LECTURE 9: MOTION IN REAL AND VIRTUAL WORLDS

Ehsan Aryafar
earyafar@pdx.edu
http://web.cecs.pdx.edu/~aryafare/VR.html

Recall: Image-Order Rendering

```
001 for (each pixel in image) {
002     Ray R = computeRayPassingThroughPixel(x,y);
003     float tclosest = INFINITY;
004     Triangle triangleClosest = NULL;
005     for (each triangle in scene) {
006         float thit;
007         if (intersect(R, object, thit)) {
008             if (thit < closest) {
009                 triangleClosest = triangle;
010             }
011         }
012     }
013     if (triangleClosest) {
014         imageAtPixel(x,y) = triangleColorAtHitPoint(triangle, tclosest);
015     }
016 }
```

Recall: Object-Order Rendering

```
001  // A z-buffer is just an 2D array of floats
002  float buffer = new float [imageWidth * imageHeight];
003  // initialize the distance for each pixel to a very large number
004  for (uint32_t i = 0; i < imageWidth * imageHeight; ++i)
005      buffer[i] = INFINITY;
006
007  for (each triangle in scene) {
008      // project vertices
009      ...
010      // compute bbox of the projected triangle
011      ...
012      for (y = ymin; y <= ymin; ++y) {
013          for (x = xmin; x <= xmax; ++x) {
014              // check of if current pixel lies in triangle
015              float z; // distance from the camera to the triangle
016              if (pixelContainedIn2DTriangle(v0, v1, v2, x, y, z)) {
017                  // If the distance to that triangle is lower than the distance
018                  // z-buffer, update the z-buffer and update the image at pixel
019                  // with the color of that triangle
020                  if (z < zbuffer(x,y)) {
021                      zbuffer(x,y) = z;
022                      image(x,y) = triangle[i].color;
023                  }
024              }
025          }
026      }
027  }
```

Recall: VR Latency Reduction Methods

- Reduce the complexity of the VR world
  - Helps with rasterization as well as VWG
- Improve rendering pipeline performance
- Remove delays along the path from rendered image to switching pixels
- Use prediction to estimate future viewpoints and world states
- Shift or distort the rendered image to compensate for last-moment viewport errors

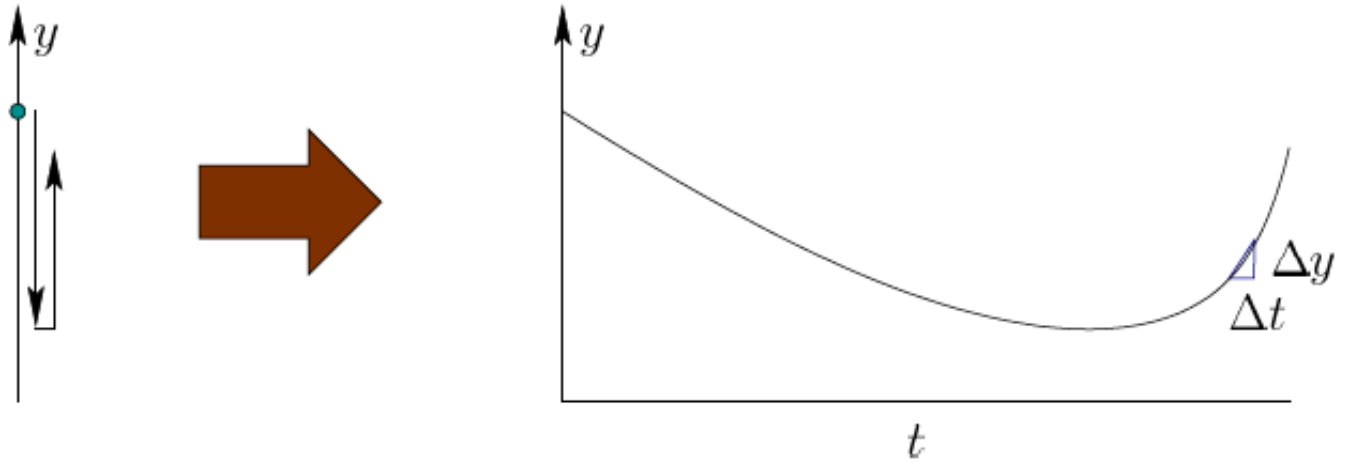
This Lecture

- Model motions for both real and virtual worlds
  - Acceleration and velocity models impact simulations in VWG
  - Also impact the methods used to track the user in real world

Outline

- Velocities, acceleration, and movement of rigid bodies
  - Fundamental math and physics concepts
- Physiology of human vestibular system, which senses velocities and accelerations
- Motion simulation in VWG
- Case study: vection

Motion in a One-Dimensional World



- Velocity, v, is defined as $$v = \frac{dy(t)}{dt}$$
- The position can also be found at any time, t

$$y(t) = y(0) + \int_0^t v(s)ds,$$

Acceleration in a One-Dimensional World

- Change in velocity, i.e., acceleration, a, is defined as

$$a = \frac{dv(t)}{dt}$$

- In general, acceleration can also vary over time, hence

$$v(t) = v(0) + \int_0^t a(s)ds$$

- Change in acceleration is called jerk, and the naming continues

Motion in a 3D World: A Moving Point

- Consider motion of a point (x,y,z) in 3D, then the coordinate values would be a function of time:

$$(x(t), y(t), z(t))$$

- Velocity and acceleration can be represented with respect to the three axis

$$(v_x, v_y, v_z) \qquad (a_x, a_y, a_z)$$

- The magnitude of velocity can be calculated as

$$\sqrt{v_x^2 + v_y^2 + v_z^2}$$

Rigid-Body Motion in 3D

- When a rigid-body moves in 3D, all its points move together
- Suppose rotation is prohibited. What does the velocity and acceleration of a single point of the rigid-body tell about the velocity and acceleration of the rest of the rigid-body points?
  - What if rotation is included?

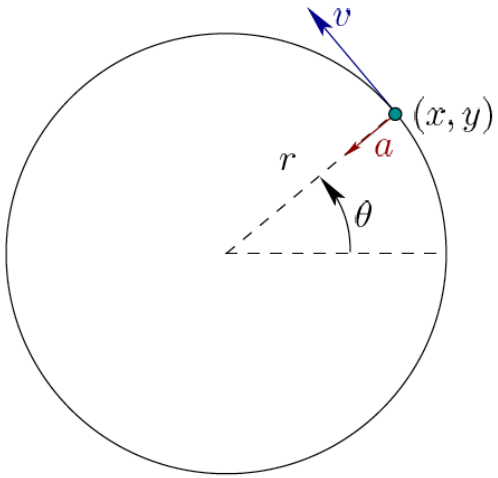
Angular Velocity

- Angular velocity can be defined as $\omega = \dfrac{d\theta(t)}{dt}$
- When angular velocity is constant we have

$$v_x = -r\omega \sin \omega t \text{ and } v_y = r\omega \cos \omega t.$$

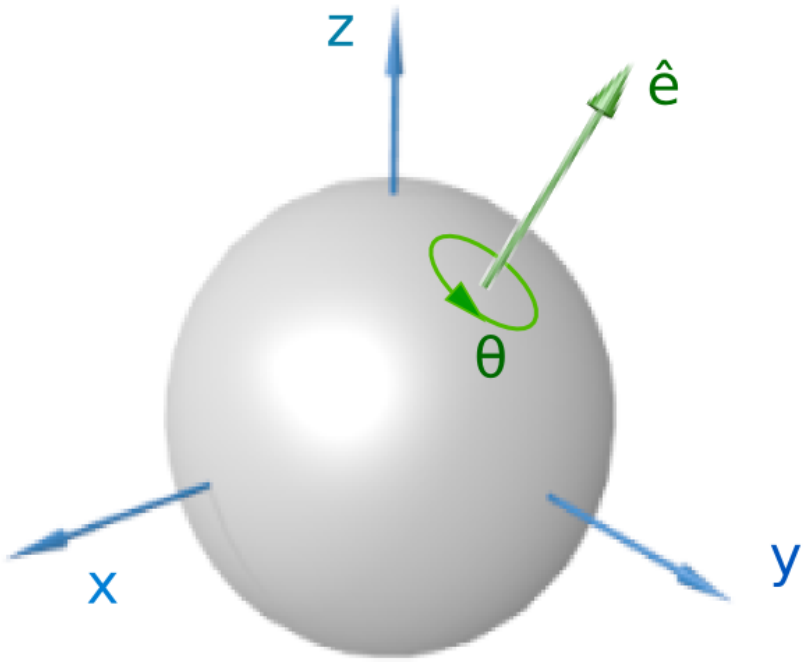
$$a_x = -r\omega^2 \cos \omega t \text{ and } a_y = -r\omega^2 \sin \omega t$$

- In this case, velocity would be tangential to the circle and the acceleration points to the center



Euler’s Rotation Theorem

- Euler's rotation theorem states that, in a three-dimensional space, any displacement of a rigid body such that a point on the rigid body remains fixed, is equivalent to a single rotation about some axis that runs through the fixed point.
- Corollary: Composition of two rotations is also a rotation.



3D Angular Velocity and Acceleration

- Consider a rigid-body rotating around a unit vector v
- This rotation can be decomposed into rotation around x, y, and z axis, resulting in the following 3D angular velocity

$$(\omega_x, \omega_y, \omega_z)$$

- Angular velocity may also change over time, resulting in angular acceleration of pitch, yaw, and roll angles

$$(\alpha_x, \alpha_y, \alpha_z)$$

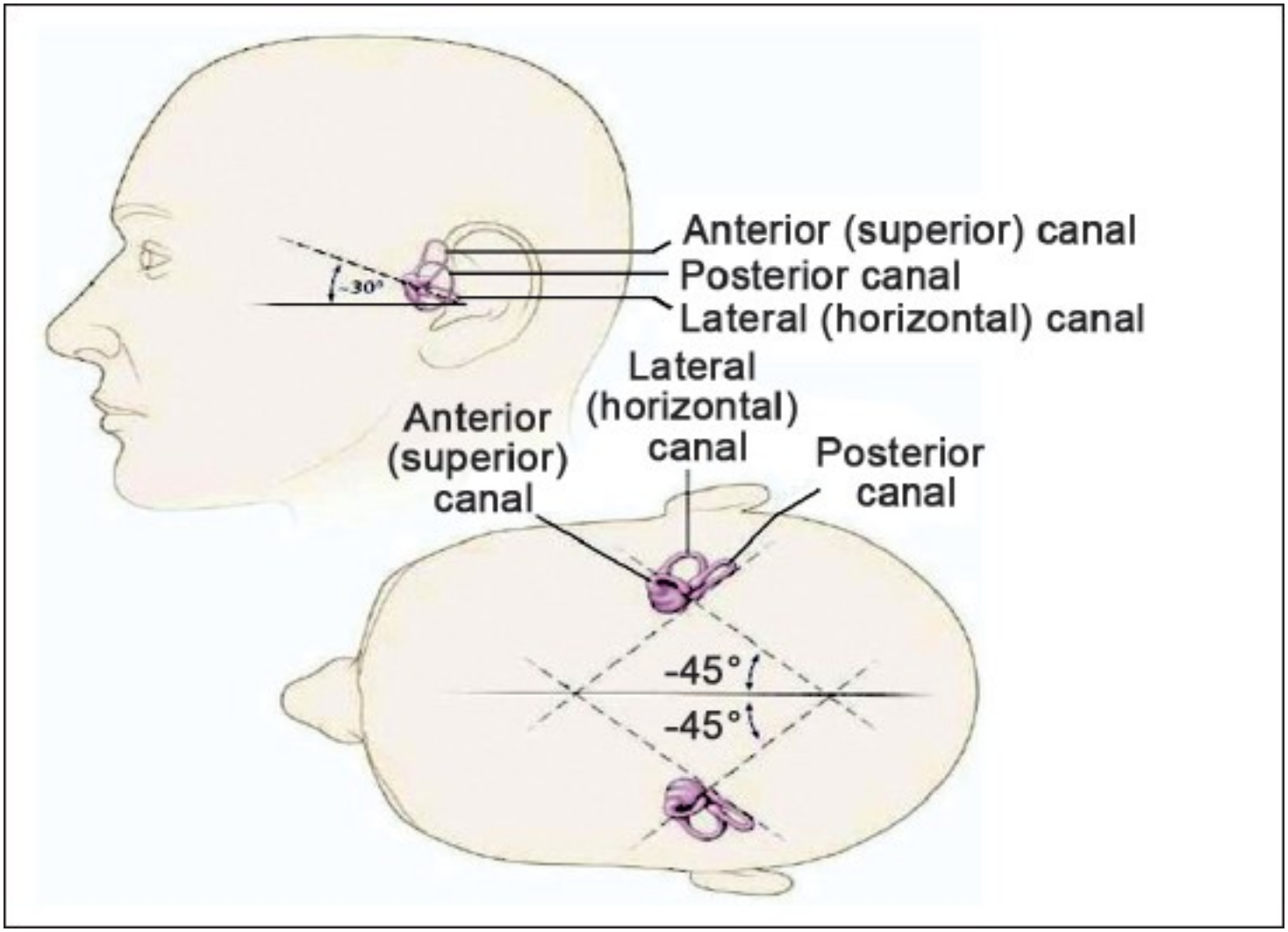
Do not mix linear velocity/acceleration with angular velocity/acceleration!

Vestibular System

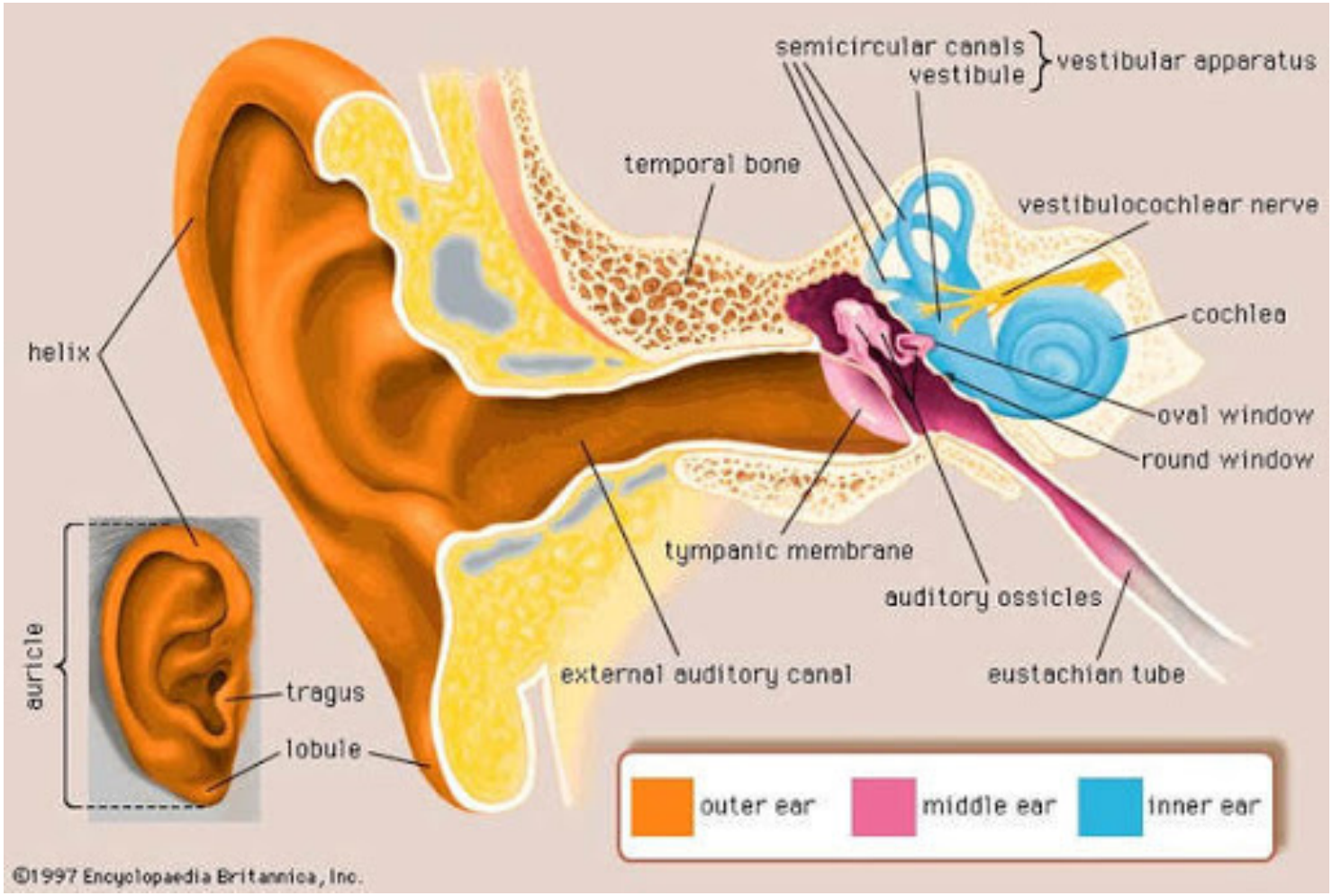
- Vestibular/balance sense: information sent to brain about how head is oriented or moving
- Vestibular system: vestibular organs along with their neural pathways are referred to as vestibular system
- Vestibular organs measure both linear and angular acceleration of the head

- VR systems and vestibular organs
  - Current VR displays do not stimulate vestibular organs
  - Research: use electrical signals or low frequency vibrations to stimulate them; multiple years until commercial use!

Physiology: Side and Top Views

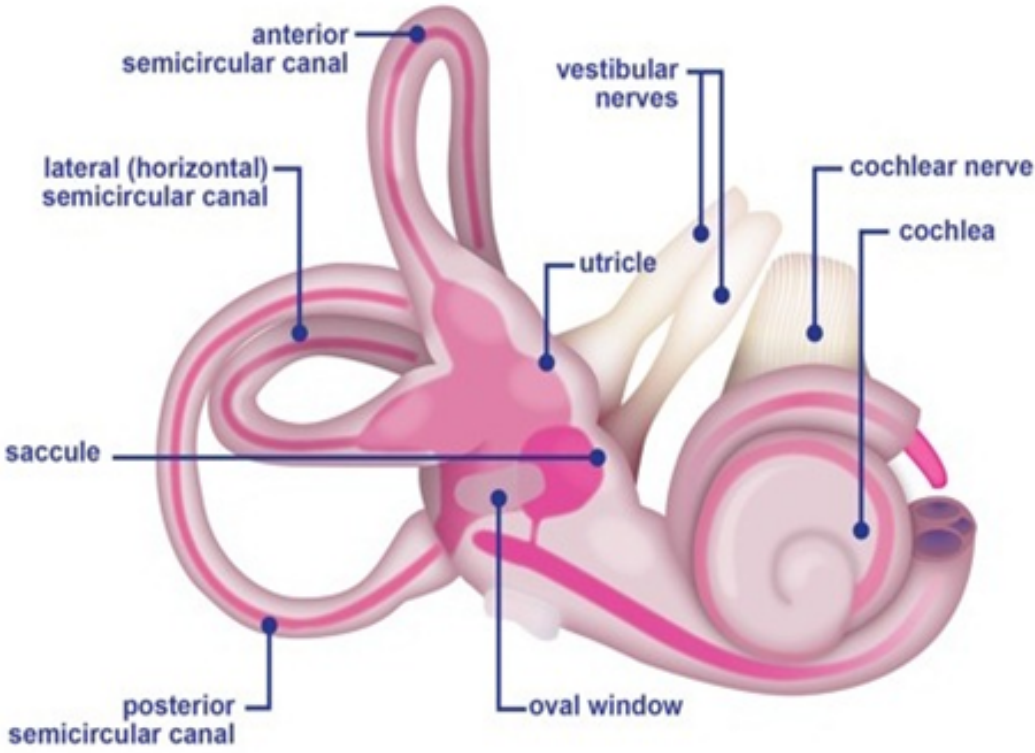


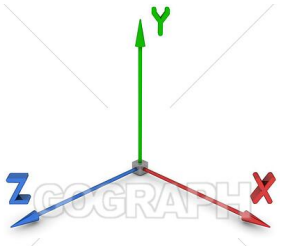
Physiology


semicircular canals
vestibule
vestibular apparatus
temporal bone
vestibulocochlear nerve
cochlea
helix
oval window
round window
tympanic membrane
auditory ossicles
auricle
tragus
lobule
external auditory canal
eustachian tube
outer ear
middle ear
inner ear
©1997 Encyclopaedia Britannica, Inc.

Vestibular Organ

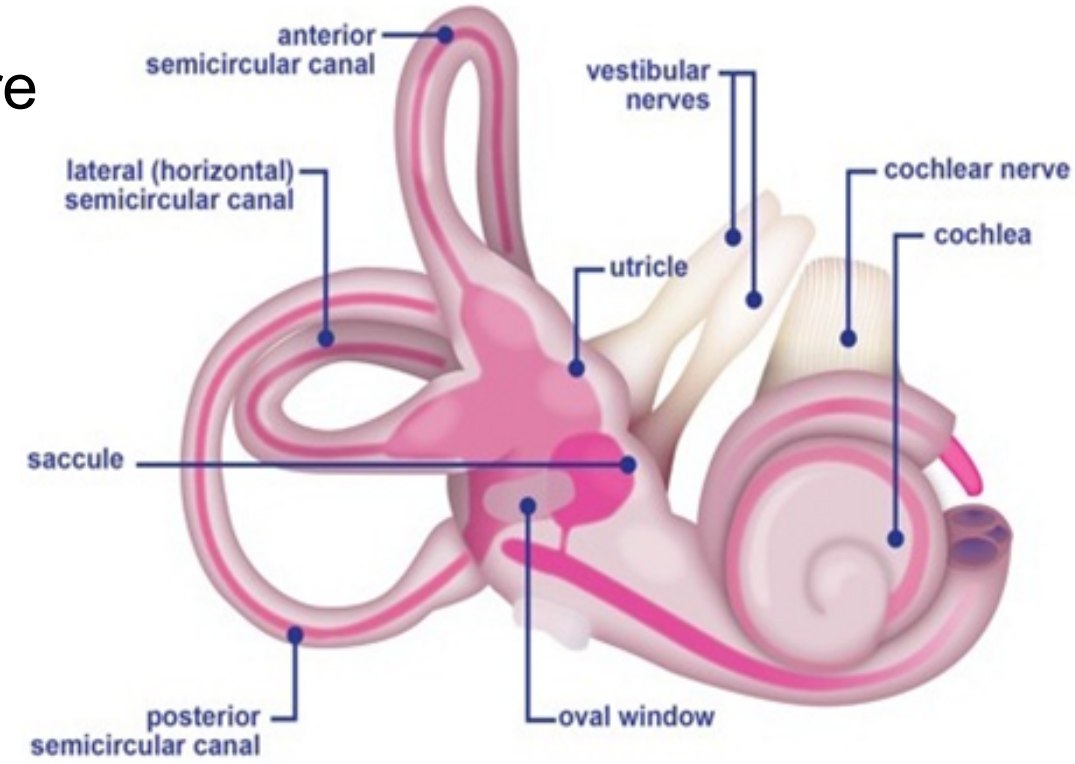
- Cochlea handles hearing
- Utricle and saccule measure linear acceleration
  - Part of vestibule
  - Together form the otolith system
  - When head is not tilted, the sensing surface of utricle mostly lies in the horizontal plane, saccule would lie vertical
  - Utricle senses linear acceleration components a_x and a_z, saccule senses a_y and a_z



