1 Connecting User Input and Viewing, cont.

1.1 Rotating the view up vector

1. Rotation for 2D data reorients the view volume, but the view window stays in the x-y plane.
2. Rotation should not change the location of the VRP or the view direction (VPN)
3. The visual point of rotation is generally the middle of the viewing screen

Process

1. Store the initial click and calculate the angle relative to horizontal (that passes through the center of the screen, because that is the horizontal that passes through the view reference point). in python, the function
   
   \text{math.atan2( y, x)}
   
   calculates the angle relative to the x-axis defined by the vector \((x, y)\).

2. Store the initial VUP vector: \(VUP_0\)

3. For each mouse motion
   (a) Calculate the current mouse angle relative to horizontal (that passes through the center of the screen) using \text{atan2}.
   (b) Subtract the angles to get the amount of rotation \(\alpha\)
   (c) Generate a rotation matrix for the angle \(-\alpha\)
      
      \[
      R(-\alpha) = \begin{bmatrix}
      \cos \alpha & \sin \alpha & 0 & 0 \\
      -\sin \alpha & \cos \alpha & 0 & 0 \\
      0 & 0 & 1 & 0 \\
      0 & 0 & 0 & 1 \\
      \end{bmatrix}
      \]  \(1\)

   (d) Rotate the initial VUP vector and update the view object
      
      \[
      VUP_\alpha = R(-\alpha)VUP_0
      \]  \(2\)

   (e) Recalculate the view matrix
   (f) Calculate the new view locations of the data
   (g) Adjust the coordinates of the visual objects
2 3D Camera Control

Translation: using typical keyboard controls (e.g. wasd for forward/left/back/right) works well. Number pad controls also work well.

- Forward/backward moves the VRP along the VPN by some step size $\gamma$.
  \[ \text{VRP}_{t+1} = \text{VRP}_t + \gamma \text{VPN} \]  
  (3)

- Left/right moves the VRP along the U axis.
  \[ \text{VRP}_{t+1} = \text{VRP}_t + \gamma U \]  
  (4)

- Up/down moves the VRP along the VUP axis.
  \[ \text{VRP}_{t+1} = \text{VRP}_t + \gamma \text{VUP} \]  
  (5)

Rotation (fly-through): rotate around VUP, anchored at the VRP, for right-left motion of the mouse and rotate around the U-vector, anchored at the VRP, for up-down motion. The rotations apply only to the orientation of the view coordinate system, which is defined as a set of vectors and is translation invariant.

- Left-right motion: translate VRP to the origin, align the axes, rotate by $\theta_h$ about the Y axis, invert the alignment, then translate back. To undo a rotation, we can multiply by the inverse matrix. The inverse of a rotation matrix happens to be its transpose, which makes inverting rotations an easy thing to do.
  \[ X_h(\theta_h) = T(\text{VRP})R_{xyz}(U, \text{VUP}, \text{VPN})^T R_y(\theta_h)R_{xyz}(U, \text{VUP}, \text{VPN})T(-\text{VRP}) \]  
  (6)

- Up-down motion: align axes, rotate by $\theta_v$ about the X axis, invert the alignment
  \[ X_v(\theta_v) = T(\text{VRP})R_{xyz}(U, \text{VUP}, \text{VPN})^T R_x(\theta_v)R_{xyz}(U, \text{VUP}, \text{VPN})T(-\text{VRP}) \]  
  (7)

Use $X_h(\theta_h)$ and $X_v(\theta_v)$ to transform the VRP point (homogeneous coordinate of 1) and the U, VUP, and VPN vectors (homogeneous coordinate of 0). Then rebuild the view matrix to update the data. The effect is that of turning your head around while the world stays fixed. Note that if both $\theta_u$ and $\theta_v$ are non-zero, you can do a single transformation process with the two rotation matrices in the center of the expression.

Rotation (Circle around data): rotate the view plane around the center of the view volume.

- Left-right motion: translate the center of view volume extent to the origin, align the VRC axes, rotate around VUP, unalign, untranslate
  \[ X_h(\theta_h) = T(\text{VRP} + \frac{E_z}{2} \text{VPN})R_{xyz}(U, \text{VUP}, \text{VPN})^T R_y(\theta_h)R_{xyz}(U, \text{VUP}, \text{VPN})T(-\text{VRP} - \frac{E_z}{2} \text{VPN}) \]  
  (8)

- up-down motion: translate the center of view volume extent to the origin, align the VRC axes, rotate around U, unalign, untranslate
  \[ X_v(\theta_v) = T(\text{VRP} + \frac{E_z}{2} \text{VPN})R_{xyz}(U, \text{VUP}, \text{VPN})^T R_x(\theta_v)R_{xyz}(U, \text{VUP}, \text{VPN})T(-\text{VRP} - \frac{E_z}{2} \text{VPN}) \]  
  (9)
Use $X(\theta)$ to transform the VRP point (homogeneous coordinate of 1) and the U, VUP, and VPN vectors (homogeneous coordinate of 0). Then rebuild the view matrix to update the data. The effect is that of the data rotating in space while the user stays fixed.

Acknowledgement: Most of the material in this lecture is taken from Bruce Maxwell’s spring 2012 CS251 notes.