CREATING DEFORMATIONS AND TUNNELS IN A SURFACE USING LAYERED GEOMETRY WITH ADAPTIVE FILTERING

A Thesis

by

JACOB BROOKS

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2005

Major Subject: Visualization Sciences

CREATING DEFORMATIONS AND TUNNELS IN A SURFACE USING LAYERED GEOMETRY WITH ADAPTIVE FILTERING

A Thesis

by

JACOB BROOKS

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Ergun Akleman
Committee Members,	John Keyser
	Richard Davison
Head of Department,	Phillip Tabb

August 2005

Major Subject: Visualization Sciences

ABSTRACT

Creating Deformations and Tunnels in a Surface Using
Layered Geometry with Adaptive Filtering . (August 2005)

Jacob Brooks, B.S., Texas A&M University
Chair of Advisory Committee: Dr. Ergun Akleman

With this thesis, I present a method for creating footprints and tunnels in a surface through the use of layered geometry. Rather than using a single geometric surface, deformations are created through the interaction of a polygonal object with multiple layered planes. Contrary to common methods such as solely using displacement maps or techniques used in fluid dynamics, none of the layered geometry moves. With adaptive filtering and layered geometry, one can create complex deformations resulting from sliding, digging, and surfacing. Its volumetric nature allows interaction to create overlapping shapes, tunnels, and holes in a surface, while alleviating the ultimate problem of broken geometry.

To my wife Amanda Brooks

ACKNOWLEDGMENTS

I would like to extend my greatest thanks to a handful of people who have given their hard work and time to make my education possible. To start off, I would like to thank Ergun Akleman, my thesis chair and professor throughout my graduate studies at Texas A&M. I am forever grateful for his unending support, wisdom, and encouragement. My committee members, John Keyser and Richard Davison, were favorite professors of mine in my undergraduate days, and I was honored to have them be a part of my graduate research. Their support, advice, and flexibility will always be remembered. Mostly, I would like to thank my wife, Amanda Brooks. She has supported me through graduate school, and I can only hope to provide a small fraction of what she has done for me for her once in the professional world. Her undying motivation, support, and love have been too great for measure. I have no idea how I would make it any one day without her.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

I.1. Motivation

With this thesis, I present a method for creating footprints and tunnels in a surface through the use of layered geometry. Rather than using a single geometric surface, deformations are created through the interaction of a polygonal object with multiple layered planes. Contrary to common methods such as solely using displacement maps or techniques used in fluid dynamics, none of the layered geometry moves. With adaptive filtering and layered geometry, one can create complex deformations and holes in a surface and alleviate the ultimate problem of broken geometry.

Imagine you are outside on a beautiful, sunny day at the beach. What visual clues make it evident that what you are seeing is the familiar place we all know as the beach? The first things noticed might be the billowy clouds, the ocean's waves, or the brilliantly bright sun. However, these evident clues are not always what make us believe that that particular space is in fact what we commonly know. The subtle interactions between various elements within that environment helps to reinforce just that. As the wind blows, the trees move; as the waves roll in, the sand beneath dampens. Interactions lead us to believe all the visual elements in the scene collectively form the common environment with which we are familiar.

In creating virtual worlds to be displayed as animated sequences of images, computer artists must represent the visual elements of a particular space to the best of their ability, while always understanding that any natural or imaginary scene could

The journal model is *IEEE Transactions on Visualization and Computer Graphics.*

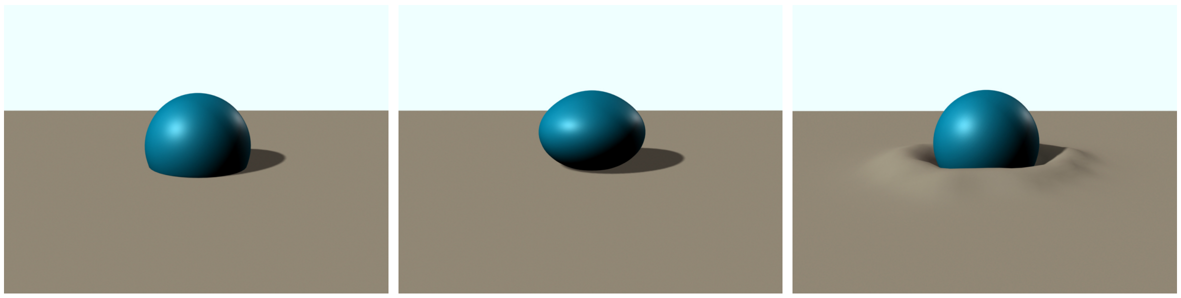


Fig. 1. Collision with penetrable surfaces generally yields some form of response to the interaction. In this case, the image on the left is missing just that.

potentially be far too complex to represent every physical element present. It is my duty as a 3-D artist to illustrate those key, yet subtle, interactions properly, in order to help the viewer better understand and relate to the environment they see. Just as it is extremely challenging to understand the plot of a movie spoken in an unfamiliar language, it is difficult to visually understand an environment that does not behave one would expect it to. For example, one of the images in Figure 1 seems to be a bit awkward.

Of course, throughout life experiences, we have all learned that when two objects collide, an effect occurs between the surfaces based on their respective material properties. The displacement of the ground around the ball (Figure 1 - right) or the ball deforming to account for contacting a more robust surface (Figure 1 - middle) are effects we are accustomed to seeing, making them more comfortable to view. The absence of some interaction effect causes the left image of Figure 1 to look a little strange. Neither the ground surface nor the ball has been affected by their interaction. We can also learn a lot about the material properties involved by analyzing such a simple interaction. It is easy to see that the soft ground structure on the right might resemble some mud-like structure, while the middle image's sphere could be made of a rubber-like material. Again, the image on the left does not tell us a whole lot. It just seems a bit confusing. While the shading and lighting in Figure 1 are far short of spectacular, these subtleties tell the viewer a lot about the environment's surroundings.

In viewing a highly complex scene such as a beach, these subtle interactions become second nature to our eyes and are often not our primary focus. So, you might initially think those interactions should be given lower priority and importance when trying to recreate the scene three-dimensionally. However, when these characteristics are absent, the trees not blowing with the wind or a foot not leaving a print in the sand, the resulting awkwardness, in turn, distracts us from the more important issue at hand, for example, the storyline or mood of a scene. Figure 2 illustrates a sequence from Blue Sky's animated film "Ice Age." In this shot, the sloth slams the ground with a rather forceful impact, yet the ground beneath him is left unchanged. One inherently expects to see some alteration of the dirt below. When this expectation is not met, its absence draws our attention. This awkwardness is what initially drove me to research methods of interaction between rigid objects and deformable surfaces, representing penetrable matter.

I.2. Goals and Scope

Knowing the time limits for developing a thesis, it was important for me to refine the scope of my research by clearly defining my goals. I had three main goals upon starting my research. I wanted to develop a way to automatically generate surface deformations given an animation, create complex interaction effects resulting from digging, surfacing, and sliding, and generate a final animation, having deformations as an addition to the story, rather than the source.

Knowing I did not want the deformations to be the immediate focus of the

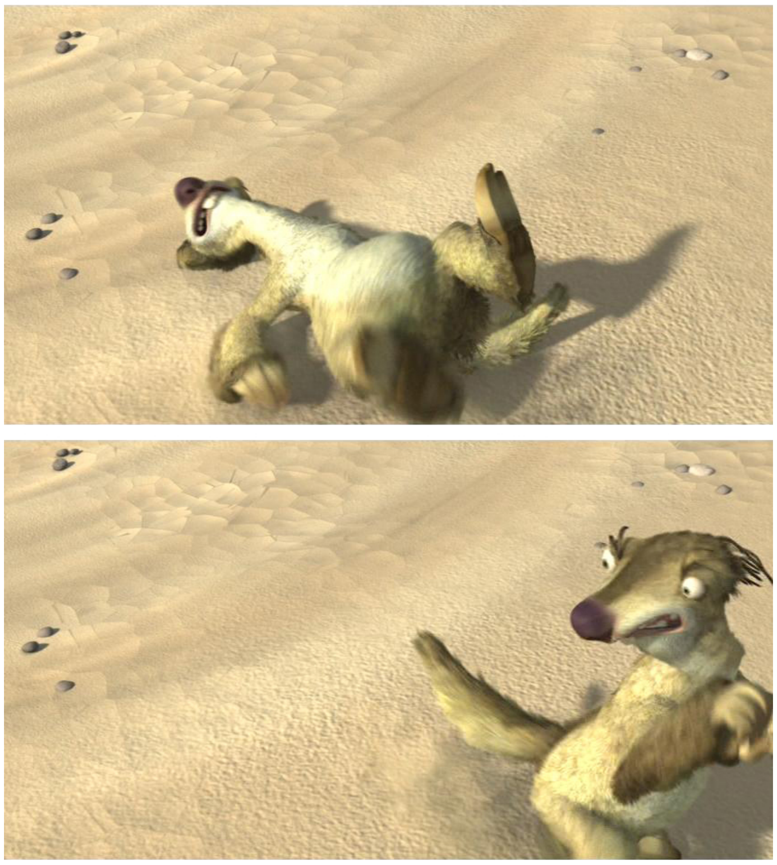


Fig. 2. After leaving the area of impact, the ground remains unaltered. Images sampled from the “Ice Age” DVD[1].

animation, I was able to create a plausible scope for my work. I did not intend for the results to look fully physically correct, so extensive research in physically based simulations was unneeded. However, due to the complexity of penetrable surface motion, referencing its physical properties proved to be a helpful step in achieving a less exact, yet effective, solution.

I.3. Other Deformation Techniques

When creating penetrable surfaces resembling sand, mud, or snow in computer graphics, it is necessary to consider a method of handling collision effects. Collisions create responses that alter the geometric structure in different ways. When interacting with surfaces of this type, the presence of horizontal movement plays an important role in illustrating a shifting of weight or a direction of motion.

The traditional use of animating displacement maps to create an effect, such as a character's footprints in snow, can be a quick and effective solution when viewed from a distance. However, when analyzed closely, the map's innate properties prove to be extremely limiting for many circumstances. For instance, Figure 3 illustrates a baseball thrown at an angle into soft mud and a sample result one might expect to occur. If soft enough, the mud will always displace along the direction of motion to some degree based on its velocity.

More importantly, it is likely that the mud-like surface will shift in multiple directions. Not only is the mud compressed beneath the ball, it is also shifted laterally with respect to the horizontal component of the ball's velocity. Therefore, in recreating this action properly in computer animation, one needs to have some means of altering the surface both vertically and horizontally. This seemingly simple necessity proves to be an inherent problem for displacement map use.

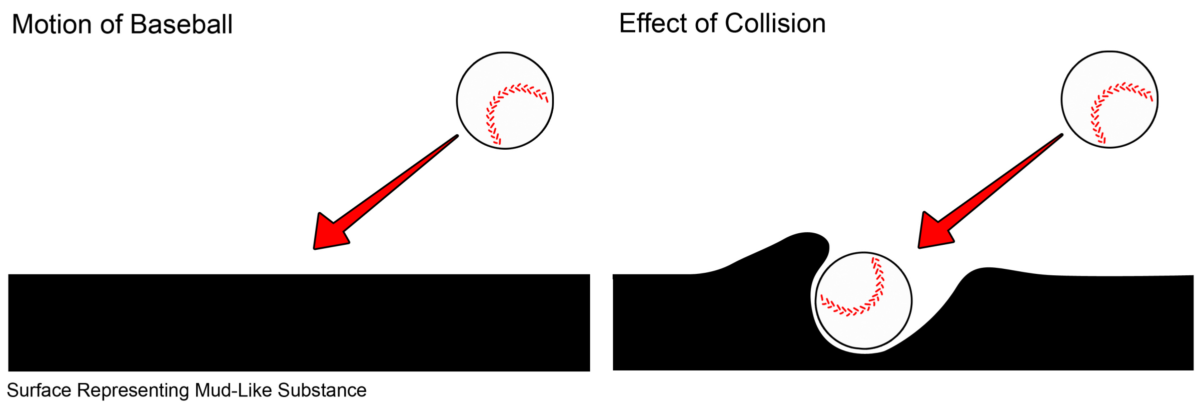


Fig. 3. With deformable surfaces, overlapping structures are often created.

A displacement map is created as a grayscale image, where the whites represent the higher elevation of a surface and the darks represent the valleys. This map is then applied to a surface, translating the geometric vertices in the vertical direction with respect to their corresponding grayscale value. This method can create a wide range of vertical variation in a surface, yet it prevents any overlapping or lateral variation from occurring. Unfortunately for the common case of an object colliding at any direction other than perpendicular to a surface, the nature of a displacement map makes it impossible to create the effect of horizontal movement. A common way to test if a particular shape can be generated from a flat plane using displacement is to use a vertical line test. If the surface intersects the vertical line in more than one place, the shape is impossible to create using a single displacement map. Figure 4 illustrates its limitations.

If only a few collisions occur in a particular scene, modeling the deformation shift in a surface might be a valuable approach. One can duplicate the original surface, alter the model to more appropriately reflect the shape resulting from the collision, and use a blend between the two shapes. This way, the animator can represent a shift of the surface over the time of interaction. The main problem with this solution

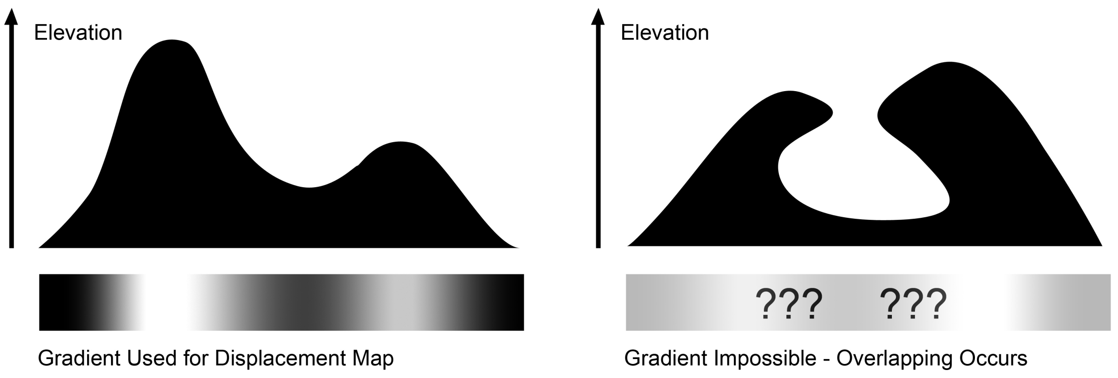

Elevation
Gradient Used for Displacement Map
Elevation
??? ???
Gradient Impossible - Overlapping Occurs

Fig. 4. With conventional displacement using a single grayscale map map, it is impossible to generate the overlapping effect shown on the right.

is that it is highly case specific and, depending on the complexity of the model, it will generally only work for a particular collision instance. If the animator were to change the moving object's animation, the modeling phase would again be revisited to account for the new shape generated. This technique is highly effective for close-ups and situations where a high level of detail is needed, however, in handling penetrable surfaces like mud and snow, there are often sequences of shots involving numerous interactions per shot. The effects of a collision with this type of surface tend to occur frequently with a variety of objects and are rarely seen at a close distance. Therefore, modeling a particular shape can be far too tedious, time-consuming, and inflexible for a complex shot.

A more effective approximation of interaction response can be done through the use of fluid dynamics and other key principles of physically based modeling. Though the results tend to be visually accurate and stunning, the computation involved in creating such effects can often be far beyond the scope of the task at hand, and extremely time consuming. Fluid effects have proven to be highly successful with their representation of water in many of today's popular computer-generated films, such as

Pixar's "Finding Nemo" or Dreamworks' "Shrek 2," however, rendering tends to be challenging and time-consuming. This application focuses on representing volumes of mass that continually interact with themselves. Its implementation requires complex grid cell calculation to approximate the fluid amount in each voxel and its current velocity with respect to its neighboring voxels. Fluid dynamics is an extremely powerful effects tool, but it can be a bit of an overkill to generate the deformation of a footprint. For my purpose, the important concept to take from fluids is creating geometry as a volume of data that can be broken down into smaller, repeatable elements.

I.4. Thinking Past the Surface Layer

Creating surface deformations is possible with any of the aforementioned methods. However, what if an object not only deformed a surface but also traveled through that surface, creating a hole or tunnel? Is it possible to create such tunnels through a surface, so that a worm, for instance, could enter the ground at a particular location and exit from a different one, leaving a clear path through the structure? With a single piece of geometry, it is not.

When modeling surface geometry, it is important to remember that the inside of the object represented is hollow. The geometry of a traditional 3-D object constructed from polygons exists only on the surface level. Just like a balloon or a blown up beach ball, the inside of the object is filled with empty space or air. It has no geometric structure on the inside.

Yet the desire to dig into a surface requires that, as we move through, the surface retain its geometric structure despite the interaction. The best visual example that comes to mind is the representation of an anthill in 3-D. Though ants will continually interact with the outside surface of the mound, if one were to poke the side of it with

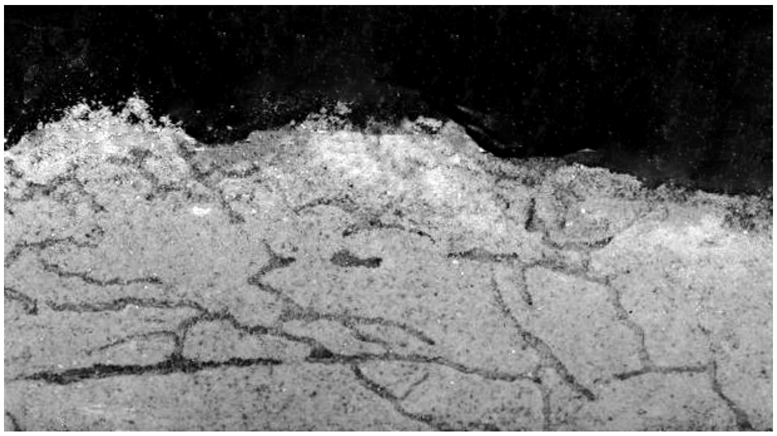


Fig. 5. Structures with holes throughout tend to be a challenging problem to create and interact with.

a stick, he would see the hundreds of others digging the tunnels beneath. Now I, personally, have never done such a cruel thing, nor has any other typical five year old I have met, yet I can imagine what it would look like. It would be extremely difficult for this continuously-changing, porous structure to be represented by a single surface model. A sliver of the tunnels complexity can be seen in the antfarm pictured in Figure 5.

Aside from modeling the anthill, imagine accounting for the changes made to its tunnels over time, as the ants continually shape and build the mound, digging new paths, and often branching into other existing ones. This is far beyond the scope of surface geometry. In the case of penetrating through another existing tunnel, how would the two tunnels merge? Simply moving geometry would not handle such an event, for the shifted vertices would penetrate through the other tunnel, giving us a completely different result than two joined paths. Eventually, when two or more paths cross, the surface geometry overlaps and breaks its geometric structure.

Broken geometry leads to unexpected results when rendering. The structure intersects itself, causing the backside of the interpenetrating faces, whose normals

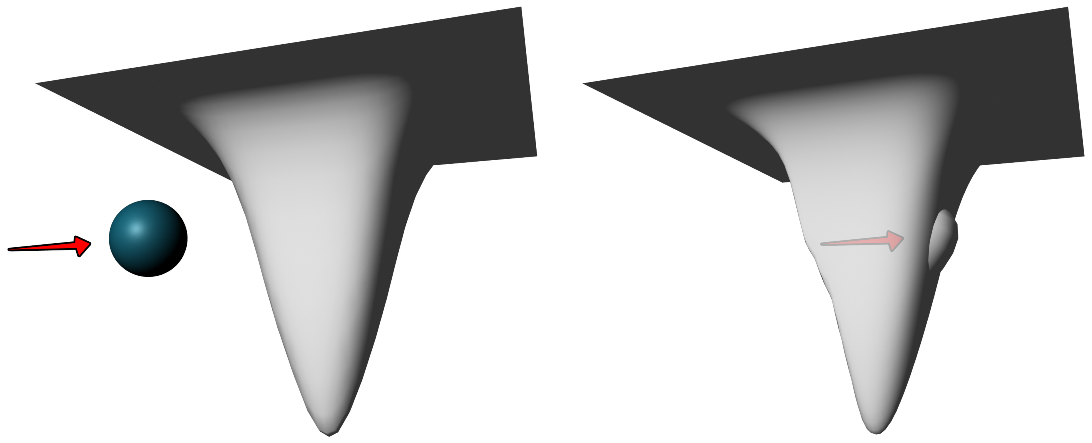


Fig. 6. Left: Sphere moves towards the original geometry. Right: Translating the side to account for interaction results in broken geometry as it penetrates the other side.

are pointing in the opposite direction, to be rendered. Most often, the visual results are unpredictable and therefore undesirable. In instances where an object is moving through a surface, it is likely that using single surface geometry will result in broken geometry, as in the examples of Figure 6 . So what other options are available?

Well, this potential problem of creating animated deformations lies with the movement of the geometry itself. Referring back to Figure 3, as the ball hits the mud, it is a common solution to shift the geometric structure downward to create the effect of a divot or an imprint made over time. As the motion of an object becomes more complex, the vertices must be moved farther to accommodate the desired shape. This all seems to make perfect sense initially. Unfortunately, as described in the case of the tunnels in the anthill's tunnels, moving vertices and pushing a shape too far is sure to result in broken geometry. So, if moving geometry tends to result in broken geometry, why not avoid moving it altogether?

This may seem like an absurd solution since we are so accustomed to creating a particular shape by altering the surface geometry. However, if we think of geometry as a group of layers stacked atop one another, much like our skin or the layers of the Earth, certain limitations of traditional deformation are alleviated. With multiple layers of interaction, it is possible to create holes throughout as well as generate vertical and horizontal variation in a surface. The volumetric nature of a layered surface presents the opportunity for convex, concave, and overlapping terrain properties and allows for the dragging or sliding of objects through the surface. More importantly, rather than pushing or pulling geometry to generate these shapes and risk breaking the geometric structure of the surface with self-intersection, layer properties such as transparency and normal direction can be manipulated to create a similar effect of motion through a surface. The actual geometry of the individual layers never changes at all.

I.5. Layers: A Plausible Solution

As with any research, the resulting product relies heavily on the initial scope of the work. If one had to only recreate a single baseball falling into the mud in one shot of an entire movie, using layered geometry is most likely overkill to represent that one instance of interaction. It could definitely do the trick, but this effect could be easily and efficiently achieved using other simpler methods. However, if creating story that relies heavily on surface interaction, for instance, “The Life and Adventures of Earl the Earthworm,” digging holes and illustrating his movement through the ground is essential to depict how he interacts with his environment. Creating a tunnel or poking through on the surface elsewhere is a necessary visual element in order for us to understand his way of life. Though it may not be the focus of the story, it is a vital element in telling the story, and awkwardness would result from its absence.

CHAPTER II

RELATED WORK

Developing new ideas almost always stems from the analysis of existing ones. Studying the current methods used elsewhere is essential for learning from previous mistakes and successes. Understanding theses successes, problems, benefits, and drawbacks leads to a more effective and timely solution. Research areas applicable to my goals include geometric modeling, physically based simulation, and surface animation.

In order to deform an object, it must first be created/modeled. A variety of shape representations are commonly used in creating geometric structures. Included are parametric surfaces (polygons, spline patches, and trimmed NURBs), subdivision surfaces, and implicit surfaces. These representations, in general, relate to the shape of the surface of an object [3].

Developing methods to represent geometric structures as a volume of elements became a popular area of research in the late nineteen-eighties. For example, in the representation of fluids, it is impossible to think of fluid as a single piece of geometry in the case of pouring and splashing into some container. A fluid can have varying masses and velocities across its surface at any given time. Thus, the concept of breaking the overall geometry into separate grid cells with individual properties is essential to parallel how molecules of fluids interact [6].

Volumetric geometry also aims to create extremely complex three-dimensional models. A common goal is to retain a higher level of detail without having to increase the geometric resolution to unmanageable sizes. Volume textures have proven to be a useful tool to do just that. Peng describes a way to use volume textures on the surface of basic geometry to achieve highly detailed results such as chain-mail or a woven

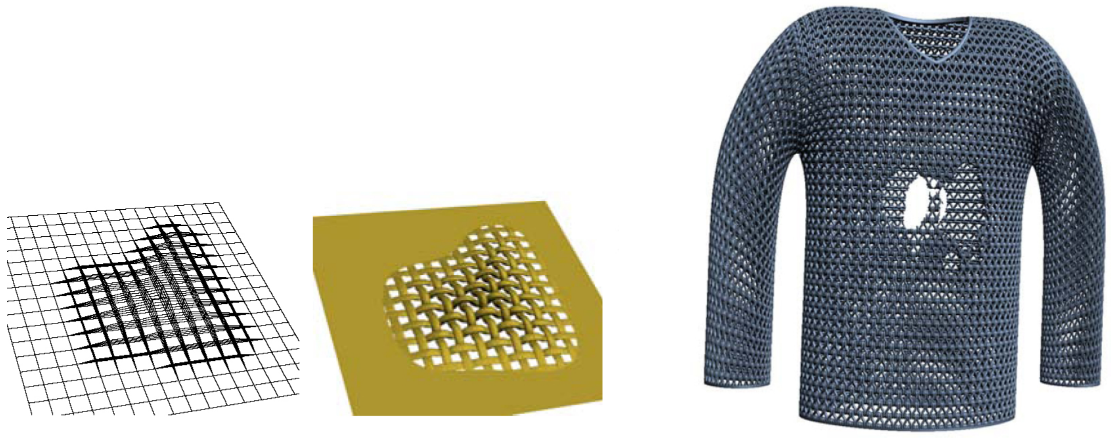


Fig. 7. Sample images from Peng's SIGGRAPH paper [5].

basket. The success of the volume textures relies on its application to a repeatable volume of geometry. Achieving this volume involves breaking the single surface into multiple layers. His methodology includes creating non-intersecting shells that form an "elastic compressible skin" about the geometry's surface (Figure 7 - left). Each of these shells has the same connectivity, allowing for repetition of the volume texture. The shells then have implicit functions within each, describing a more complex shape. This lets mathematical functions dictate the shape of the model, not the resolution of the model's surface [5]. The obvious goal here is to create an efficiently complex model, such as the chainmail on the right of Figure 7, not to handle how that model might change over time.

However, many circumstances in nature represented in computer graphics do inherently require changes in a model over time. For instance, if snow were to fall for some time period, one would expect to see accumulation of the flakes on the ground or on other objects in the scene. To do this without handling the heavy computation of particle collisions as they stack atop one another, one could devise a subdivision scheme that will subdivide the geometric mesh to a higher level of detail based on

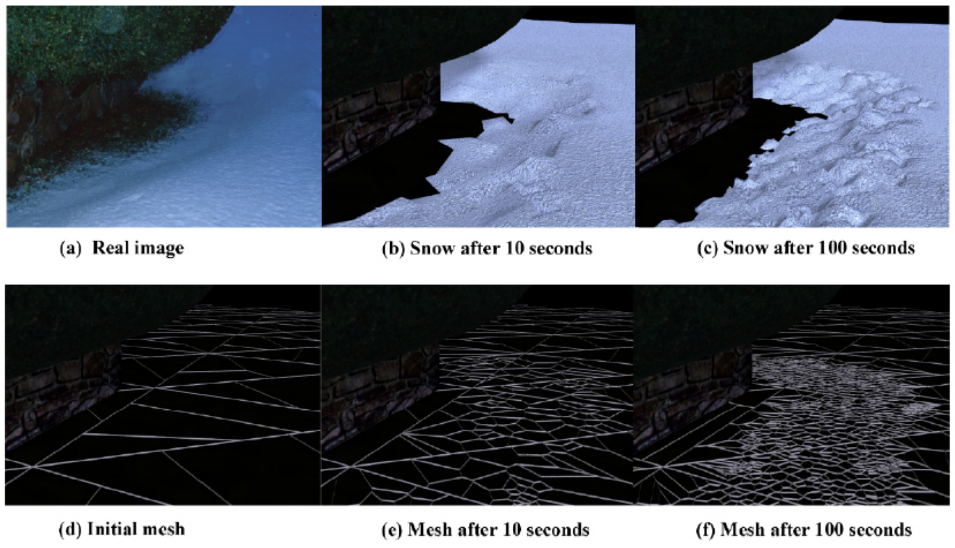

(a) Real image
(b) Snow after 10 seconds
(c) Snow after 100 seconds
(d) Initial mesh
(e) Mesh after 10 seconds
(f) Mesh after 100 seconds

Fig. 8. Sample images from Fearing's paper[2].

the current amount of snowfall. Fearing did just this in his research on "Computer Modeling of Fallen Snow." The geometry changes over time, continually sub-dividing into finer definition as the positions of specific regions of the mesh shift (Figure 8) [2]. This method is quite effective for demonstrating the change of a geometric structure over time, however, the resulting geometry continually becomes geometrically more complex and can potentially become difficult to manage when rendering.

Instead of only changing a model over time, I must be able to alter it based on its interaction with another object. Rigid body interaction with penetrable surfaces introduces this new element not yet considered. Some object will collide, intersect, and move through the ground geometry at any given frame. Handling this complexity was approached by Sumner, O'Brien, and Hodgins in their research on "Animating Sand, Mud, and Snow." The geometric structure is modeled as a height field of vertical columns that compress and move according to the impact of a rigid body model. At each frame of interaction, the columns heights are adjusted such that no column is colliding with the object. To do so, the geometry can only move vertically, forming a compressed shape around the collision. To account for various types of surface materials, parameters such as liquidity, compression, roughness and slope are adjusted by the user [7]. The results from this research are fascinating, however, overlapping ground structure cannot be generated, due to the lack of lateral motion, making digging and surfacing impossible interactions.

Given the previous research in the aforementioned areas, there does not seem to be one clear path to take in order to achieve my goals. Each does, however, contribute an essential idea or concept that aids in achieving my desired results. The key ingredients in applying my solution involve three main ideas: creating some form of geometric volume representation, developing interaction rules with that structure, and modifying that structure in some way over time. I plan to incorporate these research concepts to create a penetrable layered ground plane, whose layered sections will be treated individually as in fluid dynamics, while preserving the illusion of being one consistent piece of geometry changed over time.

CHAPTER III

METHODOLOGY

There are three critical stages involved in developing this technique. It begins with creating a layered structure, followed by defining and refining the rules of interaction for that structure, and ends with creating an animation to demonstrate the results. This bottom-up building process overlaps each new step with its predecessor, making every decision valuable for its use in the next. As in developing any visual effect in 3-D, the way to test its success is to perform test renders, analyze the results, and revisit the development stage. The most crucial and time-consuming of the stages involves fine-tuning the rules of interaction; however, without the presence of each of the other steps, it alone serves no purpose.

III.1. Creating Layered Geometry

For an object to interact with a layered surface, there must first be a way to create such a structure. Also, depending on the subject matter of the desired final animation, a user must be capable of specifying various parameters to describe the surface of interaction. Given these essentials, a tool is needed to take a user-specified surface and construct a volume from it.

Traditional 3-D modeling involves creating the surface of an object. For my purposes, the model represents a penetrable ground plane, such as mud or snow, and is interacted with by some other object moving within some region of its volume. It is essential then that a representation of that volume is created to allow this. Volume is achieved using duplicates of the original structure placed below the original layer of interaction (see Figure 9). Each layer must retain all of the properties of the

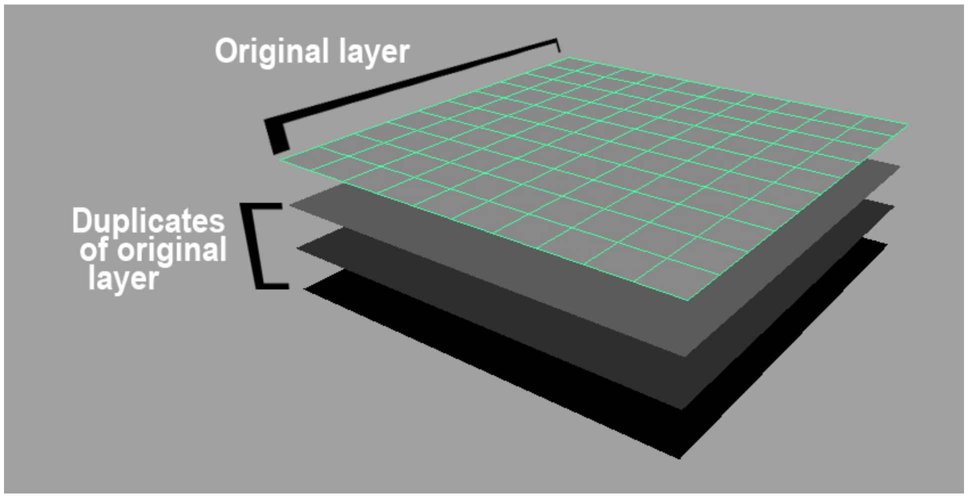


Fig. 9. An example of layered geometry. Each layer is identical to the selected original. I have spaced them out considerably for illustration purposes.

original layer, as it is translated beneath the next. Parameters of the initial surface layer include resolution, width, and length. This resulting structure should resemble one constructed by stacking Legos, except imagine stacking one beneath the other, leaving a gap between each layer as it is moved down. Figure 9 illustrates a simple surface and the layered structure created from it. Nothing more than duplication and translation is needed in order to create such a structure. The only parameter subject to change after defining the original layer is the spacing defined between each. It is important to note that this method is not limited to using a flat piece of geometry. A height field could be used to define the original layer's structure, making a more interesting shape when creating the lower layers. For simplicity's sake, I will perform calculations using a simple, flat plane.

III.2. Defining Rules of Interaction

The rules of interaction encompass all changes that could potentially occur to each of the layers' properties in one particular frame of an animation. Included are determining collision points with a specified object, handling the effects of that collision within each layer, and conserving the overall mass of the layered structure by altering its structure according to the direction, speed, and position of the animated object.

III.2.1. Handling Collision

In determining how to alter the layered structure upon a specific type of collision, we must first locate the collision when and where it occurs. Researchers have studied collision handling since the early stages of computer graphics. Collision detection determines the exact time frame when one object penetrates another. In the realm of physically based modeling, where various objects behave according to an approximation of physics laws, detecting collisions serves as the primary step for most simulations. If a ball is bouncing in a box, it must be able to detect the walls of the box, so that it does not bounce out. This would be impossible to simulate if there were no way of determining when a collision between the ball and the box's walls occurs. The same applies to the case of interacting with a layered surface. Nothing can be affected or changed until a collision is present within one or more of the layers.

Moore and Wilhelms addressed the topic of detecting collision and handling the resulting response in computer animation in the late 1980s [4]. Their methodology describes how to determine the collision between two polygonal rigid surfaces and the motion created from that collision. In dealing with convex polyhedra, the Cyrus-Beck clipping algorithm determines whether a particular point is inside a convex polygon. By viewing the polygon in a 2-D plane, the dot product of a particular side's normal vector and the vector from a corresponding vertex from that side to the point being tested yields either a positive or a negative number. After computing the dot product for all sides, if all of the results are negative, then the point is, in fact, inside the polygon. This method was then implemented to account for this occurrence three dimensionally. Others have implemented what is known as ray-triangle intersection to test whether a given ray intersects with a given triangle. This method has been used heavily in ray tracing for collision detection. Because my thesis focus is not to derive a new way for detecting collision and I will be using convex and concave polygons, I will use the ray-triangle intersection method for determining collision.

III.2.2. Handling Transparency

Once a collision is determined, there must be a way to create the collision response within each individual layer. As mentioned above, a key concept of using layers is that I will never move the geometry. I use transparency as a means of removing the mass from an area. Shading a layer's collision faces transparent helps me achieve a similar look to moving the geometry. In Figure 10, the sphere penetrates the topmost layer, and, if removed from atop the structure, a portion of the plane inside the object should no longer be visible, due to it being "compacted" by the ball's structure. This particular area of the layer is shaded to be completely transparent to reflect this action. Transparency is set on a face by face basis.

III.2.3. Adjusting Vertex Normals

To aid in blending the layers together as one piece of geometry, vertex normals lining each layer's collision region must be altered from their previous states. An example of this alteration is provided in Figure 11. I assign each vertex a new normal according to the interacting object's (the sphere's) closest contact point to that vertex. For a particular layer's vertex in collision, I calculate the inverse of the normal of the sphere's closest contact point and assign it as the new vertex normal. The desire of

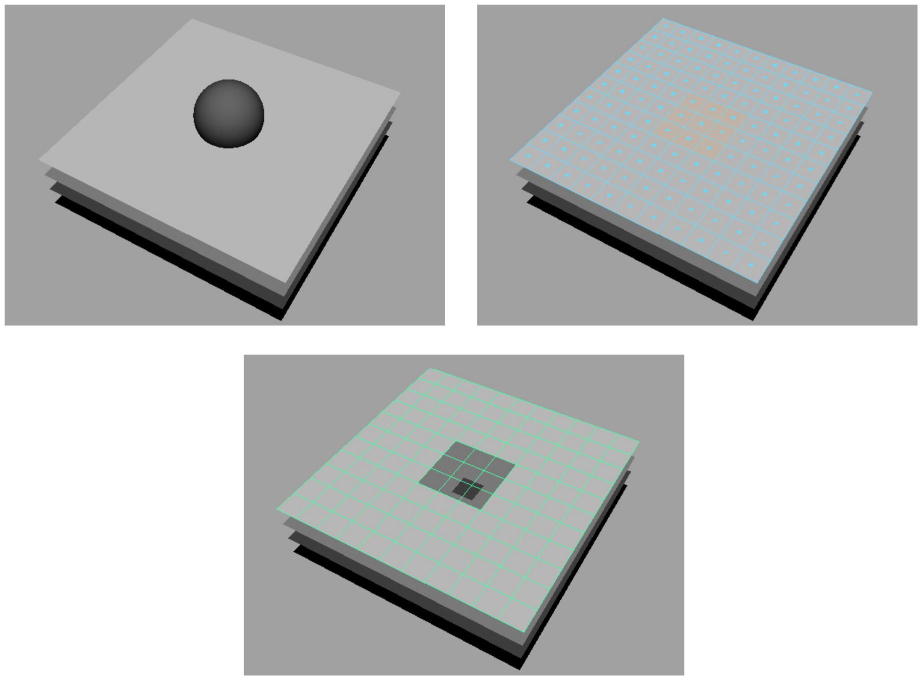


Fig. 10. Top Left: Current state of interaction. Top Right: Selection of faces in collision at that particular state. Bottom: Collision faces set to be transparent.

this action is to aid in reducing the obvious visual separation between each layer. The left image of Figure 11 illustrates the various normals of the sphere's surface with black arrows. The green vertices represent the vertices in need of vertex normal alteration. The sphere's face normals are then used to reassign the inverse to the surrounding vertices in collision. The resulting vertex normals are illustrated by the arrows in the right image. I could take this one step further and create a new geometric structure along the holy area to alleviate the layered geometry, but this would pose a bigger problem when creating tunnels or holes through the surface. Creating that complex geometry from a porous structure would be difficult.

Altering the vertex normals is a fairly simple concept for generating a single image, however, the goal is to develop a method that works over the course of an animation. When considering the motion of any object over time, the direction of

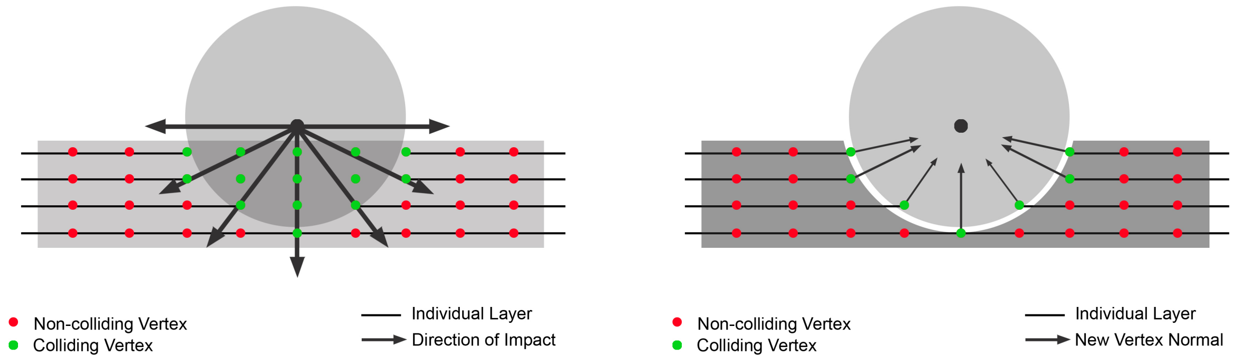


Fig. 11. Left: Vertices currently in collision with the object. Right: Adjusted vertex normals assigned according to the sphere's closest surface normal.

motion plays a large role in determining which vertex normals should be altered. The only vertex normals needing change exist in some interval area about the direction of motion. In the case of our sphere sliding through the layers in Figure 12, the vertices greater than ninety degrees away from the direction of motion should not be altered for a particular time-step.

In Figure 12's right image, the lone green vertex outside of the degree interval will not have its normal altered, even though it may have been selected as a vertex in collision. The reason for this is that once vertices are behind some region surrounding the object's direction of motion, they are no longer affected by the impact of the sphere. The vertex normal should never be changed if outside the region between the direction vector and its perpendicular vector. Anything altered beyond that point is wasted calculation, and if changed, the resulting shape often strays from the desired structure. Rather than achieving a channeled surface much like a half pipe in snowboarding, the structure will seem to have slight bumps in the areas where previously located. For any given interaction, half of the vertices can potentially be

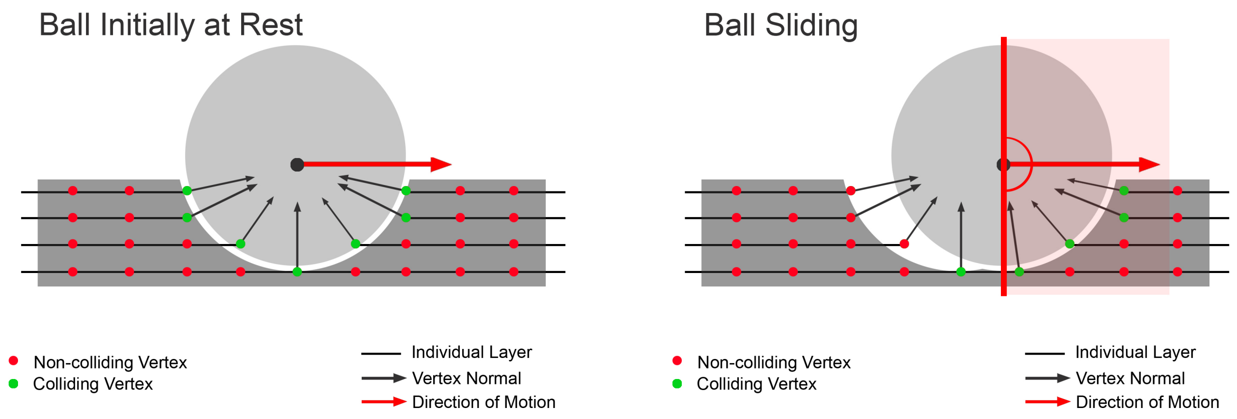


Fig. 12. Left: Vertices previously altered from collision are colored green. Right: Only change the normals for vertices located in some interval about the direction of motion.

thrown out, reducing the calculation time by half as well.

This method can also be implemented for a more complex object than a sphere. In the case of a concave shape like a peanut, of the ground vertices in collision with this object, only those in an area where the appropriate intersecting face's normal is within that ninety degree region about the direction of motion will be altered. Thus, contact with the portion of the peanut that has face normals greater than ninety degrees away from the motion direction do not need to be calculated. As in the case of the sphere, the new normal is set to the inverse of the closest face normal of the object in collision. Again, depending on the particular shape, possibly half of the vertices can be eliminated before calculation.

III.2.4. Conserving Mass

A more challenging response involves conserving the mass shifted upon interaction. As with any deformable matter, mass is always conserved in the process of interaction. It may be compressed or shifted according to the material properties, yet roughly the

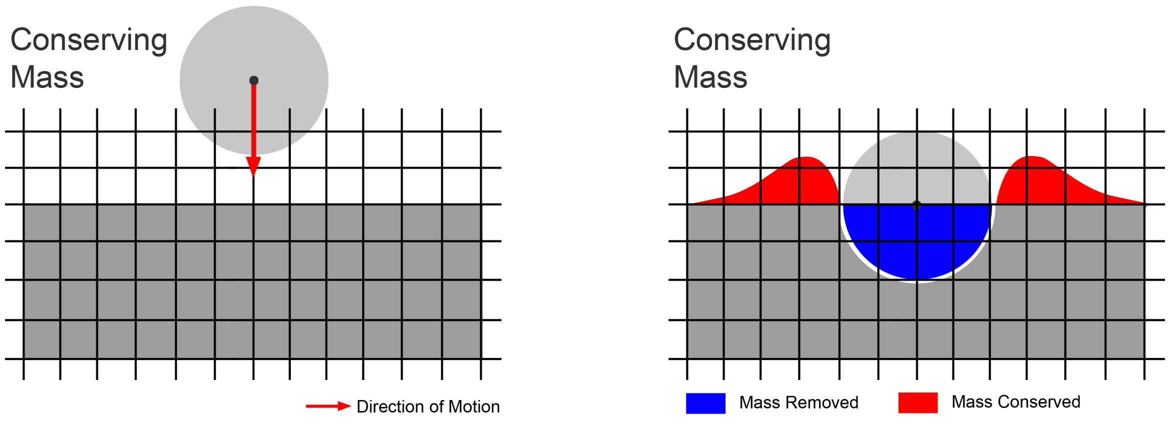


Fig. 13. Left: Motion towards the surface. Right: Mass is shifted according to the area of impact.

same amount of matter should appear to be present before and after interaction. In Figure 13, the red area represents the shifted mass resulting from the area in blue becoming depressed. Since it is generally not the focus of an image, it is unnecessary to make matter conservation an exact science; it is more important that it is present in some form. It should behave, on some level, as one would expect a real surface to if in the real world. Unfortunately, doing that is not always so simple.

We have a method for removing mass by altering transparency and vertex normals, however, there must be a solution for conserving that removed mass. Perhaps the ground should appear to shift in a particular direction, as in the case of a footprint. This scenario relies heavily on being able to create the illusion of a shift of mass within the individual layers, yet due to the nature of the lower levels, movement of the actual geometry is impossible and will break the structure. The lower layers handle the removal of mass, so how do we create the effect of its shifting elsewhere? When handling the topmost layer, a solution surfaces.

When creating the illusion of mass conservation, the alteration of the topmost

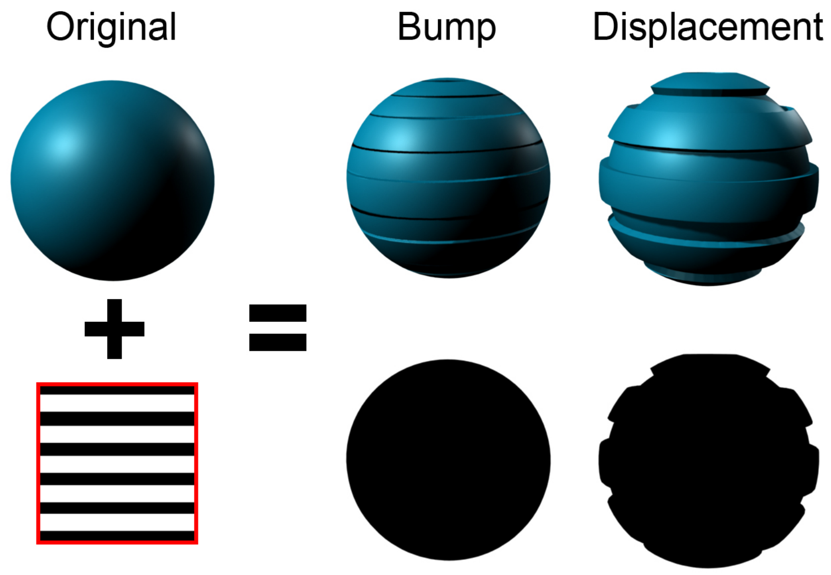


Fig. 14. In applying a striped texture map to a sphere, the difference between applying it as a bump map or a displacement map is evident.

layer proves to be extremely critical. With a layered structure, the majority of the layers are hidden beneath the top layer, and their job is to reflect the removal of mass when visible. It is then the responsibility of the topmost layer to conserve the mass properly. Various methods are capable of doing just this, each with their own respective benefits. One could solely alter the vertex normals of the top layer according to the collision response. This method illustrates the shift well, however, because the geometry itself is not altered, it may seem too flat when viewed from angles other than directly above. A common example of this problem is illustrated in Figure 14. The bump map alters only the vertex normals, while the displacement map actually translates the surface geometry. The difference is evident when looking along the outline of the geometry of each and seeing that geometry is only altered in the case of displacement.

Altering the geometry using displacement solely for the top layer is yet another solution, as long as the layers below never change. This method eliminates the overall flat look, but implementing it is yet another challenging problem. Displacing the top layer would involve no calculation at all if one were willing to individually paint the corresponding displacement map for each and every frame of animation. This solution is not only extremely time-consuming, but it also detracts from the goal of an automated process. It is my goal to derive a way to calculate these changes automatically for each frame.

In order to do so, one must utilize all of the information given at a particular time. The most important data at any moment include the selection of faces on the top layer that are currently in collision with the object, the direction and velocity of the object, and the center of mass of the object with respect to the topmost layer. Using these three pieces of data, generating a method for determining vertical displacement of the topmost layer is possible.

First of all, the faces in collision with the object form the overall 2-D topographic shape of the current collision. As with any imprint or deformation, the mass shifts around the collision shape created. This can be seen in the case of a spherical object falling into a penetrable surface. Radially from the center of the sphere, a ripple effect is created beyond the edges of the object. If falling from exactly ninety degrees above the surface, the shape generated will generally be the same going out at any angle from the center of the sphere. It may be easier to visualize this result when analyzing a 2-D slice from the deformed structure. The displaced matter is roughly identical in mass on either side of the object, yet the shape is flipped horizontally based on its orientation to the ball.

The resulting shape can be defined two-dimensionally by a number of key displacement translation points. Figure 15 illustrates the possible variation of those key points. For simplicity's sake, let's take the highest point displaced to be equivalent to

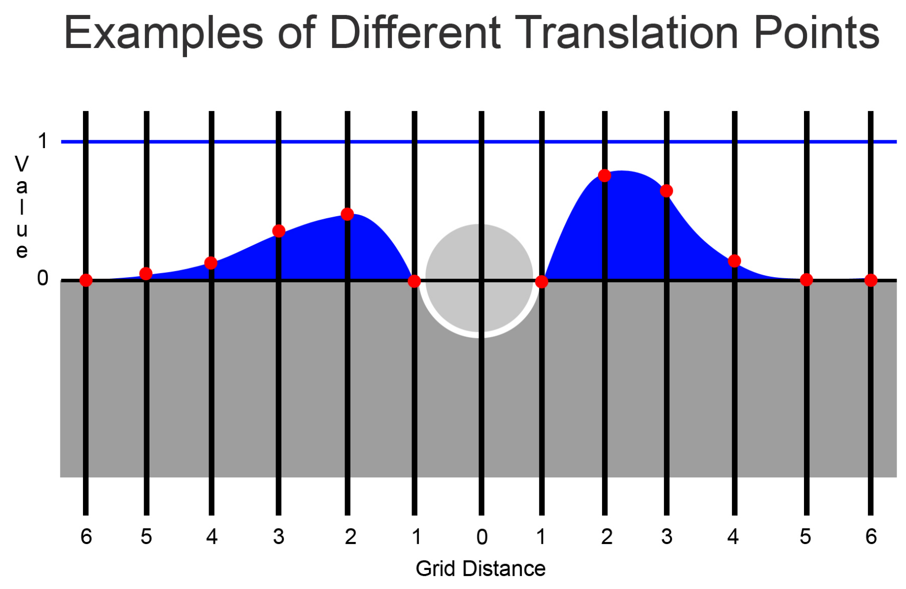


Fig. 15. Each of the red points is a sample height recorded from a desired shape. They are sampled at a repeated distance away from the area of interaction.

a vertical translation of 1, and the original top surface height to be 0. If we sample the deformed structure's height with respect to the initial height of the topmost layer, we can gather enough information to determine an estimate of how the shape was made based on its vertical translation. We can then analyze the height at a sample distance away from the collision point, repeating multiple times, until the surface resumes its original height. The red points in Figure 15 represent the interpolation points used to generate the shape of the area in blue. Their translational data collectively describes the shape of the matter displaced or shifted for a particular material. The points of translation can be altered to generate different effects and shapes useful for a variety of surface-types based on the material's physical properties. Given the approximation of the shape in 2-D, I must use that translation data to create some modification to a 3-D surface.

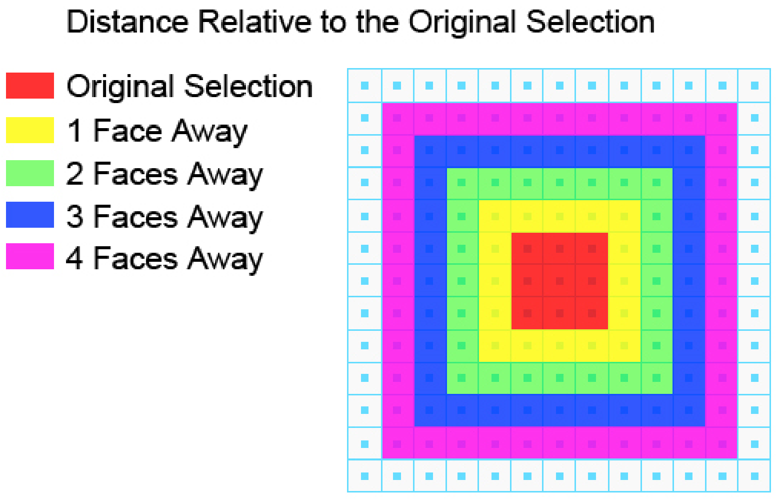


Fig. 16. The color of any particular face corresponds to the selection group it is in, all being a particular face width away from the original.

As mentioned before, the selection of faces in collision with the object is already calculated, and in Figure 16, it is shaded red. Based on this original selection, mass will shift a certain distance away. That distance is illustrated by the various colored rings in the image below. Each color represents a different area grown about the original selection. By first breaking the surrounding area into sections, each group of faces can be displaced to reflect a rough approximation for mass conservation. The pre-determined translation data from before is then applied to the respective colored region of faces. For instance, at one face width away from the original collision faces (the yellow area), we can move the geometry vertically in accordance with the first translation point. At two face widths away, we can move the next sampled translation value, and so forth. While the basic concept works, the 2-D grid gives a strong angular bias to the amount of displacement. For most cases, we want circular, smooth rings, not square rings. To create a wider variety of shapes and allow more cohesion between each particular translation, it is better to translate vertices instead of faces.

Two things must then be accomplished: a conversion from faces to vertices is needed for the collision area and a method for determining which vertices are 1, 2, 3, etc. sample distances away from the edge of collision. The solution for the first of the two is straight-forward. A transparent shader will be assigned to the collision face list prior to anything else. Given that selection of transparent faces and knowing that a face consists of four vertices, one can simply retrieve the vertex information for each face in the selection and add it to a new list of vertices. This new vertex list will be used as the base collision selection and will not be altered vertically or horizontally since it is within the colliding object's surface boundary. The only thing that will be done is re-calculating the normal of each vertex in the list according to the methodology mentioned before for the lower layers. Aside from that, this list will only serve as a base for adjusting the heights of the surrounding vertices.

Now that we have an initial selection of vertices, the second problem is to determine which surrounding vertices are a particular grid size away. It is conceptually easier to think of the surrounding vertices as being in various concentric rings of distance around the initial collision list. Figure 17 illustrates this around an initial selection of collision vertices in red.

Each surrounding vertex will then be stored in a list along with those of the same color. The goal is to have different sets of vertices, each a different grid size away from the original. With those selection groups, I can then translate each set independently based on the translation data for the particular surface being represented. It is important to note that just because two vertices are in the same set does not imply that their respective distances to the outline of the original shape is the same. The sets include vertices that are within a certain interval of the grid size away from the original selection.

Unfortunately, due to the nature of selecting a randomly shaped set of vertices on

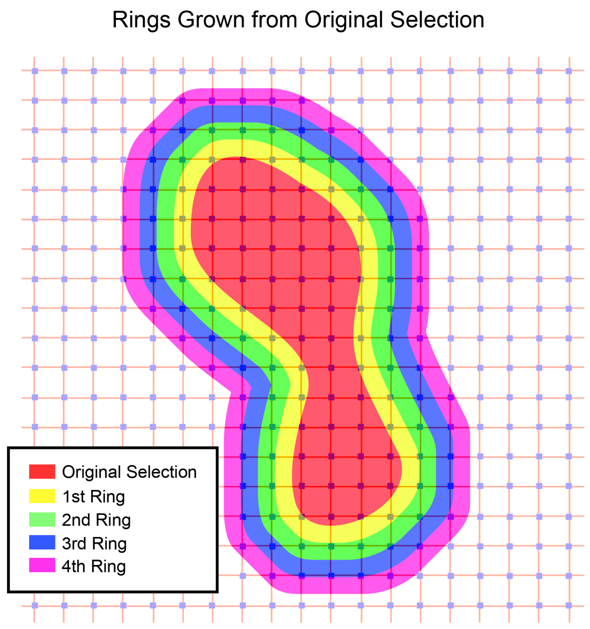


Fig. 17. Each ring of vertices is described by a different color. Vertices in each grown area will be translated according the their color.

a plane made of perfect square faces, the various rings of vertices around that shape will continually get more jagged the farther the distance. As seen in Figure 18 on the right, the largest grown ring will seem to have harsh, sharp edges composing its outline. Even though the initial shape of vertices on the left makes a smooth shape, this blockiness will always result from growing the initial region farther out.

In order to solve this blocking problem, one would need to calculate the exact distance each vertex in a particular set was away from the outlining shape of the initial collision selection. Based on that exact distance, one could interpolate linearly from the translational data to figure out an exact amount to displace the vertex. Yet again, this is a tough solution when dealing with both convex and concave selection shapes. How can we determine from which point on the outline of the selection to measure the distance? How can we determine how to weight the affects when a vertex

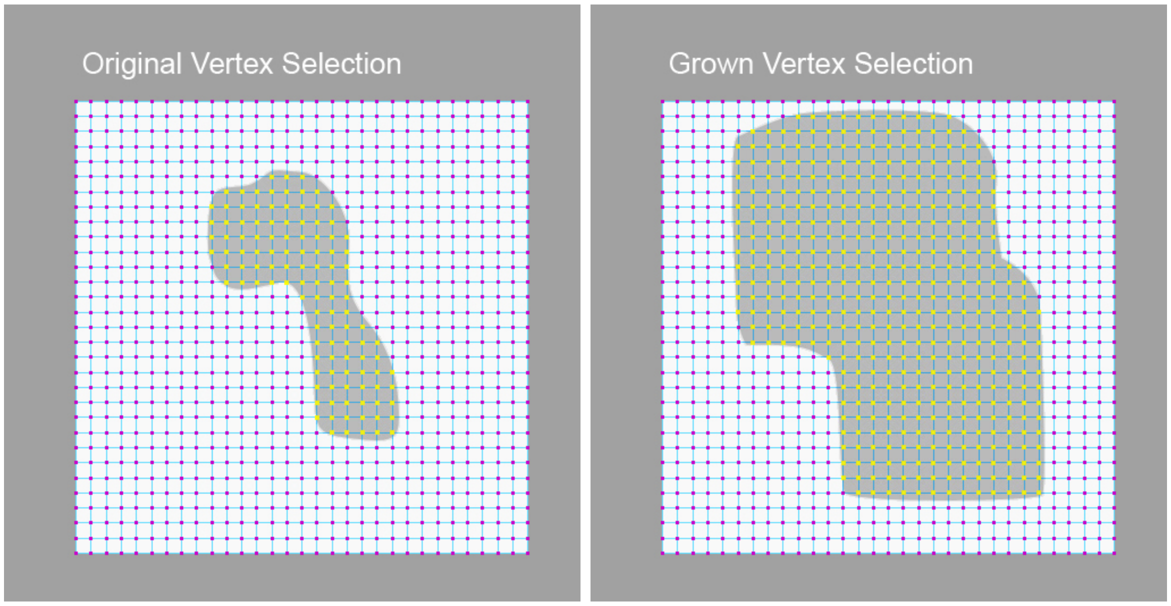


Fig. 18. Left: Original smooth selection of vertices. Right: Aliased result from growing that region multiple times.

lies within the boundaries of numerous surface edges? This is extremely problematic for concave shapes where the basic shape takes the look of the letter "V." Figure 19 illustrates this area of complication. The translation data applied to vertices in the blue area will depend on weighting the distances and the appropriate translation data from various points of the outlining shape, which is a tough solution.

Determining the actual distance to a particular vertex in a ring is shape based, so I will first treat each ring of vertices as if they are the same distance away from the original collision set. Each set of vertices will be initially translated the same distance vertically in accordance with the corresponding material translation data points. Doing so will yield a shape similar to the one below in Figure 20 which, unfortunately, results in a jaggy, blocky surface. Contrary to what is desired for most penetrable surfaces, the results are rough instead of smooth and fluid. This problematic result proves to be a valuable start to a desired solution.

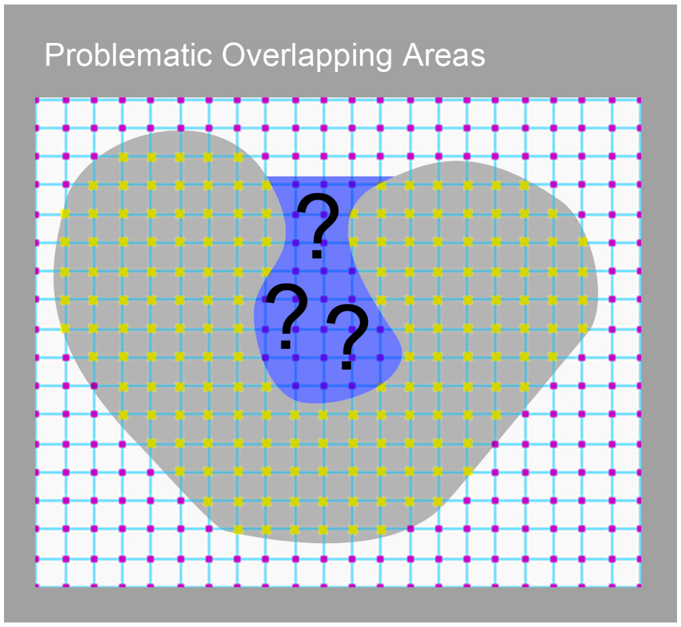


Fig. 19. In areas of concavity, it is difficult to determine how to translate the data since we have several surface points influencing the displacement.

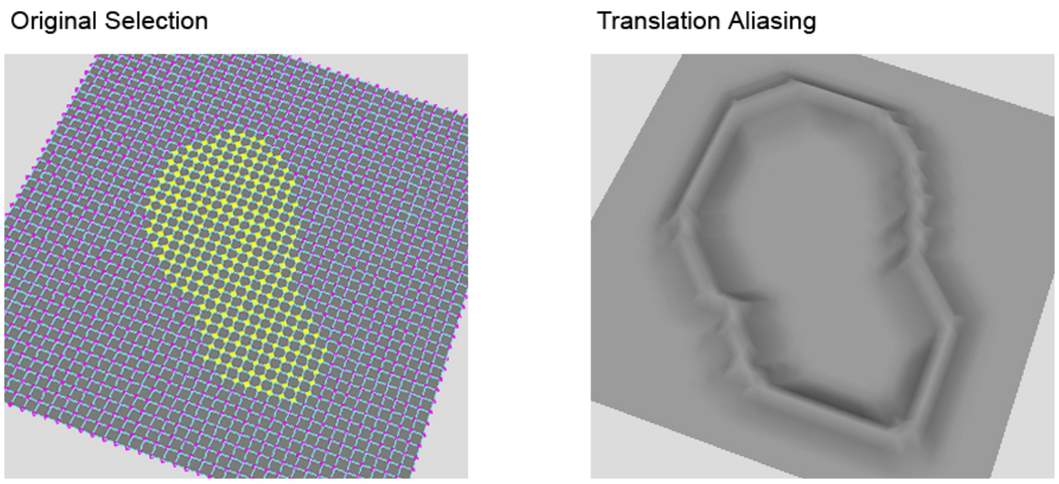


Fig. 20. Left: Original selection of colliding vertices in yellow. Right: Aliased result from applying translation data to the surrounding vertex rings.

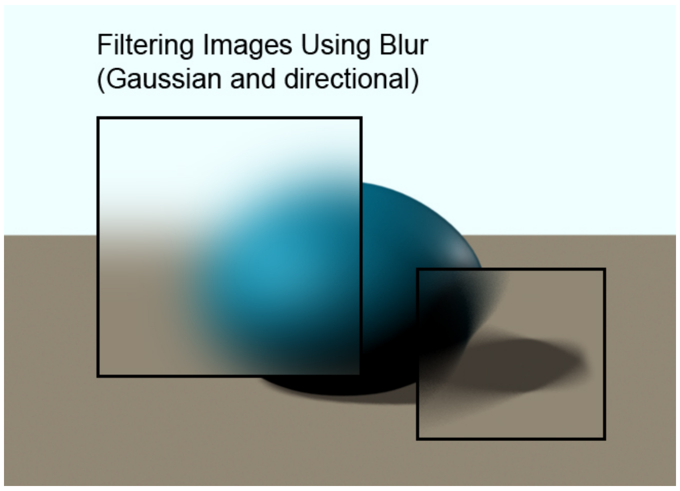


Fig. 21. Gaussian and directional blur applied in areas of a simple 2-D image.

III.2.5. Adaptive Filtering

The resulting blocky geometry serves as a building block to a much more appealing solution. It is here that we implement what we call "adaptive filtering." Adaptive filtering serves as a method of smoothing a geometric surface through the use of a variable filter technique. In the same way filters are applied to 2-D images to blur, smooth, or alter the pixel value in some way, one can use adaptive filtering to smooth the geometry of a blocky surface. An example of filtering select areas of a 2-D image is evident in Figure 21.

Adaptive filtering works as follows. For each vertex in every corresponding set, a new height value is set based on the average height of its surrounding vertices. For example, if a particular vertex has a height of 1 and the eight surrounding vertices each have a value of 0, the resulting value of the current vertex will be set to the average of those nine values. In this case, the new height for that vertex would be 1/9 instead of 1. This method is equivalent to that of a 2-D box filter. In Figure 22, one can see the difference in averaging a small region about each vertex (second image

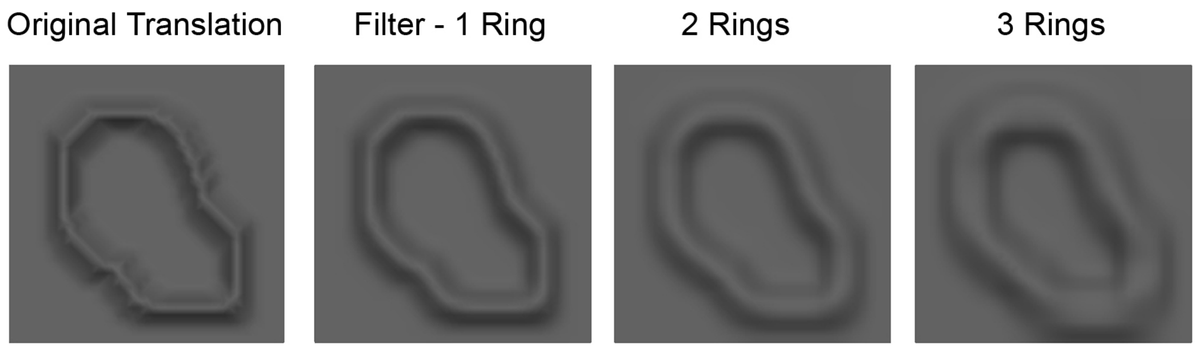


Fig. 22. Averaging the vertex heights based on surrounding vertices results in different levels of smoothing based on the filter size.

- 1 ring) and a large area around a vertex (fourth image - 3 rings). This technique adequately smooths the overall geometric structure; however, a problem still exists.

In the case of most penetrable surfaces, the further away from the collision area, the less detailed the ground shift becomes. The impact of the collision grows weaker as distance increases, causing the resulting effects to form a more subtle, rather than a distinct shift. For example, in analyzing a footprint in damp sand, one can easily see the individual toe prints in the collision area, but a few inches outside of that collision area, the shifted sand is no longer precise enough to make out the individual toe prints. This can be seen in the animal track of Figure 23.

The presence of various levels of detail is handled by the adaptive portion of this filtering method. Averaging the vertices is controlled by the set in which the vertex lies. Essentially, the further away a ring of vertices is from the initial collision set, the less complex (more smoothed) the deformed geometry will be. In order to vary the resulting area's smoothing, one can increase or decrease the number of surrounding vertices to include in calculating the new average height. Much like the rings of vertices grown about the original collision vertices, we will grow a certain number of times about the particular vertex being averaged, and according to its set, determine

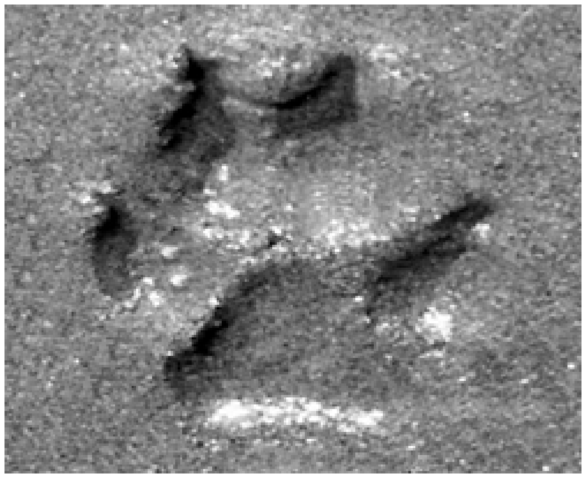


Fig. 23. The amount of detail needed to describe the shape near the collision area greatly differs from the areas further away.

surrounding heights and the resulting average. This method allows varying detail across the shifted mass and can again be fine-tuned to represent the desired material properties. Figure 24 illustrates the effect of using various filter sizes along with a result from using them adaptively as vertices increase in distance from the original selection. This maintains the detail near the original selection, and properly accounts for the area sampled between the two selections.

III.2.6. Considering Animation

We now have a method for making deformations for a single image, but we must further account for two important elements: withstanding an animation and altering the shape of the displacement to account for the interacting object's vector of motion and position.

In order for these techniques to be useful, they must be able to work over the course of an animation. Displacement with adaptive filtering helps solve the conservation of mass issue, but between any two frames the shift must blend together. With

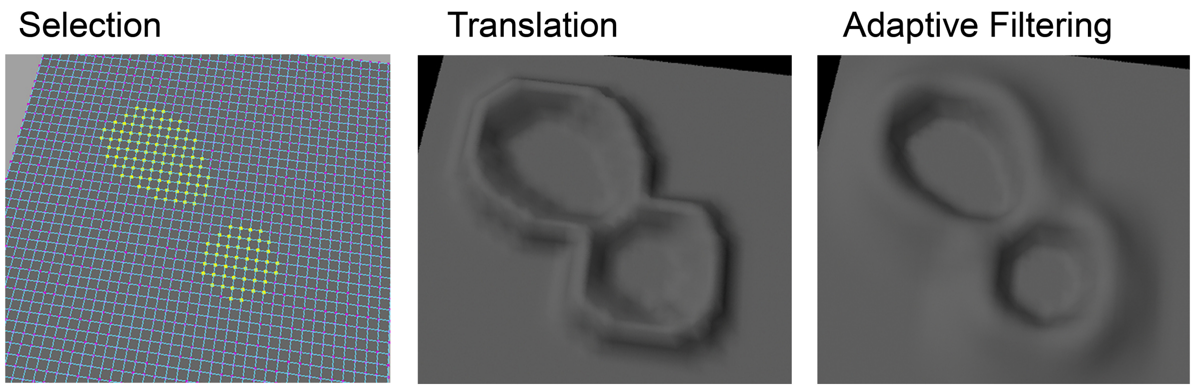


Fig. 24. A smaller sample size is used to filter the rings of vertices close to the original selection and increases as the rings size does.

the current structure, each particular collision does not know anything about the collision before it, nor does it account for the changes made prior to that particular frame. In the previous examples shown, the topmost layer is altered in all directions around the area of impact. However, if the collision object were sliding horizontally along the topmost layer over time, there is no need to alter the geometry directly behind the object since there is no force creating a change of shape in that particular direction.

In order to account for this problem, it is necessary to consider the direction of motion, just as was done in the case of altering the vertex normals of the lower layers. Only those vertices within some interval near 90 degrees of the direction of motion need to be altered. Of those vertices, another tolerance needs to be set to average the previous state of the displacement with the current stage. This creates a blend between shapes and generates a result that seems to mesh together as one collective deformable mesh from one frame to the next.

The vertices directly in front of the object's motion vector will displace the most matter. As the object's speed increases or the deeper it penetrates the surface, the higher the topmost layer will be displaced. These subtle, yet necessary, characteristics will make the layered geometric surface more believable to the viewer.

In the case of generating holes or tunnels through a surface, the colliding object's velocity and position determine the way the layers are affected. If moving upward from beneath the surface as in the case of a groundhog digging to the top layer from underground, the displacement layer must give the illusion of a shifting upward. Accounting for this will involve displacing the top layer some amount based on the distance to the topmost layer. The closer the object is to the top layer if coming from below, the displacement maximum value will increase. Once the surface level is breached, the layer will then retreat back to the traditional way of handling displacement with a hole of transparent faces in the center. Each of these animation cases must be handled appropriately and are essential for generating an aesthetic final animation sequence.

III.3. Creating an Animation

Once the layered ground is constructed and the appropriate interaction rules are set, there must be a way to display the effects generated from their use. Sample animations involving various scenarios of interaction ranging from digging, sliding, surfacing, or any combinations thereof serve as a testing ground for the functionality of the tools. Various modeling and animation software packages provide a means of animating a particular object through a surface, and their built-in renderers allow one to generate image sequences of TIFs from a particular animation. With those sequences, one can then create a movie viewable in Windows Media Player (or a similar player) and watch the animated sequence as it unfolds over time. Animation creation also serves as a test to see if the tools are working properly for a given situation. By comparing the animation samples, I will be able to fine tune the tools to achieve the desired results for a final animation.

CHAPTER IV

IMPLEMENTATION

Implementing the tools described proved to be a cyclical process, despite my hopes for linearity. In any bottom-up process, one can only hope that each stage predicts the needs of its successor. My approach followed the course most traveled, as I continued to re-visit my previous solutions to better suit the final product. The following describes the decisions I made, the successes and problems of each, and the solutions implemented to achieve my goals.

IV.1. Setting the Stage for Development

Before I began, I had to decide which software package to use for developing the tools necessary and a final animation sequence. I chose to use Alias' Maya as my foundation. Aside from having functionality for modeling and animating, Maya allows users to program scripts and plug-ins through the use of its embedded scripting language, MEL (Maya Embedded Language). Having used MEL previously and knowing its strong correlation to C programming, I opted to use it for programming the tools needed. This decision proved to be one that would yield various pros and cons throughout the development stage.

IV.2. Initial Setup

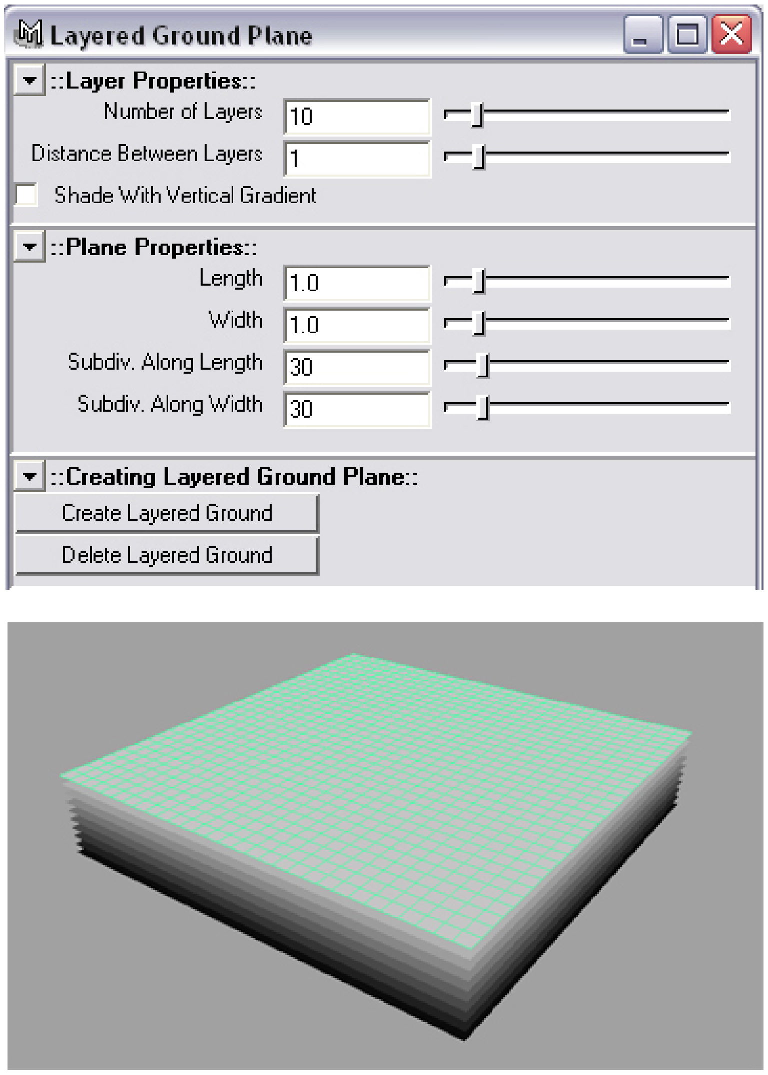
To start, I animated a simple polygon sphere to move through an inanimate, square, polygonal ground plane. The first tool needed was to create the layered geometry. A single plane obviously was no solution to having a layered ground plane, but through the use of Maya's commands for surface duplication and translation, I transformed the initial surface into various carbon copied layers, each with their own respective distance away from the initial top surface.

So, the scripting began, and through the use of MEL's built-in GUI functions, I was able to build an interactive window, allowing a user to select the makeup of the initial polygonal surface. Figure 25 is a snapshot of the menu available in creating a layered structure and a sample structure created. Options include length, width, the subdivisions along each axis, the number of layers to create, and a distance between each. Once all parameters are present, at the click of a button a layered structure is built according to specification. Below are pictures of the options available for the user when creating a layered structure and the resulting structure created when applied. I later setup a way for a user to create layered geometry from a polygonal plane they previously altered in some way.

This tool not only creates the layered structure but also serves as the basis for determining collision and its respective response. Without this clearly defined structure, moving to the next stage would be impossible. It is important to note that the lower layers of this layered surface will only be seen when a collision response occurs. Much like a grassy field, we cannot see the dirt beneath the grass unless the grass is penetrated or shifted by some force. The topmost layer in the bottom image of Figure 25 would resemble the grass, with the lower layers serving as the layers of dirt below. If we were to stand upon that top layer and look directly down, the layers below would not be visible.

IV.3. Handling Collision

Considering the geometric structure of the layers, I had to implement an efficient way for determining collision. Because the structure now has N copies of the original



Layered Ground Plane

::Layer Properties::

Number of Layers 10

Distance Between Layers 1

☐ Shade With Vertical Gradient

::Plane Properties::

Length 1.0

Width 1.0

Subdiv. Along Length 30

Subdiv. Along Width 30

::Creating Layered Ground Plane::

Fig. 25. A user can alter any of the sliders available to generate a variety of layered structures.

structure and (N)*(number of faces in a single layer) faces to handle for collision, coming up with an efficient way to determine which faces are in collision with the sphere at a particular time frame was extremely important.

The key to doing this is understanding that a sphere is basically a point in space, with an area of surface surrounding it no greater than some radial length away. It is important to throw away as much extraneous data possible to reduce the amount of calculations, just as a ray tracer does not consider objects hidden behind others in relation to the camera upon render time. If the distance from the sphere's center to a particular layer's position along the y-axis is greater than the radius of the sphere, no faces in that layer ought to be considered for collision, and it can be thrown out of the calculation. In the case where the distance is less than or equal to the radius of the sphere, we know that a collision does in fact occur at that particular time-step. However, it would be too costly to check every single face for collision.

The calculation data can be further reduced by finding the closest face to the center of the sphere, and checking for collision within some region of faces surrounding it. To generate this checklist of faces, I used one of Maya's most handy functions that grows the selection region. This built-in functionality served as a helper for many calculations in developing the interaction rules. Figure 26 provides a visual representation of this idea. If, for instance, the sphere's radius is roughly two times the width of a face, then we know that the largest cross section of intersection with that sphere could be four faces wide. Since we already have the closest face in collision, we grow that selection of faces twice to increase the area of collision to be slightly larger than that of the sphere. We then know that, for all the faces in that particular plane, it is impossible for any other face outside that grown selection to be in collision with the sphere. Thus, for any particular layer, we have already calculated the closest face in collision, and have a reduced selection of faces to check for collision.

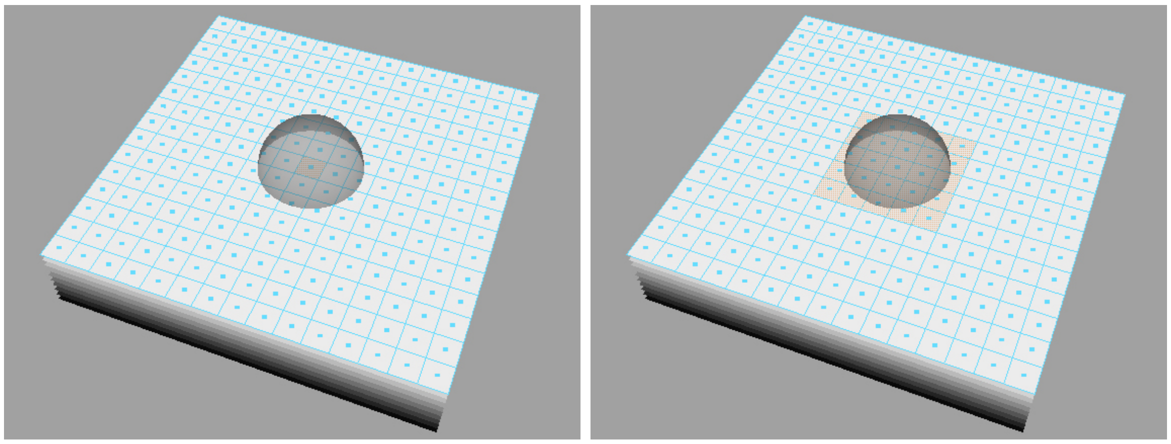


Fig. 26. Left: Selection represents the closest face in collision. Right: Selection represents a more concise set of faces that could be in collision.

For a particular face, if the distance from its center to the center of the sphere is less than or equal to that of the current radius, the face is in collision and some effect must occur. Using the ray-triangle intersection method determines collision areas with complex objects, allowing this to be implemented for a variety of different objects as well. If the collision face's center is within the boundary of the object, it will be included in the final collision set.

In determining collision areas using MEL, calculation time for a particular frame grew exponentially when the geometry composed of many faces. I suppose the reason for this is that MEL has to access each face in a long array of elements, scanning through each to find the proper one for each and every calculation. Therefore, the bigger the array (higher the resolution), the longer the calculation time. I knew I would work with some fairly heavy geometry for the animation, so I had to optimize this process.

For the interaction of one specific object with a surface, we know at any given time, its maximum size. It is therefore wasteful calculation to test faces for collision if outside some region encompassing its maximum width. By using a piece of geometry the size of its maximum width and translating it to the closest face in collision, I was able to do collision testing on the smaller test piece of geometry, and translate the resulting collision actions to the high resolution data. Using this intermediary for collision testing, I was able to alleviate a large amount of overhead in calculation time.

Once collision testing worked efficiently enough for my purposes, I had to achieve a desired deformation effect, requiring further additions to the development tool.

IV.4. Denting the Surface

Within each layer's collision area, something must be done to illustrate the effect of the material either getting compressed or removed. Since I am not moving any of the geometric structure to achieve this effect, the shading properties of the individual layers will be important.

The first thing to do in areas of collision is to shade that particular face with a completely transparent shader, giving the impression that a piece of actual geometry has been removed. Transparency plays an important role in creating the illusion of a removal of the geometric structure without actually doing so. If I were to delete the faces in collision, the plane's geometric structure would change, making it impossible to reconsider that area for future interactions. Figure 27 shows the result of shading collision faces to be completely transparent. The layers seem to have faces removed, yet due to the layer's distance apart, it looks as if the layers don't blend together as one piece of geometry. Furthermore, I have shaded each layer with a gradient in Figure 27, making the lower layers darker than the ones directly above it. If I were to shade all of the layers with the same white shader, as I will eventually, the removal of mass is not visible since the way light interacts with the surface has not changed.

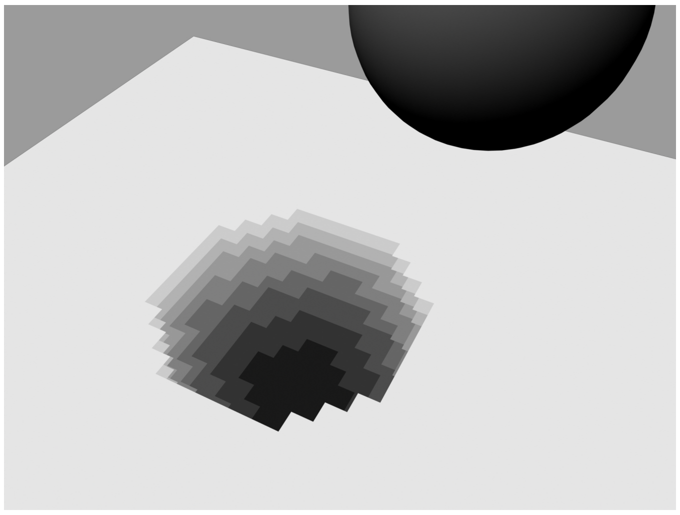


Fig. 27. Hole created by shading the collision faces of each layer to be transparent.

For this, I changed how each layer reacts to the light, by altering the vertex normals for the vertices in collision. Altering the vertex normals also improved the cohesion between the multiple layers. Maya provides a useful function that converts a selection of faces to vertices. With that new vertex selection it is important to only calculate new normals for the vertices that will be visible. Of the transparent faces in collision, only the outermost ring of vertices will be seen after the faces are set to be transparent. Shrinking the region of vertices once and removing that selection list from the original collision list of vertices further reduced the computation by throwing out extraneous calculations. As described in the methodology section, the vertex normals in the outermost ring of the collision area of a particular layer are altered to point toward the sphere's center, or the object's closest surface point. After doing this for each layer in collision, the appearance of the layers as one piece of geometry was improved but not perfectly. The result is illustrated in Figure 28.

Due to each layer's vertical separation to the next, one can see the faces surrounding the region of collision from most camera angles. Their vertex normals still

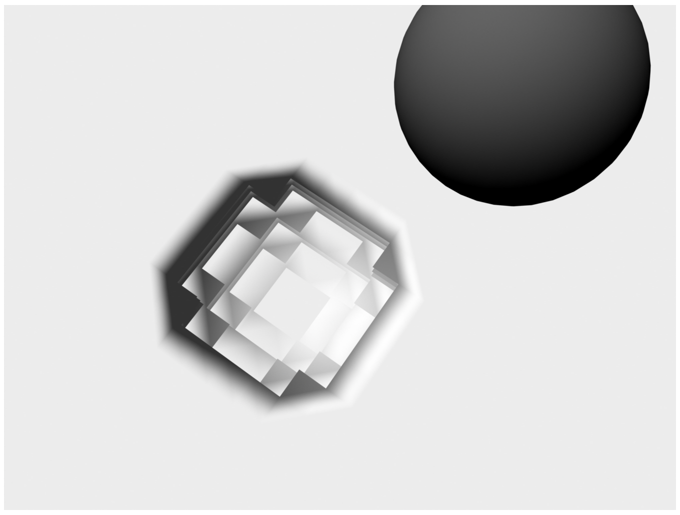


Fig. 28. Vertices in collision are altered to match the impression of the shape.

point in the direction of the original surface, upward, creating the triangular dark edges seen in Figure 28. This presence when looking between any two layers again prevents the two from meshing together as one surface as well as we had hoped.

To solve this problem, I also altered the vertex normals directly surrounding the vertices in collision to point toward the sphere's center. This required more than double the vertex calculation for each layer, however, the resulting image in Figure 29 merged together far better than any solution before. The top layer's dark ring still posed a problem, but I will address that later, when handling displacement.

IV.5. Conserving the Shifted Mass

The results so far illustrate the removal of mass, so we must now account for the shifted mass. As mentioned in my methodology, doing so relied on making alterations to the topmost layer of the structure. Mass moved below must go somewhere, and when not being compressed, it is most likely shifted upward somewhere within the structure.

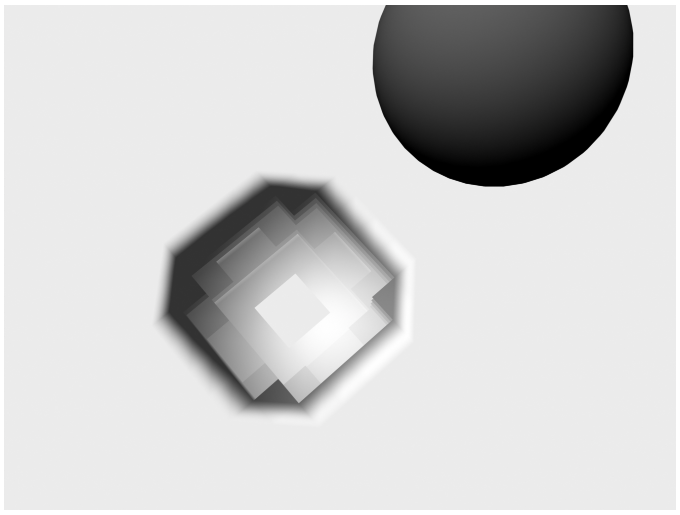


Fig. 29. Surrounding ring's vertex normals are altered to better blend from layer to layer.

IV.5.1. Vertex Normal Alteration

As a first attempt to represent this shifting of mass, I extended the method of altering the vertex normals. Since the shifting occurs roughly the same all around the center of the collision, I altered the vertex normals of the top layer based on their distance away from the sphere's center. Given two key vertex normals to interpolate between based on distance, I altered each of the vertex's normals within a certain grown region around the collision area of the topmost layer. As with the lower layers, the faces in collision for the top layer were set to transparent, yet the vertex for changing was far greater than that of any lower layer. Figure 30 shows the effected normals assigned for the topmost layer's vertices surrounding the collision.

The results from this method yielded a nice illusion of conserving the mass shifted from below (Figure 31, left), however, when looked at from practically any angle other than from directly above, it was quite noticeable that the geometry remained unaltered, resulting in an extremely flat look (Figure 31, right). As a bump map does

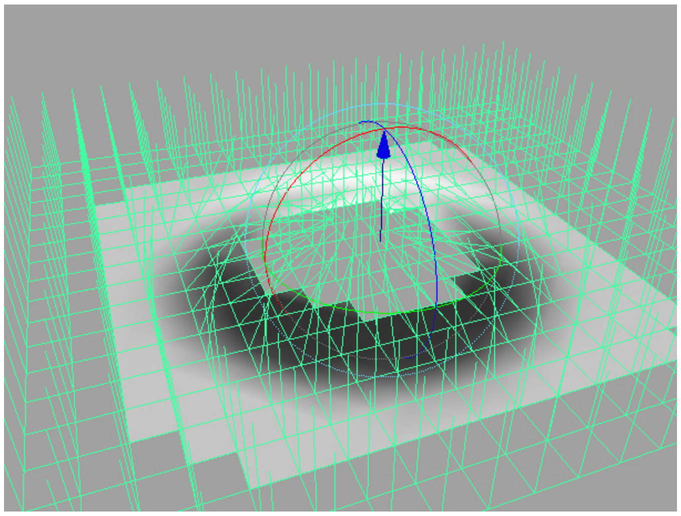


Fig. 30. Vertex normals are interpolated between key normals to create the effect of shifted mass.

not change the surface position of an object, this method prevented any shadows from being cast from the top of the surface to the layers below and gave no vertical variation on the surface.

IV.5.2. Transparent Planes Above

My initial solution was to develop a way to show the shifted mass while maintaining the layered structure. Since the geometry needed to physically represent a shift of mass, I implemented within the layered geometry creation tool a way to create layers above the topmost layer, assigning each to be completely transparent prior to collision. The idea here was to incorporate layers above the surface level that would, in a sense, “turn on” in certain areas to illustrate the shift. By shading particular faces within the transparent layers as collisions occurred, I avoided ever moving any geometry. Layers above the top were shaded opaque in regions surrounding the area of collision. Creating this effect involved selecting a number of rings grown concentrically around the faces of collision. That set of faces was then shaded opaque for the first transparent

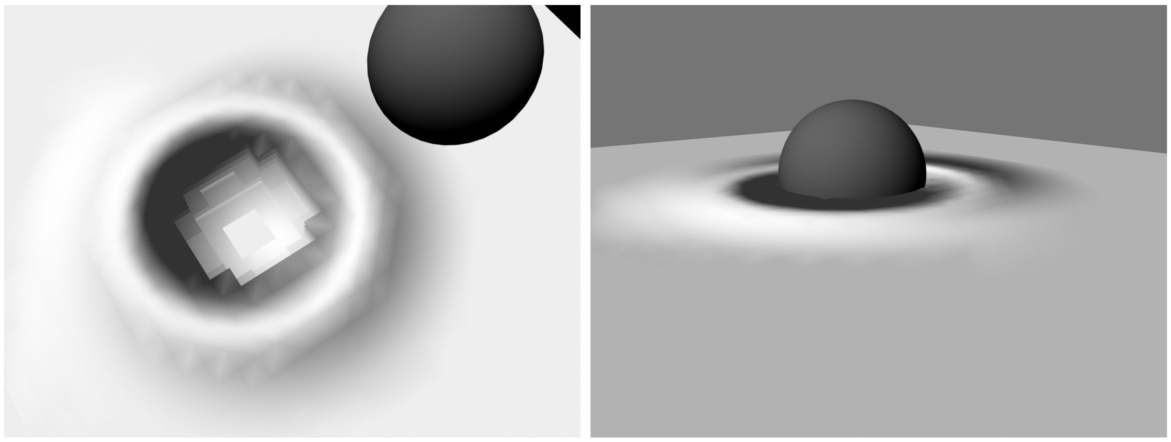


Fig. 31. Left: Top view of the result of altering the vertex normals. Right: Flaw is obviously the flatness when viewed from side angles.

layer above the original topmost layer. Shrinking that selection of faces and shading the next immediate transparent layer's faces continued until the face selection was empty. As in the case of the original topmost layer, the vertex normals in each of the faces shaded in the transparent layers were altered to give the illusion of unity.

This new solution allowed me to conserve mass and maintain geometry in areas of shifting, however, it posed bigger problems. The selection of faces grown about the initial collision faces of the topmost layer tended to result as an extremely blocky version of the original, due to the nature of geometry composed of square faces. Again, when looked at from any angle other than directly above, the semi-shaded transparent layers above seemed to clearly be separate planes of geometry. With the inability to view the shape from any angle from the side, I was back to the same problem resulting from solely changing the normals of the topmost layer. It was considerably more noticeable than the layers below that the overall structure lacked the sense of being one geometric structure. Figure 32 illustrates these problems and limitations.

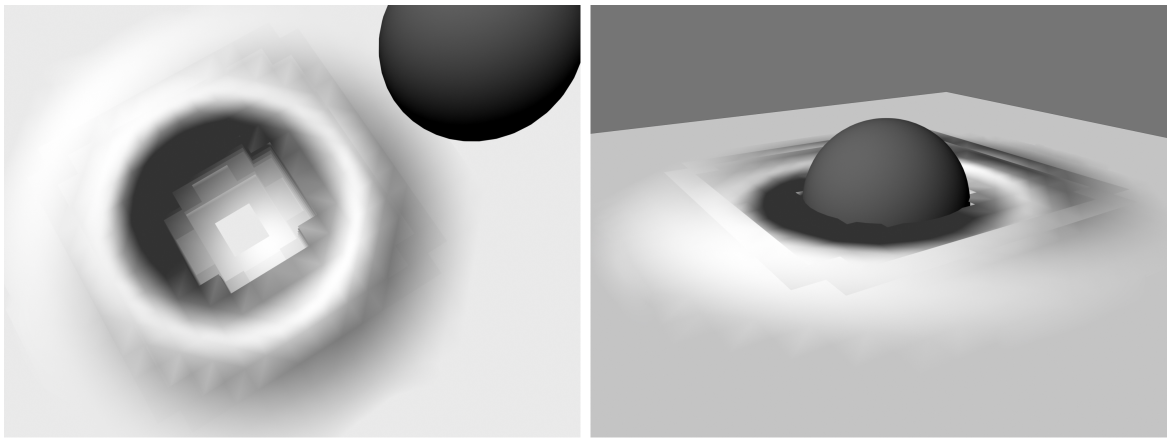


Fig. 32. Left: Top view of the transparent layers in use. Right: Flaw continues to be the flatness when viewed from side angles.

IV.5.3. Surface Level Displacement

Yet again, I was back to square one in creating the illusion of mass conservation without actually moving the geometry. Layered geometry had already shown its benefits by allowing an object to move through another, removing the mass, but its structure was extremely limiting in creating effects above the surface level. As in most scenarios, one idea alone cannot be a solution to all problems, so I began pondering the benefits and potential problems of using displacement solely on the topmost layer of geometry to create the illusion of shifting mass.

As mentioned prior, the only layer that actually shows where the mass shifts when depressed or moved is the topmost layer, given you are looking from above the top surface level, which we will be. The layers below represent matter compression or the absence of mass, while the top is responsible for illustrating how and where it is moved. To prevent seeing a gap in the overall structure, the topmost layer's collision faces will be to always set the height to equal the elevation of the plane's y-value. As long as the camera is never able to see the area labeled green in Figure 33, it could

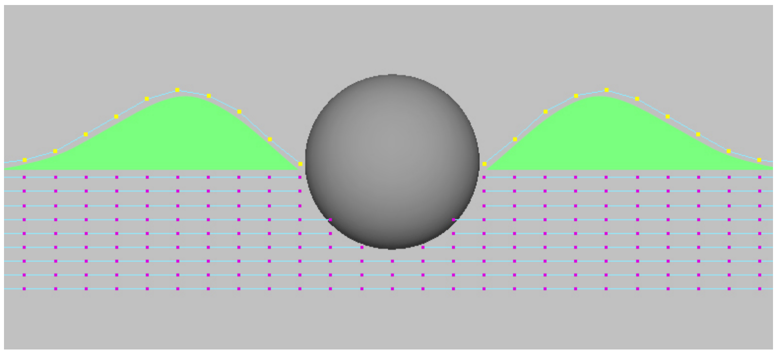


Fig. 33. Area under the displaced top layer is not viewable.

be a valuable solution. Displacement of the top layer seemed to be an easy method to implement, however, trouble lay ahead.

Creating a smoothly displaced surface required new solutions. As described in the methodology section, the shape deformed around an object directly correlates to the shape of the collision area. The collision area for the top layer is already determined, so altering the vertex heights involves selecting grown rings of vertices about that initial selection and translating those rings or sets to a particular height based on the user's predefined translation data. Each of these vertex sets is determined by growing the size of the previous selection and de-selecting the vertices in that previous set. This allows concentric sets of vertices to be created about the initial collision selection, with each being further spread out than the last. The translation data was then applied to the corresponding set of vertices to move the group a specified amount in the y direction. Doing so for every ring resulted in a blocky structure formed about the original selection, colored orange in Figure 34.

At this stage in the process, I realized this method of displacing the surface layer would not suffice if there were not some way to smooth the deformation created. By implementing adaptive filtering across the various grown sets of vertices, I was able to do just that. Adaptive filtering relies solely on averaging the translation data of

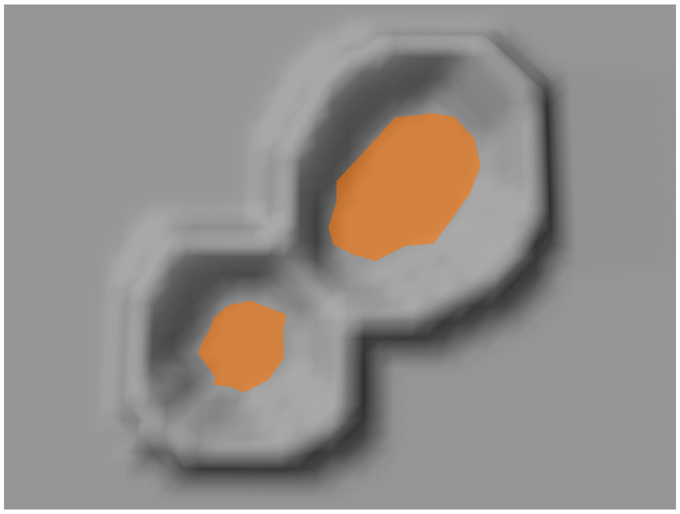


Fig. 34. Aliased top layer displacement generated using translation data.

the vertices' height values surrounding a particular vertex along with that of its own, as described in the methodology section. By revisiting each vertex in its respective ring, I was able to take a weighted average of the specified region around it.

The filter area for a particular vertex depends on what set that vertex is in. The sampling area is determined by selecting the vertex and growing the selection of vertices any number of times corresponding to the predetermined filter value for its ring. If more detail is desired, a smaller selection is used. If less is desired, a larger selection is used. In Figure 35, the vertex on the right nears the original collision and since detail is needed to help define the shifted shape, the filter area consists of itself and one surrounding ring of vertices. However, if filtering a vertex in the outermost ring, like the vertex on the left of Figure 35, it is filtered with a larger surrounding area of vertices to better blend into the surface. The resulting average height value is then assigned to the vertex that started the selection.

Using Maya's built-in selection tools for growing the vertex selection size made it much easier for me to implement adaptive filtering as a tool. In general, adaptive

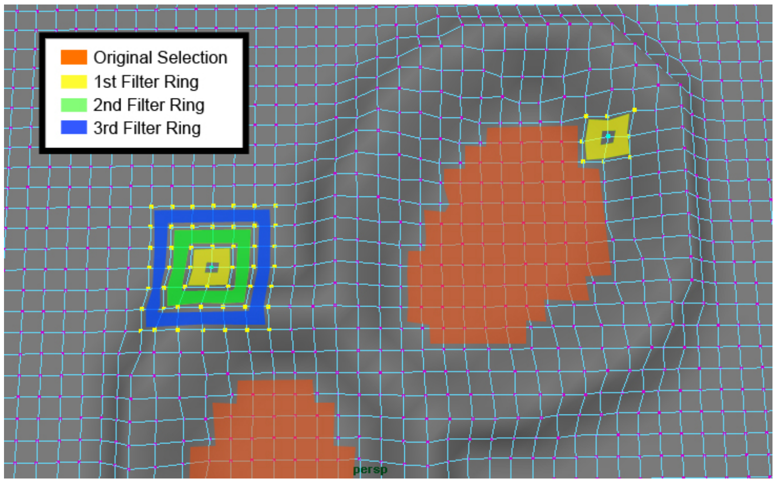
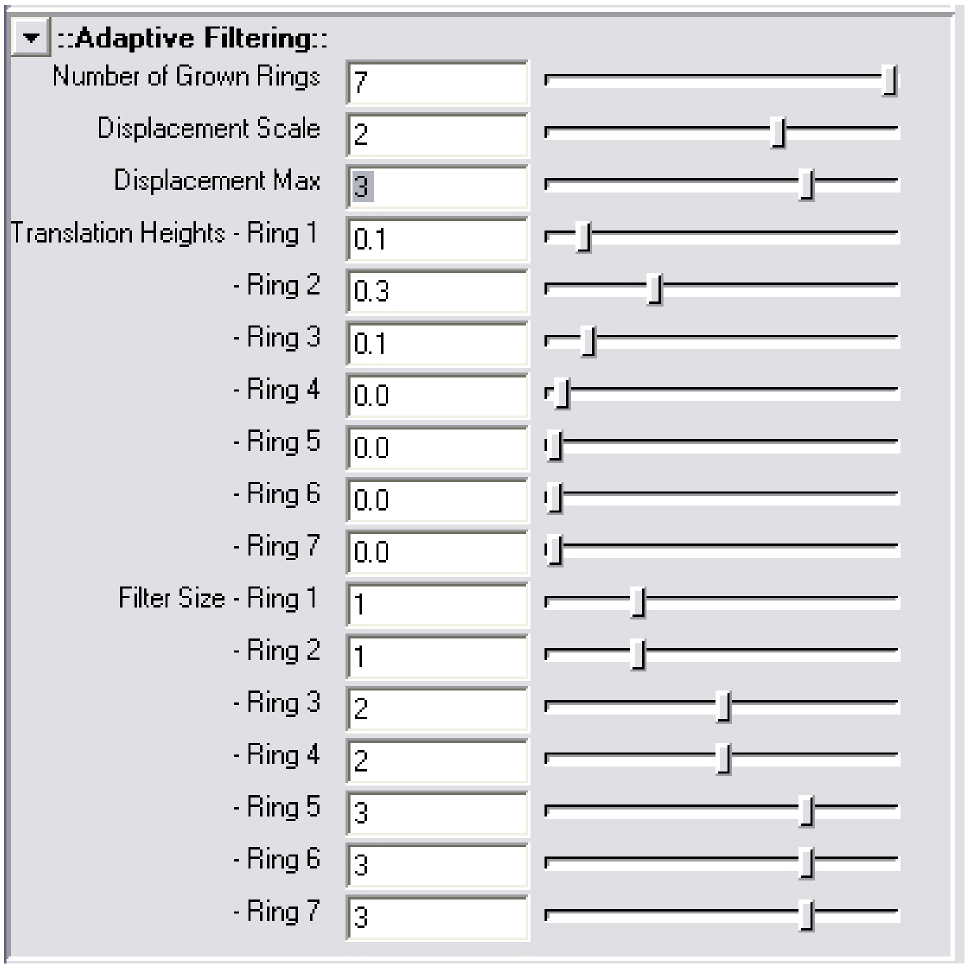


Fig. 35. Depending on the distance away from the original selection, different filter sizes are used.

filtering creates a smoothed region far away from the collision area and a detailed region near by. Of course, the user is able to specify the translation data and its respective filter size upon runtime to create a variety of shapes and effects. All results can then be tweaked for specific results by redefining the parameters for filter width and translation data for each set of vertices. The interface for using adaptive filtering is illustrated in Figure 36. Handling vertices outlining the collision selection, it is still important to set the vertex normal according to the intersecting face of the interacting object, as done when simply altering the vertex normals of the topmost layer. This maintains the blending effect between the displaced layer and the layers beneath.

IV.6. Handling Animation

With adaptive filtering enabled for the displacement of the topmost layer and layered geometry accounting for the penetration within the surface, the tools developed work well for a single image. Generating a single image well does not ensure its success



::Adaptive Filtering::

Number of Grown Rings	7
Displacement Scale	2
Displacement Max	3
Translation Heights - Ring 1	0.1
- Ring 2	0.3
- Ring 3	0.1
- Ring 4	0.0
- Ring 5	0.0
- Ring 6	0.0
- Ring 7	0.0
Filter Size - Ring 1	1
- Ring 2	1
- Ring 3	2
- Ring 4	2
- Ring 5	3
- Ring 6	3
- Ring 7	3

Fig. 36. Graphical user interface for adaptive filtering.

in creating a working animated sequence. To do so, rules and alterations must be implemented to account for the various interaction scenarios that could potentially occur in an animation.

Potential situations include an object penetrating the surface from above, pushing upward on the surface from below, and countless ways of sliding through the surface in any primarily horizontal direction. Each of these test cases required a different implementation of how the surface level is deformed. The key to determining which case applied is calculating the direction of motion of the colliding object along with the speed at which it is moving.

Animation in Maya relies on positional data captured at particular moments in time, or key frames. These frames of time serve as a method for determining the changes in a scene from one moment to the next. For a particular time frame when an object is interacting with another, it is important to gather as much data possible about each object. The velocity and position of the colliding object are the crucial factors for determining the interaction response for a particular time frame. The position of an object in Maya is always available by reading its coordinates in world space. Calculating the velocity of that object at that time frame requires backing up a time-step, getting the position, and subtracting that positional value from the current position. This calculation gave us the direction of motion along with the distance traveled in a single time-step, which was used as a way to measure its speed.

IV.6.1. Motion Downward

The methods described in the prior examples of displacement are sufficient for implementing the case where the motion of the colliding object is directed downward or into the surface. If the angle between the object's direction and the penetrable

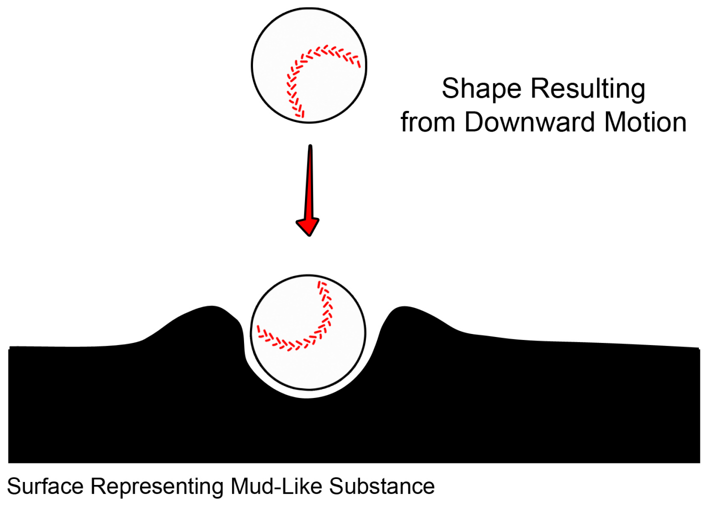


Fig. 37. Displacement generated from downward motion.

surface layer's normal is within the interval of 45 to 90 degrees, the surface will generally displace matter equally around the collision area. I calculated the angle by determining the dot product of the motion vector with the vector perpendicular to the average normal of the topmost layer. If within this interval of degrees, the result will tend to resemble the image in Figure 37. The position of the colliding object in relation to the topmost layer determines how far to displace the vertices. The further down the object moves, the higher the ground is displaced to account for the extra movement of mass. There is a maximum displacement value that can be set by the user to prevent the surface from displacing too far. In Figures 38 and 39, one can see the results when using layered geometry for this type of interaction.

Often, the object's motion is not perpendicular to the surface as the previous example. When entering at an angle from above, an overlapping shape is created, fulfilling the requirements that displacement maps cannot. Figures 40, 41, and 42 illustrate the expectations and results from angular downward motion.

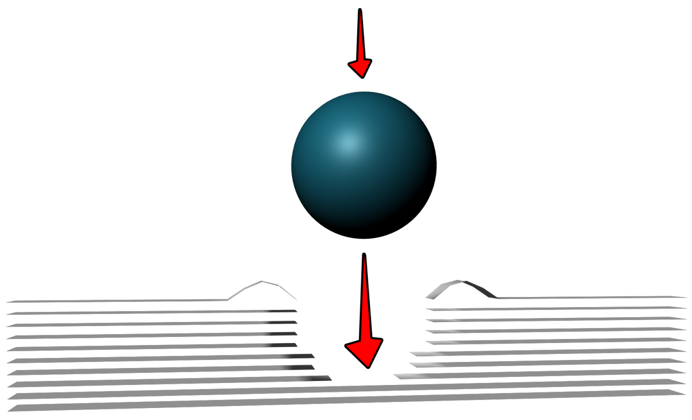


Fig. 38. Slice of layered geometry resulting from downward motion.

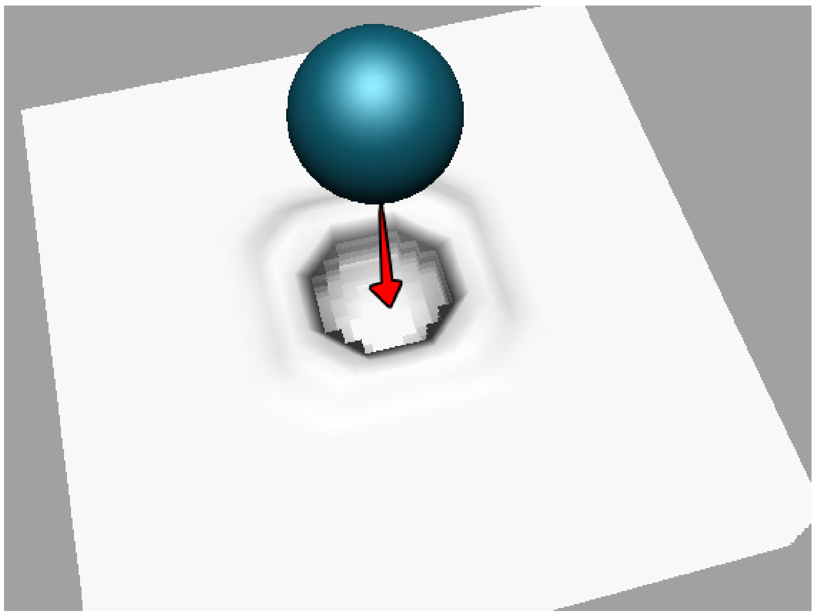


Fig. 39. Layered geometry resulting from downward motion.

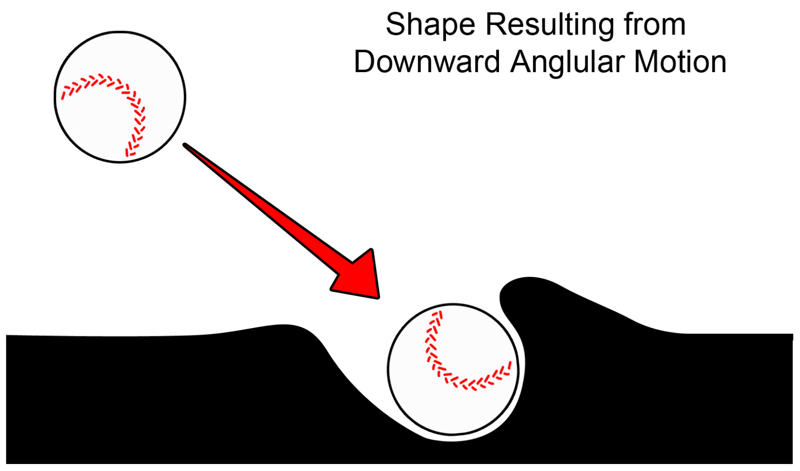


Fig. 40. Displacement generated from motion downward at an angle.

