Understanding Quaternions

Jim Van Verth
Software Engineer, Google
jim@essentialmath.com
G+: vintagejim
Twitter: cthulhim

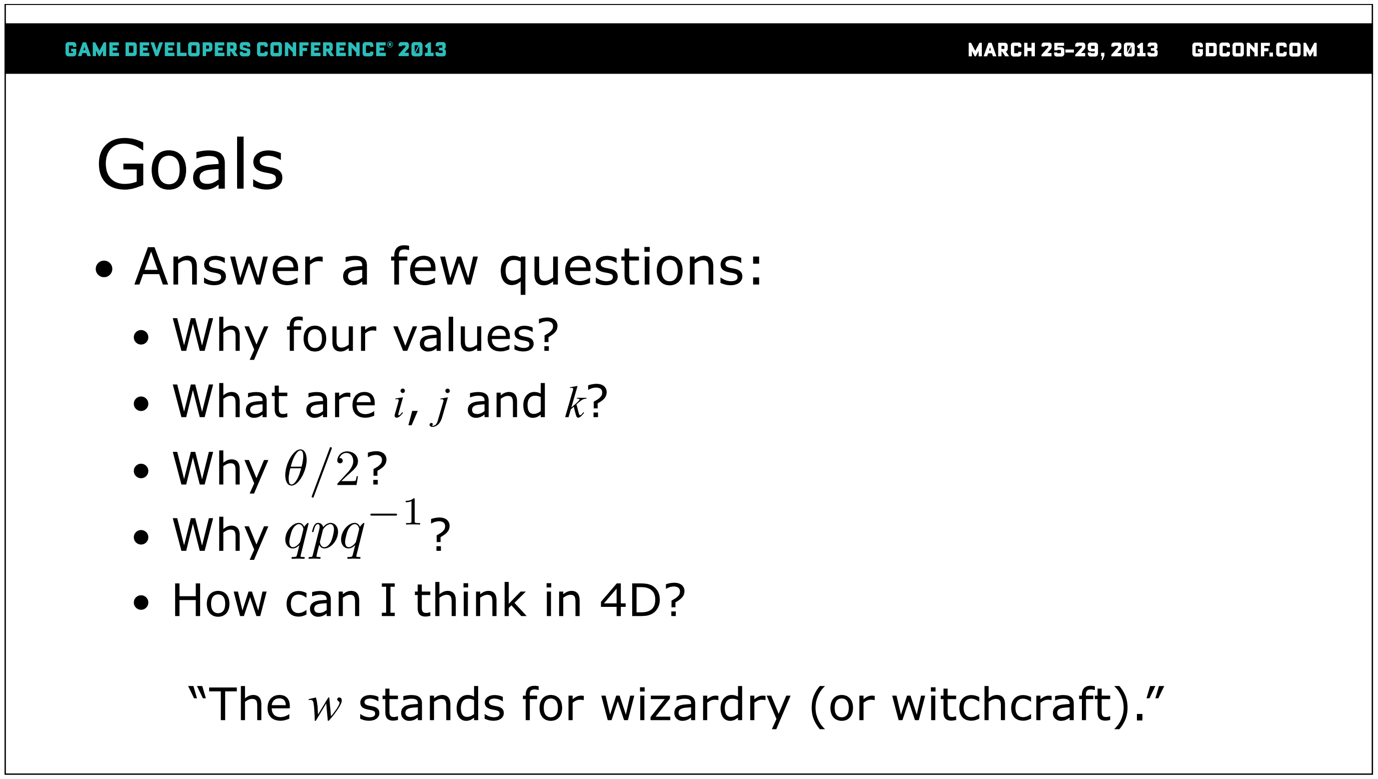
GAME DEVELOPERS CONFERENCE 2013
SAN FRANCISCO, CA
MARCH 25–29, 2013
EXPO DATES: MARCH 27–29

Goals

- Explain quaternions
- Not so much why they’re useful
- But how they work

Goals

- Quaternions, briefly
  - Used to rotate vectors
  - Four values: $w + xi + yj + zk$
  - Build via half angle $\theta/2$
  - Rotate via qpq^{-1}
  - Compact, normalize well
  - Excellent for interpolation



Goals

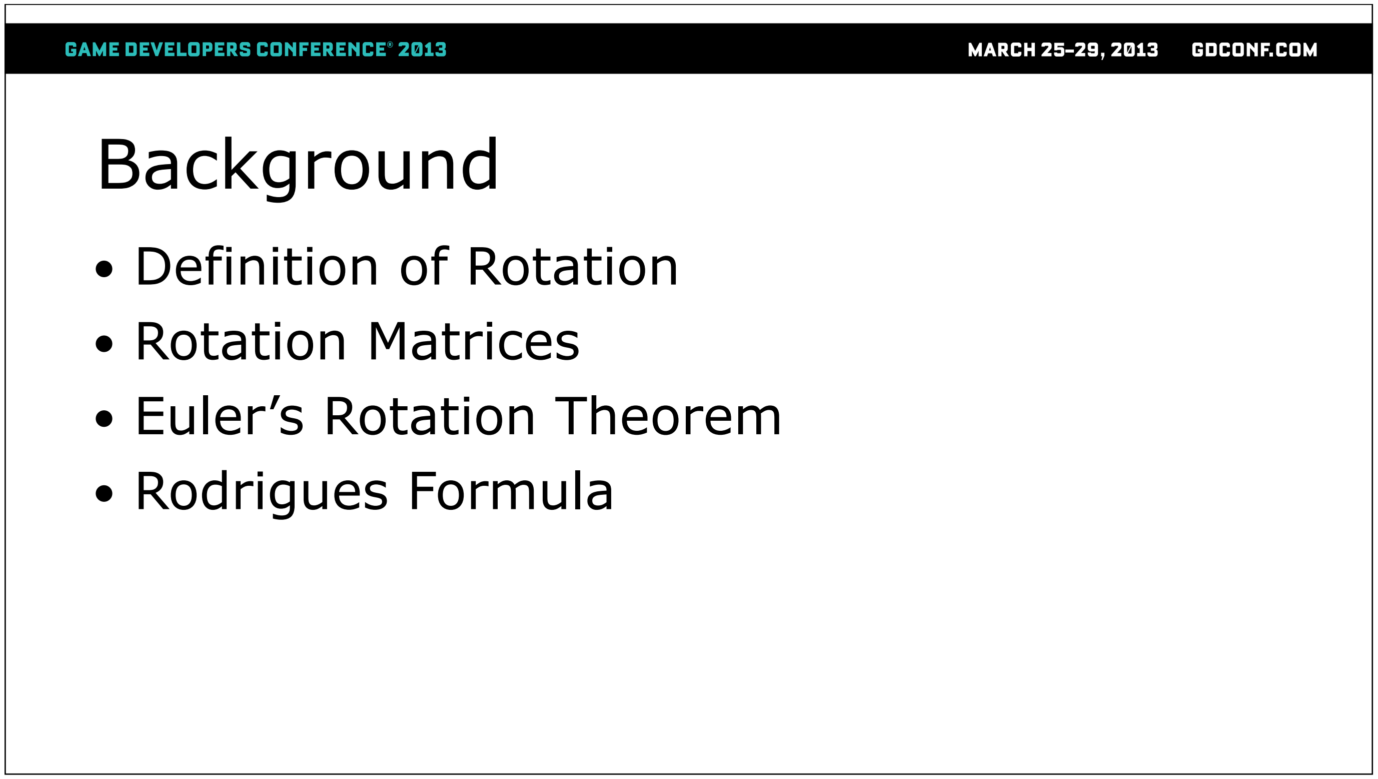
- Answer a few questions:
  - Why four values?
  - What are i, j and k?
  - Why $\theta/2$?
  - Why qpq^{-1}?
  - How can I think in 4D?

"The w stands for wizardry (or witchcraft)."

I probably won't meet these goals. Quaternions are pretty tough, and rotations show up in a lot of unexpected places. Best I can do in an hour is present a modicum of information and hope that some of it sticks.

Outline

- Background
- Complex Numbers
- Quaternions



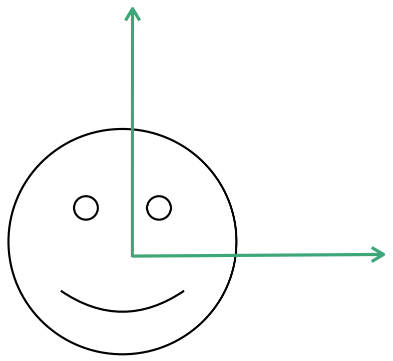
Background

- Definition of Rotation
- Rotation Matrices
- Euler's Rotation Theorem
- Rodrigues Formula

Part of this is somewhat historical -- though a bit out of order

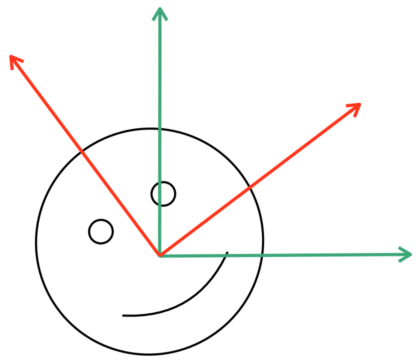
Defining Rotation

- Reference frame



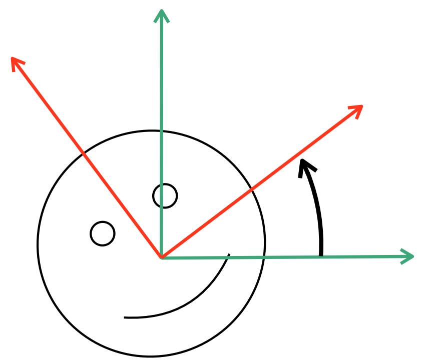
Defining Rotation

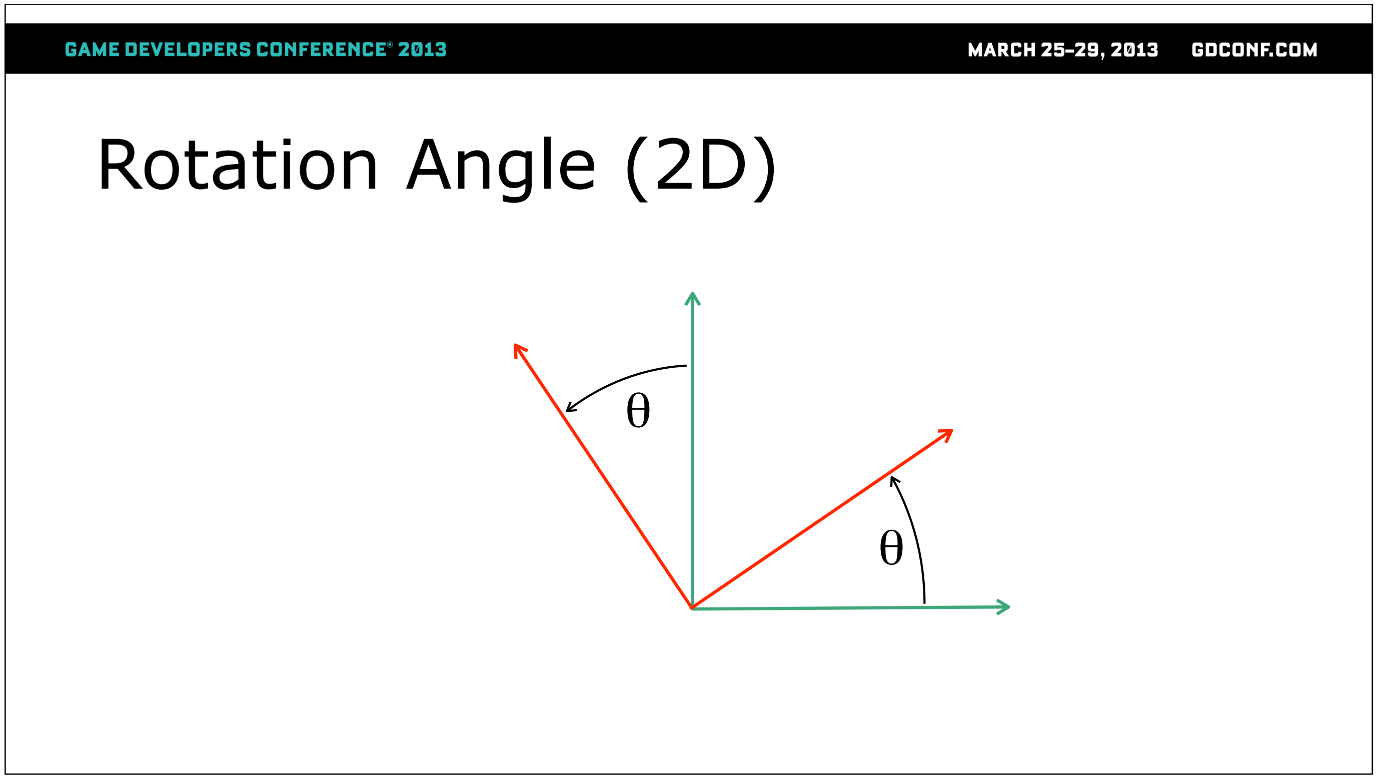
- Orientation relative to reference frame



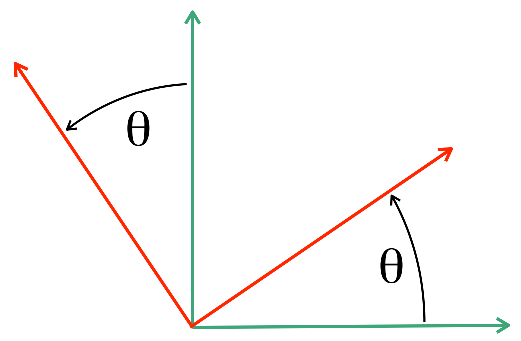
Defining Rotation

- Rotation transforms from one orientation to another

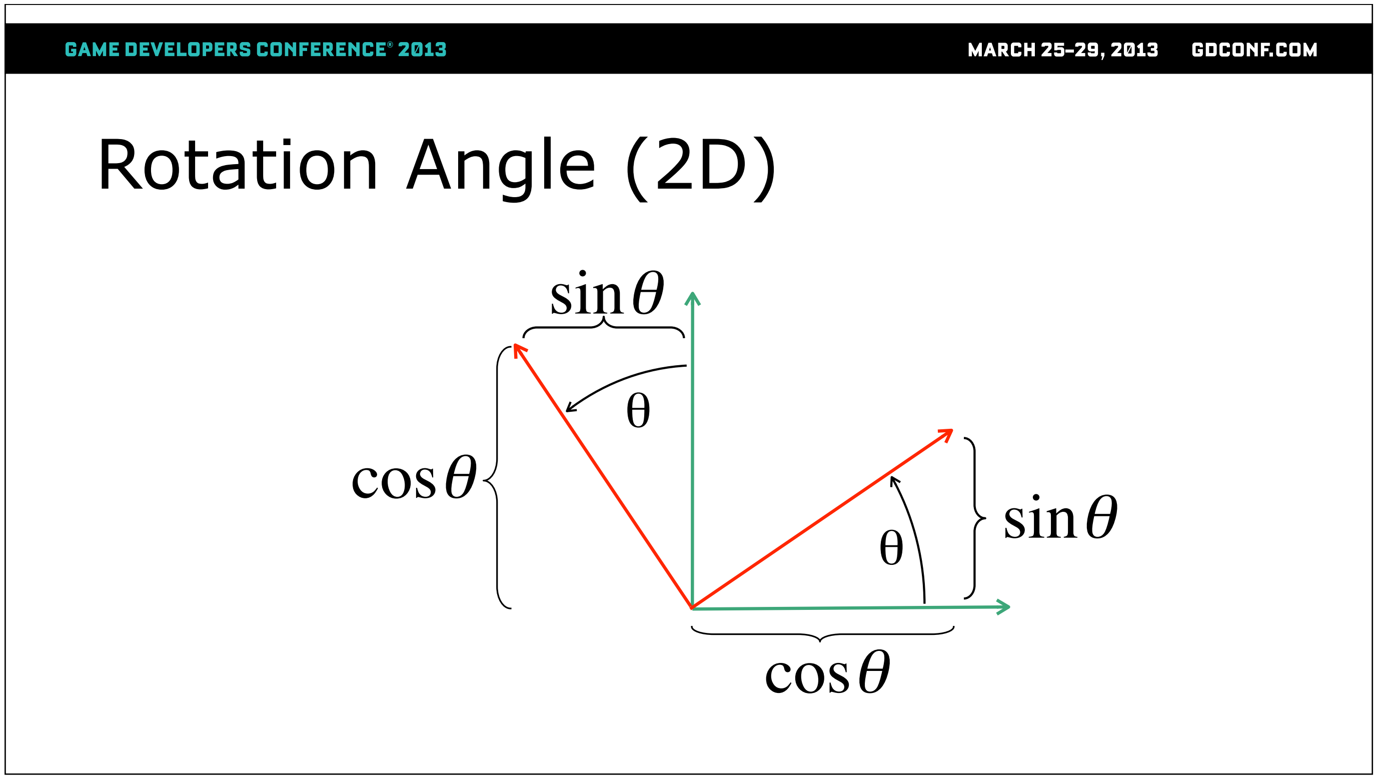




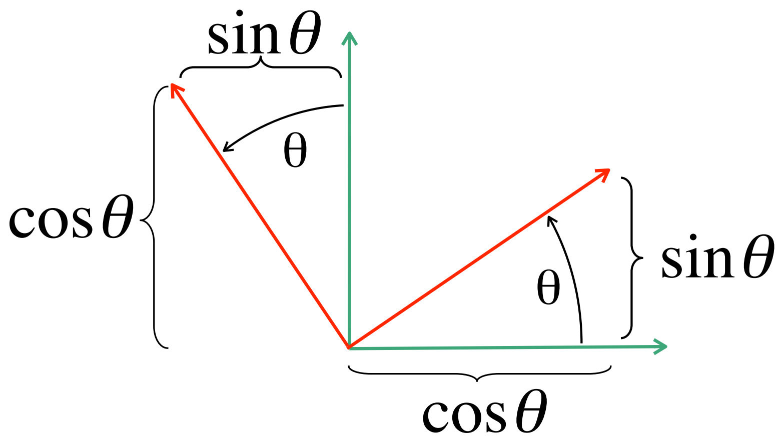
Rotation Angle (2D)



Say we want to rotate (in 2D) by an angle theta. The basis vectors will change like this.

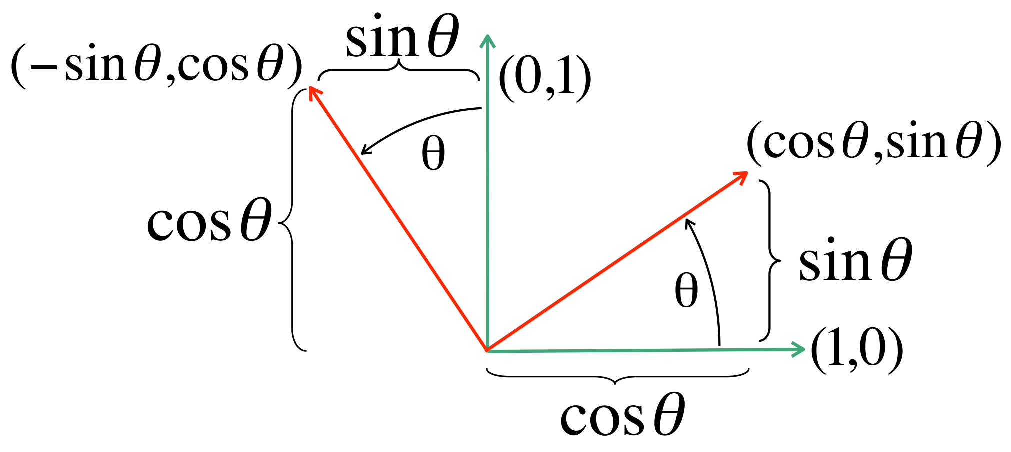


Rotation Angle (2D)

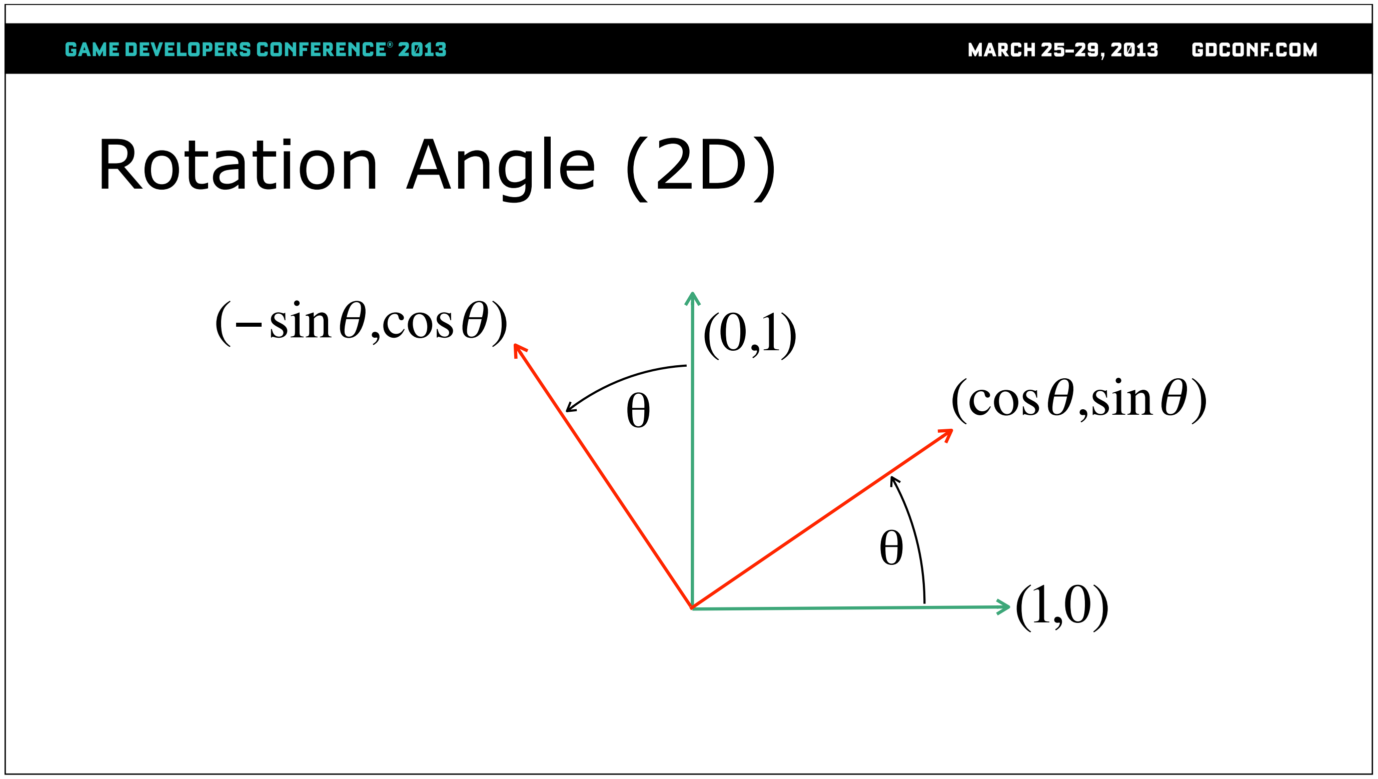


We can represent these new basis vectors as distances along the old basis vectors.

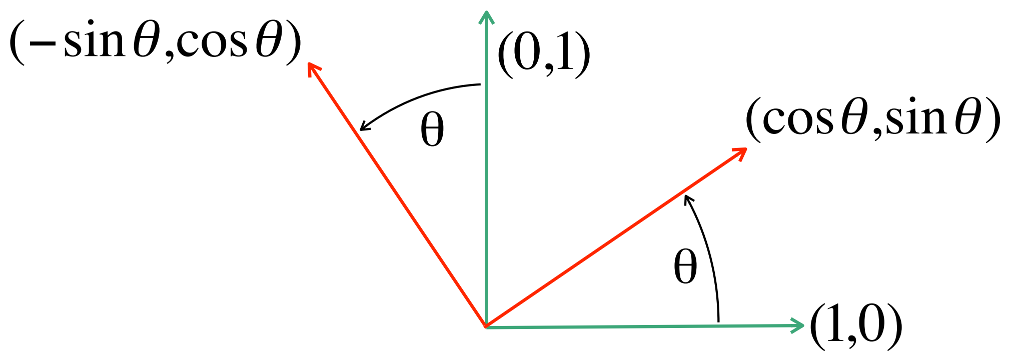
Rotation Angle (2D)



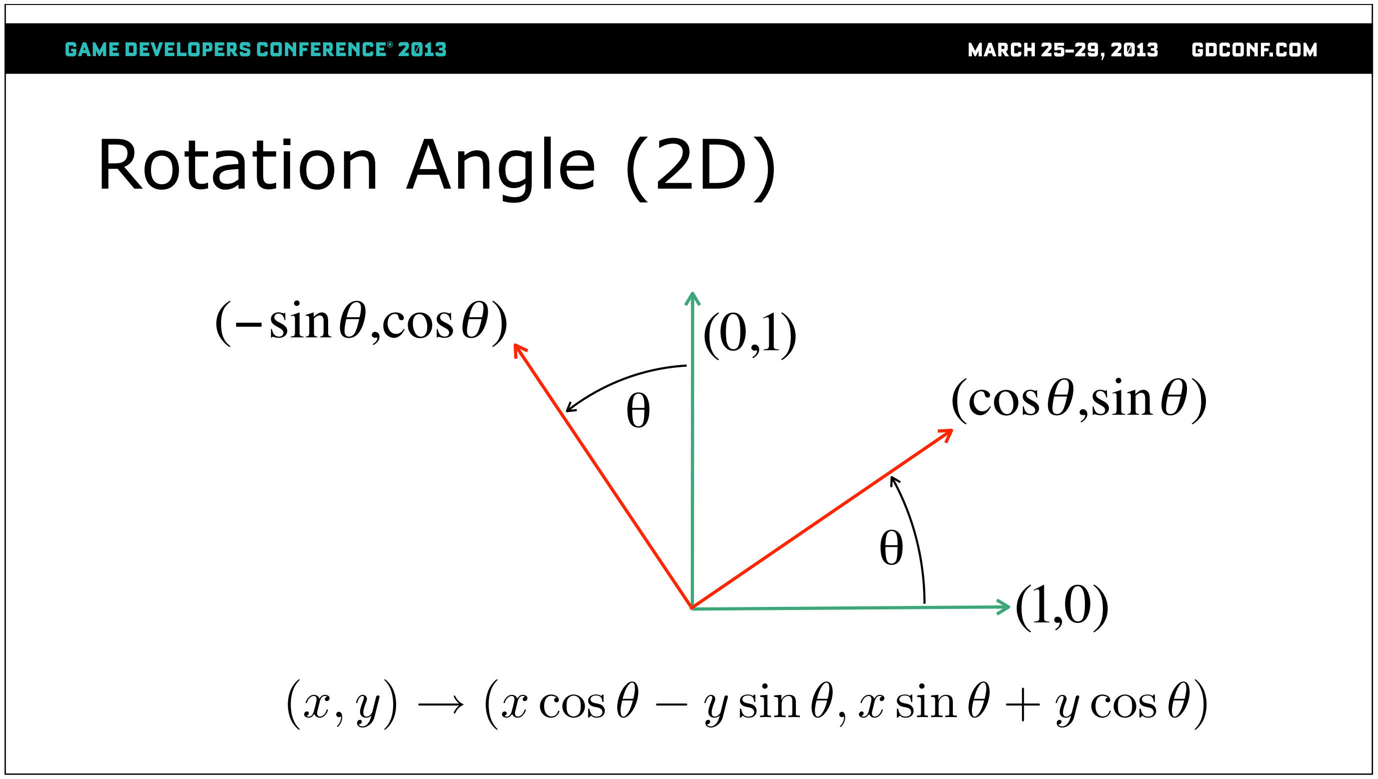
And from there get the new basis vectors



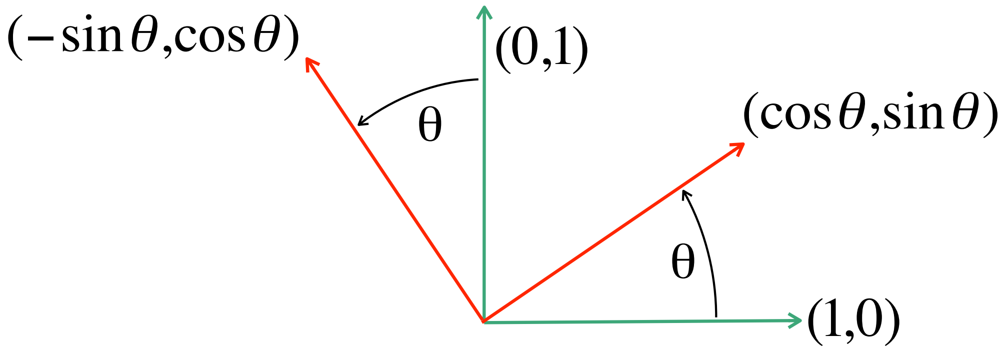
Rotation Angle (2D)



(just to make it clearer, let’s drop the cruft and just show the vectors)



Rotation Angle (2D)



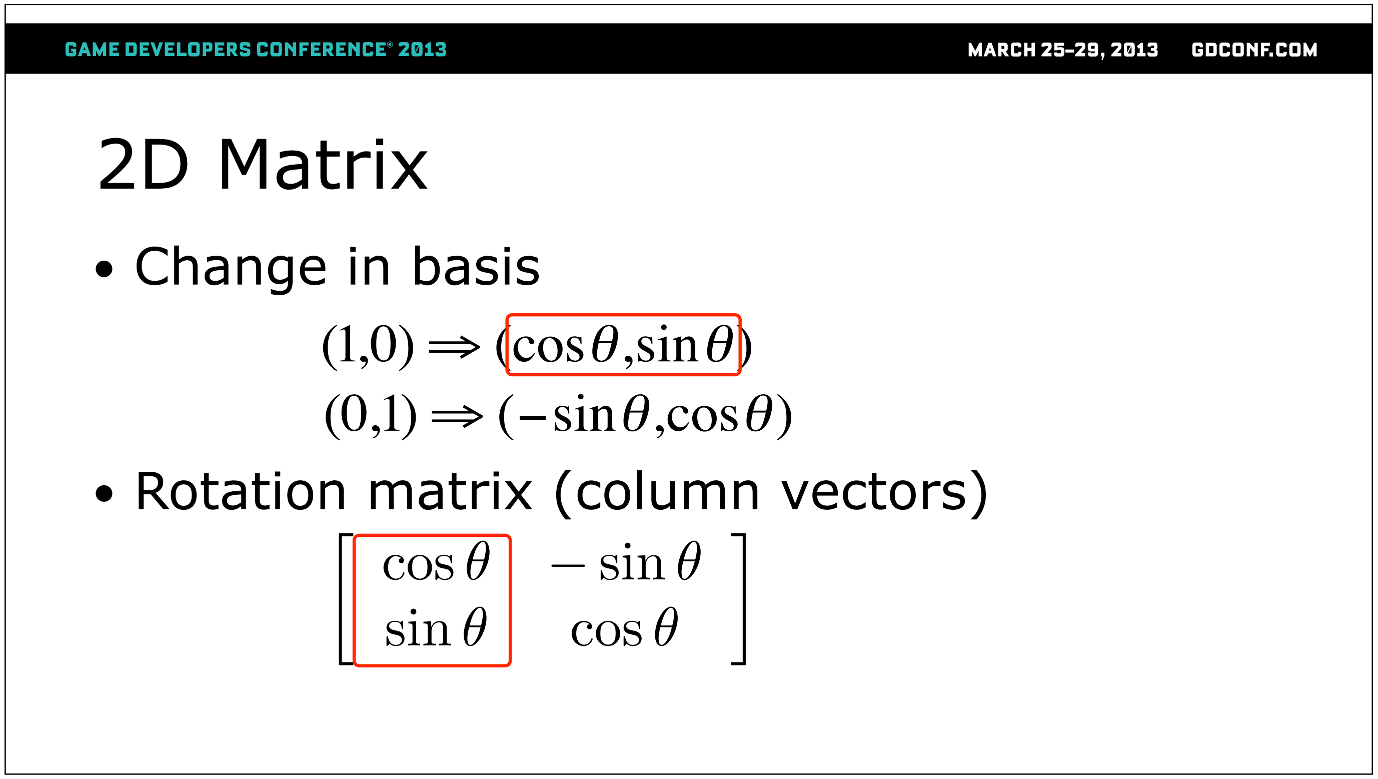
$$(x, y) \rightarrow (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$$

Knowing this we can create a 2D rotation transformation as follows:

Rotation Matrix

- Idea: Bake new basis in matrix
- Multiply by vector to rotate
- Matrix represents transformation

Squirrel (probably) already covered this, but just to review



GAME DEVELOPERS CONFERENCE® 2013 MARCH 25-29, 2013 GDCONF.COM

2D Matrix

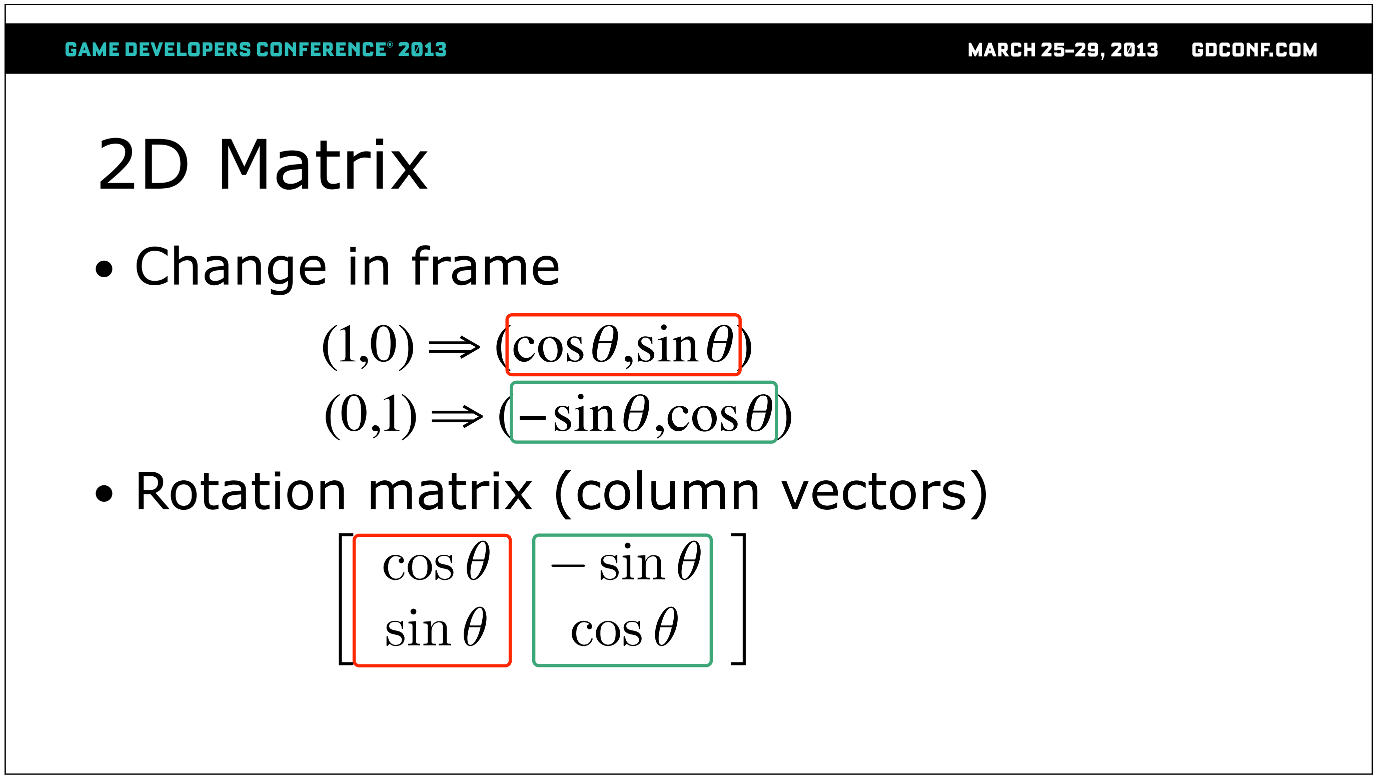
- Change in basis

$$(1,0) \Rightarrow (\boxed{\cos\theta,\sin\theta})$$
$$(0,1) \Rightarrow (-\sin\theta,\cos\theta)$$

- Rotation matrix (column vectors)

$$\left[\begin{array}{cc} \boxed{\begin{matrix}\cos\theta \\ \sin\theta\end{matrix}} & \begin{matrix}-\sin\theta \\ \cos\theta\end{matrix} \end{array}\right]$$

Our x axis basis gets transformed to cos theta, sin theta -- so that's the first column in our matrix.



2D Matrix

- Change in frame

$$(1,0) \Rightarrow (\boxed{\cos\theta,\sin\theta})$$
$$(0,1) \Rightarrow (\boxed{-\sin\theta,\cos\theta})$$

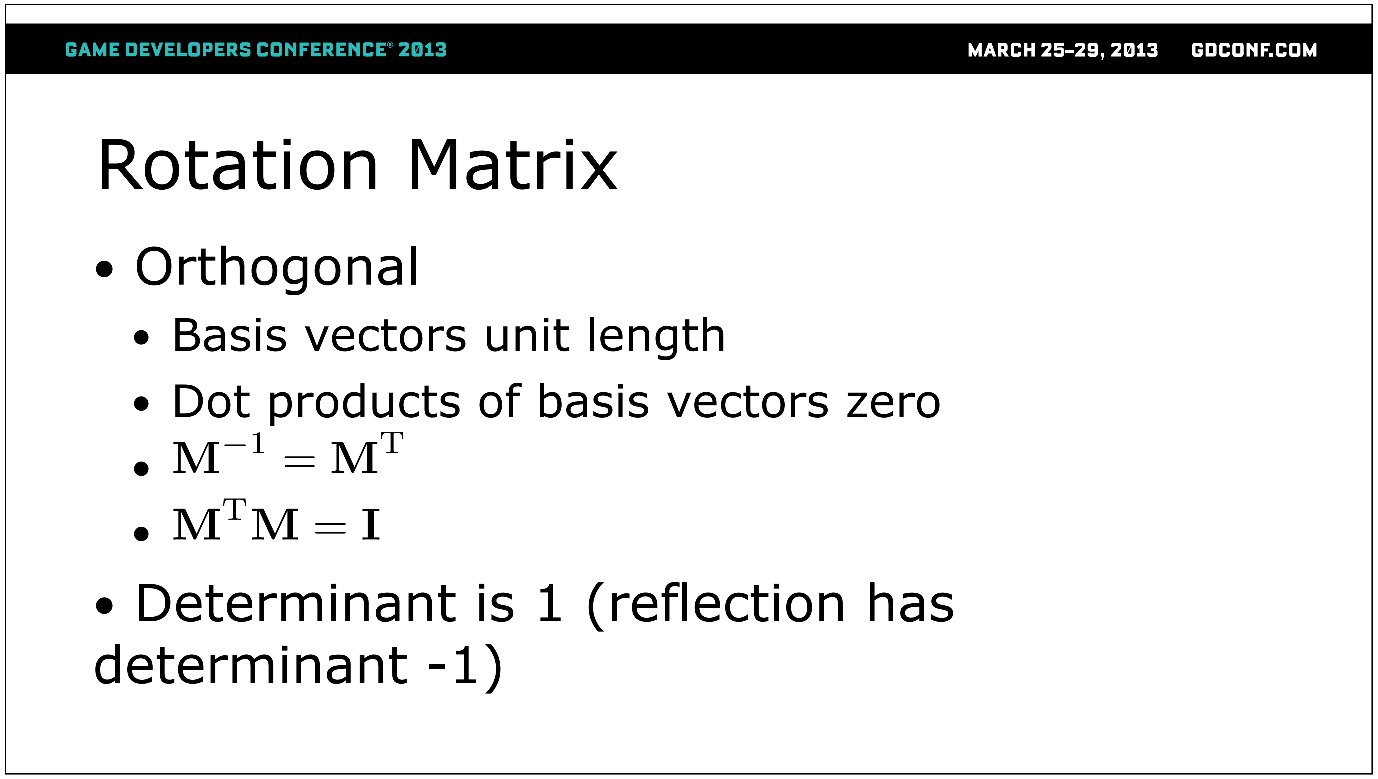
- Rotation matrix (column vectors)

$$\begin{bmatrix} \boxed{\begin{matrix}\cos\theta \\ \sin\theta\end{matrix}} & \boxed{\begin{matrix}-\sin\theta \\ \cos\theta\end{matrix}} \end{bmatrix}$$

And similarly for the transformed y axis.

