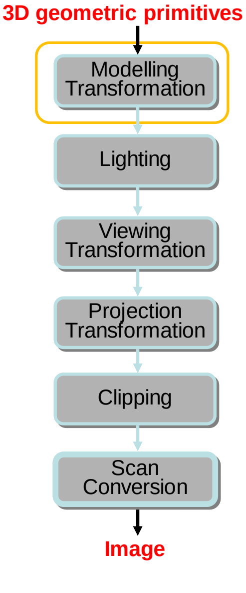
# 3D Graphics Programming Tools

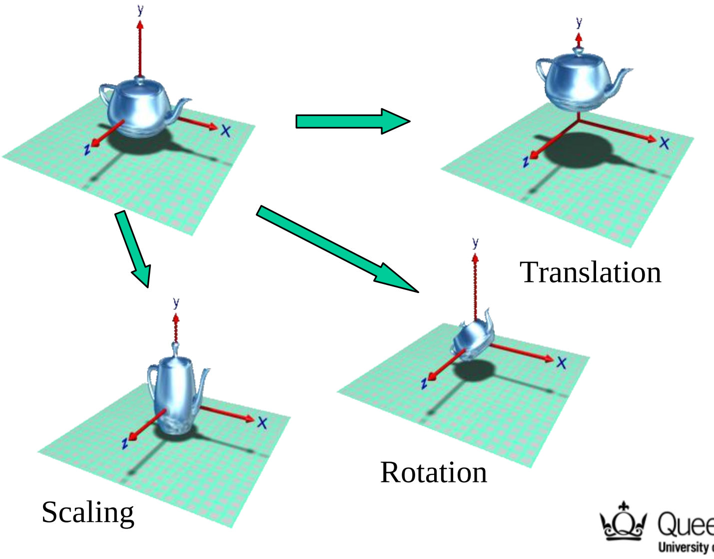
## Modelling Transformations

1




2

## Modelling transformations




3

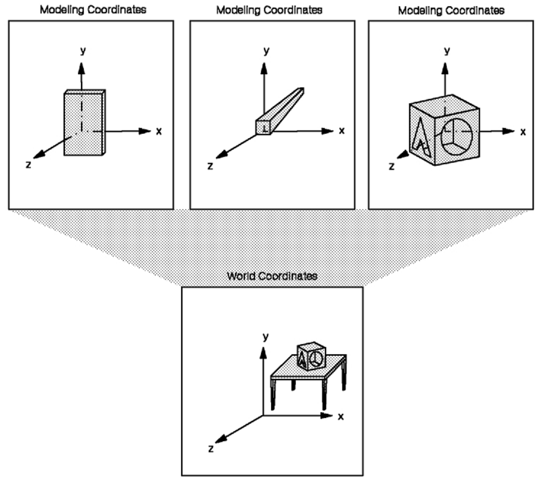
## Today’s agenda

- 2D Transformations
  - Basic 2D transformations
  - Matrix representation
  - Matrix composition
- 3D Transformations
  - Basic 3D transformations

4

## Modelling transformations

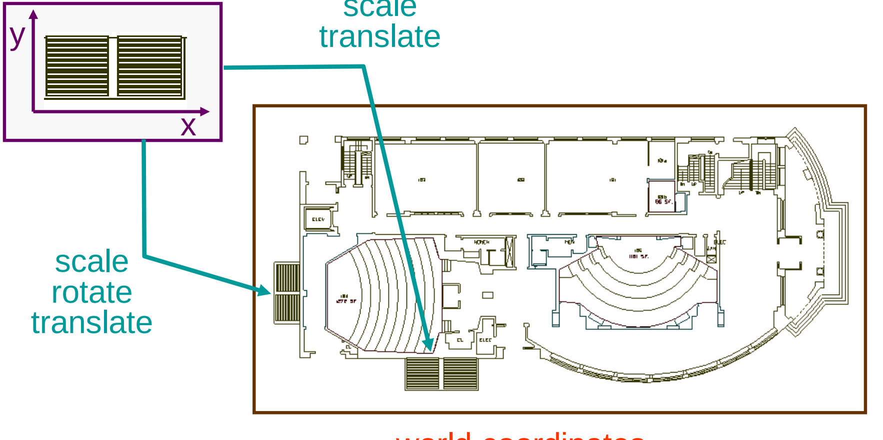
- Specify transformations for objects
  - definitions of objects in own coordinate systems
  - use of object definition multiple times in a scene




5

## 2D modelling transformations

Modelling Coordinates
(i.e. object coordinates)

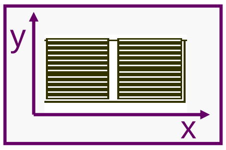


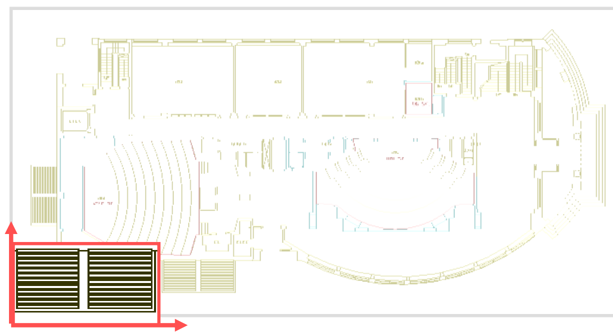

world coordinates

6

## 2D modelling transformations

modelling coordinates



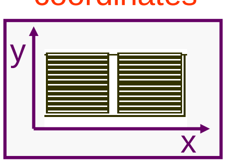



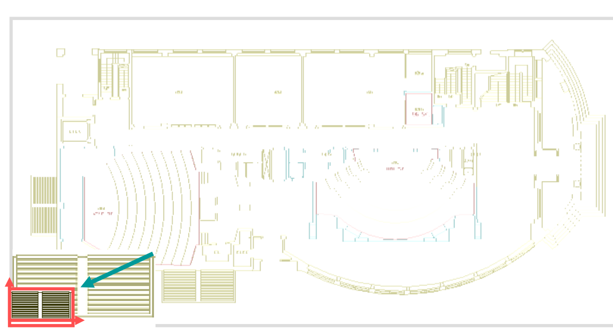
Initial location at (0, 0) with x- and y-axes aligned



## 2D modelling transformations

modelling coordinates

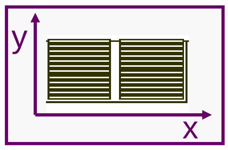


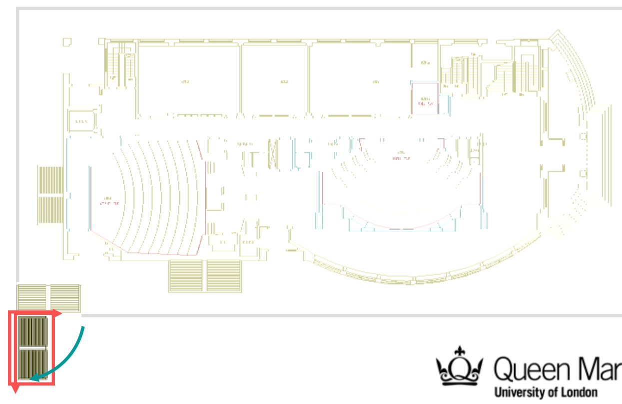
scale .3, .3

## 2D modelling transformations

modelling
coordinates




scale .3, .3
rotate -90

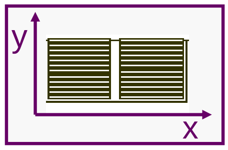


Queen Mary
University of London

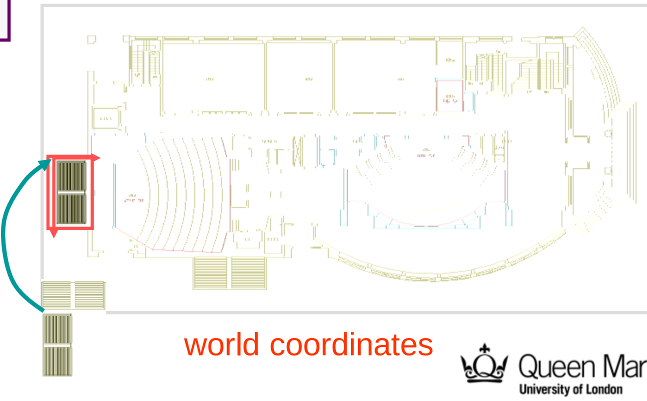
9

## 2D modelling transformations

modelling
coordinates




scale .3, .3
rotate -90
translate 5, 3



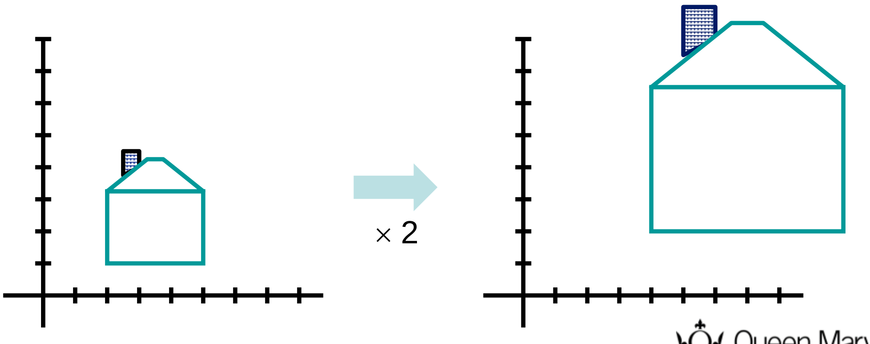
world coordinates

Queen Mary
University of London

10

## Scaling

- Scaling a coordinate
  - means multiplying each of its components by a scalar
- Uniform scaling
  - means this scalar is the same for all components

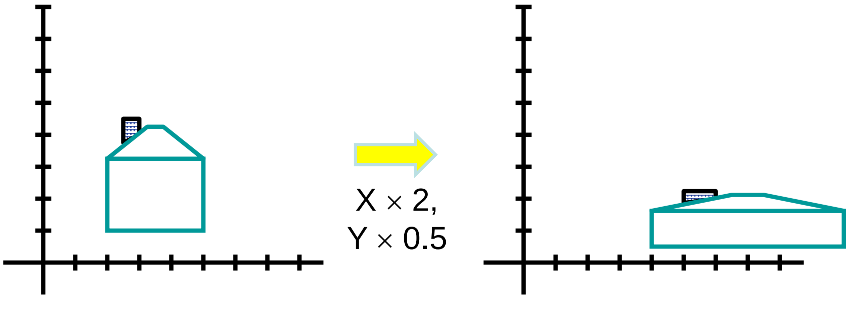



Queen Mary University of London

11

## Scaling

- Non-uniform scaling
  - different scalars per component




How can we represent this in matrix form?

Queen Mary University of London

12

## Scaling

- Scaling operation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} ax \\ by \end{bmatrix}$$

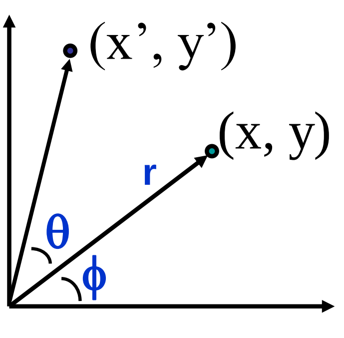
- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\textit{scaling matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Multiplying a point (or a vector) by a matrix (a transformation) yields a new transformed point (or a new vector)



## 2D rotation




x = r cos (φ)
y = r sin (φ)

x' = r cos (φ + θ)
y' = r sin (φ + θ)

trigonometric identity…
x' = r cos(φ) cos(θ) – r sin(φ) sin(θ)
y' = r sin(φ) cos(θ) + r cos(φ) sin(θ)

substitute…
x' = **x** cos(θ) - **y** sin(θ)
y' = **x** sin(θ) + **y** cos(θ)

## 2D rotation

- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

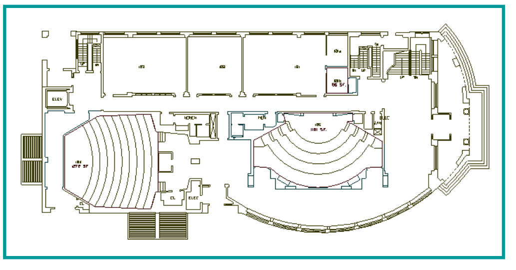
- Even though sin(θ) and cos(θ) are nonlinear functions of θ,
  - x’ is a linear combination of x and y
  - y’ is a linear combination of x and y

Queen Mary University of London

15

## Basic 2D transformations

- Translation
  - $x' = x + t_x$
  - $y' = y + t_y$
- Scale
  - $x' = x * s_x$
  - $y' = y * s_y$
- Shear
  - $x' = x + h_x * y$
  - $y' = y + h_y * x$
- Rotation
  - $x' = x * \cos\Theta - y * \sin\Theta$
  - $y' = x * \sin\Theta + y * \cos\Theta$

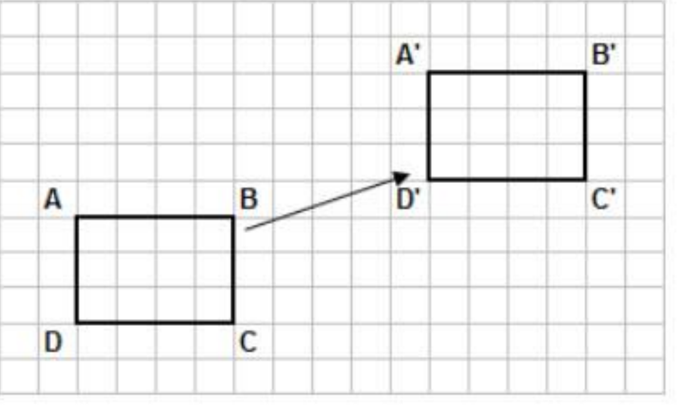


Transformations can be combined (with simple algebra)

Queen Mary University of London

16

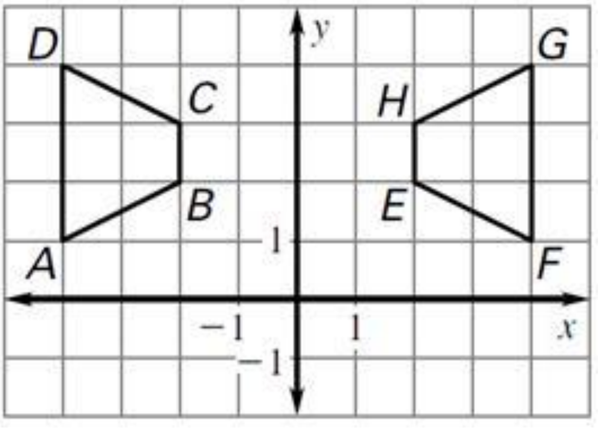
## Name the transformation!




Queen Mary University of London

17

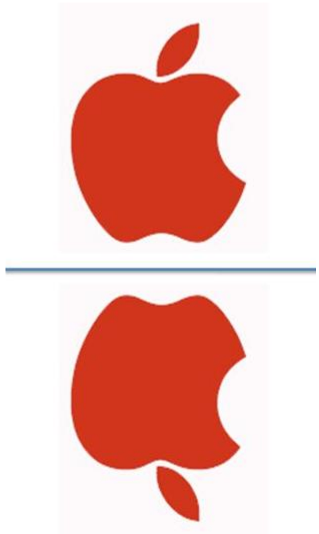
## Name the transformation!




Queen Mary University of London

18

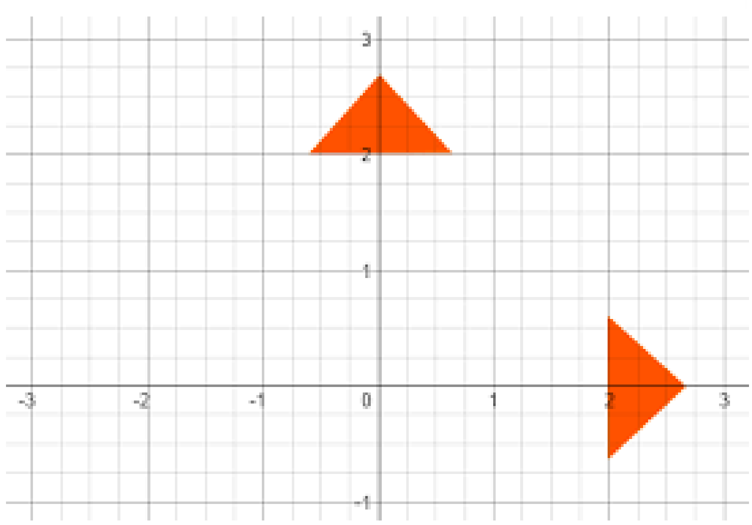
## Name the transformation!



Queen Mary University of London

19

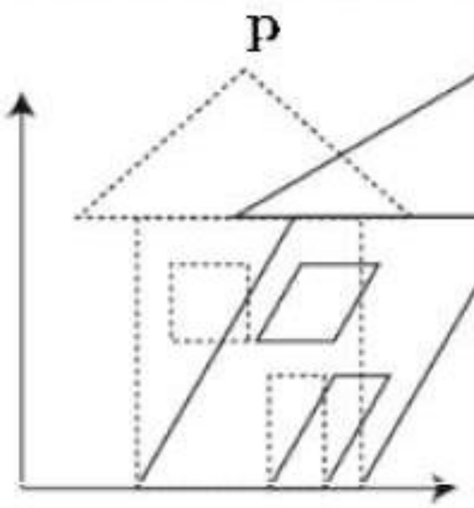
## Name the transformation!




Queen Mary University of London

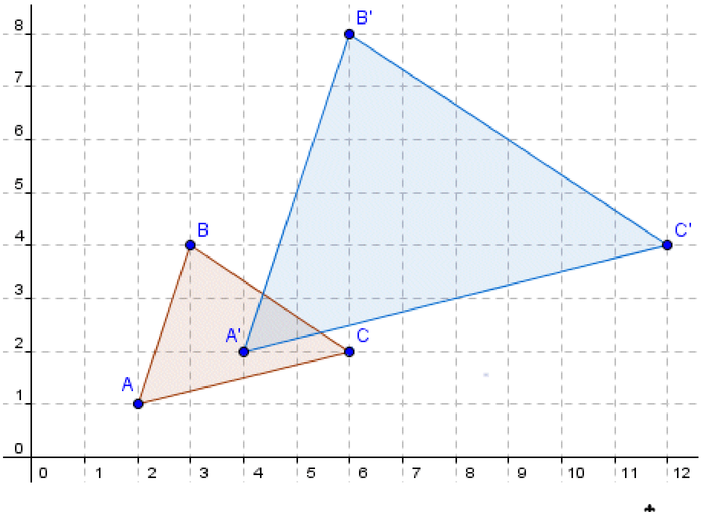
20

## Name the transformation!



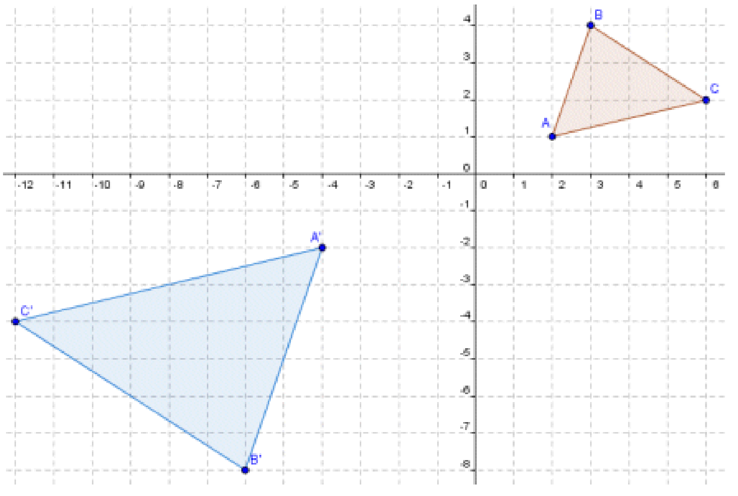

21

## Name the transformation!



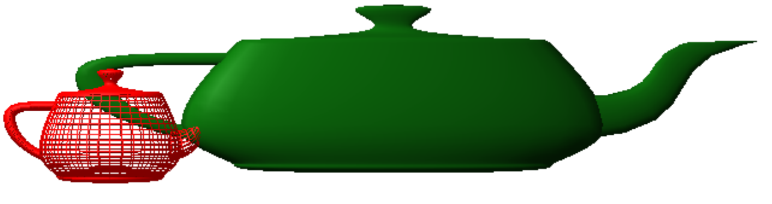

22

## Name the transformation!

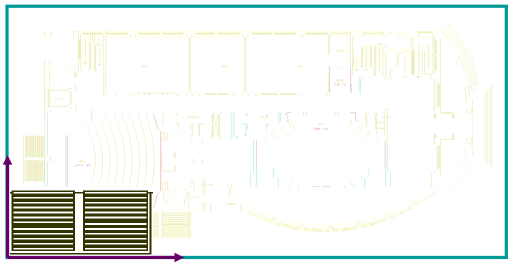





## Name the transformation!



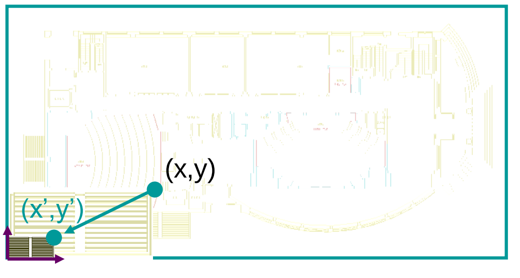
## Basic 2D transformations (combination)



25

## Basic 2D transformations (combination)

- Scale
  - $x' = x * s_x$
  - $y' = y * s_y$

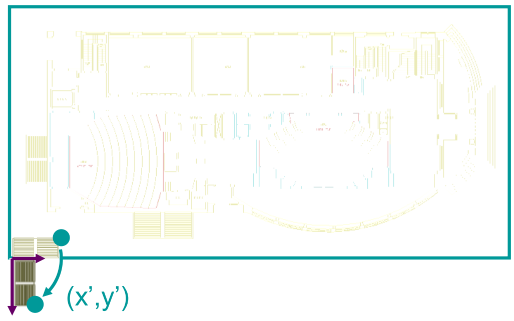



$$x' = x * s_x$$
$$y' = y * s_y$$

26

## Basic 2D transformations (combination)

- Scale
  - $x' = x * s_x$
  - $y' = y * s_y$
- Rotation
  $x' = x * \cos\Theta - y * \sin\Theta$
  $y' = x * \sin\Theta + y * \cos\Theta$




$$x' = (x * s_x) * \cos\Theta - (y * s_y) * \sin\Theta$$
$$y' = (x * s_x) * \sin\Theta + (y * s_y) * \cos\Theta$$

Queen Mary University of London

27

## Basic 2D transformations (combination)

- Scale
  - $x' = x * s_x$
  - $y' = y * s_y$
- Rotation
  $x' = x * \cos\Theta - y * \sin\Theta$
  $y' = x * \sin\Theta + y * \cos\Theta$
- Translation
  $x' = x + t_x$
  $y' = y + t_y$


$$x' = ((x * s_x) * \cos\Theta - (y * s_y) * \sin\Theta) + t_x$$
$$y' = ((x * s_x) * \sin\Theta + (y * s_y) * \cos\Theta) + t_y$$

Queen Mary University of London

28

## Today's agenda

- 2D Transformations
  - Basic 2D transformations
  - Matrix representation
  - Matrix composition
- 3D Transformations
  - Basic 3D transformations

Queen Mary University of London

29

## Matrix representation

- Represent 2D transformation by a matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

- Multiply matrix by column vector
  $\Leftrightarrow$ apply transformation to point

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad \begin{aligned} x' &= ax + by \\ y' &= cx + dy \end{aligned}$$

Queen Mary University of London

30

## Matrix representation

- Transformations combined by multiplication

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Matrices are a convenient and efficient way to represent a sequence of transformations

Matrix multiplication is not generally commutative !



## 2x2 matrices

- What types of transformations can be represented with a 2x2 matrix?

2D identity

$$x' = x$$
$$y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D scale

$$x' = s_x * x$$
$$y' = s_y * y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

## 2x2 matrices

- What types of transformations can be represented with a 2x2 matrix?

2D rotate around (0,0)

$$x' = \cos\Theta * x - \sin\Theta * y$$
$$y' = \sin\Theta * x + \cos\Theta * y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D shear

$$x' = x + sh_x * y$$
$$y' = sh_y * x + y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Queen Mary University of London

33

## 2x2 matrices

- What types of transformations can be represented with a 2x2 matrix?

2D mirror about Y axis

$$x' = -x$$
$$y' = y$$

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

2D mirror over (0,0)

$$x' = -x$$
$$y' = -y$$

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

Queen Mary University of London

34

## 2x2 matrices

- What types of transformations can be represented with a 2x2 matrix?

2D translation

$$x' = x + t_x$$
$$y' = y + t_y$$

NO!

Only linear 2D transformations can be represented with a 2x2 matrix



## Linear transformations

- Linear transformations are combinations of
  - scale
  - rotation
  - shear and
  - mirror

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

- Properties of linear transformations
  - origin maps to origin
  - lines map to lines
  - parallel lines remain parallel
  - ratios are preserved
  - closed under composition

$$T(s_1\mathbf{p}_1 + s_2\mathbf{p}_2) = s_1 T(\mathbf{p}_1) + s_2 T(\mathbf{p}_2)$$

## Homogeneous coordinates

- Homogeneous coordinates
  - represent coordinates in 2 dimensions with a 3D vector
  - seem unintuitive, but they make graphics operations much easier

$$\begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{\text{homogeneous coords}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

37

## Homogeneous coordinates

- How can we represent translation as a 3x3 matrix?
  - Using the rightmost column

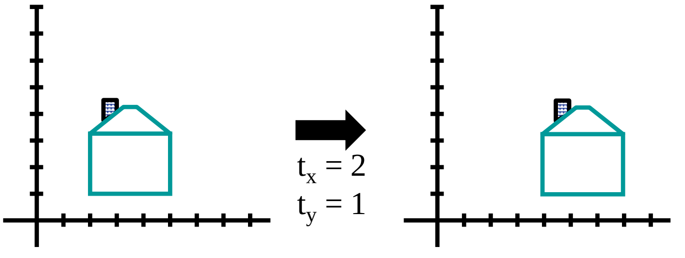
$$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}_x$$
$$\boldsymbol{y}' = \boldsymbol{y} + \boldsymbol{t}_y$$

$$\mathbf{Translation} = \begin{bmatrix} 1 & 0 & \boldsymbol{t}_x \\ 0 & 1 & \boldsymbol{t}_y \\ 0 & 0 & 1 \end{bmatrix}$$

38

## Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

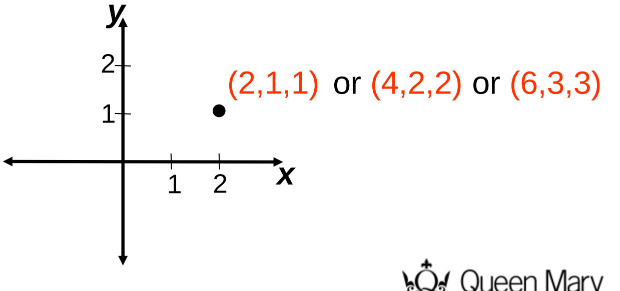



39

## Homogeneous coordinates

- Homogeneous coordinates
  - add a 3rd coordinate to every 2D point
    - (x, y, w) represents a point at location (x/w, y/w)
    - (x, y, 0) represents a point at infinity
    - (0, 0, 0) is not allowed

Convenient coordinate system to represent many useful transformations




40

## Basic 2D transformations

- Basic 2D transformations as 3x3 matrices

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

translate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

rotate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

shear

Queen Mary University of London

41

## Affine transformations

- Affine transformations are combinations of
  - Linear transformations, and
  - Translations

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

- Properties of affine transformations
  - origin does not necessarily map to origin
  - lines map to lines
  - parallel lines remain parallel
  - ratios are preserved
  - closed under composition

Queen Mary University of London

42

## Today's agenda

- 2D Transformations
  - Basic 2D transformations
  - Matrix representation
  - Matrix composition
- 3D Transformations
  - Basic 3D transformations

Queen Mary University of London

43

## Matrix composition

- Transformations can be combined by matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left( \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

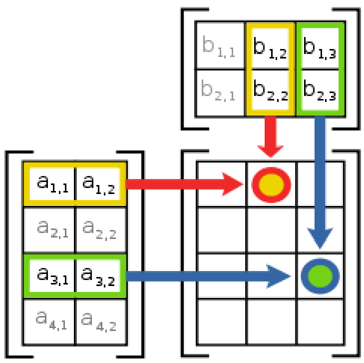
$$\mathbf{p'} \quad = \quad T(t_x,t_y) \qquad R(\Theta) \qquad S(s_x,s_y) \qquad \mathbf{p}$$

Queen Mary University of London

44

## Matrix multiplication (reminder)

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$




Queen Mary
University of London

45

## Matrix composition

- Matrices are a convenient and efficient way to represent a sequence of transformations
  - general purpose representation
  - hardware matrix multiply

**p'** = (T * (R * (S***p**) ) )

**p'** = (T*R*S) * **p**

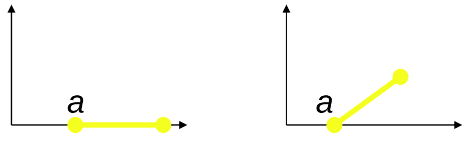
  - NB! order of transformations matters
    - matrix multiplication is not commutative

Queen Mary
University of London

46

## Example

- What if we want to rotate and translate?
  - Ex: Rotate line segment by 45 degrees about endpoint *a*

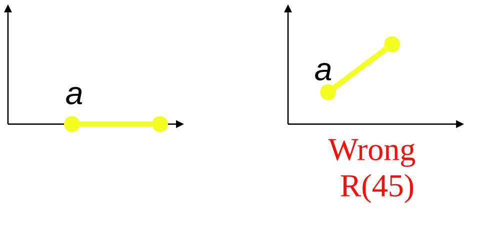



Queen Mary University of London



## Multiplication order – wrong way

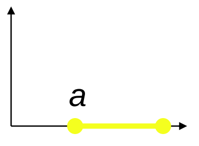
- The line segment is defined by two endpoints
  - Applying a rotation of 45 degrees, R(45), affects both points
  - We could try to translate both endpoints to return endpoint *a* to its original position, but by how much?

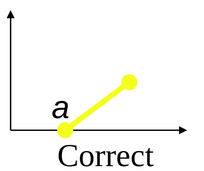



Queen Mary University of London

## Multiplication order - correct

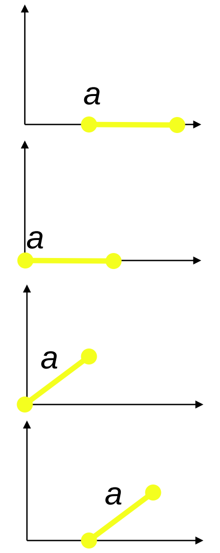
- Isolate endpoint a from rotation effects
  - First translate line so *a* is at origin: T (-3)
  - Then rotate line 45 degrees: R(45)
  - Then translate back so *a* is where it was: T(3)




Correct

T(-3,0) R(45) T(3,0)






## Example

Will this sequence of operations work?

$$\begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(45) & -\sin(45) & 0 \\ \sin(45) & \cos(45) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ 1 \end{bmatrix} = \begin{bmatrix} a'_x \\ a'_y \\ 1 \end{bmatrix}$$

Queen Mary University of London

## Matrix composition

- After correctly ordering the matrices
- Multiply matrices together
- What results is one matrix – store it (on stack)!
- Multiply this matrix by the vector of each vertex

→ All vertices easily transformed with one matrix multiply

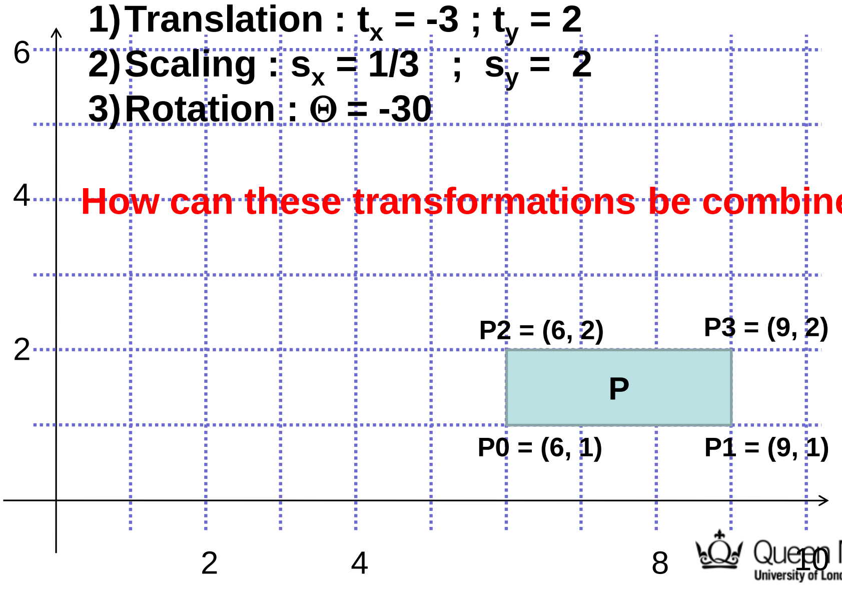
Queen Mary University of London

51

## Exercise

1) Translation : $t_x = -3$ ; $t_y = 2$
2) Scaling : $s_x = 1/3$ ; $s_y = 2$
3) Rotation : $\Theta = -30$

**How can these transformations be combined?**




Queen Mary University of London

52

## Today’s agenda

- 2D Transformations
  - Basic 2D transformations
  - Matrix representation
  - Matrix composition
- 3D Transformations
  - Basic 3D transformations

Queen Mary University of London



## 3D transformations

- Same idea as 2D transformations
  - homogeneous coordinates: (x,y,z,w)
  - 4x4 transformation matrices

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Queen Mary University of London

## Basic 3D transformations

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

identity

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ \boldsymbol{z}' \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 & 0 \\ 0 & 0 & \boldsymbol{s}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \boldsymbol{w} \end{bmatrix}$$

scale

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ \boldsymbol{z}' \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \boldsymbol{t}_x \\ 0 & 1 & 0 & \boldsymbol{t}_y \\ 0 & 0 & 1 & \boldsymbol{t}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \boldsymbol{w} \end{bmatrix}$$

translation

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

mirror about Y/Z plane

Queen Mary
University of London



## Basic 3D transformations

Rotate around Z axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 & 0 \\ \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Rotate around Y axis

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ \boldsymbol{z}' \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} \cos\Theta & 0 & \sin\Theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Theta & 0 & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \boldsymbol{w} \end{bmatrix}$$

Rotate around X axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\Theta & -\sin\Theta & 0 \\ 0 & \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Queen Mary
University of London

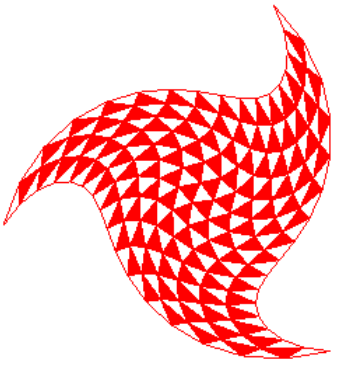
## 3D rotation

General rotations in 3D
- require rotating about an arbitrary *axis of rotation*
- deriving the rotation matrix for such a rotation directly is a good exercise in linear algebra …
- standard approach
  - express general rotation as composition of canonical rotations
    - rotations about **X, Y, Z**

Queen Mary University of London

57

## Twist



Queen Mary University of London

58

## Twist

```
void display()
{
   glClear(GL_COLOR_BUFFER_BIT);
   divide_triangle(v[0], v[1], v[2], n);
   glFlush();
}

void divide_triangle(GLfloat *a, GLfloat *b, GLfloat *c, int m)
{
   GLfloat v[3][2];
   int j;
   if(m>0)
    {
      for(j=0; j<2; j++) v[0][j]=(a[j]+b[j])/2;
      for(j=0; j<2; j++) v[1][j]=(a[j]+c[j])/2;
      for(j=0; j<2; j++) v[2][j]=(b[j]+c[j])/2;
      divide_triangle(a, v[0], v[1], m-1);
      divide_triangle(v[0], b, v[2], m-1);
      divide_triangle(v[1], v[2], c, m-1);
      divide_triangle(v[0], v[1], v[2], m-1);
   }
   else(triangle(a,b,c));
}
```

Queen Mary University of London

59

## Twist

```
GLfloat twist = 1.5;

void triangle (GLfloat *a, GLfloat *b, GLfloat *c)
{
  GLfloat v[2];
  double d;

  glBegin(GL_POLYGON);
          d = sqrt(a[0]*a[0] + a[1]*a[1]);
          v[0] = ?
          v[1] = ?
          glVertex2fv(v);
          d = sqrt(b[0]*b[0] + b[1]*b[1]);
          v[0] = ?
          v[1] = ?
          glVertex2fv(v);
          d = sqrt(c[0]*c[0] + c[1]*c[1]);
          v[0] = ?
          v[1] = ?
          glVertex2fv(v);
  glEnd();
}
```

Queen Mary University of London

60