Physics-Based Animation (SET09119)

Tutorial 01 (NOTES) - Mathematics for Physics

1 Vectors

1.1 What is a Vector?

- A scalar (a normal number) is sometimes called a magnitude - it provides a value (e.g. distance) without any notion of direction. This is fine for one dimensional work, but doesn't let us work with any other dimensions.
- A vector contains a magnitude and a direction. This allows us to model movement in two or more dimensions.

Note: be aware of what hats and arrows mean - a common notation for a vector is an arrow above the variable (e.g., $\vec{a}$). While for a unit vector, (i.e., a vector of magnitude 1) a hat is used, (e.g., $\hat{n}$).

1.2 Displacement

- Sometimes you will hear about object displacement. Object displacement is to do with how far an object has moved from its original position, rather than the entire distance it might have moved between the start and end point. For example:

  I travel from Edinburgh to London

  Current displacement is 535 KM

  I travel to Glasgow

  Current displacement is 75 KM

  I travel back to Edinburgh

  Current displacement is 0 KM

- Understanding the difference between distance and displacement is very important.

1.3 Polar Coordinates and Cartesian Coordinates

- As a vector is a magnitude and a distance, it can be described in different forms. In particular, for 2D vectors we can use the following:

  Polar coordinates - these provide a direction as an angle, and the distance as a normal scalar value. The form is as follows:

  $||A||@\theta$, where $||A||$ is the magnitude (distance) of the vector and θ the direction

Cartesian coordinates (components) - provide the vector as horizontal and vertical components. The form is as follows:

$ai + bj$, where i represents a single unit in the x direction, and j a single unit in the y direction

- We can convert from a Polar coordinate to a Cartesian coordinate using the following technique

  For Vector $A = ||A||@\theta$

  Vector $A = a_1 i + a_2 j$

  Where $a_1 = ||A||cos(\theta)$ and $a_2 = ||A||sin(\theta)$

  THIS IS A VERY USEFUL CALCULATION TO CONVERT FROM AN ANGLE TO A VECTOR

- We can also convert from a Cartesian to Polar coordinate using the following technique

  For Vector $B = b_1 i + b_2 j$

  $||B|| = \sqrt{(b_1)^2 + (b_2)^2}$ and $\theta = tan^{-1}(\frac{b_2}{b_1})$

  THIS IS ANOTHER IMPORTANT CALCULATION

- We can also use the Cartesian component approach to define a vector in 3D.

  Vector $B = b_1 i + b_2 j + b_3 k$ where i is one unit in the x direction, j is one unit in the y direction, and k is one unit in the z direction

1.4 Vector Addition and Subtraction

- Adding two vectors together is fairly intuitive, and behaves like scalar addition.

  For two vectors, A and B

  $A + B = (a_1 + b_1)i + (a_2 + b_2)j$

  Where $A = a_1 i + a_2 j$ and $B = b_1 i + b_2 j$

- Subtracting two vectors follows the same form:

  For two vectors, A and B

  $A - B = (a_1 - b_1)i + (a_2 - b_2)j$

  Where $A = a_1 i + a_2 j$ and $B = b_1 i + b_2 j$

- The addition and subtraction calculations also work for 3D vectors

  $A + B = (a_1 + b_1)i + (a_2 + b_2)j + (a_3 + b_3)k$

  $A - B = (a_1 - b_1)i + (a_2 - b_2)j + (a_3 - b_3)k$

1.5 Scalar Multiplication

- Multiplying a vector by a scalar allows us to increase the magnitude (length, distance) of the vector. This is very useful for certain calculations. To scale a vector:

  $sA = sa_1 i + sa_2 j$

  Where s is a scalar, and A is a Vector (i.e., $A = a_1 i + a_2 j$)

- Normalization is a term often used in games and graphics programming when referring to vectors. Normalization allows us to take a vector, and make it one unit long while still pointing in the same direction. The normalization calculation for 2D and 3D vectors are as follows:

- $\hat{A} = \frac{A}{||A||} =< \frac{a_1}{||A||}, \frac{a_2}{||A||} >$

  $A =< a_1, a_2 >$

  $\hat{A} = \frac{A}{||A||} =< \frac{a_1}{||A||}, \frac{a_2}{||A||}, \frac{a_2}{||A||} >$

  $A =< a_1, a_2, a_3 >$

1.6 Dot Product (a.k.a., Scalar Product)

- The dot product of two vectors is a way to multiply to vectors together. To calculate the dot product, use the following form:

  $A \bullet B = (a_1)(b_1) + (a_2)(b_2)$

  For any 2D vectors $A =< a_1, a_2 > andB =< b_1, b_2 >$

  $A \bullet B = a_1b_1 + a_2b_2 + a_3b_3$

  For any 3D vectors $A =< a_1, a_2, a_3 > andB =< b_1, b_2, b_3 >$

- The dot product has some interesting properties:

  If $A \bullet B = 0$ then A and B are perpendicular ($A \perp B$)

  If $A \bullet B < 0, \theta > 90^0$, where θ is the angle between the vectors A and B

  If $A \bullet B > 0, \theta < 90^0$, where θ is the angle between the vectors A and B

  $A \bullet B = ||A||||B||cos(\theta)$, where θ is the angle between the vectors A and B

1.7 Cross Product (a.k.a., Vector Product)

- Another way to multiply two vectors is called the cross product (sometimes called the vector product). Generally we are only interested in the cross product of 3D vectors. The calculation is as follows:

  $A \times B =< (a_2b_3 - a_3b_2), (a_3b_1 - a_1b_3), (a_1b_2 - a_2b_1)$

  For any 3D vectors $A =< a_1, a_2, a_3 >$ and $B =< b_1, b_2, b_3 >$

- The cross product also has some interesting properties

  $(A \times B) \perp A$ and $(A \times B) \perp B$

  $A \times B$ is perpendicular (orthogonal) to both vector A and vector B

  $A \times B \neq B \times A$

  $A \times B$ = -($B \times A$)

  We can calculate the surface normal (the vector perpendicular to a surface) that is defined by two 3D vectors

  $\hat{C} = A \times B = \frac{A \times B}{||A \times B||}$

  $||A \times B|| = ||A||||B||sin(\theta)$, where is the angle between the vectors A and B

- The length of the cross product of two vectors is equal to the area of the parallelogram determined by the two vectors.
- Multiplication by scalars: $(s\vec{a}) \times \vec{b} = s(\vec{a} \times \vec{b}) = \vec{a} \times (s\vec{b})$
- Distributivity: $\vec{a} \times (\vec{b} + \vec{c}) = \vec{a} \times \vec{b} + \vec{a} \times \vec{c}$
- Note that the result for the length of the cross product leads directly to the fact that two vectors are parallel if and only if their cross product is the zero vector. This is true since two vectors are parallel if and only if the angle between them is 0 degrees (or 180 degrees).

1.8 Plane Equation

- The equation of a plane in 3D space is defined with normal vector (perpendicular to the plane) and a known point on the plane
- The equation of a line in the form $ax + by + cz = d$ can be written as a dot product:

  $(a, b, c) \bullet (x, y, z) = d$, or $A \bullet X = d$
- If the normal vector is normalized (unit length), then the constant term of the plane equation, d becomes the distance from the origin.

2 Matrices

2.1 What is a Matrix?

- A matrix is essentially a way of organising values into columns and rows. When programming, we use matrix like structures whenever we have arrays, or multidimensional arrays. When dealing with matrices mathematically, we show them as a grid of values like so:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} x & y & z \end{bmatrix}$$

- The important concept to understand about matrices is that they are composed of a certain amount of rows, and a certain amount of columns. It is common to talk about matrices in the form m x n matrix, with m indicating the number of rows, and n the number of columns. From a computer game point of view, we normally use 3 x 3 and 4 x 4 matrices.
- We also describe individual elements of a matrix by using 0 based indexing of columns and rows. For example, for a 3 x 3 matrix we define the individual components as follows:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}$$

- We can also define when two matrices are equal:

  Both matrices have the same dimensions (same size)

  All corresponding components of the matrices are equal

2.2 Matrix Addition & Subtraction

- As with vectors, we want to define useful operations for working with matrices. The first operations we will look at are addition and subtraction.
- Adding matrices is a simple operation, defined as follows:

  For two matrices of the same size, add the corresponding entries.

- So for example:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} (a_{00}+b_{00}) & (a_{01}+b_{01}) & (a_{02}+b_{02}) \\ (a_{10}+b_{10}) & (a_{11}+b_{11}) & (a_{12}+b_{12}) \\ (a_{20}+b_{20}) & (a_{21}+b_{21}) & (a_{22}+b_{22}) \end{bmatrix}$$

- Subtracting matrices is similar:

  For two matrices of the same size, add the corresponding entries

- So for example:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} - \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} (a_{00}-b_{00}) & (a_{01}-b_{01}) & (a_{02}-b_{02}) \\ (a_{10}-b_{10}) & (a_{11}-b_{11}) & (a_{12}-b_{12}) \\ (a_{20}-b_{20}) & (a_{21}-b_{21}) & (a_{22}-b_{22}) \end{bmatrix}$$

- The important aspect here is that addition and subtraction only work with matrices of the same dimensions (size)

Scalar Multiplication of Matrices

- As vectors, we have different methods to multiply matrices. The first method we are going to look at is scalar multiplication. This works in a similar way as we multiply vectors, and works for matrices of any size:

  For any matrix, to multiply by a scalar s, multiply the individual elements of the matrix by s

- Therefore:

$$sA = s\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} sa_{00} & sa_{01} & sa_{02} \\ sa_{10} & sa_{11} & sa_{12} \\ sa_{20} & sa_{21} & sa_{22} \end{bmatrix}$$

2.3 Matrix Multiplication

- Let us now look at how we multiply two matrices together. This is a very important calculation going forward, and you need to understand how matrix multiplication is calculated, and the properties of matrix multiplication.
- As an example, let us look at how we multiply two 2 x 2 matrices

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} (a_{00}b_{00} + a_{01}b_{10}) & (a_{00}b_{01} + a_{01}b_{11}) \\ (a_{10}b_{00} + a_{11}b_{10}) & (a_{10}b_{01} + a_{11}b_{11}) \end{bmatrix}$$

- OK, this does look a bit crazy. But there is a pattern. Let us put this differently:

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \times \begin{bmatrix} b_0 & b_1 \end{bmatrix} = \begin{bmatrix} a_0b_0 & a_0b_1 \\ a_1b_0 & a_1b_1 \end{bmatrix}$$

- If we consider each row of the first matrix to be a vector (e.g. $\begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$, and each of column of the second matrix to be a vector (e.g. $\begin{bmatrix} b_0 & b_1 \end{bmatrix}$, each entry of the resulting matrix is the dot product of the relevant row vector of matrix A and relevant column vector of matrix B.
- Because we are using the dot product of the row of matrix A and column of matrix B, then the number of columns in matrix A must equal the number of rows in matrix B. Or, put another way:

  To multiply two matrices, A and B, the dimensions of A are defined as m x n, and the dimensions of B are defined as n x p.

- The resulting matrix has a size of the number of rows in A by the number of columns in B. Or:

  The result of multiplying a m x n matrix A by a n x p matrix B is a m x p matrix C.

- Another important property of matrix multiplication to understand is that multiplication is not commutative. This is extremely important to understand when we look at transformations next week. Put simply:

  $(A)(B) \neq (B)(A)$

2.4 Matrix Transpose

- The transpose operation we will look at with matrices is the transpose operation. The transpose operation is useful in a number of graphical applications and allowing us to convert from the right-handed space of OpenGL to the left-handed space of DirectX.
- The transpose of a matrix is relatively simple. As an example, see how we transpose the 3 x 3 matrix below:

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} A^T = \begin{bmatrix} a_{00} & a_{10} & a_{20} \\ a_{01} & a_{11} & a_{21} \\ a_{02} & a_{12} & a_{22} \end{bmatrix}$$

- The general rule for calculating a matrix transpose for any size matrix is:

  For any size matrix A, each entry a_{mn} moves to a_{nm} in A^T

3 Transformations

3.1 What is a Transformation?

- A Transformation is a term used to describe the manipulation of objects that affects their geometry. For example we can transform geometric using the following transforms:

  Rotate

  Scale

  Translate (move)
- The term affine applied to transformation means that the shape of the object doesn't change when we transform it.
- Transformation of 3D objects is normally handled on the graphics card, and utilises matrix operations to perform the calculation. This is something the graphics card is designed to do, and can do very fast.

3.2 Translation

- Translation is the movement of objects to a different position.
- There are two methods we can use to translate an object - matrix addition and matrix multiplication. If you only plan to move an object, matrix addition will work by itself. If you want to perform other transformation operations at the same time, then you must use matrix multiplication.
- For the matrix multiplication technique we use the following matrix to translate a 4D vector:

$$T_{\mathbf{v}}\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + v_x \\ p_y + v_y \\ p_z + v_z \\ 1 \end{bmatrix} = \mathbf{p} + \mathbf{v}$$

3.3 Scaling

- Scaling is achieved via matrix multiplication
- A scaling can be represented by a scaling matrix. To scale an object by a vector $v =< v_x, v_y, v_z >$, each point $p =< p_x, p_y, p_z >$ would need to be multiplied with this scaling matrix:
$$S_v p = \begin{bmatrix} v_x & 0 & 0 \\ 0 & v_y & 0 \\ 0 & 0 & v_z \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} v_x p_x \\ v_y p_y \\ v_z p_z \end{bmatrix}.$$

3.4 Rotation

- Rotation is also achieved via matrix multiplication. For 2D calculations, there is only one matrix technique:
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$
- Rotation around the x, y, and z axis, we have:

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

3.5 Concatenation

- One advantage of using matrices is that you can combine the effects of two or more matrices by multiplying them. This means that, to rotate a model and then translate it to some location, you do not need to apply two matrices. Instead, you multiply the rotation and translation matrices to produce a composite matrix that contains all of their effects. This process, called matrix concatenation, can be written with the following formula.
$$C = (M1)(M2)..(MN)$$
In this formula, C is the composite matrix being created, and $M1$ through Mn are the individual transformations that matrix C contains. For example, $M1$ could be a translation matrix, and $M2$ could rotation matrix around the z-axis with the resulting matrix being a concatenation of both operations. In most cases, only two or three matrices are concatenated, but there is no limit.