

DIGITAL SANDBOX

Integrating Landform Making and Analysis for Landscape Design

ELLEN YI-LUEN DO

*Design Machine Group, University of Washington, Seattle, WA
98195-5720, USA*

Abstract. Digital Sandbox is a computational tool for landscape architecture design that provides design and analysis capabilities in the same environment. It uses image processing of hand gestures to infer the landform sculpting actions and simulates storm water accumulation over the digital terrain.

1. Introduction

1.1. A SCENARIO

Imagine you are a landscape architect. You are commissioned to design a landscape on a hill site for green space and some residences. Imagine you could start sculpting the landform in a digital sandbox. In the sandbox you can form the hills, dig out valleys, and put vegetation and buildings on different locations with your hands. You could rotate and move the sandbox around to examine the quality of space, color and texture of your design. You could move the buildings and trees to different places. To examine how the summer lighting condition would impact your design you turn on the sunlight simulation. Realizing that storm water and flooding could be a major concern on this site, you activate the water flow simulation analysis. The terrain model would display different gradients of color. This visualization shows that certain areas in your landform design would have a high volume of water accumulation on rainy days. You quickly move the buildings, and plant more trees and the problems go away.

Imagine all these imaginings are true. What would it take to build a system like this that would provide both functions of design and analysis in a single environment? In this paper we describe a prototype system, called the Digital Sandbox, to support landscape design tasks. The Digital

Sandbox system integrates a quantitative model of an ecological process with a spatial model of a digital landform. It provides designers the ability to design a landform and simultaneously analyze the storm water flow over that landform. The system employs a gesture-based modeling interface for a designer to manipulate the spatial model using hand gestures in three-dimensional space.

1.2. BACKGROUND: WHY BUILD A DIGITAL SANDBOX?

Human hands have dramatically changed the earth's landscape over the past several hundred years. Landscape design is a complex task with multiple goals and constraints. It involves both reasoning of form and function. The landform design in three-dimensional space requires an understanding of the structure and the function of the land. Landscape architects must understand the ecological impacts of their designs as well as the formal aspects. To achieve these multiple goals, designers often employ a variety of tools from hand drawn sketches to simulation models. All these tools are indispensable to designers. However, in the design process, designers would like to move back and forth seamlessly from qualitative landform making to quantitative analysis without interruption. Designers would like to have the right tool at the right time in the same work environment. No such ideal tool exists—yet. Currently there is no landform design tool that provides design and analysis in the same environment. Designers often use sketches and cardboard models to explore the spatial qualities of a design, then use digital tools such as Geographic Information Systems (GIS) to perform ecological analysis.

We built the Digital Sandbox system to provide designers a single environment where they can sculpt a landform while obtaining feedback of their design. We call the system Digital Sandbox because it is digital, to compute ecological simulation, and it is also like a traditional sandbox, that is easy to interact with by hand. This paper is organized as follows. Section 2 reviews related work in design computing work in digital terrain modeling, gesture-based modeling methods, and research on integrating design and analysis tasks in computational environments. Section 3 provides an overview of the Digital Sandbox implementation. Section 4 describes a design use scenario of the Digital Sandbox. The paper concludes in Section 5 with a brief discussion and identifies future research directions of this new landscape design tool.

2. Related Work

The building of a digital sandbox concerns three areas in design computing: digital landform making, gesture-based modeling, and the integration of design and analysis tasks in computational environments.

To better position the Digital Sandbox, and to identify conceptual issues and methods for better earth-forming tools, this section reviews related work in these three categories.

2.1. DIGITAL LANDFORM MAKING

Landform making is fundamental to landscape architecture. Developing new terrain design tools is difficult because the representation and manipulation of topology is inherently complex (Sawyer 1998). However, there have been few innovations in the digital tools for such design task. Among these efforts, the Topographic Surface Sculptor (TSS) (Westort 1998) and Leveller™ (DaylonGraphics 2001) both utilize procedural metaphors to provide better support for creative landscape forming. Existing CAD-based tools require the designer to digitize contours to create mesh landform models. This process is tedious and time consuming. To ease this process, TSS uses a bulldozer as metaphor for landform operations (Westort 1996). It consists of three parts: a blade tool to sculpt an excavation path, a library of geometric shapes, and landform operations. A designer can select a bulldozer blade, and adjusts its shape to create a digital landform. The transformation of the digital surface is executed through the designer-defined path, operated with a given blade and shape.

A commercial height field modeler developed by Daylon Graphics called Leveller™ takes a similar but different approach. It is a raster-based paint program. The interface is a grid surface of pixels. Different colors represent different elevation heights. A designer can modify the terrain model by editing the height field using a paintbrush function, similar to using a conventional paint program. The system provides a variety of brush sizes and 'dig-and-raise' tools to excavate and modify the terrain. The plan view windows provide the ability to edit the value of the field heights. A perspective window displays the perspective of the resulting terrain.

These systems use metaphors such as 'bulldozer' and 'paint' to make it easier for a designer to understand the operations on a digital terrain map. By providing more construction-like methods for terrain manipulations, these systems improve over current CAD tools.

2.2. GESTURE-BASED MODELING

Current trends see the emergence of a number of gesture-based interfaces for design applications. These systems use image processing and pattern matching of gestures to pointing or modeling operations in 3D modeling. For example, GestureVR is a vision-based input interface for three-dimensional object creation and manipulation (Segen and Kumar 1998).

It is a single-handed gesture-recognition system with input from two video cameras. The system does not require the user to wear a data glove for gesture detection. Instead, it employs two cameras placed about three feet above the gesture space to capture the plan and profile views of user's hand gestures. By computing and processing images from the image input the system determines the spatial location and configuration of the user's hand. The information of the 3D position (x, y, and z location) of the fingertip and the azimuth and elevation angles of the finger's axis is used to draw curves and input digital objects to the three-dimensional scene by selecting from a tool palette on the screen. Each gesture is mapped to a command. For example, the user can draw a line with an index finger, and make a fist gesture to indicate a selection in Segen and Kumar's '3D Scene Composer' environment.

