COMP3421/9415
Computer Graphics

Further geometry & Transformations

Robert Clifton-Everest

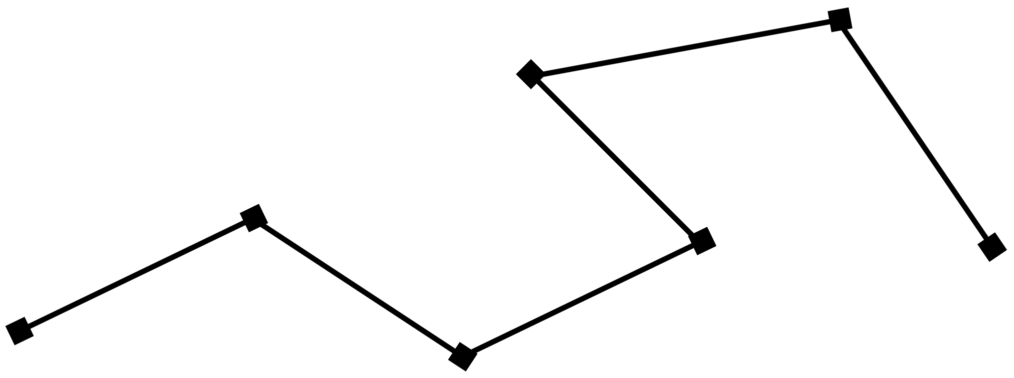
Email: robertce@cse.unsw.edu.au

Recap

- We know how to use OpenGL to draw points and lines
  - OpenGL API
  - Manual memory management
- Lab task available for practice
  - Help session at 3-4PM on Thursday and 2-3PM on Friday. In the Piano lab (K14 underground)

Line strips

- A line strip is a series of points joined by lines



- They can be drawn with GL_LINE_STRIP
- See LineStrip2D.java

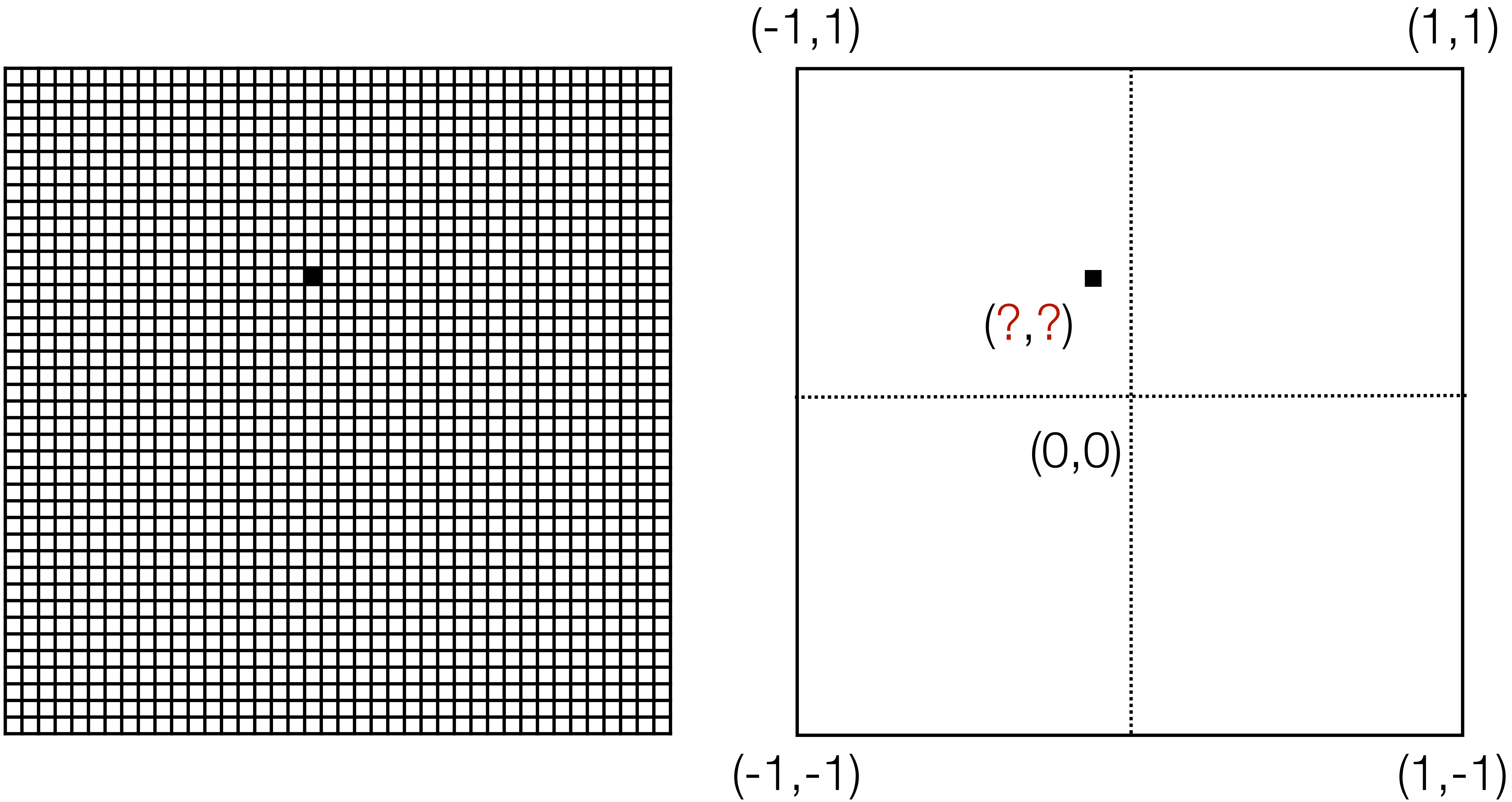
Mouse Input events

- We can add mouse event listeners to handle input.
  - http://jogamp.org/deployment/v2.3.2/javadoc/jogl/javadoc/com/jogamp/newt/event/MouseListener.html
- Adaptors let us only handle the events we care about.
  - http://jogamp.org/deployment/v2.3.2/javadoc/jogl/javadoc/com/jogamp/newt/event/MouseAdapter.html
- See LineDrawing.java

Mouse Events

- When we click on the screen we get the mouse co-ordinates in screen co-ordinates.
- We need to somehow map them back to viewport co-ordinates.

Mouse Events

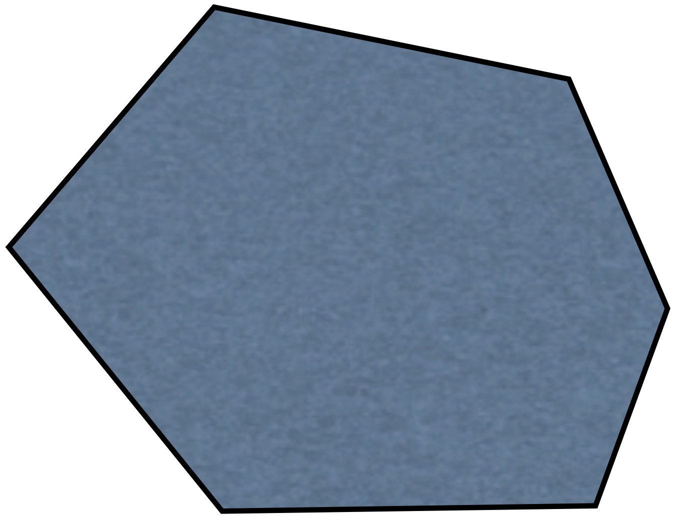


Triangles

- We can draw triangles with GL_TRIANGLES
- See Triangle2D.java and TriangleDrawing.java

