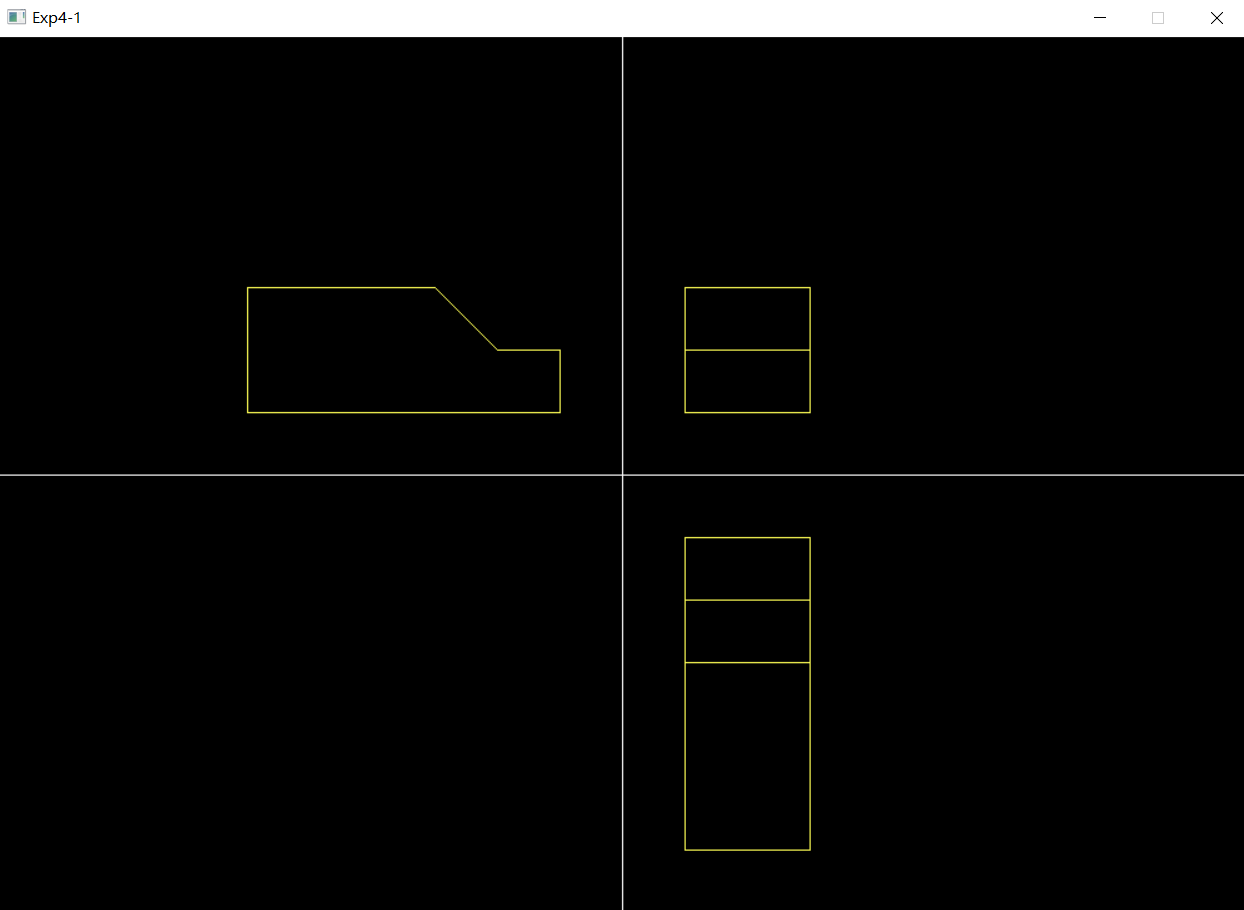
**模块4-1 三维图形投影和消隐**

**一 实验目的**

1. 编写三维图形各种变换的投影算法

**二 实验内容**

1：自行选择三维物体（不能选长方体），建立坐标系，给定点的三维坐标值，建立边表结构。完成三视图。

实验结果如下图所示：

左上显示为主视图，右上显示为侧视图，右下显示为俯视图。

**三 程序说明**

最终的实验代码如下表所示：

|  |
| --- |
| 1题 |
| //////////////////////////////////////////////////////  // 程序名称：实验4-1  // 功 能：实现预设图像的三视图显示  // 编译环境：VS2019，EasyX\_20220116  // 作 者：夏婉可<2020301010225><1597493790@qq.com>  // 最后修改：2022-4-8  #include <graphics.h>  #include <conio.h>  #include <iostream>  #include <math.h>  using namespace std;  //和分割线之间的距离  const int gap = 50;  //预设三维体的点坐标  int X[33] = { 0,0,0,0,0,0,0,0,100,100,0,100,100,100,100,100,100,100,100,0,0,0,0,100,100,100,100,0,0,0,0,100,100 };  int Y[33] = { 0,0,250,250,200,150,0,0,0,0,0,0,150,200,250,250,0,0,150,150,0,150,200,200,150,200,250,250,200,250,250,250,250 };  int Z[33]= { 100,0,0,50,50,100,100,0,0,100,100,100,100,50,50,0,0,100,100,100,100,100,50,50,100,50,50,50,50,50,0,0,50 };  //顶点总数  const int num = 33;  //三视图实现函数  void fun() {  //正视图  POINT\* front = new POINT[num];  for (int i = 0; i < num; i++) {  front[i].x = X[i] + 500 + gap;  front[i].y = 350 - Z[i] - gap;  }  //侧视图  POINT\* side = new POINT[num];  for (int i = 0; i < num; i++) {  side[i].x = Y[i] + 250 - gap;  side[i].y = 350 - Z[i] - gap;  }  //俯视图  POINT\* up = new POINT[num];  for (int i = 0; i < num; i++) {  up[i].x = X[i] + 500 + gap;  up[i].y = 600 - Y[i] + gap;  }  setcolor(YELLOW);  for (int i = 0; i < num - 1; i++) {  line(front[i].x, front[i].y, front[i + 1].x, front[i + 1].y);  line(side[i].x, side[i].y, side[i + 1].x, side[i + 1].y);  line(up[i].x, up[i].y, up[i + 1].x, up[i + 1].y);  }  free(front);  free(side);  free(up);  }  int main() {  initgraph(1000, 700);  line(0, 350, 1000, 350);  line(500, 0, 500, 700);  fun();  \_getch();  closegraph();  return 0;  } |