Editing Heightfield using History Management and 3D Widgets

M. Adil Yalcin
Computer Engineering Department
Bilkent University
Ankara, Turkey
yalcin@cs.bilkent.edu.tr

Tolga K. Capin
Computer Engineering Department
Bilkent University
Ankara, Turkey
tcapin@cs.bilkent.edu.tr

***Abstract*—In virtual environments, terrain is generally modeled by heightfield, a 2D structure. To be able to create desired terrain geometry, software editors for this specific task have been developed. The graphics hardware, data structures and rendering techniques are developing fast to open up new possibilities to the user and terrain editor functionalities are following such improvements (such as real-time lighting updates during editing operations and multi-texture blending). Yet, current terrain editors mostly fail to give the user feedback about their actions and also fail to help the users understand and undo the editing operations on the terrain. The aim of this study is to investigate the 3d-widget based visualization of possible editing (sculpturing) actions on terrain and to help user undo previous operations.**

***Index Terms*—terrain editing, interaction techniques, synchronous interaction, graphics editors.**

I. Introduction

Terrain modeling is an important problem in designing virtual environments. Generally, terrain is represented and modeled using a 2D grid of square regions, where grid corners hold the height value of the 2D position. This model, heightfield, is used to provide a direct visual representation of the world in 3D space. There exists well-known limitations in using heightfields, such as the inability to represent overhangs. Therefore, a heightfield can be considered a 2.5D structure and its geometric and visual properties can be exploited to develop efficient modification and interaction methods.

One of the problems related to terrain rendering is the editing of large-scale terrains. Smooth interaction, which is highly dependent on frame rate, is a requirement for real-time editors. Increasing the number of grid cells that are updated during an operation has a negative impact on the frame rate and can limit the interaction with the editor. Various improvements can be developed to overcome the update speed problems (for example, computing large chunks of updates in parallel). For interactions to be real-time, an efficient terrain rendering engine is also a requirement. Terrain engines generally have features such as frustum culling, level-of-detail using patches and geomorphing [5] and further shading optimizations, which are necessary to achieve continuous interaction.

In this paper, a terrain editing system is proposed. The system is designed to be scalable over large heightfields. The aim of this work is to visualize previous editing operations and to investigate out-of-order undoing of operations.

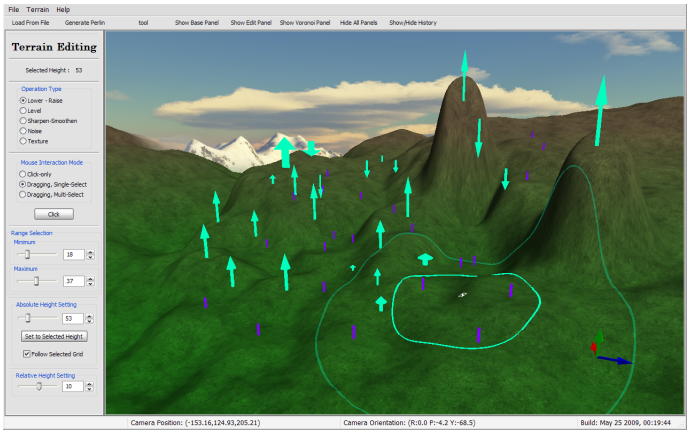


Figure 1: A screenshot from the application

II. Previous Work

Terrain editing is a common feature in most of recent 3D modeling tools. While available tools allow the user to manipulate terrain data practically, academic research can provide insight to formalization of interaction modes, operations and model properties. Papers that describe visualization techniques on terrain can be descriptive or inspiring. In a notable work, Meng [8] approaches the problem of representing and organizing ground vehicle situation information according to principles in visual perception. The proposed techniques are density based visualizations, blobs and a totem metaphor, where units and equipment in the same region are stacked for reducing visual clutter.

There exist software packages or graphics engines that allow editing terrain heightfield data and a sample set of features is listed in this section briefly. One graphics engine that has an advanced terrain modeling tool is Gamebryo [6]. It can merge multiple basic brushes to a single brushing operator. The editor supports undo and redo, but it is only through simple menu/keyboard interfaces with no 3D visual cues to the user about the operations. Many editors, such as CryEngine's Sandbox [4], support painting (texturing) operations to be constrained using terrain slopes and height . Another game engine supporting terrain editing is Unreal Editor. The supported area-editing tools are smoothing, noise, flattening and visibility (which flags a grid as visible or not)[7]. A new interesting

978-1-4244-5023-7/09/$25.00 ©2009 IEEE

ISCIS September 14-16, 2009
METU
Northern Cyprus Campus

type of brush allows direct value editing per grid (Vertex Editing). The value editing selection supports selecting/deselecting disjoint areas using multiple mouse clicks. The selections are either single-grid with each mouse click or an area click, with the strength of the operator decreasing as the points are further from the selected grid. The user has to press multiple mouse buttons and a keyboard to edit the selected grid values, and the only operation defined in this mode is raising or lowering the grids with mouse drag moves.

III. Metaphors And Design Heuristics

Metaphors are used in user interfaces to achieve natural interaction with the device. A number of metaphores that can be applied on terrain editing operations and basic user interface design heuristics are presented in this section.

Construction equipments, such as explosives, shovels, and so on, are commonly used to shape a physical terrain. Also natural phenomenas such as wind, water and even temperature can cause erosion and thus update the geometry of a physical terrain. Similar metaphors can be applied to virtual terrain editing operations, but some may violate the overhang restrictions inherent in heightfield-based data structures and some may be unpractical mostly, such as an "explosion" metaphor, which may involve many physical parameters including terrain material properties and may be unpredictable in nature. Terrain editing benefits from regional operators, rather than single grid-cell based operators, in terms of efficiency in representing randomness and smoothness through compact parameters. Matching the user's view of the system, including available brushes, mathematically and logically is also another important criteria.

In this study, a direct-manipulation based interaction is proposed. The mouse is used to select the region to perform a selected operation on. The main idea behind the direct-manipulation interface developed in this work is the "brush" and "hand" metaphors. Similar to different types of brushes in 2d paint applications (which are actually metaphors of brushes in real world), the brushes have separate parameters and visual or geometric effect on the place they are applied. The user selects the active editing operation to perform through a 2D menu interface. The active mode is visualized through the mouse projection marks on the ground and is enhanced by using color visual variable. Projection mark colors can help the user to instantly see the active mode directly on 3D terrain visualization, thus increase awareness of current editor state.

The 3D widgets that are placed on top of the ground to represent editing operations are a metaphor of street poles, but their structure and possibly animations should be designed to reflect different operational properties and logic. Achieving solid-looking and solid-behaving 3D widgets are our aim in designing 3D widgets [3].

Nielsen provided heuristics for user interface evaluation in [9]. The heuristics presented in that work are commonly used as guidelines in user interface design problems. *Visibility of system status* and *user control and freedom* are two important heuristics that lie in the core of this work. The first heuristic is applied by consistent visualization of terrain and operation status, while the second principle is directly related to undoing of operations performed. Other heuristic principles as described in the related work is referenced in this work when applicable.

IV. Editing Operations

The editing operations are based on the brush metaphor and the continuous interaction showcases the hand metaphor. The operations and interaction methods should allow easy and intuitive manipulation of terrain structure while producing realistic results. Variations of editing brushes are designed to have little high frequency detail to achieve smoothness.

A. Shared Properties of Operations

This section briefly lists the properties that are shared between different editing operations.