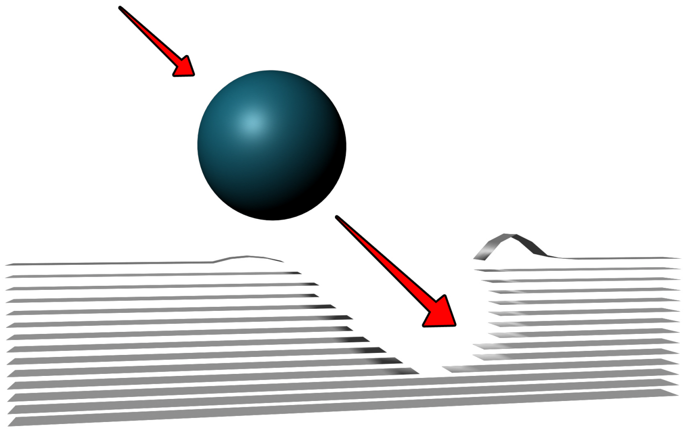


Fig. 41. Slice of layered geometry resulting from motion downward at angle.

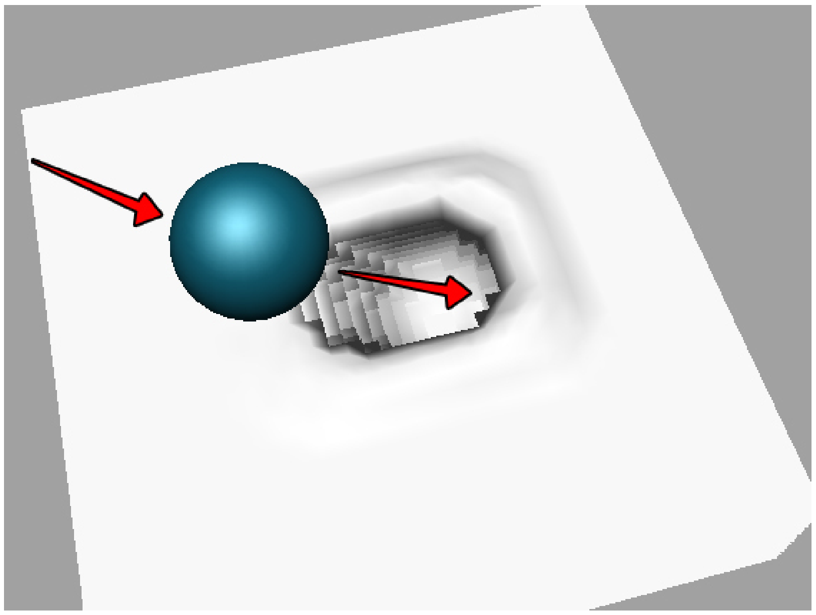


Fig. 42. Layered geometry resulting from motion downward at angle.

IV.6.2. Motion Side to Side

Lateral motion requires extra calculation to properly determine the resulting effect. When sliding a ball sideways in the sand, for instance, the area in the direction opposite the motion is not altered. Only the area within a region of 90 degrees of the motion vector will be altered, having the areas further away displacing less and less. For displacing a particular vertex, if its vector created from the center of the colliding object is greater than 90 degrees from the direction of motion, it will not be altered. In the area where the vertices are not in the strength of the direction of motion, only a percentage of the displacement value is used, averaging that portion of the height with the value from the previous frames, if there were any. Doing so allows for the blending of displacement animation from one frame to the next, rather than constantly displacing all vertices the exact amount for the current frame. Figure 43 gives an example of lateral motion, while Figures 44 and 45 illustrate the results from

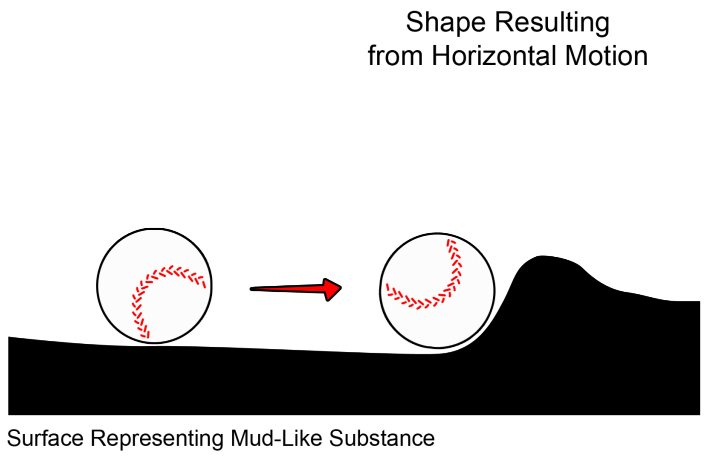

Shape Resulting from Horizontal Motion
Surface Representing Mud-Like Substance

Fig. 43. Displacement generated from horizontal motion.

this interaction using layered geometry.

IV.6.3. Motion Upward

For motion from below the top surface moving upward, yet another consideration must be accounted for. The displacement does not occur around a collision area, because the collision area is not yet defined, due to the collision having not yet occurred for the topmost layer. In the case of continual motion upward from beneath the top surface layer, the displacement must reflect the shape of the object coming from below. According to its position beneath the surface and the collision area from a lower layer, the shape is then applied to the topmost layer as an area of displacement, translating the vertices in the center the most and those on the edges the least. This displacement uses adaptive filtering as well to smooth the resulting geometry. Of course, as the object gets closer to surfacing, the displacement height gradually increases until the maximum displacement for that particular material is

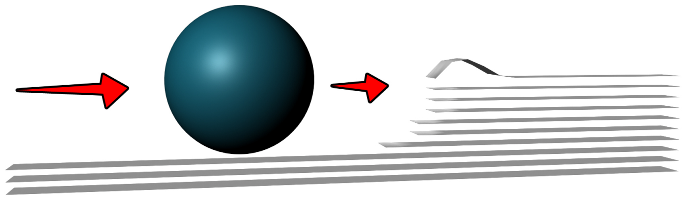


Fig. 44. Slice of layered geometry resulting from horizontal motion.

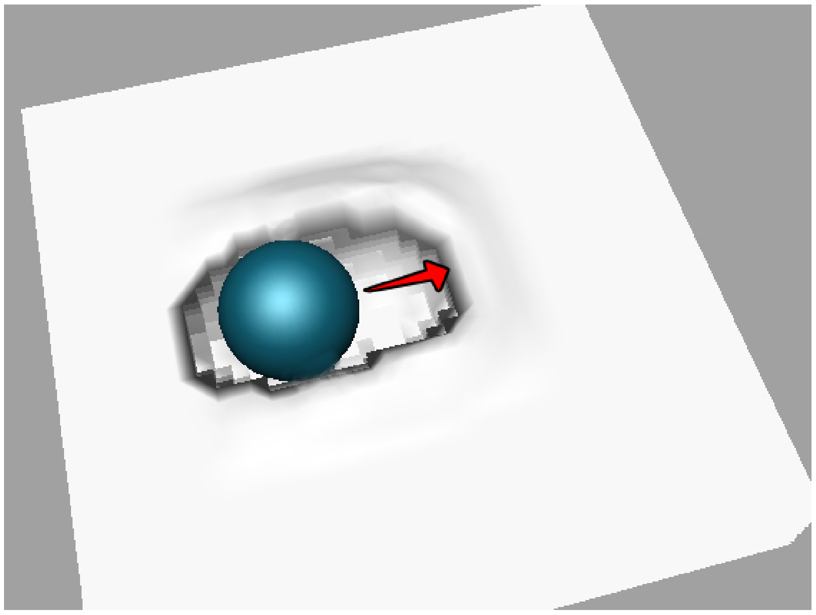


Fig. 45. Layered geometry resulting from horizontal motion.

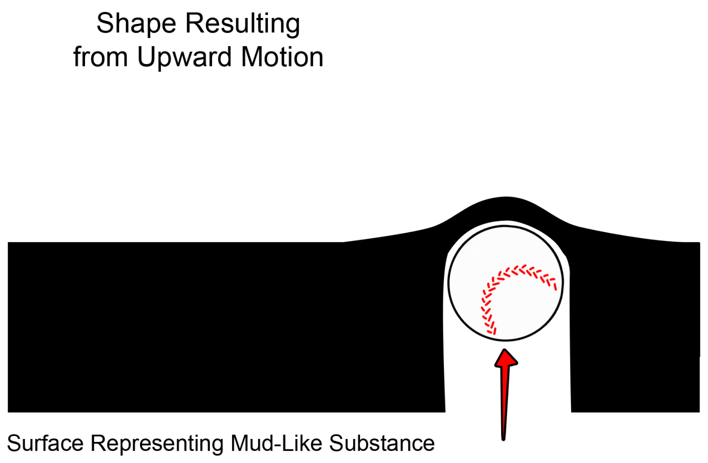

Shape Resulting from Upward Motion
Surface Representing Mud-Like Substance

Fig. 46. Displacement generated from upward motion.

met. Figure 46 illustrates the expected effect of upward motion, prior to breaching the top surface. When the object has penetrated through the topmost layer, that frame resorts back to calculating displacement in the case of downward motion, leaving a hole as the object exits the surface. Figures 47 and 48 display the results from this interaction type using layered geometry.

IV.6.4. Tunneling

When moving through a layered structure in an animation, a combination of interaction methods occur. Digging a hole through a surface generally requires downward, angular, horizontal, and upward motion. Each resulting in different effects as seen in the previous examples. Figures 49 and 50 illustrates a combination of all of the aforementioned interaction methods.

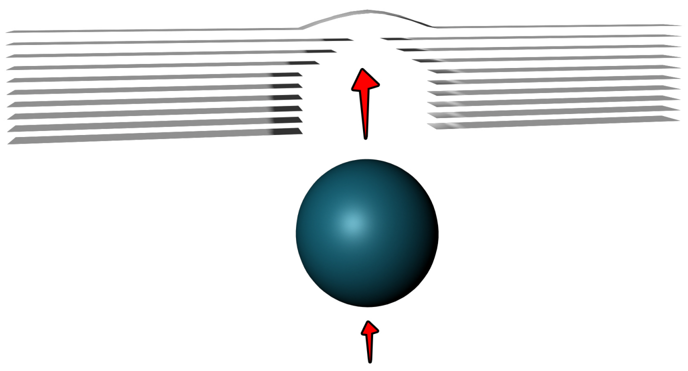


Fig. 47. Slice of layered geometry resulting from upward motion.

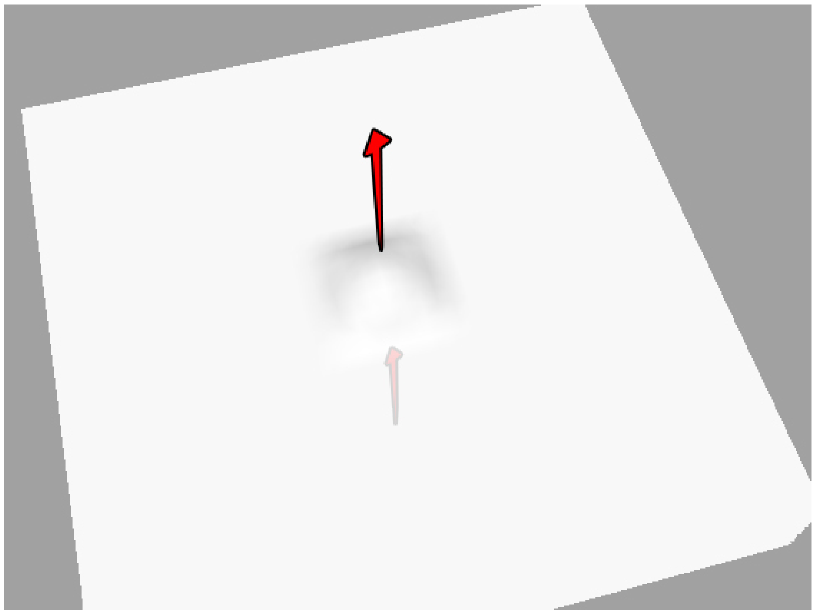


Fig. 48. Layered geometry resulting from upward motion.

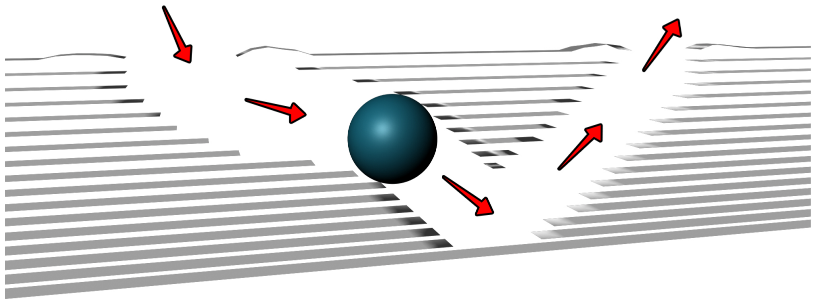


Fig. 49. Slice of layered geometry resulting from tunneling.

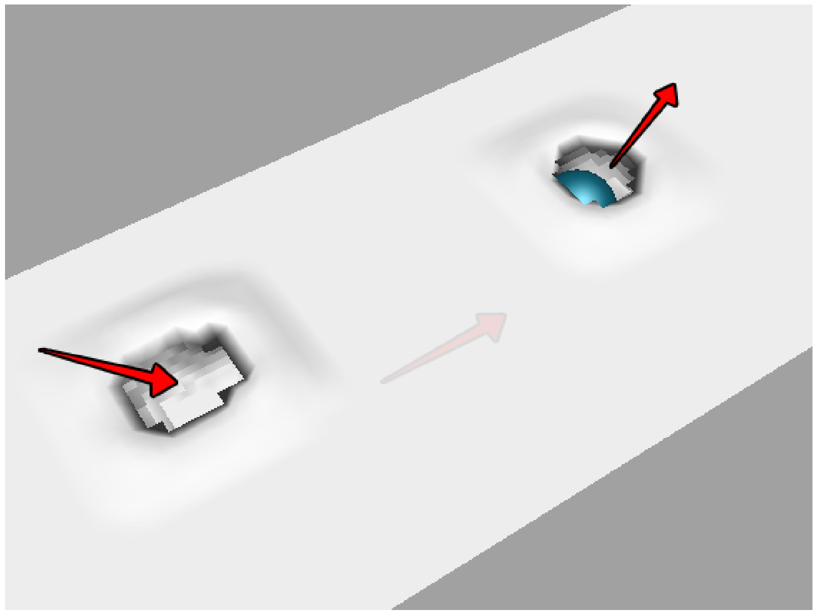


Fig. 50. Layered geometry resulting from tunneling.

IV.7. Creating a Final Animation

Once all of the rules for interaction were set and the tools for creating and deforming the layered geometry worked according to specific test scenarios, a final animation was created to illustrate the results. I wanted to ensure the animation demonstrated all of the test scenarios properly, however, I did not want the final render to seem as if it were a mere demo for the various interactions. It was more important to me that I select a subject matter that would in fact do those things in its natural environment, making the need for such tools relevant. My hope was to create an animation whose interaction with an environment made complete visual sense to the viewer, not distracting from the animation itself but adding to the realism of the actions.

In order to do this, I decided to animate an inch worm moving through a soft mud surface. His interaction with the mud includes sliding along the surface, digging beneath, and surfacing again elsewhere. With each of these actions animated and working together as one sequence of shots, I hoped to generate an effect that would be difficult to create using traditional methods of surface deformation and would enhance the animation for a necessary purpose, not merely for eye candy.

After rendering the animation as a sequence of TIFs, I composited the rendered images from Maya together to make a movie viewable in Windows Media Player. Through the use Adobe's compositing software, After Effects, I was able to generate an AVI file of the animation sequence, viewable as a movie at thirty frames per second.

CHAPTER V

RESULTS

The following section is devoted to showing still frames from my final animation. I have included stills of consecutive frames from various types of interaction to illustrate the different calculations and effects resulting from each.

My final animated short created is titled “Xing.” A brief summary of the plot involves the painstaking journey of a determined inchworm, tiring with every movement. When confronted with a detour around a twig crossing his path, he becomes disgruntled with the additional distance added to his trip. In a clever attempt to beat the system, he takes a deep breath and digs a hole beneath the twig to reach the other side. Delighted with his ingenuity, he continues his journey.

I was careful to develop a story that would require all of the interaction methods developed within my tools, however, as stated before, my overall hope was to create an animation whose story would be accented by the visual effects of surface deformation, rather than having deformation be the one and only focus.

In the opening sequence, the inchworm moves slow and steady along through the ground’s soft mud. The interaction, in this case, involves sliding along horizontally through the mud’s surface. The surface structure, unlit and unshaded, can be seen in Figure 51. The final rendered effect can be seen below in the image sequence of Figures 52 and 53.

Shortly after, he decides to try his luck by digging beneath the ground. The interaction in this case involves digging downward through an object. Again, the rough structure and the resulting effect can be seen below in Figure 54 and the image sequence of Figure 55 respectively.

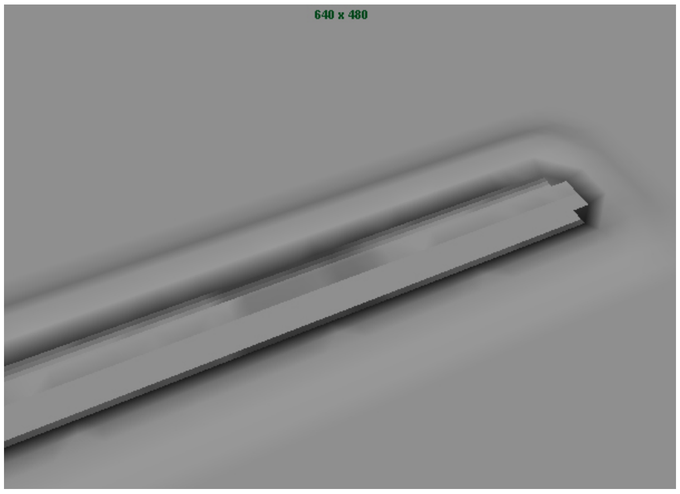


Fig. 51. Layered structure resulting from sliding.

