INTRODUCTION

MODELING AND ANIMATION, LAB 0

Mark Eric Dieckmann Erik Junholm Emma Broman

Tuesday 19th March, 2024

Abstract

This lab is intended to present the basic guidelines and familiarize you with the code structure used for the labs in the course TNM079 - Modeling and Animation (MoA). We will look at how to fetch, compile and run the application. We will also look at how to present your findings in a report.

1 Introduction

The code structure is designed to be as simple as possible and still support your work well during the labs. It offers a GUI where you can do simple transformations and manipulate the visualization of your models. Plenty of code is already written so you can focus on implementing the core ideas and theory behind the methods discussed during the course.

2 General Guidelines

These are general guidelines and rules for the organization of the labs.

2.1 Labs and lectures

The lectures focus on topics that are directly relevant for the labs. You should understand the lectures and the theory part of the lab script before you come to the labs - the scheduled lab sessions will be enough only if you come *prepared*! We have scheduled 6 lab sessions lasting 4 hours each to implement labs 1-6 and one extra lab to complete the labs that you did not finish during the official lab sessions. You work in groups of 2 during the labs. Under exceptional circumstances you can also work by yourself. But no more than two people are allowed in each group.

2.2 Presenting Results and the Report

The lab assistant will help you during the lab hours with your questions related to the implementation of the code. The lab assistant will also mark completed labs.

All 6 reports are *personal* and should be written *individually* as they will be the base for an individual grade. The reports have to be written in LaTeX and shall be submitted though Lisam. A LaTeX template, which can be used as a blueprint for your reports, will be provided through Lisam as well. You can study how to use LaTeX by looking at the source file. Alternatively, Overleaf can be used as an easy to understand browser based latex editor.

The report should be self-contained, which means that it should not be necessary to read other sources such as the papers that you list in your reference list. You can assume that the reader is familiar with the lab script and the lecture notes and explain the algorithms covered by these without going too much into detail. The reports should not contain pseudo-code! If you use information from text sources other than the lab script or the lecture notes then you have to refer to these sources. Do not copy and paste text and equations directly from the lab exercise sheets into your report. Rather than replicating the description of that theory in your report, it might be more valuable to replicate only the core of the theory and then describe what the theory means for your situation and/or problem.

Subdivide your report into an *Abstract*, a *Background* section and a *Results* section.

Abstract: Summarize in about 100 words the report's content. Try write one or two sentences about *what*, *how* and *why* additionally you should give first results. The abstract should not exceed 200 words in total.

The section *Background* should provide a description of every task you implemented. It should take up about 1 A4 page if you aim at a grade 3, 1.5 A4 pages for a 4 and about 2 A4 pages if you aim at a 5 (excluding Figures). You give in this section a more detailed description of what you have implemented in this lab. You can use 2-3 figures to support your description.

The section *Results* should describe your results. You should show about 2-4 figures (2 for grade 3, 3 for grade 4 and 4 for grade 5), which you made with your program, and describe what you see and how you achieved that. More figures do not lead automatically to a better grade. Each figure needs its purpose! This section should cover 1 A4 page if you aim for a 3, 1.5 pages if you aim for a 4 and 2 A4 pages for a 5 (including Figures). It is a good idea to split the Results section up into *Results* and *Conclusion*. In the latter the results are interpreted and evaluated. You should reason about them. The last section is about your lab partner (purely informal, since the report itself is individual work), the grade you are aiming for and the list of references if you have any.

The total length of your report should be about 3 pages for a 3 and 5 pages for a 5 (including all figures). You can use subsections in the section *Results* to distinguish the individual tasks you implemented. Do not use subsections in the *Background*. Structure the text by using paragraphs.

2.3 Writing Reports with LaTeX

LaTeX (Latex) is a document preparation system that offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing. For example, numbering and cross-referencing, tables and figures, page layout, bibliographies, math formulas and much more. For composing your report using Latex you have three options:

OVERLEAF – The first and **recommended** option is to use one of the online services (Overleaf) designed to make composing documents in Latex easy. All you have to do is 1) create an account and 2) click here and select "open as template" (you will need to sign in). On the next page you will be presented with a website showing a text editor for writing Latex code on the left side and showing the resulting PDF on the right. You can upload images and other material and/or create files and folders by accessing the "Project" menu. Click on "PDF" to download the document.