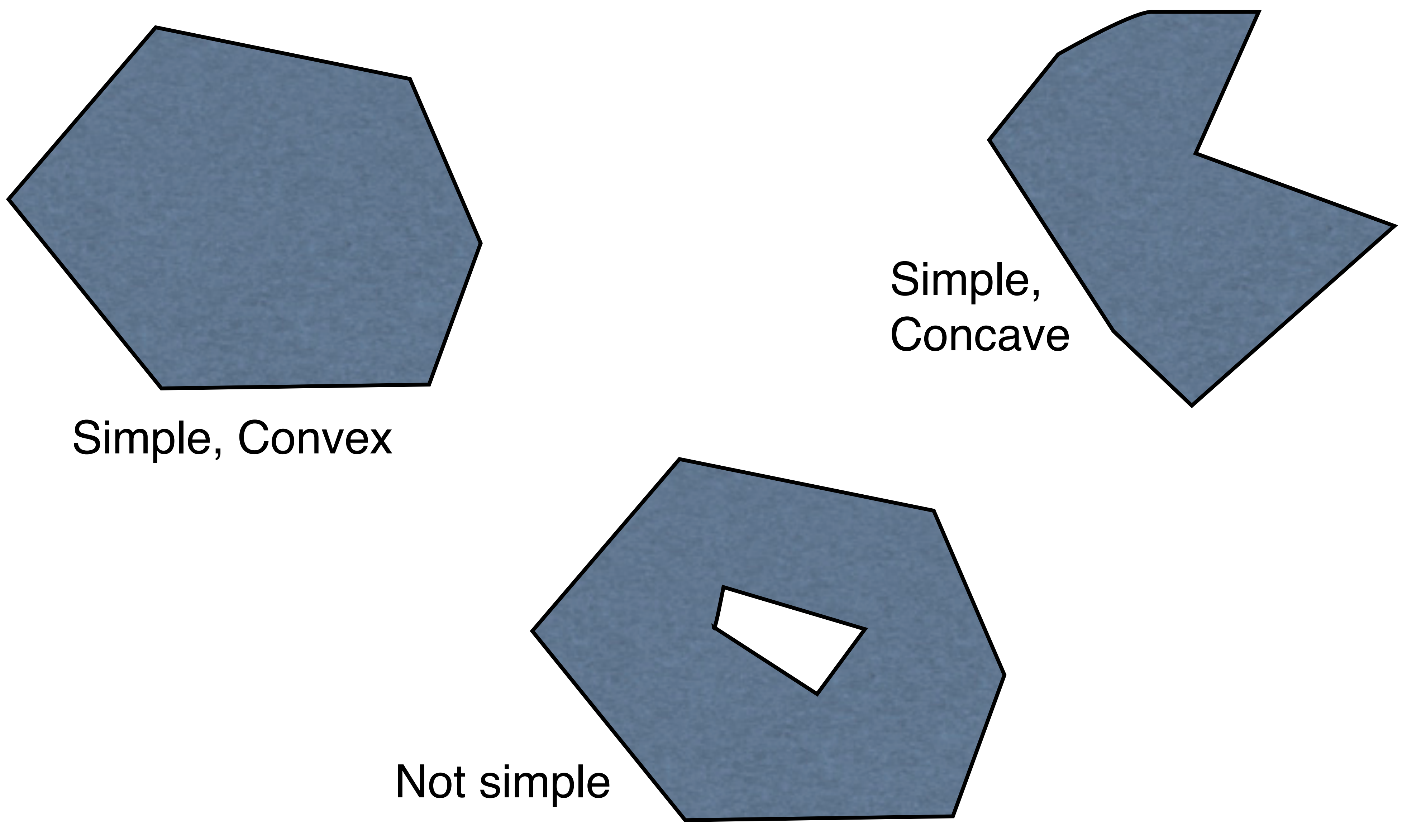
Polygons

- Shapes with an arbitrary number of sides



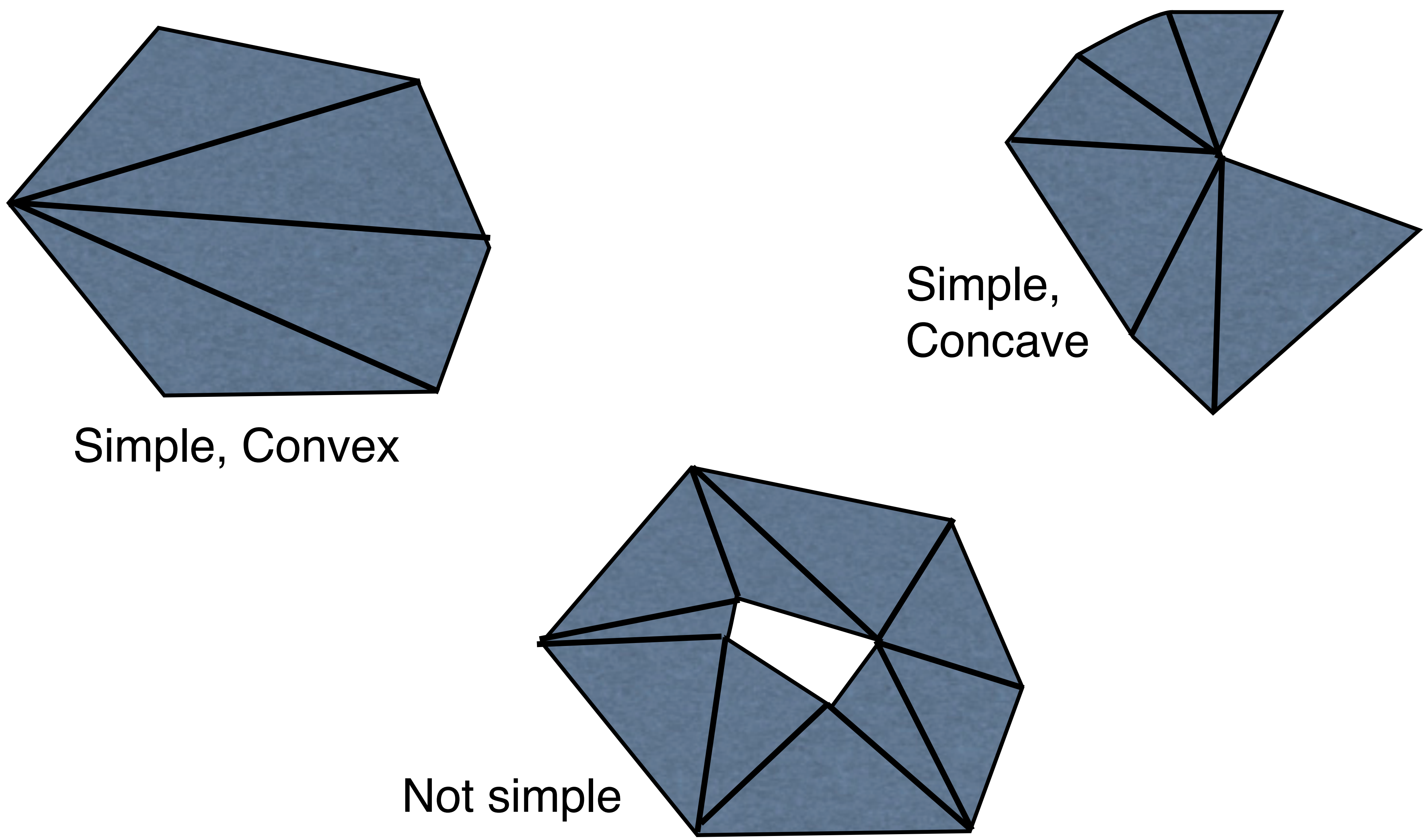
- Whether or not we can easily draw them depends on a few factors

Polygons



Tessellation

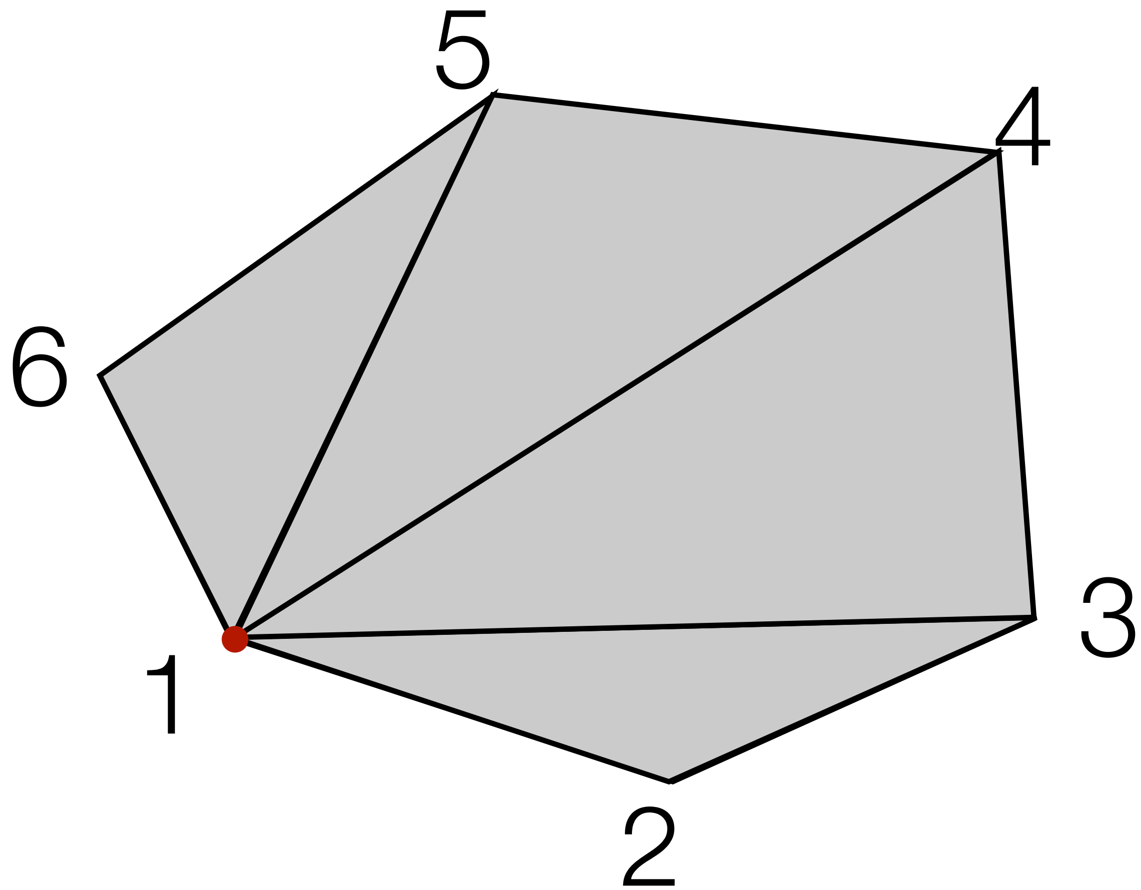
- We can draw polygons by splitting them up into simpler shapes (typically triangles)



Triangle Fans

- One simple method is to use a triangle fan.
- Start with any vertex of the polygon and move clockwise or counter-clockwise around it.
- The first three points form a triangle. Any new points after that form a triangle with the last point and the starting point.

Triangle Fans



Triangle Fans

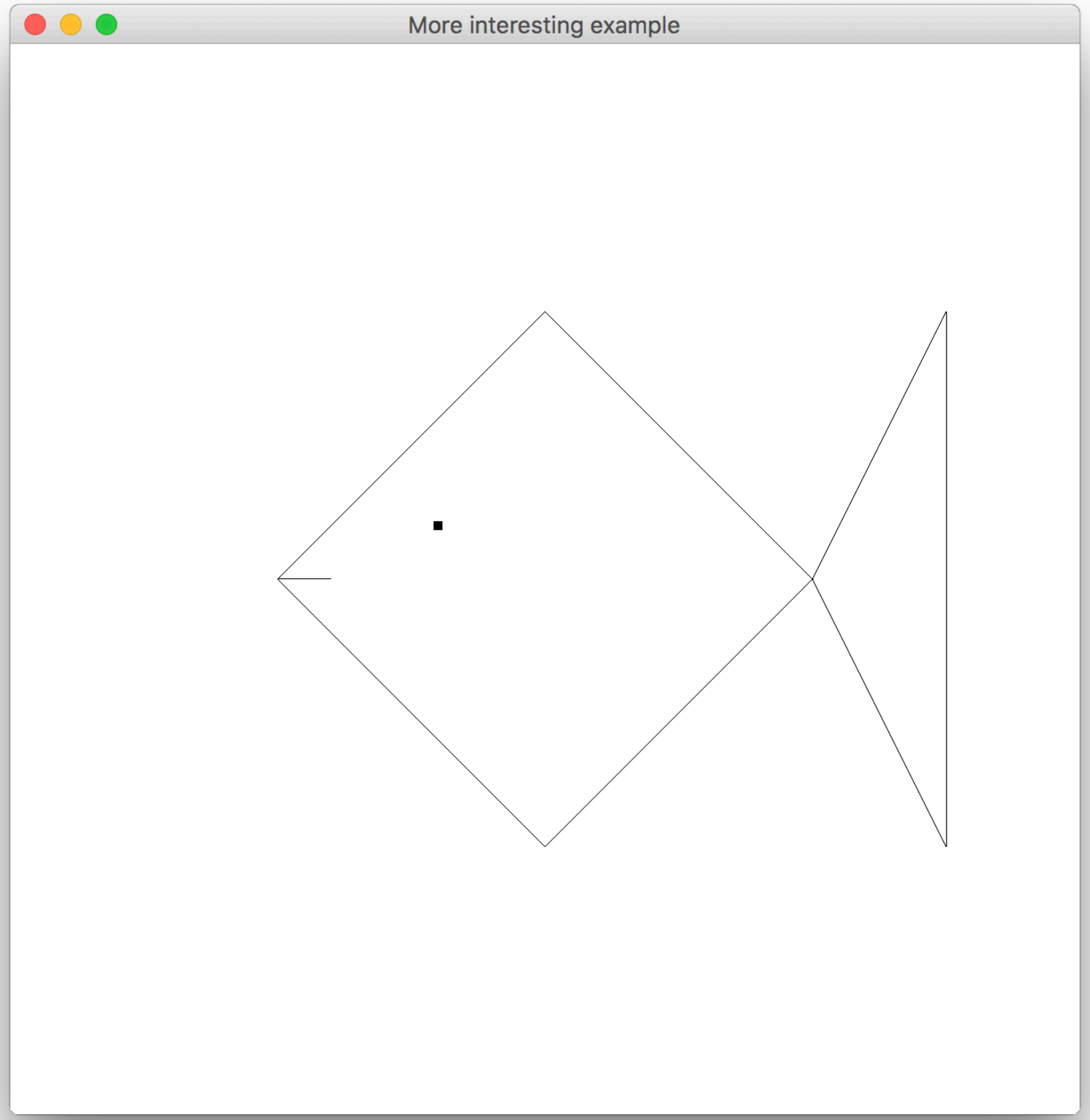
- Works for all simple convex polygons, and some concave ones
- Can be drawn with GL_TRIANGLE_FAN
- The lab task

Transformations

- One of the fundamental concepts we will cover in this course
- This week and next week we will be focusing on 2D transformations.

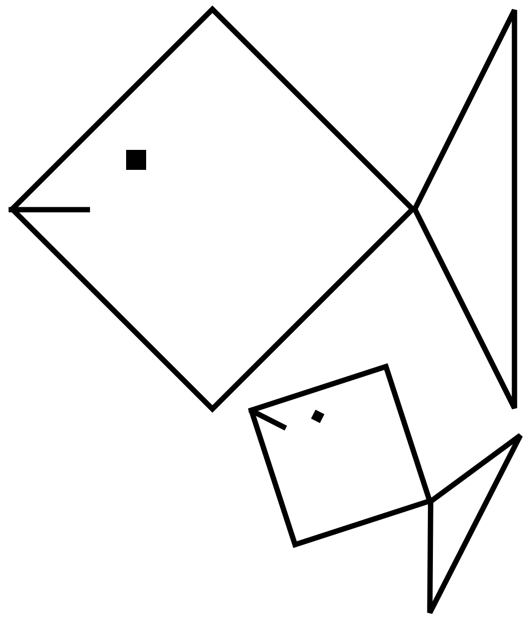
Back to the fish

- So we can draw a fish?
- What if we wanted to draw one that was somewhere else
  - … or smaller
  - … or swimming upwards



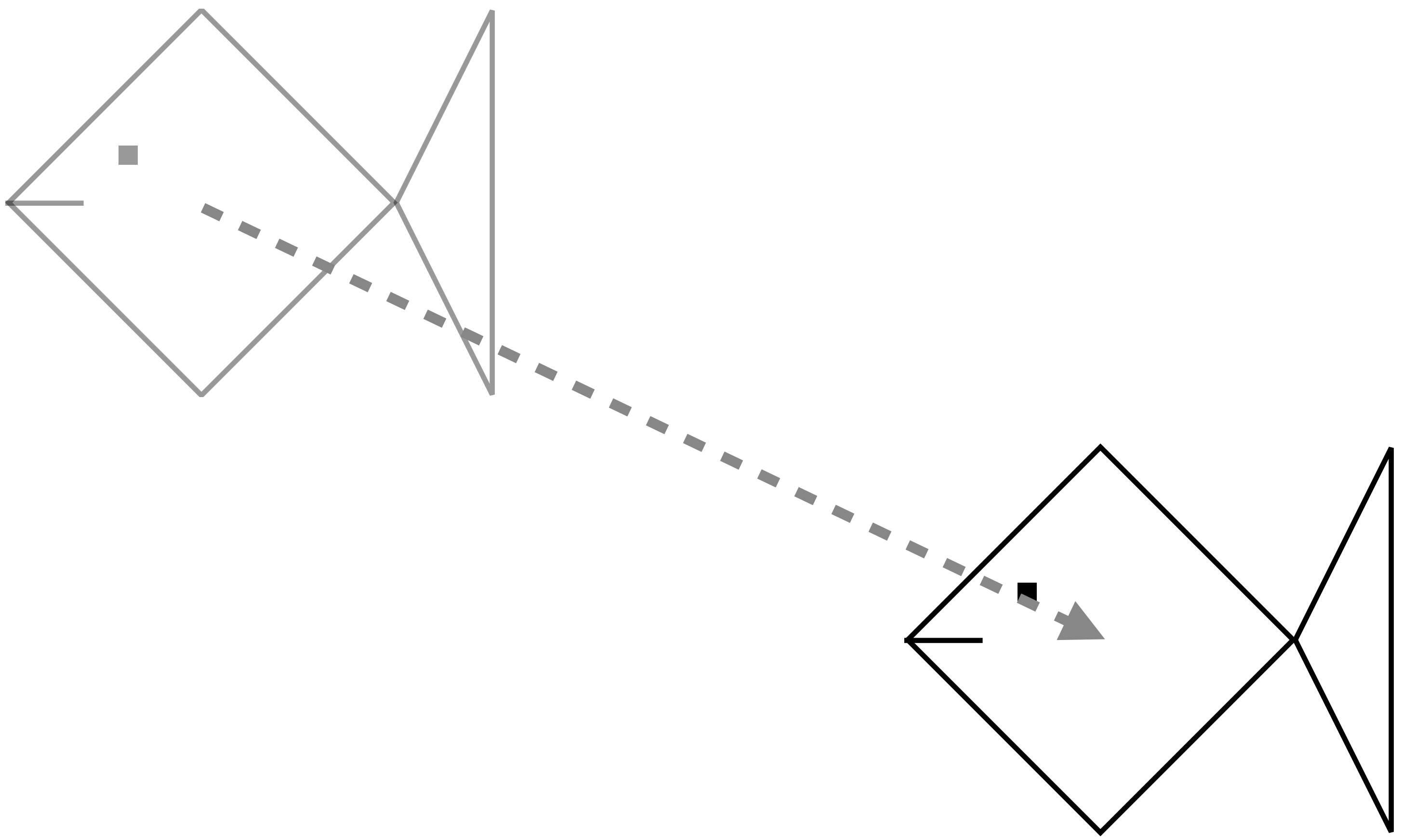
Back to the fish

- What if we wanted to draw a scene like this?



