Lecture XII
Zoom Visualization

This lecture discuss the issues arising from zooming visualization: it is the simplest form of what is often called visual navigation or walkthru visualization. Walkthru visualization are usually for 3D models. But in 2D which is our focus, so many issues are greatly simplified.

§1. Introduction

There is a basic mathematical background that is critical in graphics and visualization, called transformations. The topic arises because we have on the one hand, a geometric model M and on the other hand, an image I of this model. Certain "transformations" must be performed to convert M to I, based on the current transformation parameters. In 2-D, most of this becomes very simple. A further simplification comes from the fact that we (usually) do not want to rotate our views.

Suppose we are given a very large map M to be visualized in a zoom and pan model. In this lecture, we discuss a model for implementing this in three stages.

First we assume that M has its native coordinate system called **world coordinate system**. For us, this is basically the LON/LAT coordinate. We also need to convert the LON/LAT into absolute (world) unit of length: if D is the diameter of the earth in meters, then each degree LAT is equal to $\pi D/360$ meters. We will use $D = 12,756$.

At the end user, we have a **view window** which has **view size** $w_{\tt view} \times h_{\tt view}$ in pixels. It has its own **view coordinate system**, so that a point (x, y) in this window satisfies $0 \le x < w_{view}$ and $0 \le y < h_{view}$. This window has a (current) **view position** $p_{\tt view}$ which is in world coordinates. Essentially, this says that the origina $(0, 0)$ in the view coordinate system represents the point $p_{\tt view}$ in model M. Associated with the view window is also the (current) **zoom level** which is a number denoted $\ell_{\tt view}$. This number is in the units of **meters per pixel**.

So, the first approximation of our problem is to simply provide a transformation between world coordinates and view window. That is essentially what you did in homework 4. However, this will be inadequate and we will need to introduce intermediate structures between the world and view coordinates. This arise in two ways: from buffering as well as from level-of-detail (LOD) hierarchies in our model.

§2. Transformations

We discuss transformations only in 2-dimensions. If $p = (x, y)$ is a point, we write $(x, y, 1)$ as the **standard set** of **homogeneous coordinates** for p. In general, any triple of the form (cx, cy, c) where $c \neq 0$ is called a set of homogeneous coordinates for p.

What is a transformation T? It is a mapping between two coordinate systems. If (x, y) is a point in one coordinate system, then $(x', y') = T(x, y)$ is a point in another coordinate system. The simplest example is **translation** by a vector $t = (t_x, t_y)$. In this case,

$$T_1(x, y) = (x + t_x, y + t_y). \tag{1}$$

Another basic transformation T_2 is rotation by an angle θ. This can best written as a matrix $M(\theta)$:

$$T_2(x, y) \quad = \quad M(\theta)(x, y)^t$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}.$$

Note that we view a point (x, y) as a column vector $(x, y)^t$ (transpose of the row vector (x, y)) when we use matrix notations.

What about the composition of translation T_1 with rotation T_2? In other words, how should we represent the transformation $T_{12}(x, y) = T_1(T_2(x, y))$? We must now use the homogeneous notation for (x, y) and write $(x, y, 1)$ (or $(x, y, 1)^t$ as a column vector). Then the

$$\begin{aligned} T_{12}(x, y) &= M(\theta, t)(x, y, 1)^t \\ &= \left[\begin{array}{cc|c} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ \hline 0 & 0 & 1 \end{array}\right] \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \end{aligned}$$

Thus, multiplication by a special kind of 3×3 matrix is our most general form of transformation. Such matrix has the property that the bottom row is $(0, 0, 1)$. In general, product AB of two such matrices A, B represents the corresponding composition. Note that $AB \neq BA$ in general.

Translation and rotation, and their compositions, represents the class of Euclidean transformations: such transformations are characterized by the fact that the preserve lengths and angles, and do not flip the image into an mirror image.

Besides Euclidean transformations, the other transformations that is very important for us is **scaling**. Such matrices are represented by

$$S(s_x, s_y) = \left[\begin{array}{cc|c} s_x & 0 & 0 \\ 0 & s_y & 0 \\ \hline 0 & 0 & 1 \end{array}\right]$$

where s_x, s_y are arbitrary positive numbers. They represent the scaling in the x- and y-direction. Usually, $s_x = s_y$. If $s_x > 1$ ($s_x < 1$), this represents an expansion (contraction) in the x-direction. We may check that $s_x = s_y = 1$ is the identity.

Exercise: Describe the transformation that forms a mirror image of an image.

Exercise: Prove that scaling and translation do not change lengths.

§3. Zoom Levels

First we assume that M has its native coordinate system called **world coordinate system**. For us, this is basically the LON/LAT coordinate. We also need to convert the LON/LAT into absolute (world) unit of length: if D is the diameter of the earth in meters, then each degree LAT is equal to $\pi D/360$ meters. We will use $D = 12, 756$.

Let ℓ be any zoom level. We will partitition the zoom level into λ **simplification levels** (or, **S-level**). Let

$$0 = s_0 < s_1 < s_2 < \cdots < s_\lambda = \infty \tag{2}$$

be fixed for this discussion. Then we say ℓ is in S-level i if

$$s_i \leq \ell < s_{i+1}.$$

Write $SL(\ell) = i$ in this case.

§4. Buffer Levels

Next, we assume that there is a **buffer** whose size is $w_{\texttt{buff}} \times h_{\texttt{buff}}$ in pixels. The buffer has a (current) position denoted $P_{\texttt{buff}}$ (or just P), again in world coordinates and has the same interpretation as $p_{\texttt{view}}$. The buffer has a (current) **simplification level** denoted $L_{\texttt{buff}}$. Thus $0 \leq L_{\texttt{buff}} < \lambda$.

§5. Design for Responsiveness

In remote visualization systems, we have two major pieces of runtime software: the Server and the Client. The Server holds the data to be visualized and the Client manages the GUI and talks to the server. The major issue we want to address is "responsiveness". This is defined to be a properto of our system to act responsively to the changing environment. We address two main issues.

- Bandwidth Variations Bandwidths can temporarily drop to zero – a responsive system must act reasonably even in this setting. This may sound paradoxical – is there anything we must do when there is no bandwidth? The answer is yes. Anticipating such situations, we could build up a lower resolution model which can be used during bandwidth blackout periods. In short, our system can "amortize" the variable bandwidth.
- User changes of mind Users are fickle, and will often change their mind. One moment, the user may want to view one part of the model, and before this could be fulfilled, the user may decide to move on to another part. The user does not explicitly tell us "cancel that order". But we need to deduce this.

The issue of bandwidth variation already arises in the setting of I/O limited visualization, such as found in traditional WalkThru visualization work. There, the bandwidths are fully reliable but it is generally insufficient to satisfy the user requests. Here, the physical bandwidth does not change, but the **bandwidth demand** changes. But the end effect from a system view point is rather similar.

In the present design (unlike [1]), the Server is just the Postgresql server. The Client Program has to have all the smarts to achieve a "responsive system".

Another issue is that of redundant data – the client does not want to waste bandwidth asking for data it already have. This can be solved at various levels of granularity. At present, we will use a very crude method, which is just based on the bounding box of the buffers.

END OF LECTURE

References

[1] K. Been. *Responsive Thinwire Visualization of Large Geographic Datasets.* Ph.d. thesis, Department of Computer Science, New York University, Sept. 2002. Download from `http://cs.nyu.edu/visual/home/pub/`.