( P \) it is possible to have different sliding motion conditions, described by a different value of \( c \). Also curves with \( c = \text{const} \) (\( c \)-curves) are displayed in Figure 27. The \( c \)-curves are lines departing from the origin that represent friction conditions, where the same sliding movement is taking place between the virtual object and the finger, but with different values of the gripping force \( P \). The two limit conditions are represented by the pure rotation of the object around a fixed point (\( y \) coordinate axis corresponding to \( c = 0 \)) and the pure sliding of the object (\( x \) coordinate axis corresponding to \( c \to \infty \)).

### 7.7. The extended friction cone

The graphical representation of Figure 27 allows to produce an equivalent interpretation of the friction cone, by extending it to the case of combined rotational and linear friction. The limit condition can be rewritten as:
\[
\left( \frac{|F_i(k)|}{\mu |P(k)|} \right)^2 + \left( \frac{|M(k)|}{\Gamma |P(k)|} \right)^2 = \left( \frac{r}{\mu l} \right)^2 + \left( \frac{k_\rho}{\mu_l k/d} \right)^2 < 1 \quad (52)
\]

Equivalent \( p \)-curves and \( c \)-curves can be directly traced in the plane \( r - \dot{p} \), proving a direct geometric interpretation, useful for the implementation of the algorithm. Condition (52) is simplified when the \( \Gamma \) function is assumed to be \( \mu \ell P \):

\[
\left( \frac{r}{\mu l} \right)^2 + \left( \frac{k_\rho}{\mu_l k/d} \right)^2 < 1 \quad (53)
\]

Figure 28: The interpretation of the mixed rotational-linear friction adaptive cone

When condition (53) is not verified, the linear \( x_p(k) \) and rotational \( \alpha_p(k) \) proxies should be moved on the corresponding \( p \)-curve, by computing suitable increments \( r' \) and \( \alpha_p' \) that satisfy equation (53). The increments \( r' \) and \( \alpha_p' \) are found as the intersection in the \((r, \rho)\) plane of the \( c \)-curve through \([r, \rho]\) with the corresponding \( p \)-curve for the level of normal force \( P \), as shown in Figure 28. This shift corresponds to consider that the characteristics of the instantaneous sliding motion are preserved (\( c = \text{const} \)), but the reaction forces are changed to satisfy the physics of the system.

7.8. Combination of linear and rotational friction: the proxy algorithm with coupled friction

Given the elements exposed in section 7.5, it is now possible to formulate a god-object algorithm that combines linear and rotational friction effects. At a generic \( k \)-th time sample, step \( b \) of previous algorithm is changed to \( b' \) as follows:

\( b' \): Analysis of the friction condition. Compute \( \varepsilon = x^2 + \)

\[
y^2 \text{ by:} \quad \begin{cases} 
  x = \frac{r}{\mu_l P} \\
  y = \frac{k_\rho}{\mu_l k/d} 
\end{cases} \quad (54)
\]

When the god-object is inside the equivalent friction cone, the position of god-object is not changed and so:

\[
\begin{cases} 
  x_p(k) = x_p(k-1) \quad \text{if } \varepsilon \leq 1 \\
  \alpha_p(k) = \alpha_p(k-1) \quad \text{if } \varepsilon \leq 1 
\end{cases} \quad (55)
\]

If \( \varepsilon > 1 \), the god-object is sliding and the point \([r, \rho]\) is moved to \([r', \rho']\) on the the boundary of the corresponding \( p \)-curve, as it is shown in Figure 28. So we have:

\[
\begin{align*}
  x_p(k) &= x_g(k) + \frac{r}{\rho} (x_p(k-1) - x_g(k)) \\
  \alpha_p(k) &= \alpha_g(k) + \frac{\rho'}{\rho} (\alpha_p(k-1) - \alpha_g(k))
\end{align*} \quad (56)
\]

8. Six DOF Haptic and volume based rendering

6DOF haptics is needed in all the applications where rotational cues are required for resolution, such as manipulation of complex assemblies or virtual prototyping.

Figure 29: Examples of applications requiring volumetric rendering

Clearly the 6DOF approach introduces some challenges. For instance in common Virtual Prototyping applications it is required to:

- avoid penetration;
- rigid bodies with no shape constrain;
- complex collision detection impacts on the required haptic rate
Quality of the interaction, that means essentially stability and contact stiffness.

The following diagram in Figure 30 provides a synthetic view of all the aspects involved in 6dof haptic rendering:

![Diagram](image_url)

**Figure 30: An overview of volumetric rendering**

The standard 3DOF proxy is simple and efficient. It is normally modelled just a single point (or a small sphere when soft contact is included), but it is missing a dynamic behavior during contact. This dynamic behavior can be simulated by adopting an event based haptics.

For 6DOF we have a complex shape proxy object. We can create a Dynamic Proxy by two main options for the interaction:

- Direct Rendering
- Virtual Coupling

![Diagram](image_url)

**Figure 31: The virtual coupling allows to simplify volumetric rendering interaction**

As shown in Figure 31, the Haptic Handle is attached to a Dynamic Object. We need to simulate the Dynamic Object, but how can we choose the correct parameters? This is related to the stability issues discussed in the previous sections of this tutorial. For example suppose to have a small mass $m$, e.g. 14g. Oscillations in the coupling can be prevented by avoiding imaginary solutions in the equivalent second order dynamic system. Given mass and viscosity, the stiffness can be easily required as the one associated to the critical damping for a second order system by the equation $b_T > 4kTm$, where the parameters are physically represented in Figure 31.

The virtual coupling in 6dof can be easily implementing, by introducing a single body dynamic simulator with explicit integration:

$$
\frac{d}{dt} \begin{bmatrix} x(t) \\ q(t) \\ P(t) \\ L(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ \frac{1}{2} \omega(t)q(t) \\ F(t) \\ \tau(t) \end{bmatrix} \tag{57}
$$

and we compute the equivalent force to be applied as:

$$
\vec{F} = k_t(\vec{x}(t) - \vec{x}_h(t)) + b_t(\vec{v}(t) - \vec{v}_h(t)) \tag{58}
$$

For a 3 DOF device the rotational component can be omitted.

The penetration depth can be used as a mean for the estimation of the interaction force. Each contact on the two objects is a t-uple $(p_0, p_1, n, \delta)$, optionally with also the velocities $(v_0, v_1)$. In some cases the contact can be disregarded if the two objects at the contact points are separating, that means that the relative velocity $v_{rel} = \vec{n} \cdot (\vec{v}_1 - \vec{v}_2) \leq 0$.

![Diagram](image_url)

**Figure 32: Reference frames for computation of interaction force with penetration depth**

Alternatives to the method of penetration depth are constituted by volume and distance estimation, as shown in Figure 33. While the latter can be useful for a soft finger model, the second one is a simplified, but working approach.

When multiple contacts are present between two colliding objects, the issue of contact clustering should be analyzed. Each contact provides a force contribution and the number of contacts can vary from one step to the following:

- Iterative separation velocity selection during collision handling
- Force Contribution Mean
- Contact Clustering

© The Eurographics Association 2006.
We can cluster the contact points:
- By a threshold distance $\delta$
- In $K$ groups using $K$-mean clustering

The cluster equivalent contact point is obtained by averaging the contained contact points' properties. Also, the collision detection cost impacts on the haptic rate. Possible solutions are represented by:
- Constant-time collision detection
- Local model representation
- Multi-resolution
- Continuous collision detection

Voxel models can be efficiently used to reduce the time required for collision detection. A Voxel representation of an object is a uniform space partitioning of arbitrary object geometry, that apply also to objects from imaging technologies. Every spatial cube can be empty or full. Additionally, we can define Surface and Proximity voxels.

The collision test between a surface and a voxel model can be conducted by testing each surface point of the probing object against the static voxels of the world. Cost is limited by the number of points in the shell [MPT99b]. The force contribution of each collision can be computed by a Tangent Plane Force Model.

Multiple contacts contributions are averaged to an arbitrary number $N$. The force is sent to the device through Virtual Coupling Penetration is avoided by enlarging the surface border around objects. Additional improvements come from time coherency.

9. Application

The planning of reconstructive surgery operations currently requires the creation and manipulation of a physical model with consequent problems in precision and cost. The adoption of computational tools for the simulation and planning of surgical intervention would reduce cost and time and would increase the precision of the resulting surgical plan. Our goal is thus to allow the interactive visualization and haptic manipulation of CT scan data of an individual during preoperative planning of reconstructive facial surgery. In this scenario, the surgeon is able to manipulate the position of bones and bone fragments to identify the ideal postoperative bone placement. Effective simulation of this process requires six-degree-of-freedom (6DOF) force feedback, representing the interaction between bone fragments. This work presents a novel solution to this problem, based on a large-array voxel representation. This work extends a previous work in haptic simulation of bone surgery [MGBS05] and builds upon existing voxel-based haptic rendering algorithms [MPT99b]. The algorithm presented in this work is based on the computation of all collision points between voxel models. [MPT99b] introduces the related voxel-point-shell algorithm, in which a single moving object is point-sampled and tested against a static voxel model containing all other objects in an environment. The present work extends this approach to allow collisions among any objects in a scene, using sphere trees. Sphere trees are rotationally invariant and have a small test cost and are thus a common technique for hierarchical bounding of volumes. However, for large-array voxel models such as those required to represent CT scans, the cost of the storage of the sphere tree (or, in general, any collision tree) is prohibitive and prevents the applicability of this technique. In this work, we have overcome this problem to provide collision computation at haptic rates.

Method One of the typical approaches for the storage of voxel models is the octree, which allows reduced memory storage and a hierarchical representation of the voxel model. In this work, we have used the octree as the basis for the construction of an implicit sphere tree for collision detection. An implicit sphere tree is a tree of spheres whose center and radius are computed during the collision test and depend only on the status of the octree at a certain level.

Computing collisions between two moving voxel models requires us to account for differing reference systems. The approach presented in [MPT99b] did not address this problem, because only one object was moving and only a limited number of points of the point shell needed to be transformed to the reference system of the static voxel model. In our implicit sphere tree, we are able to compute the collision in world coordinates by transforming the relative positions of all the sphere centers in the lookup table, which corresponds to 256 or less coordinate transformations, and the relative positions of the 8 children in the octree.

The search for collision points between two voxel mod-
els starts with the top-level bounding spheres whose centers are computed in world coordinates and then descends the tree looking for all the collisions between the child spheres corresponding to the child octrees. This may proceed down to the single-voxel level, or may terminate at a certain level depending on the required precision. The descent can be optimized by taking into account only surface voxels, which are voxels with one or more free voxels as neighbors. The following image shows slices from eight levels of a typical octree; nodes drawn in purple contain surface voxels.

During collision computation, we identify all colliding points and apply the resulting force feedback to a virtual object. We use a virtual coupling technique to reduce the noise caused by rapid variation of the number of contacts. The force feedback for each collision is computed by evaluating the penetration of one of the spheres against the surface plane of the other sphere. The surface plane is stored directly inside the voxel model in the form of the distance to the nearest surface point and the normal at that point as shown in the following diagram:

The algorithm discussed in this paper has been implemented in C++ as an extension to the open source CHAI3D library for haptic interaction and tested on a 2GHz Pentium M machine using a PHANToM device (Sensible Technologies, Woburn, MA) for the haptic feedback.

The resulting implementation allows the interactive manipulation of virtual bones at haptic rates with models of 2563 voxels. The haptic rate limit (1kHz) is reached when the number of contacts is over one thousand, a condition that in practice never arises, since this induces a force feedback magnitude greater than a typical maximum device force.

A preliminary test of the algorithm with users shows that it is possible to manipulate complex objects as in Figure 1 with realistic feedback of the collisions.

![Figure 35: An application of reconstruction from CT scans](image)

9.1. HapticWeb: a new tool for rapid prototyping of web-enabled haptic applications

Haptic Web is a powerful development environment within the XVR framework (see [http://www.vrmedia.it](http://www.vrmedia.it)) for details. This part of the tutorial it will introduce the basic functionalities of the XVR development environment, with a specific description of the Haptic Web module for interfacing with haptics. See enclosed slides hands-out for further details.

10. Interaction in Virtual Worlds with haptics - Application examples

This module will introduce some application case studies based on the results of previous conducted research projects. The first case-study will present the main research issues for the interaction with digital models of sculptures in a virtual museum (http://www.pureform.org/). This case-study reviews techniques for modeling complex real shapes, starting from digital acquisition by laser scanner of real objects and then overviewing the steps needed to achieve their digital representation in VEs [FJBLO05, JBF03]. Issues for the effective presentation to users and evaluation of level of involvement in the simulation will be also presented, on the basis of the results of the exhibitions conducted in European museums within the Pure Form project. The second case-study presents an effective methodology for replicating the behavior of primary commands of car in virtual reality by means of haptic interfaces [FCM*05]. In particular simple models for simulating the behavior of a steering wheel and a manual gearshift are reviewed.

10.1. The Pure Form system

Haptic perception represents the most immediate and intimate way of interacting with sculptures and other objects of art. In traditional museums, for reasons of security and preservation, touching sculptures is not allowed to visitors, and their experience falls short of a full appreciation of the artistic value and intrinsic beauty of the collections. Moreover, the appreciation of visual art is often denied entirely to blind and visually impaired visitors. Various examples exist of systems where haptic interfaces are used to interact with art forms. Most of such examples are based on using haptic devices as creative tools [BSC00], and less often haptic devices have been used as tools for better appreciating art pieces [MSS*00]. The Museum of Pure Form [Ber, BF03], through the conjunction of virtual reality (VR) and haptic technologies, aims to explore new paradigms of interaction with sculptural pieces of art. It offers an alternative to haptic perception of artistic forms, by recreating a virtual gallery featuring three-dimensional digitised sculptures from European and worldwide museums with which visitors can interact using their senses of touch and sight. The Museum of Pure Form has had a number of deployments in cultural heritage institutions across Europe. In September 2003 for the first time a multimedia room was opened to the public at Museo dell’Opera del Duomo (OPAE), Pisa (Italy). Formal evaluation of the project took place at University College London in November 2003, where several experiments on the sense of presence were conducted using the Haptic...
Interface system in a CAVE-like environment [CNSD93], the ReaCTor. A CAVE-like room (CAVE Automatic Virtual Environment, i.e. a room with the floor and 3-4 walls back-projected with a stereoscopic visualization of a virtual computer-generated scenario, was used for the experiments. The tests were conducted within the ReaCToR at the Department of Computer Science, University College London, where the PURE-FORM exoskeleton haptic display was installed. Visitors could navigate through a virtual museum, select a virtual ancient Egypt artifact from the Petrie Museum or work of art from OPAE and interact with it.

The evaluation campaign continued at Galician Centre for Contemporary Arts (CGAC), Santiago de Compostela (Spain), and at the Nationalmuseum (NMS) of Fine Arts, Stockholm (Sweden) during the first three months of 2004. In early 2004 the Pure Form systems were exhibited throughout the month of February at CGAC, where more than 400 users explored the potential of haptic interfaces for engagement with artworks. A simultaneous audio narration with synchronized images provided an explanation of the relevant features of each work, explaining the author’s choices and situating the work in its historical period or movement.

The exhibition of Pure Form at NMS was arranged in March and April 2004 in connection with a larger exhibition “False and Genuine” featuring genuine works of art and different kinds of copies, authorized as well as forged. The virtual works of art to be explored haptically were here considered as a special kind of copy. This section will present the results of the evaluations that were conducted at the two museum exhibition sites during 2003-2004. The aim was to evaluate the impact of such systems on the public, the added value compared to traditional displays, and the potential development as a multi-media dissemination tool for museums.

10.1.1. System description
The Museum of Pure Form system is a complete Virtual Reality system with advanced human-computer interfaces, composed of the following subsystems:

1. the haptic interface system;
2. the database of the 3D models of sculptures;
3. the stereoscopic visualization system;
4. the software module dedicated to the haptic rendering and force generation;
5. the software module dedicated to the graphic rendering.

The need of a more realistic haptic interaction in VE has encouraged the development of devices that make use of multiple points of contact, when the interaction is not tool-based. In the Pure Form system two haptic devices have been specifically devised in order to provide the user forces on two contact points on the hand and were employed during the exhibitions, one exoskeleton device and one desktop device. The anthropomorphic exoskeleton HI can be worn on

© The Eurographics Association 2006.
the operator arm and is particularly suitable for installations in full immersive Virtual Reality environments, since it is possible to avoid the optical interferences between the projector beams of lights visualizing the virtual scene and the mechanical parts of the haptic interface. It is composed of two subsystems, the Light-Exos and the Pure-Form Hand-Exoskeleton.

Seventeen sculptures belonging to different historical periods were digitized through laser scanning and post-processed to produce accurate polygonal meshes. The count of polygons of meshes was then reduced in order to reduce the complexity of the models and improve the performance of both haptic and graphic rendering algorithms.

10.1.2. Motivation of the study and objectives of the evaluation

The aim of the evaluation was to evaluate the PURE-FORM haptic displays used by museum visitors. The results from evaluations at two museums CGAC and NMS are reported, where the PURE-FORM exoskeleton haptic display was used.

The methodology adopted was assessment by self-reporting and numeric assessment through ratings given over numerical scales. The questionnaire was structured into 4 main parts:

- General section
- Overall experience of the system (including haptic one)
- System usability
- User’s profile

Typical assessment of usability was performed, according to ISO 9241 definition, as composed of:

- Effectiveness: How well the user achieves the goals they set out to achieve using the system.
- Efficiency: The resources consumed in order to achieve their goals.
- Satisfaction: How the user feels about their use of the system.

Questionnaires were used at all museums. The questions were partly common in the different contexts, but they were also partly different depending on local conditions. A statistical analysis was carried out to point out the main aspects of the evaluation. A factor analysis was conducted on data and it will be outlined during the presentation in the tutorial.

10.2. Haptic systems for simulation of primary commands of cars

Virtual prototyping and digital mock-up technologies can reduce the time and costs necessary to develop and construct new products and bridge the disciplinary and distance gaps among collaborators working on the same project. Virtual prototyping is based on the simulation in a realistic...
three-dimensional virtual environment of the functionality expected from the new product. Before the real construction of the first prototype, it enables designers to evaluate the fulfillment of ergonomics, engineering and design requirements, and the feasibility/adequacy of the product to the foreseen usage. One of the key issues in Virtual Prototyping applications is the capability provided to the operator of interacting in an efficient and sensorially sufficient way with the virtual model of the object. The sense of touch appears to be fundamental to assess the properties of the prototype of a complex system, such as a vehicle. Although CAD systems are becoming widespread, there is still some important information that a CAD program alone cannot supply. The designer cannot see from the CAD how the operator will fit into cab or assess how the controls will feel during operation. During the development of new vehicles the designer must still wait until a mock-up or a prototype is built to find out how the controls feel and whether or not they are easy to use. Vehicle simulators immersed in Virtual Environments where either the contact or the inertial forces are replicated to the operator, have been successfully employed for the evaluation of ergonomics aspects in the cockpit. Vehicle simulators, and in particular aircraft simulators for use in training pilots, are generally equipped with various control devices such as acceleration, brake and clutch pedals, gear change lever and steering wheel, to enhance the realism of the simulation. In most of such simulators, control commands are passive sensorized systems, which measure only the commands executed by the user. In such systems the force behavior is regulated by the mechanics of the control commands, designed to replicate the force response of real commands. The adoption of Haptic Interfaces (HI) within simulator systems can improve the functionalities of the simulator. The mechanical response of HI can be logically programmed and controlled. In such a way in the tests within a simulator, users can evaluate the vehicle under different conditions, with changes in kinesthetics or haptic parameters. This is particularly useful when ergonomics aspects are taken into consideration.

10.2.1. The physical model of gearshift

Differently from other primary controls, the gearshift force behavior is highly non-linear and unpredictable, since it is related to the instaneous collisions that occur in the gear-box [AFS’01]. The gearshift force response was modelled through an analytical approach based on the force contents displayed during the engagement as measured experimentally, and through preliminary assessment of results. The particular force response of a gear-shift can be described as a function of the three different main stages, occurring during the gear engagement: the synchronization, the engagement and the impact against the mechanical stop. Since the gearshift is a multi-body dynamical system, each stage is associated to the interaction of different parts in the gear-box. The modelling of the force response in three stages allows to simplify its analytical description. A typical force response of an automotive gearshift, derived from experimental results, is shown in Figure 3.

\textbf{Figure 41: Example of statistical processing for factor analysis of results}

\textbf{Figure 42: Experimental force response of a gearshift according to the lever position}

In the first level of implementation of the simulator, a haptic device with two degrees of freedom, based on a parallel mechanism, allowing the lever to pivot around two independent orthogonal axes, was specifically developed to comply with the following requirements: Workspace: the knob must be able to reach every point on a spherical surface whose extension depends on lever length. In addition lever length must be variable from 110 to 160 mm. Projecting the workspace on a plane parallel to the table the knob trajectory must be included in a rectangle. Force: this HI must exert in engaging direction a force of 100 N and in selection direction a force of 60 N. Thus the haptic interface mechanism can replicate the movement of the gearshift lever on
An analytical model of the gearshift behavior was synthesized to replicate a correct force-feedback to the operator. The different phases of the gearshift were modeled through a dynamic model with both continuous and discrete states (each discrete state was associated to a gearshift stage). The model of the engagement process takes as input the user forces exerted on the knob and provides as output the knob position and velocity. The model can be divided in two parts:

- A time varying continuous dynamics module which depends on the actual gear stage.
- A discrete state machine which determines the gear stage on the basis of the knob position, the user’s force and the previous machine state.

The discrete module receives as inputs the position of the knob and the force exerted by the user. The discrete states of this module represent the different gearshift stages, i.e. the synchronization, the engagement and the end impact. For instance if the user is pushing forward during the synchronization state, a transition is activated to reach the engagement state. Conversely if the force is suddenly reverted, a transition to a free motion state is activated. The conditions under which the transitions can occur are based on position, force and time actual values. So the duration of the synchronization stage is imposed by a time-dependent out-transition from this state. Figure 5 shows an experimental test of the implemented dynamic model during the simulation of pure engagement operations with force feedback. It can be seen how experimental results produce a good fit with simulated ones.

10.2.2. - The fully virtual level of immersivity

The fully immersive development level addressed in the Virtual project has foreseen the integration of an exoskeleton haptic device, replacing the physical primary commands developed in the first level. In this development level, one of the adopted visualisation devices is a CAVE-like visualization system. The rider wears only simple shutter glasses integrated with an Head Tracker, so that he can experience a 3D stereo vision in every direction, but having the possibility to see his own body superimposed to the virtual scenario.

The force feedback is exerted on the right arm of the rider. An exoskeleton based haptic interface is in connection with the right arm of the rider at the level of the palm, of the forearm and of the upper arm. This haptic can acquire the movement of the rider’s arm and to exert controlled forces through predefined contact surfaces with the rider’s arm. The following components were integrated at this development level:
• Active pedals: to simulate the physical interaction with the primary commands of feet;
• Two Sensorized gloves: to track the position of the two arms and of the user’s hands;
• CAVE+ Shutter Glasses + Head Tracker: the graphics simulation was updated according to the current point of view of the user detected by the head tracker;
• Haptic interface: a force-feedback arm exoskeleton, to simulate both static and dynamic interaction with the primary commands.
• HeadPhones: to reproduce the audio information, according to the dynamic status of the simulation.

From the user point of view the exoskeleton haptic interface is able to make the rider feel the resistant forces of the Steering Wheel and the Gear Shift and/or the interaction force arising during possible collision of the arm with the fittings of the car. The Pedals are still present in the system and the acoustical feedback is a 3D-positional one. The system was adjustable and reconfigurable, since the exoskeleton was mounted over linear slides, allowing its positioning relatively to the driver’s seat. An optical tracker was used to detect the absolute position of the first link of the exoskeleton.

