

# Warm-up 制作无声小短片视频(HW1)

---

作业要求：

1. 编程生成一个满足以下要求的视频，生成的视频，程序运行结束后会自动保存在一个指定文件名的视频文件中
2. 内容一：刚开始是一个片头，有镜头切换，并显示浙大元素相关照片，以及个人照片。显示自己学号与姓名等信息
3. 内容二：镜头切换后，缓慢画一幅很简单的儿童画。儿童画内容自行设计，写死在程序里。不能读入一个画好的图片
4. 其他内容：自由设计
5. 内容N：编程设计一个片尾动画
6. 编程实现所有镜头切换效果。
7. 有一定故事情节，自由设计
8. 按一下空格，会暂停视频，再按一下则继续

信息：

姓名：鲍奕帆

学号：3180103499

邮箱：[3180103499@zju.edu.cn](mailto:3180103499@zju.edu.cn)

完成日期：2020年11月29日晚上10点30分

## 一、软件开发说明

- 计算机与操作系统

macOS Catalina版本10.15.5

MacBook Pro(13-inch, 2020, Four Thunderbolt 3 ports)

CPU: 2 GHz 四核 Intel Core i5

内存: 16 GB 3733 MHz LPDDR4

启动磁盘: Macintosh HD

图形卡: Intel Iris Plus Graphics 1536 MB

内核(uname -a): Darwin EvandeMacBook-Pro.local 19.5.0 Darwin Kernel Version 19.5.0: Tue May 26 20:41:44 PDT 2020; root:xnu-6153.121.2~2/RELEASE\_X86\_64 x86\_64

- opencv 版本

opencv\_4.4.0

- 编程语言

C++11

- IDE

XCode Version 12.2 (12B45b)

- 编译器

Apple clang version 11.0.3 (clang-1103.0.32.62)