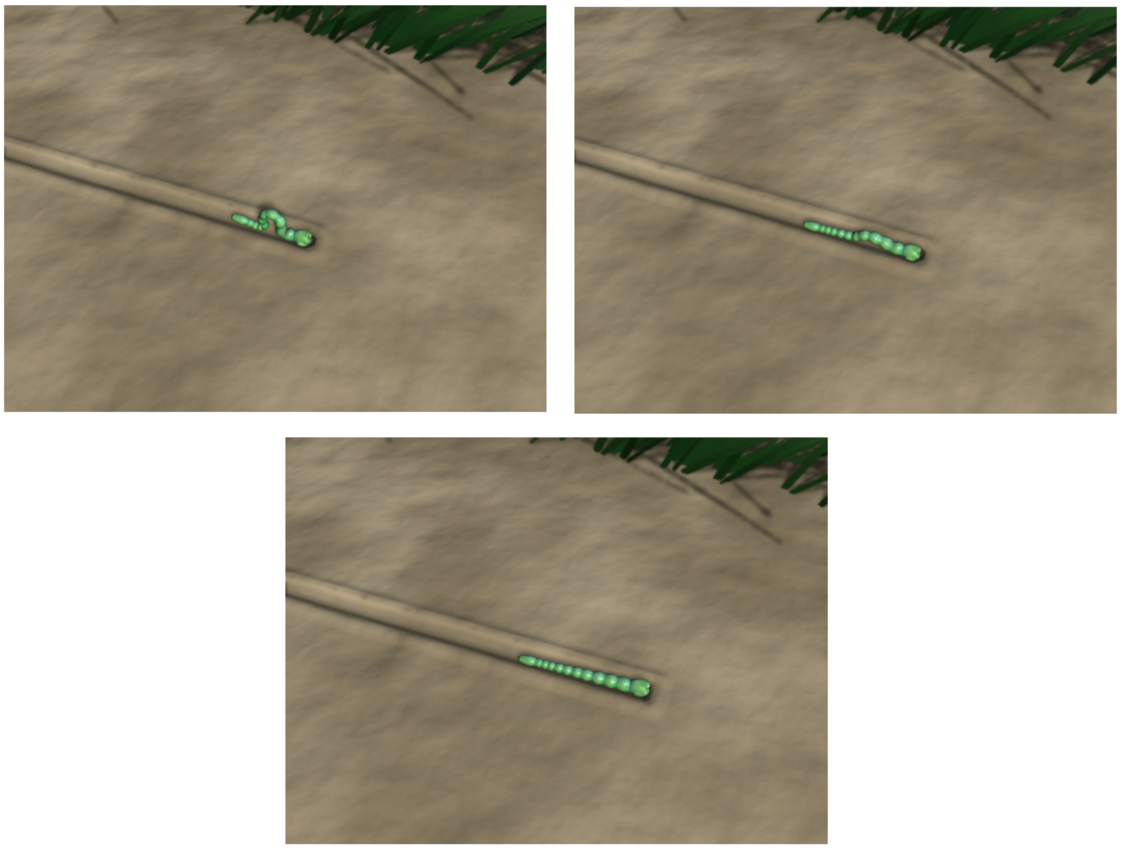


Fig. 52. Shot 10: Resulting images from horizontally sliding.

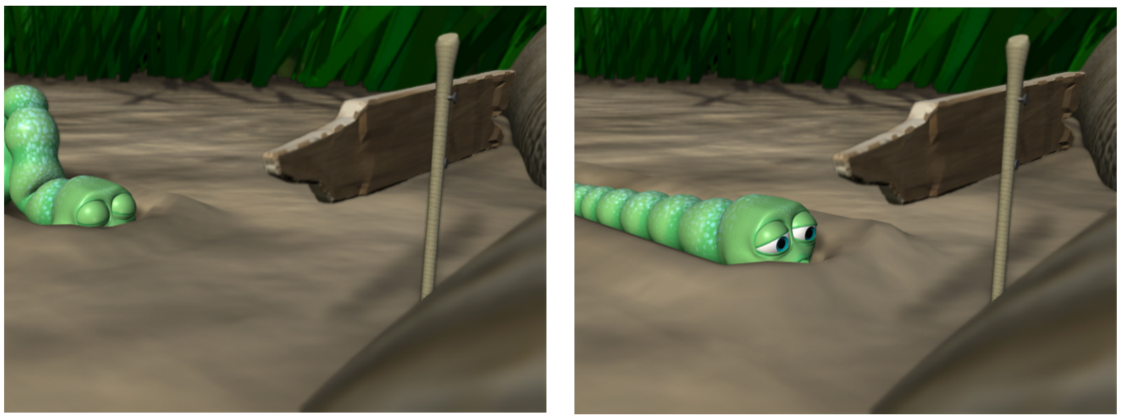


Fig. 53. Shot 20: Resulting images from horizontally sliding.

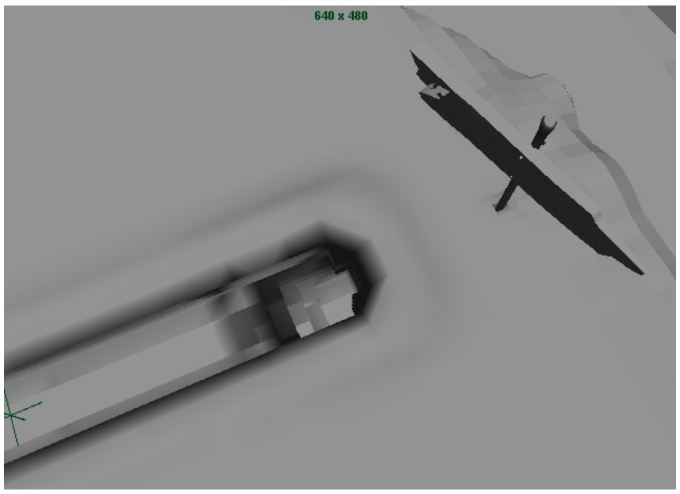


Fig. 54. Layered structure resulting from digging.

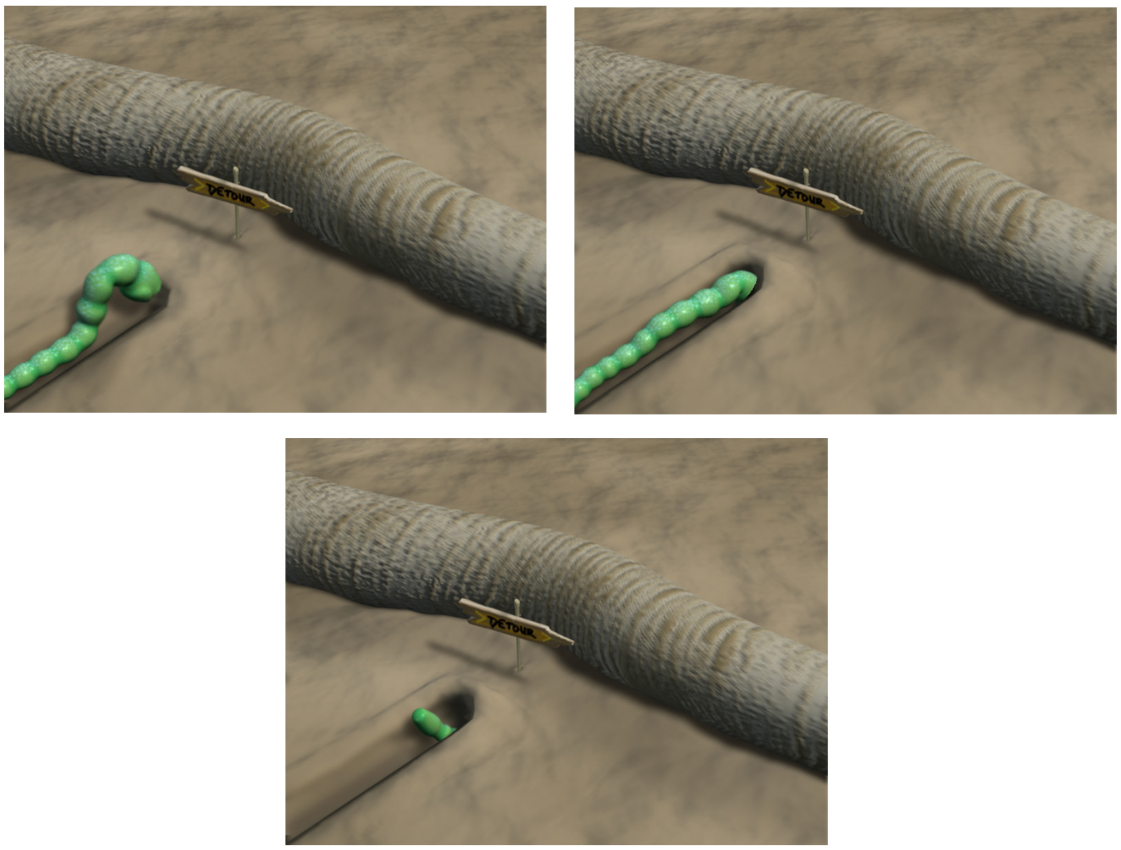


Fig. 55. Shot 60-1: Resulting images from downward digging.

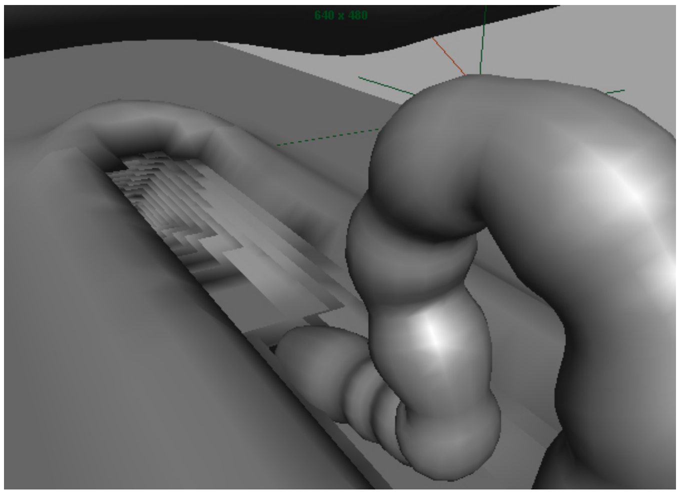


Fig. 56. Layered structure resulting from surfacing.

Once past the twig's area, he begins to approach the mud's topmost layer, surfacing from below. Figure 56 and Figure 57 contain sample stills from this type of interaction.

The final rendered images are much more aesthetically interesting than those of the rough surfaces, due to the lighting and shading of the various props and elements. I was pleasantly surprised with the blending between the various layer, as the structure seems to exist as one consistent piece of geometry. The results from this short reflect on the successes of the use of the tools developed. I define each interaction type clearly within the animation, yet, the story prevents the viewer from focusing on its demonstration of the various scenarios. With these final renders, I am better able to assess my goals and draw conclusions.

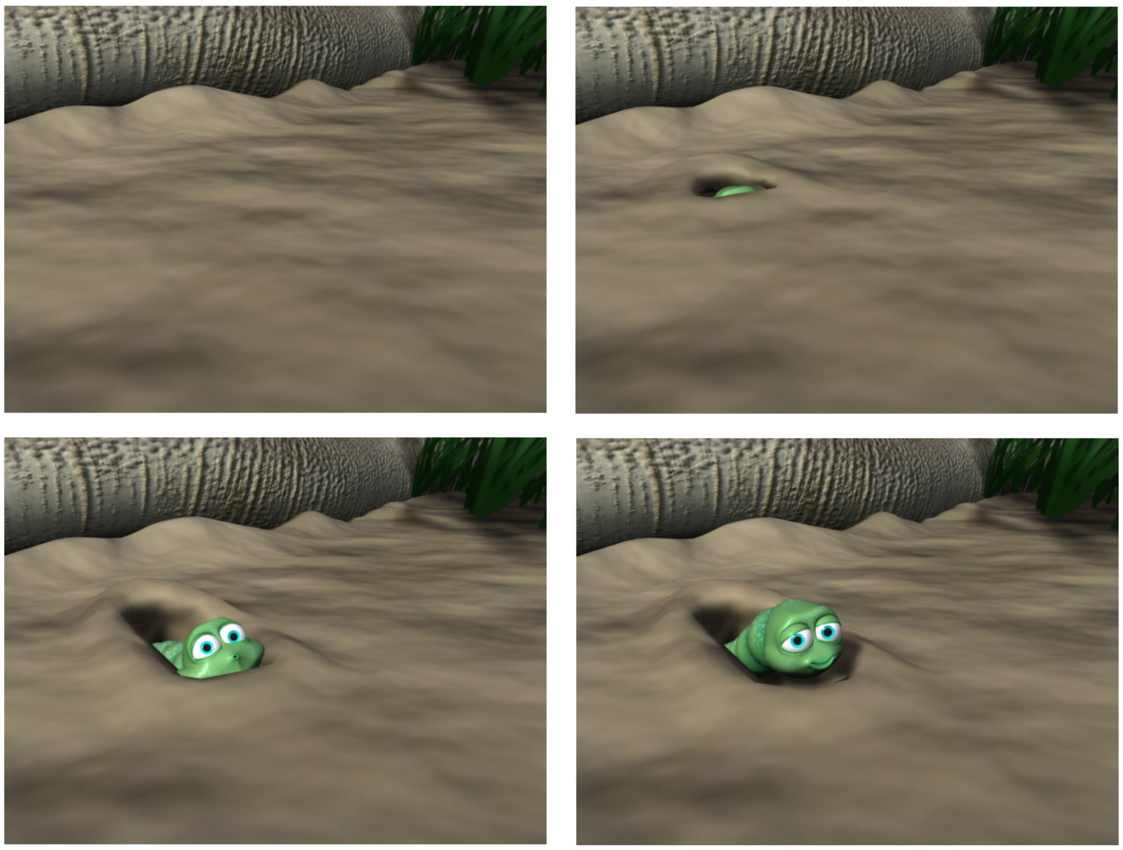


Fig. 57. Shot 60-2: Resulting images from vertically surfacing.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

VI.1. Conclusions

Given the images shown in the results section, I was able to demonstrate an automatic method for animation-dependent, geometric deformation that handles multi-directional movement, while conserving mass. The inherent benefit of treating a surface as a volume of layers is that the layered geometry never changes. Thus, the geometry never breaks, allowing the creation of interesting overlapping effects resulting from digging, dragging, surfacing, or making holes in a surface. The MEL scripts developed allow a user to create a specific layered ground plane and, with an animation of a particular object, create a sequence of images illustrating its motion and effects from interaction with a soft, penetrable surface in Maya. I attempted to give the user the most control possible to fine tune the deformation effects for a desired ground material. The overall results proved it to be a usable tool for creating complex deformations in a surface, and I feel that I successful met my initial goals.

As for the results of the process itself, there are a few things I might try differently, given the chance to do it all again. The rendered images illustrate the output from using my developed tools, but generating those images from a computing standpoint could stand improvement. As mentioned early on, selecting MEL as the coding language to build this tool led to effective results, however its innate flaws, unknown to me at the time, prevent its ease of use.

The main downfall of using MEL as a tool builder was its inability to release memory. After computing calculations for a particular time frame, the memory stays active and is never released back to the computer's operating system. Upon calling the tools' functions a second time, almost an equivalent amount of memory is allocated again, rather than reusing that which was allocated previously. No matter if I flushed the variables or history, the memory continued to remain inaccessible and un-releasable. Heavy calculations generally resulted in Maya crashing, restarting the file from where it left off, and continuing the render process.

Furthermore, there are a few additions that would make this tool even more useful. Currently, the layered structure changes from frame to frame, never allowing a user to see what the geometry looked like at a previous frame. Making the changes keyable from one frame to the next would benefit the user by allowing them to scrub back and forth through the animation to see the effects in each frame. This would also allow post modifications to the surface on a frame basis. As with any production, no tool works perfectly for every single shot, and often, computer graphics artists must manipulate certain frames of an animation that might not look exactly as desired. At their current state, my tools would not allow this fine-tuning step.

Also, there is currently nothing implemented to refine the translated geometry of the topmost layer. Thus, as the vertices are displaced, the resolution of that layer reduces, due to the greater distance between any face's translated vertices. Not accounting for this problem can result in areas of the displaced layer becoming blocky and rigid if displaced too far, when the goal is to have a smooth geometric surface. Implementing a subdivision technique to refine these stretched areas would alleviate this problem.

The final concern for my implementation occurs when traversing back through old paths. Currently, crossing an old path will not shift any matter into the old transparent area. Once the area is transparent, I have no method for re-shading it back to opaque when interacted with a second time. Implementing a way to revisit transparent sections of the structure would allow for more complex animations.

It is important to note, however, that no one technique will ever be a perfect solution for each and every situation. Many circumstances pose a strong need for a different way of handling what is difficult to do with common tools. In the situation of creating animated deformations and holes through a surface, layered geometry and displacement with adaptive filtering serves as a viable solution.

VI.2. Future Work

As in most cases, future work is always a possibility given time to act. Never is any one solution perfect in all of its methods and calculations. There is always room for improvement.

In my case, the biggest push would be to improve the calculation speed, whether it be coding in another language or reducing the number of calculations done each time step. Speed can always be improved given time to research bottlenecks in current algorithms. For instance, it would be beneficial to implement a more resourceful way of calculating collision, determining the closest collision point using iterative subdivision schemes on the geometry, rather than looping through every single geometric face. One could begin treating each layer as a single face initially, subdividing that face, calculating which of the new faces is closest to the object, and repeating until some subdivision level is met. This would reduce the unnecessary collision calculations immensely. Also, re-working the code for Maya's API environment could be beneficial in both calculation time and memory usage, or implementing the same tools in a coding language such as C++ with OpenGL could allow for more complex and lengthy animations. Unfortunately, strictly coding in C++ requires the user to setup a whole slew of other tools needed to model, animate, and render easily.

Another area of work for this technique would be to handle the interaction of the penetrated surface with itself after another object interacts with it. For instance, if some animal digs a hole beneath a surface and exits, depending on the material structure, the matter might shift to fill the hole where he used to be. In representing sand, this shifting is largely important for illustrating its granular nature as an object enters or leaves. Much of the time, sand will actually cover the geometry of the object until it exits. Generating this effect would allow users to represent a wider range of material properties.

Regarding the actual animations generated, the content is bound only by the user's imagination, leaving new ideas and effects wide open to further development.

REFERENCES

[1] Blue Sky Studios, “Ice Age,” Motion Picture: March 2002.

[2] P. Fearing, “The Computer Modeling of Fallen Snow,” Ph.D. Thesis, Dept. of Computer Science, University of British Columbia, July 2000.

[3] S. Frisken, R. Perry, A. Rockwood, and T. Jones, “Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics,” *Proc. ACM SIGGRAPH 2000,* pp. 249-254, 2000.

[4] M. Moore, and J. Wilhelms, “Collision Detection and Response for Computer Animation,” *Proc. ACM SIGGRAPH 1988,* pp. 289-298, 1988.

[5] J. Peng, D. Kristjansson, and D. Zorin, “Interactive Modeling of Topologically Complex Geometric Detail,” *Proc. ACM SIGGRAPH 2004,* pp. 635-643, 2004.

[6] J. Stam, “Stable Fluids,” *Proc. ACM SIGGRAPH 1999,* pp. 121-128, 1999.

[7] R. Sumner, J. O’Brien, and J. Hodgins, “Animating Sand, Mud, and Snow,” *Computer Graphics Forum,* vol. 18, no. 1, pp. 17-26, 1999.

VITA

Jacob Brooks
227 Bonner
New Braunfels, TX 78130
jakebrooks@tamu.edu

Education

M.S. in Visualization Sciences	Texas A&M University, August 2005
B.S. in Computer Science	Texas A&M University, May 2002

Employment

Graduate Assistant, Non-Teaching	Texas A&M University, August 2002 - May 2005