10.2.3. Interaction with the driving simulation

The force feedback of the arm exoskeleton was basically employed at two different levels: Static feedback: when the data glove detected an open posture of the hand, the user can perceive by touch the shape of the primary commands that were approximated through the simple geometries shown in the figure below.

Dynamic feedback: when the user closed his hand while in contact with one of the primary commands, the hand was assumed to hold the command and its behavior was associated to the dynamic of the command. Figure 10 shows for instance the virtual kinematic constraint that was recreated through the exoskeleton where the user was grasping the gearshift handle. On this kinematic constraint, the dynamic response associated to the engagement process was superimposed.

Figure 46: The simulation set-up and the driving interface.

(a) Steering wheel model
(b) Gearshift lever model

Figure 47: The geometries used for simulating the steering wheel and the gearshift

References


© The Eurographics Association 2006.
Antonio Frisoli & Massimo Bergamasco & Emanuele Ruffaldi / Real-time Inhabited Virtual Worlds and Interaction


Modeling virtual worlds for RT interactive applications

Nadia Magnenat-Thalmann
Alexandre Foni
Nedjma Cadi
George Papagiannakis
MIRALab – University of Geneva

Aspendos Site (161-180 A.D.)

Sources

Geometrical Restitution of the Scene
Texture restauration

Samples from the Material Gallery assembled from the on-site and photographic documentation (color corrected)

Textured Model

Scene Restitution with Statues and Textures

Velum Restitution
Roof Restitution

Chosen Interpretations

Geometrical 3D Model of the modern theatre
Modeling of the Terrain

From Procedural to Modern environment

From Modern environment to Roman environment

Modern Environment vs Roman Environment
Inclusion of Trees with use of billboards

Lighting with HDR Light probes

Illuminated 3D Model using a virtual light probe

Real time illuminated model

- Real time illuminated model using a full texture baking approach
- Light-map and diffuse tiles Real Time multi-texturing approach
Real time model of the restituted Roman Theatre illuminated using Standard lighting techniques.

illuminated using a full texture baking approach.

illumination using a Light-map/diffuse-map multi-texturing approach.

Real time reenacting of a scripted performance.

VH Database
- 3D Rendering
- Crowd Simulation
- Sound Simulation
- Skeleton Animation
- Skin Deformation
- Cloth Simulation
- Face
- Speech
- Plants
- Python Scripting
- Scenes
- UML Loader
- XML Loader.

Real time reenacting of a scripted performance.
QUESTIONS
Platform Choice for Inhabited virtual worlds = framework + components

George Papagiannakis
Nadia Magnenat-Thalmann
MIRALab – University of Geneva

facing it...

- VR-AR Research, game industry
- revolution in real-time graphics
- atomic simulation technologies affordable to nearly everyone

demand:
- Interactive, inhabited virtual worlds
- integrate multiple features / qualities / technologies
- under a single application roof
- deliver always newer, always faster, always more robust algorithms
- deliver in shorter time

effect:
- system complexity growing exponentially

need:
- methodology and tools to curb and handle dev complexity
- continuous extensions
- massive code and design reuse
- big teams of developers / experts working in parallel

integrating...

- experts / developers work independently
- producing heterogeneous technologies e.g. networking, rendering, sound, animation, vision, behaviours, AI, HCl, VR, AR...
- sure we get it...
- but "not exactly" what we wanted...

... integration issues.

- integration of heterogeneous technologies
  - e.g. net, graphics, sound, animation, behaviours, AI, HCl, VR, AR...
  - each one developed with different toolkits
  - different semantics and abstraction levels
  - > no common vocabulary
  - best effort integration is not enough
  - system complexity growing exponentially
  - we need the tools / methodology to curb and handle complexity

- numerous experts / developers
  - need to work in parallel
  - should stay within their expertise
  - should not be preoccupied with integration that
  - should adapt and follow some common rules
  - guidelines, conventions

- new projects ... new use cases ... needs
  - > to stay motivated and efficient
  - > experts too expensive to do
  - > non-centralized way

we would like to integrate them and let them "play" under a single application hood to get a feature rich and complex simulation
Searching for solutions …

- Frameworks
  - Massive design & code reuse
  - Born for extensions
  - Curbing complexity
  - Exposing ready to use design
  - Ready classes and design patterns
- Concept in short
  "Partially completed application pieces of which are customized by the user to complete the application." (G. Rogers, 1997)

... lets try to combine both ...

- Unifying development framework
  - Unifying "core architecture" providing
  - Common class vocabulary allowing for
  - Seamless collaboration of pluggable components
  - Top level SDK for app development
  - Deep SDK to access and customize deep levels
  - Domain specific fundamental/common services
  - Comprehensive set of tools and utilities
  - Ready to use and general runtime engine
  - Providing concurrent "power supply" to active elements
  - Common conventions and guidelines
- Plug-in components (active services)
  - Encapsulate heterogeneous technologies in form of active services
  - E.g., rendering, sound, animation, behaviors, AI, VR devices, interaction paradigms, etc.
  - Used on the simulation abstraction level
  - Hide complexity of their operation and state
  - Get "power supply" from the framework’s engine

Dividing, identifying, categorising …

- Investigating use cases of VR/AR applications
  - Identifying repeating architectural patterns
  - Identifying repeating mechanisms, services and components
- Categorising
  - Time-critical vs. Non-time-critical
  - Performance heavy vs. Light
  - Control vs. Event driven
  - Synchronous vs. Asynchronous
  - Containing vs. Processing
  - Data sharing vs. Transmitting
  - Local vs. Remote
  - Compiled vs. Interpreted
  - Compile-time vs. Run-time configuration

VHD++ in short

- Object oriented application development framework
  - Optimised for real-time speed
  - Interactive audio-visual performance
  - VR/AR applications
- Offering
  - Component based development
  - Fast prototyping by composition of pluggable vhdServices
  - Plug-in architecture
  - Clear O/O extension and customisation methodology
  - Platform separating middleware character
  - vhdKernel, SDK and generic vhdRuntimeEngine
- Special focus
  - Virtual human RT simulation technologies
VHD++ overview

- investigating use cases of VR/AR applications
  - identifying repeating architectural patterns
  - identifying repeating mechanisms, services and components

- categorising
  - time-critical vs. non-time-critical
  - performance heavy vs. light
  - control vs. event driven
  - synchronous vs. asynchronous
  - containing vs. processing
  - data sharing vs. transmitting
  - local vs. remote
  - compiled vs. interpreted
  - compile-time vs. run-time configuration

... fundamental services ...

- embedded into the framework kernel in order to optimise real-time performance
- low abstraction level:
  - object referencing
  - garbage collection
  - containers
  - serialization mechanisms (for loading/saving/transmission purposes)
  - active object scheduling (threads, timers, schedulers)
  - time management
  - networking
- intermediate abstraction level:
  - concurrent data sharing
  - asynchronous event handling
  - asynchronous operation requests
  - remote method invocation
  - system runtime configuration
  - dynamic scripting

... plug-in vhdServices ...

- vhdService == component
  - encapsulate any simulation technology e.g.
    - rendering, sound, animation,
    - behaviours, AI,
    - VR devices, interaction paradigms,
    - etc...
  - provides easy to use service interface
  - hides internal state and operational complexity
  - provides entry points for expert users
  - implements set of plug-in methods
  - loaded dynamically on runtime based on XML config
  - developers draw from the pool of existing vhdServices
  - each new vhdService is added to the pool for future reuse
  - developers compose applications out of vhdServices
  - quick prototyping
  - robustness as composed of tested vhdServices
**vhdProperties**

- `vhdProperty == shareable data`
  - classes encapsulating generic data e.g.
    - configuration data
    - VRML models
    - animation data
    - sound samples
    - etc...
  - developers may freely add their own new data types
- each `vhdProperty` can have sub-properties
- `vhdProperties` form tree-like structures to express generic dependencies
  - e.g. a VH model may have sound sub-property or keyframe sub-property
  - a system root may have config sub-property
- dependencies are interpreted and used by `vhdServices` operating / sharing `vhdProperties`
- `vhdProperties` provide thread-safe data sharing mechanism
  - each `vhdProperty` may be equipped with respective `vhdPropertyDecorator` able to interpret XML and to create `vhdProperty` instance
  - in this way VHD++ provides fully extensible XML configuration syntax

**vhdRuntimeEngine 1/2**

- `vhdServiceLoader`
  - loads specified `vhdServices` on runtime (based on XML config)
  - initializes loaded `vhdServices` with params read from the XML config
  - passes loaded `vhdServices` to `vhdServiceManager`
- `vhdServiceManager`
  - manages generic state of `vhdServices` on runtime
  - each `vhdService` implements `init()`, `run()`, `freeze()`, `terminate()`, `update()` methods that are used by `vhdServiceManager`
- `vhdServiceScheduler`
  - according to the XML config puts `vhdServices` on concurrent threads
  - `vhdXMLManager`
  - user provided plug-ins used by the kernel to load `vhdProperties` that encapsulate simulation content e.g. configs, VRML models, VH models, sounds, semantic information, animation data, etc.
- `vhdPropertyManager`
  - provides thread-safe data sharing mechanism to `vhdServices` operating on `vhdProperties`

**vhdRuntimeEngine 2/2**

- `vhdTimeManager`
  - maintains centralized system, simulation and time-wrap clocks that are used for synchronization of fundamental and plug-in `vhdServices`
- `vhdEventManager`
  - receives and forwards `vhdEvents`
- `vhdScriptManager` > Python scripting language
  - allows to call the API's of `vhdServices` from the Python level
- `vhdEventBroker` / `vhdServiceBroker`
  - net brokers responsible for communication with other `vhdRuntimeEngines` forming distributed `vhdRuntimeSystem`
  - exchanging `vhdEvents` with remote `vhdRuntimeEngines`
  - brokering information about existence and state of `vhdServices`
some pluggable vhdServices

- vhdSkeletonAnimationService
  - skin deformation
  - fake clothes deformation
  - physics-based cloth animation
  - skin to cloth collisions

- vhdFaceAnimationService
  - mixing of facial expressions

- vhdCrowdService
  - controls crowds of VHs

- vhdClothAnimationService
  - physical cloth animation
  - skin to cloth collisions

- vhdSoundService
  - environmental 3D effects
  - home cinema surround sound 5.1
  - multiple sound file formats

- vhdVRNavigationService
  - VR navigation paradigms
  - shutter glasses and HMD using
  - magnetic head tracking

- vhdScenarioEditorAndPlayerService

- application specific vhdServices
  - GUIs, controls, etc.

VHD++ in numbers

- framework
  - more than ~500 classes
  - grouped into ~35 packages including ~10 kernel packages
  - already ~14 existing vhdServices (big plug-in components)

- development effort
  - ~40 contributing experts / researchers / developers
  - ~10 direct developers
  - ~3 kernel designers / lead programmers
  - design and development started in March 2001 (3 years)

- now in real-world application tests
  - used in multiple EU level and national level projects
  - becomes quickly main real-time framework solution for MIRAlab and VRlab research and development in VR/AR domain

growing spectrum of applications

vhd JUST
- immersive VR health emergency personnel training

vhd STAR
- AR training of hardware maintenance professionals

vhd ERATO
- VR reconstruction of ancient theatres and plays

vhd CAHRISMA
- VR reconstruction and preservation of cultural heritage

vhd LIFEPLUS
- AR education reconstruction and preservation of ancient sites in Pompeii (Italy)

vhd JUST
- immersive VR health emergency personnel training
Interplaying technologies:

- Network deployment
  - Simulation runtime engine networked with
  - GPU equipped simulation controller

- Next-gen 3D graphics
  - Stereoscopic rendering
  - Shutter glasses or HMD support

- Immersive 3D sound
  - Home cinema 5.1 surround sound
  - Real-time environmental effects

- VR interaction
  - Immersive 3D navigation paradigm
  - Using head magnetic tracking

- Interactive scenarios
  - Interactive authoring
  - Interactive execution

- Virtual human animation
  - Skeleton animation
  - Walking engine
  - Keyframes
  - Animation mixing
  - Skin deformation
  - Face animation mixing

- Immersive 3D sound
  - Home cinema 5.1 surround sound
  - Real-time environmental effects

- Speech recognition
  - Giving orders to the system
  - Changing point of views

- Others
  - VRML, HANIM, XML, MPEG-4, ...

