Collapsible 3D Terrains for GIS Visualization

Suwen Wang, Robert Beiko, Stephen Brooks

Dalhousie University, Canada
Telephone: +(1)902-494-2512
Fax: +(1)902-492-1517
Email: sbrooks@cs.dal.ca

1. Introduction

Visualization helps users comprehend spatially referenced data, and with the rise of commodity 3D graphics, it has prompted the development of 3D representations. However, further research is needed to better utilize the potential of 3D GIS. In particular, 3D GIS often suffer from the problem that available screen space is insufficient for visualizing the entire terrain in any detail. For example, if one was interested in analyzing two or more distant regions simultaneously, one would need to 'zoom-out' significantly, severely limiting the amount of detail shown. This issue is particularly acute when operating a portable device.

In this work, we describe a 3D GIS featuring 'terrain collapsing', a user-control allowing a portion of a 3D terrain to be dynamically shrunk to occupy less space on the display, as shown in Figure 1. This feature is useful for simultaneously viewing regions which are arbitrarily distant from each other. With irrelevant areas collapsed, more screen space is dedicated for displaying regions of interest.

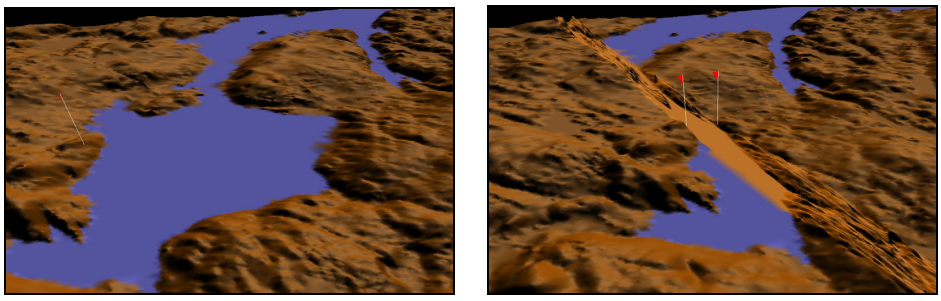

Figure 1. Left: original 3D terrain. Right: central area of terrain is collapsed.

2. Related Work

GIS has gradually moved into the third dimension, with systems such as ArcGIS (ESRI, 2006), as well as research prototypes such as GeoZui3D (Ware et al., 2001). However, 3D is often restricted to simple fly-bys. Another area of related work has developed 2D distortion techniques to overcome the problem of limited screen space within 2D interfaces. This is normally achieved by applying piecewise linear or continuous magnification functions to objects (Leung and Apperley, 1994).

3. The Interactive 3D Terrain System

Terrain collapsing allows a user-specified portion of the terrain to be dynamically shrunk. In order to proceed, the user must specify the region to be collapsed within the 3D interface. From the user's point of view the process is quite simple: the user specifies two

points on the terrain which are to be brought together. This is sufficient information for the system to perform the collapsing operation.

3.1 Point Selection on a 3D Terrain Surface

The user must be able to directly specify locations on the 3D terrain with a click of the mouse which raises technical issues. In particular, when the user selects a point, it does not specify a single point in the 3D space (due to 3D depth). Instead, it casts an imaginary ray extending from the point on the display outwards to infinity. We must determine which point on this ray intersects the terrain, using a reverse transformation of 3D projection. Using reverse projection we can trace a mouse click from a point on the screen to the target 3D point on the terrain surface.

Moreover, we must allow the user to specify *any* two points on the terrain, not just two that happen to be visible. We therefore provide camera controls, so that the user can move to any location to specify the first point, then move to any other distant location to specify the second point. As these camera controls are typical of any 3D GIS, we will not discuss them in detail here.

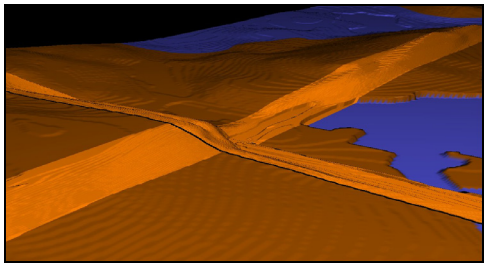

Figure 2. Multiple orthogonal collapsing zones.

3.2 Orthogonal Terrain Collapsing

If we restrict the collapsing zones to be parallel to the coordinate axes, it enables us to model the terrain as a set of parallel quad-strips. In this restricted case, the terrain is divided into three parts: the collapsing zone and two non-collapsing zones on either side. Collapsing can be achieved entirely with hardware-accelerated 3D transformations. Firstly, one non-collapsing zone is rendered without alteration. Secondly, a scaling transformation is performed to shrink the size of the collapsing zone before rendering. Finally, the remaining non-collapsing zone is moved to abut the collapsing zone.

