COMP270
Mathematics for 3D Worlds and Simulations

Week 3 Exercises: Dot Product and Matrices

This worksheet is split into two sections: Part A is a set of "traditional" maths questions to complete without a computer, while Part B involves using code to answer the same or similar questions. You can complete either section first, or swap between them; you may find that tackling the same problem using a different approach enhances your understanding of it.

PART A

Answer the following questions using pen(cil) and (graph) paper.
Pro tip: show your working – diagrams can be helpful!

1. For each of the pairs of vectors below, evaluate their dot product using the algebraic definition:
$$\mathbf{a} \cdot \mathbf{b} = x_\mathrm{a} x_\mathrm{b} + y_\mathrm{a} y_\mathrm{b}$$
and check the result against your answers to question 3 of part A of last week's exercises using the identity
$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\| \cos\theta$$
   a. $\mathbf{a} = \begin{pmatrix} 3 \\ \sqrt{3} \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} 1 \\ \sqrt{3} \end{pmatrix}$
   b. $\mathbf{a} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} -2 \\ 2 \end{pmatrix}$

2. Write down any two vectors that are (i) parallel and (ii) perpendicular to:
   a. $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$
   b. $\begin{pmatrix} -1 \\ -1 \end{pmatrix}$
   c. $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$
   d. $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$

   Verify that your answers are correct using the dot product.

3. Compute the following matrix products:
   a. $\begin{pmatrix} 1 & -2 \\ 5 & 0 \end{pmatrix}\begin{pmatrix} -3 & 7 \\ 4 & \frac{1}{3} \end{pmatrix}$
   b. $\begin{pmatrix} 6 & -7 \\ -4 & 5 \end{pmatrix}\begin{pmatrix} 3 \\ 3 \end{pmatrix}$
   c. $\begin{pmatrix} -3 & -2 \\ 5 & 4 \end{pmatrix}\begin{pmatrix} -2 & -1 \\ 2\frac{1}{2} & 1\frac{1}{2} \end{pmatrix}$
   d. $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix}$
   e. $\begin{pmatrix} 3 & 3 \end{pmatrix}\begin{pmatrix} 6 & -7 \\ -4 & 5 \end{pmatrix}$

   What do you notice about (c)? How about (b) and (e)?

Exercises may include some modified questions from
Dunn, F & Parberry, I 2011, *3D Math Primer for Graphics and Game Development*, CRC Press, Boca Raton, FL

4. Describe the transformation represented by each of the following matrices.
   Hint: consider what happens when they are applied to the *basis vectors* $\begin{pmatrix}1\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\end{pmatrix}$

   a. $\begin{pmatrix}0 & 1\\-1 & 0\end{pmatrix}$

   b. $\begin{pmatrix}\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\\\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\end{pmatrix}$

   c. $\begin{pmatrix}2 & 0\\0 & 2\end{pmatrix}$

   d. $\begin{pmatrix}4 & 0\\0 & 7\end{pmatrix}$

   e. $\begin{pmatrix}-1 & 0\\0 & 1\end{pmatrix}$

   f. $\begin{pmatrix}0 & 2\\-2 & 0\end{pmatrix}$

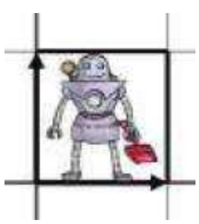
5. Match each of the following figures (1-4) with their corresponding transformations as applied to the figure on the left:

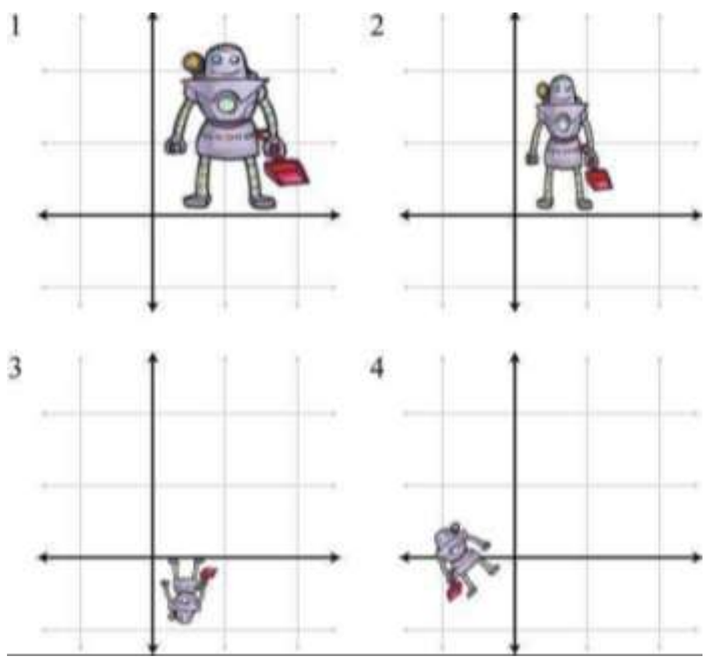
   a. $\begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}$

   b. $\begin{pmatrix}2.5 & 0\\0 & 2.5\end{pmatrix}$

   c. $\begin{pmatrix}-\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\\-\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\end{pmatrix}$

   d. $\begin{pmatrix}1.5 & 0\\0 & 2.0\end{pmatrix}$





Figures taken from *3D Math Primer for Graphics and Game Development*, Fletcher Dunn and Ian Parberry, CRC Press