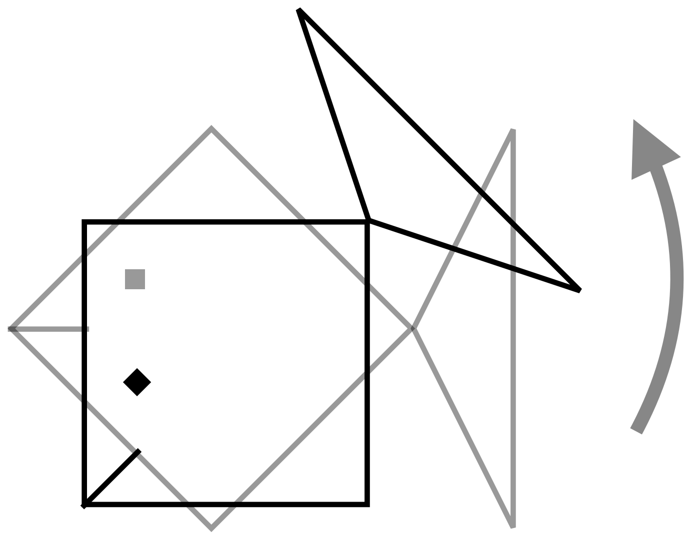
Translation

- Translation is the process of moving an object in space



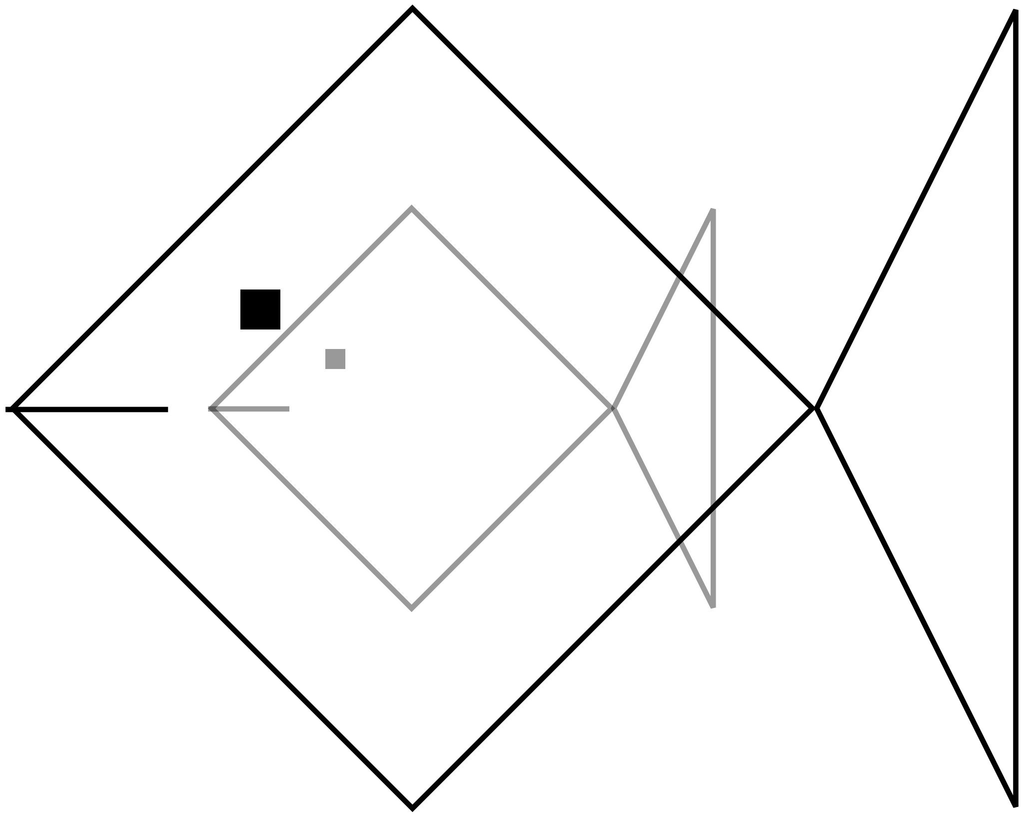
Rotation

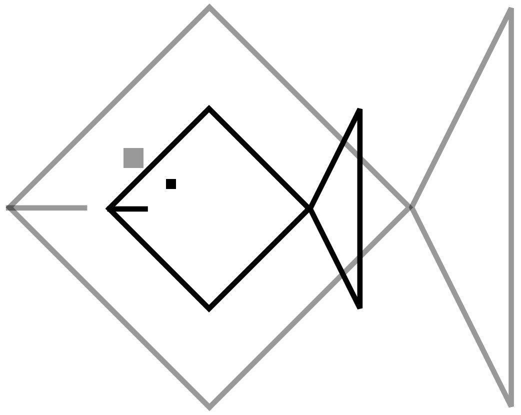
- Rotate objects around the origin



Scaling

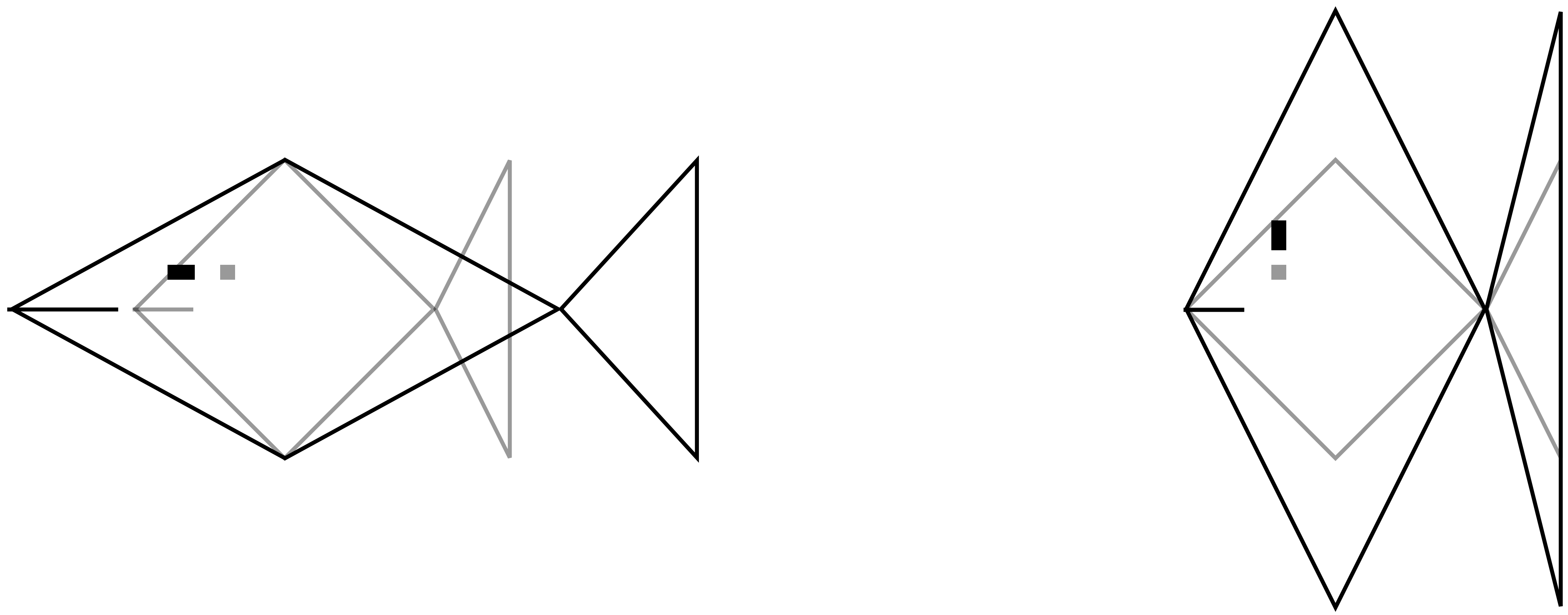
- Scale along both axes.





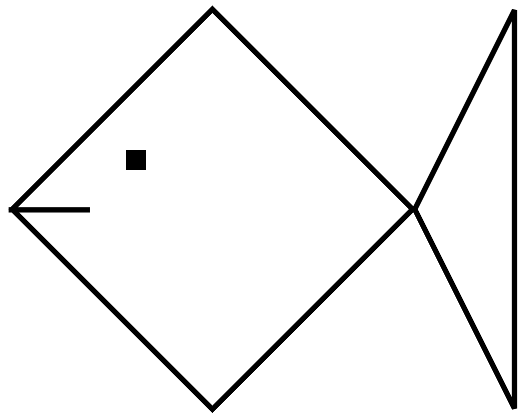
Scaling

- Or scale across only one axis



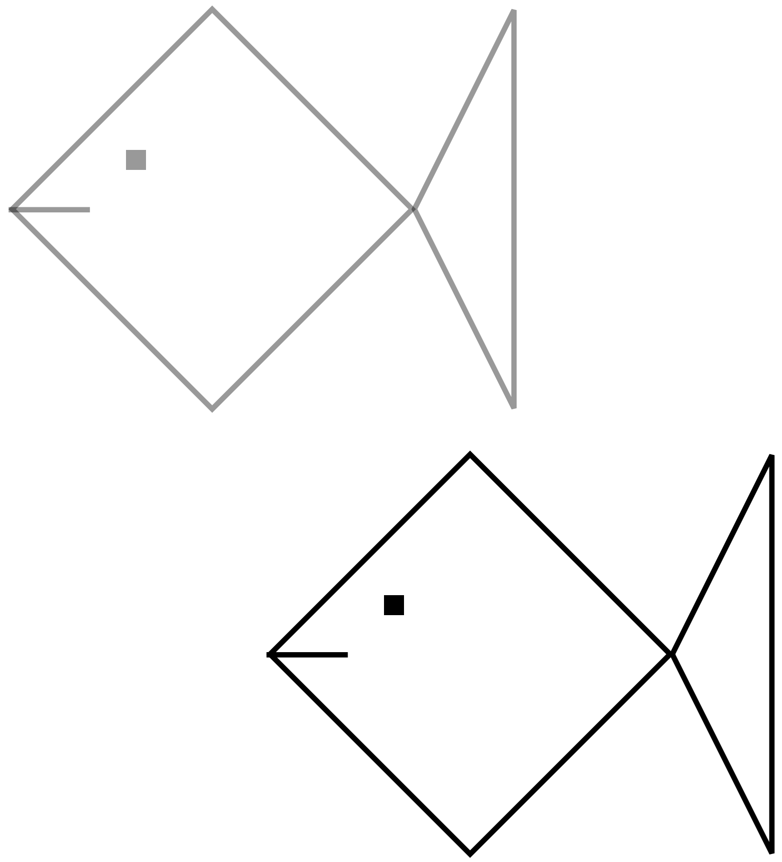
Composition

- We can compose these transforms to arrange a fish however we want e.g.



