**模块3-1 图形变换**

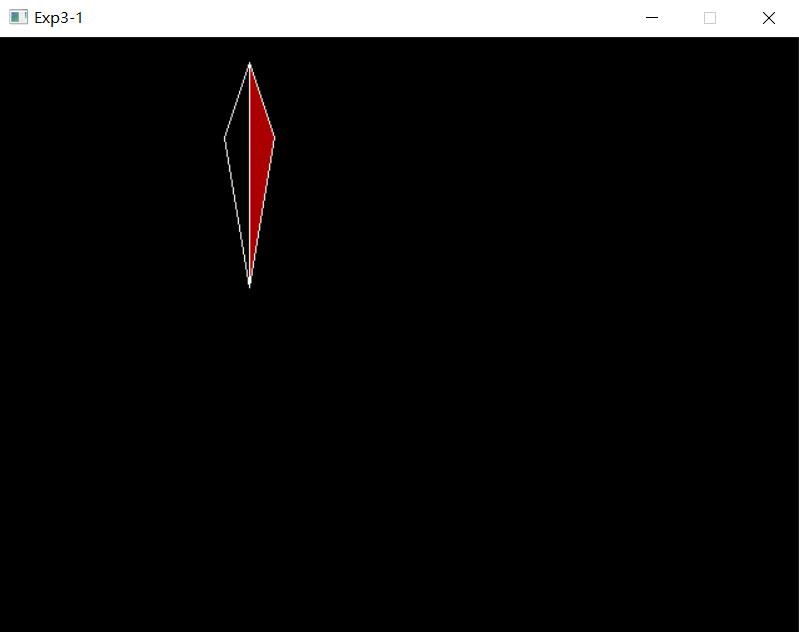
**一 实验目的**

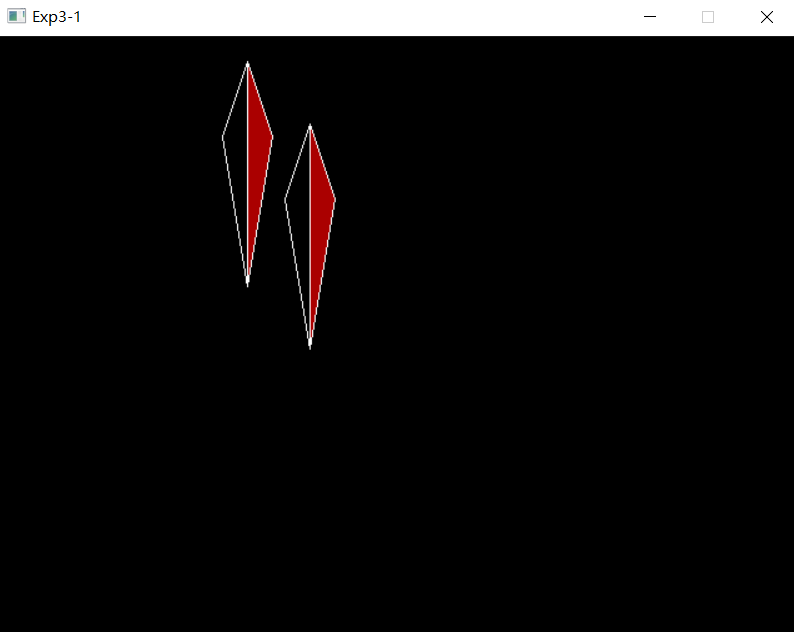
1. 编写图形各种变换的算法

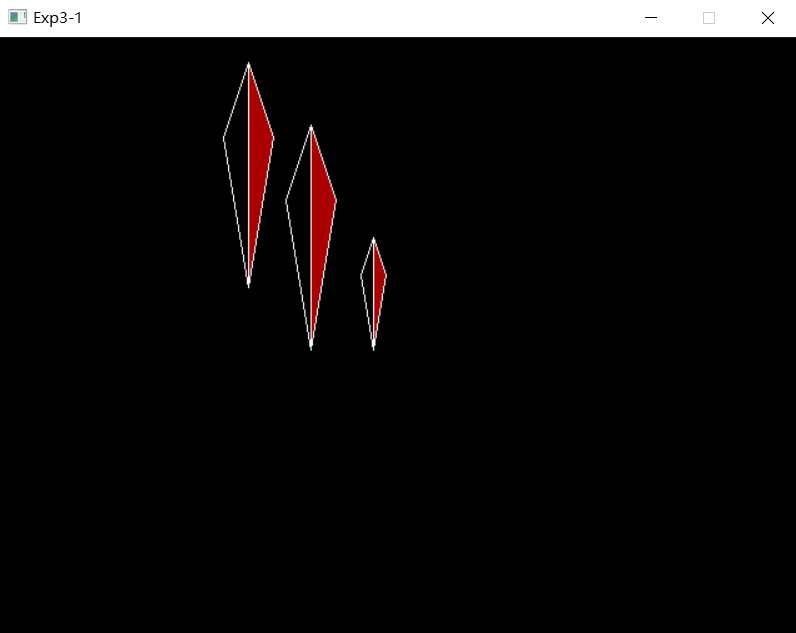
**二 实验内容**

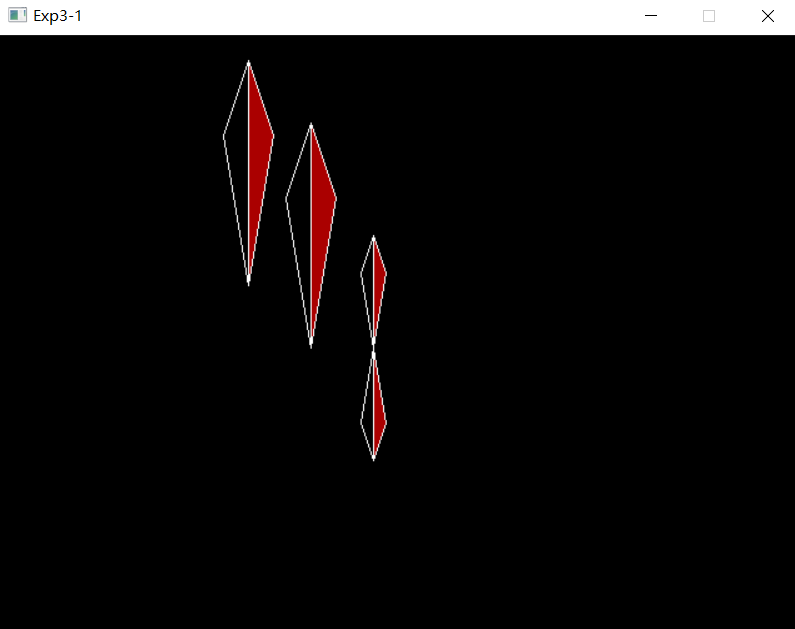
1：自行设计基本图案，完成1-5种简单变换

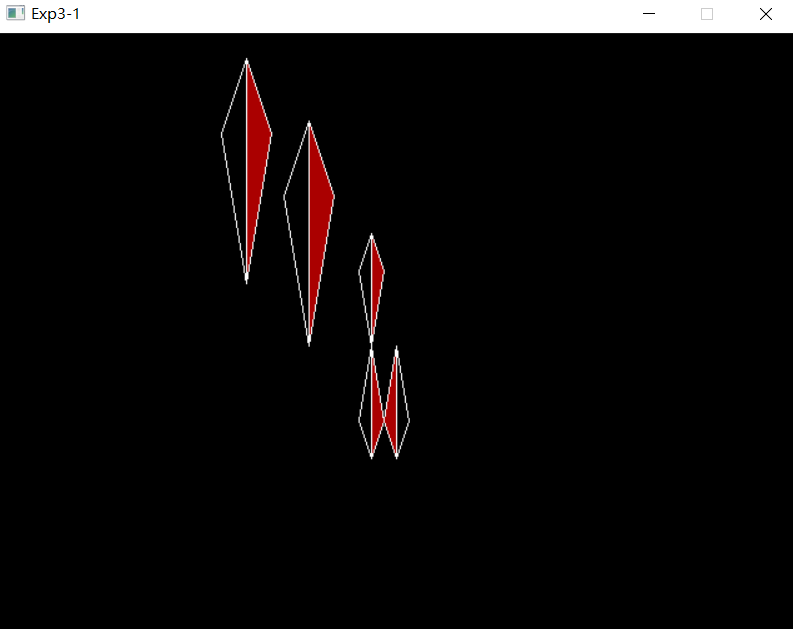
实验结果如下图所示：

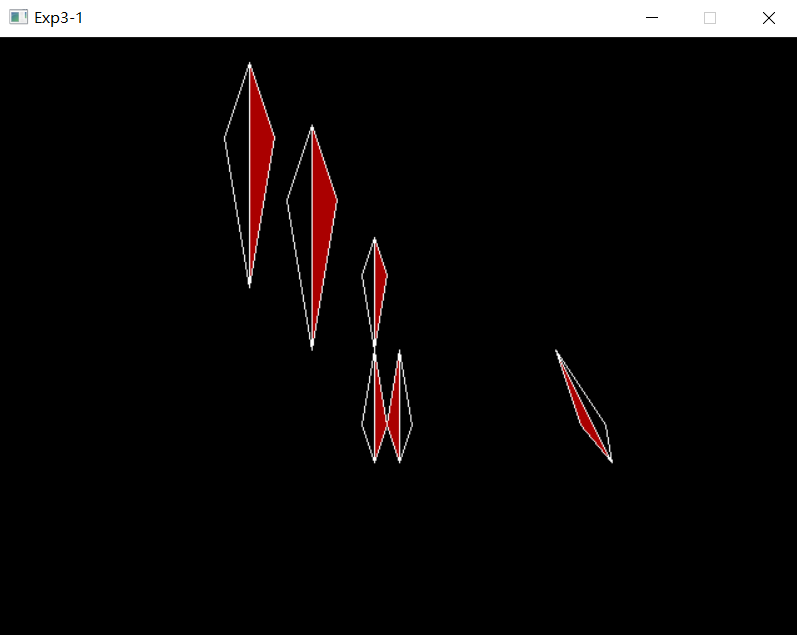
图形初始化：

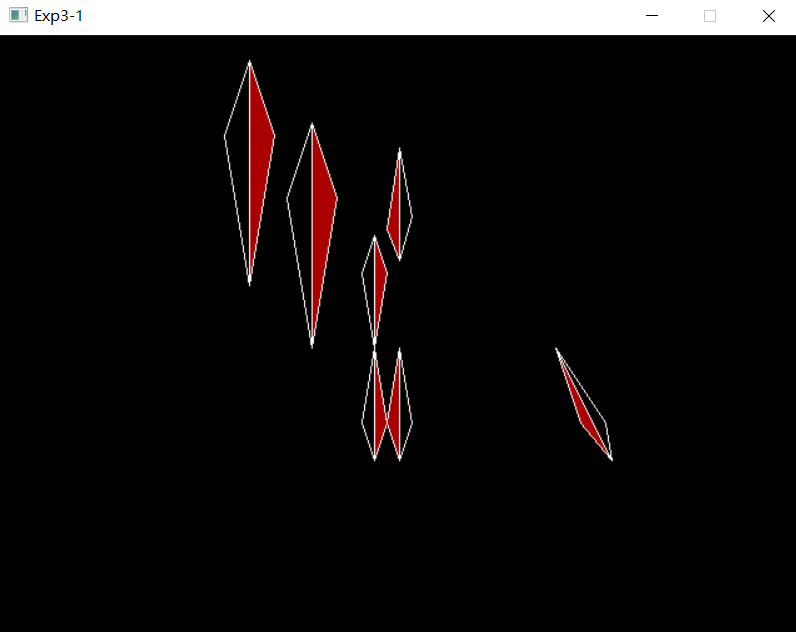
第一次点击左键，实现平移变换：

第二次点击左键，实现比例变换（同时伴有平移变换）：

第三次点击左键，实现对称变换（以平行y轴方向的直线为对称轴）：

第四次点击左键，实现对称变换（以平行x轴方向的直线为对称轴）：

第五次点击左键，实现错切变换（沿x轴方向关于y错切）：

第六次点击左键，实现错切变换（沿y轴方向关于x错切）：

2：在实验题3-1的基础上实现多步复合变换，设计动画效果

实验结果如下图所示：

初始化界面：

按任意键后出现动画效果：

**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3-1  // 功 能：实现预设图像的平移变换、比例变换等变换  // 编译环境：VS2019，EasyX\_20220116  // 作 者：夏婉可<2020301010225><1597493790@qq.com>  // 最后修改：2022-4-5  #include <graphics.h>  #include <conio.h>  #include <iostream>  #include <math.h>  using namespace std;  #define pi 3.1415926535  int main() {  POINT t1[] = { {200,200} , {200,20} , {220,80} };  POINT t2[] = { {200,200} , {200,20} , {180,80} };  int len = 3;  float Tx = 50, Ty = 50;//平移  float Sx = 0.5, Sy = 0.5;//比例  float angle = 45 \* pi / 180;//旋转，没做出来QwQ  float C = 0.5, B = -0.5;//错切  //initialize graph  initgraph(640, 480);  //勾画初始图案  setfillcolor(RED);  fillpolygon(t1, 3);  polygon(t2, 3);  //record the times of changing  int times = 0;  ExMessage m;  //全是单步变换。  while (1) {  m = getmessage(EX\_MOUSE | EX\_KEY);  float cur1x[3], cur1y[3], cur2x[3], cur2y[3];  //平移  if (m.message == WM\_LBUTTONDOWN && times == 0) {  for (int i = 0; i < len; i++) {  t1[i].x += Tx;  t1[i].y += Ty;  t2[i].x += Tx;  t2[i].y += Ty;  }  setfillcolor(RED);  fillpolygon(t1, 3);  polygon(t2, 3);  times++;  }  //比例  else if (m.message == WM\_LBUTTONDOWN && times == 1) {  for (int i = 0; i < len; i++) {  t1[i].x += 50;  t2[i].x += 50;  }  //以一顶点为缩放点  for (int i = 1; i < len; i++) {  t1[i].x = Sx \* (t1[i].x - t1[0].x) + t1[0].x;  t2[i].x = Sx \* (t2[i].x - t2[0].x) + t2[0].x;  t1[i].y = Sy \* (t1[i].y - t1[0].y) + t1[0].y;  t2[i].y = Sy \* (t2[i].y - t2[0].y) + t2[0].y;  }  setfillcolor(RED);  fillpolygon(t1, 3);  polygon(t2, 3);  times++;  }  //对称 about x axis  else if (m.message == WM\_LBUTTONDOWN && times == 2) {  float mid = t1[0].y;  for (int i = 0; i < len; i++) {  t1[i].y = 2 \* mid - t1[i].y;  t2[i].y = 2 \* mid - t2[i].y;  }  setfillcolor(RED);  fillpolygon(t1, 3);  polygon(t2, 3);  times++;  }  //对称 about y axis  else if (m.message == WM\_LBUTTONDOWN && times == 3) {  float midx = t1[2].x;  for (int i = 0; i < len; i++) {  t1[i].x = 2 \* midx - t1[i].x;  t2[i].x = 2 \* midx - t2[i].x;  //float cur1x[3], cur1y[3], cur2x[3], cur2y[3];  cur1x[i] = t1[i].x;  cur1y[i] = t1[i].y;  cur2x[i] = t2[i].x;  cur2y[i] = t2[i].y;  }  setfillcolor(RED);  fillpolygon(t1, 3);  polygon(t2, 3);  times++;  }  //错切 沿x轴方向关于y错切（x = x + cy）  else if (m.message == WM\_LBUTTONDOWN && times == 4) {  for (int i = 0; i < len; i++) {  t1[i].x += C \* t1[i].y;  t2[i].x += C \* t2[i].y;  }  setfillcolor(RED);  fillpolygon(t1, 3);  polygon(t2, 3);  times++;  }  //错切 沿y轴方向关于x错切（y = y + bx）  else if (m.message == WM\_LBUTTONDOWN && times == 5) {  for (int i = 0; i < len; i++) {  /\*  //float cur1x[3], cur1y[3], cur2x[3], cur2y[3];  cur1x[i] = t1[i].x;  cur1y[i] = t1[i].y;  cur2x[i] = t2[i].x;  cur2y[i] = t2[i].y;  \*/  //t1[i].y += B \* t1[i].x;  //t2[i].y += B \* t2[i].x;  t1[i].y = cur1y[i] + B \* cur1x[i];  t1[i].x = cur1x[i];  t2[i].y = cur2y[i] + B \* cur2x[i];  t2[i].x = cur2x[i];  }  setfillcolor(RED);  fillpolygon(t1, 3);  polygon(t2, 3);  times++;  }  }  \_getch();  closegraph();  return 0;  } |
| 2题 |
| //////////////////////////////////////////////////////  // 程序名称：实验3-2  // 功 能：实现预设图像的复合变换  // 编译环境：VS2019，EasyX\_20220116  // 作 者：夏婉可<2020301010225><1597493790@qq.com>  // 最后修改：2022-4-5  #include <graphics.h>  #include <conio.h>  #include <iostream>  #include <math.h>  #include <malloc.h>  #include <stdio.h>  using namespace std;  #define PI 3.1415926535  int dimension = 3, num = 4;  double points[50][2] = { {150,150},{150,200},{200,200},{200,150} };  //初始化  void initialize() {  initgraph(800, 640);  setbkcolor(WHITE);  setcolor(WHITE);  fillrectangle(0, 0, 800, 640);  setcolor(BLACK);  line(0, 80, 800, 80);  setcolor(BLACK);  line(0, 80, 800, 80);  //说明框矩形  RECT r = { 0,0,800,80 };  