Inflatable Icons: Diffusion-based Interactive Extrusion of 2D Images into 3D Models

Alexander Repenning
AgentSheets Inc.
4565 Gunpark Drive
Boulder, CO 80301 USA
+1 303 530 1773
alexander@agentsheets.com

Abstract

There are many applications, such as rapid prototyping, simulations and presentations, where non-professional computer end-users could benefit from the ability to create simple 3D models. Existing tools are geared towards the creation of production quality 3D models by professional users with sufficient background, time and motivation to overcome steep learning curves. Inflatable Icons combine diffusion-based image extrusion with a number of interactively controllable parameters to allow end-users to efficiently design 3D objects. Early user testing has indicated that it takes end-users, even kids, just minutes to draw icons and then use the Inflatable Icons approach to create 3D models which they can use to build video games.

Keywords: diffusion-based image extrusion, interactive user interfaces. ACM CCS descriptors: I.3.3 (Picture/Image Generation), I.3.4 (Graphics Utilities, paint systems, graphics editors), I.3.5 (Computational Geometry and Object Modeling).

INTRODUCTION

While 3D content has certainly managed to become accessible in most households in form of game consoles and personal computers, there are few tools that allow non-expert users to quickly create simple 3D shapes. Nowadays *end-user development* is quickly advancing [11] and end-users employ all kinds of authoring tools such as word processors and slide presentation tools to author their own content. 3D content is lagging behind not because of a hardware challenge – on the contrary, even the cheapest PCs now feature amazing 3D rendering capabilities – but because 3D authoring tools are mostly geared towards professional developers with proper training, time and motivation.

Inflatable Icons is the name of a new technique that can interactively extrude 2D pixel-based images into polygon-based 3D models with surprisingly little input required by users. The general idea is that through the use of an inflation process suitable 2D icon artwork can serve as input for an interactive 2D to 3D transformation process.

This makes Inflatable Icons useful for a number of end-user applications including presentation software and 3D sketching. However, our immediate application domain is the use of Inflatable Icons in multi-agent simulation authoring tools employed in education such as AgentSheets [12] or StarLogo [15]. With these tools end-users, typically kids, create complex simulations involving hundreds and even thousands of agents. These agents are visualized as icons draw by kids. Inflatable Icons add new affordances to simulations. For instance, simulation worlds no longer have to adopt a birds-eye, top down perspective. Instead, a camera can be placed anywhere into a simulation world featuring arbitrary orientation including first and third person perspectives. 3D can disambiguate the spatial relationships between objects. For instance, in AgentSheets agents can be stacked on top of each other. A vase agent can be on top of a table agent, which, in turn, can be on top of a floor agent. In a 2D orthogonal top-down view this often becomes extremely confusing to users.

This paper presets Inflatable Icons as a technique that is based on an extrusion process that adds a third dimension to a two dimensional image. In its most simplistic form this extrusion process can be compared to the inflation of a balloon. An icon defines the basic shape of the balloon as well as the image painted onto the balloon. A circular shaped balloon, for instance, will be extruded into a sphere. This paper first introduces a simple inflation mechanism and then gradually refines this mechanism by adding user controllable parameters. Users can interactively steer the extrusion process through these parameters in order to create intricate 3D objects. The specific innovation presented here is the combination of a highly extensible *biased diffusion* technique controlled by 2D images, with an interactive, user-controllable process.

The quality of a 3D model resulting from the technique presented here would not likely satisfy a professional 3D model builder but is more than sufficient for this kind of application. The focus of this work is to allow end-users to very quickly create simulation worlds containing large numbers of simple 3D objects. In a multi agent simulation application it is more important for users to be able to visualize the spatial relationships between agents then to have a high quality 3D representation of any individual agent.

Polygonizing Icons

An Inflatable Icon, like a regular icon, includes a two-dimensional pixel array. Each pixel is represented as a RGBA color. For our first example, we use only a one-bit alpha value: a pixel is either fully visible or invisible. Figure 1 shows a "Lobster" icon, which will be turned into an Inflatable Icon.

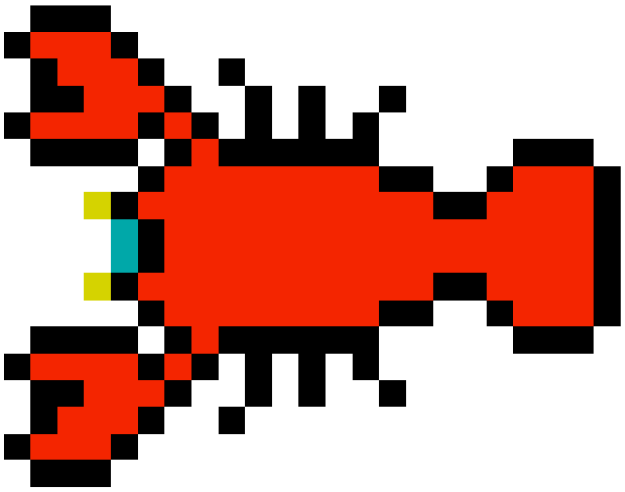

Figure 1: A low-resolution icon ready to be inflated.

The Inflatable Icon has a $z_{x,y}$ value for each x, y coordinate. These z values are initially set to zero. Unlike regular icons, an Inflatable Icon is a true 3D object in the sense that it is represented as 3D polygons. A simple but somewhat brute force approach – in terms of number of polygons required – is to use triangle strips and represent each pixel as a square consisting of two triangles. As described below, triangle strips work especially well when rendering objects with smooth shading.

As a convention, we define the upper left corner of each square to be the reference point representing the z value of a pixel. Since all the z values are initialized to zero we get a planar set of polygons (Figure 2).

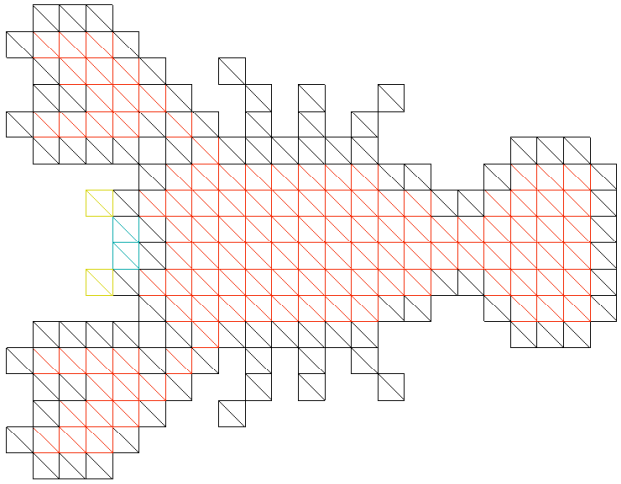

Figure 2: Each pixel is polygonized with two triangle polygons.

Diffusion-based Image Inflation

There are a number of ways of implementing an inflation process. In Teddy [8], for instance, the system uses a skeleton based approach [1] to inflate closed regions with amounts depending on the width of the regions. Wide areas become fat and narrow ones become thin. The approach presented here, in contrast, is image-based. Inflatable Icons are based on a diffusion process taking place at the pixel level. In computer graphics diffusion and related operations are often used as technique to smooth [3] or to deform [2] 3D objects.

To capture the notion of inflation we introduce the idea of a *biased diffusion* process. Unlike most diffusion applications, including the use of diffusion for smoothing, biased diffusion does not attempt to preserve the total amount of matter. A biased diffusion intentionally changes the amount of matter. The resulting process resembles inflation quite closely. The bias introduced to diffusion is a measure for pressure. Pressure added to an Inflatable Icon will gradually extrude the image-based mesh from a flat shape into a 3D model.

Biased diffusion can be represented with a difference equation computing a new value based on the four Newman neighbors (north, south, east, west) and itself.

Biased Diffusion Equation

$$z_{x,y} = D(z_{x-1,y} + z_{x+1,y} + z_{x,y-1} + z_{x,y+1} - 4z_{x,y} + p) + z_{x,y}$$

D is the diffusion coefficient [0..0.5], p is a number corresponding to pressure.

A positive pressure p will inflate, while a negative one will deflate. All z values are initialized to 0. Z values of invisible pixels remain always 0. Masked pixels surrounding our Lobster icon will clamp down, i.e., the pixels at the edge of the icon will pull the diffusion values down to zero. Because of this clamping icon inflation converges. The inflated lobster (Figure 5) was created with

only 10 iterations using a pressure p of 0.04, and a diffusion coefficient of 0.25. The converged shape of an inflated icon will only depend on the pressure p but not the diffusion coefficient D.

Pixels inside the icon far away from the edge will assume the highest z values. The highest z values are a function of the pressure p. The higher the pressure, the larger the z values will become. A pressure value of zero will flatten an already inflated icon. Negative pressure will create concave icons.

Applying pressure to the Inflatable Icon will gradually pump it up like a balloon. Z values are shown as red lines (Figure 3).

