3D Rotation: more than just a hobby

Eric Yang’s special talent is the mental rotation of three-dimensional objects.

Mental rotation is Yang’s hobby.

“I can see textures and imperfections and the play of light and shadows on the objects I rotate, too,” Yang says.

David Foster Wallace
Outline

• Rotation from Mouse Event in VTK
  – Projects 2A / 2B
  – Other options
• Rotation Concepts
  – There’s going to be math. I think it will be fun.
  – There won’t be any math. It will be a blast!
• Applications
• Optional: Gimbal Lock
• References
Outline

- **Rotation from Mouse Event in VTK**
  - Projects 2A / 2B
  - Other options

- **Rotation Concepts**
  - There’s going to be math. I think it will be fun.
  - There won’t be any math. It will be a blast!

- **Applications**

- **Optional: Gimbal Lock**

- **References**
Camera Frame:
- \( v1 = \text{position} - \text{focus} \)
- \( v2 = \text{up} \)
- \( v3 = \text{up} \times v1 \)

Label the center of the window (0, 0).

Observe click at \((w, h)\) in window.

Let \( f \) be a factor based on zoom.

Rotate camera \( f \times w \) degrees around \( v2 \) and \( f \times h \) degrees around \( v3 \).
We can implement a different interaction style:

Call the center of the window (0, 0).

Mouse click down at (w1, h1) in window.

Mouse click release at (w2, h2) in window.

dw = w2 – w1, dh = h2 – h1.

Let f be a factor based on zoom.

Rotate camera actor f * dw degrees around up and f * dh degrees around up cross view.
VTK calls this version the *joystick* interaction style.

We applied the transformation to the camera.

VTK calls this version the *trackball* interaction style.

We applied the transformation to the actor.

Can you have joystick interaction style with actor transformation? Yes!

Can you write your own interaction style? Yes!
VTK

Camera Frame:
- $v_1 = \text{position} - \text{focus}$
- $v_2 = \text{up}$
- $v_3 = \text{up} \times v_1$

What if you want to rotate around $v_1$?

YOU CAN’T!

Mouse click gives $(w, h)$ position on window. No way to get three independent pieces of information (rotation about each axis) from two.
Outline

• Rotation from Mouse Event in VTK
  – Projects 2A / 2B
  – Other options

• Rotation Concepts
  – There’s going to be math. I think it will be fun.
  – There won’t be any math. It will be a blast!

• Applications
• Optional: Gimbal Lock
• References
From VTK to Concepts

• How does VTK store and apply rotations?
  – We’ve been working with matrices: DEMO

• Actually, VTK will try to save space and only compute and cache the matrix representation on demand. We will see different ways to store a rotation in a moment …

• First, an interesting question: why not just use three values – rotation about x, y, and z – and update these to produce any rotation?
Euler Angles

- Euler’s Rotation Theorem (pt 1): any rotation can be represented as rotations about the coordinate axes.

- This is less complicated than the full rotation matrix. Why don’t we just do this?

  **DEMO**

- It doesn’t work in general. See the optional section on Gimbal Lock.

However ... if your application doesn’t hit these bad angles and you really, really care about performance, this could be the way to go.

See this video for an example in Autodesk Maya: [link](#)

People in the twenty-first century are developing complex software.

People in the eighteenth century developed complex hats.
Matrix Representation

- This is what we have been using.

- **Pros:**
  - No bad rotations!
  - Can easily handle translation and scaling

- **Cons:**
  - 16 doubles to store 4x4 matrix
  - Multiplication is kind of slow

![Matrix Representation Example](http://xkcd.com/184/)

\[
\begin{bmatrix}
\cos 90^\circ & \sin 90^\circ \\
-\sin 90^\circ & \cos 90^\circ \\
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
\end{bmatrix} =
\begin{bmatrix}
a_1 \\
\sqrt{a_1^2 + a_2^2} \\
\end{bmatrix}
\]

← can be compressed
← can be optimized
Axis-Angle Representation

- Euler’s Rotation Theorem (pt 2): any 3D rotation can be represented as a single rotation around some vector.
- Pros:
  - Sometimes useful to conceptualize a transformation this way.
- Cons:
  - How do you actually implement it?
  - How do you multiply two of these? (You need trig functions. It’s slow.)
- An API usually has methods that will accept rotations in this form.
  (Under the hood: the transformation is likely converted to some other form before being applied, but the user doesn’t have to care.)
Quatertion Representation

- Any 3D rotation can be described by a 4-tuple called a quaternion.

