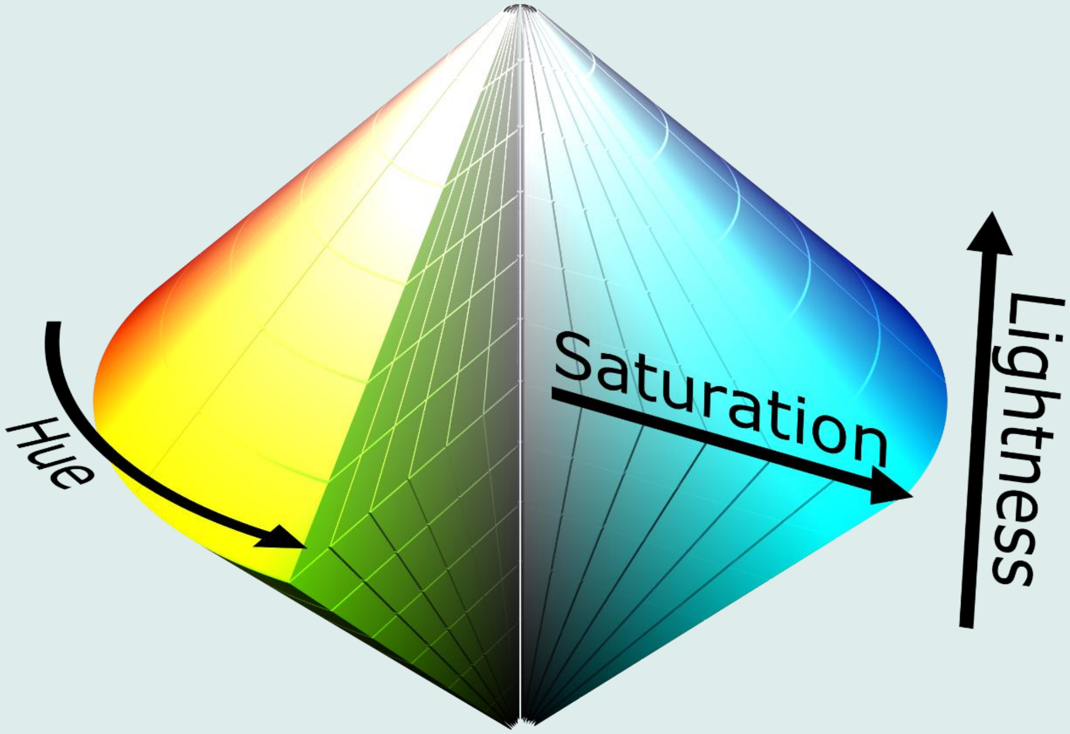
24/10/2021

HSLA image



Sayf-allah nazihe

Cyber-Security Group

École d'Ingénierie Digitale et d'Intelligence Artificielle

Introduction

Our world today is full by images, pictures and screenshots, using a lot of filters from social media application or a professional software. Those images are stored digitally as an array of pixels and every pixel represent a color defined as color space. Filters are a mathematical function that take an image a return it to new image as output. Our gool is to understand how the filters work using the HSL Image color space and coding it using C++ language and qt platform for software development.

&

Objective: Understanding The HSL images manipulation applying inheritance

Images are generally represented by a combination of three main colors called RGB color system (RED-GREEN-BLUE)
The HSL color system that we will use contain a combination of three components:

- **Hue:** define the color itself
- **Saturation:** indicates the degree to which the hue differs from a neutral gray
- **Luminance:** indicates the level of illumination.