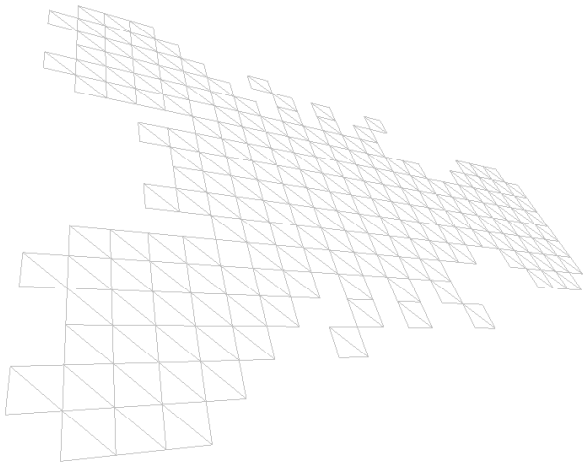

Figure 3a: Inflation: icon is still flat.

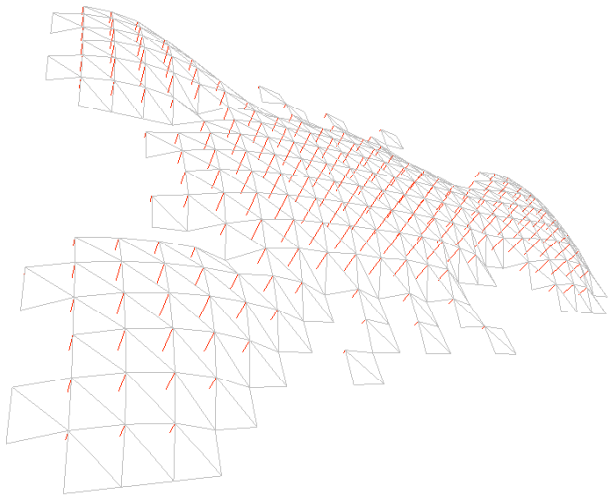

Figure 3b: Inflation: icon starts to inflate.

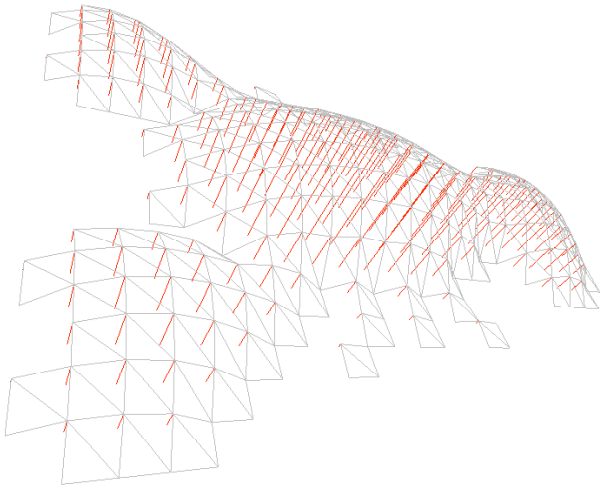

Figure 3c: Icon is fully inflated by Biased diffusion.

Rendering polygons filled and enabling lighting will make the Inflatable Icon look like a solid object.

Figure 4: A flat shaded inflated icon.

Because of the regular grid structure of the underlying image it is particularly simple to implement Gouraud shading [6] by computing normal vectors for each vertex and not just for each triangle face as follows:

$$\vec{n}_{x,y} = \begin{pmatrix} z_{x-1,y} - z_{x+1,y} \\ z_{x,y-1} - z_{x,y+1} \\ 2m \end{pmatrix}$$

m is the grid size ($m = \Delta x = \Delta y$)

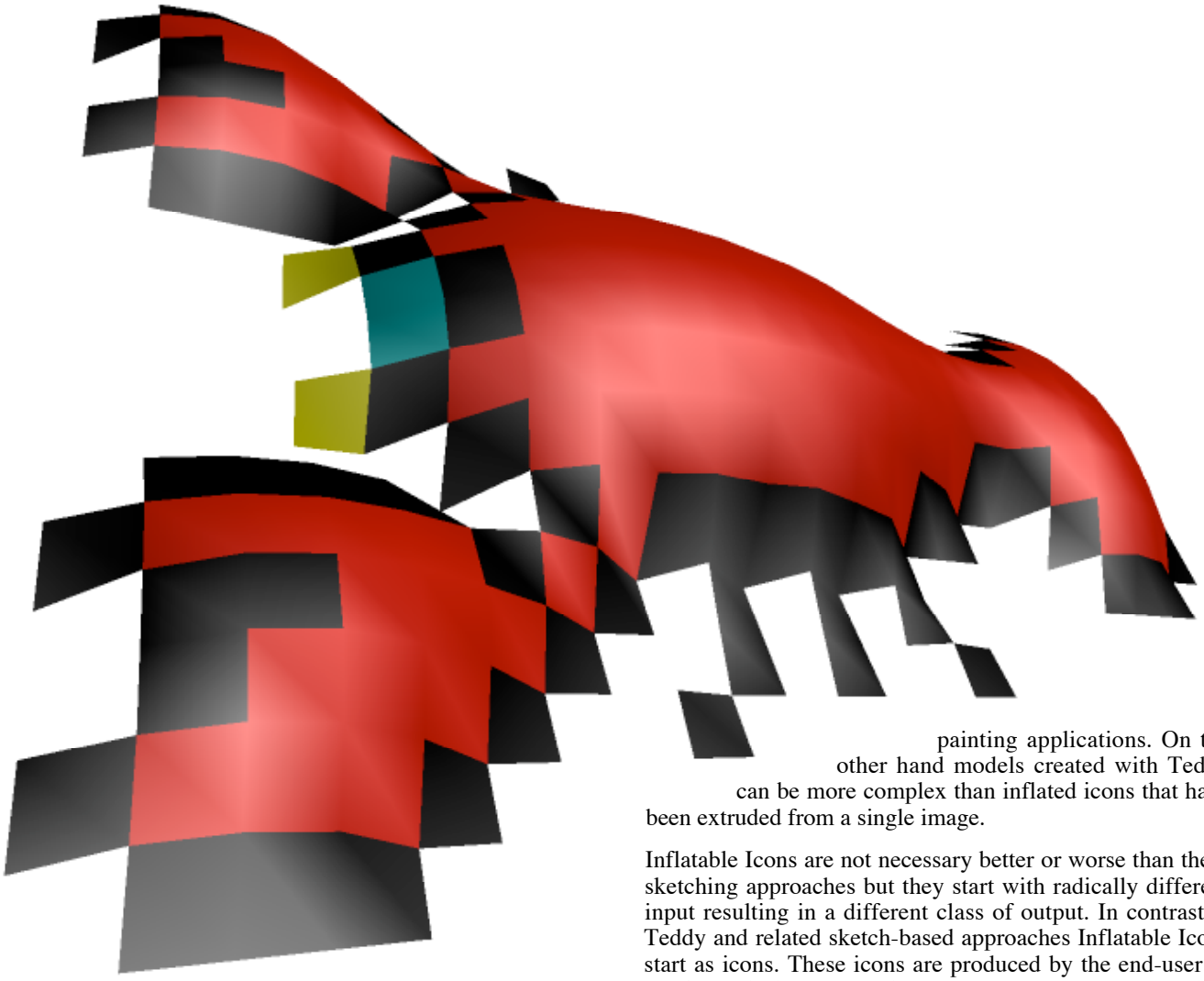

Figure 5: Gouraud shaded inflated icon.

Related Work

Sketching approaches such as the Electronic Cocktail Napkin system [7] interprets pen drawn sketches to create diagrams with semantics by matching sketches against a large and expandable set of graphical primitives with user defined semantics. Digital Clay [13] not only recognizes sketches but can also construct appropriate 3D models. Sketch VR [4] recognizes 2D geometric shapes that it can project into 3D architectural spaces. Gesture-based interfaces have also been used to create 3D models of mechanical designs [5] and CAD/CAM models [10].

Freestyle sketching approaches do not rely on domain knowledge but use sketching to create arbitrary 3D objects that are not interpreted semantically by the computer. Teddy [8] is a sketching interface to create all kinds of sophisticated 3D models that "have a hand-crafted feel (such as sculptures and stuffed animals) which is difficult to accomplish with most conventional modelers." Teddy's inflation mechanism is based on closed regions. It does not use image information to generate models nor does it include texturing tools. Models created with Teddy do not include skins and need to be textured with third party painting applications. On the other hand models created with Teddy can be more complex than inflated icons that have been extruded from a single image.

Inflatable Icons are not necessary better or worse than these sketching approaches but they start with radically different input resulting in a different class of output. In contrast to Teddy and related sketch-based approaches Inflatable Icons start as icons. These icons are produced by the end-user or are found in icon collection such as the Icon Factory [14]. Icons are not only decoration in the sense that they will be used later as the skin of the model but they actively drive the inflation process. The icon mask is used to derive the basic shape of the model. In more complex cases (explained below) the color value of each pixel is analyzed as well and is used to control the diffusion process.

A number of image-based extrusion approaches add three-dimensional looking effects to two-dimensional images without creating tree dimensional models. William's automatic airbrush [16] creates compelling three-dimensional illustrations out of images by applying complex shading functions to selected regions. Simpler versions of this idea are found in popular paint programs such as Photoshop. However, the results of these algorithms remain two-dimensional shapes with no user-accessible 3D model information.

USER CONTROL

Our user testing indicated that the pressure parameter is not sufficient to control the extrusion process. Below, a number of additional parameters that provide users with more options are described. Users suggested many of these parameters, including noise.

Aside from providing users with an ever-growing number of parameters an important general design principle that

emerged from user testing was that continuous feedback helps users enormously in building inflatable icons. Most of the parameters presented in a numerical form provide little or no information to users. The nature of the diffusion process is highly iterative. The extrusion of an image into inflated icons typically requires dozens to hundreds of iterations. Instead of keeping this computation hidden behind the scenes it was made a part of the interface explicitly revealing the diffusion process to its users as a continuous inflation animation. Especially younger users tend to playfully explore the inflation process and particularly enjoy inflating icons to extreme degrees – expecting the icons to pop.

Antialiasing

Icons are low-resolution small two-dimensional images. A typical desktop icon has a size of 32 x 32 pixels. Even so-called "huge" icons (in Apple's OS X) have a size of 128 x 128 pixels. Small size, on the one hand, makes the inflation computationally process feasible, but, on the other hand, results in shapes that have a smooth surfaces but jagged edges.

A 32 x 32 pixel, 1 bit alpha circle icon manifests highly visible aliasing.

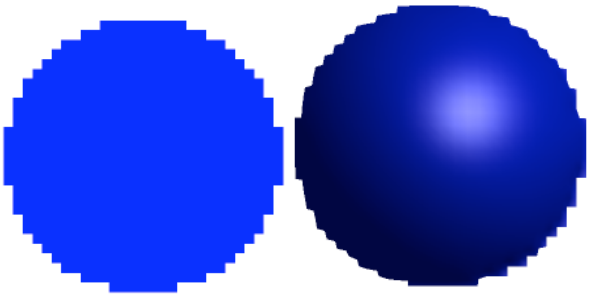

Figure 6: A circle icon with visible aliasing (left). The inflated icon (right) has a smooth surface but keeps the aliased edge.

Icon Inflation turns the circle (Figure 6, left) into a hemisphere with a smooth surface but jagged edge (Figure 6, right). In a perspective view the edge of a rotated hemisphere will look even worse (Figure 7, left). The diffusion process used for the inflation also helps to smoothen the edge.

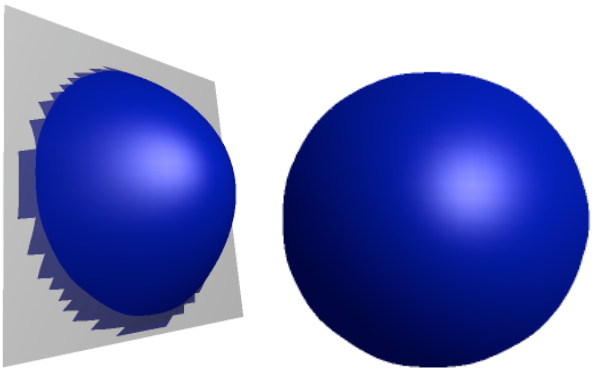

Figure 7: The "bad" aliased part of the inflated icon is cut away with a clipping plane (left) to get a smooth edge (right).

A simple, yet effective, antialiasing effect is achieved through a clipping plane (Figure 7, left) parallel to the x/y plane with a small z value. The selection of a z value for the clipping plane is user controlled through a slider interface, similar to the one shown in Figure 14. A small z value will preserve the original shape but still manifest the jagged edges. A large z value, in contrast, will make the edge of the shape smooth but will also thin out the shape. Thinning out is a problem for certain shapes that include sharp edges or intricate details. Figure 8 illustrates how the sharp tip of a heart gets lost because of the smoothing.

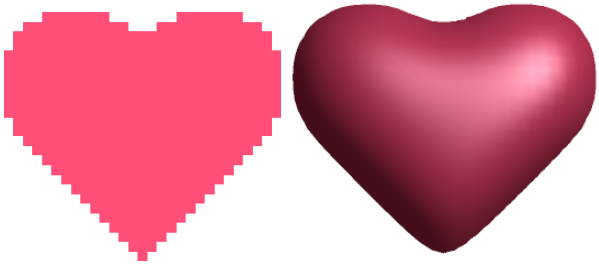

Figure 8: The sharp edge at the bottom of the heart shapes gets lost in the antialiasing.

