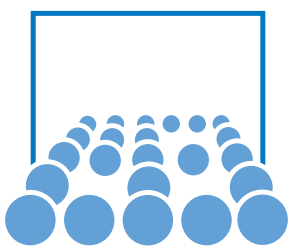
PSE Game Physics

Session (3)
Springs, Ropes, Linear Momentum and Rotations

Sebastian Rettenberger, Roland Wittmann

29.04.2016



Outline

Springs and ropes
- Overview
- Springs
- Ropes

Collisions
- Collision detection
- Collision resolution

Linear momentum
- Basics
- Conservation laws
- Why use the inverse of mass?

Rotations

Outline

Springs and Ropes

- Springs and ropes are so called **constraints** since they force different objects to behave in a specific way.
- Other constraints are e. g. a **hinge** or a **wheel** attached to a car
- Springs and ropes are frequently used in games:
  - Ropes and springs to drag & drop something
  - Building bridges out of ropes
  - Setting up a “3rd-person“ camera with damped spring
  - Connecting 2 objects
  - ...
- For this worksheet, we assume a few simplifications:
  - The spring is attached to the mass center of the object
  - Spring/Rope collisions are not considered

Springs (1/2)

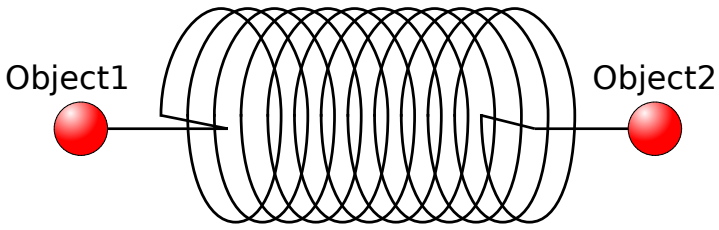


Figure: Spring

- Create a **force acting on both involved objects**
- Force depends on the **distance** of the spring anchors
- Hook's law gives us the force acting on each object:

$$\mathbf{F} = -k \cdot \mathbf{d}$$

- F: Applied force, k: Spring coefficient, d: spring elongation (= length - equilibrium length)

Springs (2/2)

- How to implement this force into our physics engine?
- This force can be implemented via a **soft constraint**:
  - We can easily **compute the acceleration** exceeded by the spring's force for each object within a time-step (use an explicit euler step).
  - Then the change of acceleration can be simply **added to the acceleration accumulator** during each time-step.
  - The **direction** is given by the line between the **anchor points** of both objects.
- Question: Are there problems with large time-steps and large spring coefficients? How can we handle those?

Ropes (1/2)

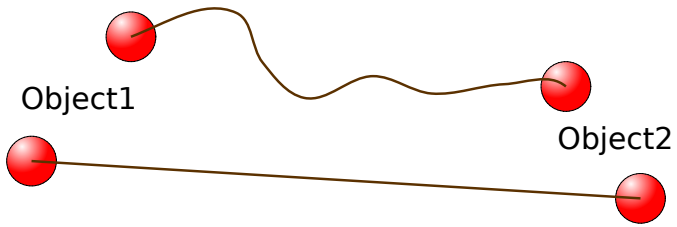


Figure: Top image: rope without exceeding its maximum length, Bottom image: “Colliding rope“

- We only consider ropes **without rope collisions**!!!
- After finishing all worksheets, you’ll be able to simulate this as well :-)

Ropes (2/2)

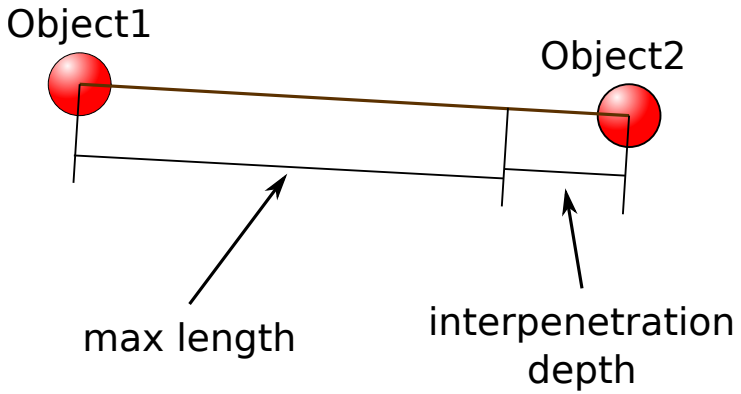


Figure: Colliding details

- Imagine 2 objects with increasing distance.
- Once the anchor points of the ropes at the objects exceed a specific rope length an **impulse acts onto both attached objects**
- This is similar to applying an impulse during collision handling
  $\rightarrow$ **hard constraint**
  $\rightarrow$ Create a simple collision dataset

Outline

Collision detection

Basic algorithmic steps for collision detection:

1. Take any two colliding objects

Collision detection

Basic algorithmic steps for collision detection:

1. Take any two colliding objects
2. (Optional) Coarse culling: find a lower bound for the distance between the two objects
   - If this lower bound is greater than zero, abort

Collision detection

Basic algorithmic steps for collision detection:

1. Take any two colliding objects
2. (Optional) Coarse culling: find a lower bound for the distance between the two objects
   - If this lower bound is greater than zero, abort
3. Fine culling: Determine collision points, normal, interpenetration depth (usually goes hand in hand)
   - If interpenetration depth is less than zero, abort

Collision detection

Basic algorithmic steps for collision detection:

1. Take any two colliding objects
2. (Optional) Coarse culling: find a lower bound for the distance between the two objects
   - If this lower bound is greater than zero, abort
3. Fine culling: Determine collision points, normal, interpenetration depth (usually goes hand in hand)
   - If interpenetration depth is less than zero, abort
4. Set collision parameters and start collision resolution for the two objects

Collision detection

Basic algorithmic steps for collision detection:

1. Take any two colliding objects
2. (Optional) Coarse culling: find a lower bound for the distance between the two objects
   - If this lower bound is greater than zero, abort
3. Fine culling: Determine collision points, normal, interpenetration depth (usually goes hand in hand)
   - If interpenetration depth is less than zero, abort
4. Set collision parameters and start collision resolution for the two objects
5. Check if any other collisions occured (always necessary?), if yes return to step 1

Collision resolution

Goals:

- No more collision between the two objects, obviously
- Very little movement, which is visually unpleasant and might cause further collisions
- Somehow physical behaviour

Collision resolution

Goals:

- No more collision between the two objects, obviously
- Very little movement, which is visually unpleasant and might cause further collisions
- Somehow physical behaviour

Steps:

- Move the objects apart from each other (along what direction?):
  - Move Object 1 by $\frac{m_1^{-1}}{m_1^{-1}+m_2^{-2}}$ of the interpenetration depth.
  - Move Object 2 by $\frac{m_2^{-1}}{m_1^{-1}+m_2^{-2}}$ of the interpenetration depth.

Collision resolution

Goals:

- No more collision between the two objects, obviously
- Very little movement, which is visually unpleasant and might cause further collisions
- Somehow physical behaviour

Steps:

- Move the objects apart from each other (along what direction?):
  - Move Object 1 by $\frac{m_1^{-1}}{m_1^{-1}+m_2^{-2}}$ of the interpenetration depth.
  - Move Object 2 by $\frac{m_2^{-1}}{m_1^{-1}+m_2^{-2}}$ of the interpenetration depth.
- (Later) Update velocities by applying forces to the objects
- Why the weird inverse masses?

Collision resolution

Goals:

- No more collision between the two objects, obviously
- Very little movement, which is visually unpleasant and might cause further collisions
- Somehow physical behaviour

Steps:

- Move the objects apart from each other (along what direction?):
  - Move Object 1 by $\frac{m_1^{-1}}{m_1^{-1}+m_2^{-2}}$ of the interpenetration depth.
  - Move Object 2 by $\frac{m_2^{-1}}{m_1^{-1}+m_2^{-2}}$ of the interpenetration depth.
- (Later) Update velocities by applying forces to the objects
- Why the weird inverse masses? Design decision: Allow infinite mass, neglect zero mass!

Outline

Linear momentum

Once there is a collision, several things have to be done:

- Resolving the interpenetration (finished during last session)
- The velocity of at least one object has to be modified according to its current state

So far, we only consider elastic collisions!

Basics

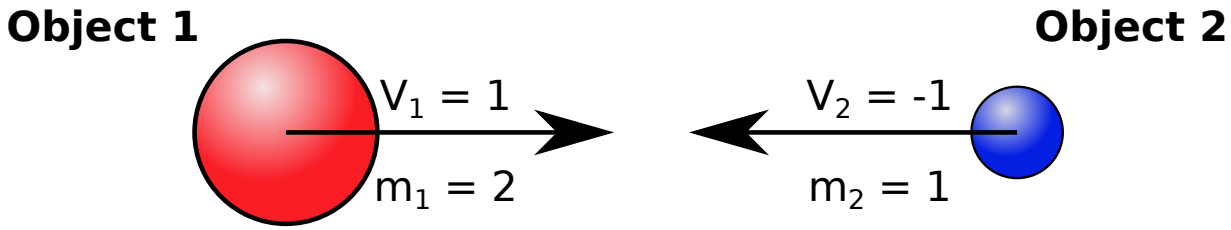


Figure: 2 colliding spheres

- Relevant object properties for linear momentum in 1D:
  - mass of objects: m_1, m_2
  - velocity of object before collision: v_1, v_2
  - velocity of object after collision: v_1', v_2'
- **Closing velocity**:

$$v_c = v_1 - v_2$$

- What's the separating velocity for each object?

Force and Linear Impulse

- A Force leads to change of acceleration.

$$F = ma$$

- Is it possible to change the velocity instantaneously?
- Idea : use the Impulse.

$$J = m\,\Delta v = \int_{t_1}^{t_2} F\,dt$$

We start with a few basic conservation laws:

- **Energy conservation** for elastic collisions:

$$\frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 = \frac{1}{2} m_1 v_1'^2 + \frac{1}{2} m_2 v_2'^2$$

- **Conservation of momentum**:

$$m_1 v_1 + m_2 v_2 = m_1 v_1' + m_2 v_2'$$

- We start by **rewriting the energy conservation law** to

$$m_1(v_1^2 - v_1'^2) = -m_2(v_2^2 - v_2'^2)$$

and avoid an expression of the velocity with a square exponent:

$$\underbrace{m_1(v_1 - v_1')}_{A}(v_1 + v_1') = -m_2(v_2 - v_2')(v_2 + v_2')$$

- By rewriting the **equation for the conservation of momentum** to

$$m_1(v_1 - v_1') = -m_2(v_2 - v_2')$$

- we can replace (A) by the right side of the upper equation:

$$-m_2(v_2 - v_2')(v_1 + v_1') = -m_2(v_2 - v_2')(v_2 + v_2')$$

- Finally, we get the first result: $v_1 + v_1' = v_2 + v_2'$ or

$$v_1' - v_2' = -(v_1 - v_2)$$

where the **separating velocity** $v_s = v_1' - v_2'$ is the exact opposite of the closing velocity $v_c = v_1 - v_2$.

Updated Velocities

- Using the equation $v_1' - v_2' = -(v_1 - v_2)$ of the previous slide, we can rearrange it to

$$v_2' = v_1 - v_2 + v_1'$$

- By putting this equation in the momentum equation, we get

$$m_1(v_1 - v_1') = -m_2(v_2 - (v_1 - v_2 + v_1')) = -m_2 v_2 + m_2 v_1 - m_2 v_2 + m_2 v_1'$$

$$\Leftrightarrow m_1 v_1' + m_2 v_1' = 2m_2 v_2 - m_2 v_1 + m_1 v_1 \overbrace{+m_2 v_1 - m_2 v_1}^{\text{Extension by 0}}$$

$$\Leftrightarrow (m_1 + m_2) v_1' = 2(m_2 v_2 - m_2 v_1) + (m_1 + m_2) v_1$$

- By dividing the equation by $(m_1 + m_2)$, the velocity of object 1 after collision is given by

$$\Leftrightarrow v_1' = 2\frac{m_2(v_2 - v_1)}{m_1 + m_2} + v_1 = -2\frac{m_2}{m_1 + m_2} v_c + v_1$$

- Normally, collisions are not completely elastic. The **coefficient of restitution** C_r is used to represent the respective deviation depending on the closing and separating velocity:

$$C_r = \frac{v_2' - v_1'}{v_1 - v_2} = \frac{-v_s}{v_c}$$

- With inclusion of the coefficient of restitution, the velocity of object 1 after collision is given (without proof) by

$$\Leftrightarrow v_1' = -(1 + C_r)\frac{m_2}{m_1 + m_2} v_c + v_1$$

- Similarly, the velocity of object 2 after the collision is given by

$$\Leftrightarrow v_2' = (1 + C_r)\frac{m_1}{m_1 + m_2} v_c + v_2$$

Getting more insight, no formulas

- Understanding the formulas is crucial. Therefore we have a look at the most important parts:

$$\underbrace{v_1'}_{\text{new velocity}} = -\underbrace{(1+C_r)}_{1+C_r}\underbrace{\frac{m_2}{m_1+m_2}}_{\text{mass fraction}}\underbrace{v_c}_{\text{closing velocity}} + \underbrace{v_1}_{\text{old velocity}}$$

- **Old/New velocity**: Velocity of object 1 *before*/*after* the collision
- **Closing velocity**: Relative speed of the two objects
- **Mass fraction**: The fraction accounts for the change of velocity depending on the fraction m_{object}/m_{total}
- $\mathbf{1 + C_r}$: Imagine a collision between a moving object 1 with mass 0 and a steady object 2 with mass 1.
  - Then object 2 doesn't move at all. Object 1 reverses its velocity and is slowed down by the factor C_r.
  - Therefore $v_1' = -C_r v_1 = -(1+C_r)v_1 + v_1 = -(1+C_r)v_c + v_1$.

Using inverse mass...

- Computing v_2' directly, we get problems when the masses of the objects become huge, i.e. approach infinity. Assuming, that $m_1 = \infty$, we get

$$\Leftrightarrow v_2' = (1 + C_r) \underbrace{\frac{\infty}{\infty + m_2}}_{\textit{undefined}!!!} v_c + v_2$$

- Without further checks, we run into severe problems due to the fact, that arithmetic datatypes are finite.
- Therefore we will work with the inverse of the mass:

$$\frac{m_1}{m_1 + m_2} = \frac{m_1\ m_1^{-1}\ m_2^{-1}}{(m_1 + m_2)\ m_1^{-1}\ m_2^{-1}} = \frac{m_2^{-1}}{m_1^{-1} + m_2^{-1}}$$

Momentum in 3D

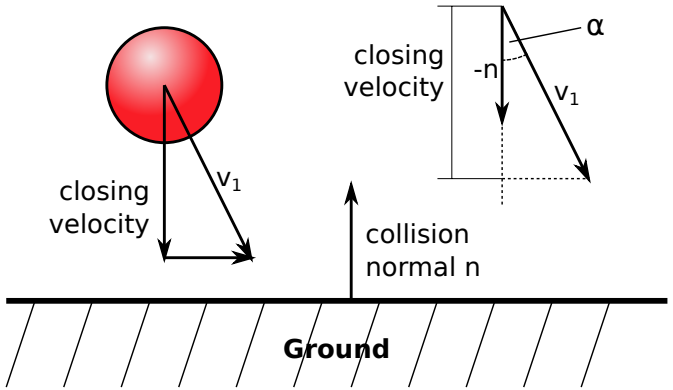


Figure: Collision involving the collision normal

- For 3D (any dimension higher than 1D), we have to take the collision normal into account.
- Only the velocity component parallel to the collision normal may be involved in computing the new velocity (no friction so far!)
- Thus the collision velocity $\overline{v}$ is computed by using only the fraction of the velocity in direction of the collision normal.
- Using the dot product we can compute this fraction by: $\overline{v} = -\vec{n}_c \cdot \vec{v}$

Collision normal? Wtf is that?

- The collision normal describes the direction at which repelling forces act on the two colliding objects upon impact.
- Imagine two arbitrary objects colliding without any friction. What restrictions must these forces meet at the time of impact?

Collision normal? Wtf is that?

- The collision normal describes the direction at which repelling forces act on the two colliding objects upon impact.
- Imagine two arbitrary objects colliding without any friction. What restrictions must these forces meet at the time of impact?
  - They sum up to zero (Newton's Third law)

Collision normal? Wtf is that?

- The collision normal describes the direction at which repelling forces act on the two colliding objects upon impact.
- Imagine two arbitrary objects colliding without any friction. What restrictions must these forces meet at the time of impact?
  - They sum up to zero (Newton's Third law)
  - They are perpendicular to the surface (No friction!)

Collision normal? Wtf is that?

- The collision normal describes the direction at which repelling forces act on the two colliding objects upon impact.
- Imagine two arbitrary objects colliding without any friction. What restrictions must these forces meet at the time of impact?
  - They sum up to zero (Newton's Third law)
  - They are perpendicular to the surface (No friction!)
- So what are the collision normals for the following cases?
  - Sphere-Sphere:
  - Sphere-Plane:
  - Sphere-Edge:
  - Sphere-Rectangle:

Collision normal? Wtf is that?

- The collision normal describes the direction at which repelling forces act on the two colliding objects upon impact.
- Imagine two arbitrary objects colliding without any friction. What restrictions must these forces meet at the time of impact?
  - They sum up to zero (Newton's Third law)
  - They are perpendicular to the surface (No friction!)
- So what are the collision normals for the following cases?
  - Sphere-Sphere: Line connecting sphere centers
  - Sphere-Plane:
  - Sphere-Edge:
  - Sphere-Rectangle:

Collision normal? Wtf is that?

- The collision normal describes the direction at which repelling forces act on the two colliding objects upon impact.
- Imagine two arbitrary objects colliding without any friction. What restrictions must these forces meet at the time of impact?
  - They sum up to zero (Newton's Third law)
  - They are perpendicular to the surface (No friction!)
- So what are the collision normals for the following cases?
  - Sphere-Sphere: Line connecting sphere centers
  - Sphere-Plane: Plane normal
  - Sphere-Edge:
  - Sphere-Rectangle:

Collision normal? Wtf is that?

- The collision normal describes the direction at which repelling forces act on the two colliding objects upon impact.
- Imagine two arbitrary objects colliding without any friction. What restrictions must these forces meet at the time of impact?
  - They sum up to zero (Newton's Third law)
  - They are perpendicular to the surface (No friction!)
- So what are the collision normals for the following cases?
  - Sphere-Sphere: Line connecting sphere centers
  - Sphere-Plane: Plane normal
  - Sphere-Edge: Shortest line segment connecting sphere center and edge
  - Sphere-Rectangle:

Collision normal? Wtf is that?

- The collision normal describes the direction at which repelling forces act on the two colliding objects upon impact.
- Imagine two arbitrary objects colliding without any friction. What restrictions must these forces meet at the time of impact?
  - They sum up to zero (Newton's Third law)
  - They are perpendicular to the surface (No friction!)
- So what are the collision normals for the following cases?
  - Sphere-Sphere: Line connecting sphere centers
  - Sphere-Plane: Plane normal
  - Sphere-Edge: Shortest line segment connecting sphere center and edge
  - Sphere-Rectangle: Two known cases: Sphere-Plane and Sphere-Edge

Outline

Rotations

Different approaches to describe rotations:

Rotations

Different approaches to describe rotations:

- Angles
- Matrices
- (Later) Quaternions

Generally, assume there are $n \in \mathbb{N}$ degrees of freedom $r \in \mathbb{R}^n$ that describe a rotation. What operators do we need for rotations then?

Rotations

Different approaches to describe rotations:

- Angles
- Matrices
- (Later) Quaternions

Generally, assume there are $n \in \mathbb{N}$ degrees of freedom $r \in \mathbb{R}^n$ that describe a rotation. What operators do we need for rotations then?

- Mapping M, which creates the description r for the rotation.
- Transformation $T_r : \mathbb{R}^d \rightarrow R^d$, which applies the rotation r to points
- Concatenation $\circ : \mathbb{R}^n \times \mathbb{R}^n \rightarrow R^n$, which allows to combine two rotations r_1 and r_2 to a new rotation $r = r_1 \circ r_2$

Rotations in 2D

Angles:

Rotations in 2D

Angles:

- Store the rotation angle α (1 DoF).
- Concatenation: Easy, $\alpha = \alpha_1 + \alpha_2$:)
- Transformation: Use 2D Rotation Matrix
  $R(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$, then $T_\alpha(x) = R(\alpha)x$

Matrices:

Rotations in 2D

Angles:

- Store the rotation angle α (1 DoF).
- Concatenation: Easy, $\alpha = \alpha_1 + \alpha_2$:)
- Transformation: Use 2D Rotation Matrix
  $R(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$, then $T_\alpha(x) = R(\alpha)x$

Matrices:

- Store a rotation matrix R (4 DoFs)
- Concatenation: Matrix multiplication $R = R_1 R_2$.
- Transformation: $T_R(x) = R\,x$
- Difference to angles?

Rotations in 2D

Angles:

- Store the rotation angle α (1 DoF).
- Concatenation: Easy, $\alpha = \alpha_1 + \alpha_2$:)
- Transformation: Use 2D Rotation Matrix
  $R(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$, then $T_\alpha(x) = R(\alpha)x$

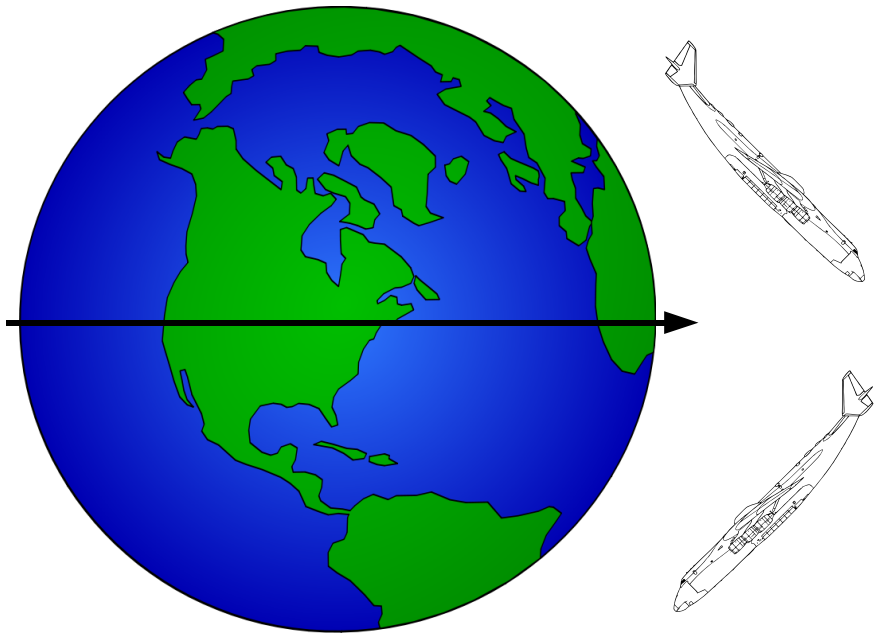
Matrices:

- Store a rotation matrix R (4 DoFs)
- Concatenation: Matrix multiplication $R = R_1 R_2$.
- Transformation: $T_R(x) = R\,x$
- Difference to angles?

Example: Rotate vector $\vec{v} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ by 90° (=$\frac{\pi}{2}$)

$$\rightarrow \qquad \vec{v}^{new} = R(90°) \cdot \vec{v} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad (1)$$

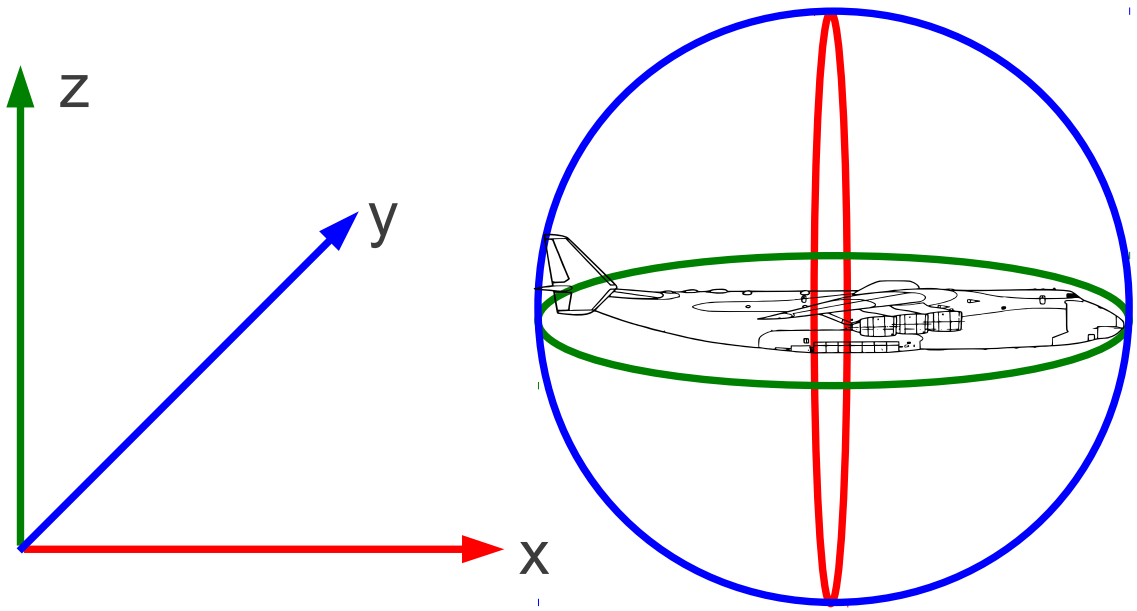
Rotations in 3D: it's not that easy...



- $\rightarrow$ Handling rotations sounds straight-forward, but definitely isn't!
- Autopilot on F-16 flipped the airplane upside-down when crossed the equator!

Euler Angles

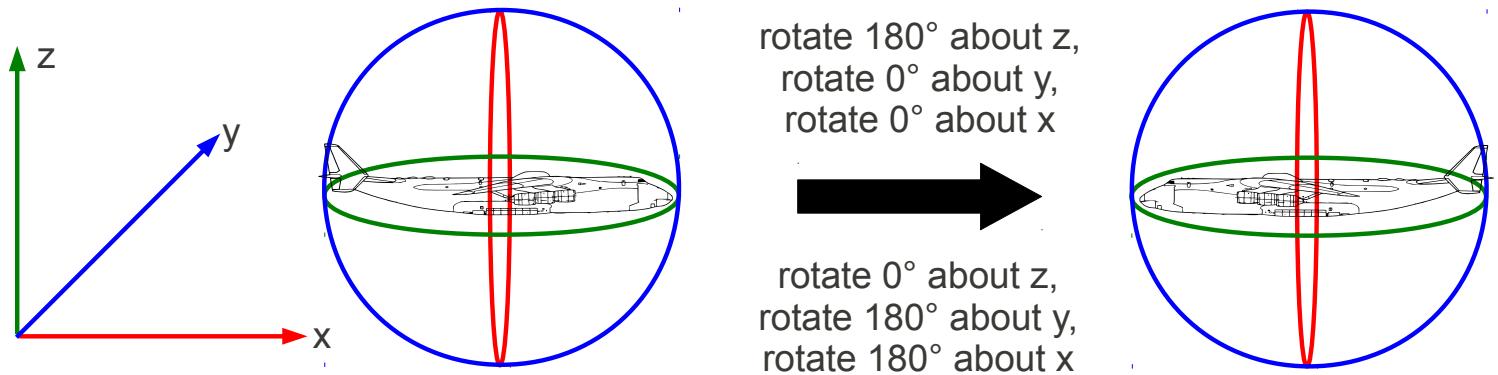
- **From 2D rotation...**: 1 rotation axis $\rightarrow$ therefore, we use one type of rotation matrix R^{2D}
- **...to 3D**: 3 possible rotation axes (x,y,z) $\rightarrow$ use three rotations to describe any rotation in space
- Euler angles: Describe rotated position by angles between this position and reference position



Euler Angles: Problems

- Different conventions/ approaches
- Example: zyx-convention
  1. Rotation about z-axis
  2. Rotation about y-axis
  3. Rotation about x-axis

$\rightarrow$ 3 degrees of freedom



$\rightarrow$ **Rotational description not unique**

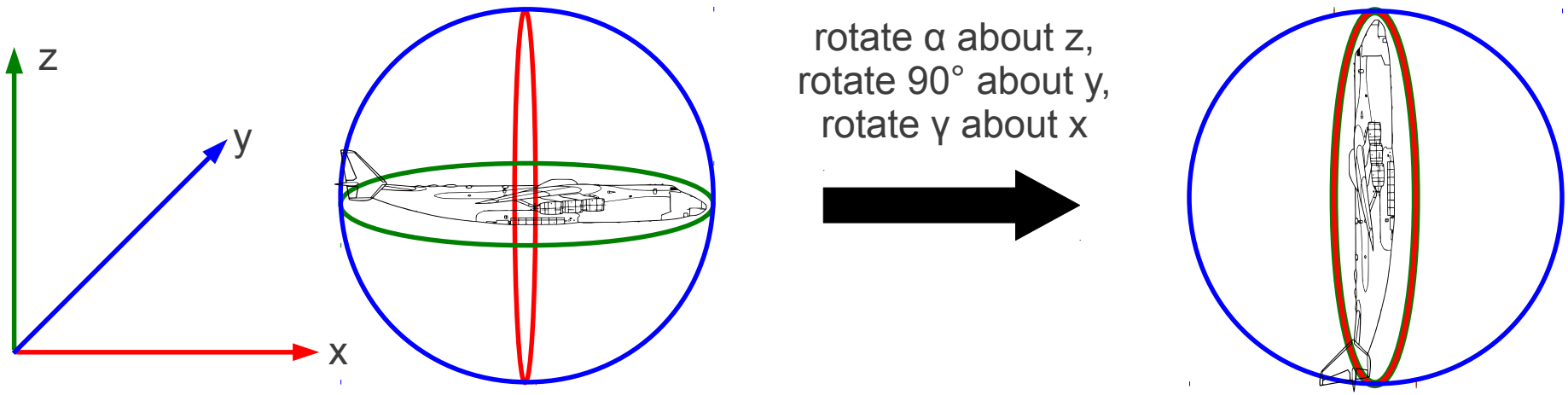
Euler Angles: Problems - Gimbal Lock (1)

Excerpt from the NASA logs and commentary (taken from *Visualizing Quaternions, A.J. Hanson,2006*):

... Stafford had the vehicle in the right attitude 10 minutes early. Cernan asked, "You ready?" Then he suddenly exclaimed, "Son of a bitch!" Snoopy seemed to be throwing a fit, lurching wildly about. He later said it was like flying an Immelmann turn in an aircraft, a combination of pitch and yaw. Stafford yelled that **they were in gimbal lock**–that the engine had swiveled over to a stop and stuck- and they almost were. He called out for Cernan to thrust forward. Stafford then hit the switch to get rid of the descent stage and realized they were 30 degrees off from their previous attitude. The lunar module continued its crazy gyrations across the lunar sky, and a warning light indicated that the inertial measuring unit really was about to reach its limits and go into gimbal lock. Stafford then took over in manual control, made a big pitch maneuver, and started working the attitude control switches. Snoopy finally calmed down.

Euler Angles: Problems - Gimbal Lock (2)

- Gimbal Lock: Loss of one degree of freedom in rotations



- Example: *z*- and *x*-axis become parallel.
- Solution: Allow rotations around axes Z, Y, X in different order.
- Better solution: Use quaternions.

Rotations in 3D

Angles:

- Store angles α, β, γ for rotations about unit vectors (3 DoFs)
- Transformation: $T_r(x) = R_1(\alpha)R_2(\beta)R_3(\gamma)x$.
- Concatenation: Same as 2D, add up angles
- Each time we rotate around one axis 1, we rotate axes 2 and 3 as well $\rightarrow$ Problem?

Rotations in 3D

Angles:

- Store angles α, β, γ for rotations about unit vectors (3 DoFs)
- Transformation: $T_r(x) = R_1(\alpha)R_2(\beta)R_3(\gamma)x$.
- Concatenation: Same as 2D, add up angles
- Each time we rotate around one axis 1, we rotate axes 2 and 3 as well $\rightarrow$ Problem?

Matrices:

- Store matrix R (9 DoFs)
- Similar to 2D, concatenation by matrix multiplication and transformation by matrix-vector product
- Difference to angles: Imagine many small updates to the rotation. What happens in both cases?

Rotations in 3D

Angles:

- Store angles α, β, γ for rotations about unit vectors (3 DoFs)
- Transformation: $T_r(x) = R_1(\alpha)R_2(\beta)R_3(\gamma)x$.
- Concatenation: Same as 2D, add up angles
- Each time we rotate around one axis 1, we rotate axes 2 and 3 as well $\rightarrow$ Problem?

Matrices:

- Store matrix R (9 DoFs)
- Similar to 2D, concatenation by matrix multiplication and transformation by matrix-vector product
- Difference to angles: Imagine many small updates to the rotation. What happens in both cases?

Quaternions:

- Try and take the best of both approaches
- Eliminate Gimbal Lock

Quaternions: Basics

- Describe a rotation θ around an *arbitrary* axis (x, y, z) in space
- Advantage: 3+1 degrees of freedom
- Solves the Gimbal Lock-Problem
- Given rotation axis (x, y, z), with rotation axis length $\|(x, y, z)^T\| = 1$ and angle θ, the corresponding quaternion q is given by

$$q := \begin{pmatrix} w \\ i \\ j \\ k \end{pmatrix} = \begin{pmatrix} \cos\frac{\theta}{2} \\ x\sin\frac{\theta}{2} \\ y\sin\frac{\theta}{2} \\ z\sin\frac{\theta}{2} \end{pmatrix} \tag{2}$$

Quaternions: Multiplication

- **Applying two rotations** given by quaternions q and p to an object means to apply the **multiplication of both quaternions**: $q \cdot p$.
- Multiplication of two quaternions is given by

$$\begin{pmatrix} w_1 \\ i_1 \\ j_1 \\ k_1 \end{pmatrix} \cdot \begin{pmatrix} w_2 \\ i_2 \\ j_2 \\ k_2 \end{pmatrix} = \begin{pmatrix} w_1 w_2 - (i_1 i_2 + j_1 j_2 + k_1 k_2) \\ w_1 i_2 + w_2 i_1 + j_1 k_2 - k_1 j_2 \\ w_1 j_2 + w_2 j_1 + k_1 i_2 - i_1 k_2 \\ w_1 k_2 + w_2 k_1 + i_1 j_2 - j_1 i_2 \end{pmatrix}$$

- In this way the rotation of p is applied first, then the rotation expressed with the Quaternion q.

Quaternions: Convertion to rotation matrix

- Convert quaternion back to **general rotation matrix** R via

$$R = \begin{pmatrix} 1-2(j^2+k^2) & 2(ij+kw) & 2(ik-jw) \\ 2(ij-kw) & 1-2(i^2+k^2) & 2(jk+iw) \\ 2(ik+jw) & 2(jk-iw) & 1-2(i^2+j^2) \end{pmatrix}$$

Things, you should remember...

Linear transformations:

- Objects are projected **from *model-space* to *world-space*** by applying the matrix M to the object.
- We can simply **invert** the model matrix M to transform "**backwards**" from world-space to object-space.
- Object coordinates are given with **homogenous coordinates** to **account for translations**.
- Use the **inverse-transpose** M^{-T} to transform **normals**.
- Use **quaternions** for rotations to avoid the **gimbal-lock**.
- Always **draw sketches**!

Any questions?