LIFEPLUS proposes an innovative 3D reconstruction of ancient frescoes - paintings through the real-time revival of real fauna and flora featuring groups of virtual animated characters with artificial life dramaturgical behaviours, in an immersive AR environment.
STAR aims at mixing images of real environments, virtual objects and virtual humans to produce mixed reality simulations of an existing environment.

STAR

http://www.realviz.com/STAR
Introduction

Components of Interactive Virtual Human

Overview

• Introduction
• Background
  – Interactive Virtual Human
  – Performance Animation
• The MIRAnim Engine
• Adding emotions to animation
• Interactive Virtual Human

FACIAL ANIMATION:
• MPEG-4 standard
• 68 FAPs to animate face
• Euclidean → Linear interpolation
Introduction

BODY ANIMATION:
- H-Anim joints
- 1 global translation
- 1 rotation for each joint

Problematic:
- Non-linear rotations
- Physical constraints
- Collisions

Background

Systems for automatically animating VHs:
- The BEAT System [Cassell01]

Example animation schedule in BEAT:
<
<
<
<
<
<
<
<
<
<


Systems for automatically animating VHs:
- The MAX System [Kopp04]

Example animation schedule in MAX:
</
</
</
</
</
</
</
</
</
</


Systems for automatically animating VHs:
- The GRETA System [Hartmann02]

Example animation schedule in GRETA:
</
</
</

Background

Global limitations of these systems:
• Motions are not very realistic, because they are produced procedurally for a few joints
• No full body motions taken into account
• No emotion simulation

In order to improve realism, several motion capture techniques exist.
• Motion graphs [Kovar02]

Other methods for segmentation and control:
• Segmentation into textons using linear dynamic system [Li02]
• Controlling motion from music rhythm [Kim03]
• Synthesizing motion path from annotated database [Arikan03]

Limitations of such methods:
- Computationally expensive distance criterion
- Real-time application depends on the type of constraints
- Precise constraints required not only for target and source frames, but also for intermediate animation
- Vertices/edges have no meaning
Background

• Stone et al. [Stone04] are the first to use motion capture techniques for gestures.

Our contributions

• A simpler distance criterion for construction motion graphs
• Real-time motion synthesis with loose constraints
• Realistic gesture synthesis (automatic calculation of dependent joint motion)
• Animation engine independent of animation type (face, body, ...)
• High-level animation control (dialogue system, emotions)
• Implementation as a part of existing framework in MIRALab
In the MIRAnim animation engine, we use different representations for orientations:

- Quaternions
  - Widely used
  - SLERP interpolation
  - Small format
  - Widely used
  - SLERP interpolation
  - Small format

- Exponential Map
  - Linear interpolation
  - Dimension reduction

- Principal Components
  - Statistical analysis
  - Dimension reduction

Cassell et al. [Cassell99] identify two types of communicative body motions:

- Gestures
- Posture shifts

Creating posture shift motions (idle motions):

- Recorded motion
- We need to add small variations in posture to make the motion look more realistic [Perlin95]
- We apply a noise function [Perlin85] onto a subset of Principal Components in stead of directly on joints

• Gesture motions → automatic creation of joint motions

We apply a scaling filter on each posture, to automatically generate dependent joint motions.

Animation blending:

Different levels of animation control
Different representations for emotions: Ekman, OCC, Plutchik, Cowie

- Mapping animation segments onto intervals on the emotion disc

Interactive Virtual Human

Creating face and body animation

I don't know what that means, but you do...
Funding/Support

- FP6 HUMAINE Network of Excellence
  (http://www.emotion-research.net)

- FP6 ENACTIVE Network of Excellence
  (http://www.enactive-network.org)

- FP5 MUHCI RTN project
  (http://poseidon.csd.auth.gr/Projects/muhci/muhci_ns.html)
Real-time illumination models for interactive virtual inhabitants

George Papagiannakis
Nadia Magnenat-Thalmann
MIRALab – University of Geneva

Summary
- Introduction
- Previous Work
- Our MR illumination model design
- Dynamic Precomputed Radiance Transfer
- All-Frequency 'key-fill' illumination model
- Results and Conclusions
- Future Work

Introduction
- Virtual Characters composited via offline Special Effects (SFX) in movie industry

Objectives
- Physically-principled illumination model
- Application for simulating and compositing dynamic virtual characters
- Low real-time computational cost
- Consistent, believable MR registration
- Input requirements, assumptions and level of automation
Problem Statement

- algorithms and extensions for an MR illumination model for virtual humans:
  - complex (multi-segmented, multi-material)
  - dynamic (animatable) skeleton-based deformable
  - rendered in real-time
  - consistent with the real environment area-light

Previous work: Nomenclature

- Lighting
- Shading
- Local illumination
- Global illumination (GI)
- GI for Real-Time (RT)
- Image-Based Lighting
- high dynamic range (HDR)

Previous work: Mathematical and Physical Principles

<table>
<thead>
<tr>
<th>Radiant term</th>
<th>Symbol</th>
<th>Relation</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radiant Energy</td>
<td>Q</td>
<td>basic light definition</td>
<td>J</td>
</tr>
<tr>
<td>Luminous Energy</td>
<td>E</td>
<td>W/sr</td>
<td>lm</td>
</tr>
<tr>
<td>Radiant Power (flux)</td>
<td>d Q</td>
<td>W/sr</td>
<td>lm</td>
</tr>
<tr>
<td>Luminous Power (flux)</td>
<td>d E</td>
<td>W/sr</td>
<td>lm</td>
</tr>
<tr>
<td>Radiant Intensity</td>
<td>I</td>
<td>W/(m² sr)</td>
<td>cd</td>
</tr>
<tr>
<td>Luminous Intensity</td>
<td>I</td>
<td>W/(m² sr)</td>
<td>cd</td>
</tr>
<tr>
<td>Irradiance</td>
<td>E</td>
<td>W/m²</td>
<td>lm/m²</td>
</tr>
<tr>
<td>Illuminance</td>
<td>E</td>
<td>W/m²</td>
<td>lm/m²</td>
</tr>
<tr>
<td>Radiance</td>
<td>L</td>
<td>W/(m² sr)</td>
<td>cd</td>
</tr>
<tr>
<td>Luminance</td>
<td>L</td>
<td>W/(m² sr)</td>
<td>cd</td>
</tr>
</tbody>
</table>

The reflectance principle in GI (left) and the concepts of Radiance (middle) and Irradiance (right)
Previous work: Precomputed Radiance Transfer equations

\[ L_x(\theta, \phi) = L_{So}(\theta, \phi) + \sum_{i=1}^{m} L_x(\theta, \phi)^t L_{So}(\theta, \phi) d\theta d\phi \]  (1)

\[ \bar{L}_x(\theta, \phi) = \int L_x(\theta, \phi)^t L_{So}(\theta, \phi) d\theta d\phi \]  (2)

\[ L_x = \sum_{i=1}^{m} L_x(\theta, \phi)^t L_{So}(\theta, \phi) d\theta d\phi \]  (3)

Previous work: Spherical Harmonics Basis

\[ Y_{lm}(\theta, \phi) = \frac{(-1)^m}{2^l l! \sqrt{\pi}} \cdot \sin^m(\theta) \cdot \cos^l(\phi) \]  (4)

Image Based Lighting [Debevec 98]

Algorithm
- Capture High Dynamic Range image based model using light probe
- This model replaces any synthetic light sources
- Scene is partitioned in three components
- An approximate material based model of the local scene needed
- Render all components with GI
- Final compositing using differential rendering: \( LS_{\text{Final}} = LS_{\text{back}} + (LS_{\text{obj}} - LS_{\text{noobj}}) \)

Irradiance Maps [Ramamoorthi01]

Assumptions
- Diffuse surfaces
- Distant illumination
- No shadowing, interreflection
- Hence, Irradiance is a function of surface normal

Classically, hemispherical integral for each pixel

Classically, hemispherical integral for each pixel
**Precomputed Radiance Transfer**

- Better light integration and transport
  - dynamic, area lights
  - self-shadowing
  - interreflections

- For diffuse and glossy surfaces

- At real-time rates

**Skin in the Dawn Demo**

- Ad hoc shader programs (vertex, fragment)
  - Tangent space bump mapping using pixel shaders
  - Offline Irradiance maps
  - Joint Dependant Deformation algorithm in GPU hardware
  - Fast performance, static area light, no self-shadow, interreflections, BRDF

**Self-Shadowing of Dynamic Objects**

- Interactive rendering of dynamic models with self-shadows due to time-varying, low frequency lighting environments

  - a hemispherical s/w rasterizer, which rapidly computes visibility
    - Rendering blocker geometry into a 2D occlusion mask with correct occluder fusion.
  - response of an object to the lighting = integrating the visibility function at each of the vertices against the spherical harmonic functions
  - Interactive performance (not real-time), supports only self-shadowing

**Previous work: Graphical Comparison**
Dynamic Precomputed Radiance Transfer: Prerequisites

- Our SH Rotation [Ivanic98]
- VH Skinning [Magnenat-Thalmann88]
- Our Uniform Grid Voxel-based Ray – Tracer [Amanatides87] [Moller97]

Dynamic Precomputed Radiance Transfer: Overview of our algorithm

- Proxy Mesh and Multi-Geometry categorization
  - Define new visibility function
  - Against Proxy Mesh M
- Receiver – Occluder categorization
- Tensor Distance Relaxation

Dynamic Precomputed Radiance Transfer: Our Diffuse Shadowed Merged Transfer
Dynamic Precomputed Radiance Transfer: Diffuse Shadowed Merged Transfer

- Comparisons

All-Frequency 'Key-fill' Illumination Model: Overview

- Inspired from traditional cinematography
- 'key' principal light responsible for all major direct scene lighting
- 'fill' light for all indirect, ambient illumination
- Registering VHs with 'natural' area lights for both VR and AR

All-Frequency 'Key-fill' Illumination Model: Area Light Capturing Process

- Environment images combined to create a high dynamic range (HDR) image [Debevec97]
- Interactive Exposure matching with dPRT

All-Frequency 'Key-fill' Illumination Model: Algorithm

- Unique combination of low-frequency dPRT and high-frequency direct lighting with Depth Shadow mapping
- Similar ideas in [Phong 75], [Blinn 77], [Wang05], [Kautz05]
All-Frequency 'Key-Fill' illumination model: Overview of our algorithm

Comparisons

- Phong
- DU
- DS
- DSM
- All Frequency 'Key-Fill'

Validation

- Offline (3DSMax) and our real-time model comparisons

Results and Conclusions: Architecture

- Our MR-Rendering system is incorporated on the VHD++ [Ponder03], component-based framework engine
Results and Conclusions:

**Shader Graphs**
- flexibly and interchangeably apply a rendering effect anywhere in the multi-segmented character node hierarchy

**LIFEPLUS**
- Marker-less, feature based camera tracking system for geometrical illumination
- AR simulation in ancient Pompeii [movie]

**ARToolkit**
- Marker-based camera tracking system for geometrical illumination

**ERATO**
- VR cultural heritage simulation in ancient Aspendos [movie]
Results and Conclusions: Statistical analysis on VR/AR Measurements

<table>
<thead>
<tr>
<th>VH</th>
<th>Number of Polygons</th>
<th>Preprocessing Time (sec)</th>
<th>dPRT (fps)</th>
<th>All Frequency Illumination (fps)</th>
<th>OPRIT Phong (fps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antigone</td>
<td>17458</td>
<td>5.1</td>
<td>12</td>
<td>9.3</td>
<td>7.5</td>
</tr>
<tr>
<td>Creon</td>
<td>20430</td>
<td>6.5</td>
<td>16</td>
<td>9.3</td>
<td>8.1</td>
</tr>
<tr>
<td>Combined</td>
<td>53799</td>
<td>17.0</td>
<td>10.1</td>
<td>9.2</td>
<td>7.0</td>
</tr>
</tbody>
</table>

Contribution I

- A new DSM Transfer for multi-segmented, skeleton-based deformable VHs [Chapter 4]
- Rotation of SH light coefficients [Ivanic98] from global to local for articulated structures:
  - a series of transformations applied each frame,
  - estimate the local to world joint transformation matrix for each vertex
  - for every frame we rotate the light according to each skinned mesh joint world space matrix
- Occluder, Receiver Posture Dependent, Posture Independent mesh identification

Contribution II

- Uniform Grid acceleration structure, [Amanatides86] voxel traversal, with [Moller97] fast ray-triangle intersection
- All frequency illumination model for articulated characters including:
  - low frequency, (dPRT)
  - high frequency (Point-Light, Shadow Map) shading/shadowing [Chapter 5]
- AR Illumination Registration for fully simulated VHs via various geometrical registration methods [Chapter 6]

Limitation and Future work I

- challenge for a truly dynamic self-shadowing occlusion
- Challenge for the kind of lighting (indoor-outdoor)
- other basis functions (frequencies of evaluation and projection) more fitted for the VH topology
  - e.g. Bessel basis functions
Where do we need to reduce complexity for VH rendering:
- In the Lighting environment
- In the Transfer vector
- In the basis function scheme (storage, access, compression)

employ our illumination registration with natural light algorithms to 3D scanned virtual humans

Thank You!
Introduction

Components of Interactive Virtual Human

Overview

• Introduction
• Background
  – Interactive Virtual Human
  – Performance Animation
• The MIRAnim Engine
• Adding emotions to animation
• Interactive Virtual Human

Emotion, Interaction and Real-Time Animation
Nadia Magnenat-Thalmann and Arjan Egges
MIRALAB- UNIVERSITY OF GENEVA

FACIAL ANIMATION:
• MPEG-4 standard
• 68 FAPs to animate face
• Euclidean → Linear interpolation
BODY ANIMATION:
- H-Anim joints
- 1 global translation
- 1 rotation for each joint

Problematic:
- Non-linear rotations
- Physical constraints
- Collisions

Introduction

Background

Systems for automatically animating VHs:
- The BEAT System [Cassell01]

Example animation schedule in BEAT:

```xml
code
<utterance>
<specification>
<behaviorspec id=”gesture_1”>
<gesture>
<affiliate onset=”t1” end=”t2” focus=”this”>
<function name="refer_to_loc">
<param name="refloc" value=”$Loc-Bar_1”>
</function>
</affiliate>
</gesture>
</behaviorspec>
</specification>
</utterance>
```


Systems for automatically animating VHs:
- The MAX System [Kopp04]

Example animation schedule in MAX:

```xml
code
<gestureclass adjectival
<gestureinstance small
<DURATION 1.5
<STARTFRAME 0.0
<FRAMETYPE stroke_start
<ARM XC YUpperP ZNear
<HAND symbol_1_open
<FINGER index bend_curved
<WRIST FBUp PalmInwards
<ENDFRAME
<FRAMETYPE hold
<ARM XC YUpperP ZMiddle
<HAND symbol_1_open
<FINGER index bend_curved
<WRIST FBDefault PalmDefault
<ENDFRAME
<FRAMETYPE hold
<ARM XC YUpperP ZMiddle
<HAND symbol_1_open
<FINGER index bend_curved
<WRIST FBUp PalmInwards
<ENDFRAME
```


Systems for automatically animating VHs:
- The GRETA System [Hartmann02]

Example animation schedule in GRETA:

```xml
code
<GESTURECLASS adjectival
<GESTUREINSTANCE small
<DURATION 1.5
<STARTFRAME 0.0
<FRAMETYPE stroke_start
<ARM XC YUpperP ZNear
<HAND symbol_1_open
<FINGER index bend_curved
<WRIST FBUp PalmInwards
<ENDFRAME
<STARTFRAME 0.4
<FRAMETYPE stroke_end
<ARM XC YUpperP ZMiddle
<WRIST FBDefault PalmDefault
<ENDFRAME
<STARTFRAME 1.0
<FRAMETYPE hold
<ARM XC YUpperP ZMiddle
<HAND symbol_1_open
<FINGER index bend_curved
<WRIST FBUp PalmInwards
```


Global limitations of these systems:
• Motions are not very realistic, because they are produced procedurally for a few joints
• No full body motions taken into account
• No emotion simulation

In order to improve realism, several motion capture techniques exist.
• Motion graphs [Kovar02]

Limitations of such methods:
- Computationally expensive distance criterion
- Real-time application depends on the type of constraints
- Precise constraints required not only for target and source frames, but also for intermediate animation
- Vertices/edges have no meaning
Background

- Stone et al. [Stone04] are the first to use motion capture techniques for gestures.

Our contributions

- A simpler distance criterion for construction motion graphs
- Real-time motion synthesis with loose constraints
- Realistic gesture synthesis (automatic calculation of dependent joint motion)
- Animation engine independent of animation type (face, body, ...)
- High-level animation control (dialogue system, emotions)
- Implementation as a part of existing framework in MIRALab

The MIRAnim Engine
Body Animation and retargeting

Nadia Magnenat-Thalmann and Etienne Lyard
MIRALab – University of Geneva

Virtual Characters

- Several models layers for a complete character

Skin deformation

- Skeletal Subspace Deformation (SSD)[1][2]
  The skeleton animation is used for deforming the skin layer
- Example based methods [3]
  A deformation model is tuned so that it reproduces a given set of example poses
- Physics based approaches
  The skin is physically deformed according to the skeleton motion: finite elements

Skeletal Subspace Deformation

- Most widely used approach
- Simplest method
- Easy to implement
- Deformations must be designed semi-automatically using production software
- Drawbacks: artifacts appear when the deformation becomes large

References:
Skeletal Subspace Deformation

- The skin vertices are deformed using nearby joints of the skeleton
- Several approaches were proposed in order to get rid of the deformation artifacts
- Can be implemented in graphics hardware for performances gain

Skeleton Animation

- Various means of creation
  - Traditional animation [4]
  - Key-framing
  - Motion capture[5,6]
  - Physics and AI based simulations[7]
- One animation does only match one particular skeleton
- Need for motion adaptation when applying a motion clip to a skeleton

Motion Retargeting

- Process of adapting a given motion clip to the particular features of a target character
- No 2 different virtual character have the same dimensions:
  - Thus there is a need to adapt the motion to the particular character it will animate.

Motion Retargeting Strategies

- Direct parameters tweak [8]
- Spacetime optimization [9,10]
- Inverse Kinematics [11,15]
- Hybrid systems [12]
- Dedicated methods for footskate cleanup [13,14]


Motion warping

- Directly modify the parameter curves of the motion
- Easy to implement
- Not really retargeting – rather motion editing
- No guarantee regarding the quality of the results

Spacetime optimization

- Constrained optimization formulation
- Take into account the motion as a whole
- Tricky to code
- High quality results which can comply with any kind of requirement from the user

Inverse Kinematics

- 2 classes:
  - Numerical – close from spacetime
  - Analytical – uses the specificities of the human body in order to use an analytical formulation of the problem
- Numerical methods can assign a strict number of priority among the constraints
- Analytical methods are super fast
**Hybrid systems**

- Combines the best of optimization and analytical inverse kinematics
- Architecture of the code tailored to the application
- Used for computer puppetry

**Footskate cleanup methods**

- Applications targeted for removing this very unpleasant artifact
- Ad-hoc methods have proven to work well [13]
- More complex approaches can also estimate the foot planting automatically [14]
**State of the art for hair**

Prof. Nadia Magnenat-Thalmann  
MIRALab – University of Geneva  
Thalmann@miralab.unige.ch

---

**Hair Simulation Tasks - Styling**

- Hair shape is a result of complex physical interaction between hair-hair and hair-body  
- Hairstyling – a constant human passion: curlers, clips, knots, braids and up-dos  
- Hair dynamics at interactive speed is impossible. Heuristic approach is needed for hair shape modeling  
- Thus, hair shape modeling is an exclusive task in computer graphics

---

**Hair Simulation Tasks - Dynamics**

Highly anisotropic physical behavior  
Solid-liquid duality  
Light weight of hair as compared to its acceleration, stiffness, friction and air drag  
Constant collisions / frictional interactions with the body  
100,000 to 150,000 hair strands on scalp  
- Hair-hair interaction, one of the unsolved problems of Computer Graphics

---

**Hair Simulation Tasks - Rendering**

- Intricate geometry of individual hair  
- Large number of hair strands  
- Complex interaction with light and shadows  
- Multiple scattering, luster, self-shadowing  
- Anisotropy in shading  
- Small thickness of the hair – anti-aliasing  
- How artists paint hair?
Open Problems in Hair Simulation

• Hair Shape Modeling
  – Many recent attempts, however not yet matured

• Hair Dynamics
  – Stiffness dynamics of individual hair was grossly approximated considering current computing power (1GHz, 1GB RAM)
  – Hair-hair interaction not attempted (until recently)

Open Problems in Hair Simulation

• Hair Rendering
  Fairly matured and is available through commercial systems. However, global-illumination for hair with detailed shading models is not yet attempted.

  • Real-time hair animation and rendering
    With possible compromise on the realism

  • Hair interaction with a user (hairdresser for example)

Our Approach in Non Real-Time

• Hair Shape Modeling*
  – Hair shape as streamlines of fluid flow

• Hair Rendering
  – Single iteration hair rendering including volumetric shadows using graphics hardware

Making of…

Placement of smoothed particles to realize hair-fluid

Our Approach (2)

- Hair Dynamics**
  - Stiffness dynamics of non-straight hair strand
  - Elaborate inertial dynamics using reduced coordinate formulation
  - Hair-hair, hair-body and hair-air interaction dynamics


Individual Hair Dynamics

- Without stiffness
- With stiffness

Body forces
Collision as fluid boundary

Results – Computer Animations

Bunch of hair

Hair fall
Results – Computer Animations

Blown by wind

Experiments with real hair

A typical experiment on bunch of hair

Low stiffness High stiffness

As stiffness increases the “swing” reduces

Virtual Experimental Setup

- Switch length – 24cm
- Number of hair strands – 1000
- Hair segment length 0.9cm
- Amplitude – 4cm
- Frequency – 1Hz
- Number of cycles - 2

Variations in Strand Stiffness

- Increase in stiffness gives considerable phase difference.
- In reality, it is the change in hair thickness which varies the bending stiffness.
- However, it is hard to model various hair thickness.
- We may simulate the process by changing the stiffness coefficient E.
- Observe the tip movement with respect to the movement of the base.

E=250 E=500 E=750
Simulation - Adjusting the air drag

- Normal hair makes 5-6 oscillations before coming to rest.
- When hair is treated with styling products, air drag increases to reduce number of oscillations to merely 1-2.

Variations in Hair-hair Interaction

- The spread in the bunch of hair is a result of hair-hair interaction.
  - One can vary the constant Kc to change this interaction.
  - An immediate effect is on the spread of the tip.
  - One can also observe how the tip makes a "bouncy" movement as Kc increases.

\[ \rho \]
\[ cK \]
Equation of state

Variations in Internal Damping

- The internal damping is associated to energy loss as elastic material undergoes deformation.
- Though the large change in internal damping has minute effect on the hair movement, it is important to increase stability of the simulation.

\[ ID=0.1 \quad ID=0.2 \quad ID=0.3 \]

Full head of hair

- By means of varying hair-hair interaction (the coefficient Kc), one can achieve different quality of hair movements.
- When Kc is low, hair appears to be straight and oily. Increase in Kc gives much "volume" to the hair and it then appears more dry and airy.
Result – Perturbations to Few Individual Hair

- Few individual hair strands break away from the flow and follow the perturbations.

Result - Shaping Hairstyle Using Fluid Elements

- Vortices as curlers

Result – Applying Clumps and Waves

- Under adhesive forces, in reality, hair strands form clumps.

Real-Time Animation of Hairstyles

- Managing Complexity of Hair Simulation
  - More than 100,000 hair strands
  - Mechanical behavior of individual strands
  - Collisions and friction on the skull
  - Collisions and friction between strands
  - Physical state (wetness, styling products)
  - Aerodynamics

- No chances to simulate an explicit mechanical model in real-time.
Real-Time Animation of Hairstyles

- Ideas for Speeding Up Hair Simulation
  - Simplification of the mechanical model
  - Reduction of the degrees of freedom
  - Macroscopic approximations
  - Interpolation

- Techniques
  - Wisp or Cluster models
  - Particle systems of variable topology
  - Volume hair models
  - Free-Form Deformations

1. Real-Time FFD Hair Animation

- The main idea: Deforming the complete hairstyle using a mechanically-animated Free-Form Deformation lattice.

1. Real-Time FFD Hair Animation

- Main advantages:
  - Drastic and controllable reduction of the number of degrees of freedom of the mechanical model
  - Real-time mechanical simulation
  - Simple and fast interpolation scheme for computing the deformed hairstyle
  - Real-time motion of any feature of the hairstyle
  - Any hairstyle can be animated

2. FFD Hairstyle Deformation

- Building the lattice around the hairstyle
  - The lattice is attached to the rigid-body motion $\mathbf{R}$ of the skull
  - The lattice is then deformed by mechanical simulation
2. FFD Hairstyle Deformation

• Attaching Strand Roots on the Skull
  – Attachment of the hair strands on the skull should be ensured whatever the deformation
  – Strand roots should be defined by the rigid motion of the skull only
  – Continuous transition between rigid motion \((\delta = 0)\) and deformed lattice \((\delta = 1)\) along the hair

\[
p = \sum_i w_i \delta \{p_i + (1 - \delta)p_i^0\}
\]

3. Animating the FFD Lattice

• Mechanical Properties to be Simulated
  – Mechanical behavior of hair:
    • Density, Elasticity
    • Interactions between strands
  – External forces exerted on hair:
    • Gravity
    • Aerodynamic effects
  – Collisions between hairs and other objects:
    • Body parts: Skull, shoulders...
    • Other objects

3. Animating the FFD Lattice

• Building Blocks: Lattice Stiffeners
  – Mechanical interaction between lattice nodes
    • Weighted contributions of lattice nodes
  – Viscoelastic behavior law \(\sigma(\varepsilon, \varepsilon')\)
    • Relating lattice forces \(f_i\) to positions \(p_i\) and speeds \(p_i'\)

\[
f_i = w_i \sigma \quad \text{with} \quad \varepsilon = \sum_j w_j p_j \quad \text{and} \quad \varepsilon' = \sum_j w_j p_j'
\]

• Particular Lattice Stiffeners
  – Lattice Attachment
    • Relates one location of the lattice to a given point in space
    • Weights: Linear interpolation coefficients of the location in the lattice
  – Lattice Spring
    • Relates two locations of the lattice
    • Weights: Linear interpolation coefficients, positive and negative
3. Animating the FFD Lattice

- Hair Strand Discretization
  - Creating lattice springs between grid edges (>45°)
  - Creating lattice attachment on root

- Decimation of the Model
  - Number of lattice stiffeners reduced at will
    - Similar stiffeners are consolidated
    - Parallelism and projections in the space of lattice node weights
  - Optimal accuracy - speed tradeoff
    - Initial strands
    - Initial model: 12100 strands, 3500 attachments
    - Low decimation: 800 strands, 200 attachments
    - High decimation: 200 strands, 50 attachments

- The Ether Model
  - Nodes surrounded by lattices empty of any hair may exhibit erratic behaviour.
  - To limit this, the whole lattice is considered as an ether medium with its own stable behaviour.
  - Additional weak viscoelastic forces are attached to the lattice nodes to rest position (rigid motion of the head)
    - Stabilize and damp the motion of the hairs
    - Improve the robustness
    - Additional custom parameters for simulating hair stiffness (short hair, styling gel...)
    - May include additional physical effects (aerodynamics)
3. Animating the FFD Lattice

- **Collision Effects**
  - Between the hair and the body
    - Head
    - Shoulders
  - **Metaball-based model of body parts**
    - Approximate modeling with a low number of primitives
    - Easy animation of body deformations
    - Attached to the rigid head motion (skull) or the body skeleton (shoulders)

- **Collisions with Metaballs**
  - 6th-order polynomial metaballs ($n=3$) modeled as lattice attachments
    - **Potential:**
      \[
      F_{\text{potential}} = \begin{cases} 
      e^0 r_0^{2n} & \text{if } s \geq 0 \\
      0 & \text{elsewhere}
      \end{cases}
      \]
    - **Force:**
      \[
      F_{\text{force}} = \frac{e^0 r_0^{2n} (s - \epsilon)^{2n-1}}{2n} \quad \text{if } s \geq 0 \\
      0 & \text{elsewhere}
      \]
      \[\sigma = -\frac{\partial e}{\partial \epsilon} = \frac{e^0 r_0^{2n}}{2n} \frac{(s - \epsilon)^{2n-1}}{2n} \quad \text{if } s \geq 0 \\
      0 & \text{elsewhere}\]

- **Customizing Metaballs on Lattice Nodes**
  - Low number of metaballs
  - Their center are attached to the skeleton
  - Their radius is customized to each lattice node according to its position
    - The metaball radius is chosen so as to:
      - Keep nodes outside the skull to a given distance from the surface
      - Prevent nodes inside the skull from moving deeper

\[
\rho_i = \rho_i + \rho_{1 - \exp \left[ \frac{ \rho_i - \rho_i \cdot p_i - p_0 }{ \rho \cdot p_i - p_0 } \right] } \quad \text{if } \rho_i - \rho_0 \geq \rho_i
\]

\[
\rho_i = \rho_i + \rho_{1 - \exp \left[ \frac{ \rho_i - \rho_i \cdot p_i - p_0 }{ \rho \cdot p_i - p_0 } \right] } \quad \text{elsewhere}
\]
3. Animating the FFD Lattice

• Customizing Metaballs on Lattice Nodes
  – Normalizing the strength of the metaballs
    • The base potential of the metaball should be
      chosen so as to give same repulsion
      (acceleration) whatever:
      – The mass of the lattice node
      – The radius of the metaball

\[ e_i = r_i^2 m_i \gamma \]

• Metaball-Based Collision Handling
  – Approximate collision response
  – Low computational overhead
  – Suited for animation

3. Animating the FFD Lattice

• Numerical Integration
  – Particle system integrated by using the implicit
    Inverse Euler method
  – Using the Conjugate Gradient as linear system solver
  – On-the-fly matrix assembly for capturing all the
    nonlinearities of the mechanical system
  – Good robustness with any mechanical system
    when using constant time steps

4. Scalability and Level-of-Detail

• Options for Adjusting the Accuracy Tradeoff
  – Number of lattice attachments
    • The more attachments, the more accurate the
      mechanical behavior of the hairs
  – Size of the free-form deformation grid
    • The finer the grid, the richer and diverse the
      deformation patterns of the hair
  – Number of metaballs
    • The more metaballs, the more accurate the collisions
      between the hair and the body
4. Scalability and Level-of-Detail

• Performance and Level-of-Detail
  – Computation times for a 1/25 s frame on a 3GHz Pentium4 PC computer

<table>
<thead>
<tr>
<th>Mesh Mod.</th>
<th>Lat. Size</th>
<th>0 Att.</th>
<th>0 Spr.</th>
<th>5 Att.</th>
<th>5 Spr.</th>
<th>10 Att.</th>
<th>10 Spr.</th>
<th>20 Att.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mech. Mod.</td>
<td>10x10x10</td>
<td>1.1 ms</td>
<td>5.2 ms</td>
<td>10.1 ms</td>
<td>13.8 ms</td>
<td>21.4 ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5x5x5</td>
<td>0.1 ms</td>
<td>0.2 ms</td>
<td>2.1 ms</td>
<td>3.8 ms</td>
<td>7.6 ms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. Scalability and Level-of-Detail

• Benefits of the FFD Model
  – Scalability, for efficient LOD implementations
  – Compatible with any hair rendering method
  – Weak reliance on the strand nature of hairs
  – Total freedom of hairstyle design using any standard design tool

In the MIRAnim animation engine, we use different representations for orientations:

- **Quaternions**
  - Widely used
  - SLERP interpolation
- **Exponential Map**
  - Linear interpolation
  - Small format
- **Principal Components**
  - Statistical analysis
  - Dimension reduction

Cassell et al. [Cassell99] identify two types of communicative body motions:

- **Gestures**
- **Posture shifts**

The MIRAnim Engine

Creating posture shift motions (idle motions):

- Creating idle motions from motion capture segments
- We need to add small variations in posture to make the motion look more realistic [Perlin95]
- We apply a noise function [Perlin85] onto a subset of Principal Components instead of directly on joints

• Gesture motions ➞ automatic creation of joint motions

We apply a scaling filter on each posture, to automatically generate dependent joint motions.

Animation blending:

Different levels of animation control

MIRALab
University of Geneva
www.miralab.ch
Eurographics 2006
Funding/Support

- FP6 HUMAINE Network of Excellence
  (http://www.emotion-research.net)

- FP6 ENACTIVE Network of Excellence
  (http://www.enactivenetwork.org)

- FP5 MUHCI RTN project
  (http://poseidon.csd.auth.gr/Projects/muhci/muhci_ns.html)
Case study: simulating life in Pompeii

Nadia MAGNENAT-THALMANN
MIRALab
University of Geneva
Switzerland

The making of clothes with our MIRALab Fashionizer platform

« Prêt-à-porter » Collection
Garment Creation Tools

Interactive tools for creating, draping, animating and prototyping garments on animated virtual characters [1]


Research collaboration with Adidas

Garment Prototyping

- Accurate Simulation of Parameters
  - Evaluation of strain, stress and contact pressure all over the cloth surface

Color coded visualizations of deformation parameters

Deformation in Warp direction
Deformation in Weft direction
Deformation in Shear direction
Color coded visualizations of deformation parameters

Haute Couture Garments

Mode, Passion et Collection - Le regard d’une femme” Musée d’Art et d’histoire, Genève

Haute Couture Garments

Lacroix

Lanvin

Nina Ricci

Paco Rabanne
Exhibition Robert Piguet/Yverdon

Modèle designed by Givenchy

CULTURAL HERITAGE: European Research Project Lifeplus: augmented life in Pompeii

• LIFEPLUS proposes new Augmented Reality narrative spaces for the innovative revival of life in ancient fresco-paintings in Pompeii
Scenario 1, e.g. Thermopolio di Vetuzio Placido

Scenario 2, e.g. La Casa dei Vettii

Scenario 3, e.g. La Villa dei Misteri

Research in CLOTHES AND FACIAL ANIMATION

- Real-time Cloth and hair simulation
  - Hybrid deformation
- Real-time Facial emotion expression and Speech animation
Research in HUMANS Simulation

- Real-time realistic skin rendering and interactive programmable shading module
- Artificial life methods for behavioural animation of virtual characters

Early Mixed Realities simulations in ancient Pompeii (offline AR)

AR Life: User Experience

- User arrives at the AR Life “hot spot”
  - Inform about AR Life Simulation option
  - Start theater-like immersive AR simulation
- User participation and interaction
  - Using HMD + head mounted camera
  - Watching AR revival of the ancient life
  - Dramatic scenario featuring high quality Virtual Humans
  - Blended into the real scenery
  - Moving around the scene, changing point of view
**AR Life: Functional Elements**

- **main functional elements**
  - real-time camera tracking: marker-less
  - real-time VR simulation: advanced Virtual Humans
  - blending

**Making of**
Simulating Interactive virtual plays in ancient theatres

Nadia Magnenat-Thalmann
George Papagiannakis
MIRALab – University of Geneva

Aspendos Site (161-180 A.D.)

