Sculpting with the "Ball and Mouse" Metaphor

André LeBlanc
Computer Graphics La, Swiss Federal Institute of Technology
Lausanne, Switzerland

Prem Kalra
MIRALab, CUI, University of Geneva
Switzerland

Nadia Magnenat Thalmann
MIRALab, CUI, University of Geneva
Switzerland
and HEC Montréal, Canada

Daniel Thalmann
Computer Graphics La, Swiss Federal Institute of Technology
Lausanne, Switzerland

Abstract

In this paper, we present a user interaction methodology based on a six degree of freedom interactive input device called the Spaceball™. This methodology is geared for computer graphics applications that require items to be positioned or displaced in three-dimensional space in a purely visual and esthetic fashion. When the Spaceball device is used in conjunction with a common 2-D mouse — the Spaceball device held in one hand and the mouse in other — overall visual depth perception on traditional 2-D displays can be considerably enhanced by exploiting a technique called motion parallax, thereby promoting full three-dimensional user interaction. This paper focusses on the application of this methodology to sculpting highly irregular polygon mesh surfaces, such as character faces or any other surfaces of arbitrary shape. The Spaceball device is used to move the object being sculpted while the mouse carries out the picking and deformation work, all of which is projected onto the screen in real time. As case studies, three common operations in polygon mesh sculpting are described in detail (vertex creation, primitive selection and local surface deformations).

Résumé

Cet article présente une méthode d'interaction basée sur un périphérique d'entrée ayant six degrés de liberté, connu sous le nom de Spaceball. La méthode permet de positionner ou de déplacer les éléments graphiques tri-dimensionnels de façon hautement interactive et visuelle, et s'avère particulièrment intéressante dans les applications où le caractère esthétique est le facteur dominant et la précision numérique est peu importante. La Spaceball est employée conjointement avec la souris 2-D traditionnelle afin d'accentuer la perception d'objets 3-D et faciliter l'interaction avec ces objets. Dans une main, on manipule les objets avec la Spaceball et simultanément, on travaille les objets avec l'autre main à l'aide de la souris. Cette technique exploite le concept de parallaxe du mouvement qui permet une bien meilleure perception de l'espace 3-D. L'article se concentre sur l'application de cette méthodologie pour la sculpture de surfaces irrégulières formées de polygones, comme des visages par exemple. Trois opérations communes sont décrites: la création de sommets, la sélection de primitives et les déformations locales de surfaces.

Keywords: User interaction, Spaceball input device, motion parallax, sculpting.

1. INTRODUCTION

Visual feedback, in a typical computer graphics application that requires items to be positioned or moved in 3-D space, usually consists of a few orthogonal and perspective projection views of the same object in a multiple window format. This layout may be welcomed in a CAD system where, in particular, an engineer might want to create fairly smooth and regular shapes and then acquire some quantitative information about his design. But in 3-D applications where highly irregular shapes are created and altered in a purely visual and esthetic fashion, like in sculpting or keyframe positioning, this window layout creates a virtually unsolvable puzzle for the brain and makes it very difficult (if not impossible) for the user of such interfaces to fully understand his work and to decide where further alterations should be made.

Until recently, the greatest obstacles in the elaboration of intuitive human-machine interaction methods for 3-D graphical applications came from basically insufficient computing power, slow frame rates and the lack of adequate multi-dimensional input devices. Highly interactive applications that require uninterrupted interaction in 3-D space rely on fast display rates in order to assure that the user may view the result of his actions without any perceived time delays. To keep up with these fast refresh rates when making changes to the state of the environment, good computing power is also an important asset. Completing this list of hardware requirements is the need for multi-dimensional input devices that can relate user information to the computer in an intuitive fashion. Indeed, advances in all these areas have provided us with a wide set of tools for building interaction methods that are more intuitive for 3-D applications.

This paper discusses work involving a user interaction methodology that is based on a six degree of freedom (DOF) interactive input device called the Spaceball™. This input device is designed to use a person's spatial intuitions to move and orient objects in space with greater dexterity. When object movements can be produced with this device and displayed in real-time, depth perception on traditional 2-D displays may be considerably enhanced by exploiting a psychological phenomena called *motion parallax* (Cahen,Forrest). With this added mobility and depth perception, we show how visualy specifying 3-D locations and displacing them in space with the help of a *single projection window* can be made a simpler task.

In this paper, we are particularly interested in studying how this methodology may be applied to sculpting highly irregular surfaces, for instance the face of a cartoon character or the body of some fictitious animal. As case studies, three polygon mesh sculpting operations using this methodology are described in detail — vertex creation, geometric primitive selection and real-time local surface deformations. Transformation matrices corresponding to these operations are also described in detail.

2. THE BALL AND MOUSE METAPHOR

2.1. Object Motion Dexterity

In essence, *motion parallax* consists of the human brain's ability to render a three-dimensional mental picture of an object simply from the way it moves in relation to the eye. Rotations offer the best results because key positions located on the surface move in a larger variety of directions. Furthermore, in a perspective projection, depth perception is further accentuated by the speed in which features flow in the field of view — points located closer to the eyes move faster than the ones situated in back. In a 3-D application, if motion parallax is to be used effectively, this implies the need for uninterrupted display of object movements and thus the requirement for hardware capable of very high frame rates.

To acquire this depth perception and mobility in a 3-D application, we make use of a 6 DOF interactive input device called the Spaceball (Figure 1). This is essentially a "force" sensitive device that relates the forces and torques applied to the ball mounted on top of the device. These force and torque vectors are sent to the computer in real time where they are interpreted and may be composited into homogeneous transformation matrices that can be applied to objects. Buttons mounted on a small panel facing the user control the sensitivity of the Spaceball and may be adjusted according to the scale or distance of the object currently being manipulated. Other buttons are used to filter the incoming forces to restrict or stop translations or rotations of the object.

Since our main usage for the Spaceball is to manipulate objects in space (as opposed to animating the eyepoint), we adhere to the "Scene in hand" metaphor as described in (Ware and Osborne). This means that we consider the viewpoint of the camera to be static and let the user move and rotate the object in eye coordinates with the Spaceball (see section 3).

Unlike the Polhemus 3Space™ digitizer (another 6 DOF input device), the Spaceball does not relate absolute position and orientation information in 3-D space. Instead, all incoming data from the Spaceball represent a relative change in position and orientation; this makes the Spaceball a good 6 DOF "steering" mechanism. In our experience, steering an object in space does not pose any problems for the average user. It usually takes on the order of 10 to 20 minutes to get comfortable with the device and there is little difficulty manipulating the object thereafter. Additionally, the user's arm is at rest at all times while moving the object, which is a clear advantage over the Polhemus where the stylus has to be held in midair.

Coming back to the discussion on motion parallax, it is important to note that the illusion of three-dimensionality caused by this phenomena is immediately lost once the object stops moving. Therefore, intermittently using the Spaceball device for purposes other than object movements would abruptly paralyze the user's freedom of movement and lose his 3-D visual perception capabilities. If motion parallax is to be used to its fullest in a 3-D application as the primary means of object examination, the Spaceball device should, in principle, be used exclusively for the purpose of moving objects.

2.2. The Ball and Mouse Metaphor

Since the operator of the Spaceball device needs only one hand to move and examine objects on the screen, his other hand is free to use another input device. This is the essence of the user interaction methodology that we refer to in this paper as the Ball & Mouse metaphor.

The metaphor closely resembles the kind of visual interface that a real-life's microtechnician would encounter when working with a very fine and delicate object:

1. A magnifying screen is needed in order to see the very small details.
2. One hand is used to hold the object in place or to move it around in order to examine it from other viewpoints.
3. The other hand is used to operate various tools to work on the object.

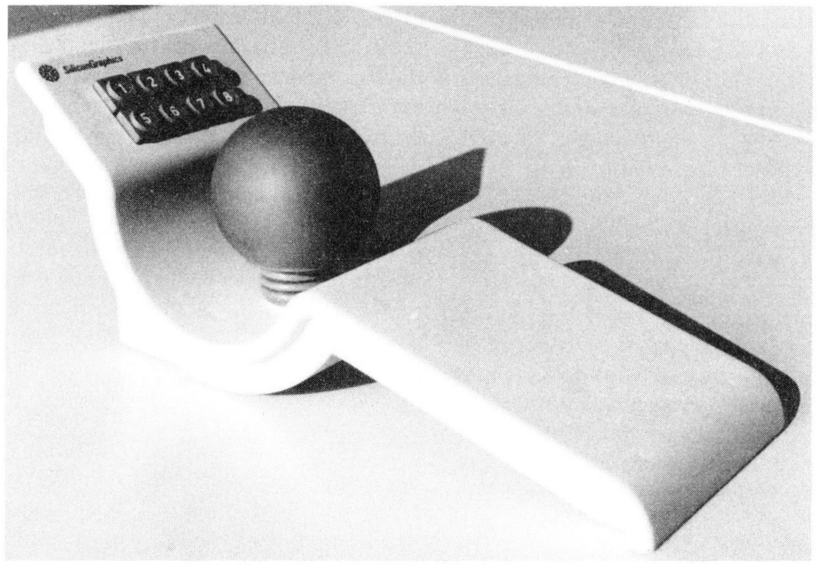
Figure 1. Spaceball device

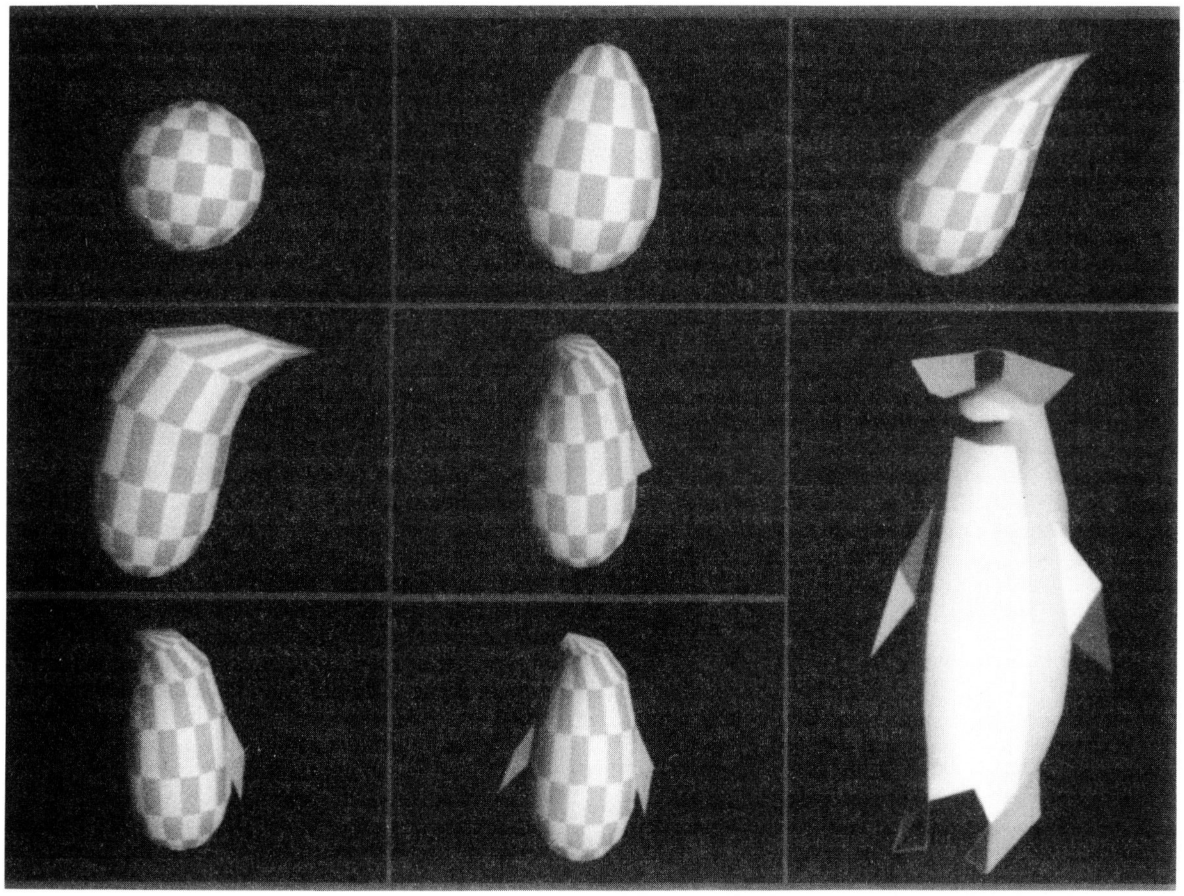
Figure 2. Sequence of images showing how a penguin caracature may be created from a sphere

In the case of the microtechnician's interface, the magnifying screen through which he examines small objects resembles, in a purely metaphoric sense, to the common CRT display screen through which virtual objects are visualised in computer graphics. By itself however, the 2-D CRT display screen can neither restitute binocular vision nor motion parallax caused by head movements of its user. If a feasible solution to these problems could be found, by using stereo imaging material and by tracking the position and orientation of the head in space, a true 3-D screen (or in this case, a type of hologram box) could then be closely emulated.

Although binocular vision would considerably enhance visual depth perception, we note that it is not an absolute necessity (Forrest) if good motion parallax is present — we could very easily imagine a microtechnician working with one eye. Additionally, the lack of motion parallax caused by head movements can be compensated by added object motion dexterity.

2.3. Mouse Device

For the choice of the user input device used by the second hand, we have experimented with a standard 2-D mouse. This device has the advantages of being widely available on all graphic workstations and is particularly useful in windowed applications for selecting widgets. Additionally, like in the case with the Spaceball device, the user's arm is at rest at all times.

In section 4 on applications to polygon mesh sculpting, all sculpting operations are given using a standard 2-D mouse. Experimentation with devices of higher degree of freedom, like a 3-D mouse or Polhemus 3Space, remains an open research topic.

3. COORDINATE TRANSFORMATION NOTATION

Here we define the coordinate system notation that we will use in the following sections to describe various homogeneous matrix transformations. We define:

M	Modeling matrix that transforms object coordinates into a second modeling coordinate system, called Node coordinates, where Spaceball transformation matrices are applied.
N	Node matrix, on which Spaceball transformations are applied, that transforms the second modeling coordinates into eye (or viewing) coordinates.
PV	Standard Viewing and Projection matrices that transform eye coordinates into normalized coordinates in such a way that $-1 \leq x,y,z \leq 1$.

Transforming vectors $p_i = [x, y, z]^T$ from model coordinates to normalized coordinates is done by multiplying $PVNM\ p_i$.

4. APPLICATIONS TO POLYGON MESH SCULPTING

For the purposes of this paper, we think of sculpting as the purely visual process by which a polygon mesh surface is created — for example by positioning vertices in three-dimensional space and linking them into a mesh — and then altered by locally displacing vertices in space in a controled manner (local deformations). Rough-hewed shapes may be generated by an external software application and loaded into the sculpting software as the starting point of a sculpting session.

Creating or adding a vertex in 3-D space requires the user to indicate its position in relation to an existing object. Then, a good geometric primitive selection mechanism is needed in order to select the vertices to be polygonized or moved, or to select existing polygons or edges to which various other operations like splitting (Allan et al.) would be applied. This requires the ability to move the object around and "point" towards the wanted primitives. And finally, altering the shape of the surface necessitates the knowledge of the 3-D direction in which selected vertices are to be moved. In this section, we show how these sculpting operations are implemented using the Ball & Mouse metaphor.

4.1. Vertex Creation

Typically, the sculpting process may be initiated in two ways; by loading and altering an existing rough-hewed shape or by simply starting one from scratch. For example, our experience shows that a designer will use a sphere (Figure 2) as a starting point for the head of a person or use cylinders for limbs or torsos. He will then add or remove polygons according to the detail or slope of the surface and apply local deformations to alter its shape. Starting from scratch — by placing vertices in 3-D space and polygonizing — offers an alternative but is usually more tedious and time consuming. Combining the two makes it possible to link disjointed surfaces together, close holes on surfaces and make small add-ons. Here, we describe a method for creating a single vertex in 3-D space.

The idea comes from the fact that a 3-D point can easily be located by finding the closest point of any two non-parallel lines.

This is done in three quick steps:

1. The user aims with the mouse cursor to specify where, in relation to an existing reference on the screen, the new point should be located. The software generates a reference line corresponding to this direction.

2. He turns the object with the Spaceball in such a way that he senses a comfortable parallax.

3. He aims a second time, from this new viewpoint, in the direction where the point should be located.

The new vertex will be located at the convergence point of the two lines entered by pointing.

Suppose for example that the user wishes to close a hole in a surface by creating a middle vertex (Figure 3), he simply aims in the direction of the center of the hole, rotates the object slightly with the Spaceball, and aims again in the direction of the center.

