Assignment 04: World View Projection

In Assignment 04 you have to create the world, view and projection matrices to support a simple truck simulation application. The simulator, will allow the user to experience driving both in first person (from the inside of the truck), and in third person (looking the truck from outside). For reason that will be explained in a future lesson, an application needs both the world-view-projection matrix and the world matrix alone to correctly compute the colors of points of the objects. The simulation then requires two types of procedures: one to compute the view-projection matrices, and another to create the world matrix. Inside the code that supports the assignment, the application will then make the product of the two to create the required world-view-projection matrix.

Since the application supports both a first and a third person view, it requires the constructions of the view matrix with either the look-in-direction, and the look-at models. Moreover, since all objects have been already modeled with the right size, scale transformation in the world matrix is not required (it can be considered always equal to 1.0).

You should then implement, inside file **`WVP.hpp`** , the following three functions:

```
glm::mat4 MakeViewProjectionLookInDirection(glm::vec3 Pos, float Yaw, float Pitch, float Roll, float FOVy, float Ar, float nearPlane, float farPlane)
```

Crates a *View-Projection matrix*, where the view matrix uses the *look-in-direction* model, and the projection matrix supports perspective. Parameters are the following:

1. Perspective matrix:
   - The vertical field of view *Fov-y* is defined in formal parameter **`float FOVy`** .
   - The *aspect ratio* is contained in **`float Ar`** .
   - *Near Plane* distance defined in **`float nearPlane`** .
   - *Far Plane* distance is passed into **`float farPlane`** .
2. View matrix (defined with the look-in-direction model):
   - The *position of the camera* is defined in formal parameter **`glm::vec3 Pos`** .
   - *Looking direction* is defined in **`float Yaw`** .
   - *Looking elevation* is passed in **`float Pitch`** .
   - *Camera roll* is contained in **`float Roll`** .

```
glm::mat4 MakeViewProjectionLookAt(glm::vec3 Pos, glm::vec3 Target, glm::vec3 Up, float Roll float Ar, float nearPlane, float farPlane)
```

Crates *a View-Projection matrix*, where the view matrix uses the *look-at* model, and the projection matrix supports perspective. Parameters are the following:

1. Perspective matrix:
   - The vertical field of view *Fov-y* is defined in formal parameter **`float FOVy`** .
   - The *aspect ratio* is contained in **`float Ar`** .
   - *Near Plane* distance defined in **`float nearPlane`** .
   - *Far Plane* distance is passed into **`float farPlane`** .
2. View matrix (defined with the look-in-direction model):
   - The *position of the camera* is defined in formal parameter **`glm::vec3 Pos`** .
   - The *target of the camera direction* is defined in **`glm::vec3 Target`** .
   - The *up vector* is passed in **`glm::vec3 Up`** .
   - *Camera roll* is contained in **`float Roll`** .

`glm::mat4 MakeWorld(glm::vec3 Pos, float Yaw, float Pitch, float Roll)`

Crates a World matrix, where rotation is specified using Euler's angles, with the zxy convention (z for roll, x for pitch and y for yaw). Scaling is considered always equal to 1, and thus it is not passed to the procedure. Parameters are the following:

- The *position of the object* in world space is defined in formal parameter **`glm::vec3 Pos`** .
- *Euler's angles yaw direction* is defined in **`float Yaw`** .
- *Euler's angles pitch direction* is passed in **`float Pitch`** .
- *Euler's angles roll angle* is contained in **`float Roll`** .

If the matrices are correct, the user should be able to move the truck with the controls shown below. Pressing SPACE allows to change between third- and first-person views. After the two views have been shown once, the third press of the SPACE key will save the screenshots of your results in files **`A04_1.png`** to **`A04_8.png`**. Please check that their content matches your window, as such files will be an important part of the final delivery of this assignment.

In third person view, the controls are the following:

In first person view, the controls are instead: