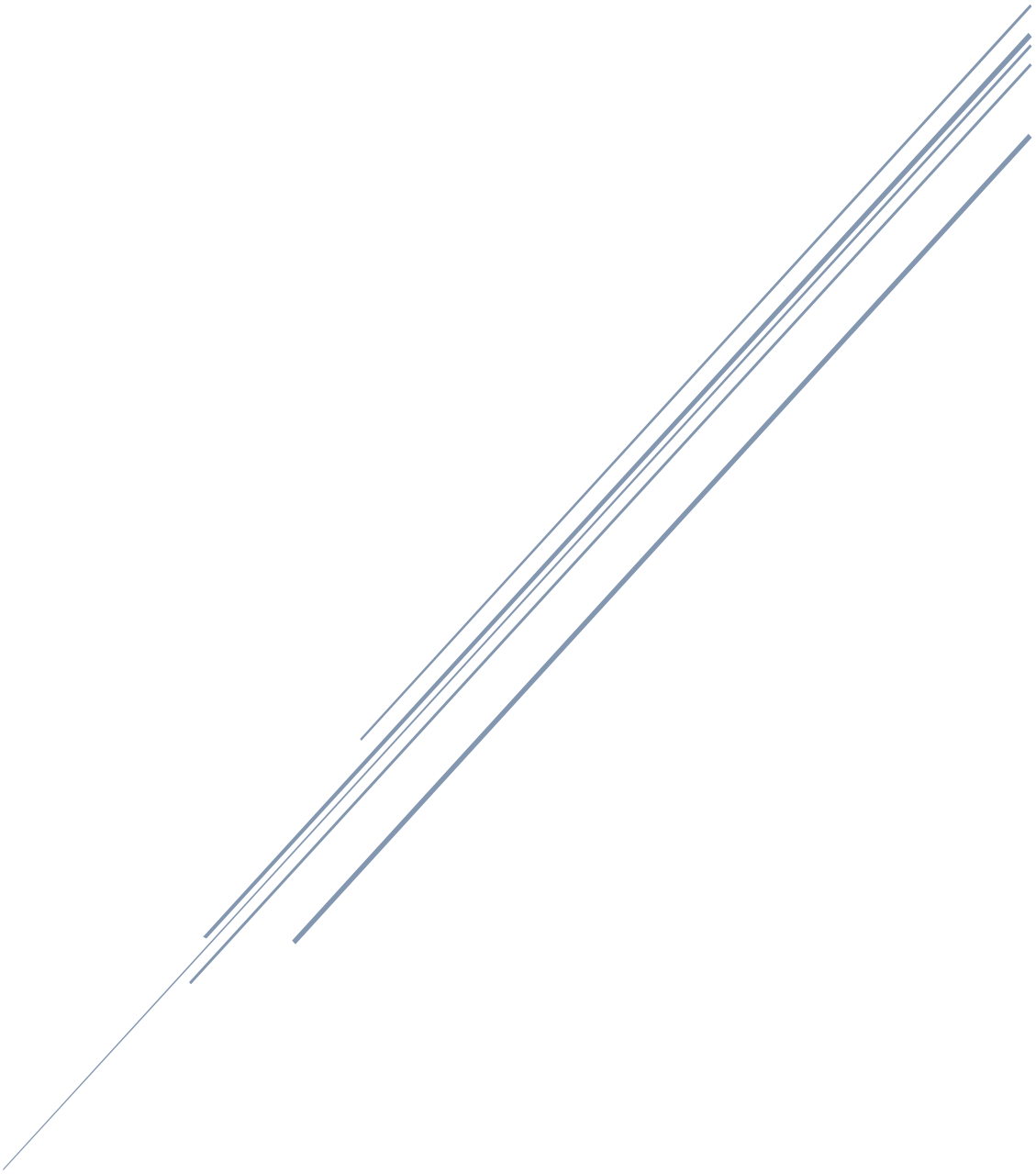
表

高程荣誉班·章节作业之柒



同济大学

高级语言程序设计

目录

陈宇飞

题目纯属玩，

作业内容

写一个尽量好看的表。
使用工具：VS2019 C++ easyX

得分点

能吸引到老师或 TA 的点。~~TA 说了算~~
1 没有锯齿
2 独特的设计
3 其它图像处理算法

扣分点

1 使用类(class)等后续知识点；
2 不使用 struct
3 基本做的跟作业题上附的 demo 差不多。

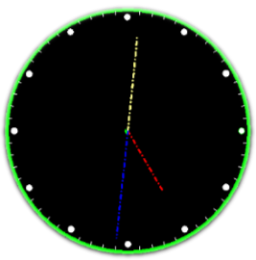
设计思路和算法分析

在埋头写代码前，拿出纸笔或者数位板，让灵感飞一会儿。
在写之前就设计好自己的钟表的大致效果，减少大面积修改和时间浪费。

表针设计

百度“钟表设计”以获取别人的灵感 wwww 甚至可以找一些（不一定是钟表）的艺术设计网站。

简约

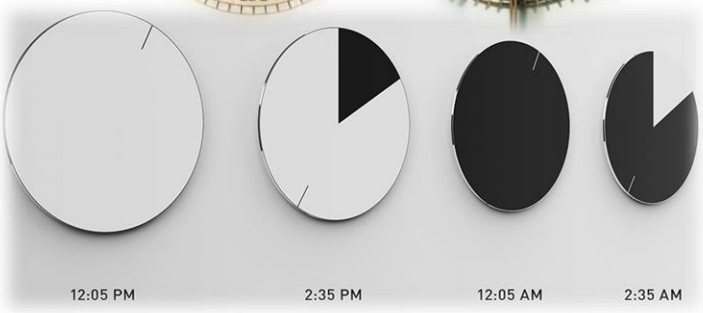


因为实现钟表的正常转动是最基础的要求，因此不揭示实现方法了。
XOR 渲染：setwritemode(R2_XORPEN);
调用一次画表针，再完全相同调用一次擦除表针。
缺点：非黑色背景难调整颜色
由此，一个比被辞退的员工的表好看一点，且有一定特色的表已经写完了。

华丽


想办法利用直线+自带函数构建右侧表的图形。

异形

anomaly .mp4

表盘设计

普通表盘

生活中，大部分时候我们见到的种还是“基础款”的。在此基础上，我们着重讲一个颜色的配搭。

嗟来之食：

https://colorhunt.co/

空格摇摇乐 https://coolors.co/

上传一张图片，还你一套配色（配合图片式背景食用）：

https://www.canva.com/colors/color-palette-generator/

http://www.colorfavs.com/

http://colormind.io/

输入你想分析的网站，返回该网站的配色分析：

http://webcolourdata.com/

好看的渐变色（小思考：怎么画出渐变色？）：

https://webgradients.com/

https://uigradients.com/#Venice

https://www.grabient.com/

异形表盘

有没有同学心里想画一个漂亮的表盘，但是找不到相关图形的函数，最后妥协了？

Task.A 怎么画一个正三角形的表盘？

三条直线

给直线中围着的的表盘上色可能比较麻烦。

Task.B 怎么画一个心形表盘？

两个圆弧+两条直线

Task.C 怎么画一个手表？

接下来立刻讲解手写直线和曲线。

另外，WWS 将额外展示一些漂亮的高级图形的画法，这里先卖个关子。

手写直线生成

走上架空 easyX 的不归路。

DDA 算法

Q：如何从点(0,0)开始，生成直线 y=3x，直到(100,300)结束？

我们将这两个坐标相减，发现：这条直线跨越了 300 个行像素点，100 个列像素点。因此，我们选择以 y 为步长，一行一行的渲染。

Q：为什么选 y？何时选 x？

//打开带方格的画图讲解

函数的普适画法

列出参数方程。在参数范围内设置精度，以一定的步长打点，然后将点连成直线。好比在圆周上均匀的取一些点，然后把他们连起来。我们知道，取点越多，最终连起来的曲线越均匀，而代价是随着取点变多，对机能的要求也会增大。

Q：我们应该取多少个点最为合适？A：看上去足够光滑即可。

Q：难列参数方程的函数怎么办？A：模仿上述 DDA 算法，以像素点为步长，按行/列渲染图像。

E：写一个玫瑰曲线。$\rho = a \cdot \sin\left(\frac{n}{d}\theta\right)$

首先改写为参数方程：

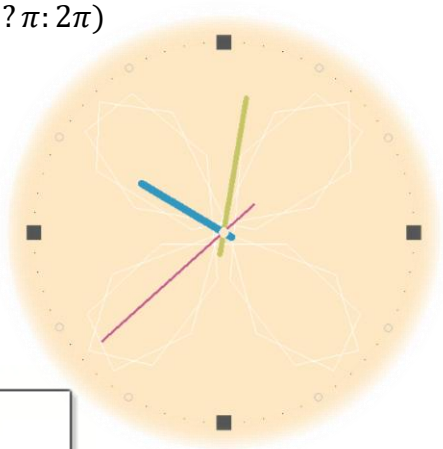
$$\begin{cases} x = asin(n\theta\pi)\cos(d\theta\pi) \\ y = asin(n\theta\pi)\sin(d\theta\pi) \end{cases} (0 \le \theta \le n \cdot d \,\%\, 2?\pi:2\pi)$$

让θ每次增加 0.001，并相应计算出 x、y 的值

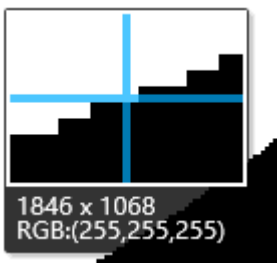
//演示精度过低（取点太少）的结果

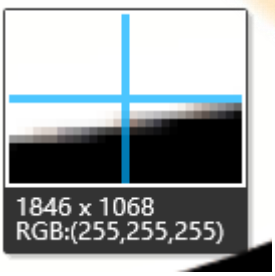
让 n 和 d 的取值分别和分针、秒针相关联，

得到随时间变化的玫瑰曲线。



抗锯齿





出现锯齿出现的原因就是屏幕的分辨率不够。如左图右下角，白色和黑色的交界处是一段圆弧，方框中是这段圆弧的放大。可以明显的看出，每一个像素点非黑即白；一旦放大，就会显现出所谓“锯齿状”，经不起细品。于是呢，我们把他的边缘做一个渐变。如右图，边缘从白色慢慢地过渡到黑色，

从而整个画面看起来更平滑。这就是所谓的“消除锯齿”，以此达到假装更高分辨率的效果

不难发现，抗锯齿必然需要额外的运算，所以就有五花八门的抗锯齿算法，目的都是以尽量小的性能损耗来尽量大的优化建模边缘的视觉感受。他们的效果随场景不同，各有千秋。

搜索小技巧：搜索“直线抗锯齿算法”，而非“直线 抗锯齿”。后者很难找到相应算法讲解。当然，可以尝试挑战自己开发抗锯齿。

除了直线抗锯齿外，以下介绍一些适用性更高的抗锯齿算法。

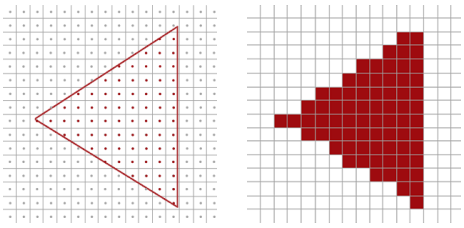
SSAA 算法（朴素超级采样抗锯齿）

SDF 算法

基本非直线图形（比如椭圆）的抗锯齿

Super Sampling Anti Aliasing (SSAA)

- 考虑这样一个图形的抗锯齿，首先我们考虑它最基本画出来的形态（右图）
- 由于屏幕像素总量的限制，有些边缘的像素能够被渲染出来，而有些则不会，结果就是我们使用了不光滑的边缘来渲染图元，导致之前讨论到的锯齿边缘



双击以查看完整 ppt。

环境设计

主体部分已经完成，我们还能如何“放飞自我”呢？

背景设计

Q：上文提到的渐变背景色如何实现？

A：渲染直线的同时改变像素颜色，由此设计出函数 f(x,y)=RGB

图片：相对路径插入图片和提交注意事项

Q：如何写一个简单的粒子效果？

A：重构 sleep()，随机生成星星的坐标和移速，按帧移动。

更漂亮的背景光点：见高级图形学算法。

Q：如何实现“阳光”的效果？

把表的一部分较亮，另一边调暗；亮暗交界限随时间而变化？

互动设计

画面对鼠标移动反馈

1950591 黄茂森.mp4

此外，可以设计如表针对鼠标点击的反馈。例如：鼠标每点击一下，就更换一次表盘的颜色。
~~比如摇摇乐点空格~~
当然，也可以考虑是按钮+对话框式的传统交互，不追求代码量，而是服务于钟表的整体效果。

视听设计

不展开。可能会在后续大作业时分享 bgm 相关(比如直接插入||用自带音效自制 bgm)

