3. User manual

The program is started by launching “**Front.scala**” as a Scala Application. This opens up the starting menu as a window, in which you can operate by using the mouse or keyboard. Arrow keys allow movement between buttons, same as tab, and with space you can press any button. Remember that if you are using arrow keys and enter a slider, it is not possible to get out of it by left and right arrow keys, so use instead tab/up-down arrow keys. To enter fullscreen mode you can press “F” any time, same to exit it. This is especially preferable on the Keyboard shortcuts scene, to get a good view of the whole picture.

When you are done tinkering with settings, get back to the first scene, from which you can choose “Start” to enter the main program. One final note about settings is that the intersection math is rather heavy so if your hardware is having trouble, you can turn of clipping under “Show more” in settings, but then you have to remember to always have the objects in the world in front of you, as the clipping is integral to filtering out objects on the backside of camera.

The controls are as follows. WASD to move around, mouse to turn around. Shift to run, control to crouch. While crouched, Q and E to lean to the sides. Plus to zoom in, minus to zoom out. Z to reset zoom level to the original. To exit/enter full-screen, press F, and to exit the program Esc. Space to jump around/ fly.

The program is best used as a visualization tool, so use of CSVWriter to make on shapes is encouraged.

4. Program Structure

The main window is Front.scala which handles everything related to ScalaFx excluding styling, which resides inside src/stylesheets. Front is resposible for the whole visualization of the program, including both the starting menus and the main program. The substructure consists of scenes, each handling one view of the program, and is as follows:

* bootstrap: The opening screen with navigation to the main screen, options, and quit. Quitting is always possible with Esc, but here is a fall-back incase that fails.
* options: All the options handling “game mechanics”, which includes everything that affects the player directly. This includes things such as player speed, as well as all the “cheats”. From here it is possible to navigate “Back” -> back to the opening screen, “Controls” -> to view the keyboard navigation tips, and “Show more” -> to show the rest of the options.
* more:
  + This includes setting the clipping of the main screen. This clips all the to-be-drawn triangles against the plane to the sides of the player, with normal vector to the “viewing direction”, in other words where the camera is pointing. Disabling it increases performance greatly but then it is required that all the objects reside in front of player all times.
  + Here is also settings for adjusting screen size. A good thing to note is that the window is always at max the width or height of the screen, so if the screen is already full with 16:9 “small”, the result will be exactly the same with 16:9 “medium”. The aspect ratio affects mostly the windowed mode, as fullscreen always fills the entire screen, but if the image remains distorted, changing the aspect ratio to the right one might help. Also pressing “Shift+F” when exiting fullscreen changes it to full-screen with the screen only filling the size a corresponding windowed mode would fill.
  + Last in the scene is file loading. Only csv works. Obj-files load as well but are corrupted, if the ordering of the points in the triangles are not the exact same as the one used in the program. This is mainly manifested in the clock-wise order of the points of the triangle viewed from the side the triangle should be visible from e.g. a wall that is only visible from either inside or outside. So if you want a house to have walls visible from inside and outside, create two tringles facing opposite ways. Press the “Load” to load a preferred file. If the program did not receive the string, it will ask for one, likewise it will inform in the window if the file was found. If the file does not exist it the resources-folder, the program will close. In the console printout on the top row there is the custom exception which will tell the tried and failed path. Usually you have to scroll up in the console to find the exception.
* threeD: As the name implies, this is the core of the product. All the referrals before to the “main screen” mean this one. This is where all the game mechanics are handled. You can see all the flags established at the beginning, as they are prepared when the whole program launches for the first time, so they will not be resetted when the game is on-going. The dynamic part of all the scenes are the event handlers like “onKeyPressed” which can be used to alter the scene content while in the scene. The recurring operation, which happens approxiamately once every 0.01 seconds, is the AnimationTimer, and everything inside it. Even though the program only executes the loop multiple times, all the event listeners of threeD work parallel to it’s running. The exact order of execution is unknown to me, but effectively they work parallel. From inside threeD there is calls to Projector to prepare the data for visualization, and to Camera, to handle the rotational aspects as well as moving around. The call to projector gives first of all the 2D converted shapes, but also the final 3D shapes to allow for finding out the right draw order. The timer also hosts all the listeners to the flags established in the scene/stage, that need dynamic functionality. At the end of the timer, everything is drawn on the screen