Running a javafx or B4J jar file in linux.

You can use a linux shell script (bash) to automate the process of running a java executable jar file.

You can find a good tutorial at:
https://www.freecodecamp.org/news/bash-scripting-tutorial-linux-shell-script-and-command-line-for-beginners/

Let's create a bash script with the name "run_javafx_jar.sh".
Open your favorite linux editor (in this case nano) with the file name:

```
sudo nano run_javafx_jar.sh
```

Type in your sudo password if asked.
The script contains these 2 lines to type:

```
#!/bin/bash
java -Dprism.verbose=true -Dprism.forceGPU=true -jar --module-path $PATH_TO_JAVAFX --add-modules ALL-MODULE-PATH "$@"
```

Make sure there is a space after --module-path.
Press CTRL+O and ENTER to save the changes and press CTRL+X to exit the editor.
To make the script executable you need to give the following command:

```
sudo chmod +x run_javafx_jar.sh
```

The second line needs some more explanation.
The command used to run a jar file is: java
This means you must have a java JDK installed in linux.
You can check this with the command:

```
java -version
```

```
openjdk version "21.0.6" 2025-01-21
OpenJDK Runtime Environment (build 21.0.6+7-Ubuntu-120.04.1)
OpenJDK 64-Bit Server VM (build 21.0.6+7-Ubuntu-120.04.1, mixed mode, sharing)
```

If you need to install the JDK 21 you can use this command:

```
sudo apt install openjdk-21-jdk
```

The java command takes several arguments that are separated by a space character.
Depending on your needs you may have to add some more arguments.

The -Dprism arguments are used when your jar file uses graphics.
The -Dprism.verbose=true argument shows the rendering process in the terminal window.
The -Dprism.forceGPU=true argument instructs the command to use the GPU.

The -jar argument indicates that you are going to run a jar file.

The --module-path argument specifies the path to the javafx modules.
For this you can set up a linux environment variable. Use your editor and change the file "/etc/environment".

```
sudo nano /etc/environment
```

Add a new line:

```
PATH_TO_JAVAFX="javafx-sdk-21.0.6/lib"
```

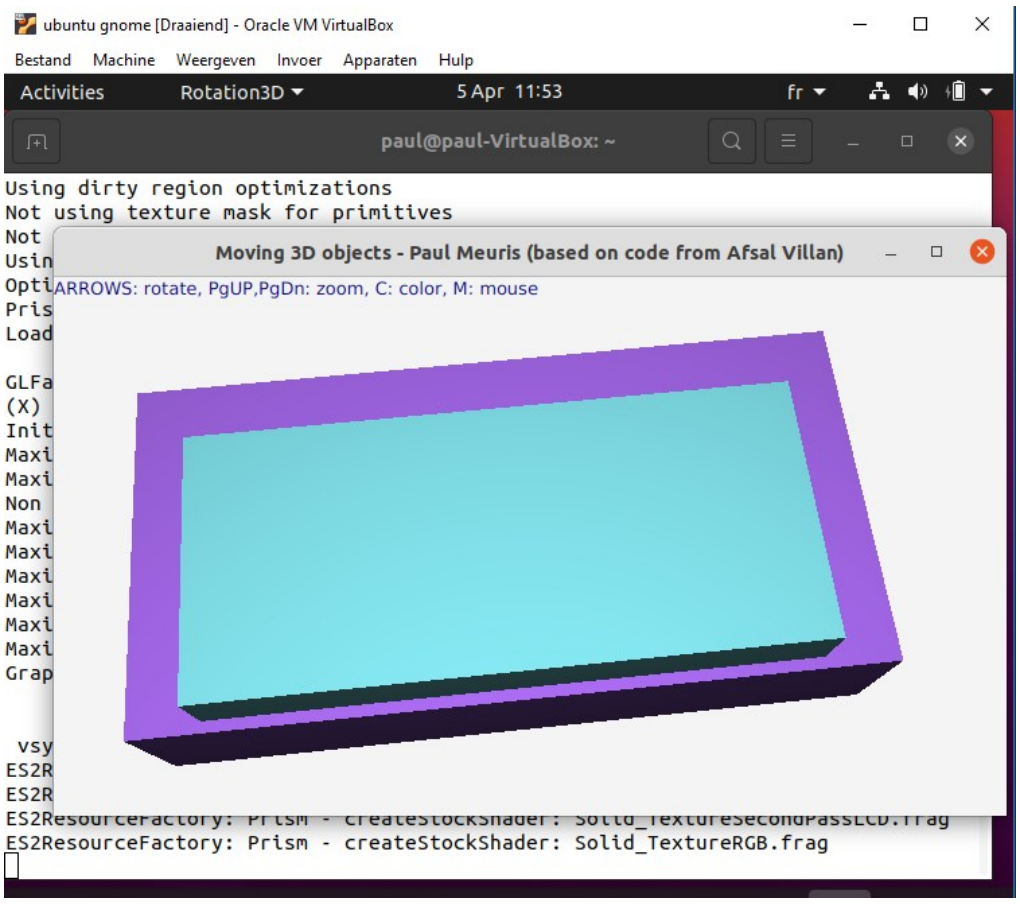
And save the file with CTRL+O, ENTER, CTRL+X
You need to adjust the path string to where you copied the javafx-sdk-21.0.6 folder. In my case this is in the home folder.

The --add-modules argument lists the modules that are needed.
For this you can use the predefined variable ALL-MODULE-PATH or you list specific modules: javafx.controls,javafx.fxml,javafx.graphics
The individual modules are separated with a comma and there is NO space character after the commas because it's considered to be one argument!

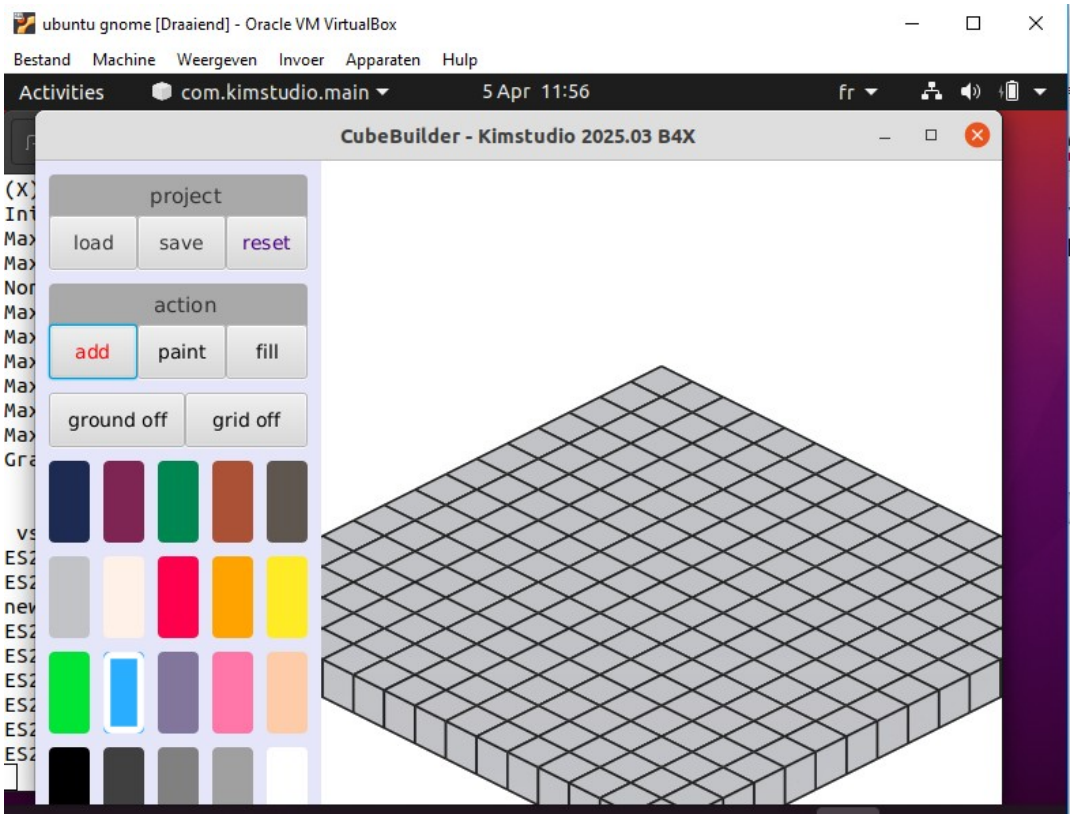
The "$@" argument is a placeholder for the command line argument from the bash script. It is replaced with the path and filename you type in the command line after the name of the script.

