Set-up and Introduction to 2D and 3D Graphics

Repository: https://github.com/raaavioli/DH2323-Computer-Graphics

I was not able to make SDL work on my hardware, so I used SDL2 together with the following skeleton linked in canvas: https://github.com/lemonad/DH2323-Skeleton

Dependencies

- SDL2
- CMake

Build

- git clone https://github.com/raaavioli/DH2323-Computer-Graphics
- cd DH2323-Computer-graphics-and-interaction/Lab-1-Interpolation
- mkdir build && cd build
- cmake ..
- make

Notice that CMakeLists are configured to compile for arm64 if running Apple (MacOS), since I'm working on Apple Silicone.

So if you're running **MacOS** on an **x86** processor and you're having problems, **comment out** the following lines in CMakeLists:

```
IF(APPLE)
  SET(CMAKE_OSX_ARCHITECTURES "arm64" CACHE STRING "Build architectures for Mac OS X" FORCE)
ENDIF(APPLE)
```

Run

Part 2 on interpolation is compiled into *Colors*

- *./Colors*

Part 3 on starfield is compiled into *StarField*

- *./StarField*

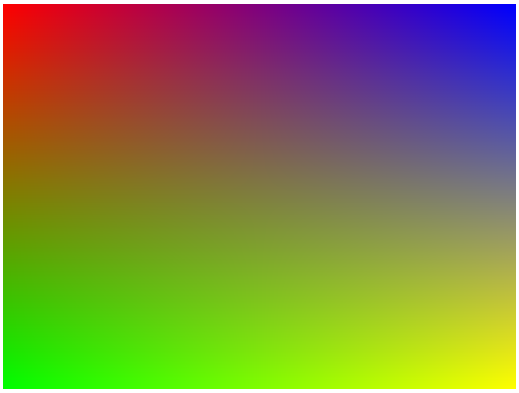
Implementation - Colors

When implementing the interpolation function, I did not really encounter any major problems. Starting with the float-float interpolation, the most tricky thing may have been to get an inclusive range. I did not get any unexpected bugs however, and the trick is to iterate over all values in the vector, and for each value calculate the interpolating factor, which is the index of the element divided by "*size - 1*".

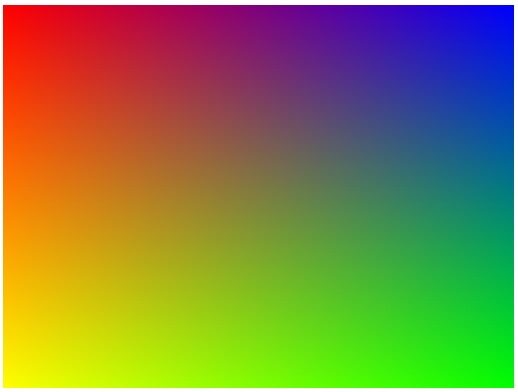
The reason for using "*size - 1*" is simply as the array is 0-indexed, and we want to include fractions evenly distributed between 0 and 1, and since the first index is 0, then i / (*size - 1*) = 0. Since the last index is "*size - 1*", then i / (*size - 1*) = 1, and all indices in between will be linearly interpolated between 0 and 1.

Result

The first result looked like the following due to the skeleton setting up the green color in bottom left and yellow in bottom right.



After changing the order of the yellow and the green, the result looks like the expected result according to the assignment.



Implementation - StarField

When implementing the starfield there were not any real problems that occurred. First I was a little confused regarding the directions of the stars, causing them to move out from the screen, however that was just fixed with a change of sign when incrementing z-position.

Other than that, I quickly realized the rather simplified formula z_i = z_i-1 + v * dt needed a little modification to take into account that the time received from SDL_GetTicks is a discrete value representing the number of milliseconds since SDL-initialization. To fix this, an upper-bound was set of 1 second per frame, and the delta-time “dt” was then clamped between 0 and 1000, and divided by 1000, to create a value between 0 and 1, that could be multiplied with the velocity.

Doing this, the result of the implementation looks like the following, while the small white dots are travelling towards the observer and are reset back to z = 1 once they reach z = 0.