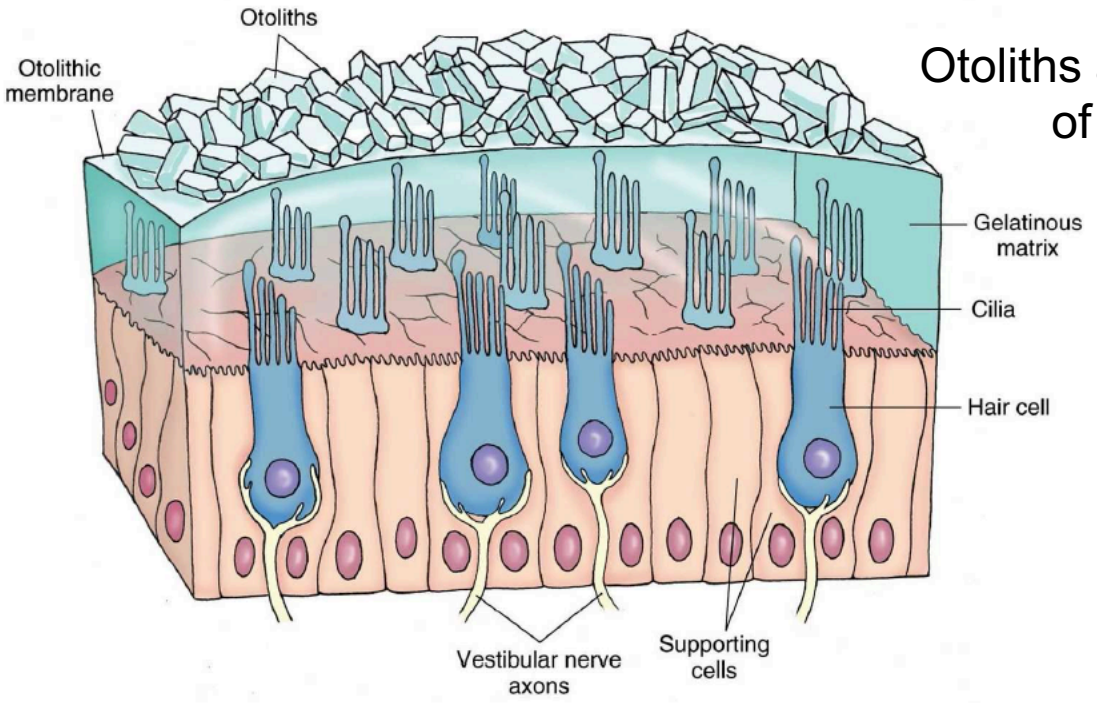


Vestibular Organ

- Semicircular canals measure angular acceleration
  - 0.2-0.3 mm canal diameter
  - Each circular arc has 2-3 mm diameter
  - The three canals are perpendicular so they independently measure three components of angular acceleration
  - Not directly match to our x, y, z axis!

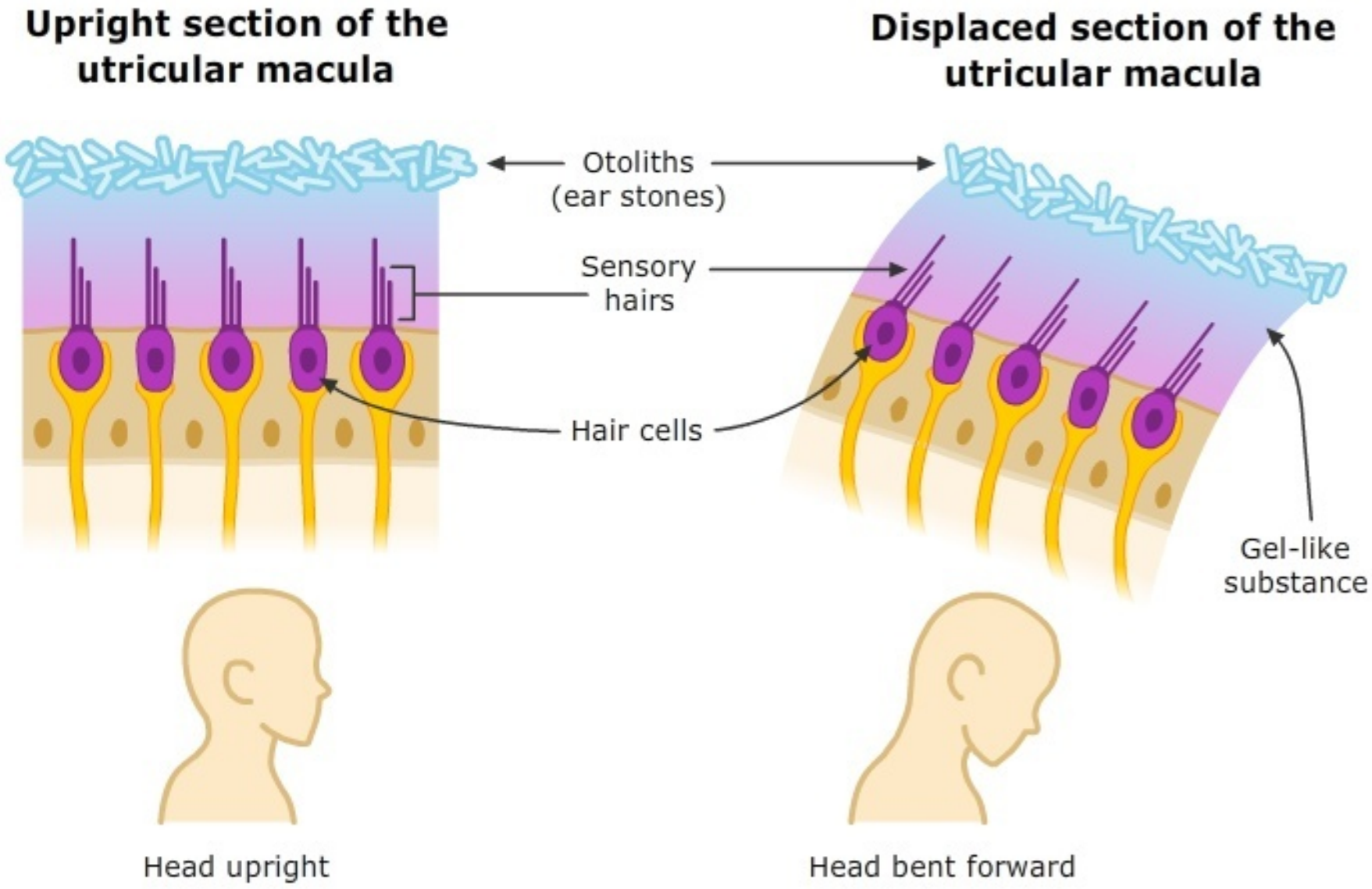


Sensing Linear Acceleration

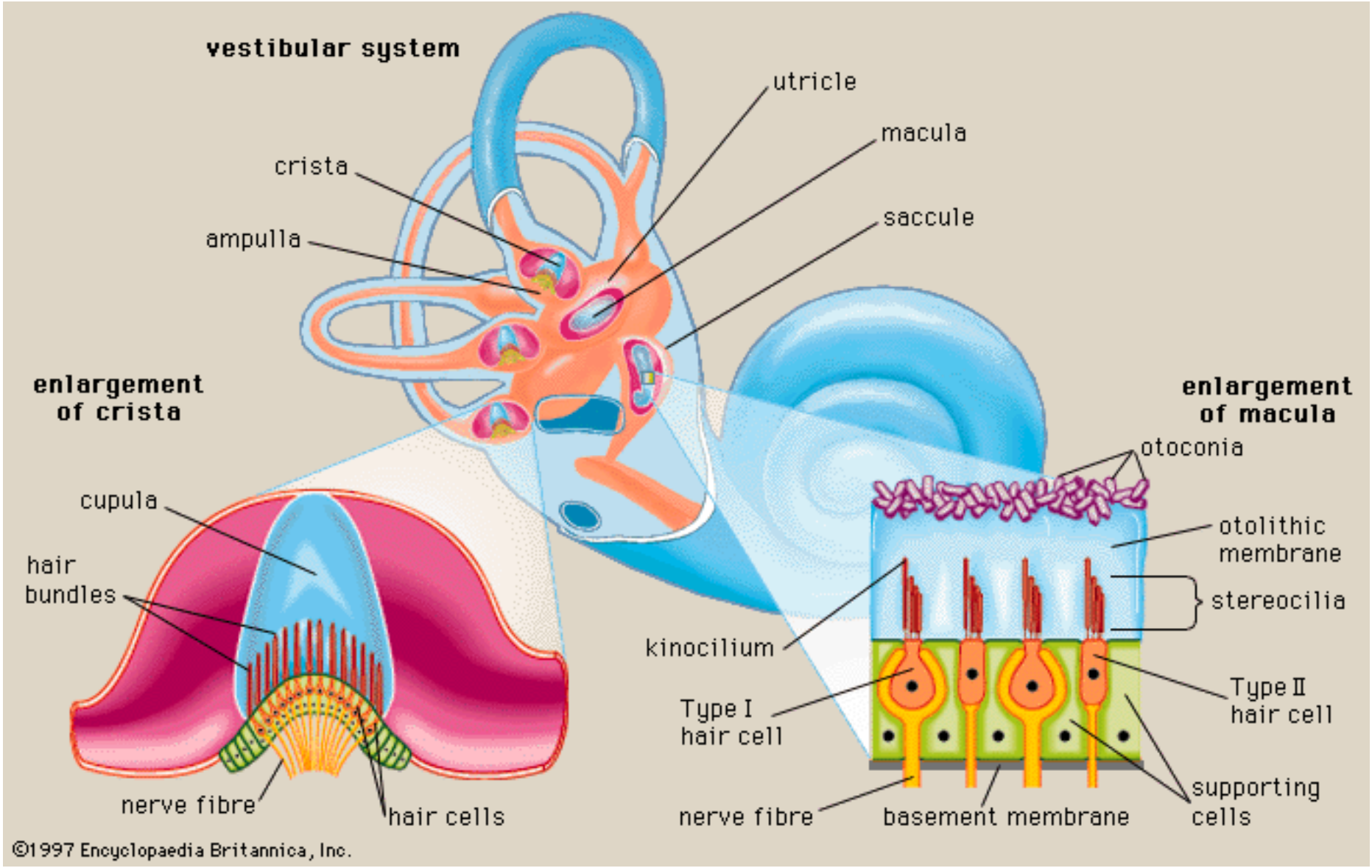


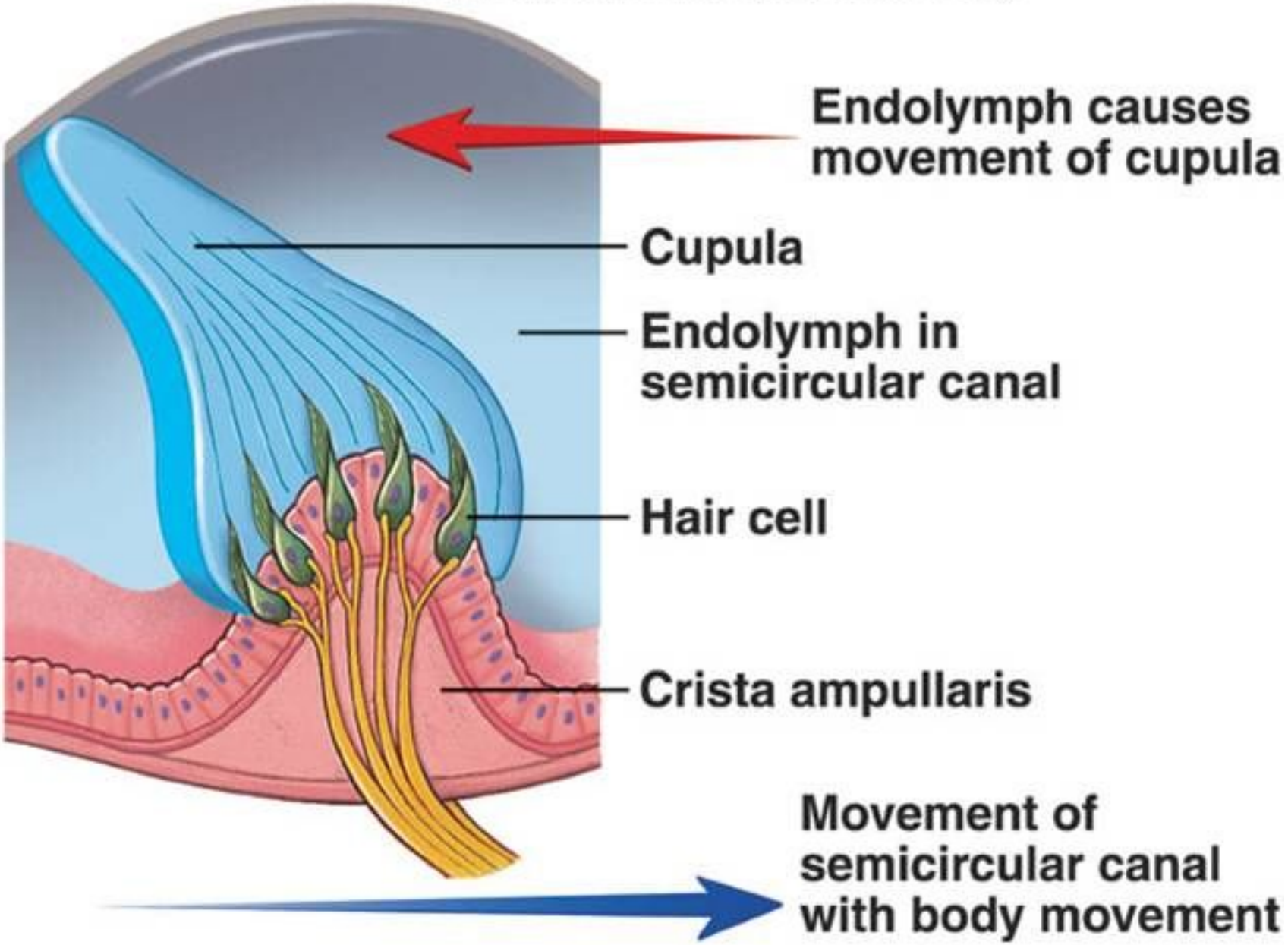
Otoliths are a thin layer of crystals!

- Depiction of an otolith organ (utricle or saccule)
- Mechanoreceptors, in the form of hair cells, convert acceleration into neural signals
- Each hair cell has cilia that are embedded in a gelatinous mass



Sensing Angular Acceleration





Impact of Vestibular Cues on Perception

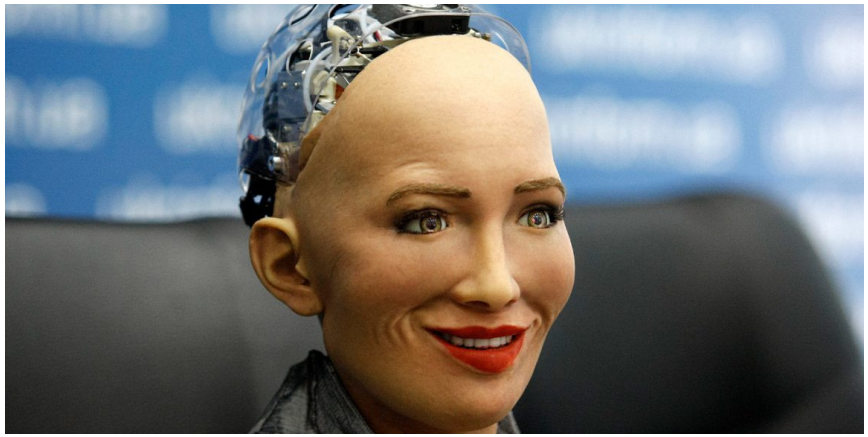
- Vestibular cues are generally weaker than other senses such as vision
- Often, vestibular cues work well as they accompany other cues
- Some people more susceptible to vestibular cue mismatch, which can make them experience vertigo
  - Vertigo: sense of being off-balance
  - Feeling the world around them is spinning
  - Common symptoms: nausea, vomiting, and sweating