LOCAL LATEX INSTALLATION – The second and more cumbersome option is to install special software. This software is readily available in the lab computers, if that is where you wish to write your reports. If you want to install it at home, Latex is available for most platforms, see Latex Project. The necessary Latex files are contained in the repository/zip archive (see Sec. 3).

VISUAL STUDIO CODE – The third option is using Visual Studio Code, which is a modern text editor which also supports extensions. Note that for this option you would also need a local Latex installation, since Visual Studio Code does not come with a Latex compiler. Install and open VS Code, Navigate to "Extensions" on the left side and search for "LaTeX Workshop" After the extension is installed commands as "Build with Recipe" and "View LaTeX pdf file" are available. The necessary Latex files are contained in the repository/zip archive (see Sec. 3).

3 Getting the Code

The source code for the application can be forked or cloned at GitLab. If Git is new to you, familiarize yourself with it! You should understand the concept of forks, branches, commits, staging, pushing and pulling! The software has been tested on 64bit Windows with Visual Studio 2019 and 2022. A couple of years back it was also working on Mac and Linux (Ubuntu) with CLion, both 64bit. However, we cannot promise that this is still the case. For those who wish to work on their personal computers, detailed instructions on how to set up the development environment are available in the Readme file.

When working on the lab computers, you will have to store the code either in your User folder on C:, for example: 'C:/Users/liuid123', or on the network drive called something with 'fillager.liu.se'. If you store it anywhere else, you will not be able to run the executable file due to permission restrictions. We recommend storing it on the local hard drive (C:), since this speeds up compilation times. But remember to commit your changes or move your files to another storage (e.g. a USB stick) before leaving the lab! The users folder on the local drive should be considered temporary storage and might be wiped between lab sessions.

Note that if you want to fork/clone the repo, you will have to create an account on ITNs GitLab server using your student credentials.

4 Compiling

In the following, we will shortly describe how to compile the code on different platforms.

WINDOWS – We will use CMake to create the project files for Visual Studio. Set the source code directory (first field on the top) to where you have unzipped the source code or cloned the repo, respectively, e.g. `I:/tnm079/TNM079-code`. Subsequently, set the build directory (second field on the top) to something like `I:/tnm079/build/`. Hit `Configure`. A window will pop up, asking you what kind of project file you wish to generate. Please check, which version of Visual Studio is installed. At the time of writing this document ***Visual Studio 2022*** is available on the lab computers. We also recommend setting the optional Windows 64 bit flag for the platform.

Press `Finish`. Hit Configure again, to update the variables marked in red in the list. If an error persists and you do not know how to fix it feel free to ask the lab assistant. After the configuration step terminated successfully you can hit `Generate` and open the generated solution. You can compile the code in Debug or Release mode. While you are working on the solution it might be advisable to use Debug so you can (you guessed it) debug the code, use breakpoints, etc. When you are done, you can switch to Release. This has the effect that the application will run significantly faster.

MAC AND LINUX – If you have completed the setup and have CLion installed you can simply click on "open project" on startup.

4.1 Notes

Note that in the CMake options you can set which lab parts you want to compile. So, if you are working on lab 2 you do not need to compile parts 3-6. When you move from one lab to the other later on, make sure you enable the next lab. Generally, you might as well enable all the parts right away, the code will just need a little longer to compile.

5 Sample Data

Data sets are available in a separate repository.

6 Code Structure

After successfully compiling and running the application we can have closer look at the code structure. The `GUI` is implemented in `GUI/FrameMain` so you can start by looking through this class (we have also supplied the file `GUI.fbp` which you can use to modify the GUI in wxFormBuilder). An important class is also `GUI/GLObject` which all objects added to the scene inherit from (such as meshes, implicits, cut planes etc.). A base class for geometrical objects is `Geometry/Geometry` which adds transformations and the vital `Update()` and `Initialize()` functions.

Feel free to browse around and take your time to familiarize yourself with the code.

7 Known issues

As of now, there are no known *major* issues within our testing environment. If you run into a persistent problem, please report it to the lab assistant or create an issue on GitLab.

Unfortunately, we do not have the resources to continuously support Mac and Linux computers as well as Windows. In previous years we have observed some problems with the UI on Mac computers, as well as some problems with compiling due to outdated dependencies. We, therefore, do not recommend using Mac computers for the labs. If you decide to do so and overcome the issues, please let us know so that we can add the updates to the instructions or code base. Any contribution is greatly appreciated.

8 Acknowledgements

Originally developed by Gunnar Läthén, Ola Nilsson, Andreas Söderström and Stefan Lindholm.