Virtual Trackball

CSE 781

Han-Wei Shen
Winter 2010
3D Rotations with Trackball

- Imagine the objects are rotated along with a imaginary hemi-sphere
Virtual Trackball

- Allow the user to define 3D rotation using mouse click in 2D windows
- Work similarly like the hardware trackball devices
Virtual Trackball

- Superimpose a hemi-sphere onto the viewport
- This hemi-sphere is projected to a circle inscribed to the viewport
- The mouse position is projected orthographically to this hemi-sphere
Virtual Trackball

- Keep track the previous mouse position and the current position
- Calculate their projection positions p1 and p2 to the virtual hemisphere
- We then rotate the sphere from p1 to p2 by finding the proper rotation axis and angle
- This rotation (in eye space!) is then applied to the object (call the rotation before you define the camera with gluLookAt())
- You should also remember to accumulate the current rotation to the previous modelview matrix
Virtual Trackball

- The axis of rotation is given by the normal to the plane determined by the origin, $p_1$, and $p_2$
  \[ n = p_1 \times p_1 \]
- The angle between $p_1$ and $p_2$ is given by
  \[ | \sin \theta | = \frac{|n|}{|p_1||p_2|} \]
Virtual Trackball

- How to calculate p1 and p2?
- Assuming the mouse position is \((x,y)\), then the sphere point \(P\) also has \(x\) and \(y\) coordinates equal to \(x\) and \(y\)
- Assume the radius of the hemi-sphere is 1. So the \(z\) coordinate of \(P\) is \(\sqrt{1 - x^2 - y^2}\)
- Note: normalize viewport \(y\) extend to -1 to 1
- If a point is outside the circle, project it to the nearest point on the circle (set \(z\) to 0 and renormalize \((x,y)\))
Virtual Trackball

Visualization of the algorithm
Example

- Example from Ed Angel’s OpenGL Primer
- In this example, the virtual trackball is used to rotate a color cube
- The code for the colorcube function is omitted
- I will not cover the following code, but I am sure you will find it useful
Initialization

#define bool int  /* if system does not support
    bool type */
#define false 0
#define true 1
#define M_PI 3.14159 /* if not in math.h */

int winWidth, winHeight;

float angle = 0.0, axis[3], trans[3];

bool trackingMouse = false;
bool redrawContinue = false;
bool trackballMove = false;

float lastPos[3] = {0.0, 0.0, 0.0};
int curx, cury;
int startX, startY;
The Projection Step

```c
void trackball_ptov(int x, int y, int width, int height, float v[3])
{
    float d, a;
    /* project x,y onto a hemisphere centered within width, height ,
    note z is up here*/
    v[0] = (2.0*x - width) / width;
    v[1] = (height - 2.0F*y) / height;
    d = sqrt(v[0]*v[0] + v[1]*v[1]);
    v[2] = cos((M_PI/2.0) * ((d < 1.0) ? d : 1.0));
    a = 1.0 / sqrt(v[0]*v[0] + v[1]*v[1] + v[2]*v[2]);
    v[0] *= a;    v[1] *= a;    v[2] *= a;
}
```
glutMotionFunc (1)

Void mouseMotion(int x, int y)
{
    float curPos[3],
    dx, dy, dz;
    /* compute position on hemisphere */
    trackball_ptov(x, y, winWidth, winHeight, curPos);
    if(trackingMouse)
    {
        /* compute the change in position
           on the hemisphere */
        dx = curPos[0] - lastPos[0];
        dy = curPos[1] - lastPos[1];
        dz = curPos[2] - lastPos[2];
    }
if (dx || dy || dz)
{
    /* compute theta and cross product */
    angle = 90.0 * sqrt(dx*dx + dy*dy + dz*dz);
    axis[0] = lastPos[1]*curPos[2] -
                lastPos[2]*curPos[1];
    axis[1] = lastPos[2]*curPos[0] -
                lastPos[0]*curPos[2];
    axis[2] = lastPos[0]*curPos[1] -
                lastPos[1]*curPos[0];
    /* update position */
    lastPos[0] = curPos[0];
    lastPos[1] = curPos[1];
    lastPos[2] = curPos[2];
}
}
Idle and Display Callbacks

```c
void spinCube()
{
    if (redrawContinue) glutPostRedisplay();
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    if (trackballMove)
    {
        glRotatef(angle, axis[0], axis[1], axis[2]);
    }
    colorcube();
    glutSwapBuffers();
}
```
void mouseButton(int button, int state, int x, int y) {
    if(button==GLUT_RIGHT_BUTTON) exit(0);

    /* holding down left button
       allows user to rotate cube */
    if(button==GLUT_LEFT_BUTTON) switch(state) {
        case GLUT_DOWN:
            y=winHeight-y;
            startMotion( x,y);
            break;
        case GLUT_UP:
            stopMotion( x,y);
            break;
    }
}
Start Function

```c
void startMotion(int x, int y)
{
    trackingMouse = true;
    redrawContinue = false;
    startX = x;
    startY = y;
    curx = x;
    cury = y;
    trackball_ptov(x, y, winWidth, winHeight, lastPos);
    trackballMove=true;
}
```
Stop Function

void stopMotion(int x, int y)
{
    trackingMouse = false;
    /* check if position has changed */
    if (startX != x || startY != y)
        redrawContinue = true;
    else
    {
        angle = 0.0;
        redrawContinue = false;
        trackballMove = false;
    }
}