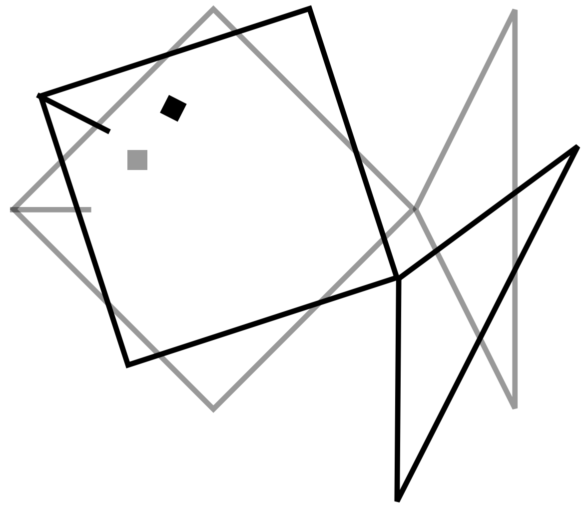
Composition

- We can compose these transforms to arrange a fish however we want e.g.
  - Translate



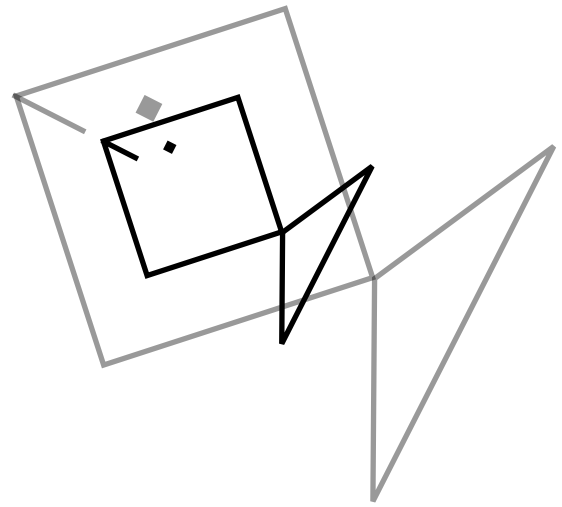
Composition

- We can compose these transforms to arrange a fish however we want e.g.
  - Translate
  - Rotate



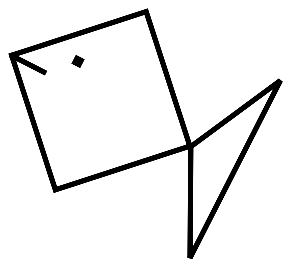
Composition

- We can compose these transforms to arrange a fish however we want e.g.
  - Translate
  - Rotate
  - Scale



Composition

- We can compose these transforms to arrange a fish however we want e.g.
  - Translate
  - Rotate
  - Scale



Transformations

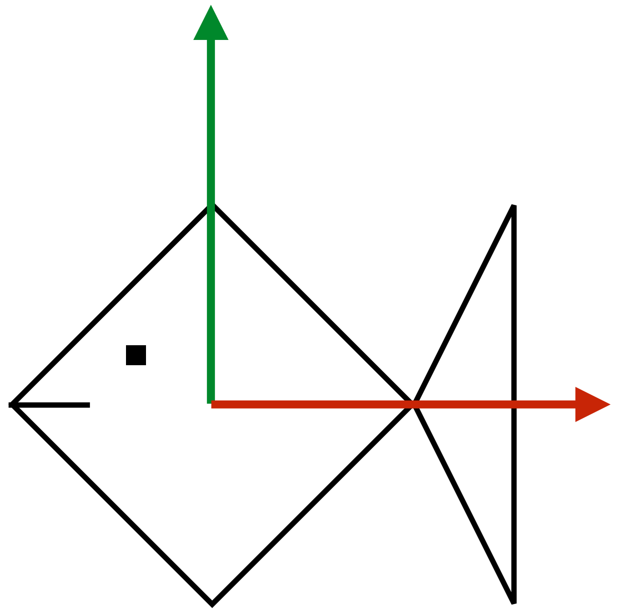
- We can think of transformations in two ways

1. Extrinsic: An object being transformed or altered within a **fixed co-ordinate** system.

2. Intrinsic: The co-ordinate system of the object being transformed. **This is generally the way we will think of it**.

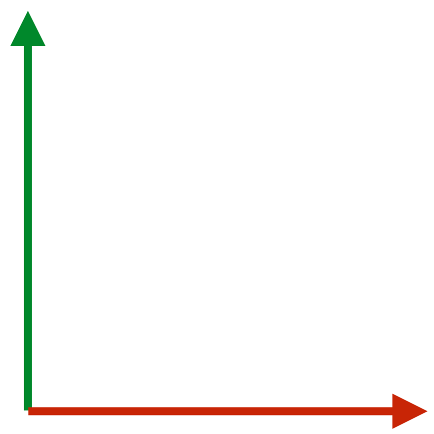
Intrinsic transformations

- We transform the coordinate system the fish is in. e.g.



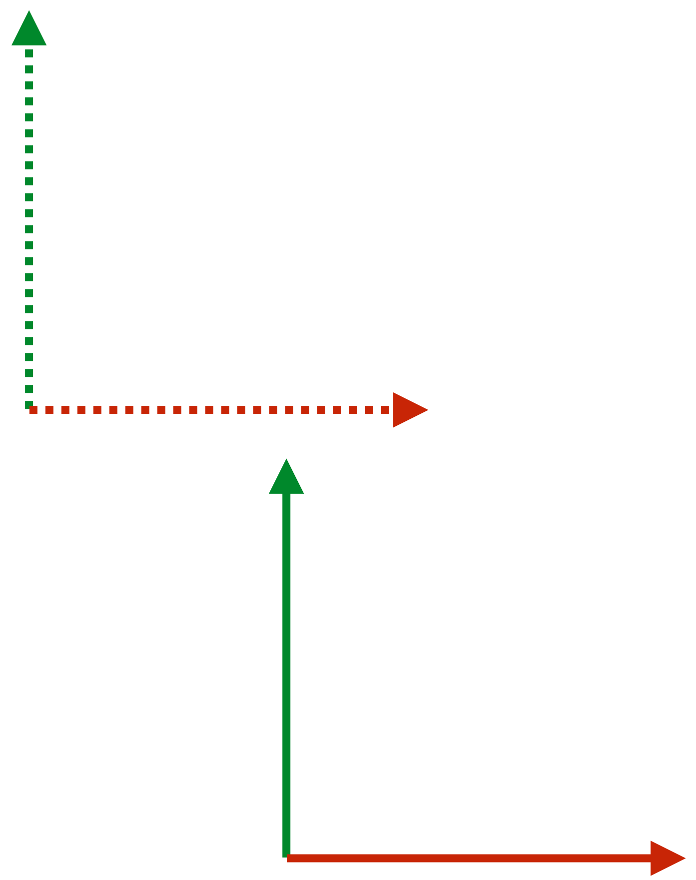
Intrinsic transformations

- We transform the coordinate system the fish is in. e.g.



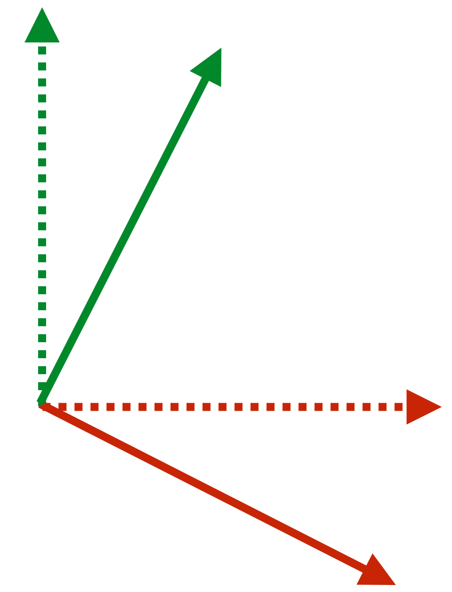
Intrinsic transformations

- We transform the coordinate system the fish is in. e.g.
  - Translate



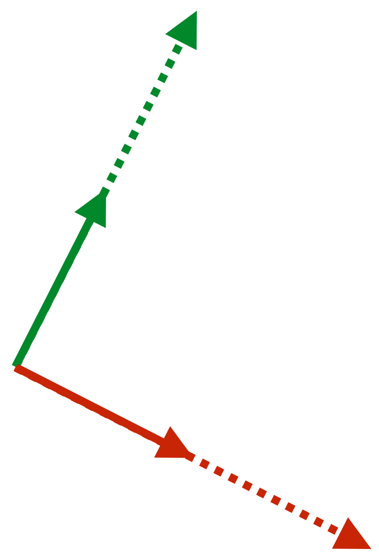
Intrinsic transformations

- We transform the coordinate system the fish is in. e.g.
  - Translate
  - Rotate



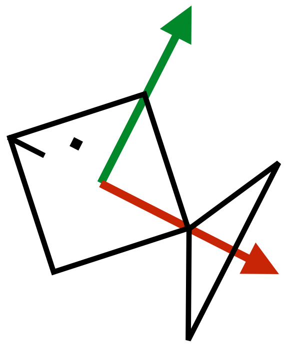
Intrinsic transformations

- We transform the coordinate system the fish is in. e.g.
  - Translate
  - Rotate
  - Scale



Intrinsic transformations

- We transform the coordinate system the fish is in. e.g.
    - Translate
    - Rotate
    - Scale
    - Draw fish



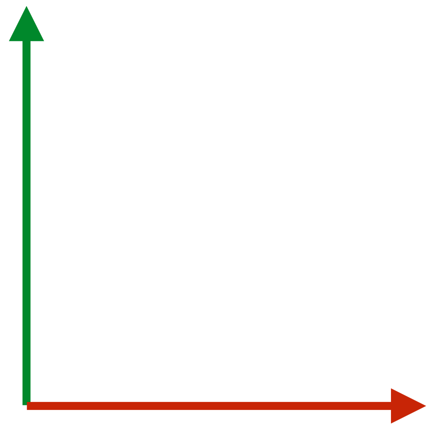
Model transformation

- A model transformation describes how a local coordinate system maps to the world coordinate system.

- Each object in a scene has its own local coordinate system.

Coordinate frames

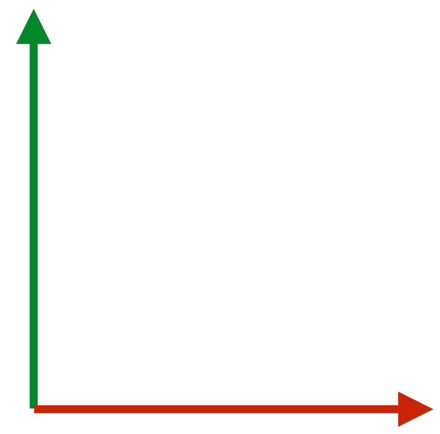
- We define a coordinate system by a coordinate frame.



- It defines the origin and the direction and scale of the x and y-axes.

Coordinate frames

- A helpful analogy is to think of a coordinate frame as a more *general* form of cursor.
  - Mouse cursors can *only* be translated
  - Coordinate frames can *also* be rotated and scaled (and more).



Identity frame

- The coordinate frame with:
  - an origin at (0,0)
  - y-axis vertical and of length 1
  - x-axis horizontal and of length 1
- … is referred to as the identity frame

Coordinate Frames in UNSWgraph

- The CoordFrame2D class represents a coordinate frame in 2D.
  - Constructed via the static method CoordFrame2D.identity()
  - Has methods for generating transformed coordinate frames.
  - See TransformingFish.java.

Transformations

- We can apply different transformations to the coordinate frame:

  - `translate(float x, float y)`
  - `rotate(float degrees)`
  - `scale(float x, float y)`

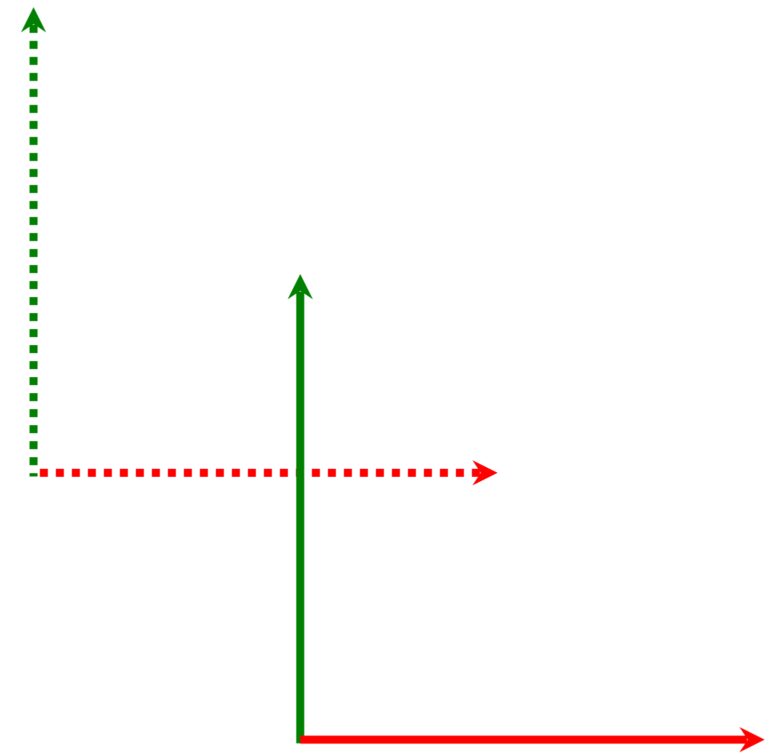
- Giving the frame as an argument to the draw methods will draw them in the coordinate system represented by the frame. e.g.

