Lecture 13-14:
Mid-trimester Recap, Animation started

CGRA 354 : Computer Graphics Programming

Instructor: Alex Doronin
Cotton Level 3, Office 330
alex.doronin@vuw.ac.nz

With slides from: Holly Rushmeier, Yale; Steve Marschner, Cornell; Taehyun Rhee, CMIC; Zohar Levi, ECS;

This week

Assignment 1 and mid-trimester test: Grades released Wednesday (please review), Assignment 2 due

Today: Recap and Assignment 3

Friday:

- ***Assignment 3 tutorial***
- ***Assignment 2/3 helpdesk***

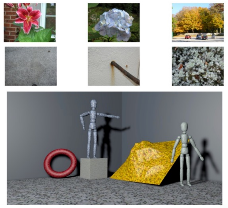
Recap: What is Computer Graphics?

Creating new images using computers

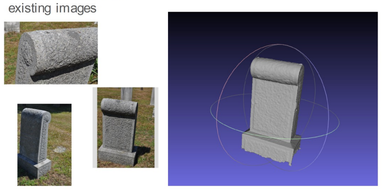
May be created from:

- Existing images
- 3D models
- User strokes
- High dimensional data

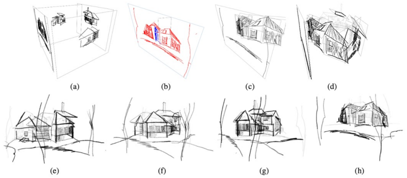
From existing images + 3D models



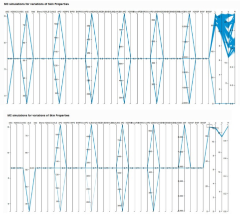
New 3D models from existing images



From user strokes

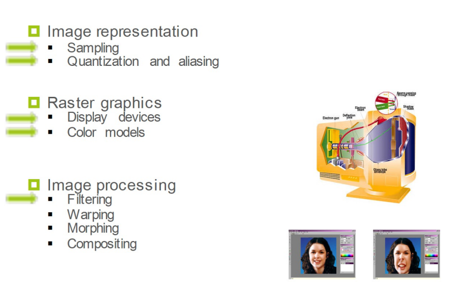


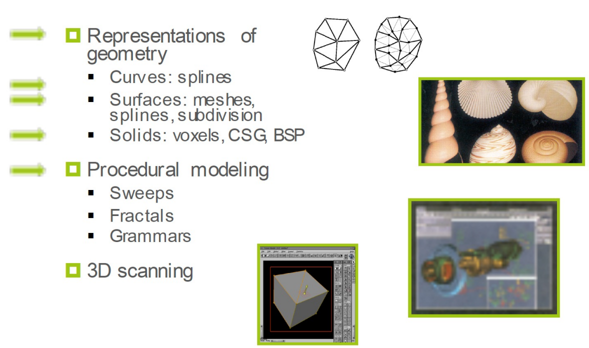
From high dimensional data

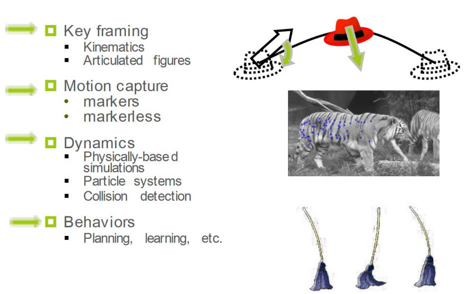


Recap: Areas in Computer Graphics

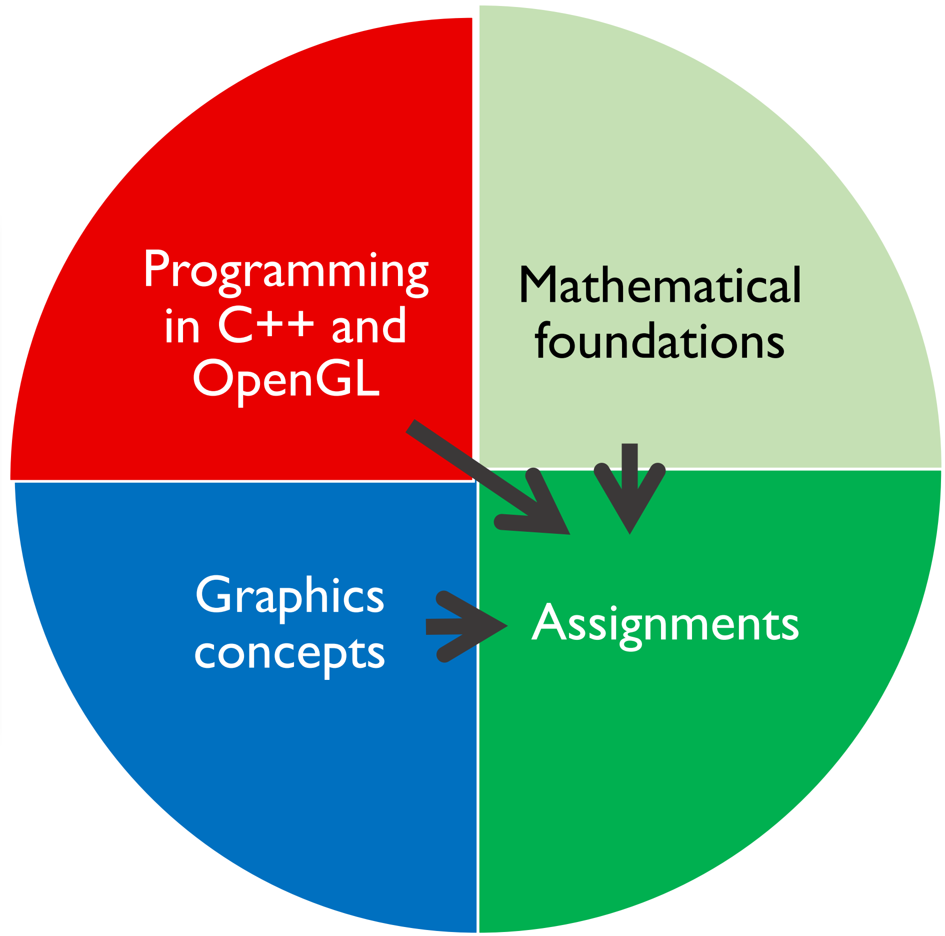
- **Imaging** = representing 2D images
- **Modeling** = representing 2D/3D objects
- **Rendering** = 2D images from 2D/3D models
- **Animation** = simulating changes over time







Recap: Outline

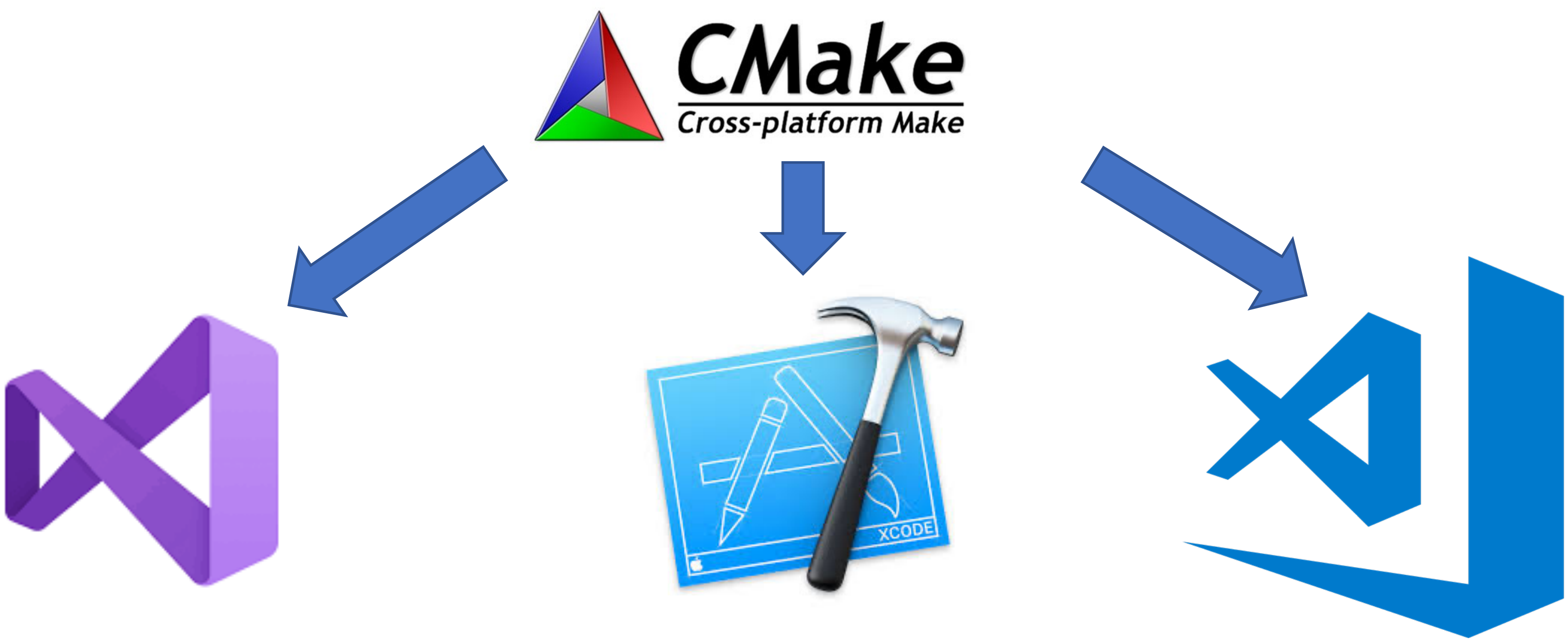


What is OpenGL?

