作业4

作业4算是很简单的啦，复习了一下 de Casteljau 算法，然后按照实验指导书完成代码如下：

```
cv::Point2f recursive_bezier(const std::vector<cv::Point2f> &control_points, float t)
{
    // TODO: Implement de Casteljau's algorithm
    if(control_points.size()==1)
        return cv::Point2f(control_points[0]);
    std::vector<cv::Point2f> points;
    for (int i =1;i<control_points.size();i++)
    {
        points.emplace_back(control_points[i].x*t+control_points[i-1].x*(1-t),control_points[i].y*t+control_points[i-1].y*(1-t));
    }
    return recursive_bezier(points,t);

}

void bezier(const std::vector<cv::Point2f> &control_points, cv::Mat &window)
{
    // TODO: Iterate through all t = 0 to t = 1 with small steps, and call de Casteljau's
    // recursive Bezier algorithm.

    for (double t = 0; t < 1; t+=0.001)
    {
        auto point = recursive_bezier(control_points, t);
        window.at<cv::Vec3b>(point.y, point.x)[1] = 255;
    }

}
```

提高部分

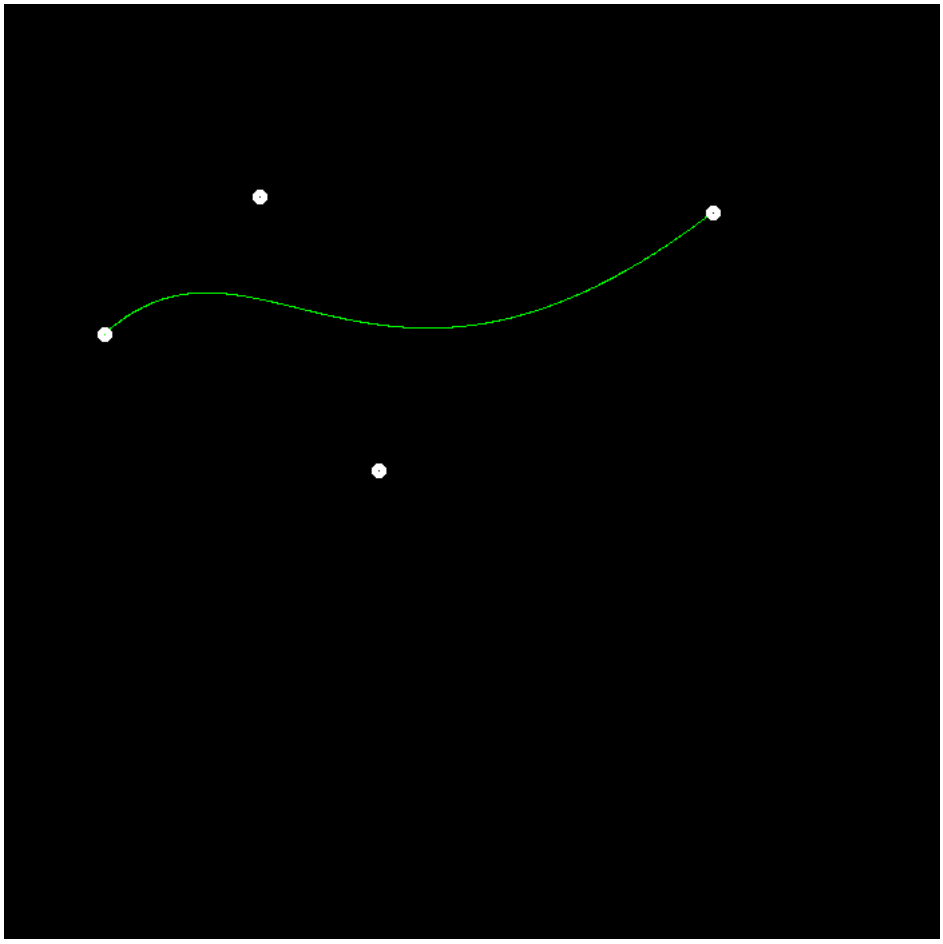
提高部分有比较多需要注意的地方：

- opencv像素的绘制是调用 `window.atcv::Vec3b(int y, int x)` 函数，没错，你没看错，y在前，x在后
- 像素的中心是对应 int 类型加上 0.5f
- 如果想要进行反走样，就需要把当前得到的曲线上的点的周围最近的四个像素点绘上颜色，颜色可以通过这四个像素点距离曲线上的点的距离乘以想要的颜色值得到，代码如下：

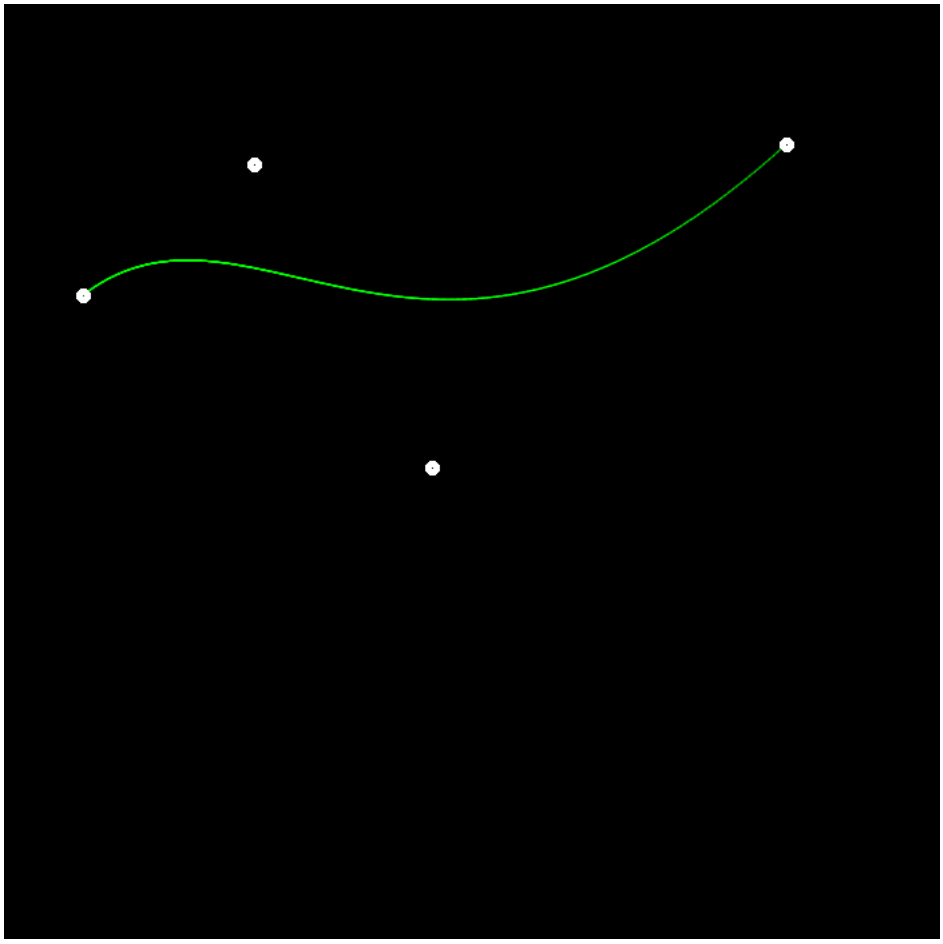
```
void bezier_interpolated(const std::vector<cv::Point2f> &control_points, cv::Mat &window)
{
    for (double t = 0; t < 1; t += 0.001)
    {
        auto point = recursive_bezier(control_points, t);

        cv::Point2i p0(point.x-std::floor(point.x)<0.5?std::floor(point.x):std::ceil(point.x),
                       point.y - std::floor(point.y)<0.5?std::floor(point.y):std::ceil(point.y));
        std::vector<cv::Point2i> p_around = {p0,cv::Point2i(p0.x-1,p0.y),
                           cv::Point2i(p0.x,p0.y-1),cv::Point2i(p0.x-1,p0.y-1)};
        float sum_d = 0.f;
        float max_d = std::sqrt(2);
        std::vector<float> dis;
        for(int i = 0;i<4;i++)
        {
            cv::Point2f pp(p_around[i].x+0.5f,p_around[i].y+0.5f);
            float d = max_d - std::sqrt(std::pow(point.x-pp.x,2)+std::pow(point.y-pp.y,2));
            dis.push_back(d);
            sum_d += d;
        }
        for(int i = 0;i<4;i++)
        {
            float w = dis[i]/sum_d;
            window.at<cv::Vec3b>(p_around[i].y, p_around[i].x)[1] = std::min(255.f,window.at<cv::Vec3b>(p_around[i].y, p_around[i].x)[
        }
    }
}
```

效果图对比：

无反走样：



有反走样：



至此，作业4完成！