Assignment 01: Python Basics

Erich Flock 117954
Anton Kliuyeu 118026

P.S. the console output is encoded with a grey font color.
All the code is functional. You can try to run the ./main.py to make sure that the Matrix and Vector framework works correctly. We've used the selfmade framework to answer some questions in this assignment.

Exercise 1.1

```
a = TMatrix(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)
b = TMatrix(1,3,5,7,9,11,13,15,2,4,6,8,10,12,14,16)
c = a.mult(b)
```

```
c = [
        152, 168, 184, 200,
        376, 424, 472, 520,
        180, 200, 220, 240,
        404, 456, 508, 560
]
```

Exercise 1.2

```
a = TMatrix()
a = a.make_trans_mat(1, 2, 3)
a = a.make_rot_mat(45, 'x')
a = a.make_rot_mat(90, 'y')
a = a.make_rot_mat(120, 'z')
a = a.make_scale_mat(1, 2, 3)
```

```
a = [
        -3.061616997868382e-17, -0.9659258262890682, 0.25881904510252085, 0.0,
         1.0605752387249069e-16, 0.5176380902050417, 1.9318516525781364, 0.0,
         -3.0, 1.2989340843532398e-16, 1.29893408435324e-16, 0.0,
         1.0, 2.0, 3.0, 1.0
]
```

Exercise 1.3

```
a = euclidean_distance(
        Vector4(2, 4, 6, 2),
        Vector4(0, 0, 0, 1))
```

```
a = 7.48331477355
```

Exercise 1.4

```
a = TMatrix(
        -3.061616997868382e-17, -0.9659258262890682, 0.25881904510252085, 0.0,
         1.0605752387249069e-16, 0.5176380902050417, 1.9318516525781364, 0.0,
         -3.0, 1.2989340843532398e-16, 1.29893408435324e-16, 0.0,
         1.0, 2.0, 3.0, 1.0
)

c = a.mult_vec(Vector4(1,2,3,1))

c = [
        -8.0,
        2.0693503541210156,
        7.1225223502587935,
        1.0
]
```

Exercise 1.5

The base Node object contains all the informations about for example transformation matrix, name of the node, who is a parent of this Node and so on... The TransformNode is a child (inheritance) of the parent Node object. The TransformNode may not have additionally functionality, but it's necessary to keep constistency of the framework, because there are other instruments to design a scene. The TriMeshNode class that is a child class of Node too, contains a Mesh (defined with 3D vectors). This Mesh will be multiplied with the Matrix of Node so that the Object can be manipulated. Other instances of the Node object are for example CameraNode, LightNode and ScreenNode, that have it's own special functionality that is based on the parent class. For example the CameraNode contains a Camera Matrix that distorts the whole scene for a perspective view. The ScreenNode defines a virtual Viewport. The Light it's a mental abstraction that creates a virtual light source interpreted by shaders.

Exercise 1.6

The WorldTransform of a Node is a multiplication result of all matrices on every branch from this Node to it's very first Parent. It defines the absolut location of this Node. On other side the Transform Matrix defines a local transformation on this Object like rotation of the Moon around the Earth (that rotates around sun).

Exercise 1.7

First, we going to calculate the local position of all the objects. After this we will calculate the World Transformation Matrix. As you see, we need to multiply all the matrices from the top to bottom to give our object the correct absolute/global position.

LOCAL MATRIX CALCULATION

```
root = TMatrix()

cycle_transform = TMatrix()
cycle_transform = cycle_transform.make_trans_mat(5, 0, -2)
cycle_transform = cycle_transform.make_rot_mat(90, 'y')

cycle_transform = [
        6.123233995736766e-17, 0.0, 1.0, 0.0,
        0, 1, 0, 0,
        -1.0, 0.0, 6.123233995736766e-17, 0.0,
        5, 0, -2, 1
]

body_geom = TMatrix()
body_geom = body_geom.make_rot_mat(90, 'y')

body_geom = [
        6.123233995736766e-17, 0.0, 1.0, 0.0,
        0, 1, 0, 0,
        -1.0, 0.0, 6.123233995736766e-17, 0.0,
        0, 0, 0, 1
]

wheel1_trans = TMatrix()
wheel1_trans = wheel1_trans.make_trans_mat(-2, -1.5, 1)

wheel2_trans = TMatrix()
wheel2_trans = wheel2_trans.make_trans_mat(2, -1.5, 1)

wheel2_trans = [
        1, 0, 0, 0,
        0, 1, 0, 0,
        0, 0, 1, 0,
        2.0, -1.5, 1.0, 1.0
]

wheel1_geom = TMatrix()
wheel1_geom = wheel1_geom.make_scale_mat(.5, .5, .5)

wheel2_geom = TMatrix()
wheel2_geom = wheel2_geom.make_scale_mat(.5, .5, .5)

wheel2_geom = [
        0.5, 0.0, 0.0, 0.0,
        0.0, 0.5, 0.0, 0.0,
        0.0, 0.0, 0.5, 0.0,
        0, 0, 0, 1
]
```

GLOBAL MATRIX CALCULATION

```
cycle_transform_WT = TMatrix()
        .mult(root)
        .mult(cycle_transform)

cycle_transform_WT = [
        6.123233995736766e-17, 0.0, 1.0, 0.0,
        0, 1, 0, 0,
        -1.0, 0.0, 6.123233995736766e-17, 0.0,
        5, 0, -2, 1
]

body_geom_WT = TMatrix()
        .mult(root)
        .mult(cycle_transform)
        .mult(body_geom)

body_geom_WT = [
        -1.0, 0.0, 1.2246467991473532e-16, 0.0,
        0.0, 1.0, 0.0, 0.0,
        -1.2246467991473532e-16, 0.0, -1.0, 0.0,
        5.0, 0.0, -2.0, 1.0
]

wheel1_trans_WT = TMatrix()
        .mult(root)
        .mult(cycle_transform)
        .mult(wheel1_trans)

wheel2_trans_WT = TMatrix()
        .mult(root)
        .mult(cycle_transform)
        .mult(wheel2_trans)

wheel2_trans_WT = [
        6.123233995736766e-17, 0.0, 1.0, 0.0,
        0.0, 1.0, 0.0, 0.0,
        -1.0, 0.0, 6.123233995736766e-17, 0.0,
        4.0, -1.5, 0.0, 1.0
]

wheel1_geom_WT = TMatrix()
        .mult(root)
        .mult(cycle_transform)
        .mult(wheel1_trans)
        .mult(wheel1_geom)

wheel2_geom_WT = TMatrix()
        .mult(root)
        .mult(cycle_transform)
        .mult(wheel2_trans)
        .mult(wheel2_geom)

wheel2_geom_WT = [
        3.061616997868383e-17, 0.0, 0.5, 0.0,
        0.0, 0.5, 0.0, 0.0,
        -0.5, 0.0, 3.061616997868383e-17, 0.0,
        4.0, -1.5, 0.0, 1.0
]
```

Exercise 1.8

To represent the influence of rotation and scale, we are placing an initial vector with the direction of (0, -1) on the coordinate's origin.