drawtext(\_T("\n\n依次展示旋转，放大，平移，关于直线对称，关于x错切"), &r, DT\_CENTER | DT\_VCENTER);  HRGN rgn = CreateRectRgn(1, 81, 799, 639);  setcliprgn(rgn);    setcolor(BLACK);  rectangle(0, 0, 800, 640);  setcolor(RED);  }  //矩阵乘法  void multiply(double a[5][5], int ar, int ac, double b[5][5], int br, int bc) {  if (ac != br) {  cout << "matrix invalid";  return;  }  double c[5][5];  for (int i = 0; i < ar; i++) {  for (int j = 0; j < bc; j++) {  c[i][j] = 0;  }  }  for (int i = 0; i < ar; i++) {  for (int j = 0; j < bc; j++) {  for (int k = 0; k < ac; k++) {  c[i][j] += a[i][k] \* b[k][j];  }  }  }  for (int i = 0; i < ar; i++) {  for (int j = 0; j < bc; j++) {  b[i][j] = c[i][j];  }  }  }  //平移变换  void trans(double tx, double ty) {  double T[5][5] = { {1,0,tx},{0,1,ty},{0,0,1} };  double point[5][5];  for (int i = 0; i < num; i++) {  point[0][0] = points[i][0];  point[1][0] = points[i][1];  point[2][0] = 1;  multiply(T, dimension, dimension, point, dimension, 1);  points[i][0] = point[0][0];  points[i][1] = point[1][0];  }  }  //旋转变换  void rotate(double degree) {  double theta = degree / 180 \* PI;  double R[5][5] = { {cos(theta),-sin(theta),0},{sin(theta),cos(theta),0},{0,0,1} };  double point[5][5];  for (int i = 0; i < num; i++) {  point[0][0] = points[i][0];  point[1][0] = points[i][1];  point[2][0] = 1;  multiply(R, dimension, dimension, point, dimension, 1);  points[i][0] = point[0][0];  points[i][1] = point[1][0];  }  }  //缩放变换  void scale(double sx, double sy) {  double S[5][5] = { {sx,0,0},{0,sy,0},{0,0,1} };  double point[5][5];  for (int i = 0; i < num; i++) {  point[0][0] = points[i][0];  point[1][0] = points[i][1];  point[2][0] = 1;  multiply(S, dimension, dimension, point, dimension, 1);  points[i][0] = point[0][0];  points[i][1] = point[1][0];  }  }  //对称变换  void symmetry(int flag) {  if (flag == 0) {  for (int i = 0; i < num; i++) {  points[i][1] = -points[i][1];  }  }  else if (flag == 1) {  for (int i = 0; i < num; i++) {  points[i][0] = -points[i][0];  }  }  else {  return;  }  }  void anysymmetry(int x1, int y1, int x2, int y2) {  double k = 0, b = 0;  if (x1 == x2) {  trans(-x1, 0);  symmetry(1);  trans(x1, 0);  }  else if (y1 == y2) {  trans(0, -y1);  symmetry(0);  trans(0, y1);  }  else {  k = double((y1 - y2) / (x1 - x2));  b = y1 - k \* x1;  trans(0, -b);  rotate(-atan(k) \* 180 \* 1.0 / PI);  symmetry(0);  rotate(atan(k) \* 180 \* 1.0 / PI);  trans(0, b);  }  }  //输出图形  void paint() {  for (int i = 0; i < num; i++) {  if (i == num - 1) {  line(points[i][0], points[i][1], points[0][0], points[0][1]);  break;  }  line(points[i][0], points[i][1], points[i + 1][0], points[i + 1][1]);  }  }  //错切变换  void cut(int flag, int degree) {  double point[5][5];  if (flag == 0) {  double C[5][5] = { {1,0,0},{tan(double(degree) / 180 \* PI),1,0},{0,0,1} };  for (int i = 0; i < num; i++) {  point[0][0] = points[i][0];  point[1][0] = points[i][1];  point[2][0] = 1;  