Physics in the Virtual World

- We need to fool our brain into thinking that we live in a virtual world
- Motion should behave in a similar manner
- VWG need to contain a physics engine that governs the motion of objects in the virtual world
  - Forces acting on rigid-bodies, gravity, collision between physical bodies should be captured in a convincing manner
- Game engines such as Unity 3D and Unreal Engine were quickly adopted for VR, since they have physics implemented in them already
  - At the moment no widely adopted VR engines exist

Physics Engine Design

- Physics engine design may be simple or overwhelming
  - Depends on a variety of factors, e.g., will user interact with objects, will multiple users share the same virtual space? Will the matches zone remain fixed, or will the user need to be moved by locomotion?
- We need to keep the virtual world frequently updated so that the interactions between users and objects are well synchronized and renderers provide a low latency projection onto displays
- The goal may not always be to perfectly match reality (comfort vs reality)
  - Sometimes, perfectly matching reality makes things look creepy
  - Make some problems such as vection more pronounced



More realism might lead to increased creepiness!

Numerical Simulation of Rigid-Bodies

- Virtual world contains many rigid-bodes
- Job of the VWG is to determine the position and orientation of each and every object for any given time
  - This snapshot of the virtual world is also referred to as world state
  - With known state, we can use the chain of transformation to place objects onto the user display
- Typically, a rigid-body has six degrees of freedom (DoF)
- Sometime, DoF are lost due to constraints
  - Example: how many DoF does a ball that roll on the ground have?

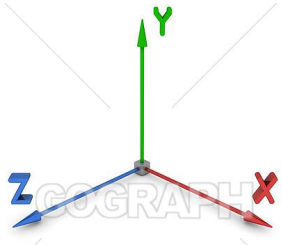
Rigid-Body DoF

- For many models, rigid-bodies are attached together in a way that allows relative motions
  - Referred to as multibody kinematics
- How many DoF would be needed to model a car alone as well as when wheels can also rotate?

Partial Differential Equations

Extra!
Not in Exam

- To fully model human body 244 DoF may be needed
  - High computational complexity and difficult implementation
- Rigid-bodies may not be sufficient to model the world
  - Example: waves rippling realistically across a lake
  - Can model wave location as a continuous function: y = f(x,z)
  - Motions described as partial differential equations



- Let x = (x_1, …, x_n) denote an n-dimensional state vector
  - x_i corresponds to a position or orientation for a rigid-body
  - Let the time derivative or velocity for each parameter be $\dot{x}_i = \frac{dx_i}{dt}$
  - To obtain the state at time t, the velocities need to be integrated over time

$$x_i(t) = x_i(0) + \int_0^t \dot{x}_i(s)ds$$

How to Derive the Integral Values?

Extra!
Not in Exam

- Time is discretized into steps
  - Delta-t is the step size or sampling rate
  - For example, sampling rate of 1 msec means the physics engine updates all parameters (positions and orientations) every 1 msec
  - Think of it as the frame rate for the physics engine
  - Physics engine frame rate is much higher than renderer and display frame rates to have a much higher accuracy
- Euler approximation for state variable x_i:

$$x_i((k+1)\Delta t) \approx x_i(0) + \sum_{j=1}^{k} \dot{x}_i(j\Delta t)\Delta t$$

$$x_i[k+1] \approx x_i[k] + \dot{x}_i(k\Delta t)\Delta t$$

Runge-Kutta Integration Method

- Yields a much better performance that Euler method
  - Although has a higher computational cost

Extra!
Not in Exam

- Let f(t,x) = $\dot{x}_i = \frac{dx_i}{dt}$ then

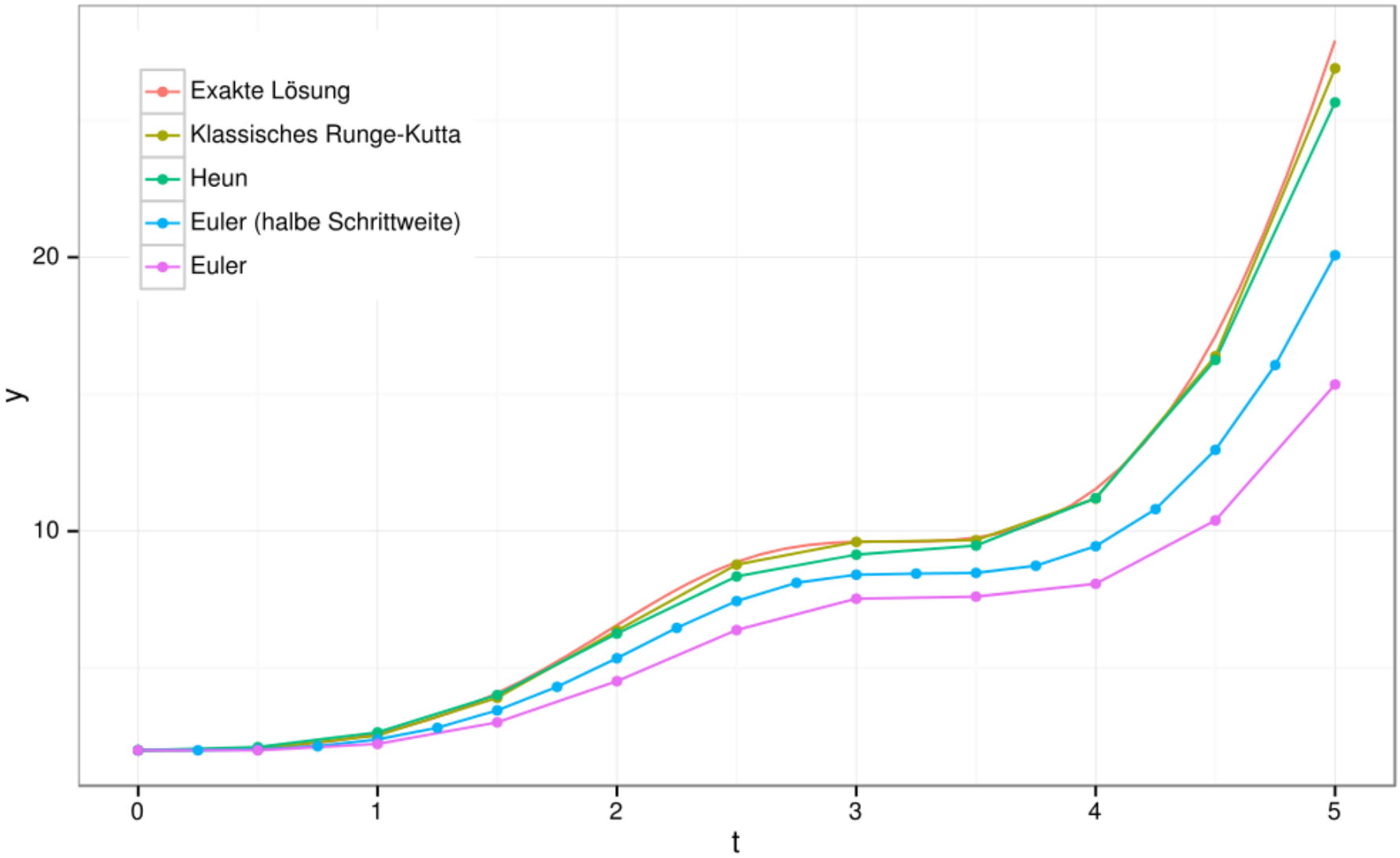
$$x_i[k+1] \approx x_i[k] + \frac{\Delta t}{6}(w_1 + 2w_2 + 2w_3 + w_4)$$

$$w_1 = f(x_i(k\Delta t))$$

$$w_2 = f(x_i(k\Delta t + \tfrac{1}{2}\Delta t) + \tfrac{1}{2}\Delta t\, w_1)$$

$$w_3 = f(x_i(k\Delta t + \tfrac{1}{2}\Delta t) + \tfrac{1}{2}\Delta t\, w_2)$$

$$w_4 = f(x_i(k\Delta t + \Delta t) + \Delta t\, w_3)$$



Comparison of the Runge-Kutta methods for the differential equation y'=sin(t)^2*y (orange is the exact solution)