  - `line.draw(gl,frame)`

`translate(x,y)`

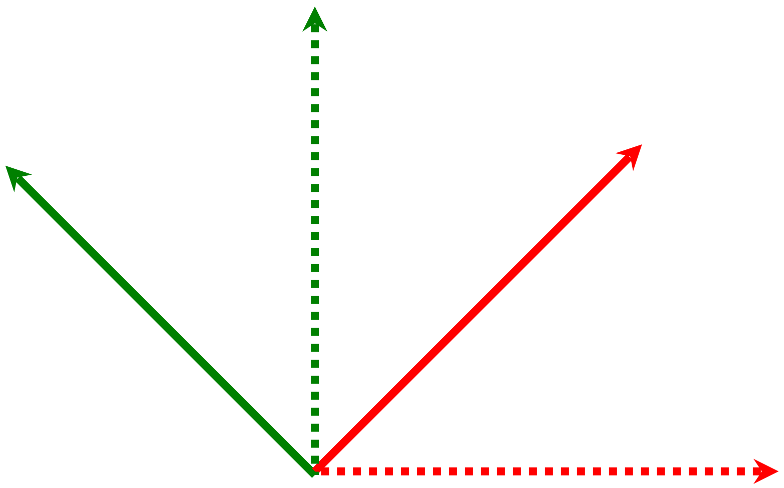
- Translate the coordinate space by the specified amount along each axis.

- In this case the origin of the co-ordinate frame moves.



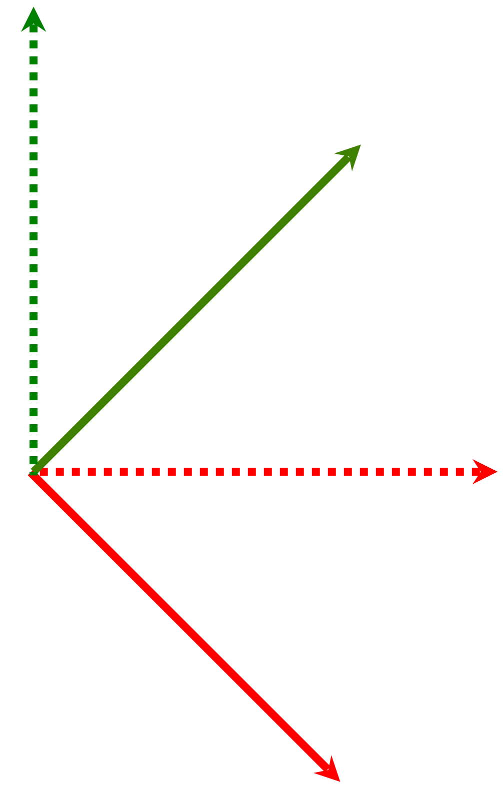
rotate(degrees)

- Rotate the coordinate space by the specified angle.
- Notice, the origin of the co-ordinate frame doesn't move



`rotate(degrees)`

- Angles are in degrees.
- Positive rotations are rotating x towards y.
- Negative rotations are rotating y towards x.

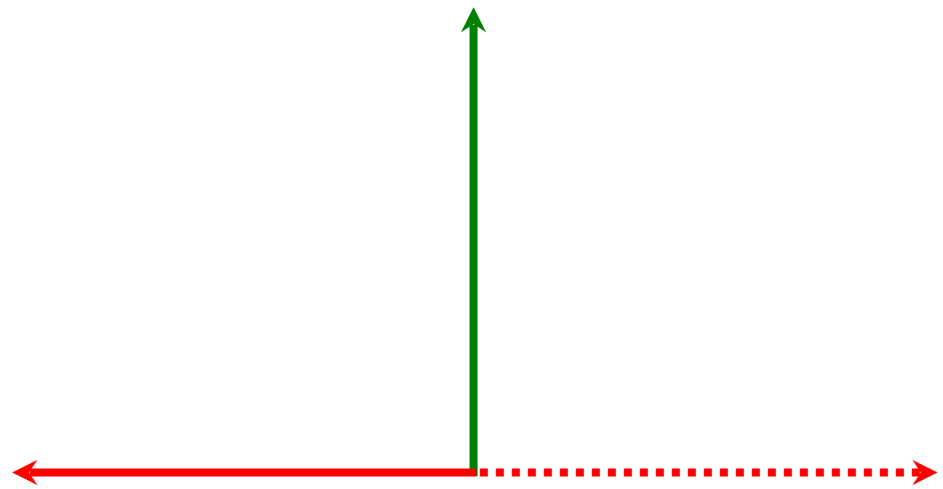


`scale(x, y)`

- Scale the coordinate space by the specified amounts in the **`x`**, **`y`** directions.
- Notice again, the origin of the co-ordinate doesn't move.

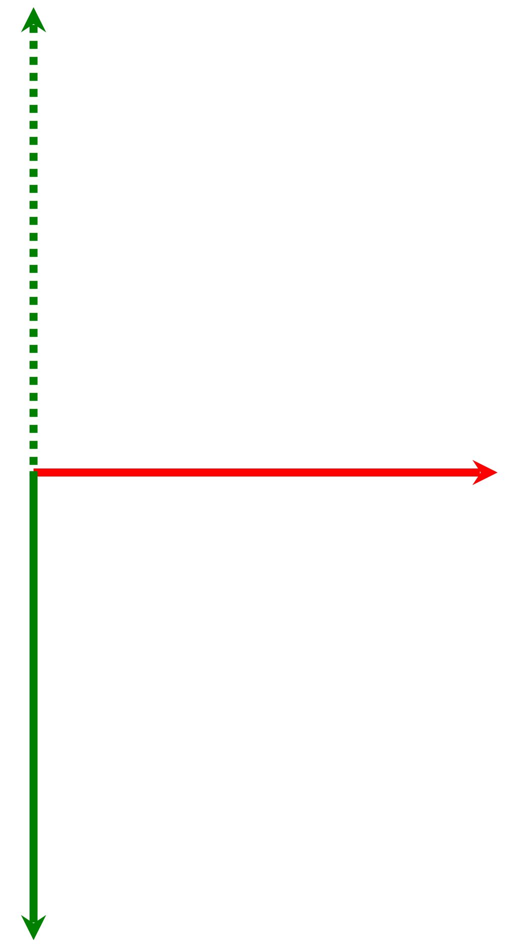
`scale(x, y)`

- Negative scales create reflections.
- e.g. scale(-1,0)
- Flip horizontally



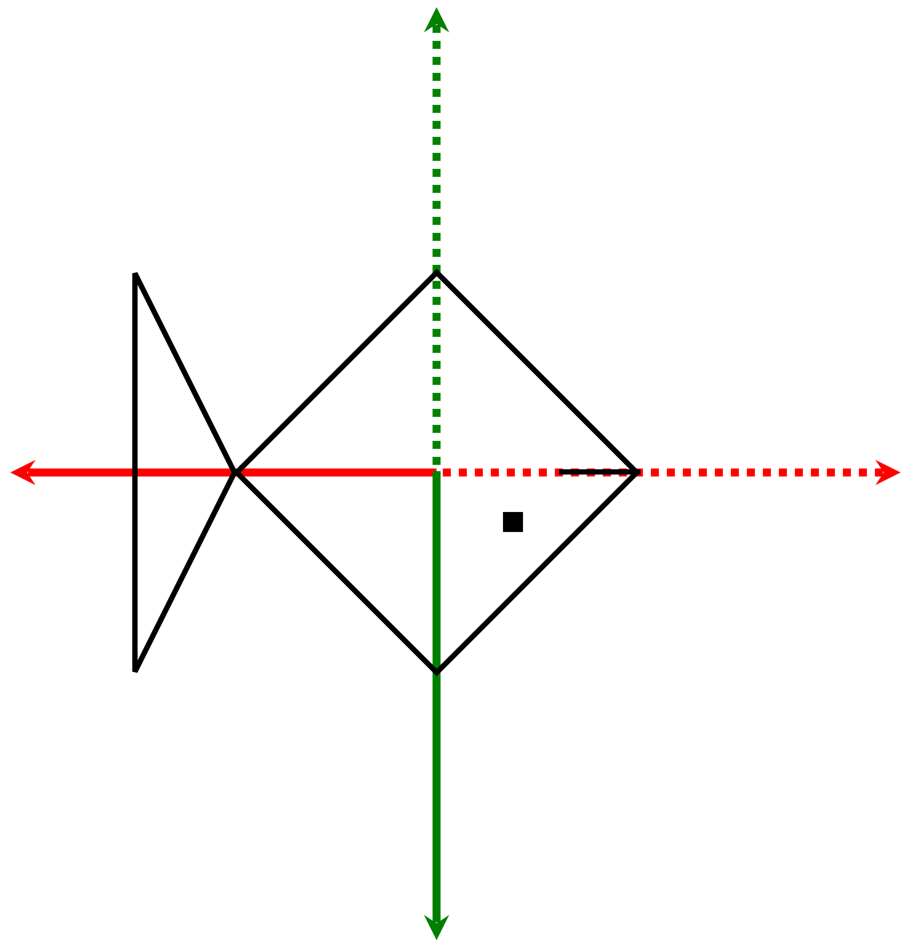
`scale(x, y)`

- or scale(0,-1)
- Flip vertically



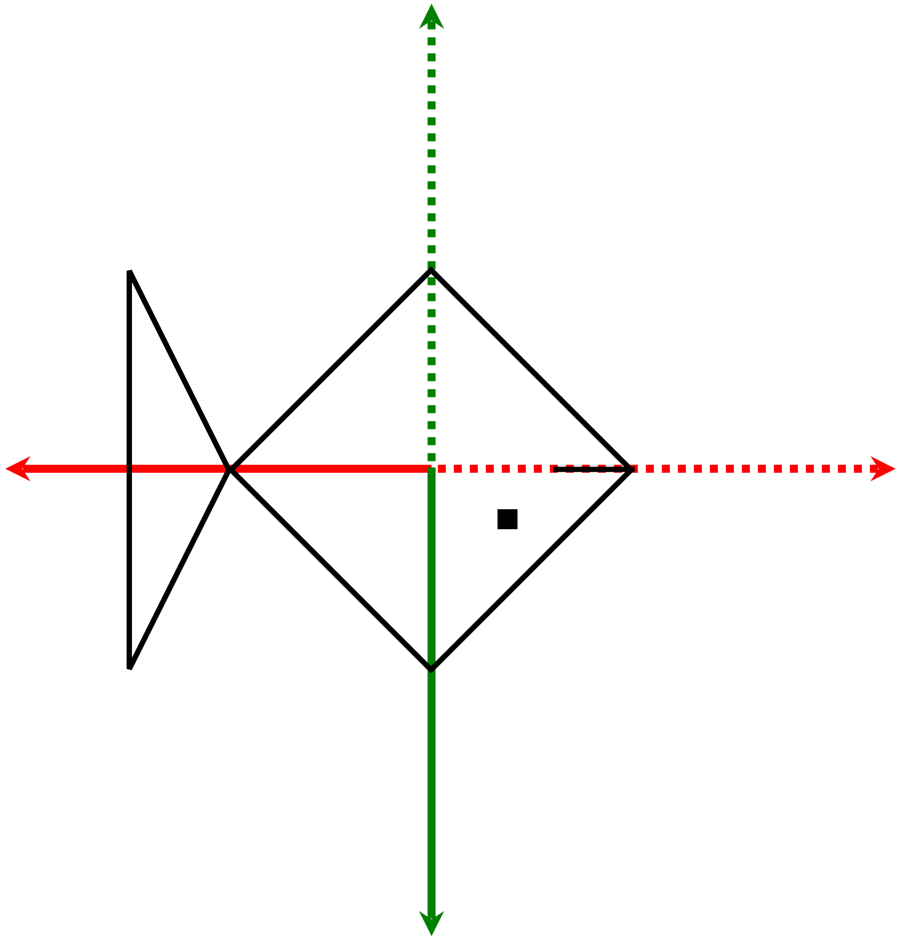
Exercise

- What transformation/s would give us this result?



