Project Proposal

Carnegie Mellon University

15-112: Fundamentals of programming and Computer Science

Diram Tabaa

Project Name

Rendorium, a 3D rendering and editing program based on Blender

Brief Description

Rendorium alpha is a 3D rendering and editing software that is based on Blender. Despite being implemented on a pre-existing 3D engine, Rendorium aims to provide its users with a basic set of implements that can be relevant to people outside the 3D design circle. Those tools range from an intuitive GUI that can handle 3D navigation of final-render quality scenes, to a variety of options through which a user can load templates, modify objects or scenes in such templates or use their own custom templates in the form of .blend files. The program also serves the users by providing intuitive export options for still scenes as well as animations. Those export options include but are not limited to: .png, .jpg, .mkv, .avi, etc. Finally, the user will have the ability to save the changes he made on his local device, such that when the user launches the program again and loads his saves, the GUI will be set to the state the user left it when it was saved. With all such features, users with limited exposure to complex 3D software packages can gain an insight into the world of 3D design, as well as produce viable designs themselves.

Competitive analysis

While Rendorium is indeed a slower 3D rendering program than most of the ones available on the market. Nevertheless, there are certain features available in Rendorium that gives it an edge over its competitors. Firstly, Rendorium is more user-friendly than most 3D editing or rendering software available on the market. The user does not need to learn obscure commands or learn the different ways a viewport can be navigated to use Rendorium. This is especially true for users with limited expertise in using computers in general, such as elementary school children. Rendorium compared to other 3D rendering programs (even Blender itself) on the market can be thought of as akin to what MS paint is to Adobe PhotoShop. While Paint features are more limited, it is more intuitive and easier to use and learn than Adobe Photoshop. It is the case for Rendorium since it is not aimed towards people who are professional in 3D design. Secondly, Rendorium offers pre-built templates that are unavailable in most 3D rendering applications like Autodesk Maya, those templates range from realistic indoor scenes to vibrant outdoor scenes. While some 3D rendering applications like InteriCAD do offer such templates, many of them charge a hefty price for a single template alone. Rendorium offers templates free of charge that are copyright free as well.

Structural Plan

Rendorium is based on two key components: the front-end GUI and the back-end Blender Process. The front-end GUI has been developed in PyQt5 and is being carefully modified to match with modern GUI in terms of usability and aesthetics. The Back-end process, which uses data generated and collected from the user interaction with the GUI, runs as an application subprocess on the local system. Blender has been chosen as the application on which the back-end code runs for multiple reasons: the availability of an API through which 3D geometry can be handled, the ability to run Blender in command line mode, and finally the intensive documentation available for the python API. When the back end is run, the data collected from the user's interaction with the GUI is processed to translate the data into a series of commands and/or functions. Arguments can also be passed down from the front-end to the back end. This means of communication is facilitated by .data files stored in the local directory. The backend script will be a single class through which different functions will be implemented, the execution of such functions will depend on the commands received from the main GUI script. The GUI script is divided into 3 different classes, one class runs on a different thread and handles the subprocess, another class will handle the interaction between the loading screen and the main screen, and the last and most important class will handle all functions related to the GUI elements and their interactions with the user. Scenes rendered will be stored as temporary image files in the local directory. A compiled version of blender is also available within the files of Rendorium. All the graphics are handled by images produced by me personally (which include non-copyrighted material)

Link for car model: https://free3d.com/3d-model/bugatti-chiron-2017-model-31847.html

Algorithmic Plan

The biggest challenge presented by this project has been the implementation of an interactive GUI that allows for the implementation of long-running tasks on the backend. Simply executing the long-running task through a command from the GUI makes the GUI unresponsive till the task is completely executed. This presented a problem since most of the tasks running on the backend were rendering tasks that understandably take a very long time to implement. The solution came in part through inspiration from a StackOverflow answer, which highlighted a class present in PyQt5 through which threading can be handled. This class, called "QThread", allows the creation of threads that are linked to the GUI framework. Once a thread is created, a process or a function can be moved into that thread through an attribute of QWidget. Subsequently, the initialization of a thread can be connected to the task that has long execution times. For this project, those features were utilized by creating a thread through QThread, move the function which initializes the subprocess (long-running since it ends after all back-end commands have been executed) to the thread created, and then connecting the initialization of the thread to the execution of that function (which is present in a different class). Whenever the subprocess was needed, the initialization of the thread is called, subsequently enabling the execution of the subprocess on that thread. Once the function has been executed completely, a PyQtsignal is emitted, which when received by the main class, signals the termination of the thread. Such termination allows the user to re-run the subprocess as many times as needed, and it also makes sure that the thread doesn't stay even when the main window has been destroyed.

Link to thread: https://stackoverflow.com/a/33453124

Timeline

17th to 19th Nov: Completion of all save/import/export features, fixing minor bugs

20th to 21st Nov: Completion of pre-made template features, implementation of basic editing tools

22nd to 24th Nov: Completion of editing tools and features (adding or deleting meshes, changing material, etc.)

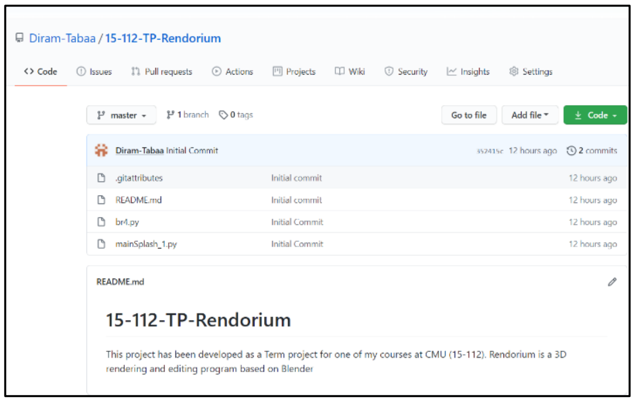
24th to 26th Nov: Implementation of animation-related features (GIF preview, switching between still preview and GIF preview in the GUI), fixing bugs

26th to 28th Nov: Completion of animation-related features. Implementation of customizability features (dark/light theme, docking widgets).

28th Nov to 1st Dec: Debugging. A short user-guide is written. Project version is updated to alpha 0.2

Version Control Plan

All python code that is written for this project is uploaded to GitHub on daily basis. In case of any storage failure or loss of data. The public repository will always be present to get the latest commit back. In the case of loss of functionality due to errors caused by a newly implemented feature, the commits can be reverted to their previous state before that feature has been implemented.



Diram-Tabaa Initial Commit 352415c 12 hours ago 2 commits

.gitattributes	Initial commit	12 hours ago
README.md	Initial commit	12 hours ago
br4.py	Initial Commit	12 hours ago
mainSplash_1.py	Initial Commit	12 hours ago

README.md

15-112-TP-Rendorium

This project has been developed as a Term project for one of my courses at CMU (15-112). Rendorium is a 3D rendering and editing program based on Blender

Link: https://github.com/Diram-Tabaa/15-112-TP-Rendorium

Module List

- PyQt5
- subprocess
- blender modules (bpy & mathutils)
- os
- sys
- pathlib

TP2 Update

Pre-made template idea has been delayed till major core features are implemented. The range of functionality in terms of mesh manipulation or physics simulations is to be limited. GIF based animations are still in the plan. Export feature should be added soon. Some features, like light/dark theme, have been implemented way ahead of time.