Examples:
I have copied some examples in the shared folder in linux that is linked to the shared folder from my Windows host (VirtualBox setting).

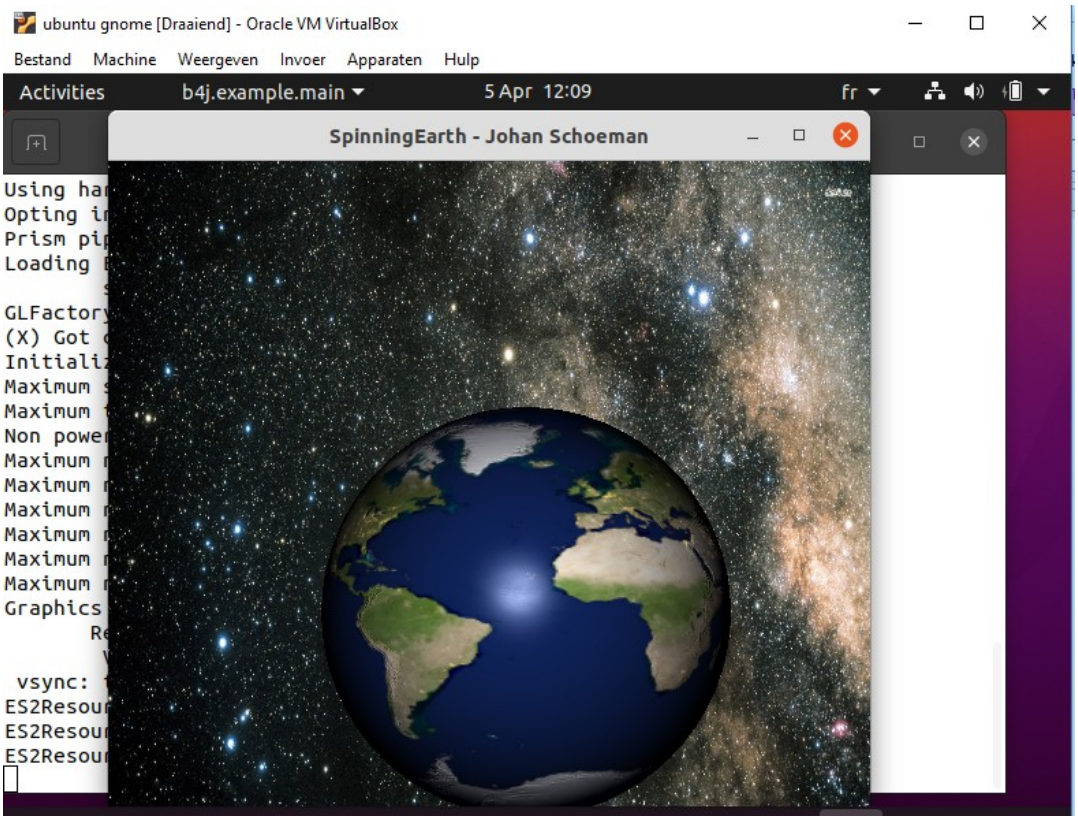
```
./run_javafx_jar.sh windows_share/Rotation3D.jar
```



