In this exercise we will look at two methods of interpolating colours. When interpolating colours, we must somehow define which colours are in-between, say, what is the colour half-way between red and green?

In computer graphics, colours are represented in a colour space, and the geometry of this space implicitly answers this question. However, there exists a lot of colour spaces, and different colour spaces can yield different results. We will look at two two colour spaces in this exercise, the RGB and HSV colour spaces.

The most commonly encountered colour space in computer graphics is the RGB colour space. Here, we specify colours as a mix of Red, Green, and Blue. This is a natural choice since this is how colours are formed on computer screens. In addition, it is closely related to the physiology of the human eye (which decomposes light into red, green and blue). However, this colour space is not particularly intuitive seen from a psychological point of view; humans do not describe colours in this way.

A more intuitive colour space is the HSV colour space. In this space, we describe the Hue of colour, that is, if the colour is red-ish or green-ish. We also describe the Saturation of the colour, that is, how washed-out the colour is (is it almost grey or pure?). Finally, we describe the Value of the colour, which is simply the intensity of the colour.

To get a feel for these two colour spaces and the relation between them, fire up The Gimp and open up the Colour Selection dialogue, see Figure a.

THE APPLICATION

The application should be given six floating point numbers. These six numbers are the RGB values of two colours, the colour we are interpolating from and the colour we are interpolating to. For example,

```
./oblig1 1.0 1.0 0.0 0.5 0.5 1.0
```

specifies the first colour to be yellow \((1,1,0)\) and the second to be light blue \((\frac{1}{2}, \frac{1}{2}, 1)\).

To visualize the colours, split the screen into two and let the top half show the gradient representing the RGB interpolation and let the bottom half represent the HSV interpolation, see Figure b. Make sure that the colours are smooth.
The file `ColorSpace.hpp` contains two functions to convert colours back and forth between the RGB and HSV colour spaces, which you may use.

Let $c_1$ and $c_2$ be the first and second colour, then linear interpolation is defined as

$$c(t) = (1 - t)c_1 + tc_2, \quad t \in [0, 1]$$

where $c$ is the interpolated colour. You can use this directly to interpolate in the RGB space.

Interpolating in the HSV space is slightly different. In the $S$ and $V$ components, the linear interpolation formula given above works fine, but the hue is an angle, which is periodic. Interpolation may therefore be done clockwise or counterclockwise. Your application must choose the shortest distance along the circle between the start and end hues. For example, if the starting hue is 5, and ending hue is 340, your interpolation should go something like 5 - 0/360 - 355 - 350 - 345 - 340, instead of increasing from 5 to 340. If the angles are 180 degrees apart, both directions are equally good, and your choice may be arbitrary.

**Handling in the exercise**

The assignment is individual and everyone shall create their own program. If you choose to use code or derivations of code that is not your own, its source and author shall be explicitly cited. We will normally accept some degree of unoriginal work, but we may require that parts of the unoriginal code must be rewritten.

We reserve the right to do an oral examination of the student for each submitted exercise. The failing or passing of this examination will decide whether you pass this exercise, and this decision is final.
The program can be either a C or C++ program, and the visualisation shall be done using OpenGL. The choice of windowing toolkit is the student’s. Document your efforts by commenting the code.

All source code as well as a Makefile shall be included. The Makefile should have at least a target to build the program as well as a clean-target that deletes all object-files as well as the executable. Assume that your user name is `foo` (exchange “foo” with your user name). Put all files in a directory called `foo-1`,

```
cd foo-1
make clean
```
to remove compiled files and

```
cd ..
tar -cvfz foo-1.tgz foo-1
```
to create the archive `foo-1.tgz`. Handing in the assignment is performed by sending an email to the tutor of the course with the subject “INFx320 Oblig 1” with an archive attached within the deadline. Make sure that the program compiles and runs cleanly on the computers at IfI.

*Good luck!*