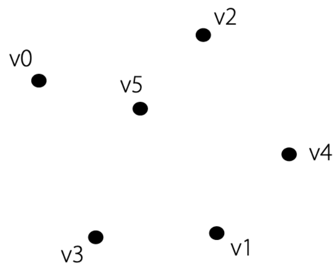
- Specialized hardware — Graphics Processing Unit (GPU)
- API to interact with the hardware
- Cross-platform
- Additional libraries:
    - GLEW: extension
    - GLFW: basic window with OpenGL rendering context
    - Etc.
- Start with the classic OpenGL?

Recap: Building Frameworks

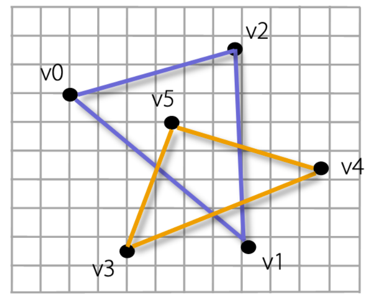
- *Integrated development environments (IDEs) on Windows, Mac, Linux and CMake tools*



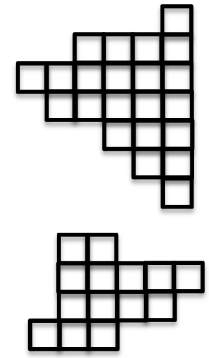
Recap: Computer Graphics pipeline entities



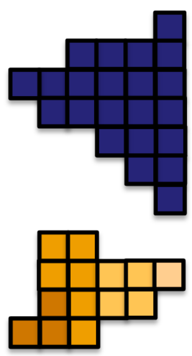
Vertices



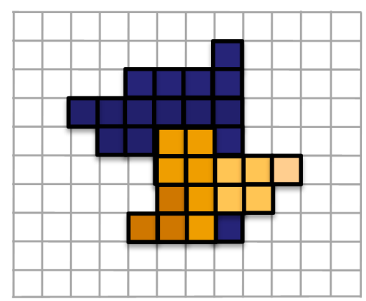
Primitives



Fragments



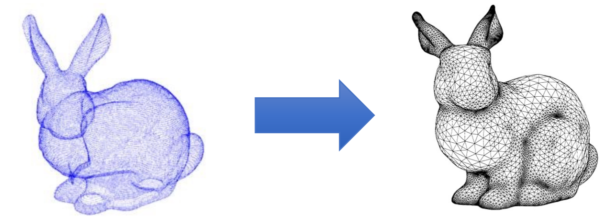
Fragments (shaded)



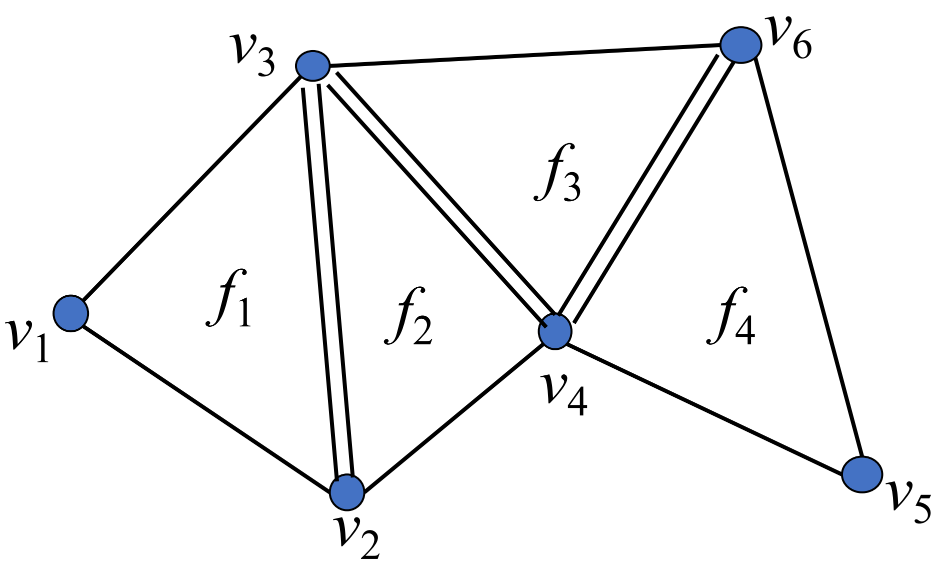
Pixels

Recap: Geometry and Polygon Mesh

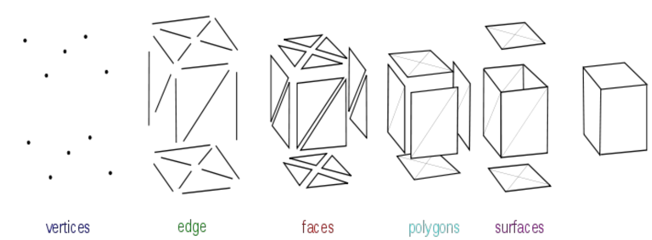
- Face list
  - Lists of coordinates
- Polygons are unrelated



Point Cloud Polygon Mesh



face	vertices (ccw)
f_1	(v_1, v_2, v_3)
f_2	(v_2, v_4, v_3)
...	...



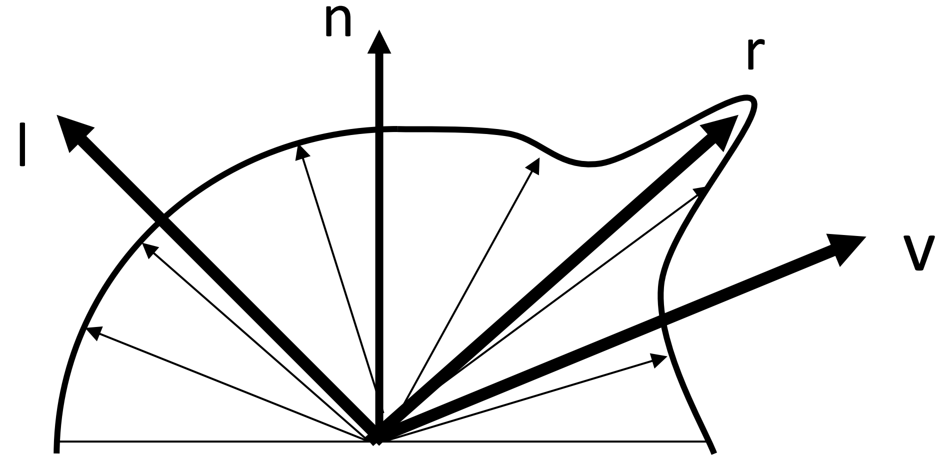
- What are the nearest neighbor of a vertex ?
- What are the adjacent triangles of a vertex ?

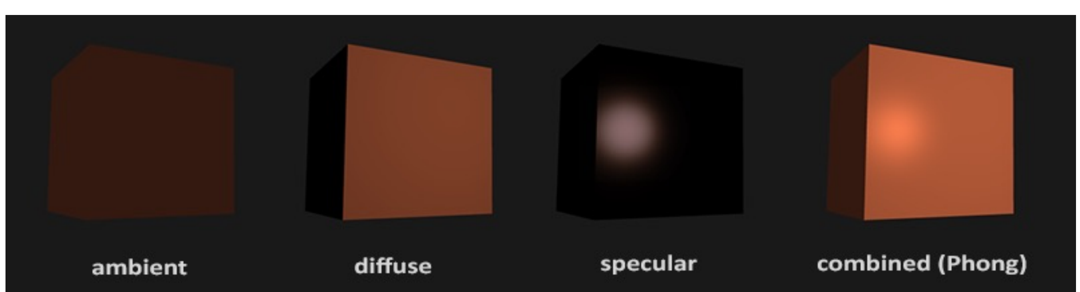
Phong Shading Model in OpenGL

- Phong illumination model is combination of
  - Ambient i_{amb}+ Diffuse i_{diff} + Specular terms i_{sepc}
  - Developed by Bui Tuong Phong at Univ. Utah 1973

$$\mathbf{I} = k_a i_a + k_d i_d (\mathbf{n} \bullet \mathbf{l}) + k_s i_s (\mathbf{r} \bullet \mathbf{v})^{m_{shi}}$$

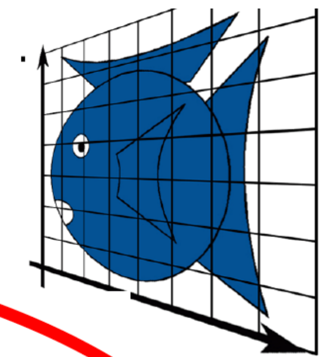
  - k_a k_d k_s are material properties having RGB components

