Assignment 4

/*叶俊利 3130100672*/

1. Translation

First, calculation the size of the new image and allocate space for the new image:

```
int h = imgOriginal.GetHeight();
int w = imgOriginal.GetWidth();
int x0 = 100;
int y0 = 100;
CImage imgNew.Create(w+y0, h+x0, imgOriginal.GetBPP());
```

Then, according to the equation

$$\begin{cases} x' = x + x_0 \\ y' = y + y_0 \end{cases}$$

So we can process the image according to the equation.

The code is shown below.

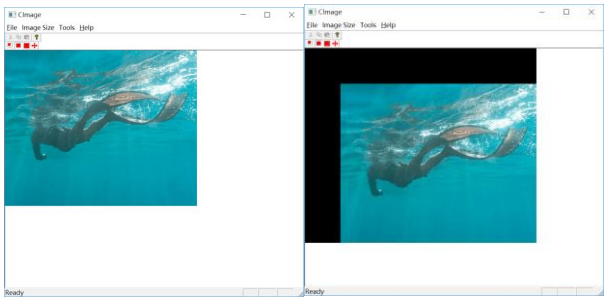
```
for (int i = 0; i < h; i++)
for (int j = 0; j < w; j++){
        pixel = imgOriginal.GetPixel(j, i);
        imgNew.SetPixel(j+y0, i+x0, pixel);

}
```

The unset pixel of the new image is black.

At last, copy the content of the new image into the original image:

```
imgOriginal.Destroy();
imgOriginal.Create(imgNew.GetWidth(), imgNew.GetHeight(), imgNew.GetBPP());
for (int i = 0; i < imgNew.GetHeight(); i++){
        for (int j = 0; j < imgNew.GetWidth(); j++){
                pixel = imgNew.GetPixel(j, i);
                imgOriginal.SetPixel(j, i, pixel);
        }
}
```

According to the this, we can get the result:



2. Rotation

First, calculation the size of the new image and allocate space for the new image:

```
int h = imgOriginal.GetHeight();
int w = imgOriginal.GetWidth();
CImage imgNew.Create(round(w*cos(theta) + h*sin(theta)), round(w*sin(theta) +
h*cos(theta)), imgOriginal.GetBPP());
```

Then, according to the equation

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases}$$

In order to do interpolation, I assign the values of pixels in the new image by the pixel on inverse location in the original image. So we can process the image according to the equation.

The code is shown below.

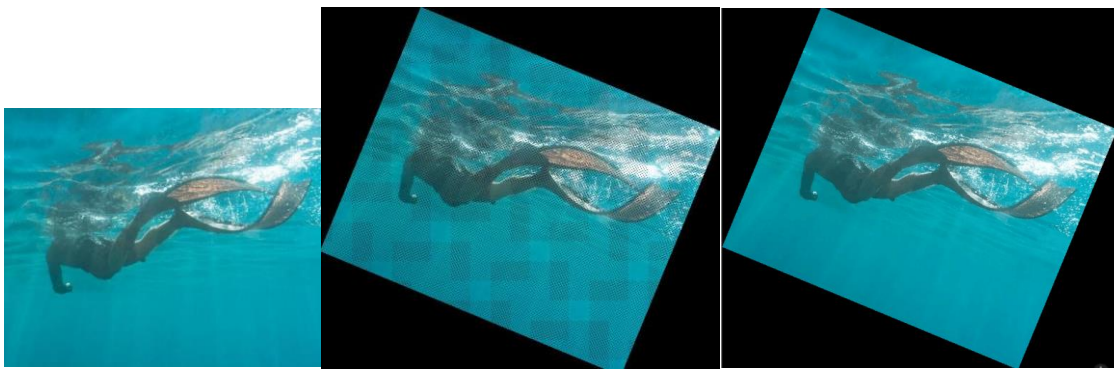
```
for (int i = 0; i < round(w*sin(theta) + h*cos(theta)); i++)
for (int j = 0; j < round(w*cos(theta) + h*sin(theta)); j++){
    pixel = imgOriginal.GetPixel(round((j - h*sin(theta))*cos(-theta) - i*sin(-
theta)), round((j - h*sin(theta))*sin(-theta) + i*cos(-theta)));
    imgNew.SetPixel(j, i, pixel);
}
```

The unset pixel of the new image is black.

At last, copy the content of the new image into the original image:

```
imgOriginal.Destroy();
imgOriginal.Create(imgNew.GetWidth(), imgNew.GetHeight(), imgNew.GetBPP());
for (int i = 0; i < imgNew.GetHeight(); i++){
    for (int j = 0; j < imgNew.GetWidth(); j++){
        pixel = imgNew.GetPixel(j, i);
        imgOriginal.SetPixel(j, i, pixel);
    }
}
```

According to the this, we can get the result:



Original Without Interpolation After Interpolation

3. Scale

First, calculation the size of the new image and allocate space for the new image:

```
double c = 1.5;
double d = 1.5;

int h = imgOriginal.GetHeight();
int w = imgOriginal.GetWidth();
imgNew.Create(w*c, h*d, imgOriginal.GetBPP());
```

Then, according to the equation

$$\begin{cases} x' = cx \\ y' = dy \end{cases}$$

In order to do interpolation, I assign the values of pixels in the new image by the pixel on inverse location in the original image. So we can process the image according to the equation.

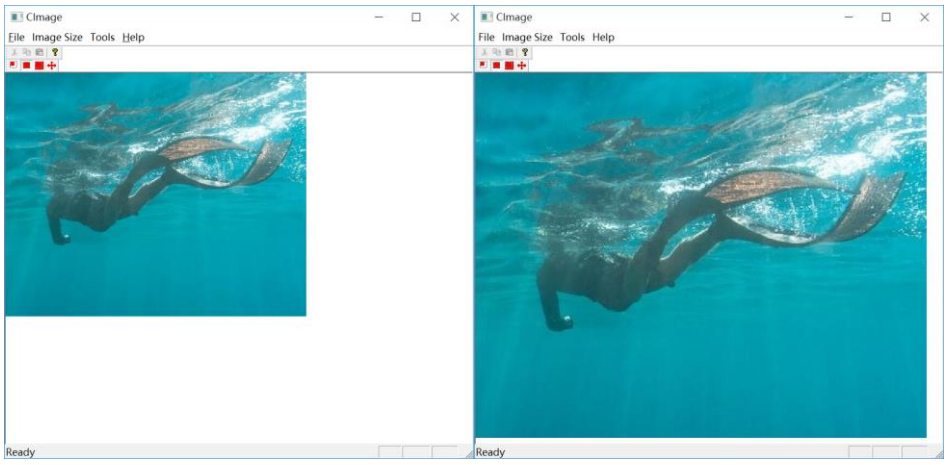
The code is shown below.

```
for (int i = 0; i < h*d; i++)
for (int j = 0; j < w*c; j++){
        pixel = imgOriginal.GetPixel(j/c, i/d);
        imgNew.SetPixel(j, i, pixel);
}
```

At last, copy the content of the new image into the original image:

```
imgOriginal.Destroy();
imgOriginal.Create(imgNew.GetWidth(), imgNew.GetHeight(), imgNew.GetBPP());
for (int i = 0; i < imgNew.GetHeight(); i++){
        for (int j = 0; j < imgNew.GetWidth(); j++){
                pixel = imgNew.GetPixel(j, i);
                imgOriginal.SetPixel(j, i, pixel);
        }
}
```

According to the this, we can get the result:



4. Shear

First, calculation the size of the new image and allocate space for the new image:

```
double d = 0.2;
int h = imgOriginal.GetHeight();
int w = imgOriginal.GetWidth();
imgNew.Create(w,  h+d*w, imgOriginal.GetBPP());
```

Then, according to the equation

$$\begin{cases} x' = x + x_0 \\ y' = y + y_0 \end{cases}$$

So we can process the image according to the equation.

The code is shown below.

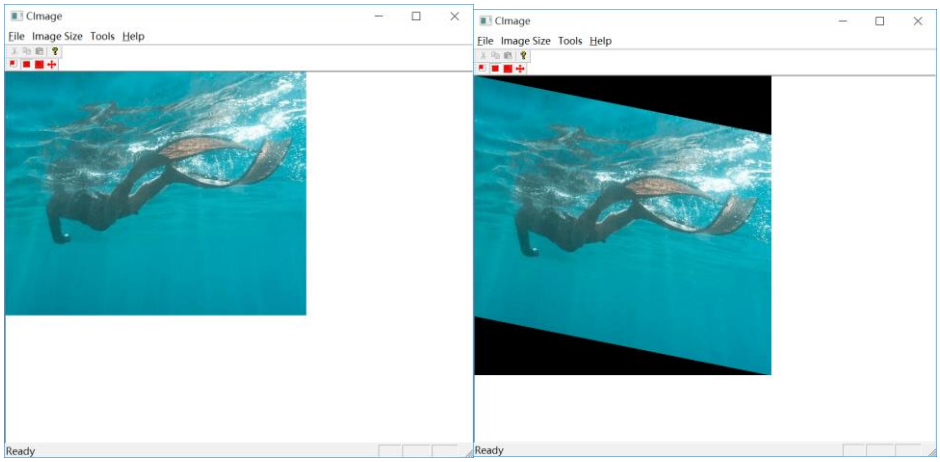
```
for (int i = 0; i < h + d*w; i++)
for (int j = 0; j < w; j++){
        pixel = imgOriginal.GetPixel(j, i-d*j);
        imgNew.SetPixel(j , i, pixel);
}
```

The unset pixel of the new image is black.

At last, copy the content of the new image into the original image:

```
imgOriginal.Destroy();
imgOriginal.Create(imgNew.GetWidth(), imgNew.GetHeight(), imgNew.GetBPP());
for (int i = 0; i < imgNew.GetHeight(); i++){
        for (int j = 0; j < imgNew.GetWidth(); j++){
                pixel = imgNew.GetPixel(j, i);
                imgOriginal.SetPixel(j, i, pixel);
        }
}
```

According to the this, we can get the result:



5. Mirror

First, calculation the size of the new image and allocate space for the new image:

```
int h = imgOriginal.GetHeight();
int w = imgOriginal.GetWidth();
imgNew.Create(w, h, imgOriginal.GetBPP());
```

Then, according to the equation

$$\begin{cases} x' = x \\ y' = -y \end{cases}$$

So we can process the image according to the equation.

The code is shown below.

```
for (int i = 0; i < h; i++)
for (int j = 0; j < w; j++){
        pixel = imgOriginal.GetPixel(j, i);
        imgNew.SetPixel(j, h-i, pixel);
}
```

At last, copy the content of the new image into the original image:

```
imgOriginal.Destroy();
imgOriginal.Create(imgNew.GetWidth(), imgNew.GetHeight(), imgNew.GetBPP());
for (int i = 0; i < imgNew.GetHeight(); i++){
        for (int j = 0; j < imgNew.GetWidth(); j++){
                pixel = imgNew.GetPixel(j, i);
                imgOriginal.SetPixel(j, i, pixel);
        }
}
```

According to the this, we can get the result: