Sculpting Via Physical Proxy Using Magnets

Michael Margel[*]

Abstract

I explore the idea of using magnets and ball bearings as a means of sculpting a virtual model via a physical proxy. By using a motion capture system, we can track the position of each bearing, and use them to build a skeleton, which can then be used to construct a 3D model. I attempt to implement a system using open source software that allows users to manipulate a virtual model by physically manipulating magnets and ball bearings. While conceptually rather simple, there were a large number of difficulties that were encountered, and potential solutions for these issues are explored. Finally, Liu explored the use of cameras and physical clay in sculpting virtual objects [Liu 2004].

1 Introduction

Cameras that can detect and track fine hand gestures, such as the Microsoft Kinect and Leap Motion are become much more affordable, and as a result, common, than similar systems have been historically. At the same time, there is more research being done into how the physical properties of clay can be simulated. It is because of this overlap that I have decided to explore sculpting virtual objects by using a physical proxy that simulates some of the properties of clay: magnets. Similar to clay, magnets can be detached and reattached elsewhere. Magnetic structures can also be built upon the same way that clay structures can be, by simply adding existing structures to the current structure.

In this paper, I explore the idea that a motion capture system can be used to track the magnets and the metal balls that they are attached to in order to develop an interface that can be used to model virtual clay to determine whether magnets can be used in either commercial or personal applications, in order to allow people to create virtual sculptures without necessarily having any significant experience with computers. I present MagDrop, a program based on DragDropTool [Schmidt and Singh 2009], a program that can be used to sculpt models by dragging and dropping different meshes and by removing and relocating parts of the mesh, in a manner similar to clay. MagDrop heavily modifies the interface, and is updated to allow users to manipulate the mesh with minimal input from a mouse or keyboard by using a Vicon motion tracking system to determine how the different pieces of the mesh should be moved by tracking how a user manipulates a physical proxy.

2 Previous Work

There has been very limited work in the field of sculpting via physical proxy. While a significant amount of work has been done on simulating clay, there had been very little work on the idea of using a physical object to represent virtual models. Pihuit et al. explored the idea of using a haptic device as a way of interacting with virtual objects [Pihuit et al. 2008]. Sheng et al. explored the idea of using a sponge as a physical proxy for clay, where deforming the sponge would have a similar effect on the virtual clay [Sheng et al. 2006].

[*]e-mail:mmargel@cs.toronto.edu

3 Equipment

This experiment used a Vicon 6 system with 6 cameras and a set of standard motion capture markers. Physical models were built using a Geomag magnetic construction toy, containing 54 magnets, each approximately 2.5cm long, and 32 metal ball bearings, each approximately 1cm in diameter. The Vicon iQ and MagDrop software were run on a PC running 64-bit Windows 7, with a 3.2GHz dual core Xeon processor and 4GB of RAM. MagDrop is written in C++, is based on the DragDropTool source code, and uses the libgeometry and WildMagic4 libraries.

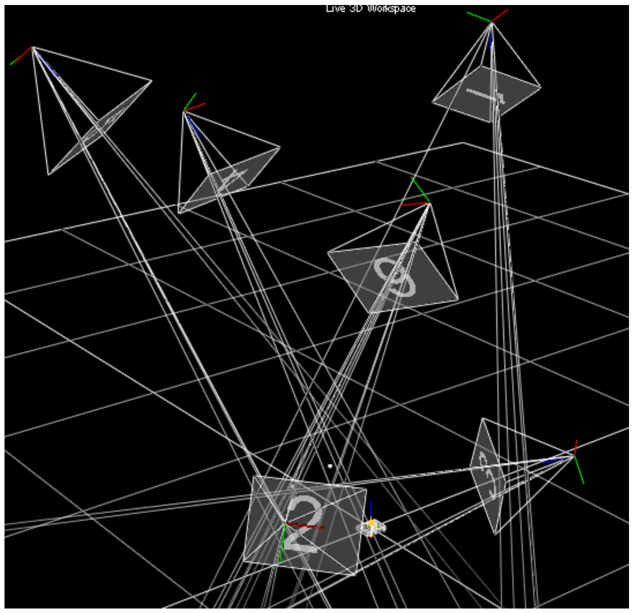

Figure 1: *Arrangement of the Vicon cameras. Cameras 1, 4, and 5 are located above and in front of the user. Cameras 2 and 3 are located beside and behind the user on the floor. Camera 6 is located almost directly above the user.*

4 System Overview

In this system, 6 Vicon cameras are arranged around a plastic table, with 3 of them suspended from the ceiling on front of the user, 2 were based on tripods behind and on either side of the user, and 1 was suspended from the ceiling almost directly above the table, as seen in Figure 1. It's expected that the cameras can be configured in any way, as long as the center of the table is always visible to multiple cameras. To start using the MagDrop software, the user needs to first build a skeleton of the model that they want to sculpt using the magnets and ball bearings. Once this is done, the user must attach motion capture markers to each of the bearings using some sort of mounting putty.

Once the basic skeleton has been built, the user can use the Vicon

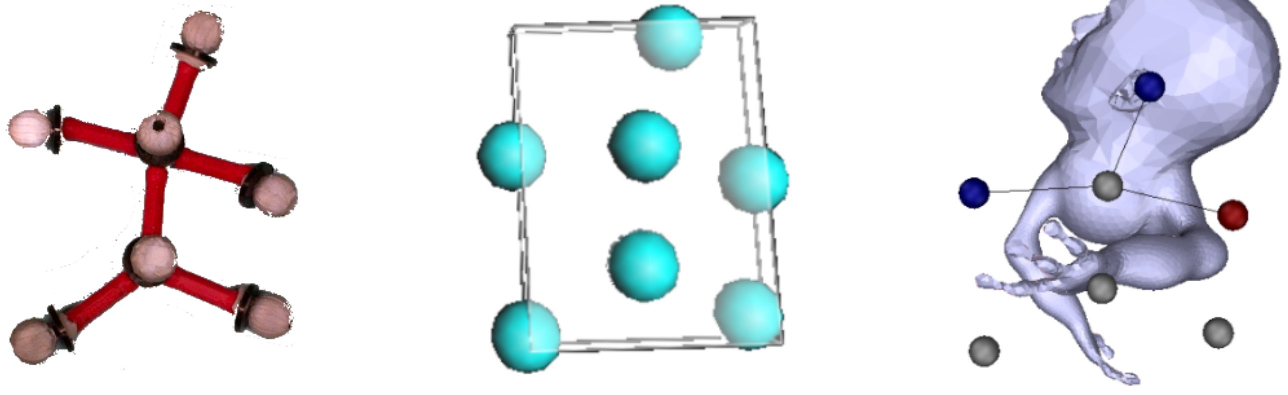

Figure 2: *The process pipeline. Shows the physical model (left), the iQ body (center), and the MagDrop skeleton and rendering (right). The slight discrepancies between the physical and iQ model are due to the viewing angle. The slight discrepancies between the iQ and MagDrop models are a result of projecting the iQ model into 2D.*

iQ software to group the markers into a body. The markers can be added to the body in any order, but the last marker selected will compose the "core" of the virtual model.

After the body has been constructed in iQ, MagDrop can import the marker data, creating multiple nodes. At this point, the user must build a skeleton in MagDrop. This skeleton will be used to render the final model. To join nodes together, the user can click on a node and drag the mouse to another one. Users can assign a mesh fragment (part of a previously created mesh) to a node by clicking on the node and then selecting the mesh fragment from a list.

Users can render the full mesh by pressing the spacebar. The mesh is rendered by starting at the "core" node (the last node added to the body using iQ) and attaching the mesh fragments for all nodes that are attached to it. These fragments are attached where the ray from the current node to the base node intersects the mesh. This is done recursively for all nodes.

If the position of the markers has changed, (e.g. the user moved the arm of a figure they were modeling), they can press the tilde button (~) to update the nodes in the MagDrop software. Pressing the spacebar again will render the model using the updated positions.

5 Analysis

The results of this experiment were underwhelming. Due to a number of limitations with the software, as outlined below, it was only possible to render very crude models, as seen in Figure 2.

The majority of issues were due to issues with the DragDropTool source code, which the MagDrop software was based on, while some were due to problems with the hardware and the medium that was being used for sculpting.

The first issue with DragDropTool is that it was built on libraries that used features specific to Microsoft Visual Studio 2008, which were removed in newer versions. As a result, the only compatible C++ compiler used the C++03 standard. There was no way to use a compiler that supports the newer C++11 standard, since the Visual Studio 2008 compiler does not support it, and there is no way to change the compiler it uses. The main consequence of this was that the entire program was run in a single thread. As a result, there was no way to update the model in the MagDrop software in real time. Attempting to update the model in the same thread that rendered the model and handled user input would simply cause the program to freeze. This also caused issues that prevented parts of the mesh from being rotated, which is discussed in the next section.

The other issues were simply a result of using the DragDropTool code in a way that it was never intended to be used. DragDropTool was designed with the intention that there is a single stationary mesh, and additional meshes would be added onto it. It was designed to be used with a mouse as an input device, so most of the functionality does not work correctly when automated. For instance, there is no way to reliably scale objects based on the distance between nodes. The scaling could only be uniform, and would either scale the mesh fragment before attaching it, which could cause issues with collision detection, and may not allow the user to attach the mesh, or after attaching it, which would cause distortions in the mesh fragment. There was also an issue where the direction that a mesh fragment should be facing could not consistently be determined. The DragDropTool code would simply use the normal vector at the point where a mesh fragment is attached as the up direction for that fragment, and there was no way to use a custom direction instead.

