# CPSC 314 - W22
# Notepack 1: Getting Started

Instructor: Dinesh K. Pai

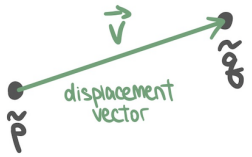
| Difference in Notations, this course vs textbook | | |
|---|---|---|
| | CPSC 314 | Textbook |
| Point | $\widetilde{\boldsymbol{p}}$ | $\widetilde{\boldsymbol{p}}$ |
| Vector | $\vec{\boldsymbol{v}}$ | $\vec{\boldsymbol{v}}$ |
| Column Matrix | $\bar{\mathsf{v}}$ | $\boldsymbol{v}$(bold) |
| Row Matrix | $\underline{\mathsf{v}}$ | $\boldsymbol{v}^T$ (bold) |
| Basis | $\underline{\vec{\boldsymbol{b}}}$ | $\vec{\boldsymbol{b}}^T$ (bold) |

## 1 Points, Vectors and Coordinates

Points $\neq$ Vectors $\neq$ Arrays

A **point** is a real location in space:

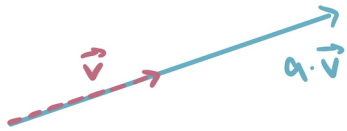

A **vector** is an algebraic object. A canonical example is the *displacement vector* from one point to another.



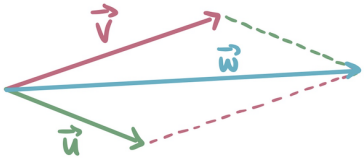

With vectors, we can:

1. Multiply with a scalar:

$$\text{a} \times \vec{\boldsymbol{v}} \rightarrow \text{a new vector}$$



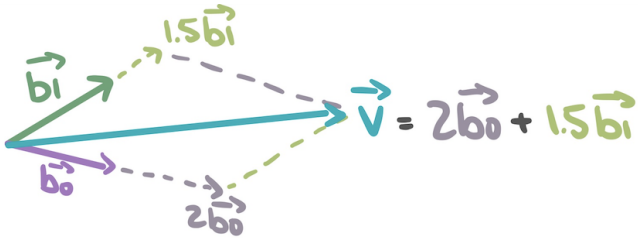

2. Add 2 vectors together




$$\vec{u} + \vec{v} = \vec{w}$$

## 1.1 Basis

A basis is an independent set of vectors that can produce any vector in a given space by linear combination. The size of the basis is the dimension of the space.




## 1.2 Coordinates of a Vector in a Basis $\underline{\vec{b}}$

We can write any vector $\vec{v}$ as:

$$\vec{v} = v_0 \times \vec{b}_0 + v_1 \times \vec{b}_1$$

$$\vec{v} \xrightarrow{\text{in basis } \underline{\vec{b}}} \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} = \bar{v}$$

**NOTATION**

$$\overline{\mathsf{v}} = \begin{pmatrix} v_0 \\ v_1 \end{pmatrix}$$

*column matrix*

$$\underline{\mathsf{v}} = \begin{pmatrix} v_0 & v_1 \end{pmatrix}$$

*row matrix*

Then, the basis can be written as:

$$\underline{\vec{\boldsymbol{b}}} = \begin{pmatrix} \vec{\boldsymbol{b}}_0 & \vec{\boldsymbol{b}}_1 \end{pmatrix}$$

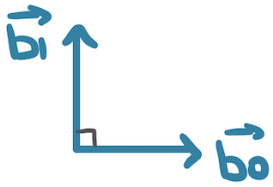
We can then write this very compactly

$$\vec{\boldsymbol{v}} = v_0 \times \vec{\boldsymbol{b}}_0 + v_1 \times \vec{\boldsymbol{b}}_1 = \begin{pmatrix} \vec{\boldsymbol{b}}_0 & \vec{\boldsymbol{b}}_1 \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} = \underline{\vec{\boldsymbol{b}}}\, \overline{\mathsf{v}}$$

, where $\vec{\boldsymbol{v}}$ is a real vector, $\underline{\vec{\boldsymbol{b}}}$ is a basis, and $\overline{\mathsf{v}}$ is the column matrix of coordinates of $\vec{\boldsymbol{v}}$ in basis $\underline{\vec{\boldsymbol{b}}}$.