Color-Based Pressure Modulation

Uniform diffusion will not always work well. In many cases some regions of an icon need more and others need less pressure for an inflated icon to look right. The inflated version of a teddy bear icon does not look particularly compelling especially when viewed from the side. One would expect the black nose to be more prominently extruded and the ears more pronounced. Without this the resulting shape has the appearance of an inflated balloon with the face of the teddy bear merely painted on (Figure 9, right).

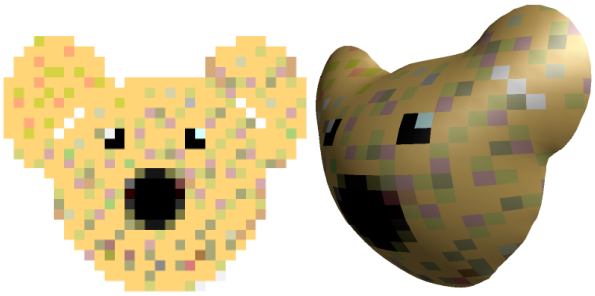

Figure 9: A general inflation turns a teddy icon (left) into a non-compelling inflated icon (right)

To achieve the desired effect the universal pressure p in our general diffusion equation is replaced with pixel coordinate depending function. This function modulates the pressure for each pixel based on the pixel color.

$$p_{x,y} = p(1 + \mathrm{m}(color_{x,y}))$$

For the teddy bear icon we define

m(black) = 1;

m(white) = -3;

otherwise m = 0.

The black pixels representing the nose modulate the pressure positively whereas the three white pixels serving as edge between the main sphere of teddy's head and his ears modulated the pressure negatively (Figure 10).

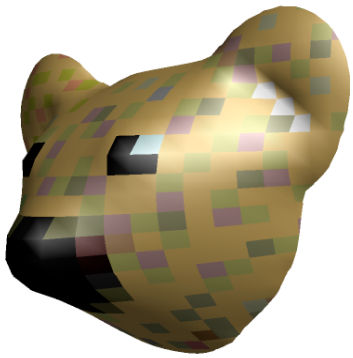

Figure 10: Black pixels have more pressure, white pixels have less pressure

End-users, of course, do not specify color modulation functions. Instead, they pick colors (or color ranges) from palettes. For each selected color they use a slider with a limited positive and negative range to define a modulation bias. Sliders provide a good interactive means to control color-based pressure modulation.

Noise

Objects found in nature such as rocks often do not have smooth surfaces. Noise can be added to give inflatable icons a more organic look. Users are provided with two noise parameters:

• ***Amount***: Control the noise amplitude.

• ***Bumpiness***: Should the surface be more spiky or smooth?

We want to create a model of a rock. A gray blob icon is drawn quickly (Figure 10, left) and inflated (Figure 10, right).

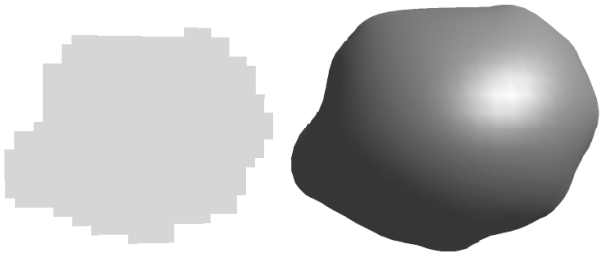

Figure 11: A gray blob (left) turns into a balloon (right)

Using a random function, noise gets added to each $z_{x,y}$ of the inflated icon with values in between –amplitude and +amplitude. In many cases, including this one, the random spikes added to the surface require smoothing (Figure 12).

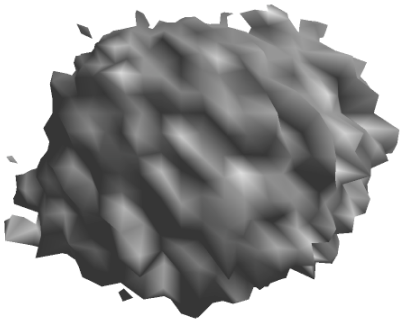

Figure 12: Noise added to the balloon makes it look like a sponge and not like a rock.

Non-noisy diffusion is applied again to smoothen the surface. The bumpiness parameter controls the number of follow-up smoothing diffusion cycles. With each additional diffusion cycle, the inflated icon becomes smoother. Continued smoothing will eventually turn the inflated icon back into its original shape (before the noise was applied).

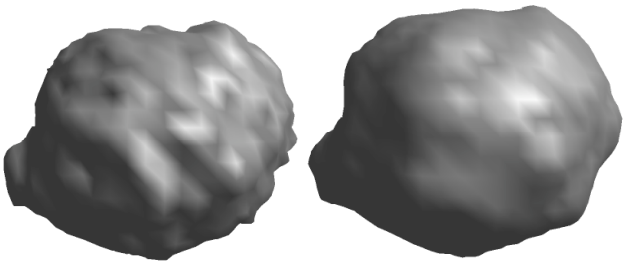

Figure 13: Two iterations of diffusion (left) and three iterations of diffusion (right) applied to the sponge create a rock look.

INTERFACE

The interface is kept simple with the main design goal to facilitate exploration. The main pressure parameter is accompanied with a number of optional parameters such as

the ones described above. Additionally users can specify *orientation* (should an inflated icon lay on the ground, such as the lobster icon, or should it be upright, such as the icon of a tree), *symmetry* (e.g., a circle can be turned into a complete sphere combining positively and negatively inflated copies of a circle icon), and simple *composition* of different inflatable icons into more complex shapes.

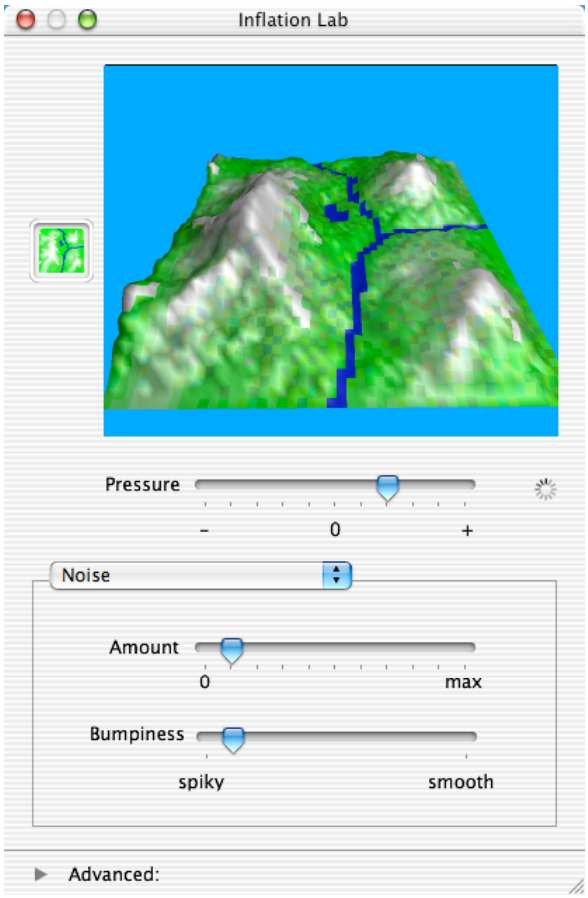

Figure 14: A simple user interface to inflate icons.

Figure 14 shows the inflation of an 48 x 48 pixel icon representing a landscape. Positive pressure inflates the landscape. Color-based pressure shapes the mountains and the river valley: white color represents snow; blue color represents the river. Noise makes the mountains look rough. The view showing the Inflated Icon is an OpenGL viewer that includes camera controls (zoom, rotate, pan), which allow users to explore their 3D world from different perspectives.

The interactive nature of the process keeps users in the design loop. A change of pressure will result in inflation/deflation of the icon by running the diffusion process for about 50 steps. The inflation process is stable in the sense that it converges. Only for extreme values of pressure will the icon inflation become numerically instable. Our first prototype tested by middle school students did not include a pressure limit. Users enjoyed entering extreme pressure values seeing over inflated icons pop. Following each individual diffusion step the display is updated providing users a sense of the process. At the same time users may continue to edit their icons changing the shape, the texture and, in the case of color-based pressure modulation, selectively controlling the ongoing diffusion process.

DISCUSSION

Not every icon is suitable to icon inflation. Especially more modern icons that have a three dimensional look may not inflate well for two reasons. First, their shape is pre-distorted to look like a 3D object. The inflatable icon approach assumes that icons are flat two-dimensional projections. A second problem can be shading that, again, is used to create a three dimensional appearance. The inflated icon will inherit the statically shaded texture from its two dimensional icon. Since the inflated icon is a true three-dimensional object placed into a three-dimensional world with potentially many light sources the result will be highly confusing. In some cases the user can clean up the icon with an icon editor. However, depending on the complexity of the icon this may require a considerable effort.

The polygonization approach used creates models with large mesh sizes. Even a small 32 x 32 icon may have over 1000 polygons. Assuming that applications such as simulations may feature thousands of Inflatable Icons performance could become an issue. We have compared the time required to draw a regular icon with the time to draw an Inflatable Icon.

Test machine: Macintosh G4, 800Mhz, OS X 10.2.3, Video Board, ATI Mobile Radeon; test icon: Lobster (Figure 1).

> 2D Icon: 280μs – using native operating system call
>
> 3D Inflatable Icon: 220μs - OpenGL 1.3, display lists, one light source, Gouraud shading.

The absolute numbers are irrelevant, however, the fact that even without optimizations such as OpenGL extensions (e.g., compiled vertex arrays) it took less time to create a 3D inflatable icon than a regular 2D one is amazing.

Models produced by icon inflation tend to be smooth making them viable candidates for well-established mesh reduction algorithms [9] if optimization is required.

A somewhat constraining aspect of Inflatable Icons is the biased diffusion process, which does not scale well to large images. The problem is that it will take pixels far away from the icon edge a long time to rise because the pixels that need to raise the most are surrounded by a very large number of pixels pulling them down. The result is that it takes many more iterations to inflate an icon properly. One way to overcome this limitation is by first downsampling large images, inflate the downsampled image and then to apply the original image as the skin of the resulting 3D model. This way the inflation technique can be applied to images of arbitrary size.

It is important to stress that Inflatable Icons are not meant to replace more complex approaches and sophisticated tools for creating 3D models. In applications such as simulation authoring environments end-user such as kids

need to be able to create good-enough 3D models in seconds or minutes but not in hours.

The type of models that can be extruded from a single surface is intrinsically limited. On the other hand, the approach of Inflatable Icons can be extended to introduce more complex extrusion functions and to add sophisticated composition functions. Any number of icons could be inflated and then composed into an aggregate model.

REFERENCES

1. J. Bloomenthal and B. Wyvill, "Interactive techniques for implicit modeling," presented at Symposium on Interactive 3D Graphics, 1990.
2. G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr, "Dynamic Real-Time Deformations using Space and Time Adaptive Sampling," presented at SIGGRAPH '01, 2001.
3. M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow," presented at SIGGRAPH 99, 1999.
4. Y.-L. Do, "Drawing Marks, Acts, and Reacts, toward a computational sketching interface for architectural design," *AIEDAM, Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 16, pp. 149-171, 2002.
5. L. Eggli, C. y. Hsu, B. D. Bruderlin, and G. Elber, "Inferring 3d models from freehand sketches and constraints," *Computer-Aided Design*, vol. 29, pp. 101-112, 1997.
6. H. Gouraud, "Illumination for Computer Generated Pictures," *Communications of the ACM*, vol. 18, pp. 311-317, 1971.
7. M. D. Gross, "The Cocktail Napkin, the Fat Pencil, and the Slide Library," presented at Association for Computer Aided Design in Architecture (ACADIA '94), St Louis, 1994.
8. T. Igarashi, "Teddy: A Sketching Interface for 3D Freeform Design," presented at *Proceedings of ACM SIGGRAPH 99*, Los Angeles, 1999.
9. R. Klein, G. Liebich, and W. Straßer, "Mesh reduction with error control," presented at Visualization 96, 1996.
10. D. Lamb and A. Bandopadhay, "Interpreting a 3D object from a rough 2D line drawing," presented at Visualisation '90, 1990.
11. H. Lieberman, *Your Wish Is My Command: Programming by Example*: Morgan Kaufmann Publishers, 2001.
12. A. Repenning, A. Ioannidou, and J. Zola, "AgentSheets: End-User Programmable Simulation," *Journal of Artificial Societies and Social Simulation*, vol. 3, 2000.
13. E. Schweikardt and M. D. Gross, "Digital Clay: Deriving digital models from freehand sketches," *Automation in Construction*, vol. 9, pp. 107-115, 2000.
14. Icon Factory, http://www.iconfactory.com/
15. StarLogo, http://education.mit.edu/starlogo/
16. L. Williams, "Shading in Two Dimensions," presented at Graphics Interface '91, 1991.

WEB INFORMATION

The figures from this paper, as well as short movie, are available at

http://www.acm.org/jgt/papers/Repenning04