Sketch-Based Annotations in Google Earth

Christian Schulze Laurens Nienhaus
Hochschule Bremen (University of Applied Sciences)

Jörn Loviscach*
Fachhochschule Bielefeld (University of Applied Sciences)

1 Introduction

Painting in a 3D geobrowser is interesting both for artistic uses such as virtual graffiti and for commercial applications such as architectural sketches. It *seems* straightforward to turn Google Earth into a 3D painting tool: Store the 3D mouse data that the software returns and construct polylines from them. However, this approach has two vexing drawbacks: First, when the user paints past an edge, the end of the stroke will be placed at an incorrect depth, see Figure 1; second, it is not possible to sketch in mid-air, for instance to indicate a planned height extension of a building.

These issues could be solved easily if 3D meshes were available. However, these data remain mostly inaccessible, partly due to copyright issues. We present a prototype of a collaborative Web-based solution that constructs simplified painting surfaces from 3D mouse points collected through Google Earth. (Note that the stand-alone version of the Google Earth browser offers tools to draw paths and polygons onto the scene, but is limited to ground-level drawings.)

Figure 1: *Even though 3D mouse data are available from Google Earth, the results can be surprising when one paints past edges (left: view during painting, right: rotated).*

Before starting the actual painting, the user is asked to brush with the mouse across the interesting region of the building. We apply RANSAC to find up to three planes in the 3D data thus collected. This allows dealing, for instance, with typical corner points. Afterward, the user can select from three painting modes:

- Strokes can only be applied to the convex surface formed by the planes.
- Strokes can only be applied to the surface formed by a single, selected plane, within the limits given by the intersections with the other planes.
- Strokes are applied to a single, selected plane, but now are allowed on its entire area to support painting in mid-air.

The user may adjust the color, the opacity and the size of the brush. This is implemented by creating appropriate 3D polylines inside Google Earth. For fine adjustment, the detected planes can be nudged in the direction or against the direction of their normal. This is helpful to correct planes that are offset due to our RANSAC-based algorithm getting stuck in recesses or to prevent z-fighting artifacts between the 3D paint strokes and the buildings.

*e-mail: joern.loviscach@fh-bielefeld.de

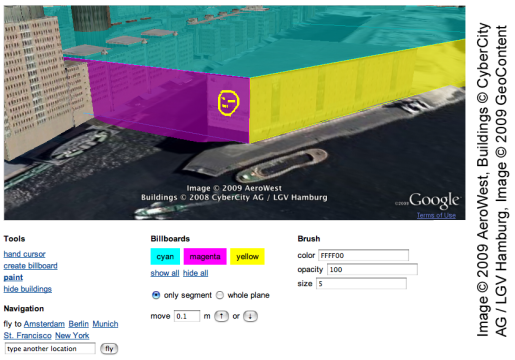

Figure 2: *The user interface combines the 3D view (Google Earth plug-in) with drawing tools and view controls (HTML, JavaScript).*

2 Implementation

The front end of our system consists of a Web page containing the Google Earth plug-in accompanied by controls based on HTML, see Figure 2. The back end is formed by the Google Earth servers and by our own application server running Ruby on Rails to provide the users' annotations. Already existing annotations, in particular those of other users, are inserted from our server into Google Earth's 3D scenes by providing KML files.

A client—be it the standalone version or the Web-browser plug-in of Google Earth—once receives a main KML file containing all current annotations from our server. This KML file periodically sends requests for updates to the server using a KML NetworkLink. Another KML file containing instructions on which drawings to add, update or delete is then served to and processed by the Google Earth client. The KML NetworkLinkControl technique for modifying elements in a KML document is employed here.

Each request for updates contains the timestamp of the last successful update in order to only send annotations that are new or that have been changed in the meantime. Other users can access work in progress as soon as a user has sent his or her changes to the server. The access is possible with only a short delay due to the periodical requests for updates described before.

3 Conclusion and Outlook

We presented a system that leverages existing 3D content despite its restricted availability. Graffiti-like objects can be built with and integrated into standard Web technology.

For architects and engineers distributed over a construction site, one can imagine a Web-based solution where everybody carries a mobile computer, uses this to paint markings and look at other persons' inputs, and communicates with the others through a chat interface realized with Google Earth's placemark balloons.

On the side of entertainment, one can imagine a voting system for graffiti or a game that is based on different roles such as sprayer, janitor, and police. The set of drawing tools could be extended. Entertainment applications may include effects such as color dripping and the automatic weathering of graffiti with time.