A final issue that arose from using the DragDropTool code was that the mesh fragments could not be updated in real time. Instead, users need to press the spacebar to render the scene and the tilde button to update the position of the nodes. This is due to performance issues with DragDropTool, which suffers from lag when trying to do this, making it unusable. As well, because DragDropTool was not designed with the intention of being able to move sections of the mesh without selecting them manually, attempting to automate the process and update it in real time caused the program to become incredibly unstable.

As a result of the inability to scale and rotate parts of the mesh, and the inability to define the "up" direction for each mesh fragment, the only models that could be constructed were rather crude.

There were some issues that resulted from using the Vicon system. The largest of these issues stems from the fact that the Vicon system is unable to detect any object that is not a retroreflector. In order for the cameras to properly detect an object, it must either be retroreflective or have a retroreflective marker attached to it. Because the ball bearings used in the Geomag magnet toy are simply steel bearings, the Vicon cameras cannot detect them. As a result, it was necessary to attach motion tracking markers to the ball bearings.

When comparing the position of the bearing to the position of the marker, there was a discrepancy of approx. 1cm. To accommodate for this, the model in MagDrop was projected into 2D, to prevent this discrepancy from having too large of an impact on the model. It may be possible to avoid this in the future by wrapping the ball bearings in retroreflective tape. This was not done here due to time and cost constraints.

6 Limitations

There are some limitations that result from using magnets, which cannot be avoided. The largest of these is that magnets have discrete lengths. For instance, every magnet is approximately an inch long. This means that adjacent bearings will be roughly an inch apart, with a similar distance between adjacent nodes. While the distance can be increased by adding more magnets, the distance will always be in increments of 1 inch. Without buying shorter magnets, it's impossible to place bearings at different lengths (e.g. 1.5 inches). This causes problems when trying to construct a model that requires parts of the mesh to be placed at different intervals.

Another major limitation was the inability to rotate a mesh fragment around it's "up" axis. This is due primarily to the fact that a magnet with a ball bearing is symmetric, and there is simply no way for any camera to detect the rotation by comparing the magnet before and after the rotation. As a potential workaround multiple markers could be attached to each node, but this is cumbersome would make it much harder to control the shape of a model, since the markers would interfere with the user's ability to position the magnets. Another alternative, which may be more viable, would be to attach markers to the user's fingertips (either directly using putty, or onto a glove that they could wear). The Vicon system could then track the motion of the user's had, and determine when the user grabbed a bearing or magnet, and when they rotated it. This could only work when the software is multithreaded, but as mentioned above, the DragDropTool code does not support multithreading, and attempting to access the Vicon data in real time would cause the software to freeze.

Another issue with using magnets is the weight of the magnets and ball bearings. While the magnets seem to weight very little, they are heavy enough to cause issues with structural integrity. In short all but the smallest sculptures are too heavy to support themselves, and will need some other form of support.

7 Conclusion and Outlook

In this experiment, I proposed a new way of sculpting virtual models by using magnets as a physical proxy. During the course of developing the MagDrop software, I encountered a large number of issues that stemmed from the original DragDropTool source code. These include the inability to update the model skeleton in real time, the inability to scale objects properly, and the inability to properly determine the orientation of objects. While these bugs could be fixed by rewriting significant portions of the code base, there are other problems that resulted from the magnets themselves, such as the inability to rotate objects and the fact that the distance between nodes is in increments of 1 inch. While it should theoretically be possible to develop software to track the user's fingers and determine when they are trying to rotate a magnet, the inability to place magnets between nodes (instead of at 1 inch intervals) means that it's very difficult to get precise placement. This means that there are some models that might be impossible to build properly using magnets, simply due to imprecision.

While most of the problems encountered here can be solved by rebuilding the entire program from scratch, some of the problems, such as the scaling and rotation issues, as well as the lack of precision when positioning nodes, lead me to conclude that magnets may not be a viable medium to use as a physical proxy. However, because so many of the problems were caused by the code that was used to develop this software, I believe that further experimentation must be done in order to safely draw a conclusion.

References

LIU, X. 2004. *Editing Digital Models Using Physical Materials [microform].* Canadian theses. Thesis (M.Sc.)–University of Toronto.

PIHUIT, A., KRY, P., AND CANI, M.-P. 2008. Hands on virtual clay. In *Shape Modeling and Applications, 2008. SMI 2008. IEEE International Conference on*, 267–268.

SCHMIDT, R., AND SINGH, K., 2009. Drag-and-drop surface composition. `http://papervideos.s3.amazonaws.com/DragDropSurfComp09.pdf`.

SHENG, J., BALAKRISHNAN, R., AND SINGH, K. 2006. An interface for virtual 3d sculpting via physical proxy. In *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, ACM, New York, NY, USA, GRAPHITE '06, 213–220.