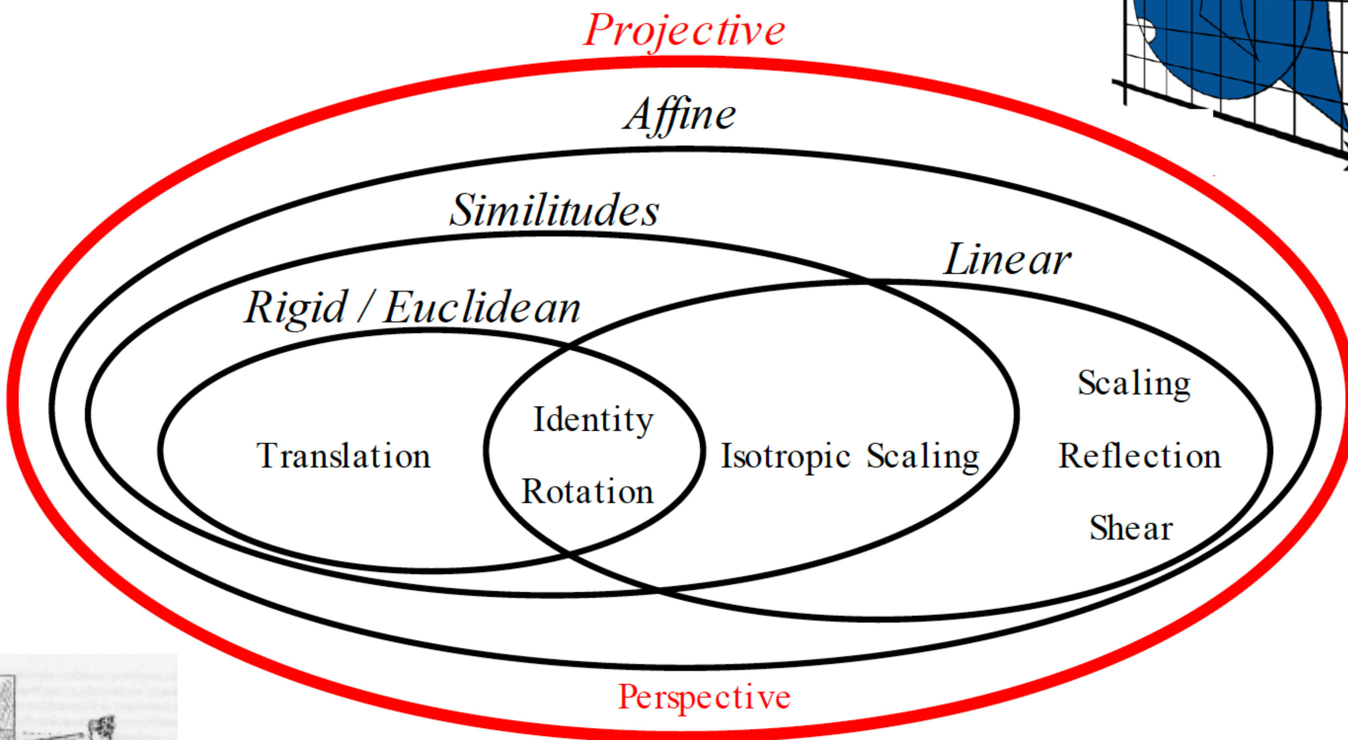


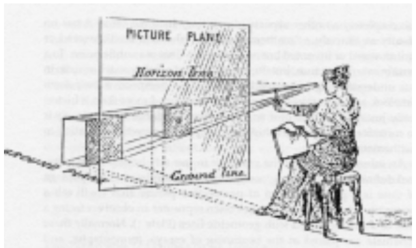


Transformations

- preserves lines



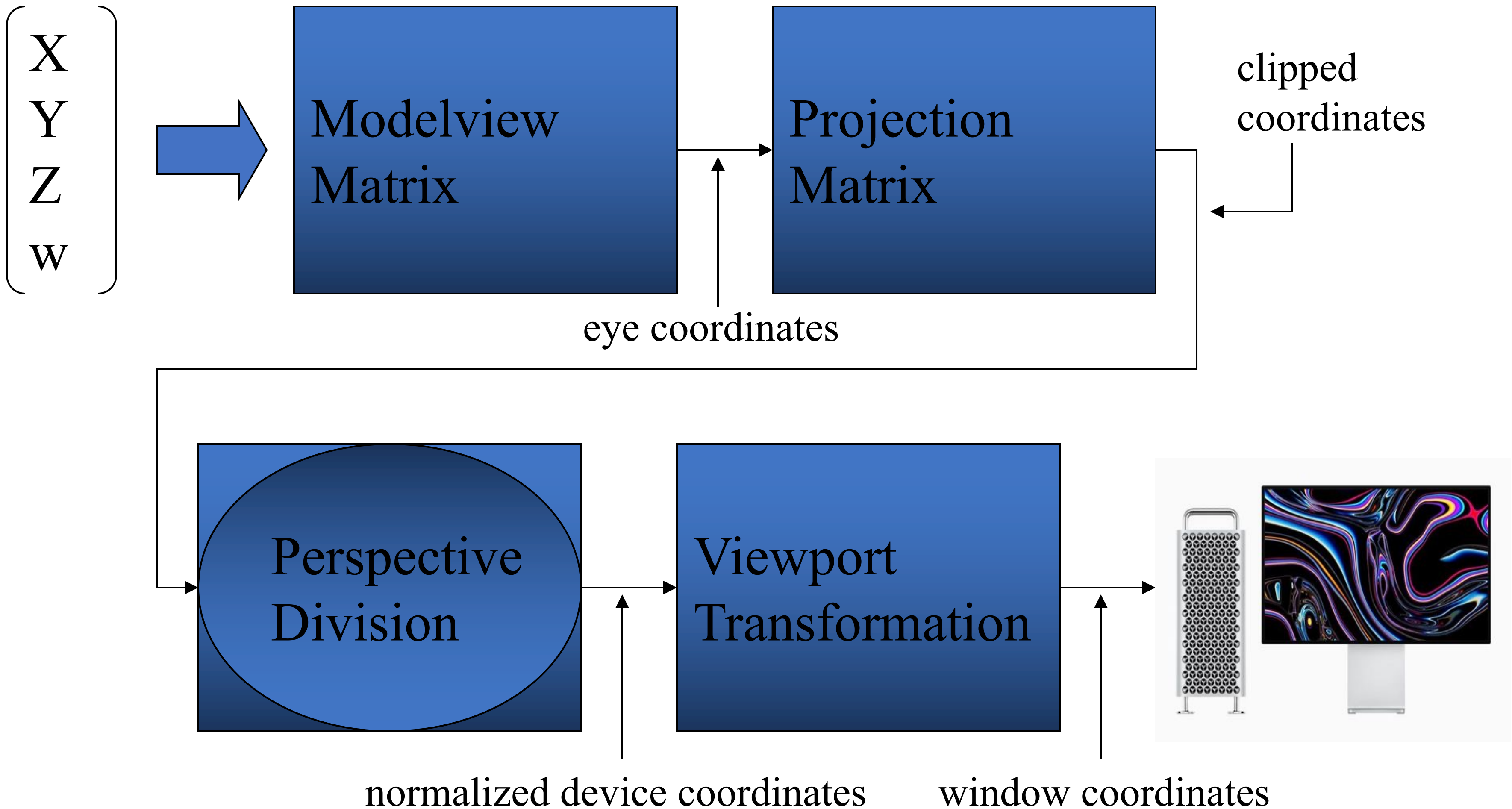




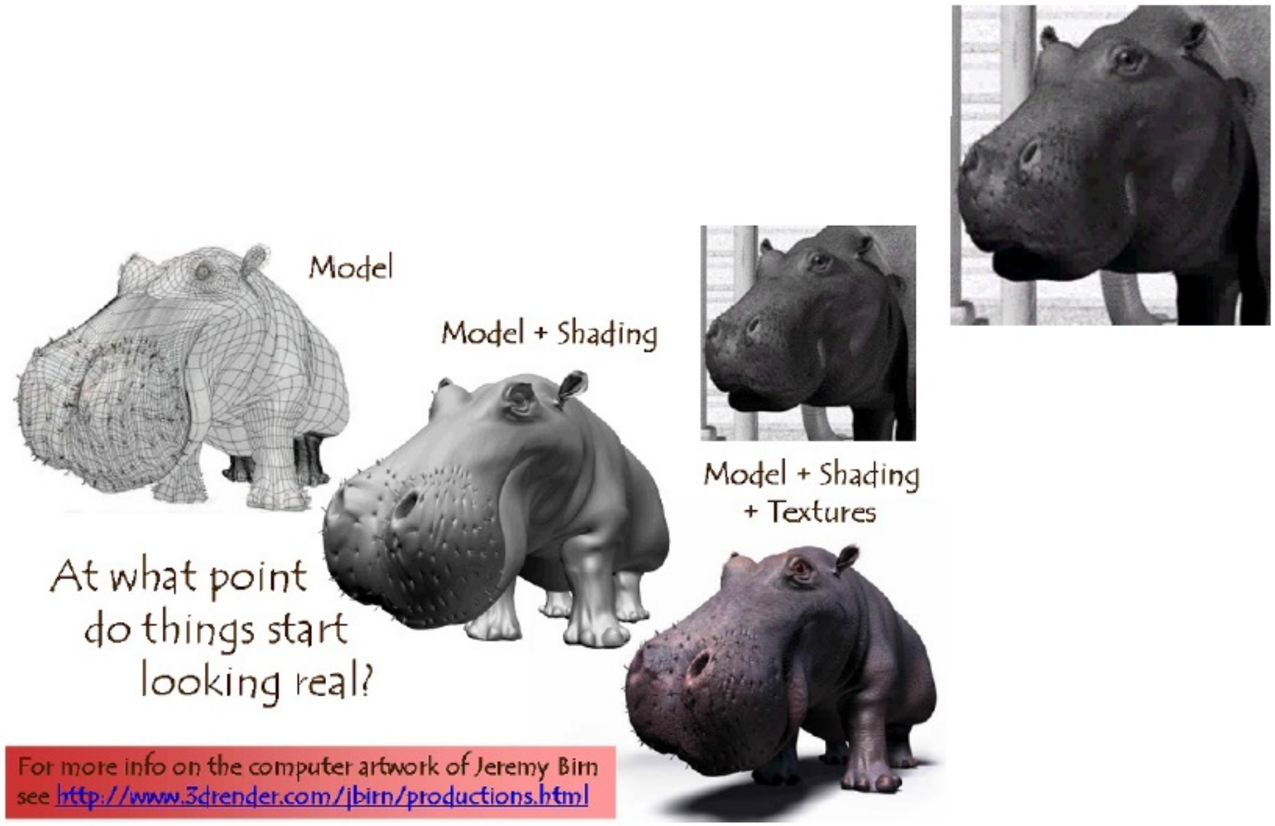
Instancing

- transformations allow you to define an object at one location and then place multiple instances in your scene