Using the inheritance diagram our goal will be writing additional classes that inherit from a PNG class and add the functionalities:

```
class PNG{

   PNG();   //default constructor

   PNG(int, int): //constructor with width and height

   ~PNG();       //Destructor

   bool readFromFile(string);  //read from a file

   bool writeToFile(string);  //write content to a file

   HSLAPixel  getPixel(int x, int y); //get content for pixel x, y


};
```

Inheritance diagram:

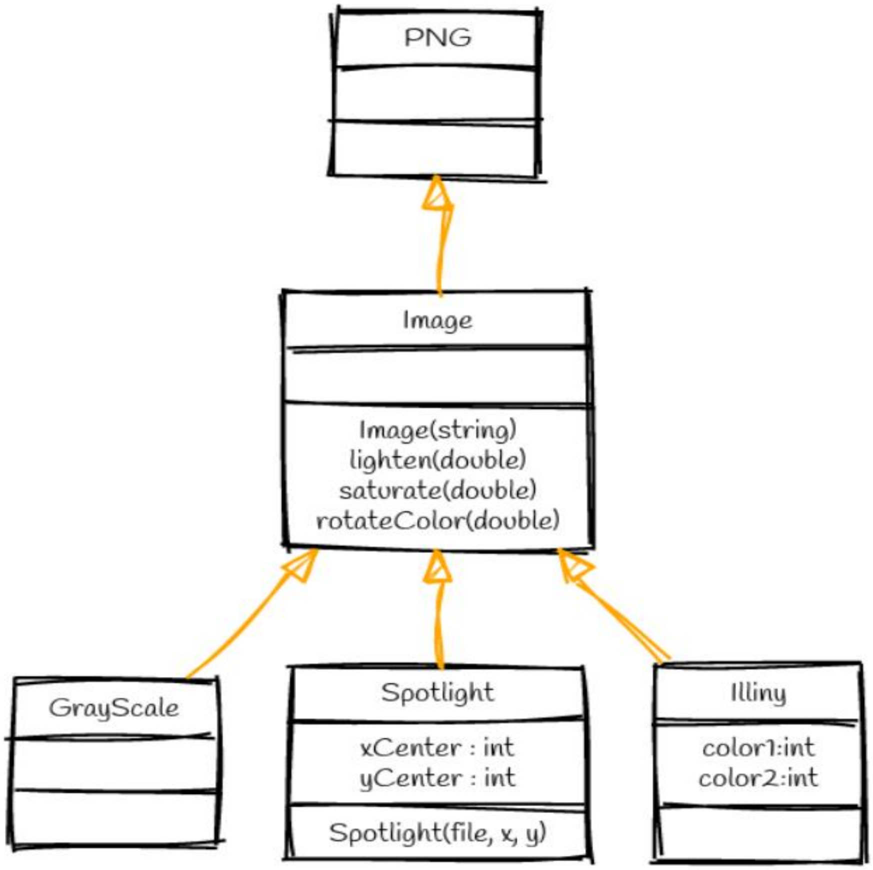


Image class:

This class is inherited from the png class that mean all the attributes in png class are useful in the image class. The class contain a special constructor and three methods:

Image.h

```
using std::string; // stand for standard library
class Image: public PNG { //inheritance from the PNG class
  public:
using PNG::PNG; //use all the attributes that locate in PNG class
    Image(string ); //Constructor

    void lighten(double amount=0.1); //void lighten that contain the
amount as double variable remains in the range [0.1]
```

```
    void saturate(double amount=0.1); // change the luminace using the
amount as double variable remains in the range [0.1]

    void rotateColor(double angle); add the value of angle to each
pixel remains in the range [0.360]
};
```