3D Matrix

- Much the same as 2D
  - Map transformed axes, store as columns (or rows) of matrix
  - Rotate via vector multiplication



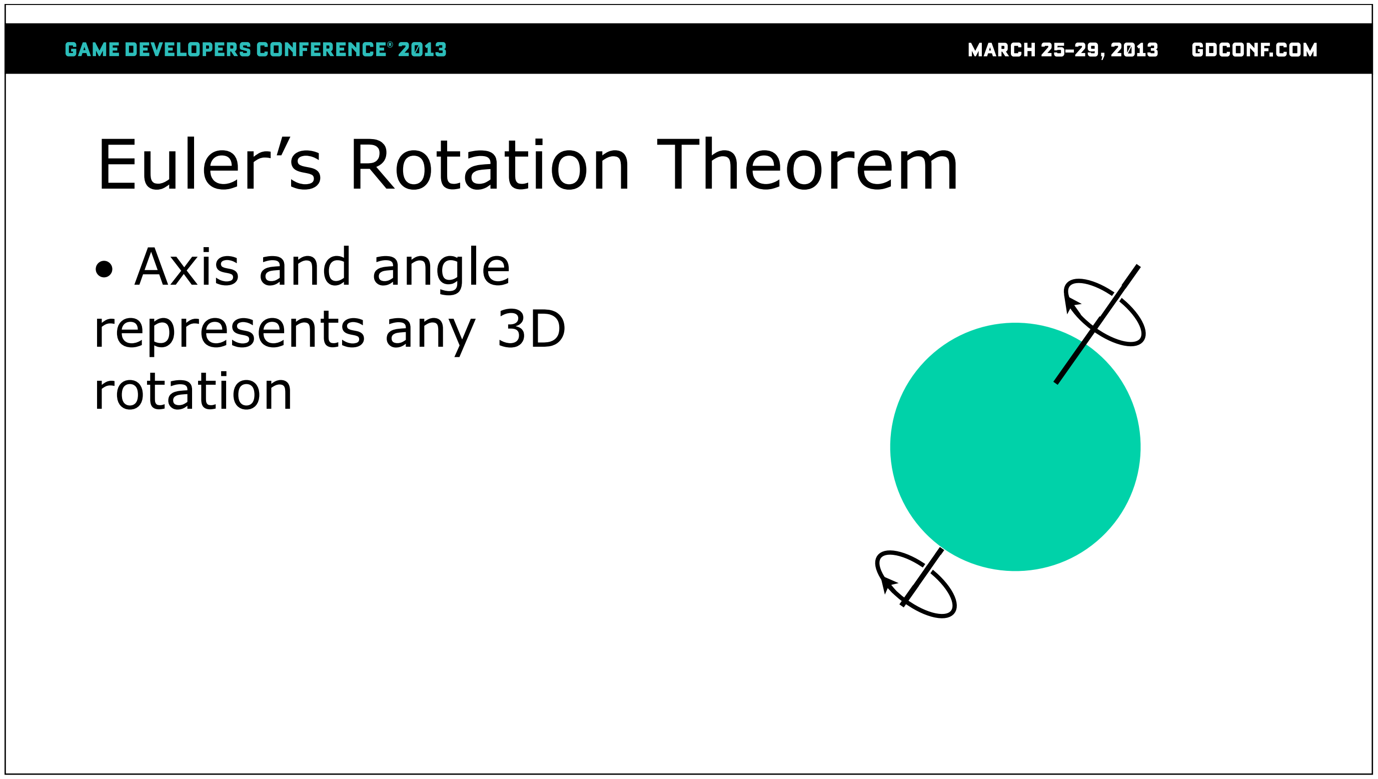
Rotation Matrix

- Orthogonal
  - Basis vectors unit length
  - Dot products of basis vectors zero
  - $\mathbf{M}^{-1} = \mathbf{M}^{\mathrm{T}}$
  - $\mathbf{M}^{\mathrm{T}}\mathbf{M} = \mathbf{I}$
- Determinant is 1 (reflection has determinant -1)

Orthogonal, by definition, means that it has orthonormal basis vectors. As a result, its transpose is its inverse. Reflections are also orthogonal. Determinant being 1 means that after transformation, size and shape doesn't change (circles remain circles of the same radius). Knowing how to recognize a rotation matrix is a useful thing that we'll make use of later.

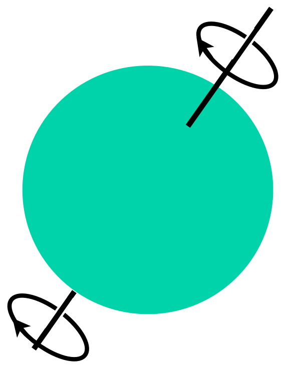
Leonhard Euler

- Master of rotation
- Gave us Euler angles
  - (we forgive him anyway)

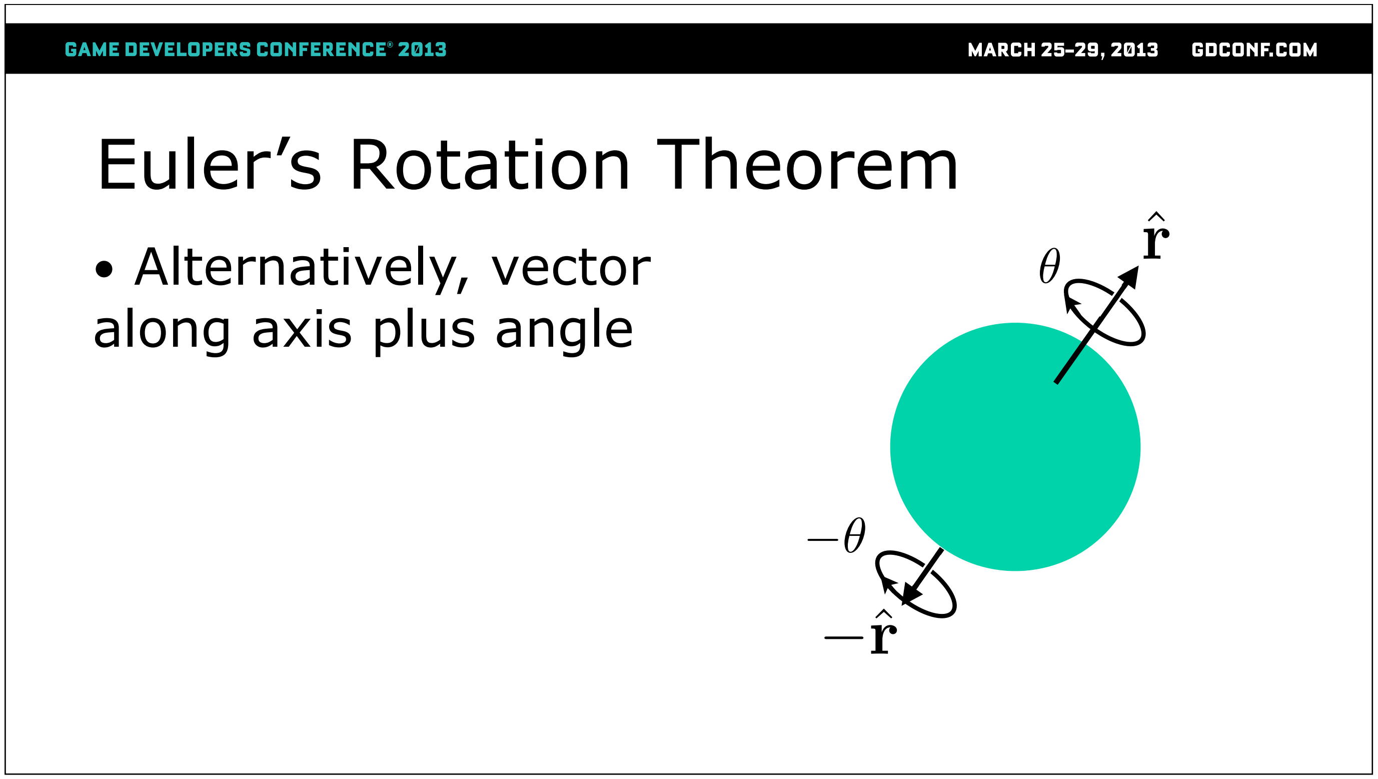


Euler’s Rotation Theorem

- Axis and angle represents any 3D rotation

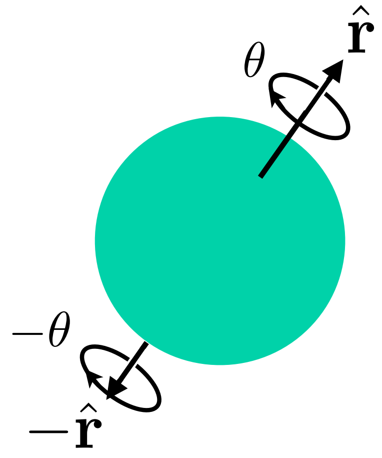


This (in a historically backwards way) leads us to Euler’s rotation theorem. 3D rotation can be represented by a line of points that don’t change (axis), and amount of rotation (angle).

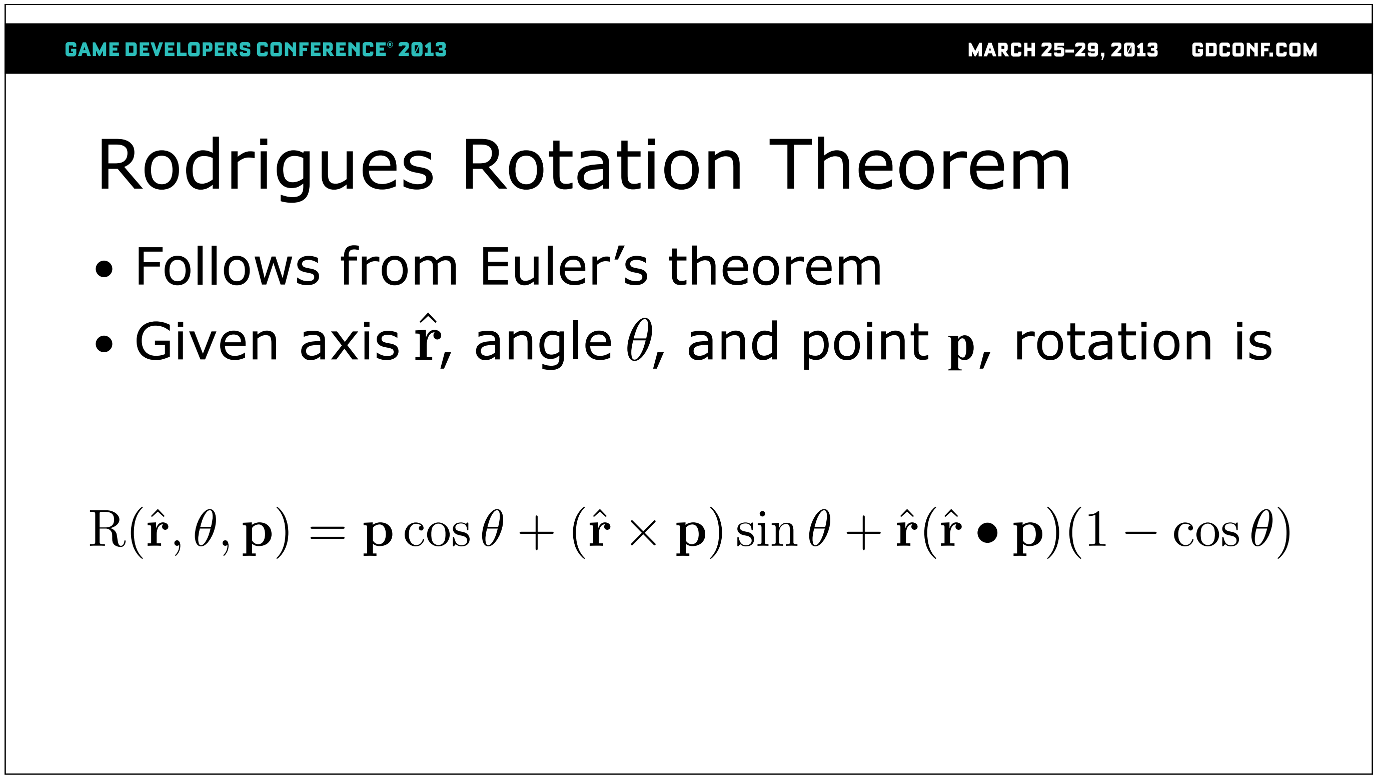


Euler’s Rotation Theorem

- Alternatively, vector along axis plus angle



In our modern notation we can represent this as a vector and an angle. For each rotation there are two possibilities, i.e. two antiparallel axes and angles (one axis and angle is negation of the other). For those who care, for a 3D rotation matrix, the eigenvector corresponding to the eigenvalue 1 is the axis of rotation. For those who don’t, don’t worry about it, we won’t mention it again.

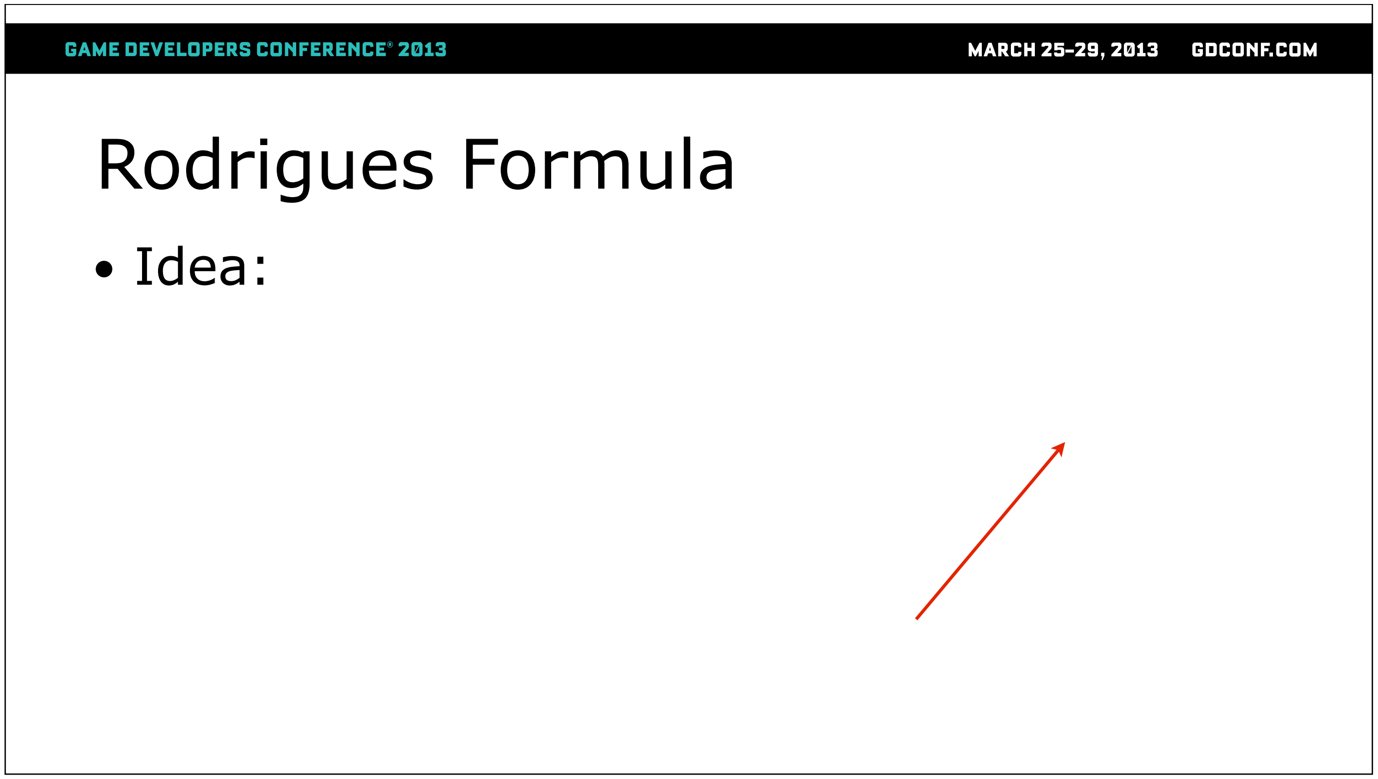


Rodrigues Rotation Theorem

- Follows from Euler's theorem
- Given axis $\hat{\mathbf{r}}$, angle θ, and point $\mathbf{p}$, rotation is

$$\mathrm{R}(\hat{\mathbf{r}}, \theta, \mathbf{p}) = \mathbf{p}\cos\theta + (\hat{\mathbf{r}} \times \mathbf{p})\sin\theta + \hat{\mathbf{r}}(\hat{\mathbf{r}} \bullet \mathbf{p})(1 - \cos\theta)$$

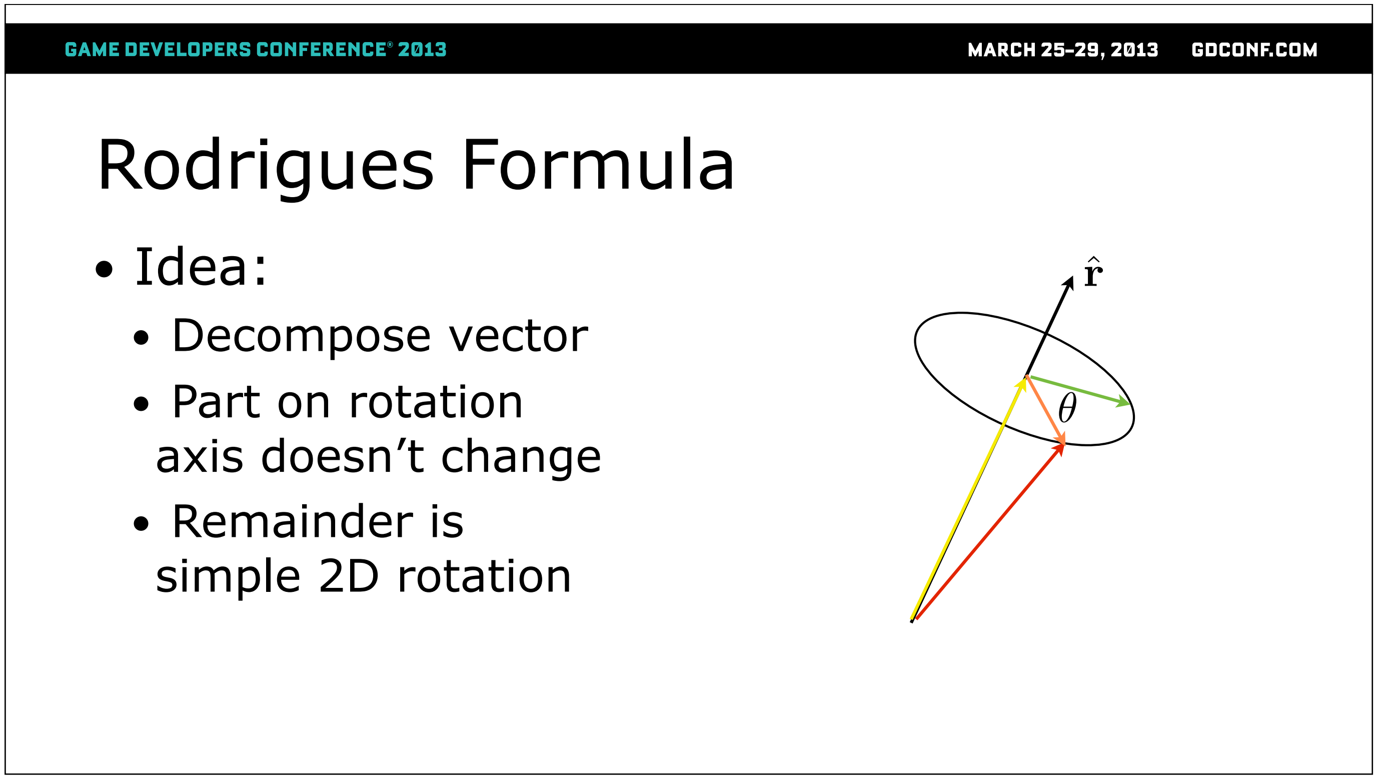
Benjamin Olinde Rodrigues (1795–1851), more commonly known as **Olinde Rodrigues**, was a French mathematician who is best known for his formula for Legendre polynomials. He is second best known (and best known in the graphics community) for this formula.



Rodrigues Formula

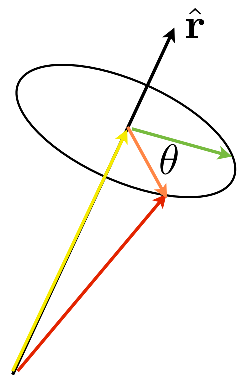
- Idea:

So suppose we have some vector in space we want to rotate.

