# Tutorial

## Introduction

LayerPlus is basically an extension to the CImg library, a C++ template image processing toolkit. In this tutorial, you will see how it can be used to give layer support to an image processing system. This tutorial assumes you have some basic knowledge of CImg, but the specific CImg usage will still be explained.

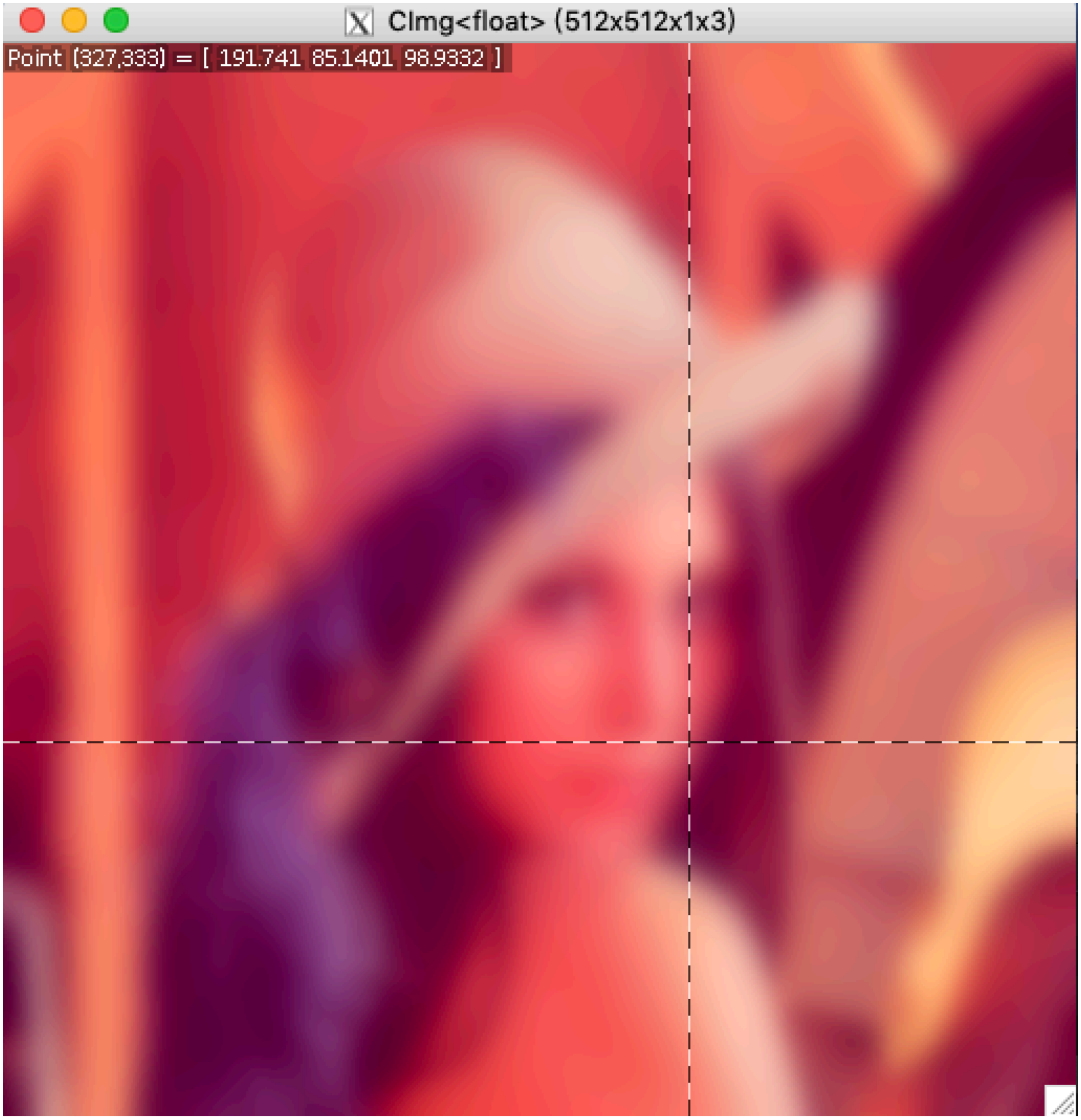
## Getting Started

### Single layer

Now let's get the idea by starting to write our first program. This will demonstrate how to load images and create layers. Assume we want to load a color image lena.jpg into a layer, smooth it, display it in a windows. It can be very simply done by using the LayerPlus library. Just look at the code below:

```
#include "Layer.h"
using namespace cimg_library;

int main() {
  CImg<float> image("Lenna.jpg");
  cimg_extension::Layer<float> layer(image);
    cimg_extension::Layer_System<float,10> sys;

    cimg_extension::Layer<float> layer_first = *(sys.blur_gradient_layer(layer, 5));

    CImg<float> res = layer_first.data();
    res.display();
    return 0;
}
```

Here is a screenshot of the resulting program:


CImg<float> (512x512x1x3)
Point (327,333) = [ 191.741 85.1401 98.9332 ]

And here is the detailled explanation of the source, line by line :

```
#include "Layer.h"
```

Include the main and only header file of the Layer library.

```
using namespace cimg_library;
```

Use the CImg library namespace to ease the declarations afterward.

`CImg<float> image("Lenna.jpg");`
Initialize a CImg instance using CImg library.
`cimg_extension::Layer<float> layer(image);`
Initialize a layer, load a CImg instance into layer.
`cimg_extension::Layer_System<float,10> sys;`
Initialize the layer system.
`cimg_extension::Layer<float> layer_first = *(sys.blur_gradient_layer(layer, 5));`
The layer system blurs the current layer, with a standard variation of 5 and load it into a new layer.

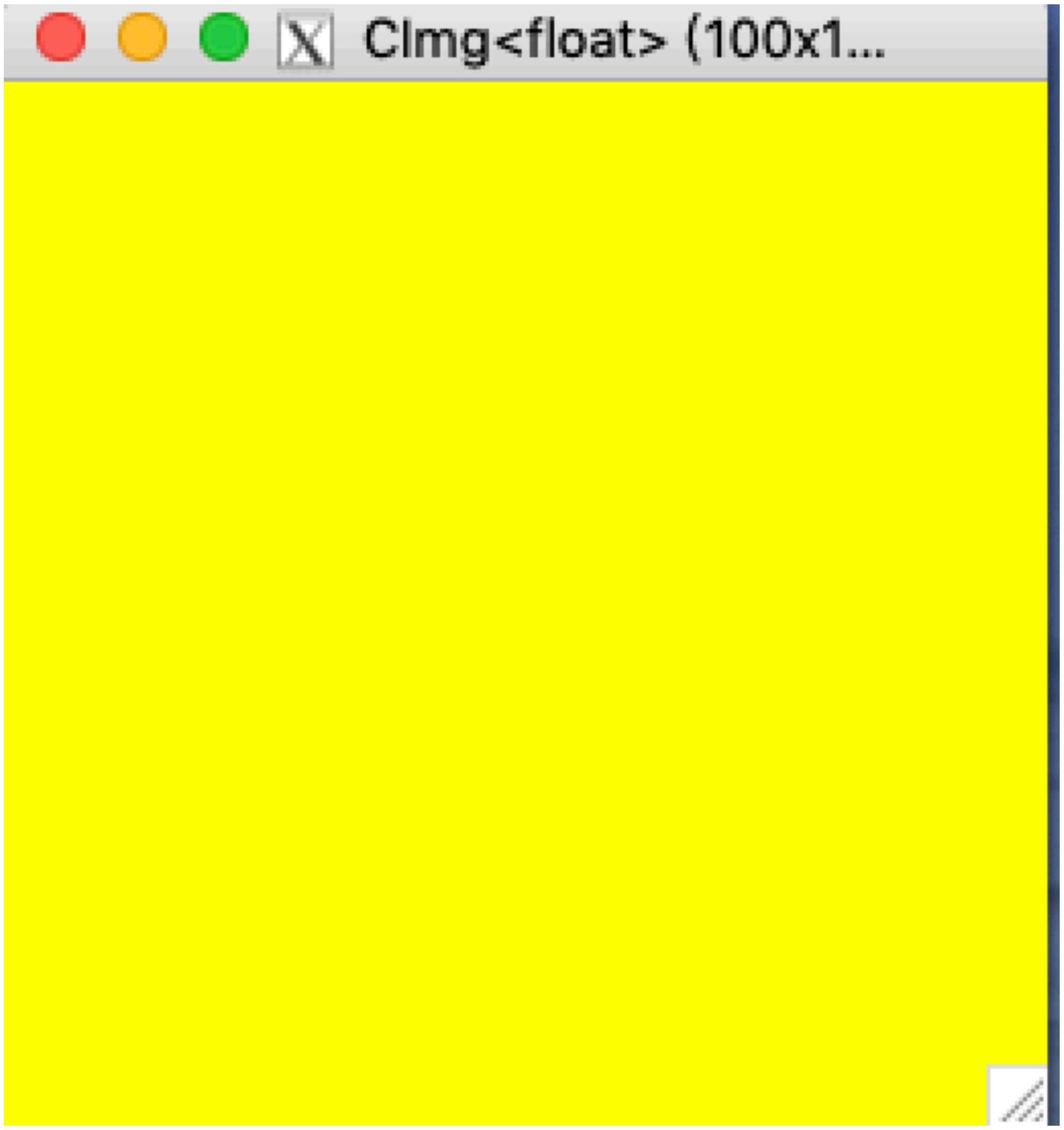
`CImg<float> res = layer_first.data();`
`res.display();`
Extract the CImg instance and display on windows. That's all! The code is very intuitive and small.

## Multiple layers

Now let's try to add another layer, in which the image is set to a low brightness. We just need to modify our former program a little bit.

```
#include "Layer.h"
using namespace cimg_library;

int main() {
    CImg<float> image("Lenna.jpg");
    CImg<float> img("overlay.jpg");

    cimg_extension::Layer<float> layer(image);
    cimg_extension::Layer_System<float,10> sys;

    cimg_extension::Layer<float> layer_first = *(sys.blur_gradient_layer(layer, 5));
    cimg_extension::Layer<float> layer_second(img);

    sys.add_layer(layer_first);
    sys.add_layer(layer_second);

    cimg_extension::Layer<float> top = sys.get_top_layer();
    top.data().display();
    return 0;
}
```

Here is a screenshot of the resulting program:



Here are detailed explanations line by line:

```
cimg_extension::Layer<float> layer_second = *(sys.exposure_layer(layer, 0.5));
```

Generate a new layer. It loads a CImg with a gamma 0.5, which has lower brightness.

```
sys.add_layer(layer_first);
```

```
sys.add_layer(layer_second);
```

Add layers into the layers system. The layer_first located at the bottom while layer_second is on the top.

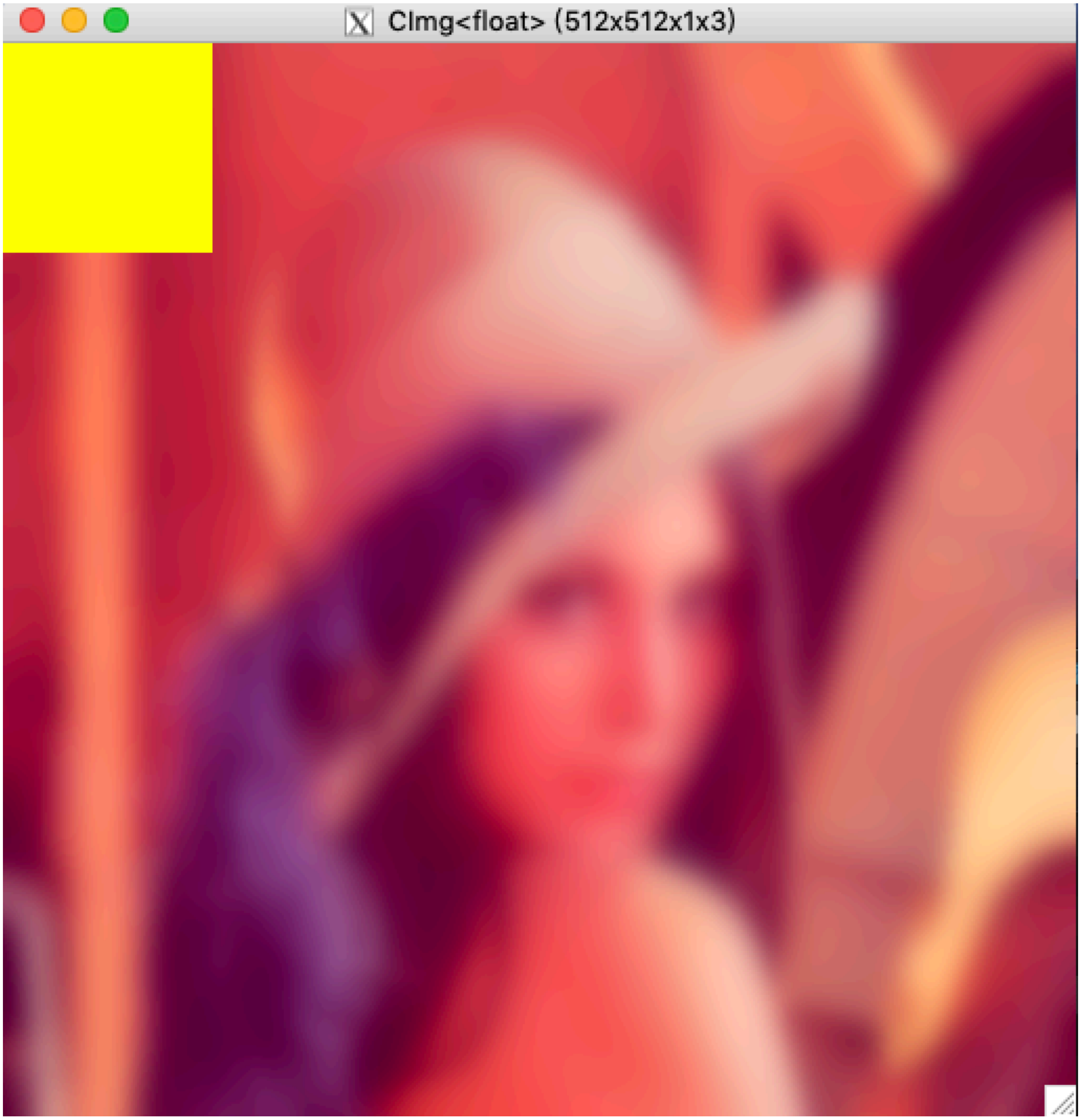
```
cimg_extension::Layer<float> top = sys.get_top_layer();
```

We can easily get the top layer and display the within CImg.

To merge all the layers and generate a new layer, we can simply use:

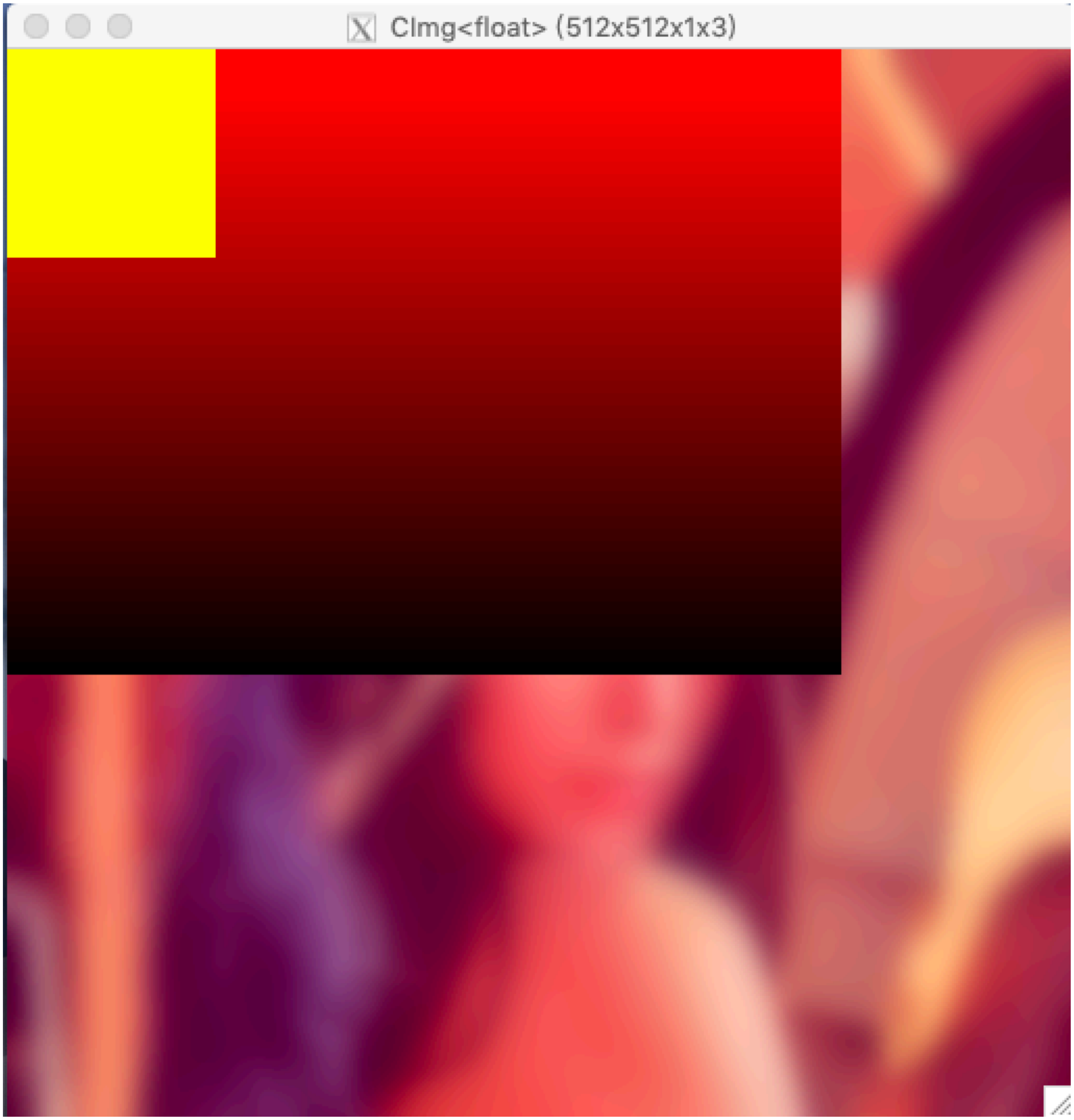
```
cimg_extension::Layer<float> layer = *(sys.merge_layer());
```

The image we generated is as below:



## Layer visibility

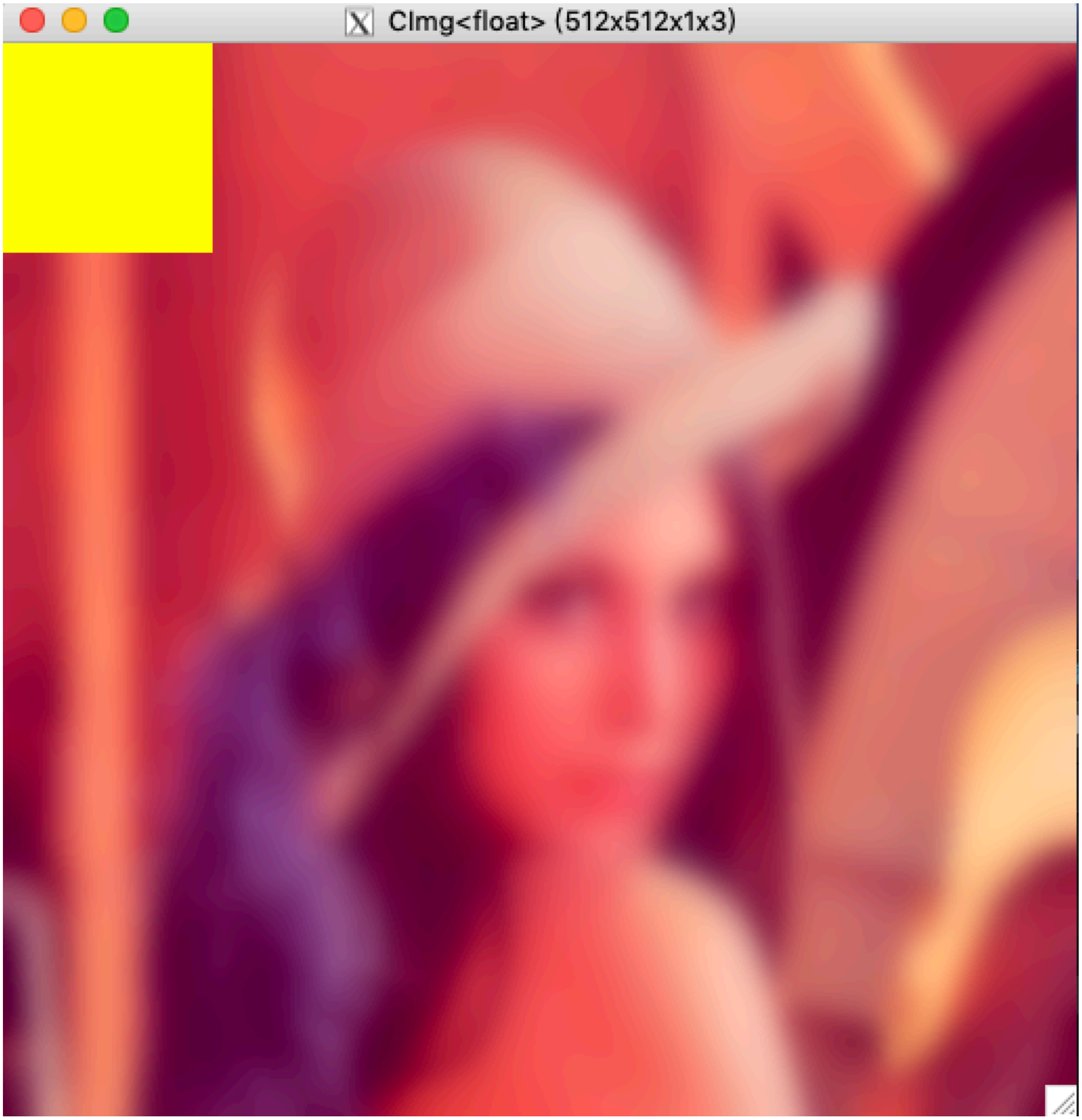
Now suppose we have three layers, the resulted image is as below:



We can set a specific layer to be invisible by using:

`sys.set_invisible(sys.data(1));`

`sys.data(1)` refers to the second layer, it's now set invisible. So the result is:



You can modify the code in example.cpp to try whatever combination you like.