1 Matrix operations

The graphics and physics engine of Year Googol rely heavily on matrix computation. Your implementations in the Matrix class will involve the following operations.

1.1 Entry-wise Matrix Arithmetic

\[
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix} \times (3) \rightarrow \begin{bmatrix}
3 & 6 \\
9 & 12 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix} + \begin{bmatrix}
5 & 6 \\
7 & 8 \\
\end{bmatrix} \rightarrow \begin{bmatrix}
6 & 8 \\
10 & 12 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix} + \begin{bmatrix}
5 \\
7 \\
\end{bmatrix} \rightarrow \text{throws IncompatibleMatrixSizeException}
\]

\[
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix} - \begin{bmatrix}
4 & 3 \\
2 & 1 \\
\end{bmatrix} \rightarrow \begin{bmatrix}
-3 & -1 \\
1 & 3 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
12 \\
-7 \\
\end{bmatrix} \mod \begin{bmatrix}
5 \\
5 \\
\end{bmatrix} \rightarrow \begin{bmatrix}
2 \\
3 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 2 \\
\end{bmatrix} \cdot \begin{bmatrix}
4 \\
1 \\
2 \\
\end{bmatrix} \rightarrow 1 \cdot 4 + 0 \cdot 1 + 2 \cdot 2 = 8
\]
1.2 Sub-matrices

\[
\begin{bmatrix}
1 & 0 & 2 & 0 \\
0 & 3 & 0 & 4 \\
5 & 0 & 6 & 0 \\
0 & 7 & 0 & 8 \\
\end{bmatrix}
\]

\text{subMatrix}(1, 0, 2, 3) \rightarrow \begin{bmatrix} 0 & 3 & 0 \\ 5 & 0 & 6 \end{bmatrix}

\[
\begin{bmatrix}
1 & 0 & 2 & 0 \\
0 & 3 & 0 & 4 \\
5 & 0 & 6 & 0 \\
0 & 7 & 0 & 8 \\
\end{bmatrix}
\]

\text{subMatrix}(1, 0, 2, 5) \rightarrow \text{throws}
\begin{align*}
\text{MatrixOutOfBoundsException}(1,4,\text{this}) \\
\text{or} \\
\text{MatrixOutOfBoundsException}(2,4,\text{this})
\end{align*}

1.3 Matrix Multiplication

Entry \((i,j)\) of the matrix product \(A \times B\) is the dot-product of the \(i^{th}\) row of \(A\) and the \(j^{th}\) column of \(B\). The number of columns in \(A\) must equal the number of rows in \(B\). For example:

\[
\begin{bmatrix}
1 & 0 & 2 \\
0 & 2 & 3 \\
\end{bmatrix}
\times
\begin{bmatrix}
1 \\
0 \\
3 \\
\end{bmatrix}
\rightarrow
\begin{bmatrix} 1 \cdot 1 + 0 \cdot 0 + 2 \cdot 3 & 1 \cdot 0 + 0 \cdot 2 + 2 \cdot 1 \end{bmatrix}
= \begin{bmatrix} 7 & 2 \end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & 2 \\
0 & 2 & 3 \\
\end{bmatrix}
\times
\begin{bmatrix}
1 \\
0 \\
2 \\
\end{bmatrix}
\rightarrow
\text{throws IncompatibleMatrixSizeException}
\]

1.4 Matrix Equality

The limited precision of floating-point CPU arithmetic leads to small rounding errors that can accumulate over multiple operations. Two matrices should still be considered equal even if there are small deviations in their corresponding entries: Deviations less than \(\text{YearGoogol}.\text{PRECISION}\) are allowed. For example, if \(\text{YearGoogol}.\text{PRECISION}=0.01\):

\[
\begin{bmatrix} 0.001 & 0.505 \end{bmatrix}
\text{.equals}(\begin{bmatrix} -0.001 & 0.507 \end{bmatrix}) \rightarrow \text{true}
\]

\[
\begin{bmatrix} 0.001 & 0.505 \end{bmatrix}
\text{.equals}(\begin{bmatrix} -0.001 & 0.515 \end{bmatrix}) \rightarrow \text{false}
\]
1.5 Plane Rotations