Collision Detection

- Key challenge in physics engine: handling collisions
- Collision detection algorithms are plentiful because of widespread use in computer graphics and video games
- Housdorff distance between two models A and B, is the Euclidean distance between the closest pair of points, taking one from A and another from B.
- Algorithm
  - Consider the distance
  - If A intersection B is non-empty, then Housdorff distance is zero and there is collision
  - If distance is greater than zero, then there is no collision
  - If the distance is a large number, then A and B are going to be collision free in the near future as well

Mismatched Obstacles in VR

- Collision detection in VR has applications applied to mismatches between real and virtual worlds
- 1) Real Obstacles in the real world
  - Potentially dangerous (hot cup not present in the VR world)
  - Solution: boundary of the matched zone rendered as user gets close
  - Solution: real objects that block user can be rendered on display
- 2) Virtual Obstacles Only
  - A wall in the virtual world not present in the real world (frustrating)
  - Mismatch can be uncomfortable for the user
- 3) Real and virtual obstacles
  - If obstacles match, the effect can be very powerful
  - For example, imagine standing on a slightly raised platform while the virtual world shows you standing on a building rooftop

Mismatched Motion and Vection

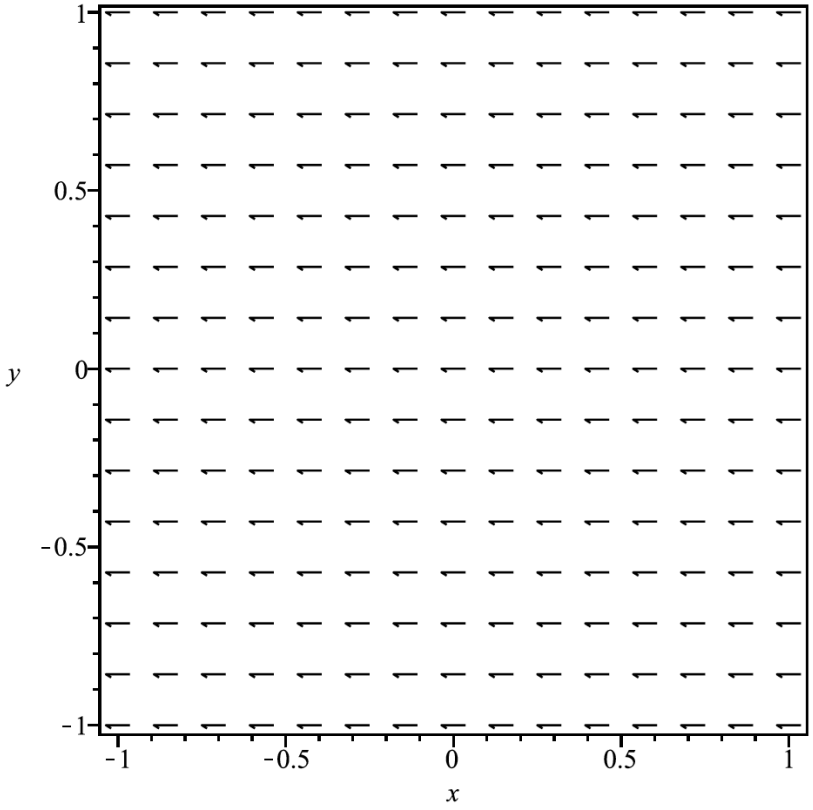
- Recall vection: illusion of self-motion
  - Commonly induced in VR by moving the user's viewpoint while there is no corresponding motion in the real world
  - Vection is a prime example of mismatched cues
- If a headset is better in terms of spatial resolution, frame rate, tracking accuracy, field-of-view, and latency, then the potential is higher for making people sick through vection and other mismatched cues
  - If the headset more accurately mimics reality, then the sensory cues are stronger, and our perceptual systems become more confident about mismatched cues
  - Mismatches may lead to fatigue, as the brain works hard to resolve conflicts. In worst case, VR sickness can occur.

Optical/Optic Flow

- Optic flow: pattern of apparent motion of objects in a visual scene
- Motion of visual features on the retina/display is modeled as a vector field
- Assigns a velocity vector at every point along a surface
- Consider xy plane, then

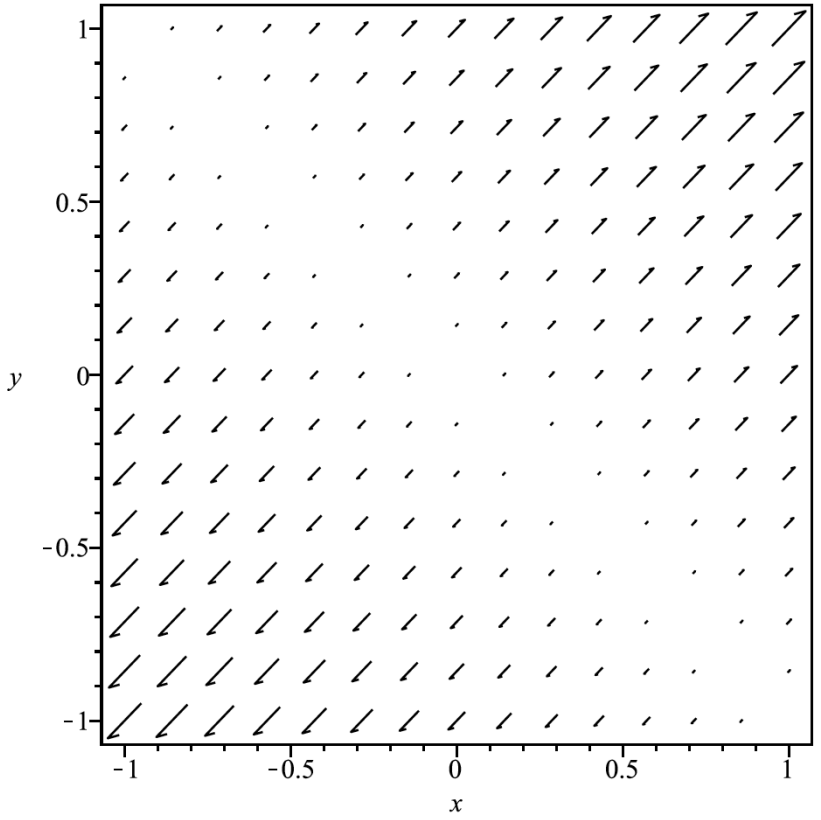
$$(v_x, v_y) = \left(\frac{dx}{dt}, \frac{dy}{dt}\right)$$

