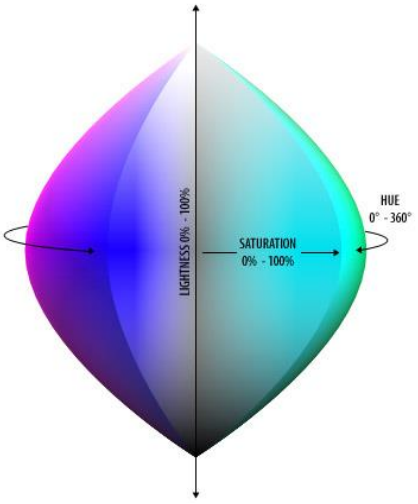
École d'Ingénierie Digitale et d'Intelligence Artificielle

GUI Programming with Qt

Report: Applying inheritance knowledge to manipulate HSLa images.



This PNG image named " aa "will be used for tests

SQUALLI HOUSSAINI YASSINE

BOUZIDI SALAH

THE TASK :

Create a class named `Image` that **inherits** from the `PNG` class. This means that your class will inherits all the attributes and members from the `PNG` class. Meaning that anything you can do with a `PNG` you can do with an `Image`.

Addionally, you should add the following methods:

- `Image (string filename)`: a special constructor that loads the image from the given `filename`.
- `lighten (double amount)` changes the luminance of each pixel by amount.
  - The function must ensures that luminance remains in the range [0,1]
- `saturate` changes the **luminance** by amount.
  - Again you should verify that the luminance stays in [0,1].
- `rotateColor(double angle)`: add the value of angle to each pixel.
  - The value of a color is in cyclic value [0,360].

Image.h

```
#ifndef IMAGE_H
#define IMAGE_H

#include "PNG.h"
#include "HSLAPixel.h"
using std::string;
class Image: public PNG // PNG class inheritance
{public:
using PNG::PNG;
    Image(string );
  // void lighten();
    void lighten(double amount=0.1);
    //void saturate();
void saturate(double amount=0.1);
 void rotateColor(double angle);
 };
```

Image.CPP

```
Image::Image(string filename):PNG(){
    readFromFile(filename);}
```

//Add the implementation of the constructor Image

We add the implementation of the lighten in Image.cpp:

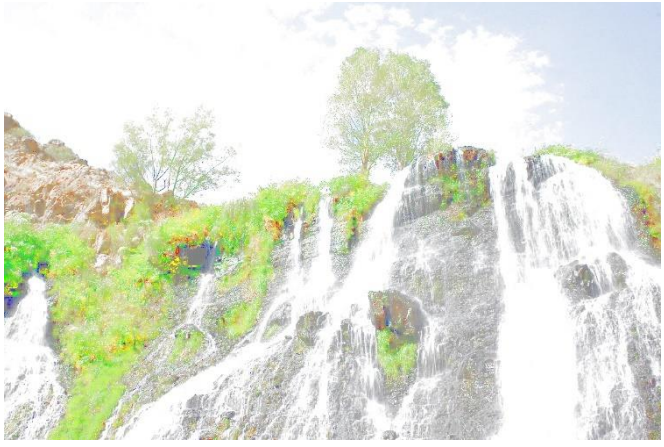
```
void Image::lighten(double amount) {
  for (unsigned x = 0; x < this->width(); x++) {
    for (unsigned y = 0; y < this->height(); y++) {
      HSLAPixel & pixel = this->getPixel(x, y);

 // Lighten an Image by increasing the luminance of
every pixel by amount.
      pixel.l += amount;
        pixel.l=(pixel.l<1)?pixel.l :1;
        pixel.l=(pixel.l<0)?0 :pixel.l;
    }
  }
}
```

we add the lighten to the main class :

```
Image I ;
I.readFromFile("res/aa.png");                    //call aa.png located in the res file
I.lighten(0.6);//to add the amount of the lighten
I.writeToFile ("res/lighten. png");//helps create a new image called lighten.png
that have  the changes of the image
```

The result :

