azPNG

C++ PNG Creator

Made by
AZAR Majed Saliou
majed.azar7@gmail.com
majedazar.com

Lome, Togo
17 December 2020

Table of Contents

Overview

Language : C++

Platforms : Windows, Linux

Summary : A library used to create .png images using shapes (pixels, lines, rectangles, etc...)

Description :

The azPNG library has been written primarly to create pictures using the .png format. Images can be drawn by the programmer using pixels, lines, rectangles or other geometric shapes. The functions used to draw are quite elementary but simple to pick up and understand.

Since this library is focusing only on drawing the image, it doesn't have any feature to display the output (if you want to see the results, open the .png image created using another software). Here, the .ppm format is used to create the image and then the output is converted to .png. Due to this fact, you can use 24bits colors (red, green and blue from 0 to 255) but there's no support for transparency.

Set Up

In order to use the azPNG library inside a C++ project, you first need to include the header file associated with it :

```
#include "azPNG.h"
```

Then, the next step depends on your system.

On Linux : You must install **netpbm** to be able to convert your .ppm images into .png :

```
sudo apt-get install netpbm
```

On Windows : You must copy the file **convert.exe** next to the executable of every program which uses the azPNG library. This way, it'll use it in order to convert your .ppm images into .png.

When you're done with the setup, you can finally use the objects inside the namespace azPNG :

```
azPNG::Image newImage(640,480,azPNG::Color::White());
```

Color Object

The Color object is the object used to represent pixels. It's using 3 variables to represent the intensity of the red, green and blue light to form a color. 0 is the minimum level of intensity (black) and 255 is the max (white).

Constructors

The Color object comes with 3 differents constructors. The 1st one creates the color black, the 2nd creates a grey variant and the 3rd can create any color.

```
Color()
```
Description : Creates a black pixel

Parameters : none

```
Color(short const& I_GREY)
```
Description : Creates a grey pixel using grayscale

Parameters :

- I_GREY : integer representing the intensity of the grey (from 0 to 255)

```
Color(short const& I_RED, short const& I_GREEN, short const& I_BLUE)
```
Description : Creates a colored pixel using rgb standards

Parameters :

- I_RED : integer representing the intensity of the red (from 0 to 255)
- I_GREEN : integer representing the intensity of the green (from 0 to 255)
- I_BLUE : integer representing the intensity of the blue (from 0 to 255)

Getters

The getters linked with the Color object can provide a way to get the intensity of the red, green and blue used for the image.

```
unsigned short GetRed() const
```
Description : Returns the intensity of the red used inside the pixel

Parameters : none

Return : an integer representing the intensity of the red

```
unsigned short GetGreen() const
```
Description : Returns the intensity of the green used inside the pixel

Parameters : none

Return : an integer representing the intensity of the green

```
unsigned short GetBlue() const
```
Description : Returns the intensity of the blue used inside the pixel

Parameters : none

Return : an integer representing the intensity of the blue

```
std::string GetColor() const
```
Description : Returns all the intensity of all the colors used inside the pixel

Parameters : none

Return : a string representing the intensity of each color separated by a space (ex : "0 127 255")

Setters

The setters linked with the Color object can provide a way to set the intensity of the red, green and blue used for the image.

```
bool SetRed(short const& I_INTENSITY)
```
Description : Sets the intensity of the red used inside the pixel to a given value

Parameters :

- I_INTENSITY : integer representing the value of the intensity (from 0 to 255)

Return : true if the intensity was changed or **false** if the value given was incorrect

```
bool SetGreen(short const& I_INTENSITY)
```
Description : Sets the intensity of the green used inside the pixel to a given value

Parameters :

- I_INTENSITY : integer representing the value of the intensity (from 0 to 255)

Return : true if the intensity was changed or **false** if the value given was incorrect

```
bool SetBlue(short const& I_INTENSITY)
```
Description : Sets the intensity of the blue used inside the pixel to a given value

Parameters :

- I_INTENSITY : integer representing the value of the intensity (from 0 to 255)

Return : true if the intensity was changed or **false** if the value given was incorrect

```
bool SetColor(short const& I_RED, short const& I_GREEN, short const& I_BLUE)
```
Description : Sets the color inside the pixel to a given value

Parameters :

- I_RED : integer representing the value of the intensity of the red (from 0 to 255)
- I_GREEN : integer representing the value of the intensity of the green (from 0 to 255)
- I_BLUE : integer representing the value of the intensity of the blue (from 0 to 255)

Return : true if the color was changed or **false** if the color given was incorrect

```
bool SetColor(Color const& CL_COLOR)
```
Description : Sets the color inside the pixel to a given value

Parameters :

- CL_COLOR : Color Object representing the color which needs to be applied

Return : true if the color was changed or **false** if the color given was incorrect

Predefined colors

The Color object also features some predefined colors that you can use without having to manually give the intensity of the rgb components (ex : azPNG::Color::Black()).

```
Color static RandColor()
```
Description : Returns a random color

Parameters : none

Return : a Color Object representing a random color

```
Color static RandRed()
```
Description : Returns a random variant of red

Parameters : none

Return : a Color Object representing a random variant of red

```
Color static RandGreen()
```
Description : Returns a random variant of green

Parameters : none

Return : a Color Object representing a random variant of green

```
Color static RandBlue()
```
Description : Returns a random variant of blue

Parameters : none

Return : a Color Object representing a random variant of blue

```
Color static RandYellow()
```
Description : Returns a random variant of yellow

Parameters : none

Return : a Color Object representing a random variant of yellow

```
Color static RandCyan()
```

Description : Returns a random variant of cyan

Parameters : none

Return : a Color Object representing a random variant of cyan

```
Color static RandMagenta()
```
Description : Returns a random variant of magenta

Parameters : none

Return : a Color Object representing a random variant of magenta

```
Color static RandGrey()
```
Description : Returns a random variant of grey

Parameters : none

Return : a Color Object representing a random variant of grey

```
Color static Black()
```
Description : Returns a the color black

Parameters : none

Return : a Color Object representing the color black

```
Color static Red()
```
Description : Returns a the color red

Parameters : none

Return : a Color Object representing the color red

```
Color static Green()
```
Description : Returns a the color green

Parameters : none

Return : a Color Object representing the color green

```
Color static Blue()
```
Description : Returns a the color blue

Parameters : none

Return : a Color Object representing the color blue

```
Color static Yellow()
```
Description : Returns a the color yellow

Parameters : none

Return : a Color Object representing the color yellow

```
Color static Cyan()
```
Description : Returns a the color cyan

Parameters : none

Return : a Color Object representing the color cyan

```
Color static Magenta()
```
Description : Returns a the color magenta

Parameters : none

Return : a Color Object representing the color magenta

```
Color static White()
```
Description : Returns a the color white

Parameters : none

Return : a Color Object representing the color white

```
Color static DarkRed()
```
Description : Returns a the color dark red

Parameters : none

Return : a Color Object representing the color dark red

```
Color static DarkGreen()
```
Description : Returns a the color dark green

Parameters : none

Return : a Color Object representing the color dark green

```
Color static DarkBlue()
```
Description : Returns a the color dark blue

Parameters : none

Return : a Color Object representing the color dark blue

```
Color static DarkYellow()
```
Description : Returns a the color dark yellow

Parameters : none

Return : a Color Object representing the color dark yellow

```
Color static DarkCyan()
```
Description : Returns a the color dark cyan

Parameters : none

Return : a Color Object representing the color dark cyan

```
Color static DarkMagenta()
```
Description : Returns a the color dark magenta

Parameters : none

Return : a Color Object representing the color dark magenta

```
Color static Grey()
```
Description : Returns a the color grey

Parameters : none

Return : a Color Object representing the color grey

```
Color static VeryDarkRed()
```
Description : Returns a the color very dark red

Parameters : none

Return : a Color Object representing the color very dark red

```
Color static VeryDarkGreen()
```
Description : Returns a the color very dark green

Parameters : none

Return : a Color Object representing the color very dark green

```
Color static VeryDarkBlue()
```
Description : Returns a the color very dark blue

Parameters : none

Return : a Color Object representing the color very dark blue

```
Color static VeryDarkYellow()
```
Description : Returns a the color very dark yellow

Parameters : none

Return : a Color Object representing the color very dark yellow

```
Color static VeryDarkCyan()
```
Description : Returns a the color very dark cyan

Parameters : none

Return : a Color Object representing the color very dark cyan

```
Color static VeryDarkMagenta()
```
Description : Returns a the color very dark magenta

Parameters : none

Return : a Color Object representing the color very dark magenta

```
Color static DarkGrey()
```
Description : Returns a the color dark grey

Parameters : none

Return : a Color Object representing the color dark grey

Image Object

The image object is used to create our image. It's a grid of I_WIDTH x I_HEIGHT pixels (the Color object). This grid can be modified pixel by pixel but it also supports geometric shapes such as lines, rectangles, triangles, trapeze and circle. These shapes can be drawn empty (only the outline will then come out) or they can be drawn filled.

