# Homework 3: C++ Classes

Tiziano Guadagnino, Saurabh Gupta, E-Mail tiziano.guadagnino@igg-uni-bonn.de

Handout : 15.05.2023
Handin: 21.05.2023 at 23:59:59 (CET)

In this homework you will write your first C++ class. In particular you will create the class `MyPointCloud` that represents a 3D Point Cloud and gives the possibility to perform a variety of operation on points. You will also learn how to use the C++ library `Eigen3` for linear algebra, using it to handle the set of 3D points. We provide you with:

- The folder structure of the project: a folder `my_pointcloud` in which you will write your custom class and a folder `apps` in which you will find the `main.cpp` (hint: use it to create the executable).
- Two utility functions defined in the files `my_pointcloud/Utilities.cpp-hpp`, to load and print your point cloud (hint: these functions need the library `Open3D` to work, refer to homework 1 to check how to link it to your library).
- A skeleton for your executable in the file `apps/main.cpp` to read a filename from the command line and visualize the corresponding point cloud (hint: use it to test your class' methods).

General rules:

1. You need to provide the build system for this homework. This means, you need to provide as many **CMakeLists.txt** files as you think is needed.
2. Follow the `header-source-separation` principle, i.e. declare the methods in the provided header (**MyPointCloud.hpp**) file and the corresponding definition in the provided source file (**MyPointCloud.cpp**).
3. Use the type `Eigen::Vector3d` defined in the library `Eigen3` to define a single 3D point; on Ubuntu, you can install it with:

```
$ sudo apt install libeigen3-dev
```

4. Make good use of the `const` qualifier for the class' methods, to let the compiler to force you to not modify the class' attributes when it is not needed.

# A Classes

## A.1 MyPointCloud Attributes

The class will have only an attribute `points` that will represent the set of the points in your cloud (hint: use a combination of `std::vector` and `Eigen3::Vector3d` to represent it).

## A.2 MyPointCloud Costructors

Create two constructors:

1. One that allows to create an enmpty point cloud.
2. One that allows to create a point cloud from a vector of 3D points.

Furthermore, ensure that your class will never be copied. For this you need to take care of the **copy-constructor** and **move-constructor**. Take care also of the **copy-assignment operator** and the **move-assignment operator**.

## A.3 MyPointCloud Operators

Overload the member access operator `[ ]` that allows to get a single point in the cloud, given the index (hint: do some check on the dimension of the cloud).

## A.4 MyPointCloud Methods

Implement the following methods for the class `MyPointCloud`:

- `Clear` → remove all the points from the cloud
- `Size` → return the number of points in the cloud
- `IsEmpty` → check if the cloud is empty
- `GetPoints` → return the points of the cloud
- `Transform` → given a transformation matrix, rotate and translate each point in the cloud according to it
- `GetMinBound` → obtain the point representing the lower bound in all axes
- `GetMaxBound` → obtain the point representing the higher bound in all axes
- `VoxelDownSample` → voxelize the point cloud given a **`voxel size`** and how many points you want to keep for each voxel (hint: use the **`Voxel`** struct the we give you and use it in a **`hash map`** exploiting the **`hash function`** that we give you for the type **`Voxel`**); do not use the center of the voxel as representative for such voxel, but actual points in the cloud
- `RemoveDuplicatePoints` → remove duplicate points (hint: you can exploit the `VoxelDownSample` method)
- `Threshold` → remove all points over a given **`threshold`** for a given **`axis`**
- `ComputeMeanAndCovariance` → compute the point representing the **`mean`** of the cloud and the **`covariance matrix`** of the cloud
- `ComputeGroundNormal` → compute the directional vector representing the normal of the plane of the cloud (hint: you can exploit the method `ComputeMeanAndCovariance`)