Stages of Vertex Transformations



Surface Details

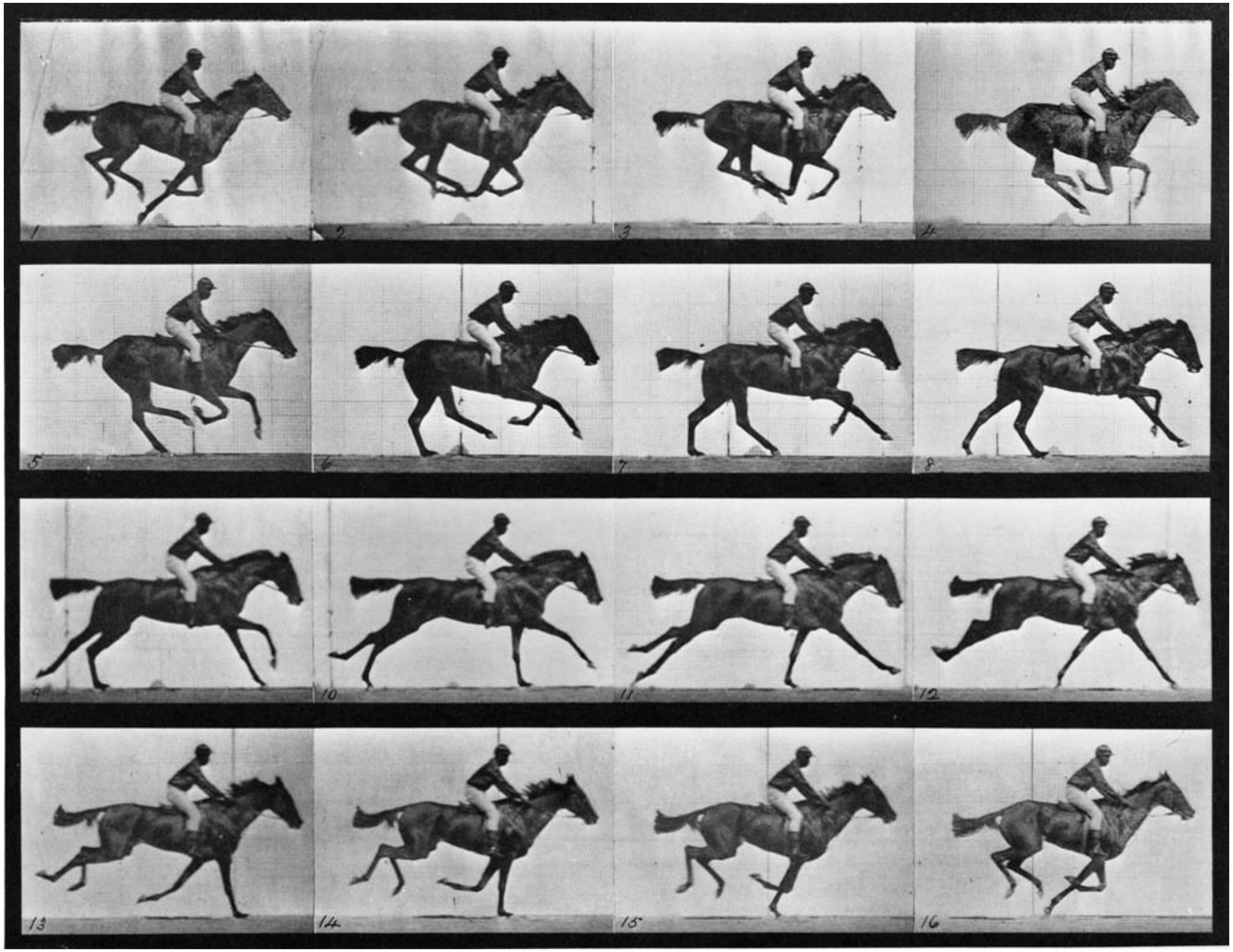


Perception of Motion

- How is it even possible to make it look things look like they are moving, or even alive with the computer?

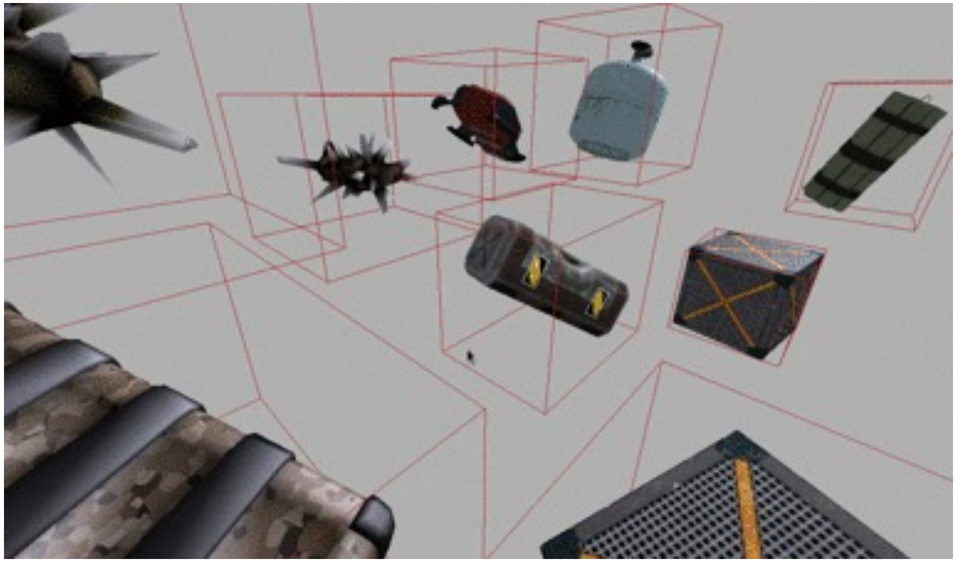
Image credit: https://giphy.com/gifs/animated-loop-walking-XGnWMiVXL87Xa

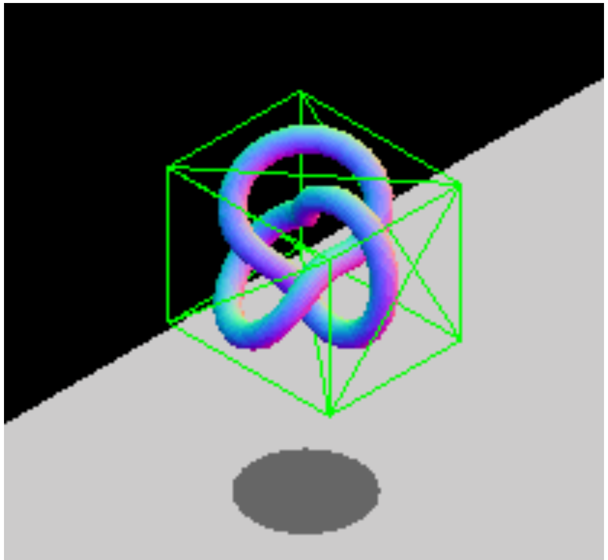
Freeze frame

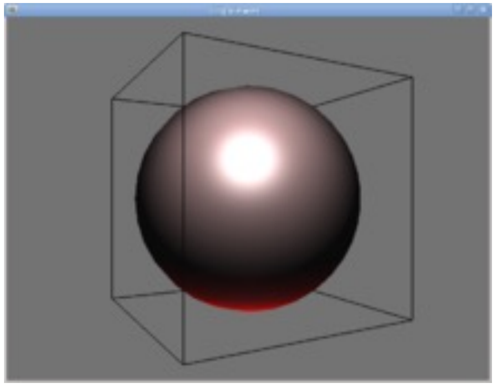


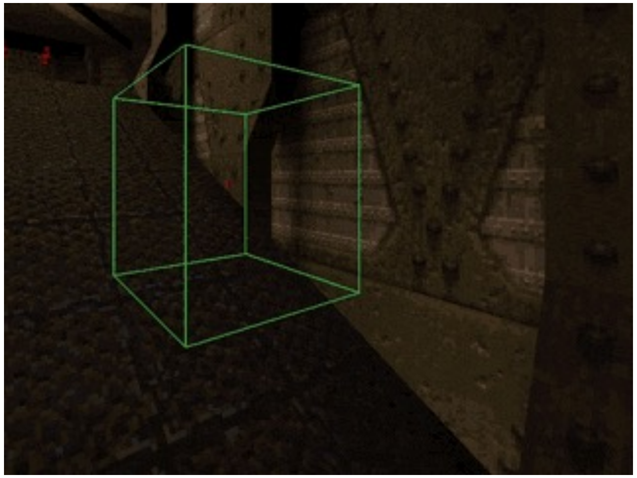
https://americanhistory.si.edu/muybridge