- A separate color is assigned to each operation. This assigned color is used in terrain selection and terrain history 2D list UI display. It is important to choose operation colors as distinguishable as possible, among different operations and possible terrain colors. The number of operations should be small accordingly. The color codes enhance usability in terms of *recognition rather than recall* heuristic guideline.
- Each editing operation is assigned its inverse operation (for applying operation undo).
- Each editing operation is applied to a specific grid in terrain. Note that for editing operations that can be applied to a complete heightfield in a single step (like a smoothing or erosion filter over complete terrain), this property does not hold.
- An inner and an outer circle can be set for each operator. The difference between the inner and outer circles denote the fall-off parameter found in many surface editing tools. The inner and outer circle have the same upper and lower bound. An additional constraint is applied to ensure that the inner circle is smaller than the outer circle. The upper bound limit should be set by the application so that the interactive editing frame-rate is achievable in all possible values, but note that this also affects the expressiveness of the operations. To set radius parameters, a double-slider is the ideal 2D widget. Two single-sliders have been used in sample implementation for this purpose, and the above logical constraints are applied to these sliders.

B. Mouse Interaction Types

Three types of mouse interaction techniques for applying an operation have been defined. These three simple types are able to define rich interaction techniques for efficient direct-manipulation based terrain editing.

1) Single Click: On mouse click event, if the mouse points to the 3D terrain, the selected grid cell is updated to point the grid which the mouse aims at and the active operation is applied to that selected grid cell and its region. A button is inserted into 2D GUI to allow the user simulate the single-click mouse event (without changing the active selected cell).

2) Click and Drag (Single cell selection): On mouse click event, the terrain cell pointed by the mouse is selected. While the mouse button is down, dragging along an axis (x or y) updates a parameter of the active operation and applies the operation as required.

3) Click and Drag (Multiple cell selection): On mouse click event and on mouse drag event (while a mouse button is clicked), the terrain cell pointed by the mouse is selected (the selected grid is changed). After each selection update, the active operation is applied to selected cell and its region as required.

C. Types of Editing Operations

In this section, basic types of editing operations are listed. Each operation can define its own set of required parameters and its interaction method. Restricting the operations to a small list decreases the system's perceived complexity in the user and increases the user's overall efficiency of interaction. Refer to Figure 2 to see editing operations applied on a regular terrain, with different parameters.

1) Lowering / Raising Operation: An additive image kernel is generated using inner radius, outer radius and modulation parameters. All the grids of the filter inside inner circle store the value 1, while the grids outside the outer circle store the value 0. The in-between gradient is filled using Gaussian distribution. Each filter cell is multiplied by m_{rel}, a separate relative height modulation parameter supplied through a 2D GUI element.

In single-click mouse interaction, the user selects a grid and applies the additive filter to selected grid. The modulation is set to m_{rel}. Dragging mouse interaction with multiple selection scheme applies the same filter to all terrain cells that are pointed while the mouse button is pressed. Dragging mouse interaction with single selection scheme allows the user to select the center ground cell of operation and maps the y-axis motion of the mouse to relative height modulation. m_{rel} is then derived from the initial height of selected grid cell and the projected height of a ray originating from camera to the mouse pointer.

2) Leveling Operation: A multiplicative image kernel is generated using inner radius, outer radius and absolute height parameters. The kernel content is created as described in lowering/raising operation, but with no modulation multiplication defined. The region that this kernel is applied is blended to the absolute height using the filter value and the equation:

$$h_{new} = h_{old} * (1 - filter) + h_{abs} * (filt)$$

In our implementation, setting absolute value h_{abs} can be done in three ways:

1) Setting a slide-bar or spinner-control of a 2D GUI.
2) Tracking the selected grid's height automatically in each update.
3) Updating to latest selected grid's height.

These different modes allow easy control over the absolute parameter and enhances user experience and expressibility.

In single-click mouse interaction, the user selects a ground point and applies the multiplicative filter to selected grid. Dragging mouse interaction with multiple-cell selection scheme applies the same filter to all terrain cells that are pointed while the mouse button is pressed.

3) Sharpening / Smoothing Operation: This operation is very similar to its 2D image processing usage. This operation also defines an inner and an outer radius, to be able to define a blend fall-off region.

4) Noise Operation: This operation allows the user to generate random variance in the terrain structure. Many types of noise models can be used for this purpose. The noise can be generated from a pseudo-random number generator and the seed can be stored as an operation parameter.

5) Texturing Operation: This operation modifies the color (painting) on the terrain. This operation does not generate geometric modifications and thus is not further analyzed.

D. Constraints

The operation constraints are applied to benefit from *error prevention* heuristic principle. Each operation shares the bounding limits constraint of the terrain (lowest-highest value bounds). If the operation applied makes a grid height value exceed the limits, the application of the related operation should be canceled.

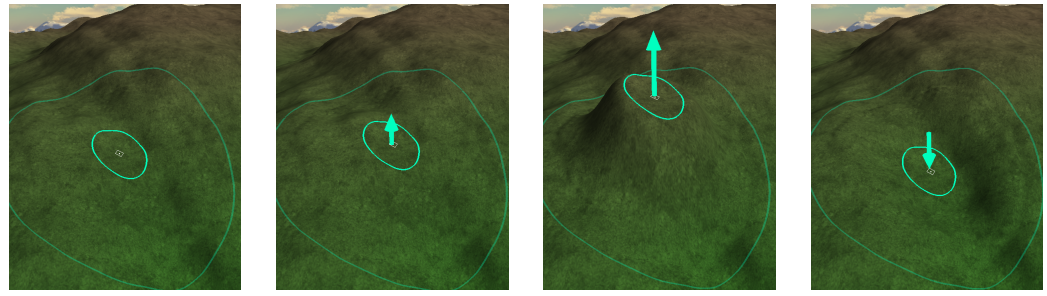
E. System Level Editing Operations

An operation's system implementation can define new system-level operations to represent an operation applied by different interaction methods to be able to add required logical behaviors and undo generators easier. Aggregate system operations pack multiple additive (lower-raise) operations under a single operation and the resulting aggregate operation represents a multi-selection drag operation. Also, system-level operations can be used to store sequences of the operation applied. The related subtypes respond to Start (click), Update (drag) and End (release) operations. Start operation can be further subdivided to be able to represent different drag schemes (single and multi selection.)

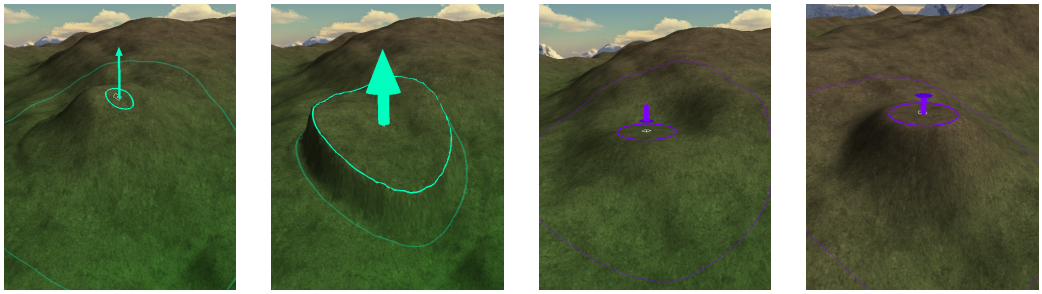
V. Selection Feedback

According to heuristic design guideline *visibility of system status*, the user has to be notified of the region that an editing operation will update. The inner and outer radius parameters must be visualized for this purpose. Another important requirement arises from the underlying data structure (2D grid-heightfield data) : the operations are applied to (mapped to) discrete grid cells rather than continuous world coordinates. Yet, providing only cell-based selection feedback produces noticeable and distracting skipping of visualized selection data. As a result, a continuous feedback of editing region must be available for smooth interaction and it must be supported by a discrete feedback also to show the user the actual grid the operation will be applied.

The proposed structure for visualizing selected circular grids on terrain is described below.