Constructors

The Image object comes with 2 constructors. The 1st creates a black image and the 2nd creates an image filled with the color you defined.

```
Image(short const& I_WIDTH, short const& I_HEIGHT)
```
Description : Constructs an Image filled with the color black

Parameters :

- I_WIDTH : integer representing the width of the image
- I_HEIGHT : integer representing the height of the image

```
Image(short const& I_WIDTH, short const& I_HEIGHT, Color const& CL_COLOR)
```
Description : Constructs an Image filled with the color given in parameter

Parameters :

- I_WIDTH : integer representing the width of the image
- I_HEIGHT : integer representing the height of the image
- CL_COLOR : Color Object representing the color which needs to be applied

Getters

In this class, there is conventional getters to get the width, the height or even a pixel on the image but there is also getters to get a cropped or scaled version of an image.

```
unsigned short GetWidth() const
```
Description : Returns the width of the image

Parameters : none

Return : an integer representing the width of the image

```
unsigned short GetHeight() const
```
Description : Returns the height of the image

Parameters : none

Return : an integer representing the height of the image

```
Color GetPixel(short const& I_X, short const& I_Y) const
```
Description : Returns a pixel inside the image

Parameters :

- I_X : integer representing the x coordinate of the pixel
- I_Y : integer representing the y coordinate of the pixel

Return : a Color Object representing a pixel inside the image

```
Image GetCrop(short I_X1, short I_Y1, short I_X2, short I_Y2) const
```
Description : Returns a cropped version of the image

Parameters :

- I_X1 : integer representing the x coordinate of the 1st point
- I_Y1 : integer representing the y coordinate of the 1st point
- I_X2 : integer representing the x coordinate of the 2nd point
- I_Y2 : integer representing the y coordinate of the 2nd point

Return : an Image Object representing the cropped image

```
Image GetScale(short const& I_RATIO) const
```
Description : Returns a scaled version of the image

Parameters :

- I_RATIO : integer determining by how many pixels we should scale the image

Return : an Image Object representing the scaled image

Drawing routines

The Image object is packaged with various functions to draw geometric shapes. You can draw pixel, lines, rectangles, cubes, triangles, trapeze and circles filled or not.

```
void DrawPixel(float const& F_X, float const& F_Y, Color const& CL_COLOR = Color(255))
```

Description : Draw a pixel inside the image

Parameters :

- F_X : float representing the x coordinate of the pixel
- F_Y : float representing the y coordinate of the pixel
- CL_COLOR (default = Color::White()) : Color Object representing the color of the pixel

Return : none

```
void DrawLine(float const& F_X1, float const& F_Y1, float const& F_X2, float const& F_Y2, Color const& CL_COLOR = Color(255))
```

Description : Draw a line inside the image

Parameters :

- F_X1 : float representing the x coordinate of the 1st point used to draw
- F_Y1 : float representing the y coordinate of the 1st point used to draw
- F_X2 : float representing the x coordinate of the 2nd point used to draw
- F_Y2 : float representing the y coordinate of the 2nd point used to draw
- CL_COLOR (default = Color::White()) : Color Object representing the color of the line

Return : none

```
void DrawRectangle(float F_X1, float F_Y1, float F_X2, float F_Y2, Color const& CL_COLOR = Color(255))
```

Description : Draw a rectangle inside the image

Parameters :

- F_X1 : float representing the x coordinate of the 1st point used to draw
- F_Y1 : float representing the y coordinate of the 1st point used to draw
- F_X2 : float representing the x coordinate of the 2nd point used to draw
- F_Y2 : float representing the y coordinate of the 2nd point used to draw
- CL_COLOR (default = Color::White()) : Color Object representing the color of the shape

Return : none

```
void FillRectangle(float F_X1, float F_Y1, float F_X2, float F_Y2, Color const&
CL_COLOR = Color(255))
```

Description : Draw a filled rectangle inside the image

Parameters :

- F_X1 : float representing the x coordinate of the 1st point used to draw
- F_Y1 : float representing the y coordinate of the 1st point used to draw
- F_X2 : float representing the x coordinate of the 2nd point used to draw
- F_Y2 : float representing the y coordinate of the 2nd point used to draw
- CL_COLOR (default = Color::White()) : Color Object representing the color of the shape

Return : none

```
void DrawCube(float F_X, float F_Y, float F_C, Color const& CL_COLOR = Color(255))
```

Description : Draw a cube inside the image

Parameters :

- F_X : float representing the x coordinate of the point used to draw
- F_Y : float representing the y coordinate of the point used to draw
- F_C : float representing the length of the cube
- CL_COLOR (default = Color::White()) : Color Object representing the color of the shape

Return : none

```
void FillCube(float F_X, float F_Y, float F_C, Color const& CL_COLOR = Color(255))
```

Description : Draw a filled cube inside the image

Parameters :

- F_X : float representing the x coordinate of the point used to draw
- F_Y : float representing the y coordinate of the point used to draw
- F_C : float representing the length of the cube
- CL_COLOR (default = Color::White()) : Color Object representing the color of the shape

Return : none

```
void DrawTriangle(float const& F_X1, float const& F_Y1, float const& F_X2, float
const& F_Y2, float const& F_X3, float const& F_Y3 , Color const& CL_COLOR =
Color(255))
```