Objectives

- Interactive Real time Visualization
- Inclusion of Virtual Humans
- Reenacting of a Roman Play
- Simulation of physically correct illumination
- Restitution of the original appearance of the site

Statues of the Scene (I)

Sources

The Gods

Imperial family
Animated Actors, Musicians and Choir

Animated 3D Actors using Motion captured data

Real time Actors

Real Time Illuminated Actors
Real-time reenacting of a scripted performance

- VR Database
- vhdRuntimeEngine
- 3D Rendering
- Sound Simulation
- Skeleton Animation
- Skin Deformation
- Cloth Simulation
- Face
- Speech
- Python Scripting
- Scenario
- VRML Loader
- XML Loader

All-frequency MR Illumination model for deformable virtual characters

- 'key-fill' illumination setup approach consisting of:
  - High-Frequency Shadow Mapping (key)
  - Low-Frequency Dynamic Precomputed Radiance Transfer (fill)

- Dynamic Precomputed Radiance Transfer
  - New Diffuse Shadow Merged Transfer
  - Multi-mesh, multi-material, articulated hierarchies
  - Real-time low order Spherical Harmonics basis for Global Illumination effects

All-frequency MR Real Time Illumination model: Results

Final Prototype Real Time Application (I)

- Based on the VHD++ MR framework for virtual characters (soon to be released as open-source project)
- Real-time simulated virtual actors with body, cloth, skin, speech simulation
- All-frequency illumination model
Final Prototype Real Time Application (I)

- Complex scene of ~300,000 triangles
- Virtual Aspendos Theatre
- Surrounding Flora simulation
- Sophocles: Antigone play excerpt

QUESTIONS
Introduction to the XVR technology: the basic framework
Franco Tecchia, Emanuele Ruffaldi, Antonio Frisoli, Massimo Bergamasco
PERCRO, Scuola Superiore Sant’Anna

General concepts

XVR: An IDE for VR applications

The importance of technology integration

XVR Architecture

XVR Workflow
What does XVR provide

- A complete and very fast 3D Scene Graph Manager
- Low level OpenGL wrapping for added flexibility
- Vertex and pixel-shaders (OpenGL2.0 GLSL)
- Basic GDI and 2D GUI functions (menus, text)
- Sound management: MIDI, MP3, 3D Positional Sound
- Collision detection & Real-time Physics
- Haptic devices control
- Network data management
- Basic streaming support
- Support for advanced devices: Trackers, Stereo projection, HMDs, Haptic Interfaces, CAVE

Why: from demos to complex applications

- Demos as single-minded applications
- Very limited interaction
- Problems can be ignored or postponed
- Most of the work to pass from concept demo to application stage
- We learn from our work and our students

Features           VS.            Complexity
---------           -------            -------
Linear increase    Exponential grow

The simplest architecture: the single loop

One loop to manage everything:
- Graphic Rendering
- User Input
- Animation
- Collision detection
- Physical simulation

~ 60 Hz

Refresh rate is usually the upper bound
Collision and physics can give troubles

Second option: separate rendering from the rest

- Animation independent from frame rate
- No more missed collisions!

~ 60 Hz

100-200 Hz

- Variables need to be shared between processes
- Atomicity should be carefully addresses
- Two threads, one rendering context?

Third step: Haptic Devices get into the picture

- The Haptic Loop is "very" fast
- More representations for the same objects

~ 60 Hz

100-200 Hz

>1kHz

- Problem: How does Haptics influence Physics?

Going further: loops are everywhere

- Network lag and dead reckoning

~10Hz

60 Hz

100-200 Hz

>1kHz

- Can’t you poll for your tracking?

~30Hz
The XVR approach: simple but not too simple

3 Main Call-backs

OnFrame()  OnTimer()

\sim 60 \text{ Hz}

100-1000 \text{ Hz}

OnEvent()

All variables are shared

\text{VR\_EVENT\_1}

\text{VR\_EVENT\_2}

\text{VR\_EVENT\_3}

\text{VR\_EVENT\_4}

The Language

The XVR scripting language (.S3D)

• It’s the fundamental glue that keeps everything together
  • Similar syntax to JAVA and BASIC
  • Byte code – virtual machine model
  • Type less
  • Object oriented

Does it have an impact on performances?

• No scripting language is as fast as C or C++
  • XVR uses bytecode generation/execution – speed similar to JAVA
  • A script-only function (e.g., Quicksort) is about 20 times slower than a C++ equivalent
  • Most of the CPU/GPU workload outside the scripting language anyway
  • In practical application we experimented very small to no impact
  • A detailed overview of the virtual machine later

But there are advantages

• Simplified syntax: script constructs tailored on VR needs
  • Light, self-sufficient development environment -> easy to set-up
  • Not much framework jargon: straight to the business
  • VERY fast compilation time (almost 0)
  • Compact bytecode, web - enabled
  • Dominant technologies are doing the same: see VirTools

S3D Statements

IF..ELSE
WHILE
DO..WHILE
FOR
BREAK
CONTINUE
SWITCH..CASE

SET

SCENE\_FOV  Field of view
SCENE\_NEAR Near clipping plane distance
SCENE\_FAR For clipping plane distance
AUDIO\_MODE 3D Audio Setup mode
Preprocessor commands

#DEFINE
#IFDEF
#ENDIF
#ELSE
#INCLUDE
#PRAGMA

Preprocessor directives, such as #define and #ifdef, are typically used to make source programs easy to change and easy to compile in different execution environments. Directives in the source file tell the preprocessor to perform specific actions. For example, the preprocessor can replace tokens in the text, insert the contents of other files into the source file, or suppress compilation of part of the file by removing sections of text. Preprocessor lines are recognized and carried out before macro expansion.

Functions

• Same as usual
• Parameter type is not specified
• Type of the return value is not specified

```c
void MoveCameraTo(float d) {
    float temp;
    temp = tan(d); // Compute a tangent
    temp = cos(d); // Compute another tangent
    temp = tan(d); // Compute yet another tangent
    return (temp);
}
```

S3D Variables

• Variable type is not explicitly declared
• The same variable can assume different types as the program evolve
• The construct “reset” can be useful!

```c
VAR i;
VAR x = 2;
VAR y = 3.1;
VAR s = "Don't panic!!!";
k = 4.3;
i = 8;
i = 9.2;
VAR v = [0.0, 1.0, 0.0];
```

S3D Types

INT, FLOAT, BOOLEAN, STRING
VECTOR: (of floats - monodimensional)
```
Var1 = [ 0.2, 2, 3.33, 4 ];
Var2 = Vector(16); // Create a 16-elements vector
Var3 = Var1[0]; // Var3 equals 0.2
```
MATRICES: Although not directly supported, often XVR functions return, or accept as input, 16-dim vectors representing matrices. In this case the matrix is stored in column-major order, like in OpenGL.

```c
VAR a = 10;
VAR a = 7;
reset a; // a = 10
```

ARRAY

An Array is a set of elements which can be accessed through an index. The elements of a S3D array can be of different type. The array size can be changed at run-time.
To specify the index in order to access the desired element, the operator [ ] is used.
```
VAR1 = { 1, true, { 12, 3 } }; // Array made of different types
VAR2 = Array(12); // Create an array of 12 elements
VAR3 = Var1[0]; // Var3 equals 1
VAR4 = Var1[2][0]; // Var4 equals 12
VAR1[1] = false; // New Var1[1] equals "false"
```

Classes

• S3D is heavily object-oriented
• Built-in objects
• Script-based classes
• Imported objects (DLLs)
Programming with XVR

How is an application organised?

- Any XVR application is based on 6 fundamental functions (built in all the scripts)
- Currently there are 2 threads calling these functions. There could be more in the future.
- Atomicity of the operations is guaranteed. Global variables are properly shared between threads.

OnDownload()

- Every time an XVR script is executed it creates a temporary working directory (inside the user temp)
- OnDownload is used to move files from their original location to the working directory
- Zip files are automatically exploded
- Generic code can be executed inside OnDownload

OnInit()

- Once the download is complete, this is the right place for initialization code
- Please note that OnInit accepts a "params" value. This is the "UserParam" string reported in the HTML code
- There is an implicit OpenGL "SwapBuffer" when OnInit ends

OnFrame()

- Function dedicated to rendering. By default it gets called after each screen refresh (60 Hz).
- Desired framerate can be changed using SetFramerate(Value)
- It’s the ONLY function that has access to the OpenGL frame buffer
- All the rendering code must be inside this function (or inside functions called by OnFrame)
- The maximum Frame Rate allowed is 100 Hz

OnTimer()

- Function normally dedicated to manage a high speed loop. By default it gets called at 100Hz
- Desired time-step can be changed using SetTimestep(Value)
- Minimum time-step = 1ms
- Independent from framerate (unless the system is overloaded)
- Excellent place for updating objects positions, sound, real time physics, haptic interfaces and network programming.
- It has no access to the framebuffer
OnEvent()

- Function under construction, coming soon
- Used for Event-driven programming
- It gets called when an EVENT arrives
- EVENTS can be generated inside XVR (GenerateEvent(val, param)) or outside XVR (The Win32 Events will be intercepted)
- Extremely useful in state machines programming

The typical rendering code

- Handling Mouse (or keyboard) input
- SceneBegin(), SceneEnd(), a very common XVR graphical construct useful for multipass rendering, picture in picture, stereo rendering
- Mixing low-level calls to scene graph rendering

The XVR Scene graph

- A flexible built-in scene graph
- Plenty of optimizations and tricks, good performances
- GLSL Vertex and Pixel Shaders support

A simple, script based GUI (EasyGUI)

Knowing how XVR is organised

- XVR architecture is historically web-oriented
- Applications normally run inside a webpage
- The core module is an ACTIVEX control

Embedding the XVR ActiveX in a webpage

```
<OBJECT ID="My3DCanvas"
        WIDTH = 640
        HEIGHT = 480
        CLASSID="CLSID:55589287-1496-4223-AE64-65FA0785858A"
        TYPE="application/x-oleobject" align="left" border="2"
        CODEBASE="http://client.vrmedia.it/XVRPlayer.cab#Version=1,0,0,900">
  <PARAM NAME="ScriptName" VALUE="AppName.s3d.bin">
  <PARAM NAME="EngineVersion" VALUE="0136">
  <PARAM NAME="BackgroundColor" VALUE="#000066">
  <PARAM NAME="Params" VALUE "DEPTH=24">
  <PARAM NAME="UserParams" VALUE "...">
</OBJECT>
```
HapticWeb
Haptic rendering and interaction with XVR

Overview

• Introducing Haptics for XVR
• Programming Model
• Features
  • Devices
  • Mesh Objects
  • Effects
  • Capture
  • Integration with Physics
  • High Level Example
• Future Directions

Haptics for Virtual Reality

3D Graphics Application

Low Level
OpenHaptics
Delta
VRPN Haptics

Hi Level
H3D
OSG Haptics
EHAP
CHA3D

HapticWeb - Haptics for XVR

Delta
Sensible
GRAB
Access to Devices
Virtual Device
Low Level

Auto Device
Device Independence

Haptic SceneGraph
High Level Haptic Rendering
OpenGL Capture

HapticWeb structure

XVR

HapticXVR
GRAB
Delta
Additional Devices

NovodexXVR
Dynamic Simulation Layer

HapticWeb Examples

XVR
Provides Haptic Rendering using the CHAI3D library

Haptic Web structure
Programming Model

- Haptic Rendering
  - Low level force command to the device
  - Haptic Scene Graph with Geometry and Effects

- Haptic Loop
  - External Thread Haptic Loop
  - OnTimer based

Haptic Scene Graph

- Based on the CHAI 3D scene graph
- Simplifies the Haptic Rendering
- Hierarchy parallel to the Graphics one

Haptic Loop

- Based on the CHAI 3D scene graph
- Simplifies the Haptic Rendering
- Hierarchy parallel to the Graphics one

Application Setup

```javascript
#include "hapticweb.h.s3d"
var h_world;
function OnInit()
{
  HLoadModule ();
  h_world = HWorld ();
  ...
  h_world.run ();
}
```

