SCALABLE SOFTWARE CONTROL OF MILLION-ELEMENT CYBER-PHYSICAL SYSTEMS USING A GRAPHICS PROCESSING UNIT

by
VELJKO KRUNIC
B.S., University of Colorado, 2000
M.S., University of Colorado, 2003

A thesis submitted to the Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirement for the degree of
Doctor of Philosophy
Department of Computer Science
2010

This thesis entitled:

Scalable Software Control of Million-Element Cyber-Physical Systems
Using a Graphics Processing Unit

written by Veljko Krunic

has been approved for the Department of Computer Science

____
Associate Professor Richard Han, Advisor

____
Professor Clayton H. Lewis

____
Assistant Professor Adjunct Willem Schreuder

____
Professor Yung-Cheng Lee

____
Randolph Ware, PhD

Date___________________

The final copy of this thesis has been examined by the signatories, and we
Find that both the content and the form meet acceptable presentation standards
Of scholarly work in the above mentioned discipline.

Krunic, Veljko (Ph.D., Computer Science)

Scalable Software Control of Million-Element Cyber-Physical Systems Using a Graphics Processing Unit

Thesis directed by Associate Professor Richard Han

Abstract

Cyber-Physical Systems consisting of hundreds of thousands of elements are emerging, with even bigger systems likely to emerge in the immediate future. However, in order for emerging and reasonably anticipated systems to be practical, the software control of million-element Cyber-Physical Systems needs to be addressed.

This PhD thesis describes the software control algorithms necessary for the realization of million-element Cyber-Physical Systems. This work will show that Graphics Processing Unit (GPU) based control of such Cyber-Physical Systems provides significant benefits, both in the form of fast control of large numbers of elements, as well as in terms of providing a viable and scalable option by using inexpensive, off-the-shelf hardware. GPU control will be shown to be particularly well suited for the combination of the virtual environment with the manipulation of the physical shape of the environment in which the user resides.

The main contributions of this PhD thesis consist of novel algorithms that utilize existing off-the-shelf GPUs to control the Constrained Motion Cyber-Physical Systems comprised of multi million element systems, and demonstrate the feasibility and scalability of such control algorithms. It will be shown how both control and coordination of the elements can be achieved, while at the same time accounting for the physical limitations of the Cyber-Physical System elements. The approach presented here results in the ability to control the position of the actuation elements in Cyber-Physical Systems, as well as additional physical attributes of the system such as temperature, perceived elasticity of the actuating elements, slipperiness of the ground in large scale systems, etc. We describe how to further extend our approach to deal with existing Cyber-Physical Systems like catoms/Claytronics [1], [2], CirculaFloor [3] and MEMS-based tactile devices, as well as describe an approach to addressing the physical safety of the user in large scale Cyber-Physical Systems.

Za moju baku, koja vise nije sa nama.

I za ostatak moje porodice, svugde u Svetu.

Vasoj ljubavi i podrsci dugujem sve svoje uspehe.

Dedicated to my grandma who is no longer with us.

And to the rest of my family all over the world.

Your love and support made my success possible.

Acknowledgments

I was lucky to benefit from the expertise, help, insight and patience of many people while I worked on this PhD thesis. I would like to thank the following people, without whom this work wouldn't be possible in its present form:

1. I would like to thank my wife Marija Krunic for love and understanding during the long process of the obtaining my PhD and the much more important process of living and enjoying life.

2. I am grateful for a lot of invaluable feedback from Professor Clayton Lewis. I found that in multiple situations where I needed help in scoping and understanding broader implications of my doctoral work, his suggestions, help and experience helped me find the right track. He suggested that further uses of the software control algorithms originally developed for PRE should be investigated in other areas, and that helped me expand the scope of this work from its initial PRE/Holodeck focus to what it is today. Furthermore, he provided many invaluable suggestions in the areas of the HCI and psychophysics research, that relate to the assistive uses of this technology, and he also provided many other suggestions about the PhD studies process. Finally, he challenged me to consider further work in the area once this thesis has been completed.

3. Professor Willem Schreuder provided help and expertise in many areas of this work, and I strongly benefited from his insight, experience and knowledge. He was the first one to point out similarities between the approach used for the

calculation of the moxel position and shadow buffer algorithm. He suggested that multiple output channels (moxel displacement, heat, etc) should be investigated in relation to PRE and proposed use of the OpenGL render buffer. He also advised me to consider feedback from the physical systems in my algorithms as well as to investigate mechanism(s) for directly accessing data that are already on the graphics card with a combination of OpenCL and OpenGL. He further suggested that I should consider overlapping surfaces when combining catoms and moxels, proposed limiting the surface of the rendering to only the area immediately around the user in combining moxels and CirculaFloor, proposed that “halfway up” failed moxels could be addressed by the raising of other moxels and proposed “relaxing” moxels below the user in the case that user is falling as a safety measure. Furthermore, he provided many suggestions related to both the style of this document as well as the substance of work described, and he challenged me to think about potential future research areas. I am grateful for the experience and insight he shared with me during multiple brainstorming sessions we had. His help was crucial in identifying areas for possible improvement and shaping this work to its current form.

4. Professor Yung-Cheng Lee acted as a liaison in working with his students involved with building a physical assistive device [4]. His help was crucial in steering me toward MEMS devices and applying my work to this area. He had many suggestions in the area of the physical characteristics of MEMS devices. Furthermore, he provided the insight that the ability to perceive the texture of

the image will be very useful even for people who are not visually disabled (e.g. ability to touch the display and feel the texture of the physical object on the image) and pointed me to TeslaTouch [5]. Finally, he suggested that I should consider adopting my algorithms to account for power consumption of the mechanical system.

5. Professor Steven Oullette helped me on short notice with the review and interpretation of the data collected during the performance test. That helped me understand the implications of the data, not only from the perspective of the benchmark, but also from the perspective of the statistical process control. He helped me understand the nature of the data (including the fact that we are dealing with more than one distribution) and he also pointed out what type of statistics are appropriate to calculate from the data we have, as well as what type of graphs will be the most appropriate representation for those data. Furthermore, his looking at the data from the perspective of the statistical process control helped me understand why these are more appropriate forms of data representation to use. I found his help invaluable and appreciate his time and willingness to help me on the short notice given.

6. Along with my advisor, Professor Richard Han, I had published original papers describing PRE [6], [7]. That work provided an outline of the corresponding algorithms that are expanded on in this document. He provided many suggestions in relation to the organization and presentation of material in this document, suggested writing an overview of the problem domain and

contributions that is now in Chapter 4. He also pointed to median filter being related to one of the steps in how this work addresses moxels stuck in down position. Finally he provided help in grammatical and organizational correction of Abstract and Chapter 1 of this document and many other suggestions related to the style of this document.

7. Randolph Ware, PhD provided many suggestions relating to the practical applications of the PRE/Holodeck environment and challenged me to better understand broader implications of the use cases that I investigate in this PhD thesis. He pointed me towards carbon nanotubes as an area in which my work could be applicable and provided other suggestions relating to this document.

8. I would like to thank to Professor Kevin Shaub for consultation regarding Shapiro-Wilk exponential test.

9. Professor Hiroo Iwata pointed me to the CirculaFloor Project after learning about PRE/Holodeck and responded to questions regarding current use of the vertical dimension in CirculaFloor.

10. Professor Dale Lawrence proposed the term “moxel” after hearing about our PRE/Holodeck idea.

Contents

Tables

Figures

Chapter 1 - Introduction

We are moving toward a world that offers an artificially generated, dynamic environment in which we will be able to control the physical shape of such environment in real-time. Our vision for Cyber-Physical Systems is that it will be possible to combine computer graphics and immersive virtual environments with morphing physical environments that deform in concert with the virtual environment. That would enable the user to produce a movement in the computer generated environment and perceive physical objects through the morphing physical environment. One of the major challenges in achieving that vision is software control of the millions of elements comprising larger Cyber-Physical Systems.

Systems of a similar nature have been recently proposed, and include systems such as MEMS [8] on the microscale, as well as Dynamic Physical Rendering (DPR) or Catoms [1] and PRE [7], [6] on the larger scale. Figure 1 shows an example of a MEMS system, and Figure 2 depicts an example of a Catom [1] based system.

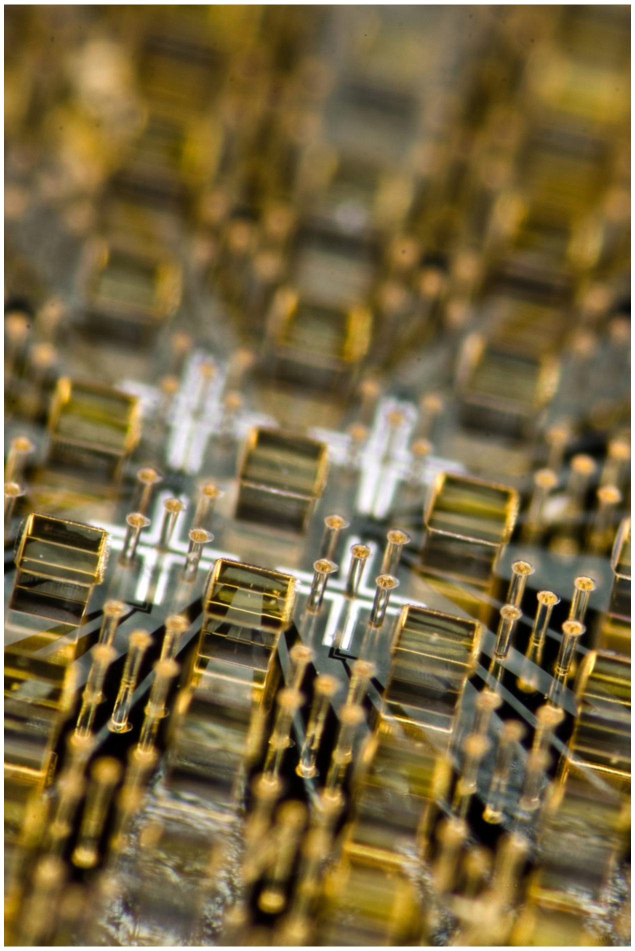

Figure 1. A Microelectromechanical System (MEMS) based force sensors [9].

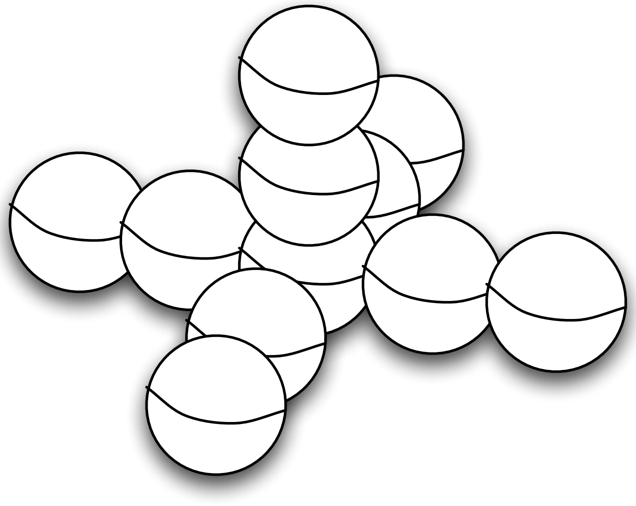

Figure 2. Catoms Forming Shape (the author's illustration).

One particularly interesting class of Cyber-Physical Systems is the Constrained Motion Physically Rendered Environment characterized by elements limited to up and down, one dimensional movement. The algorithms that will be presented in this thesis are particularly well suited for this class of Constrained Motion Cyber-Physical Systems.

The first example of the aforementioned system is the author's earlier work integrating virtual environments with the ability to perform physical deformation of the environment in which the user is standing [6], [7]. This work proposed a Physically Rendered Environment (PRE) or "Holodeck"[1], a system capable of showing realistic impressions of a combination of the physical and virtual environment. That system could be combined with additional systems like CirculaFloor[2] [10] or treadmills [11], [12] in order to provide the user with an environment featuring the illusion of infinite physical space .

Figure 3 shows the opening ceremony at the Beijing Olympics in which a human powered PRE-like system was used:

[1]The Holodeck name was chosen based on the science fiction television show"Star Trek the Next Generation". Although such a system is obviously not practical today, the ability to combine virtual environments with physical environment deformation could be considered related and was why we had chosen that name in our earlier work [6], [7].

[2] The author would like to thank to Professor Hiroo Iwata for pointing me to the CirculaFloor after learning about our PRE work.

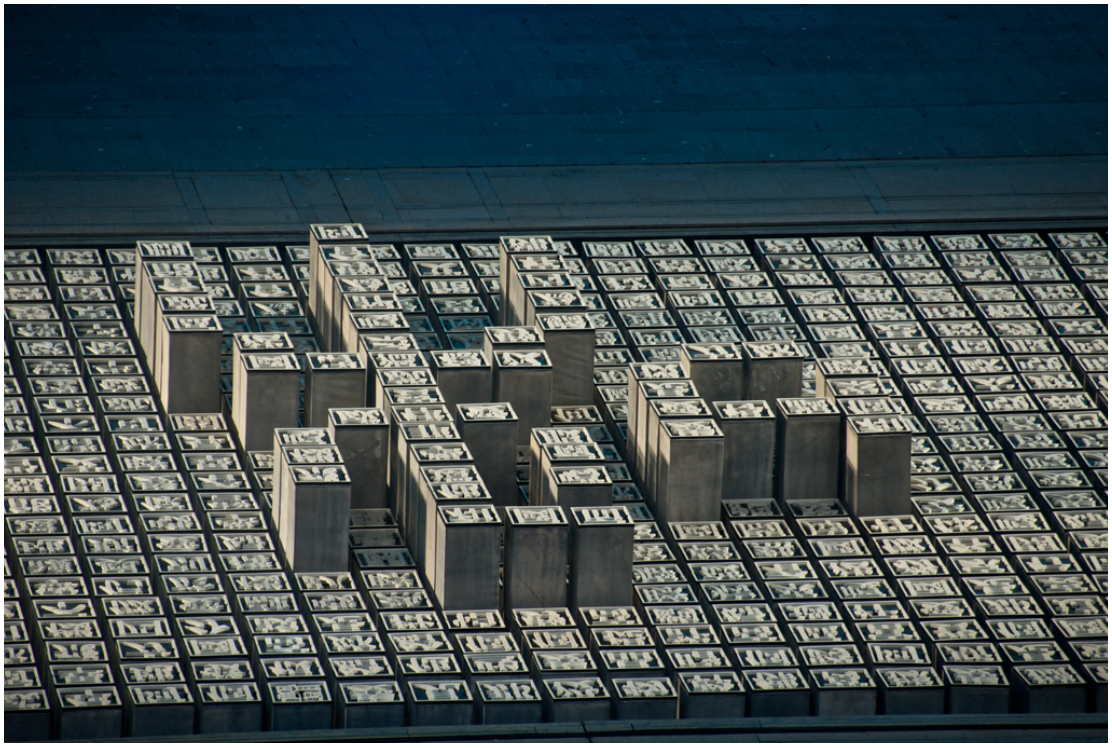

Figure 3. Opening ceremony at Beijing Olympics [13].

The second example of such Constrained Motion Cyber-Physical Systems consists of portable systems. Figure 4 shows a PinPoint toy (see e.g. [14], a vendor) that consists of a set of needles that are free to move only in the up/down direction. If a user presses his hand into the needles on one side of the toy, the needles raise on the opposite side in the approximate 3D shape of the hand, as shown in the figure below. This device has inspired multiple systems including Digital Clay [15], [16], [17] and the author's earlier work [7], [6].

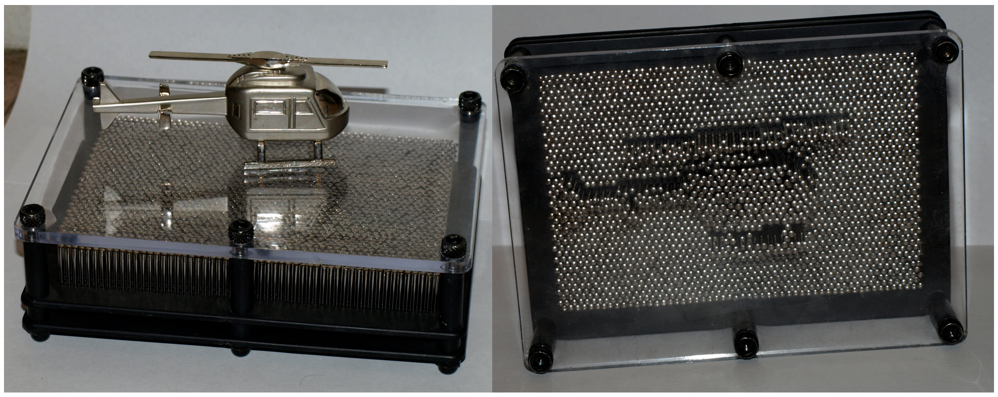

Figure 4. PinPoint Toy {Pin Art/Pin Point Toy} showing a picture of a toy chopper.

Another related example consists of tactile and MEMS based assistive devices [18], [19] for people with visual disabilities, allowing them to perceive text as well as computer generated graphics. Furthermore, applications of these devices include sensory substitution, which is advantageous for both visually impaired users and users with normal vision [20].

A further example of the portable sized system is the BrailleEye [4], [21] portable assistive device system for people with visual disabilities. This system is currently being developed in the Mechanical Engineering Department at CU Boulder.

Finally, and to highlight the importance of the ability to scale software control to massive numbers of elements in Cyber-Physical Systems, recent work shows that carbon nanotubes could be constructed in, what is essentially a Constrained Motion, 1D system[3] [22].

[3] The author would like to thank Randolph Ware, PhD for pointing out carbon nanotubes and how they could potentially be related to this project.

As mentioned in our previous work [7], [6], we call each of the actuating elements in these classes of Constrained Motion systems "moxels", [4] short for moving pixels[5]. By augmenting this "moxel based" motion with computer controlled actuation of the moxels in real time, a system becomes capable of rendering different environmental elements such as dynamic surfaces, terrains, and even forms of motion. Within the space of a designated room, a deformation of the ground, walls, and ceiling, as created by moxels, would simultaneously create an entire 3D environment with the user being able to stand and move.

The previous systems offer a large potential for significant research advances, as well as a large potential to significantly impact the world around us. Portable MEMS based Constrained Motion PREs like BrailleEye [4], [21] combined with the computer control of deformation will allow us to present tactile rendering of images and videos (including real-time videos of the surrounding area) to the users. This enables significant assistance to visually impaired users at a very reasonable price[6].

Furthermore, combining the Holodeck, PRE (in possible additional combination with the CirculaFloor [3]) with complementary technology such as virtual or augmented reality, (e.g. immersive graphics visualization via projection and/or head-mounted

[4] The author would like to thank to Professor Dale Lawrence for coining the term *moxel* for a moving pixel after hearing our idea.

[5] Note that there is no consensus in the community on what to call these moving elements - we used the term *moxel* in our previous papers ([6], [7] and continue to use that term in this paper. Other terms proposed include *RGBH* (H standing for height in the moving element that could present RGB color) as proposed in [23], *taxel* as in [24], [25], *haptcel* [26]. Another term for the concept of movable elements is *tactile pin arrays, a* term used by [27].

[6] Current budget for BrailleEye project, including prototype building, is below $1,000. The final device, when mass produced, is likely to be substantially cheaper.

helmet displays) [28], would further enhance the user experience of “being there” and is believed to be the next step in the evolution of computer graphics. The ability of the computer to control physical deformation of the terrain was one of the early visions of computer graphics pioneers [29].

1.1 Our Contribution

With the clear importance of the previous hardware technologies shown, the key question remains: do we currently have the problem of software control of Cyber-Physical Systems resolved?

To answer that question, we must first start by asking: how many elements do these systems have? To get some feeling of the scale of the system, let's consider a couple of examples:

1. Suppose that a moxel in a PRE is 1cm by 1cm and that the size of the room is 10m by 10m meaning that we would need 1M moxels just for the floor.

2. Suppose that MEMS element is 0.5mm by 0.5mm in size. The entire Kindle DX device is about 264mm by 182mm (10.4 inches by 7.2 inches) [30]. If that device surface is modeled as a MEMS system with elements on the scale of 0.5mm, the system would require 192,121 elements[7]. Note that 0.5mm was chosen as it is a good a resolution allowing a user the ability to discriminate in a tactile manner between two disjoint points [31], [32], and that for presenting different texture feelings of the surface, we might need significantly higher moxel densities[8].

[7] Due to the approximate nature of the calculation, in other parts of this thesis and the fact that Kindle DX is just one portable device (with the best format for the portable devices still being open questions) when referring to the number of moxels needed to cover the surface of Kindle DX, we would use an approximation of 200,000 moxels of the size 0.5 mm x 0.5 mm per moxel and 50,000 moxels of the size 1 mm x 1 mm per moxel.

[8] Humans can differentiate surface period changes on the order of as little as 20-40 μm [28], meaning that we might need to control much higher moxel densities even for MEMS based systems.

These examples clearly show that to provide software control of PRE and MEMS devices of such reasonable sizes, we would need the ability to control system on a scale of hundreds of thousands or even millions of elements (e.g. the Digital Clay prototype [33] demonstrates a physical system that, although built on the scale of 5x5 uses technologies, would allow for building on a 1,000x1,000 scale of linear actuators). The ability to control even a significantly larger number of elements is beneficial, as in the previous discussion we didn't account for the frequency of the image change nor for the fact that even smaller element size/larger surface areas might be beneficial. So the question of the best approach to the software control is clearly non-trivial.

This PhD thesis will show that software control of million-element moxel based Cyber-Physical Systems could be achieved using commodity hardware and that it could easily be scaled to size needed by practical hardware systems. That capability is important as there is little point in building these types of systems unless we have a way to provide software to control them.

For that software control to be implemented, it is important to notice that the number of moxels involved approaches and in some cases exceeds the number of pixels on the computer screen. In addition, per-moxel calculations are required as we not only need to calculate the position of the moxel, but account for its physical limitations in order to control it properly.

Trends in computer graphics have clearly shown that when such numbers of elements are involved, GPUs offer a significant speed advantage compared to CPUs as long as the control problem can be formulated in a way that utilizes an existing GPU's hardware rendering pipeline [34], [35], [36], [37].

In fact, trends in the last two decades and especially in the last few years clearly show that for problems that are well suited for GPUs, the GPU provides a significant performance advantage over the CPU, and that GPU based computing is challenging CPU based solutions in a large number of areas:

1. GPUs have been used for general purpose computing in systems including physical simulations (like boiling, convection, dendritic ice crystal growth, fluid flow problems, finite differences, spring mass dynamics, cloth simulation), signal and image processing, image segmentation, computer vision, Constructive Solid Geometry, databases and data mining (as surveyed by [38]).
2. The current state of heterogenous computing is surveyed in [39]. According to that survey, GPUs provide significantly higher computation and memory bandwidth performance advantages over CPUs. For example, single precision peak performance of AMD's FireStream 9270 is 1200 GFLOPS, NVIDIA's Tesla C1060's is 936 GFLOPS and Intel's Core i7-965 Quad Extreme's single point peak performance is only 102.4 GFLOPS. Furthermore, GPUs are leading ahead of CPUs (and other heterogeneous computing solutions like FPGA and CBEA) in floating point performance, memory access speed (counting access to accelerator memory), GFLOPS per watt, and MFLOPS per dollar [39].

3. The scientific community generally believes in the strength of GPUs for general purpose computing. Even the examples of the work that could be considered somewhat dissenting from the belief that GPUs are much faster than CPU still conclude that for throughput computational kernels NVIDIA GTX280 still beats Intel Core i7 960 2.5 times on average. The authors reached this conclusion after performing extensive optimization of the computing kernels on both CPU and GPU [40][9].

4. GPUs are well suited for applications operating on data streams but they are suffering from expensive synchronization [39]. Compared to other heterogenous computing solutions (FPGA, CBEA), GPUs are an attractive solution for computing with significant advantages in highly parallel problems[10].

With all of the above taken into consideration, GPU based implementation is a natural choice for addressing our highly parallelizable control problem. It will be shown in this thesis that we can use existing GPU hardware without any need to modify it, and in a way that easily scales to the control of a million moxels per second.

The main contributions of this PhD are:

[9] It is worth noting that the team that published [40] has authors who are all associated with Intel Throughput Computing Lab and Intel Architecture Group of Intel Corporation, and that based on that, authors should be considered experts on how to perform CPU based optimizations. Tests performed included both kernels that are better suited for the CPU than GPU and vice versa. Although the conclusion was that “This puts CPU and GPU roughly in the same performance ballpark for throughput computing”, the fact that even dissenting work is acknowledging the advantage of GPUs still shows that GPU has matured as a platform for general purpose computing. Finally, note that moxel calculation is much closer to a rendering problem and that as such, it is much better suited for the GPU than problems investigated in the paper.

[10] Like the one of the control of a large number of moxel systems - author’s emphasis.

1. Recognition that GPUs can be used to solve an important class of problems concerning software control of million-element Constrained Motion Cyber-Physical Systems.

2. Algorithms are devised for using existing GPUs for the control of the PRE type of environment, specifically the calculation of the physical position of the moxels using a Z buffer [41], [34], and for using the fragment shaders for controlling the physical characteristics of each moxel.

3. For the group of algorithms mentioned in the previous point, we have developed a simulator of the PRE and demonstrated the viability and scalability of the algorithms to millions of moxels.

4. An extension of the proposed approach was described for a range of systems, including CirculaFloor [3], Catoms [1] and MEMS based systems.

5. This thesis describes how to address physical characteristics of the individual moxels in the range of cases, including physical characteristics of the moxels, manufacturing process variability and addressing most common moxel failures.

In terms of the scope of this PhD, we:

1. Constructed a simulator of the Physically Rendered Environment that uses GPU along with Z buffer algorithm [41], [34] to control moxel based systems.

2. It is demonstrated through the simulator that our algorithms scale to controlling millions of actuated elements per second using a simple model of a moxel's physical characteristics, on commodity class hardware.

3. This thesis describes how system can take the physical characteristics of the actuation elements into account when calculating the positions of the elements. The fragment shader was used for controlling moxels based on the piecewise linear response curve.

4. In addition to the previous points that were all demonstrated on the simulator, in this PhD thesis it is also shown how GPU control can be extended to the wide range of systems including CirculaFloor [10], Catoms [1], and MEMS based assistive technologies [4]. Furthermore, we show how GPU based control could be extended to address multiple scenarios related to the physical safety of the user in the large scale system. Note that these particular extensions are not yet implemented as a part of the simulator[11].

Finally, although the scope of this PhD is limited to software, the software proposed here is necessary for achieving hardware scaling. Once both software and hardware systems are completed, there is significant potential for future expansions on both the software and hardware side. Such expansions would most likely be in areas of software control, HCI aspects of tactile based assistive technologies, sensory substitution, and the combination of Cyber-Physical Systems with virtual

[11] The reason is that implementation on the simulator of those algorithms was outside of the scope of the original thesis proposal. Furthermore, there is no significant concern about viability of their implementation on the GPU so it can be argued that implementation of them on simulator would represent more engineering then research contribution.

environments[12]. It is reasonable to expect that all of these areas would have significant research potential.

[12] Many examples of such works are discussed throughout the rest of this thesis.

Chapter 2 - Related Work

There are two broad groups of work related to this PhD thesis:

1. Work related to the hardware, which benefits from the approach proposed in this section and could be considered a motivational application for our algorithms.
2. Work related to the use of the GPU and software for the control of the physical systems consisting of millions of elements.

The following sections will provide more detail relating to these categories of work.

2.1 Related Work in the Area of Physical Environment Deformation and Manipulation

The Cyber-Physical Systems presented in this section benefit from the approach taken in this PhD thesis, and, as such could be considered motivational for our work and are some of the use cases for the proposed algorithms.

This work can be further divided in two groups:

1. Work related to the actuation and deformation of the environment in which the user is physically located. These are macro scale systems.
2. Work related to providing tactile feedback to the users. Main examples of these are tactile systems, in particular the assistive technologies for the visually impaired persons.

2.1.1 Macro Scale Environments

Macro scale environments are hardware environments large enough to accommodate the user within them. They are related to the vision of the convergence of robotics, vision and computer graphics for the creation of the fully immersive environment for the user, as described in [42] and some implementation examples, challenges and ideas are described in [11], [43], [44], [45], [10], [46], [47], [6], [7] and [48]. This section expands on the most related of the works enumerated above.

On the hardware side, the first example of the Cyber-Physical System used as motivation for this thesis is our earlier work - Holodeck PRE [6], [7] that proposed a large scale moxel based Cyber-Physical System controlled by the GPU (currently implemented as a HoloSim simulator). Algorithms presented in this thesis are directly related and applicable within a PRE based environment.

Intel's Dynamic Physical Rendering (DPR)/catoms/Claytronics [49], [2], [50], [51] is a vision of using small physical elements (e.g. small balls) called catoms, to create arbitrary 3D shapes as seen in Figure 5. The vision of physical balls is expected to be practical by 2012 [1], with a previous physical system being the significantly larger Planar Catom V7 [52], [1] and current catom-based physical systems based on the 1mm x 10mm cylinder [53].

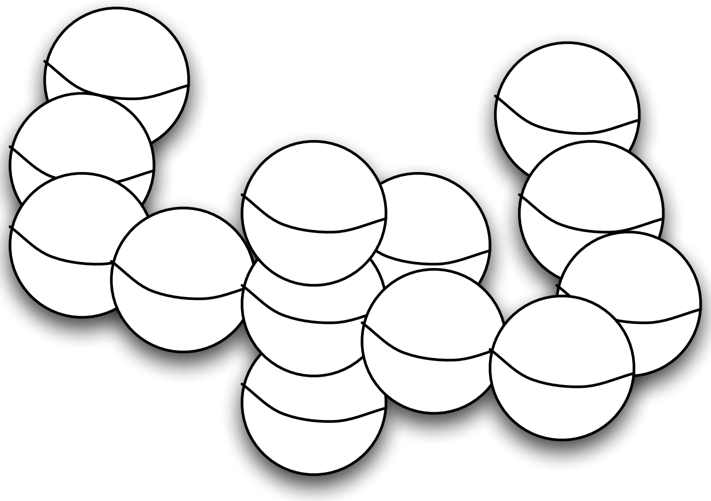

Figure 5. Catoms forming 3D shape (the author's illustration based on {Geller, 2009, Communications of the ACM}).

Catoms [54], [1]/Claytronics [2] are able to orient among themselves in arbitrary positions towards each other. The DPR concept is similar to micro-robot ensembles and programmable matter [54].

On the software control side, work on catom's control is in its early stages and is unclear just how fast these software algorithms are, when it comes to rendering of a large number of catoms. Note that catom's vision is based on the decentralized software control but it is fair to say that there is no total consensus among researchers that decentralized software control is the best approach for controlling large scale systems [1][13].

[13] Furthermore, as [1] is pointing out, for catoms to be useful they have to perform some function on the larger scale (e.g. what do bodies made of catoms do) - and if that is the case, then we would need to have a central control somewhere in the system. We submit that the computer that is responsible for providing central control would likely already be equipped with the GPU, and that makes ability to use it an interesting research question.

In order to address the problem of coordination of a large number of catoms, some of the latest efforts were based on the hierarchical software control of catoms where a large number of catoms are treated as a group [55] as well as anatomical/biological inspired control schemas where catoms emulate anatomical structures (bones, muscles) [56]. The experimental results of [55] have shown the ability to control modules consisting of few thousand catoms. However, it is unclear how these schemas would behave when trying to control millions of catoms and [56] still considers emulation of the complete body an open question. Considering these issues, it is fair to say that software control of the system on a scale of millions of catoms is a research question yet to be fully addressed.

Related work in the area of self-reconfigurable systems consisting of large number of robotic elements [57] states that new algorithms will be required for the proper control of large systems, and that, although algorithms for handling million element systems were developed under ideal conditions, significant challenges remain in the area of software control of systems using large number of robotic elements: work that demonstrates coordination of number of elements on the order of 421,875 elements [58] shows that a system consisting of a cube with 75 elements needs 10,345 timestamps[14]. Furthermore, the object was significantly changed even while moving on the flat ground.

[14] Within a timestamp, each element had a chance to move so that, after completing all timestamps, the end result is movement of the cube equivalent to 74 element diameters.

In addition, the number of catoms needed to simulate a surface is significantly larger than the number of moxels[15].

The relation between the catoms and PRE is summarized in the author's earlier work [6], , which notes that PRE's ability to quickly render large surfaces makes it a good compliment for the DPR, as PRE could be used to render ground plane while DPR is used to render objects that couldn't be easily represented with moxels.

Similarly, the author's earlier work [7] states:

"Ideal environment is likely to use the Holodeck approach for ground areas, and DPR approach for smaller and more complex objects in the room."

Another related work is CirculaFloor [3]. From the author's previous work [7]:

"CirculaFloor [10] is a locomotion interface that uses a set of movable tiles to achieve an omni-directional motion while providing the illusion of infinite distance. CirculaFloor simulates only flat surfaces, so PRE and CirculaFloor are complements. Combining a moxel-based surface with the tiles of CirculaFloor allows for the extension of the PRE which is capable of simulating unlimited environments."

Software algorithms presented in this PhD are able to directly control a Holodeck environment and here we present how a Holodeck type of the environment could be combined with catoms [1] and CirculaFloor environments.

[15] Moxels are covering only surface ground plane with mechanical elements while catoms are covering a complete surface and in some instances the inside of the object they model. Some proposals for use of moxel might even result in covering the inside of the volume in order to provide, for example, a complete and anatomically correct model of the human body [56].

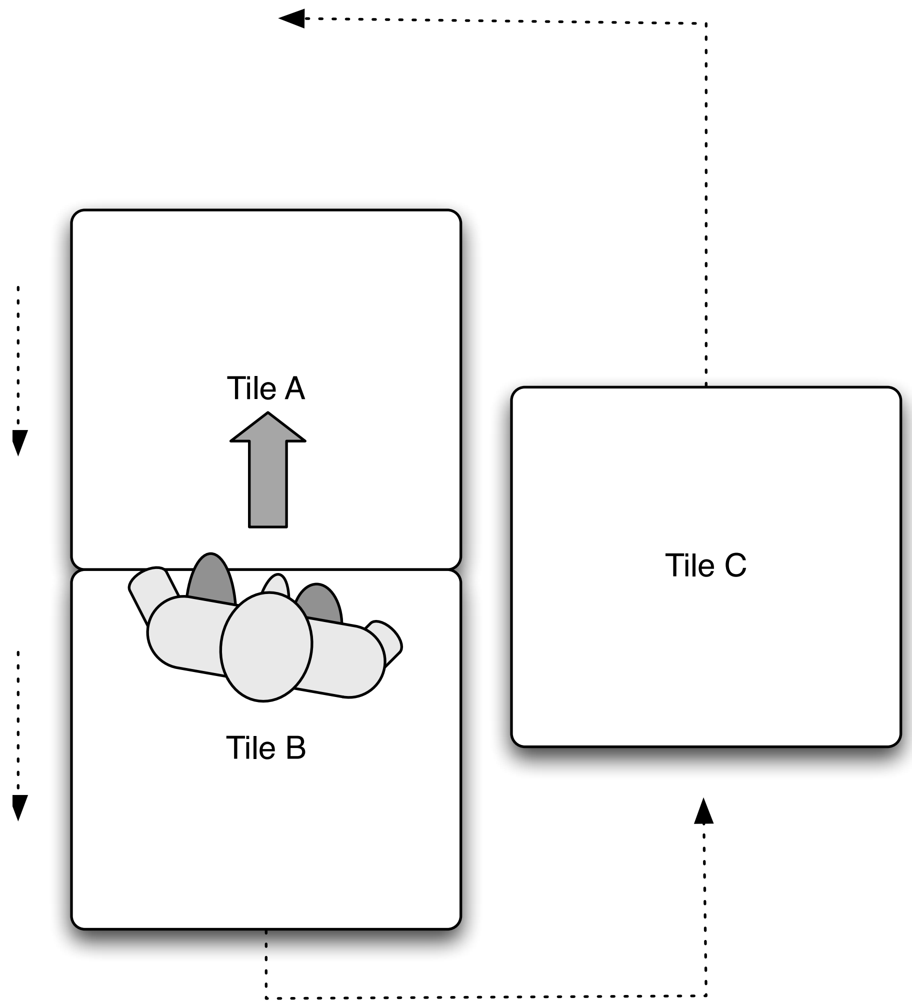

Figure 6. CirculaFloor (the author's conceptual illustration based on [10]). Note that circulation mechanism and number of tiles in current CirculaFloor is different, and that previous picture is used for the illustration purposes. See [3] and [10] for details.

Virtual reality systems combine physical objects with computer generated imaging [28]. Many concepts from virtual reality such as helmet-based split-view immersive environments [28], are quite complementary with the vision of physically rendered environments. Depending on the application, we envisioned that users navigating through a PRE could be equipped with helmet-mounted immersive displays, and the terrain around the user would change in real time in order to conform to what the user is seeing.

Omnidirectional Treadmills [12], [59] and the Torus Treadmill for Infinite Floor [60] allow for user movement in all directions while simulating a floor surface of an unlimited size, but are limited to the a surface only. Related work includes treadmills that can angle themselves such as Sarcos [11]. They relate to PRE as they provide similar functionality, but inherently PRE provides for better control of the local slope and greater freedom of movement (as some of the approaches for the simulation of the local slope use tethering of the user at torso and applying force on him to simulate slope) [43], [61], [62].

In regards to virtual reality, the related system, is SGI's CAVE [63]. Although it allows for a very impressive virtual 3D environment, it is unable to physically deform terrain to conform to the environment pictured - you see stairs and they might look "almost real", but don't try to step on them.

Omnidirectional treadmills can be combined with SGI's CAVE system, though this combination is limited to the simulation of flat surfaces. Attempts to simulate sloped surfaces tethered users to the environment [61], [62] and, as such, obviously limited user mobility. PRE type environments are a natural complement to these systems.