A matrix can be used to represent rotation of points in a plane. A plane rotation matrix contains 1 in each diagonal entry, and 0 everywhere else, except at four special entries. If the plane corresponds to the $a_1^{th}$ coordinate axis and $a_2^{th}$ coordinate axis, and the angle of rotation is $\theta$, then:

- entry $(a_1, a_1) = \cos \theta$
- entry $(a_1, a_2) = -\sin \theta$
- entry $(a_2, a_1) = \sin \theta$
- entry $(a_2, a_2) = \cos \theta$

For example, in 3D space, the $0^{th}$ axis corresponds to $x$, and the $1^{st}$ axis corresponds to $y$. So a rotation of $\theta$ in the $xy$ plane is given by:

$$\text{Matrix.planeRotation}(3,0,1, \theta) \rightarrow \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In 4D space, a rotation of $\theta$ in the $xz$ plane ($0^{th}$ and $2^{nd}$ axes) is given by:

$$\text{Matrix.planeRotation}(4,0,2, \theta) \rightarrow \begin{bmatrix} \cos \theta & 0 & -\sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.6 Unit hyper-cubes

Unit hyper-cubes are used to render the Exoverse. The $n$-dimensional unit hyper-cube has $2^n$ vertices. The vertex coordinates are precisely the bits of the binary numbers 0 through $2^n - 1$. This is illustrated in Figure 1. The vertices can be organized as the columns of a matrix, where entry $(i, j)$ contains the $i^{th}$ bit of the binary number $j$ (ordered from least significant bit to most significant). For example, the 3D cube has associated matrix:

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

The $i^{th}$ bit of $j$ can be found via the formula

$$\frac{j}{2^i} \mod 2$$

or equivalently with the bit-shift operators.
Figure 1: The 3-dimensional unit “hyper”-cube. Vertex coordinates are precisely the bits of the binary numbers 0 through $2^3 - 1 = 7$.

### 1.7 2D Vectors

The `Vec2D` class is used to represent positions, velocities, forces, and paths in the Year Googol plane. This class has been implemented for you. In fact, `Vec2D` is a sub-class of `Matrix`: A `Vec2D` object is a `Matrix` with 2 rows and 1 column. It can do everything a matrix can do, including addition, scalar multiplication, dot-product, and modulo. See Figure 2 for an illustration of vector arithmetic. `Vec2D` also contains some additional functionality, documented in the API, which will come in handy. For example:

\[
\begin{bmatrix} 3 \\ 4 \end{bmatrix} \cdot \text{dotSelf()} \rightarrow 25 \\
\begin{bmatrix} 3 \\ 4 \end{bmatrix} \cdot \text{norm()} \rightarrow 5 \\
\begin{bmatrix} 3 \\ 4 \end{bmatrix} . \text{normalize}(10) \rightarrow \begin{bmatrix} 6 \\ 8 \end{bmatrix} \\
\begin{bmatrix} 3 \\ 4 \end{bmatrix} . \text{normalize()} \rightarrow \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix}
\]

The `norm` of a vector $v$ is its length, denoted $|v|$.

### 2 Year Googol Physics

Several formulae in this section involve a very short time interval, which is denoted by $\Delta t$. This time interval is a small fraction of a second. For your project, the value of $\Delta t$ is a constant, stored in the variable `YearGoogol.DELTA_T`.

Particle positions, velocities, forces, and paths are 2-dimensional vectors, indicated in **bold**, and represented in your project by `Vec2D` objects.
2.1 drift

Given:

- \( \mathbf{v} \): The current particle velocity
- \( \mathbf{p}^{\text{(orig)}} \): The initial particle position

The updated position is:

\[
\mathbf{p}^{\text{(new)}} = \mathbf{p}^{\text{(orig)}} + \mathbf{v} \Delta t
\]

The expression \( \mathbf{v} \Delta t \) means that the vector \( \mathbf{v} \) is multiplied by the scalar \( \Delta t \).

The Year Googol plane wraps around on itself: If a particle drifts off one edge of the view, it reappears on the opposite side. The size of the Year Googol plane is stored in the variable \( \text{YearGoogol}.\text{SIZE} \). This variable is a \text{Vec2D} object, whose first component is the width and whose second component is the height. See Figure 3 for an example of wrapping.