Description : Draw a triangle inside the image

Parameters :

- F_X1 : float representing the x coordinate of the 1st point used to draw
- F_Y1 : float representing the y coordinate of the 1st point used to draw
- F_X2 : float representing the x coordinate of the 2nd point used to draw
- F_Y2 : float representing the y coordinate of the 2nd point used to draw
- F_X3 : float representing the x coordinate of the 3rd point used to draw
- F_Y3 : float representing the y coordinate of the 3rd point used to draw
- CL_COLOR (default = Color::White()) : Color Object representing the color of the shape

Return : none

```
void FillTriangle(float const& F_X1, float const& F_Y1, float const& F_X2, float const& F_Y2, float const& F_X3, float const& F_Y3 , Color const& CL_COLOR = Color(255))
```

Description : Draw a filled triangle inside the image

Parameters :

- F_X1 : float representing the x coordinate of the 1st point used to draw
- F_Y1 : float representing the y coordinate of the 1st point used to draw
- F_X2 : float representing the x coordinate of the 2nd point used to draw
- F_Y2 : float representing the y coordinate of the 2nd point used to draw
- F_X3 : float representing the x coordinate of the 3rd point used to draw
- F_Y3 : float representing the y coordinate of the 3rd point used to draw
- CL_COLOR (default = Color::White()) : Color Object representing the color of the shape

Return : none

```
void DrawTrapeze(float const& F_X1, float const& F_Y1, float const& F_X2, float const& F_Y2, float const& F_X3, float const& F_Y3, float const& F_X4, float const& F_Y4 , Color const& CL_COLOR = Color(255))
```

Description : Draw a trapeze inside the image

Parameters :

- F_X1 : float representing the x coordinate of the 1st point used to draw
- F_Y1 : float representing the y coordinate of the 1st point used to draw
- F_X2 : float representing the x coordinate of the 2nd point used to draw

- F_Y2 : float representing the y coordinate of the 2nd point used to draw
- F_X3 : float representing the x coordinate of the 3rd point used to draw
- F_Y3 : float representing the y coordinate of the 3rd point used to draw
- F_X4 : float representing the x coordinate of the 4th point used to draw
- F_Y4 : float representing the y coordinate of the 4th point used to draw
- CL_COLOR (default = Color::White()) : Color Object representing the color of the shape

Return : none

```
void FillTrapeze(float const& F_X1, float const& F_Y1, float const& F_X2, float const& F_Y2, float const& F_X3, float const& F_Y3, float const& F_X4, float const& F_Y4 , Color const& CL_COLOR = Color(255))
```

Description : Draw a filled trapeze inside the image

Parameters :

- F_X1 : float representing the x coordinate of the 1st point used to draw
- F_Y1 : float representing the y coordinate of the 1st point used to draw
- F_X2 : float representing the x coordinate of the 2nd point used to draw
- F_Y2 : float representing the y coordinate of the 2nd point used to draw
- F_X3 : float representing the x coordinate of the 3rd point used to draw
- F_Y3 : float representing the y coordinate of the 3rd point used to draw
- F_X4 : float representing the x coordinate of the 4th point used to draw
- F_Y4 : float representing the y coordinate of the 4th point used to draw
- CL_COLOR (default = Color::White()) : Color Object representing the color of the shape

Return : none

```
void DrawCircle(float const& F_X, float const& F_Y, float const& F_R, Color const& CL_COLOR = Color(255))
```

Description : Draw a circle inside the image

Parameters :

- F_X : float representing the x coordinate of the point used to draw
- F_Y : float representing the y coordinate of the point used to draw
- F_R : float representing the radius of the circle
- CL_COLOR (default = Color::White()) : Color Object representing the color of the shape

Return : none

```
void FillCircle(float const& F_X, float const& F_Y, float const& F_R, Color const& CL_COLOR = Color(255))
```

Description : Draw a filled circle inside the image

Parameters :

- F_X : float representing the x coordinate of the point used to draw
- F_Y : float representing the y coordinate of the point used to draw
- F_R : float representing the radius of the circle
- CL_COLOR (default = Color::White()) : Color Object representing the color of the shape

Return : none

```
void Clear(Color const& CL_COLOR = Color(255,255,255))
```

Description : Fill the image using a color

Parameters :

- CL_COLOR (default = Color::White()) : Color Object representing the color applied to the image

Return : none

Saving

Finally, the following function can be used to save an Image object once the work is done.

```
bool Save(std::string const& S_PATH = "image.ppm", bool const& B_CONVERT = true)
```

Description : Saves the image to .ppm and converts it to .png

Parameters :

- S_PATH (default = image.ppm) : string representing the path and the name where the image should be saved (ex : blank.ppm)
- B_CONVERT : boolean determining if the image while be converted after having been saved or not

Return : true if the image could be saved else **false**