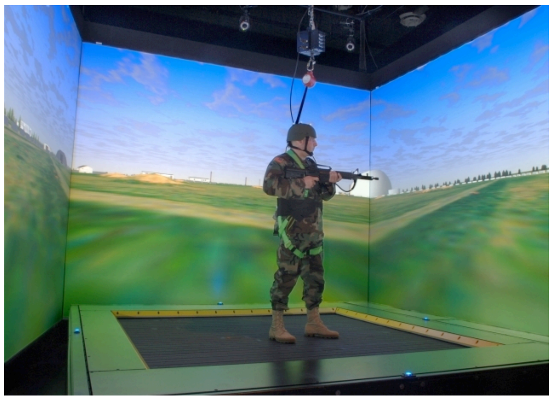

Figure 7. Soldier on an omni-directional treadmill, inside of CAVE [64].

2.1.2 Portable Scale Environments

Portable scale environments are used mostly in the areas of the assistive technologies and sensory substitution. The idea is to partially substitute a sense of sight with the sense of touch, by manipulating the surface of small scale (and sometimes portable) devices.

The best known example of this kind of work is a Braille enabled terminal [65], like the one seen in Figure 8. Although in use for a significant period of time, these terminals have limited number of moving elements, are limited to showing only text and cannot show graphics.

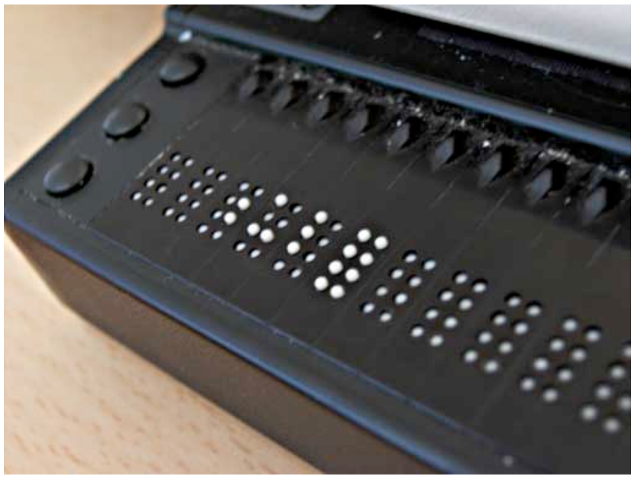

Figure 8. Refreshable Braille terminal [66].

Braille terminals are just one example of tactile displays. Tactile displays use mechanical, electrical and thermal stimuli of the skin, and are implemented with static or vibrating pins, focused ultrasound, electrical stimulation, surface acoustic waves and other techniques [18], [67], [68].

Furthermore, most of the work in the tactile feedback field was related to the haptic display of a static image, with the bulk of the work being done in the area of haptic devices as opposed to surface texture change [69], and a recent study of the field acknowledges that it is still in the early stages of development [70]. Moreover, there has been little work done so far on what will be the equivalent of animation (in the computer graphics sense), and the work done is limited to low density actuators [71], [72], [20], [24]. Some examples of work done on defining “tactile language” are given in [67], [73].

There is previous work done on automating tactile image translation [74], [75], [76] but that work focuses on simplification of the images for low resolution displays (and [74] is helping the tactile specialists in creating images and as such is not oriented toward real time processing of the images).

Although, historically, these devices feature low resolution, this is not due to human inability to distinguish finer resolutions but due to limitations in technology. Humans are able to discriminate fine grained texture elements on a sub-millimeter scale with the fingertip [77] as well as tip of the tongue [32] (there are systems that stimulate the tongue surface so as to reflect the image captured by the video camera).

Some examples of the expansion of Braille terminals to higher number of pins are tabletop surfaces based on rod actuation like Digital Clay [78], [79], [17]. Another group of related work are systems that combines video projector with the matrix of rods that actuates canvas surface - FEELEX [80], MATRIX [81], Northrop Gruman's TableTerrain [82], [83] and Relief [84], [85]. The number of moxel-like moving elements in such a system could be very large, with [33] suggesting that proposed manufacturing methods used for 5x5 prototype developed could be extended to arrays with 1M elements.

Consequently, there is a need for actuation and control of a large number of elements. Recent advances in the area of MEMS devices [86], [67], [87] pave the way for the future development of hardware capable of providing sensitivity approaching discrimination thresholds of the tip of the finger or the tongue. One example of such a system is BrailleEye [4], [21] currently being developed in the Mechanical Engineering department at CU Boulder. At the moment, most of the tactile systems are still in the development process and are limited to few tens of pins [18]. However, as we show, based on the reasonable surface size and human touch threshold, this situation is transitory and number of elements is likely to significantly increase in the future.

Another method for tactile feedback based on electrovibration was proposed in [5]. The authors used electrovibration to simulate the feeling of friction as sensed by a sliding finger. In regards to hardware implementation, this system offers advantages over the MEMS based systems due to its reliability and uniformity of the presentation of the tactile feedback, although it has the disadvantage of not being able to stimulate a stationary finger and having a somewhat smaller magnitude of tactile sensation it could render [5]. However, from the software control system point of view, this system is very similar to the MEMS based system and provides similar motivational advantages for our work to MEMS systems.

In addition to the assistive uses of those technologies, tactile interfaces are also useful for normal-sighted users. One very common use case is related to the rise of touch-screen phones like the iPhone [88], featuring virtual keyboards. Their users could benefit from tactile feedback, and work is currently being done on dynamic displays that are able to provide tactile feedback by physical deformation of the screen surface [89] or electrovibration of the screen [5]. It is reasonable to assume that as users will be able to see the display changing, even higher rate (frequency) of change of the tactile screen will be needed during tactile reconfiguration, as tactile display would need to match the visual display.

Although carbon nanotubes are still an active area of fundamental research, carbon nanotubes could be considered Constrained Motion, 1D system [22], [90]. At the nanoscale, we will need to control of even higher number of elements.

Thus, we have a clear need and trend toward increasing both spatial resolution and frequency of movement of the elements in the refreshable displays, and we currently do not have a definitive answer on software approach that should be used for the controlling of these devices in the future. This PhD thesis is addressing that gap by providing software algorithms for highly scalable control of millions of elements, using commodity hardware.

2.2 Related Work in the Area of Software Algorithms

The general trend in the last few decades is toward implementation of graphics rendering engines that use GPUs [34], [35]. This is caused by the fact that GPUs are optimized for computer graphics related calculations, and are much faster than CPUs on graphics intensive tasks. As an additional positive effect, GPUs are cheap and ubiquitous in modern personal computers, appearing even in consumer electronics such as smartphones like e.g. iPhone [88], with the 3G S version supporting OpenGL ES 2.0 and programmable shaders [91], [92], [93].

Programmable GPUs have been used for acceleration of rendering of computer graphics for a number of years, and many practical uses were discussed in e.g. [36], [35] and others. On a related note, GPUs were used for computation of images in the volumetric displays [94], [95], with an interesting use case being the planning of radiation therapies [96].

In addition to rendering support, GPUs were used for a wide variety of general computation related tasks with some examples being volumetric rendering, fast fluid dynamics simulation, ultrasound visualization [97], flow simulation, option pricing [98], image correction in scan-beam projectors [99] and an acceleration of the reliability analysis of MEMS devices and simulation of electromagnetic wave propagation [100].

Although widely used to accelerate rendering speeds and general computation, GPUs have not been used as often in relation to robotic control, and even when used it has been mostly for computation and vision recognition as opposed to the direct control and coordination of a large number of robotic devices. Examples of the use of the GPU in these areas include ultrasound image analysis and control of a robotic arm (as proposed in [101] and implemented in [102] and [103]), proposal to use GPU for computation related to the shape recognition [104], solving of the kinetic equations [105], as well as proposed use for the speedup of the Constrained Motion planning equations [106].

Still, with all of the previous work, there was little work done on the coordination of the large number of mechanical elements using GPU. Our earlier work [6], [7] describes algorithms allowing the use of the GPU for the software control of the Holodeck/PRE like systems. Earlier work on the Digital Clay system briefly mentions possibility of the GPU use [79], and later work on Relief system used GPU [85].

Work in this PhD thesis features implementation and enhancement of the author's previous proposals as outlined in my earlier work [7], [6]. This thesis specifically investigates the problem of software control of PRE and MEMS based Constrained Motion Cyber-Physical Systems, shows viability and scalability of the previously proposed ideas and extends those ideas to a number of different systems like catoms [1], CirculaFloor [10] and MEMS [21] based devices. Finally, we propose extensions on how to deal with some of the expected moxel failure patterns in the PRE type of the environment.

On the software side, algorithms proposed in the author's previous work [6], [7] and modifications made in this thesis are significant contributions in the area of application of GPU on the control on the Cyber-Physical Systems. Furthermore, observation of the particular suitability of the commodity, relatively inexpensive GPU hardware to the range of the million elements Constrained Motion Cyber-Physical Systems is also an important contribution.

Before we discuss differences between the most related work in the area of the software algorithms in detail, we would like to highlight that, compared to all other work by other the authors (including my previous work), there is a unique contribution of the work presented in this thesis that is, to my knowledge, the only body of work at this point in time to cover, in the context of the Cyber-Physical Systems, the following topics:

1. Addressing the range of systems from tabletop to room sized systems, while being primarily based on the combination of the Z-buffer and the fragment shader calculation of the position of the moxels.

2. Addressing the integration of the moxel based system with a diverse range of systems including catoms/Claytronics [1], [2], CirculaFloor [3] and MEMS based multistate and bistate devices, using GPU.

3. Expanding on the physical limitations of the moxels that could be per moxel specific, vary from moxel to moxel to account for limitations of the manufacturing system, and account for physical characteristics of moxels in which position of one moxel affects other neighborhood moxels, as well as dynamically account for multiple failure modes in the moxel actuation mechanisms.

4. Present some early results on the physical safety of the users in room-sized systems.

5. Finally, focus on scalability of the software control of large number of element mechanical systems using GPU hardware.

Noted below are works related to what is presented in this thesis as pertains to the area of software algorithms:

1. The author's earlier work [6], [7] proposed a combination of orthogonal projection and Z-buffer readout, as well as the use of the fragment shader for taking into account physical characteristics of the moxels in the context of the large scale moxel based physical environment. This PhD thesis extends that work by providing further details on implementation of the proposed techniques, proof of feasibility and scalability of the proposed software control concepts, as well as all of the elements identified in the previous part as unique to this thesis.

2. Shadow Mapping [107] has similarities with our idea of using Z buffer with respect to the idea of the manipulation of the camera position and use of the depth buffer for the test, but our work is different with the respect to requiring matching pixel in the rendering resolution to the size of the moxel, area of the application (device control), extendability on the display surfaces that are not fixed in space (we can combine with movable surfaces like CirculaFloor [10] or catoms [1]). Furthermore, our basic Z buffer based control needs only a single

rendering pass as opposed to two rendering passes that Shadow Mapping requires.

3. Digital Clay [16], [78] actuates vertical arrays of pins for haptic feedback and would qualify as a Constrained Motion Cyber-Physical System as defined in this thesis. That work briefly mentions the possibility of use of the Z-buffer for the calculation of the vertical arrays positions as one of possible research direction they investigated in the project [79]. This thesis expands by providing how exactly an implementation of the Z-buffer control will be achieved, using GPU to much greater extent than previous Digital Clay related work proposed [16], [78], and also by taking into account the physical characteristics of the mechanical elements on GPU, using the fragment shader, and addressing imperfections in the fabrication process. In addition, all elements previously identified as unique to this thesis are also different from Digital Clay.

4. Distance Fields [108] are a volumetric representation of the distance of a point in space to 3D objects, and [109] discuss use of the GPU for the calculation of discrete values in the distance field using Z-buffer for the purpose of fast distance queries. Some related work on the distance field calculations includes [110] and [111]. Although these works have similarities with and are a further extension of Z-buffer based distance calculation from the ground plane proposed in our earlier work [6], they are working in completely different domain addressing proximity queries from geometric objects. Furthermore, aside from

the use of the Z buffer, all other already described considerations that make this work different than e.g. Shadow Buffer [107] apply to Distance Fields, too.

5. The author's earlier work [6], [7] proposed combination of the orthogonal projection and Z-buffer readout, as well as the use of the fragment shader for taking into account the general concept of physical characteristics of the moxels. Later work on Relief [84] is another example of the 3D pin actuation system. [85] proposes a similar mechanism to the one we are using here for camera positioning and Z buffer readout mechanism and applies shaders to account for the limited travel of actuation rod. Relief [85] uses multiple output channels (e.g. color channels to signal that a rod is out of the range of motion). This thesis expands on the approach proposed in [85] for passing an actuator height map for addressing differences in the physical position between calculated and actual position of the actuation elements in respect to GPU based reaction to the physical feedback on the moxel system. However, we allow for the more general case of reacting on any physical characteristic of the moxel, as well as account for moxel manufacturing variations. Similarly, the approach we are using for the control of the multiple output channels could be considered a generalization of the communication of the rod being out of the range of motion proposed in [85] to controlling other properties of the physical system using GPU output. Beyond those similarities, there are many differences - to start with, this thesis focuses on GPU use for the acceleration of large scale software control of moxel like systems with high moxel density, while [84] is a tabletop system with limited moxel density. Furthermore, all the

elements identified in this chapter as unique to this thesis differ from the Relief system.

6. There are some similarities with volumetric displays in the sense that they calculate the image from many different positions [94], [95]. However, while they use many perspectives similar to that of the viewer of the image, we are using a rendering perspective that is very different than the one of the user, and we are oriented toward mechanical device control as opposed to volumetric image display on an optical device and we don't need to render anything but the geometry of the image (so there is no need for e.g. texture application, fog display etc). Furthermore, while they are proposing a full new graphics language for integration in the graphic pipeline [95], we are proposing integration in the rendering pipeline using AOP [112], [113] and limited program modification. Finally, all other previously mentioned differences with e.g. Digital Clay system would apply to this case, too.

7. Finally, there are similarities with MATRIX [81] in the recognition that systems using moxels would benefit from the non-CPU based control. MATRIX uses FPGA [81], while we propose use of the GPU for our software control.

8. Digital Clay and Relief are further expansion on the range of prior work including project FEELEX [80]. All differences between our work and Digital Clay and Relief already described apply to those systems, too.

This thesis presents a software approach that addresses important practical problems in the field, while showing viability of use of the well known and broadly available commodity hardware in the form of the GPU. Furthermore, the body of related work in the field shows that this work is addressing problems related to the broader community..

Chapter 3 - Requirements of the Software For Controlling Constrained Motion CPS

There is significant use for computer controlled deformable physical surfaces consisting of large numbers of movable elements.

On the micro scale, there is a significant potential for the use of devices that have similar characteristics to Constrained Motion PREs in the field of assistive and tactile display technology [114], [8], [31], [86], [72], [115], [116], [117].

In particular:

1. MEMS based screens could be used as assistive technology, allowing a person with vision impairment to perceive 3D depth and shape. Combining such a system with a video camera would allow us to render graphics and even video in a way that a visually impaired user could perceive. One such system uses a tactile system to stimulate user's torso based on the image from the video camera [73]. Furthermore, there is a project currently in progress in Mechanical Engineering department of CU Boulder that is developing a MEMS based screen, with plans to use some of the software algorithms proposed by the author in one of its future implementations [21].

2. Utility of tactile systems is not limited to assistance of visually impaired users. Tactile systems could be used in combination with touchscreen phones [89] to provide tactile feedback to normal-sighted users. There are many other uses in which tactile systems are beneficial. Such tactile feedback was found useful

even for users without any disability - [118] evaluates tactile cueing to helping soldiers doing navigation in challenging environments and quotes examples of the tactile feedback used by pilots under high-G force [119] and astronaut training [120].

On the larger scale, deformation of the terrain on which the user stands is even more intriguing. In our earlier work on PRE [6], [7]) we proposed combination of the physical deformation of the terrain with a helmet on the user's head [28] which is used to provide a computer graphics generated environment. In that combination the user sees computer synthesized image, and feels under his feet the computer controlled deformable terrain.

That will be the next logical step in the evolution of virtual environments, and would allow for the following use cases (following use cases are adopted from the author’s earlier work [6], [7]):

1. In the flight simulations, it is possible for the instructor to setup a scenario that is too difficult, dangerous or expensive to in a the real plane [121]. An environment like this could allow for the equivalent scenarios like “a soldier is fighting in the cave, and just as he sees the enemy, he slips on his right foot”.

2. It is possible to train search and rescue personnel and civilians in building evacuation in the case of fire. Our approach would allow much more immersion then previous work [122] (which was using virtual environments and omnidirectional treadmill alone) would allow for.

3. The system is opening the possibility of distributed training in which multiple systems of this kind are networked, providing an impression that all team members are at the same location. This is very intriguing possibility for the next step in telepresence research.

4. Once this system is possible to create at a reasonable price, it will be a breakthrough in the entertainment world as it would allow very high levels of physical interaction within the game world virtual environment. Success of the platforms like Nintendo's Wii [123] are showing that there is a strong interest in video entertainment that would involve certain levels of physical activity.

These examples illustrate a variety of exciting applications of PREs. For them to happen, software control of the potentially million element Constrained Motion PRE needs to be resolved.

In order to address such a system, the software control system needs to have the following properties:

1. It needs to be able to control millions of elements per second. For example, if the diameter of the rod is 1 mm, it needs to control 10,000 moxels per square meter, leading to the millions of moxels required just to cover the floor surface.

2. It needs to offer precise local control of each moxel, taking its physical properties into account. For example, how fast can the moxel start or stop? How to account for moxel’s inertia? How about other physical parameters?

3. Integration with virtual environments should be possible, so that the same or at least a similar description of the environment could be used for both graphical rendering and physical control of the system.

4. If the hardware has the ability to simulate slipperiness of the terrain on which the user is standing, the software should have the ability to control that slipperiness.

5. The approach taken with the software algorithms should be applicable to a wide range of the Constrained Motion PREs.

6. Latency must be appropriate for the calculation of the position of the elements in the physical system[16].

7. The proposed approach should be applicable not only to desktop and server computer systems, but also to consumer electronics such as smartphones, allowing devices such as smartphones to be used as computer control devices for the assistive technology solutions.

This PhD thesis proposes a novel software approach allowing for the control of Constrained Motion PRE systems using common of the shelf hardware, namely GPU. It will show that:

[16] Appropriate latency clearly depends of the physical system used. Later chapters would discuss latency, as well as relation of GPU and real time systems.

1. Our algorithms allow common, off the shelf GPU, present in almost every modern desktop system and becoming increasingly common in smartphones such as iPhone [93] to be used to control Constrained Motion PRE systems.

2. Said algorithms do not require any modifications of the existing GPU hardware, meaning there will be no expenses necessary for the design and fabrication of the modified GPUs.

3. Said algorithms are able to calculate the position of hundreds of thousands and even millions of moxel elements per second on commodity class desktop hardware.

As is demonstrated by this work and other work quoted in the related work section, coordination of the million elements systems is not a simple problem. Due to the large number of elements involved, doing that coordination at an interactive rate is, while accounting for the physical properties of the moxel, a computationally intensive operation.

There is a definitive analogy between Holodeck/PRE and computer graphics. The GPU is more efficient in per-pixel calculations than the CPU - there is little doubt that the historical direction of graphics related calculation is moving toward specialized hardware in the form of the GPU [34], [36], [35], [28], [124], [37]. We will show that GPU based control of the PRE and MEMS based systems would provide significant benefits in the form of the fast control of the large number of elements, using relatively inexpensive, off the shelf hardware.

To summarize, the PRE as envisioned in [6], [7] will be capable of the following features:

- Computer-controlled rendering of arbitrary physical surfaces on a grid of moxels
- Control executed in real time
- With realistic tactile feedback and visual feedback
- Able to support the weight of a user standing, walking or running in designated environment

Although Holodeck/PRE will be used as a primary motivator of the GPU based software control, GPU is in no way limited to the Holodeck like system. The scope of this PhD thesis will be limited to the software control in Holodeck, but here we will show that there is a generalization of the software algorithms used on the other systems like CirculaFloor/Holodeck crossover [10], [6], [7], assistive technologies [86], [8], [125], etc.

Chapter 4 - Overview of the Contributions by This Thesis

The work that was performed as a part of this PhD research falls into two categories:

1. Algorithms and methods for the scalable software control of multimillion and larger Cyber-Physical Systems. As a part of it, it will be described how these algorithms and methods could be applied on the multiple classes of the Cyber-Physical Systems including large scale Constrained Motion Cyber-Physical Systems, MEMS based tactile technologies and integration with systems like CirculaFloor [3].

2. Demonstrates that proposed approaches to calculation of the moxel position in the Z buffer and accounting for physical response curve are highly scalable and capable of controlling multimillion elements Constrained Motion Cyber-Physical Systems. In order to do that, a simulator of the Constrained Motion cyber physical system, HoloSim, was developed.

Simulator was developed because of the fact that previously mentioned Cyber-Physical Systems are expected to be expensive (as in the case of Holodeck environment), and in some cases (like BrailleEye [21]) are still in process of being developed. This necessitates that problems of the software control of such environment must be resolved prior to building a physical environment.

Although we have simulator of the physical environment, that simulator demonstrates all techniques necessary for the control of the physical environment and is performing range of calculation necessary for the control of the full scale physical environment[17]. In that respect, from the perspective of the software control of the physical environment, the simulator goes beyond the level of complexity required from the software needed to control a physical system if one was already in existence. This is so, since the simulator requires all parameters necessary to control physical system's element position in addition to functionality particular to the simulation and visualization of the position in the physical system that wouldn't be necessary if we had physical system.

Considering hardware interface for the control of the physical system, standards like DVI [126] that are allowing very high control bandwidth of millions of discrete electronic elements (in this case, pixels) are widely used. That demonstrates that communication with the hardware is not a difficult engineering problem[18].

The diagram in Figure 9 shows all elements of the problem domain that this work is addressing. Shaded parts are the parts that required performance demonstration in order to prove scalability and due to that were addressed both in the

[17] Calculations performed are limited on calculation of the position of the elements in the environment, as that is the scope of this PhD work. Some physical environment might require additional software and firmware that is dependent of the environment that is necessary for the physical implementation of the environment. As this will beimplementation specific, simulator didn't covered them.

[18] At least for any system that is having moxel densities comparable with the pixel densities of the modern displays, and systems of that size already are multimillion elements even on relatively small sizes. It is possible that there could be challenges related to the control if we are to go above currently available pixel densities (especially in the area of nanotechnology), but these are challenges in the area of the mechanical and electrical engineering. Finally, although DVI was mentioned as one way to get data out of the tactile system, as discussed later in the document, it is not the only (or even the best) way.

text of the thesis and the simulator. Non-shaded parts are addressed in the text of this thesis.

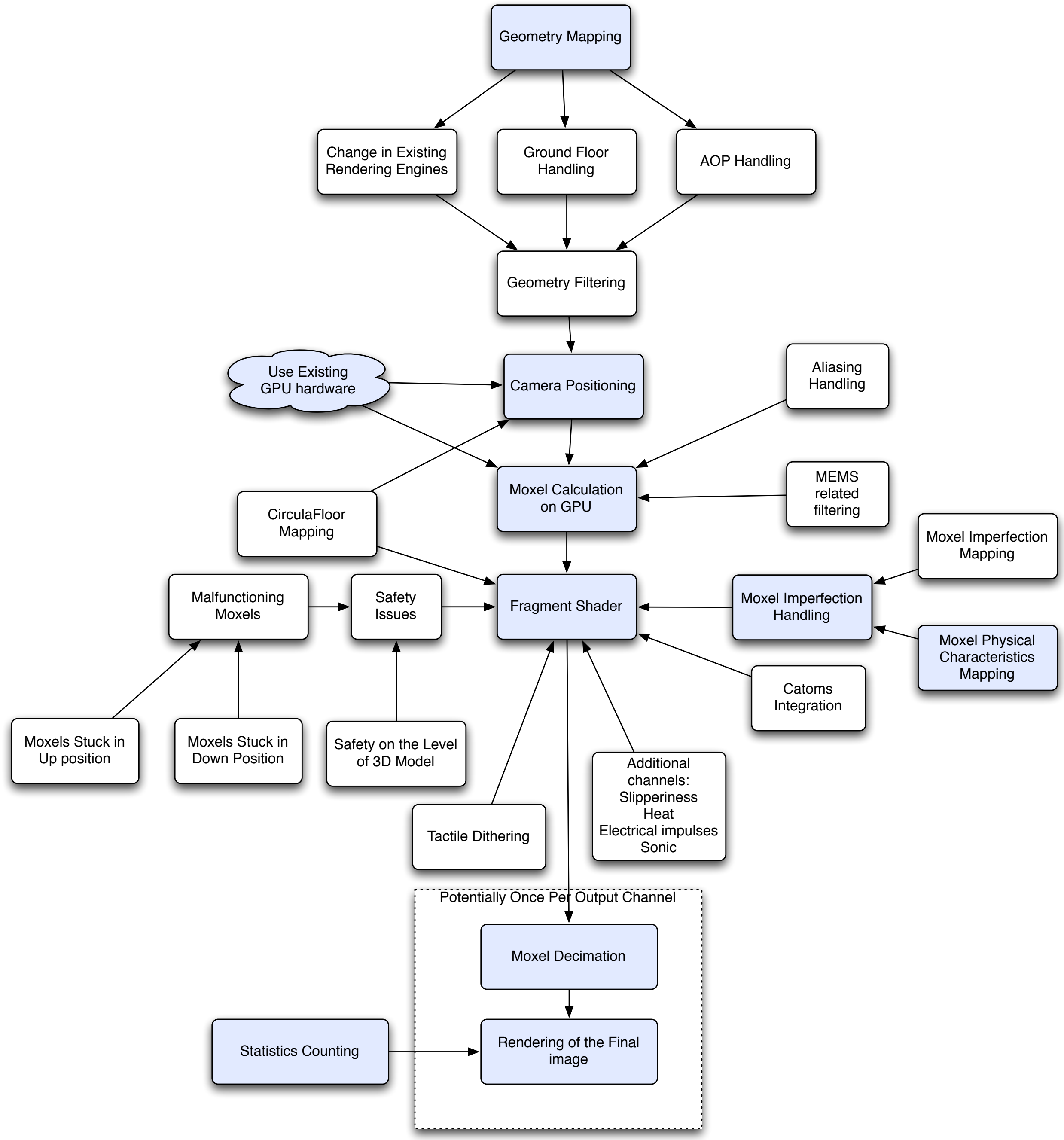
Geometry Mapping
Change in Existing Rendering Engines
Ground Floor Handling
AOP Handling
Geometry Filtering
Use Existing GPU hardware
Camera Positioning
Aliasing Handling
Moxel Calculation on GPU
MEMS related filtering
CirculaFloor Mapping
Moxel Imperfection Mapping
Malfunctioning Moxels
Safety Issues
Fragment Shader
Moxel Imperfection Handling
Moxel Physical Characteristics Mapping
Catoms Integration
Moxels Stuck in Up position
Moxels Stuck in Down Position
Safety on the Level of 3D Model
Additional channels: Slipperiness Heat Electrical impulses Sonic
Tactile Dithering
Potentially Once Per Output Channel
Moxel Decimation
Statistics Counting
Rendering of the Final image

Figure 9. Overview of the problem domain addressed by this thesis. Non-shaded areas are addressed in the text of the PhD this and shaded areas are addressed in text of the PhD thesis and prototyped in HoloSim.

As a part of the original vision of earlier work on PRE [6], [7], we believe that the following capabilities should be present in the final software system controlling PRE-type environments:

1. It should allow for the reuse of the 3D model definitions that might already exist for the purpose of the computer graphics visualization of the environment, so that most of the work that went into the defining 3D models is usable both for the computer graphics based visualization, as well as for the physical rendering of that environment in the physical system [7].

2. It should be capable of integration with systems like catoms [1], CirculaFloor [3] and MEMS based systems, while at the same time taking into consideration physical imperfections of the system as well as addressing various moxel failures.

4.1 System Description And Research Considerations

We will now provide a brief overview of the elements in Figure 9, in order to present a brief outline of the system. Subsequent sections will expand on parts of the system much further.

Geometry Mapping - In order to be able to reuse existing 3D models, the first step that needs to be performed is to take the 3D geometry description from the 3D world coordinates (in the OpenGL sense [124]) and transform them into the format that is appropriate for the physical rendering. That is achieved by mapping the existing 3D world description on the ground plane around user (in the case of the large scale systems) or on the user viewport (in the case of the MEMS systems).

Change in Existing Rendering Engines - We must ask: how can existing rendering engines be changed so that they incorporate Holodeck in their pipeline? Some of the techniques that could be used in this respect were discussed in the author’s previous work [6] including the point that the techniques used in the modern rendering engines that result in different visual output from the basic 3D model (like Bump Mapping and Displacement Mapping) would require modification on the level of the Holodeck system.

Ground Floor Handling - in the 3D graphic with output on 2D display, it is customary to remove parts of the picture that are behind user. [124], [34]. This is a perfectly reasonable approach for the visualization purposes, but as pointed out by the author's earlier work [6] for the PRE purposes this technique become very problematic if user decides to step back in a large scale physical environment. This implies that while rendering, OpenGL clipping planes must be set in a position that encompasses the area immediately behind the user when drawing a scene into the Holodeck based environment, and that the position of the culling planes is function of both user position and predicted maximum speed of user movement.

AOP Calculation - Is there a way to depend on the existing OpenGL libraries without having access to the source code? Although a “wrapper” OpenGL library could be written, another proposed option is that combination with the AspectC++ [112] compiler would allow for the AOP based [113] approach to be implemented. This allows for a better architecture in which we can intercept every call, and not require that calls have to be proxied, as discussed in the author's previous work [6], [7]. This is especially important because OpenGL is not a trivial specification, and even "thin proxy" on top of the OpenGL would be a significantly bigger chunk of work than AOP based interception

CirculaFloor Mapping - would address mapping of the existing techniques that are designed for a static array of the moxels in limited 2D space into CirculaFloor system [10], [3] that is designed to simulate infinite space. In order to successfully perform those operations, it is important to account for the fact that moxel based system suffers from the limited range of motion in Z coordinate too. Thus, the challenge was to devise a system that not only uses GPU to calculate static position of the moxel in the room, yet allow for XY movement to map that calculation to the dynamically moving system of CirculaFloor, while at the same time not losing scalability properties of the system. That requires that mapping happens completely on the GPU. Furthermore, a system had to be devised that would allow presentation of the infinite height in the limited 3D space, in effect requiring re-centering of the users in Z coordinate, again while performing only GPU based calculation.

Geometry Filtering - challenge here is which parts of the 3D model geometry are to be mapped into the Holodeck based system? For example, transparent polygons shouldn't be presented to the Holodeck system, nor any polygon representing material the user is not able to stand on. This would likely require combination of per geometric primitive work (e.g. checking alpha channel value for primitive) and some modifications in the Rendering Engine (in order to allow to programmer to tag particular polygons as “not to be displayed” in Holodeck if e.g. structure is too weak to support user's weight).

Camera Positioning - how to position camera in a way that does not require change in the existing GPU accelerators, while allowing for the massively scalable computation of the moxel positions? Furthermore, the challenge is to accomplish this without requiring complete redoing of existing 3D geometry models, as well as how to minimize changes in the rendering engine. These issues are addressed by adopting new algorithm that is in its complexity and approach similar to the Shadow Buffer [107], but is applied in the field of the robotic calculation.

Moxel Calculation on GPU - it is necessary to fit in the constraint of the GPU - e.g. map our work in the way that inputs are processable on GPU (which is what we did in camera positioning) and that outputs are meaningful outside of the GPU. This requires matching combination of previously described work in camera positioning and matching moxel calculation granularity to the output moxel resolution.

Moxel Decimation - In order to allow for the real time speed of visualization (in addition to the real time calculation that system is already allowing), it is necessary to simplify existing output once per-moxel calculation was complete. Please note that this step is required only in the simulation - if we are controlling real physical system, this task wouldn’t be required. This part was not hard to implement, but it is interesting that visualization of even very simplified system takes much more time then calculation of the final system many times its size and that some aliasing effects related to the PRE control were uncovered during this process.

Statistics Counting - In order to demonstrate that this system has reached its performance goals, there was a need to implement performance statistics based counting. There was no scientific challenge here but this part of the system was required in order to validate research results. Furthermore, with the imperfection of the computer clock, we had to implemented engineering system that, as a part of the build process, measures performance and clock resolution and fails the build if required performance parameters are not met.

Tactile Dithering - In the case of control of the MEMS system, what is the best way to present “tactile textures” of the material, and how to scale it on the very large number of moxels that are needed to simulate fine grained surface? This issue has two components - how to provide dithering fast (which will be described as a part of this PhD work) and what is the best dithering pattern (which is HCI topic that requires tests on the physical system and is outside of the scope of this PhD work).

Moxel Imperfection Handling - Moxels are physical systems, subjects to manufacturing defects. As a part of the BrailleEye [21], variations as large as 50% were noticed in the volume of the thermal element used for the control of the particular moxels, which would imply very high variability of the control signals necessary for two moxels to reach same position. Similar problems (although likely of somewhat lesser magnitude) are expected in larger scale systems. Software control must be able to account for the physical imperfection of the moxel while staying in the confines of the GPU. This was addressed by making modifications of the per-moxel fragment shader.

Moxel Physical Characteristics Mapping - In addition to the aforementioned imperfections, moxels posses physical characteristics that have to be taken into account when we are addressing their positioning. Challenge is how to run per-moxel calculation without having to do it on CPU. This was addressed by using the fragment shader.

The Fragment Shader - has to take into account all of the previous per moxel based phenomena.

MEMS Related Filtering - Challenge here is how to present a surface that allows for the best discrimination using only a sense of touch, while allowing finer grained elements to preserve surface feel. Some basic techniques [75], [76] that lend itself to the GPU based acceleration will be discussed. With that being said, the challenge identified here can provide ample opportunities for the further research in HCI area once MEMS based hardware is available.

Additional Channels - GPU have limited output capabilities on per-moxel basis (e.g. they are limited on the color and Z-buffer per pixel info). There is a whole array of methods that could be used for communication with the user, including but not limited to mechanical, electical, vibrotactile and heat related [18], [89], [24], systems like tactons that combine rhythm and tactile response to convey information [127], [128] as well as effects like slipperiness in large scale systems [7]. How could we provide output on the multiple channels using only limited capabilities of the GPU? Here, we propose an approach that is a generalization of the approach used in [85].

Aliasing Handling - working with a discrete system, as is the case here,raises questions about aliasing in both spatial and temporal domain. Most of the anti-aliasing work in literature was concentrated on effects of the aliasing in the signal processing, sound, image and visual animation [34], [129], [130]. This brings the question: do same principles translate to the micro and macro scale of the PRE based systems? Although final resolution of that question is definitely in the area for further research and outside of the scope of this PhD thesis, we can demonstrate that macro scale systems may require different approaches than the antialiasing techniques used in image processing.

Physical Safety - this topic is relevant to the large scale PRE since there is an unavoidable amount of failed moxels in a system of this size. The thesis explores the issue of how some of the anticipated moxel failure modes including moxels stuck in the fully retracted and partially extended position could be addressed, as well as what are safety implications of the 3D geometry used in the large scale Holodeck type of the environment.

Catoms Integration - physical environment based exclusively on the moxel actuation cannot present 3D shapes that could not be described with the 2D discrete function. However, mechanism for the combination of the moxel and catom based environments in the presence of the GPU based control of the moxel environment was devised.

4.2 Engineering Consideration

In addition to all of the previous scientific considerations, the simulator of the system was a significant amount of work on the engineering side. This section will briefly describe what work was done on the engineering side of this project:

1. HoloSim extensively uses unit testing [131] and test driven development [132]. It is setup in such a way that as a part of every build a set of tests is run and if those tests fail, build fails.

2. Performance goals are included as a part of each unit test (e.g. as a part of build, program would test that performance goals stated in the PhD thesis proposal were met and fail if not).

3. Program is written on OS X in XCode. Portability on the iPhone 3GS and later appears relatively simple to accomplish, but has not been done yet. This might be relevant if we are to use this software later in a mobile scenario.

4. Open source libraries were used when available for the supporting tasks, but code relevant to the demonstration of core contributions of this PhD thesis is not using third party libraries (except the ones that are shipped as a part of OS like OpenGL library, math library, GUI library etc). Open source libraries were used to reduce the burden of mundane tasks only (e.g. loading Collada [133] models).

5. The code is fully documented and automated generation of documentation is part of the build process. This was done by using an automated documentation generation tool [134].

6. High level architecture was accomplished by using MVC (Model View Controller) design pattern [135], with the separate model classes responsible for calculation and decimation of the moxel based model.

Overall, HoloSim is designed, implemented and documented using the best current engineering practices in the industry.

Chapter 5 - GPU Based Control of Large Scale Constrained Motion PRE

Software control of the Holodeck PRE will be done by employing high-level system architecture described in the author's earlier work [6], [7] depicted in Figure 10. The architecture is based on utilizing the capabilities of the GPU for dual purpose - graphics rendering (on the left side of the picture) and moxel control (on the right side of the picture). 3D Model data describes the surface being used by both pipelines. Software algorithms proposed for calculation of the moxel position using Z buffer and the fragment shader were tested in simulation, and simulation was used as a means to prototype and test various concepts prior to building complex and expensive hardware. We will describ a simulation environment in a separate section.

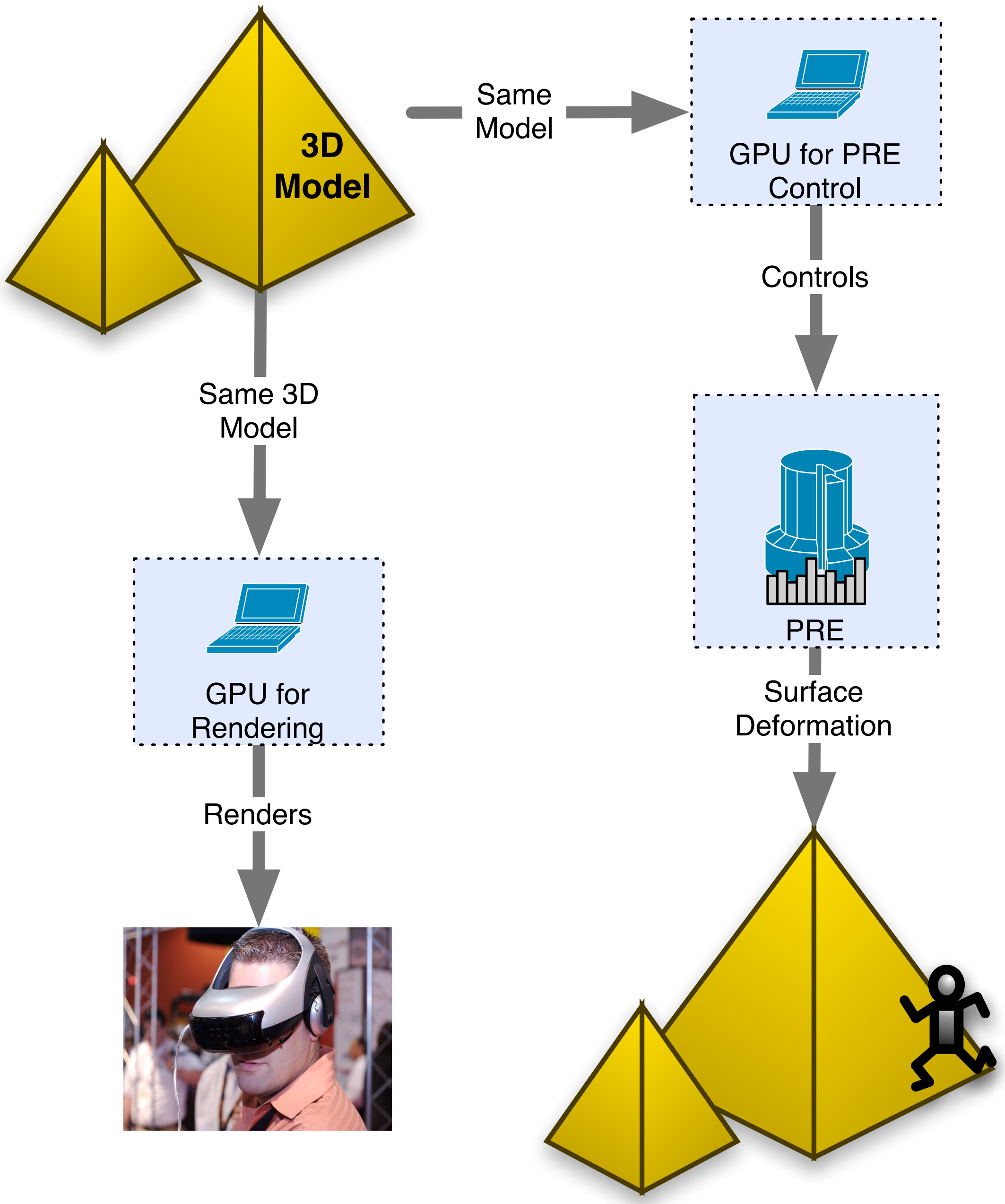

Figure 10. GPU Based Control of the PRE. Used [136] for helmet picture.

The author's earlier work [7] states that:

"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 the users 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 system creates an immersive environment [7] by presenting an image to the user wearing a virtual helmet [28], and at the same time deforming terrain under the user's feet in order to conform to the same terrain image that user sees. We have shown in this thesis that we can use existing programs that generate 3D images and modify them to control Holodeck system's ground deformation and slipperiness of the terrain.

The example in Figure 10 demonstrates deformation of only one plane, e.g. the ground plane, but the concept is straightforward to extend to deforming other edges of the room besides the floor, e.g. ceiling and walls. Thus, the physical rendering engine may be drawing as many as six different surfaces (floor, ceiling and walls in the room).

Figure 11 takes a deeper look at how our software uses a typical GPU's rendering pipeline. We are able to use elements of that pipeline for the GPU based control of the Constrained Motion PRE.

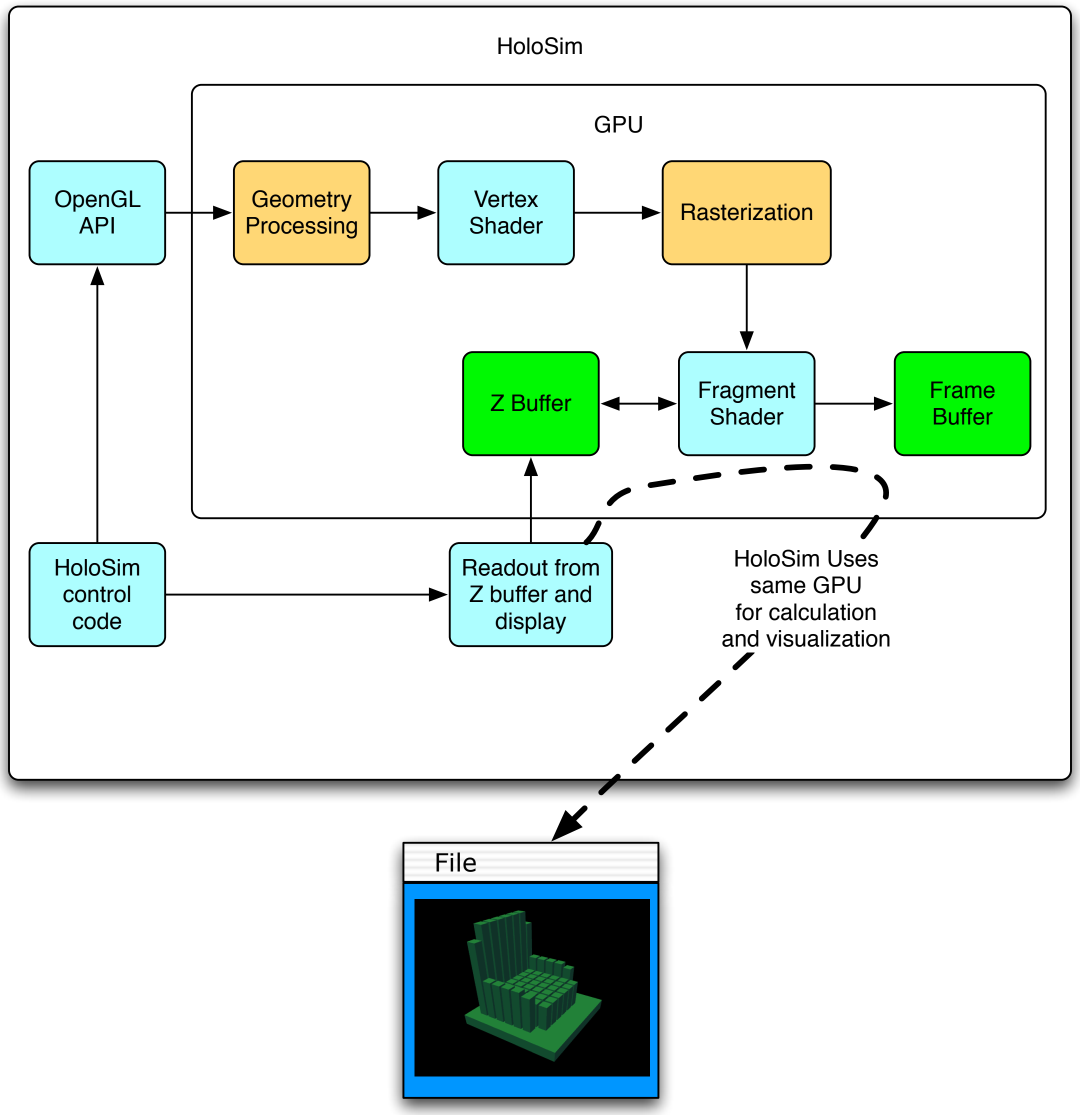

Figure 11. Simplified version of how our software, HoloSim uses programmable pipeline available in most modern GPUs. Only relevant stages of the OpenGL pipeline areshown (the author's illustration based on [137]). Green color identifies buffers used, cyan identifies code and olive are relevant stages in the OpenGL pipeline.

Most of the elements of the previous graphics pipeline are hardware accelerated on modern GPUs [138], [35], [36]. This thesis shows that we are able to use most of the elements of the graphic pipeline without requiring any functionality that is not already a part of the existing rendering pipeline:

1. By the appropriate transformation of the viewport (described in detail in the next chapter) we are able to use all elements of the existing graphic pipeline prior to the Fragment Shader in the same way any prior rendering engine will be using them.

2. The fragment shader allows us to run SIMD [139] style parallel programs that will be executed on a per-pixel basis. We will use this functionality in the next chapter to account for the physical characteristics of the movable element.

3. All elements of the rendering pipeline after the fragment shader will be used without any changes, as in any other graphics program.

4. At the end of the frame being rendered, we read the content of the Z buffer of the rendered image, and use it in the next section.

Although at first glance you might consider using OpenCL [140] and CUDA [141] for the calculation of the moxel positions we will show that a more natural approach manipulates a viewport directly and as such eliminates a need for, or benefit from, OpenCL and CUDA due to its ability to directly control and access Z buffer.

5.1 GPU Based Control of the Moxel Displacement in Constrained Motion PRE

When it comes to software algorithms, this project is basing the basic calculation algorithm on the approach taken in the author's earlier work [7], [6].

In the following discussion, lets assume for simplicity's sake that there is one planar plate on the floor, and that its pins are perpendicular to the plate, rising up from the floor as shown in Figure 12.

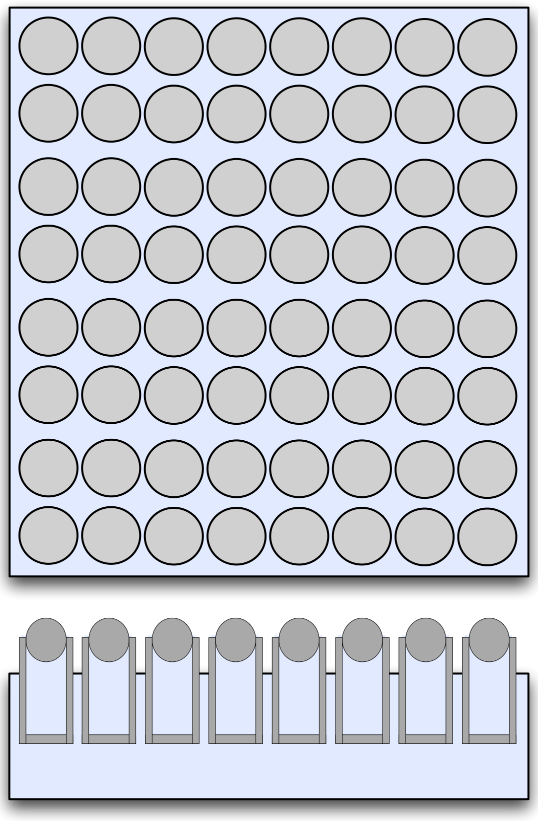

Figure 12. Physically Rendered Environment.

5.1.1 Use of the Z Buffer for Displacement Calculation in Constrained Motion PRE

The author's earlier work [7] proposes an outline of the algorithm for moxel control, where the viewport is positioned so that the combination of camera position and resolution, in which every moxel matches one pixel, allows us to read moxel displacement directly from the Z-buffer[19].

That situation is shown in Figure 13. Looking at the plate from above, Figure 12 shows similarity with the array of the pixels on the computer screen. This brings the following observation, as elaborated in Figure 13:

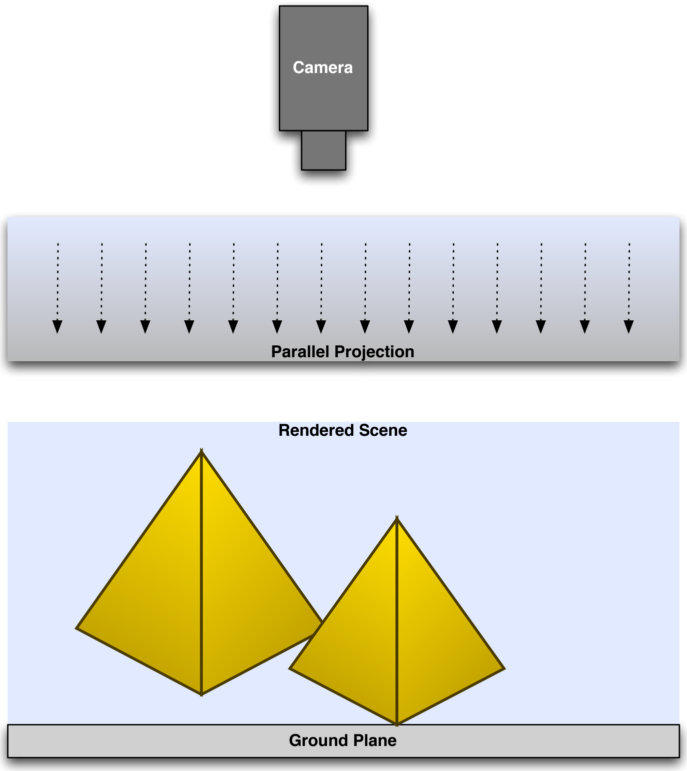

Figure 13. Viewport Position while calculating moxel position in PRE using GPU.

[19] Note that it is possible to position camera both above and below the scene, and that moxel displacement would depend both on how we interpret Z buffer value as well as the behavior of the fragment shader - whether smaller or larger value of the Z occupies Z buffer after the test.

Suppose that we render an angled plane in the room, as shown in Figure 13 - the user point of view.

Now, let’s look from the point of view #2, with the orthographic projection on the plate, with the viewer positioned above the plate.

Algorithm Moxel Position Calculation:

1. Suppose that we consider plate position to be at the screen position, and the resolution is set so that there is one to one correspondence between each moxel and (non-supersampled) pixel.

2. If the previous assumptions are met, then rendering of the plane and reading back the Z-buffer for each pixel would determine how much each moxel needs to rise.

3. The vertical distance that moxel needs to move should be equal to the value in the Z-buffer which corresponds to the XY position of the moxel.

An important outcome of these observations is the fact that they allow us to use hardware acceleration available in conventional GPUs to calculate the moxel position, as well as standard APIs like OpenGL [124], [124] for controlling these calculations.

We will use the frame buffer and render buffer functionalities in OpenGL [37] for rendering the image using this algorithm[20].

With a simple visual scene a typical GPU easily reaches 70Hz resolutions on a million pixels (and earlier works in computer graphics field [37], [138], [36], [35], [34] shows it to be much more efficient for this purpose than the CPU), the GPU is ideally suited for the software control of the large number of moxels.

In order to export results of the calculation from the GPU, we are using OpenGL renderbuffer extension and readout from the OpenGL framebuffer object[21].

[20] Benefit of this approach is that it allows us to set buffer resolution independent of the current display hardware, but is otherwise identical to the normal OpenGL output to the display.

[21] Author would like to thank to Professor Willem Schreuder for suggestion to use this functionality.

5.1.2 Use of the Fragment Shader for Adaptation to the Physical Limitations of the Moxels

As noted in the author's earlier work [7], each moxel is a physical entity, and as such, subject to physical laws of inertia that can be accounted for in the fragment shader. Suppose that moxel has a response curve for its position shown on Figure 14. For the calculation of the final position, we could use GPU.

But what about intermediate positions? What if we for example have a response curve for the moxel speed like on Figure 14? How can we calculate intermediate positions of the moxels?

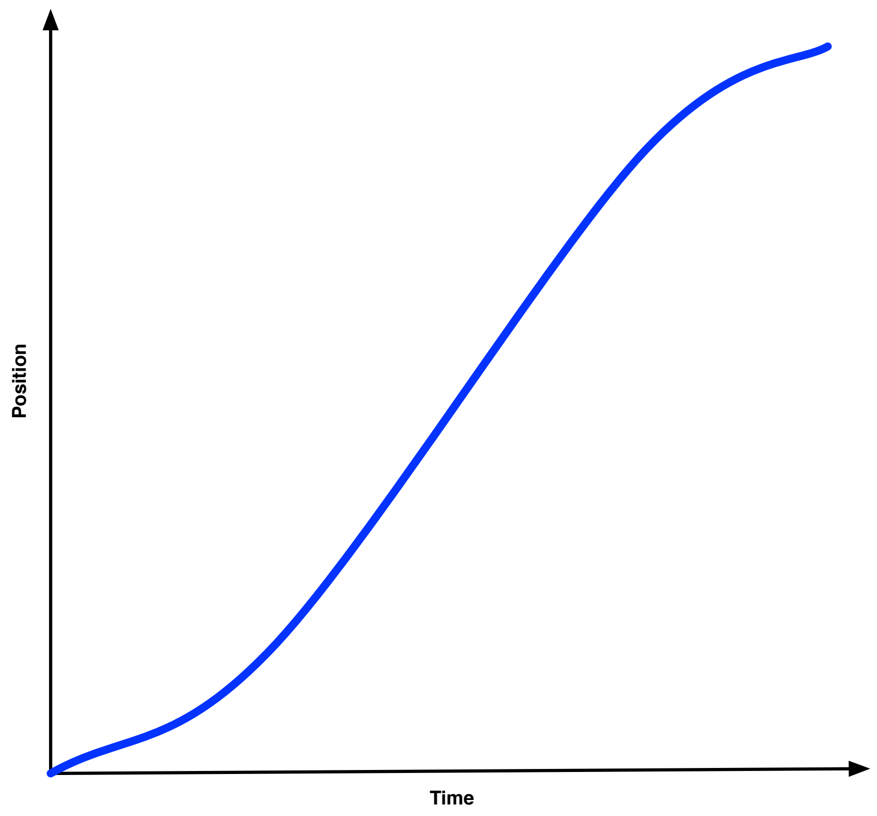

Figure 14. One example of the response curve describing physical limitations of the moxel position over time. Picture is modification of the figure from the author's earlier work [7]

The author's earlier work [6] proposes use of the pixel (fragment) shader present in modern GPUs and performing quantization of the function (presented in the Figure 14) into a texture[22] and passing the texture to the fragment shader with a uniform variable representing the interpolation step that we want the fragment shader to perform. From [6]:

> *"The fragment shader would then perform the necessary interpolation, store the intermediate position of the moxel in the color buffer or intermediate texture (in the situation that we need to account for the intermediate position in the successive shader iteration), and actuate the moxel with the appropriate displacement."*

Previous procedure addresses correcting for the physical limitations of the moxel. Note that (as it will be explained in the section "The Fragment Shader Adoption for Moxel Imperfection Handling") the procedure can be modified to account for the physical limitation of the individual moxel too (e.g. to address manufacturing variation between different moxels). Furthermore, it is possible to combine previously described approach with the information about current position of the moxel (as for example height map of current height info as described in [85]) or external forces applied to moxel (e.g. from the actuation mechanism as in [33], [16], [78], [17]) to have moxel texture take into account moxel's previous position, physical characteristics of the moxel, manufacturing variability, user position and external forces in the fragment shader calculation. This allows us to address general physical characteristic of a single moxel completely on the GPU.

[22] Although originally 1D texture was proposed, packing 1D arrays in 2D texture better utilizes GPU texture caches [38].

In addition to addressing the position of a single moxel, it is important to note that sometimes there are issues with cross-talk of the moxels, in which actuation of one moxel could cause surrounding moxels to be actuated (similar to crosstalk issues described in [17] and [142]). Although the individual fragment shader can't account for cross-talk, rendering of the moxel positions to the intermediate texture allows for a subsequent rendering pass in which we use the fragment shader again on the texture to account for the position of the surrounding moxels in the gather step [38][23].

In effect, to address cross-talk issues, we use multiple rendering passes, in which the first rendering pass renders to a texture and subsequent rendering passes read that texture and process it as an image processing operation, similar to e.g. blurring operation in computer graphics [143], [129], and we could use similar approach to processing including derivatives of the pyramidal methods described in [143] to address relation between closely spaced moxels among which blurring did occur.

Note that as this is a GPU localized operation, it would also account for the concerns about CPU impact of this correction in software mentioned in [17].

At the moment, we believe there are no significant quantization issues in relation to the positions of the moxels in Z direction, and, if they exist, they are more likely to be artifact of the mechanical implementation of the system chosen than its software control.

[23] Although not implemented in the current simulator.

Further sections describe various other aspects of the software control on the GPU. Per the scope of this PhD research project defined at the time of the thesis proposal, algorithms proposed in the subsequent sections were not included in HoloSim.

5.1.3 Adoption of the Fragment Shader for Moxel Imperfection Handling

When working with physical systems, we have to take into account the fact that there might be variations between the moxels that are results of the manufacturing process. As just one of the examples, when the prototype of the BrailleEye project was made at CU Boulder [21], differences between the amount of thermal element used to power individual moxels had variation as high as 50% (in this case as a result of the drilling tolerances). Although improvement of the manufacturing process is certainly an interesting topic, the fact remains that there is always some variability present in manufacturing of physical objects[24]. Furthermore, from the perspective of the whole integrated system, if differences between moxels could be compensated for with the software, we will be able to tolerate higher variability in the manufacturing process.

In order to control for moxel imperfection, we will assume that:

1. It is possible to determine moxel imperfections on per moxel basis (e.g. during physical system's calibration test).
2. That information about moxel imperfection was collected for every moxel in the system.
3. That information about moxel imperfection could be described as an array of numbers, e.g. X = [X0, X1, ... , XN].
4. That it is possible to correct for the moxel imperfection knowing that set of numbers from the previous point.

[24] Well known effect in any manufacturing operations. See e.g. [144] for the descriptions of this effect.

As an example, if we have varying mass of moxels that are controlled by an electrical motor, it is possible that the motor would have to be run with higher power for a moxel that has larger mass in order to achieve same acceleration as the moxel that has smaller mass.

In order to address imperfection, we need to pass information about moxel imperfections to the fragment shader, so that it can reconstruct the information and address it. In the general case, we would need to pass an array of the values corresponding to each moxel to completely describe its behavior (e.g. moxel could have a problem only in particular vertical intervals, and we would need to pass those intervals and nature of the limitation info). As moxels are a 2D structure and we need to pass 1D array to each, we have a 3D structure that needs to be passed to the fragment shader unit.

There are multiple ways to describe this structure. If the length of the arrays to be passed to the moxel is similar across the moxels, then we could use 3D texture (see e.g. [37] for description of 3D textures) in combination with the fragment shader. With the previous approach in mind, the following algorithm will be used:

<u>Algorithm Moxel Imperfection Mapper:</u>

1. Setup a 3D texture in which XY plane will be of the same size as moxel field

2. Set Z coordinate of the texture to be equal to N + 1, where N is maximum of cardinality of set X across all the moxels.

3. For each moxel at X, Y fill up 3D texture's Z dimension in the format of ZF = [Code, X0, X1, ... , XN] where

    a. Code discriminates how to interpret the remaining numbers in array (e.g. if some imperfections could be described with only one number and other imperfections could be described with two numbers, then we can have a code as 1 and 2 for previously enumerated moxels).

    b. X0, X1, ... XN is array describing moxel imperfections.

4. In the fragment shader during moxel calculation, sample the 3D texture at the XY coordinates corresponding to the moxel coordinates in order to extract imperfection information corresponding to each moxel.

5. Use the fragment shader to compensate for the physical moxel imperfections based on the previous information.

Previous algorithm would work for situations when the size of the moxel field in XY direction could be represented with the 3D textures. If hardware capabilities don't allow for the creation of the 3D texture that is of the same size as moxel field size in XY direction, we have two approaches:

1. Use multitexturing capabilities to bind multiple 3D textures [37]. This will be helpful. However, the maximum number of textures is still limited so we could

still run into the problem that the moxel field is larger then what texturing capabilities would allow addressing in a single pass on the fragment shader.

2. Use multiple rendering passes - have one pass render only portion of the moxel field that is big enough to fit in texture space we have, then second pass address adjacent part of the moxel field.

3. Employ different packaging form - e.g. store imperfection arrays consecutively in the 2D texture, and have another 2D texture that is holding indexes to the first texture (this is based on the handling of the sparse data structures, as described in [145] and [38]). This is preferred solution when length of the arrays could vary a lot [25].

Clearly, which one of the previous approaches we chose would determine whether it is possible to address moxel imperfections in a single pass or in multiple passes. If we need a single pass to correct moxel imperfections, the question is: should we combine that correction with the moxel calculation step, so that we need in total a single pass to calculate moxel positions and correct for imperfections?

When making that decision, the following has to be considered:

1. There are clearly considerations with the capabilities of the graphics cards - e.g. if we are to combine two passes that need textures in single pass, then we

[25] This approach inspired by the multiple levels of the texture presented in the Pyramidal Array approach [143] and gather approach [38] for splitting info in various positions in the texture . The fragment shader reads array as “subimage” that is part 2D texture.

would reduce total number of textures that each pass can use[26]. So engineering considerations might impose that we have a separate pass here.

2. Correction of imperfections has to be the last point in the moxel calculation pipeline because moxel imperfections have to be taken into account after the program has finished all the calculations for the moxel position. If we already need to have multiple passes for other reasons[27], we might be able to combine moxel imperfection correction as last pass (or consider combining it with the last pass in the pipeline).

The reason why moxel imperfection correction has to be the last point in the pipeline is that moxel imperfection potentially has to take into account all the things that moxel does. So if we apply moxel imperfection correction and then have a subsequent rendering pass that modifies moxel position, then previous moxel imperfection corrections might be invalidated.

As previously noted, it is possible that we might need to address cross-talk between moxels in which position of one moxel impacts nearby moxel positions [17], and additional rendering pass on the texture of rendered moxels would allow us to do that.

[26] As max number of textures is typically limited. See [37] for details.

[27] We would discuss reasons why additional passes might be needed elsewhere in this thesis, when we describe full software pipeline for the moxel based systems.

5.2 Integration of the Proposed Algorithm with the Existing Rendering Pipelines

As discussed in the author's earlier work [6], [7], once we know how to render a given surface in Holodeck, the question remains how do we know what surface to render? In a broader content of simulation, Holodeck is simulating physical ground in the context of a bigger simulation environment, where Holodeck simulates ground planes and the image of the remainder of environment is displayed to the user.

Could we integrate generation of the terrain image with the control of the terrain that the user is standing on, whilst using the same code to control both? Is it possible to reuse Holodeck with existing graphics program so that Holodeck shows the user's perspective from the ground plane?

One additional problem to account for is that some objects residing on the ground plane, e.g. the ball in Figure 15, could not be easily rendered in the Holodeck environment, as noted in our earlier work [6]. If the top surface of the ball is rendered using the Z-Buffer method, then we would fill in the shaded area in the gap beneath the ball.

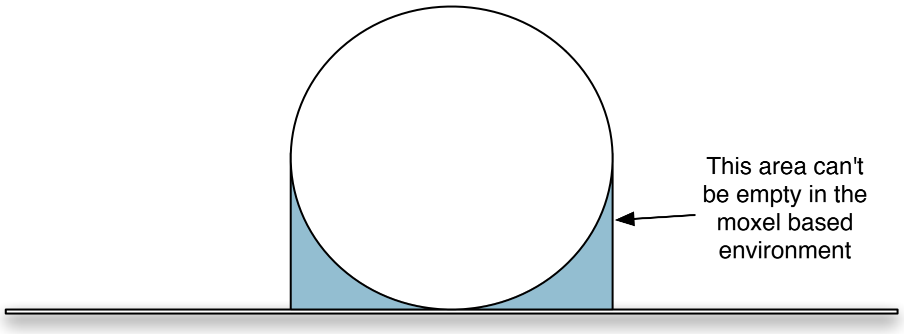

Figure 15. Example of the physical shape that can't be rendered in PRE.

In some use cases like e.g. combat simulations, it is likely that the user might try to take cover with the area directly above him protected. Although this might be partially addressable by providing moxel system in which moxels are both on the ceiling and the floor, this will not work on all surfaces as it allows us a maximum of two z buffer values per any area in the XY plane and remains a consideration in use cases such as combat simulations[28].

Another related problem is that some objects (e.g. grass or bushes) could be on the ground, but we could not stand on them - the user's weight is supported by the ground in the outside environment, not by the grass on that ground [6].

That is, the problem is how to effectively distinguish the ground plane from the objects on the ground plane. Based on our earlier work [6], 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 modifications needed to help us distinguish between the ground plane and objects resting on its surface are simple. In fixed OpenGL pipeline (without changing geometry description in vertex shaders), we need to add just two API commands that would identify the beginning and the end of the code section within which all rendered geometry will be considered to be part of the ground plane. Only these OpenGL commands will be physically rendered into the Holodeck's ground

[28] See later section on combination with catoms for description of how this problem could be resolved in the system that is combining catoms and PRE type of the environment. The author would like to thank to the Professor Willem Schreuder for pointing use case of the combat simulations to me.

geometry. An example would resemble the following (example repeated from the author's earlier work [6]):

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

// All subsequent drawing will appear on both the screen and in physical Holodeck
hglGroundBegin();

// ... all geometric objects that are part of the Holodeck's ground plane ...

// Drawings following this command would not appear in the Holodeck, but only on the screen
hglGroundEnd();
```

So, how do we render the picture shown in Figure 15? As is apparent, in the Holodeck environment, this situation cannot be rendered correctly. As discussed in our earlier work [6], by providing the ability to the programmer to control what is part of the ground plane and what is not, we could declare the ball to be a part of the ground plane (in which case the shaded area will be incorrectly rendered). Or we can declare that the ball is not a part of the ground plane (so only the ground plane is rendered, and we ignore the physical representation of the ball or simulate it with a separate haptic interface). In effect, Holodeck would draw only those surfaces that could be represented by 2D functions.

It is an interesting question, how to best intercept OpenGL calls between hglGroundBegin() and hglGroundEnd(). One possibility is to use wrapper library around OpenGL calls, so that it would intercept OpenGL calls and send them to both the Holodeck and graphics screen. Another possibility presented in the author's earlier work [6] is to use AOP [113], [112] to get the same effect. In effect, we would consider

HoloSim to be a cross-cutting concern for the visualization, and we would use AOP advise to centralize in one place in the code interception of the OpenGL based calls, and use one pointcut to “select” all OpenGL calls to be advised with the previous advise.

5.3 Slipperiness of the Terrain in PRE

Let's take a closer look at a single moxel, as detailed in Figure 16. The author's earlier work [6], [7] sets a basic outline for the capabilities of a moxel, by proposing that we equip the top of the moxel with a surface whose coefficient of friction can be varied (e.g. ball with the brake), so that we can control both slipperiness and the height of the surface, allowing for the simulation of the slippery terrains like e.g. inside of a cave.

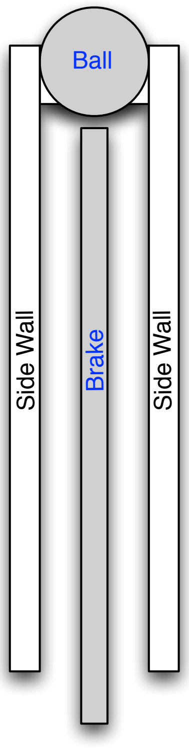

Figure 16. Moxel capable of simulating slipperiness. Image reused from the author's earlier work [6].

The author's earlier work [7] points out that we need more information to control slipperiness of the terrain, as information about slipperiness is not presented to

OpenGL and proposes use of the fragment shader to vary slipperiness of each ball on top of each moxel in accordance with the value of a “slip texture” for that moxel.

Slipperiness of the terrain is obviously scene specific but the fragment shader portion of the slipperiness calculation is not, so the same shader could be used for all the user programs. As a result, the user program would need to be modified to pass additional texture information for each surface on which the user is standing. This is typically not a problem unless the number of already used textures is limited by the texturing capabilities of the card.

As discussed in our earlier work [6], 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 would then be used by the Holodeck's fragment shader. In the long run, we could modify material libraries in the modeling packages typically used by artists to specify a look and feel of objects [146] (which typically include multiple texture maps). The modifications could be extended to include a slipperiness map of the surface, too. This is somewhat similar to the concept of the library of the haptic recordings that is mentioned as one alternative for haptic rendering by [42].

5.4 Controlling Additional Physical Properties Using Proposed Approach

Slipperiness is just one example of the additional physical properties that can be beneficial for the user immersion. There are many others:

1. By modifying the elasticity of the terrain on which the user is standing we could simulate the sensation of being on concrete or sand. This relates to the perceived elasticity of the moxel. GaitMaster [44] proposed varying resistance on the pedals user is standing on to provide impression of moving over different virtual terrains, and we are proposing extension of this approach to moxel based systems.

2. Temperature of the moxel surface.

3. If the users skin is in direct contact with the moxel, additional tactile feedback can be provided by the means of tactile, electrical, vibration or ultra-sound stimulation [18], [67], [68].

The above examples could be calculated using a similar approach already outlined for the calculation of the slipperiness. The question is how to provide additional outputs in our framework, as each of the aforementioned could have values coming from the analog scale.

Our solution here is to remember that in addition to the Z-Buffer information, each pixel of the GPU would have RGBA information with 8 bits per channel, in effect allowing us to output 32 bits per pixel.

We could divide these 32 bits among the values that need to be outputted - e.g. if there are two values that could be outputted, we can allocate 16 bits to each, ignoring the color channel (R, G, B, A) boundaries and providing 16 bit quantized output per channel. Obviously, the more channels we have, the less quantization levels would exist per channel.

5.5 MEMS Based Tactile Devices

Previously enumerated approaches could be used for the control of MEMS based screens in situations where we are rendering the user interface elements or computer generated images.

In addition, in situations where a visually impaired user needs to perceive a large number of pixels all at the same time (e.g. by touching MEMS based screen), it is possible to show not only computer generated images, but graphics and video streams [114]. The combination of the display of the graphics and video streams goes far beyond the ability to present only Braille based alphabet [147], because it allows the visually impaired person to perceive not only text but graphics as well (one similar system projecting video image on the torso is described in [73]).

In the case of tactile displays, it is beneficial to be able to control a large number of MEMS elements.

1. Human skin is very sensitive to the difference of the small surface structures with point discrimination threshold (TPDT) of 2-3mm at the fingertips, and 7-10mm at the palm of our hands. Experiments were conducted [117], [116] indicating that gratings with a period of 0.7mm to 1mm could be distinguished with 75% correct level when the difference between the periods is in 5%-10% range.

2. Work was done on sliding surface over immobilized hand and it was found that perceivable differences in the period along longitudinal direction of 0.64-0.8 mm

and that frequency discrimination thresholds for the vibrotactile displays will be ranging from 16.5 Hz to 20.0 Hz (100 Hz standard) [77].

3. Furthermore, [31] suggest that spacing of less than TPDT/2 is necessary for the device to be able to produce any tactile feeling (under conditions when positioning of the hand could be precisely controlled). It is likely that even higher spatial resolutions will be needed if the position of the hand could not be precisely controlled.

4. Previous work investigates elements with a diameter of 0.5mm, with 250Hz movement frequency for presenting texture feedback [86], [68].

5. Previous examples indicate that MEMS systems with elements smaller than 1mm in diameter are likely to be needed in at least some circumstances for the realistic presentation of tactile surfaces.

6. On the tip of the human tongue, spatial resolution is shown to vary between individuals in the range between 0.254mm-0.762mm [32].

7. Experiments performed by [115] and by measuring neurological responses in the anesthetized non-human primates [148] indicate that if humans are allowed to move their fingers across a surface, they can perceive differences in the spatial distance between 0.65mm raised dots of only 20-40 microns (measured by threshold of distance that is needed to achieve 75% correct discrimination between two surfaces with different frequencies)!

8. A survey of various studies of tactile discrimination done in [27] concludes that the threshold for the vertical indentation for amplitude in vibration for four types of receptors in our skin varies in the range of 0.01 to 40 microns, for the sensitivity of the receptor occupying various frequency ranges from 1 to 1,000Hz and highest peak sensitivity in RAII (PC) type of receptors being at 250 Hz. Other studies showed vertical sensitivity in 1-3 microns with different experimental setup and associated frequency range [149].

9. Touch has much lower bandwidth than sight (four orders of magnitude below sight), with about 100 bits per second but is about five times faster then sight [18]. Consequently, although we need to control significant number of moxels prior to touch and might need to control high frequencies of the vibration, for the purely assistive technology uses with users that are unable to see, refresh rate bandwidth while user is "scanning" doesn't need to be high, provided that latency of the page refresh allows for the complete page change prior to repeated touch[29].

10. In order to benefit from the economy of scale, assistive technologies should be useful for sighted users, too.

Taking all of the above into account, an argument could be made for MEMS elements as small as 20 microns in diameter, but we will be conservative in our estimate and assume they need to be 500 microns to 1mm in size. Even with that assumption,

[29] Ideal interface for visually impaired people proposed in [24] has 76,800 elements on the size of 32cm x 24cm and refresh rate below 10s. However, for people with the normal sight we would need much higher refresh rate, as well as to take into account not only single point discrimination threshold but also surface period discrimination threshold as demonstrated by [115].

there is a need to control ~50,000 to ~200,000 MEMS elements (for MEMS elements varying between 0.5mm and 1mm in diameter) in order to cover surface of the size of Kindle DX (including both screen and enclosure) with these elements [30].[30]

Large surfaces that could be touched with both hands have an advantage in recognition of haptic images [25]. In the past, there have been attempts to provide a small tactile surface that could show a tactile picture to one finger, and combine it with a tablet that uses a pen and the other hand to point to a particular location in the picture. So far researchers have found that users have significant problems with shape recognition in those scenarios [25]:

> *"The aggregation process is too difficult for them in comparison with haptic images: for taxel-based information only a small part of the digit skin is involved in the reconstruction process, while for haptic images all the hand is used."*

In the frequency domain, it is unclear how often elements need to refresh the picture. Blind users need substantial time to inspect a single picture [24], while sighted users would expect that tactile picture be synchronized with the visual picture. With 16.5-20Hz vibration discrimination thresholds, it is reasonable to assume that 20Hz should be a lower bound of frequency with which we need to refresh the display if we are to use vibrotactile feedback for blind users. Additionally, tactile displays that are logical extension of the Dynamic Displays [89] and TeslaTouch [5] idea (which are to be used by sighted users to get tactile feedback on a touchscreen phone) might need even higher refreshment frequencies.

---

[30] Author would like to thank to Professor Clayton Lewis for pointing him toward Psychophisical research in relation to this project.

Alternatively, it is possible that some systems might combine vibrations and displays in the same system, necessitating calculation and control of both displacement and vibration in our system.

This goes to show that we need an ability to provide software control at the rates of the hundreds of thousands and possibly even millions of elements per second even with very conservative set of assumptions. Those assumptions being, we need to reach only 0.5mm moxel size and the very modest frequency of 20Hz.

As we can feel much finer textures then TPDT/2 when the position of the hand cannot be precisely controlled (as is the case for the proposed assistive devices), it is likely that an even higher number of elements might be needed. This shows that there is a strong benefit from software that can scale control of millions of elements per second. In other words, assistive technologies strongly benefit from our proposed approach.

In order to present graphics as applied to tactile devices, we have used the following approach:

1. We define displacement of the moxel as a function of color of its corresponding pixel. The best mapping from color to the tactile pattern would likely depend on the exact physical implementation of the mechanical system used for the tactile rendering and as such is an area for the future research. There is a significant body of research performed on the topic of the human perception of tactile systems that HCI based research of the color mapping could build upon, including [115], [67], [67], [127], [27], [150] and [24].

2. Due to the mechanical limitations of different devices, it would be beneficial if we could deal with the situation where moxels have only a limited number of states they could occupy (e.g. binary, “up” and “down”). This in turn could become an issue when we want to map large number of colors in graphic image to moxels.

Analogous problems were previously encountered in computer graphics, when a color picture had to be shown on a monochrome screen [34]. The solution in that case was dithering, where groups of black or white pixels are arranged in such a way that they provide perception of shades of gray, as shown on Figure 17:

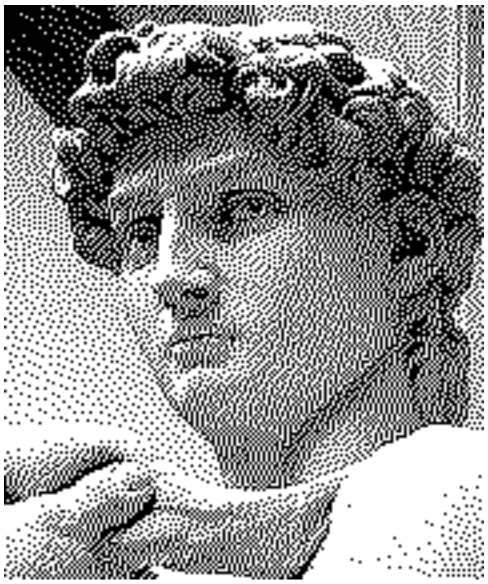

Figure 17. Michelangello’s David rendered using Floyd-Steinberg dithering [151].

Inspired by that approach and haptic recordings as discussed in [42], we can use dithering on a group of MEMS elements mapped to the single pixel in order to map different colors. In effect, we are defining a function that maps the color of the pixel on screen to the texture emulated by the group of the MEMS elements.

This dithering could be performed in the fragment shader in order to determine the color of the pixel using the multiple fragment shader units which are available on the GPU in parallel, ensuring massive scalability of the calculation approach. In effect, the following approach will be taken:

1. The fragment shader program will be responsible for the calculation of the MEMS element positions on a per element basis. In addition, it is likely that the image enhancement algorithms such as edge detection, blurring and segmentation will be beneficial for better tactile perception of the image [75]. GPU is capable of accelerating each one of these tasks.

2. Initial color image will be passed as the texture to the fragment shader.

Currently, it is unclear whether the current dithering algorithms accepted in computer graphics offer the best approach to performing dithering on the picture. Further HCI work will be needed in the future to make that determination. For the purpose of this work, we are focusing only on the moxel software control and scalability of that control. The question of most appropriate dithering algorithms for the touch is left as an area for future research once we have available physical MEMS systems with fine enough resolution.

Finally, note that the dithering based approach and displacement based approach can be combined if the MEMS based device allows for more then two states of the MEMS element, as for example the tactile devices presented in [71] are capable of.

Software algorithms for GPU control of MEMS based systems will be described in more details in the subsequent sections.

In relation to the MEMS based devices, it is interesting to note that there are recently proposed alternatives in the form of systems using electrovibration to provide the sensation of differing surface friction to the sliding finger, as described by [5]. Although these systems are very different from the MEMS based systems in the respect to the physical implementation of the system, from the perspective of the software control, they offer a number of similarities to the MEMS based systems:

1. Feeling of different surface frictions is just one example of the tactile mechanism for transferring information to the human finger. Algorithms for the MEMS based system could be easily adopted, as electrovibrational effects are just another output channel.

2. Need to account for the different detection thresholds for different electrovibration levels, as discussed in [5], is just another form of accounting for the physical characteristics of the moxel. We discuss the particulars of the moxel physical characteristics elsewhere in this document.

Chapter 6 - HoloSim, Implementation of Algorithms for the Constrained Motion PRE

HoloSim software fulfills two distinctive roles:

- It is a proof of concept of the proposed algorithm for ***Algorithm Moxel Position Calculation***, that also takes into account physical limitations of the moxel[31].

- At the same time, as there is currently no physical prototype of the PRE environment available, HoloSim performs the additional task of the visualization of the results in such an environment.

As a result, HoloSim is not an environment simulator - it is actually a proof of concept, as well as a mechanism for the visualization of the PRE environment. Figure 18 illustrates output from HoloSim, in which a throne is rendered inside of the PRE environment[32].

[31] This is done using simple fragment shader that is using piece-wise linear function describing one possible physical characteristic of the moxel.

[32] Also, note that Moiré pattern would not exist in the real PRE environment. See later section discussing aliasing in the moxel based environment for more details.

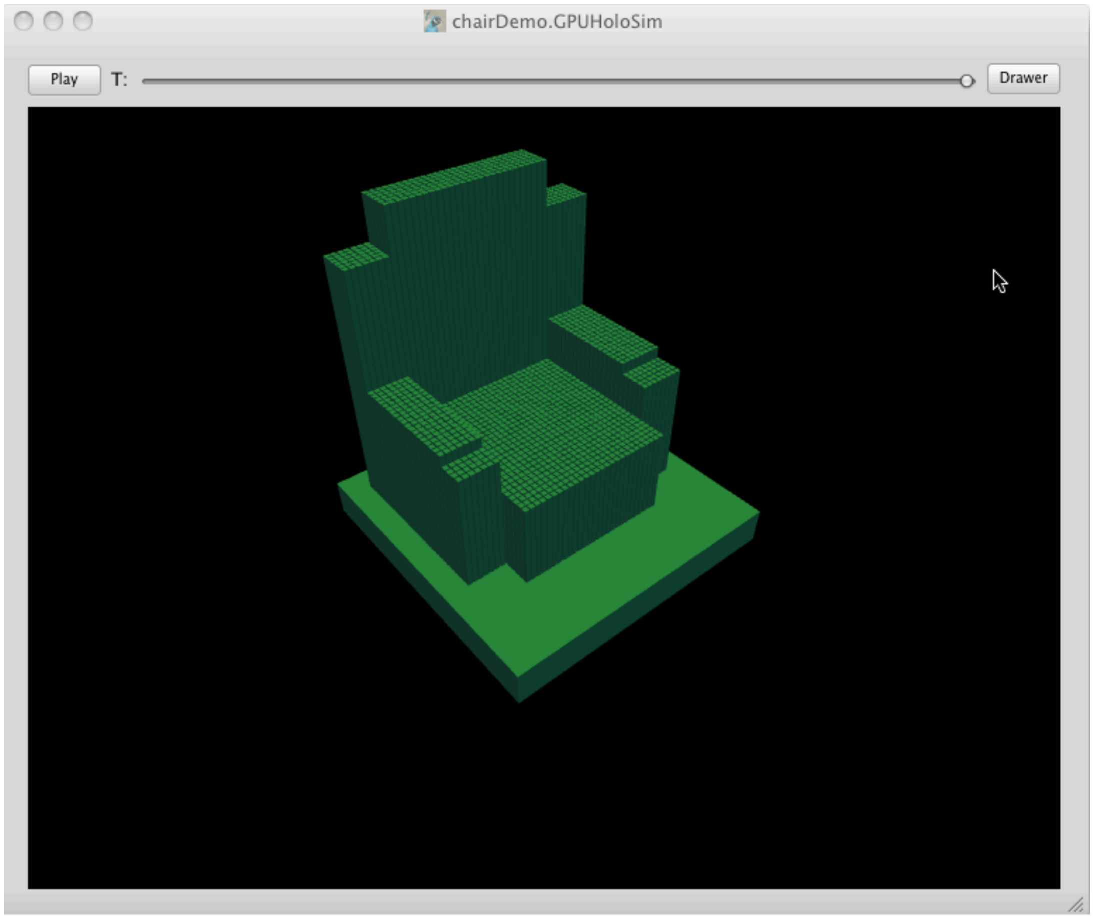

Figure 18. HoloSim, a simulator of the PRE.

HoloSim features following capabilities:

- Support for a subset of the Collada format [133], allowing 3D models to be made in modeling programs[33].

- Ability to calculate moxel positions on GPU using previously proposed algorithm.

[33] Collada is complex format, and it was not needed for the purpose of this project to support all parts of it (e.g. HoloSim doesn't currently use normals, texturing or independent coordinate systems for objects in Collada file). Supported functionality allowed use of Google SketchUp [152] for the definition of the HoloSim models shown in this thesis.

- Ability to take physical characteristics of the moxel into account on the fragment shader. For the purpose of the simulator, we are using the fragment shader performing piecewise-linear moxel time/position control function.
- One channel output of the displacement over Z-buffer values.
- Visualization component that visualizes position of the moxels in the 3D space. It also provides GUI for control of the viewport.
- Performs tracking and presentation of the moxel calculation statistics
- Simplifies visualization of a large number of moxels, by grouping multiple moxels into the single visualized moxel[34].
- Provides basic loading functionality for model description.
- HoloSim code is written in the C++, and GUI portion of the HoloSim is written in Objective C, using Apple Cocoa framework.
- Although simulator currently works on Max OS X, majority of the non-GUI code should be portable on iOS platforms supporting OpenGL ES 2.0, including iPhone versions supporting OpenGL ES 2.0[35].

Note that although HoloSim simulates physical environment, the calculation engine included in HoloSim would not need to change for the code to control e.g. physical PRE. We would just need to replace output to the visualization portion of the

[34] Position of single visualized moxel is based on the averaging position of the grouped moxels.

[35] iPhone 3GS is the first version of the iPhone that supports OpenGL ES 2.0 [91], [92], [93].

HoloSim with output to the physical system. That means that HoloSim is, on the purely software side, more complex then it will be if it's only purpose is controlling the physical system, as we wouldn't need visualization, speed measurement and view control functionality in the real physical system.

Figure 19 shows the difference between functionality available in the current HoloSim and minimum functionality that will be needed if we are controlling real physical system:

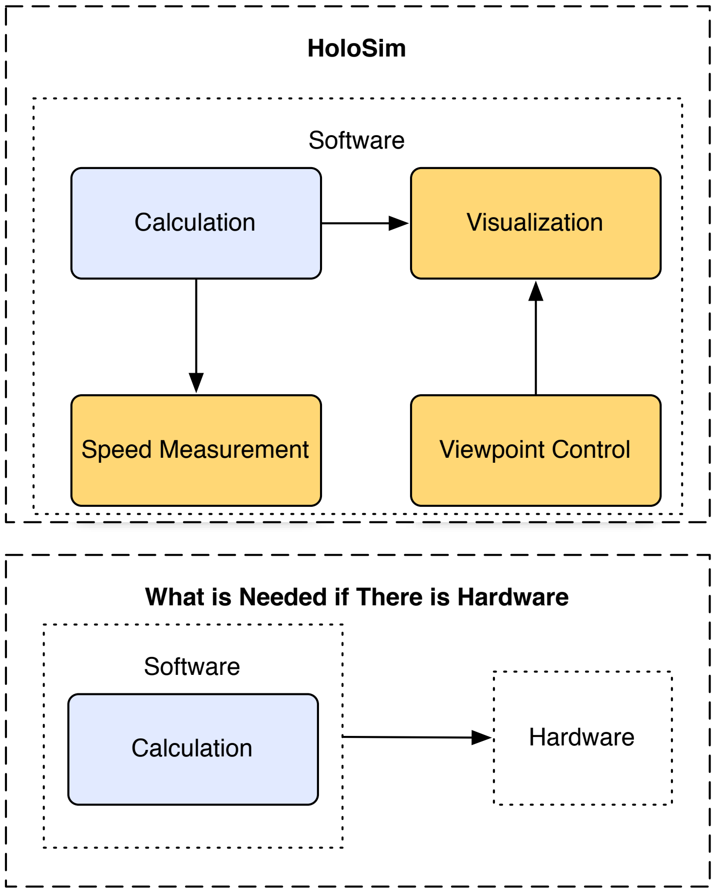

Figure 19. Current HoloSim required more software then will be needed if there was hardware.

Current implementation of HoloSim is:

• Showing feasibility of the use of algorithm ***Moxel Position Calculation***.

• Showing that it is feasible to take into account physical characteristics of the moxels based on piecewise linear (time vs position) physical response curve.

• Showing that proposed algorithm could scale to support million elements and larger Constrained Motion PREs.

• Providing infrastructure for the visualization, viewpoint control and moxel calculation statistics tracking and measurement.

Following sections will expand on the results that were found in the HoloSim.

6.1 Measured Performance in HoloSim

In respect to the performance of the moxels, two components impact rendering speed:

1. 3D scene complexity - number of polygons, as well as size and overdraw of those polygons. This area is affected in absolutely the same way as classical Z buffer algorithm, and as such the factors influencing it, as well as appropriate ways to optimize it, are well understood in the computer graphics community. In addition GPUs have already demonstrated in practice an ability to scale well in the respect to the geometric complexity of visual scenes.

2. Scalability to the various number of moxels, for the given scene complexity. This is the area that is directly related to control of the moxel based systems: while we can control in software the complexity of the input 3D scene, we can't really control number of moxels once the physical system has been produced. Consequently, we have paid a special attention to this area.

All measurements were performed on a 15” MacBook Pro 2.66 GHz from early 2009 (Apple model identifier MacBook Pro 5,3), using NVIDIA GeForce 9600M GT graphics card. At the time of purchase (March 2010), machine cost less then \$2,000 [36]. The goal of the test was the proof of the scalability of the proposed algorithms rather

[36] Note that measurement was based on what was perceived as a “worst case of moxel calculation” - e.g. we counted time needed to read back buffer as part of the moxel calculation, and system was using GPU for both moxel calculation and visual drawing, necessitating saving and restoring of the OpenGL state between moxel drawing. It is therefore likely that even higher performance then the one presented here could be achieved. However, scope of this PhD thesis is just showing scalability and feasibility of the approach, so we have setup conservative measurement characteristics based on the close to worst case scenario described above.

than determining the maximum number of moxels that given configuration could support. The test was set with a goal of ensuring that we are measuring the lower bound of what is practical on the given commodity hardware.

The test was performed on a 1,680x1,050 external monitor, with the size of the HoloSim viewing area of 733x555 pixels, and moxel's physical transfer function consisting of three piece-wise linear segments determining moxel position vs timestamp value used for the model deformation, with the model cyclically deformed. Screen saver/screen blanking/Time Machine were disabled for the duration of the test, and all front end applications except XCode, HoloSim and Terminal shut down. The test was performed by loading a model that is deformable based on the value of the timestamp. HoloSim was setup to automatically start rendering successive timestamps of the model. We took 1,000 frames from each model starting with the second frame[37]. Statistics and rate were calculated based on these frames, using moxels per second as a unit of data.

The same graphics card was used for the purposes of the moxel calculation as well as visualization in the HoloSim. Any time needed for the save/restore of the graphic card state and read back of the Z buffer was included as a time needed to calculate position of the moxels. No simplification of the moxels was done during these tests for the purpose of the visual simplifications.

[37] Initialization of the GPU data structures (e.g. render buffer) is performed in the first frame, so the first frame is not representative of the system in a stable state and was therefore not included when calculating statistics. The author would like to thank all of the PhD committee members, who pointed out during the defense of this thesis that the first frame shouldn't be included when calculating statistics.

All of the previous measures taken assure that any bias in our tests would lean toward moxel performance lower than what GPU is actually able to achieve for the given scenario. We believe that this was appropriate for the goal of understanding whether a proposed solution can scale and control the million element moxel based system. We have chosen a fairly conservative approach to reporting our results and in demonstrating scalability and viability of the approach[38] rather than reporting only the maximal performance that GPU based control will be capable of when controlling the physical system.

HoloSim performance measurements were performed on a simple model from the Google SketchUp. That model has 160 triangles in it.

[38] As well as the fact that minimum performance level outlined as a goal for the PhD thesis was achieved.

Table below shows measurements performed for the moxel calculation speeds based on the number of moxels:

Model Size (moxels)	Sample Mean (Kmoxel/s)	Sample Min (Kmoxel/s)	Sample Median (Kmoxel/s)	Sample Max (Kmoxel/s)	Sample Average Frame Latency (ms)	Median Frame Latency (ms)
100 (10x10)	155.37	4.22	162.07	202.84	0.960	0.617
2,500 (50x50)	3,943.63	90.74	4,051.86	4,940.71	0.906	0.617
10,000 (100x100)	14,993.54	719.01	15,408.32	18,552.88	0.715	0.649
40,000 (200x200)	9,465.64	2,998.05	9,013.07	62,992.13	4.377	4.438
90,000 (300x300)	20,146.25	6,818.70	20,179.37	96,670.25	4.678	4.460
160,000 (400x400)	41,252.59	4,885.50	27,605.30	141,718.33	6.840	5.796
250,000 (500x500)	48,072.03	10,324.18	43,576.78	161,290.32	5.872	5.737
562,500 (750x750)	79,826.95	16,234.23	83,475.55	168,766.88	7.484	6.739
1,000,000 (1,000x1,000)	93,884.89	32,000.00	100,811.69	180,505.42	11.536	9.920
2,250,000 (1,500x1500)	136,572.45	59,272.92	138,491.41	181,393.10	16.909	16.247
4,000,000 (2,000x2,000)	146,108.45	42,028.73	149,664.19	181,669.54	27.833	26.727
6,250,000 (2,500x2,500)	158,623.97	46,074.80	163,479.90	181,490.84	39.728	38.231
9,000,000 (3,000x3,000)	160,173.07	105,682.18	162,971.15	181,865.94	56.432	55.225

Table 1. Performance results of rendering model versus various number of moxels. All models had an equal number of moxels in the X and Y direction.

In the graphical form, results are shown on the graph below[39]:

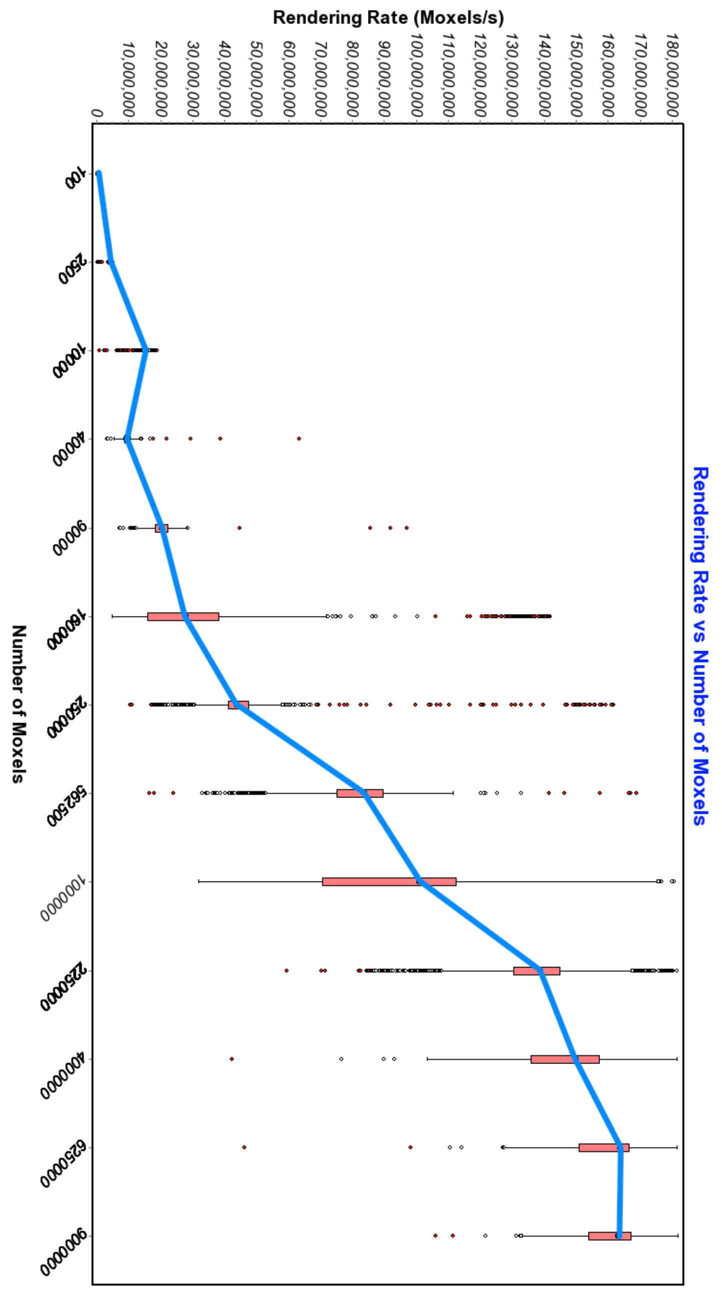

Figure 20. Moxel rendering rate as a function of the number of moxels in a model. Note that in order to fit all the measurements on it, graph is using ordinal scale on the X axis, as opposed to linear or logarithmic scale.

[39] Author would like to thank to Professor Steven Ouellette for help on how this data can be analyzed if looked at from the perspective of the statistical process control.

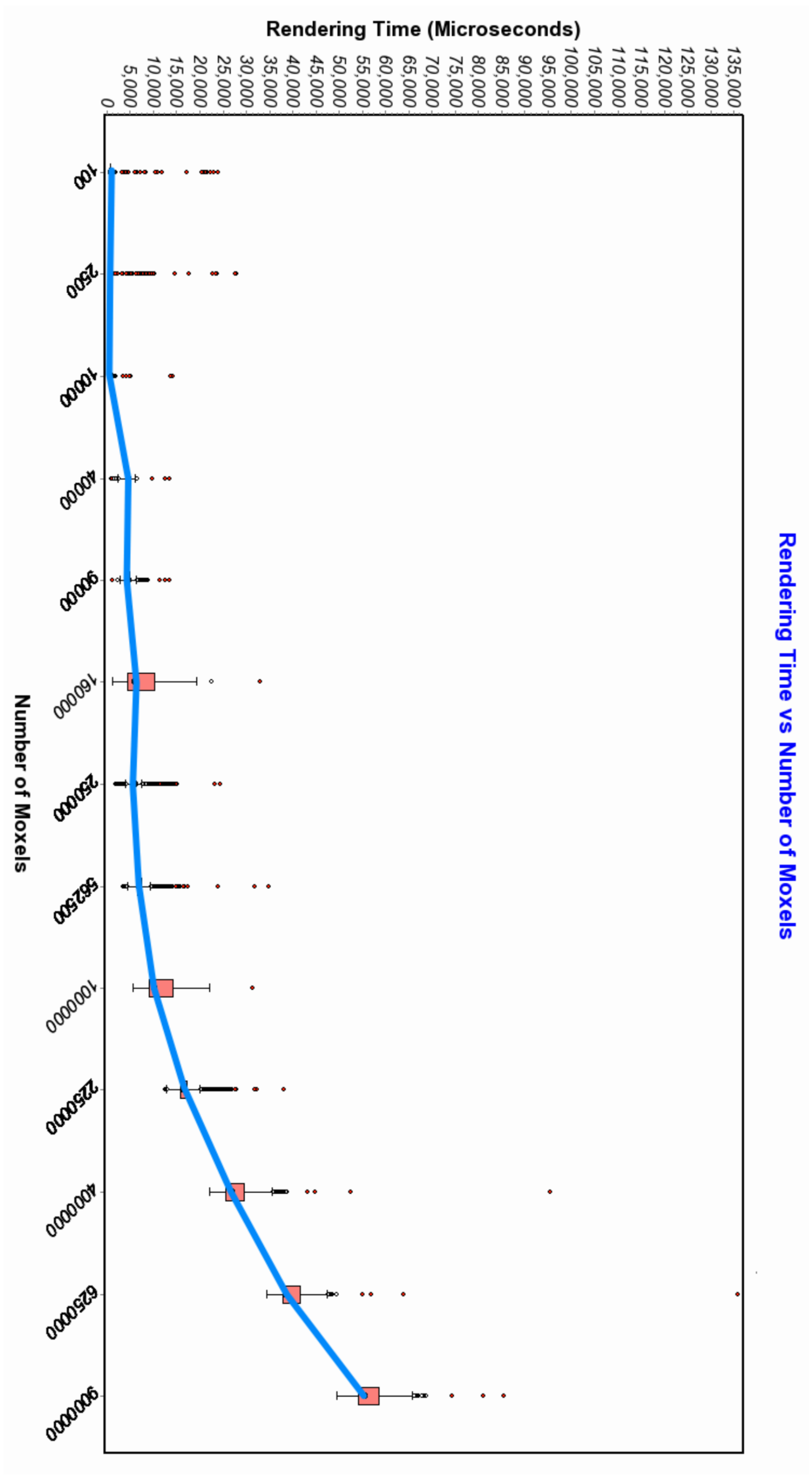

Figure 21. Moxel rendering latency as a function of the number of moxels in a model. Note that in order to fit all the measurements on it, graph is using ordinal scale on the X axis, as opposed to linear or logarithmic scale.

As the figures show, HoloSim clearly demonstrates an ability to control millions of physical elements per second, while taking physical characteristics of the moxels in account [40]. That being said, there are some additional things that could be noticed from the previous graphs:

1. We are clearly exceeding rate of 500,000 moxels/second on all but the smallest of models (10x10 moxel size) for which setup time heavily dominates calculation time (and that are small enough that they could be easily controlled on both GPU and CPU). From that perspective, it is clear that we are exceeding the goal that was set at the time of the thesis proposal. Note that this is achieved with the conservative approach to measurement (e.g. we are calculating OpenGL state change and initialization of the OpenGL renderbuffer as a part of the moxel calculation time).

2. Median of the latency of the frame rendering time is staying well within interactive range even for the models holding millions of the moxels, with the median of the latency of the 9M moxel model being 55.225 ms.

3. There is significant variation in the rate of moxel rendering between frames, as well as couple of outliers in the first 1,000 frames.

    a. Cause of those variations could be attributed to the impact of the state changes between frames, in relation to the use of the same GPU for both moxel calculation and visualization, as well as the associated cache

[40] Note that maximum model tested was 9M moxels with the number of moxels per second calculated based on the time it took to render 9M moxel model.

changes. Furthermore, it is possible that there are variations in the cache behavior between frames.

b. We are using the same GPU for visualization and calculation. That means that we are not only requiring frequent state changes, but that we are using GPU resources (e.g. caches) for the purposes different than rendering, and that furthermore we are alternating resource uses between two different goals with different optimal policies. This increases variability[41].

c. With that being said, definitive cause of the variation and factors impacting interframe variation are related to the question of what are maximum rates achievable on the particular GPU, and as such outside of the scope of this work that was concerned with showing feasibility of the GPU use for the purpose of the control of the large scale moxel systems.

Note that previous is GPU based performance on the given model for which tests were performed. Statistical tests performed on the distribution didn't show conformance to any well-known statistical distribution at the 95% confidence level. We are likely dealing with a data sample that is not coming from a population that should be described with the single distribution[42]. Consequently, statistical inference on what are parameter values in population this data shouldn’t be performed.

---

[41] Again, we are measuring in the conservative way, with the goal of being able to show feasibility of the control of at least 500,000 moxels/s. Consequently, our measurements are actually overestimating frame rendering time. As such, peak rendering rates and best optimizations to achieve those rates fall outside of the scope of this thesis.

[42] One implication of this is that all given statistics are values in sample. Another is that box and whisker is better representation of underlying data then e.g. average would be.

As we can see in the previous table, system performance depends on the resolution of the moxel model, and highest results were achieved on the larger models. Note that except for the smallest models that could be easily calculated on both CPU and GPU, we are clearly reaching our goal of controlling millions of moxels per second. Furthermore, frame rendering latency grows with the number of moxels in the model. Average and median latency is 56.432 ms and 55.225 ms respectively, for the moxel model including 9M moxels, which is certainly acceptable performance.

Chapter 7 - Beyond Pure Moxel Based System

It is possible to combine the previously described approaches with the larger class of the Cyber-Physical Systems. This chapter will discuss combination of the moxel based system with multiple Cyber-Physical Systems.

7.1 CirculaFloor and Moxels

CirculaFloor [10], [3] is a locomotion interface which uses a set of movable tiles to provide an infinite walking surface while staying in a limited confined space, as shown in Figure 6.

Infinite walking surface is provided by moving tiles in the opposite direction of the movement of the user as shown on Figure 6, so that if user is moving forward to Tile A, Tile B on which he is standing is moved backwards, keeping user in the same space. At the same time, Tile C behind the user will be moved in front of him, allowing him to step onto the next tile[43].

Combination of the CirculaFloor [3] with the Holodeck type of environment would allow simulation of an infinite surface not limited to flat surfaces [6], [7]. We can now present techniques by which proposed algorithms can be extended to support of CirculaFloor.

[43] There are actually multiple possible circulation modes in the CirculaFloor. For the complete description of circulation modes, see [10] and [3]. Figure 23 and description given here are showing simplified conceptual picture, using only three tiles.

First step in the extension of the ***Algorithm Moxel Position Calculation*** needs to map calculation of the moxel coordinates to the CirculaFloor tiles. In order to achieve this, we propose a following approach as outlined in the Figure 22:

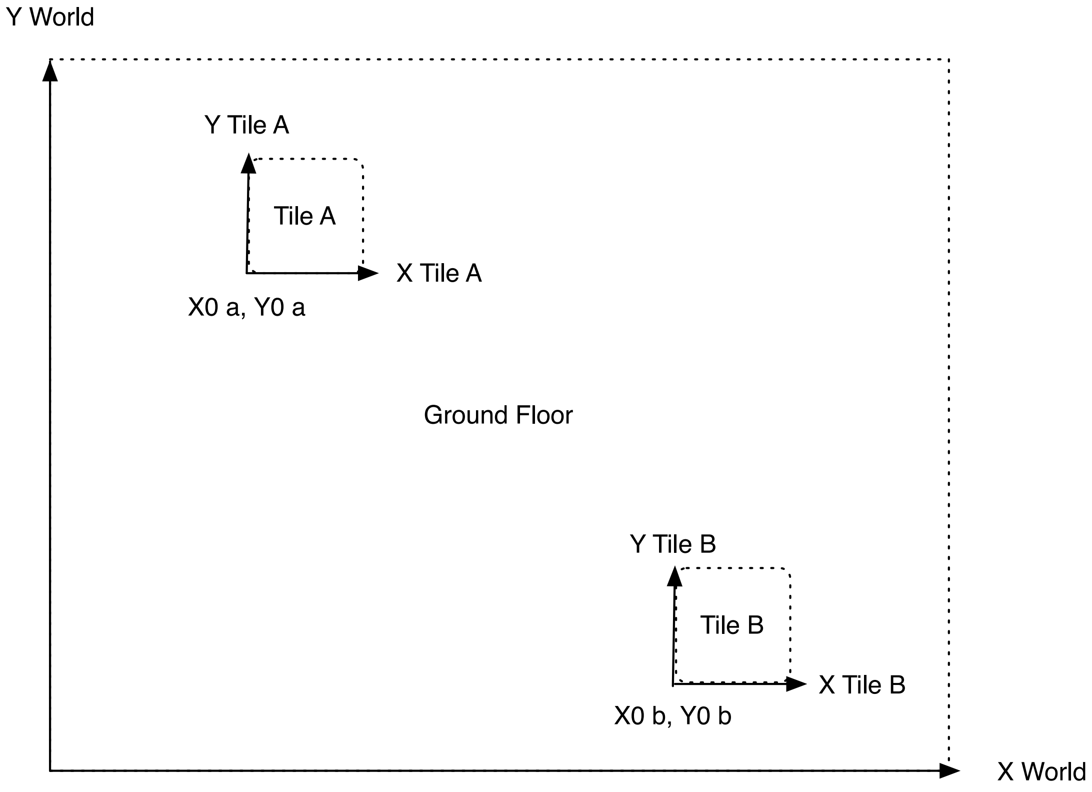

Figure 22. Mapping of the calculated moxel coordinates to the CirculaFloor.

Conceptually, for clarity of explanation, we will assume that we will perform a moxel calculation on the whole ground plane which includes all the space where CirculaFloor tiles are able to move. In effect, we are going to calculate position as if the whole floor is covered with moxels. Then we will apply the following Affine transformation between coordinates in the Tile A's coordinate system and world's coordinate system[44]:

[44] Although for the sake of clarity we decided to use non-matrix form, this translation could be shown in the matrix form using homogenous coordinates [34].

$X_{world} = X0_a + X_{TileA}$

$Y_{world} = Y0_a + Y_{TileA}$

Position of the moxel in the local coordinate system of CirculaFloor tile is:

$X_{TileA} = X_{world} - X0_a$

$Y_{TileA} = Y_{world} - Y0_a$

Now that we know how to map global coordinate system to any single tile using simple translation we can ensure that translation step will not have a high performance impact.

However, there is one more correction needed to account for the physical delay in raising the moxel on the tile. In the case where a tile is moving with some speed (Vx, Vy), what has to be accounted for is that coordinates corresponding to the moxel on the tile will be changing over the period of the time that moxel needs to move (as different moxels on the moving time would correspond to the different moxels on the floor). We will assume that the time interval is small enough that (Vx, Vy) could be considered constant. Where this is not the case a similar approach could be easily extended by taking any projected trajectory the user would take in that period of time.

For the sufficiently short moxel moving period Δt, we will assume that speed of the movement of the tile (V_x, V_y) will be constant. Furthermore, we will be interested in the final position of the tile at the end of movement. So transformation between position of the origin of the coordinate system of the tile $X0_a\ (t_0)$, $Y0_a\ (t_0)$ at the time t_0 when moxel starts moving and time t_1 when the moxel stops moving is:

$X0_a\ (t_1) = X0_a\ (t_0) + V_x\ \Delta t$

$Y0_a(t_1) = Y0_a(t_0) + V_y \Delta t$

Again, this step wouldn't have high performance impact as common term could be calculated on per-tile basis once; we would need one addition and one multiplication per moxel to find its final position.[45]

At this point, we have three possible approaches to addressing the calculation of the moxel position. Which one is the most appropriate depends on the moxel densities on the circulating floor tile:

1. We could have single GPU calculating position of the moxels across the whole ground floor and mapping it to the particular tile occupying the space of the pixel. This approach allows us to control system of the multiple tiles with the single GPU, and is appropriate in the situations when the moxel densities are such that whole room in which tiles are can be served using single GPU.

2. It is possible to dedicate GPU per tile, and to perform calculation only over the space of the single tile. This approach is appropriate in the situation when moxel densities are high (e.g. 1mm on 10m times 10m tile at 1 Hz requires 100M moxels/s).

3. Combination of the previous two approaches is possible[46], in which we cover area that is immediately around the CirculaFloor tile with calculation. This

[45] Common term in this case wouldn't include Δt because different moxels would start from the different positions, and consequently would need different times amounts to reach final position.

[46] The author would like to thank to Professor Willem Schreuder for suggestion of the hybrid method.

hybrid approach would work in situations when the room is much bigger than the area immediately around the tile, but moxel density is not big enough to warrant having single graphics card per CirculaFloor tile.

Proposed approach for calculation of moxel displacement in the XY plane takes its inspiration from texture mapping [34], in which parts of the texture maps are mapped to the flat polygon. In this case, we could consider that each tile in the CirculaFloor is analogous to the texture, and that ground plane is analogous to the polygon on which texture map is applied, with the further distinction that instead of calculating results (output color) to apply to the ground floor (analogous to polygon in texture mapping), we are calculating result (moxel displacement) to apply to tile in CirculaFloor (analogous to mapped texture) and that we take user's movement into the account. Note that although general approach is modeled based on the texture mapping, math used had to be modified to fit our case.

Similar approach to what CirculaFloor is doing in the XY space to provide impression of the infinite height achieved with the moxels that could be moved only limited travel space could be done by adopting CirculaFloor’s XY repositioning approach to the Z coordinate, as shown on Figure 23.

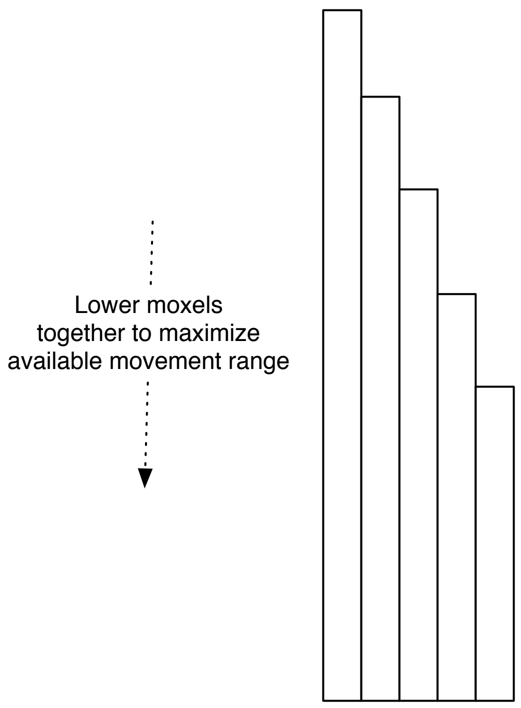

Figure 23. Z coordinate handling in moxelated CirculaFloor.

In this approach, lowest coordinate in Z direction would determine how far we need to raise a moxel, as the lowest moxel could stay at position 0 without being extended. This is conceptually similar to the direction that an extension of CirculaFloor [47]is using for moving a complete surface up and down in its “Stair stepper demo” as reported by [155] and currently implemented for [154], with addition of ability to provide finer local control of the surface because we can control individual moxel as opposed to controlling complete tile surface.

Note that it is possible to include Z coordinate range normalization both on the CirculaFloor tile itself, as well as on the GPU, by implementing additional reduce step,

[47] The are no publications currently available for this extension in vertical dimension [153]. There is museum exhibition of the enhanced CirculaFloor device [154], as well as various reports on the work [155].

as described in e.g. [97] and [156][48] with the reduction step on per-tile basis[49]. Furthermore, the Z reduction step doesn't need to be run all the time - it could be triggered on-demand, in the situation when we are running out of the moving range for the rods in Z direction.

Note that although the proposed framework on the CirculaFloor would use fewer moxels than if we were to take an approach in which a surface not modeled with CirculaFloor is covered with the moxels, it doesn't eliminate need for the control of the large number of moxels. Reason is that smaller tile allows us to reduce moxel size (e.g. while at 100m x 100m field even with diameter of 1cm we would need 100M moxels, on the 10m x 10m tile using 1mm diameter, we have the same number of moxels needed). This only gets more demanding in the approaches in which the complete floor is rendered, and tiles are then mapped based on result of the output from the floor.

Although combination of the CirculaFloor and moxels allows for the illusion of the infinite surface and as such makes CirculaFloor important technology, CirculaFloor can't avoid the same problems that any repositioning mechanism does, namely that it is not possible to move user smoothly without user noticing that he has been repositioned[50]. CirculaFloor can't provide stable walking surface to 22% of users [3].

[48] Also, similar technique as to what we are describing could be implemented in OpenCL [140] or CUDA [157], but as we are already need Z buffer related calculation, we decided to formulate this problem in the form of the classical graphics calculation.

[49] Interesting topic of the future research is could we use geometry shaders to help to this process, as described in the [158]. However, at the current time Geometry Shaders are not likely to be supported on all architectures that this work is potentially interested in (e.g. cell phones with low end OpenGL support) so this work is not using them.

[50] Because some acceleration is required in order to reposition a user - system is reacting on the user movement with some finite delay.

Based on these findings, there is no reason to believe that adding moxels to CirculaFloor would in any way improve user's perception of stability of the surface. This is a part of the larger trend - similar systems reported a need for user training so that the system could be used at all [12].

Consequently, though the proposed approach allows for the simulation of infinite space within a confined space, moxel covered static surfaces would likely provide better experience. However, this is a limitation of the CirculaFloor as a physical surface and not in any way aggravated by the approach to the software control that we have chosen to use here.

7.2 Extension of the Proposed Approach to Systems Combining MEMS and Moxels

The proposed approach could be extended to small scale systems, e.g. tactile displays [18]. These type of systems have significant usage in assisting visually disabled people [24]. The also offer an interesting research direction as Dynamic Displays [89] and TeslaTouch [5].

In each case, portion of tactile display could be considered to be 1D Constrained Motion system (moxel). With regard to the mechanisms for the software control, tactile displays can be divided, in relation to their hardware capabilities, into displays capable of raising elements to multiple positions (e.g. fully down, quarter of the way up, halfway up etc) and bimodal (on or off) displays. GPU based algorithms are useful for both classes of dynamic displays.

For the tactile displays capable of multiple positions of the moxels, the previously proposed algorithm for control of the Holodeck environment applies naturally to the visualization of the 3D models, possibly with the additional transfer function (e.g. function that transfers Z range in the physical model to the Z range available in the MEMS based device). As previously demonstrated, this is a special case accounting for the physical limitations of the moxels and could be addressed with the extension of the proposed fragment shader/texture input algorithm we are using.

Meanwhile, for the tactile displays capable of only bimodal (on/off) state, GPU supplies following capabilities:

1. Mapping of colors to the tactile texture would allow for representation of the tactile feel of the color. Although best mapping is an area of further research in respect to this thesis, some work related to this area is [67], [67], [127], [27], [150] and [24].

2. For the visually impaired people GPU provides the ability to accelerate image enhancement operations needed for the better tactile perception of the images [75], [76]. There is significant previous work in the area of GPU acceleration of image enhancement algorithms - one example [159] discusses use of the GPU for image space transformation including edge and corner detection.

3. Previously proposed algorithms allow us to calculate the ideal height to which a moxel should be raised; having this calculated we could use a combination of tactile dithering and rounding of the position to represent height of the surface.

7.2.1 Tactile Dithering Acceleration Using GPU

In the case of tactile dithering acceleration, we use the GPU fragment shader to quickly calculate position of the moxel in order to form dither pattern, based on the input pattern.

For example, we can say that following dither pattern corresponds to the particular color, as shown on the Figure 24:

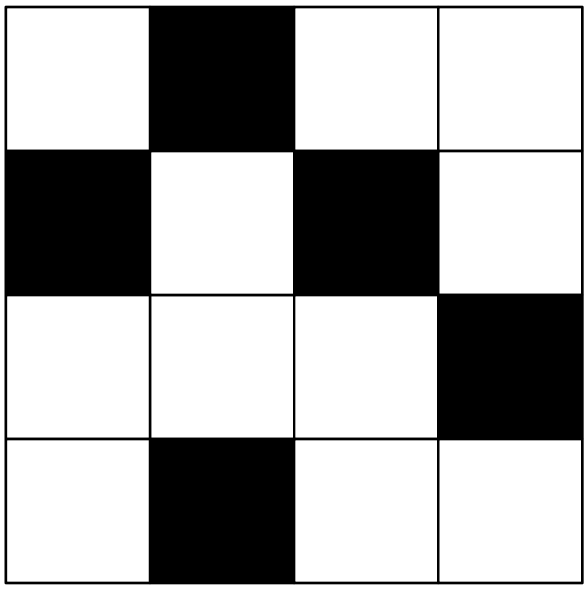

Figure 24. One possible tactile dithering pattern using 4x4 moxel grid to represent particular color.

If we have an input picture for which we want to represent tactile dithering equivalent, we would need to map every pixel to the set of pixels (e.g. we map blue to the 4x4 pixel pattern shown on the Figure 24). We would achieve this by using the following multi-pass approach:

<u>Algorithm Tactile Dithering with GPU:</u>

1. In the first pass, we would generate the normal picture, but instead of rendering it to the screen, we would render it to the 2D texture. Output of this pass is texture that has colors of the particular pixels in the original color, without tactile dithering.

2. Then we would use that 2D texture as an input to the second pass of he shader in which each pixel in the texture will be mapped to the group of output moxels, with the color being used as index in the second texture that is holding dithering patterns for the color.

3. Using multitexturing capabilities of the GPU, we will be able to provide tactile dithering in two passes.

Previous approach is inspired by the scatter/gather mapping step (as described in [38] and Percentage-Closer Filtering [160], in which multipass render to textures is used in order to emulate scatter step on the GPU, but is modified to be applicable to the moxel based rendering for bistate capable moxels).

Although the previous approach is relatively straightforward to implement, there is a limitation in relation to the maximum size of the textures, with many GPUs being limited in the maximum texture size that they can support [37]. That in effect means that the picture shown as a part of step 1 has to be of limited resolution (limited to the max texture resolution). If we need to have a pre-dithered picture of higher resolution then what fits in the maximum texture, we have couple of options to address it:

1. Use “reduce step” as described in [38] to scale resolution of the input picture (if we don’t need full resolution of the input picture).

2. Use multiple rendering targets in order to split the picture into a set of smaller output images (one way to render to the multiple targets is using EXT_framebuffer_object [161] or equivalent functionality of the later OpenGL versions). Note that, due to the limit of the number of multiple rendering targets supported, multiple passes might be required for the different parts of the picture[51].

[51] Clearly, multiple rendering passes at the full resolution would slow down calculation. Number of passes required would depend on the proportion of the max texture resolution and resolution of the image that we want to render.

7.2.2 MEMS Tactile Systems for General Population

In addition to the clear benefit of tactile displays to visually impaired people, tactile feedback has potential to be useful for the general user as well:

1. Dynamic displays [89] and TeslaTouch [5] are clearly useful to the general population as they allow combination of the touch screen with the tactile feedback.

2. Work on touching reverse surface of the phone to avoid situation in which user fingers are obscuring the display and preventing precise selection[52] of objects on the screen [162]. Such a situation necessitates a tactile screen that is refreshed with a similar rate to that with which we refresh a display.

Previous two use cases are interesting for the following reasons:

1. Users that are visually disabled do not require high refresh rates, due to the limited speed with which this group of users can scan tactile display [75]. However, these users might need further image processing in order to better comprehend picture (e.g. edge detection), which, as we pointed out previously, could be implemented via GPU.

2. Furthermore, users that are sighted would require similar refresh rates as current computer displays are providing as they would want to touch what they see.

[52] So called “fat finger” problem [162].

Combined with high pixel densities required if user is to “scan” with his finger over surface (based on 20-40 microns period change being perceivable [115]), we clearly need to be able to control number of moxels comparable with number of moxels needed to control Holodeck based systems of large scale.

As an example, lets repeat our conservative estimate of the 200,000 moxels (based on the moxel diameter of 0.5mm and Kindle DX size of surface) and lets assume that they are controlled at 20Hz - this would require software control of 4M moxel/s. If we are looking at control of the moxel elements 20 microns in size, we would need to control billions of moxels/s (at 20 Hz and Kindle DX sized touch surface[53]). Clearly, control of such high numbers of elements would tremendously benefit from the GPU.

One also must take in the account that the carbon nanotubes could be considered essentially Constrained Motion, 1D system [22], [90]. At the nanoscale, we might encounter the need to control even higher number of elements.

To conclude - there is clearly a need for control of a massive number of mechanical elements when talking about MEMS devices. Here we could see how GPU based approaches could be used for the calculation of the desired position.

[53] Based on the external dimensions of the complete KindleDX device.

7.3 Catoms Integration

As shown in this thesis the advantage of moxel based systems is in their ability to quickly control millions of elements, and in doing so approximate large areas. The disadvantage of moxel based systems is that they are limited to showing only 2D surfaces.

Catoms, [49] on the other hand, are able to represent arbitrary surfaces. However, the work on software control of a large number of catom-like elements is still in the early stage [55], [55], [56], [57], [58] and due to the challenges with the software control, catoms are not well suited for the situations where we are in a need of simulating large ground planes.

A combination of the Holodeck and catoms has the potential to combine the best characteristics of both, with the catoms used for the simulation of the objects that are not well approximated with the 2D functions (e.g. ball on the Figure 15) and the HoloSim being used for the description of objects that are well approximated with the 2D functions.

Furthermore, the nature of previously proposed moxel algorithms could provide information to the Catom control algorithms about the position of every moxel, which will be needed in order to combine systems. In effect, we would calculate position of every moxel and declare any area that is between fully retracted and current moxel position as occupied.

This raises a question: how can we quickly determine which objects are suitable for rendering in the HoloSim using moxels, and which objects would require combination of the moxels and catoms for rendering?

In order to answer this question, lets suppose that we have set X = [X0, X1, X2, ... XN] of objects with 3D geometry. What is needed is a GPU based algorithm that allows us to quickly determine in which areas we have objects that are requiring simulation using Catom based rendering.

In this example, as we look at the set of moxels, we first notice that we could divide moxels based on what part of which distinct region of XY plane they occupy, as in Figure 25. We would designate moxels that are in the shaded area as set C, and the moxels that are in the white areas as the moxels in set H.

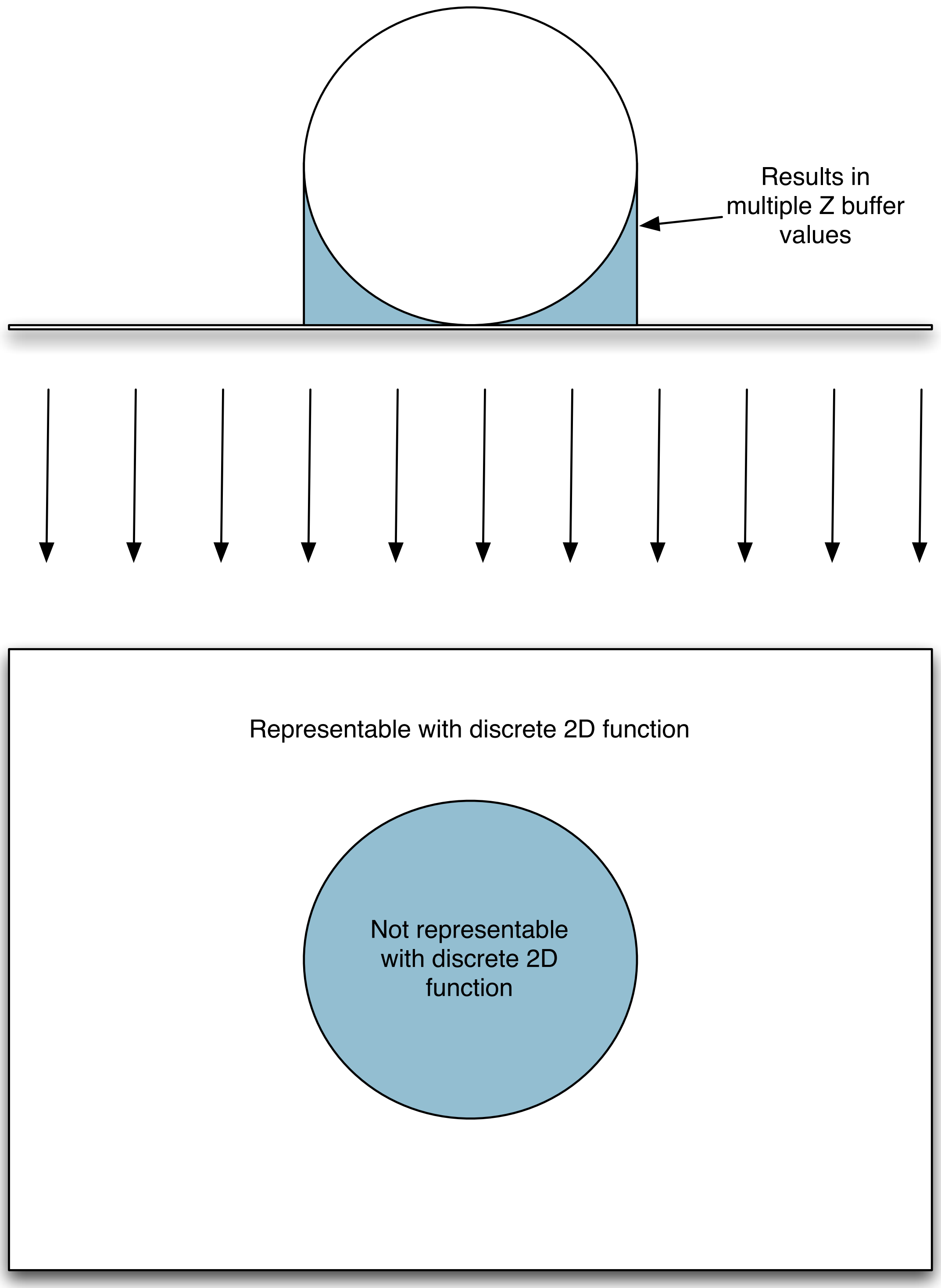
Results in multiple Z buffer values
Representable with discrete 2D function
Not representable with discrete 2D function

Figure 25. XY plane (lower part of the picture) could be divided in two distinct areas,: one that cannot be represented by 2D discrete function (shaded area) and the other that can be represented by discrete 2D function (white area). This figure is based on the figure found in the author's earlier work [6].

Now, we will setup a multipass rendering algorithm which will allow us to use GPU to determine whether a moxel belongs to the set C or set H.

<u>Algorithm Calculating Catom and Moxels Regions:</u>

1. Initially, set all Z buffer values to 0.

2. Render two output targets - Z buffer and output texture which will indicate whether a moxel belongs to set C or set H.

3. Render all objects using Holodeck rendering algorithm.

    a. A modification of the fragment shader checks the existing Z buffer value, and if the value of Z buffer is not 0, we know that an overwrite occurred.

    b. The fragment shader indicates in the output texture whether a moxel belongs to set C or set H[54].

4. Now, we have a texture showing XY plane area moxels that can't be represented with the moxels. Set new rendering pass with following inputs:

    a. Texture showing current value of the Z buffer

[54] This approach assumes that if there is only single polygon above moxel, that polygon represents ground plane. Variations on the approach are possible in which we decide minimum difference in Z positions before we declare that something has to be rendered in catoms (so that for example very thin "roofs" extending over building and consisting of only single polygon are not rendered in HoloSim nor catoms).

b. Texture showing C or H belonging.

5. Have the fragment shader in that pass output Z buffer value for the moxels that belong in set H and Z_{min} for the moxels that belong to set C.

Now, we have successfully segmented the system into the areas where catoms are needed and where only moxels are needed. We can still use moxels to represent the height up to the ground plane, to further reduce number of catoms needed. In order to do that, we would use following approach:

Algorithm for Catom and Moxel Rendering

1. Render ground plane, with CH texture and Z buffer values from the previous pass as an input.

2. At moxels that are in C area, use the ground plane value for the Z coordinate. In the other areas, use values of the Z buffer. If there is no groundplane at the area, leave moxel down[55].

The result of this algorithm is that moxels in the set C (area that can't be represented with the Holodeck correctly) are still in their lowest position. We can use catoms to fill up remaining holes occupied by moxels that belong to set C.

[55] Net effect is that we would rise moxel up to the level of the groundplane but not higher in the C area.

This algorithm is based on the implementation of the Z buffer in a programmable pipeline as described in e.g. [37], but modified for the detection of the overwrite and classification of the primitives based on the overwrite.

Note that backface culling [34] has to be disabled for this algorithm to work correctly, as well as that techniques for rendering to more than one output textures and “tiling” that textures as described in tactile dithering section of this work may need to be done in order to render moxel spaces exceeding max texture size for the single texture.

Furthermore, note that in some cases, objects in computer graphics could be overspecified (e.g. have some polygons inside of the object, as a results of the objects consisting of intersecting parts that assemble in the whole object and count on the Z-buffer to correctly render the object with intersection)[56]. In those situations, the previous algorithm would have a tendency to expand C region to include those over-specified objects, even if that region could be rendered only using moxels! Although this doesn’t impact correctness of the previous algorithm (in a sense that surfaces would be correctly rendered), it does impact optimality (as catoms are used more often then needed). In order to address this issue, one possibility is to limit or eliminate “inside” polygons in the object.

Please note that upcoming Catom pass would significantly benefit from the detection pass in which we detect which objects need to be rendered with it. We can

[56] Author would like to thank to Professor Willem Schreuder for this insight and pointing out that one solution is to make sure that all polygons describe only outside surfaces.

again perform this calculation on the GPU, using following approach on the fragment shader[57]:

<u>Algorithm Calculating Objects Using Catoms:</u>

1. We will start by assuming that each object has Axis Aligned Bounding Box (AABB) around it (AABB generation of them is described in [163]).

2. Pass AABB as primitives, as well as H-C set membership texture. Have color of the element represent object that we want to identify.

3. Set output textures indexed by the index of the 3D objects input, one representing objects that have moxels in C, and another with the moxels in H.

4. For each AABB, pass H-C texture for it, mapped in a way that would map AABB to the part of the H-C texture it is covering.

5. Have the fragment shader check H-C texture for the presence of moxels in H or C. Mark output texture appropriately[58].

[57] Please note that in this particular situation, we can benefit from the stream model presented in the OpenCL [140], [140] or CUDA [141], [157] too and that algorithm proposed here is general algorithm appropriate even for the devices that are not having OpenCL/CUDA interfaces exposed (e.g. some mobile platforms).

In conclusion, we show how catoms and HoloSim based environments could be combined, how HoloSim could be modified to account for catoms, and how GPU based calculation could achieve the aforementioned goals. The proposed approach could be used for the combination with other non-catoms based systems that could show shapes that can’t be achieved with a moxel only system (e.g. joint arrays described in [164] could potentially present those functions if arbitrary angle between joints is permitted).

7.4 Aliasing in Moxel Based Systems

As expected based on the fact that we are using a discrete system to represent a broad range of shapes (and some of those shapes having components which frequency domain representation would go above Nyquist frequency of the HoloSim [130]), HoloSim exhibits aliasing behavior while representing series of shapes.

Figure 26 shows the same object (chair) reassembled at two different moxel densities - picture on the right was obtained by downsampling picture on the left. We used averaging (box filter) in order to determine height of the aggregated moxels on the right side of the picture.

Please note that the Moiré pattern visible on the upper chair is result of the aliasing artifacts in the visualization part of the HoloSim simulator, and not the fully realistic representation of what user would see in the HoloSim[59].

[59] Although Moiré patterns could be seen even when observing with the naked eye, in the case of the Holodeck environment, the user is expected to wear a VR helmet and watch a computer generated image of the Holodeck environment, with computer visualization being responsible for the visual and Holodeck being responsible for the tactile feedback. That is why it is not expected that visual Moiré will be a problem in the real Holodeck, although it is quite possible that there will be some tactile equivalent of the Moiré that we will need to take into account. That would require testing in the physical model of the Holodeck, though.

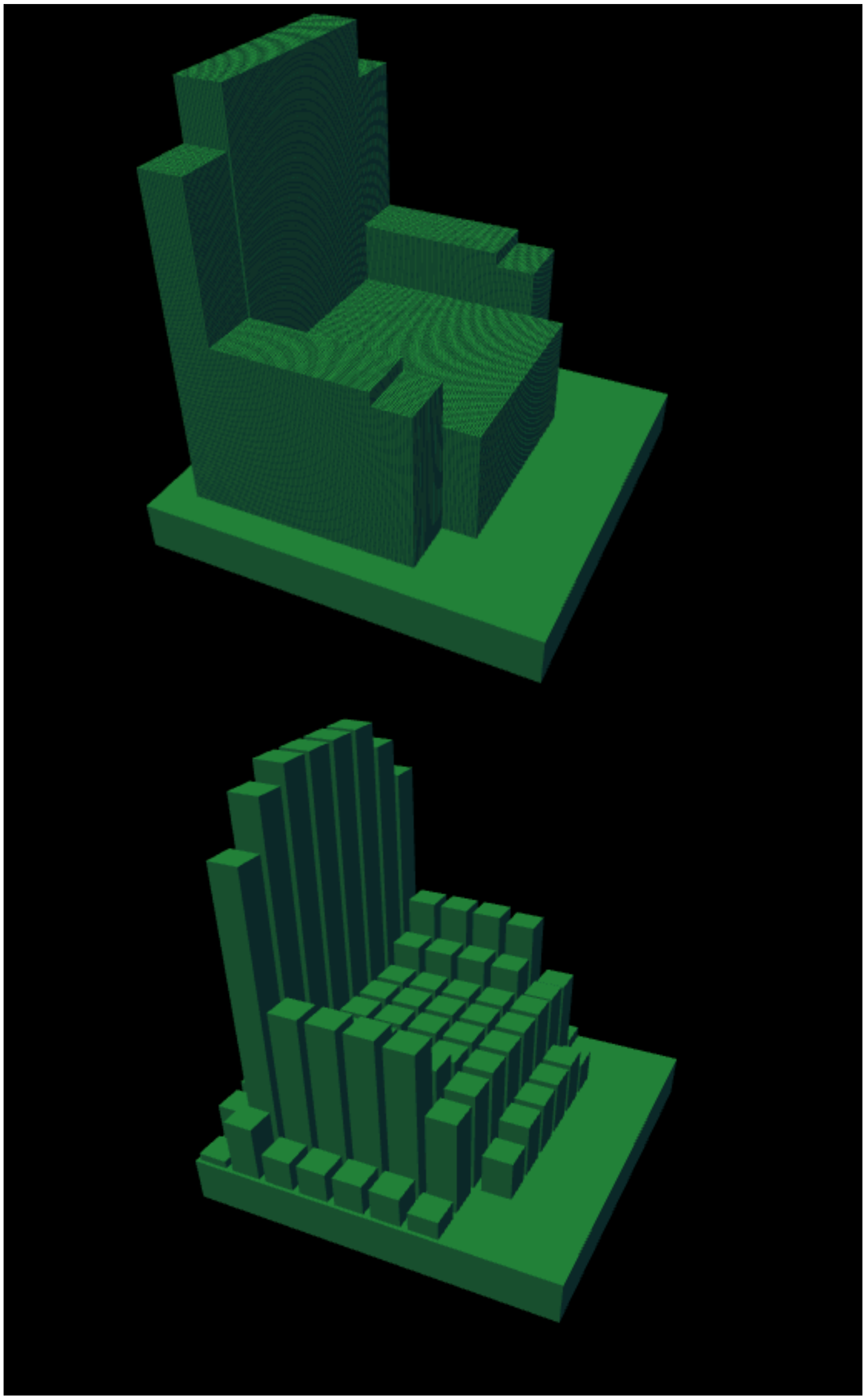

Figure 26. Simplification of the moxel based geometry, done by averaging height of the moxels in the area. Note how that results in “steps” on the chair in the right side of the picture.

Before we continue with the details of this section, please note that the details on aliasing phenomena about to be described here, lay outside of the scope of this thesis work, and are described here in order to point to the potential for the future research that exists in this area.

As it can be seen from that picture, “gradual moxel transition” between fully erected moxels and the moxels on the floor provides a very unnaturally looking representation of the chair[60].

Figure 27 clearly looks better. The surprising thing is that that picture was obtained by rendering the chair in the lower resolution, and raising every moxel to the position that the center of the moxel occupies. At least in this particular case, we might be better off using an approach that is different then the approach typically used to address aliasing in the computer graphics! In computer graphics antialiasing in effect uses variation of the color intensity on the edges of the structure to soften visual perception of the aliasing (the author’s description of the net effect of various anti-aliasing methods that were described in [130]).

Previous example is not conclusive evidence that traditional anti-aliasing methods wouldn’t work in the Holodeck type of the environment - effectiveness of the anti-aliasing depends of the resolution of the display and rendered geometry, and it is possible that what is seen in this particular case can’t be generalized[61].

[60] Maybe even potentially dangerous as there are more “bumps” to catch and trip the user. However, final judgement on this topic would probably require testing of the physical prototype, so it is still too early to make claim in this respect.

[61] Author would like to thank to Professor Clayton Lewis for providing this insight.

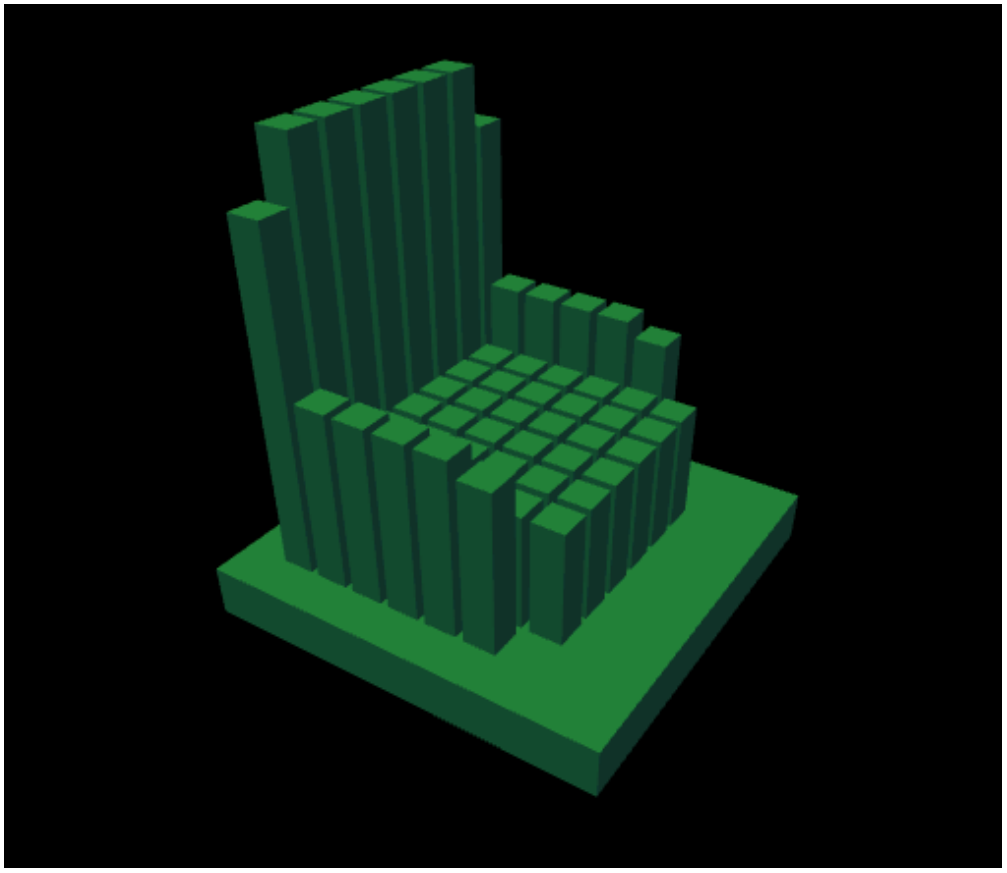

Figure 27. Rendering of the chair on the left side of the Figure 26 in lower resolution. GPU would choose height of the moxels based on the Z buffer value in the center of the moxel.

The conclusion here is that further research might be needed to address the issue of spatial aliasing in the physical Holodeck and that assumption shouldn't be made that anti-aliasing techniques used in computer graphics are applicable to the spatial Holodeck systems[62]. This, of course, is likely to apply to the hardware based anti-aliasing mechanisms present on modern GPUs, too.

[62] Similar research opportunity exists in the area of the tactile displays, and for some examples of related work, see [165].

In addition to the aliasing in the spatial domain, there is a question of the aliasing in the time domain in the Holodeck type of the systems. What if the required frequency of the moxel movement exceeds possible physical frequency of the moxel movement? Do time based anti-aliasing approaches from the other fields (e.g. computer graphics) apply to HoloSim based environment?

The answer to previous questions is almost certainly “no”, at least not without significant changes. The reason for that is that, in addition to user perception, we have to take into account user safety. This is very different problem than anti-aliasing in the field of computer graphics in which single pixel can’t injure the user. The next section will discuss issues of the user’s physical safety in more details.

7.5 Physical Safety of User in Holodeck System

Holodeck is a large scale system in which the user can move. As we previously demonstrated, it is expected that Holodeck could have millions of moxels in it. As moxels are physical entities, when manufacturing millions of them, some of them will be defective. Also, during use, some of the moxels may become defective.

Consequences of defective moxels vary, depending on the failure mode:

1. Moxels that are still movable, and whose physical characteristics for some reason change (e.g. upper part of the moxel lost) could be addressed using previously described the fragment shader algorithm for the moxel imperfection handling as they could be considered just a special case of the moxel imperfection with the unusually large variation compared to other moxels. This type of moxels is not likely to represent safety hazard.

2. Individual moxels that are stuck in the lower position are not a serious safety hazard. Their implications are likely to be limited to imperfect scene representation.

3. Closely grouped moxels that are stuck in the lower position could form “holes” in which users’ feet could fall. We would need to limit users access to those area.

4. Moxels that are stuck in the “up” position represent obstacles, but more worrisome individual moxels that are stuck in the upright position could injure a

user. We have to address presence of the individual “spikes” to prevent impalement of the user.

5. Moxels that are immediately around user foot or in his path must not be moved rapidly, as that could cause user to trip and fall as a result from moxels moving[63].

6. Moxels stuck in “halfway up” position are for the purposes of the physical safety similar to the moxels that are stuck in the fully extended position[64]. If there is only a few moxels that are stuck in the “halfway up” position and reduction in the range is not too severe, it is possible to address this situation by collectively rising the rest of the moxels to that level at which single moxel is stuck[65].

We devised an approach that would allow us to address each one of the previous situations that could affect physical safety of the user[66]. The advantage of running safety algorithms on the level of moxel is that we could account for the moxel failure on much more granular keel without introducing too many objects in simulator, and that we could combine it with adaptive resolution/movement optimization.

---

[63] Please note that there could be use cases in which we would purposely cause the user to fall as a part of training (e.g. training soldiers to react on losing balance during firefight), but those use cases are rare and would likely require some protective equipment for the user. We are limited here to describing the more common case in which we want to prevent user from tripping.

[64] Because a moxel stuck in the “halfway up” position is equivalent to a moxel that is stuck fully up, but for which the range of travel is just half of the range of the travel of the original moxel.

[65] The author would like to thank to Professor Willem Schreuder for this insight.

[66] Please note that for the final evaluation of the effectiveness of the proposed approach, experiments with real users in physically sized system are needed. This thesis work is limited t the software work possible without having a physical system built (physical system is unlikely to be built if there is no way to control it so control algorithms have to come first).

7.5.1 Addressing Moxels Stuck in the Up Position

Moxels that are stuck in the fully extended position would represent “spikes” on which a user might injure himself. In order to address those spikes, we have to “smother” the spike, as in Figure 28.

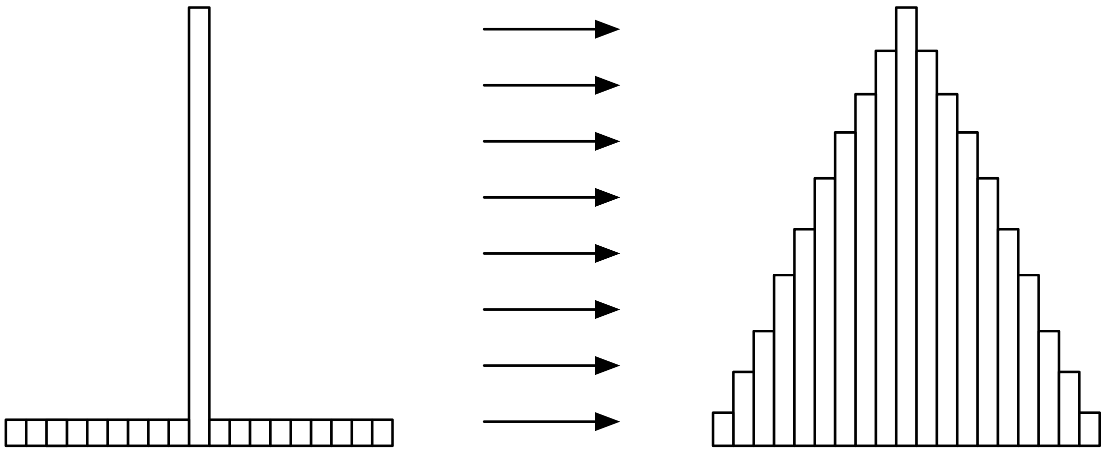

Figure 28. Addressing physical safety of the moxels that are stuck in the “up” position. Side view is shown in the picture, but in 3D space moxels will be risen in a radial pattern around the moxel that is stuck.

As we can see in the previous figure, the approach taken in addressing the pixels stuck in extended position is to smooth the transition. One way to smooth the transition is by using Gaussian blur (as described in e.g. [130]) around the stuck pixels.

We would use following algorithm to address pixels that are stuck in the upward extended position:

Algorithm for Stuck Moxel Isolation:

1. Have 2D texture (safety texture) representing moxels stuck. The value in the texture would represent the height on which a moxel was stuck[67].

2. Perform smoothing of the moxels around failed moxels. This is done by applying Gaussian blur on the 2D texture, so that moxels near the moxel that failed would get their corresponding texels partially risen, as in Figure 28. The blur operation could be achieved using CPU or GPU based algorithms (e.g. [143]), and has to happen only once when the moxel failure is detected.

3. In the fragment shader, have "safety texture" as an input to the fragment shader during moxel position calculation. If the safety texture indicates that the pixel should be elevated, elevate the pixel to the maximum of Z-buffer input and safety texture input for that pixel.

Note that in this approach, it is not a problem if the maximum texture size is smaller than the number of moxels in the Holodeck, because for the blur type of operations could have multiple moxels corresponding to the single texel, and raise all those moxels together.

[67] For the purpose of this approach, there will be no difference between addressing moxels that failed half-way up and addressing moxels that failed in the fully extended position.

As proposed algorithms modify physical landscape, communication with the visual engine will be beneficial so that user can be shown visual representation of the obstacle.

Finally, note that the calculation and recalculation of the safety texture could be performed while Holodeck is running. That means that it is possible to react on the indication that new moxel failed in Holodeck without the need to take current Holodeck session offline.

7.5.2 Addressing Groups of Moxels Stuck in Down Position

In the situation where we have a group of moxels stuck in the down position, we have a danger of user's feet being trapped in the hole created. Similarly, if we have a large number of failed moxels in a confined space, it is possible that they would impact physical characteristics of the surface they represent by reducing strength of the surface or by making the actually represented surface much different than what we are expecting it to be, as in Figure 29. Because of this, we would treat any situation in which more than some predefined percentage of the moxels fails in down position as a failed region[68].

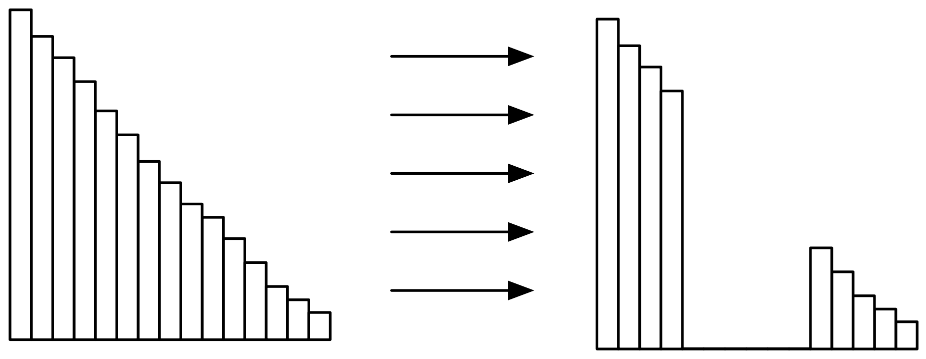

Figure 29. Failure of the set of closely grouped moxels causes gradually downward slopped surface to finish with the hole in it.

In order to address group of moxels that are stuck, we first need to identify them. In order to do that, we would use following algorithm that could be implemented on GPU:

[68] Please note that probability of closely grouped moxels stuck in the down position depends heavily of the hardware implementation used for the control of the moxels. If chance of having each moxel fail is uniform across all the moxels, chance of large group of adjacent moxels failing is likely to be small. However, if there is hardware implementation using common hardware elements in order to control group of closely grouped moxels, then it is possible that whole group of moxels would fail.

<u>Algorithm to Protect Stuck Moxels:</u>

1. Define texture that maps to the whole Holodeck moxel grid, with texel being marked with value 1 if there is a failed moxel in the down position that corresponds to the moxel position in the document.

2. Use reduce step [38] on the texture to count total number of moxels that failed in the region[69]. Output of this reduce step is new texture, where each texel corresponds to a group of moxels, and value of the texel corresponds to the number of moxels that failed in a given area.

3. If the number of failed moxels exceeds a threshold, declare the whole region to be a "hole".

4. Repeat previous steps until the area of the hole is big enough not to represent tripping hazard (e.g. much larger then user ‘s feet). We would call this texture SL.

Previous algorithm needs to be done only once a new moxel gets stuck in the down position. Output from it is a texture representing regions in which moxels form holes (step A on Figure 30). In that texture, texel will be active if they are representing

[69] Author would like to thank to Professor Richard Han for pointing out that this is a similar approach to median filter as described in e.g. [129].

region in which moxels failed as a hole. We now will do edge detection step on those texture (on GPU, as described in e.g. [159]) and get boundary region in which we have series of correctly working moxels.

Now, we would use that region of correctly functioning moxels to raise a "fence" around the "holes" as shown in Figure 30, step C. However, we can’t just raise fence as square pattern around stuck moxels because corners of the square would represent sharp points (only one moxel in diameter) that could injure the user. Instead, we need to raise a circular fence around the moxels that failed. To achieve that, we would use the following fragment shader algorithm:

Algorithm for Moxel Fence Builder:

1. Start in Step A from Figure 30, where SL texture is calculated.

2. For every moxel, sample circular area to determine if the moxel is near enough to the failed moxels in SL texture that it is part of the fence. It is considered near enough (red moxel in Step B on the Figure 30) if there is any texel in SL texture within radius R from the moxel that is marked as “failed region” (grey moxels in SL texture in Figure 30)[70].

[70] Radius R is determined by the physical characteristics of the moxels so that it is large enough that user could put a whole feet in or that it is large enough to impact user's stability (so that he doesn’t injure the foot if he steps on the fence) and that thickness of the fence is enough to prevent user from accidentally stepping over it and entering problematic region in SL texture. Furthermore, parameters would have to be chosen to ensure that fence is of some minimum radius, so that it doesn't represent spike.

3. If moxel is in the “failed” region in SL texture (grey moxels in Step B in Figure 30), leave it down. If it is in the boundary region (red moxels in Step B on Figure 30), then raise it up.

4. Output of Step B in Figure 30 is which moxels should be down if it is in failed region (grey pixels in Step B on Figure 30), and up if it is part of the “fence” (red pixels in Step B on Figure 30).

5. However, as we have some moxels that really physically failed and can’t be controlled, we would really be able to control only red and white pixels in Step C in Figure 30. Still, the result is identical to the result calculated at the end of Step B of that figure.

Previous algorithm is adaptation of the scatter/gather GPU algorithm described in [38] and sampling of the texture map (fairly common operation in GPU programming, see e.g. [37] for example)[71], with the elements of the pyramid methods [143] (due to the multilevel view of the areas of stuck moxels).

[71] As sampling could be done on the texture of size smaller then the moxel field (Step A in the Figure 30) demand on the texture memory bandwidth will be helped, and the chance of the SL texture fitting in the texture cache is increased.

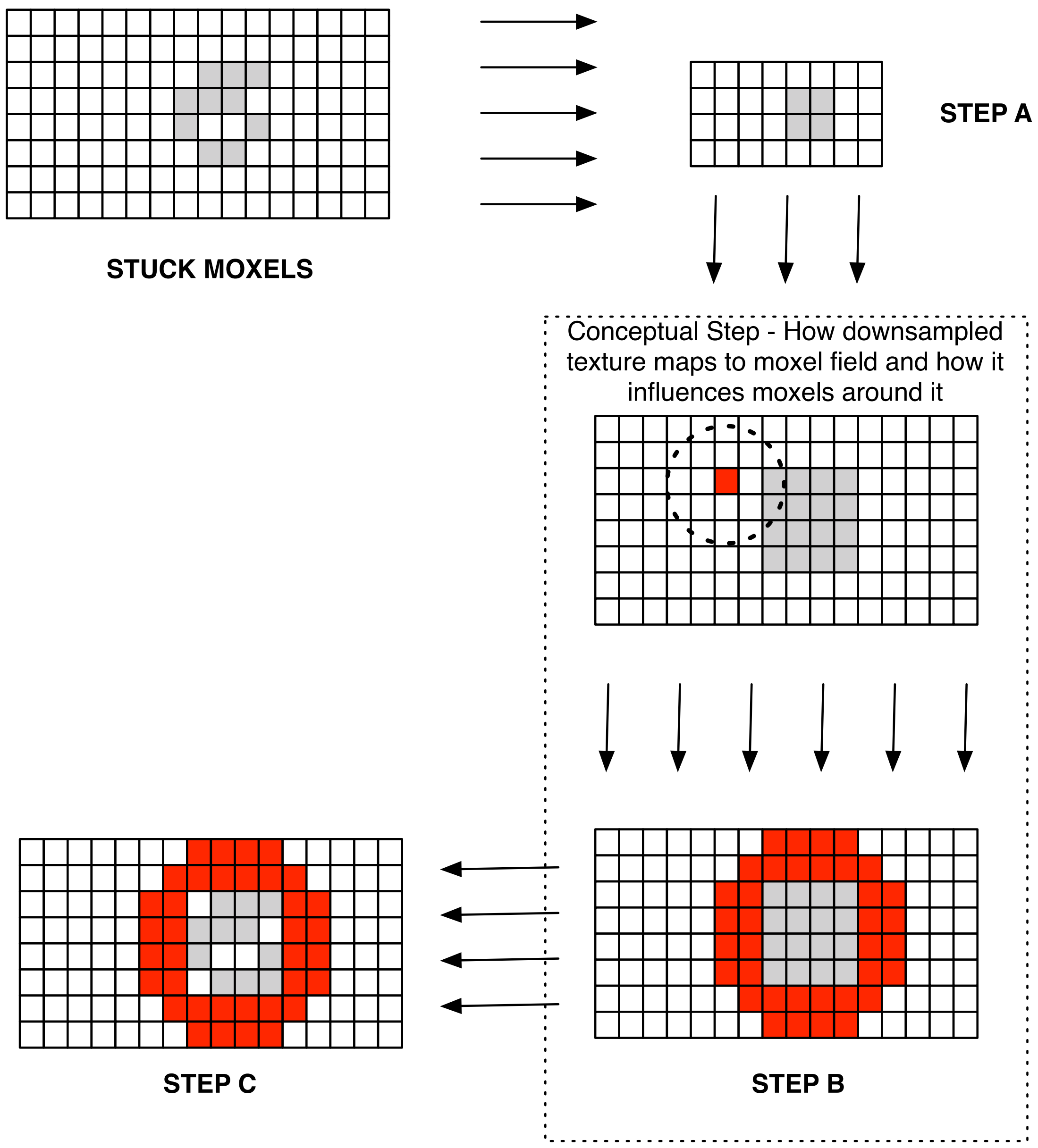
STUCK MOXELS
STEP A
Conceptual Step - How downsampled texture maps to moxel field and how it influences moxels around it
STEP C
STEP B

Figure 30. Addressing groups of moxels stuck in down position (grey color) by raising “fence” moxels (red color). Note that steps B is conceptual - in step C, non functional moxels can’t be rised.

Please note that in this particular case it is not a problem if moxel field is larger then the maximum texture size, as initial texture is just an input to the reduction step. If maximum size of the texture is smaller then the moxel field size, we would simply have initial texture where texel corresponds to the multiple moxels and is initialized with the number of moxels that failed in the given area.

Previous algorithm has one problem - it can split the region in such a way that we don't exceed threshold in two successive regions individually, but that the threshold is exceeded in a region that is shared by two adjacent regions, as on the Figure 31:

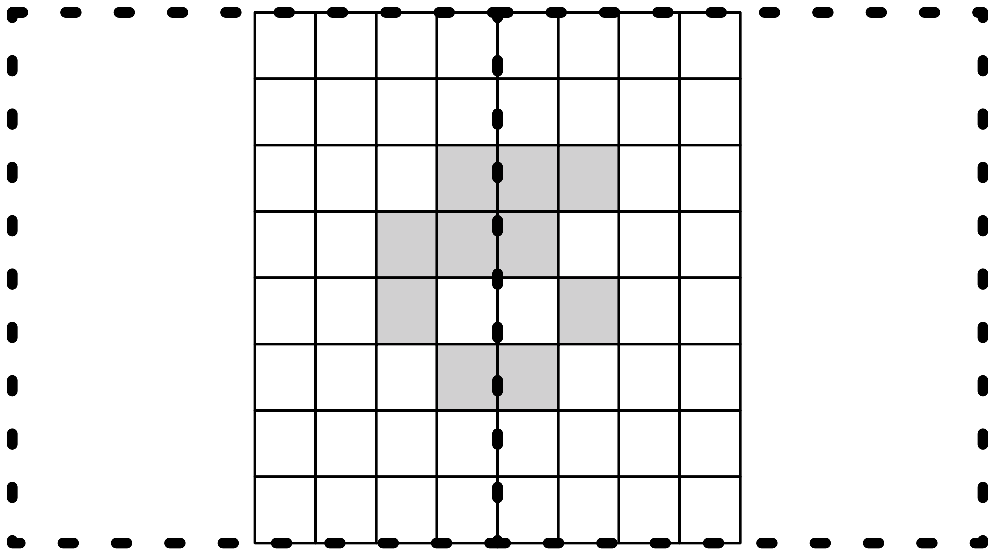

Figure 31. It is possible that a group of failed moxels can be split between adjacent regions so that it doesn't reach the threshold level in either of the two regions, although the number of failed pixels would reach threshold if they were part of a single region. Dotted line shows regions for threshold comparison.

In order to address this problem, we have following options:

1. Have threshold setup in such a way that we avoid the problem. In effect, we are adding a safety factor in thresholds by making thresholds smaller[72].

[72] This is the appropriate approach in situations when the chance of the moxels forming holes is small enough that a decrease in threshold doesn't create many false positives. However, this approach is heuristic in nature.

2. Use 2D approach inspired by Marching Cubes algorithm [166] in which we perform series of the successive passes over texture that are shifted for one pixel, as in Figure 32.

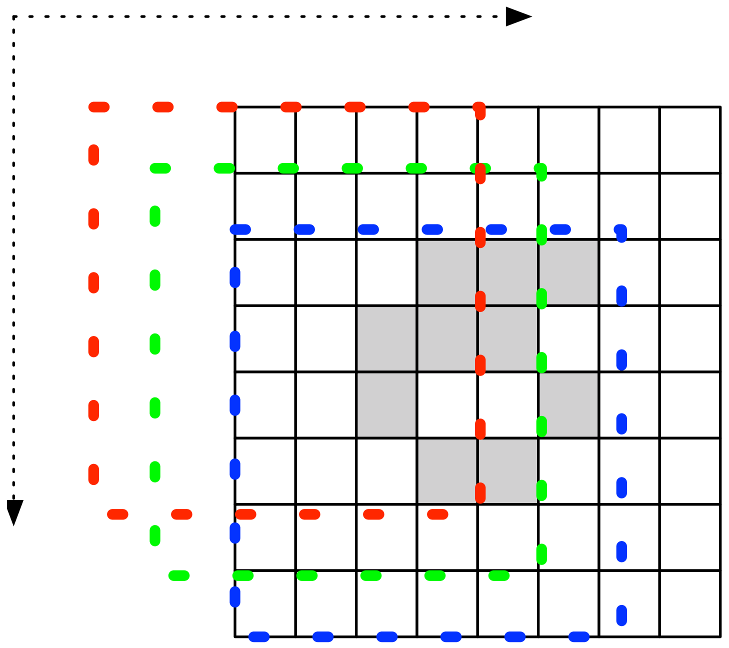

Figure 32. In order to ensure that large enough moxel group would always trigger detection, we would “march” detection region so that it covers every texel in the texture. Red, green and blue are three different regions we would examine. Note that only some detection regions were shown on the picture (there will be one detection region per every moxel that maps to the same texel in SL texture).

We describe the approach from Figure 34 in more detail:

Algorithm for Square Walk:

1. Output of the algorithm is final texture of failed moxel zones SL, as in the previous algorithm.

2. We would start by using previous algorithm and populating SL in it.

3. Now, we would use step inspired by Marching Cubes algorithm and shift the reduction region as shown in Figure 32. That would result in shifted region corresponding to between one and four texels in the final texture SL.

4. We would check in reduction step is threshold exceeded. If so, we would mark all corresponding texels from the previous step in the SL as failed.

5. Rest of the steps after construction of the SL would proceed as in the previous algorithm.

Previous algorithm would require N additional passes, where N is the number of moxels that fit in the texel in the SL. As this process needs to be repeated only when a new moxel fails in the down position[73], it is not expected that the time constraints will be a problem[74].

Again, as proposed algorithms modify the physical landscape, communication with the visual engine will be beneficial so that user can be shown visual representation of the obstacle.

[73] Previously discussed considerations for the hardware being able to address "in session failures" apply.

[74] And if it is we have an approximate heuristic approach described in this section that we can use to address the problem.

Finally, note that algorithms ***Algorithm for Failed Moxel Isolation*** and ***Algorithm for Moxel Fence Builder*** could be combined, so that the area around a moxel that failed in the partially up position could be fenced. In that approach, instead of building a cone around failed moxel with *Failed Moxel Isolation* algorithm, we can instead raise all moxels in the cone to their maximum extension, transforming "spike" into the small hole.

The question of comparing and contrasting these two approaches to addressing moxel failure is an interesting one. Building cone around failed moxels has advantage of providing gradual slope that allows user to still use areas of the Holodeck that have failed moxels. Fencing the areas with the failed moxels is likely to be safer in the case of the arbitrary moxel failure pattern (e.g. enough moxels failed that stable walk is impossible) and as such is a more conservative solution, but denies areas of the Holodeck to the user. Optimal balance between these two approaches to addressing failure of the moxel is outside of the scope of this thesis, though an interesting area for future research.

7.5.3 Addressing Moxels Near the User Position

In the Holodeck environment, moving moxels near user position could represent hazard to the user - we can cause the user to trip, as well as to lose balance. Although some use cases in which this is beneficial were previously discussed, it is generally a good thing to be aware of the user position and be able to modify moxel movement accordingly.

It is clearly possible (but wasteful) to consider all moxels within particular distance from user position potentially affected. Instead, we would propose an approach that significantly reduces the number of potentially affected moxels we need to consider, based on area user will walk into in the upcoming time interval.

The key observation in this approach is to notice that affected moxels come from two sources:

1. Moxels near the current position of the user - where user is represented by the terrain immediately around the user's legs and represents the requirement of stable ground which shouldn't be rapidly moved as it can affect user balance.

2. Moxels that will be in the user's vicinity in the immediate future. In addition to the current position of the user, direction and speed of the user movement are important as they provide us a region in which the user is expected to arrive soon. That region should be free of rapid movement of the moxels that could make it significantly different in physical rendering than what the user sees rendered on his visual channel. For the purpose of this discussion, we would assume constant speed of user's movement in the timestamp.

Figure 33 is showing previously described regions, in relation to user movement:

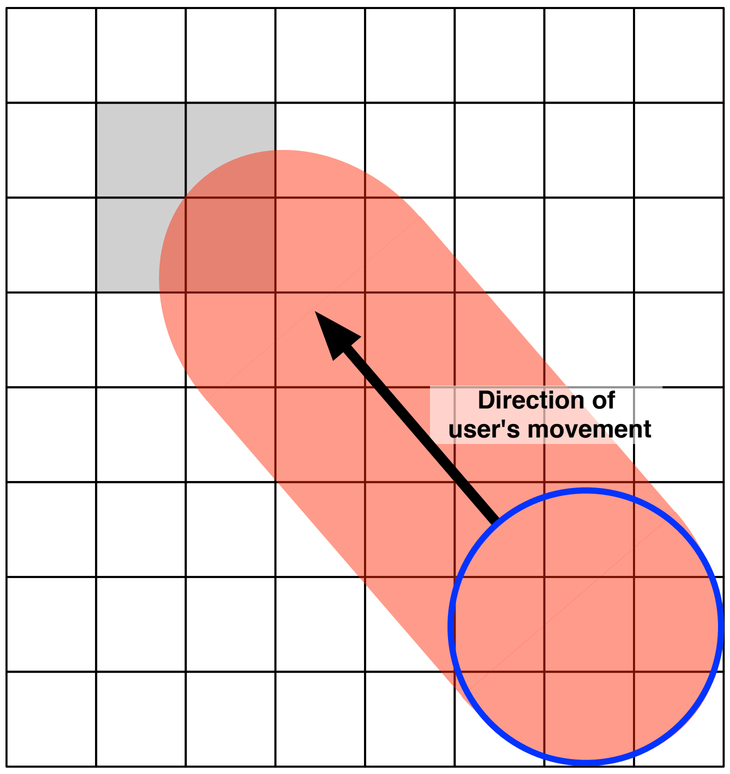

Figure 33. User affects two groups of the moxels - those around him (blue circle) and those that are near the path of his movement (red shape in the picture).

Both of those regions could be addressed in the fragment shader in the respect of limiting movement speed of the moxels. We would assume that moxel moving time is low enough that we could assume that maximum velocity with which the user could move in the direction of moxel is known, and notice that the problem of finding the moxels belonging to the shaded area is analogous to finding moxels that are at the center of the circle intersecting or touching line that is collinear with user's speed vector (with the origin of the line in the user's current position), leading us to the following algorithm to decide if a moxel is in the affected region:

Algorithm for Moxel Affected With User Movement:

1. Let t_m represent maximum amount of time needed for moxel to move to any position in the Z range from its current position.

2. Calculate an uniform variable line defined by the current position of the user and where user will be after time period t_m (based on its current speed vector) has elapsed.

3. In the fragment shader, find whether the circle representing the area occupied by the user (bounding circle for the user) intersects the previously defined line[75].

Previous calculations could be easily integrated in the fragment shader's calculation of the moxel position by passing information about user movement as a uniform variable to the shader[76] and using previous calculations to determine if the moxel is affected, and should we limit it's maximum movement speed.

In addition to the previous approach addressing stability of the ground in the user movement path, we can address smoothness of the terrain in the region M

[75] Intersection of the line and circle is well understood problem, and e.g. [167] describes one implementation which computational complexity is appropriate for the implementation in the fragment shader. Note that this approach is based on the assumption that user's path can be reasonably described with the line. Some more complex paths could be problematic to intersect with the circle within the fragment shader, but if that is the case, we could use bounding box around the path.

[76] See [37] for the description of the uniform variables in shader.

(corresponding to region shaded in red) to avoid having clearly defined obstacles that user can trip on, as on the Figure 34:

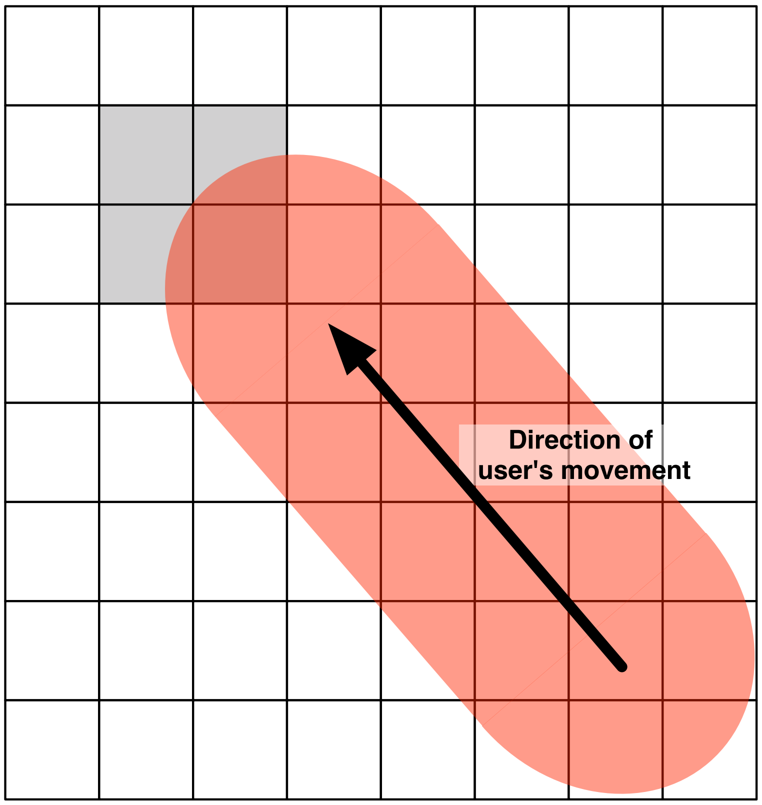

Figure 34. Obstacle in the user path (shown in the grey color) can cause user to trip. We can provide moxel level terrain smoothing to avoid tripping the user.

To address that situation, we could have following algorithm run:

<u>Algorithm to Detect Moxels Affected by User:</u>

1. Let Hu be current height of the user legs in Z direction.

2. For every moxel in region M calculate height above Hu. This could be done on the GPU on the fragment shader during the moxel Z coordinate calculation.

3. For each moxel for which height differential exceeds threshold, move that moxel to the height of Hu instead.

A combination of the previously described algorithms allows us to address physical safety directly in front of user. But these algorithms are working on the scale of the individual moxels and closely spaced groups of moxels and are not necessary appropriate for addressing large scale artifacts in the 3D models represented in Holodeck. The next section will talk about addressing those artifacts in more details.

7.5.4 Implications for Safe Geometry

Finally, it is clear that 3D models in which the user is moving have an impact on user safety. If the model is showing sharp edges on the larger scale (e.g. as on the Figure 35), although we can do moxel level processing operation to blunt edges, those operations would have negative influence on edges that should remain sharp and are not directly in the path of user (e.g. green objects on that figure).

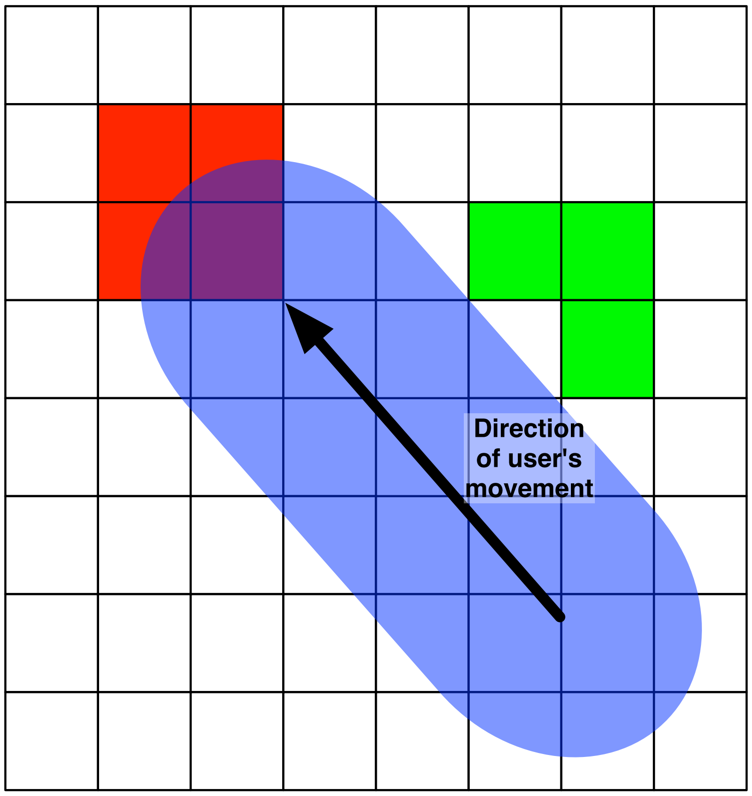

Figure 35. Obstacles in the user path (shown in the red) are more dangerous because of the direction of user's movement. Although of the same sharpness, green obstacles are not a problem because user is not moving toward them.

This example shows that some artifacts couldn't be addressed by just looking at the moxel scale - e.g. we would want to blunt the sharp edge the user is moving toward without affecting steps on the stairs in the picture. Although some moxel level safety mechanisms are helpful on the local level (as we saw in the previous section), it is important to understand that physical environment safety is an issue that likely couldn't be addressed in a fully automated way for every 3D environment. For example, the angle of the stairs could impact physical safety, and is significantly easier to address on the level of the 3D model than on the level of the individual moxels.

Another way to address physical safety of the user is to “relax” moxels that are near user when/if we detect that the user is falling, in order to “soften” the fall[77]. Modification of ***Algorithm for Moxel Affected With User Movement*** could be used to determining affected areas.

The take away message here is that to address physical safety issues, we need to review 3D models that are imported in Holodeck for physical safety, and that in some situations change in the 3D models is likely to be required in order to ensure physical safety[78].

[77] The author would like to thank to Professor Willem Schreuder for this insight.

[78] Although clearly outside of the scope of this work, there might be further research opportunities in automating identification of the features of the 3D models that could represent safety hazard for the user.

7.6 Software Control of Integrated Physical Systems

This section will present examples of the combination of the proposed algorithms with the various systems mentioned in this thesis. Please note that all diagrams are showing logical stages in the system, and that it is possible that some stages will be combined so that for example two logical stages on the diagram are implemented as a single pass on the GPU[79].

[79] How many rendering stages will be needed is primarily dependent on the GPU's hardware capabilities (e.g. number of textures that could be combined in the single pass). As we are going to control embedded systems (and would use known hardware for that control), these capabilities are known in advance. Software design of the shaders so that we can easily vary number of rendering passes is one possible area for future work.

Lets start with Figure 36, showing software control of a pure Holodeck environment integrating all the elements previously discussed in this thesis, as shown in the Figure 36.

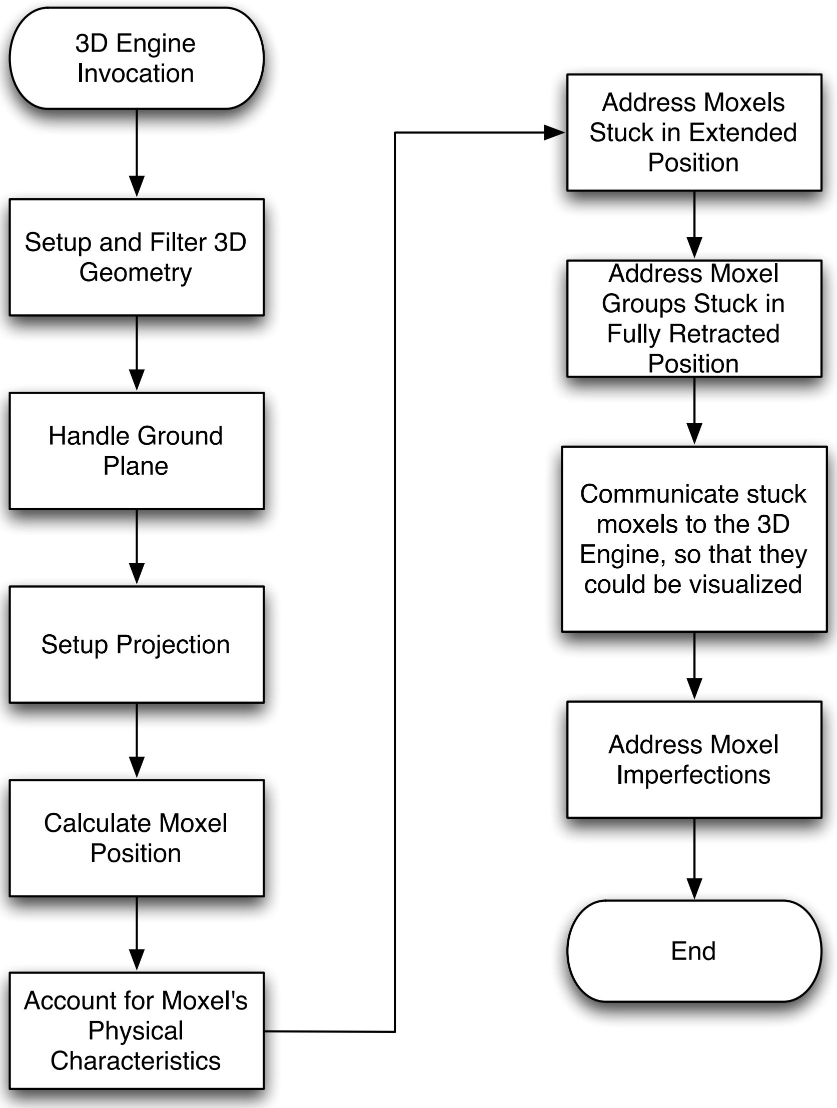
3D Engine Invocation
Setup and Filter 3D Geometry
Handle Ground Plane
Setup Projection
Calculate Moxel Position
Account for Moxel's Physical Characteristics
Address Moxels Stuck in Extended Position
Address Moxel Groups Stuck in Fully Retracted Position
Communicate stuck moxels to the 3D Engine, so that they could be visualized
Address Moxel Imperfections
End

Figure 36. Flowchart of the Holodeck. Please note that stages presented are logical stages - some of the action steps could be combined in a single rendering stage on GPU.

Figure 37 shows a flowchart for the combination of a Holodeck environment with a catoms [49] based environment.

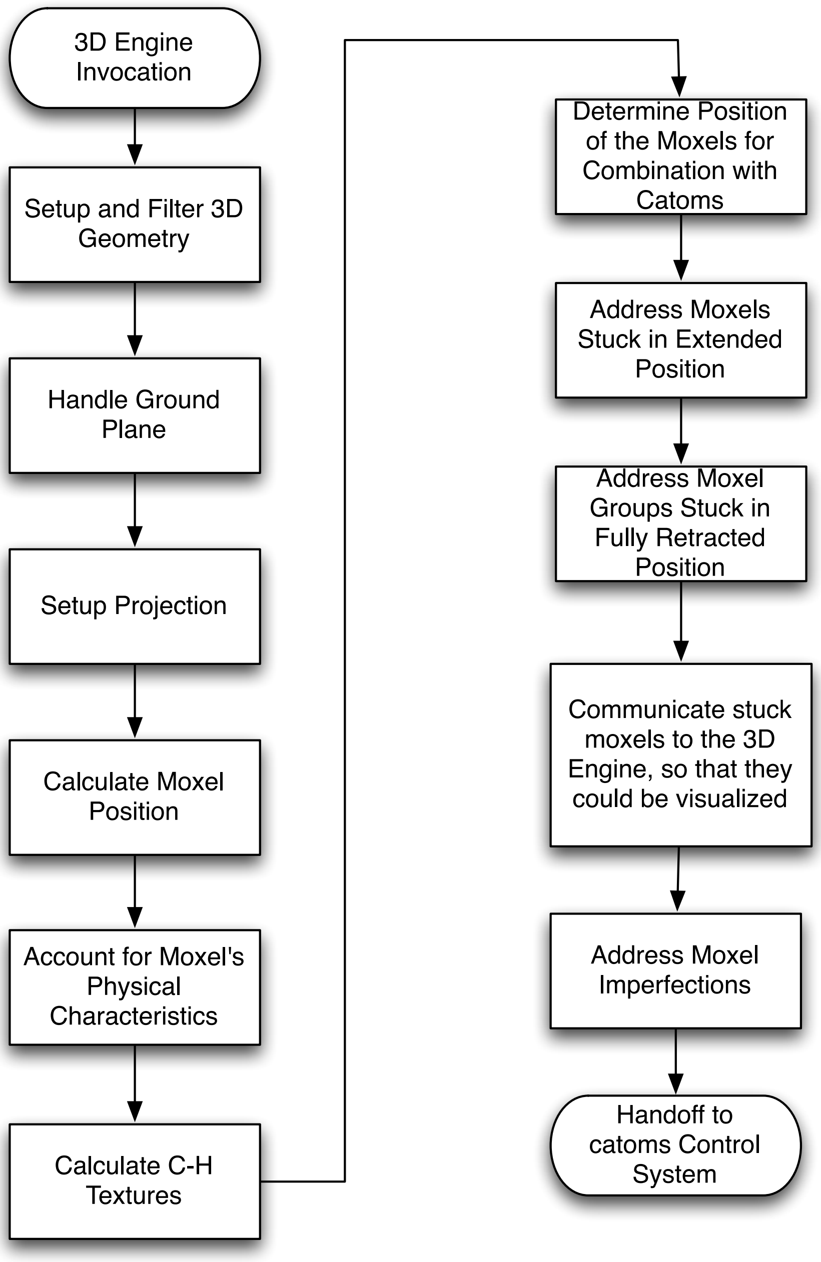
3D Engine Invocation
Setup and Filter 3D Geometry
Handle Ground Plane
Setup Projection
Calculate Moxel Position
Account for Moxel's Physical Characteristics
Calculate C-H Textures
Determine Position of the Moxels for Combination with Catoms
Address Moxels Stuck in Extended Position
Address Moxel Groups Stuck in Fully Retracted Position
Communicate stuck moxels to the 3D Engine, so that they could be visualized
Address Moxel Imperfections
Handoff to catoms Control System

Figure 37. Flowchart of the pipeline for the combination of the Holodeck with a catoms environment. Please note that stages presented are logical stages - some of the action steps could be combined in the single rendering stage on GPU.

Figure 38 shows a combination with a MEMS based system that is capable of only bistate (up/down) positions:

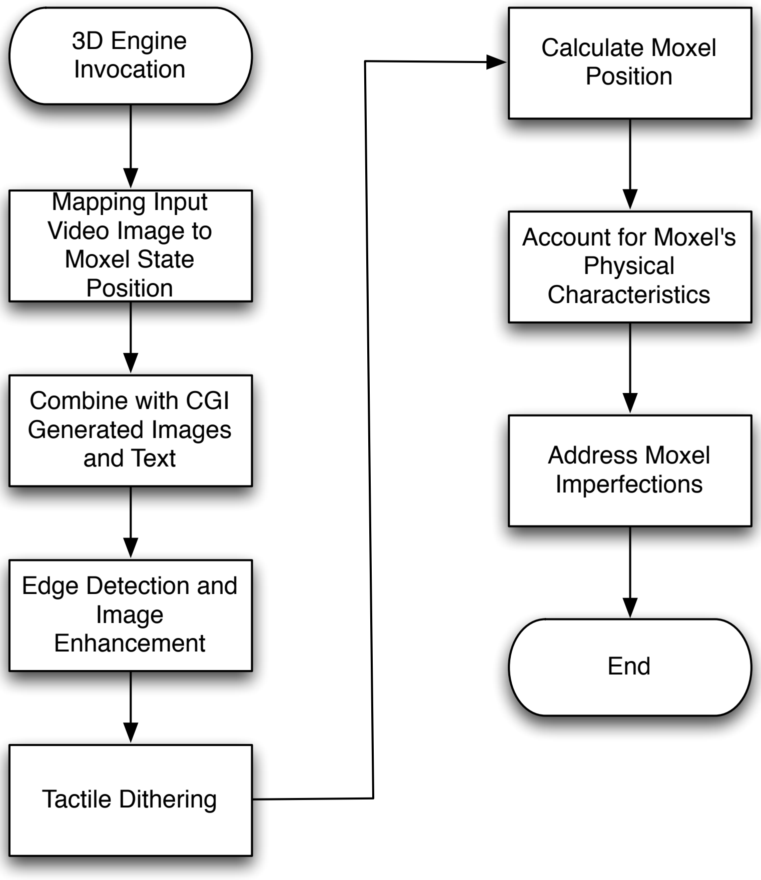
3D Engine Invocation
Mapping Input Video Image to Moxel State Position
Combine with CGI Generated Images and Text
Edge Detection and Image Enhancement
Tactile Dithering
Calculate Moxel Position
Account for Moxel's Physical Characteristics
Address Moxel Imperfections
End

Figure 38. Flowchart of the pipeline for the control of the bistate capable MEMS system. Please note that stages presented are logical stages - some of the action steps could be combined in the single rendering stage on GPU.

Chapter 8 - Uses of GPU In CPS Beyond Moxel Position Control

Previous chapters have shown that an OpenGL based approach to controlling moxels is a viable and scalable approach to the control of tmoxel based systems. This chapter will build on previous work to expand upon and discuss GPU use in the more general context of robotic control.

8.1 Relation of the Proposed Approach to OpenCL and CUDA

In addition to the OpenGL based API, many modern GPUs can be programmed using CUDA [157] or OpenCL [140] or even combination (as done by Apple's Grand Central Dispatch [168]). OpenCL and CUDA represent stream based approaches to data processing using GPU and represent memory hierarchy in a way that is more familiar to programmers used to distributed and parallel systems.

Although these approaches are very well suited to general purpose computation on the GPU, in the case of the Holodeck and moxel control, an OpenGL based environment is much better suited because our problem domain maps naturally to the use of the Z-Buffer based approach for the calculation of the moxel position. If we were to use OpenCL or CUDA based approaches, we would have to reimplement Z-Buffer logic on those approaches, and it is much more natural, elegant and likely faster to use existing capabilities and combine Z buffer based position calculation with the output using the Render Buffer extension, allowing us to render outputs in frame buffer instead of the color buffer[80].

[80] Author would like to thank to Professor Willem Schreuder for pointing me to the Render Buffer based approach.

However, it is possible that some steps that were framed in the OpenGL based formulation in this thesis could be implemented on a OpenCL/CUDA based platform. In particular, scatter and reduce based steps [38] could be implemented in OpenCL and/or CUDA based formulation.

8.2 Geometry Shader Use

The latest generations of graphics card have an additional shader unit – the geometry shader - that is capable of producing [158] variable length output from the shader. This section will briefly discuss some advantages of the use of geometry shaders in the context of moxel based algorithms.

One of the main advantages of geometry shaders is that they are able to implement efficient scatter operation [38] and variable length output on GPU [158].

In the past, in the area of general purpose computing on the GPU, some examples of geometry shader use are computer vision (corner detection, Hough transform) and histogram building [158]. In the context of moxel based systems, they could be used for:

1. The more effective detection of the moxel regions that are stuck in the "down" position by eliminating need for the reduction steps in the failure detection algorithms - in effect, we would directly output fence geometry as a circle around the region of stuck moxels[81].

2. More effective building of the physical fences around failed moxels, instead of use of the ***Algorithm for Moxel Fence Builder*** - we would not only "fence failed moxels in" based on the fragment based calculation, but directly define geometry of the "fence" in the geometry shader in the subsequent rendering pass.

[81] Clearly, similar approach would apply to the regions that are stuck in the up position (if we are to build fence around them).

3. It remains an area for the further research which parts of the unsafe surface geometry for the user on the level of the 3D geometry could be detected on the level of the geometry shader.

4. As indicated in [158], geometry shaders could be used for minimizing the size of the output from the GPU that should be read by the CPU (e.g. by outputting to only small region of the framebuffer and reading only the portion of the frame buffer, as explained in the [158]). This allows faster communication of information about failed moxel regions.

8.3 Transfer of Results From the GPU

The question of the best mechanism for the transfer of the calculated data from the GPU to the mechanical system would depend on the implementation of the mechanical system used but is definitely an engineering as opposed to a fundamental research problem. This is particularly important because the bandwidth of the data transfer from the GPU to main memory is currently significantly lower than from the GPU accelerator memory to the GPU itself [39], so repeated transfers from accelerator to CPU memory would have an unnecessary performance impact[82].

With the previous in mind, the following approaches for the transfer of the calculated state are possible:

1. Highest read bandwidth with the smallest impact on the GPU performance would likely be achieved if we use the mechanism that current GPUs are already using, namely output of the color buffer to monitor (e.g. on DVI port [126]). This approach has the significant disadvantage of requiring decoding of the signal on the physical system side and is generally mentioned only to show that output is possible at the full speed even with the current generation of GPU hardware. Furthermore, this approach requires us to use resolutions that are supported by the display hardware[83].

[82] Note that our performance tests were done with the readback of the rendered buffer from the GPU and calculation on the CPU, so our results showing viability of the GPU based calculation of the moxel position take this effect into account because the time to read Z buffer is included in the performance measurement as part of the moxel calculation time.

[83] The author wishes to thank to thank to Professor Willem Schreuder for this insight.

2. A more appropriate approach for systems consisting of a few million moxels might include reading values from the graphics cards buffers and using network (e.g. 10 Gigabit ethernet) for the transfer of the data [169], with possible aggregation of the several ethernet links to achieve higher bandwidth [170].

3. As previously discussed, some parameters of the output (e.g. when we are not needing to export position of all moxels but just few parameters like failed moxels) could be communicated using geometry shader [158].

4. Finally, an interesting question is whether we could combine OpenCL/CUDA and GPU based computation without requiring transfer back and forth to the main memory[84]. This is likely possible on the level of existing hardware[85], and there are APIs in both OpenCL and CUDA that allow shared buffer between OpenGL and CUDA[86] (see e.g. [157] and [171] for the details of the APIs). It will be an interesting opportunity for future work to extend our framework so that we can combine OpenGL and OpenCL based computation of the same model That way, we would have ability to use OpenGL for the problems that are best formulated in the context of computer graphics, and use OpenCL for the problems that are better formulated in the context of the SIMD based processing.

[84] Author would like to thank to Professor Willem Schreuder for challenging me to think more about this question.

[85] After all, we just need bits and bytes that are already in the memory of the accelerator card.

[86] Note that CPU and GPU don't directly share memory even when this approach is used - any transfer would still need to happen over bus and there is a need for the copy. That copy, although fast, would be significantly slower then direct access of GPU to accelerator memory [39]. As a result, shared buffers that minimize the need for data to be copied would help.

8.4 GPU Use for Addressing Physical Feedback of the System

Extension of the GPU to take into account physical feedback of the system is certainly possible. Accounting for the user's feedback is a central theme of multiple previous systems including [84], [85] and Digital Clay [15], [16], [17] systems which allow user's manipulation of the surface based on the user force applied. The following section will propose how some of the techniques already pioneered by these systems could be extended to be used in the general case of reacting to any feedback on the GPU.

Previous work in the literature shows how a height map of the current actuator position could be passed to the system and used by the system (as discussed by [85]). There is no reason to limit that technique only to the height map - we could pass any feedback from the physical system as one of the input textures to the fragment shader, and the fragment shader could account not only for physical characteristics of the moxel in isolation, but of the physical characteristics of the moxel with the appropriate external forces being applied to it.

Note that use of the GPU is not limited to addressing force feedback on the level of the fragment shader - higher level integration with the rendering engine would allow for the addressing of the physical forces on the system by modifying geometry before we ever reach GPU - e.g. it would be possible to have a system in which the user manipulates some elements of the system using physical force and modifies the environment to account for that movement (one easy to foresee use case is user pressing down on the moxel wall that would cause that moxel wall to move, as done by the Relief [84], [85] and Digital Clay [15], [16], [17] systems).

Finally, it might be possible to use physical feedback of the system to the user in order to address presentation of the environment. For example, somatogravic illusions allow us to "trick" the body[87] so that upward acceleration could be perceived as linear and vice versa[88], as experienced by pilots [172] and exploited in flight simulators [173].

[87] Author wishes to thank to Professor Willem Schreuder for pointing to similarity and relevance of the somatogravic illusions.

[88] However, note that this might require hard real time control (guarantee that mechanism would always move by the particular deadline), and that additional mechanisms past the pure GPU control might be needed to address hard-real time aspect of this problem, as discussed in the section "GPU's Suitability for Various Areas of Use".

8.5 GPU Use for Decimation for the Purpose of Visualization

Although in the current system we are doing decimation of the output of a large number of moxels for the purpose of visualization on the CPU, it is possible to do decimation of the existing moxels on the GPU by using reduce-like output [38]. The GPU could be used to perform various image space transformations as described in [143] and the extension of multiple digital image processing algorithms described in [129] is likely possible on the GPU (e.g. [174] describes how image segmentation algorithms could be implemented on the GPU).

With the previous being said, we need to keep in mind what is the possible purpose of decimation:

1. In this work, decimation was used for the exclusive purpose of visualization so that the demo of a system that calculates a large number of moxels is not slowed down by the visualization component of the simulator. As this is not fundamental work for our research, there was not much point in running it on the GPU (and this functionality would likely not be needed on the full physical system).

2. In the larger context, as previously pointed out, the question of the best anti-aliasing methods for the moxel based system is still an open research question. In this area, use of the GPU is likely to be beneficial, but as previously pointed out, this area is outside of the scope of our research.

Based on the previous, we decided not to implement decimation of the moxels on GPU in this version of the simulator.

8.6 GPU's Suitability for Various Areas of Use

Based on the results presented in the previous section, what can we conclude about GPU use for the control of large scale moxel based systems? In which situations does a GPU represent a better approach than CPU based control?

There is no question that the raw power of the GPU in the terms of the memory bandwidth and computational power is higher than for the CPU, and that as such GPUs hold huge promise for computational class of problems. In the problems that lend themselves to parallel processing, GPUs are clearly superior [39], [40], [174] to the CPU. This works shows that problems of moxel controls fit clearly in that category, and that furthermore GPUs are especially well suited to the control of moxel systems due to the applicability of Z buffer based control to the calculation of the moxel position, and general ability of the GPU to address areas of moxel position calculation and the ability to address physical characteristics of the moxels.

Furthermore, we have shown that a GPU based approach to moxel control extends naturally to combinations with systems like CirculaFloor [3], [10] and catoms [49], [1] and that GPU based control is applicable to controlling multistate and bistate MEMS based systems.

At the same time, this is early work on GPU based robotic control. There are areas of control for which the GPU is not well suited, or needs further research in applicability.

Those areas are:

1. Theory of scheduling of the hard real time control is well developed on the CPU. For example, [175] and [176] allow us to prove the schedulability of a set of tasks scheduled with a preemptive scheduler. On the contrary, the mathematical framework for scheduling on the GPU is at best in the very early stages of research and is currently limited to soft real time aspects [177], with no work on hard real time, that the author is aware of. Consequently, in systems in which not finishing computation by a particular deadline would result in the catastrophic failure of the system or harm to the users, the GPU will be need to be combined with traditional real-time capable control. In the case of Constrained Motion Cyber-Physical Systems this problem doesn't inherently present itself as we are not using moxels to actively balance the user, and the only limitations are moxel hardware imposed[89].

2. Similarly, in the problem domains that are dominated by the requirement for minimal latency of movement, GPU latency characteristics are less well understood then CPU latency characteristics[90].

[89] For all other issues, we can fallback to the anytime type of the algorithm in which we leave moxel where it is if its position is still not calculated. For addressing limitations of the hardware of the single moxel, on the level of movement of the single moxel, we could have GPU holds position calculation of large number of moxels, and on the level of the moxel employ hard-real time control that might be implemented by having microprocessor/custom hardware controlling just a hard-real time aspects of the group of moxels. Again, this is limited only on the very small subset of hard-real time functionality if any is needed for the individual moxel and could be done on the level of the individual moxels, still benefiting from the scalable calculation of moxel position on the level of the whole system. Note that software control system for the Digital Clay [16], [33] proposed layered system of the software control, with the rod control layer being separate layer from the control of the calculation of the rod position and that such system would work well for the hard real time constraints that could be addressed on the lower level of software.

[90] That is another way to say that hard real time domain is better understand on the CPU.

3. If massive synchronization among multiple moxels is needed, the GPU is at a disadvantage to the CPU [39], [40]. So for the problems in which there is asymmetry in the control system of the physical moxel implementation that impose constraints like "moxel X must not be calculated before calculation of the position of moxel Y is completed", a GPU based solution is less suitable than CPU. Note that this is limited on the calculation - there is no problem if control system imposes actuation constraint (e.g. refreshment in the scan lines like approach proposed in [33] is completely acceptable).

Although previous problems are significant, they are today endemic in the whole space of massive coordination of large numbers of robotic elements. It is unclear how and if any of the work presented on algorithms for the movement of the large swarms of elements (as in e.g. [58], [55] and [56]) could be extended to the hard real time domain. For that matter, GPU based algorithms have the advantage of the ability to understand position of every element of the system on the global level, so analysis of e.g. stability of the user standing on the moxel surface as well as of the completion of the whole frame prior to deadline is likely to be easier to perform than of the user standing on a group of catoms that are not even under the central control[91].

With the previous limitations in mind, it is fair to say that we have shown that GPU based computing of the moxel position is viable alternative and a field offering very interesting research options in the future.

[91] Previous observation reflects negatively on the whole idea of the decentralized control, as it likely or puts you in position that you have to impose limitations on the movement of the catoms as proposed in [56] and [55] or force you to account for the impact of next move Catom is about to make on the whole structure.

8.6 GPU Use for Power Saving In Mechanical Device

Among other physical characteristics of the moxel that could be taken into account, power consumption of the moxel is particularly interesting for portable systems[92]. An interesting thing about the power consumption is that it could be both a local (per moxel) and a global phenomenon (multiple moxels among the picture).

On the local level, it is possible that different methods of actuation of the moxels would have different power characteristics (e.g. faster actuation using more power), and the previously described algorithm for determining the moxel zone that is next to be encountered by the user is beneficial in calculating urgency with which a moxel should be moved to achieve its final position.

On the global level, we have a question of how we could achieve minimal energy expenditure for the transition between two moxel arrays states. This approaches has both a spatial component (among all the moxels in the model) and time component (frame to frame coherence) and it is an open research question how well GPUs would map to those problems. As two states could be considered images, image comparison algorithms between current and desired frame could be used for comparison of the states on the GPU (similar to the proposal of comparison of the height maps proposed in [85]) and as such would give us starting position for the energy minimization algorithms that could be run on the GPU. With that being said, that research, while interesting future research topic is outside of the scope of this PhD.

[92] Author would like to thank to Professor Yung-Cheng Lee for the suggestion to look in the power consumption further.

8.8 Adaptive Resolution and Distribution Among Multiple Users

There is a significant set of work in the computer graphics field dealing with adaptive resolution rendering for the purpose of graphics acceleration - for some examples of that work, see e.g. [178] and [179].

Extensions of that work, combined with the work presented in this thesis of determining areas of the PRE that are going to be accessed by the user next allow us to use adaptive resolution for moxel control, in which different resolutions are used to calculate moxel positions in the immediate surrounding area around the users, while calculating other moxel areas in lower resolution[93].

Note that clearly this adaptive resolution scheme could be hierarchical, with areas very close to the user needing highest resolution and areas further away needing progressively lower resolutions, as a function of the distance from the user and maximum moxel movement speed. This is an interesting extension for further research, and as a basis for it we can use the following algorithm:

Algorithm Adaptive Resolution Moxel Calc:

1. Use ***Algorithm Moxel Affected With User Movement*** to detect moxels affected by user movement, based on the maximum time that the moxel would need to move from its current position to any other position in the range.

[93] Question might be asked why calculate position of the moxels in the area not in the proximity of user at all, and the answer here is that approximate positioning of the moxels is likely to help them reach target position earlier when user ultimately move in that area.

2. If we are controlling a Holodeck environment with multiple users, then clearly previous calculation has to be done taking into account every single user.

3. In the regions where we find that moxels are not affected by the user(s), we can raise only a portion of the moxels, e.g. every fourth or eight. (e.g. to affect line of sight and erect obstacles between users). An approach similar to the one described in ***Algorithm Tactile Dithering with GPU*** could be used to provide dithering, based on the average height and slope of the terrain that should be shown in the region.

Previous algorithm could be used to reduce power consumption of the system, too. Note that although in its previous form algorithm was presented for use in the large scale, Holodeck type of environment, the algorithm could be used to provide adaptive resolution for a MEMS based tactile system - we would treat every finger as a user in the previous algorithm and provide finer image to the fingers that are “scanning”, while most of the screen is rendered in the lower resolution.

Finally, note that if adaptive resolution is used in a large scale system occupied by the multiple users, we clearly have to consider position of every single user in an area as an area where we should do high-res rendering.

Chapter 9 - Future Work

This PhD thesis addresses important practical problems of software control of Constrained Motion Cyber-Physical System environments consisting of millions of elements. Although this work is important to enable the construction and practical applicability of those systems (and as such represents a tangible contribution) it is just an early contribution in what is likely to be a field with very significant research potential.

9.1 Mobile Systems as Assistive Technology

The combination of MEMS based systems with a portable device opens a set of interesting possibilities for larger use cases in which proposed MEMS based systems could be used as assistive technology for users that are visually impaired. Although this is an area of future work that goes outside of the scope of this thesis, it opens exciting research possibilities that we will discuss in this chapter.

The current situation of assistive technologies for blind users is not encouraging, with many of the devices that are on the market being rejected by blind users due to their limitations and the need for better interfaces for communicating presence of obstacles [180]. As an example, systems that use sound to indicate the presence of an obstacle seen on the camera interfere with hearing [180], and overview of the tactile pin arrays [27] quotes multiple tactile vision substitution systems like Optacon that were using camera to present tactile image.

A key observation here is that for most manufacturing processes, the cost of the unit produced decreases with bigger numbers of units being produced[94]. That

[94] Because, among other things, fixed costs gets absorbed among larger amount of units.

means that assistive technology that is useful to sighted users will be cheaper than technology that is useful for only visually impaired users. So if we can produce system that is useful to the sighted users, it would eventually result in cheaper assistive technology for the visually impaired users.

As discussed elsewhere in this dissertation, tactile screens for visually impaired users could present graphics in a way that could be perceived only by touch. This is clearly very useful to the people who are visually impaired, but is also useful to sighted users for the following reasons:

1. Current touch screens do not lead themselves well to the touch typing or dialing phone numbers without looking at them, so physical keyboards are still beneficial[95].

2. However, physical keyboards take additional space and are fixed, showing only one set of keys. Virtual keyboards have the advantage of better using space available[96].

3. The previous point could be addressed with a combination of the tactile and visual screen, and early works in that area are Dynamic Displays [89] and TeslaTouch [5]. However, combining dynamically changing surface and display is clearly more difficult then just providing dynamic surface by itself.

[95] Blackberry Torch is is combining physical keyboard with the touchscreen [181].

[96] E.g. Apple's iOS in some applications presents only numerical characters on keyboard for numerical input, making space for every key much larger then it will bepossible if we were representing all alphanumerical characters on the same space [182].

Furthermore, TeslaTouch is not capable of providing tactile feeling to a stationary finger, and requires the finger to move [5].

4. However, combining MEMS based surface on the back of the phone, so that user could touch reverse side of the phone to avoid occlusion of the display by fingers e.g. having two touchscreens on front (LED based) and back (MEMS or electrovibration based[97] is clearly simpler to implement with currently available technology. This is a modification of the ideas proposed in [162], [184], [185] and [186] in which various mechanisms were used to help user manipulate touchscreen by touching the back of the surface).

5. There is early work on organic user interfaces in which computers could take any physical form. It is conceivable that they would have areas that lead to feedback over actuation of moxel like systems described [187].

So we have use cases that lead us to believe that we might have touch phones with MEMS based surfaces being mass produced at some point in the not too distant future. Let's recapitulate how such a smartphone is likely to look:

1. It would have a tactile display on the back.

2. It will be equipped with a GPS chip[98].

3. It would have GPU with a programmable graphical pipeline.

---

[97] Author would like to thank to Professor Yung-Cheng Lee for sharing idea of touchscreen being useful to the sighted users to feel textures of the surface.

[98] At the time of this writing, many popular smart phones like current flagship models from Apple [88], Motorola [183] and RIM [181] are equipped with the GPS.

4. It would have map information related to user position accessible.

5. It would have CPU, memory and wireless connection speed that are comparable with the what desktop computers had a number of years ago.

In itself, none of the previous ideas and technologies are new or original. However, technology is finally catching up to the point when all of the previous ideas are likely to soon be available in a package of the size of a mobile phone. Taken together, they provide significant potential for a visually impaired person, and are interesting areas for future work.

So what can we do with such a system? Some of the capabilities of such systems are already shown by products that use GPS and focus range information available from the camera to determine what object the user is looking at and provide virtual tour of it, as done in e.g. [189]).

We can combine that functionality with the use of the software algorithms presented here to show output from camera on the tactile display on the back or torso (one such system presenting output on the user's torso is discussed in [73]) or even on the phone display itself, while at the same time using GPS position and rangefinder information available from the camera. All of this without requiring hardware modification of the device (smartphone with the tactile display) that we expect to become ubiquitous and cheap at some point in future.

Furthermore, note that there is work in progress for MEMS based stimulation of the retina in the blind users [114] and that it is still not clear what will be the best software control of such devices. We submit that once MEMS based epiretinal stimulation devices reach sufficient density of MEMS elements, GPU based control will be relevant for them too.

This combination is an interesting area for future research. Its ultimate potential is to be a useful substitute for guide dogs and as such, this vision could be considered logical extension of the work already being done in smartphone based assistive technologies, as discussed in work done by [190] on vibrotactile sensor on the back of the phone and [73] for the combination of the camera with a wearable tactile feedback display.

Furthermore, it is interesting to notice that benefits of the tactile displays exist not only for the people with visual disabilities, but that tactile feedback is useful for people who are able to see normally. It was already demonstrated that even people who see normally benefit from the tactile feedback for touchscreen typing [191]. Note that these benefits of touch are not limited to just sense substitution - sense of touch is faster than sight, allows consecutive stimuli that are only 5ms apart [191] and tactile GPS units were shown to significantly help navigation of soldiers in the challenging navigation environments, while reducing workload for user [118]. Furthermore, various locations were investigated for positioning tactile feedback devices (waist, ankle, wrist) [192].

9.2 Automated Tactile Translation for Haptic Systems

Although one possibility of producing tactile images is by mapping existing graphics image to the tactile device (e.g. as proposed in [73]). While this is the most obvious approach for sensory substitution and a good value for people who are gifted with normal sight (as it allows them to touch what they can see), there are indications that for visually disabled people modification of the image leads to better recognition performance:

1. Image manipulation based on image enhancement algorithms helps the people who are visually disabled to better perceive the image [76], [75]. As mentioned elsewhere in this thesis, GPUs are a good solution for those image enhancement operations.

2. In addition to the pixel level simplification, there are indications that for the best tactile perception of the images by visually impaired and blind users, haptic drawing has to be different than just direct translation of the visual picture to the haptic surface, and that changing the shape of the picture in a format that is different then classical visual drawing helps to better perceive the image [193].

Combination of the previous factors indicates that there is significant advantage in using GPU hardware for the processing of haptic images and that image level operations definitely help. However, it might be necessary to do more than image level operations for best presentation of the images to users that are unable to see and those operations can’t be performed in the image space or GPU alone, as they require change at the level of the 3D model. These operations fall outside of the scope of this work, but are mentioned for the sake of completeness of this work.

9.3 Other Areas for Future Research

Hardware devices consisting of millions of elements would have significant transformational potential, in respect not only to research in one area, but multidisciplinary research and ultimately the world around us:

1. Assistive, MEMS based technologies could restore part of the ability to perceive graphics and images to users that are visually impaired.

2. Previous devices are not only limited to assistive technologies, but might allow sensory substitution in general. E.g. can we provide tactile feedback to users who are not visually disabled? E.g., can we get information to pilots or drivers using not only sight and sound as we do today, but by e.g. changing tactile feedback they get? Some early work in this area is [72] as well as work summarized in [24].

3. Integration of assistive technology with mobile phones and applications that provide location aware augmented reality (like [194], [189] and [195]) would provide significant additional help to visually impaired people and could potentially augment some of the functions of a guide dog.

4. Holodeck based PRE as well as related locomotion research that could be combined with PRE (e.g. CirculaFloor [10]) will be nothing less than the next step in the evolution of immersive environments, allowing us not only to see the

environment, but to physically move in it with minimal limitations on the type of movement the user can do. This would bring us closer to the early visions of computer graphics pioneers [29].

In order for previous visions to happen, much additional research will be needed and would involve disciplines like Computer Science, Mechanical Engineering, Electrical Engineering, Psychophysics and others.

Some examples of future work relating to Computer Science are (in the area of portable Constrained Motion Cyber-Physical Systems):

1. HCI for tactile interfaces and work relating to MMHCI [196]. What is the best way for the user to interact with a high resolution tactile interface?

2. What are the requirements of a smartphone based virtual layer [194], [189, 195] for assistive use of visually impaired users? What other location aware services could be presented to the users?

3. Work in the area that best presents information for sense substitution for users that are not visually disabled, in order to enhance their situational awareness. Could we provide tactile instruments to e.g. pilots that would provide tactile feedback from the plane's instruments? Could we make the users feel slight physical discomfort if instruments are indicating dangerous situations? What are the best ways to do that? What are appropriate situations in which to use these facilities? What are the best HCI approaches here? [118] evaluates tactile

based navigation in challenging environments and surveys some of the previous uses of tactile feedback.

4. What are the best HCI approaches for Dynamic Displays [89] and TeslaTouch [5] for users who are not visually impaired? How can we combine a GUI with tactile interfaces once that (or similar) work progresses to the point that it is able to physically deform individual pixel?

5. How do we control output from multiple tactile feedback channels (e.g. mechanical, thermal, vibration, soundwave based)? Need for the mutliple output channels is demonstrated with systems like tactons that combine rhythm, roughness and location to convey tactile information [127], [128].

Note that benefits to the user of tactile feedback were demonstrated in a number of applications for users that have perfectly normal health (e.g. [118] points to the examples of the use of tactile feedback by pilots under high-G force and astronaut orientation and reports results of soldier navigation in challenging environments). This indicates that a potential area of further research is extending proposed GPU algorithms in the area of the HCI interaction over touch.

Opportunities for the further research are equally exciting in the area of large scale Constrained Motion Cyber-Physical Systems. These are just some examples:

1. Although we discussed some basic approaches to addressing user safety in a Holodeck type of environment, the larger question of how to ensure the physical safety of the users in the environment remains. Are all transformations

of the surfaces safe or are there limitations? What is the best way to enforce those limitations?

2. Related to the previous, how can we address mechanical failure in the system in a way that ensures physical safety of the user?
3. What is the best approach to providing immersion and tactile feedback? What is appropriate slipperiness for icy cave versus sand beach? What is the elasticity of the grass versus mud?
4. What is the best way to present realistic places in the Constrained Motion environment?
5. What is the best approach for networking multiple PREs (in order to allow for distributed training)?
6. What is the best way to address temporal aliasing problems and tactile aliasing in a PRE?

Finally, when we are talking about large scale systems with many moving elements, we have shown that the system could address tens of millions of moxel calculations per second on commodity class hardware. However, even larger Cyber-Physical Systems could be on the horizon - e.g. if we are talking about Carbon Nanotubes based systems [22], then we might be talking about even larger numbers of elements that need to be controlled.

Combination of the techniques proposed in this work with the techniques used for the cluster rendering for the large screen displays [197] is an interesting area of future work as it would allow us to control even larger systems.

In conclusion, this PhD thesis is addressing important problems, but is not a self-contained work, that is an end in itself, with no potential for future work. Quite the contrary, the area of software control of million element Cyber-Physical Systems is likely to present many opportunities for further research in the years to come.

References

1. T. Geller: “Shaping the Future”, *Communications of the ACM*, 52, 2009, pp. 16-18.

2. S. C. Goldstein and T. Mowry: “Claytronics: A scalable basis for future robots”, *Robosphere*, November, 2004, pp. 1-6.

3. H. Iwata, H. Yano, H. Fukushima and H. Noma: “CirculaFloor: A locomotion Interface Using Circulation of Movable Tiles”, *Proceedings of the IEEE Virtual Reality Conference 2005 (VR 2005)*, Bon, Germany, 2005, pp. 223-230.

4. B. Britton, C. Rogers, M. Rodriguez and A.Moklestad: "BrailleEye", *University of Colorado Senior Project team working under direction of Professor Y.C. Lee*, 2009.

5. O. Bau, I. Poupyrev, A. Israr and C. Harrison: “TeslaTouch- Electrovibration for Touch Surfaces”, *The 23rd Annual ACM Symposium on User Interface Software and Technology (UIST 2010)*, New York, NY, 2010, pp. 283-292.

6. V. Krunic and R. Han: “Towards Physically Rendered Environments”, in 2007, pp.1-12, *University of Colorado at Boulder Technical Report CU-CS-1033-07*, available at http://www.cs.colorado.edu/department/publications/reports/docs/CU-CS-1033-07.pdf, visited on October 1, 2007.

7. V. Krunic and R. Han: “Towards Cyber-Physical Holodeck Systems Via Physically Rendered Environments (PRE's)”, *Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems Workshops*, 00, 2008, pp. 507-512.

8. E. Enikov, K. Lazarov and G. Gonzales: “Microelectrical Mechanical Systems Actuator Array for Tactile Communication”, *COMPUTERS HELPING PEOPLE WITH SPECIAL NEEDS Lecture Notes in Computer Science*, 2398/2002, 2002, pp. 245-259.

9. Doll JC, Harjee N, Klejwa N, Kwon R, Coulthard SM, Petzold B, Goodman MB and Pruitt BL: “Creative Commons licensed image from article "SU-8 force sensing pillar arrays for biological measurements." in *Lab Chip. 2009 May 21;9(10):1449-54*, also available on Flickr URL http://www.flickr.com/photos/jcdoll/3833701083/sizes/l/in/photostream/, visited on August 16, 2010.

10. H. Iwata, H. Yano, H. Fukushima and H. Noma: “CirculaFloor [locomotion interface]”, *Computer Graphics and Applications, IEEE*, 25, 2005, pp. 64-67

11. J. Hollerbach, Y. Xu, R. Christensen and S. Jacobsen: "Design Specifications for the Second Generation Sarcos Treadport Locomotion Interface", *Haptics Symposium, Proc. ASME Dynamic Systems and Control Division*, Orlando, Florida, 2000, pp. 1293-1298.

12. R. Darken, W. Cockayne and D. Carmein: "The Omni-Directional Treadmill: a Locomotion Device for Virtual Worlds", *Proceedings of the 10th annual ACM symposium on User interface software and technology*, Banff, Alberta, Canada, 1997, pp. 213-221.

13. Flickr user Mr eNil: "Image from Beijing Olympics Opening Ceremony (Creative Commons licensed image)", in Beijing, China, 2008, available at http://www.flickr.com/photos/enil/2800434486/sizes/l/in/photostream/, visited on September 18, 2010.

14. Dino Direct: "Pin Art/Pin Point Toy", internet shopping site at http://www.dinodirect.com/Pin-Point-Impression-Toy.html, visited on September 27, 2010.

15. J. Gargus, B. Kim, J. Rossignac, I. Llamas and J. R. a. Shaw: "Finger sculpting with Digital Clay: 3D Shape Input and Output Through a Computer-Controlled Real Surface (Technical Report)", *Technical Report GIT-GVU-02-22 GVU Center and College of Computing Georgia Institute of Technology*, 2010, available at http://smartech.gatech.edu/bitstream/1853/3285/1/02-22.pdf, visited on September 5, 2010.

16. H. Zhu and W. Book: "Control Concepts for Digital Clay", *Robot control 2003 (SYROCO'03): a proceedings volume from the 7th IFAC Symposium*, 2, Wrocław, Poland, 2003, pp. 347-352.

17. H. Zhu and W. Book: "Embedding and Multiplexing Large Scale Sensor Arrays for Digital Clay", *Proceedings of ASME-IMECE ASME International Mechanical Engineering Congress and Exposition (IMECE2005)*, Orlando Florida, 2005, pp. 1489-1495.

18. V. Chouvardas, A. Miliou and M. Hatalis: "Tactile displays: Overview and recent advances", *Displays*, 29, 2008, pp. 185-194.

19. P. Dario: "Tactile sensing: Technology and applications.", *Sensors and Actuators*, 1991, pp. 251-256.

20. L. Johnson and C. Higgins: "A Navigation Aid for the Blind Using Tactile-Visual Sensory Substitution", *Proceedings of the 28th IEEE EMBS Annual International Conference*, New York City, USA, 2006, pp. 6289-6292.

21. B. Britton, A. Moklestad, M. Rodriguez, C. Rogers and S. R. Sokol: “BrailleEye - Haptic User Interface for the Visually Impaired”, poster presented at *The Ninth Annual Coleman Institute conference, Cognitive Disability, Inequality, and Technology in an Age of Economic Uncertainty*, Westminster, Colorado, 2009, pp. 1.

22. C. Li, E. Thostenson and T. Chou: “Sensors and Actuators Based on Carbon Nanotubes and Their Composites: A Review”, *Composites Science and Technology*, 68, 2008, pp. 1227-1249.

23. I. Poupyrev, T. Nashida and M. Okabe: “Actuation and tangible user interfaces: the Vaucanson duck, robots, and shape displays”, *Proceedings of the 1st international conference on Tangible and embedded interaction (TEI'07)*, Baton Rouge, LA, USA, 2007, pp. 205 - 212.

24. F. Vidal-Verdu and M. Hafez: “Graphical Tactile Displays for Visually-Impaired People”, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 15, 2007, pp. 119-130.

25. R. Chellali, L. Brayda, C. Martinoli and E. Fontaine: “How Taxel-based Displaying Devices Can Help Visually Impaired People to Navigate Safely”, *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on*, 2009, pp. 470-475.

26. X. Wu: "A reconfigurable tactile display based on polymer MEMS technology", PhD, Georgia Tech, 2008, available at https://smartech.gatech.edu/handle/1853/22623, visited on September 2, 2010.

27. S. Wall and S. Brewster: “Sensory substitution using tactile pin arrays: Human factors, technology and applications”, *Signal Processing*, 86, 2006, pp. 3674-3695.

28. O. Bimber and R. Raskar: "Spatial Augmented Reality: Merging Real and Virtual Worlds", Prentice Hall, 2005, ISBN-13: 978-1568812304.

29. I. E. Sutherland: “The Ultimate Display”, *Proceedings of the IFIP Congress*, 1965, pp. 506-508.

30. Amazon.com, Inc.: “Amazon Kindle DX Technical Specs”, in 2009, available at http://www.amazon.com/Kindle-Wireless-Reading-Display-Generation/dp/B0015TCML0, visited on November 10, 2009.

31. N. Asamura, T. Shinohara, Y. Tojo, N. Koshida and H. Shinoda: “Necessary Spatial Resolution for Realistic Tactile Feeling Display”, *IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*, 2, Coex, South Korea, 2001, pp. 1851-1856.

32. A. Arnoldussen, B. Besta, A. Ferber and E. Fisher: “Spatial Resolution on the Tongue as Applied to a Prosthetic BrainPort (R) Vision Device”, *Investigative Ophtalmology and Visual Science*, 49, 2008.

33. H. Zhu: "PRACTICAL STRUCTURAL DESIGN AND CONTROL FOR DIGITAL CLAY", PhD Thesis, Georgia Institure of Technology, 2005, available at http://smartech.gatech.edu/handle/1853/7270, visited on September 2, 2010.

34. J. D. Foley, A. v. Dam, S. K. Feiner and J. F. Hughes: "Computer Graphics: Principles and Practice in C (2nd Edition)", Addison-Wesley Professional, 1995, ISBN-13: 978-0201848403.

35. A. Watt and F. Policarpo: "Advanced Game Development with Programmable Graphics Hardware", A K Peters, Ltd, 2005, ISBN-13: 978-1568812403.

36. W. Engel: "ShaderX3: Advanced Rendering with DirectX and OpenGL (Shaderx Series)", Charles River Media, 2004, ISBN-13: 978-1584503576.

37. R. J. Rost: "OpenGL(R) Shading Language (2nd Edition)", Addison-Wesley Professional, 2006, ISBN-13: 978-0321334893.

38. J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn and T. Purcell: “A Survey of General-Purpose Computation on Graphics Hardware”, *Computer Graphics Forum*, 26, 2007, pp. 80-113.

39. A. Brodtkorb, C. Dyken, T. Hagen, J. Hjelmervik and O. Storaasli: “State-of-the-Art in Heterogeneous Computing”, *Scientific Programming*, 18, 2010, pp. 1-33.

40. V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal and P. Dubey: “Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU”, *Proceedings of the 37th annual international symposium on Computer architecture*, 38(3), Saint-Malo, France, 2010, pp. 451-460.

41. E. Catmull: "A Subdivision Algorithm for Computer Display of Curved Surfaces", PhD, University of Utah, 1974.

42. J. Hollerbach, W. Thompson and P. Shirley: “The Convergence of Robotics, Vision, and Computer Graphics for User Interaction”, *International Journal of Robotics Research*, 18, 1999, pp. 1088-1100.

43. D. Tristano, J. Hollerbach and R. Christensen: "Slope Display on a Locomotion Interface", *EXPERIMENTAL ROBOTICS VI - Lecture notes in control and information sciences*, 250/2000, 2000, pp. 193-201.

44. H. Iwata, H. Yano and F. Nakaizumi: "Gait master: A Versatile Locomotion Interface for Uneven Virtual Terrain", *Proceedings of the IEEE VR2001 Conference*, Yokohama, Japan, 2001, pp. 131-137.

45. H. Yano, K. Kasai, H. Saito and H. Iwata: "Sharing Sense of Walking With Locomotion Interfaces", *International Journal of Human-Computer Interaction*, 17, 2004, pp. 447-462.

46. J. Hollerbach, D. Grow and C. Parker: "Developments in Locomotion Interfaces", *Proc. 9th Int'l Conf. on Rehabilitation Robotics*, Chicago, Illinois, 2005, pp. 522–525.

47. J. Hollerbach: "Some Current Issues in Haptics Research", *IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*, San Francisco, California, 2000, pp. 757-762.

48. J. Yoon and J. Ryu: "A novel locomotion interface with two 6-dof parallel manipulators that allows human walking on various virtual terrains", *The International Journal of Robotics Research*, 25(7), 2006, pp. 689-708.

49. Intel Corporation: "Dynamic Physical Rendering", in 2010, available at http://techresearch.intel.com/ProjectDetails.aspx?Id=120, visited on September 28, 2010.

50. Carnegie Mellon: "Claytronics Project", in 2010, available at http://www.cs.cmu.edu/~claytronics/, visited on September 28, 2010.

51. P. Pillai and J. Campbell: "Sensing and Reproducing the Shapes of 3D Objects Using Claytronics", *Proceedings of the 4th international conference on Embedded networked sensor systems (DEMONSTRATION SESSION: Demonstration papers)*, 2006, pp. 369-370.

52. B. T. Kirby, B. Aksak, J. D. Campbell, J. F. Hoburg, T. C. Mowry, P. Pillai and S. C. Goldstein: "A modular robotic system using magnetic force effectors", *In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'07)*, San Diego, CA, 2007, pp. 2787-2793.

53. M. Karagozler, S. Goldstein and J. Reid: "Stress-driven MEMS Assembly+ Electrostatic Forces= 1mm Diameter Robot", *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009.*, St. Louis, Missouri, 2009, pp. 2763-2769.

54. S. Goldstein, J. Campbell and T. Mowry: "Programmable Matter", *Computer*, 38, 2005, pp. 99-101.

55. P. Bhat, J. Kuffner, S. Goldstein and e. al.: "Hierarchical Motion Planning for Self-Reconfigurable Modular Robots", *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006*, Beijing, China, 2006, pp. 886-891.

56. D. Christensen, J. Campbell and K. Stoy: "Anatomy-Based Organization of Morphology and Control in Self-Reconfigurable Modular Robots", *Neural Computing & Applications*, 19, 2010, pp. 787-805.

57. Mark Yim,Wei-Min Shen,Behnam Salemi, D. Rus, M. Moll,Hod Lipson,Eric Klavins and G. S. Chirikjian: "Modular Self-Reconfigurable Robot Systems", *IEEE Robotics and Automation Magazine*, 1, 2007, pp. 43-52.

58. R. Fitch and Z. Butler: "Million Module March: Scalable Locomotion for Large Self-Reconfiguring Robots", *The International Journal of Robotics Research*, 27, 2008, pp. 331-343.

59. Z. Wang, K. Bauernfeind and T. Sugar: "Omni-Directional Treadmill System", *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings.*, Los Angeles, California, 2003, pp. 367-373.

60. H. Iwata: "Walking About Virtual Environments on an Infinite Floor", *Proceedings of the IEEE Virtual Reality*, Houston, Texas, 1999, pp. 286-293.

61. J. Hollerbach, R. Mills, D. Tristano, R. Christensen, W. Thompson and Y. Xu: "Torso Force Feedback Realistically Simulates Slope on Treadmill-Style Locomotion Interfaces", *The International Journal of Robotics Research*, 20, 2001, pp. 939.

62. J. Hollerbach, D. Checcacci, H. Noma, Y. Yanagida and N. Tetsutani: "Simulating Side Slopes on Locomotion Interfaces Using Torso Forces", *Proceedings of the 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'03)*, Los Angeles, CA, 2003, pp. 91-98.

63. C. Cruz-Neira, D. J. Sandin and T. A. DeFanti: "Surround-Screen Projection-Based Virtual Reality: the Design and Implementation of the CAVE", *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, Anaheim, CA, 1993, pp. 135-142.

64. Wikimedia Commons Image: "Image of the soldier in the CAVE environment (public domain as work from United States Federal Government)", in 2007,

available at http://en.wikipedia.org/wiki/File:ARL_ODT.jpg, visited on November 11th, 2009.

65. American Foundation for the Blind: “Refreshable Braille Display”, in 2010, available at http://www.afb.org/Section.asp?DocumentID=3652&SectionID=7&SubTopicID=97&TopicID=330, visited on September 28, 2010.

66. Wikimedia Commons: “Image of the refreshable Brail Display”, in 2006, available at http://en.wikipedia.org/wiki/File:Refreshable_Braille_display.jpg, visited on September 28, 2010.

67. M. Hafez: “Tactile Interfaces: Technologies, Applications and Challenges”, *The Visual Computer*, 23, 2007, pp. 267-272.

68. Y. Ikei, M. Yamada and S. Fukuda: “A new design of haptic texture display-Texture Display2-and its preliminary evaluation”, *Proceedings of IEEE Virtual Reality 2001 Conference*, Yokohama, Japan, 2001, pp. 21-28.

69. D. Ruspini, K. Kolarov and O. Khatib: “The Haptic Display of Complex Graphical Environments”, *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, Los Angeles, California, 1997, pp. 345-352.

70. J. Roberts and S. Panëels: “Where are we with Haptic Visualization?”, *Proceedings of the Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Tsukuba, Japan, 2007, pp. 316-323.

71. M. Shinohara, Y. Shimizu and A. Mochizuki: “Three-Dimensional Tactile Display for the Blind”, *IEEE Transactions on Rehabilitation Engineering*, 6, 1998, pp. 249-256.