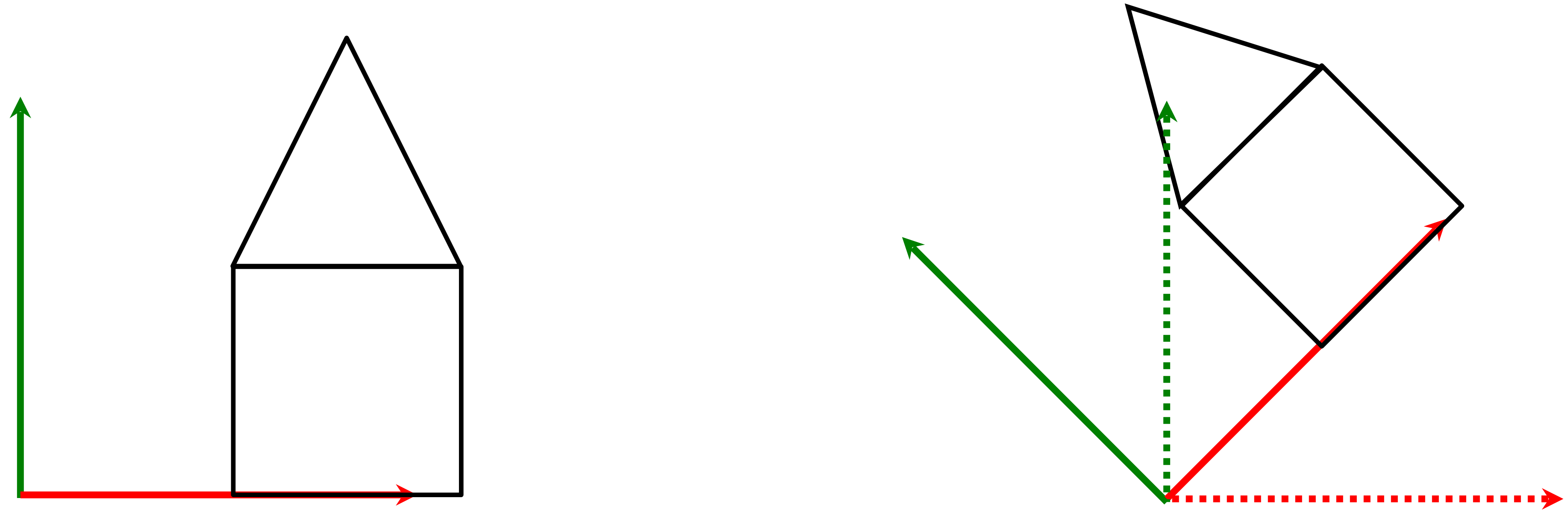
Solution

- `scale(-1,-1)`
- or `rotate(180)`
- or `rotate(-180)`



Rotation

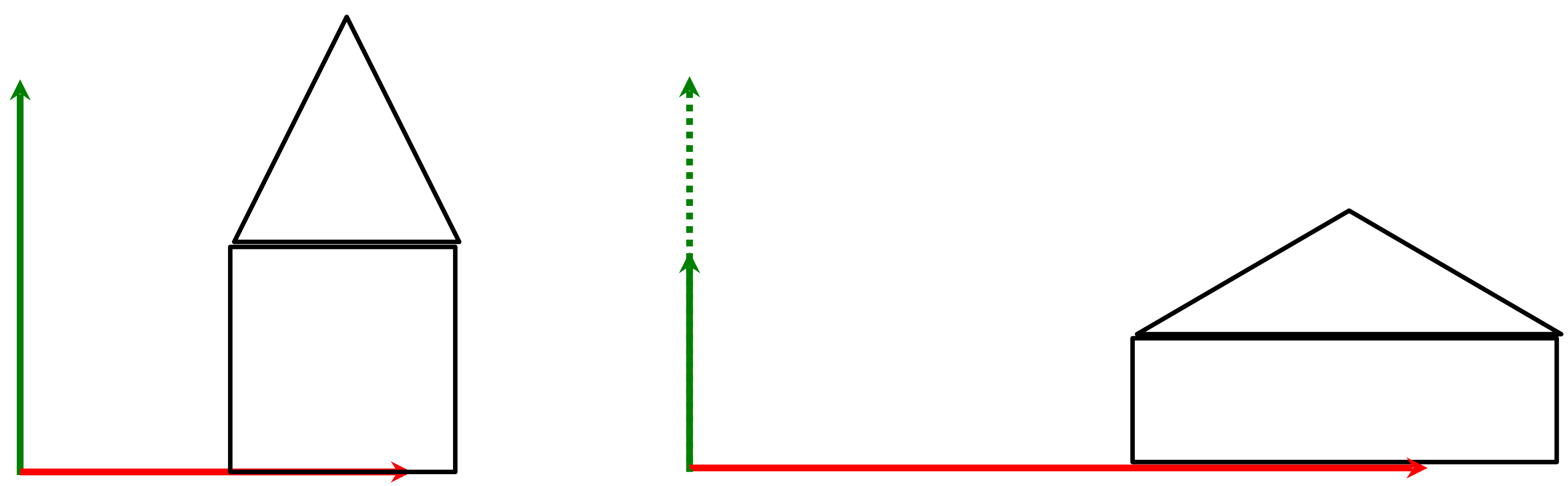
- If the object is not located at the origin, it might not do what you expect when its co-ordinate frame is rotated.



- The origin of the co-ordinate frame is the pivot point.

Rotation

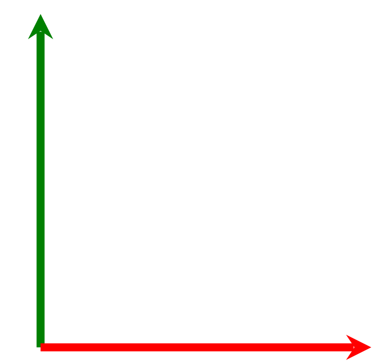
- If the object is not located at the origin, the object will move further from the origin if its co-ordinated frame is scaled



- Only points at the origin remain unchanged.

Exercise

- Draw the co-ordinate frame after each successive transformation.



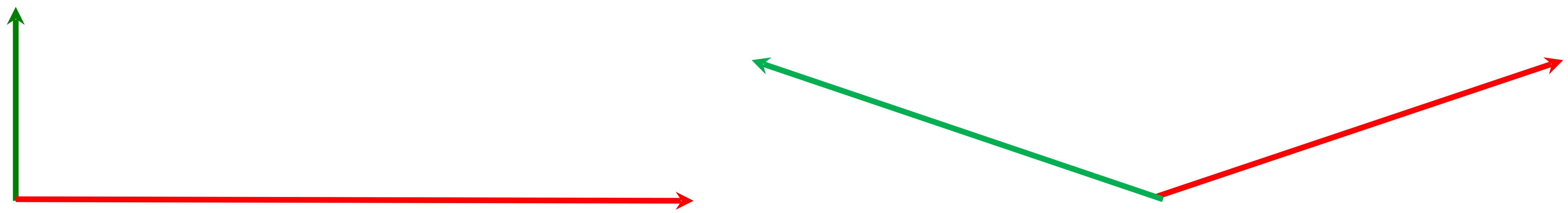
```
CoordFrame2D.identity()
    .translate(-1, 0.5)
    .rotate(90)
    .scale(1, 2)
```

Order matters

- Note that the order of transformations matters.
- translate then rotate != rotate then translate
- translate then scale != scale then translate
- rotate then scale != scale then rotate

Non-uniform Scaling then Rotating

- If we scale by different amounts in the x direction to the y direction and then rotate, we get unexpected and often unwanted results. Angles are not preserved.



Rotating about an arbitrary point.

- So far all rotations have been about the origin. To rotate about an arbitrary point.

1. Translate to the point

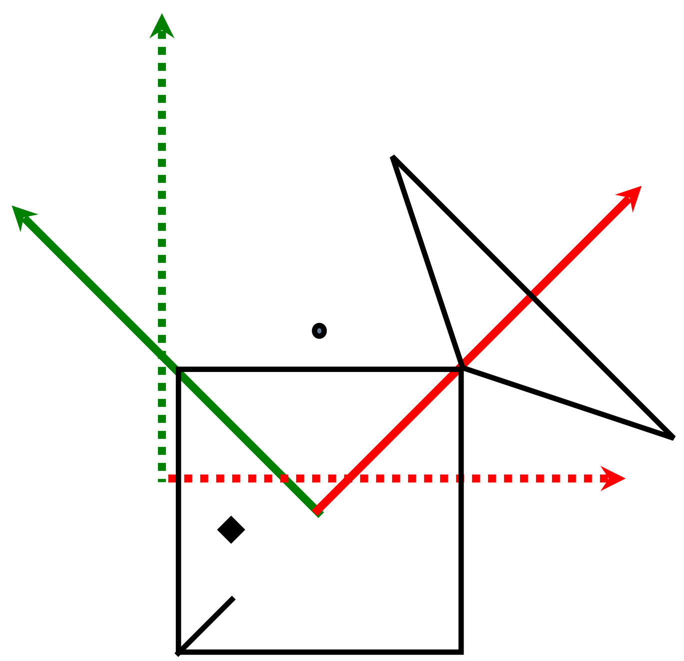
    `-translate(0.5,0.5)`

2. Rotate

    `-rotate(45)`

3. Translate back again

    `-translate(-0.5,-0.5)`

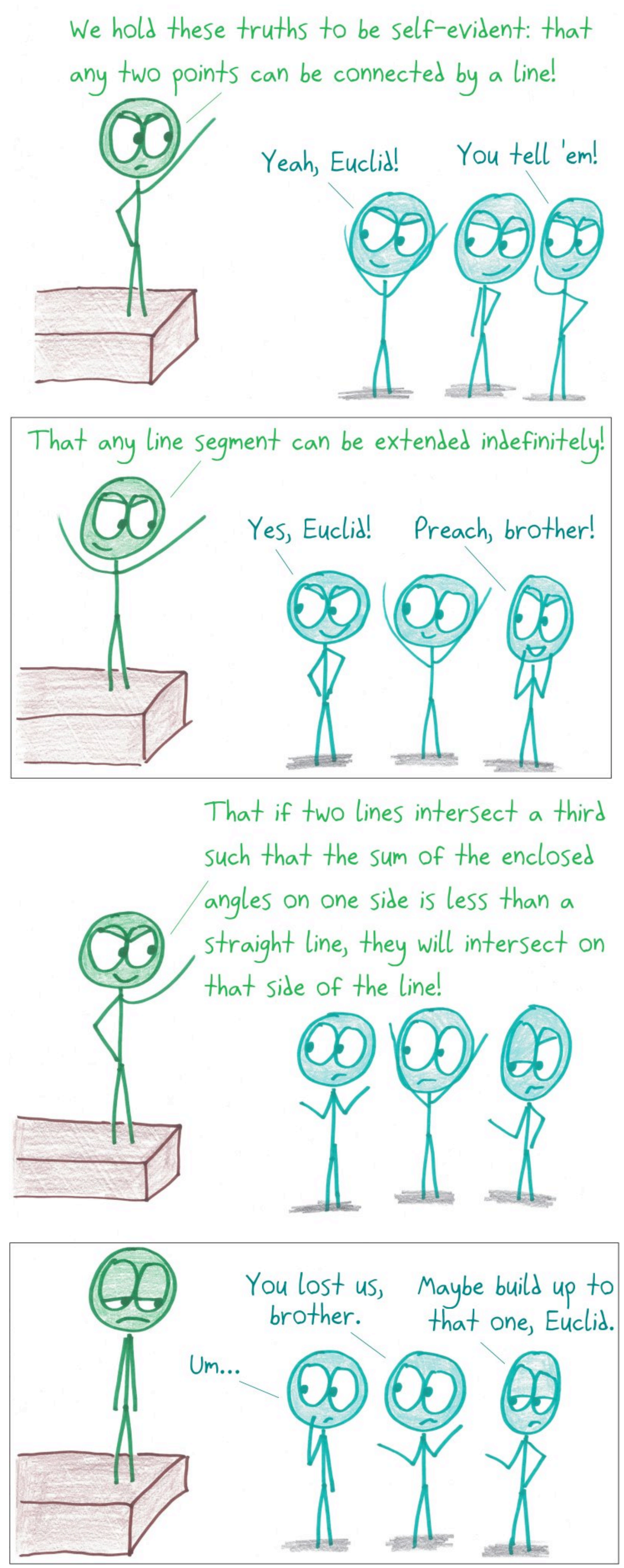


Storing history

- Often we want to store the current transformation/coordinate frame, transform it and then restore the old frame again.

- The CoordFrame2D class is immutable, so we can store intermediate frames

```
CoordFrame2D fishFrame0 = CoordFrame2D.identity().scale(0.5f, 0.5f);
CoordFrame2D fishFrame1 = fishFrame0.translate(1, -1);
CoordFrame2D fishFrame2 = fishFrame0.translate(-1, 1);
```



Source: mathwithbaddrawings.com

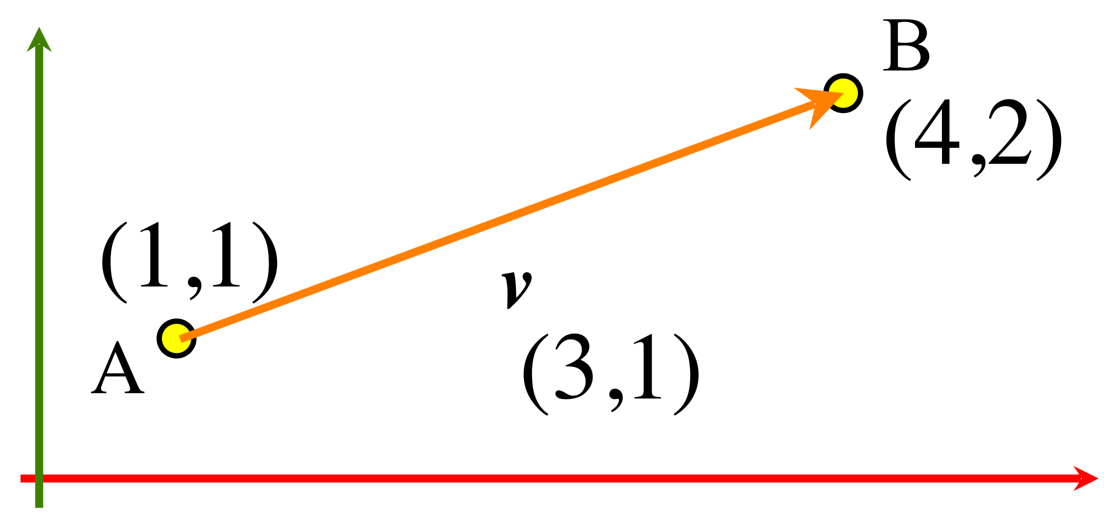
Vector and Matrix Revision

- To represent coordinate frames and easily convert points in one frame to another we use vectors and matrices.

