Practical Work 2

Homogeneous Coordinates

This practical work is designed to illustrate the use of Homogeneous Coordinates for representing 3D points, transformations and calculus.

Preparation

We're using the Python language and the MatPlotLib library. The MatPlotLib library can be installed with the command:

```
pip install matplotlib
```

or for conda installation:

```
conda install -c conda-forge matplotlib
```

The MatPlotLib library displays geometric rendering in an independent interactive window. Depending on the Python environment, this window may be rendered as a frozen image, preventing user interaction. To correct this problem, here are a few solutions:

Jupyter Notebook:

Execute following code within the Jupyter Notebook:

```
%matplotlib qt
```

PyCharm

Go to `Settings / Tool / Python Plot` and uncheck the option `Show plots in tool windows`.

Spyder

Go to `Tools / Preferences / IPython console / Graphics / Backend:Inline` and change "`Inline`" to "`Automatic`". Click OK button and restart the IDE.

Julien SEINTURIER
http://www.seinturier.fr / julien.seinturier@univ-tln.fr

Representing Vectors and Matrices

This work relies on Numpy for the vector and matrix representation. The Numpy library can be integrated within Python program with the import:

```
import numpy as np
```

Representing points and vectors

Points can be represented as Numpy array. Creating a point $p = (x, y, z)$ can be done using the python instruction:

```
p = np.array([x, y, z])
```

Simple operations

Let $p = (x, y, z)$ and $q = (t, u, v)$ two points, the result of adding, subtracting, or multiplying p and q are given respectively by:

```
p = np.array([x, y, z])

q = np.array([t, u, v])

sum = p + q

diff = p - q

mult = p * q
```

Dot and cross product

Let $p = (x, y, z)$ and $q = (t, u, v)$ two vectors, the result of scalar product (dot) or vectorial product (cross) are given respectively by:

```
p = np.array([x, y, z])

q = np.array([t, u, v])

dot = p.dot(q)

cross = np.cross(p, q)
```

Exercise 1

Create a python program that define two points a = (1, 0, 0) and b = (0, 1, 0) and that computes display: a+b, a-b, a*b, $a \cdot b$, and $a \times b$

Representing matrix

Matrix can be represented as Numpy arrays. Let the matrix M with l lines and c columns defined such as:

$$M = \begin{bmatrix} m_{00} & \cdots & m_{0j} & \cdots & m_{0c} \\ \vdots & \ddots & \vdots & & \vdots \\ m_{i0} & \cdots & m_{ij} & \cdots & m_{ic} \\ \vdots & & \vdots & \ddots & \vdots \\ m_{l0} & \cdots & m_{lj} & \cdots & m_{lc} \end{bmatrix}$$

The representation of M within a python program is given by:

```
M = np.array(((m00, …, m0c), …, (mi0, …, mic), …, (ml0, …, mlc)))
```

Exercice 2

Create a python program that define and display the matrix:

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

Matrix / vector multiplication

Let M a matrix and V a vector such as:

$$M = \begin{bmatrix} m_{00} & \cdots & m_{0j} & \cdots & m_{0c} \\ \vdots & \ddots & \vdots & & \vdots \\ m_{i0} & \cdots & m_{ij} & \cdots & m_{ic} \\ \vdots & & \vdots & \ddots & \vdots \\ m_{l0} & \cdots & m_{lj} & \cdots & m_{lc} \end{bmatrix}, \text{and } V = \begin{bmatrix} v_0 \\ \vdots \\ v_j \\ \vdots \\ v_c \end{bmatrix}$$

The multiplication of V by M, denoted $R = MV$, is given by:

```
V = np.array([v0, …, vj, …, vc])
M = np.array(((m00, …, m0c), …, (mi0, …, mic), …, (ml0, …, mlc)))
R = M.dot(V)
```

Exercise 3

Write a python program that compute and display the matrix / vector product:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

Matrix / matrix multiplication

Let M and K two matrices such as:

$$M = \begin{bmatrix} m_{00} & \cdots & m_{0j} & \cdots & m_{0c} \\ \vdots & \ddots & \vdots & & \vdots \\ m_{i0} & \cdots & m_{ij} & \cdots & m_{ic} \\ \vdots & & \vdots & \ddots & \vdots \\ m_{l0} & \cdots & m_{lj} & \cdots & m_{lc} \end{bmatrix}, \text{ and } K = \begin{bmatrix} k_{00} & \cdots & k_{0t} & \cdots & k_{0c} \\ \vdots & \ddots & \vdots & & \vdots \\ k_{j0} & \cdots & k_{jt} & \cdots & k_{jc} \\ \vdots & & \vdots & \ddots & \vdots \\ k_{c0} & \cdots & k_{ct} & \cdots & k_{ct} \end{bmatrix}$$

The multiplication of K by M, denoted $R = MK$, is given by:

```
M = np.array(((m₀₀, …, m₀c), …, (mᵢ₀, …, mᵢc), …, (mₗ₀, …, mₗc)))
K = np.array([v₀, …, vⱼ, …, vc])
R = M.dot(V)
```

Exercise 4

Write a python program that compute and display the matrix / vector product:

$$R = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 2 & 0 & 4 \\ 0 & 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} 4 & -1 \\ 3 & 2 \\ 2 & -5 \\ 6 & 4 \end{bmatrix}$$

Transformations within homogeneous coordinates

We are now focusing on 3D point transformation implementation and display.

Representing Point and Vector

Let $P = (x, y, z)$ be a point expressed within Euclidean Space with Cartesian Coordinates. A representation of P within Homogeneous coordinates is given by:

$$P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Let Q a vector expressed within Homogeneous coordinates such as:

$$Q = \begin{bmatrix} t \\ u \\ v \\ w \end{bmatrix}$$

Q can be represented within Euclidean Space as $Q = \left(\frac{t}{w}, \frac{u}{w}, \frac{v}{w}\right)$

Exercise 5

Create a Python program named `homogeneous.py` and implement a function `toHomogeneous(v: tuple) -> np.array` that convert the given tuple expressed within Euclidean Space to its representation within Homogeneous Coordinates. Test the program by converting the point $(1.0, 2.0, 3.0)$ to Homogeneous Coordinates.

Exercise 6

Within `homogeneous.py`, add a function `toEuclidean(v: tuple) -> np.array` that convert the given tuple expressed within Homogeneous Coordinates to its representation within Euclidean Space. Test the program by converting the point $(2.0, 4.0, 6.0, 2.0)$ to Euclidean Space.

Translation

Let $P = (x, y, z)$ a 3D point. A translation is an application, denoted $T(\alpha, \beta, \gamma)(P)$ that can be represented within Homogeneous coordinates by:

$$T(\alpha,\beta,\gamma) = \begin{bmatrix} 1 & 0 & 0 & \alpha \\ 0 & 1 & 0 & \beta \\ 0 & 0 & 1 & \gamma \\ 0 & 0 & 0 & 1 \end{bmatrix}, with\ P = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Exercise 7

Within `homogeneous.py`, add a function `translate_point_hc(point, alpha, beta, gamma)` that takes in parameter an array representing the vector `[x, y, z, w]` and that return an array that represents the translated vector along vector (α, β, γ).

Test the function translate by displaying the point `(4.0, 3.0, 2.0)` and by displaying the result of the translation along vector `(0.0; 1.0, 1.0)`.

Rotation

Let $P = (x, y, z)$ a 3D point. A rotation is an application, denoted $R_i(\theta)(P)$ that rotate the point P around the axis i by an angle θ. A rotation can be defined within Homogeneous Coordinates.

The rotation $R_x(\omega)$ around X axis by an angle ω is defined such as:

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

The rotation $R_y(\varphi)$ around Y axis by an angle φ is defined such as:

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

The rotation $R_z(\kappa)$ around Z axis by an angle κ is defined such as:

$$R_z(\kappa) = \begin{bmatrix} \cos(\kappa) & -\sin(\kappa) & 0 & 0 \\ \sin(\kappa) & \cos(k) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Exercise 6

Write a function `rot_x_point(point, omega)` that takes in parameter a point represented by a tuple `(x, y, z)` and returns the tuple `(xr, yr, zr)` that represents the rotated point around X axis by an angle omega.

Test the function by displaying the point `(4.0, 4.0, 4.0)` as a black circle and displaying the result of its rotation by an angle of $\frac{\pi}{4}$ as a red circle.

Exercise 7

Write a function `rot_y_point(point, phi)` that takes in parameter a point represented by a tuple `(x, y, z)` and returns the tuple `(xr, yr, zr)` that represents the rotated point around Y axis by an angle `phi`.

Test the function by displaying the point `(4.0, 4.0, 4.0)` as a black circle and displaying the result of its rotation by an angle of $\frac{\pi}{4}$ as a green circle.

Exercise 8

Write a function `rot_z_point(point, kappa)` that takes in parameter a point represented by a tuple `(x, y, z)` and returns the tuple `(xr, yr, zr)` that represents the rotated point around Z axis by an angle `kappa`.

Test the function by displaying the point `(4.0, 4.0, 4.0)` as a black circle and displaying the result of its rotation by an angle of $\frac{\pi}{4}$ as a blue circle.