72. E. Chabot and Y. Sun: “Visual-to-Tactile Interface to Detect Motions in Real-time for Persons with Visual Impairments”, *Proceedings of the IEEE 32nd Annual Northeast Bioengineering Conference, 2006.*, Easton, Pennsylvania, 2006, pp. 63-64.

73. M. Pereira and F. Kassab: “An Electrical Stimulator for Sensory Substitution”, *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, August 30 - September 3, 2006, 2006, pp. 6016-6020.

74. R. E. Ladner, M. Y. Ivory, R. Rao, S. Burgstahler, D. Comden, S. Hahn, M. Renzelmann, S. Krisnandi, M. Ramasamy, B. Slabosky, A. Martin, A. Lacenski, S. Olsen and D. Groce: “Automating Tactile Graphics Translation”,

Assets '05: Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility, New York, NY, 2005, pp. 50-57.

75. T. P. Way and K. E. Barner: “Automatic Visual to Tactile Translation-Part I: Human Factors, Access Methods, and Image Manipulation”, *IEEE Transactions on Rehabilitation Engineering*, 5, 1997, pp. 81-94.

76. T. P. Way and K. E. Barner: “Automatic Visual to Tactile Translation. II. Evaluation of the TACTile Image Creation System”, *IEEE Transactions on Rehabilitation Engineering*, 5, 1997, pp. 95-105.

77. F. Alarya, M. Duquettea, R. Goldsteina, C. E. Chapmanb, P. Vossa, V. L. Buissonnière-Arizaa and F. Leporea: “Tactile acuity in the blind: A closer look reveals superiority over the sighted in some but not all cutaneous tasks”, *Neuropsychologia*, 47, 2009, pp. 2037-2043.

78. H. Zhu and W. Book: “Practical Structure Design and Control for Digital Clay”, *2004 ASME International Mechanical Engineering Congress and Exhibition*, Anaheim, California, 2004, pp. 1051.

79. J. Rossignac, M. Allen, W. Book, A. Glezer, I. Ebert-Uphoff, C. Shaw, D. Rosen, S. Askins, J. Bai and P. Bosscher: “Finger Sculpting with Digital Clay: 3d Shape Input and Output Through a Computer-Controlled Real Surface”, *International Conference on Shape Modeling and Applications*, May 12-15, 2003, 2003, pp. 229 - 231.

80. H. Iwata, H. Yano, F. Nakaizumi and e. al.: “Project FEELEX: Adding Haptic Surface to Graphics”, *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, Los Angeles, California, 2001, pp. 469-476.

81. D. Overholt, E. Pasztor and A. Mazalek: “A Multipurpose Array of Tactile Rods for Interactive eXpression”, *ACM SIGGRAPH 2001, Sketches & Application Session*, 2001, pp. 232.

82. D. J. Page: “Reconfigurable surface. US Patent, June 2005. US 6903871.”, 2005, pp. 1-11.

83. W. Hillis and F. B. Daniel: “Raised display apparatus. US Patent, October 2008. US 7436388”, 2008, pp. 1-11.

84. D. Leithinger and H. Ishii: “Relief: a Scalable Actuated Shape Display”, *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction (DEMONSTRATION SESSION: Demonstrations)*, Cambridge, Massachusetts, 2010, pp. 221-222.

85. D. Leithinger: "Design and Implementation of a Relief Interface", MS Thesis, Massachusetts Institute of Technology, 2010, available at http://dspace.mit.edu/handle/1721.1/57899, visited on October 16th, 2010.

86. M. Benali-Khoudja, M. Hafez, J. Alexandre and A. Kheddar: "Tactile Interfaces: a State-of-the-art Survey", *ISR 2004, 35th International Symposium on Robotics*, Paris, France, 2004.

87. S. Takagi, H. Sasaki, M. Shikida and K. Sato: "Electrostatic Latch Mechanism for Handling Projection on Arrayed Vertical Motion System", *TRANSDUCERS & EUROSENSORS '07, The 14th International Conference on Solid-State Sensors, Actuators and Microsystems*, Lyon, France, 2007, pp. 1147-1150.

88. Apple, Inc.: "iPhone", in 2009, available at http://www.apple.com/iphone/, visited on September 3, 2010.

89. C. Harrison and S. Hudson: "Dynamic Displays", *ACM Crossroads*, 16, 2009, pp. 6-7.

90. R. H. Baughman, C. Cui, A. A. Zakhidov, Z. Iqbal, J. N. Barisci, G. M. Spinks, G. G. Wallace, A. Mazzoldi, D. D. Rossi, A. G. Rinzler, O. Jaschinski, S. Roth and M. Kertesz: "Carbon Nanotube Actuators", *Science*, 284, 1999, pp. 1340-1344.

91. C. Ziegler: "iPhone 3G vs. iPhone 3G S: the tale of the tape", in engadget.com, 2009, available at http://mobile.engadget.com/2009/06/08/iphone-3g-vs-iphone-3g-s-the-tale-of-the-tape/, visited on March 23, 2010.

92. C. Ziegler: "iPhone 3G S supports OpenGL ES 2.0, but 3G only supports 1.1 -- will the App Store splinter?", in 2009, available at http://www.engadget.com/2009/06/10/iphone-3g-s-supports-opengl-es-2-0-but-3g-only-supports-1-1/, visited on August 3, 2010.

93. H. Nguyen: "iPhone 3GS: Prepare For a 3D Graphics Shock", in ubergizmo.com, 2009, available at http://www.ubergizmo.com/15/archives/2009/06/iphone-3gs-gpu.html, visited on February 11, 2010.

94. G. E. Favalora: "Volumetric 3D Displays and Application Infrastructure", *Computer*, 38, 2005, pp. 37 - 44.

95. W. Chun, J. Napoli, O. S. Cossairt, R. K. Dorval, D. M. Hall, T. J. P. II, J. F. Schooler, Y. Banker and G. E. Favalora: "Spatial 3-D Infrastructure: Display-Independent Software Framework, High-Speed Rendering Electronics, and Several New Displays", *Stereoscopic Displays and Virtual Reality Systems*

XII, edited by Andrew J. Woods, Mark T. Bolas, John O. Merritt, Ian E. McDowall, Proceedings of SPIE-IS&T Electronic Imaging, SPIE, 5664, 2005, pp. 302-312.

96. J. Napoli, S. Stutsman, J. Chu, X. Gong, M. Rivard, G. Cardarelli, T. Ryan and G. Favalora: “Radiation Therapy Planning Using a Volumetric 3-D Display: PerspectaRAD”, *Proceedings of SPIE*, 6803, San Diego, California, 2008, pp. 680312.

97. R. Fernando: "GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics", Addison-Wesley Professional, 2004, ISBN-13: 978-0321228321.

98. M. Bunnell and F. Pellacini: "GPU Gems 2 - Programming Techniques for High-Performance Graphics and General-Purpose Computation", Addison-Wesley Professional, 2005, ISBN-13: 978-0321335593.

99. M. Liao, Z. Zhang and J. Lewis: “Software-Based Distortion Compensation for a Scanned Beam Display”, *Proceedings of the 5th ACM/IEEE International Workshop on Projector camera systems*, Marina del Rey, California, 2008, pp. 13.

100. C. Chen, B. Miao and D. Prather: “Enhanced Electro-Optical Effect in Silicon”, *Proc. of SPIE Vol*, 6477 64770Y-1, San Jose, California, 2007, pp. 6477-6432.

101. F. Bourger, C. Doignon and M. D. Mathelin: “A Model-free Vision-based Robot Control for Minimally Invasive Surgery using ESM Tracking and Pixels Color Selection”, *2007 IEEE International Conference on Robotics and Automation*, Roma, Italy, 2007, pp. 3579-3584.

102. P. Novotny, J. Stoll, N. Vasilyev, P. Del Nido, P. Dupont, T. Zickler and R. Howe: “GPU Based Real-Time Instrument Tracking with Three-Dimensional Ultrasound”, *Medical image analysis - Special Issue on the Ninth International Conference on Medical Image Computing and Computer-Assisted Interventions - MICCAI 2006*, 11, 2007, pp. 458-464.

103. P. Novotny, J. Stoll, P. Dupont and R. Howe: “Real-time Visual Servoing of a Robot Using Three-Dimensional Ultrasound”, *IEEE International Conference on Robotics and Automation*, Rome, Italy, 2007, pp. 2655-2660.

104. E. Dedu, K. Boutoustous and J. Bourgeois: “An Exhaustive Comparison Framework for Distributed Shape Differentiation in a MEMS Sensor Actuator Array”, *Proceedings of the 2008 International Symposium on Parallel and Distributed Computing*, Krakow, 2008, pp. 429-433.

105. A. Frezzotti, G. P. Ghiroldi and L. Gibelli: “Solving Kinetic Equations on GPUs I: Model Kinetic Equations”, *Preprint Submitted to Elsevier, 24 March*, 2009.

106. R. Gayle, M. Lin and D. Manocha: “Constraint-based motion planning of deformable robots”, *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005. ICRA 2005., Barcelona, Spain, 2005, pp. 1046-1053.

107. L. Williams: “Casting curved shadows on curved surfaces”, *ACM SIGGRAPH Computer Graphics*, 12(3), 1978, pp. 270-274.

108. M. W. Jones, J. A. Baerentzen and M. Sramek: “3D distance fields: A survey of techniques and applications”, *IEEE Transactions on Visualization and Computer Graphics*, 12, 2006, pp. 581-599.

109. T. Morvan, M. Reimers and E. Samset: “High Performance GPU-Based Proximity Queries Using Distance Fields”, *Computer Graphics Forum*, 27, 2008, pp. 2040-2052.

110. A. Sud, M. Otaduy and D. Manocha: “DiFi: Fast 3D distance field computation using graphics hardware”, *EUROGRAPHICS*, 23, 2004, pp. 557-566.

111. K. Hoff, T. Culver, J. Keyser, M. Lin and D. Manocha: “Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware”, *Proceedings of ACM SIGGRAPH*, 1999, pp. 277-286.

112. O. Spinczyk, A. Gal and W. Schoder-Preikschat: “Aspect C++: An Aspect-Oriented Extension to the C++ Programming Language”, *Proceedings of the Fortieth International Conference on Tools Pacific: Objects for internet, mobile and embedded applications*, 21, Sydney, Australia, 2002, pp. 53-60.

113. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J. Loingtier and J. Irwin: “Aspect Oriented Programming”, *European Conference on Object-Oriented Programming*, 1241, 1997, pp. 220-242.

114. W. Mokwa: “MEMS Technologies for Epiretinal Stimulation of the Retina”, *Journal of Micromechanics and Microengineering*, 14, 2004, pp. 12.

115. G. Lamb: “Tactile Discrimination of Textured Surfaces: Psychophysical Performance Measurements in Humans.”, *The Journal of Physiology*, 338, 1983, pp. 551-565.

116. J. Morley, A. Goodwin and I. Darian-Smith: “Tactile Discrimination of Gratings”, *Experimental brain research*, 49, 1983, pp. 291-299.

117. Y. Tojo, N. Asamura and H. Shinoda: "A study on tactile resolution of human skin", *SICE 2002. Proceedings of the 41st SICE Annual Conference*, 5, 2002, pp. 3205-3207.

118. L. Elliott, J. van Erp, E. Redden and M. Duistermaat: "Field-Based Validation of a Tactile Navigation Device", *IEEE Transactions on Haptics*, 3, 2010, pp. 78-87.

119. J. B. F. v. Erp, L. Eriksson, B. Levin, O. Carlander, J. A. Veltman and W. K. Vos: "Tactile Cueing Effects on Performance in Simulated Aerial Combat with High Acceleration", *Aviation, Space and Environmental Medicine*, 78, 2007, pp. 1128-1134.

120. J. B. F. v. Erp, E. L. Groen, J. E. Bos and H. A. H. C. v. Veen: "A Tactile Cockpit Instrument Supports the Control of Self-Motion During Spatial Disorientation", *Human factors*, 48, 2006, pp. 219-228.

121. J. M. Rolfe and K. J. Staples: "Flight Simulation (Cambridge Aerospace Series)", Cambridge University Press, 1988, ISBN-13: 978-0521357517.

122. T. Jung and I. Haulsen: "Immersive Escape-Route Scenario with Locomotion Devices", *Proceedings of Workshop on Spatial Cognition in Real and Virtual Environments*.

123. Nintendo Corporation: "Nintendo Wii", 2009, available at http://www.nintendo.com/wii, visited on November 10th, 2009.

124. O. A. R. Board, D. Shreiner, M. Woo, J. Neider and T. Davis: "OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R), Version 2 (5th Edition)", Addison-Wesley Professional, 2005, ISBN-13: 978-0321335739.

125. K Christian: "Design of Haptic and Tactile Interfaces for Blind Users", in 2000, available at http://www.otal.umd.edu/UUGuide/kevin/, visited on November 5, 2009.

126. Digital Display Working Group: "Digital Visual Interface (DVI)", Revision 1.0, 1999, available at http://www.ddwg.org/lib/dvi_10.pdf, visited on September 6, 2010.

127. S. Brewster and L. Brown: "Tactons: structured tactile messages for non-visual information display", *Proceedings of Australasian User Interface Conference*, Dunedin, New Zealand, 2004, pp. 15-23.

128. L. Brown, S. Brewster and H. Purchase: “Multidimensional tactons for non-visual information presentation in mobile devices”, *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, 2006, pp. 238.

129. K. R. Castelman: "Digital Image Processing", Prentice Hall, 1995, ISBN 978-0132114677.

130. A. S. Glassner: "Principles of Digital Image Synthesis", Morgan Kaufmann, 1995, 978-1558602762.

131. CppUnit Project: “CppUnit Wiki”, in 2008, available at http://sourceforge.net/apps/mediawiki/cppunit/index.php?title=Main_Page, visited on May 2nd, 2007.

132. K. Back: "Test Driven Development: By Example", Addison-Wesley Professional, 2002, ISBN-13: 978-0321146533.

133. Khronos Group: “COLLADA – Digital Asset Schema Release 1.5.0 Specification”, in 2008, available at http://www.khronos.org/files/collada_spec_1_5.pdf, visited on December 30, 2009.

134. Dimitri van Heesch: “Doxygen: Doxygen: Generate Documentation From Source Code”, in 2010, available at http://www.stack.nl/~dimitri/doxygen/, visited on September 20, 2010.

135. G. E. Krasner and S. T. Pope: “A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System”, *Journal of Object Oriented Programming*, 1, 1988, pp. 26-49.

136. Flickr user Joel Johnson: “Creative Commons licensed image of VR helmet”, in 2007, available at http://www.flickr.com/photos/joeljohnson/353246613/sizes/m/in/photostream/, visited on August 16, 2010.

137. Khronos Group: “OpenGL ES 2.X and the OpenGL ES Shading Language”, in 2010, available at http://www.khronos.org/opengles/2_X/, visited on September 2, 2010.

138. NVIDIA: “GeForce 7300 GPUs”, in 2009, available at http://www.nvidia.com/page/geforce_7300.html, visited on November 10, 2009.

139. J. L. Hennessy and D. A. Patterson: "Computer Architecture: A Quantitative Approach, 4th Edition", Morgan Kaufmann, 2006, ISBN-13: 978-0123704900.

140. J. E. Stone, D. Gohara and G. Shi: “OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems”, *Computing in Science and Engineering*, 12, 2009, pp. 66-73.

141. NVIDIA: “NVIDIA CUDA Programming Guide”, in NVIDIA Corporation, 2009, pp.1-145, available at http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.3.pdf, visited on March 5, 2010.

142. M. Sadeghi, H. Kim and K. Najafi: “Electrostatically Driven Micro-Hydraulic Actuator Arrays”, *IEEE 23rd International Conference on Micro Electro Mechanical Systems (MEMS)*, Wanchai, Hong Kong, 2010, pp. 15-18.

143. M. Strengert, M. Kraus and T. Ertl: “Pyramid Methods in GPU-based Image Processing”, *Vision, modeling, and visualization 2006: Proceedings*, Aachen, Germany, 2006, pp. 169-176.

144. J. T. Luftig and M. V. Petrovich: "Quality with Confidence in Manufacturing", Luftig & Warren International/SPSS Inc, Chicago, IL, 1997, ISBN 1-56827-153-0.

145. J. Fung: “Implementing Efficient Parallel Data Structures on GPU (Sparse Data Structures subchapter)”, in *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, 2005, pp. 535-540.

146. Blender Team Project: “Blender Material Library”, in 2010, available at http://www.blender.org/download/resources/, visited on November 10, 2009.

147. Omniglot - writing systems & languages of the world: “Braille”, in 2010, available at http://www.omniglot.com/writing/braille.htm, visited on September 2008, 2010.

148. G. Lamb: “Tactile Discrimination of Textured Surfaces: Peripheral Neural Coding in the Monkey.”, *The Journal of Physiology*, 338, 1983, pp. 567-587.

149. R. LaMotte and J. Whitehouse: “Tactile Detection of a Dot on a Smooth Surface: Peripheral Neural Events”, *Journal of Neurophysiology*, 56, 1986, pp. 1109-1128.

150. A. Chang and C. O’Sullivan: “Describing Haptic Phenomena”, in 2005, available at http://www.dcs.gla.ac.uk/haptic/haptics%20web%20pages_files/Chang%20and%20Sullivan.pdf, visited on September 28, 2010.

151. Wikimedia Commons: “Image of Michelangelo David”, in 2007, available at http://en.wikipedia.org/wiki/File:Michelangelo%27s_David_-_Floyd-Steinberg.png, visited on September 27, 2010.

152. Google, Inc.: “Google SketchUp Home Page”, in 2010, available at http://sketchup.google.com/, visited on January 5, 2010.

153. Personal communication with Dr. Hiroo Iwata, October 25, 2010.

154. Tokyo National Museum of Emerging Science and Innovation, Japan: “Laboratory for New Media 3rd Exhibition, "Hiroo Iwata: Dr. Strange Device"”, in 2010, available at http://www.miraikan.jst.go.jp/en/info/090108102774.html, visited on October 25, 2010.

155. Kevin Hall: “CirculaFloor simulates 360° of movement in VR — and now goes up, too”, in 2009, available at http://dvice.com/archives/2009/02/circulafloor_si.php, visited on September 12, 2010.

156. B. He, W. Fang, Q. Luo and N. K. Govindaraju: “Mars: a MapReduce Framework on Graphics Processors”, *Parallel Architectures and Compilation Techniques (PACT) PACT’08*, Toronto, Ontario, Canada, 2008, pp. 260-269.

157. nViDIA Corporation: “NVIDIA CUDA Compute Unified Device Architecture Programming Guide Version 1.0”, in 2007, pp.1-112, available at http://developer.download.nvidia.com/compute/cuda/1_0/NVIDIA_CUDA_Programming_Guide_1.0.pdf, visited on April 9, 2010.

158. H. Nguyen: “Using Geometry Shader for Compact and Variable-Length GPU Feedback”, in *Gpu Gems 3*, Addison-Wesley Professional, 2007, pp.891-907, ISBN:9780321545428.

159. J. Fung: “Chapter 40. Computer Vision on the GPU”, in GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation, Addison-Wesley Professional, 2005, pp.649-666, ISBN-13: 978-0321335593.

160. M. Bunnell and F. Pellacini: “Chapter 11 - Shadow Map Anti-Aliasing”, in *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, Addison-Wesley, 2004, pp.185-192, ISBN-13: 978-0321228321.

161. Kurt Akeley, Jason Allen, Bob Beretta, Pat Brown, Matt Craighead, Alex Eddy, Cass Everitt, Mark Galvan, Michael Gold, Evan Hart, Jeff Juliano, Mark Kilgard, Dale Kirkland, Jon Leech, Bill Licea-Kane, Barthold Lichtenbelt, Kent Lin, Rob Mace, Teri Morrison,Chris Niederauer, Brian

Paul, Paul Puey, Ian Romanick, John Rosasco, R. Jason Sams, Jeremy Sandmel, Mark Segal, Avinash Seetharamaiah, Folker Schamel, Daniel Vogel, Eric Werness and Cliff Woolley: “OpenGL Extension Registry for EXT_framebuffer_object”, in 2008, available at http://www.opengl.org/registry/specs/EXT/framebuffer_object.txt, visited on January 4, 2010.

162. D. Wigdor, C. Forlines, P. Baudisch, J. Barnwell and C. Shen: “Lucid Touch: a See-Through Mobile Device”, *Proceedings of the 20th annual ACM symposium on User interface software and technology*, Newport, Rhode Island, USA, 2007, pp. 269-278.

163. C. Ericson: "Real-Time Collision Detection", Morgan Kaufmann, 2005, ISBN-13: 978-1558607323.

164. D. Rosen, A. Nguyen and H. Wang: “On the Geometry of Low Degree-of-Freedom Digital Clay Human-Computer Interface Devices”, *Proceedings ASME Computers and Information in Engineering Conference*, Chicago, Illinois, 2003, pp. 2-6.

165. J. Lee, C. Wagner, S. Lederman and R. Howe: “Spatial Low Pass Filters for Pin Actuated Tactile Displays”, Proceedings of 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003., Los Angeles, California, 2003, pp. 57-62.

166. W. E. Lorensen and H. E. Cline: “Marching cubes: A High Resolution 3D Surface Construction Algorithm”, *Computer Graphics*, 21, 1987, pp. 163-169.

167. Wolfram Research: “Circle-Line Intersection”, in 2010, available at http://mathworld.wolfram.com/Circle-LineIntersection.html, visited on September 24, 2010.

168. Apple, Inc.: “Grand Central Dispatch (GCD) Reference”, in 2010, available at http://developer.apple.com/library/mac/#documentation/Performance/Reference/GCD_libdispatch_Ref/Reference/reference.html, visited on October 28, 2010.

169. Intel Corporation: “10 Gigabit Ethernet Technology Overview”, in 2003, available at http://www.intel.com/network/connectivity/resources/doc_library/white_papers/pro10gbe_lr_sa_wp.pdf, visited on September 3, 2010.

170. IEEE Computer Society: “IEEE Standard for Local and metropolitan area networks — Link Aggregation (IEEE Std 802.1AXTM-2008)”, in 2010, available at http://standards.ieee.org/getieee802/download/802.1AX-2008.pdf, visited on September 23, 2010.

171. Khronos Group: “OpenCL 1.1 Specification”, in 2010, available at http://www.khronos.org/registry/cl/specs/opencl-1.1.pdf, visited on October 14, 2010.

172. Melchor J. Antunano: “Federal Aviation Administration Pilot Safety Brochure - Spatial Disorientation”, in 2003, available at http://www.faa.gov/pilots/safety/pilotsafetybrochures/media/SpatialD.pdf, visited on October 24, 2010.

173. J. M. Rolfe and K. J. Staples: “6. Motion Systems”, in *Flight Simulation (Cambridge Aerospace Series)*, 1988, pp.111-129, ISBN-13: 978-0521357517.

174. K. Mueller, F. Xu and N. Neophytou: “Why Do Commodity Graphics Hardware Boards (GPUs) Work so Well for Acceleration of Computed Tomography?”, *SPIE Electronic Imaging*, 6498, 2007, pp. 64980N.

175. C. L. Liu and J. W. Layland: “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment”, *Journal of the ACM (JACM)*, 20, 1973, pp. 46-61.

176. N. C. Audsley, A. Burns and M. F. Richardson: “Hard Real-Time Scheduling: The Deadline Monotonic Approach”, *Proceedings 8th IEEE Workshop on Real-Time Operating Systems and Software*, 1991, pp. 133-137.

177. G. A. Elliott and J. H. Anderson: “Real-Time Multiprocessor Systems with GPUs”, to appear in Proceedings of the 18th International Conference on Real-Time and Network Systems, 2010.

178. R. Dimitrov: “Cascaded Shadow Maps”, in 2007, pp.1-16, available at http://developer.download.nvidia.com/SDK/10.5/opengl/src/cascaded_shadow_maps/doc/cascaded_shadow_maps.pdf, visited on March 7, 2010.

179. P. Longhurst, K. Debattista and A. Chalmers: “A gpu based saliency map for high-fidelity selective rendering”, *Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, 2006, pp. 21-29.

180. D. Calder: “Travel Aids For The Blind - The Digital Ecosystem Solution”, *7th IEEE International Conference on Industrial Informatics, 2009. INDIN 2009.*, Cardiff UK, 2009, pp. 149-154.

181. BlackBerry, Inc.: “BlackBerry Torch 9800 Specifications”, in 2010, available at http://na.blackberry.com/devices/blackberrytorch/#!phone-specifications, visited on September 1, 2010.

182. Apple, Inc.: “iPhone Features - Keyboard”, in 2010, available at http://www.apple.com/iphone/features/keyboard.html, visited on September 19, 2010.

183. Motorola, Inc.: “DroidX by Motorola”, in 2010, available at http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/Mobile-Phones/Motorola-DROID-X-US-EN, visited on September 20, 2010.

184. M. Sugimoto and K. Hiroki: “HybridTouch: an intuitive manipulation technique for PDAs using their front and rear surfaces”, *Proceedings of the 8th conference on Human-Computer Interaction with Mobile Devices and Services*, Helsinki, Finland, 2006, pp. 137-140.

185. S. Hiraoka, I. Miyamoto and K. Tomimatsu: “Behind Touch: A Text Input Method for Mobile Phone by The Back and Tactile Sense Interface.”, *Information Processing Society of Japan, Interaction 2003*, 2003, pp. 131-138.

186. D. Wigdor, D. Leigh, C. Forlines, S. Shipman and e. al.: “Under the Table Interaction”, *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, Montreux, Switzerland, 2006, pp. 259-268.

187. D. Holman and R. Vertegaal: “Organic User Interfaces: Designing Computers in Any Way, Shape or Form”, *COMMUNICATIONS OF THE ACM*, 51, 2008, pp. 48-55.

188. Apple, Inc: “Chapter 4 - Drawing Offscreen”, in *OpenGL Programming Guide for Mac OS X*, 2010, pp.47-57, available at http://developer.apple.com/library/mac/#documentation/GraphicsImaging/Conceptual/OpenGL-MacProgGuide/OpenGLProg_MacOSX.pdf, visited on February 16, 2010.

189. Layar.com: “Layar's Technical Overview”, in 2009, available at http://layar.com/api/, visited on July 16, 2009.

190. M. Fukumoto and T. Sugimura: “Active Click: Tactile Feedback for Touch Panels”, *CHI '01 extended abstracts on Human factors in computing systems, Poster Session*, 2001, pp. 121-122.

191. I. Poupyrev, S. Maruyama and J. Rekimoto: “Ambient Touch: Designing Tactile Interfaces for Handheld Devices”, *Proceedings of the 15th annual*

ACM symposium on User interface software and technology, Paris, France, 2002, pp. 51-60.

192. E. Hoggan and S. Brewster: “Crossmodal Spatial Location: Initial Experiments”, Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles, Oslo, Norway, 2006, pp. 469-472.

193. M. Kurze: “Rendering Drawings for Interactive Haptic Perception”, Proceedings of the SIGCHI conference on Human factors in computing systems, Atlanta, Georgia, 1997, pp. 423-430.

194. M. Hickins: “Smartphones Get Killer App”, in bnet.com, 2009, available at http://industry.bnet.com/technology/10002375/smartphone-gets-killer-app/, visited on May 6, 2010.

195. wikitude.org and Mobilizy Software: “Wikitude World Browser”, in Mobilize Software, 2010, available at http://www.wikitude.org/, visited on September 28, 2010.

196. A. Jaimes and N. Sebe: “Multimodal Human–Computer Interaction: A Survey”, *Computer Vision and Image Understanding*, 108, 2007, pp. 116-134.

197. M. Song, S. Park and Y. B. Kang: “A Survey on Projector-based PC Cluster Distributed Large Screen Displays and Shader Technologies”, *ENTERTAINMENT COMPUTING – ICEC 2007, Lecture Notes in Computer Science*, 4740/2007, 2007, pp. 445-449.

198. Sony Computer Entertainment: “SCEA Shared Source License 1.0”, in 2010, available at http://research.scea.com/scea_shared_source_license.html, visited on September 19, 2010.

199. Boost C++ Libraries: “Boost Software License”, in 2003, available at http://www.boost.org/users/license.html, visited on September 20, 2010.

200. Jean-loup Gailly and Mark Adler: “zlib License”, in 2010, available at http://www.zlib.net/zlib_license.html, visited on September 21, 2010.

201. Apache Software Foundation: “Apache License, Version 2.0”, in 2004, available at http://www.apache.org/licenses/LICENSE-2.0.html, visited on September 28, 2010.

202. Sen:te: “Sen:te Free Software License v4”, in 2004, available at http://www.sente.ch/software/OpenSourceLicense.html, visited on September 7, 2010.

Appendix A - Licenses For the Software

In the development of HoloSim, we used multiple open source packages to speed up development of the simulator. All packages used have licenses allowing for the given use, and additionally none of those packages already performed functionality that is a core area of this research - they were used for peripheral tasks like file loading.

Licenses for the used packages are as follows.

A.1 Apple Examples

Apple examples were used for the initialization of OpenGL on OS X and its interface with the Cocoa toolkit and window system, as well as for unit testing of the shader initialization system. Apple examples FBOBunnies and GLSLBasicCocoaDL were used as a starting point for the framebuffer object initialization and as a template of how to use aforementioned Apple's technologies [99].

All of the previous files and examples are used under the following license:

"Disclaimer: IMPORTANT: This Apple software is supplied to you by Apple Computer, Inc. ("Apple") in consideration of your agreement to the following terms, and your use, installation, modification or redistribution of this Apple software constitutes acceptance of these terms. If you do not agree with these terms, please do not use, install, modify or redistribute this Apple software.

In consideration of your agreement to abide by the following terms, and subject to these terms, Apple grants you a personal, non-exclusive license, under Apple's copyrights in this original Apple software (the "Apple Software"), to use, reproduce, modify and redistribute the Apple Software, with or without modifications, in source and/or binary forms; provided that if you redistribute the Apple Software in its entirety and without modifications, you must retain this

[99] Furthermore, code performing initialization and usage of the Frame Buffer Object was written based on the outline of the process provided in [188],

A.2 COLLADA DOM

ColladaDOM was used for reading files of the Collada format, as written by the Google Sketchup! and other modeling tools. Modifications of the code were limited to the various modifications needed to compile on my machine and are available upon request.

Full text of license, from [198] is included below:

3. No Right to File for Patent.

In exchange for the rights that are granted to You free of charge under this License, You agree that You will not file for any patent application, seek copyright protection or take any other action that might otherwise impair the ownership rights in and to the Software that may belong to SCEA or any of the other contributors/authors of the Software.

4. Contributions.

*SCEA welcomes contributions in form of modifications, optimizations, tools or documentation designed to improve or expand the performance and scope of the Software (collectively "**Contributions**"). Per the terms of this License You are free to modify the Software and those modifications would belong to You. You may however wish to donate Your Contributions to SCEA for consideration for inclusion into the Software. For the avoidance of doubt, if You elect to send Your Contributions to SCEA, You are doing so voluntarily and are giving the Contributions to SCEA and its parent company Sony Computer Entertainment, Inc., free of charge, to use, modify or distribute in any form or in any manner. SCEA acknowledges that if You make a donation of Your Contributions to SCEA, such Contributions shall not exclusively belong to SCEA or its parent company and such donation shall not be to Your exclusion. SCEA, in its sole discretion, shall determine whether or not to include Your donated Contributions into the Software, in whole, in part, or as modified by SCEA. Should SCEA elect to include any such Contributions into the Software, it shall do so at its own risk and may elect to give credit or special thanks to any such contributors in the attached copyright notice. However, if any of Your contributions are included into the Software, they will become part of the Software and will be distributed under the terms and conditions of this License. Further, if Your donated Contributions are integrated into the Software then Sony Computer Entertainment, Inc. shall become the copyright owner of the Software now containing Your contributions and SCEA would be the Licensor.*

5. Redistribution in Source Form

You may redistribute copies of the Software, modifications or derivatives thereof in Source Code Form, provided that You:

a. Include a copy of this License and any copyright notices with source

b. Identify modifications if any were made to the Software

c. Include a copy of all documentation accompanying the Software and modifications made by You

6. Redistribution in Object Form

Licensed under the SCEA Shared Source License, Version 1.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at:

http://research.scea.com/scea_shared_source_license.html

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License."

A.3 Boost C++ Libraries

Boost is used as a file I/O layer by the Collada Dom and other libraries included in this project. Full text of the license is included below [199]:

"Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE."

A.4 LibZ

Zlib is used as a compression library by the other open source libraries used in the system, as well as for file I/O. Full text of its license from [200] is included below:

A.5 Apache Xerces-C++ XML Parser

Xerces-C++ was used for XML document parsing and by other open source libraries. Full text of its license is included below [201]:

“Apache LicenseVersion 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an

A.6 CPPUnit

CPPUnit [131] is used for unit and automated functional testing of the C++ code as a part of our project's build system. Full text of its license from CppUnit 1.12.1 distribution is included below:

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free

Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

A.7 OCUnit

OCUnit was used for automated unit testing of the Objective-C code. Full text of its license is included below [202]:

A.8 Doxygen

Doxygen was used for automated HTML documentation generation from the source code comments in Objective-C, Objective-C++ and C++. Full text of its license is included below[100] [134]:

> *"Copyright © 1997-2009 by Dimitri van Heesch.*
>
> *Permission to use, copy, modify, and distribute this software and its documentation under the terms of the GNU General Public License is hereby granted. No representations are made about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty. See the GNU General Public License for more details.*
>
> *Documents produced by doxygen are derivative works derived from the input used in their production; they are not affected by this license."*

[100] Full text of GNU General Public License referenced in the text of Doxygen's license can be found at http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

A.9 ColladaDOM Use Example

ColladaDOM use example was used as the template for code that is loading COLLADA model code. It is licensed under MIT license. Full text of its license is included below:

Appendix B - License For This Thesis

The author believes that this work should be made available free of charge to anyone who can benefit from it, provided that proper attribution of the work is done. As a result, author is making this work available under the Creative Commons Attribution Non-Commercial 3.0 Unported license. Keep in mind that this thesis uses (with the proper attributions) other material in the public domain or under the various versions of Creative Commons license, and that it is your responsibility to comply with the terms of the original licenses, where appropriate. Full text of the Creative Commons Attribution Non-Commercial Unported 3.0 license is available below and at http://creativecommons.org/licenses/by-nc/3.0/legalcode. For the short summary of your obligations under this license, see http://creativecommons.org/licenses/by-nc/3.0/legalcode.

other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

*"**Collection**" means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined above) for the purposes of this License.*

*"**Distribute**" means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.*

*"**Licensor**" means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.*

*"**Original Author**" means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.*

*"**Work**" means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a*

copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and, (iv) consistent with Section 3 (b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

d. *For the avoidance of doubt:*

    i. ***Non-waivable Compulsory License Schemes**. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;*

    ii. ***Waivable Compulsory License Schemes**. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License if Your exercise of such rights is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b) and otherwise waives the right to collect royalties through any statutory or compulsory licensing scheme; and,*

    iii. ***Voluntary License Schemes**. The Licensor reserves the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License that is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(c).*

e. *Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform* *the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.*

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. *EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.*

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this

License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

b. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

f. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

Appendix C - HoloSim License

HoloSim code is available at https://github.com/krunic/HoloSim. It is released as open source software under the terms of the Affero GPL v3. Full text of its license is included below:

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to *the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.*

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial,

by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work