- Some **revision** first.

Vectors

- Having the right vector tools greatly simplifies geometric reasoning.
- A vector is a displacement.



- We represent it as a tuple of values in a particular coordinate system.

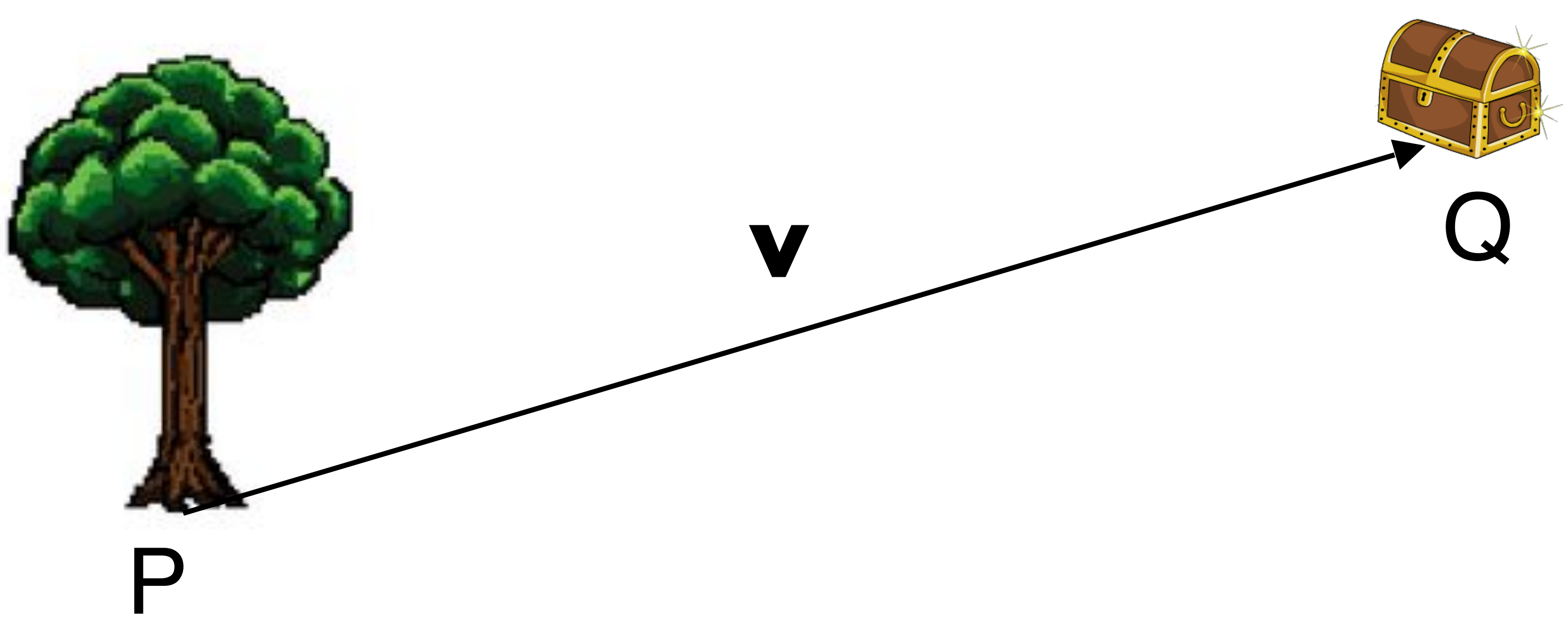
Points vs Vectors

- Vectors have
  - length and direction
  - no position
- Points have
  - position
  - no length, no direction

Points and Vectors

- The sum of a point and a vector is a point.

$P + \mathbf{v} = Q$



Points and Vectors

- The sum of a point and a vector is a point.

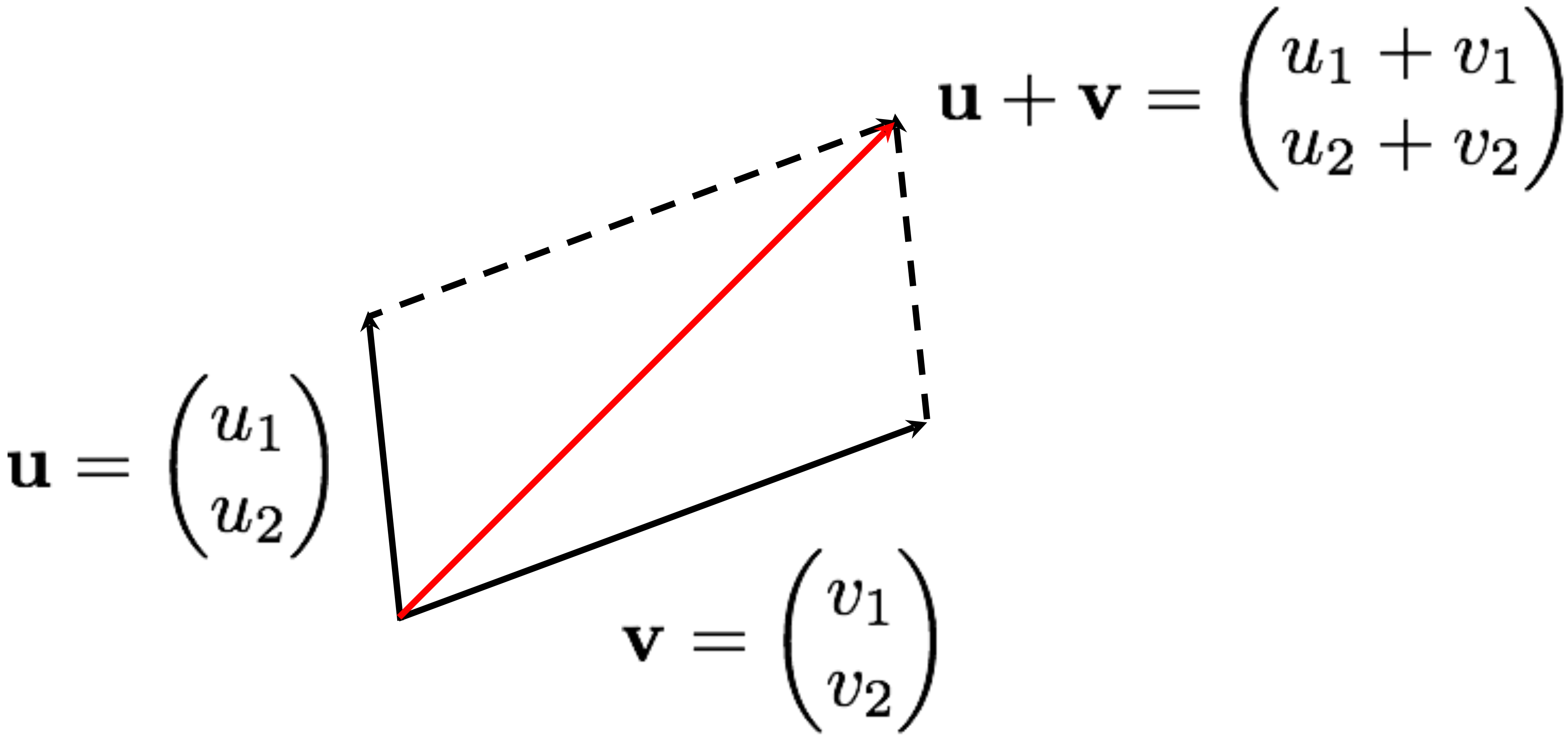
  $P + \mathbf{v} = Q$

- Which is the same as saying
  - The difference between two points is a vector:

    $\mathbf{v} = Q - P$

Adding vectors

- By adding components:

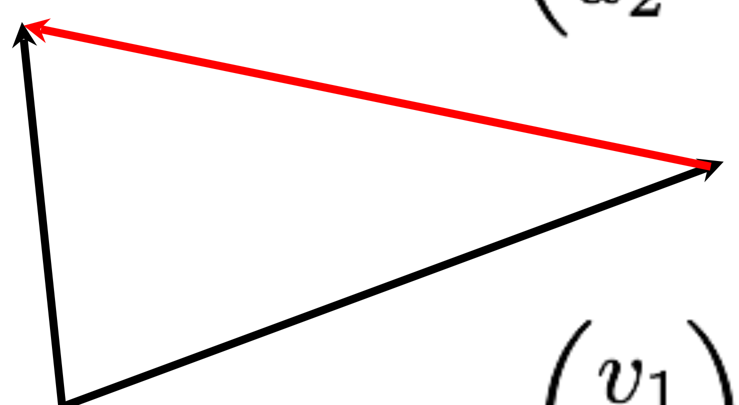


Subtracting vectors

- By subtracting components:

$$\mathbf{u} - \mathbf{v} = \begin{pmatrix} u_1 - v_1 \\ u_2 - v_2 \end{pmatrix}$$

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$



$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

Magnitude

- Magnitude (i.e. length)

$$|\mathbf{v}| = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$$

- Normalisation(i.e. direction):

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{|\mathbf{v}|}$$

$$|\hat{\mathbf{v}}| = 1$$

- Warning: You can't normalize the zero vector

Exercises

1. What is the vector **v** from P to Q if $P = (4,0)$, $Q = (1,3)$?

2. Find the magnitude of the vector $(1,2)$

3. Normalise the vector $(8,6)$

Dot product

Definition: $$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n$$

Example: $$(1,2) \cdot (-1,3) = 1 \times (-1) + 2 \times 3 = 5$$

Properties:
$$\begin{aligned}
\mathbf{u} \cdot \mathbf{v} &= \mathbf{v} \cdot \mathbf{u} \\
(a\mathbf{u}) \cdot \mathbf{v} &= a(\mathbf{u} \cdot \mathbf{v}) \\
\mathbf{u} \cdot (\mathbf{v} + \mathbf{w}) &= \mathbf{u} \cdot \mathbf{v} + \mathbf{u} \cdot \mathbf{w} \\
\mathbf{u} \cdot \mathbf{u} &= |\mathbf{u}|^2
\end{aligned}$$

Angle between vectors

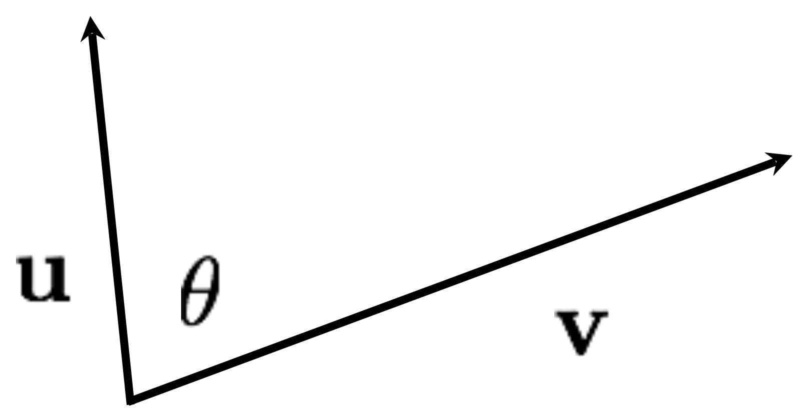
$$\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}||\mathbf{v}| \cos\theta$$

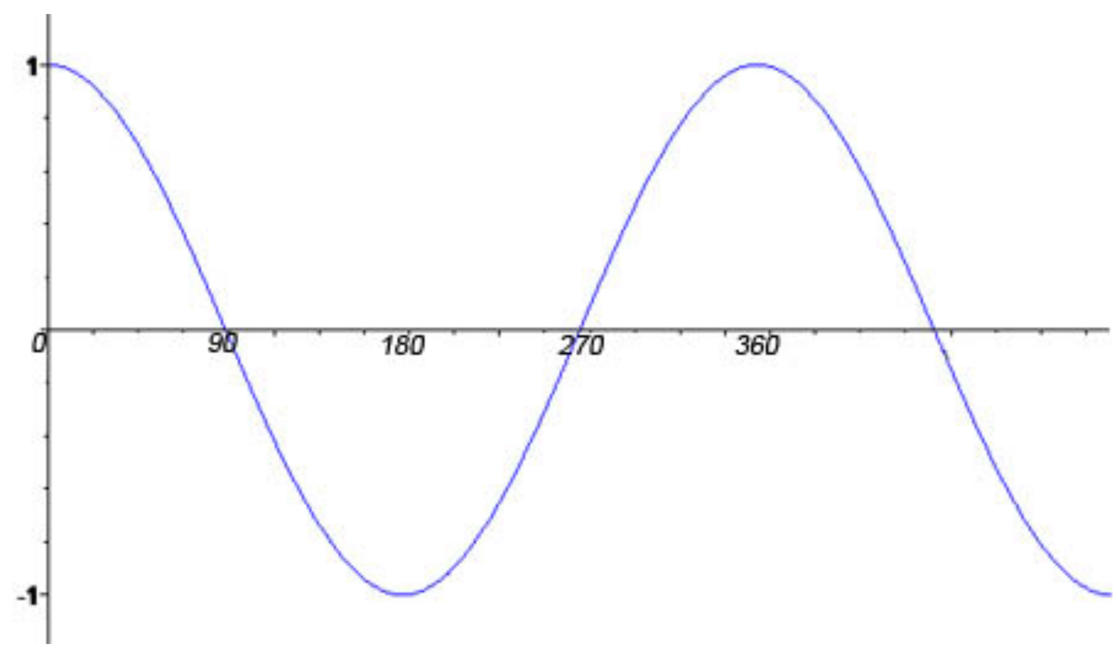
$$\cos\theta = \hat{\mathbf{u}} \cdot \hat{\mathbf{v}}$$

$$\mathbf{u} \cdot \mathbf{v} > 0 \implies \theta < 90^\circ$$

$$\mathbf{u} \cdot \mathbf{v} = 0 \implies \theta = 90^\circ$$

$$\mathbf{u} \cdot \mathbf{v} < 0 \implies \theta > 90^\circ$$





Normals in 2D

If two vectors are perpendicular, their dot product is 0.