- Pros:
  - Only need to store four doubles.
  - Multiplication is faster than matrices.
  - Renormalizing (due to floating point precision loss) is faster than matrices.
  - Methods for interpolating quaternions.

- Cons:
  - Applying the transformation to geometry is slower than matrices.
  - Related to axis-angle form, but not as easy to directly visualize.

Here as he walked by on the 16th of October 1843 Sir William Rowan Hamilton in a flash of genius discovered the fundamental formula for quaternion multiplication

\[ i^2 = j^2 = k^2 = ijk = -1 \]

& cut it on a stone of this bridge
Outline

• Rotation from Mouse Event in VTK
  – Projects 2A / 2B
  – Other options

• Rotation Concepts
  – There’s going to be math. I think it will be fun.
  – There won’t be any math. It will be a blast!

• Applications

• Optional: Gimbal Lock

• References
Applications

- Animation: could be lots of tiny transformations, matrix renormalization (orthogonalization) will be slow, want to use quaternions
- Robotics: the robot’s sensors need to communicate with each other about their orientation in world space
- Camera Calibration: noisy readings from camera need to be “averaged” – (s)lerping quaternions!
Outline

• Rotation from Mouse Event in VTK
  – Projects 2A / 2B
  – Other options

• Rotation Concepts
  – There’s going to be math. I think it will be fun.
  – There won’t be any math. It will be a blast!

• Applications

• Optional: Gimbal Lock

• References
Gimbal Lock

Gimbal: A mechanical device for rotation.

Widely used in navigation, robotics, photography, etc.

Actually need **four** gimbals to guarantee three degrees of freedom under any circumstances!
Gimbal Lock

Can represent axis rotations using matrices:

- **X**: \[
\begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(a) & -\sin(a) \\
0 & \sin(a) & \cos(a)
\end{bmatrix}
\]
- **Y**: \[
\begin{bmatrix}
\cos(b) & 0 & \sin(b) \\
0 & 1 & 0 \\
-\sin(b) & 0 & \cos(b)
\end{bmatrix}
\]
- **Z**: \[
\begin{bmatrix}
\cos(c) & -\sin(c) & 0 \\
\sin(c) & \cos(c) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Let \( b = 90 \):

- **Y**: \[
\begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
-1 & 0 & 0
\end{bmatrix}
\]

Suppose rotation order is XYZ:

- **XYZ**: \[
\begin{bmatrix}
0 & 0 & 1 \\
\sin(a + c) & \cos(a + c) & 0 \\
-\cos(a + c) & \sin(a + c) & 0
\end{bmatrix}
\]

change in either angle \( a \) or angle \( c \) has exact same effect!! (not good)
Gimbal Lock Solved

Can represent axis rotations using matrices:

• X: \[
\begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(a) & -\sin(a) \\
0 & \sin(a) & \cos(a)
\end{bmatrix}
\]
• Y: \[
\begin{bmatrix}
\cos(b) & 0 & \sin(b) \\
0 & 1 & 0 \\
-sin(b) & 0 & \cos(b)
\end{bmatrix}
\]
• Z: \[
\begin{bmatrix}
\cos(c) & -\sin(c) & 0 \\
\sin(c) & \cos(c) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
R_1: \begin{bmatrix}
a = 0, b = 90, c = 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{bmatrix}
\]

\[
R_2: \begin{bmatrix}
a \text{ free}, b = 0, c \text{ free} \\
\cos(c) & -\sin(c) & 0 \\
\cos(a)\sin(c) & \cos(a)\cos(c) & -\sin(a) \\
\sin(a)\sin(c) & \sin(a)\cos(c) & \cos(a)
\end{bmatrix}
\]

This works fine! The variables a and c are independent and this produces the desired transformation. This works because we use two full rotation matrices for the transformation.
Outline

• Rotation from Mouse Event in VTK
  – Projects 2A / 2B
  – Other options

• Rotation Concepts
  – There’s going to be math. I think it will be fun.
  – There won’t be any math. It will be a blast!

• Applications

• Optional: Gimbal Lock

• References
References

• VTK Examples: http://www.vtk.org/Wiki/VTK/Examples
  – lots of great VTK starter code

• Lecture Demos:
  http://ix.cs.uoregon.edu/~hampton2/graphics/

• Great book for learning more about the mathematical structure of rotations (and lots of other fun stuff): Algebra, Michael Artin

• The excerpt on the title page is from a story in Brief Interviews with Hideous Men, David Foster Wallace, pp. 196 – 199. It’s a great book!