Image.cpp

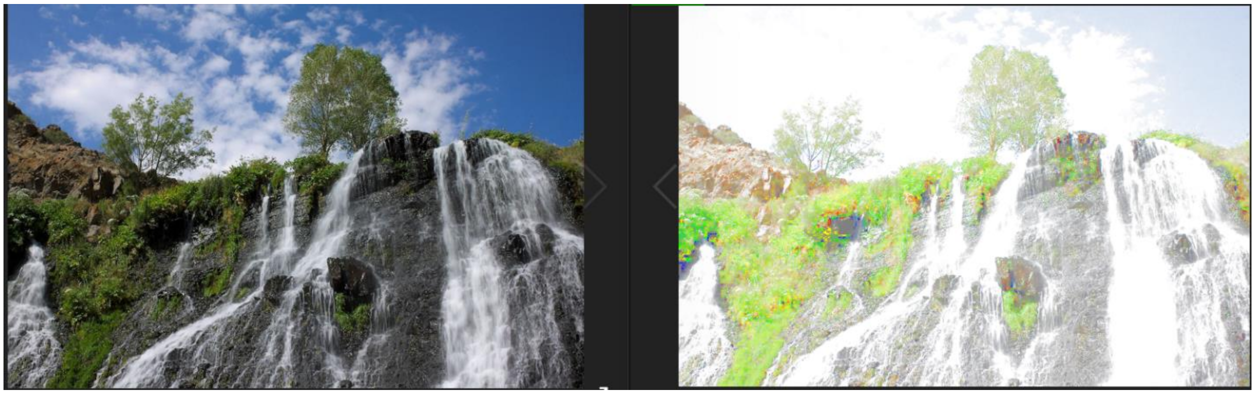
Add the implementation of the constructor Image:

```
Image::Image(string filename):PNG(){
    readFromFile(filename);//to read the file
}
```

Add the implementation of the lighten:

```
void Image::lighten(double amount) {
  for (unsigned x = 0; x < this->width(); x++) {
    for (unsigned y = 0; y < this->height(); y++) {      //loop the
image pixel

      HSLAPixel & pixel = this->getPixel(x, y);//reference on the
pixel

      pixel.l += amount;//increase the luminance of every pixel by
amount

        pixel.l=(pixel.l<1)?pixel.l :1;
        pixel.l=(pixel.l<0)?0 :pixel.l;      //ensure that the
                                              luminance stay in
                                                  the range [0.1)
    }
  }
}
//the implementation of the lighten in the main class:
    Image I;
I.readFromFile("res/aa.png");//image that call aa.png locate in the
res file
I.lighten(0.6);//add the amount of the lighten
I.writeToFile("res/lighten. png");//create a new image called lighten.png that
content the changes of the image
```