Before



After

We add the implementation of the saturation in Image .cpp:

```
void Image::saturate(double amount) {
  for (unsigned x = 0; x < this->width(); x++) {
    for (unsigned y = 0; y < this->height(); y++) {
      HSLAPixel & pixel = this->getPixel(x, y);

      // Lighten an Image by increasing the
saturation of every pixel by amount.

      pixel.s += amount;
        pixel.s=(pixel.s<1)?pixel.s :1;
        pixel.s=(pixel.s<0)?0 :pixel.s;
    }
  }
}
```

we add saturation to the main class

```
Image I;
I.readFromFile("res/aa.png");
I.saturate(0.6);
I.writeToFile("res/saturate. png");;
```

The result:

Before

After

We add the implementation of the rotateColor in Image .cpp:

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

  for (unsigned x = 0; x < this->width(); x++) {
    for (unsigned y = 0; y < this->height(); y++) {
      HSLAPixel & pixel = this->getPixel(x, y);

      pixel.h += angle;

      if (pixel.h > 360) {
        pixel.h = pixel.h - 360;
      } else if (pixel.h < 0) {
        pixel.h = pixel.h + 360;
      }
    }
}
```

we add rotateColor to the main class:

```
Image I;
I.readFromFile("res/aa.png");
I.rotateColor(60);
I.writeToFile("res/rotateColor. png");
```

The result:

Before

After

- Grayscale :Now you should write a simple class `Grayscale` that inherits from the `Image` class. This is a simple class that eliminates all the colors and represents the image using only a `grayscale` level.
- Illini: Create a class called `Illini` that inherits from the `Image` class. An `Illini` image has only two colors that are defined as attributes.
- SpotlightA : create a **spotlight** centered at a given point `centerX`, `centerY` defined as attributes.

Grayscale.h

```
class Grayscale: public Image //inheritance from the image class
{
public:
    using Image::Image;
 using PNG::writeToFile;
    Grayscale(string filename);
   void CreateGrayscale();//to eliminate the colors into grayscale
};
```

Grayscale.cpp

```
#include "grayscale.h"
Grayscale::Grayscale(string filename):Image()
{
    readFromFile(filename);
saturate(-1);
}//Add the implementation of the constructor


void Grayscale:: CreateGrayscale() Add the implementation of the method
{for (unsigned x = 0; x <width(); x++) {
    for (unsigned y = 0; y < height(); y++) {
      HSLAPixel & pixel = getPixel(x, y);
      pixel.s = 0;
    }
  }
}
```

Illini.h

```
class Illini:public Image //inheritance from the image class
{
public:
     using Image::Image;
     using PNG::writeToFile;
     //to inherit the method writeTofile from the PNG class

      Illini(string filename,int color1=11,int
color2=216);// constructor
};
```

Illini.cpp

```
#include "illini.h"
Illini::Illini(string filename,int color1,int
color2):Image()
{
     readFromFile(filename);
     for(unsigned x = 0; x < width() ; x++)
       for(unsigned y = 0; y < height(); y++)//loop
       {
          //reference on the pixel
          HSLAPixel &P = getPixel(x, y);
          //modifiy the element of P
if(P.h>11 && P.h<318)
{
int distance1=abs(P.h-color1);
int distance2=abs(P.h-color2);
if(distance1<distance2)
     P.h=color1;
else P.h=color2;
}
else
     P.h=color1;
   }}
```

Spotligh .h

```
#ifndef SPOTLIGHT_H
#define SPOTLIGHT_H
#include "image.h"

class spotlight: public Image //inheritance from the image class
{

public:
    using Image::Image;
    using PNG::writeToFile;
 //inheritance of the writeTofile method from the PNG class
    int centerX;
    int centerY;
    spotlight(string filename,int centerX,int
centerY);
    void changeSpotPoint(int centerX,int centerY);
};
```

Spotligh .cpp

```
#include "spotlight.h"
#include "math.h"

spotlight::spotlight(string filename,int centerX,int
centerY):Image(){
    readFromFile(filename);
```

```
    for(unsigned x=0;x<width();x++)
        for(unsigned y=0;y<height();y++)
            //loop
           { double distance = sqrt((x-
centerX)*(x-centerX)+(y-centerY)*(y-centerY));
            HSLAPixel &P =getPixel(x,y);
            if(distance<160){

                P.l=abs(P.l-
(distance)*0.005*P.l);
            }
            else {
                P.l=0.2*P.l;
            }
}
}
```

TEST RESULTS: