Towards Physically Rendered Environments

Veljko Krunic, Richard Han

Department of Computer Science
University of Colorado at Boulder

Technical Report CU-CS-1033-07

September 5, 2007

Towards Physically Rendered Environments

Veljko Krunic, Richard Han
University of Colorado, Boulder
krunic@ieee.org, Richard.Han@colorado.edu

Abstract

We present an early vision of a system that allows computer controlled rendering of physical surfaces, terrains, and environments by manipulating grids of "physical pixels" or rods whose heights can be raised and lowered on command. A user would be free to walk within such a dynamically deformable physically rendered environment (PRE). The system would be able to create on demand the floor, ceiling and sides of the "Holodeck" in which the person resides, and vary the slipperiness of the surfaces to provide authentic tactile feedback in real time. Ideally, the system would support discreet relocation while a person walked to create the impression of infinite distance. Such a system could be combined with immersive graphic visualization and futuristic programmable matter to convey realistic impressions of a walking tour of a remote site, the experience of climbing a steep mountain, or navigation through a maze. The potential applications of this system range from entertainment to police and infantry training to education, and extend as far as the imagination allows. We propose mechanisms with which we believe that such a vision could be implemented with the technology of today. This paper is an early work intended to raise interest in this idea, as opposed to presenting a finished solution for achieving the vision of a Holodeck.

1. Introduction

The "Holodeck" of StarTrek fame presented the vision of a room in which an immersive virtual world is created around a user by a computer, which is capable of creating arbitrary objects and terrain of high realism in real time. In addition, the user was free to move through such an environment, and objects could be materialized seemingly out of thin air. While this complete vision is still science fiction, this paper offers a pathway towards achieving a Holodeck by describing how the technology of today can be used to generate physically rendered environments (PREs) of emulated three-dimensional terrains with geometric and tactile realism.

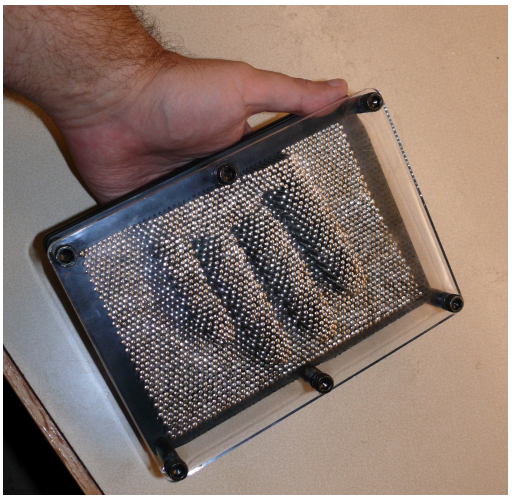

Figure 1. PinPoint toy with the impression of a human hand rendered in the physical pixel rods.

Figure 1 shows a PinPoint toy that consists of a set of needles that are free to move in the up/down direction. If a user presses their hand on the bottom of the needles, the needles would raise in the approximate 3D shape of the hand, as shown in the figure.

By augmenting this rod-based motion with computer controlled actuation of the rods in real time, dynamic surfaces, terrains, and even forms of motion, e.g. waves, would be capable of being rendered, as shown in Figure 2. Over the size of a room, deformation of the ground, walls, and ceiling would simultaneously create entire 3D environments within which a user could stand and move. The realism would be further enhanced by equipping the tip of each rod with a surface, such as a ball bearing, disc or ring, whose coefficient of friction could be varied. This would provide tactile feedback that approximates the slipperiness of actual physical surfaces. Combining the PRE with complementary tech-

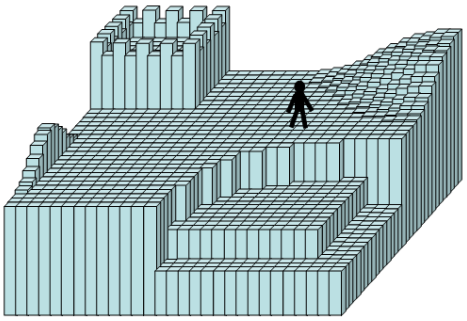

Figure 2. A physically rendered environment (PRE) or Holodeck consisting of a grid of computer-controlled physical rods or pixels whose displacements can be raised and lowered.

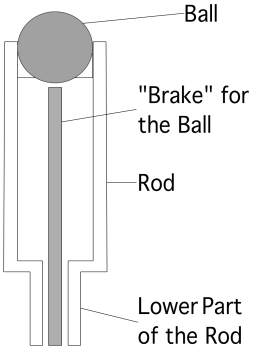

Figure 3. Each physical pixel consists of a rod whose height can be raised and lowered. The tip of each rod provides a mechanism for controlling the coefficient of friction, in this case a ball whose rotation can be adjusted or braked.

nology such as virtual or augmented reality, e.g. immersive graphics visualization via projection and/or head-mounted helmet displays [7], would be an active area for further exploration.

A user would be able to stand within this computer controlled physically rendered environment, indeed even on top of the rods themselves. A user would be able to grasp onto protrusions in the walls and lift themselves just as in a climbing wall, except this wall would be programmable, e.g. any part of a famous climb could be rendered on demand in one's living room.

To enhance the interactivity of the PRE, our goal is to convey the effect of infinite distance via subtle displacements that re-center a user while they're walking or climbing. Achieving such an effect would be one of our main challenges. Offering such a capability would allow a user to conduct arbitrarily long hikes, tours, or climbs. Our intent is to offer such a translation effect in two ways. First, the surface would be translated by raising and lowering the rods in the appropriate sequence to shift the surface back towards the center of the PRE, while preserving the shape of the surface during the shift. The user standing on or climbing this surface would be translated along with the surface by the up/down actions of many rods. The effect would also be adaptive, i.e. translation would stop as soon as the surface and user are shifted enough to be re-centered. Second, the coefficient of friction on the rods would be varied in concert with the raising and lowering of rods to ease the job of sliding the user back towards their original position. For example, a user standing on an incline could be re-centered by having the surface friction of rods underneath the user subtly change to slippery, so that the user's own weight would slide them down until they hit a rougher surface stopping the slide. The same two approaches would be used to translate other objects in a PRE, not just the human user. These techniques could be combined with treadmill-based approaches.

