Homework. 2: C++ Basic Syntax

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

Handout : 30.04.2024
Handin: 07.05.2024 at 23:59:59 (CET)

In this homework you will write some functions to create your own fantastic library that performs operations over a 3D vector `type`. We provide you with a custom type implemented using a C++ `struct` that you have to use as the base type called `Vector3d`.

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 functions in the provided header (**.hpp**) file and the corresponding definition in the provided source file (**.cpp**)

A Variables, Functions and Control Structures

A.1 Desired Library API

Implement the following functions in the given (**.hpp**) and (**.cpp**) files:

- `sum` → returns a vector that is the element-wise sum of the two input vectors
- `scale` → takes an input vector and a scale value and returns a scaled vector
- `multiply` → returns a vector that is the element-wise product of the two input vectors
- `norm` → returns the norm of an input vector
- `normalize` → returns a normalized copy of the input vector
- `setConstant` → assigns a constant value to all elements of an input vector
- `dotProduct` → returns the dot product of two input vectors
- `minmax` → returns a `tuple` containing the minimum and maximum element of an input vector
- `isZeros` → returns a `bool` value indicating if the input vector has all elements equal to zero

A.2 Ray Sampling with the Custom Vector Library

Once you have your beautiful library implemented, you perform a simple task of 3D ray sampling in the (**main.cpp**) file:

- Write a code snippet that computes and prints to the terminal 10 equally spaced vectors between the provided vectors start_vec and end_vec
- Use as many functions as you defined in the my_vector library
- Feel free to add more functions to the library if necessary to accomplish this task

B Tips

- Take help from `homework_1` to write the **CMake** build system generator for this project
- We provide an example function `printVector` using the `Vector3d` type to illustrate how to use the custom type
- Make sure you use the concept of `const-correctness` and `pass-by-reference` appropriately
- The `main.cpp` file can also be used to test the correctness of your function implementation on simple test cases