## 二、实验结果展示与分析

- 开场视频效果

### 1. 开场中国电影视频+浙江大学图标

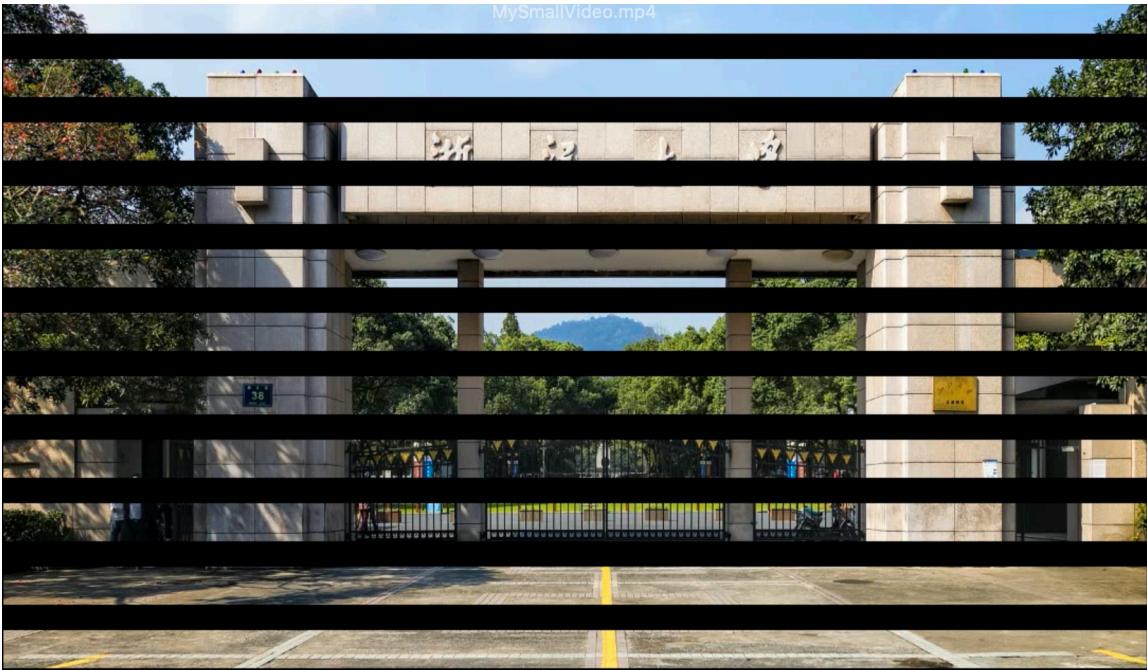


### 2. 浙大元素风景图片显示与转场

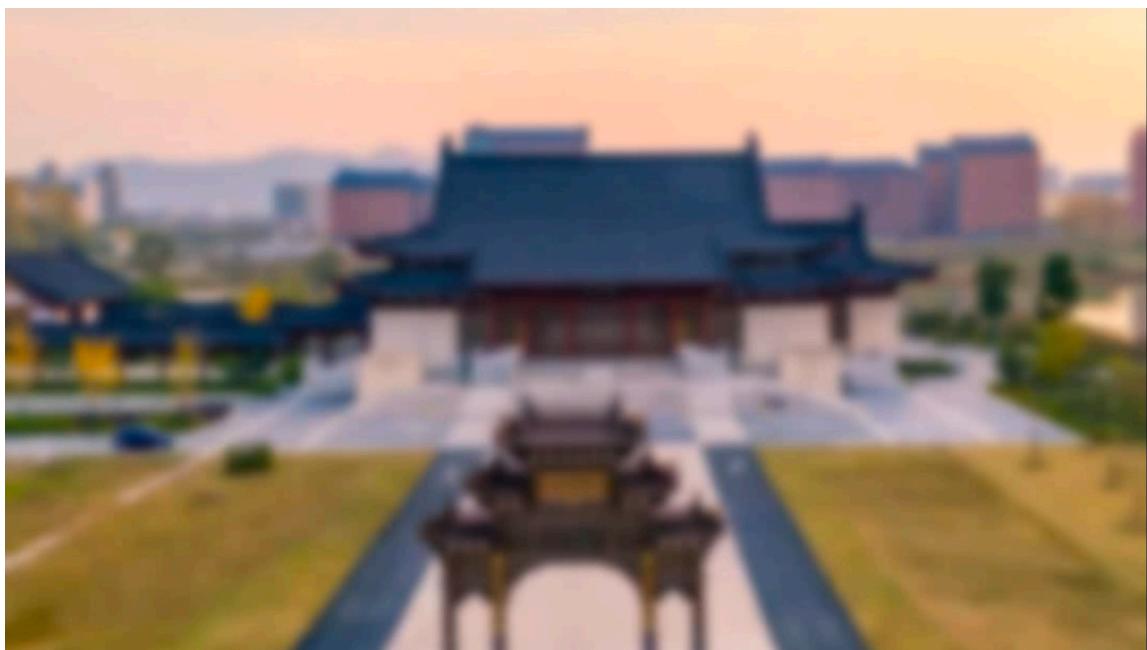
转场效果1



转场效果2 百叶窗



转场效果3 高斯模糊



转场效果4 水平翻页



转场效果5 向上平移(无法直接表示) + 个人照片与信息展示



信息进一步展示

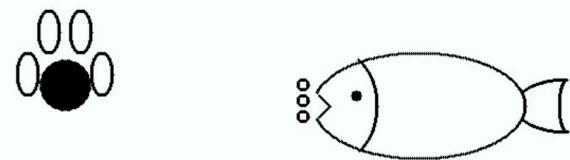
name : Bao Yifan

OpenCV is powerful

Student ID: 3180103499

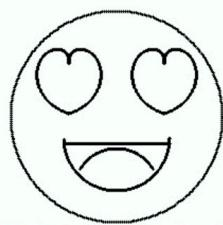
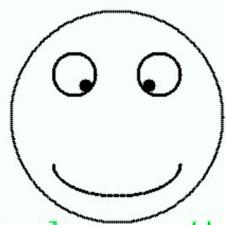
- 儿童画绘制

图画1+设计故事情节1(哲理：鱼和熊掌不可兼得。人不应该有过分的欲望)



You can not have it both ways

图画2+设计故事情节2(哲理：保持平常心，爱其所爱。心想小处，做在实处。)

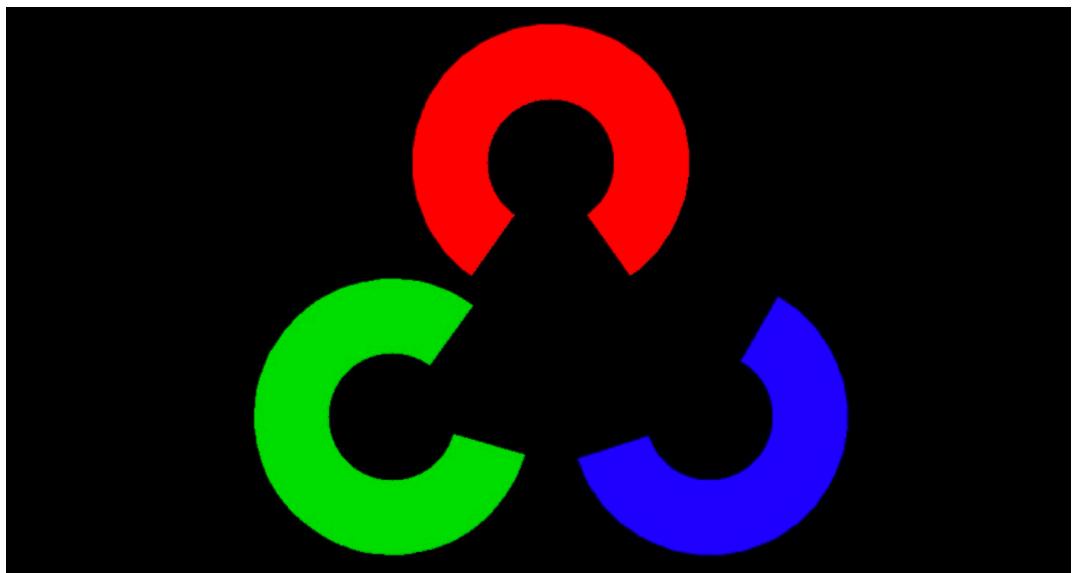


But it doesn't mean you are unhappy

Think small and do big



镜头切换(转场)--opencv logo 动态缓慢绘制

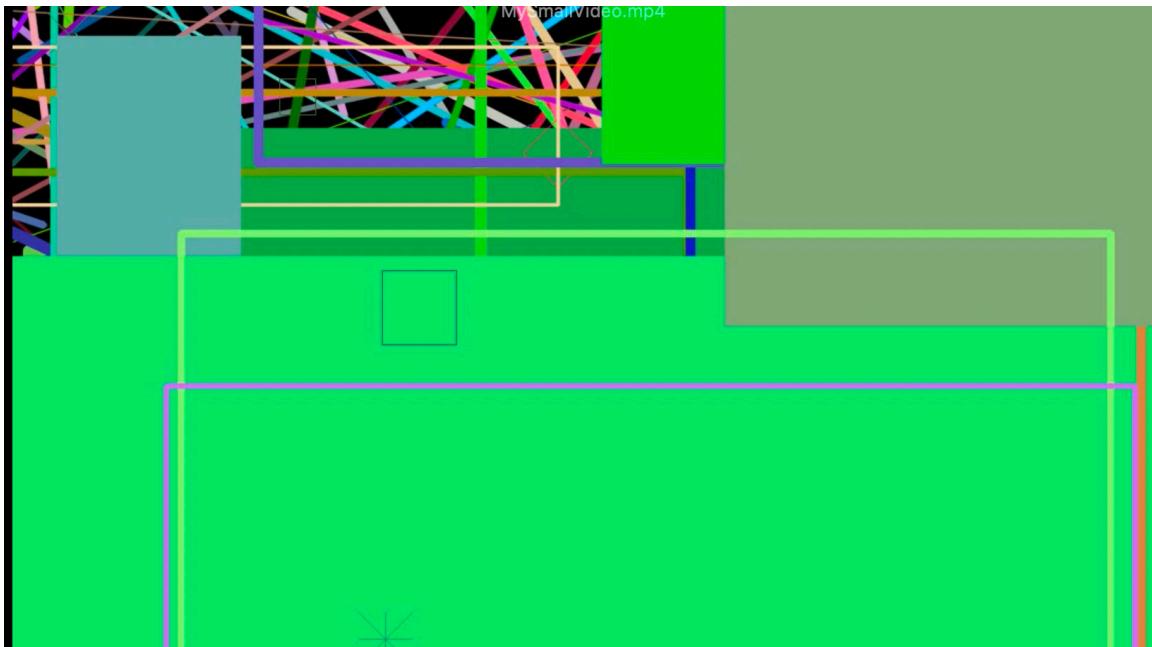


- 片尾动画

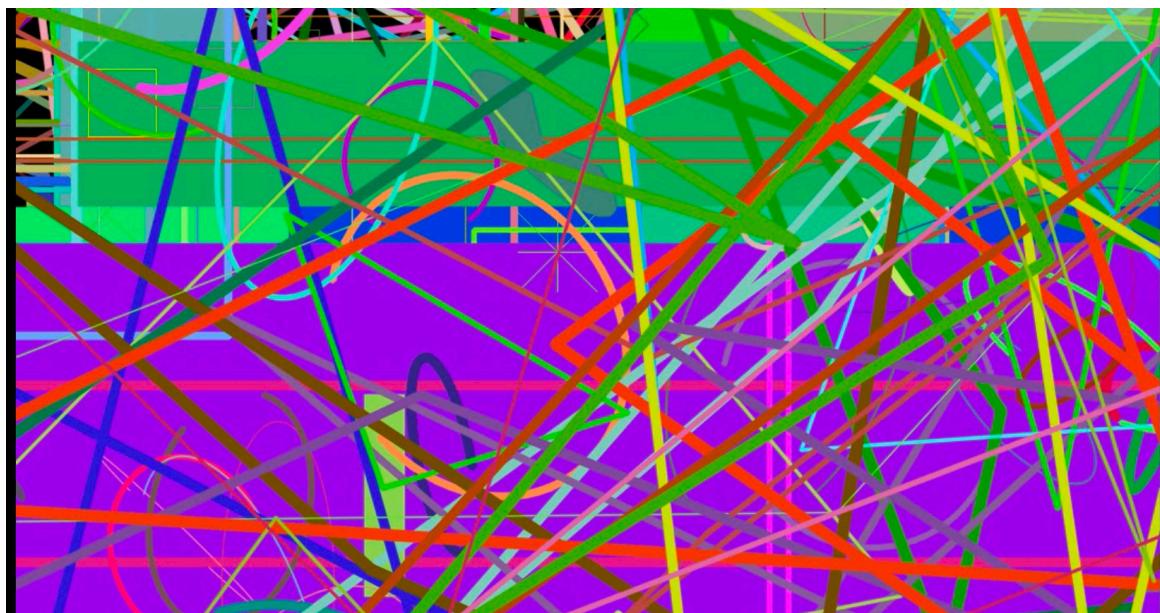
直线线条斑驳，动态快速绘制，形成视觉冲击



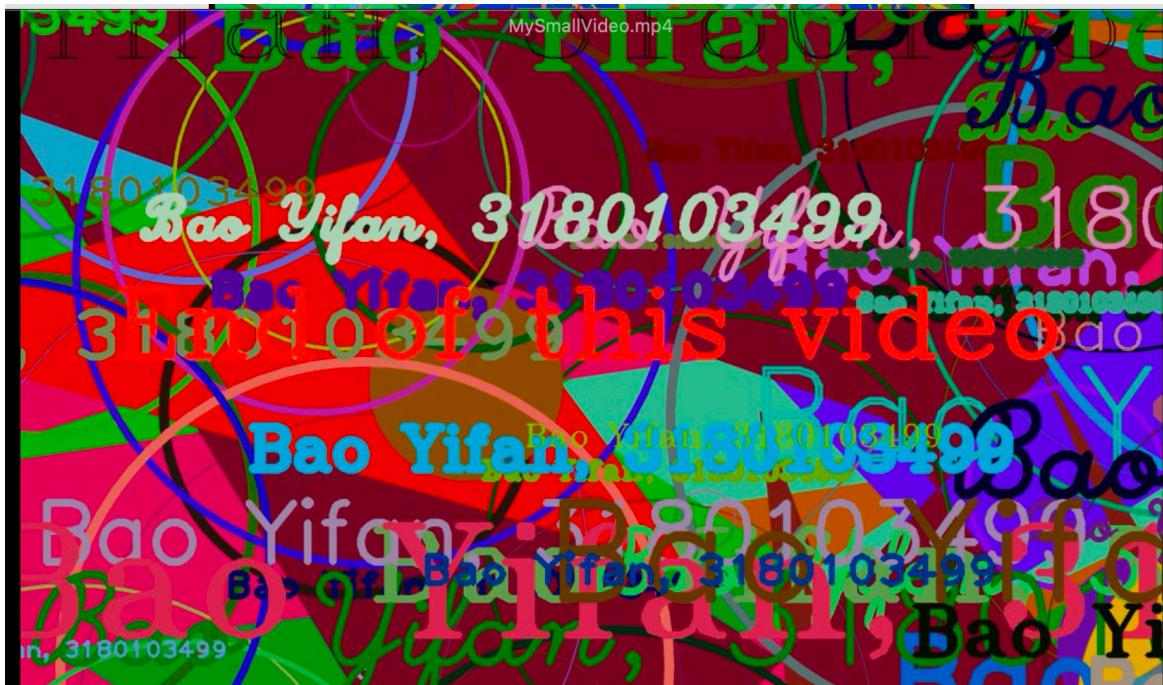
方形Mark交织，动态快速绘制，形成视觉冲击



各种形状绘制，形成酷炫效果。

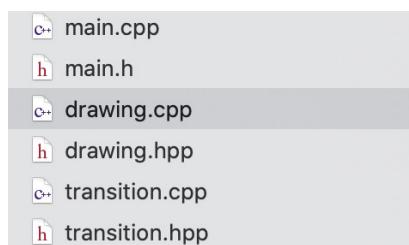


底部显示大量个人信息，渐变消失。最终显示End of this video



### 三、程序基本描述

- 源文件结构



一共有三个cpp文件，分别是main.cpp, drawing.cpp, transition.cpp

下面叙述各个模块的大致功能。

- main.h

```

#define PIC_NUM 6
#define WIDTH 1080
#define HEIGHT 720
#define FPS 30
using namespace cv;
using namespace std;
extern Mat image;
bool getPictures(string& path, vector<Mat>& pictures);
void writeBeginVideo(VideoWriter& writer, string& beginVideoPath);
void writePictures(VideoWriter& writer, vector<Mat>& pictures);
void play();

```

getPictures函数根据路径读取对应的图片到向量

writeBeginVideo函数绘制开头视频

writePictures函数绘制图片，并添加转场效果

play函数在namedWindow中播放视频，并按空格可以暂停播放。

- main函数

```

int main(int argc, char* argv[])
{
    if(argc != 2)
    {
        cout << "invalid argument!" << endl << "./program --create: create
video"
        << "./program --play: play video";
    }
    if(strcmp(argv[1], "--create") == 0){
        string path = "../label_pic/";           //视频和图片存储路径
        vector<Mat> pictures;                  //图片向量 用于保存读入图片
        string beginVideoPath = "../label_pic/begin.mp4"; //开场视频路径
        char writerPath[50] = "../generated_video/MySmallVideo.mp4"; //绘制视频写
路径
        //读入图片并判断是否成功
        if(!getPictures(path, pictures))
        {
            cout << "read picture failed, exit";
            return 1;
        }
        namedWindow("Video");
        //VideoWriter 视频写类
        VideoWriter writer(writerPath, VideoWriter::fourcc('m', 'p', '4', 'v'),
FPS, Size(WIDTH,HEIGHT));
        writeBeginVideo(writer, beginVideoPath); //写片头
        writePictures(writer, pictures);       //写视频并过渡
        //缓慢图画绘制
        init(writer,writerPath);
    }
}

```

```

drawBackground(true); //第一次绘图背景
drawText(); //文本内容
drawFish(); //绘制 鱼
drawBearPaw(); //绘制 熊掌
aphorism(); //哲理
drawBackground(false); //背景刷新
drawWeirdFace(); //笑脸
drawLovingFace(); //爱脸
aphorism2(); //哲理
drawBackground(false); //刷新
image = Mat::zeros(HEIGHT,WIDTH,CV_8UC3);
writer<<image;
logo(); //黑色背景 opencv logo显示
Ending(); //结尾动画 绘制
writer.release(); //释放句柄
}
else if(strcmp(argv[1], "--play") == 0)
{
    play();
}
else
{
    cout << "invalid argument!" << endl << "./program --create: create
video"
    << "./program --play: play video";
}

```

main函数包含两块，根据命令行参数确定。在命令行是 ./program --create的时候根据程序创建视频，在命令行是./program --play的时候根据已经创建好的视频播放。

## 四、程序模块实现算法

- 视频过渡特效

1. 向上平移。图像自下而上滑动。在每一帧中，图像上方的连续行将被复制到画布下方相同数量行的矩形区域中。

```

void transition_effect1(cv::VideoWriter& writer, const cv::Mat& img)
{
    Mat img_translation(img.size(), img.type());
    for (int i = 1; i <= FPS; ++i)
    {
        img_translation.setTo(cv::Scalar(0, 0, 0));
        //图像上方的连续行将被复制到画布下方相同数量行的矩形区域中。
        cv::Rect rect(0, img.rows * (1 - (float)i / (float)FPS), img.cols,
img.rows * (float)i / (float)FPS);
        img.rowRange(0, img.rows * (float)i /
(float)FPS).copyTo(img_translation(rect));
        writer<<img_translation;
    }
}

```

2. 百叶窗。使用mask遮挡图像即可实现。mask中有白条纹。白条纹的宽度逐渐变化，这导致所显示图像的范围发生变化

```

void transition_effect2(cv::VideoWriter& writer, const cv::Mat& img)
{
    const int interval = FPS * 2;
    cv::Mat img_masked(img.size(), img.type());
    cv::Mat mask(img.size(), img.type()); //设置mask
    for (int i = 0; i < FPS; ++i)
    {
        cv::Mat img_tmp;
        int whiteWidth = i * 2;           //白条纹
        mask.setTo(cv::Scalar(1.0, 1.0, 1.0));
        for (int j = 0; j < mask.rows / interval; ++j)
        {
            img_tmp = mask.rowRange(j * interval + whiteWidth, (j + 1) *
interval);
            img_tmp.setTo(cv::Scalar(0.0, 0.0, 0.0));
        }
        if ((int)(mask.rows / interval) * interval + whiteWidth < mask.rows)
        {
            img_tmp = mask.rowRange((int)(mask.rows / interval) * interval +
whiteWidth, mask.rows);
            img_tmp.setTo(cv::Scalar(0.0, 0.0, 0.0));
        }

        cv::multiply(img, mask, img_masked);
        writer<<img_masked;
    }
}

```

3. 高斯模糊

直接使用GaussianBlur函数即可。动态改变模糊度

```
void transition_effect3(cv::VideoWriter& writer, const cv::Mat& img)
{
    cv::Mat img.blur;
    for (int i = FPS; i > 0; --i)
    {
        cv::GaussianBlur(img, img.blur, cv::Size(2 * i - 1, 2 * i - 1), 0);
        writer<<img.blur;
    }
}
```

#### 4. 擦除(水平翻页)

从右到左拉有黑白渐变的画布。实现中设置两个mask，一个用于原始图像，另一个用于画布。条带的颜色随空间而变化，条带的宽度随时间而变化。代码相对较长，节省篇幅，不展示。

#### 5. 二值化

首先绘制图像的轮廓，然后通过更改阈值对图像着色。与一般的二值化不同，这里使用的二值化直接处理彩色图像，显示了着色过程。

```
void transition_effect5(cv::VideoWriter& writer, const cv::Mat& img)
{
    cv::Mat img_threshold(img.size(), img.type());
    for (int i = 0; i < FPS; ++i)
    {
        threshold(img, img_threshold, i * 255 / FPS, 255, cv::THRESH_BINARY);
        writer<<img_threshold;
    }
}
```

- 获取图片getPictures()

根据路径遍历即可，关键代码入下。

```
for(int i = 1; i < PIC_NUM; i++)
{
    char buffer[50];
    sprintf(buffer, "%s%d.jpg", path.c_str(), i);
    cout << buffer << endl;
    Mat temp = imread(buffer);
    pictures.push_back(temp);
}
```

- 写开场视频与写图片，writeBeginVideo(), writePictures()

基本思想一致，读取，while循环，写入。

```

while (true)
{
    capture.read(img);
    if (img.empty())
    {
        break;
    }
    cv::resize(img, imgResized, cv::Size(WIDTH, HEIGHT));

    zjuIcon.copyTo(imgResized(roi_rect));
    writer.write(imgResized);
}

```

- 缓慢绘图特效

**基本思想(key idea):** 利用画点，每次画点后增加一帧的方法实现缓慢绘制。通过画点实现基本的图形形状的绘制，而后使用基本的图形形状完成儿童画的绘制。

主要的基本几何图形绘制算法

1. 缓慢绘制直线drawLine()

使用DDA算法(计算机图形学中一种基于直线的微分方程来生成直线的方法)

其主要思想是由直线公式 $y = kx + b$ 推导出来的。

我们已知直线段两个端点P0(x0,y0)和P1(x1,y1)，就能求出 k 和 b。

在k, b均求出的条件下，只要知道一个x值，我们就能计算出一个y值。如果x的步进为1（x每次加1，即 $x = x + 1$ ），那么y的步进就为k+b；同样知道一个y值也能计算出x值，此时y的步进为1，x的步进为 $(1-b)/k$ 。根据计算出的x值和y值，向下取整，得到坐标(x',y')，并在(x',y')处绘制直线段上的一点

[https://www.omegaxyz.com/2018/09/23/dda\\_cg/](https://www.omegaxyz.com/2018/09/23/dda_cg/)

2. 缓慢绘制圆弧drawArc()

利用圆的参数方程得到各个点，而后逐点描出，即步进一定角度，利用角度和半径得到对应点坐标。

核心代码

```

for(double r = start_angle; r <= end_angle; r += foot)
{
    arc.x = center.x + radius * cos(r);
    arc.y = center.y + radius * sin(r);
    if(r == start_angle)
    {
        s = arc;
    }
    if(r == end_angle)
    {
        s = arc;
    }
    circle(img, arc, thick, color, -1);
}

```

```
    writer.write(image);
}
```

### 3. 缓慢绘制椭圆弧drawEarc()

利用椭圆的参数方程得到各个点，而后逐点描出，同样，步进一定角度，利用角度和a、b得到对应点坐标。

### 4. 缓慢绘制爱心 drawStar (Mat img,Point center,int a,Scalar color,int thick)

与前面类似利用心形参数方程绘制。

核心代码如下

```
for (double i = -PI; i <= PI; i+=foot)
{
    arc.x = center.x + a*i*sin(PI*sin(i) / i);
    arc.y = center.y + a*abs(i)*cos(PI*sin(i) / i);
    circle(img, arc, thick, color, -1);
    writer.write(image);
}
```

### 5. drawBackground()

背景画布绘制，仅在儿童画绘制阶段使用。同时可以用于清除屏幕

```
void drawBackground(bool flag)
{
    rectangle(image, Point(0, 0), Point(WIDTH, WIDTH), Scalar(255, 255, 255),
-1, 8);
    if(flag == true)
    {
        writer.write(image);
    }
    rectangle(image, Point(0, 550), Point(WIDTH, WIDTH), Scalar(159, 200, 0),
-1, 8);
}
```

## ● 儿童画

### 1. 鱼的绘制drawFish()

基本算法：采用肉眼观测加适度计算的方法，找准对应的位置与角度，利用基本的工具绘制出最简单的鱼。

核心代码

```
Point bubbles[] = {Point(90, 140), Point(90, 155), Point(90, 170)};
//鱼身子
drawEarc(image, Point(200, 150), 0, (1 - 0.082) * PI, 100, 50, Scalar(0, 0,
0), 1, true);
```

```

drawLine(image, Point(105, 140), Point(115, 150), Scalar(0, 0, 0), 1);
drawLine(image, Point(115, 150), Point(105, 160), Scalar(0, 0, 0), 1);
drawEarc(image, Point(200, 150), (1 + 0.082) * PI, 2 * PI, 100, 50,
Scalar(0, 0, 0), 1, true);
//鱼眼睛
circle(image, eye, 5, Scalar(0, 0, 0), -1);
//鱼脸部分割
drawEarc(image, Point(125, 150), 0, 0.33 * PI, 35, 50, Scalar(0, 0, 0), 1,
true);
drawEarc(image, Point(125, 150), 1.67 * PI, 2 * PI, 35, 50, Scalar(0, 0,
0), 1, true);
//鱼尾巴
drawEarc(image, Point(340, 150), 1.17 * PI, 1.5 * PI, 50, 25, Scalar(0, 0,
0), 1, true);
drawEarc(image, Point(340, 150), PI, 1.5 * PI, 8, 25, Scalar(0, 0, 0), 1,
true);
drawEarc(image, Point(340, 150), 0.5 * PI, PI, 8, 25, Scalar(0, 0, 0), 1,
true);
drawEarc(image, Point(340, 150), 0.5 * PI, 0.83 * PI, 50, 25, Scalar(0, 0,
0), 1, true);

for(int i = 0; i < 3; i++)
{
    drawEarc(image, bubbles[i], 0, 2 * PI, 5, 5, Scalar(0, 0, 0), 1, true);
}

```

## 2. 熊掌的绘制drawBearHand()

基本算法：采用肉眼观测加适度计算的方法，找准对应的位置与角度，利用基本的工具以及opencv自带的circle函数绘制出最简单的熊掌。

核心代码

```

drawEarc(image, Point(-170, 180), 0, 2 * PI, 10, 20, Scalar(0,0,0), 1, true);
drawEarc(image, Point(-150, 220), 0, 2 * PI, 10, 20, Scalar(0,0,0), 1,
true); //都是熊爪子对应的椭圆形
drawEarc(image, Point(-120, 220), 0, 2 * PI, 10, 20, Scalar(0,0,0), 1,
true);
drawEarc(image, Point(-100, 180), 0, 2 * PI, 10, 20, Scalar(0,0,0), 1,
true);
Point center(-135, 170);
toCenter(center);
circle(image, center, 12, Scalar(0,0,0), 24);

```

## 3. 笑脸的绘制drawWierdFace()

算法思想与前面类似。

核心代码

```

drawArc(image, Point(-210, 130), 100, 0, 2 * PI, Scalar(0, 0, 0), 1);
//眼眶
drawArc(image, Point(-170, 170), 20, 0, 2 * PI, Scalar(0, 0, 0), 1);
drawArc(image, Point(-250, 170), 20, 0, 2 * PI, Scalar(0, 0, 0), 1);
//眼球
circle(image, left_eye, 6, Scalar(0, 0, 0), -1);
writer.write(image);
circle(image, right_eye, 6, Scalar(0, 0, 0), -1);
writer.write(image);
//鼻子
drawEarc(image, Point(-210, 85), 0, PI, 60, 30, Scalar(0, 0, 0), 1, true);

```

#### 4. 爱脸的绘制drawLovingFace()

核心代码

```

drawArc(image, Point(210, 130), 100, 0, 2 * PI, Scalar(0, 0, 0), 1);
//眼眶爱心
drawStar(image, Point(255, 180), 15, Scalar(0,0,0), 1);
drawStar(image, Point(165, 180), 15, Scalar(0,0,0), 1);
//笑 嘴巴张大
drawEarc(image, Point(210, 110), 0, PI, 50, 50, Scalar(0,0,0), 1, true);
drawEarc(image, Point(210, 60), 1.17 * PI, 1.83 * PI, 40, 40,
Scalar(0,0,0), 1, true);
drawLine(image, Point(210, 105), Point(260,125), Scalar(0,0,0), 1);
drawLine(image, Point(210, 105), Point(160,125), Scalar(0,0,0), 1);

