Arcball Technique

Flavia R. Cavalcanti
Com S 437

What can we do with it?

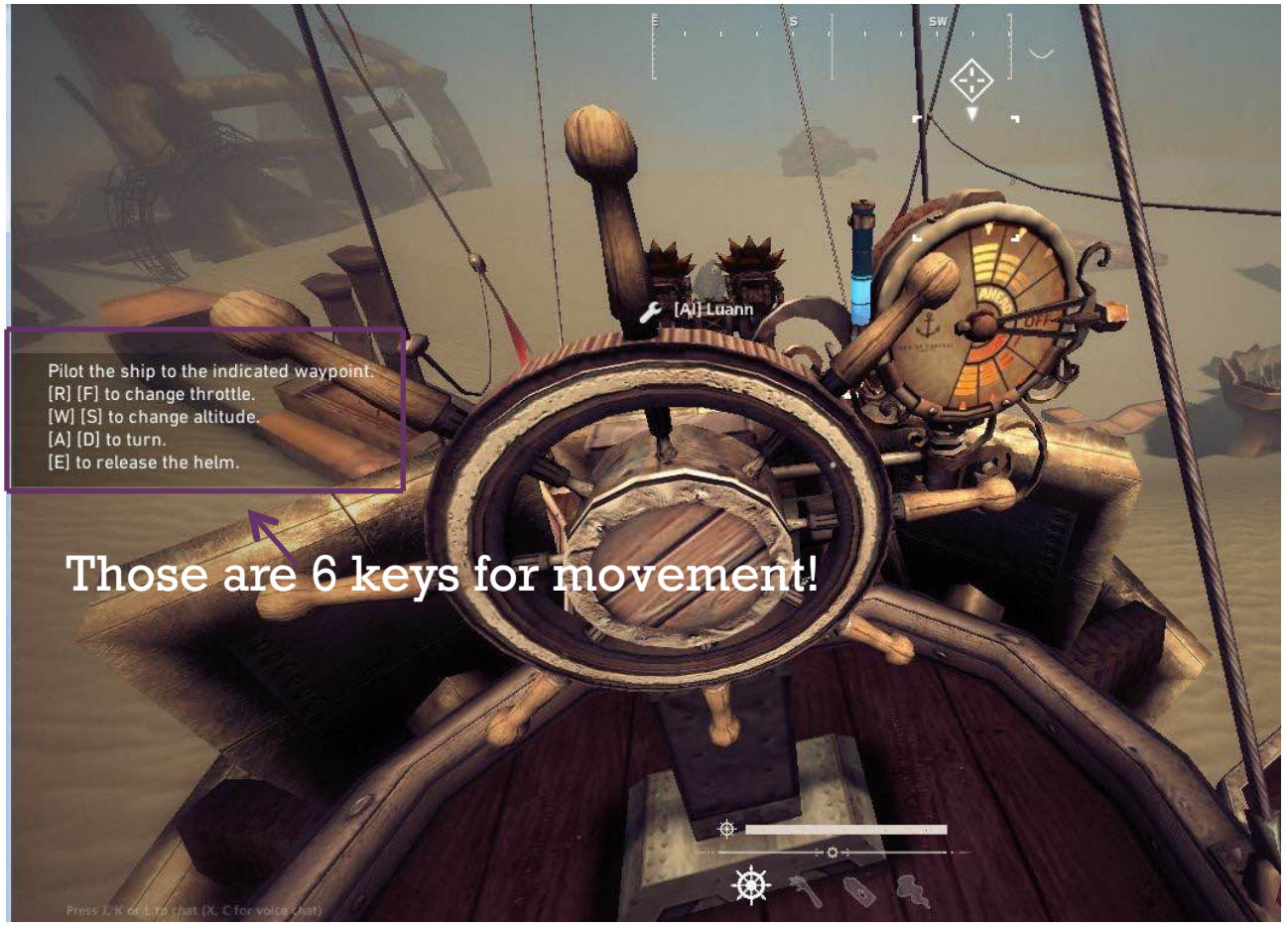
- Goal: Implement an algorithm to rotate an object, using the Arcball paradigm
- Input: Mouse movement
- Output: a rotation matrix for a 3D object

Demo

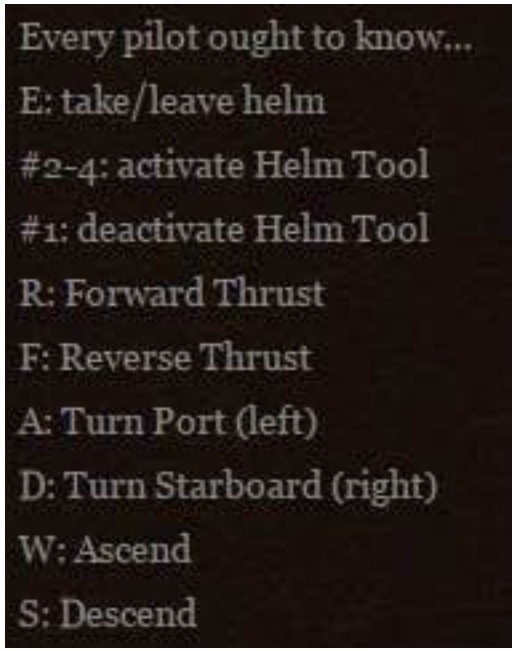
- Let's first see what arc ball can do before digging into the theory…
- Go demo go!

Why is it useful?

Convoluted Controls



Those Controls Get Worse…



Every pilot ought to know…

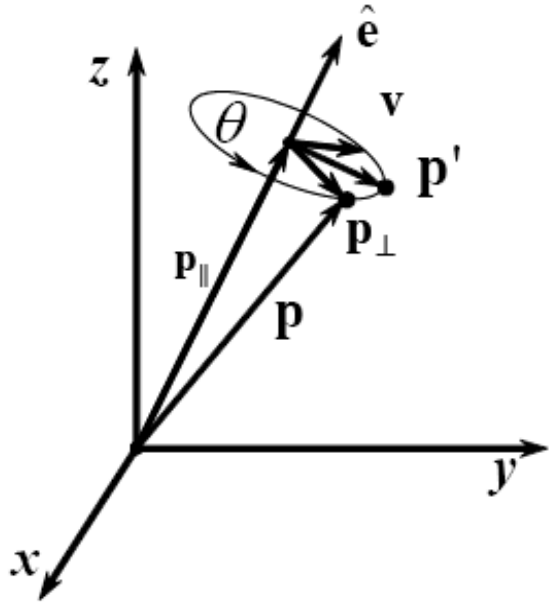
That you have **5** action keys.

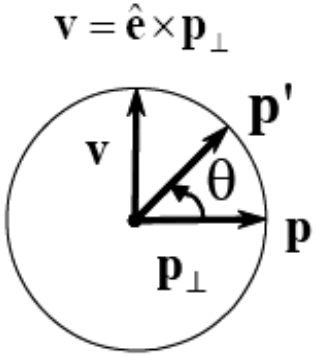
That you have **6** movement keys.

And as such…
You have to deal with **11** keys simultaneously per game!

Rotating Around Vector ê [2]

P' is described as an orthogonal base $\hat{e}, p_\perp, v, \left\|\hat{e}\right\| = 1$





$$p = p_\perp + p_\parallel$$

$$p' = R(p_\perp) + R(p_\parallel)$$

$$v = \hat{e} \times p_\perp$$

$$p' = p_\parallel + (\cos\theta)\, p_\perp + (\sin\theta)\, v$$

$$p_\parallel = (\hat{e} \cdot p)\hat{e}$$

$$p_\perp = p - (\hat{e} \cdot p)\hat{e}$$

$$v = \hat{e} \times p_\perp = \hat{e} \times (p - (\hat{e} \cdot p)\hat{e}) = \hat{e} \times p - (\hat{e} \cdot p)\hat{e} \times \hat{e} = \hat{e} \times p$$

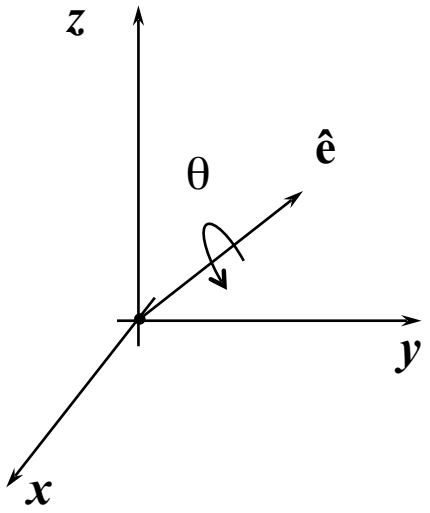
$$p' = (\hat{e} \cdot p)\hat{e} + (\cos\theta)(p - (\hat{e} \cdot p)\hat{e}) + (\sin\theta)(\hat{e} \times p)$$

$$\underline{p' = (\cos\theta)\, p + (1 - \cos\theta)(\hat{e} \cdot p)\hat{e} + (\sin\theta)(\hat{e} \times p)}$$

Rotation Matrix

- Rotation about an axis.

ê – axis of rotation
θ – rotation angle



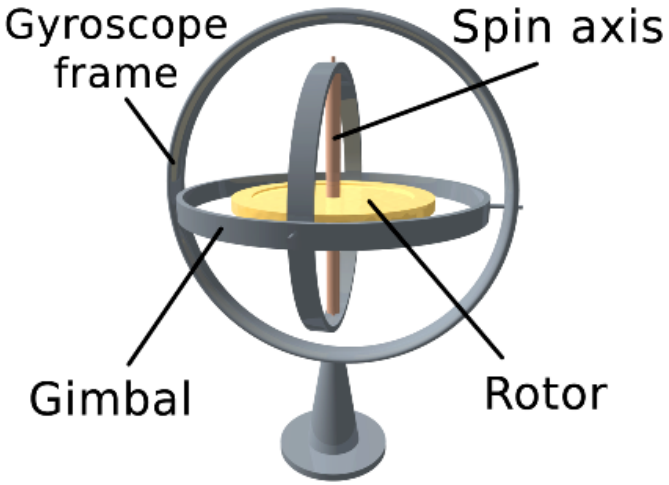
  - OpenGL - glRotatef(θ, ex, ey, ez)

- Rotation matrix, after changing coordinate system from to x,y,z: $p'(p) = (\cos\theta)p + (1-\cos\theta)(\hat{e}\cdot p)\hat{e} + (\sin\theta)(\hat{e}\times p)$
  - Columns of M are given by: p'(1,0,0), p'(0,1,0), p'(0,0,1)

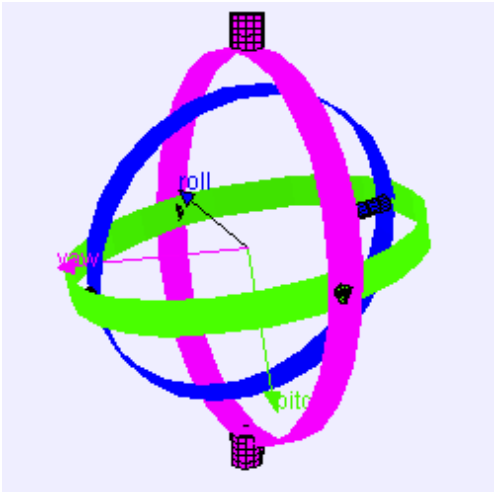
$$\mathbf{M} = \begin{bmatrix} \cos\theta + (1-\cos\theta)e_x^2 & e_y e_x(1-\cos\theta) - e_z\sin\theta & e_z e_x(1-\cos\theta) + e_y\sin\theta & 0 \\ e_x e_y(1-\cos\theta) + e_z\sin\theta & \cos\theta + (1-\cos\theta)e_y^2 & e_z e_y(1-\cos\theta) - e_x\sin\theta & 0 \\ e_x e_z(1-\cos\theta) - e_y\sin\theta & e_y e_z(1-\cos\theta) - e_x\sin\theta & \cos\theta + (1-\cos\theta)e_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Gimbal Lock [3]