Let's take a closer look at a single physical pixel or rod, as in Figure 3. On top of the rod is either a ball or a surface whose coefficient of friction can be varied. In the case of a ball, its ability to rotate and/or rate of rotation would be adjustable, e.g. by braking, to give varying degrees of resistance. When both the height of each rod and the spin of each rods ball are computer controlled, we would control not only the shape of the terrain, but also the slipperiness of the surface under the users feet. This would allow simulation of an icy cave, or a rough climbing surface.

To summarize, the PRE that we envision would be capable of at least the following features:

- computer-controlled rendering of arbitrary physical surfaces on a grid of physical pixel rods
- in real time
- with realistic tactile feedback
- and realistic visual rendering
- able to support the weight of a standing, walking, or climbing person
- giving the impression of infinite distance

This white paper is *early* work that is intended to raise interest in the vision of computer controllable deformable terrains. Many questions about practical implementation of this vision, its limitations, price and practical applicability still remain open. However, we believe that a prototype of

this vision is achievable with the technology available today, both on the software and hardware side.

Our paper is organized as follows. Section 2 elaborates on motivating applications for this work. Section 3 presents related work, and Section 4 describes the architecture of the system. Section 5 discusses software control of the PRE, and Section 6 discusses how physical implementation of this system could be achieved. Section 7 describes simulation of the PRE. Section 8 summarizes the paper.

2. Motivating Applications of the PRE

Potential applications of the PRE include but are not limited to:

1. Remote tours of landmarks and facilities that are distant in place and/or time. For example, a user could climb the steps of the Great Pyramid, hike the Great Wall, navigate through a remote factory assembly plant, walk through a famous golf course (potentially in concert with playing a simulated golf game), or tour Ancient Rome as it once was. Versions of such an application are already available in programmable treadmills in which elevation changes are simulated by adjusting the steepness of the treadmill.
2. A climbing wall in one's home entertainment room. The PRE could be programmed with any number of difficult or famous climbs and traverses, such as scaling Mt. Everest or Aconcagua.
3. Interactive gaming in one's living room. The PRE could be programmed to simulate a virtual environment, say a subterranean cave that never existed, a fairy tale castle, or a futuristic city.
4. Assistive technology. A scaled down version of this system the size of a box could be used as assistive technology, allowing a blind person to perceive 3D depth and shape.
5. Search and Rescue Training on a large scale. The scale of the PRE could be extended to the size of a building or beyond. Obstacles and rooms would be created on demand. Firefighters could be trained to navigate through terrain that keeps changing, i.e. obstructions could appear in front of and/or behind the trainee, emulating a real fire that blocks the way forward and/or the exit behind. Different members of a fire team could be distributed across different rooms in the PRE while simultaneously facing a sudden failure scenario that requires collective team action to overcome. Police SWAT teams could be trained to navigate through interiors in which a large number of rooms could be formed and changed in fast succession with the raising and lowering of physical pixel rods without a need for the team to pause for the rooms to be rearranged. Slipperiness of floors could be adjusted on the fly, to simulate icy conditions, wet floors, etc.
6. If such a building-scale PRE is filled with water, then it would be possible to train divers to navigate through sunken ships.
7. Combat training, especially in urban areas. A PRE would allow soldiers to experience how to navigate urban terrain that is created on demand to simulate any number of dangerous scenarios. Just as in immersive flight simulators[12], parameters could be changed suddenly to test for a critical failure scenario, e.g. force a foot to slip during a critical portion of a reconnaissance mission. A more benign version of this application would enable paintball and lasertag games in maze-like buildings whose layouts are created on demand and capable of morphing during a battle.
8. Rapid prototyping of 3D objects. The PRE can be used to quickly create rough injection molds. The outer shell of the mold would be formed by the PRE, and would then be injected and dried. This enables quick approximation of the desired 3D shape, albeit with pixelated surfaces, which may be appropriate for certain types of rapid prototyping.
9. Wall art and furnishings. Arbitrary surfaces could be rendered on a wall on demand, e.g. mosaic patterns, carvings, sculpted art, even a human face, and more simply, shelves.

These examples illustrate a variety of exciting applications of PREs. In addition, there are several interesting properties of a PRE that emerge. When the scale of a PRE is extended beyond the size of a living room, then the PRE can be used to conduct team-based training exercises. The real-time adaptability of the PRE enables paths and layouts to change during an exercise, simulating blocked passages and new-found obstacles to navigate around. Also, the central "atmosphere" of a PRE can be injected with water or another substance to enable new applications of a PRE.

3. Advanced Capabilities of the PRE

To enhance the realism of large scale PREs, it would be beneficial to hierarchically embed PREs within PREs. This would be useful, for example, in creating protrusions from walls in the interior of a PRE. Otherwise, only the outer edges of a PRE would provide any emulation of non-flat walls, while all interior walls would be flat. The same concept can be scaled down to a room-sized PRE to enable, for example, the creation of a table within a PRE. This would be accomplished by embedding a PRE-like *"object"* within a PRE whose surfaces are all pixel-based and are capable

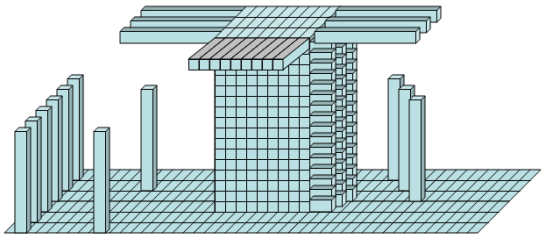

Figure 4. A PRE *object* within a PRE enables the rendering of shapes in the interior of a PRE.

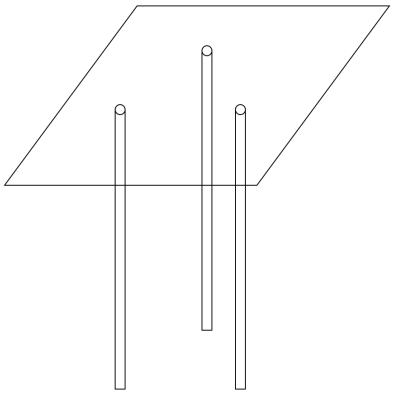

Figure 5. Plate articulation based on mounting a plate on three rods.

of extending into the PRE. A limited number of such objects would be embedded in the PRE and could be raised and lowered into/out of the PRE much like a single pixel. However, a PRE object would displace a large number of pixel rods. Once raised into the PRE, the object's various surfaces could be extended out into the room. This capability is shown in Figure 4. A similar mechanism would be used to create a window in a wall. Telescoping physical pixels would extend the range of pixels beyond the displacement of the object.

Another example of hierarchy involves rod layout and manipulation. The floor of the PRE could be segmented into plates. Each plate would be independently controllable in terms of angle and height by a spherical joint and/or three rods, as shown in Figure 5. Each plate would contain its own grid of rods. Such a system would enable rendering of pixels at angles other than perpendicular to the ground plane. Such a system may allow for more convenient or faster manipulation of groups of pixels on a plate granularity. Parallels on the hierarchy can be made with computer graphics, in which surfaces are shown coarsely as a series of polygons, while fine details are drawn by texture mapping. Here, plates and large rods are analogous to polygon-scale manipulation, while small rods on the plate are equivalent to the texture map.

Another capability of the PRE enables zooming into and out of physical scenes. In an Alice-In-Wonderland effect, surfaces and shapes in the room could be quickly scaled larger or smaller in comparison to the user.

We expect that sensing technology will play an important role in the PRE to provide feedback of user actions. For example, if a user sits in a rendered throne, the seat of the throne could become contoured to fit the user. If a user walks on certain pixels representing swampy or sandy ground, the pixels could give way as the user steps on them. The user's location and/or orientation within the PRE also should be known.

As mentioned earlier, PREs are capable of translating objects, in particular human users, by raising and lowering rods while also changing their tips' coefficients of friction. Other objects that are placed in the PRE can also be translated, e.g. a soccer ball or furniture. In addition, PREs can deform the ground underneath objects, causing tilting or other motion.

In addition, the PRE system is complementary to other technologies such as immersive head-mounted displays and interactive user interfaces such as the Nintendo Wii [13]. It would be advantageous to combine the PRE with such technologies to enhance the Holodeck-like experience. If the PRE is to be viewed directly without the assistance of a head-mounted display, then color realism could be enhanced by emitting light of appropriate colors through the tips of translucent rods, thereby painting in effect a mosaic on top of a 3D surface.

Networking together distributed Holodecks would be another facet of team-oriented training, where individuals in the team need not all be in the same place. Instead, individuals in geographically separate PREs can be networked together to train as a team, even though each individual's PRE only renders a local physical view as seen by that user. The graphical views of each user would also need to be coordinated, as in a cooperative multi-player game.

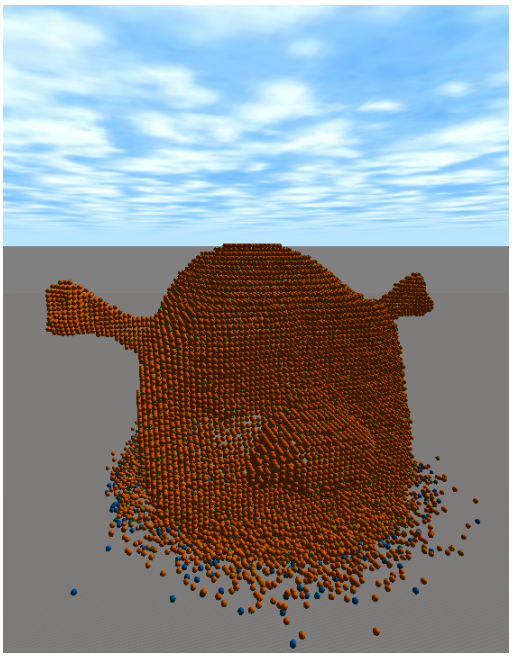

Figure 6. Catoms forming 3D face [27].

4. Related Work

Intel's Dynamic Physical Rendering (DPR) [1] is a vision of using small physical elements called *catoms*, e.g. balls, to create arbitrary 3D shapes as seen in Figure 6. Catoms are able to orient among themselves in arbitrary positions. The DPR concept is similar to micro-robot ensembles and programmable matter [2]. DPR is in its early stages, with the current prototype using two cylinders with electromagnets on them to achieve orientation, but work is under way for a more advanced prototype. We view PRE technology as able to render large surfaces quickly and with modest computational cost, making it a natural complement to DPR technology. Shapes internal to a PRE, such as a ball, would be rendered using DPR, while other shapes, such as a table, could be a hybrid combination of PRE rods, PRE objects, and DPR rendering.

The proposed PRE system relates to Solid Freeform Fabrication (SFF) [22], [21], [23], as 3D objects could be rendered by the PRE to assume the form of a desired object surface. Moldable material could then be injected into the PRE. In this way, the PRE could provide a rapid but approximate method to mold and print 3D objects.

Omnidirectional treadmills [3] give a user the impression of walking through an effectively unlimited environment even though they are physically staying in the same place. The Sarcos treadmill [6] combines an angled treadmill with a mechanical tether and a CAVE-like immersive graphical interface. The angle of inclination can be controlled. The mechanical tether presents forward resistance and sideways pressure to the user. There are a variety of limitations to this approach. The bundling of the user to the tether limits locomotion, i.e. kneeling and rolling are difficult to realize. On the other hand, a PRE is designed to permit full human motion. Also, our PRE-based Holodeck is designed to enable fine-grained realization of irregular terrain and slopes that are not flat, whereas the treadmill approach is largely limited in what it can emulate by its single planar surface. The PRE also enables geographically differentiated emulation of varying degrees of slipperiness, which is not addressed by the treadmill approach. Both the PRE and treadmill approaches may benefit from sharing of ideas. For example, we expect to incorporate lessons from the treadmill on how to reposition a user as imperceptibly as possible. In addition, the PRE could be augmented with rod-based treadmills, i.e. a treadmill whose surface consists of a grid of rods is inserted into a PRE. One way to realize such a treadmill is to subdivide the surface into separate plates. Conversely, treadmills may benefit from incorporating PRE technology, e.g. so that the slipperiness of the surface can be varied with fine granularity as the treadmill rolls.

An alternative approach that gives a user the physical illusion of walking arbitrary distances is offered by devices like the GaitMaster [4], shown in Figure 7. The device has two pedestals or pedals on which a user stands, one for each leg. The user's complete weight is supported by the two pedals. The pedals move with the user's legs, and the combined motion of the pedals keeps the user in place while giving the user the illusion of walking forward, e.g. as the user is walking forward, one pad is moved forward for the front foot, and another pad is moved backwards. In addition, the platform is mounted on a spherical joint that moves the same way that the user does. This type of device confines human locomotion and does not fully support our objectives for the PRE, namely the ability for a human user to move freely through the PRE, e.g. by laying down, kneeling or rolling. As with treadmills, cross-fertilization of ideas may benefit the PRE both in terms of learning how the user's sense of balance is affected by constant repositioning in the GaitMaster[5] and in how lowering/raising the pedals can simulate mashy ground and clay. Conversely, pedals on the GaitMaster could be modified with the PRE pins to present local textures, a concept which we term "porcupine pedals".

Augmented reality systems combine physical objects with computer generated imaging [7]. Many concepts from augmented reality such as helmet-based split-view immersive environments [7] are quite complementary with the vision of physically rendered environments. Depending on the application, we envision that users navigating through a PRE can be equipped with helmet-mounted immersive displays. The PRE provides computer-controlled rendering of

Figure 7. GaitMaster keeps a user's legs on the movable pedals [28].

physical surfaces, walls, steps, hills, and deformable ojbects on demand in a coordinated fashion with what the user is viewing in the head-mounted display. In some cases, the user's entire vision may be immersed by the helmet, while the PRE provides the realistic physical feedback or "feel" of the surfaces.

The designers at amusement parks, e.g. Imagineers at Walt Disney [19], as well as the designers of sophisticated stage sets [18] explore effects that are distantly related to the PRE-based concepts that we are proposing. Objects and props rise from below to the stage floor or descend from the ceiling in electromechanical coordination. However, these objects are typically quite specialized and fixed in form beforehand, unlike the PRE where much more universal surfaces and forms can be rendered.

Haptic interfaces [20] allow computer stimulation of the user's sense of touch. This approach shows a lot of promise for allowing the perception of touch in the environment, but haptic interfaces are not generally not concerned with locomotion in an environment. Our work is complementary in that it is likely that we will need to integrate concepts from haptic interfaces in order to provide textures that give the right degree of slipperiness or roughness.

Finally, somewhat related work is a Braille enabled terminal [17]. The system that we are proposing could enhance a Braille terminal, as it would not only enable reading of the Braille alphabet, but would also enable perception of new 3D shapes.

5. System Architecture

To realize the vision of a Holodeck through physically rendered environments, we intend to employ a high-level system architecture such as depicted in Figure 8. The architecture is similar to the flow of data through graphics-based rendering engines, and consists of both a software component for issuing commands to control the displacement of the rods, and actuation technology for raising/lowering the rods. Building such a system will require substantial innovations both in computer science and electro-mechanical engineering. We describe in Section 6 the structure of the software computer control, modeling and rendering system. As a means to prototype and test various concepts prior to building complex and expensive hardware, we describe a simulation environment in Section 7 that renders pixel rods and their motions in a virtualized graphical world, but as if in a real PRE, e.g. the inertia of rods is factored into the simulation.

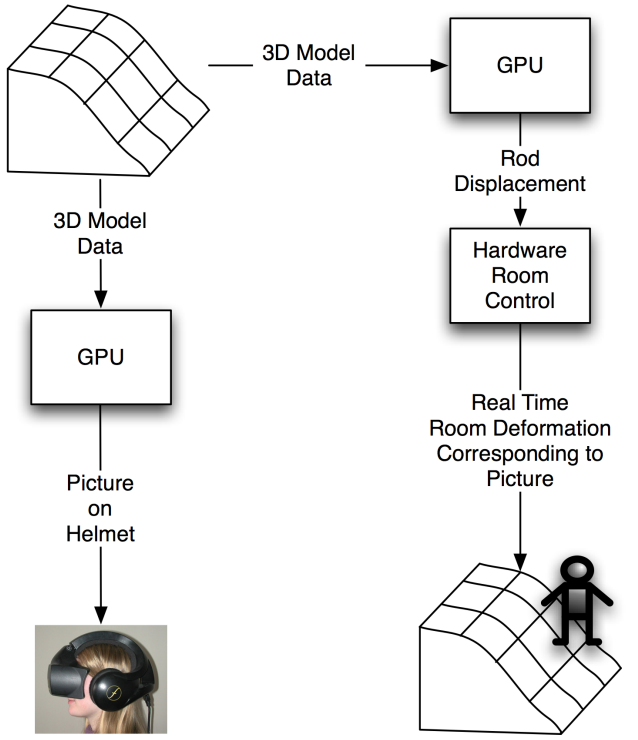

Figure 8. System Architecture

The system begins by creating a 3D model in software of the environment to be rendered. The 3D model must contain physical characteristics of surfaces being modeled, including shape, texture and slipperiness. From the 3D model, we extract the sequence of actions needed to render the physical surfaces in the environment. From the same 3D model, we can generate both graphical images that are shown within a user's helmet-mounted display as well as corresponding physical terrains that are rendered within the PRE, thereby providing an even deeper sense of immersion. Thus, two coordinated rendering paths emerge from the same core 3D model.

