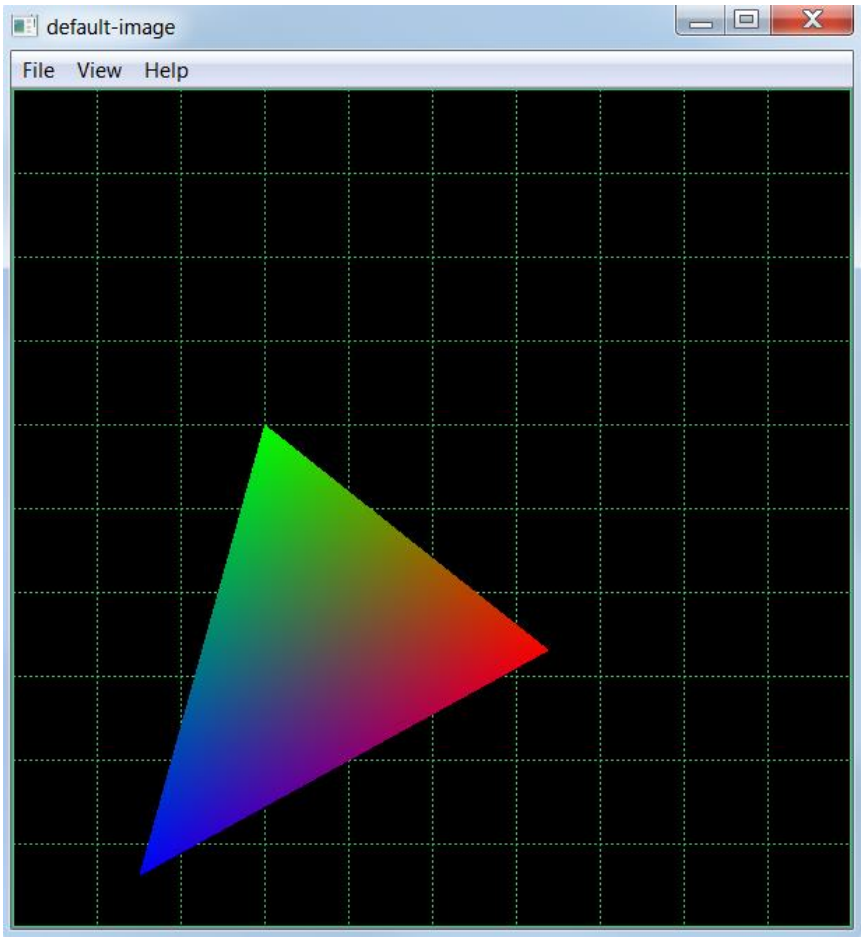
POLYTECH
NICE-SOPHIA

Lab3: Part2, Basic Qt

The objective of Lab3 is to discover an IDE used in the embedded world: Qt (“cute”). Qt is in fact more than a simple IDE but a full framework to develop complete application with graphic interfaces. Many of the specific displays found in the automotive, medical or consumer electronic sectors have been developed using Qt framework.
You will find tons of youtube videos on Qt and dedicated displays, if you are interested you can start with https://www.youtube.com/watch?v=6mcU4exDz4k

1 - Generation of a New Image

Your goal is the update the code you have tested in part1 (`image_base`) to generate a new image of size 600x600 which will look like the following:

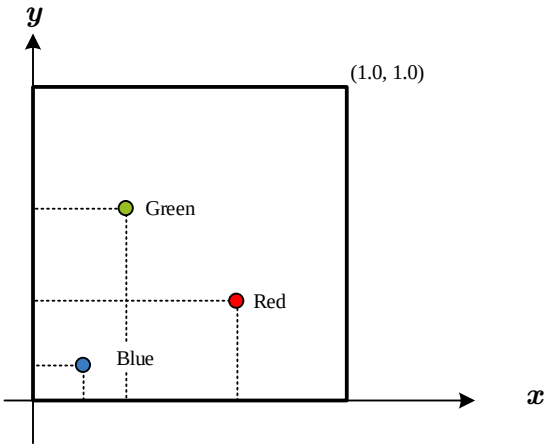


1.1 Real World Coordinate

The vertices of the triangle are given in the "*real world*" coordinate:

	X	Y
Red	0.64	0.33
Green	0.3	0.6
Blue	0.15	0.06

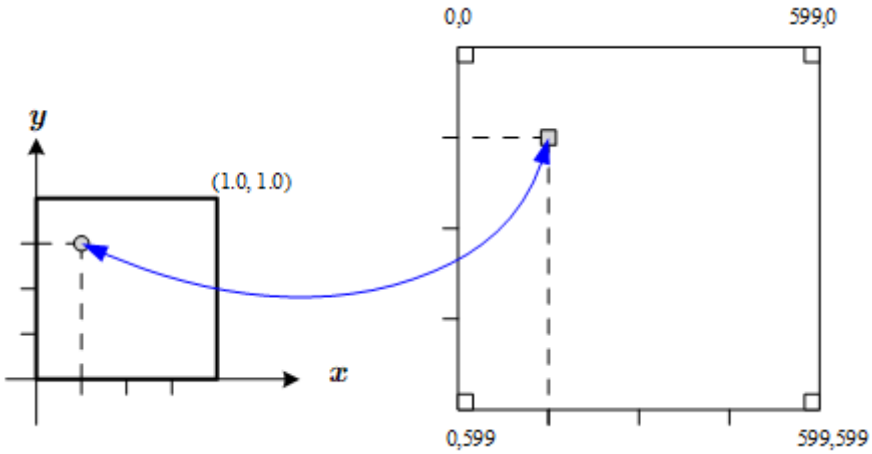
The overall image shall fit in the unit square, i.e. length of each side is 1.



real world

1.2 World / Pixel Conversions

World coordinates must be converted into pixel coordinates.
You can use the following figure to help you perform this conversion: two simple linear interpolations one for each axis.



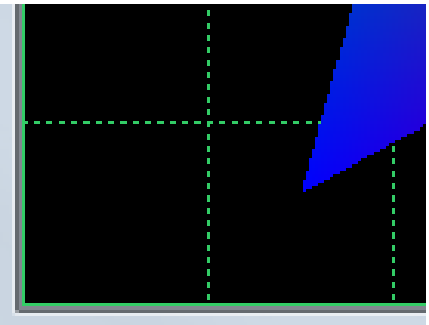
real world *pixel space*

Ultimately, you need to draw pixel, so it is always better to go from *pixel space* to *real world* and for each pixel coordinate, obtain the coordinate in the real world then compute its colors and paint the corresponding pixel with that color.

1.3 Drawing the Grid

Drawing the grid is your first step, ensure that all the pixels of the images are set to black color and draw a grid as shown in the target picture.
The spacing of the grid is 0.1 in the real world.
The RGB color for the grid is (51, 204,102).

You can start drawing you grid with plain lines, then move to dash pattern.



1.4 Drawing the Colored Triangle

Must read: http://en.wikipedia.org/wiki/Barycentric_coordinate_system

If you are not familiar with RGB color model in digital image, please read:
https://web.stanford.edu/class/cs101/image-1-introduction.html"

1.4.1 Monochrome Triangle as a Starter

Start with a simple monochrome triangle (red for example).
For each pixel of the image, you need to convert the pixel coordinates (X, Y) to real world coordinates (x, y).
Then find if the (x, y) point is inside or outside the triangle by computing the lambda values λ_1, λ_2 and λ_3. If inside, paint the corresponding pixel in red.

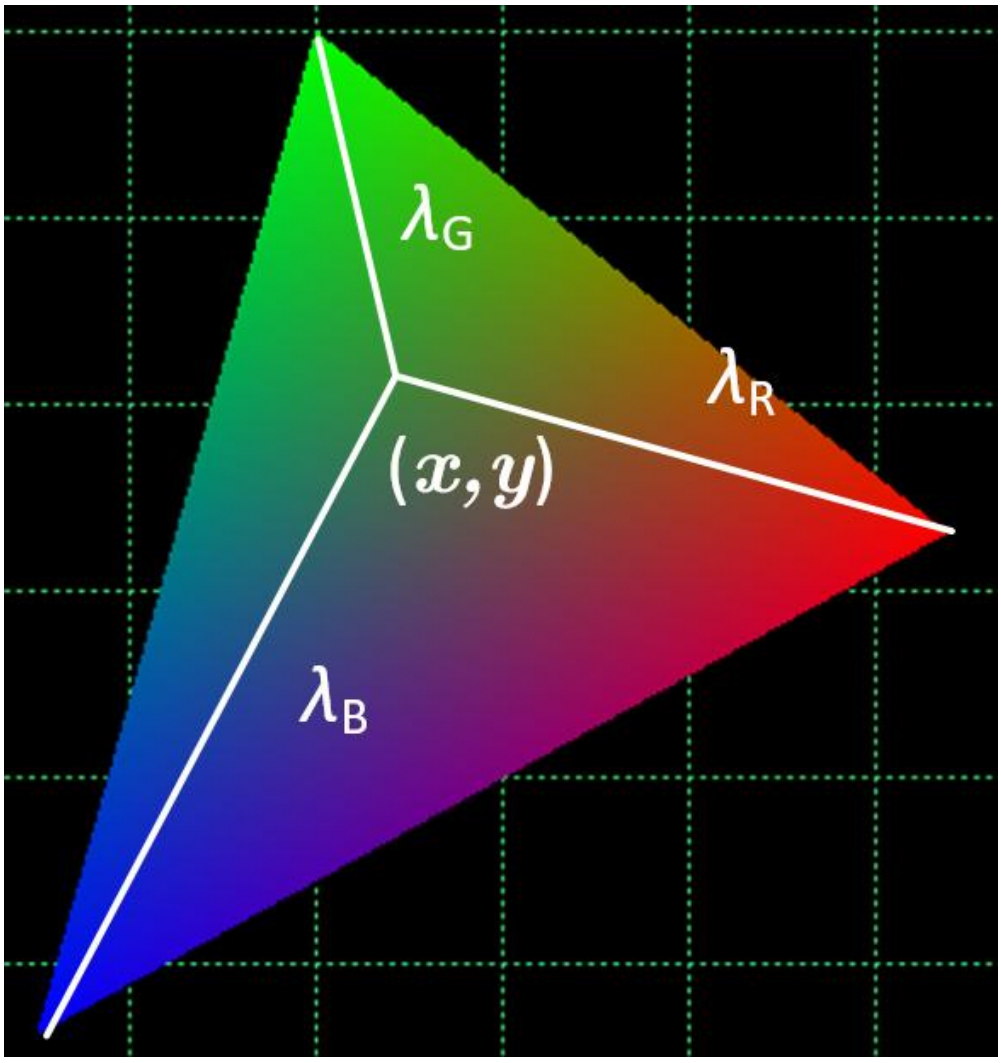
How can you optimize the algorithm to avoid scanning the entire image?

1.4.2 Colored Triangle

For better understanding, we shall use λ_G, λ_R and λ_B, instead of the indices 1, 2 or 3, as shown in the screen capture below.

With barycentric coordinates, the closer the point (x, y) is to a vertex, the closer the matching λ is from 1, note that the sum of the 3 lambdas is always 1 when point is within the triangle. So beware that the length of each white segments in my figure is *not* λ_i.

To know the color of a point, you need to compute the value of each primary color (red, green blue) as function of the lambdas. Remember that the value of each primary color is an unsigned integer from 0 to 255.



1.5 Reference

Interesting QT methods:

```
QRgb qRgb(int r, int g, int b);
void QPainter::drawLine(const QPoint &p1, const QPoint &p2);
void QPainter::setPen(const QPen & pen)
void QPen::setDashPattern(const QVector<qreal> & pattern)
void QImage::setPixel(int x, int y, uint index_or_rgb)
```