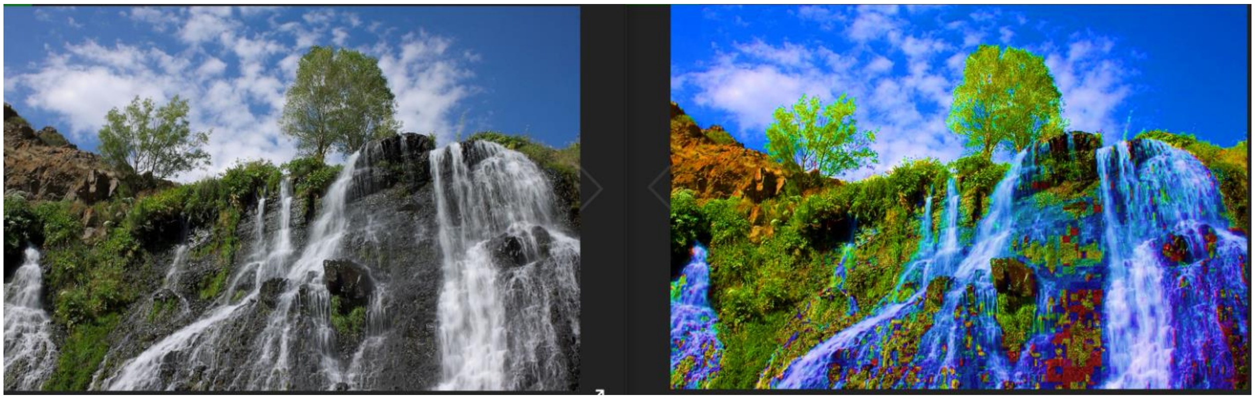
Result of the lighten image



Add the implementation of the saturation:

```
void Image::saturate(double amount) {
  for (unsigned x = 0; x < this->width(); x++) {
    for (unsigned y = 0; y < this->height(); y++) {       //loop the
image pixel

      HSLAPixel & pixel = this->getPixel(x, y);//reference on the
pixel

      pixel.s += amount;//increase the saturation of every pixel by
amount

        pixel.s=(pixel.s<1)?pixel.s :1;
        pixel.s=(pixel.s<0)?0 :pixel.s;       //ensure that the
saturation stay

                                                        in the range [0.1)
    }
  }
}
//the implementation of the saturation  in the main class:
   Image I;
I.readFromFile("res/aa.png");//image that call aa.png locate in the
res file
I.saturate(0.6);//add the amount of the saturation
I.writeToFile("res/saturate. png");//create a new image called saturate.png that
content the changes of the image
```

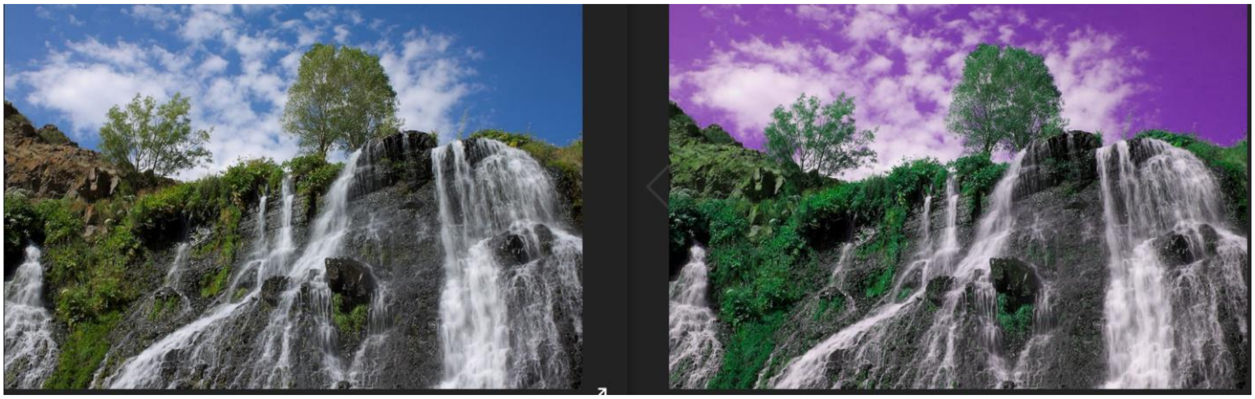
Result of the saturate image



Add the rotate color implementation:

```
void Image::rotateColor(double angle) {

  for (unsigned x = 0; x < this->width(); x++) {
    for (unsigned y = 0; y < this->height(); y++) {        //loop the
image pixels

   HSLAPixel & pixel = this->getPixel(x, y); //reference on the pixel


    pixel.h += angle; //increase the rotation of every pixel by angle


      if (pixel.h > 360) {
        pixel.h = pixel.h - 360;
      } else if (pixel.h < 0) {                   //ensure that the rotate
                                                          range bee
                                                      between[0.360]

        pixel.h = pixel.h + 360;
      }
    }
  }
}
//the implementation of the saturation  in the main class:
    Image I;
I.readFromFile("res/aa.png");//image that call aa.png locate in the
res file
I.rotateColor(60);//add the amount of the saturation
I.writeToFile("res/rotateColor. png");//create a new image called rotateColor.png
that content the changes of the image
```

Result of the rotateColor image



Grayscale class:

A class that inherit from image class to eliminate all the colors on the image using grayscale level:

Grayscale.h

```
class Grayscale: public Image //inheritance from the image class
{
public:
    using Image::Image;//all the attributes of image class are useful in this class
using PNG::writeToFile; //write a content to a file using it from the PNG class
    Grayscale(string filename);//know the name of the file that we wanna transform it
  void CreateGrayscale();//a method to eliminate the colors into grayscale level

};
```

Grayscale.cpp

Add the implementation of the constructor:

```
Grayscale::Grayscale(string filename):Image()
{
    readFromFile(filename);//read the file
}
```

Add the implementation of the method:

```
void Grayscale:: CreateGrayscale(){
```

```
for (unsigned x = 0; x <width(); x++) {
    for (unsigned y = 0; y < height(); y++) {  }  //loop the image
pixels

      HSLAPixel & pixel = getPixel(x, y);//reference on the pixel

      pixel.s = 0;//the saturation of every pixel is set to 0
    }
  }
}
```

Illini class:

This class inherits from the image class, and represent the image using two colors (orange=11 and blue=216)

Illini.h

```
class Illini:public Image //inheritance from the image class
{
public:
    using Image::Image;//all attributes of image class are useful in
this class
using PNG::writeToFile;//inherit the method writeTofile from the PNG
class
   Illini(string filename, int color1=11, int color2=216);// a special
constructor that contain the colors and their numbers and the file
name

};
```

Illini.cpp

Add the constructor implementation:

```
Illini::Illini(string filename,int color1,int color2):Image()
{
    readFromFile(filename);

    for(unsigned x = 0; x < width() ; x++)       }   //loop the image
                                                             pixels
      for(unsigned y = 0; y < height(); y++)
      {

         HSLAPixel &P = getPixel(x, y); //reference on the pixel
if(P.h>11 && P.h<318)
{
int distance1=abs(P.h-color1);
int distance2=abs(P.h-color2);          //we calculate the abs(for
                                          positive values) distance of
                                        the colors to choose which color is
                                               the closest and admitted
                                                     on the image
if(distance1<distance2)
    P.h=color1;
else P.h=color2;
}
else
    P.h=color1;
   }}
```

Spotlight class:

This class inherits from the image class, it adjust the luminance by 0.5% per 1 pixel, up to 80% decrease of luminance

Spotlight.h

```
class spotlight: public Image //inheritance from the image class
{
public:
    using Image::Image;//inheritance from the image class all the
attributes are useful in this class
    using PNG::writeToFile;//inheritance of the writeTofile method
from the PNG class
    int centerX;     //to indicate the center
    int centerY;
    spotlight(string filename,int centerX,int centerY);//special
constructor that have the filename and two points
  void changeSpotPoint(int centerX,int centerY); //the method that change the
position of the spotlight using the two points of center
};
```

Spotlight.cpp

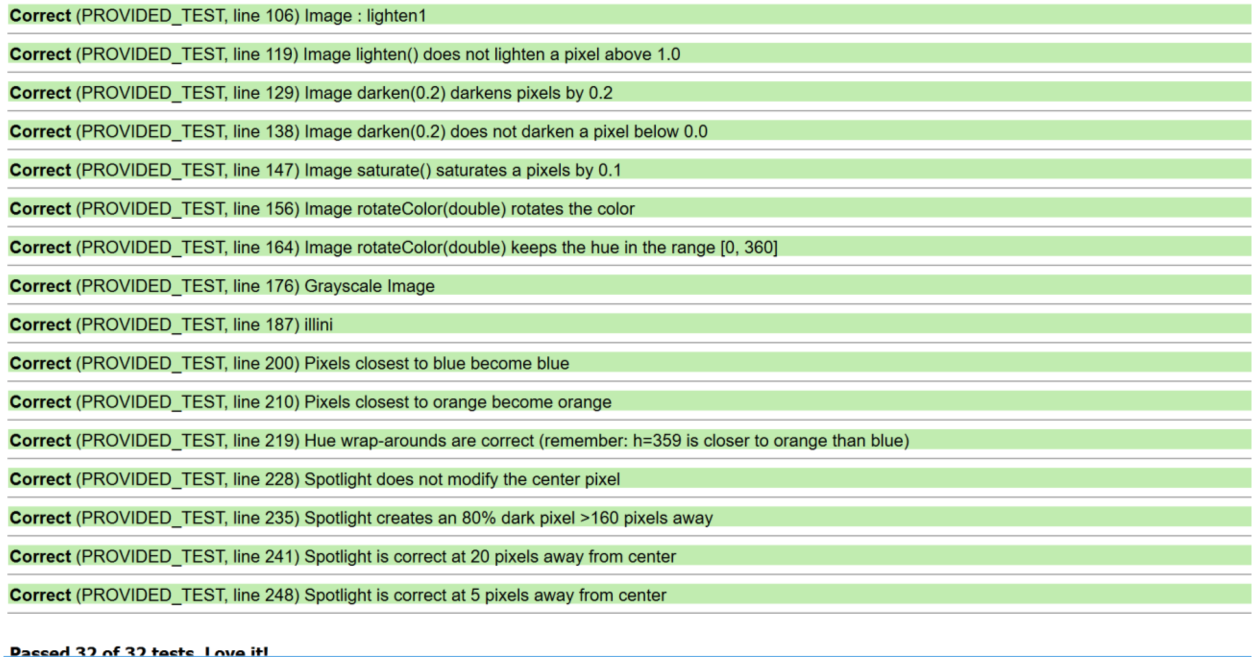
Add the implementation of the constructor and the method createSpotlight():

```
spotlight::spotlight(string filename,int centerX,int centerY):Image(){
    readFromFile(filename);//to read the file name

    for(unsigned x=0;x<width();x++)
        for(unsigned y=0;y<height();y++){     //loop the image pixels
```

```
            double distance = sqrt((x-centerX)*(x-centerX)+(y-
centerY)*(y-centerY));//calculate the distance of the pixel that is
away from the center

            HSLAPixel &P =getPixel(x,y);//reference on the pixel
            if(distance<160){

                P.l=abs(P.l-(distance)*0.005*P.l);




            }
            else {
               P.l=0.2*P.l;
            }
}
}
```

//if the Distance is over than 160 pixels the luminance will be decreased

Tests Results :



Correct (PROVIDED_TEST, line 106) Image : lighten1

Correct (PROVIDED_TEST, line 119) Image lighten() does not lighten a pixel above 1.0

Correct (PROVIDED_TEST, line 129) Image darken(0.2) darkens pixels by 0.2

Correct (PROVIDED_TEST, line 138) Image darken(0.2) does not darken a pixel below 0.0

Correct (PROVIDED_TEST, line 147) Image saturate() saturates a pixels by 0.1

Correct (PROVIDED_TEST, line 156) Image rotateColor(double) rotates the color

Correct (PROVIDED_TEST, line 164) Image rotateColor(double) keeps the hue in the range [0, 360]

Correct (PROVIDED_TEST, line 176) Grayscale Image

Correct (PROVIDED_TEST, line 187) illini

Correct (PROVIDED_TEST, line 200) Pixels closest to blue become blue

Correct (PROVIDED_TEST, line 210) Pixels closest to orange become orange

Correct (PROVIDED_TEST, line 219) Hue wrap-arounds are correct (remember: h=359 is closer to orange than blue)

Correct (PROVIDED_TEST, line 228) Spotlight does not modify the center pixel

Correct (PROVIDED_TEST, line 235) Spotlight creates an 80% dark pixel >160 pixels away

Correct (PROVIDED_TEST, line 241) Spotlight is correct at 20 pixels away from center

Correct (PROVIDED_TEST, line 248) Spotlight is correct at 5 pixels away from center

Passed 32 of 32 tests. Love it!

Passed 32 of 32 tests. Nice work!

Conclusion

This report will teach you a lot of skills and techniques using the inheritance that make the code easier , and also how the filters work on your images by using mathematics functions and C++ language . I hope you enjoy my report 😊