Vector Field



$$(x, y) \mapsto (-1, 0)$$

Constant vector field



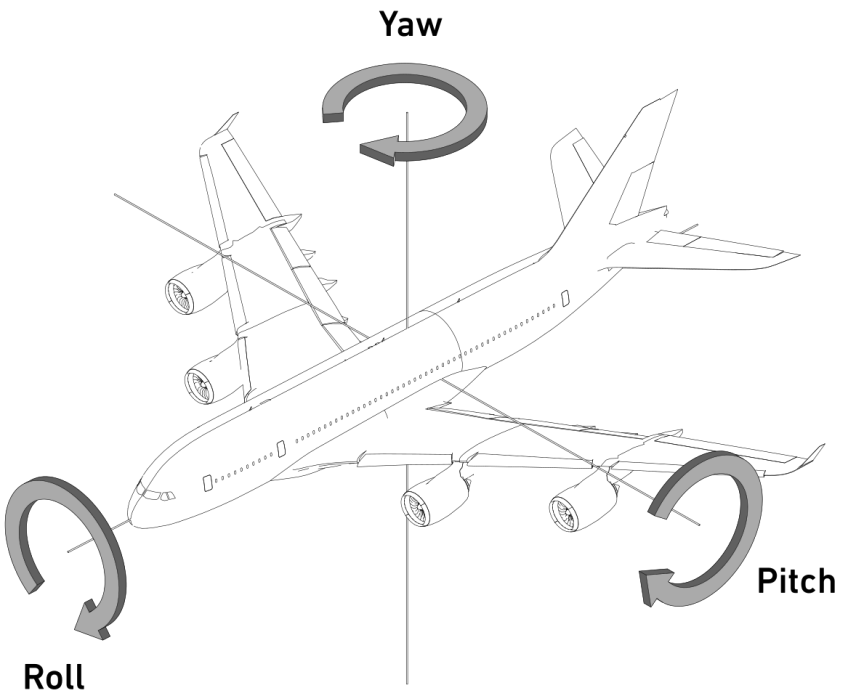
$$(x, y) \mapsto (x + y, x + y)$$

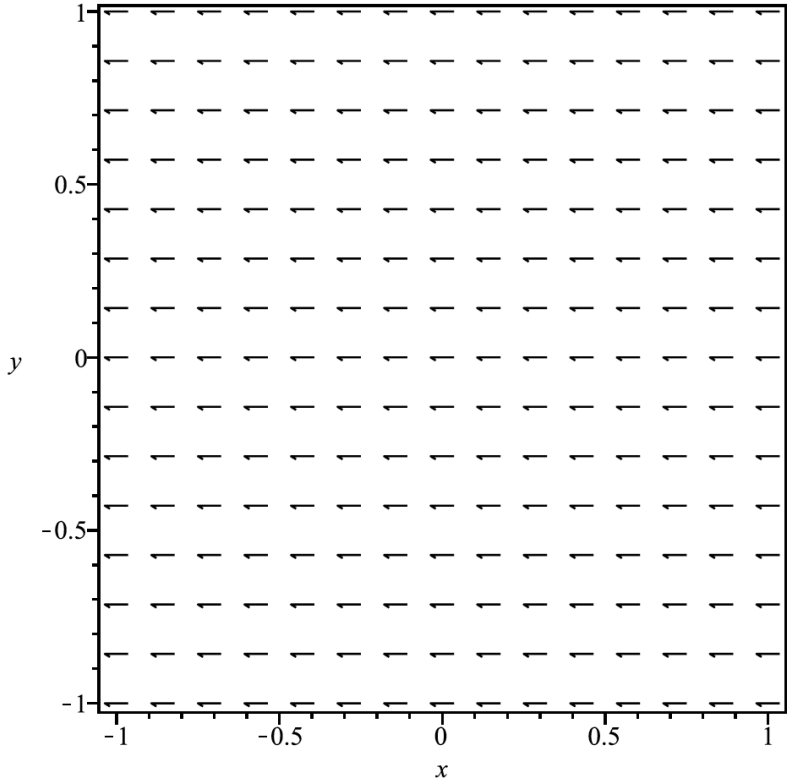
Non-Constant vector field

Types of Vection

- Vection can be caused by any combination of angular and linear velocities of the viewpoint in the virtual world
- Viewpoint velocities are commonly divided into three linear components:
  - … and three angular components
- Types of vection:
  - Yaw, Pitch vection: sometimes called circular vection
  - Roll vection
  - Lateral, vertical, forward/backward vection: linear vection

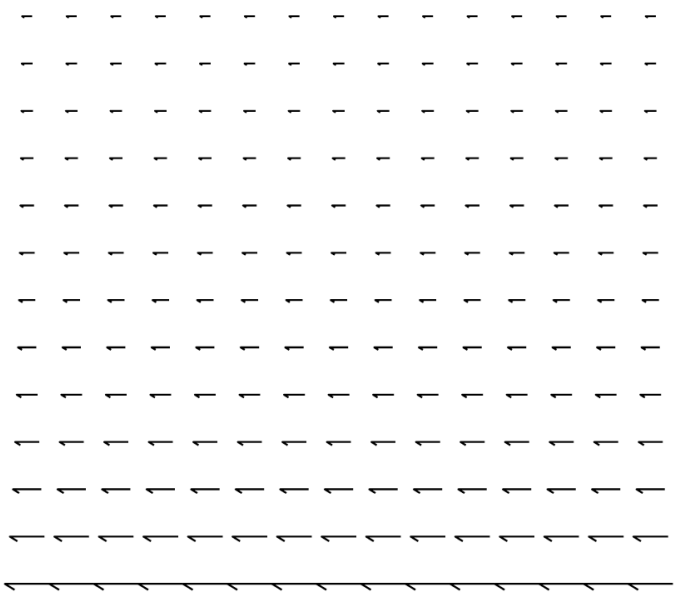
Yaw Vection





- If the viewpoint is rotated counterclockwise about the y axis, then all visual features move from right to left at the same velocity.
- Essentially, the virtual world is rotating clockwise, however, self-motion is perceived in the opposite direction. The user feels as if she riding a merry-go-round.

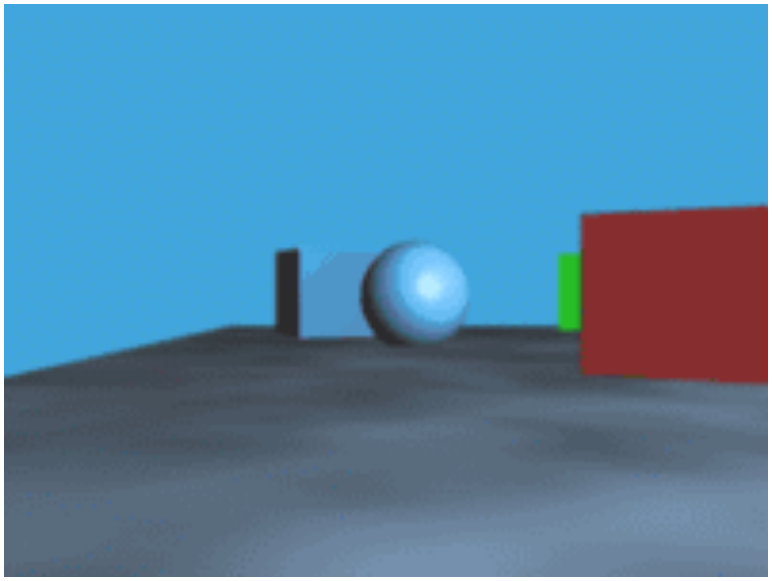
Lateral Vection



- Viewpoint is translated to right. Features move horizontally to the left.
- Distinction from yaw vection: features that correspond to closer objects move more quickly than from distant ones
  - **Parallax effect**
- Example above, assume lower in the depth field is closer

Parallax

- Parallax is a displacement in the apparent position of an object viewed along two different lines of sight. Due to foreshortening, nearby objects show a larger parallax than farther objects when observed from different positions, so parallax can be used to determine distances.



Vection and Mismatches

- All forms of vection cause perceived velocity
- If there is a change in velocity, it may cause perceived acceleration

- Type of vection mismatches
  - Mismatch with vestibular sense organ: in case of linear or angular perceived acceleration, there would be a vestibular mismatch
  - Conflict with sensory motor system: lack of movement by body

Factors that Affect Sensitivity to Vection

- Percentage of field-of view: increased FoV, strengthens vection cues
- Exposure time: higher exposure time, strengthens vection
- Spatial resolution: increased spatial resolution or images with more features, optical flow becomes stronger
- Prior training or adaptation