If $n = (x_n, y_n)$ is a normal to

$$p = (x, y)$$

$$p \cdot n = x_n x + y_n y = 0$$

So, unless one is the 0 vector, either

$n = (-y, x)$ or $n = (y, -x)$

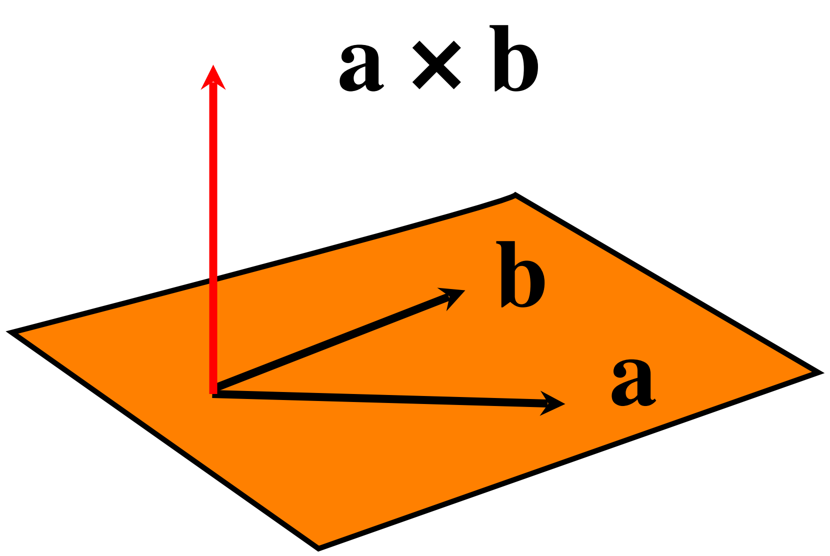
Cross product

- Only defined for 3D vectors:

$$\mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix}$$

- Properties:

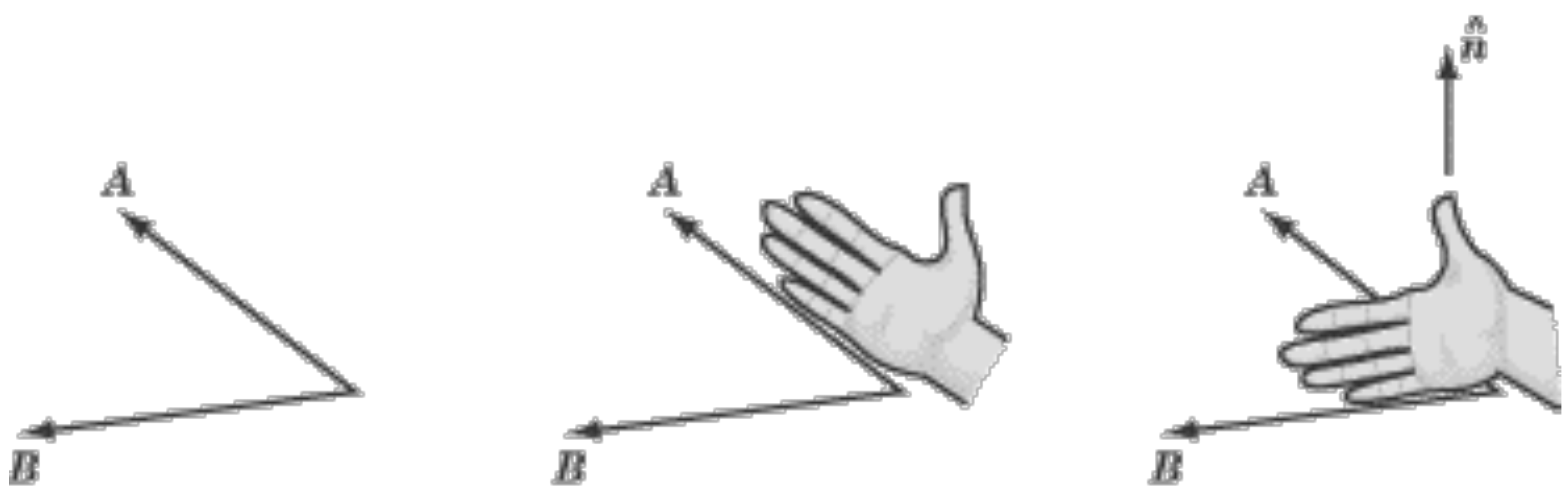
$$\mathbf{a} \times \mathbf{b} = -(\mathbf{b} \times \mathbf{a})$$
$$\mathbf{a} \cdot (\mathbf{a} \times \mathbf{b}) = 0$$
$$\mathbf{b} \cdot (\mathbf{a} \times \mathbf{b}) = 0$$



- Can use to find normals (more on this in later weeks)

AxB vs BxA

- Assuming a right handed co-ordinate system: to find the direction of AxB curl fingers of your right hand from A to B and your thumb shows the direction. BxA would be in the opposite direction.



Determinant form

- For this who know, the cross product can be defined as a determinant of a matrix.

$$a \times b = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

- It is not necessary to understand determinants in this course

Memory Aid

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} \\ \\ \end{pmatrix}$$

Memory Aid

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 \\ \\ \\ \end{pmatrix}$$

Memory Aid

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ \\ \\ \end{pmatrix}$$

Memory Aid

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 \\ \end{pmatrix}$$

Memory Aid

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ \end{pmatrix}$$

Memory Aid

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 \end{pmatrix}$$

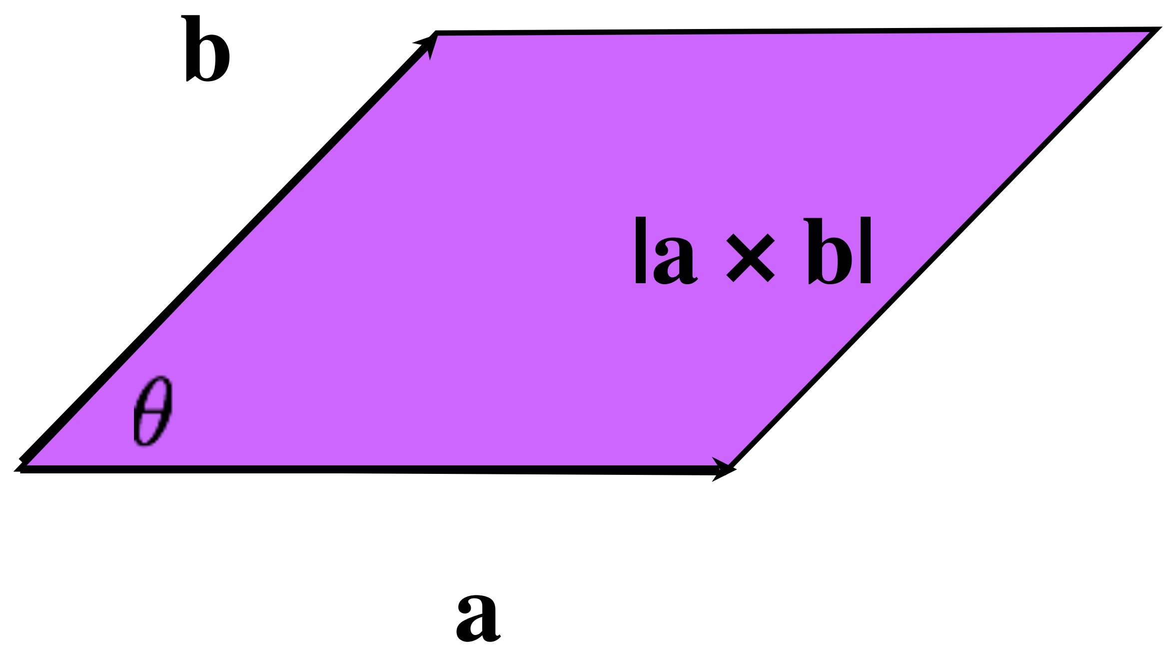
Memory Aid

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix}$$

Cross product

- The magnitude of the cross product is the area of the parallelogram formed by the vectors:

$$|\mathbf{a} \times \mathbf{b}| = |\mathbf{a}||\mathbf{b}| \sin \theta$$



Exercises

1. Find the angle between vectors (1,1) and (-1,-1)

2. Is vector (3,4) perpendicular to (2,1)?

3. Find a vector perpendicular to vector $\mathbf{a}$ where $\mathbf{a} = (2,1)$

4. Find a vector perpendicular to vectors $\mathbf{a}$ and $\mathbf{b}$ where $\mathbf{a} = (3,0,2)$ $\mathbf{b} = (4,1,8)$

Matrices

We can think of a matrix as a 2D array of numbers

$$\begin{pmatrix} 1 & 0 & 3 \\ 2 & 3 & 4 \\ 0 & 0 & 1 \end{pmatrix}$$

And vectors as a matrix with a single column

$$\begin{pmatrix} 2 \\ 3 \\ 0 \end{pmatrix}$$

Matrix multiplication

$$\begin{pmatrix} 1 & 0 & 3 \\ 2 & 3 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 2 \end{pmatrix} = \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}$$

Matrix multiplication

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{3} \\ 2 & 3 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{2} & 1 & 1 \\ \mathbf{0} & 0 & 1 \\ \mathbf{1} & 1 & 2 \end{pmatrix} = \begin{pmatrix} ? & & \\ & & \\ & & \end{pmatrix}$$

$$1 \times 2 + 0 \times 0 + 3 \times 1 = 5$$

Matrix multiplication

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{3} \\ 2 & 3 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{2} & 1 & 1 \\ \mathbf{0} & 0 & 1 \\ \mathbf{1} & 1 & 2 \end{pmatrix} = \begin{pmatrix} 5 & & \\ & & \\ & & \end{pmatrix}$$

Matrix multiplication

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{3} \\ 2 & 3 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & \mathbf{1} & 1 \\ 0 & \mathbf{0} & 1 \\ 1 & \mathbf{1} & 2 \end{pmatrix} = \begin{pmatrix} 5 & ? & \\ & & \\ & & \end{pmatrix}$$

Matrix multiplication

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{3} \\ 2 & 3 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & \mathbf{1} & 1 \\ 0 & \mathbf{0} & 1 \\ 1 & \mathbf{1} & 2 \end{pmatrix} = \begin{pmatrix} 5 & 4 & \\ & & \\ & & \end{pmatrix}$$

Matrix multiplication

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{3} \\ 2 & 3 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & \mathbf{1} \\ 0 & 0 & \mathbf{1} \\ 1 & 1 & \mathbf{2} \end{pmatrix} = \begin{pmatrix} 5 & 4 & ? \\ & & \\ & & \end{pmatrix}$$

Matrix multiplication

$$\begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{3} \\ 2 & 3 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & \mathbf{1} \\ 0 & 0 & \mathbf{1} \\ 1 & 1 & \mathbf{2} \end{pmatrix} = \begin{pmatrix} 5 & 4 & 7 \\ & & \\ & & \end{pmatrix}$$

Matrix multiplication

$$\begin{pmatrix} 1 & 0 & 3 \\ \mathbf{2} & \mathbf{3} & \mathbf{4} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{2} & 1 & 1 \\ \mathbf{0} & 0 & 1 \\ \mathbf{1} & 1 & 2 \end{pmatrix} = \begin{pmatrix} 5 & 4 & 7 \\ ? & & \\ & & \end{pmatrix}$$

Matrix multiplication

$$\begin{pmatrix} 1 & 0 & 3 \\ \mathbf{2} & \mathbf{3} & \mathbf{4} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{2} & 1 & 1 \\ \mathbf{0} & 0 & 1 \\ \mathbf{1} & 1 & 2 \end{pmatrix} = \begin{pmatrix} 5 & 4 & 7 \\ 8 & & \\ & & \end{pmatrix}$$

And so on…

Matrix multiplication

$$\begin{pmatrix} 1 & 0 & 3 \\ 2 & 3 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 5 & 4 & 7 \\ 8 & 6 & 13 \\ 1 & 1 & 2 \end{pmatrix}$$

Homework

- Revise basics of vectors and matrix multiplication if you need to as we will use them extensively from next week on.