In step (2) above, maximum parallax will be obtained by turning the object at a 90° angle relative to the first line. This will permit the user to enter his new vertex with greater precision. However, good precision can be obtained by turning the object at a much inferior angle and repeating the pointing procedure (third step) relatively quickly. We suspect that this is due to the human brain's ability to obtain depth information by motion parallax. In any case, if the vertex is not properly positioned in space, it may easily be moved afterwards (section 4.3).

4.2. Item Selection

For selecting parts of the object, the mouse can be used in conjunction with the Spaceball to quickly mark out the wanted primitives in and around the object. In our implementation, this amounts to pressing the mouse button and sweeping the mouse cursor on the screen while moving the object with the Spaceball. All primitives (vertices, edges and/or polygons) that have passed underneath the area of the cursor will then be candidates for selection. Mass picking may be done by moving the object away from the eye (assuming a perspective projection) and careful, minute picking may be done by bringing the object closer.

4.3. Local Surface Deformations

After vertices have been created and positioned in space, the sculptor calls on tools to move them about, individually or in groups, in three-dimensional space. These tools make it possible to produce local elevations or depressions on the surface and to even out unwanted bumps once the work is nearing completion.
In this section, we show how local surface deformations are applied using the Ball & Mouse methodology. The idea is that while the Spaceball device is used to move the object and examine the progression of the deformation from different angles, *mouse movements* on the screen are used to produce vertex movements in 3-D space from the current viewpoint.

Much in the same manner as (Allan et al.), we wish to displace vertices in a controlled fashion according to some decay function that is radially symmetric about a vertex. When this vertex, called the apex vertex, is moved in 3-D space, other vertices follow in the same direction with amplitudes that vary as a function of their initial distance to this vertex. Many different deformation functions are possible to use depending upon the final shape desired for the selected region of vertices. Some of the functions, for example, are bell, cusp, single, box, sine, alps and wave.

This methodology is intended to be a metaphor analogous to pinching, lifting and moving a piece of a stretchable fabric material. Pushing the apex vertex inwards renders the effect of pressing a mould into clay. In the Ball & Mouse user interface methodology, this translates into "pinching" the apex vertex with the mouse and moving this vertex in 3-D space (Figure 4).
We define a local deformation function $f(r)$ as follows:

$$f(r) = 1 \quad \text{for } r = 0.$$
$$0 \leq f(r) \leq 1 \qquad \text{for all } 0 < r \leq 1.$$

where r is a scalar value from 0 to 1 inclusively. In three-dimensional model coordinates, r is calculated as follows:

Let

$p = \{ p_i = (x,y,z) \}$, the set of selected vertices p_i,

$p_a \in p$, the vertex where the apex of the deformation will occur,

then

$rmax = max(|p - p_a|)$,

$r_i = |p_i - p_a| \,/\, r_{max}$, is the distance from p_a to $p_i \in p$ divided by the distance r_{max} to the farthest vertex of p.

Therefore, $f(r_i)$ is a radially symmetric function about the axis passing through p_a. If p_a is the apex of the deformation, we define the transformed set of vertices as follows:

Let

$t = \{ t_i = (x,y,z) \}$, the set of transformed vertices p,

$t_a \in t$, the transformed apex vertex p_a,

$A = t_a - p_a$, the vector of the current deformation,

then

$t_i = p_i + f(r_i) * A$, for all $i \in [1..n]$.

This means that moving vertex p_a in a direction A will cause all other selected vertices p_i to follow in the same direction A scaled as a function $f(r_i)$. If A is a null vector, no deformation will take place — this can be used to undo a deformation. As t_a is moved arbitrarily in three-dimensional space, A increases and accentuates the deformation in that particular direction (Figure 4).

As a side note, since the values of $f(r_i)$ do not change during the interactive deformation process, they may be computed and stored beforehand. This optimization is welcomed on smaller systems in order to maintain interactivity when transforming a large number of vertices.

In essence, applying a local surface deformation comes down to choosing the apex vertex amongst the selected group, computing distances and values of the function in relation to this vertex and then moving it in three-dimensional space.