The example in the figure demonstates deformation of

only one plane, e.g. the ground plane, but the concept is straightforward to extend to deforming other edges of the room beside the floor, e.g. ceiling and walls. Thus, the physical rendering engine may be drawing six different surfaces, or even more if internal PRE objects are factored into the scene.

The Graphics Processing Unit (GPU) normally takes specific graphics commands and renders them on screen. The GPU renders specific types of compute-intensive commands more quickly than generic processors. The next section proposes ways to adapt GPUs to support real time physical rendering.

6. Basis of Software Control

This section describes the basis of software control of the grid of rods. The exact control algorithms depend on the larger system in which the PRE-based Holodeck is implemented. In the following discussion, we assume for simplicity that there is one planar plate on the floor, and that its pins are perpendicular to the plate, rising up from the floor.

What are some of the major technical challenges faced in software control of the PRE?

- How do we render a physical surface from a standard 3D graphical model?
- How do we integrate physical limitations of the rods into a standard 3D graphical model?
- How do we minimize the number of changes to the rendering engine in order to realize software control of the Holodeck?
- What is the best way to move the rods to give the most realistic impression of natural motion to the user?
- How do we factor the user's safety into the physical rendering decisions?

The first three questions are addressed in the following subsections.

6.1. Z Buffer and GPU Usage

Our first observation is that the Z-Buffer found in typical graphical systems can be exploited to help us identify and render physical surfaces. In a standard graphical environment, where a viewer is looking at a rendered 3-D scene, the Z-Buffer contains the distance of each pixel from the viewer. If we position the point of view of the user to look up from the bottom of the scene, and use orthogonal projection to render the scene, then the Z-Buffer would contain the distance of each pixel from the floor, i.e. the Z-Buffer contains the height of each rod on the physical surface that we desire to render. Given a standard 3D graphical model, we need only specify the point of view of the user as being from the bottom looking up in an orthogonal projection, and read out the values from the Z-buffer, and then raise each rod in the PRE to the appropriate height from the floor.

An important outcome of this observation is that it allows us to use hardware acceleration available in conventional GPUs to calculate rod position, as well as standard APIs like OpenGL [8] for controlling this calculation.

6.2. Adapting to Physical Limitations of the Rods

Each rod is a physical entity and subject to physical laws of inertia. While the Z-Buffer approach can help us calculate the final position of the rod, it does not account for how the rod reaches that final position from its current position. The rod itself may have a response curve for its position that looks like Figure 9. How can we integrate these physical limitations into the motion of each rod?

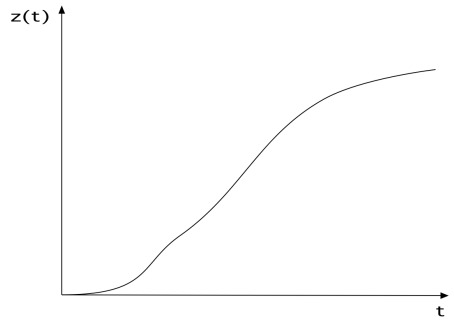

Figure 9. One possible function for describing position of a rod versus time.

Our solution is to utilize the programmable hardware acceleration available in today's graphics cards. Modern consumer GPUs have programable pixel (fragment) shaders, allowing us to perform per pixel calculations [9]. We would perform quantization of the function from Figure 9 into a 1D texture, and then pass to the fragment shader that texture and a uniform (constant for frame) variable representing the interpolation step that we want the fragment shader to perform. The fragment shader would then perform the necessary interpolation, store the intermediate position in the color buffer or intermediate texture, and actuate the pixel rod with the appropriate displacement.

6.3. Detailed Rendering Challenges

We present in this section special cases that illustrate some of the more detailed challenges we will encounter in adapting existing graphics pipelines to realize physically rendered environments. As the examples in the previous

subsections show, our intent is to leverage as much as practicable existing graphics pipelines in both software and hardware to realize physical rendering of surfaces and objects. In the following, we use OpenGL-based image generation for our examples, but the same techniques and generation apply to the other graphic APIs (e.g. DirectX [14]).

6.3.1. Fixed OpenGL Pipeline Rendering Only the Ground Plane The simplest case is the situation in which a graphics program is rendering a single surface or ground plane with no objects resting on that surface (the human user stands in the PRE but is not rendered). Such a scenario could be easily adapted to the Holodeck environment by using the same OpenGL commands to render an environment both on the graphical system and the physical Holodeck system. The user would be in the same position for both environments, though the clipping planes would differ. Clipping planes for physical rendering need to be set to the boundaries of the Holodeck environment or room. However, the perspective rendered from the user's view, e.g. the user is wearing a head-mounted display, would be given as that between the near and the far clipping plane, as in Figure 10. If any optimizations (beyond setting of the near clipping plane to be in front of the user) have been made in the graphics program to discard geometry between the viewer's position and the near clipping plane, they need to be removed in order for the Holodeck environment to correctly render the full physical scene. For example, if the user is looking forward, but there is a chair behind the user, then a user should be able to back into and bump the rendered chair, even though it is not shown in the user's helmet view, due to the user's orientation. In this case, the physical rendering should capture and draw all surfaces within the physical clipping planes, even though they may not appear in the graphical view. In this example, the near clipping plane would be the floor. In general, the clipping planes should be reconfigured so the whole area around the user is rendered. If there are no optimizations made but the near clipping plane is in front of the user, it should be moved (in orthogonal projection only) to encompass the point on which the user is standing.

An issue that could occur in this situation is that the physical height of the user who is in the Holodeck might be different than the "assumed height" of the virtual user for which a graphical image is rendered. For example, the image might be generated for a user that is 5 feet high, while the physical user is 6 feet tall, causing what we dub a "Gulliver effect", where ground features under the user's feet are either smaller or larger in the physically rendered Holodeck than what the user sees on the helmet screen. We can address this issue of matching the physical and graphical scales in two ways: the graphical scale can be held fixed,

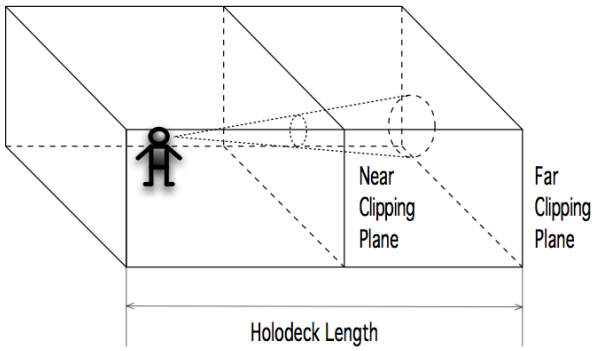

Figure 10. As the Holodeck PRE needs to render the complete physical area around the user, the adapted graphics program should not clip the immediate area around the user even if it is not visible to the user.

and the physical scale can be adjusted to match the user's height, e.g. if the graphical world is rendered from the perspective of a 6 foot human, then the 5 foot user who is peering over the top of a 6 foot high graphical wall should have the wall physically rendered to 5 feet to receive the correct tactile feedback; alternatively, the graphical scale can be adjusted to match the actual height of the user, while keeping the physical scale constant, e.g. for a 5 foot tall user, the graphical view is adjusted so that it is rendered from the 5 foot tall perspective, and a wall that is 6 feet high in the physical world will remain as 6 feet high, regardless of the height of the user. Future research will determine which of these is the best approach.

6.3.2. Fixed OpenGL Pipeline Complex scenes with objects resting on surfaces pose a more difficult challenge in terms of adapting OpenGL commands to physical rendering of surfaces. In the fixed OpenGL pipeline, final rendering depends only on the rendering APIs that are called for the program, and it is known how the picture would look based only on those API calls (as opposed to how the picture would be transformed based on programmable shaders [9]). One problem that exists in this environment is that some objects residing on the ground plane, e.g. the ball in Figure 11, could not be easily rendered in the Holodeck environment.

This problem is effectively one of distinguishing the ground plane from the objects on the ground plane. For this to be done, we need the help of the programmer, who must annotate the OpenGL program so that we know which geometry parts are the ground plane only, and which ones are not.

We believe the modification needed to help us distin-

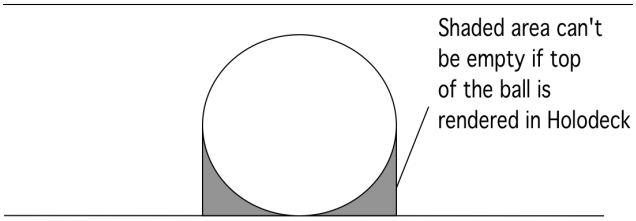

Figure 11. Ball on the ground - this image could not be easily rendered in the Holodeck, as the top surface of the ball could be rendered, but not also the gap between the ball and the ground.

guish between the ground plane and objects resting on its surface is simple, and effectively is just the addition of two API commands that would identify the beginning and the end of the code section within which all rendered geometry are included as part of the ground plane. Only these OpenGL commands will be physically rendered into the Holodeck's ground geometry. An example would resemble the following:

```
... Draw elements that are not
    part of the ground plane ...
DrawBall();

// All subsequent drawing appears both
// on screen and in physical Holodeck
hglGroundBegin();
... all geometric objects that are part
    of the Holodeck's ground plane ...
hglGroundEnd();
```

6.3.3. Bump Mapping, Displacement Mapping and the OpenGL Programmable Pipeline Bump mapping [26] is a technique of simulating surface detail by varying the surface normal on a per pixel basis to create the impression of small 3D details on the surface of an object. Displacement mapping [25] is displacement of the actual geometric points on the surface of the object, mostly done to simulate fine detail.

Both of those techniques share a problem that the actual geometry of the surface passed to the GPU does not contain the fine detail. As a result, if the ground is bump mapped, it would look rough on the graphical screen, but smooth when physically rendered in the Holodeck, given the techniques applied so far.

Fortunately, as long as the depth map or Z buffer at the point is correct, bumps would be correctly rendered. A similar observation applies to displacement mapping. So as long as we use a *programmable* fragment shader to vary the depth buffer at a pixel to simulate bumps, the Holodeck environment could render bump and displacement mapping surfaces.

From that prospective, the Holodeck integrates well with programable shaders. The main problem that exists with programable shaders is that if the depth buffer at a physical pixel does not correspond to the depth value of the graphically rendered pixel (as might be the case in some parallax mapping [25] implementations), the Holodeck would not render the correct surface. The solution then is to modify the shader to synchronize depth values at a physical pixel with the value corresponding to the graphically rendered pixel.

6.3.4. How to Perform Effective API Call Interception One solution for realizing effective API call interception for the previously described techniques is to modify the graphics driver to issue rendering calls both to the graphic card and the physical Holodeck system. The problem with this approach is that it requires access to the source code of the driver, which is typically not available. Even worse, GPU hardware is typically not documented. That means that it is in practice often impossible to write a driver from scratch.

That leaves two possibilities:

1. Wrap OpenGL calls in a custom library that would issue required double calls. This approach requires extensive modification of the rendering code in programs that are using OpenGL.
2. Recognize that this interception is effectively aspect-oriented [10] and employ a system like AspectC++ [11] to perform interception.

We prefer aspect-based interception. The code for such an aspect is shown below:

```
around any OpenGL call
        if Holodeck_affected
                issue Holodeck rendering

        proceed
```

6.4. Slipperiness Simulation

Our discussion thus far has related to how to control displacement of the rod. When it comes to controlling slipperiness of terrain, typically no information presented to the OpenGL pipeline for the purpose of scene rendering could help us to determine the slipperiness of the surface.

Fortunately, we can encode slipperiness of the surface in the texture, and then use the fragment shader to vary slipperiness of each ball on top of each rod in accordance with the value of "slip texture" for that pixel. Implementation of this is fairly straightforward in the fragment shader.

Although the fragment shader portion of the slipperiness calculation would not be scene specific (so the same shader could be used for all user programs), slipperiness of

terrain is obviously scene specific. As a result, a user program would need to be modified to pass additional texture for each surface on which a user is standing. This is typically not a problem unless the number of already used textures is at the limit of the texturing capabilities of the card.

Our approach is to require the program to be modified to provide surface slipperiness information. That information could be passed in the form of the coefficients of frictions in a texture map equivalent, that then would be used by the Holodeck's fragment shader. In the long run, material libraries in the modeling packages artists are using to specify look of the objects [15] (and which typically include multiple texture maps) could be extended to include a slipperiness map of the surface, too. This is a rich area for future work, as it is unclear whether the coefficient of friction of the material correctly captures all characteristics that are really perceived as slipperiness, i.e. for ice it is likely that the coefficient of friction is an accurate representation of the perceived *slipperiness* of the terrain, while for gravel the perceived slipperiness is likely to be higher than the coefficient of friction of stone and sand would indicate.

6.5. Integration with Rendering Engines

Although the Holodeck integrates well with the multiple parts of the rendering pipeline , its integration with the complete rendering engine [24] might be more involved, as multipass rendering techniques, e.g. shadow maps [25], or multipass rendering using shaders [24] will not reproduce realistic Holodeck pictures even if all OpenGL calls are instrumented. This is because the results of some passes, such as for shadow map generation [25], are not directly drawn in the scene but are used for intermediate steps in rendering the final scene.

To resolve this problem, it is necessary to modify the rendering engine so that calls that are not intended to affect the screen buffer are not rendered in the Holodeck. There are three approaches to achieve this:

1. Change Holodeck interception so that instead of the interception of the OpenGL calls in real time, final values of the OpenGL depth buffer when all rendering is done are read. If the depth buffer is a realistic representation of the surface of the environment, then this technique would be all that is needed. In full rendering engines, this is rarely the case in practice, e.g. bump mapping doesn't update the depth buffer to simulate real slipperiness of the surface.
2. Exploit the possibility that passes that aren't to be rendered might be fairly easy to identify, and mark each pass as "Holodeck compatible" or not, similar to the earlier approach of annotating the graphics calls to identify which applied to the ground plane. In this case, we would intercept OpenGL calls only if they apply to the Holodeck.
3. While the previous two approaches involve modest changes to the pipeline, they are not comprehensive. For the best results, the rendering engine itself may need to be changed to account for the Holodeck PRE.

In addition to the changes in the graphic engine necessary to make it compatible with the Holodeck, it is likely that the Holodeck would impact how code of the rendering engine is written. In particular, rendering calls that are GPU intensive are often interleaved with rendering calls that are CPU intensive [24]. The PRE-based Holodeck would introduce one additional area where interleaving could be exploited for enhancing total system performance. How much advantage is to be gained in interleaving the Holodeck's physical rendering code with graphical rendering, and what are the best ways to do it are areas of future research.

7. HoloSim - Simulating the PRE-based Holodeck

It is our intention to construct a simulator as a first step towards realizing this PRE system. This HoloSim approach is not only a cost saving measure, but also the simulator is likely to be able to address many of the open research questions before the expensive task of building the physical electro-mechanical system.

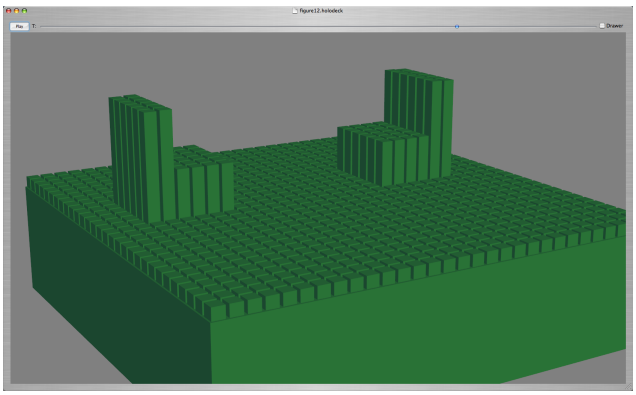

Figure 12. A screen capture of the HoloSim rendering a surface with physical rod-like pixels, in this case two thrones facing each other.

The HoloSim is intended to be able to answer many of the following questions:

1. How would the physical system look like without a helmet? Would running the simulator in a CAVE [16]

provide insights on how a completed physical system would look like?

2. The simulator would be enough to demonstrate feasibility of the proposed Z buffer and fragment shader based control approach.
3. The simulator would provide additional insight as to the best software control algorithms for rod movement. Should we choose algorithms that allow rods to reach their final position the fastest? Or algorithms that would minimize discontinuity between roads, to minimize sharp edges in simulator?
4. Fail-safeness approaches could be partially tested in simulator environment, as well as the impact on physical safety.
5. Finally, the majority of the simulator code could be used as a basis for the software control module of the physical system once it is realized.

Details of the design and implementation of the simulator are beyond the scope of this work. Figure 12 illustrates output from our current HoloSim.

8. Summary

This paper has presented an ambitious original concept for realizing the vision of a Holodeck through physically rendered environments (PREs), i.e. computer controlled actuation of a grid of physical pixel-like rods wherein the displacement of each rod can be increased or decreased. The result is the ability to render arbitrary physical surfaces and terrains. In addition, such a PRE would be able to support a human user, who will be able to walk over these surfaces, and roll and kneel as well. The aim is to further achieve an effect whereby subtle displacements would re-center a user imperceptibly, to give the perception of infinite distance. The coefficient of friction, or slipperiness, of the tip of each of rod would be computer-controlled, to give the user different impressions of slipperiness.

We have presented how we believe that software control of a PRE-based Holodeck can be achieved using the technology of today. We have described methods for adapting existing graphics pipelines, such as OpenGL software and GPU accelerators, so that 3D graphical models of environments can be used to render physical surfaces.

We proposed HoloSim as a simulation environment that mimics the rendering actions of a true PRE. HoloSim gives us the opportunity to evaluate the most promising rendering approaches prior to building the full physical system. HoloSim will incorporate the effects of physical limitations of the rods, e.g. inertia. We can then study the effect of different approaches to achieve the most realistic rendering. We believe the HoloSim, suitably modified, will grow into the software front end for controlling the hardware back end.

Finally, this is still early work that is presenting an ambitious concept. We have raised some of the technical challenges, but they are not meant to be exhaustive. For example, we have not addressed issues of reliability, safety, or human perception of motion. We have only briefly introduced some of the electromechanical issues. Graphical issues such as physical aliasing also were omitted. Our hope is that this white paper has raised interest in the idea of computer controllable rendering of physical terrains and environments, and spurs new research and development towards realizing the vision of the Holodeck.

References

[1] Intel Corporation: “Dynamic Physical Rendering”, available at http://www.intel.com/research/dpr.htm, visited on September 1st, 2007.

[2] S. C. Goldstein, J. D. Campbell, T. C. Mowry: “Programmable Matter”, Invisible Computing, June 2005, pp. 99-101.

[3] R. P. Darken, W. R. Cockayne, D Carmein: “The Omni-Directional Treadmill: A Locomotion Device for Virtual Worlds”, ACM Symposium on User Interface Software and Technology, 1997, pp. 213-221

[4] H. Iwata, H. Yano, F. Nakaizumi: “Gait Master: A Versatile Locomotion Interface for Uneven Virtual Terrain”, Proceedings of the Virtual Reality 2001 Conference (VR’01), 2001, pp 131, ISBN:0-7695-0948-7

[5] H. Yano, K. Kasai, H. Saito, H. Iwata, “Sharing Sense of Walking With Locomotion Interfaces”, International Journal Of HumanComputer Interaction, 17(4), 447462.

[6] J.M. Hollerbach, Y. Xu, R. Christensen, S.C. Jacobsen: “Design specifications for the second generation Sarcos Treadport locomotion interface”, Haptics Symposium, Proc. ASME Dynamic Systems and Control Division, DSC-Vol. 69-2, Orlando, Nov. 5-10, 2000, pp. 1293-1298.

[7] O. Bimber, R. Raskar: “Spatial Augmented Reality: Merging Real and Virtual Worlds”, A K Peters, Ltd. (July 31, 2005)

[8] D. Shreiner, M. Woo, J. Neider, T. Davis: “OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R), Version 2 (5th Edition)”, Addison-Wesley Professional; 5 edition, August 1, 2005, ISBN:978-0321335739

[9] R.J. Rost: “OpenGL(R) Shading Language (2nd Edition)”, Addison-Wesley Professional; 2 edition, January 25, 2006, ISBN: 978-0321334893

[10] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, J. Irwin, “Aspect-Oriented Programming, Proceedings of the European Conference on Object-Oriented Programming, 1997, vol.1241, pp.220242.

[11] O. Spinczyk, A. Gal, W. Schrder-Preikschat, “AspectC++: An Aspect-Oriented Extension to C++”, Proceedings ofthe 40th International Conference on Technology of Object-

Oriented Languages and Systems (TOOLS Pacific 2002), Sydney, Australia, 2002

[12] J. M. Rolfe (Editor), K. J. Staples (Editor): "Flight Simulation (Cambridge Aerospace Series)", Cambridge University Press; Reprint edition (May 27, 1988), ISBN: 978-0521357517

[13] Wikipedia entry on Wii, available at http://en.wikipedia.org/wiki/Wii, visited on August 23rd, 2007.

[14] Microsoft Corportation: DirectX Resource Center, available at http://msdn2.microsoft.com/en-us/xna/aa937781.aspx, visited on August 23rd, 2007.

[15] Blender Material Library, available at http://www.blender.org/download/resources/#c2511, visited on August 23rd, 2007.

[16] Wikipedia entry on CAVE, available at http://en.wikipedia.org/wiki/CAVE, visited on August 23rd, 2007.

[17] List of commercially available Braille terminals, available at http://www.tiresias.org/equipment/eb7.htm, visited on February 25th, 2007.

[18] Wikipedia entry on Stagecraft, available at http://en.wikipedia.org/wiki/Stagecraft, visited on February 25th, 2007.

[19] Wikipedia entry on Walt Disney Imagineering, available at http://en.wikipedia.org/wiki/Walt_Disney_Imagineering, Visited on February 25th, 2007.

[20] Thomas H. Massie and J. K. Salisbury: "The PHANTOM Haptic Interface: A Device for Probing Virtual Objects", Proceedings of the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Chicago, IL, Nov. 1994.

[21] E. Malone, H. Lipson, "Freeform Fabrication of Complete Devices: Compact Manufacturing for Human and Robotic Exploration", AIAA Space 2006, San Jose, CA, 19-21 Sept 2006, AIAA 2006-7406.

[22] "What is Solid Freeform Fabrication?", available at http://www.msoe.edu/reu/ssf.shtml, visited on May 5th, 2007.

[23] J.J. Beaman, John W. Barlow, D.L. Bourell, R.H. Crawford, H.L. Marcus, K.P. McAlea, "Solid Freeform Fabrication: A New Direction in Manufacturing", Springer; 1 edition (December 31, 1996), ISBN: 978-0792398349.

[24] W. Engel (editor): "ShaderX3: Advanced Rendering with DirectX and OpenGL", Charles River Media; 1 edition (November 2004), ISBN: 978-1584503576, pp. 499-519.

[25] A. Watt, F. Policarpo: "Advanced Game Development with Programmable Graphics Hardware", A K Peters, Ltd. (August 1, 2005), ISBN: 978-1568812403.

[26] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes: "Computer Graphics: Principles and Practice in C", Addison-Wesley Professional; 2 edition (August 4, 1995), ISBN: 978-0201848403.

[27] Reprinted with permission from http://www.pittsburgh.intel-research.net/dprweb/.

[28] Reprinted with permission from http://intron.kz.tsukuba.ac.jp/gaitmaster/gaitmaster2.jpg.