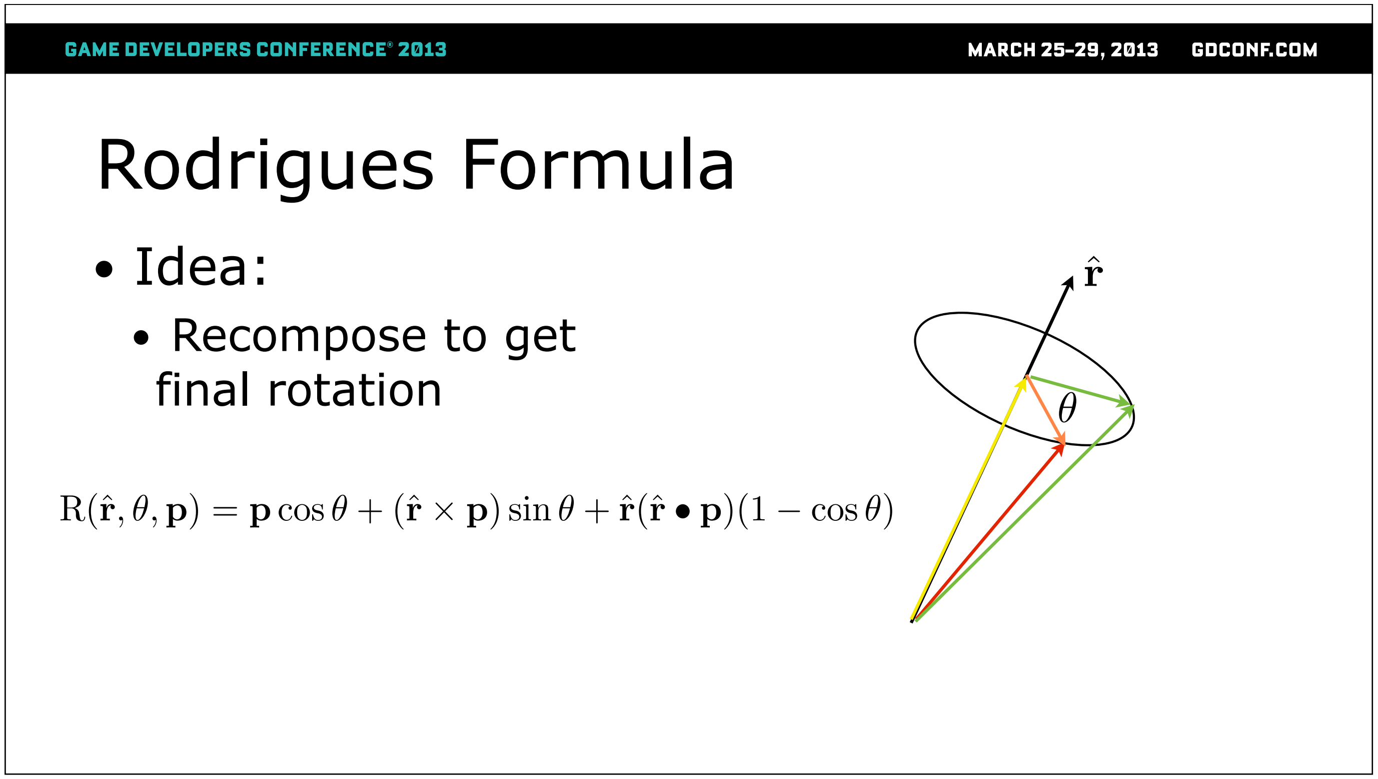


Rodrigues Formula

- Idea:
  - Decompose vector
  - Part on rotation axis doesn’t change
  - Remainder is simple 2D rotation



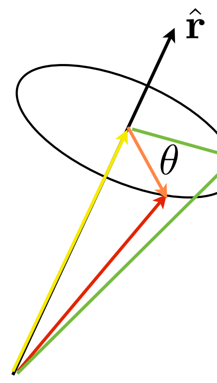
We begin by breaking it into two pieces. One is parallel to the rotation axis, the remainder is orthogonal to it. Then we can rotate the remainder in the plane orthogonal to the axis, using a 2D rotation.



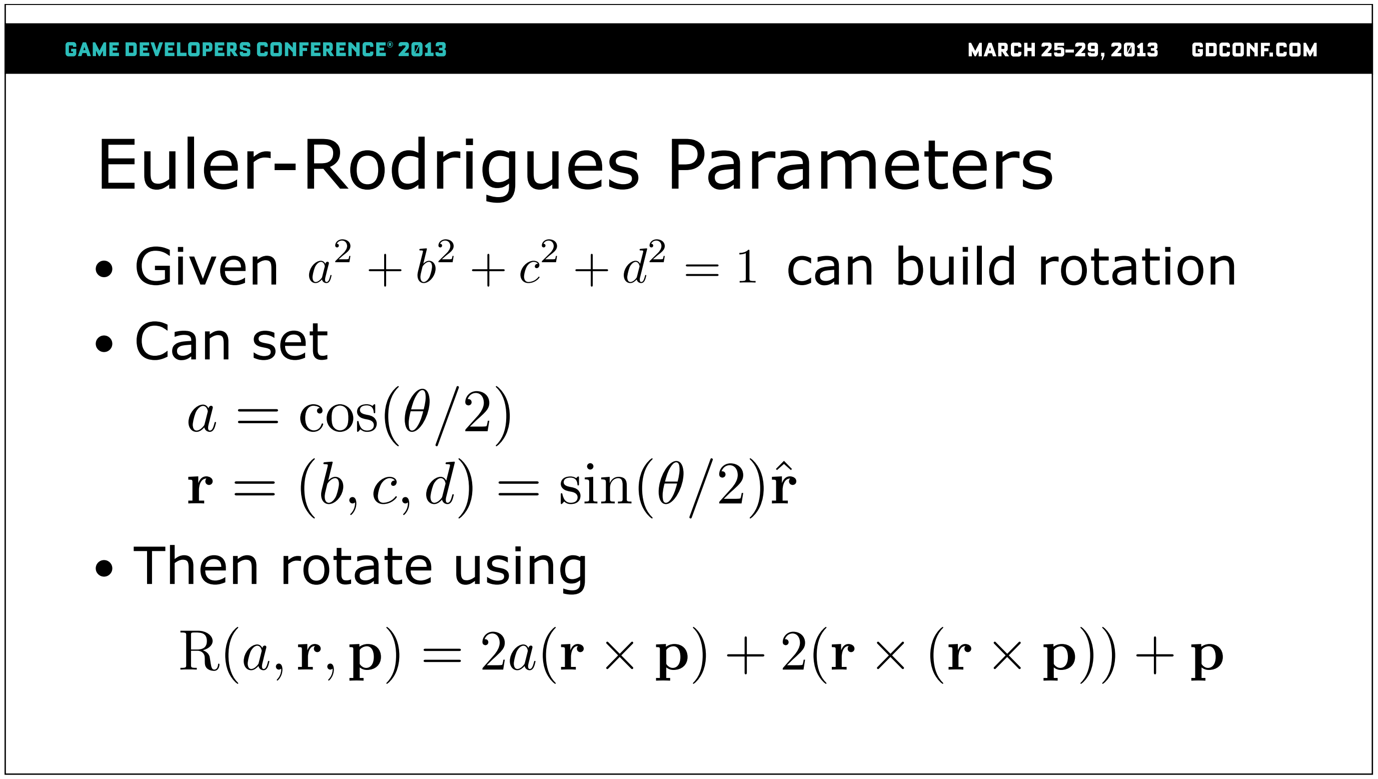
Rodrigues Formula

- Idea:
  - Recompose to get final rotation

$$R(\hat{\mathbf{r}}, \theta, \mathbf{p}) = \mathbf{p}\cos\theta + (\hat{\mathbf{r}} \times \mathbf{p})\sin\theta + \hat{\mathbf{r}}(\hat{\mathbf{r}} \bullet \mathbf{p})(1 - \cos\theta)$$



Finally we add the rotated remainder to the parallel part to get the final result. One thing to note about this, is that for both 2D and 3D rotation, we rotate through only one plane -- anything on that plane stays on that plane. And that plane is always orthogonal to the rotation axis. So rather than talking about an axis of rotation, we can talk about a plane of rotation.



Euler-Rodrigues Parameters

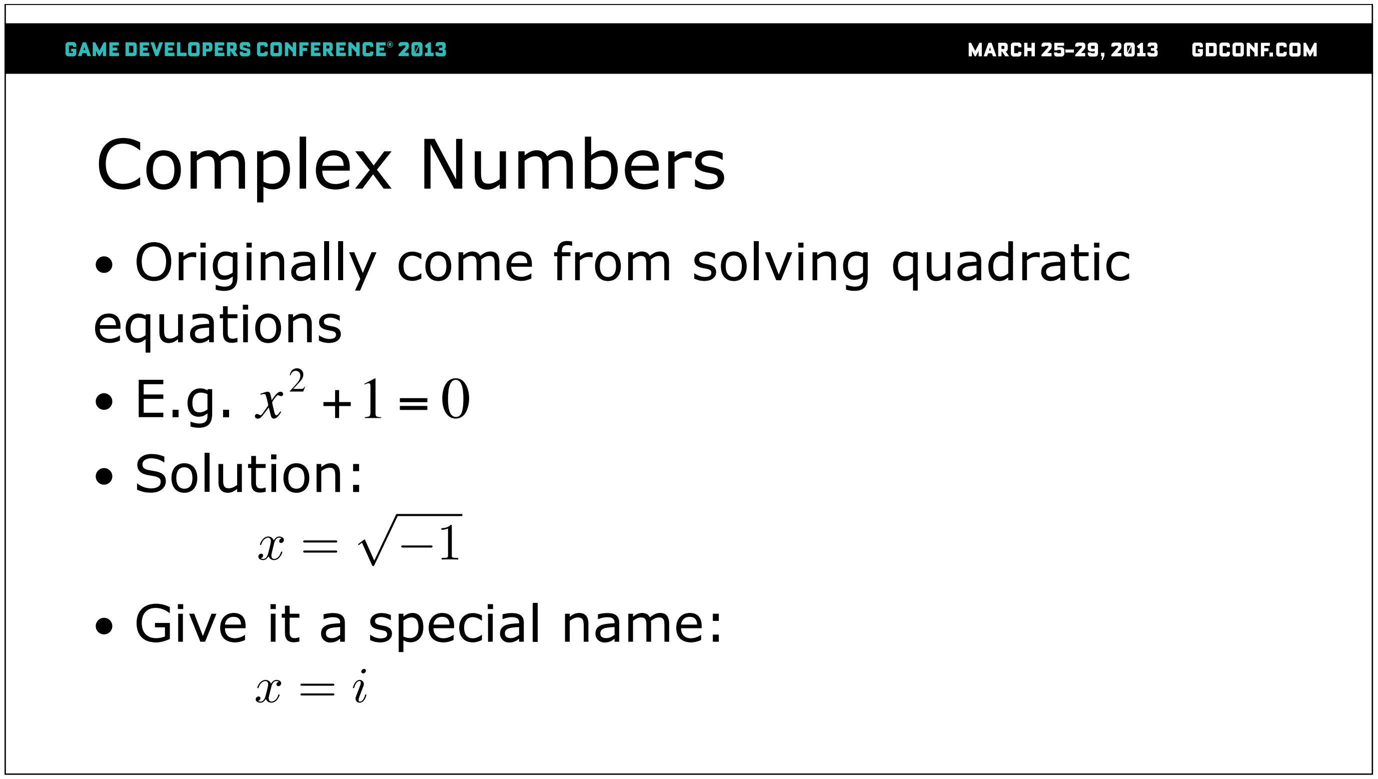
- Given $a^2 + b^2 + c^2 + d^2 = 1$ can build rotation
- Can set

$$a = \cos(\theta/2)$$
$$\mathbf{r} = (b, c, d) = \sin(\theta/2)\hat{\mathbf{r}}$$

- Then rotate using

$$\mathrm{R}(a, \mathbf{r}, \mathbf{p}) = 2a(\mathbf{r} \times \mathbf{p}) + 2(\mathbf{r} \times (\mathbf{r} \times \mathbf{p})) + \mathbf{p}$$

As a follow-on to this, Rodrigues derived this set of four parameters that can be used to create a 3D rotation. A, b, c, and d again boil down to an axis and angle format. But this does give us an alternative way of writing the Rodrigues formula. Why Euler-Rodrigues? Well, these terms were actually discovered by Euler in 1771, and again by Rodrigues in 1840. Of course, they didn't have vector notation, but it boils down to the same thing. And for those of you who know something of quaternions, this may seem somewhat familiar...



Complex Numbers

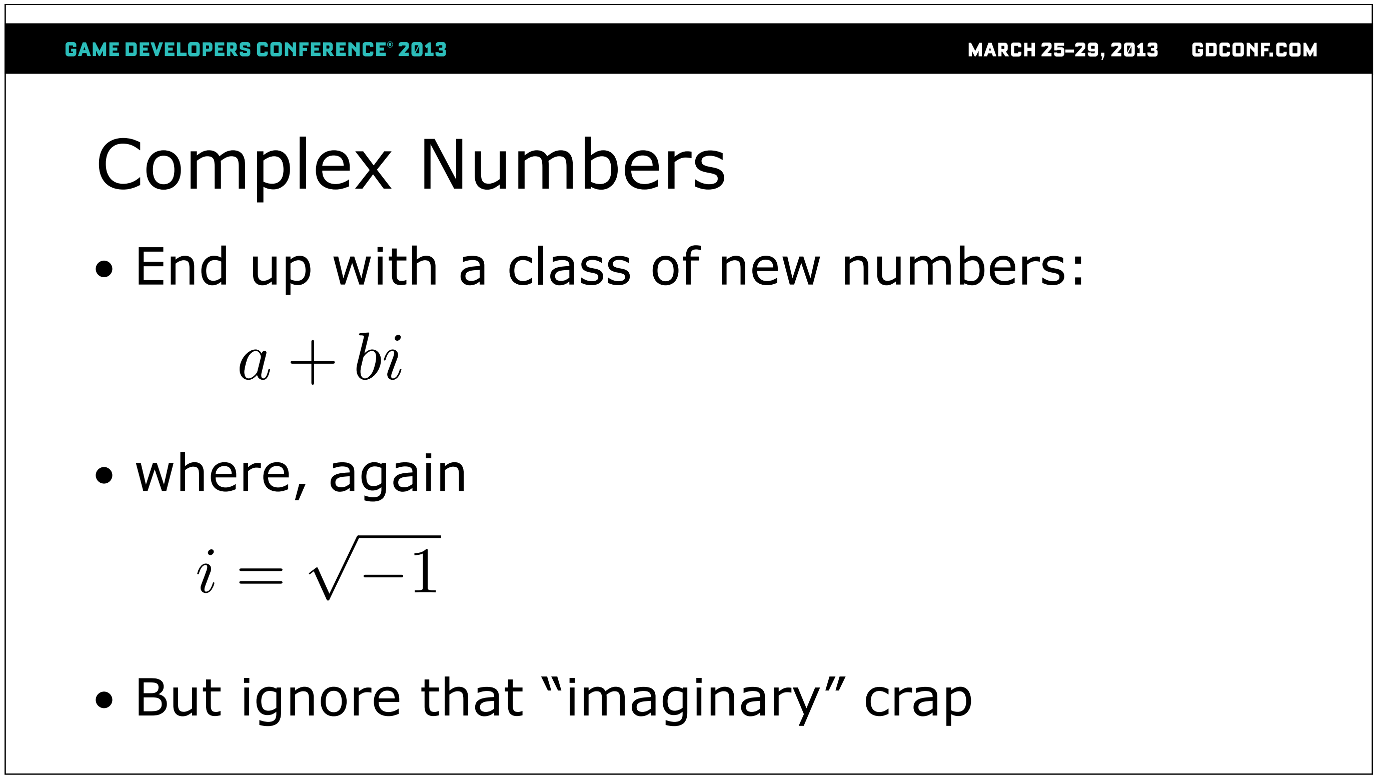
- Originally come from solving quadratic equations
- E.g. $x^2 + 1 = 0$
- Solution:

$$x = \sqrt{-1}$$

- Give it a special name:

$$x = i$$

So that's it for background regarding rotations. Let's move on to complex numbers.



Complex Numbers

- End up with a class of new numbers:

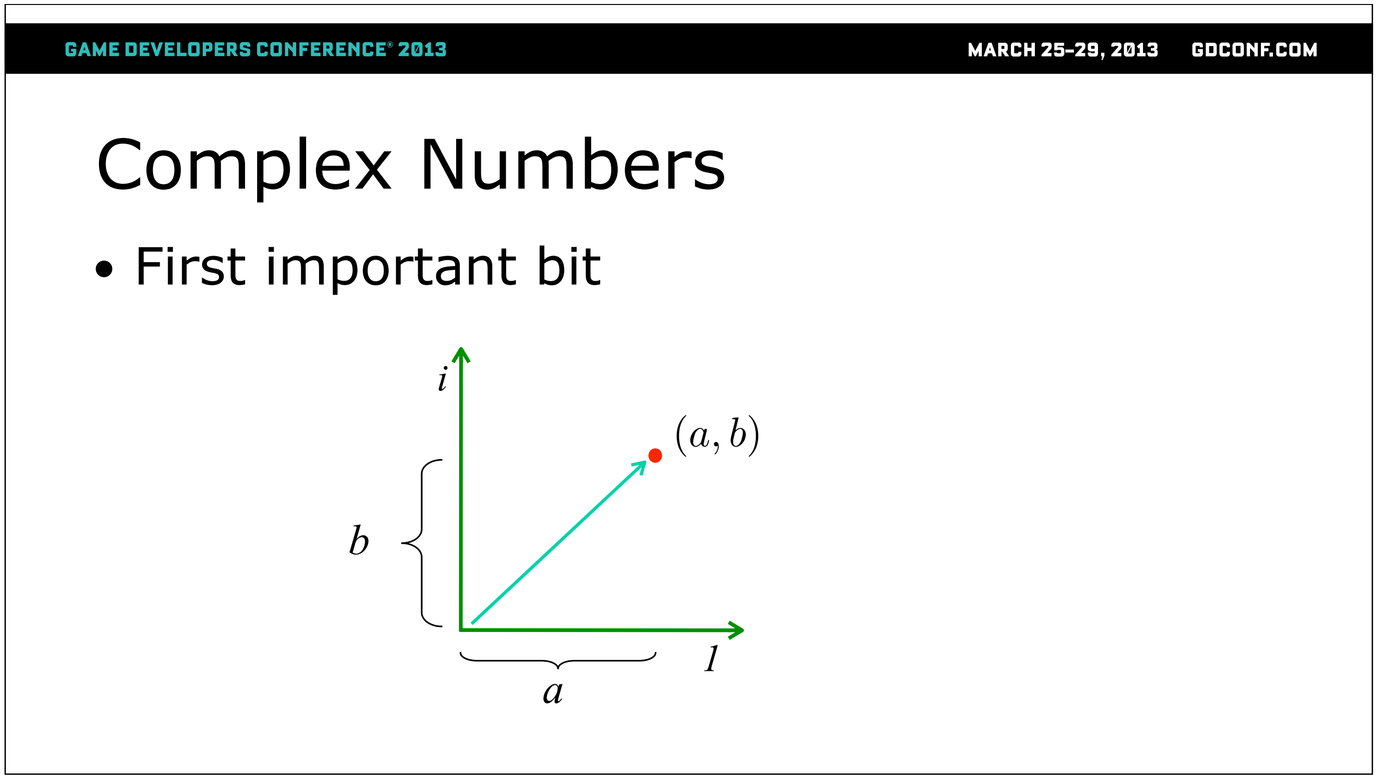
$$a + bi$$

- where, again

$$i = \sqrt{-1}$$

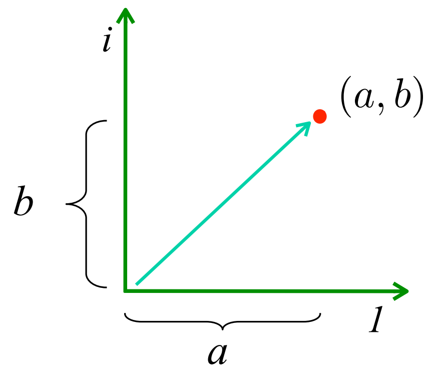
- But ignore that "imaginary" crap

I got hung up on the name imaginary for a long time. It's far better to think of them as a new class of numbers, with slightly different rules.



