题目预测

填空&选择题（50 分）

1. 字符串裁剪精度：串精度，字符精度，像素精度
2. 三维模型：线框模型，表面模型，实体模型
3. 曲线（面）表示：非参数表示（显式，隐式），参数表示（代数形式，几何形式）
4. 贝塞尔曲线（1962 年设计，1972 年使用，UNISURF 设计系统）

——以逼近为基础

——应用领域：CAD，建筑物设计，移动机器人运动规划

5. 字库的两种类型：点阵型，矢量性
6. b 样条（节点数/控制定义域）：

——节点数：控制点数+阶数

——控制定义域：下标：次数 ~ 控制顶点数（有基函数支持的区间）

7. 图形学历史：第一次确立科学分支地位：1962：Ivan E. Sutherland，论文：Sketchpad

为计算机生成和显示图形提供可能：CRT（1950）

第一次使用具有指挥和控制功能的 CRT 显示器—— SAGE（50 年代末）->标志着交互式图形技术的诞生

第一个官方标准：GKS，事实标准：OpenGL（SGI）

8. 消隐算法（2 分）：

——按消隐对象分类：线消隐，面消隐

——消隐结果与观察物体，视点有关

——Z Buffer 存储：颜色值，深度值

9. 多边形扫描算法步骤：求交，排序，配对，填色

多边形的表示方法：顶点表示，点阵表示

光栅图形学的一个基本问题：把多边形的顶点表示->点阵表示，a.k.a 多边形的扫描变换

——补充：确定最佳逼近图形的像素集合，并用指定属性写像素的过程：光栅化/扫描转换

10. OpenGL 选择题（10 分左右）：

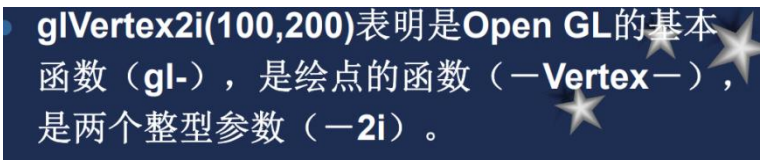
——核心库（gl），实用库（glu），工具库（glut），辅助库（aux），Windows 专用库(wgl)

——着色模式：RGBA，颜色索引

——静态链接库：.lib，动态链接库：.dll，说明文件：.h

——加速方式：硬件，三维图形，纯软件

——辅助库（aux）不能运行在所有的 OpenGL 平台上



——实用符号常量：以 GLU_为开头，均用大写字母，并用下划线与关键词分开，如 GLU_RGB。

——数据类型：Open GL 定义的数据类型以 GL 开头，例如（gl.h 文件中可见）

- glViewport()设置在屏幕上的窗口大小
- glOrtho()设置投影方式为平行投影
- gluPerspective()设置投影方式为透视投影

——OpenGL 变换：

（1）视点（视图）：**gluLookAt**

（2）模型变换：

平移变换：glTranslatef，glTranslated

旋转变换：glRotatef，glRotated

缩放变换：glScalef，glScaled

（3）投影变换：

在定义投影变换之前，必须使用如下两个函数，才能让随后的变换作用于投影矩阵而不是模型变换矩阵，避免产生复合的投影变换：

```
1 GlMatrixMode（GL_PROJECTION）；//启动投影矩阵
2 glLoadIdentity（）；          //初始化为单位阵調
```

平行投影：**glOrtho**

透视投影：**glFrustum，gluPerspective**

（4） 视口变换：**glViewPort**

11. 三角网格选择题（2 分）：简化，细分（Loop），重剖（特征敏感距离，欧氏距离—各向同性网格重剖），光顺
12. 光学知识填空（1 分）
13. 反走样方法：提高分辨率，区域采样，加权区域采样，半色调技术
14. 图形学的杂志和会议：

会议：Siggraph，Eurograph，IEEE Virtual Reality, IEEE Visualization Conference

杂志（期刊）：ACM Transaction on Graphics, IEEE Computer Graphics and Application, IEEE Visualization and Computer Graphics, IEEE T. on Visualization and Computer Graphics, Graphical Models, Computer Graphics Forum, The Visual Computer

15. HSV 颜色模型（三要素：色调，饱和度，亮度）
16. 纹理（概念）：物体表面的细小结构。（最早的工作：Catmull 74 年的博士论文）

——较好地表现模型表面的细节

17. 图形的要素（填空）：几何要素（点线面），非几何要素（色彩，线型，线宽）
18. 计算机图形学：利用计算机研究图形的表示，生成，处理和显示的学科
19. 图形设备（3 种）：输入，输出，处理

输入设备：键盘和鼠标，跟踪球和空间球，数据手套（补充：扫描仪，数字化仪）

输出设备：包括图形的显示和绘制

显示设备：彩色 CRT 显示器，球面显示器与柱面显示器，LCD, LED

——补充：LCD 技术指标（3 个）：可视角度，点距与分辨率，展望。CRT 工作原理：荧光物质在高速电子的轰击下会发生电子跃迁，高能态不稳定返回低能态发出荧光。

处理设备：显卡（核心：显示主芯片，a.k.a GPU，相当于 CPU，存储：现存，相当于内存）

——补充：显存：FPM, RAM, EDORAM, SGRAM, SDRAM, DDR SDRAM

20. 曲线光滑度度量方式：**参数连续，几何连续**
21. 在坐标变换中，**模型变换**和**视图变换**可以达到同样的效果

简答题（18 分）

激光打印机

原理：利用**电子成像**技术，激光束扫描感光鼓，将墨粉吸附到感光区域，再将墨粉转印到打印介质上，最后通过**加热装置将墨粉熔化固定到打印介质**上

明暗处理

分为**双线性光强插值，双线性法向插值**
双线性光强插值步骤：
1. 计算多边形顶点的平均法向量
2. 用 Phong 光照明模型计算顶点的平均光强
3. 插值计算离散边上的各点光强
4. 插值计算多边形内域中各点的光强

双线性法向插值步骤：
1. 求顶点的平均法向
2. 线性插值求线上的法向
3. 线性插值求面上点的法向
4. 根据各点的法向，用 Phong 简单光照明模型计算各点的光强

多边形裁剪

裁剪：分解成边界的线段逐段裁剪
四种情况输出：
1. S, P 均在可见一侧：输出 P
2. S, P 均在不可见一侧：无输出
3. S 在可见一侧，P 在不可见一侧：输出 SP 的交点 I
4. S 不可见，P 可见：输出 SP 交点 I 和 P

不能裁剪凹多边形，会出现错误的连线，解决办法：分割成凸多边形再裁剪

计算题（32 分）

中点画线

基本思想：假如当前点(x, y)，下一点的选择：(x + 1, y) or (x + 1, y + 1)，选择方法：判断理想直线与横坐标 x + 1 处的交点离谁更近就取谁

考虑直线：F(x, y) = ax + by + c->$\begin{cases} = 0， & 线上 \\ > 0， & 上方 \\ < 0， & 下方 \end{cases}$，将(x + 1, y + 0.5)代入 F(x, y)进行判断：

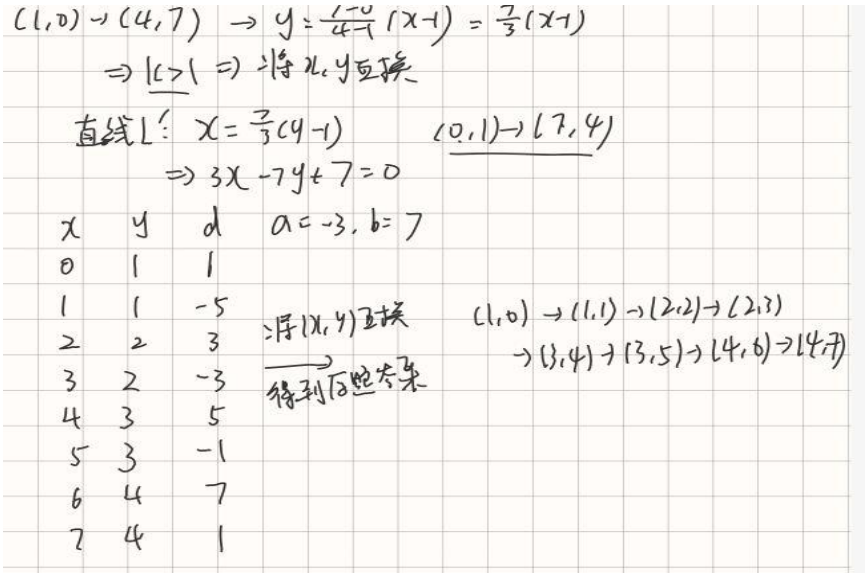
令 d = F(x + 1, y + 0.5)，初始值 d = F(x , y) + a + 0.5b，为摆脱浮点数计算，我们将 d->2d，此时：初始值 d = 2a + b，判断如下：

A. d >= 0，取下一个点为(x + 1, y)，判断下一像素位置(x + 2, y + 0.5)， d += 2a

B. d < 0，取下一个点为(x + 1, y + 1)，判断下一像素位置(x + 2, y + 1.5)， d += 2a + b，代码：

```
void MidpointLine(int x0, int y0, int x1, int y1, int color) {
    int a, b, d1, d2, d, x, y;
    a = y0 - y1, b = x1 - x0, d = 2 * a + b; // 直线其实是由(x0, y0), (x1, y1)两点确定的，点斜式->一般式
    d1 = 2 * a, d2 = 2 * (a + b);
    x = x0, y = y0; // 初始值
    drawpixel(x, y, color);
    while (x < x1) {
        if (d < 0) {
            // 中点在 Q 下方，取(x + 1, y + 1)
            x += 1, y += 1, d += d2;
        }
        else {
            x += 1, d += d1;
        }
        drawpixel(x, y, color);
    }
}
```

习题 2.2：



OpenGL 程序

基本上是画一个正方形：

1. 根据显示器分辨率（一般：800*800）画一个大框：左上角(0, 0)，右下角(800, 800)
2. 根据 gluinitwindowPosition()函数确定窗口左上角位置，根据 gluinitWindowSize()函数确定窗口大小，画一个小框，右上角标上(1, 1)，左下角标上(-1, -1)
3. 颜色标注

小框：

——glClearColor(0.0f, 0.0f, 0.0f, 1.0f)，也可以：glClearColor(GL_COLOR_BUFFER_BIT)

——glClearColor(1.0f, 1.0f, 1.0f, 1.0f)

——glClearColor(1.0f, 0.0f, 0.0f, 1.0f)

——glClearColor(0.0f, 1.0f, 0.0f, 1.0f)

——glClearColor(0.0f, 0.0f, 1.0f, 1.0f)

像素：

——glColor3f(0.0, 0.0, 0.0)

——glColor3f(1.0, 1.0, 1.0)

——glColor3f(1.0, 0.0, 0.0)

——glColor3f(0.0, 1.0, 0.0)

——glColor3f(0.0, 0.0, 1.0)

4. 确定点：

——坐标（简单）

——确定形状：glBegin(GL_POINTS)->描点即可，其他的 TRIANGLE，SQUARE 就是按具体要求画图形即可

猜测代码：

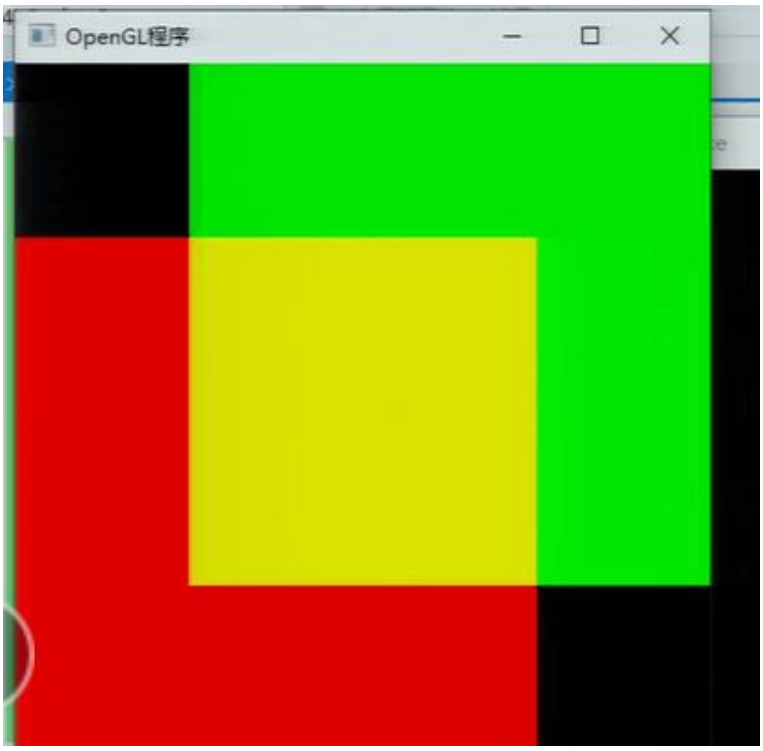
```
void myDisplay(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glEnable(GL_BLEND);
    //glBlendFunc(GL_ONE, GL_ZERO);
    //glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glBlendFunc(GL_ONE, GL_ONE);
    glColor4f(1, 0, 0, 0.5);
    glRectf(-1, -1, 0.5, 0.5
    glColor4f(0, 1, 0, 0.5);
    glRectf(-0.5, -0.5, 1, 1);


    glFlush();
    glutSwapBuffers();

}
```