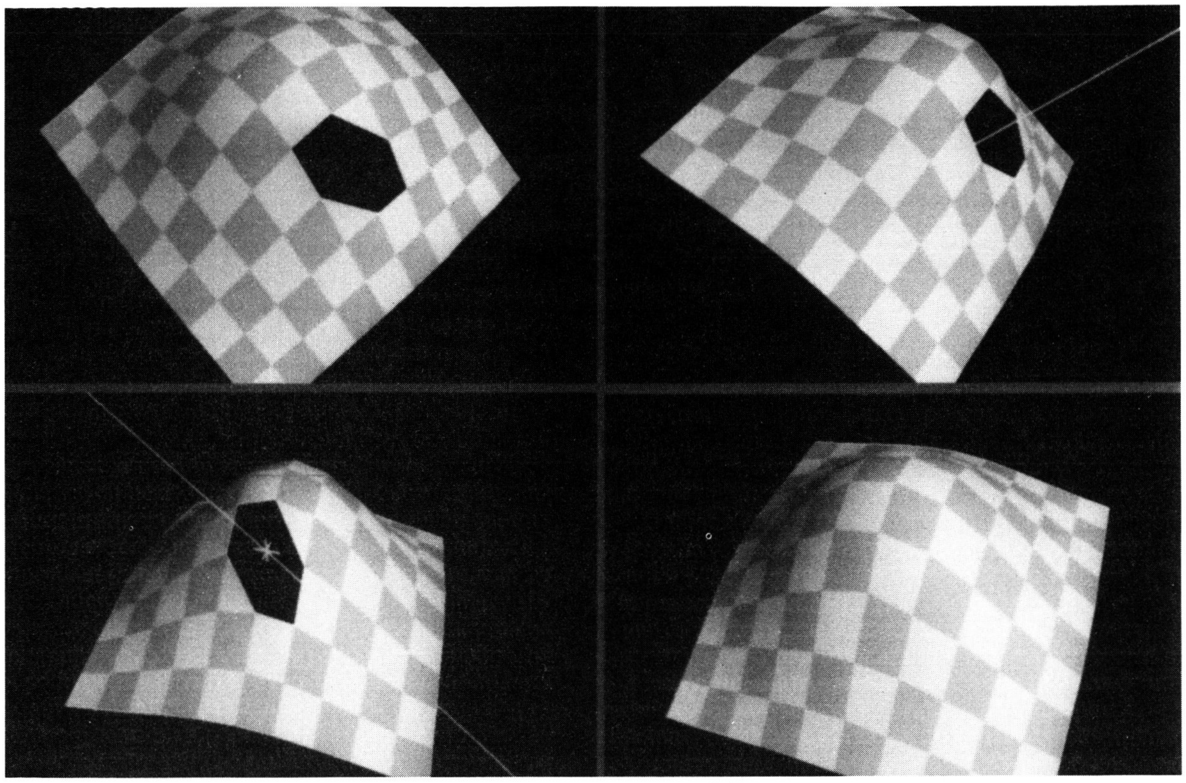
Figure 3. Locating a point in 3-D

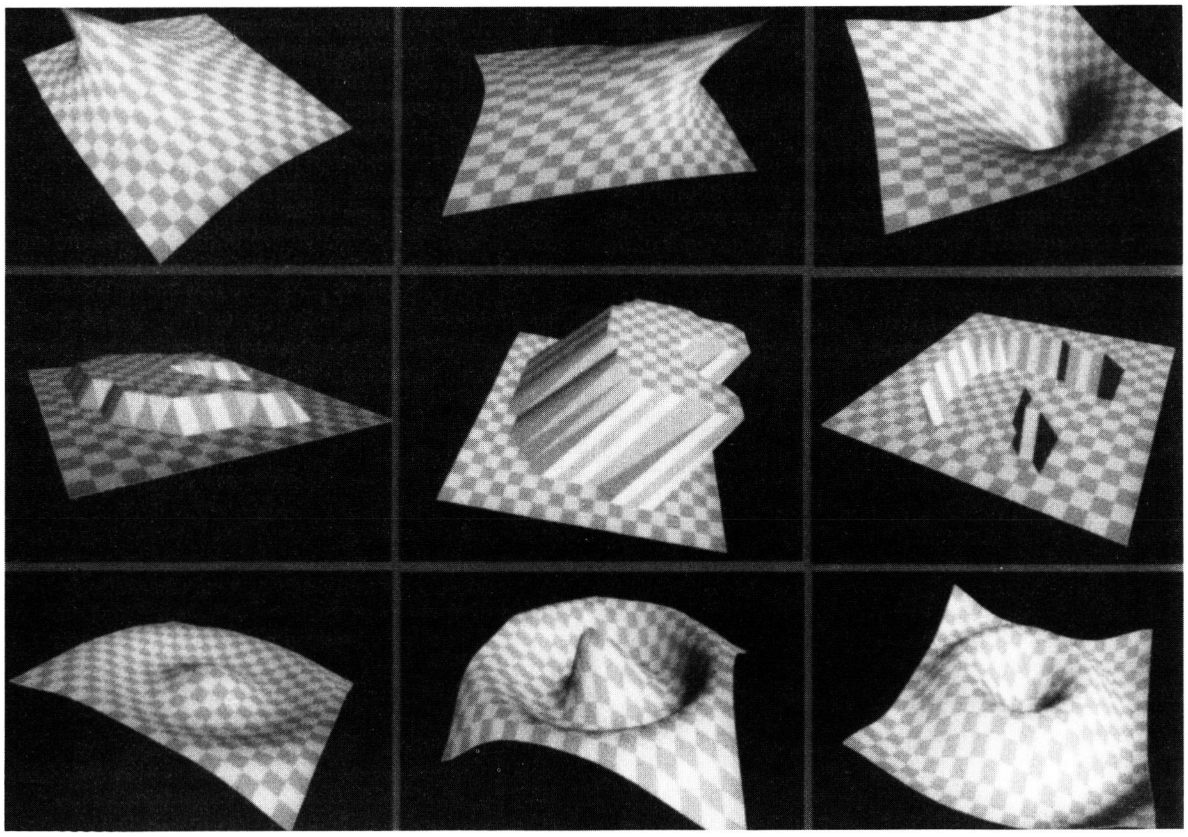
Figure 4. Example of local deformation functions. The top row shows a cusp, middle row a box type deformation and bottom row a sine wave

Calculating the Deformation Amplitude Vector

To be consistent with the pinching metaphor, the apex vertex should move at the same speed on the screen as the mouse cursor. This should be the case independently of its depth position in relation to the viewpoint. Therefore, for every mouse movements on the screen, the following equation should be maintained at all times:

$$P\,V\,N\,M\,t_a' = \Delta\,P\,V\,N\,M\,t_a$$

where

t_a' is the new value of t_a,

Δ is the transformation matrix resulting from a mouse movement on the screen.

The homogeneous transformation matrix Δ produces a translation that correspond to a 2-D mouse movement (on an XY plane) in the normalized coordinates of the display:

$$\Delta = \begin{pmatrix} 1 & 0 & 0 & 2dx/DX \\ 0 & 1 & 0 & 2dy/DY \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

dx and *dy* are mouse movement in pixels,
DX and *DY* are the viewport dimensions in pixels.

Notice that no z translation occur in display device coordinates as a result of the mouse movement. Finally, solving for t_a' gives

$$t_a' = M^{-1}\,N^{-1}\,V^{-1}\,P^{-1}\,\Delta\,P\,V\,N\,M\,t_a$$

In general, a user will apply deformations in two conceptual steps:

1. Starting with the selection of the apex vertex and a first mouse motion to induce an initial alteration,

2. Followed by rapid rotations of the object to see the result of this first alteration and apply smaller corrections from other viewpoints before terminating.

With this methodology, not only can the user see his object from every angle but he can also apply and correct deformations from every angle in real time. Furthermore (again assuming a perspective projection), deformations of largely different magnitudes can be made with the same mouse motion by varying the surface's distance from the viewpoint. Larger deformations can be induced by moving the object farther away while minor, more detailed corrections can be made by zooming in on a focussed part of the object.

5. CENTER OF OBJECT ROTATION

As pointed out at the beginning of this paper, our Ball & Mouse methodology adheres to the "Scene in hand" metaphor (Ware & Osborne). This means that the forces and torques applied on the Spaceball induce respectively a translation and rotation movement of the object and keep the viewpoint static.

Rotating an object requires the knowledge of a reference frame that defines the center about which the object may pivot. If this center of rotation is misplaced and located far from the area where work is needed, the user has to compensate rotations with translation forces on the ball in order to keep the work area from flying out the field of view. The more he focusses on areas that are distant from the center of rotation, the more he has to fight back with translations. Therefore, when this situation arises, a feature must be present in the software to relocate the center of rotation and to bring it close to the workarea.

To implement this, we use a Ball & Mouse method analogous to moving the apex vertex in the last section. The center of rotation, displayed with the object on the screen, is translated in space with mouse movements while the Spaceball is used to move the object and see when the correct position of the center of rotation has been attained. In order that the reference frame follows mouse movements on the screen, the following equation must be kept:

$$P\,V\,N'\,M' = P\,V\,N\,M$$

where

N' is the new node matrix,
M' the new modeling matrix.

Solving for M' gives

$$M' = N'^{-1}\,V^{-1}\,P^{-1}\,P\,V\,N\,M$$
$$M' = N'^{-1}\,N\,M$$

Therefore, M' is calculated such that the node translations produced by mouse movements do not induce any movement of the object on the screen — only the Spaceball will produce object movements. The new center of rotation N' is calculated as follows:

$$[N_{tx}, N_{ty}, N_{tz}]'^T = V^{-1}\,P^{-1}\,\Delta\,P\,V\,[N_{tx}, N_{ty}, N_{tz}]^T$$

where

$[N_{tx}, N_{ty}, N_{tz}]^T$ is the old translation vector of matrix N.

$[N_{tx}, N_{ty}, N_{tz}]'^T$ is the new translation vector of matrix N.

Δ is the same mouse movement matrix as described in section 4.3.

6. CONCLUSION

In this paper, we have proposed a methodology for three-dimensional graphics interaction based on the "Ball & Mouse" metaphor. This methodology has been used for designing and implementing a 3-D sculpting program (see Fig.5). The program is already being used by artists for the development of a short film with human characters.

Most users found, after about 20-30 minutes of practice, that the use of the system was much more natural than the classical "mouse and multiple windows" metaphor. In the near future, we plan to improve our approach using a stereo display.

As our methodology applied to sculpting has proved to save time and significantly improve the communication between the artist and the program, we are now extending the approach to the specification of skeleton animation and facial animation.

Acknowledgements

Sincere thanks go to members of the computer graphics lab at EPFL for their helpful comments and suggestions on this paper and for having put this user interaction methodology to the test in the SurfMan sculpting software. This research was partially sponsored by "Le Fonds National Suisse pour la Recherche Scientifique" and the Natural Sciences and Engineering Research Council of Canada.

BIBLIOGRAPHY

Allan JB, Wyvill B & Witten IH, "A Methodology for Direct Manipulation of Polygon Meshes", New Advances in Computer Graphics, CGI Proceedings, 1989, pp. 451-469.

Bier EA, "Snap-Dragging in three dimensions", Computer Graphics 24 (2), March 1990, pp. 193-204.

Cahen O, "L'image en relief, de la photographie stéréoscopique à la vidéo 3D", 1990, MASSON.

Forrest AR, "User Interfaces for Three-Dimensional Geometric Modelling", Proceedings 1986 Workshop on Interactive 3D Graphics, ACM Press, October 1986, pp. 237-249.

Ware C & Osborne S, "Exploration and Virtual Camera Control in Virtual Three Dimensional Environments", Computer Graphics, 24 (2), pp. 175-183.

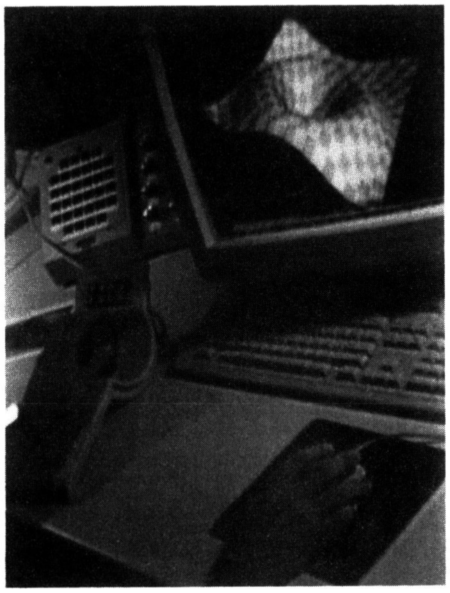
Figure 5. The use of the sculptor with the ball and mouse