- Loss of one degree of freedom in a 3D, 3-gimbal mechanism
- Occurs when the axes of 2 gimbals are driven into a parallel configuration -> locking the system into place



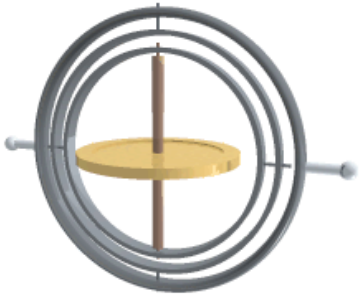
Gyroscope with 3 gimbals



Gimbal with roll, pitch, and yaw axis

Gimbal Lock Sucks When You're in Space [5]

- Problems with Apollo 13
- Apollo used an **I**nertial **M**echanism **U**nit for navigation, same as airliners do today.
- Given a starting point, such a system can record where you are in relation to that point at any time. Each gyroscopes will have a "null point", where the gyro lines up perfectly with its gimbals. If all three enter this state at once, the platform instantly loses it knowledge of where it is. It goes into "gimbal lock".



Gimbal Lock In CG [4]

- Bad Analogy – problem only occurs with Euler angles
  - Blender and Maya use three numbers, R_x, R_y, R_z, instead of a 4 dimensional array (quaternion) to store a rotation.
  - A general rotation R is then calculated, by multiplying these three matrices (or, some say, applying three "rolls"):

$$R(\theta_1,\theta_2,\theta_3) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta_1 & 0 \\ 0 & -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & 0 & -\sin\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_3 & \sin\theta_3 & 0 & 0 \\ -\sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Euler Angles and Gimbal Lock [10]

- If the y-roll happens to be 90°, then the "*lock*" occurs:

$$R(\theta_1,\theta_2,\theta_3)=\begin{pmatrix} c_2c_3 & c_2s_3 & -s_2 & 0 \\ s_1s_2c_3-c_1s_3 & s_1s_2s_3+c_1c_3 & s_1c_2 & 0 \\ c_1s_2c_3+s_1s_3 & c_1s_2s_3-s_1c_3 & c_1c_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, s_i=\sin\theta_i, c_i=\cos\theta_i$$

$$R(\theta_1,\frac{\pi}{2},\theta_3)=\begin{pmatrix} 0 & 0 & -1 & 0 \\ \sin(\theta_1-\theta_3) & \cos(\theta_1-\theta_3) & 0 & 0 \\ \cos(\theta_1-\theta_3) & -\sin(\theta_1-\theta_3) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, s_2=1, c_2=0$$

- Loss of freedom in the y - axis

How to avoid Gimbal Lock?

- Lock occurs because Euler angles are being changed, in a fixed order, one at a time, so axes can line up.
- Another possibility is using a “fourth gimbal”, not allowed to align, which take us to a…
- Better solution-> quaternions.

Sir Rowan William Hamilton

- Hamilton, in 1830, tried to generalize complex numbers to R^3 (a complex volume, with two imaginary axes)
  - $a + ib$, $i^2 = -1$
- He realized in 1843 that it was impossible on R^3 (triples of numbers are not closed under multiplication), but not on R^4 or R^8
  - $q = a + bi + cj + dk = (a,v)$, $i^2 = j^2 = k^2 = ijk = -1$
  - $ij = k$, $jk = i$, $ki = j$, $kj = -i = ji = -k$, $ik = -j$
- The multiplication of two quaternions is a quaternion
  - Closed under multiplication => mathematical group.
  - Quaternion multiplication is not commutative! $q_1.q_2 \neq q_2.q_1$

Broome Bridge in Dublin [6]

Quaternions [2]

- Complex Numbers on R^4
  - $q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$
  - $q = (s, v)$,
    - s – real component and
    - v – vector representing the imaginary component
- Simplifies the calculation of rotations about an axis

$$q = \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\hat{\mathbf{e}}\right)$$

ê – unit vector (rotation axis)
θ – rotation angle

- Rotations using product of quaternions
  - Rotation of a point p about an axis v
    - $R(p) = q\, p\, q^{-1}$
      - $p = (0, r)$ - point represented as a pure quaternion
      - $q = (s, v)$ - rotation (angle and axis) represented as a quaternion

