SHAPE MANAGEMENT : Shape Class

The `Shape` class is used as a base for all our shapes' handling inside our program.
It is imperative to use it if you want to implement a new shape that is compatible with our program.

All the methods used in graphic libraries to display an element in the screen take a class inheriting from `Shape`. For each class that you want to add that herit from the `Shape` class, you must implement a `draw` method in all the graphic libraries as well.

Shape's methods :

This is the list of methods that your new shape will have by heriting the Shape class.

```
void setPosition(const vec2int& newPosition)
```

- Sets a new position for the shape.

```
vec2int getPosition() const
```

- Returns the shape's position in the `vect2int` format.
- The `vec2int` structure contains a pair of int `x` and `y`.

```
void setColor(const color_uint8& color)
```

- Sets the new color for the shape.
- It takes a reference to a `color_unit8` structure in parameter.
- It contains three `unsigned char` named `r`, `g` and `b` in order to handle colors with RGB.

```
color_uint8 getColor() const
```

- Gets the shape's color in the `color_uint8` format.

Shape's attributes:

This is the list of attributes that your new shape will have by heriting the Shape class.

```
vec2int shapePosition
```

- The initial position of your shape, stored as a `vec2int`. By default the value is set to `{-1, -1}`.

```
color_uint8 shapeColor = {0, 0, 0};
```

- The initial color of your shape, stored as a `color_uint8`. By default the value is set to `{0, 0, 0}` (black).

Feel free to add more methods and attributes according to your shape's needs. For example, our `Rectangle` class, that herits from `Shape` has a boolean `isFilled` attribute in order to create filled and unfilled rectangle shapes.