6. Compute the inverses of the matrices in question 4 using the formula

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

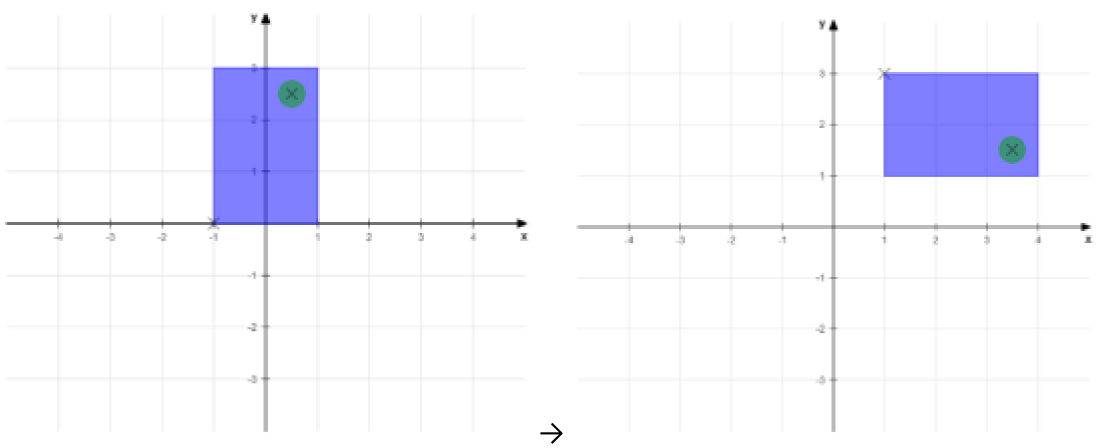
   What transformation does each inverse represent?

7. You may have noticed that a class of transformation is missing from the examples above: translation. This requires the use of *homogeneous coordinates*, where the point (x, y) is represented by $(x, y, 1)$ so that it can be multiplied by a 3x3 matrix, $\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$.

   a. Write the displacement from the origin to the point $(-1, 3)$ as a homogeneous column vector.

   b. Write down the homogeneous matrix to describe a 2D translation of 1 unit in the x direction and 2 units in y, and apply it to the vector in part (a).

   c. Combine your matrix from part (b) with an appropriate rotation matrix to represent the following transformation:



   What happens if you reverse the order in which you combine the matrices?

   d. What transformation would you need to apply to rotate the shape through 180° about the centre of the green dot (marked with a cross) in its top right corner, leaving the dot's position fixed? Express your answer as a single matrix.

PART B

We're going to carry on using the same program as before for this section; make sure you have completed at least the first two questions from part B of last week's exercises, as you'll need the functionality in order to do this week's (the answers are available on LearningSpace if you need them).

Note: you'll need to uncomment the relevant function calls in `Scene::setup()` to be able to complete the exercises.

1. The dot product is an extremely useful operation for checking vector directions.
   a. Implement the dot product formula in `Vector2::dot()`; you can check the answers it gives against your own for question 1 of part A.
   b. The function `Scene::week3_exercise1()` creates a random set of vectors; edit the code to only display ones that are pointing in roughly the same direction (within a range of 180°).
   c. Can you narrow the range even further, e.g. to only display vectors at an angle of less than 45° from any other vector shown?

2. Being able to find the normalised, unit vector that isolates the direction from the length is also useful.
   a. Implement the formula for computing the unit (normalised) vector in `Vector2::normalised()`.
   Hint: you can make use of one of the functions you implemented last week.
   b. The function `Scene::week3_exercise2()` creates two random vectors. Add code to compute (and draw) the projection of the first vector onto the second.
   Hint: you may find it helpful to break the computation into stages, displaying results at each point.

3. The file *Matrix22.h* contains a class, `Matrix22`, to represent a 2×2 matrix; again, some of the functions are unfinished. The matrix components are represented by variables `mij`, where i is the row and j is the column number.

   a. Complete the implementation by adding the following operations:
      i. Multiplication with another `Matrix22` in `Matrix22::operator*()`.
      ii. Multiplication with a `Vector2` in the relevant `operator*()` function in *Vector2.h*.
      iii. Computing the inverse matrix in `Matrix22::inverse()`.

   b. Use the Matrix22 class to apply the transformations in questions 4-6 of part A to the objects created in `Scene::week3_exercise3()` (and/or any others you care to add); also try combining these matrices by multiplying them in different orders. Does the order matter for all combinations? Can you explain why/why not?
   Hint: the matrices from 4a and 4e have been added to get you started, but you'll need to create the rest yourself.

4. The translations required in question 7 of part A cannot be computed using the `Vector2` and `Matrix22` classes, as they only represent objects in $\mathbb{R}^2$. There are partially-implemented homogeneous version in *Vector2h.h* and *Matrix22h.h*, which can be converted to/from the original classes to allow the full set of transformations to be applied.
    a. Complete the implementations for the two homogeneous classes, `Vector2h` and `Matrix22h`, by finishing the following functions:
        i. Multiplication with another `Matrix22h` in `Matrix22h::operator*()`.
        ii. Multiplication with a `Vector2h` in the relevant `operator*() function in` *Vector2h.h*.
    b. Use these classes to test the transformation matrices in question 7 of part A (the code for this has been started in `Scene::week3_exercise3()`).

5. *Bonus exercise*: the `update()` method in the `Application` class is called during the main run loop; see if you can animate the objects by updating their transforms in this function.
Hint: you'll need to add functions that the `Application` can call to affect the objects, which you'll need to keep track of after they've been created.