HapticWeb Deployment

- Which files are needed by HapticWeb?
  - Use provided hapticxvr.zip
- Core Haptics
  - hapticxvr.dll - the extension module
  - PhantomDriver.dll and PhantomXXX.dll
  - Sphere.aam - for the graphics
- Additional Devices
  - dlportio.dll - for Delta
  - GrabExtdll.dll - for the PERCRO’s GRAB
- Physics
  - Novodexvr.dll
  - NxFoundation.dll and NxPhysics.dll

Loading Modules during Development

- The HapticWeb module can be loaded using
  `LoadModule("hapticxvr.dll");`
- But the additional files for drivers are provided by the
  HapticXVR.zip
- During the local development it is possible to reduce the
  unpacking time by using the AutoLocal approach:
  1. The AutoLocal function returns true if we are running on
     local instead of Web
  2. Use the GetScriptPath function to get the script directory
  3. Change directory using SetCurrentDir
     ```javascript
     if(AutoLocal(forceLocalMode))
     SetCurrentDir(GetScriptPath());
     ```
HOobject

- The basic haptic class that corresponds to the cGenericObject of CHAI3D
- HOobject provides:
  - Hierarchy (add/removeChild, getChild)
  - Bounding Box
  - Transformation (rotate, translate, rotation, position)
  - Rendering (for debugging)
  - Additional XVR data: tag

Haptic Devices

- HapticWeb supports all the CHAI3D devices and PERCRO devices
- The Haptic Interface is represented by a 3/6DOF Haptic Point of Contact of class HTool
- The device can be specified by using an URL or automatically detected
- If nothing is found the CHAI3D Virtual Device is activated (if dhdVirtual.exe is present)

Haptic Devices URL

- We use an URL for using a specific device or automatically selecting the attached device
- Currently supported URLs:
  - null: - useful for debugging and custom virtual device
  - ehap://grab/left and ehap://grab/right for the left and right arm of the GRAB
  - virtual: - standard CHAI3D virtual device
- The URL will be used to specify remote devices or additional device parameters

HTool class

- Manages a 6DOF point of contact and a proxy
- Device:
  - forceEnabled : bool
  - devicePosition : vector in device coordinates
  - deviceOrientation : vector as 4x4 matrix
  - deviceLocalPosition: vector in tool coordinates
  - force : vector of force in N
  - torque : vector of torque in N mm
  - buttonPressed : bool
  - buttonCount : int
  - deviceURI : string (useful for testing initialization and device)
  - initialize() start() stop()

Virtual Device

- The standard CHAI3D virtual device based on an external application (dhdvirtual.exe) connected by Shared Memory
- The contact point is controlled by the Mouse
- Useful for debugging or when the device is not available
- The Shared Memory can be used for interfacing other devices (e.g. using Simulink)
- The Virtual Device could be implemented in XVR inside the application using the null: device
  - just set devicePosition before updatePose

HTool Workspace

- Workspace and scaling are fundamental aspects of Haptic interfaces
- By default the devicePosition is in mm
- The property workspace returns the current workspace of the device. By default it is mm. For PHANTOMs it is a Cube
- The workspace can be changed setting the workspaceScale factor of the tool
- IMPORTANT: stiffness should be expressed in N/mm and divided by workspaceScale before assigning it to meshes
  - virtualStiffness = stiffness / workspaceScale
HTool Proxy

- The classic God-Object proxy algorithm for Haptic Rendering
- Extended with Friction
- Properties:
  - contactObject : HMesh
  - hasContact : bool
  - lastNormal : vector if hasContact
  - proxyPosition : vector
- In OnTimer mode is updated by computeForces

The Haptic Loop

- When using the External thread model HapticWeb manages the haptic rendering loop
- Simply call `h_world.run()` in the OnInit

The Haptic Loop Diagram

1. `computeForces` - computes forces using the proxy and the Haptic Effects
2. `applyForces` - sends the forces to the device. It allows to modify the generated force

Example of Sphere rendering

```c
#include <Script3d.h>
#include "hapticweb.s3d"
var h_world;
var h_tool;
var sphereCenter = [0,2,10];
var sphereRadius = 5.0;
var sphereStiffness = 2500; // N/mm
function OnDownload()
{
  FileDownload("hapticxvr.zip");
}
function OnInit()
{
  HLoadModule();
  h_world = HWorld();
  h_tool = HToolGL(h_world);
  h_tool.tool.start();
  h_world.run(false);
}
function OnTimer()
{
  var tool = h_tool.tool;
  tool.updatePose();
  var delta = tool.devicePosition - sphereCenter;
  var d = modulus(delta);
  if(d > sphereRadius || d == 0)
    tool.force = [0,0,0];
  else
    tool.force = sphereStiffness*(sphereRadius-d)*(delta/d);
  tool.applyForces();
}
```

Adding a mesh

- An `HMesh` object is a triangular mesh that can be haptically touched using the standard proxy algorithm
- Geometry
  - Created from AAM/3DS/Obj files
  - Created from CVmMesh/CVmNewMesh/CVmObj (with all children)
  - Created explicitly using newVertex/newTriangle
  - Created from OpenGL capture using begin/endCapture
- Surface Properties
  - friction – set using `setFriction(ms, md)`
  - stiffness – should be scaled by workspaceScale

Adding a mesh

From a file

```c
var h_world;
var h_mesh = HMesh(h_world);
var aamMesh = loadFromMesh("x.aam");
```

From a CVmObj or CVmNewMesh

```c
var h_world;
var h_mesh = HMesh(h_world);
h_mesh.loadFromMesh(mymesh);
h_world.addChild(h_mesh);
h_mesh.stiffness = 10;
```

Creating a Mesh

```c
var h_world;
var h_mesh = HMesh(h_world);
h_mesh.newVertex(0,0,0,0,0);
h_mesh.newVertex(size.x,0,0,0,0);
h_mesh.newTriangle(0,1,2);
h_mesh.newTriangle(0,2,3);
h_mesh.computeBoundaryBox();
h_mesh.computeAllNormals();
```
Capturing a Mesh

- It is possible to use the OpenGL Feedback mode to construct a HMesh from a block of OpenGL primitives

```javascript
var h_mesh = HMesh(h_world);
h_mesh.beginCapture();
glBegin(GL_TRIANGLES);
...
glEnd();
h_mesh.endCapture();
```

- The primitives are mapped from the world frame into the mesh local frame, but only if visible in the viewport

Haptic Effects

- Programmable Haptic effects imposed over the Haptic Rendering

- Constant spring friction point snap line constraint plane constraint

The Haptic Rendering Pipeline

- HEffect class
- Add to the HWorld by addChild
- Common Members
  - type
  - enable
  - Constant: direction, magnitude
  - Spring: gain
  - Viscous: gain
  - Friction: gain
  - Plane: direction, gain

- Explicitly enabling by enabled
- Time based behavior for triggered effects:
  - Set duration in seconds using duration
  - Trigger the effect using trigger

Constrained Haptic Effect

- HEffect allows to define force constraints that are applied to a point of contact
- Constraints can be:
  - point
  - line
  - Plane
- A constraint is active if the distance to that is inferior to the snapDistance value.
- The constraints are created by using the OpenGL commands between begin/endCapture

Integration with Physics

- Depends on the Interaction Model
- In general we have three representations

- Touch and push model:
  - Synchronization of Position and Rotation
  - Send forces accumulated during the Haptic Loop by the touching. Use threshold for allowing surface exploration
Virtual Coupling

- Useful for simulating the grabbing of a Body
- Allows to perceive the collisions and the forces applied to the attached Body
- It is based on a damped spring that connects the Haptic Contact Point with a Body
- Two state interaction
  - Selection phase in which objects can be touched
  - Grabbing phase in which the Haptic Tool drags a Body
- The force feedback is just the stiffness of the string by the string extension. It can be done both in thread mode and in the OnTimer
- The dynamic simulation can be done manually using the applyForce or better using a Spherical Joint with a Spring

(Haptic 1998)

VirtualObject

- A XVR class that provides VirtualCoupling
- **Haptic**: using a H_SPRING effect
- **Physics**: compute the spring force manually
- Usage:
  1. Create using NSim and HWorld
  2. Attach to a physics object
  3. Call it before and after the Simulation

Virtual Coupling Example

```javascript
var h_world, p_sim;
var h_tool, p_tool, h_virtual;
var p_obj;

function OnInit()
{
  HLoadModule();
  NLoadModule();
  h_world = HWorld();
  p_sim = NSim();
  h_tool = HToolGL(h_world);
  h_virtual = HVirtualObject(h_world, p_sim, h_tool);
  h_virtual.stiffness = 5;
  p_obj = p_sim.createBody(N_GEO_SPHERE, 5);
  p_obj.mass = 0.5;
  h_virtual.attach(p_obj);
  h_world.run();
  ...
}

function OnFrame()
{
  h_virtual.preStep();
  p_sim.step();
  h_virtual.postStep();
  ...
  h_virtual.render();
}
```

Entertainment Application: Tennis Table

S. Marcheschi

Introduction

Experimental set-up of virtual tennis table game

General architecture

- **OnDownload()**
  - Load Haptic Interface (HI) lib
- **OnTimer()**
  - Evaluate integration timestep
  - Get position data from HI
  - Update dynamic loop
  - Send force data to HI
- **Online()**
  - Load mesh
  - Initialize data structures
  - Initialize HI lib
  - Set time step to 1 ms
  - Move camera with mouse
  - Draw table, pad and ball
  - Get keyboard input
  - Display data
**Data structures**

<table>
<thead>
<tr>
<th>Class</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ball</td>
<td>Data, View, Din, Radius</td>
</tr>
<tr>
<td>Pad</td>
<td>Data, View, Din, Radius</td>
</tr>
<tr>
<td>Table</td>
<td>Data, Din, Lx, Ly, VerX, VerY</td>
</tr>
</tbody>
</table>

**ObjData**
- Mass
- Inertia
- Stiffness
- AirViscosity

**ObjDin**
- Force
- Torque
- Position
- Velocity
- Acceleration

**ObjView**
- ScaleMesh
- ScaleGlo
- RotMatrixNotScaled
- Obj

---

**Integration Timestep: Differential Integration Step**

\[ T = \text{GetPerformanceTimer}() \]
\[ \Delta T = T_i - T_f \]
- **PRO**: Real-time simulation
- **CONTRA**: Integration step discontinuities

---

**Integration Time Step: Fixed Integration Step**

\[ T = \text{GetPerformanceTimer}() \]
\[ \Delta T = \text{GetPerformanceTimer}() - T \]
- **PRO**: Continuous integration step
- **CONTRA**: Non-real time (delayed) simulation

---

**Integration Time Step: Variable Integration Step**

\[ \text{Trendr} = 0.0; \]
\[ \text{Tcurr} = \text{GetPerformanceTimer}() / 1000.0; \]
\[ \text{if} \ (\text{Tcurr} - \text{Trendr}) < \text{DTmax} \]
\[ \text{Trendr} = \text{Tcurr}; \]
\[ \text{else} \]
\[ \text{Trendr} = \text{Trendr} + \text{DTmax}; \]
- **PRO**: Quasi-continuous integration step
- **CONTRA**: Quasi-real time simulation

---

**Dynamic loop - Ball**

\[ X_{t+1} = X_t + \frac{F}{m} \Delta T \]
\[ v_{t+1} = v_t + \frac{\Delta T}{\Delta t} \]
\[ \Delta v = \frac{\Delta X}{\Delta T} \]
\[ a = \frac{\Delta v}{\Delta T} \]

---

**Dynamic loop - Pad**

\[ F = F_b + F_g \]
\[ F_b = \text{Collision with ball} \]
\[ F_g = \text{Gravity} \]
Future Directions

- Freezing the API
- Clearer Device Instantiation
- More support for multi point
- Networked Device
- Support for PERCRO EHAP haptics library using the same XVR interface

Exploring more

HapticWeb http://www.hapticweb.org

Thanks to the CHAI 3D’s team