Lecture 7

Recap

- HTML Canvas
  - Comparison to SVG browser graphics
  - Drawing shapes
  - Basic Trigonometry
  - Basic collision detection
  - Animation and User Interaction
    - Examples
  - Advanced collision handling

Today

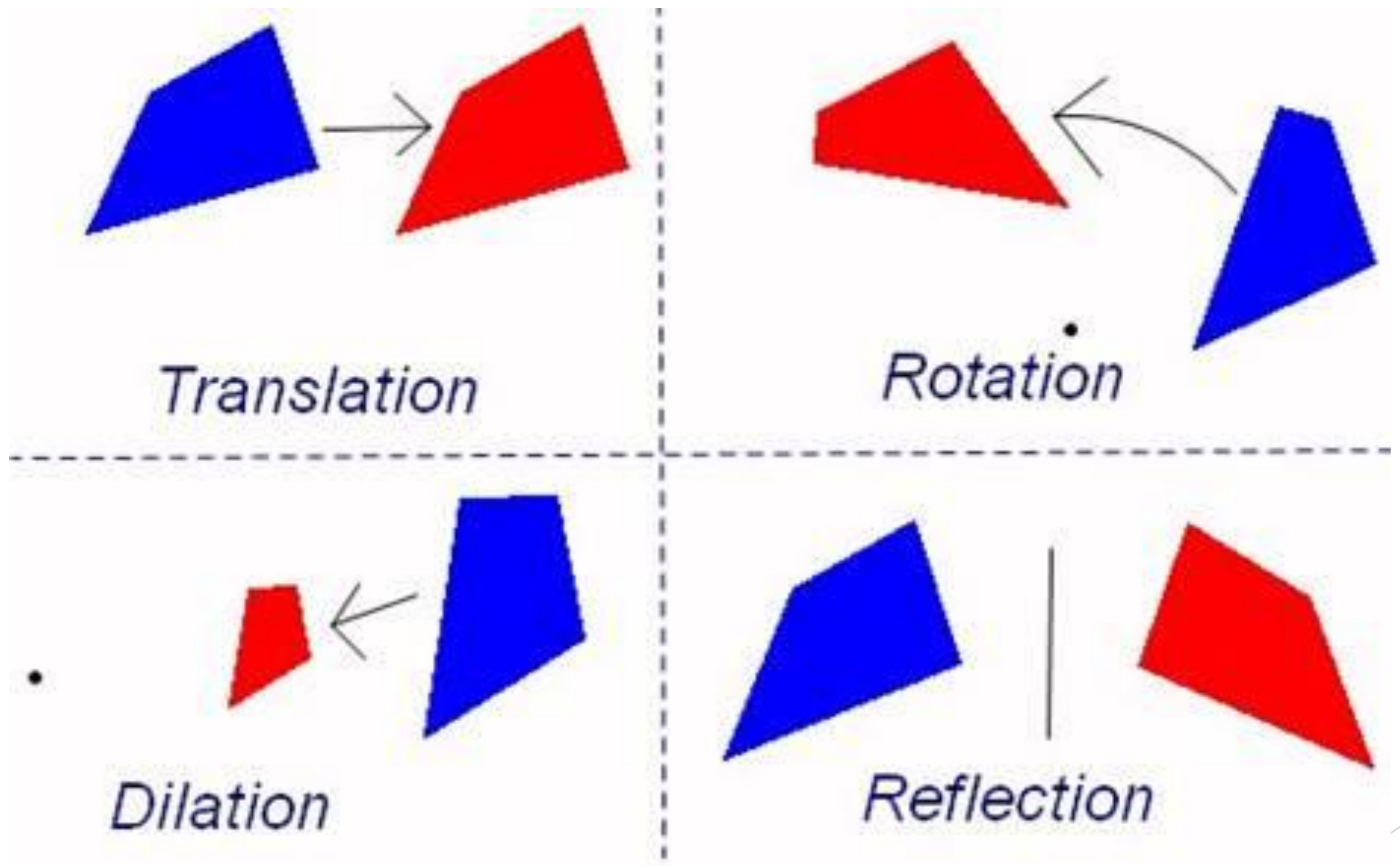
- Linear Algebra and Transformations

Labs

- Bouncing/Colliding Balls/Orbits
- Submission of Labs 4, 5 and 6 on 5th November at 23:00

- Following that moving onto SVG, D3.js

Transformations



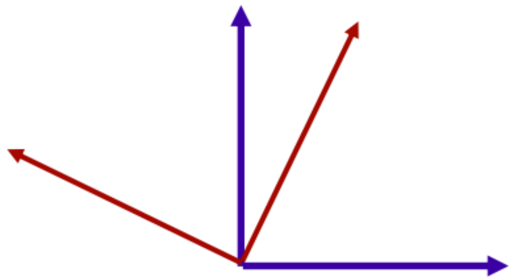
Coordinate systems

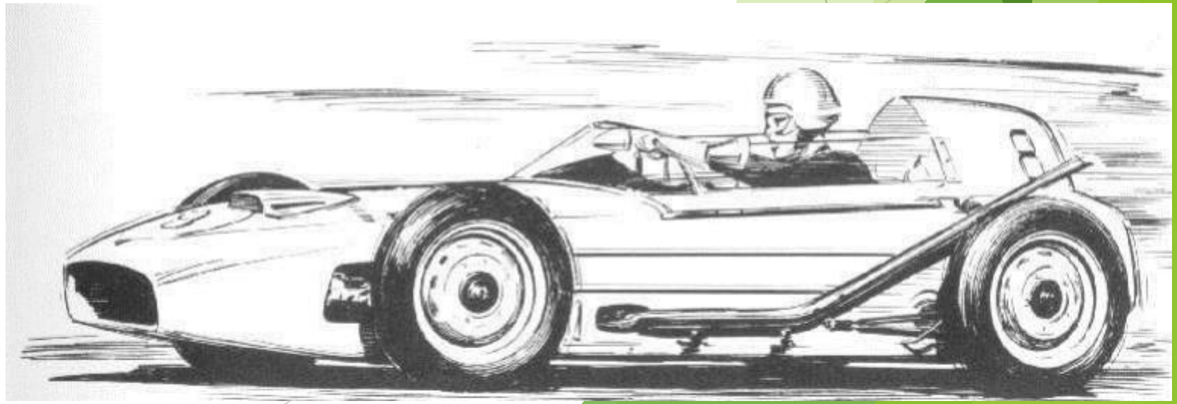
- Many coordinate systems:
  - Origin relative to:
    - Car
    - Driver
    - Arm
- Points represented as tuples (x1,y1)
  - Tuples are meaningless without a clear coordinate system

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

could be this point in the red coordinate system ●

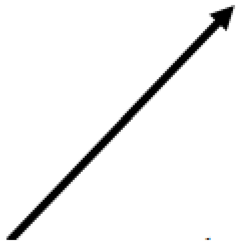
could be this point in the blue coordinate system ●





Coordinate systems

- Points
  - Represent locations
- Vectors
  - Represent movement, force, displacement from A to B
- Normals
  - Represent orientation, unit length
- Coordinates
  - Numerical representation of the above objects **in a given coordinate system**



$$\begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Point – Formal Definition

- A scalar represents magnitude, and is given by a real number – a,b,…,x,y
- A **point** in n-dimensional space is given by an *n*-tuple $P = (p_1, p_2, \ldots, p_n)$ where each coordinate p_i is a scalar number.
  - We will write $P = (p_i)$ as a shorthand for this n-tuple. The position of a point is relative to a coordinate system with an origin $0 = (0,0,\ldots,0)$
- Thus, a 3-dimensional (3D) point can be given by a triple $P = (p_1, p_2, p_3)$ whose coordinates are relative to the axes (1,0,0), (0,1,0), and (0,0,1) which are commonly referred to as the *x*, *y*, and *z*-axes.
  - Because of this established convention, we sometimes write $P = (x, y)$ for 2D points and $P = (x, y, z)$ for 3D points.

Vectors – Formal Definition

- A **vector** represents magnitude and direction in space, and is given by an n-tuple $V = (v_1, v_2, \dots, v_n)$ where each coordinate v_i is a scalar.
  - We also write $v = (v_i)$ as a shorthand for the vector's n-tuple.
  - The vector **v** is interpreted to be the magnitude and direction of the line segment going from the origin $0 = (0, 0, \dots, 0)$ to the point $V = (v_1, v_2, \dots, v_n)$. However, the vector is not this point. It only gives a standard way of visualizing the vector.
  - We can also visualize the vector as a directed line segment from any initial point $P = (p_i)$ to a final point $Q = (q_i)$. Then, the vector from *P* to *Q* is given by:

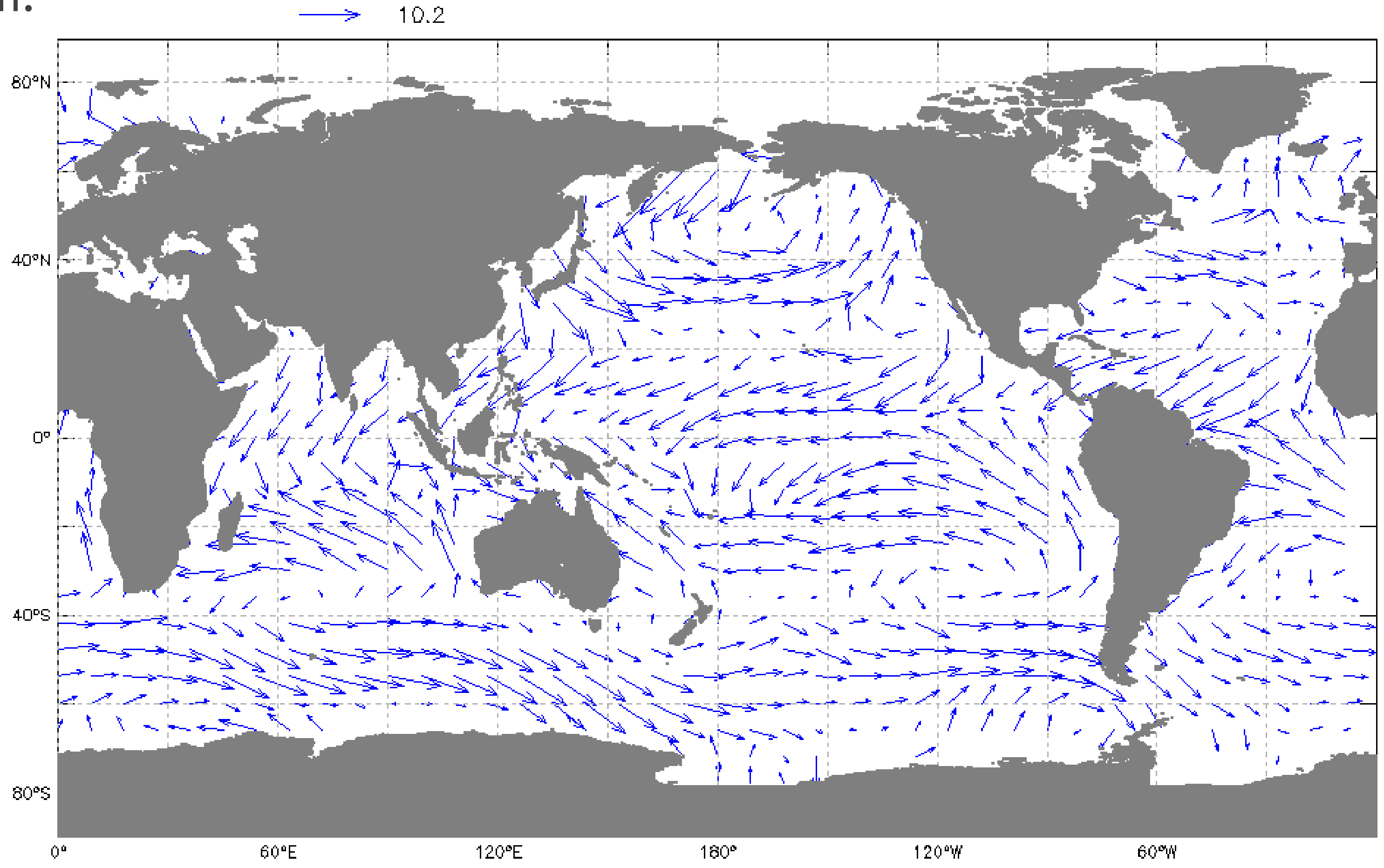
Point Difference Definition $$\mathbf{v} = Q - P = (q_i - p_i)$$

showing that the difference of any two points is considered to be a vector.

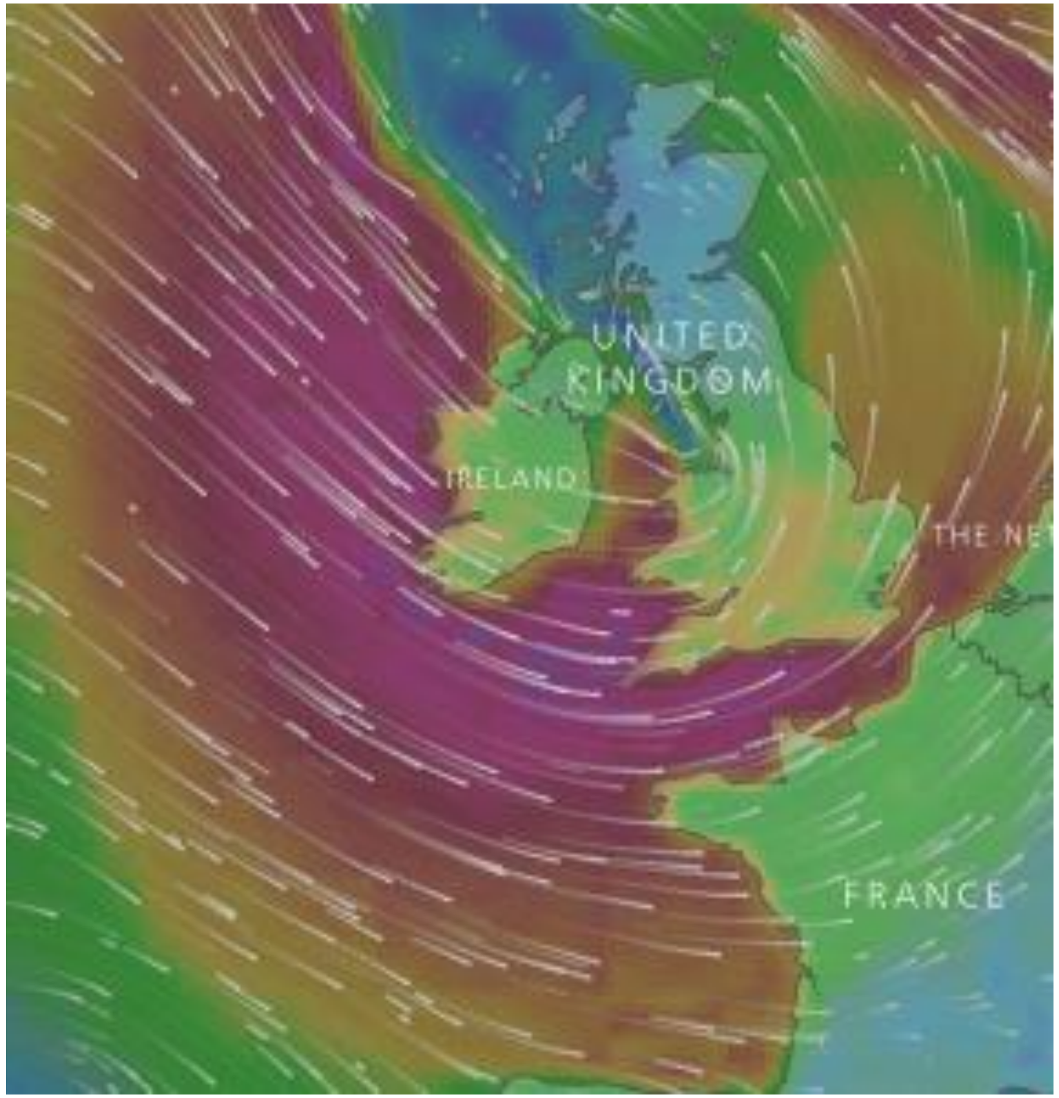
- So, vectors do not have a fixed position in space, but can be located at any initial base point P. For example, a traveling vehicle can be said to be going east (direction) at 50 mph (magnitude) no matter where it is located.

Vectors – Formal Definition

- We could even visualize a field of vectors, one at each point of a space; such as vectors for the wind direction and speed at each point on the surface of the earth.

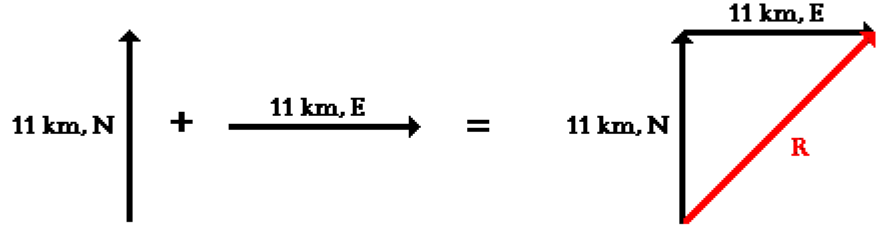


Storm Brian

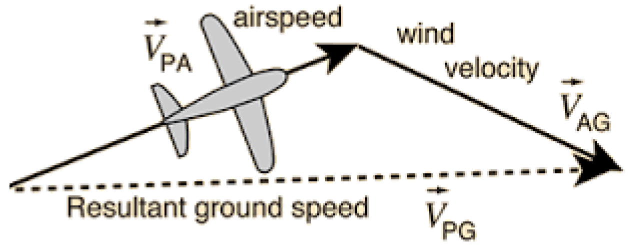


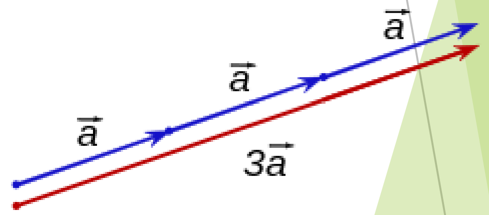
Vectors – Formal Definition

- Vectors can be added together and multiplied by a scalar (changing their length)



$$11^2 + 11^2 = R^2$$
$$242 = R^2$$
$$15.6 = R$$



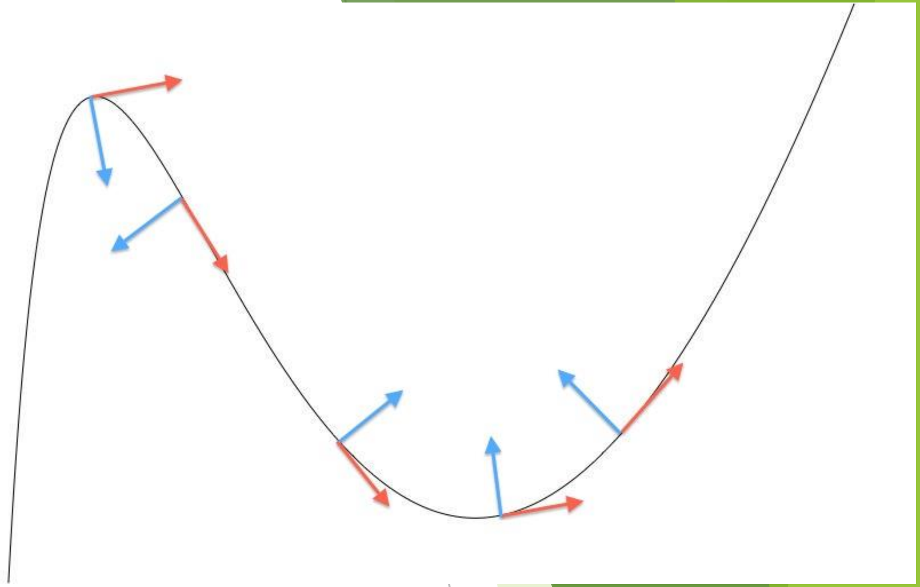


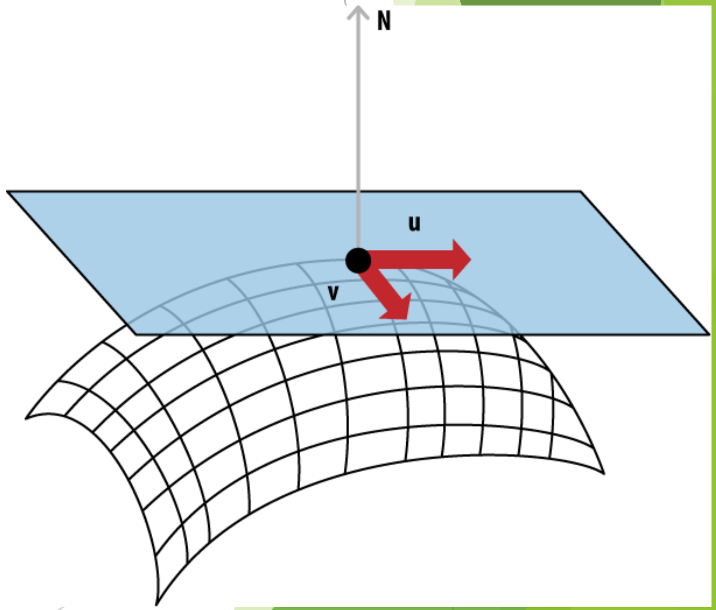
Vectors – Formal Definition

- Vectors can be added together and multiplied by a scalar (changing their length)
- There is a special element, the zero vector
  - No displacement, no force
- The length of a vector v is denoted by $|v|$, and is defined as: **Vector Length Definition** $$|\mathbf{v}|^2 = \sum_{i=1}^{n} v_i^2$$
  - This gives the standard Euclidean geometry (Pythagorean) length for the line segment representing a vector. For a 2D vector $\mathrm{V} = (v_1, v_2)$, one has: $|v|^2 = {v_1}^2 + {v_2}^2$, which is the Pythagorean theorem for the diagonal of a rectangle.

Normal Vector

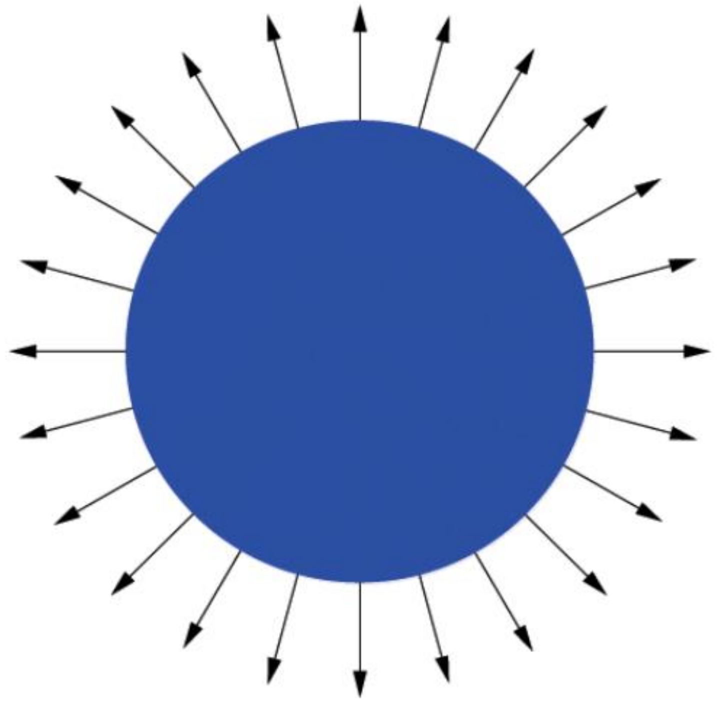
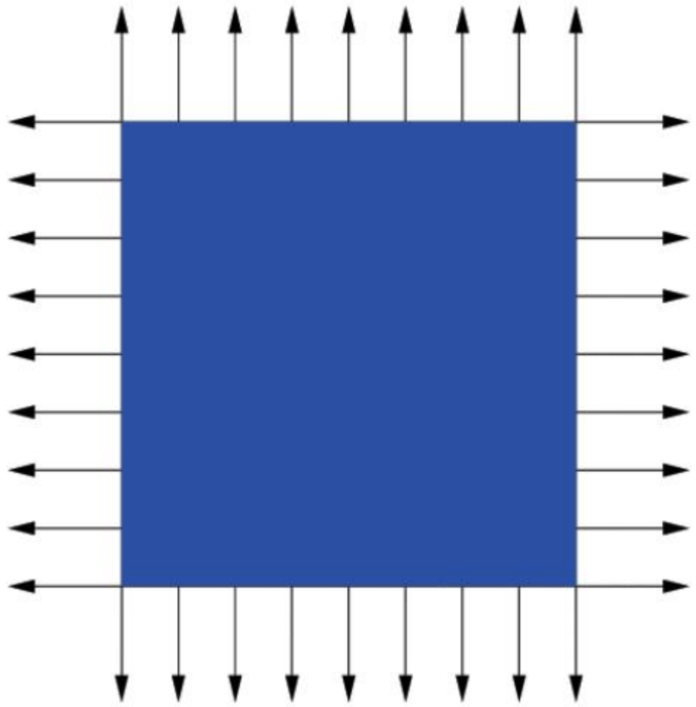
- A normal vector is one that is of unit length and is perpendicular to a given object.
- Recall – a tangent at a given point is the straight line/surface that "just touches" the curve/surface at that point.
- In 2d:
  - The normal vector to a curve at a given point is the line perpendicular to the tangent line to the curve at the point.
- In 3d:
  - The normal vector to a surface at a given point is the line perpendicular to the tangent plane to that surface at the point.
- Application:
  - The normal is often used in computer graphics to determine a surface's orientation toward a light source for flat shading
  - Makes things look 3D!





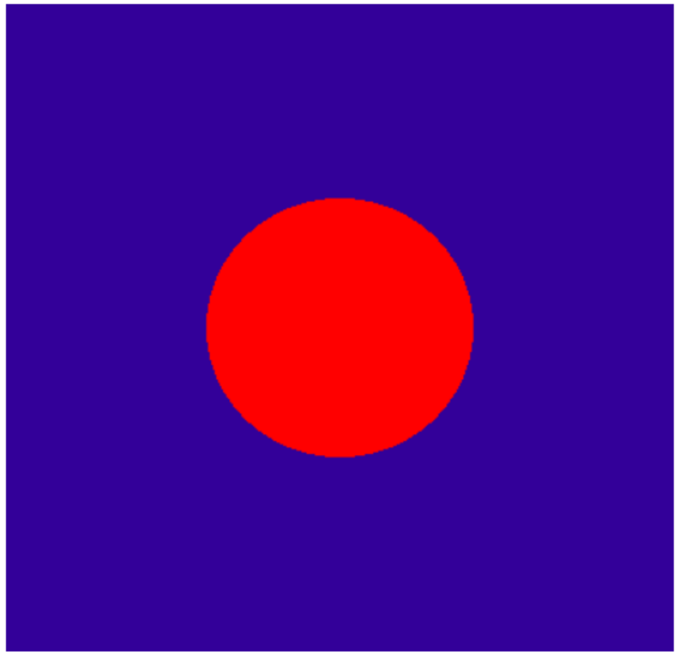
Normal Vectors - Illustration

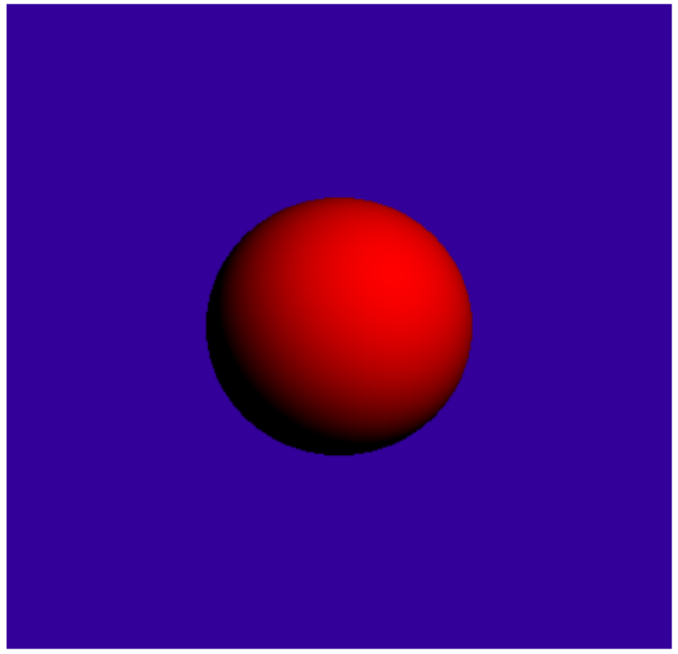
- Surface Normal: unit vector that is locally perpendicular to the surface

Why is the Normal important?

- It's used for shading — makes things look 3D!


object color only


Diffuse Shading

Difference between Points & vectors

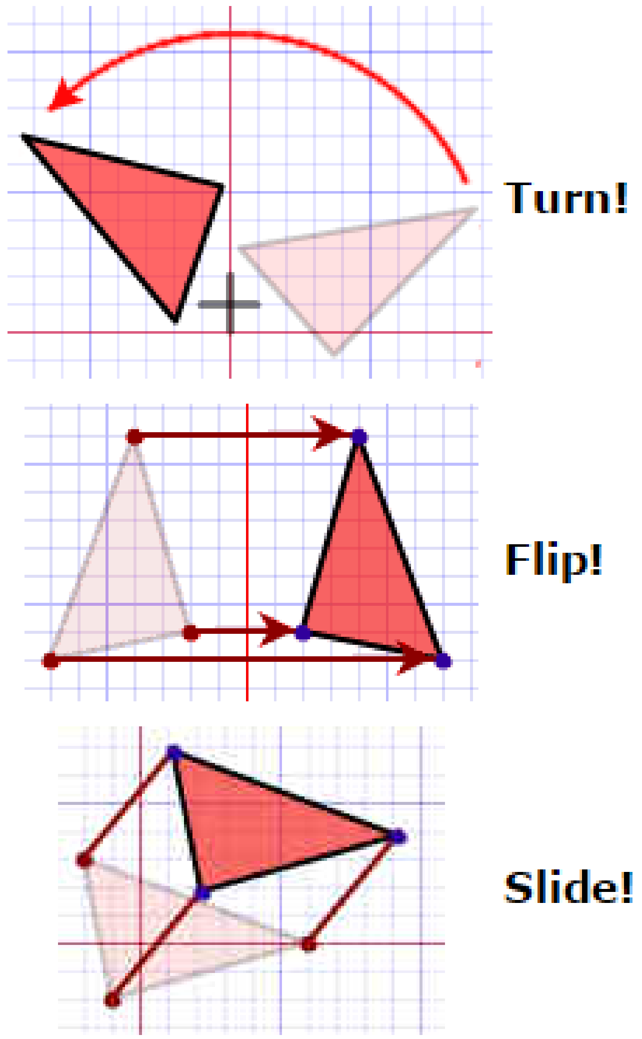
- A point is a location
- A vector is a motion between two points
- Adding vectors is meaningful
  - Going 3km North + 4km East = going 5km North-East
- Adding points is not meaningful
  - Galway city centre + Cork city centre = ?
- Multiplying a vector by a scalar changes the length of a vector
- Multiplying a point by a scalar is meaningless
- The 0 vector has a fundamental meaning: no movement, no force
- The 0 point is used as a point to reference from

Difference between Points & vectors

- It’s meaningful to add vectors, not points
- Adding a vector to a point gives another point
- Vectors are invariant under translation while points are not.
- Example - Moving car
  - Points describe location of car elements
  - Vectors describe velocity of a point/distance between pairs of points
- If the moving car is translated to somewhere else
  - The points (location) change
  - The vectors (speed, distance between points) don’t

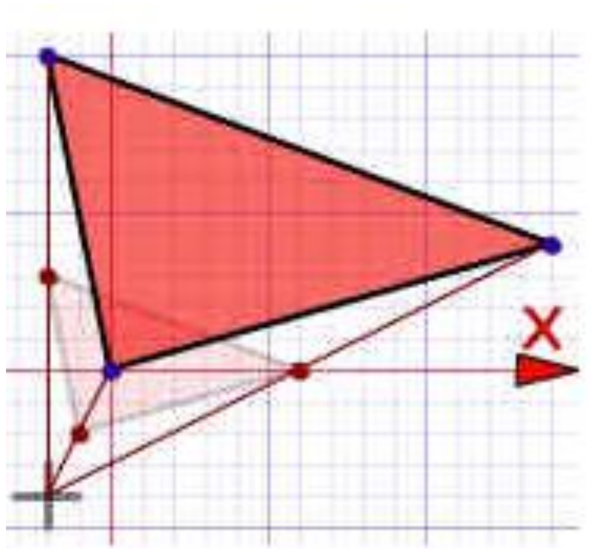
Transformations

- These are Transformations:
  - Rotation
  - Reflection
  - Translation
- After any of these transformations (turn, flip or slide), the shape still has **the same size**, **area**, **angles** and **line lengths.**



Scaling/Resizing

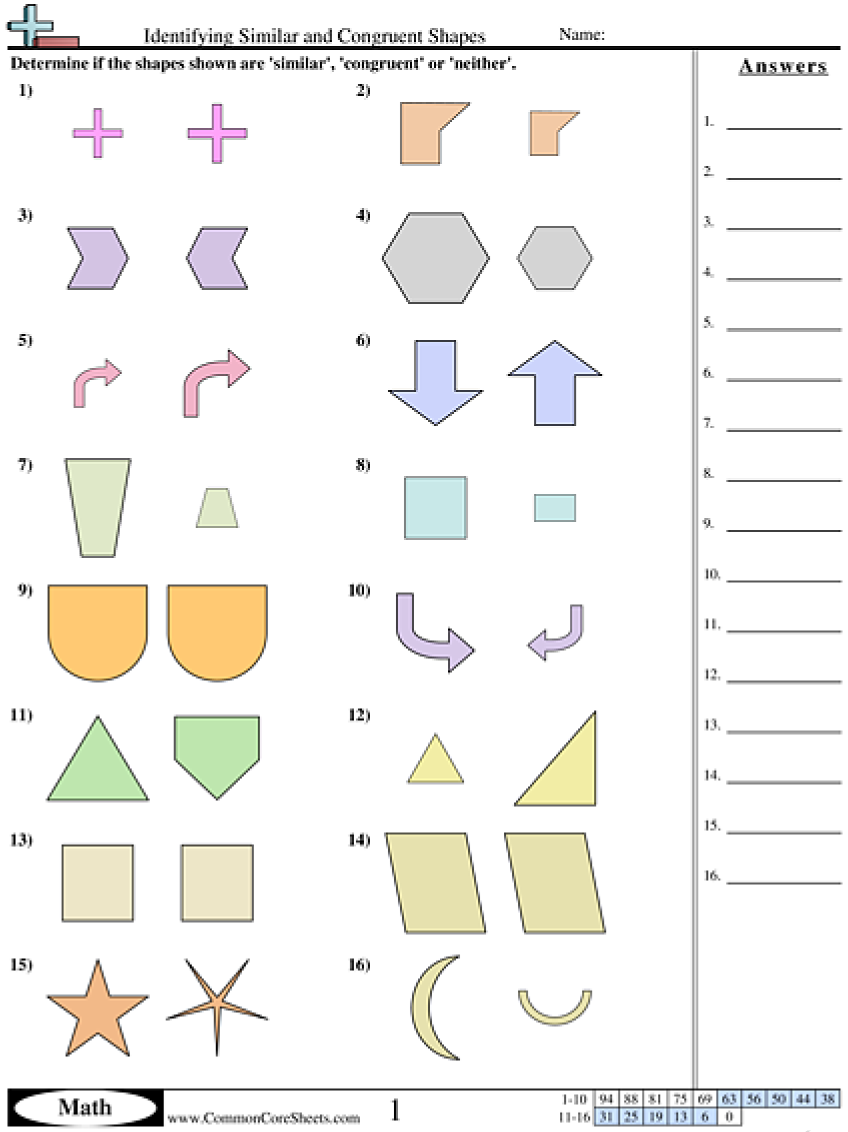
- The other important Transformation is Scaling (also called *resizing*, *dilation*, *contraction*, *compression*, *enlargement* or even *expansion*).
- The shape becomes bigger or smaller:



Congruence and Similarity

- If one shape can become another using only Rotation, Reflection and/or Translation, then the two shapes are called **Congruent**.
- Two shapes are Similar if you need to Scale one shape to become another (you may also Rotate, Reflect or Translate).
- So, if one shape can become another using transformation, the two shapes might be Congruent or just Similar

Congruence and Similarity



Identifying Similar and Congruent Shapes Name:

Determine if the shapes shown are 'similar', 'congruent' or 'neither'.

1) 2) 3) 4) 5) 6) 7) 8) 9) 10) 11) 12) 13) 14) 15) 16)

Answers

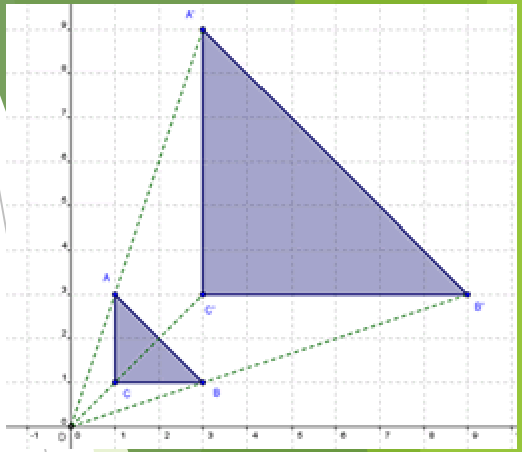
1. ____________
2. ____________
3. ____________
4. ____________
5. ____________
6. ____________
7. ____________
8. ____________
9. ____________
10. ____________
11. ____________
12. ____________
13. ____________
14. ____________
15. ____________
16. ____________

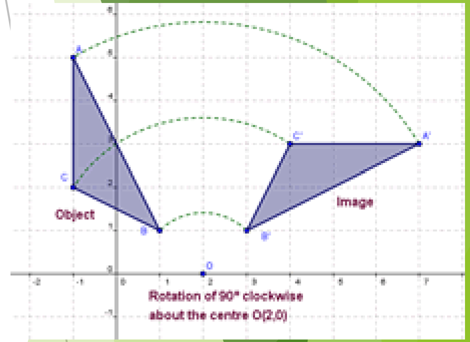
Math www.CommonCoreSheets.com 1

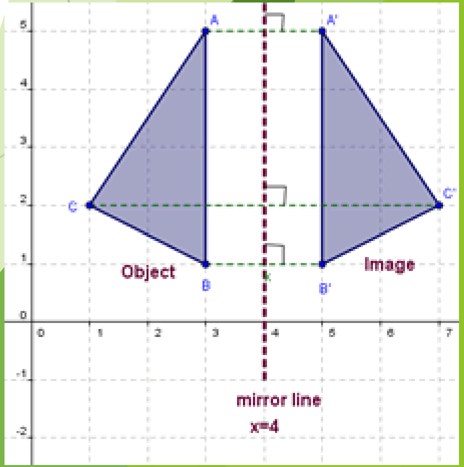
1-10	94	88	81	75	69	63	56	50	44	38
11-16	31	25	19	13	6	0				

Transformation Definitions

- Transformations are operations that apply to all the points in the coordinate system.
- Translation (2d)
  - Move all of the points by given amount in the x and y directions
- Scaling
  - Scale the location of each point by given amounts in both directions. This has the effect of changing what a unit means in each direction
- Rotation
  - Rotate each point about a reference point (known as the Centre of Rotation) by a given angle.
- Reflection
  - Reflection in a line produces a mirror image in which corresponding points on the original shape and the mirror image are always the same distance from the mirror line.

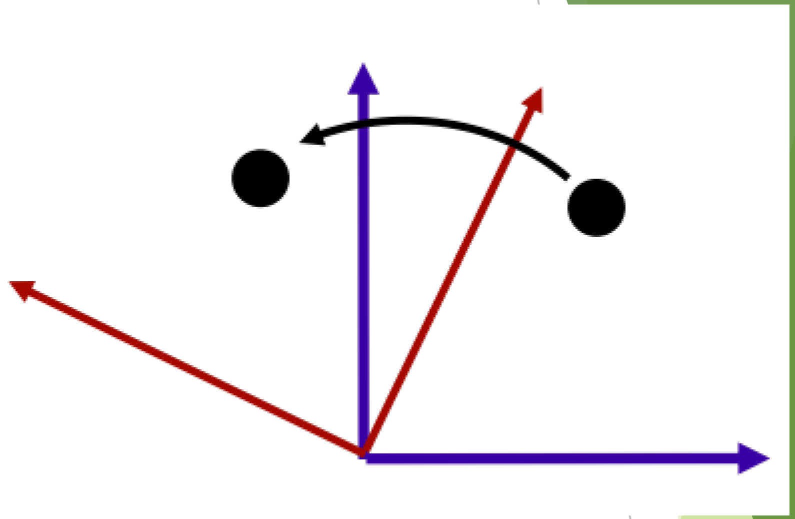


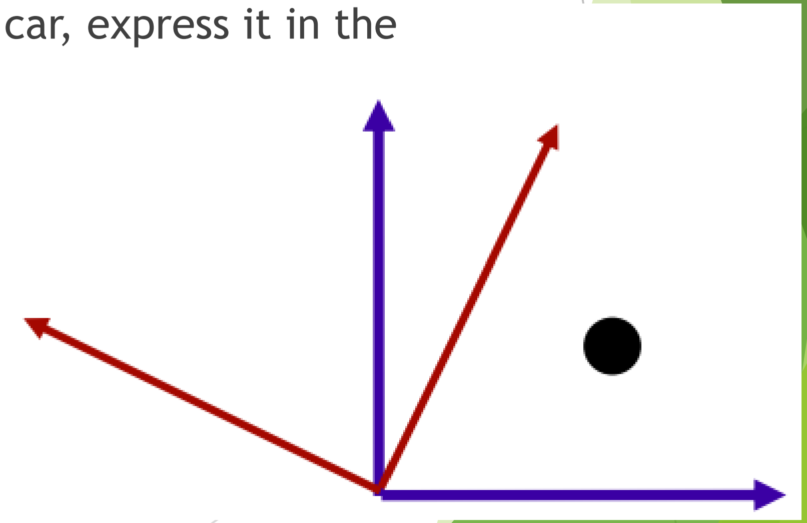




Matrices have two purposes

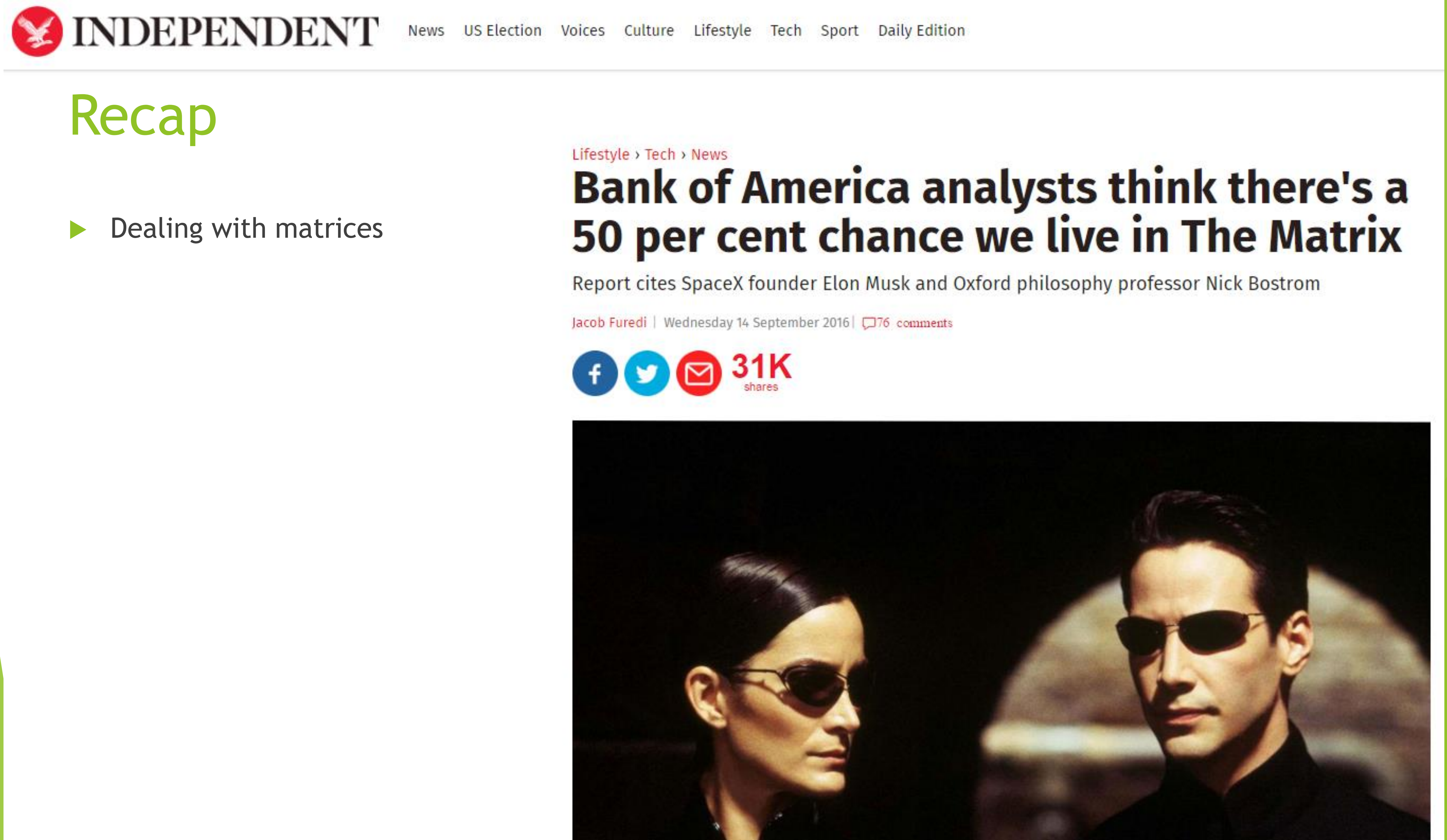
- (At least for geometry)
- Transform things
  - E.g. rotate the car from facing North to facing East
- Express coordinate system changes
  - E.g. given the driver's location in the coordinate system of the car, express it in the coordinate system of the world





Recap

- Dealing with matrices



INDEPENDENT News US Election Voices Culture Lifestyle Tech Sport Daily Edition

Lifestyle › Tech › News

Bank of America analysts think there's a 50 per cent chance we live in The Matrix

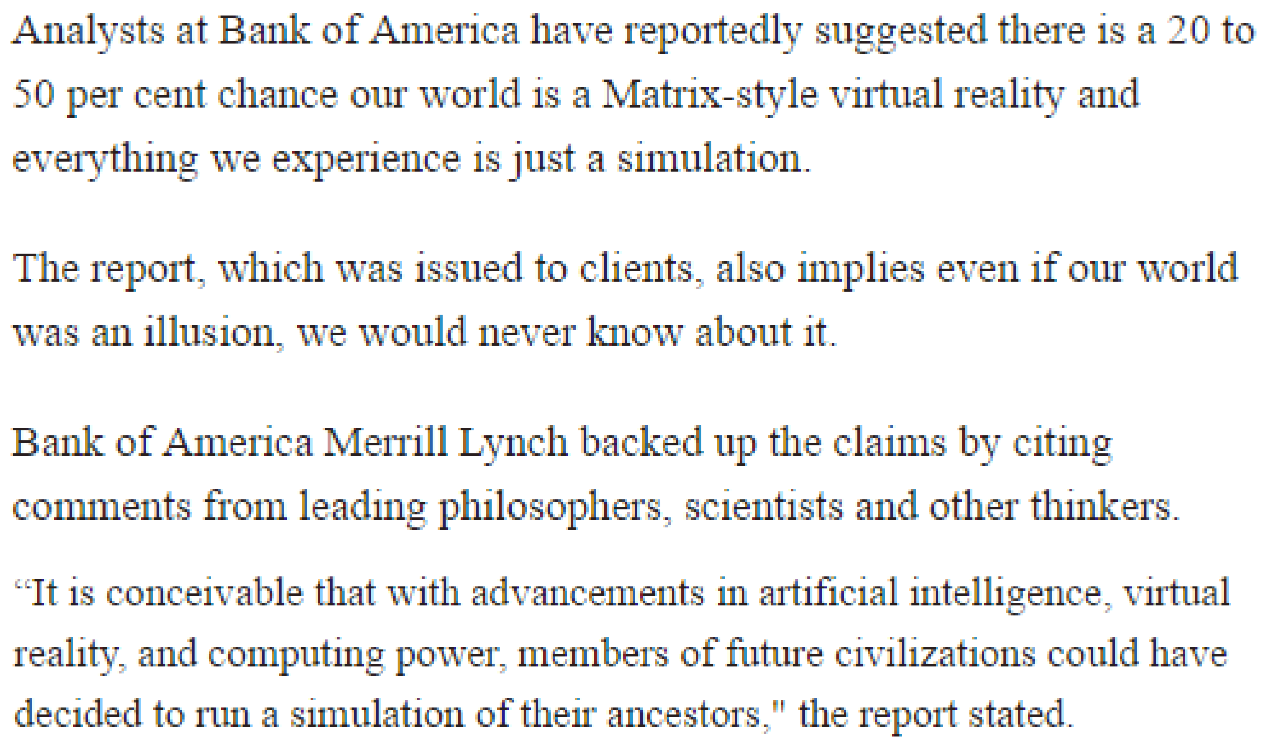
Report cites SpaceX founder Elon Musk and Oxford philosophy professor Nick Bostrom

Jacob Furedi | Wednesday 14 September 2016 | 76 comments

31K shares

Recap

- Dealing with matrices:
- http://www.independent.co.uk/life-style/gadgets-and-tech/news/bank-of-america-the-matrix-50-per-cent-virtual-reality-elon-musk-nick-bostrom-a7287471.html



Analysts at Bank of America have reportedly suggested there is a 20 to 50 per cent chance our world is a Matrix-style virtual reality and everything we experience is just a simulation.

The report, which was issued to clients, also implies even if our world was an illusion, we would never know about it.

Bank of America Merrill Lynch backed up the claims by citing comments from leading philosophers, scientists and other thinkers.

"It is conceivable that with advancements in artificial intelligence, virtual reality, and computing power, members of future civilizations could have decided to run a simulation of their ancestors," the report stated.

Matrix multiplication

- Always multiply the corresponding row in the left hand matrix by the corresponding column in the right hand matrix.
- Can anyone spot the typo?

For example

$$\begin{aligned} u_1 &= 2x_1 + 3x_2 + 3x_3 \\ u_2 &= 2x_1 + 4x_2 + 5x_3 \\ u_3 &= x_1 + x_2 + x_3 \end{aligned}$$

Express using matrix product

6

$$\begin{bmatrix} 2 & 3 & 3 \\ 2 & 4 & 5 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

$$AX = U$$

- Another example

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}$$

$$a_x = a_1 x_1 + a_2 x_2 + a_3 x_3$$

$$a_y = a_1 y_1 + a_2 y_2 + a_3 y_3$$

$$a_z = a_1 z_1 + a_2 z_2 + a_3 z_3$$

Transformations

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

- Have to multiply each point (x,y) by the Transformation matrix
- Need to convert (x,y) into a 3x1 matrix as follows. Note the last component is always 1

$$(x, y) = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Matrix transform is thus:

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

HTML5 Canvas - Transforms

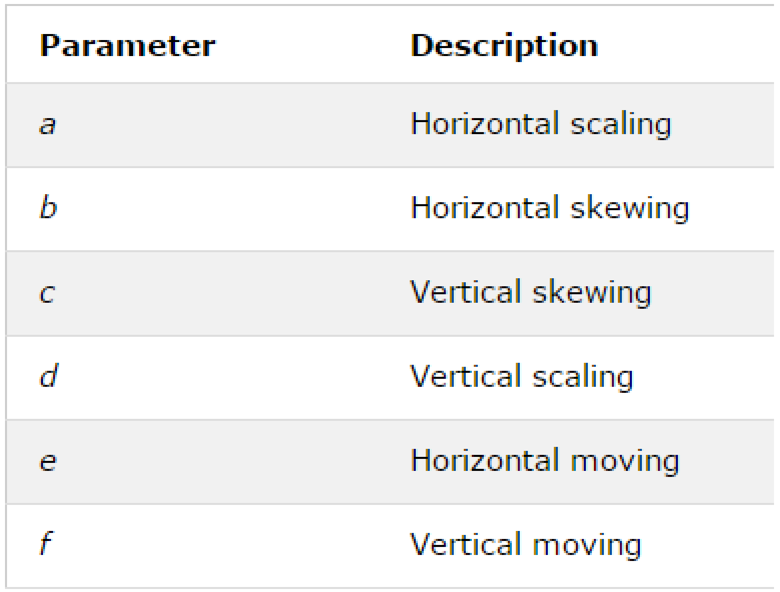
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- HTML5 canvas provides methods which allow modifications directly to the transformation matrix. The transformation matrix must initially be the identity transform. It may then be adjusted using the transformation method:
  - *transform(a, b, c, d, e, f)*
  - This method changes the transformation matrix to apply the matrix given by the arguments.
- The '*transform(a, b, c, d, e, f)*' method must multiply the current transformation matrix with the matrix described by:

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

HTML5 Canvas - Transforms

- **Note:** The transformation will only affect drawings made after the transform() method is called.
- **Note:** The transform() method behaves relatively to other transformations made by rotate(), scale() or translate(). Example: If you already have set your drawing to scale by two, and the transform() method scales your drawings by two, your drawings will now scale by four.
- Parameter Values:



Parameter	Description
a	Horizontal scaling
b	Horizontal skewing
c	Vertical skewing
d	Vertical scaling
e	Horizontal moving
f	Vertical moving

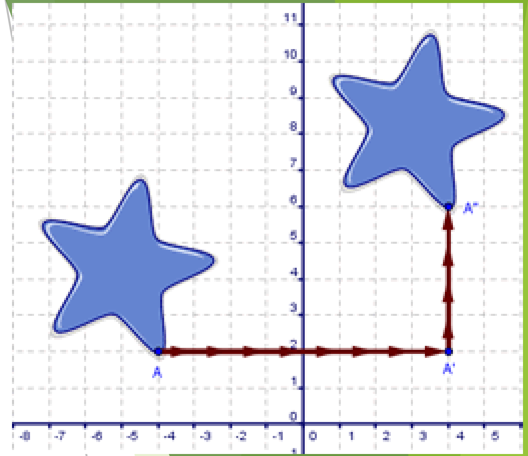
HTML5 Canvas - Transforms

- Identity Matrix
  - ctx.setTransform(1, 0, 0, 1, 0, 0);

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Translation

$$\begin{bmatrix} 1 & 0 & e \\ 0 & 1 & f \\ 0 & 0 & 1 \end{bmatrix}$$



- Translation is the name we give to moving an object from one location to another. An object might be a rectangle, circle, triangle or something more complex.
  - e in the above matrix corresponds to the translation in the direction of the x axis,
  - f in the above matrix corresponds to the translation in the direction of the y axis.
- If translation is all we want to do, we leave the four other letters as they are in the identity matrix. Below is an example of a matrix that will translate objects by (4,6).

$$\begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{bmatrix}$$

Translation Example

$$\begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{bmatrix}$$

- To use this matrix on a point (x,y), we map (x,y) to a 3x1 matrix as follows:

$$(x,y) = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Next, multiply the 3x3 translation matrix by the point matrix, and then map the product back to a point.
- Example: Below we translate the point (1,1) by 4 in the x-axis and 6 in the y axiz

$$(1,1) \to \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \\ 1 \end{bmatrix}$$

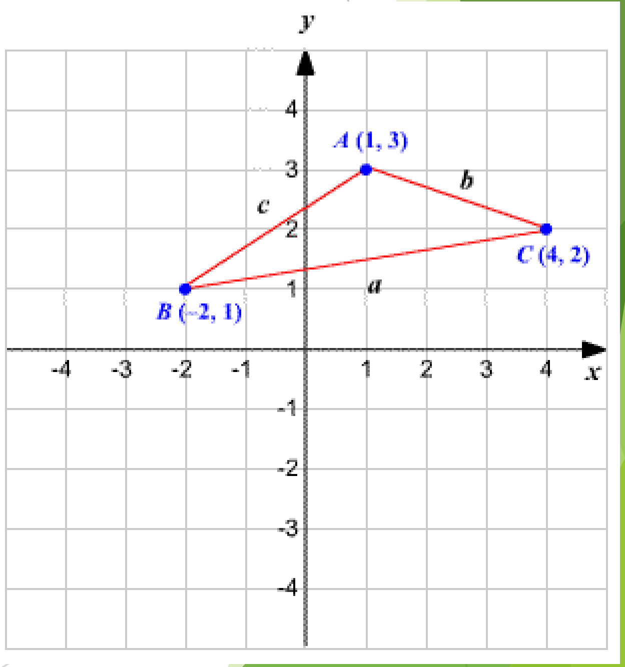
$$\begin{bmatrix} 5 \\ 7 \\ 1 \end{bmatrix} \to (5,7)$$

Scaling

$$\begin{bmatrix} a & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Scaling is achieved by replacing the first two numbers in the matrix diagonal, a and d.
  - The number a will scale the object along the x axis.
  - Likewise, the number d will scale along the y axis.
- For instance to scale by (3,2), the point (1,1) becomes (3,2)

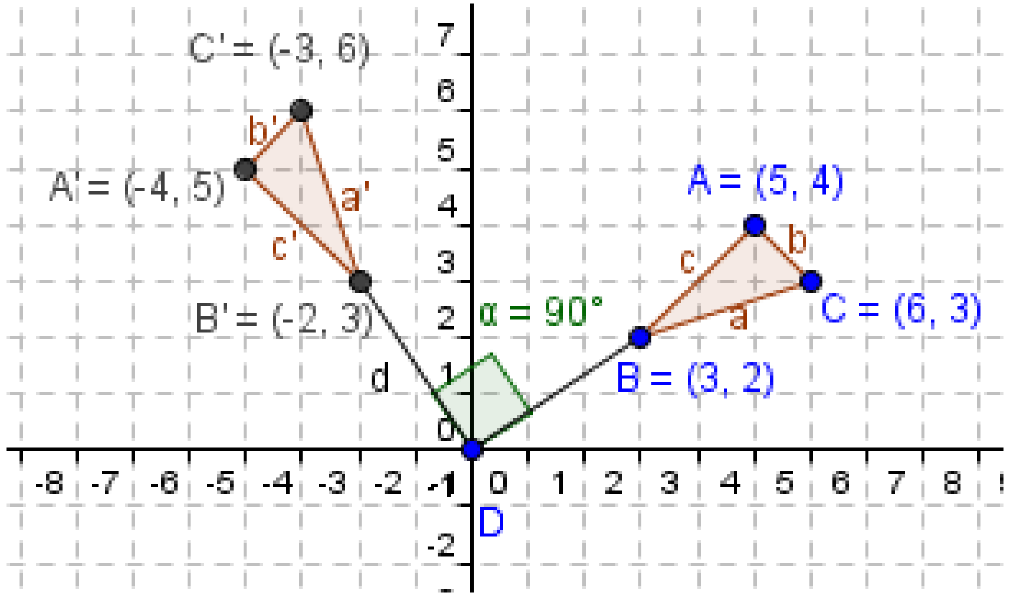
$$\begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$



Rotation

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Rotation, about the **origin**, is achieved using the matrix above.
- As before, θ is measured in radians and the origin is in the top left corner of the canvas (unlike in the diagram below):



Rotation Example

- Again we multiply each point by the transformation (rotation) matrix.
- Under a rotation of 90 degrees, or π/2 radians, the point (1,1) becomes (-1,1)

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

Rotation – Origin note

- The HTML5 Canvas has its origin in the top left
- This is different to the usual coordinate system (origin in bottom left)
- The 2 main differences are:
  - Translating and scaling in the y-direction is reversed
  - Rotations move clockwise about the origin in Canvas
  - Rotations move anti-clockwise about the origin in standard coordinate system
- Exam situation:
  - If you are asked in a written exam situation to perform a transformation, you should always assume that the origin is in the bottom-left (the usual location in mathematics)

Combining Transformations

- One of the major benefits of using transformation matrices is that it’s easy to apply a series of transformations together.
- Consider translating any point by (3,2) and then rotating it counter clockwise by π/2 radians.
  - The point (1,1) first translates to (4,3) and then rotates to (-3,4)

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

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -3 \\ 4 \\ 1 \end{bmatrix}$$

Combining Transformations – Single Matrix

- The 2 separate translation and scaling transformations on the previous slide can be combined so that the operation is performed at once.
- This is achieved by combining both transformation matrices into a new transformation matrix as follows:

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

- Now we can apply the transformation to translate the point (1,1) to (-4,3) in one go.
- This makes the process a lot more efficient if the transformation has to be applied to many points.

Combining Transformations – Reverse Order

- Note the reversing the order of transformations usually results in a different result.
  - If we rotate first and then translate, the point (1,1) instead goes to (2,3).

$$\begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 3 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -1 & 3 \\ 1 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

Luckily for HTML5 Canvas - Simplified functions

- Translation:
  - ctx.translate(x, y);
- Rotation:
  - ctx.rotate(radians);   //rotate 25 degrees.
  - As mentioned, all shapes are rotated around the upper left corner of the canvas (0,0). But, what if you wanted the shape to be rotated around a different point? For instance, rotating the shape around its own center?
  - To rotate a shape around its own center, you must first translate the canvas to the center of the shape, then rotate the canvas, then translate the canvas back to 0,0, and then draw the shape.

```
ctx.translate(cx, cy);              //translate to center of shape

ctx.rotate( (Math.PI / 180) * 25);  //rotate 25 degrees.

ctx.translate(-cx, -cy);
```

Luckily for HTML5 Canvas - Simplified functions

- Scaling:
  - ctx.scale(scaleX, scaleY);

Lab this week

- Create a Canvas
- Draw a pacman and a pizza
- Translate, scale and rotate shapes
- Animate