```

#### 5. 文本的显示drawText()、aphorism()、aphorism2()

算法思想：使用opencv自带的putText函数，结合清屏函数与for循环来实现文字的滑动与显示，

核心代码

```

for(int j = 0; j <= count; j++)
{
    temp.copyTo(image);
    putText(image, other, Point(j*(WIDTH/8)/count, HEIGHT/2),
FONT_HERSHEY_COMPLEX, 1.5, Scalar(0, 255, 0));
    writer.write(image);
    if(j == count)
    {
        //延迟一段时间
        for (int i = 0; i < 200; i++)
        {
            writer.write(image);
        }
    }
    drawBackground(false);
}

```

- 片尾动画

1. 片尾转场 opencv logo显示, logo()

opencv logo由三个不完整的环组成。要绘制圆环，首先通过调用ellipse()绘制一个扇形，然后绘制一个黑色的圆柱体以挖出中间部分。动画效果是通过更改角度来实现的。

对应代码

```
void logo()
{
    // draw the red part
    for (int i = 0; i <= FPS; ++i)
    {
        cv::ellipse(image, cv::Point(WIDTH / 2, HEIGHT / 2 - 160),
cv::Size(130, 130), 125, 0, i * 290 / FPS, CV_RGB(255, 0, 0), -1);
        cv::circle(image, cv::Point(WIDTH / 2, HEIGHT / 2 - 160), 60, CV_RGB(0,
0, 0), -1);
        writer<<image;
    }
    // draw the green part
    for (int i = 0; i <= FPS; ++i)
    {
        cv::ellipse(image, cv::Point(WIDTH / 2 - 150, HEIGHT / 2 + 80),
cv::Size(130, 130), 16, 0, i * 290 / FPS, CV_RGB(0, 255, 0), -1);
        cv::circle(image, cv::Point(WIDTH / 2 - 150, HEIGHT / 2 + 80), 60,
CV_RGB(0, 0, 0), -1);
        writer<<image;
    }
    // draw the blue part
    for (int i = 0; i <= FPS; ++i)
    {
        cv::ellipse(image, cv::Point(WIDTH / 2 + 150, HEIGHT / 2 + 80),
cv::Size(130, 130), 300, 0, i * 290 / FPS, CV_RGB(0, 0, 255), -1);
        cv::circle(image, cv::Point(WIDTH / 2 + 150, HEIGHT / 2 + 80), 60,
CV_RGB(0, 0, 0), -1);
        writer<<image;
    }
    for (int i = 0; i < FPS / 3; ++i)
    {
        writer<<image;
    }
}
```

2. 片尾动画

基本思想，采用随机化的方法，绘制大量有颜色的直线，箭头，矩形、圆形、各种Marker('+'，'三角形'等)、多边形、有填充的多边形、个人信息。最后将"End of this video"输出。渐变减弱的方式通过for循环，将原图片减掉逐渐增加的等值矩阵的值。

核心代码

随机化

```
for (i = 0; i < NUMBER; i++)
{
    Point center;
    center.x = rng.uniform(x1, x2);
    center.y = rng.uniform(y1, y2);
    circle(image, center, rng.uniform(0, 300), randomColor(rng),
           rng.uniform(-1, 9), lineType);
    writer.write(image);
}
```

渐变与输出

```
for( i = 0; i < 255; i += 2 )
{
    image2 = image - Scalar::all(i); //实现渐变
    putText(image2, "End of this video", org, FONT_HERSHEY_COMPLEX, 3,
            Scalar(i, i, 255), 5, lineType);
    writer.write(image2);
}
```

- 视频播放play()

读取对应文件夹下的视频，而后在namedWindow上播放即可，while{}语句（前面已经有简单叙述）。

通过设置waitKey(32)，等待键盘空格，循环写入，实现暂停。较为简单。

## 编程体会

通过这个实验，我更加熟悉了OpenCV库的使用，并且掌握了图像处理的基本技能，主要是OpenCV中几个常用类的构造和使用，例如Mat，VideoCapture，VideoWriter等。以及一些常用的函数，如putText，ellipse，rectangle，circle等。同时也理解了mask的使用。自己也学习了缓慢绘制图像的算法，并花费了大量的时间绘制儿童画，锻炼了自己的耐心和细心。

最后还是要感谢助教的耐心指导和帮助！

数码大头照