#define GL_ONE 1
扩展到: 1
联机搜索

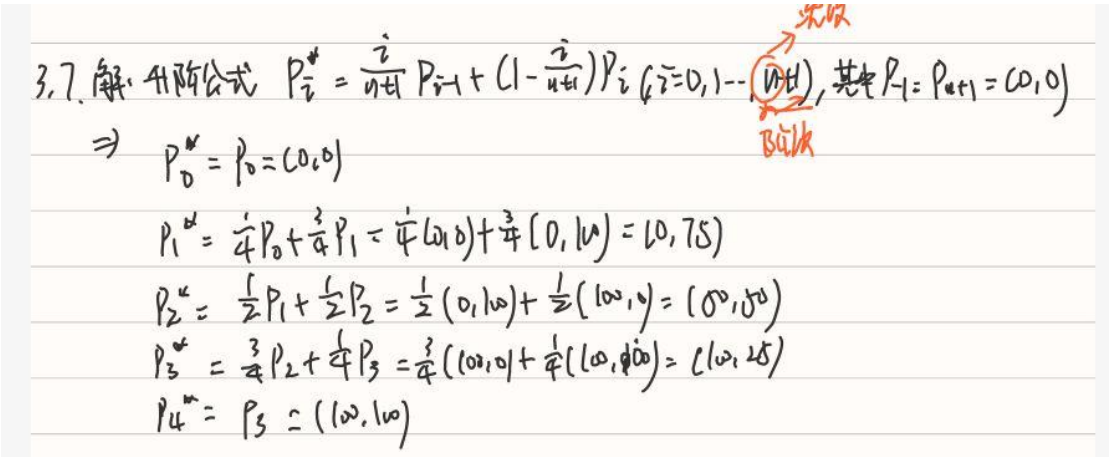


若 glBlendFunc(GL_ONE, GL_ZERO)，无黄色，绿在红上面
若 glBlendFunc(GL_ZERO, GL_ONE)，无黄色，红在绿上面

曲面递推

贝塞尔曲线（升降阶/控制曲线）

1. 升阶：



2. 降阶

$P_0(0,0)$, $P_1(0,75)$, $P_2(50,50)$, $P_3(100,25)$, $P_4(100,100)$ 降阶

解：升阶：$P_i^* = \frac{i}{n+1}P_{i-1} + \left(1-\frac{i}{n+1}\right)P_i$, $i=0,1,2,3,4$ → $n+1$

$P_0^* = (0,0) = P_0$

$P_1^* = (0,75) = \frac{1}{4}P_0 + \frac{3}{4}P_1$

$P_2^* = (50,50) = \frac{1}{2}P_1 + \frac{1}{2}P_2$ ↑ $P_1 = (0,100)$

$P_3^* = (100,25) = \frac{3}{4}P_2 + \frac{1}{4}P_3$ ⇒ $P_2 = (100,0)$

$P_4^* = (100,100) = P_3$

⇒ 降阶结果为 $P_0(0,0)$, $P_1(0,100)$, $P_2(100,0)$, $P_3(100,100)$