2.2 apply

Given:

- \( \mathbf{f} \): The applied force
- \( m \): The particle mass
- \( \mathbf{v}^{\text{(orig)}} \): The initial particle velocity

The updated velocity is:

\[
\mathbf{v}^{\text{(new)}} = \mathbf{v}^{\text{(orig)}} + \frac{\mathbf{f} \Delta t}{m}
\]
Figure 3: Wrapping. In this example, \texttt{YearGoogol.SIZE} is (800,600). If \( p^{(\text{new})} \) is found to lie out of bounds, it must be replaced with the corresponding position after wrapping. The \texttt{Vec2D mod(...)} method is designed expressly for this purpose.
Figure 4: Shortest-path vectors. In this example, YearGoogol.SIZE is (800, 600). The shortest path from a to b is not simply \( b - a = (650, 350) \). There is a shorter path if one moves up and to the left, and wraps around: (-150, -250).

2.3 Shortest-path vectors

In the ordinary cartesian plane, the shortest-path vector from position a to b is the vector difference:

\[ b - a \]

However, this formula is not sufficient for the Year Googol plane, which wraps around on itself. The shortest-path vector must allow the possibility of moving off one edge of the plane and reappearing on the other side. This is illustrated in Figure 4. The shortest-path vector from particle A to particle B is denoted \( \mathbf{r}_{AB} \).

2.4 Gravitational force

For any pair of particles A and B, each exerts a gravitational force on the other. The strength of this force depends on a physical quantity called the “gravitational constant,” which is denoted by \( G \). For your project, the value of this constant is stored in YearGoogol.G.
Given:

• \( m_A \) and \( m_B \): The masses of \( A \) and \( B \), respectively.
• \( r_{BA} \): The shortest-path vector from \( B \) to \( A \).

The force that \( A \) exerts on \( B \) is calculated as follows:

1. Compute the magnitude of the gravitational force:
   \[
   \|f_{AB}\| = G \frac{m_A m_B}{\|r_{BA}\|^2}
   \]

2. Normalize \( r_{BA} \) to have the foregoing magnitude. The normalized version is \( f_{AB} \).

The force that \( B \) exerts on \( A \) is equal and opposite to the force that \( A \) exerts on \( B \):

\[
f_{BA} = -f_{AB}
\]

### 2.5 Particle intersection

Two particles \( A \) and \( B \) intersect when the distance between them is shorter than their summed radii. In other words, intersections may be detected as follows:

1. Compute the shortest-path vector from \( B \) to \( A \): \( r_{BA} \).
2. Compute the magnitude of this vector: \( \|r_{BA}\| \)
3. Compute the sum of particle radii: \( R = r_A + r_B \).
4. If \( \|r_{BA}\| < R \), then the particles intersect.

This is illustrated in Figure 5.

### 2.6 Collision force

When two particles \( A \) and \( B \) collide, each exerts a very large force on the other, over a very short time interval.

Given:

• \( m_A \) and \( m_B \): The masses of \( A \) and \( B \), respectively.
• \( v_A \) and \( v_B \): The velocities of \( A \) and \( B \), respectively.

The force that \( A \) exerts on \( B \) can be calculated as follows:

1. Compute the “reduced mass” of the system:
   \[
   \mu = \frac{m_A m_B}{m_A + m_B}
   \]
Figure 5: Particle intersection. These particles do not intersect, since the sum of radii \( r_A + r_B \) is less than the length of the shortest-path vector: \( \| r_{AB} \| \).

2. Compute the shortest-path vector from \( B \) to \( A \): \( r_{BA} \).

3. Compute the “line of collision” \( c_{BA} \), which is the shortest-path vector, normalized to length 1.

4. Compute the “scalar velocities” along the line of collision:
   \[
   u_A = v_A \cdot c_{BA} \\
   u_B = v_B \cdot c_{BA}
   \]
   Here \( \cdot \) denotes dot-product.

5. The force that \( A \) exerts on \( B \) is given by:
   \[
   f_{AB} = c_{BA} \frac{2\mu(u_A - u_B)}{\Delta t}
   \]
   The force that \( B \) exerts on \( A \) is equal and opposite to the force that \( A \) exerts on \( B \):
   \[
   f_{BA} = -f_{AB}
   \]