The Two-handed Gesture Environment Shell (Nishino, et al. 1997), on the other hand, uses an instrumented data glove to interact with the virtual environment. Unlike Gesture VR, it is not vision-based. Instead, input from the data gloves is parsed through a driver for initial processing. The commands triggered by the hand gestures are identified through the shape, position, and orientation information from the data stream. The system employs pattern matching by a neural network to process dynamic hand gestures. A similar approach by Wexelblat places sensors at various points on the user's hands and upper body (Wexelblat 1995). He argues that a model of the user's body is necessary to understand human's natural gesture in a furniture layout design task. The system has two major components. First, the analyzer receives and analyzes input from the body model and sends the data to the interpreter. The interpreter then uses gesture input to add or move furniture in the virtual room. Gesture modeling is also used in artistic sculpturing to create 3D shapes. Surface Drawing (Schkolne, et al. 2001) uses a semi-immersive virtual environment called Responsive Workbench (Kruger and Frohlich 1994). The system combines the use of stereoscopic glasses and a data glove to create three-dimensional organic shapes. Users of the system can use hand movements and strokes in a 3D space to create digital objects. In addition to gestures, the system also provides a variety of tangible tools to operate upon objects. For example, a user can use tongs to move and rotate the drawing, or an eraser to remove portions of the drawing. All viewing and interactions occur in user's physical space.

These gesture-based modeling tools have great potential to be used in landform design. The ideal tool would integrate the creation of 3D geometry with analysis abilities.

2.3. INTEGRATING DESIGN AND ANALYSIS TASKS

To be useful for designers, computational environments should provide both design and analysis functionality. Many research projects have attempted this integration by providing a more visual interface to geographic information data, or to bring CAD and GIS applications into the same environment.

One such example of a hybrid GIS-CAD application was developed by Mayall et al. (Mayall, et al. 1994). To assess the visual impact of a proposed landscape design, a designer can input attributes such as building heights, colors, vegetation type and size to the GIS database and the system would invoke its display in a three-dimensional modeling environment of a CAD program. The system uses a simple scripting language to map and translate the GIS data into three-dimensional landscape objects in CAD representations. A project by Ervin (Ervin 1992) also creates 3D models from GIS data. It associates unique land use codes in the GIS database with a predefined library of object symbols – buildings, houses, vegetation, and paved surfaces. These systems are improvements over the existing tools because they link visualization and analysis together. However, designer can only make design changes on the GIS database, not in the CAD model. Often there is a significant lag time between analysis and visualization.

There are also projects focusing on visualization systems and providing interactive feedback. For example, the SmartForest project (Uusitalo, et al. 1997) provides real time update of the visual display of design changes. The system provides dialog boxes for user to change the growth, crowding, and thinning parameters of tree type, size and health in different growth models. These changes update the underlying database of forest composition and produce a new visualization display of the forest landscape environment. Similarly, the Toronto University Urban Modeling System (Danahy and Wright 1988) also provides user real-time feedback about design impact. Users can import a parcel file, and choose from pull-down menus to assign spatial zoning constraints and add buildings to the parcel. Design factors such as floor-space index and construction cost would then be displayed in a spreadsheet window. It provides numeric information display of the design impact after a designer's manipulation of the design.

These projects illustrate the efforts toward, and the continuing need for, integrating design and analysis tools to support better decision making in landscape design.

3. Implementation of the Digital Sandbox

In Spring 2001 we built a working prototype of the Digital Sandbox system. The Digital Sandbox is a gesture-based interface that integrates design and analysis tasks to support landscape design. A designer can use the Digital Sandbox to sculpt a digital terrain, add trees and buildings to the landform, and run a storm water accumulation model to predict environmental impacts. In the design process, designer can engage in landform sculpting, addition and manipulation of the landscape objects, as well as running the water flow simulation as often as needed, in a single work environment. These design tasks are executed by mapping hand gestures to specific predefined commands. The prototype system was built to explore the viability of such an application in digital land forming.

3.1. SYSTEM OVERVIEW

There are two major parts of the Digital Sandbox system. First, it uses a machine-vision approach called Gesture Modeling (Gross and Kemp 2001), developed at our research lab, the Design Machine Group (DMG 2001). The second part of the system is the SandboxToy module, also developed at our research lab. This module consists of a set of object classes to accept changes of attributes and methods applied to them to compute and display the water flow simulation. The system building was reported as part of a Master of Landscape Architecture thesis by Robert M. Harris (Harris 2001).

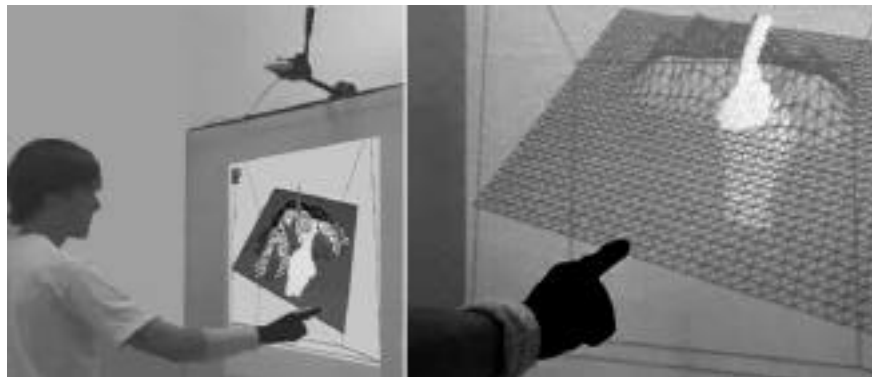


Figure 1. Designer interacting with the Digital Sandbox

The physical setup of the system has two primary components: a gesture modeling space and a 3D modeling display. The 3D modeling space is displayed on a translucent screen with rear projection of horizontally inverted image. The gesture modeling space is defined under

a simple, desktop video camera positioned approximately three feet above a white tabletop in front of the modeling screen Figure 1 shows a designer interacting with the Digital Sandbox.

To sculpt a landform mesh, the designer wears a black glove and gestures under the camera. With the contrast between the black glove and the white background, the system can easily identify the user's gestures. There is no sensor or instrumentation embedded in the glove.

Raw images captured by the camera are transmitted continuously to the computer and passed on to Vision SDK 1.2, a low-level library of methods to acquire live images. These gesture images data are compared to known image templates to find appropriate matches and to issue commands to graphic rendering implemented in Microsoft Direct3D. Figure 2 shows the two program windows on display. The large screen displays a simple three-dimensional scene for the modeling actions, and the smaller gesture recognition window displays the image captured by the camera.