Object interaction: Bounding box/volume









https://www.are.na/tetlie/boundingbox
https://en.wikibooks.org/wiki/OpenGL_Programming/Bounding_box

Boids

simulation

animation

emergent behaviour

vector algebra

Our inspiration

- **National Geographic flight of the starlings**

  https://www.youtube.com/watch?v=V4f_1_r80RY

- **A murmuration of starlings**

  https://www.youtube.com/watch?v=eakKfY5aHmY&t=75s

https://bio.kuleuven.be/ento/ferrante/images/Flock-of-Starlings.jpg

Real flocks of birds & schools of fish

- Upper bound limited only by actual population
  - Starlings: flock of 1.5 million birds
  - Herring: school of hundreds of millions covering dozens of square kilometres
- Localized reasoning *must* be used
  - A bird/fish only knows about its nearest neighbours

https://youtu.be/U9T0OlAOv0c

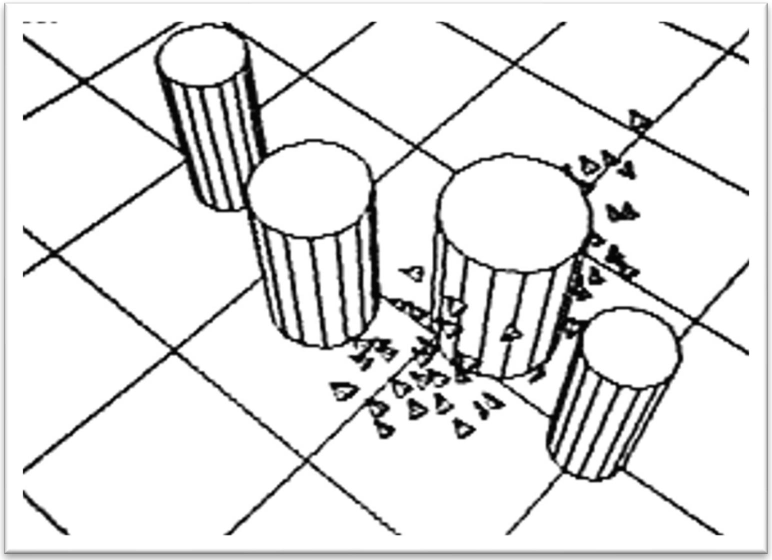
http://thefishbowlnetwork.com/blog/wp-content/uploads/2013/10/school-free-of-fish.jpg

Why go to school?

- Socialization
- Reproduction
- Protection from predators
- Foraging for food is quicker

Simulating flocking: Boids

- Early work by Reynolds
  - SIGGRAPH 1987



- Boid: a member of the flock

- Key idea: local rules produce *emergent behaviour* giving believable-looking flocking

https://www.red3d.com/cwr/boids/images/flocking_around_19.gif

- Original Boids video by Reynolds

  https://www.youtube.com/watch?v=86iQiV3-3IA

  Notice the very simple quality of the graphics, generated on one of the fastest computers at the time.
- The first use of simulated flocking in a movie was *Batman Returns* (1992). It used a modified version of the original boids software to simulate
  - bat swarms
    https://www.youtube.com/watch?v=jCVwdeAobYc
  - penguin flocks
    https://www.youtube.com/watch?v=APs3qbAE1FY

- *The Lion King* (1994) included a wildebeest stampede where a boids-like simulation was the only sane way in which to produce the effect of hundreds of animals charging down a gorge in a realistic manner
https://www.youtube.com/watch?v=XM_VHtSDMIQ
- Orcs march on Minas Tirith
https://www.youtube.com/watch?v=bPh1KXA8egU

Reynolds' rules

- Avoidance
- Alignment
- Cohesion

https://www.howitworksdaily.com/wp-content/uploads/2015/07/wallpaper-of-a-flock-of-flying-birds-hd-bird-wallpapers.jpg

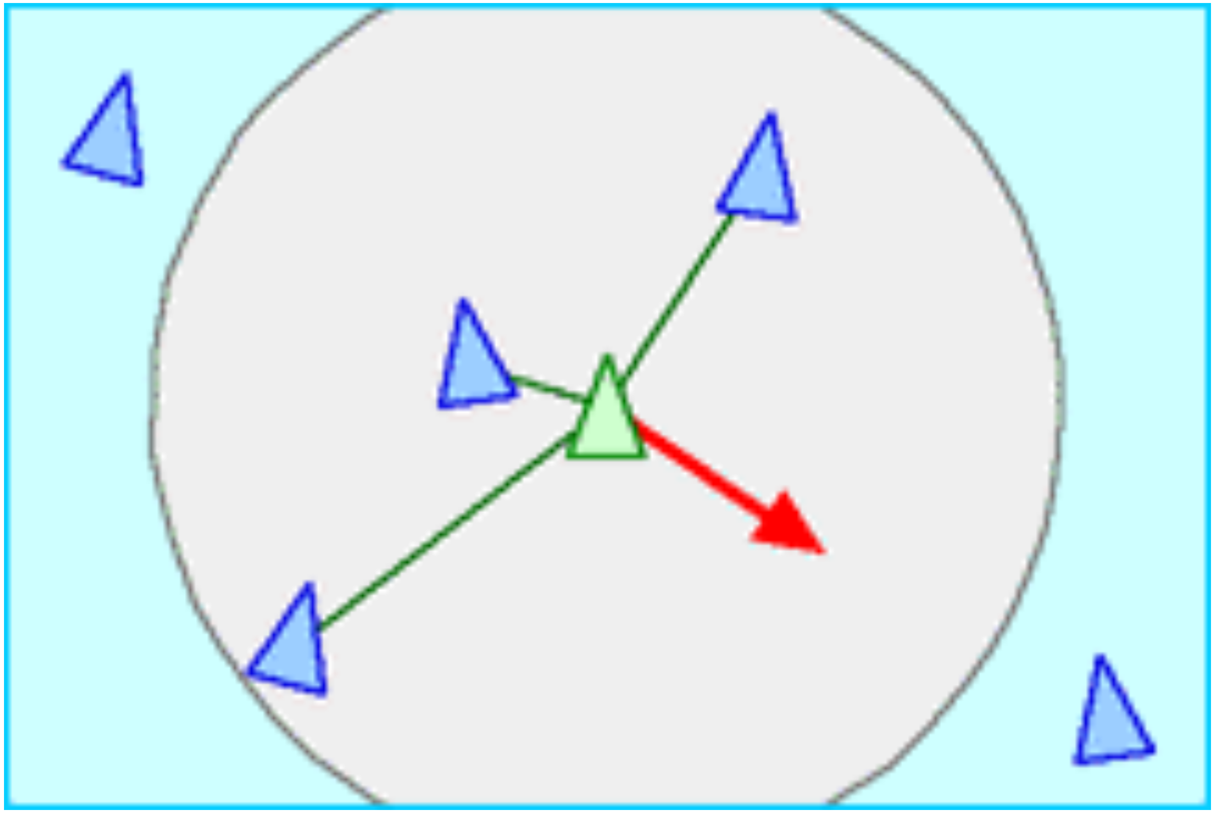
How do boids work?