(a) Initial ground (b) Low modulation raise (c) High modulation raise (d) Lowering Operation



(e) Operation with small inner radius (f) Operation with large inner radius (g) Leveling(abs. height = current) (h) Leveling (abs. height > current)

Figure 2: Possible edit operations and 3d widget configurations

- The inner and outer radius of circular selection is rendered using lines. The line geometry is generated using the height data in the graphics client (application). Note that level-of-detail systems can introduce complexity on detail level changes. A GPU-based circle projection system can be favored as well, such as projected decals.
- The inner circle is rendered opaque, and the outer circle is rendered %70 transparent. The colors of circles match the colors of active/applied editing operation's color. Thus, the user attention is focused on selected opaque inner circle, while the outer circle selection allows the user to understand the complete region the operation affects.
- Each continuous selection point data (P_{cont}) can be mapped to a terrain grid (P_{grid}). P_{grid} is selected as the nearest neighbor cell to P_{cont}. P_{grid} is displayed by using point 3D primitive. In a terrain system using level of detail, the rendered height of the position can be different than the actual height of the grid (if low level of detail is used for that region). The depth testing can be disabled so that the point is always visible even if lower level detailed data occludes the point. To enhance the grid-based selection feedback, the neighbor grids of P_{grid} are rendered using line 3D primitives.

The selected terrain grid can be updated in the sample application by the following user inputs:

- *Clicking / dragging left mouse button*: The user may prefer first to select the terrain by using left-click selection, and apply the active editing operation later (possibly using a keyboard or 2D GUI). This can help user specify the selected grid more precisely. While the left mouse button is down, the selected grid information is updated using ray-casting to heightfield visualization.
- *Clicking middle mouse button*: Direct manipulation interaction involves selecting the pointed grid on operation application time. Thus, clicking the middle button, which is used as the operation application button, updates the selected grid.

VI. 3D Widget Representations of Operations and Widget Interactions

Each editing operation can be assigned a 3D widget for direct visualization of an applied operation over terrain. This section briefly discusses possible widget types, possible interactions with widgets and visual effect enhancements on widgets. As Baecker et. al. [1] present, suitable animations on icons (3d widgets in this study) can help user to better understand the icon meaning, capabilities and methods of interaction. Thus, possible animation enhancements on these 3D widgets have also been described. Refer to Figure 2 to see the widget representations of some editing operations applied.

A. Lowering / Raising Operation Widget

The 3D widget that represents this operation is an arrow widget. The head of arrow (conic) points upwards if the editing operation is raising operation, else it points downwards. The radius of the arrow base cylinder is mapped to the inner radius of the operation. The length of the arrow base cylinder is mapped to logarithm of modulation, compressing high modulation ranges to smaller ranges. The length of the arrow head is mapped to the thickness of the arrow. An animation is implemented to stress the movement direction. The animation scales the arrow head only in the up-direction. Scaling whole

arrow is avoided since it would change the length of the base cylinder and the coordinates of the base plane of the cone, both introducing unwanted visual variance.

B. Leveling Operation Widget

The 3D widget that represents this operation is a 2D cylinder, whose main axis is the terrain up axis. Its radius is mapped to the inner radius of the operation. Leveling operation either lowers or heightens the selected grid . The cylinder is extended with a circle cap on the upper side if the leveling operation caused the applied terrain grid to raise. If the circle is below the cylinder, it can be inferred by the user that the operation lowered the region as a result. This is similar to the arrow widget used in lowering-raising operation, but with a circle plane instead of a conic arrow head. The animation can be effectively used to update the circle plane radius, emphasizing the resulting lowering or raising behavior of the operation applied.

C. Interaction with 3D Widgets

Right-clicking a 3d widget undoes the editing operation that the widget represents. Activating a previously applied operation as current operation is done by left-clicking. Middle-click locks/unlocks the editing operation.

D. Shading / Effects on 3D Widgets

- 3D widgets can become transparent with an ease-out animation based on time interval after it is created. The fade-out animation should also start in around a few seconds after the widget is created (editing operation is finalized). If a transparent 3D widget is focused by (hovered over) the mouse, that widget becomes opaque.
- A glow effect on the selected 3d widget can be used to reflect that the widget is currently selected.
- Currently, a simple shading, which assigns a constant color to the widgets in rendering phase. Shading the widgets using scene lighting information can enhance the solidity of the 3D widgets, yet may also infer with the color code information of the operation, in brightly shaded or shadowed areas.