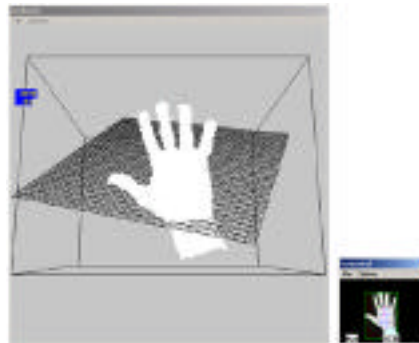


Figure 2. 3D modeling space (left) and gesture-recognition window (right)

The Digital Sandbox is written in C++. The system consists of several classes; each has a group of objects. These objects contain attributes and methods to operate upon these attributes. For example, the size, color, and shape are the attributes for an object. Object behavior is implemented through defined methods of operation such as changing the size, color, or shape under certain conditions. Each object in the system encapsulates attributes and behaviors defined by its classes. Each class thus functions independently of others. As is customary in object oriented systems, class inheritance enables each object to inherit properties from its parent classes and thus only class specific features need to be modified and defined. Thanks to this class hierarchy and individual object definitions, a module of the program can be tested and expanded easily. At present the Digital Sandbox only has a water simulation function, but using this

infrastructure, we could add other dynamic ecological models (e.g., light, wind, growth, and fire) in the future.

Currently there are two facets of the Digital Sandbox. Gesture Modeling deals with how mesh geometry is modeled with gesture commands. The SandboxToy supplies the attributes and behaviors for water accumulation and flow model simulation. Two general class categories were developed for each of these functions. The following subsections discuss these implementations.

3.2. IMPLEMENTATION OF GESTURE MODELING

3.2.1. Initial Image Processing

Input from the camera is a 160 by 120 bitmap image. An intensity level is set as threshold to isolate the glove from the background. Each pixel in the image is compared to the threshold value. When the intensity value is less than the threshold, it is replaced by a white pixel. Intensities above the threshold value are replaced with black pixels. The result of threshold operations is a black silhouette of the glove (hand) against a white background. The system then draws a bounding box to contain all areas of black pixels to distinguish the hand from the rest of the raw image captured by the camera.

3.2.2. Mapping of Gestures to Commands

Currently the system recognizes seven different gestures. These seven examples shown in Figure 3 capture the essence of generic gestures for 3D manipulations. They are point, thumbs up, two (v symbol), five (open palm), fist, gun, and pinch. These gestures suffice for most landscaping operations. Certainly more gestures can be added to the system. However, adding more items might increase the user's cognitive load (Hix and Hartson 1993) and thus reduce usability.

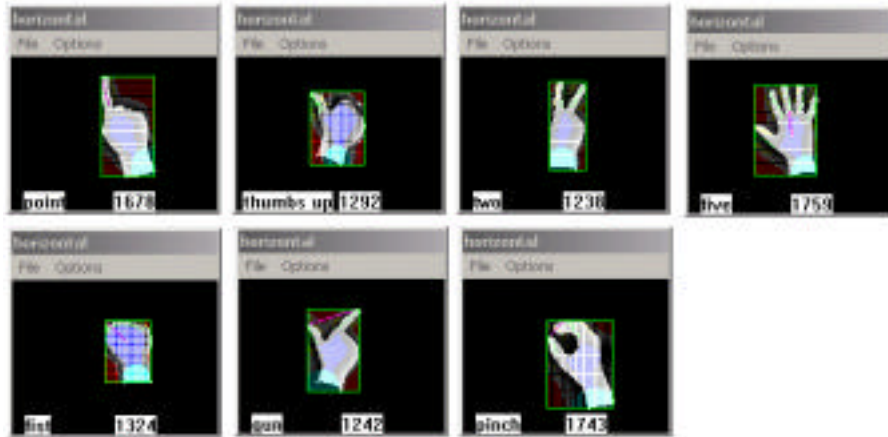


Figure 3. Gestures recognized by the Digital Sandbox.

The system uses gesture commands to add, edit, move, and analyze three-dimensional geometry. The association of gestures with commands is the basis of the system. Each input gesture in the system is associated with a command to execute an action in the digital environment. Each gesture is mapped to an operational command as shown in Table 1. Designers can use these gestures to perform specific landscape forming tasks. For example, a "point" gesture can deform the mesh to sculpt a digital landform. When the "point" gesture is recognized, the system finds the coordinates of the index finger from the image, and identifies the intersection point. When the designer's index finger moves up, this point acts as the control to deform the mesh and creates a peak or a hill.

TABLE 1. The mapping of gestures to commands for the Digital Sandbox

Gesture	Command
Point	Deforms the digital mesh terrain.
Thumbs up	Activates the storm water accumulation simulation.
Two	Adds a building (a cube).
Five	Selects a building (when intersected with the hand gesture).
Fist	Moves a building.
Gun	Adds a tree (a cone above a cylinder).
Pinch	Selects and moves a tree.

3.2.3. Pattern matching with template images

The initial image processing determines the location of the hand in the three-dimensional space. The position of the bounding box determines the x and y coordinates of the hand. The size of the bounding box determines the depth or z coordinate. The closer the hand to the camera, the larger its image becomes. After initial processing of raw images, the second part of the gesture recognition process performs pattern matching with known template images.

The system has a set of pre-stored template images for pattern recognition. Once the shape of the hand gesture is distinguished from the background and contained with a bounding box, it is compared to a set of template images to find the closest match. Figure 4 shows some of the templates images stored by the system.

Each template is a spatial map of image features such as segments, edges and lines. The templates of the hand gestures are constructed (currently through manual editing) using images captured by the camera. To facilitate easy comparison, important spatial features are identified with different colors. The background white pixels (empty space) are replaced by red pixels. The black pixels indicating palm area are replaced by blue ones. The finger axes are drawn as green lines. These axes are further outlined with yellow lines to indicate finger boundaries. The wrist and arm areas are colored cyan. The region of gray pixels remains unmodified as buffer are for pattern-matching.

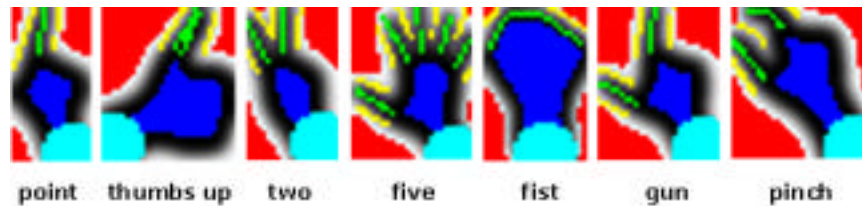


Figure 4. A sample of template images for pattern recognition.