- A Flocking Simulation

  https://www.youtube.com/watch?v=QbUPfMXXQIY

Avoidance

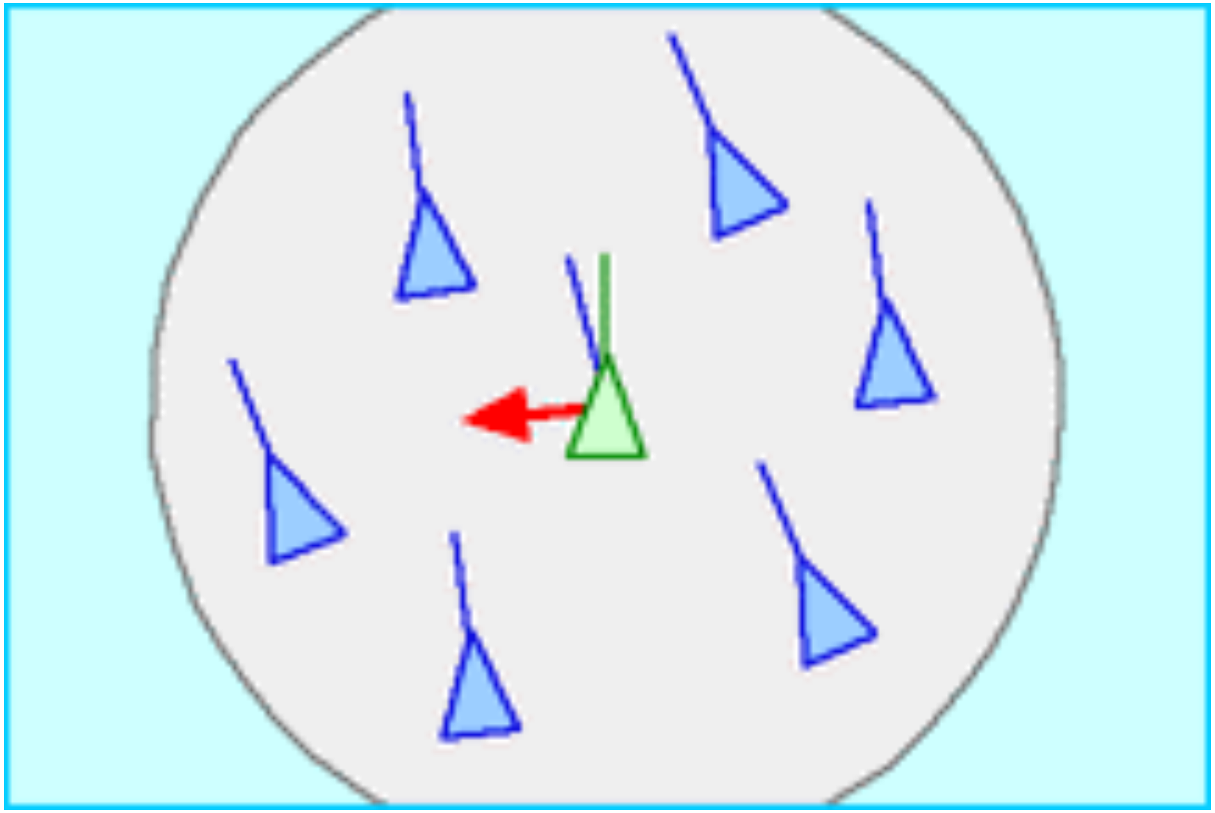
- Boids avoid crashes



- Each boid looks at the flock mates in its neighbourhood and applies a force to push it away from its neighbours

Alignment

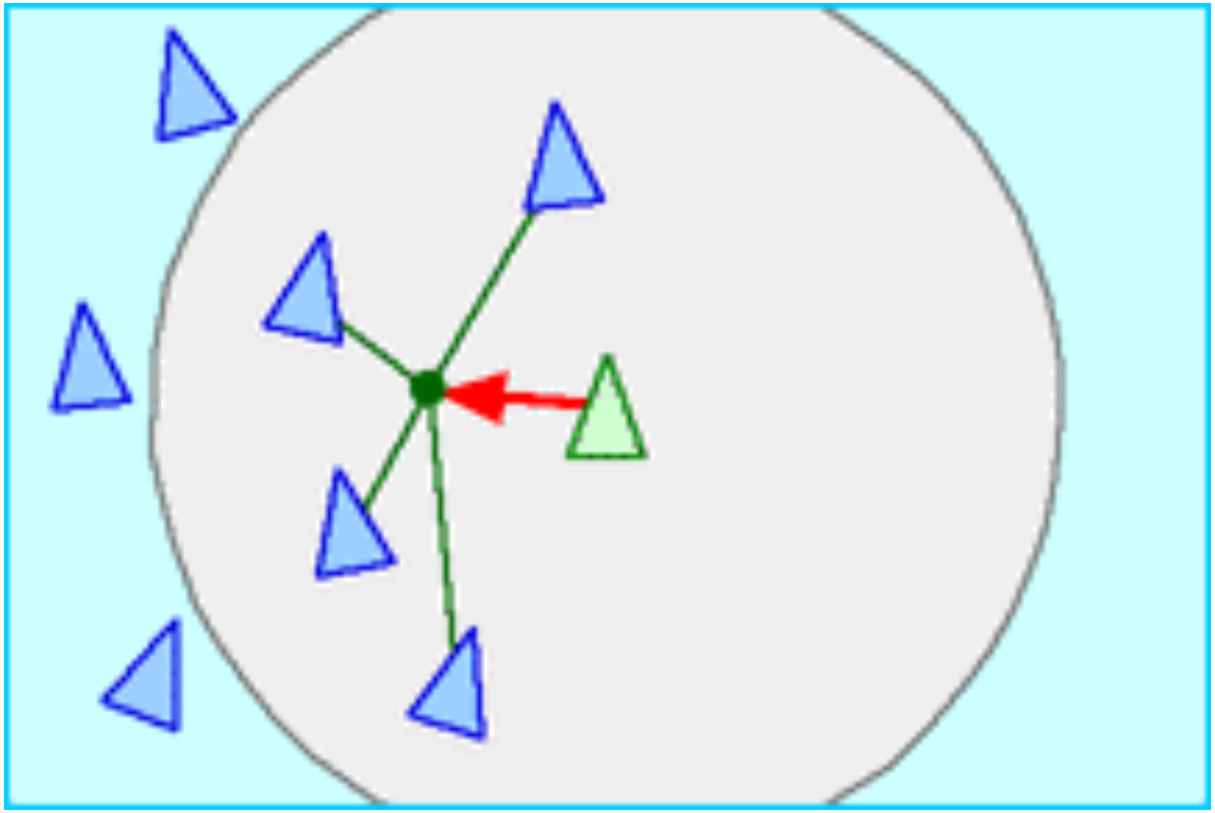
- Boids want to fly in the same direction



- Each boid looks at the flock mates in its neighborhood and applies a force to line it up with the average direction of its neighbours

Cohesion

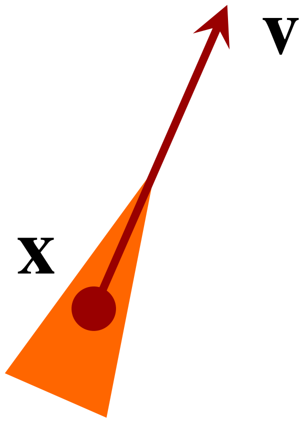
- Boids want to be near their flock mates



- Each boid looks at the flock mates in its neighborhood and applies a force to move towards the average position of its neighbours

Representing a boid

- A boid has
  - A position $$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$
  - Velocity $$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$



Updating a boid's position

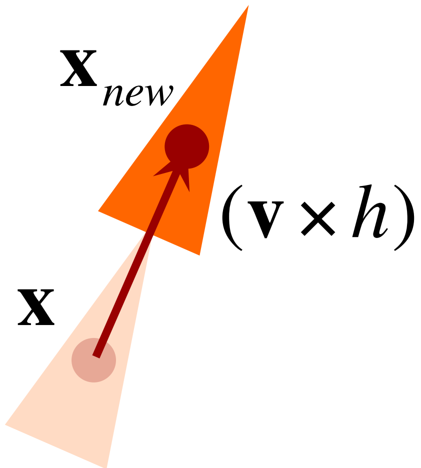
- Add velocity (times the time-step) to position

$$\mathbf{x}_{new} = \mathbf{x} + (\mathbf{v} \times h)$$

$$x_{new} = x + (v_x \times h)$$

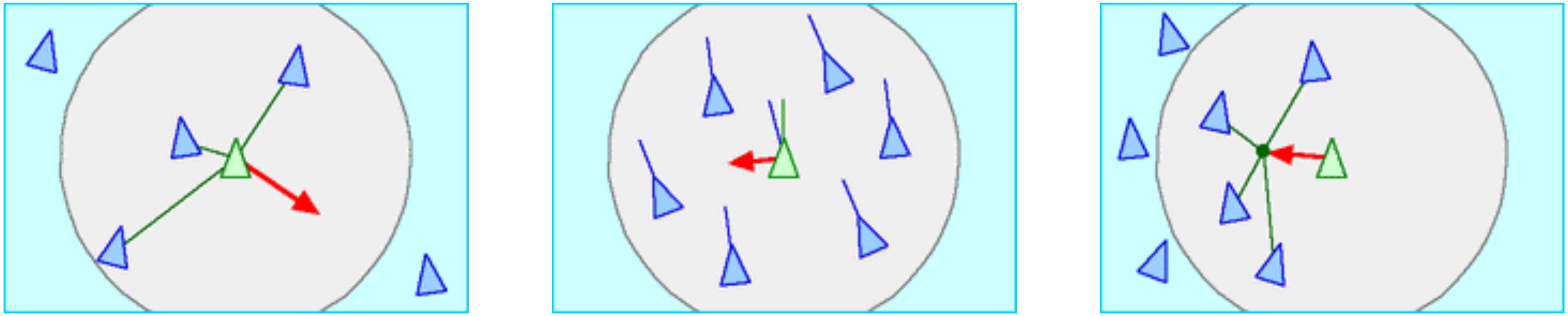
$$y_{new} = y + (v_y \times h)$$

$$z_{new} = z + (v_z \times h)$$



- h is the time-step (scalar)

How do the forces work?



$$\mathbf{x}_{new} = \mathbf{x} + (\mathbf{v} \times h)$$

- Position is updated by velocity

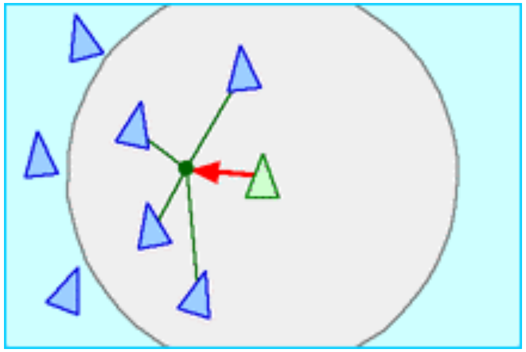
$$\mathbf{v}_{new} = \mathbf{v} + (\mathbf{a} \times h)$$

- Velocity is updated by acceleration

$$\mathbf{a} = \mathbf{f} / m$$

- Acceleration is force/mass

Cohesion



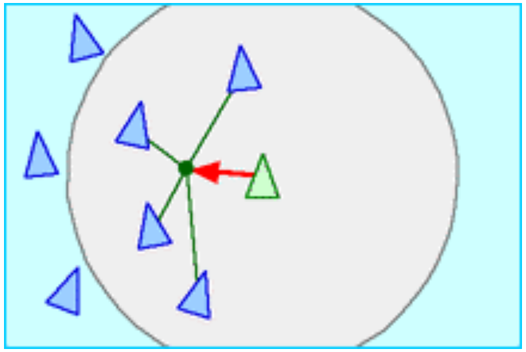
- Find the centroid of the neighbours' positions

$$\mathbf{x}_c = \sum_{j \in N(i)} \mathbf{x}_j / |N(i)|$$

- Create a force that goes from your position to the centroid

$$\mathbf{f}_x = k_x(\mathbf{x}_c - \mathbf{x}_i)$$

Who are the boids in your neighbourhood?

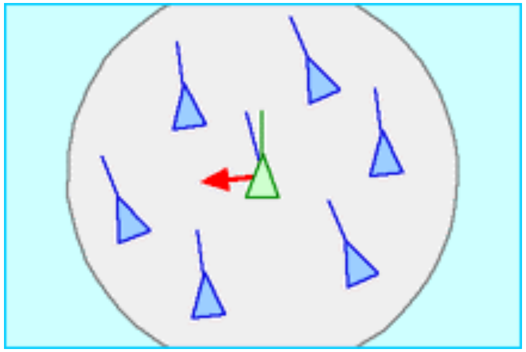


- We specify a distance d that a boid can "see"
- We check all other boids to see if they are within distance d

$$N(i) = \left\{ j : j \neq i \wedge \left| \mathbf{x}_j - \mathbf{x}_i \right| < d \right\}$$

Google: "Sesame Street who are the people in your neighbourhood"

Alignment



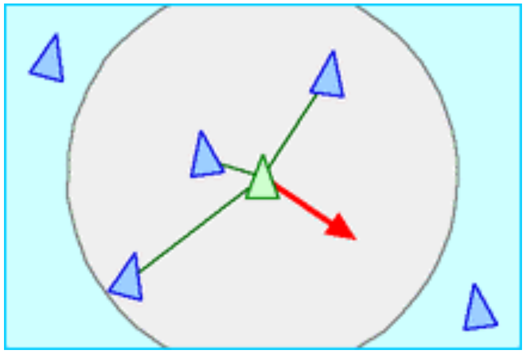
- Find the average of the neighbours' velocities

$$\mathbf{v}_c = \sum_{j \in N(i)} \mathbf{v}_j / |N(i)|$$

- Create a force that adjusts the boid's velocity to be closer to the average speed

$$\mathbf{f}_v = k_v(\mathbf{v}_c - \mathbf{v}_i)$$

Avoidance



$$\mathbf{f}_a = k_a \overbrace{\sum_{\underbrace{j \in N(i)}_{1}}}^{4} \underbrace{\frac{1}{|x_i - x_j|}}_{3} \underbrace{\frac{(x_i - x_j)}{|x_i - x_j|}}_{2}$$

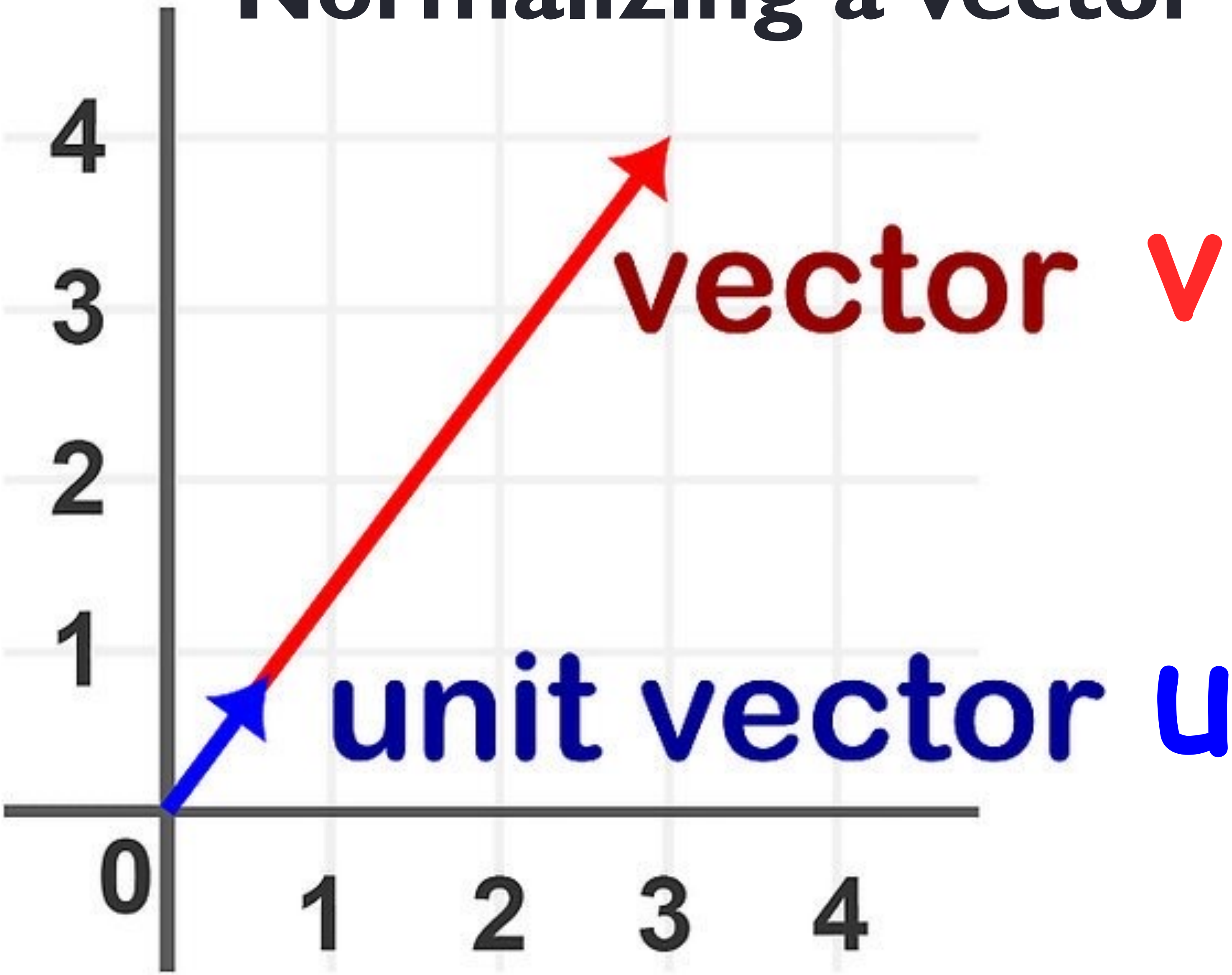
1 For each boid in the neighbourhood create a force that

2 pushes away from the boid,

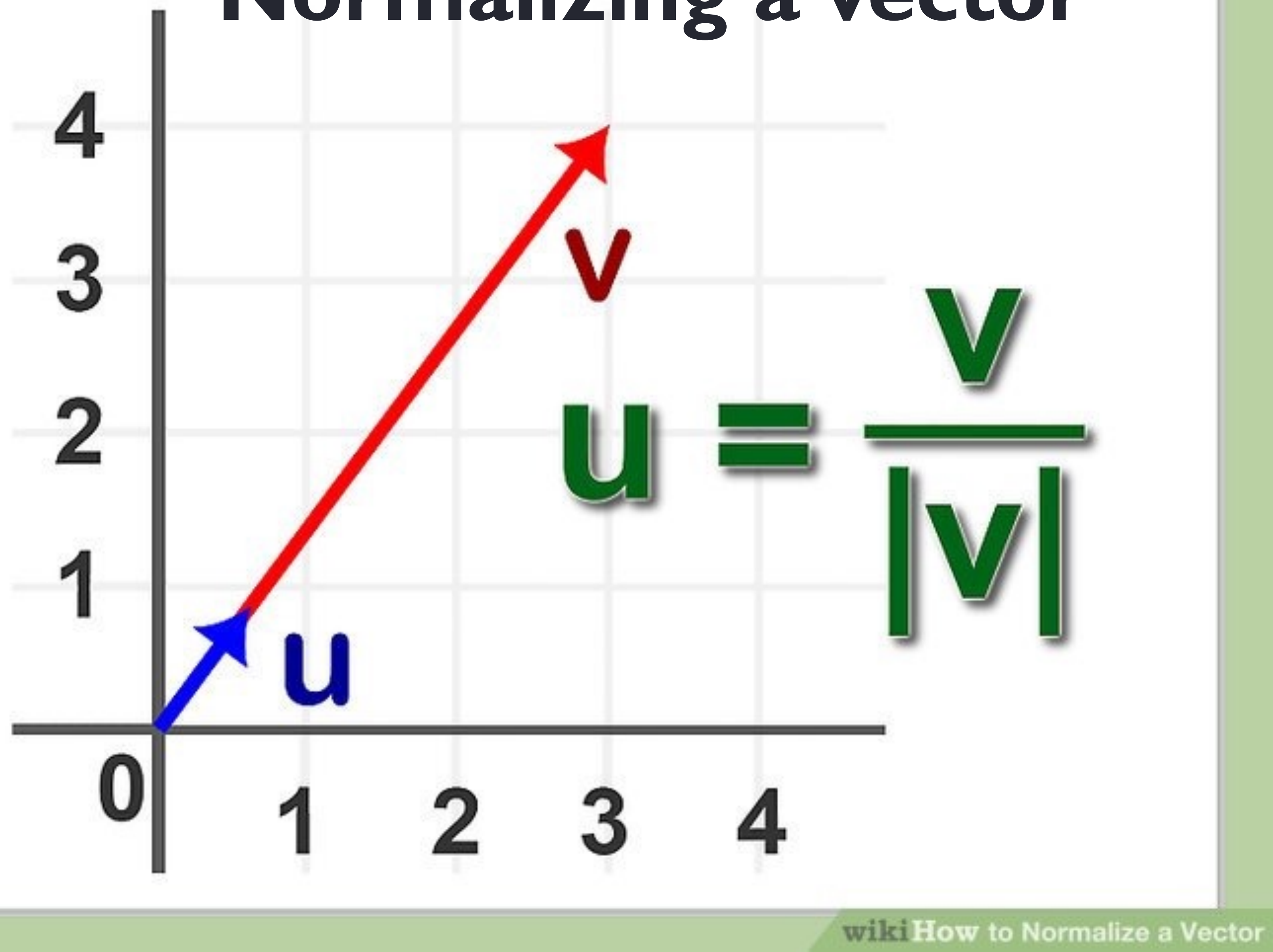
3 weighted by the inverse of the distance,

4 add all these forces together

Normalizing a vector



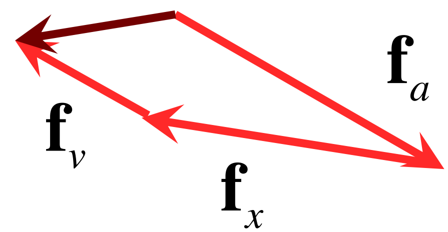
Normalizing a vector



Applying those forces

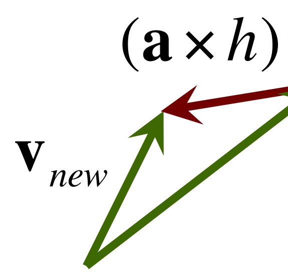
- Add up the forces and divide by the boid's mass

$$\mathbf{a} = (\mathbf{f}_x + \mathbf{f}_v + \mathbf{f}_a)/m$$



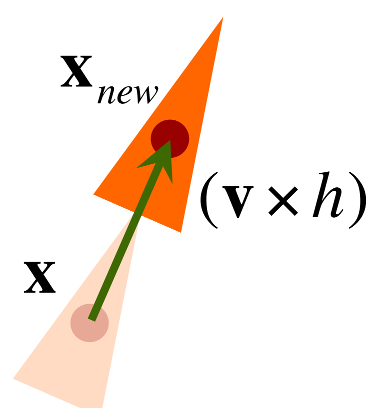
- Update velocity

$$\mathbf{v}_{new} = \mathbf{v} + (\mathbf{a} \times h)$$



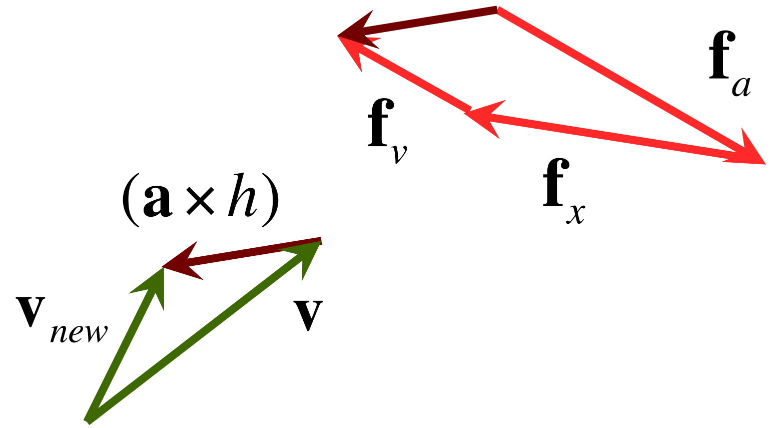
- Update position

$$\mathbf{x}_{new} = \mathbf{x} + (\mathbf{v} \times h)$$

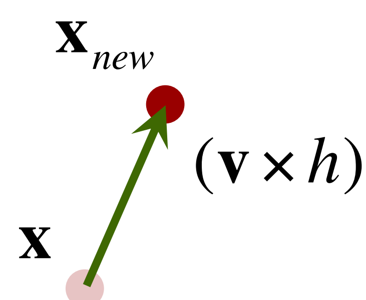


What operations make sense?

- vector = vector + vector
- vector = vector – vector
- vector = scalar × vector
- vector = point – point

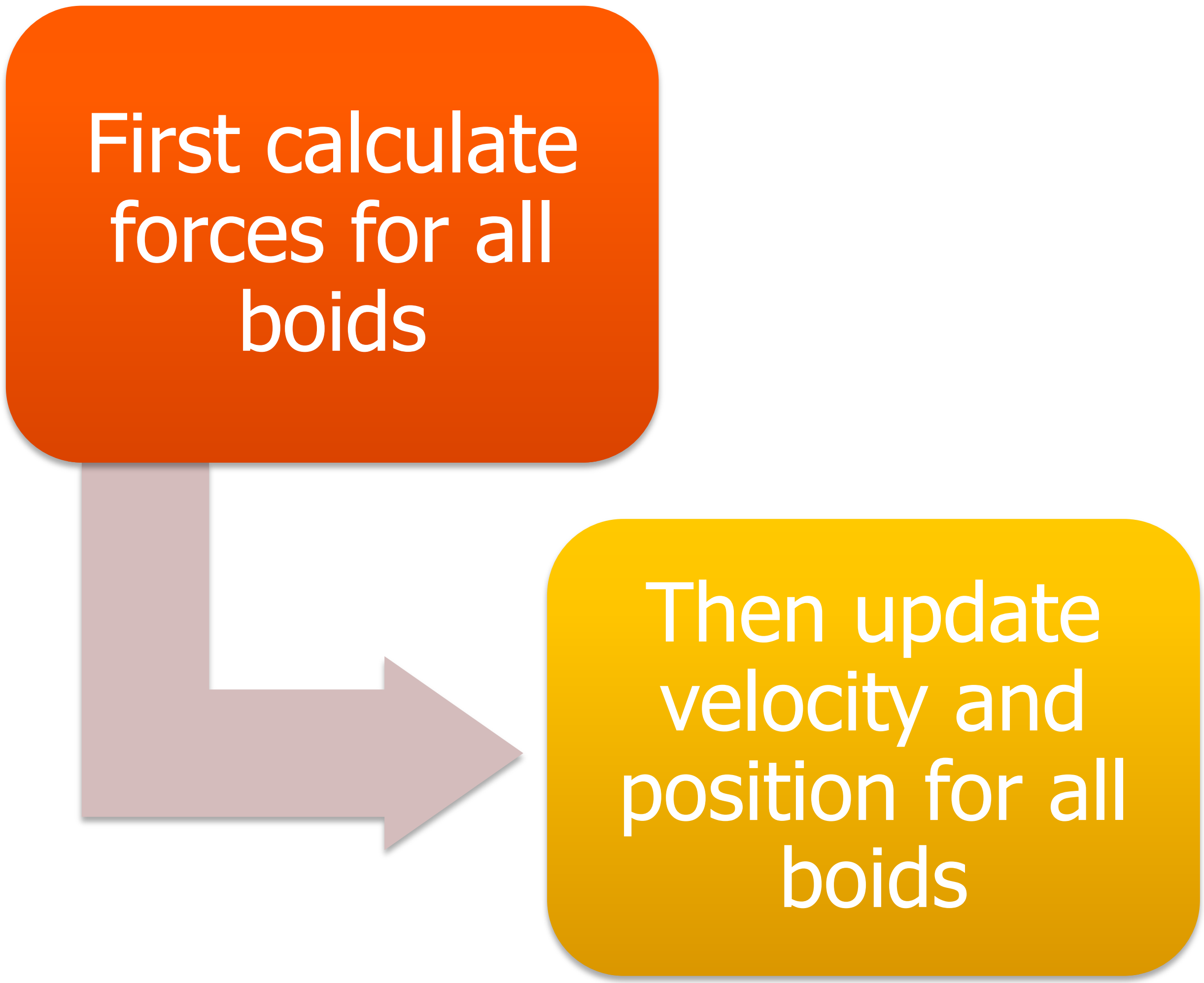


- point = point + vector
- point = point – vector
- point = average of points



$$\mathbf{x}_c = \sum_{j \in N(i)} \mathbf{x}_j / |N(i)|$$

Must update all boids together



What they don't usually tell you

- Need to balance forces carefully (experiment)
  - Careful choice of k_x, k_v, k_a
- Mass is an arbitrary number
  - If all boids weigh the same, can pretend that $m=1$
- Need to limit speed and force
  - Enforce a maximum speed and a maximum force
  - Apply maximum force limit to each force individually

How do I limit a vector?

Limiting a scalar

$$\text{if } f > \max$$

$$\text{then } f_{new} = \max$$

Limiting a vector

$$\text{if } |\mathbf{f}| > \max$$

$$\text{then } \mathbf{f}_{new} = \max \times \frac{\mathbf{f}}{|\mathbf{f}|}$$

What else haven't you told me?

- You need to stop the boids from flying off into the distance
- Define an axis-aligned box to keep the boids in and then:

Force *or* ***Bounce*** *or* ***Wrap***