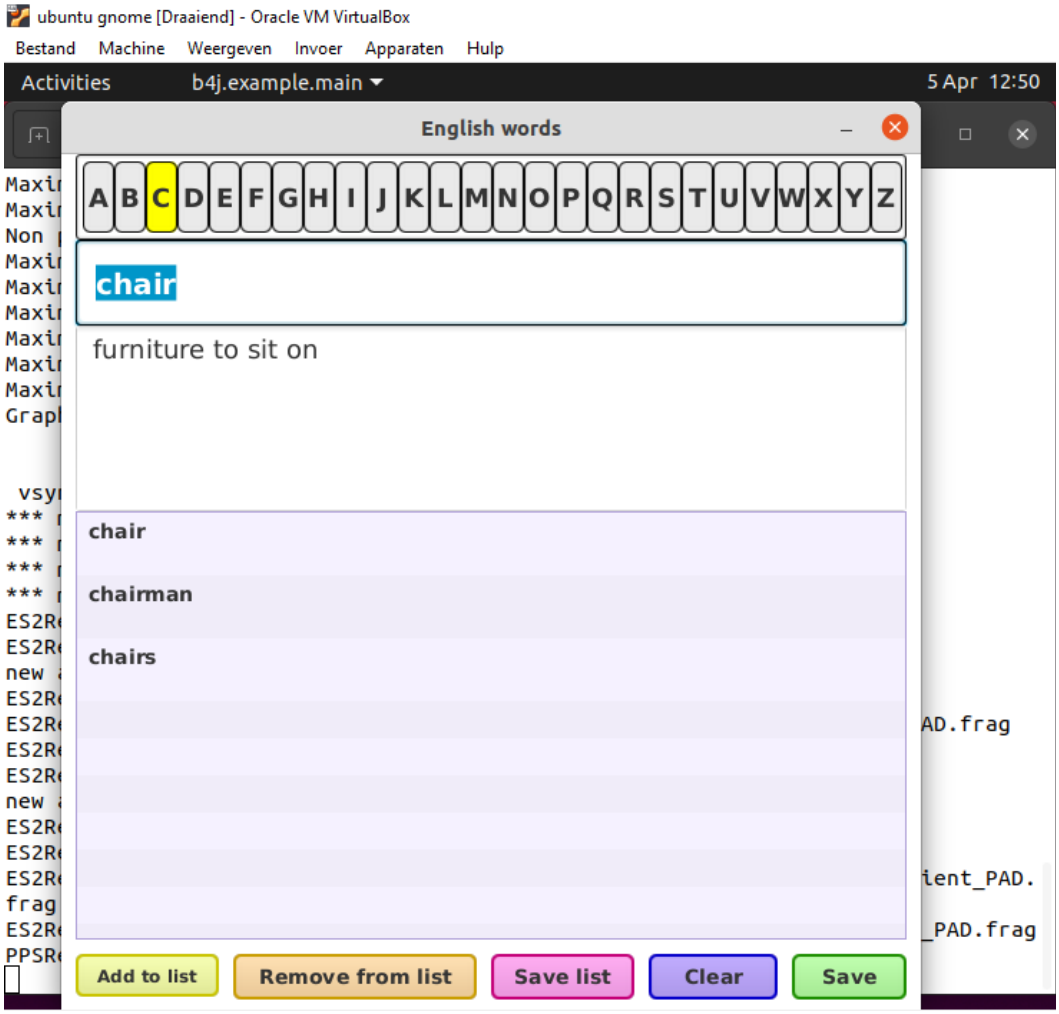
./run_javafx_jar.sh windows_share/Cubebuilder.jar



./run_javafx_jar.sh windows_share/SpinningEarth.jar



./run_javafx_jar.sh windows_share/english_words.jar



You may also need to use some of the following arguments:

--add-exports javafx.graphics/com.sun.javafx.geom=ALL-UNNAMED
--add-opens javafx.graphics/javafx.scene.canvas=ALL-UNNAMED

The .jar files are created using the B4J IDE or the BlueJ IDE.

If you run a B4J application in Release mode then you can find the generated .jar file in the folder B4J\<project name>\B4J\Objects\<project name>.jar

In the BlueJ IDE you can use the menu item Project / Create Jar File… , indicate the main class that needs to be executed and save the file in a folder of your choice.