VII. History Management And Supporting Undo

Undo is an important functionality for usability of software as described in Section III. Most of the current software allows only the last operation to be undone. In this approach, going back to previous state in time is only possible through undoing all the actions performed in sequence. Yet, the user may prefer to undo only a specific operation. This feature is for example, very useful if the updated data is large and updated regions are separated from each other.

There are also numerous academic research done in undoing operations[10], [12], [2], [11], [13]. These previous works describe the details of managing the operation history in detail, including multi-user environment problems. Some of them are directly related to graphics / image editing. Berlage presents [2] a notable work that provides a taxonomy of different undo mechanisms and approaches the undo problem in many aspects, such as software engineering, user interfaces and synchronous interaction.

A. History Based Visualization

The editing operations the user performs on the terrain are book-kept by using the type of the operation and its parameters. Since operations can be undone in any order, an efficient way of storing operation history would be using a linked list implementation.

The operations performed are presented to the user sequentially in a 2D list widget. Multiple operations can be selected using this menu widget. The selected operations are visualized on the terrain using previously described selection methods, but without grid-base rendering. These enhancements are designed to guide the user on terrain selection time, so can not provide an improvement to visualization problem of previously applied editing operations. See Section VI for the description of history-based interaction with widgets.

Additional UI design heuristics applied to history management layer are *consistency and standards guideline* for reversing the last operation done using commonly used Control+Z key combination and *visibility of system status guideline*, noting the number of operations that could not be undone on a multi-operation undo operation, either because no inverse operation for the selected editing operation is defined or the selected operation is undo-locked.

B. Generating Inverse Editing Operations

Inverse of some editing operations can be defined easily. One such operation is additive operation. The inverse of raising/lowering modulation that updates the selected height position by modulation m is the same operation with same parameters, but this time the modulation parameter is set to $-m$. Also, for other additive operations (such as noise blending), the same idea can be also applied. The undo of noise blending operation uses the seed parameters to generate a new additive noise data on demand, using negative pseudo-random modulations.

For other operators, simple and compact models may be impossible. For example, in leveling operation all the previous state height data cannot be generated back in the regions where the active blend filter is 1. One way to store history in these types of cases is to duplicate all the height-image data in each operation, which can be impractical on large terrains. Yet, some operations and interactions with terrain require storing complete previous state of the heightfield image or storing modulation for each grid. Also, multi-select dragging operations require storing initial terrain height state and generating height updates using initial height state so that consistent results are generated until the mouse button is released and drag operation is completed.

VIII. Results And Discussions

It has been observed that if the updated regions are in separate locations on terrain, out-of-order undoing allows the

user select and reverse operations based on regions, without affecting other operations performed on terrain. Otherwise, if frequent updates are performed on the same region, the 3D widgets might become over-crowded in that region, and undoing an operation may affect the other previous or latter operators that influenced the same regions on terrain. Such crowded operations are generally an indicator that the user is making an effort to achieve a geometric structure and continuously fails, updating the same region in turn.

The proposed continuous and discrete (grid-based) selection feedbacks follow *system status visualization* heuristic, and also avoid distracting large-scale skips that are inherent in grid-based only feedbacks. Selection visualization on terrain can have problems emerging from heightfield representations and level of detailed terrain rendering systems, where the rendered height of a grid position may differ from underlying heightfield data.

The basic set of presented operations are aimed at updating the terrain using circular regions. Thus, these operations can fail to provide precise operation on single grids and circular feedbacks fail to visualize single-grid operations efficiently. Generating special shapes, such as helix-like geometries, using basic regional operators is not intuitive. Also, the absolute and relative mode settings in 2D menu UI may not capture highest vertical resolution, to allow the user to roughly select in large ranges faster. Thus, height updates can at times fail to represent detailed height adjustments. The operations and interactions proposed in this work are more likely to be efficient in rough terrain editing, but fast regional updates and undo functionality helps the user to explore possible configurations on terrain easier.

Updating heightfield data while allowing interactive frame rates is observed to be a problem for update grids larger than 200x200 units in sample implementation. The requirement for duplicating previous height data in many operations is likely to affect update rate, so they should be avoided by preferring compact operators, such as additive lowering-raising operator, if possible.

It should also be stressed that the computational power requirement and algorithmic complexities to generate inverse operators can vary between different operations and interaction types. Generating system-level operations to represent multi grid-cell selections or complete height data history is also proposed as an efficient way to store and handle some complex interactions.

IX. Conclusion and Future Work

A basic methodology to approach terrain editing problem using history management and 3D widgets has been proposed, which is likely to enhance the user's perception of the operations performed on the terrain to a large extent and allow a basic framework which allows undoing of editing operations efficiently.

Representing multi-cell operations is an important problem that have to be addressed in future work. This is both required for applying operations on terrain and undoing them. Compressing previous height data on heightfield update and decompressing that data on state rollback is one approach that can handle complex cases where the previous state of the heightfield is lost after an operation is applied. Visualizing multi-grid operations can also be extended to include non-circular operations and their widget representations can be analyzed. A complete, simple and powerful formal language for operation types and interaction modes is yet to be defined for the terrain operation, although this work can be considered as an initial approach to formally representing this idea. Interactions using multi-touch screens and multi-user system extensions can be other fruitful works related to presented work. Other minor extensions would be grouping of previous operations into a single operation by the user (to reduce widgets cluttering in regions where a lot of operations have been performed) and region-based undo-locking of applied operations.

References

[1] Ronald Baecker, Ian Small, and Richard Mander. Bringing icons to life. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1–6, New York, NY, USA, 1991. ACM.

[2] Thomas Berlage. A selective undo mechanism for graphical user interfaces based on command objects. *ACM Trans. Comput.-Hum. Interact.*, 1(3):269–294, 1994.

[3] Bay-Wei Chang and David Ungar. Animation: from cartoons to the user interface. In *UIST '93: Proceedings of the 6th annual ACM symposium on User interface software and technology*, pages 45–55, New York, NY, USA, 1993. ACM.

[4] Crytek. Cryengine 2 technology. Website, 2009. http://www.cryengine2.com/index.php?pnr=1&conid=2.

[5] Willem H. de Boer. Fast terrain rendering using geometrical mipmapping. Website, 2000. http://www.flipcode.com/archives/Fast_Terrain_Rendering_Using_Geometrical_MipMapping.shtml.

[6] emergent.net. Gamebryo 2.6 terrain development tools | emergent. Website, 2009. http://www.emergent.net/en/Products/Gamebryo/Tools/Terrain/.

[7] Epic Games. Unreal engine terrain editor - editing height maps document. Website, 2009. http://udn.epicgames.com/Two/EditingTerrainMaps.html.

[8] Foo Meng and Hian Beng. Perceptual based visualization techniques for improving ground situation picture understanding. In *The 12th ICCRTS*, 2007.

[9] Jakob Nielsen. Usability inspection methods. In *CHI '95: Conference companion on Human factors in computing systems*, pages 377–378, New York, NY, USA, 1995. ACM.

[10] Atul Prakash and Michael J. Knister. A framework for undoing actions in collaborative systems. *ACM Trans. Comput.-Hum. Interact.*, 1(4):295–330, 1994.

[11] Chengzheng Sun. Undo any operation at any time in group editors. In *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 191–200, New York, NY, USA, 2000. ACM.

[12] Chengzheng Sun and David Chen. Consistency maintenance in real-time collaborative graphics editing systems. *ACM Trans. Comput.-Hum. Interact.*, 9(1):1–41, 2002.

[13] Xueyi Wang, Jiajun Bu, and Chun Chen. Achieving undo in bitmap-based collaborative graphics editing systems. In *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 68–76, New York, NY, USA, 2002. ACM.