Properties of Quaternions [2]

- Multiplication and Addition

$$q_1 = (s_1, v_1),\ q_2 = (s_2, v_2)$$
$$q_1 q_2 = (s_1 s_2 - v_1 \cdot v_2,\ s_1 v_2 + s_2 v_1 + v_1 \times v_2)$$
$$q_1 + q_2 = (s_1 + s_2,\ v_1 + v_2)$$

- Conjugate and reciprocal: $q^{-1} = \dfrac{\bar{q}}{\|q\|^2}$

$$\bar{q} = (s, -v)$$

$$q\bar{q} = (s, v)(s, -v) = s^2 + \|v\|^2 = s^2 + x^2 + y^2 + z^2 = \|q\|^2$$

- Norm: $\|q\| = \sqrt{q\bar{q}} = \sqrt{\bar{q}q} = \sqrt{s^2 + \|v\|^2}$

- Rotation: take a pure quaternion $p = (0, r)$ and a **unit** quaternion $q=(s,v)$, where $q\bar{q}=1$, and define:
  - $R(p) = q\, p\, q^{-1}$

Orientations as Quaternions [2]

$$q = (\cos(\theta), \sin(\theta)\hat{\mathbf{e}}),\ q^{-1} = (\cos(\theta), -\sin(\theta)\hat{\mathbf{e}}),\ \|\hat{\mathbf{e}}\| = 1,\ p = (0, r)$$

$$R_q(p) = qpq^{-1} = (0, (\cos^2\theta - \sin^2\theta)r + 2\sin^2\theta(\hat{e}\cdot r)\hat{e} + 2\cos\theta\sin\theta(\hat{e}\times r)) =$$

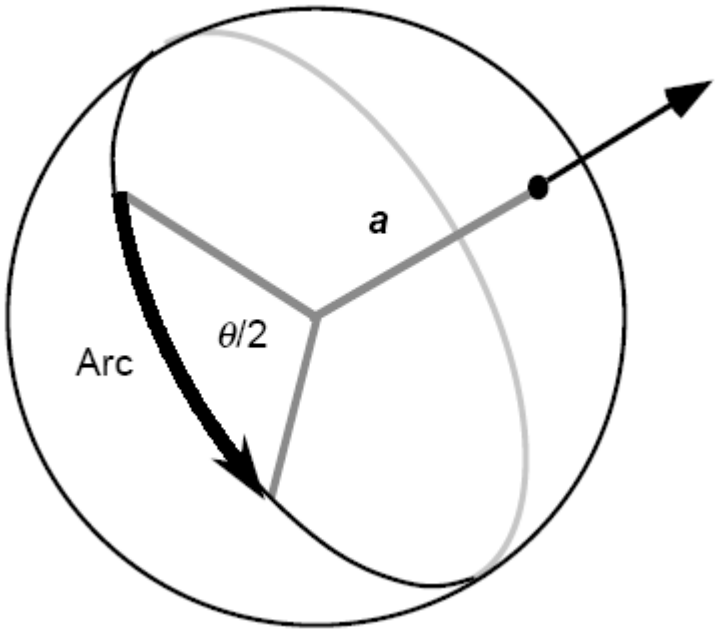
$$= (0, (\cos 2\theta)r + (1-\cos 2\theta)(\hat{e}\cdot r)\hat{e} + \sin 2\theta(\hat{e}\times r))$$

- This is exactly the same equation in transparency 4
  - aside from a factor of 2, and *p* written as *r*
- Therefore, orientations can be parameterized as:

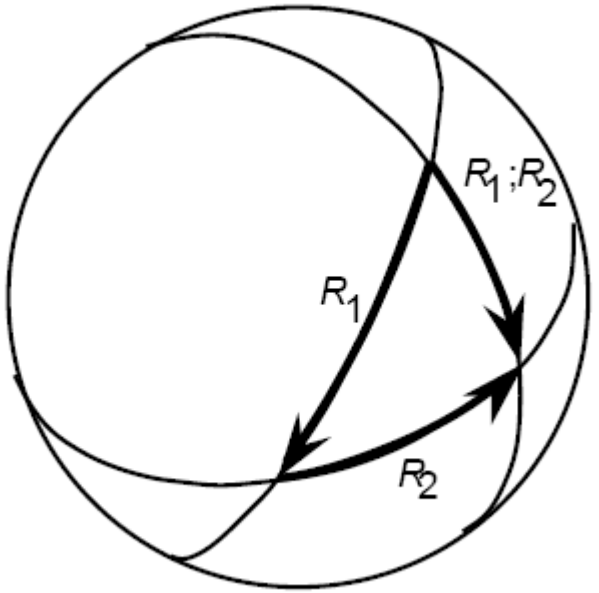
$$q = \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\hat{\mathbf{e}}\right),\ \|\hat{\mathbf{e}}\| = 1$$

ArcBall Technique

- Interface to rotate a 3*D* object using just a mouse



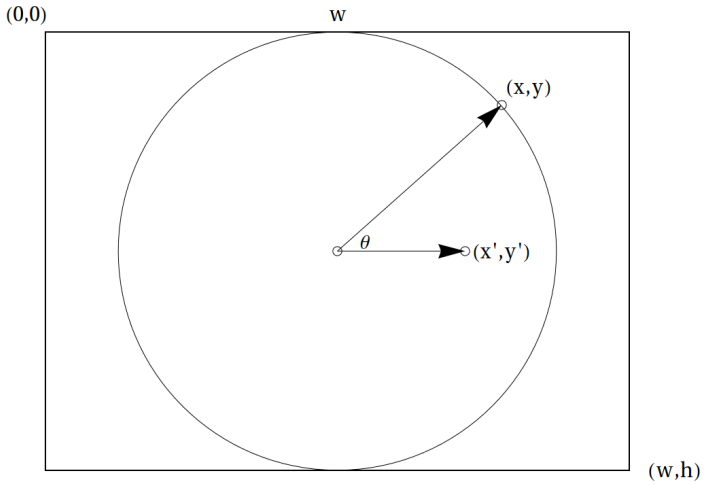
Arc
Interpretation
a – rotation axis
θ – rotation angle



Arc Combination

Sphere As a Virtual Trackball [7]

1. Consider screen coordinates from (0,0) to (w,h).
2. Mouse coordinates (x,y) are used to sample a function, z(x,y):



3. Points inside the circle are mapped onto a sphere, while points outside are mapped onto plane z=0.

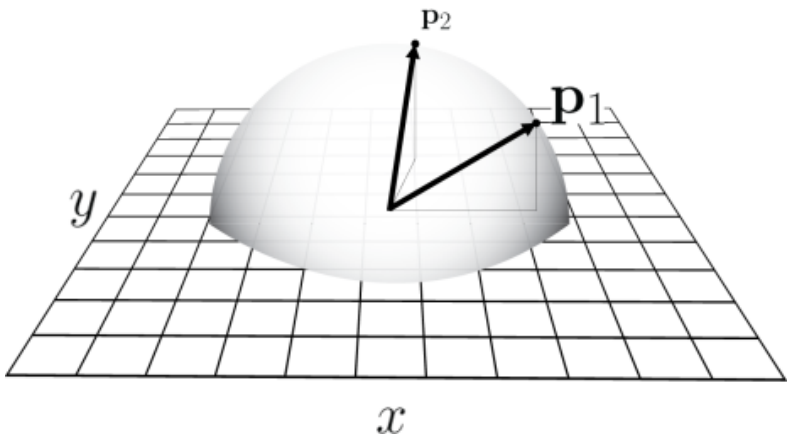
Hemisphere Inscribed in Screen Area [7]

- Circle centered at the origin:

$$x^2 + y^2 + z^2 = R^2 \Rightarrow z = \sqrt{R^2 - x^2 - y^2}$$

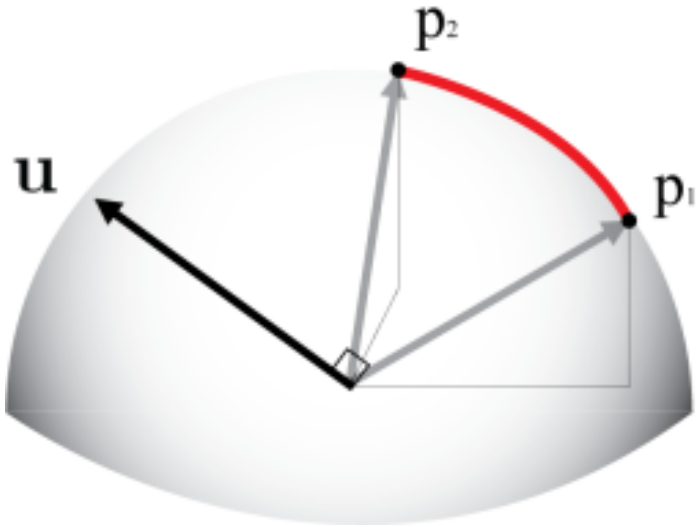
- Circle centered at (w/2, h/2):

$$z(x,y) = \sqrt{\left(\frac{h}{2}\right)^2 - \left(x - \frac{w}{2}\right)^2 - \left(y - \frac{h}{2}\right)^2}$$



Angle-axis Transformation [7]

- Mouse dragged from $p_1 = (x, y, z(x, y))$ to $p_2 = (x', y', z'(x', y'))$
- Rotation angle: $\theta = \arccos\left(\frac{p_1 \cdot p_2}{||p_2|| \cdot ||p_1||}\right)$
- Rotation axis: $u = \frac{p_1 \times p_2}{||p_2|| \cdot ||p_1||}$



Rotation as Two Reflections [8]

- Practical implementations use two reflections instead of a rotation
  - Product of two reflections in intersecting planes is equivalent to a rotation.
  - The axis of this rotation is the line of intersection of the planes, while the angle of rotation is twice the angle between the two planes.

Example [8]

- Put two mirrors at 90° to each other and put an object in front of them.
- The reflection on the second mirror will appear to have been rotated by 180° relative to the original object.
- The rotation angle 180° is twice the angle between mirrors, which is 90° in this example.

Simple-Rotator.js [10]

- setView = function(viewDirVec, viewUpVec, viewDistance)
  - Create view matrix using (u_x, u_y, u_z) as column vectors
    - viewDirectionVector --> u_z
    - viewUpVector x u_z --> u_x
    - u_z x u_x --> u_y
- getViewMatrix = function()
  - Return a rotated view matrix (after dragging the mouse)
- function toRay(x,y)
  - Create a ray from the center of the screen to the point on the sphere, which projects onto (x,y)
  - Sphere origin is the center of screen, and radius is the minimum (width, height)

Simple-Rotator.js [10]

- function doMouseDrag(evt)
  - Create two rays: ray_1,ray_2
  - map them to world coordinates, by using the view matrix inverse (transpose)
  - applyTransvection(ray_1,ray_2);
- function applyTransvection(e_1, e_2)
  - Rotate view matrix (u_x, u_y, u_z) around (e_1 x e_2), by applying two reflections on each vector.

That's All Folks

- Questions?

References

- [1] *ARCBALL: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse* - Ken Shoemake
- [2] *Advanced Animation and Rendering Techniques* – Alan Watt
- [3] https://www.youtube.com/watch?v=zc8b2Jo7mno
- [4] https://sundaram.wordpress.com/2013/03/08/ mathematical-reason-behind-gimbal-lock- in-euler-angles/
- [5] http://www.universetoday.com/119984/13-more-things-that- saved-apollo-13-part-9-avoiding-gimbal-lock/

References

- [6] https://www.youtube.com/watch v=mHVwd8gYLnI&nohtml5=False
- [7] https://braintrekking.wordpress.com/2012/08/21/ tutorial-of-arcball- without-quaternions/
- [8] https://sureshemre.wordpress.com/ 2014/07/19/a-rotation-equals-two- reflections/
- [9] http://math.hws.edu/eck/cs424/notes2013/webgl/ skybox-and-reflection/simple-rotator.js
- [10] http://en.wikipedia.org/wiki/Euler_angles