The global rotation of a point within a 3D space is obtained by applying the three rotations around the X, Y and Z axis. Let $P = (x, y, z)$ a 3D point, the rotation of P around the X, Y and Z axis by the angles ω, φ, κ is such that:

$$P_r = R_z(\omega) \circ R_y(\varphi) \circ R_x(\kappa)(P)$$

Exercise 9

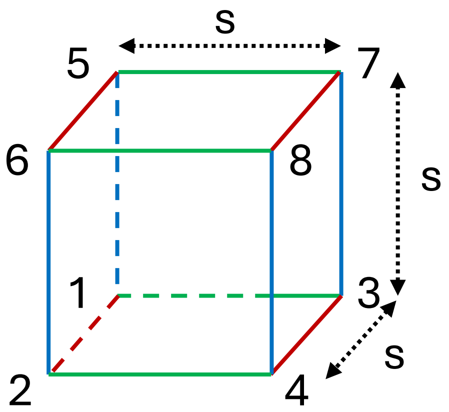
Write a function `rot_point(point, omega, phi, kappa)` that takes in parameter a point represented by a tuple `(x, y, z)` and returns the tuple `(xr, yr, zr)` that represents the rotated point around X, Y and Z axis by the angles `omega`, `phi`, `kappa` respectively.

Test the function by displaying the point `(4.0, 4.0, 4.0)` as a black circle and displaying the result of its rotation by three angles of $\frac{\pi}{4}$ as an orange right cross.

Ensure that when using only one angle value (by setting others to 0), the behavior of `rot_point` is the same as `rot_x_point`, `rot_y_point` and rot_z-point.

Working with shape

For the rest of the work, a cube is represented as Python array of 8 points corresponding to its vertices.



according to the figure below, a cube of size s is defined by the following array:

[(-s, -s, -s), (s, -s, -s), (-s, s, -s), (s, s, -s), (-s, -s, s), (s, -s, s), (-s, s, s), (s, s, s)]

1 2 3 4 5 6 7 8

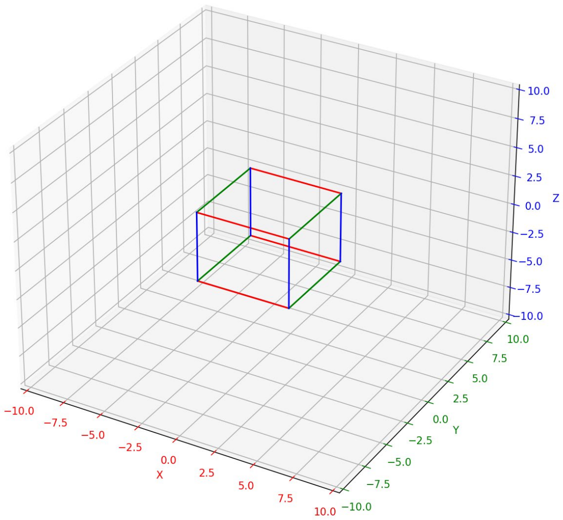
Exercise 10

Write a function `cube(size)` that takes in parameter a `size` and create a cube represented by an array of 8 tuples corresponding to its vertices. The vertices order has to respect the figure above.

Exercise 11

Write a function `display_cube(vertices)` that take in parameter an array of tuples that represent the `vertices` of a cube and that display the cube within the 3D environment.

Each edge of the cube has to be colorized with the same color as its parallel axis (see image below).



Julien SEINTURIER
http://www.seinturier.fr / julien.seinturier@univ-tln.fr

Translation

Translating a shape can be done by translating all its vertices.

Exercise 12

Write a function `translate_cube(vertices, alpha, beta, gamma)` that takes in parameter an array of tuples that represent the `vertices` of a cube and that return an array of tuples that represent the vertices of the translated cube along vector (α, β, γ).

Test the function `translate_cube` by displaying the result of the translation of a cube of size 3.0 along vector `(1.0; 2.0, 3.0)`.

Rotation

Rotating a shape can be done by rotating all its vertices.

Exercise 13

Write a function `rotate_cube(vertices, omega, phi, kappa)` that takes in parameter an array of tuples that represent the `vertices` of a cube and three rotation angles `omega, phi, kappa` and that rotate the cube according to the given angles.

Test the function `rotate _cube` by displaying the result of the rotation of a cube of size 3.0 for the angles $\omega = \frac{\pi}{4}$, $\varphi = \frac{\pi}{3}$ and $\kappa = \frac{\pi}{2}$.

Merging transformations

Translation and rotation can be combined in order to locate shapes.

Exercice 14

Using previous functions, create a cube with a size of 2.0 and display simultaneously

- The cube transformed by a translation $T(\alpha, \beta, \gamma)$ where $\alpha = 0.25$, $\beta = 0.50$ and $\gamma = 0.75$ then a rotation $R_z(\omega) \circ R_y(\varphi) \circ R_x(\kappa)$ where $\omega = \frac{\pi}{6}$, $\varphi = \frac{\pi}{4}$ and $\kappa = \frac{\pi}{3}$
- The cube transformed by a rotation $R_z(\omega) \circ R_y(\varphi) \circ R_x(\kappa)$ where $\omega = \frac{\pi}{6}$, $\varphi = \frac{\pi}{4}$ and $\kappa = \frac{\pi}{3}$ then a translation $T(\alpha, \beta, \gamma)$ where $\alpha = 0.25$, $\beta = 0.50$ and $\gamma = 0.75$

Do the two transformed cube share the same location? Why?