Complex Numbers

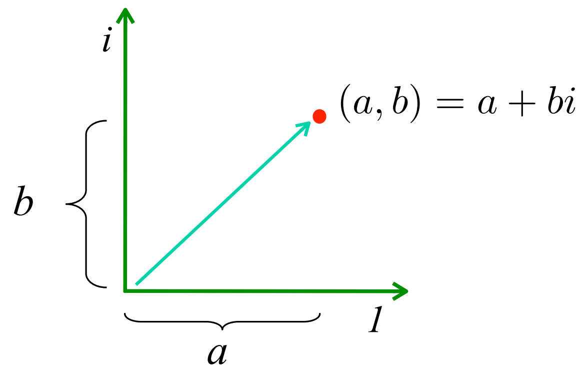
- First important bit



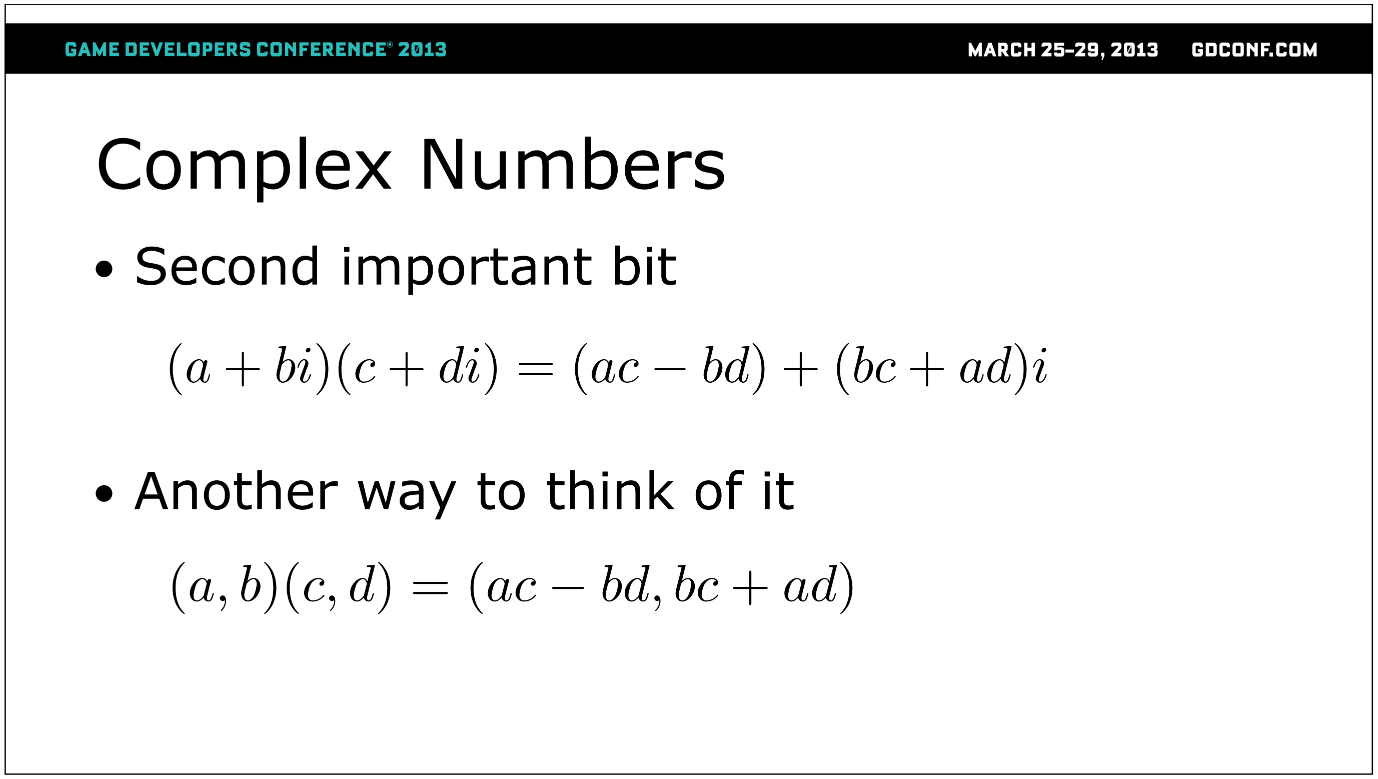
Can represent complex number as a point/vector on a 2D plane. Think of the real and imaginary axes as being the 1 and the i axes.

Complex Numbers

- First important bit



So this point/vector represents a complex number.



Complex Numbers

- Second important bit

$$(a+bi)(c+di) = (ac-bd) + (bc+ad)i$$

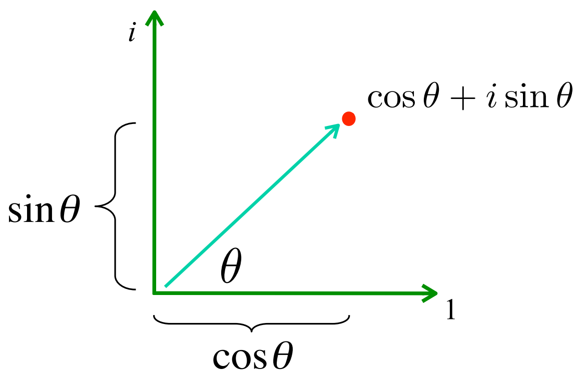
- Another way to think of it

$$(a,b)(c,d) = (ac-bd, bc+ad)$$

So a complex number is just a 2D vector with a special multiplication operation.

Complex Numbers

- Now suppose: restrict to unit length



Complex Numbers

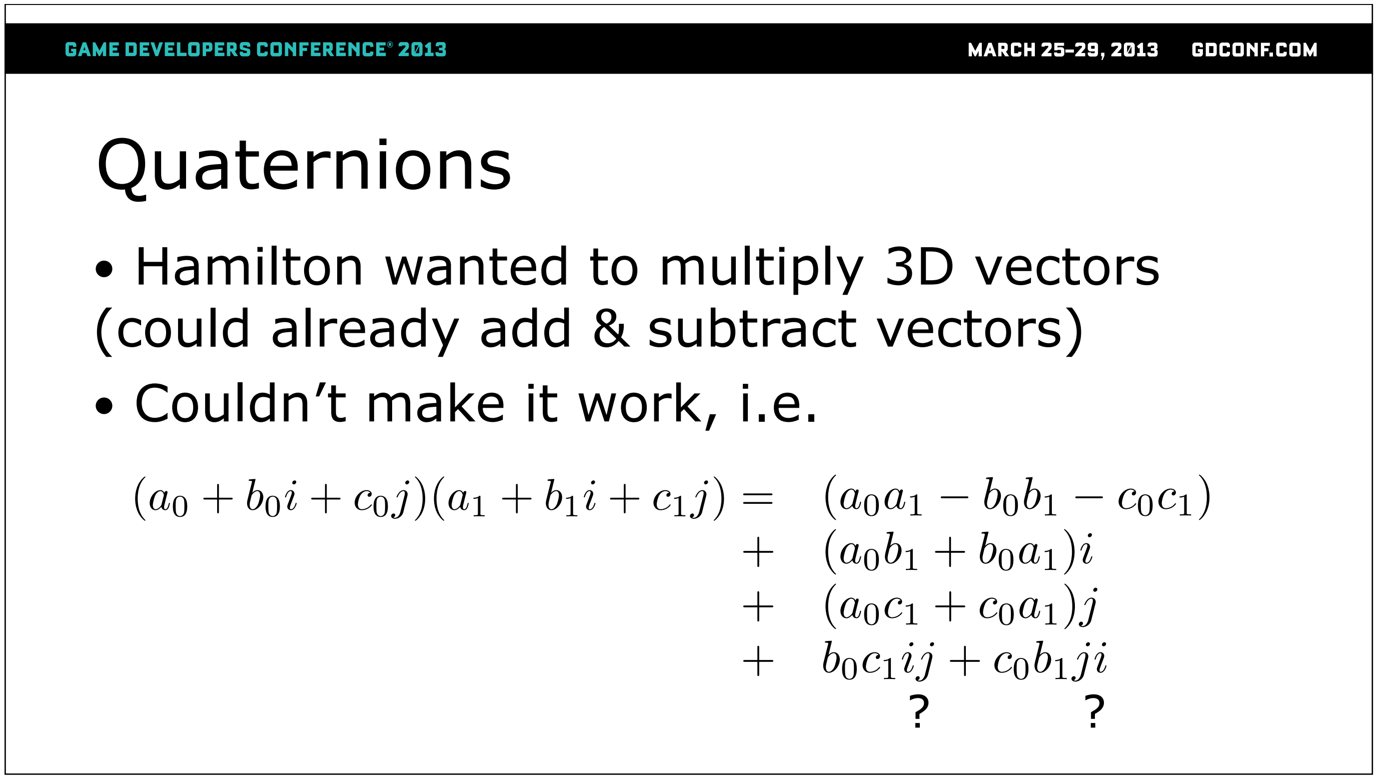
- Multiply general complex number by unit one

$$(x + yi)(\cos\theta + i\sin\theta) = \\ (x\cos\theta - y\sin\theta) + i(x\sin\theta + y\cos\theta)$$

- Look familiar?
- Gives us 2D rotation!

Complex Numbers

- Good for rotation! Who knew!
- (He did)



Quaternions

- Hamilton wanted to multiply 3D vectors (could already add & subtract vectors)
- Couldn't make it work, i.e.

$$\begin{aligned}(a_0 + b_0 i + c_0 j)(a_1 + b_1 i + c_1 j) = & \quad (a_0 a_1 - b_0 b_1 - c_0 c_1) \\ + & \quad (a_0 b_1 + b_0 a_1) i \\ + & \quad (a_0 c_1 + c_0 a_1) j \\ + & \quad b_0 c_1 \underset{?}{ij} + c_0 b_1 \underset{?}{ji}\end{aligned}$$

There's a story that Hamilton wrote in his memoirs, about him coming down to breakfast and his son asking him, "Papa, can you multiply triplets?" and he would sadly respond, "No, I can only add and subtract them." The problem is those ij and ji terms (and as it turns out, the order matters). Also want multiplication that avoids zero divisors, i.e. we don't want non-zero numbers that multiply together to produce zero (every non-zero number needs an inverse -- this is known as a division algebra). Nothing he came up with three terms would work. (Cross product doesn't work -- cross product of parallel vectors is zero.)

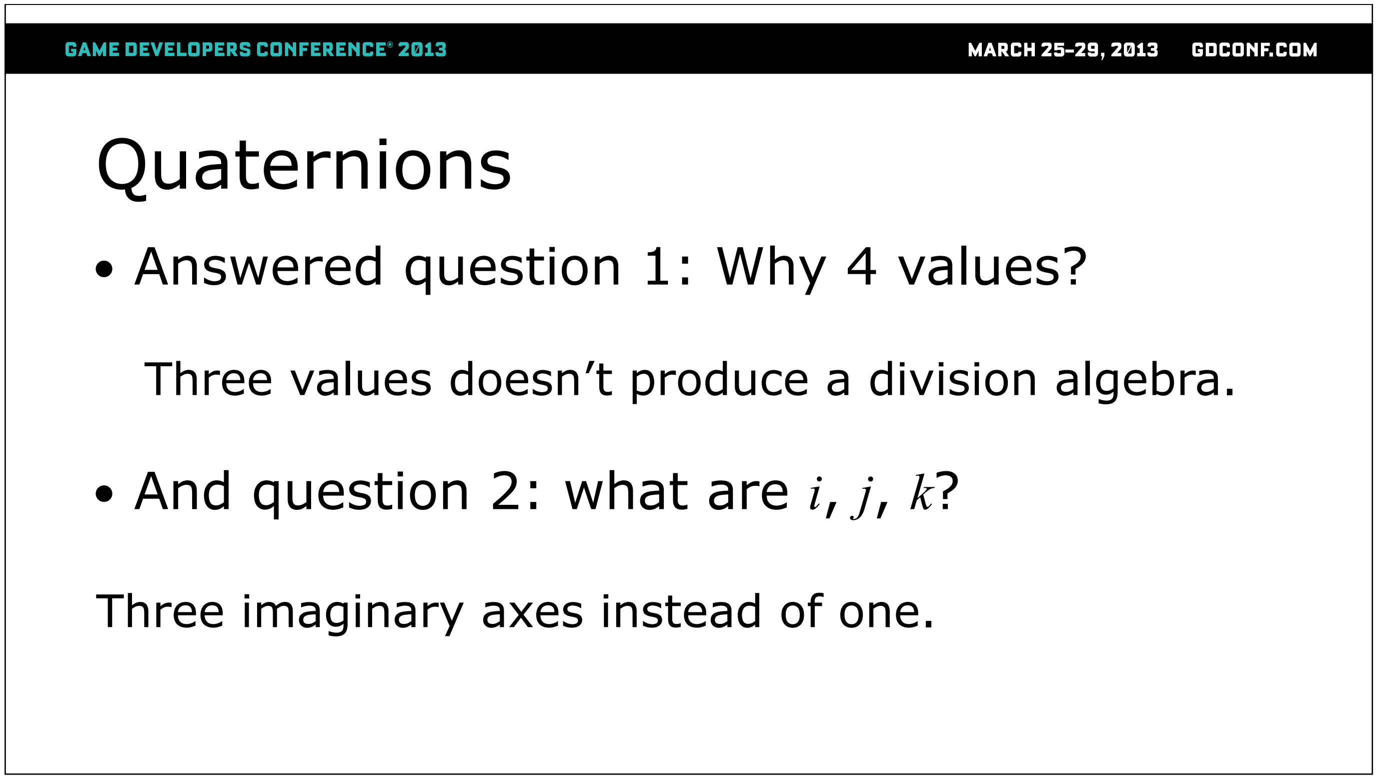
Quaternions

- Insight: multiplication possible with 3 imaginary values

$$i^2 = j^2 = k^2 = ijk = -1$$

$$\begin{array}{lllllllll} ij & = & k & jk & = & i & ki & = & j \\ ji & = & -k & kj & = & -i & ik & = & -j \end{array}$$

In 1843, as Hamilton was walking along the Royal Canal under the Brougham Bridge in Dublin, he had his insight -- with 3 imaginary values he could get a complete system of numbers where multiplication provided proper inverses. He was so excited he carved the first set of equations onto the bridge. Notice that now multiplication is no longer commutative -- the imaginary values are anticommutative. If you're familiar with cross products this may seem very familiar -- in fact the cross product comes from quaternion math.



Quaternions

- Answered question 1: Why 4 values?

  Three values doesn't produce a division algebra.

- And question 2: what are i, j, k?

Three imaginary axes instead of one.

Again, a division algebra means that all numbers other than zero have a multiplicative inverse. Why is a division algebra important? Well, if we want to do rotations, we want to be able to undo them, i.e. take inverses. So having an algebra that handles inverses is necessary.

Quaternion

- Complex numbers extended

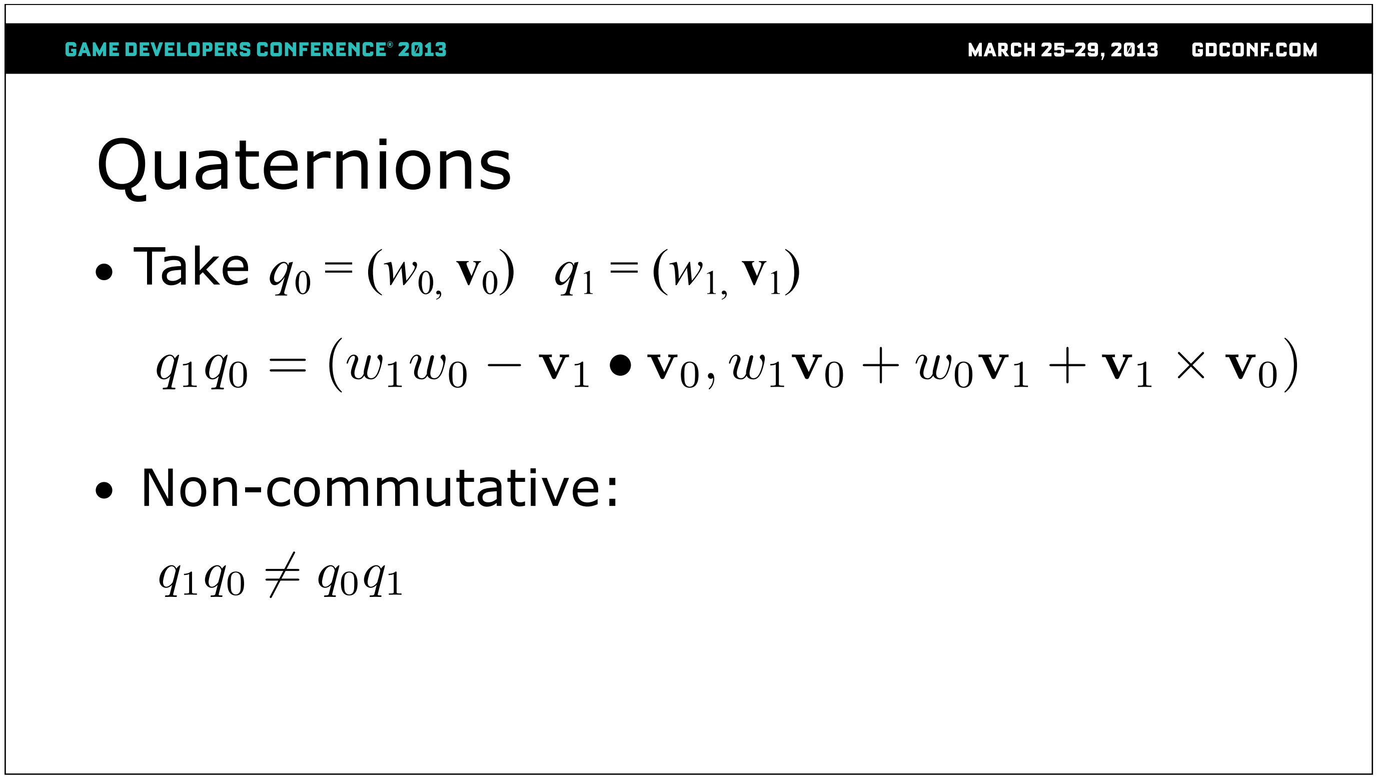
$$a + bi \Rightarrow w + xi + yj + zk$$

- Can represent as coordinates

$$(w, x, y, z)$$

- Or scalar/vector pair

$$(w, \mathbf{v})$$



Quaternions

- Take $q_0 = (w_{0,}\ \mathbf{v}_0)$ $\quad q_1 = (w_{1,}\ \mathbf{v}_1)$

$$q_1 q_0 = (w_1 w_0 - \mathbf{v}_1 \bullet \mathbf{v}_0, w_1 \mathbf{v}_0 + w_0 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_0)$$

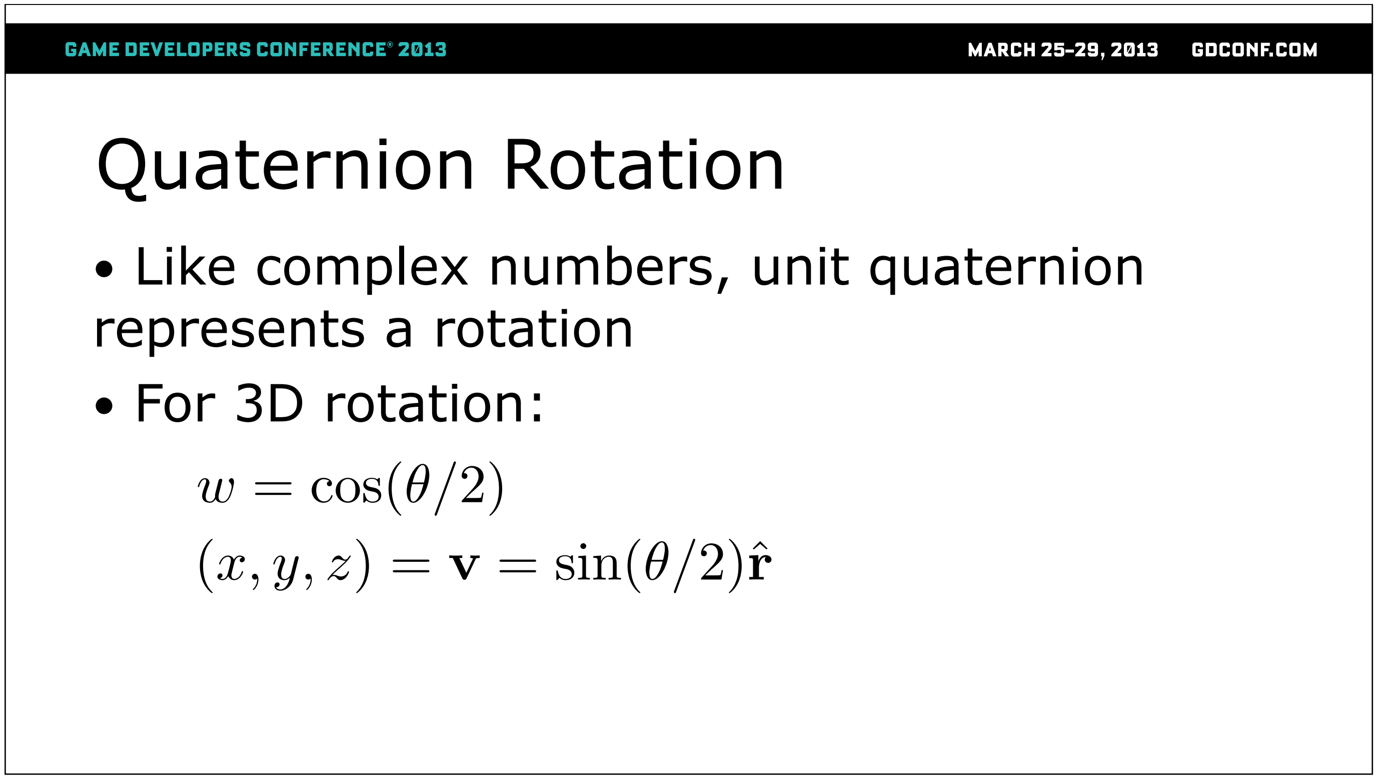
- Non-commutative:

$$q_1 q_0 \neq q_0 q_1$$

Using our familiar vector operations we can multiply two quaternions together as follows. Notice again, that due to the cross product, that this is not commutative.

Quaternions

- Multiplicative identity is $(1, 0, 0, 0)$
- q^{-1} is inverse
  - i.e. $q \cdot q^{-1}$ or $q^{-1} \cdot q$ = $(1, 0, 0, 0)$

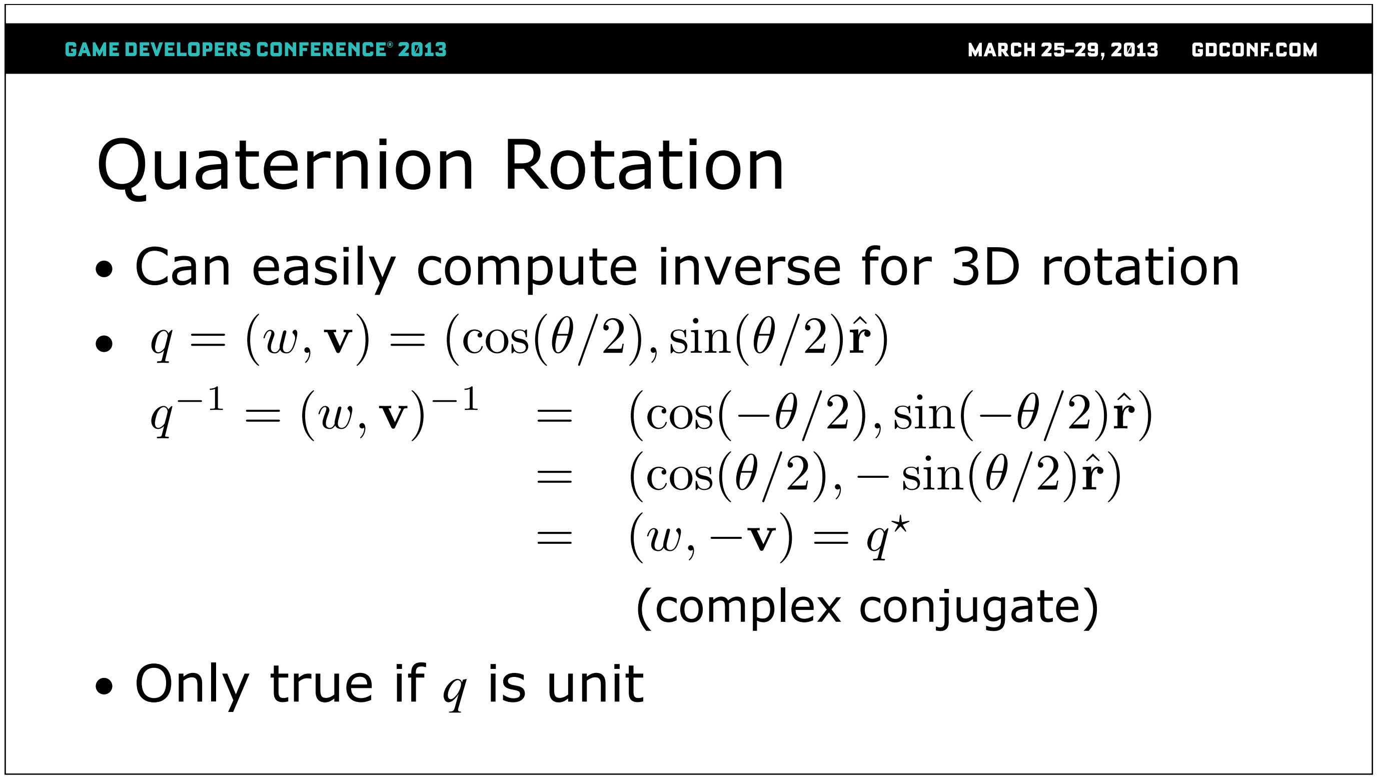


Quaternion Rotation

- Like complex numbers, unit quaternion represents a rotation
- For 3D rotation:

$$w = \cos(\theta/2)$$
$$(x, y, z) = \mathbf{v} = \sin(\theta/2)\hat{\mathbf{r}}$$

This may seem somewhat familar... in any case, now we’ll show how to use this quaternion to rotate vectors.



Quaternion Rotation

- Can easily compute inverse for 3D rotation
- $q = (w, \mathbf{v}) = (\cos(\theta/2), \sin(\theta/2)\hat{\mathbf{r}})$

$$\begin{aligned} q^{-1} = (w, \mathbf{v})^{-1} &= (\cos(-\theta/2), \sin(-\theta/2)\hat{\mathbf{r}}) \\ &= (\cos(\theta/2), -\sin(\theta/2)\hat{\mathbf{r}}) \\ &= (w, -\mathbf{v}) = q^{\star} \end{aligned}$$

(complex conjugate)

- Only true if q is unit

Using this representation we can determine what the inverse would be. We just negate the angle, move some signs around and we discover that the conjugate is the same as inverse. But as it says, this is only true for unit quaternions.

Quaternion Rotation

- Have vector $\mathbf{p}$, unit quaternion q
- Treat $\mathbf{p}$ as quaternion $p = (0, \mathbf{p})$
- 3D rotation of p by q is

$$p' = qpq^{-1}$$

Quaternion Rotation

- Have vector $\mathbf{p}$, unit quaternion q
- Treat $\mathbf{p}$ as quaternion $p = (0, \mathbf{p})$
- 3D rotation of p by q is

$$p' = qpq^{-1}$$

- Boils down to

$$\mathbf{p}' = \mathbf{p} + 2w(\mathbf{v} \times \mathbf{p}) + 2(\mathbf{v} \times (\mathbf{v} \times \mathbf{p}))$$

Hang on, this seems somewhat familiar...

Quaternion Rotation

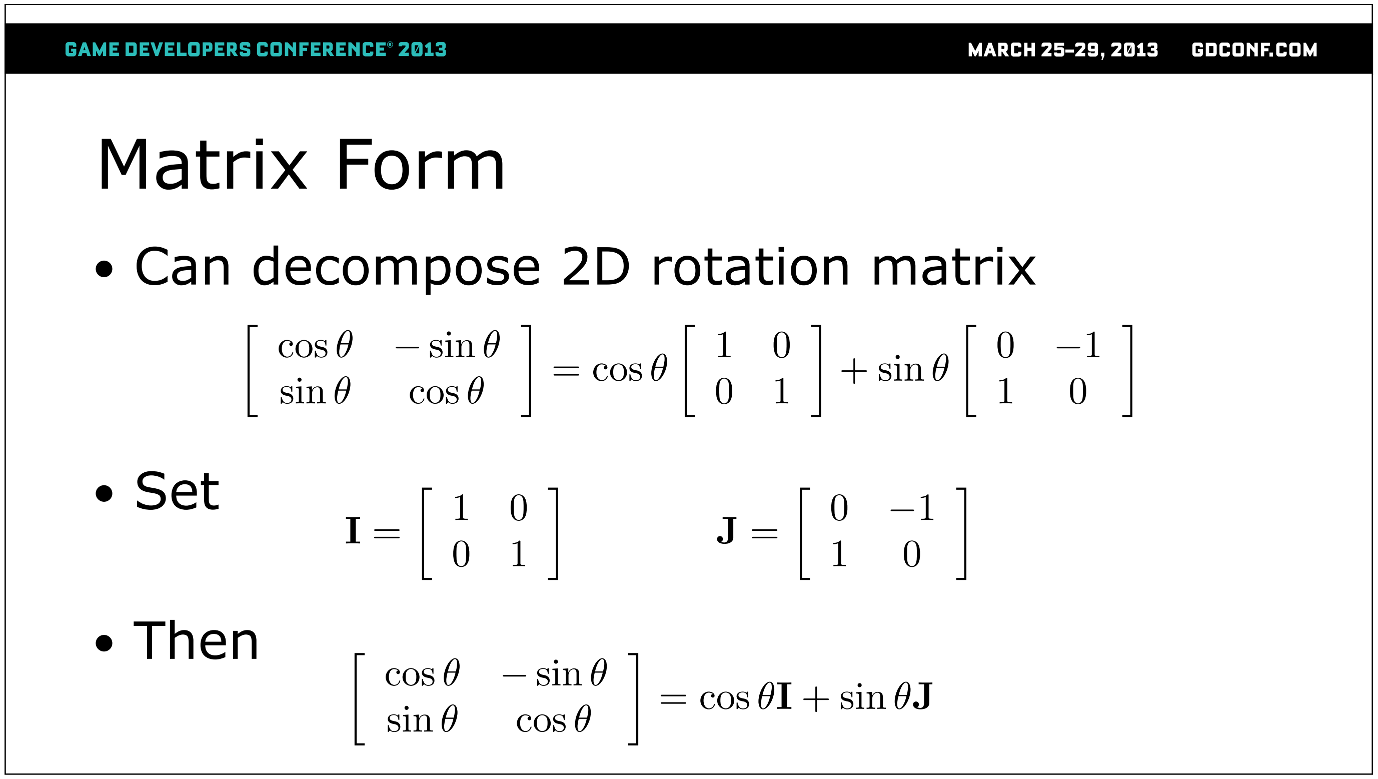
Quaternion	Euler-Rodrigues
$w = \cos(\theta/2)$	$a = \cos(\theta/2)$
$(x, y, z) = \mathbf{v} = \sin(\theta/2)\hat{\mathbf{r}}$	$(b, c, d) = \mathbf{r} = \sin(\theta/2)\hat{\mathbf{r}}$
$\mathbf{p} + 2w(\mathbf{v} \times \mathbf{p}) + 2(\mathbf{v} \times (\mathbf{v} \times \mathbf{p}))$	$\mathbf{p} + 2a(\mathbf{r} \times \mathbf{p}) + 2(\mathbf{r} \times (\mathbf{r} \times \mathbf{p}))$

Quaternions rotate in 3D!

Quaternions

- Proved, but not appealing
- Still: why half-angles? What does qpq^{-1} do?

- Let's try another perspective



Matrix Form

- Can decompose 2D rotation matrix

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \cos\theta \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \sin\theta \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

- Set

$$\mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{J} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

- Then

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \cos\theta\mathbf{I} + \sin\theta\mathbf{J}$$

This discussion is from a section written by Ken Shoemake in David Eberly's book Game Physics. J in this case is a rotation counterclockwise by 90 degrees. So we've replaced two scalars by two matrices that do exactly the same thing.

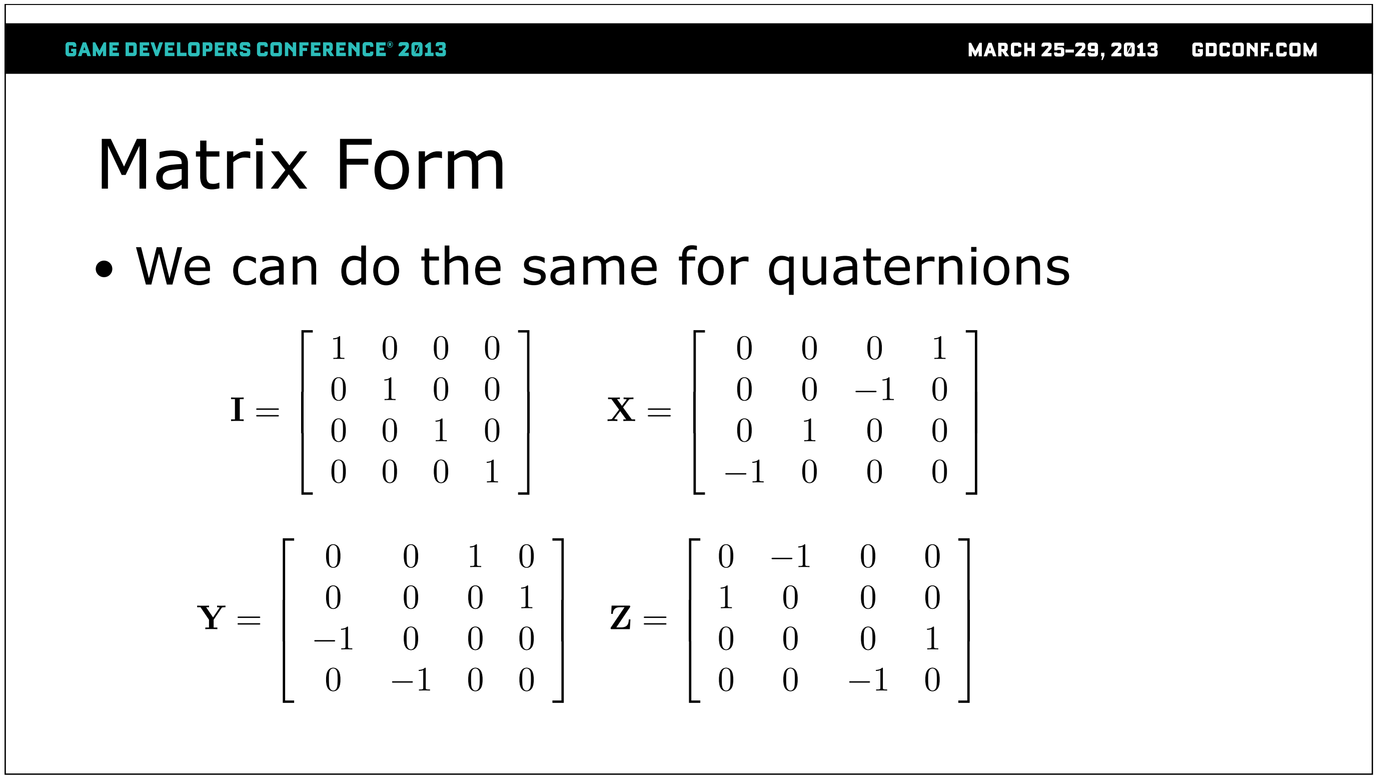
Matrix Form

- $\mathbf{I}$ and $\mathbf{J}$ act just like 1 and i

$$\mathbf{J}^2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = -\mathbf{I}$$

$$(a\mathbf{I} + b\mathbf{J})(c\mathbf{I} + d\mathbf{J}) = (ac - bd)\mathbf{I} + (bc + ad)\mathbf{J}$$

- Complex numbers in another form!



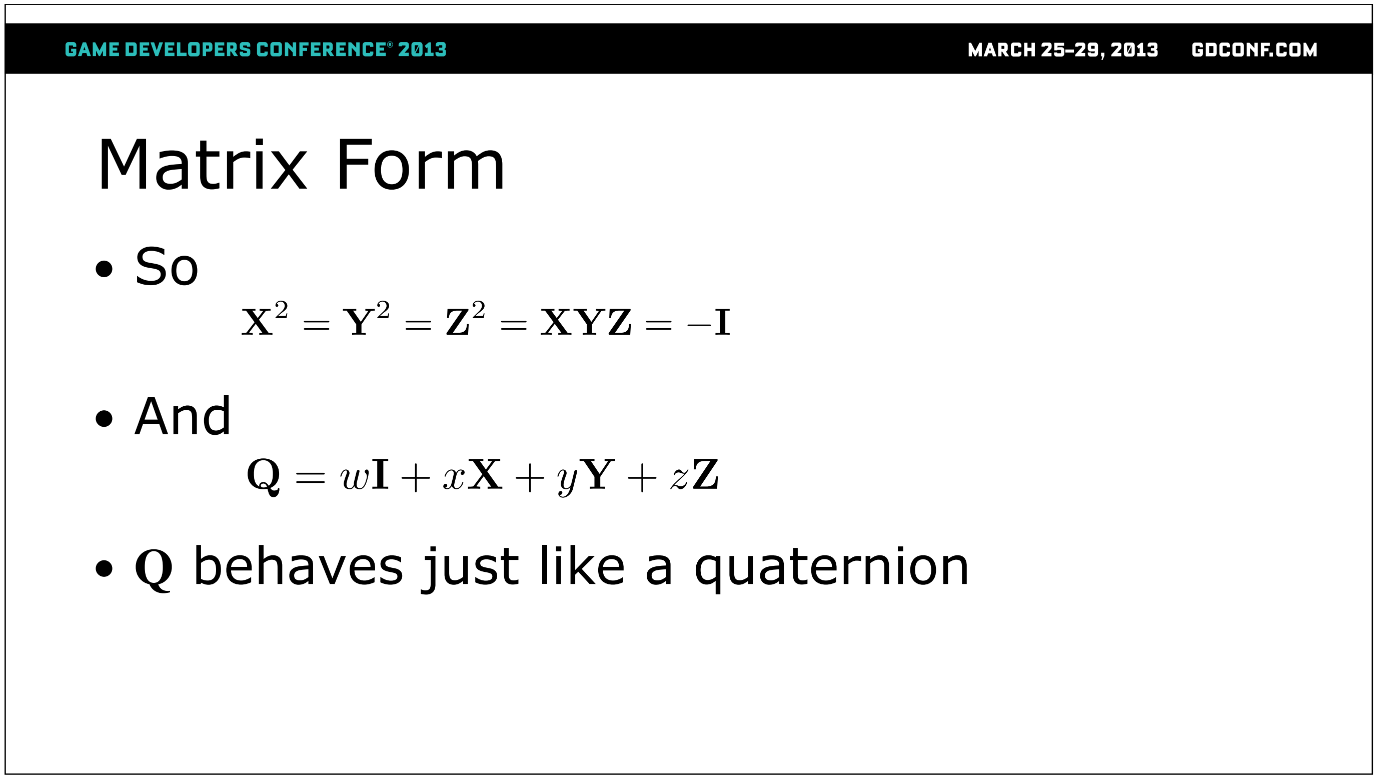
Matrix Form

- We can do the same for quaternions

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \quad \mathbf{Z} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

This is Shoemake’s notation, I have no idea if it’s standard.



Matrix Form

- So

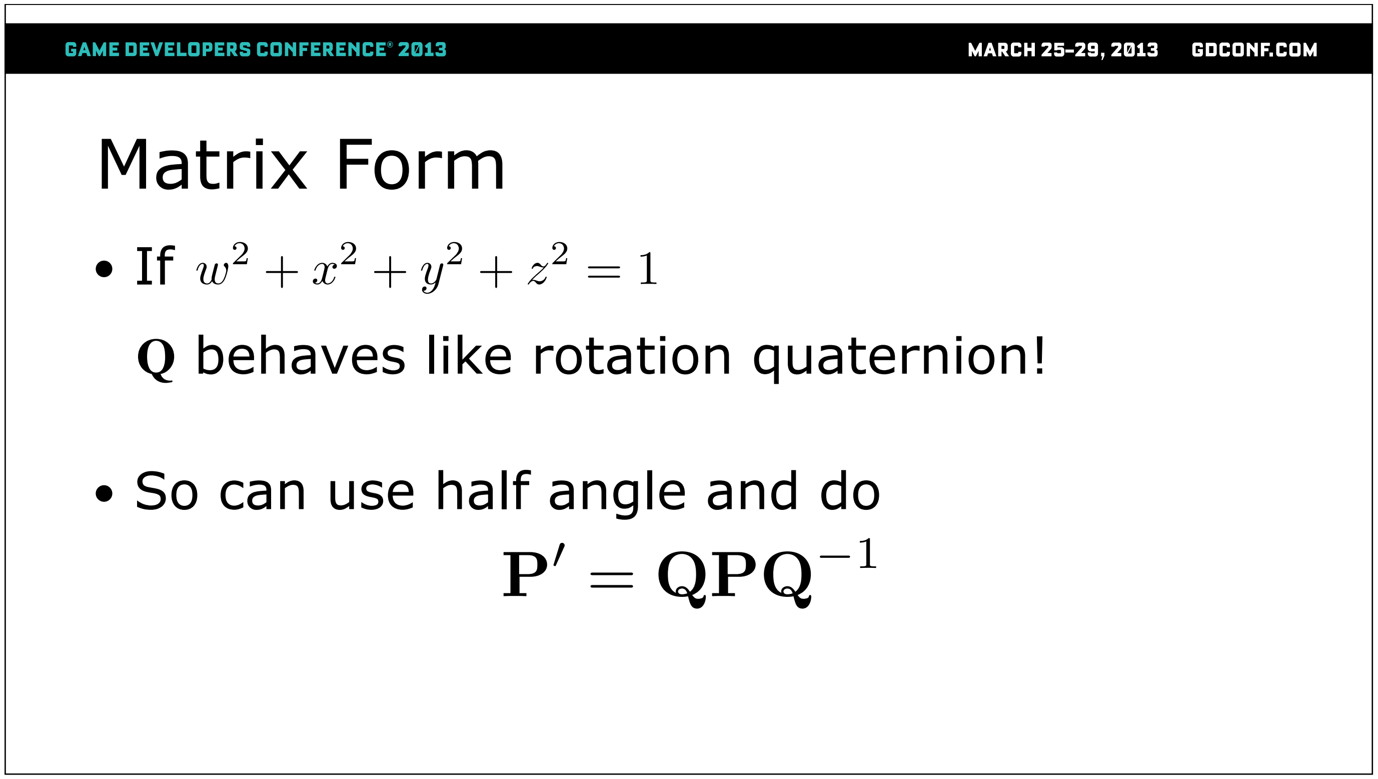
$$\mathbf{X}^2 = \mathbf{Y}^2 = \mathbf{Z}^2 = \mathbf{XYZ} = -\mathbf{I}$$

- And

$$\mathbf{Q} = w\mathbf{I} + x\mathbf{X} + y\mathbf{Y} + z\mathbf{Z}$$

- $\mathbf{Q}$ behaves just like a quaternion

In fact, it is a quaternion, just in another form.