1950394 杨泽华.mp4

性能优化+动态渲染

简单的部分

Q：有的同学发现，自己的表出现了跳秒的问题。比如，秒针有时会直接从 29 秒跳过 30 秒指向 31 秒。这是为什么？

A：您的程序可能没有使用 sleep()；或者使用了 sleep(1000)，也就是每 1 秒渲染一次。考虑渲染耗时。

Q：增加完这些花里胡哨的功能后，我的表开始频繁出现图像闪现的问题，怎么办？

A：BatchDraw()。

解决以上问题后，我们就可以设计更漂亮的界面了。

我们把之前的玫瑰曲线切成 250 份，将 sleep 间隔改为 40ms，也就是每间隔 40ms 渲染曲线的其中一份。这样，我们一共需要 40ms×250=10s 来渲染出一个完整的玫瑰曲线。

这样，我们以 10s 为一周期，得到了动态的玫瑰曲线。10s 过后，再渲染下一条新曲线。

最后，我们在 10s 周期的第 7-10s 对曲线的颜色进行调整，不断的让它从白色调整为接近背景色，这样两个曲线之间就有了过渡。

Q：改成动态渲染以后，我的曲线会盖到表针上面，怎么办？

A： 暴力的办法：每一帧渲染完曲线以后重新渲染一下表针。

正常的办法：渲染曲线上的每一段前先判断会不会触碰表针。

事实上，还有一些其它细节（比如连线断点，整秒曲线被刷）。经过一番 debug 后，一个简单的钟表就写好啦！

到此，我们的云写表就告一段落了。每一位参与者可以给其他人也给自己点一个赞 2333

复杂的部分

1. easyx 的效率问题，当自行实现 SDF 抗锯齿算法后，可以考虑直接操作底层 BUFFER 而抛弃使用 easyx 的函数

2. 若没有自行实现 SDF，那么可以考虑离屏/离线渲染（预处理），将一些绘制很复杂的图元在启动时预处理好（如高斯模糊）

3. 更进阶的，自行生成维护一个 float 类型的 BUFFER（底层是 4 字节的 BUFFER），并使用 release 编译启用向量优化加速

Performance Optimization

- 前面讲的很多算法都有一定程度上的效率问题，加上本身EasyX的效率极低，所以这一板块提出一些高阶的性能优化方法
- 底层像素操作
- 离屏渲染
- 向量优化

双击以查看完整 ppt。

高级图形学算法

Advanced Graphics Algorithms

- 介绍几个经典的图形学算法，但不一定简单
- 高斯模糊
- 贝塞尔曲线
- Alpha通道叠加

双击以查看完整 ppt。

代码构建

函数的封装+注释

```
//时钟大小随画布大小X、Y自适应改变
const int X = 350;
const int Y = 375;
//背景曲线速度，如果机能较低曲线不完整请调至4
const int S = 3;
//背景星星数量和速度
const int STAR = X / 3;
const int SPEED = 3;
//以上参数均可自由调整

//以下参数不建议调整
//渲染精度
const int C = 5;
//最大帧率限制
const int FPS = 30;
```

全局函数到底香不香？

Struct 到底香不香？

Q：这份代码应该分成几大块？

代码的拓展性

尝试一下：

Level1：将画布大小设为全局常量，让你的表的中心随 X Y 改变，表盘大小和表针粗细随 X Y 自动调整，etc.

Level2：最大帧率限制。过高反而会卡死，过低会导致画面不流畅。//约 30FPS 较好

Level3：其它你觉得可以设计为允许程序员自由调整的参数。

提高可玩性的同时，也对 sleep()等功能以及代码的整体逻辑有更深的理解。

Q：对有动态渲染的表，如何防止它的移动速度被 sleep()/FPS 的修改而影响？

A：渲染速度函数和 FPS 挂钩

资料收集

百度。从设计到算法，信息常常是一门面向搜索引擎学习的学科。

多看 easyX 帮助文档，善用搜索功能。

E：按照函数分类找到 BatchDraw

Q：如何在画布上输出文字？

A：搜索框中键入 text 即可找到。进而发现，文字可以开启抗锯齿。

题外话：切换成 release（以提高生成文件的兼容性），重新编译而后在项目内找到相应 exe。分享 exe 可以减少录屏的麻烦且文件很小，缺点是只能在电脑端运行。

尾声

如果你发现，自己仿写出被辞退的员工的表已经费尽九牛二虎之力。请不要灰心，相反，这更说明你应该花更多的时间在高程的操练上。在不影响其他学科的前提下，牺牲一些娱乐时间，把这个“小作业”当成一次“小挑战”。你会发现，刚才所说的设计思路和算法，有许多只需要十几行即可实现。

祝大家逐渐发现写代码的魅力，下一个在这里分享(秃头)的就是你！

同济大学新生院 陈泓仰

同济大学艺术与传媒学院 王维斯