The pattern-matching process involves two classes: `KnownImage` and `KnownImageSet`. The template images are objects that belong to the `KnownImage` class. The class functions as a data structure and uses a data type from the `VisionSDK` library to store dimensions and spatial structure for each image. The `KnownImageSet` class compares the template images to input images from the camera in real time. The system performs image processing on a per-frame basis. To determine which template is the closest match, a similarity score is generated as each input image is compared to the templates. Matching on each image frame produces

three best matching templates. The system then selects the gesture that matches the largest number of the last several frames' top-ranked templates.

Template matching includes three major processes. Each of these processes is a test to determine the next step of comparison. These test conditions are a set of if/then rules. First the system compares aspect ratio of the images. A template is considered an inappropriate match if the aspect ratio of the images differs by more than a constant amount. The template is discarded and comparison stops. The next template is then considered, starting the comparison process over. The second step compares spatial features. The red and blue pixels in the template are compared to corresponding white and black pixels in the input image. Gray pixels are compared as well. If the percentages of each color composition are lower than a certain value, the template is not a match. Otherwise, the matching score is incremented by an amount proportional to the percentage of matching pixels. Finally, a similar process compares green and yellow pixels and checks against the template's eight nearest neighbors. The percentage of green pixels with black neighbors and yellow pixels with white neighbors are calculated. The score is then further incremented by an amount proportional to these percentages. The gray area serves as the buffer zone to provide possible match when there are variations in the two images.

The system proceeds sequentially through each stored template to form a list of potential matches in a ranked order. The system then selects the top-ranked gesture template of the last five frames and replaces the last one in the list with this new gesture. The most frequent gesture in the list is then considered to be the gesture for the current frame. This five-frame "buffer" approach is taken to provide stability to the system. By disregarding short-term (fewer than 5 frames) candidates, the system avoids a 'flicker' of wrong match fluctuations.

3.3. IMPLEMENTATION OF THE SANDBOX TOYS

A set of classes was developed to represent a traditional, physical sandbox. A mesh class represents the 'sand' in the sandbox. Landscape objects are toys made of by simple geometric primitives (e.g., cones, cylinders, and cubes).

There is also a separate class for the sandbox application. Each object has its own methods and interacts with each other to compute the storm water accumulation model. These toy objects are also embedded with attributes of their real-world counterparts (e.g., trees intercept water). Below we discuss these classes, their attributes and methods.

3.3.1. TheSandbox

The top-level control structure of the system is TheSandbox class. TheSandbox class directs the user interactions and creation of new toys. The main components of TheSandbox are the mesh and the active toy objects in the sandbox. The attributes for this class includes (1) *Toy* - an array listing currently active SandboxToys, (2) *CurrentlyPlaying* - a variable indicating, if any of the toys in the sandbox are being manipulated, (3) *ActiveToy* - A pointer to the current active toy.

There are four methods for this class: (1) *CreateNewToy* - produces a SandboxToy object (i.e., a tree or a building). (2) *AddToyToSandbox* - adds the newly created toy to the Toy array. (3) *InteractWithToys* - directs interaction, executes commands based on the recognized gestures to create, select and reposition toys in the sandbox, deform the terrain mesh, and activate the storm water accumulation simulation. (4) *IntroduceTheNeighbors* - finds each toy's position and its location on the mesh to decide and to modify the interception or infiltration of the mesh beneath the toy.

3.3.2. SandboxToy

The SandboxToy class is the parent class for Mesh, Tree and Building classes. Figure 5 shows that the SandboxToy class is the base (parent) for Mesh, Tree and Building. The Mesh contains MeshTriangle(s). The Tree contains TrunkType and CanopyType classes.

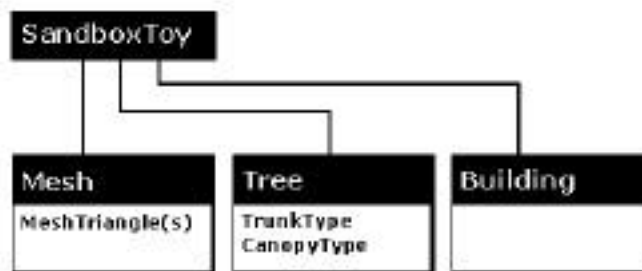


Figure 5. Inheritance relationship of the sandbox toys.

The SandboxToy class has two attributes: (1) *ID*, to identify the toy type - a Mesh, Tree, or Building. (2) *PlayMode*, to signify the current status of the toy (active or not). The method for the SandboxToy is *Interact* - to direct toy interactions based on recognized gestures. Each

derived class (Mesh, Tree, Building) has its own implementation of the method.

3.3.3. Mesh

The Mesh class makes up the digital sand in the sandbox. The Mesh constitutes the terrain surface. This mesh surface is the base for landform sculpting. It is therefore a permanent object that can not be created or destroyed by the user. The mesh is formed by a series of MeshTriangles. These MeshTriangles are stored and organized into an array of rows and columns. The Mesh class is also embedded with the storm water accumulation model so it can display itself according to the simulation results.

The Mesh deforms itself in response to gesture input, allowing the user to sculpt the surface. It also displays itself according to storm water accumulation model when the user initiates the simulation command. Deformation of the Mesh is achieved by modifying the z-coordinates of its component triangles. This calculation is expressed through the Gaussian equation,

$$f(x) = \left(\frac{1}{2\sigma} \right)^{1/2} e^{-\left(\frac{x - \mu}{2\sigma} \right)^2} \quad (1)$$

where μ is the control point where the finger touches the mesh, the midpoint of the progression, and σ is the variance. The σ value controls the shape of the deformation. Figure 6 shows the Gaussian equation distribution, and the σ value decides the appropriate number of points to be moved around μ . Currently, the system has a hard-coded value of 0.2. The goal is to have the deforming curve slope gently. The σ value was derived from the trials that produce not excessively pointed or flat curves. It could also be a user set value.

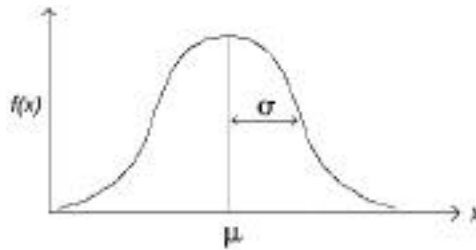


Figure 6. Mesh deformation is calculated according to Gaussian equation.

The storm water accumulation model was designed to predict flow paths across the digital mesh surface of the sandbox. There are complex approaches for modeling storm water accumulation (e.g., (Amer and Blais

2000)) that route flow through structural features of the terrain (e.g., ridges, channels, and valleys). A simpler approach was adopted for the Sandbox. Each discrete mesh unit is treated as an independent component. Each mesh unit receives water from all sources, route flow across itself and passes water on to the neighboring surfaces. This approach is similar to the Triangulated Irregular Network approach (Silfer, et al. 1987). The model uses local rules and the topological relationships between adjacent triangles to route flow across the network of triangles. Two adjacent mesh triangles are joined at their edges. Boundary conditions for these triangles are thus relatively easy to specify. Therefore, to derive local rules in a triangulated model to specify the flow between triangles is easier than using a point-based grid Digital Elevation Model (Wilson and Gallant 2000).

Using the point as the component unit has one major problem for calculating the storm water accumulation model because the adjacent surfaces created from point matrices may not always be coplanar. These surfaces are also potentially more complex. In contrast, a surface derived from triangular planes guarantees adjacent triangulated units of the mesh will always be coplanar. Therefore, all possible boundary conditions for every triangle in the model are easier to handle. To route flow across the mesh the model only needs to define flow across an individual triangle.

Computing the storm water accumulation model involves several steps. First, all mesh triangles are sorted according to their maximum heights. The runoff calculation starts from the top – the triangle with the highest elevation. Next, the runoff is routed from the triangle to the nearest downhill neighbor. This is determined by calculating the orientation of the line of steepest descent. Flow is routed based on the orientation of this line, to one of six possible directions. Figure 7 shows these six directions intersect either the triangle's three adjacent faces or one of its three corners. The next procedure is to check whether this neighbor has lower elevation to the triangle. If it is lower than the current triangle, then the flow is routed to it. If not, the system finds the next lowest triangle along the line of steepest descent from a wider neighborhood. When a lower neighbor is found, flow is routed to it. If no lower neighbor is found, the triangle is regarded as a sink that absorbs the flow (instead of routing it). When neighboring triangles are facing each other, the slope line for flow routing is computed by averaging their aspects. The model proceeds step-wise to the next triangle. The procedure is repeated until all triangles in the sorted list are exhausted. These simple, local rules direct the routing of the flow across the entire mesh surface from the highest to the lowest.

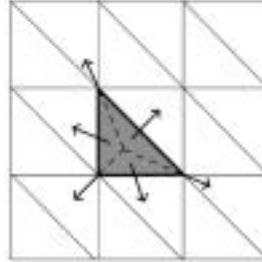


Figure 7. Flow can be routed to any of the six possible directions in a triangle.

3.3.4. MeshTriangle

The Mesh is subdivided into, and consists of MeshTriangle class objects. Each mesh triangle can perform several tasks. It can modify its vertices when the mesh is deformed. It calculates and updates its gradient, aspect, and elevation. It also computes its storm water runoff; displays its color and fill mode based on the runoff value. The equation for storm water runoff calculation is based on the following model (Burrough and McDonnell. 1998),

$$\text{Runoff} = \text{Input} - \text{Interception} - \text{Infiltration} - \text{Evaporation} + \text{Runon} \quad (2)$$

where Runon is the runoff quantity routed to the triangle by its uphill neighbor. The values used by the model are currently hard-coded into the system. The system could easily provide a dialog box for the user to explore and adjust different numeric values for the model.

The fill mode and color for each mesh triangle is determined by its computed runoff value. If the runoff exceeds a threshold value the triangle is painted with a color. If the value is below the threshold, then the triangle remains in wire-frame mode and displays no color. Currently the threshold value is hard-coded into the system. The user could also specify this value when an input interface is provided. The color for the triangle is determined according to a gradient color scheme. Lower runoff values are represented with less saturated colors, while higher-saturation colors correspond to higher runoff values.

There are several important components for this MeshTriangle class. The attributes for this class includes (1) *Elevation*, the height of the triangle; (2) *Aspect*, orientation of the steepest descent line for the triangle; (3) *Gradient*, rate of elevation change across a triangle; (4) *InputPrecipitation*, precipitation quantity; (5) *Evaporation*, water evaporation, not set for the current runoff model but would be useful for future module of atmospheric or ecological simulation model; (6)

Interception, precipitation intercepted by objects located above the triangle; (7) *Infiltration*, amount of precipitation infiltration for the triangle; (8) *RunoffFromNeighbors*; runoff volume received from neighbors, and (9) *RunoffToNeighbors*, amount of runoff to be routed to a neighbor.

3.3.5. Tree

The Tree class contains a TrunkType and a CanopyType. It is a class derived from the SandboxToy class. The Tree class determines attributes of its trunk and canopy. It uses these attributes to intercept rainfall water. Currently there is only one tree type available for the Digital Sandbox. However, the current framework is extensible to accommodate multiple types of trees. In the future implementation, each tree type could potentially interact differently with the terrain mesh by intercepting different quantities of rainfall water (just like the real trees they simulate). The attributes of this class include (1) *Trunk*, and (2) *Canopy*. The methods for this class include (1) *Interact*, move position according to recognized gesture, and (2) *SnaptoGrid*, that ensure the tree is always positioned on the surface of a mesh triangle instead of floating in space. The TrunkType class and CanopyType class both maintain their dimensions and can be moved by the user. Future extension could contain different methods and attributes to specify different water interception behaviors of the trunk and canopy types.

3.3.6. Building

The Building class is also a descendant of the SandboxToy. Buildings are typically impermeable to rainfall. Therefore, when a Building object is positioned on a mesh triangle, it decreases the infiltration values of mesh triangles located downhill. Like Tree, Building also uses methods to maintain its own dimensions and location in the sandbox. The current *Interact* method for Building is more complex than the Tree's method. Two gestures are involved to manipulate the building. The gesture "five" is used to select and deselect a building, and "fist" grabs the building and moves the building around. Therefore, the *Interact* method responds to a sequence of recent gestures (instead of one) to determine the current play mode status. For example, the user may be selecting the building ("five" followed by "fist"), releasing the building ("fist" followed by "five"), or moving the building ("fist").

4. Interacting with the Digital Sandbox

The goal of the Digital Sandbox project is to create a new type of tool to facilitate a more straightforward and direct process for landscape design. The Digital Sandbox project takes advantage of image processing for gesture recognition, and object oriented programming for embedding object behaviors and dynamic simulations. The idea is to enable designers to use simple freehand gestures to create and edit a digital landform. It is also intended to bridge the separation between design and analysis tasks. The Digital Sandbox system provides the ability for a designer to sculpt a landform, activate a storm water accumulation model simulation, and display the result on the landform, all within the same work environment.

Let's come back to consider the design scenario described earlier. The following paragraphs describe the sequence of interactions in a design process within the Digital Sandbox.

A landscape architect might begin a design session wearing the black glove and gesturing in the input space under the desktop camera. The system provides a flat mesh at ground zero for sculpting interaction. The model space on the screen displays the gestures captured by the camera. Figure 9 shows the designer deforming the mesh terrain by moving a hand with index finger extended. Unlike normal CAD applications that require the input of contour lines before any terrain can be formed, simple gestures are enough to sculpt the mesh in the Digital Sandbox.

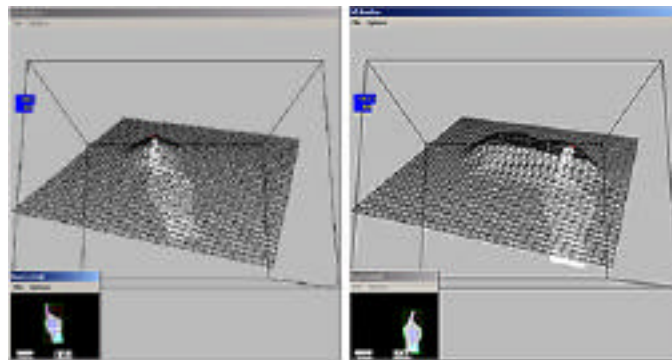


Figure 8. The “point” gesture sculpts and deforms the digital mesh. Left: initial sculpting. Right: a hill is generated after continuous “point” command.

A designer can create peaks and hills by pointing and deforming the digital mesh model. This process is very intuitive. Informal user feedback shows that the majority of users (designers, high school students, etc) can walk up to and use the point gesture to sculpt landform without formal

training. The designer can analyze the landform design at any point in the design process. The simulation of storm water flow on the site can be activated by simply makes a “thumbs up” gesture. Figure 9 shows the "point" gesture being used to sculpt the landform, and the "thumbs up" gesture initiates the storm water accumulation model. The system then computes the elevations and orientations of each mesh triangle and routes flow across the terrain surface. The accumulation of water is displayed by a gradient of colors on the mesh triangles. More highly saturated color indicates higher runoff volume. The less saturated colors represent lower runoff values. The white mesh triangles indicate the regions that do not exceed the runoff value threshold.

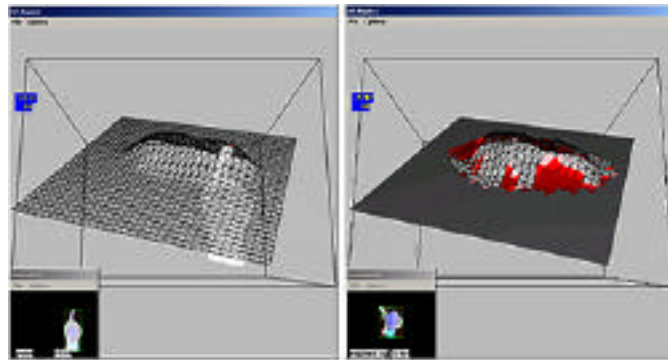


Figure 9. The “thumbs up” gesture initiates the storm water accumulation model. Left: original mesh model being sculpted. Right: mesh with painted color showing storm water values resulting from the water accumulation analysis.

The system painted all mesh triangles whose runoff volume exceeds a certain threshold value (indicative of erosion danger). The colored mesh triangles indicate the locations for flow paths. This means the simulation result is displayed directly on the landform, in the design environment. Each triangle is painted according to the water accumulation value it derives. Having analysis displayed on the terrain mesh enables the designer to visualize the flow and relative concentration areas. With this information the designer can make better assessment about the design impact on the ecological system of the landscape.

To redirect the storm water flow, the designer can continue to sculpt the mesh. The iterative cycle of adjusting the landform and analyzing flow paths can be carried out repeatedly. The designer can also place some trees to intercept the rainfall without changing the landform. To change the concentration of water on various parts of the site, the designer can add trees to the landform uphill from the areas of concern. Figure 10 shows the “gun” gesture being used to add a single tree to the

landform, and the “pinch” gesture being used to move and reposition the tree to a new location.

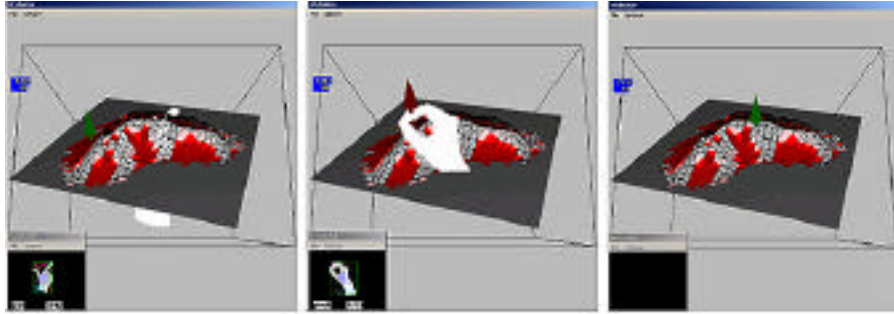


Figure 10. Creating, selecting and moving a tree. Left: the designer uses the “gun” gesture to create a tree. Middle: the “pinch” gesture grabs and moves the tree. Right: the tree is released to new location when no gesture is recognized.

To create additional trees, the designer can repeat this process. The designer can run the storm water simulation model again after placing the trees on the site. Each tree intercepts rainfall, and thus decreases the storm water accumulation on the areas immediately below it. Figure 11 shows that a cluster of trees in a particular area would significantly decrease surface flow in the area below.

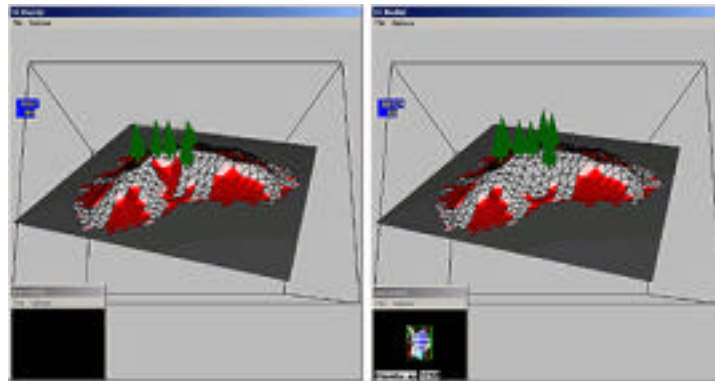


Figure 11. More trees decrease storm water accumulation on the area immediately below. Left: the model with trees. Right: tree clusters cause the disappearance of colored triangles immediately below.

To add buildings to the sandbox terrain, the designer uses a similar method. Figure 12 shows the gesture “two” being used to create a simple cube shaped building.

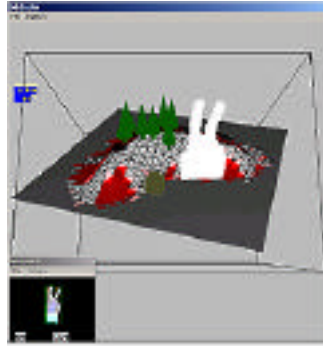


Figure 12. The “two” gesture creates a building.

Figure 13 shows the process of moving a building. The designer uses a combination of “five” and “fist” gestures to move the building. First, the “five” gesture is used to select the building. When the designer's hand intersects with the building it changes color to signify its position and selection status. A quickly formed “fist” gesture grabs the building. Once the building is grabbed, it follows the hand, and can be moved to a new location on the landform. To release the building, the designer makes the gesture “five” to deselect the building.

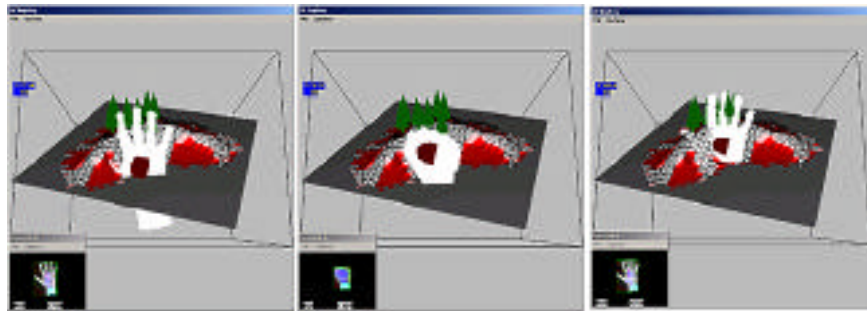


Figure 13. Left: the “five” gesture selects the building. Middle: the “fist” gesture grabs and moves the building. Right: a “five” gesture after “fist” released the building to new position.

Figure 14 shows the storm water simulation showing (before and after adding a building) how the building affects the runoff values of the area immediately downhill. Buildings are impermeable to rain fall. Therefore, the existence of a building would increase storm water accumulation in the area immediately below it.

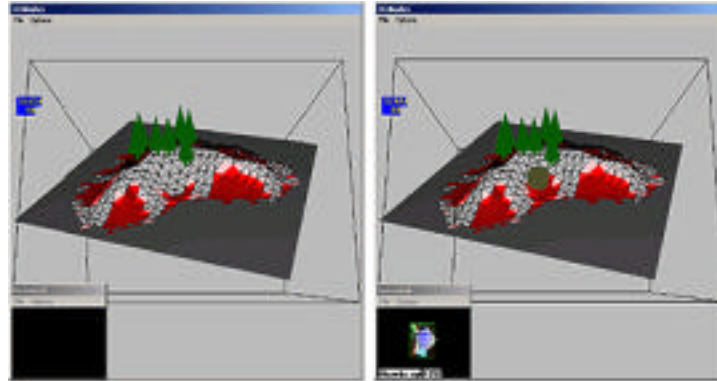


Figure 14. A building increase storm water accumulation on the area immediately below. Left: water runoff on the terrain. Right: more colored mesh appears immediately below the building indicating more water accumulation value.

The above scenario illustrates how the Digital Sandbox supports integration of design and analysis tasks in an environment for digital landscape forming. By providing the geometric sculpting and simulation analysis in a single tool, the iterative processes of design and analysis are better supported. The system provides designers the ability to use hand gestures to sculpt terrain in space, create objects like trees and buildings and add them to the landform, and activate a storm water accumulation model to see the design impact.

5. Discussion and Future Work

In sum, the Digital Sandbox is motivated by the need to integrate form making and ecological functions in the same environment. Current landscape design practice employs different tools for different tasks. These tools are useful but they do not support a seamless transition between design and analysis tasks. Digital Sandbox enables the designer to employ hand gestures to design a landform and to analyze the storm water flow anytime during the design process.

The inspiration for the design of the Digital Sandbox comes from the physical sandbox. The traditional sandbox provides an intuitive environment in which to play and explore landform design. The hands-on experience with sand helps people, children and adults alike, understand the nature and the structure of landscape forming. A computational environment ought to have similar characteristics in order to provide better human computer interactions.

The Digital Sandbox system uses an object-oriented framework for implementation, and to accommodate possible future extension. A set of

object classes has been developed to represent real-life objects such as trees and buildings. Each class is embedded with behaviors and attributes to keep track of its interactions with others, current location, computation and display of the storm water accumulation model. Designers can manipulate these objects and issue commands using hand gestures in three-dimensional space. Each gesture is mapped to one particular command. This gesture-command association enables interaction with landform without having to stop to type or find a pull down menu. The designer can sculpt a digital landform, add trees and buildings, and run the storm water accumulation model to display the environmental impact to the landform. The results of the storm water flow analysis are painted directly on the landform mesh triangles. This instant visual feedback on the same environment could empower the designer to make more informed decisions without having to translate data to different platforms or to change the focus to a different task.

Researchers in artificial intelligence have argued over two different research methodologies (Boden 1995, Schank and Childers 1984). Research projects are classified as focusing on advancement of either intelligent applications, or theories of human mind. The Digital Sandbox project addresses the task-oriented approach of AI research. It is a task-directed technological approach. We are concerned with the technology of getting computers to do things that people already do. It responds to demands of a particular task. Designers regularly perform complex tasks such as design and analysis interchangeably in a design session.

There are several research directions for future work. One obvious next step is to add more simulation models and the ability to import existing data sources, such as Digital Elevation Models into the Digital Sandbox system. The goal would be to use the tool on an actual site with existing topography. We would like to add the capabilities of adding multiple trees and buildings as well as different types of landscape objects that would respond differently to various ecological models, for example, trees with different water infiltration rates or roadwork that provide different permeability and drainage functions. It would also be important to make the interactions more realistic and practical. For example, the landform model can be embedded with physical constraints such as slope limits or environmentally sensitive area boundaries. Connection to a spatial data engine such as a conventional Geographic Information System would allow more parametric manipulations of the behaviors and attributes of the sandbox toys. We could investigate making the modeling space displayed in three-dimensional holographic space instead of the 2D screen. Adding more cameras as inputs could potentially provide more operational opportunities and help calibrate image recognition. For example, with the ability to capture the hand gestures in both plan and

profile view could enable the calculation of roll, pitch, and yaw for greater degrees of freedom in manipulating the model. However, we still wish to keep the interface simple and inexpensive (not using an instrumented glove or head mounted display). Another possible direction is to embed physical media with the intelligence of a Digital Sandbox. For example, tactile feedback could be conveyed through physical sand that is tangible while the simulation model is displayed onto the physical terrain model to provide analysis feedback.

We are exploring building a gesture-command mapping module for users to input new gesture templates and to associate them with intended operational commands. For example, a designer could give examples to train the system to recognize new gesture templates. Different designers might create their personal preference for gestures and commands. Wearing the black glove is a minor inconvenience, though it is more economical and less cumbersome than the data glove and head mounted display set-up that other research projects require. It would also be useful to develop more complex image-processing algorithms capable of detecting the user's hand without the user needing to wear a glove to ensure contrast to the background.

Acknowledgements

A more complete description of the Digital Sandbox project can be found in Robert M. Harris's master of landscape architecture thesis. This research is based upon work supported in part by the National Science Foundation under Grant No. IIS-00-96138. The views contained in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

- Amer, A. and J. Blais, 2000, *Object-oriented Digital Terrain Modeling for Geomatics Applications*, *Geomatica*, **54** (1) 43-52.
- Boden, M. A., 1995, *AI's Half-Century*, *AI Magazine*, **16** (4) 96-99.
- Burrough, P. and R. McDonnell., 1998, *Principles of Geographical Information Systems*, Oxford University Press, Oxford.
- Danahy, J. and R. Wright, 1988, *Exploring Design Through 3-Dimensional Simulations*, *Landscape Architecture*, **78** (4) 64-71.
- DaylonGraphics, 2001, <http://www.daylongraphics.com/products/leveller>, Leveller Heightfield Editor
- DMG, 2001, Design Machine Group, <http://dmg.caup.washington.edu>, Design Machine Group, University of Washington
- Ervin, S., 1992, *Integrating Visual and Environmental Analyses in Site Planning and Design*, *GIS World*, 26-30.

- Gross, M. D. and A. Kemp, 2001, Gesture Modeling: Using Video to Capture Freehand Modeling Commands, *CAAD Futures 2001*. B. d. Vries, J. P. v. Leeuwen and H. H. Achten, Kluwer Academic Publishers, Eindhoven: 33-46.
- Harris, R. M., 2001, *The Digital Sandbox: Integrating Design and Analysis in a New Digital Earth-forming Tool*, Master Thesis, Landscape Architecture, University of Washington, 71.
- Hix, D. and H. R. Hartson, 1993, *Developing User Interfaces - Ensuring Usability Through Product & Process*, John Wiley & Sons, New York.
- Kruger, W. and B. Frohlich, 1994, *The Responsive Workbench*, IEEE Computer Graphics and Applications, (May) 12-15.
- Mayall, K., G. Hall and T. Seebohm, 1994, *Integrate GIS and CAD to Visualize Landscape Change*, GIS World, 7 (9) 46-49.
- Nishino, H., K. Utsumiya, K. Yoshioka and K. Korida, 1997, *Interactive Two-handed Gesture Interface in 3D Virtual Environments*, ACM Symposium on Virtual Reality Software and Technology, 1-8.
- Sawyer, C., 1998, *Representing Landscapes Digitally*, Landscape Australia, 20 (2) 128-132.
- Schank, R. C. and P. G. Childers, 1984, *The Cognitive Computer - On Language, Learning, and Artificial Intelligence*, Addison-Wesley Publishing,
- Schkolne, S., M. Pruett and P. Schroder, 2001, Surface Drawing: Creating Organic 3D Shapes with the Hand and Tangible Tools, *Conference on Human Factors in Computing Systems*. J. Jacko, A. Sears, M. eaudouin-Lafon and R. Jacob, ACM, New York: 261-268.
- Segen, J. and S. Kumar, 1998, GestureVR: Vision-based 3D Hand Interface for Spatial Interaction, *Sixth ACM International Conference on Multimedia*. W. Effelsberg and B. Smith, ACM, New York: 455-464.
- Silfer, A., G. Kinn and J. Hassett, 1987, A Geographic Information System Utilizing the Triangulated Irregular Network as a Basis for Hydrologic Modeling, *Auto-Carto 8: Proceedings of the Eight International Symposium on Computer-Assisted Cartography*. N. Chisma, American Society for Photogrammetry and Remote Sensing, Falls Church, VA: 129-136.
- Uusitalo, J., B. Orland and K. Liu, 1997, A Forest Visualization Interface for Harvest Planning, *ACSM/ASPRS Annual Convention and Exposition*, American Society for Photogrammetry and Remote Sensing, Bethesda, MD, 4: 216-223.
- Westort, C., 1996, *Sculpting DTMs: An Example Tool*, Anthos, 35 (1) 42-45.
- Westort, C., 1998, *Methods for Sculpting Digital Topographic Surfaces*, Ph.D. Thesis, University of Zurich,
- Wexelblat, A., 1995, *An Approach to Natural Gesture in Virtual Environments*, ACM Transactions on Human-Computer Interaction, 2 (3) 179-200.
- Wilson, J. and J. Gallant, 2000, Digital Terrain Analysis, *Terrain Analysis: Principles and Applications*. J. P. Wilson and J. C. Gallant, John Wiley and Sons, Inc, New York: 1-28.