Matrix Form

- If $w^2 + x^2 + y^2 + z^2 = 1$

  $\mathbf{Q}$ behaves like rotation quaternion!

- So can use half angle and do

$$\mathbf{P}' = \mathbf{Q}\mathbf{P}\mathbf{Q}^{-1}$$

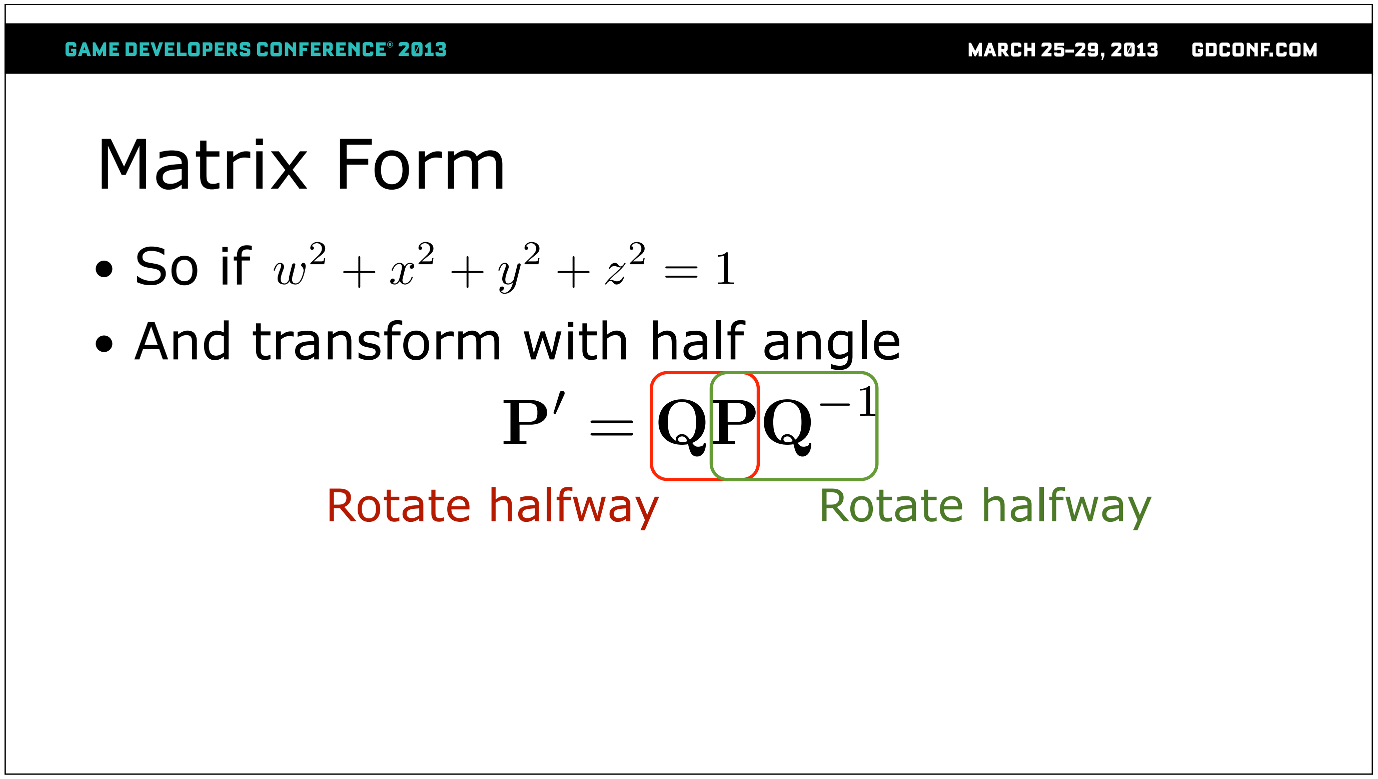
Note we need to use matrix form for P (can’t multiply a column vector on the right)

Matrix Form

- Examine Q more closely:

$$\mathbf{Q} = \begin{bmatrix} w & -z & y & x \\ z & w & -x & y \\ -y & x & w & z \\ -x & -y & -z & w \end{bmatrix}$$

- Unit basis vectors, dot products 0
- Determinant 1
- Rotation matrix!



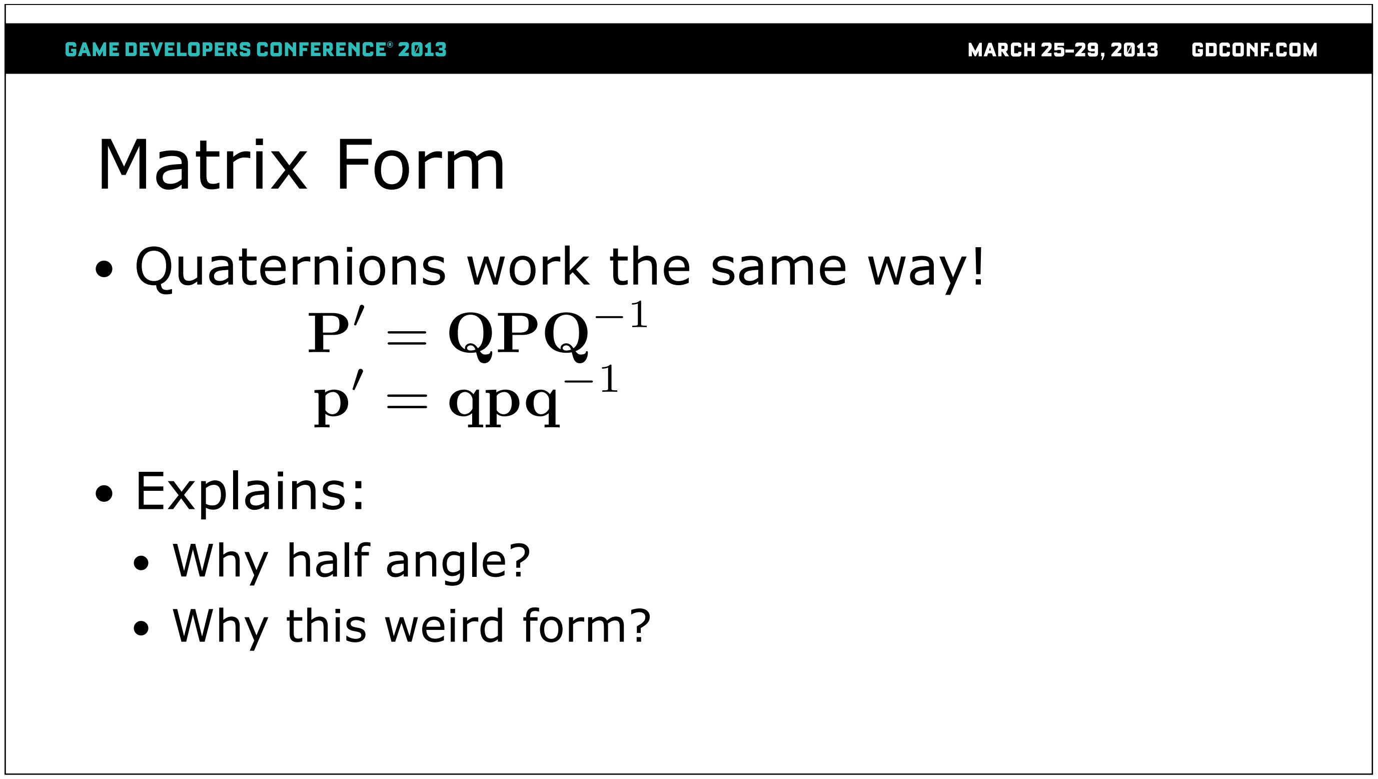
Matrix Form

- So if $w^2 + x^2 + y^2 + z^2 = 1$
- And transform with half angle

$$\mathbf{P}' = \mathbf{Q}\mathbf{P}\mathbf{Q}^{-1}$$

Rotate halfway Rotate halfway

So knowing this, we might naively break it down as: multiply on the left to rotate halfway, then rotating on the right by the inverse rotates the remainder. This is not entirely correct, as we'll see, but it's a starting point.



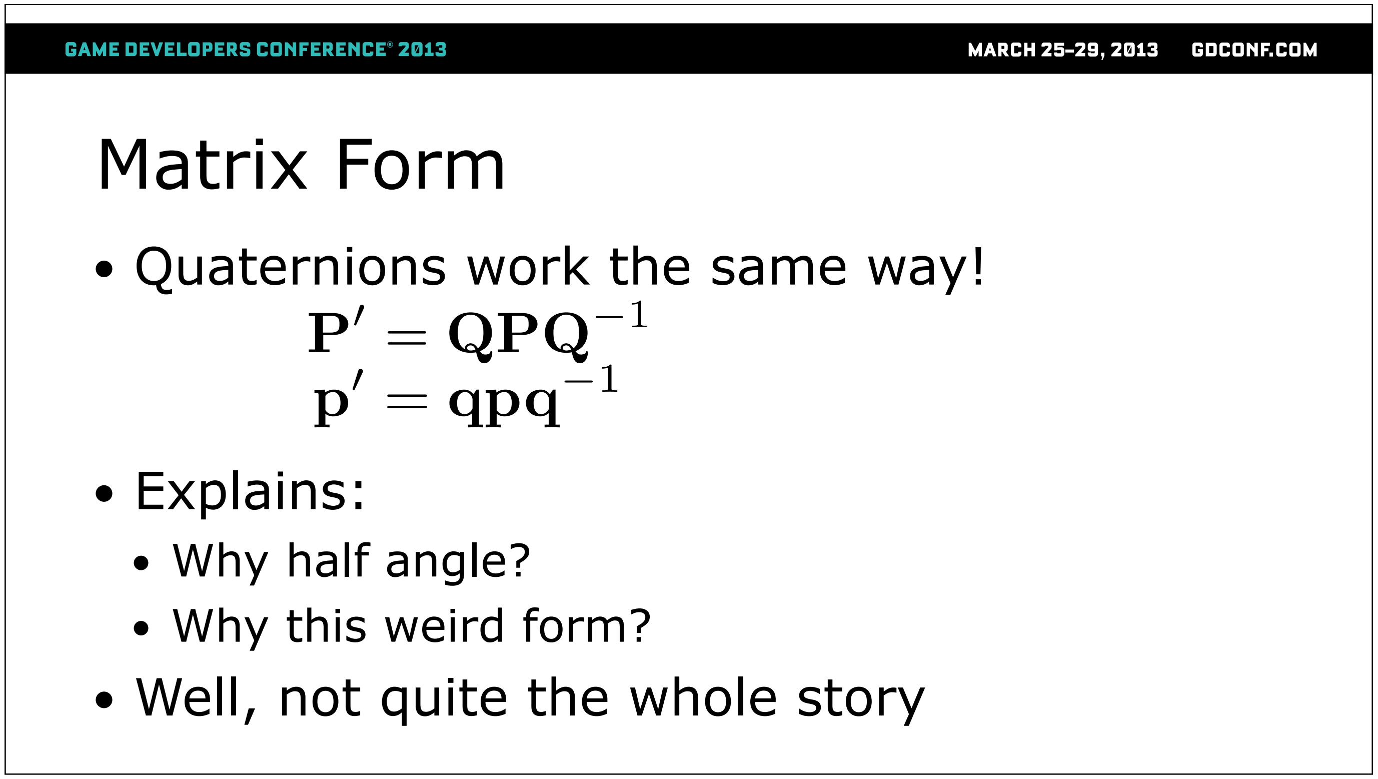
Matrix Form

- Quaternions work the same way!

$$\mathbf{P}' = \mathbf{Q}\mathbf{P}\mathbf{Q}^{-1}$$
$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}$$

- Explains:
  - Why half angle?
  - Why this weird form?

So we might conclude we have the entire picture.



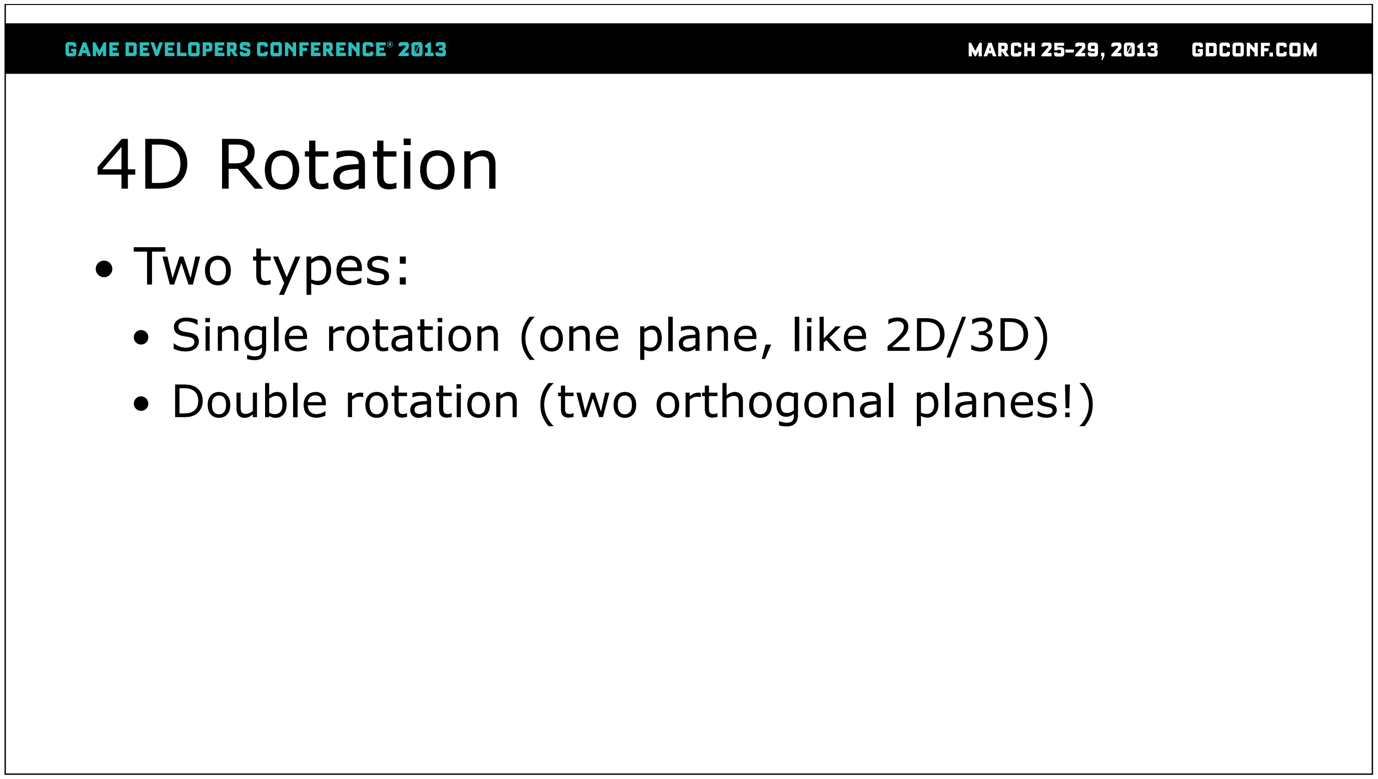
Matrix Form

- Quaternions work the same way!

$$\mathbf{P}' = \mathbf{Q}\mathbf{P}\mathbf{Q}^{-1}$$
$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}$$

- Explains:
  - Why half angle?
  - Why this weird form?
- Well, not quite the whole story

But we don't. In truth this skips over a number of details -- there's a little more to it than that. (In particular, multiplying on the right by the inverse of the matrix doesn't rotate in the same way as multiplying on the left by the matrix, whether it's 3D or 4D)

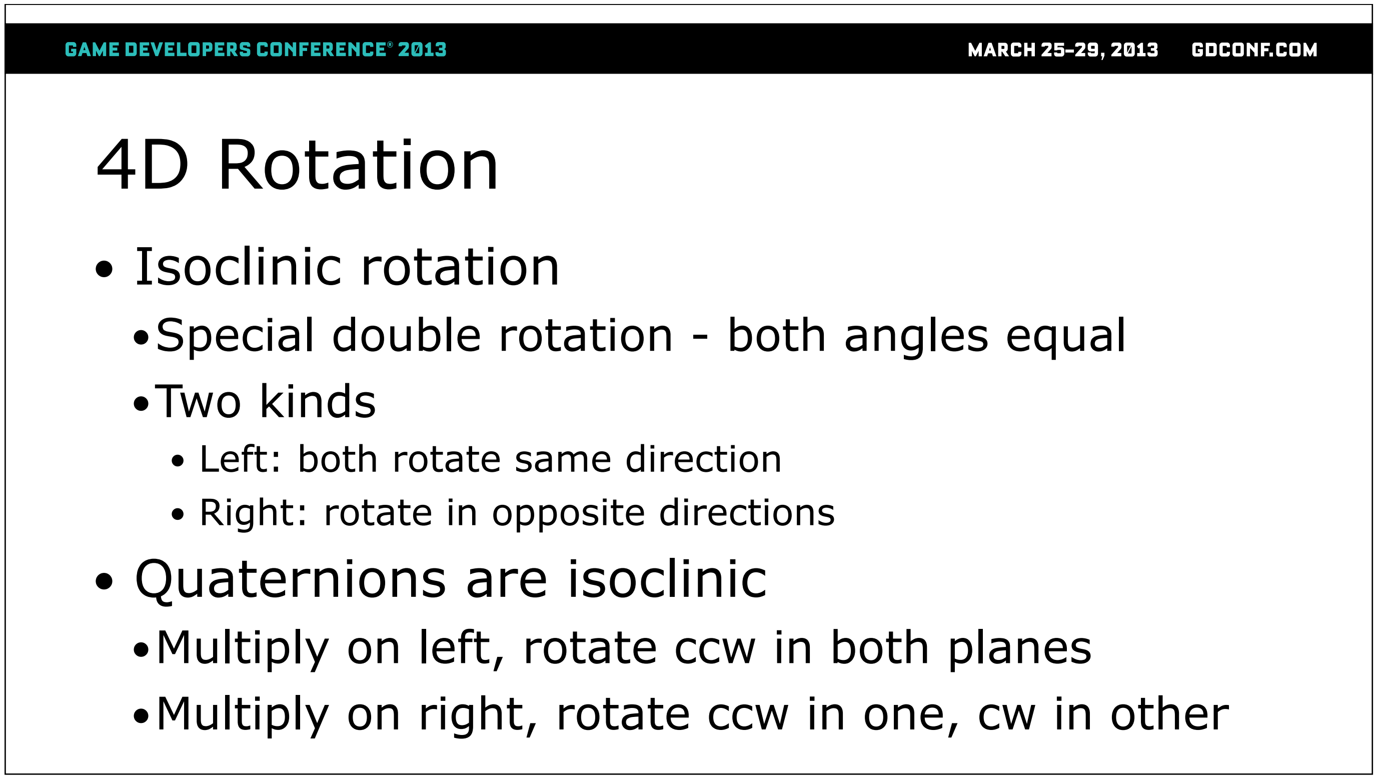


4D Rotation

- Two types:
  - Single rotation (one plane, like 2D/3D)
  - Double rotation (two orthogonal planes!)

4D rotation is -- shall we say -- unusual. I used the term weird in an article and one reviewer wrote, “To you!” In any case because we have 4 components to play with, there are two different classes of rotations.

Quaternions do not perform general 4D rotation. However, any double rotation can be decomposed into two isoclinic/quaternion rotations -- one on the left and one on the right.



4D Rotation

- Isoclinic rotation
  - Special double rotation - both angles equal
  - Two kinds
    - Left: both rotate same direction
    - Right: rotate in opposite directions
- Quaternions are isoclinic
  - Multiply on left, rotate ccw in both planes
  - Multiply on right, rotate ccw in one, cw in other

In isoclinic rotations we rotate by the same angle in both planes, but not necessary in the same direction.