We can also perform multiple orthogonal collapses simultaneously. This requires a pair of points to be selected by the user for each collapsing zone. For example, in Figure 2, there are two collapsing zones allowing four regions to be brought together.

3.3 Improving Efficiency with Level of Detail

The previously described method is overly restrictive and inefficient. We improve efficiency by incorporating level of detail (LOD) techniques into the terrain, which increases frame rates without a significant loss of visual fidelity. LOD operates by adapting the number of polygons used at different locations of the terrain, using a higher number of polygons only where needed.

We implemented a view-dependent LOD framework using a quadtree data structure (Lindstrom et al, 1996). A quadtree representation of a surface is defined by recursively

subdividing the surface to form a tree, in which each node represents a portion of the surface. The area covered by each node is recursively divided into four quadrants, which form the four children of the node.

The procedure for rendering the appropriate LOD is composed of two steps: updating and rendering. The algorithm recursively subdivides the mesh and visits the corresponding quadtree nodes in a depth-first order. Two types of vertex error checks are carried out at each step to determine whether further polygons need to be included at a given location. After the updating step, all selected vertices are rendered recursively by forming triangle fan geometric primitives. Any crack artifacts caused by inconsistencies between different LOD levels are eliminated by reinforcing vertex dependencies. Results using LOD are shown in Figure 3 (right).

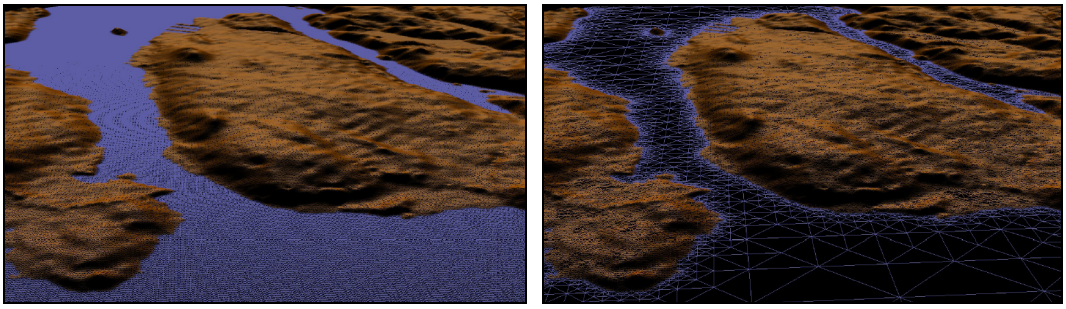

Figure 3. High-density wireframe without (left) and with (right) level-of-detail.

3.4 Terrain Collapsing at Arbitrary Angles with Level of Detail

In addition to improving efficiency, we want the collapsing to operate at any arbitrary angle. Also, when using LOD, it is difficult to group vertices into discrete geometric objects and more importantly, the quadtree is rendered recursively. We therefore cannot rely on standard OpenGL transformations to perform the collapsing as in section 3.2 and must instead calculate a new temporary coordinate for each vertex.

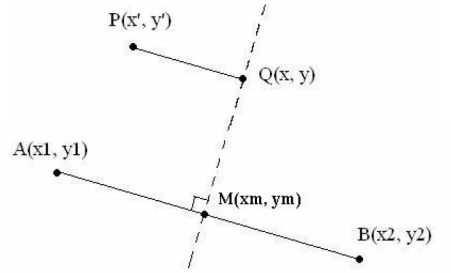

Figure 4. Calculating the collapsing axis.

Depending on the location of a vertex relative to the collapsing zone, we either scale its coordinates about the collapsing axis or translate it towards the collapsing axis. We derive the transformed temporary coordinates as follows. Given the two user-selected target points $A = (x_1, y_1)$ and $B = (x_2, y_2)$, the mid-point is calculated as:

$$M(x_m = \frac{x_1+x_2}{2},\ y_m = \frac{y_1+y_2}{2}) \quad (1)$$

For the point $P(x', y')$ we wish to transform, we find the closest point $Q(x, y)$ on the line l_n that passes through M and is perpendicular to line $\overrightarrow{AB}$, as shown in Figure 4. Let k_{AB} denote the slope of $\overrightarrow{AB}$:

$$k_{AB} = \frac{y_1 - y_2}{x_1 - x_2} = \frac{y - y'}{x - x'} \tag{2}$$

and:

$$\frac{y - y_m}{x - x_m} = -\frac{1}{k_{AB}} \tag{3}$$

We then solve:

$$k_{AB}(x - x') + y' = -\frac{1}{k_{AB}}(x - x_m) + y_m \tag{4}$$

$$\therefore \begin{cases} x = \frac{k_{AB}x' + \frac{x_m}{k_{AB}} + y_m - y'}{k_{AB} + \frac{1}{k_{AB}}} \\ y = k_{AB}(x - x') + y' \end{cases}$$

For any point P, we find its projection Q on the line perpendicular to $\overrightarrow{AB}$. Then its new coordinates after collapsing are derived as follows. If P is inside the collapsing zone,

$$\begin{cases} x_{collapse} = x + r(x' - x) \\ y_{collapse} = y + r(y' - y) \end{cases} \tag{5}$$

where r is the collapsing ratio. Otherwise, the point is outside the collapsing zone, and only needs to be translated to the appropriate location. From trigonometric relations:

$$\begin{aligned} \Delta x &= \cos(arctg(k_{AB}))(1 - r)d \\ \Delta y &= \sin(arctg(k_{AB}))(1 - r)d \end{aligned} \tag{6}$$

where d is the distance between P and Q. The new coordinate for a translated vertex is:

$$\begin{cases} x_{collapse} = x' \pm \Delta x \\ y_{collapse} = y' \pm \Delta y \end{cases} . \tag{7}$$

This entire process is dynamic with the collapse occurring gradually as shown in Figure 5.

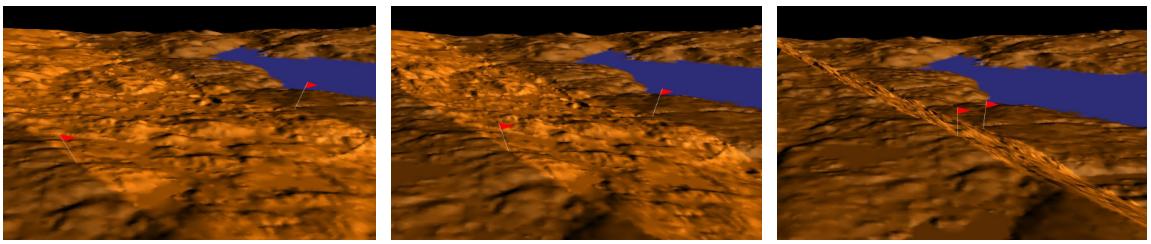

Figure 5. Dynamically collapsing at an arbitrary angle.

4. Conclusion and Future Work

We have introduced a system for interactively collapsing unimportant regions of terrains within a 3D GIS. We propose that this will be useful for comparing regions which are

arbitrarily distant from each other, without sacrificing detail. Thus far, we have implemented the collapsing functionality and have made progress towards LOD rendering of the terrain. But there are additional improvements which we aim to pursue.

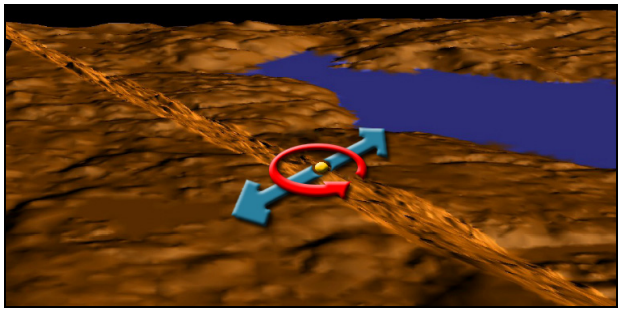

Figure 6. Diagrammatic sketch of additional interactive controls.

A promising avenue would be the development of controls for the precise positioning and orientation of existing collapsing zones. Figure 6 shows a diagrammatic sketch of what these controls might look like. It would allow the user to continuously rotate the collapsed zone or interactively add more terrain to the collapsed zone. For example, clicking-and-dragging the red arrow would rotate the line of collapse around its centre, clicking-and-dragging one of the blue arrows would draw in more terrain into the collapse on that side, and clicking-and-dragging the yellow ball in the centre of the control would slide the control itself along the axis of collapse.

5. Acknowledgements

This work was supported by NSERC and the Canadian Foundation for Innovation.

6. References

ESRI *ArcGIS*. Retrieved April 20, 2007, http://www.esri.com/software/arcgis/.

Leung Y and Apperley M, 1994, A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160.

Lindstrom P, Koller D, Ribarsky W, Hodges L, Faust N, and Turner G, 1996, Real-time, continuous level of detail rendering of height fields. *Proceedings of SIGGRAPH '96*, 109–118.

Stota J and Zlatanova S, 2003, 3D GIS, where are we standing? *IPRS Joint Workshop on Spatial, Temporal and Multi- Multi-Dimensional Data Modeling and Analysis*.

Ware C, Plumlee M, Arsenault R, Mayer L and Smith S, 2001. GeoZui3D: data fusion for interpreting oceanographic data. *Proceedings of the MTS/IEEE Conference*, 3, 1960-1964.