## 1.3 Orthonormal Basis

Basis vectors are of unit length, and perpendicular to each other.




Computing the components of a vector is easier in an orthonormal basis.
Ex. To calculate the component along $\vec{\boldsymbol{b}}_0$, compute a dot product:

$$\vec{\boldsymbol{v}} \cdot \vec{\boldsymbol{b}}_0 = 1.1 \times \vec{\boldsymbol{b}}_0 \cdot \vec{\boldsymbol{b}}_0 + 0.9 \times \vec{\boldsymbol{b}}_0 \cdot \vec{\boldsymbol{b}}_1 = 1.1 \times 1 + 0.9 \times 0$$

## 1.4 Change of Basis

Suppose we have 2 basis, $\underline{\vec{\boldsymbol{a}}}$ and $\underline{\vec{\boldsymbol{b}}}$. Given that $\vec{\boldsymbol{v}} = \underline{\vec{\boldsymbol{b}}}\, \overline{\mathsf{v}}_b$, we want to find its coordinates in $\underline{\vec{\boldsymbol{a}}}$ instead (i.e., $\overline{\mathsf{v}}_a$).

- Remember that both coordinates represent the same physical vector:
  $\vec{\boldsymbol{v}} = \underline{\vec{\boldsymbol{b}}}\, \overline{\mathsf{v}}_b = \underline{\vec{\boldsymbol{a}}}\, \overline{\mathsf{v}}_a$

- Express $\underline{\vec{\boldsymbol{b}}} = \begin{pmatrix} \vec{\boldsymbol{b}}_0 & \vec{\boldsymbol{b}}_1 \end{pmatrix}$ in $\underline{\vec{\boldsymbol{a}}}$:
  $\vec{\boldsymbol{b}}_0 = \underline{\vec{\boldsymbol{a}}} \begin{pmatrix} l_{00} \\ l_{10} \end{pmatrix}$ and $\vec{\boldsymbol{b}}_1 = \underline{\vec{\boldsymbol{a}}} \begin{pmatrix} l_{01} \\ l_{11} \end{pmatrix}$
- The exchange from basis $\underline{\vec{\boldsymbol{a}}}$ to basis $\underline{\vec{\boldsymbol{b}}}$ is then:
  $\underline{\vec{\boldsymbol{b}}} = \underline{\vec{\boldsymbol{a}}} \begin{pmatrix} l_{00} & l_{01} \\ l_{10} & l_{11} \end{pmatrix} = \underline{\vec{\boldsymbol{a}}}\ \underline{\mathsf{L}}$
- Can finally express: $\underline{\vec{\boldsymbol{a}}}\ \underline{\mathsf{L}}\ \overline{\mathsf{v}_\mathsf{b}} = \underline{\vec{\boldsymbol{a}}}\ \overline{\mathsf{v}_\mathsf{a}}$

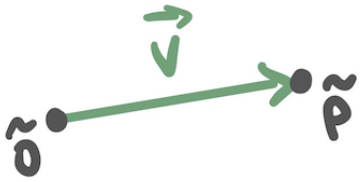
Therefore, the desired transformation of coordinates is

$$\boxed{\underline{\mathsf{L}}\ \overline{\mathsf{v}}_b = \overline{\mathsf{v}}_a}$$

*Tip: Think of a basis as a type of currency. Going from one basis to another is similar to converting a monetary amount from CAD to USD.*

## 1.5 Representing Points (and Vectors)

Given a vector $\vec{\boldsymbol{v}}$ and a basis $\underline{\vec{\boldsymbol{b}}}$, we have seen that we can obtain its coordinates in this basis. We can also say that $\vec{\boldsymbol{v}}$ represents the displacement from the origin $\widetilde{\mathbf{0}}$ (a special point) to another point $\widetilde{\boldsymbol{p}}$.




We can extend the + operation to allow the addition of a point and a vector, which will give us another point:

$$\widetilde{\boldsymbol{p}} = \vec{\boldsymbol{v}} + \widetilde{\mathbf{0}} = \vec{\boldsymbol{b}}_0 v_0 + \vec{\boldsymbol{b}}_1 v_1 + \widetilde{\mathbf{0}} = \begin{pmatrix} \vec{\boldsymbol{b}}_0 & \vec{\boldsymbol{b}}_1 & \widetilde{\mathbf{0}} \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ 1 \end{pmatrix} = \underline{\widetilde{\boldsymbol{b}}}\ \overline{\mathsf{p}}_m$$

...where $\underline{\widetilde{\boldsymbol{b}}} = \begin{pmatrix} \vec{\boldsymbol{b}}_0 & \vec{\boldsymbol{b}}_1 & \widetilde{\mathbf{0}} \end{pmatrix}$ is called a **coordinate frame** and $\overline{\mathsf{p}}_m = \begin{pmatrix} v_0 \\ v_1 \\ 1 \end{pmatrix}$ is the **homogenous coordinates** of a point (in 2D).

*Take-home question 1: How can we represent a vector using a coordinate frame?*

*Take-home question 2: What are the columns of $\underline{\widetilde{\boldsymbol{b}}}$ in 3D space?*

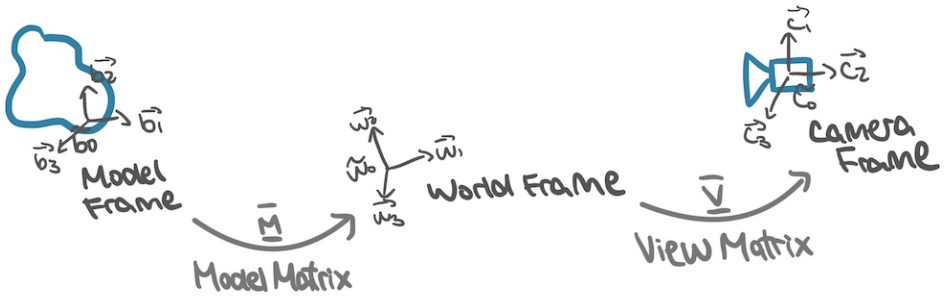
## 1.6 Three Important Frames

The three important frames are:

1. Model frame $\underline{\widetilde{\boldsymbol{b}}}$ (1 per object)
2. World frame $\underline{\widetilde{\boldsymbol{w}}}$
3. Camera frame $\underline{\widetilde{\boldsymbol{c}}}$ (1 per camera)

To go from one frame to another, we use transformation matrices:

1. Model matrix $\underline{\overline{\mathsf{M}}}$
2. View matrix $\underline{\overline{\mathsf{V}}}$




The transformation matrices can be inverted or combined with each other for different operations. For instance, the ModelView matrix is the concatenation of the Model and View matrices.

Ex. Suppose we have a point $\widetilde{\boldsymbol{p}}$ whose coordinates in its model frame are $\overline{\mathsf{p}}_m$. How can we obtain its coordinates in the world frame?

$\longrightarrow$ Answer: $\overline{\mathsf{p}}_w = \underline{\overline{\mathsf{M}}}\ \overline{\mathsf{p}}_m$

Recall that $\widetilde{\boldsymbol{p}} = \underline{\widetilde{\boldsymbol{b}}}\ \overline{\mathsf{p}}_m = \underline{\widetilde{\boldsymbol{w}}}\ \overline{\mathsf{p}}_w$. From the expression in the example above, we can multiply both sides by the world frame and obtain:

$$\underline{\widetilde{\boldsymbol{w}}}\ \overline{\mathsf{p}}_w = \underline{\widetilde{\boldsymbol{w}}}\ \underline{\overline{\mathsf{M}}}\ \overline{\mathsf{p}}_m = \widetilde{\boldsymbol{p}}$$

From this modified expression, we can see that $\underline{\widetilde{\boldsymbol{w}}}\ \underline{\overline{\mathsf{M}}} = \underline{\widetilde{\boldsymbol{b}}}$. We now have a method of converting 1 frame to another.

## 1.7 Anatomy of a Transformation Matrix

### 1.7.1 The Model Matrix $\underline{\overline{\mathsf{M}}}$

In 3D space, $\underline{\widetilde{\boldsymbol{b}}} = \begin{pmatrix} \vec{\boldsymbol{b}}_1 & \vec{\boldsymbol{b}}_2 & \vec{\boldsymbol{b}}_3 & \widetilde{\boldsymbol{b}}_0 \end{pmatrix}$ where $\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{b}}_3$ are the unit vectors that form the basis of the model frame and $\widetilde{\boldsymbol{b}}_{\mathbf{0}}$ is the origin of the model frame.

Since we know that $\underline{\widetilde{\boldsymbol{w}}}\,\underline{\overline{\mathsf{M}}} = \underline{\widetilde{\boldsymbol{b}}}$, we can see that the first 3 columns of the Model matrix must be the coordinates of $\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{b}}_2, \vec{\boldsymbol{b}}_3$ expressed in the world frame $\underline{\widetilde{\boldsymbol{w}}}$ (respectively). Similarly, the last column of the Model matrix must be the coordinates of the origin of the model frame expressed in the world frame.

The Model matrix has the form of:

$$\underline{\overline{\mathsf{M}}} = \begin{pmatrix} a & d & g & j \\ b & e & h & k \\ c & f & i & l \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

*Take-home question 3: Think about why the values in the last row are (0 0 0 1).*

### 1.7.2 The View $\underline{\overline{\mathsf{V}}}$ and Camera $\underline{\overline{\mathsf{C}}}$ Matrices

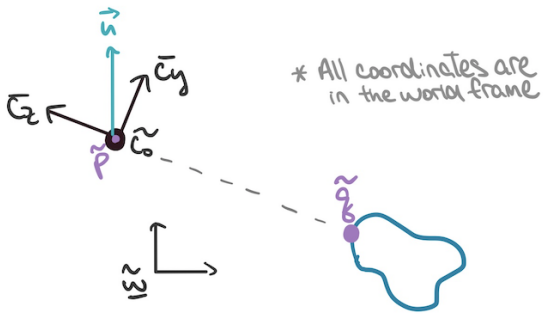
*Note: In Three.js, we can use the lookAt() function to find the View matrix, which is the inverse of the Camera matrix.*

The Camera matrix has the following form:

$$\underline{\overline{\mathsf{C}}} = \begin{pmatrix} \overline{\mathsf{c}}_x & \overline{\mathsf{c}}_y & \overline{\mathsf{c}}_z & \overline{\mathsf{c}}_0 \end{pmatrix}$$

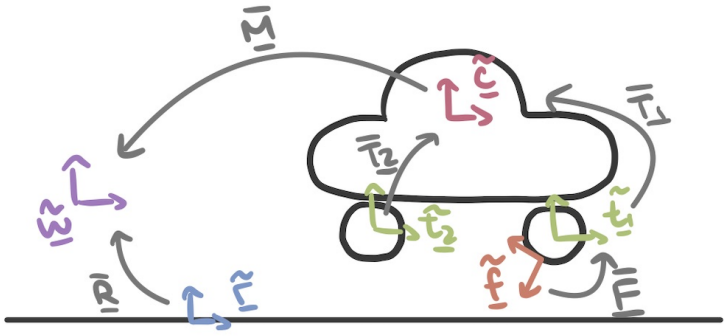
To find each column in $\underline{\overline{\mathsf{C}}}$:

1. The origin of the camera frame ($\overline{\mathsf{c}}_0$) is its coordinates in the world frame: $\overline{\mathsf{c}}_0 = \overline{\mathsf{p}}_w$
2. By convention, $\vec{\boldsymbol{c}}_z$ points away from the direction it is looking at. Thus, $\overline{\mathsf{c}}_z = normalize(\overline{\mathsf{p}} - \overline{\mathsf{q}})$
3. Pick a (user defined) vector $\vec{\boldsymbol{u}}$ to be the "up vector".
4. Pick $\vec{\boldsymbol{c}}_x$ to be perpendicular to the $\vec{\boldsymbol{u}}$ and $\vec{\boldsymbol{c}}_z$: $\overline{\mathsf{c}}_x = normalize(\overline{\mathsf{u}} \times \overline{\mathsf{c}}_z)$
5. Calculate $\vec{\boldsymbol{c}}_y$: $\overline{\mathsf{c}}_y = \overline{\mathsf{c}}_z \times \overline{\mathsf{c}}_x$



## 1.8 Scene Graphs and Hierarchies of Transformations

In Assignment 1, we placed several objects in the scene, such as an armadillo and a sphere. However, they are not connected in any way. In many cases, we want objects to move together, as shown in the following example:

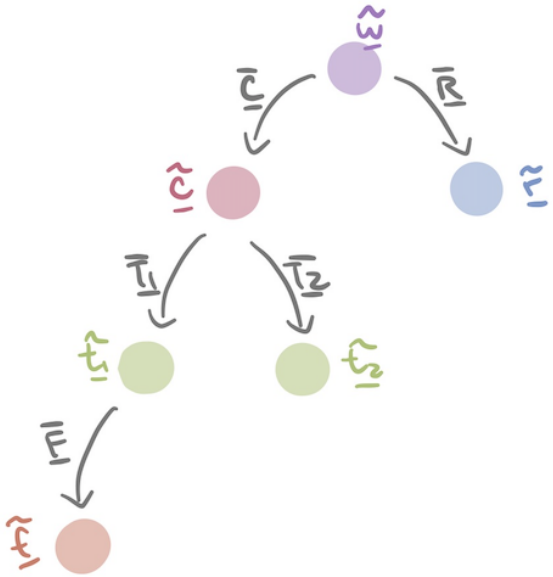


In this scene, a car is moving on the road. The tires of the car are moving along with the car, but they are also rotating. On one of the tires, there is a fly that is rotating with the tire.

Ex. Suppose we have a vertex on the fly with coordinates $\bar{\mathsf{p}}_f$ in the fly frame $\widetilde{\boldsymbol{f}}$. How can we find its coordinates $\bar{\mathsf{p}}_w$ in the world frame $\widetilde{\boldsymbol{w}}$ and what is the model matrix of the fly?

$\longrightarrow$ Answer: $\bar{\mathsf{p}}_w = \overline{\mathsf{C}}\ \overline{\mathsf{T}_1}\ \overline{\mathsf{F}}\ \bar{\mathsf{p}}_f$, where $\overline{\mathsf{C}}\ \overline{\mathsf{T}_1}\ \overline{\mathsf{F}}$ is the model matrix of the fly.

A scene graph is a data structure to represent the hierarchy of objects in the scene:



In three.js, the nodes in the graph above are called Object3D. Each Object3D stores its transformation to its parent `.matrix` and its transformation to the world frame `.matrixWorld`. In

three.js, the 'world' is also called a scene. We can add an edge to the scene graph by calling `Object3D.add()`.

*Take-home question 4: Is this matrixWorld the same as the model matrix?*

## 1.9 Interpreting Chains of Transformations (Optional)

In the car example, we can express any point $\widetilde{\boldsymbol{p}}$ in any of the frames:

$$\widetilde{\boldsymbol{p}} = \underline{\widetilde{\boldsymbol{w}}}\ \overline{\mathsf{p}}_w = \underline{\widetilde{\boldsymbol{c}}}\ \overline{\mathsf{p}}_c = \underline{\widetilde{\boldsymbol{t}}}\ \overline{\mathsf{p}}_t = \underline{\widetilde{\boldsymbol{f}}}\ \overline{\mathsf{p}}_f = ...$$

We have also seen that we can concatenate transformation matrices together:

$$\widetilde{\boldsymbol{p}} = \underline{\widetilde{\boldsymbol{w}}}\ \overline{\mathsf{p}}_w = \underline{\widetilde{\boldsymbol{w}}}\ \underline{\overline{\mathsf{C}}}\ \underline{\overline{\mathsf{T}_1}}\ \underline{\overline{\mathsf{F}}}\ \overline{\mathsf{p}}_f$$

There are two ways of interpreting this:

1. Transformation of coordinates from one frame to another:

$$\widetilde{\boldsymbol{p}} = \underline{\widetilde{\boldsymbol{w}}}\ (\underline{\overline{\mathsf{C}}}(\underline{\overline{\mathsf{T}_1}}(\underline{\overline{\mathsf{F}}}\ \overline{\mathsf{p}}_f))) = \underline{\widetilde{\boldsymbol{w}}}\ \underline{\overline{\mathsf{C}}}\ \underline{\overline{\mathsf{T}_1}}\ \overline{\mathsf{p}}_t = \underline{\widetilde{\boldsymbol{w}}}\ \underline{\overline{\mathsf{C}}}\ \overline{\mathsf{p}}_c = \underline{\widetilde{\boldsymbol{w}}}\ \overline{\mathsf{p}}_w$$

2. Moving any point (or vector) in a frame:

$$\widetilde{\boldsymbol{p}} = (((\underline{\widetilde{\boldsymbol{w}}}\ \underline{\overline{\mathsf{C}}})\underline{\overline{\mathsf{T}_1}})\underline{\overline{\mathsf{F}}})\ \overline{\mathsf{p}}_f = \underline{\widetilde{\boldsymbol{c}}}\ \underline{\overline{\mathsf{T}_1}}\ \underline{\overline{\mathsf{F}}}\ \overline{\mathsf{p}}_f = \underline{\widetilde{\boldsymbol{t_1}}}\ \underline{\overline{\mathsf{F}}}\ \overline{\mathsf{p}}_f = \underline{\widetilde{\boldsymbol{f}}}\ \overline{\mathsf{p}}_f$$

## 1.10 Transformation Sequences and Matrices

It is very important to pay attention to the order in which matrices are multiplied, because matrix multiplication is not commutative. We often apply sequences of transformations, with the order specified by the user or the problem. For example, if a car is first translated and then rotated, then the translation transformation must be applied before the rotation.

However, sometimes (e.g., when defining a Object3D), parts of a transformation matrix maybe specified separately. E.g., the position of the frame's origin can be set by setting an Object3D's `.position` property, and the rotation maybe set by changing the `.rotation` property. In this case, the order is not the order in which these properties were set, but we use a standard pattern for a series of simultaneous transformations:

$$\underline{\overline{\mathsf{M}}} = \underline{\overline{\mathsf{T}}}\ \underline{\overline{\mathsf{R}}}\ \underline{\overline{\mathsf{S}}}$$

...where $\overline{\underline{\mathsf{T}}}$ is the translation matrix, $\overline{\underline{\mathsf{R}}}$ is the rotation matrix, and $\overline{\underline{\mathsf{S}}}$ is the scaling matrix. By default, the `matrixAutoUpdate` propery is set to true, and Three.js will recalculate the matrix, in the same order, when any part is changed.

See `https://threejs.org/docs/#manual/en/introduction/Matrix-transformations`.

### 1.10.1 Translation

Translations are affine transformations. The translation matrix has the form:

$$\overline{\underline{\mathsf{T}}} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

...where $t_x, t_y$ and $t_z$ are the units of translation in the $x, y$ and $z$ directions, respectively.

*Take-home question 5: Why are translations considered affine transformations?*

### 1.10.2 Scaling

Scaling operations are linear transformations. The scaling matrix has the form:

$$\overline{\underline{\mathsf{S}}} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

...where $s_x, s_y$ and $s_z$ represent the amount of scaling in the $x, y$ and $z$ directions, respectively.

### 1.10.3 Rotation

Rotation operations are also linear transformations. The rotation matrix has the form:

$$\overline{\underline{\mathsf{R}}} = \begin{pmatrix} a & d & g & 0 \\ b & e & h & 0 \\ c & f & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

An important property of rotation matrices is that the operation preserves dot products between vectors:

$$u = Ru$$

$$||u|| = ||v|| = ||Ru||$$

$$||u|| = \sqrt{u_x^2 + u_y^2 + u_z^2} = \sqrt{u^T u}$$

$$u^T u = v^T R^T R v = v^T v$$

From the equation above, we can see that $R^T R = I$.

The $3x3$ matrix in the top-left corner of the rotation matrix depends on the angle, axis, and direction of rotation. For example, for a rotation $+\theta$, the $3x3$ matrices are:

$$Rot_x : \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}$$

$$Rot_y : \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

$$Rot_y : \begin{pmatrix} 0 & 0 & 1 \\ \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \end{pmatrix}$$

*Take-home question 6: How would you write the $3x3$ matrix for a rotation in the $-z$ direction?*