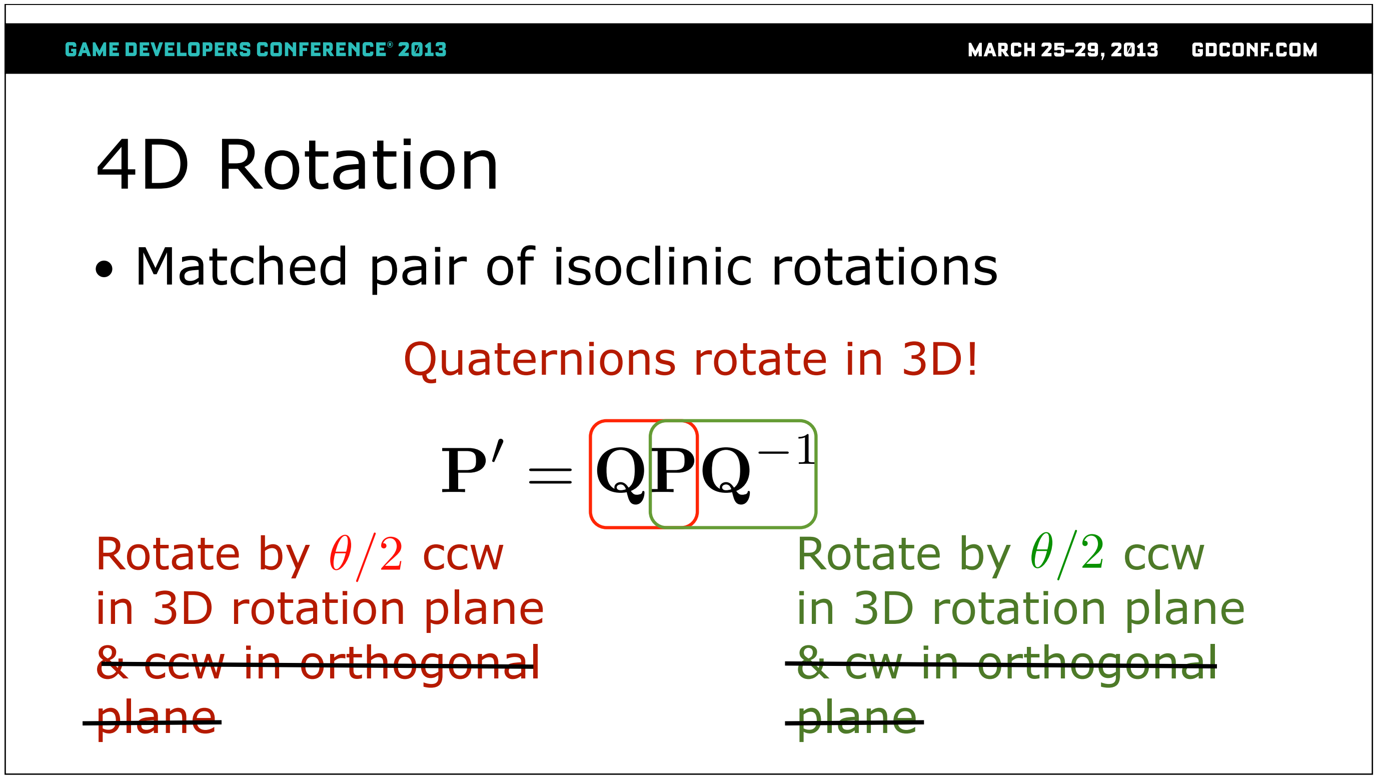
4D Rotation

- Matched pair of isoclinic rotations

$$\mathbf{P}' = \mathbf{Q}\mathbf{P}\mathbf{Q}^{-1}$$

Rotate by $\theta/2$ ccw in 3D rotation plane & ccw in orthogonal plane

Rotate by $\theta/2$ ccw in 3D rotation plane & cw in orthogonal plane



4D Rotation

- Matched pair of isoclinic rotations

Quaternions rotate in 3D!

$$\mathbf{P}' = \mathbf{QPQ}^{-1}$$

Rotate by $\theta/2$ ccw in 3D rotation plane ~~& ccw in orthogonal plane~~

Rotate by $\theta/2$ ccw in 3D rotation plane ~~& cw in orthogonal plane~~

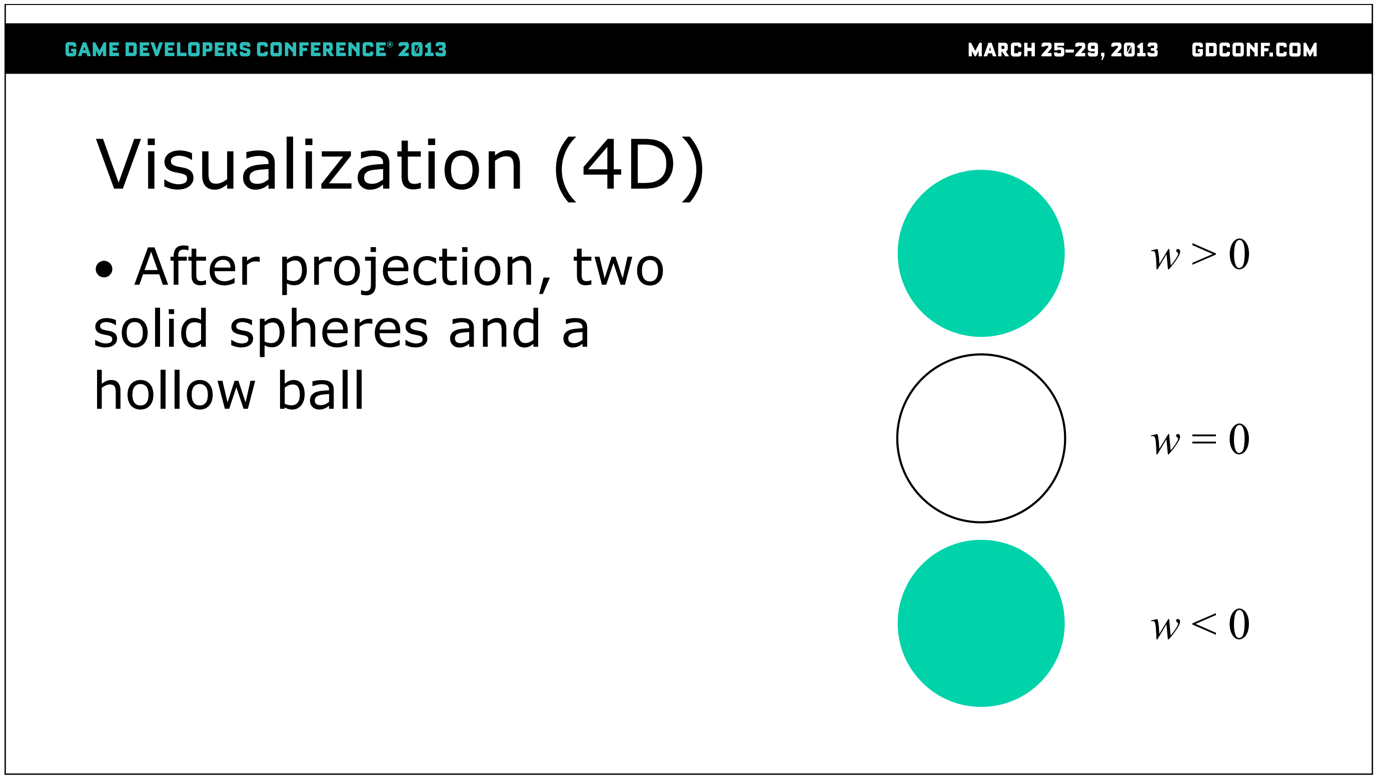
In the end, the rotations through the second plane cancel each other out, and we end up with just a rotation along the 3D plane.

Visualization

- Final question:
  - We are 3D creatures
  - How can we visualize a 4D concept?

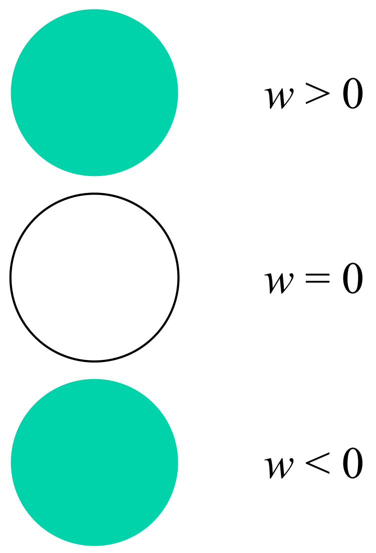
Visualization

- Project into 3D as follows:
  - Break 4D hypersphere into three pieces
  - 4D hemisphere projects to 3D sphere
  - I.e. drop w and use its sign

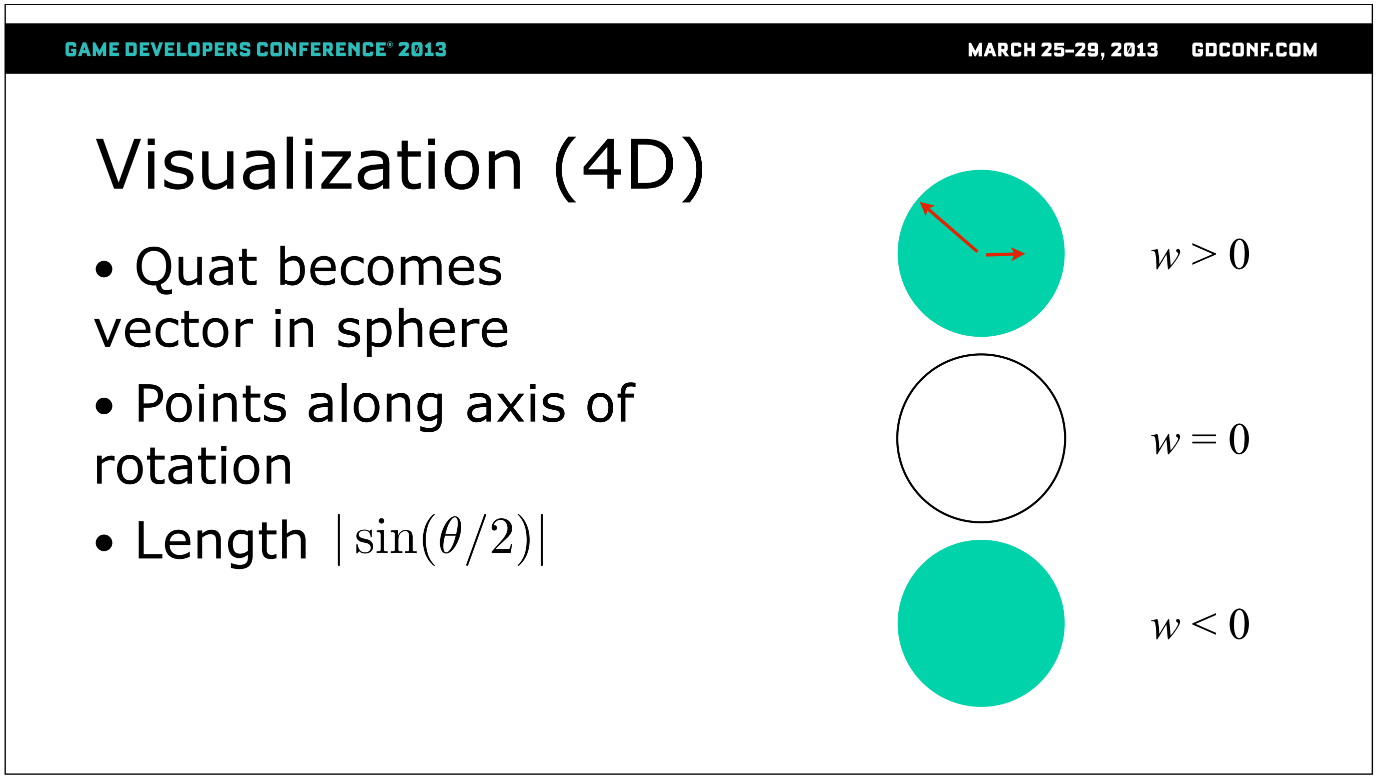


Visualization (4D)

- After projection, two solid spheres and a hollow ball

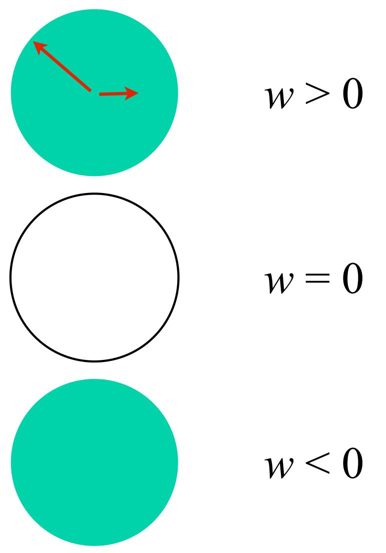


In the fourth dimension, the two solid spheres are connected together via the hollow ball.

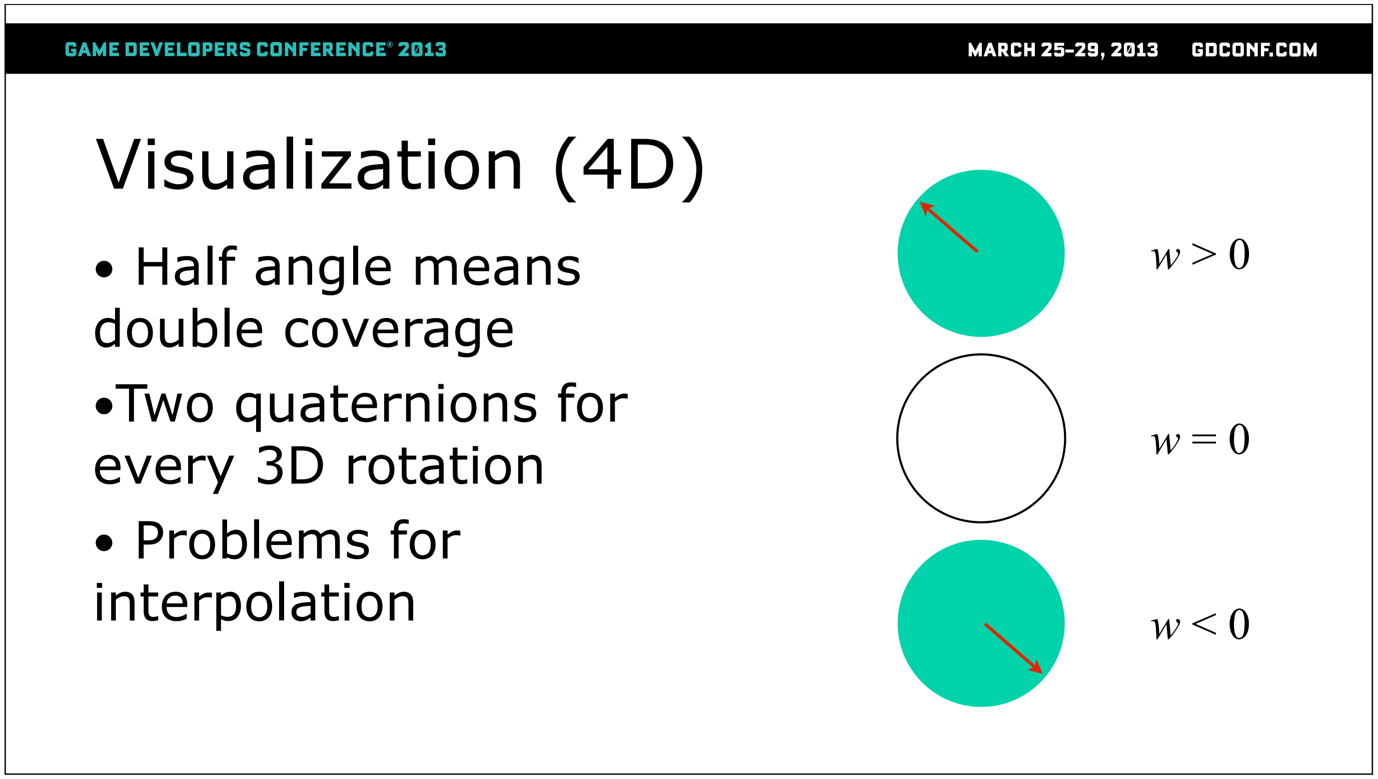


Visualization (4D)

- Quat becomes vector in sphere
- Points along axis of rotation
- Length $|\sin(\theta/2)|$

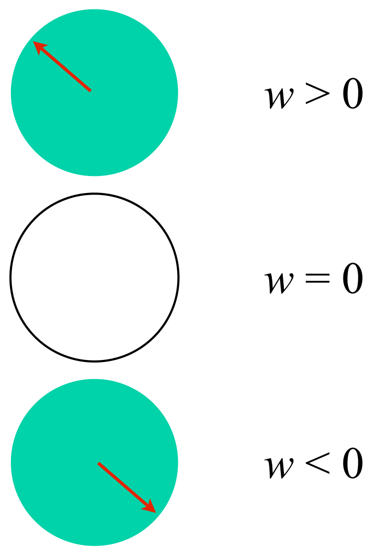


So the longer the vector, the more rotation you get. If the vector's length is zero, then you have the identity quaternion. Quaternions in the w>0 sphere represent rotations of -pi to pi. If w < 0, then the rotations are still from -pi to pi, but the axis will be flipped. Anything on the w=0 ball represents a rotation of 180 degrees.

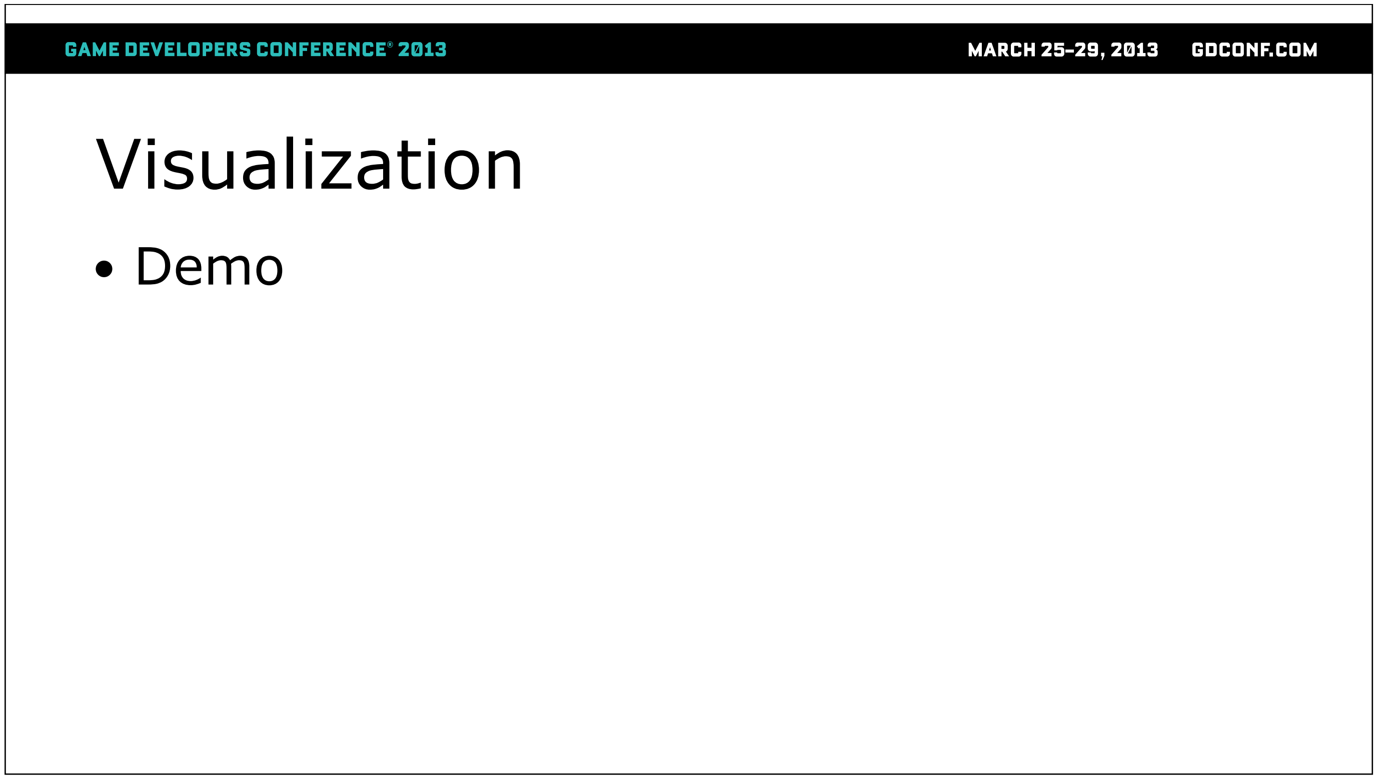


Visualization (4D)

- Half angle means double coverage
- Two quaternions for every 3D rotation
- Problems for interpolation



If we try to interpolate from a quaternion in the w < 0 sphere to the w > 0 sphere, we end up taking the long way around. This why it's recommended when you're interpolating to take the dot product of the quaternions -- if the result is negative, negate one and then interpolate.



Visualization

- Demo

This makes my head hurt, so maybe it's better to see it in action.

Wrap Up

- Why four values? Want a division algebra.
- What are i, j and k? Imaginary axes
- Why $\theta/2$? Two step rotation.
- Why qpq^{-1}? Can't rotate directly w/quaternion.
- How can I think in 4D? Think axis scaled by sine of half angle.

References

- Andrew Hanson, *Visualizing Quaternions*
- Ron Goldman, *Rethinking Quaternions*
- David Eberly, *Game Physics* (especially Shoemake section)
- Van Verth and Bishop, *Essential Mathematics for Games and Interactive Applications*

Questions?