multiply(C, dimension, dimension, point, dimension, 1);  points[i][0] = point[0][0];  points[i][1] = point[1][0];  }  }  else if (flag == 1) {  double C[5][5] = { {1,tan(double(degree) / 180 \* PI),0},{0,1,0},{0,0,1} };  for (int i = 0; i < num; i++) {  point[0][0] = points[i][0];  point[1][0] = points[i][1];  point[2][0] = 1;  multiply(C, dimension, dimension, point, dimension, 1);  points[i][0] = point[0][0];  points[i][1] = point[1][0];  }  }  else if (flag == 2) {  double C[5][5] = { {1,tan(double(degree) / 180 \* PI),0},{tan(double(degree) / 180 \* PI),1,0},{0,0,1} };  for (int i = 0; i < num; i++) {  point[0][0] = points[i][0];  point[1][0] = points[i][1];  point[2][0] = 1;  multiply(C, dimension, dimension, point, dimension, 1);  points[i][0] = point[0][0];  points[i][1] = point[1][0];  }  }  else {  return;  }  }  //复合变换  void multitrans() {  paint();  int i = 50;  while (i > 0) {  i--;  Sleep(50);  clearcliprgn();  trans(-150, -150);  rotate(-20);  trans(150, 150);  paint();  }  i = 8;  while (i > 0) {  i--;  Sleep(250);  scale(1.1, 1.1);  clearcliprgn();  paint();  }  i = 40;  while (i > 0) {  i--;  Sleep(50);  trans(-1, -1);  clearcliprgn();  paint();  }  i = 4;  while (i > 0) {  i--;  Sleep(250);  anysymmetry(250, 100, 300, 560);  clearcliprgn();  setcolor(BLACK);  line(250, 100, 300, 560);  setcolor(RED);  paint();  }  i = 20;  while (i > 0) {  i--;  Sleep(150);  cut(0, 1);  clearcliprgn();  paint();  }  }  //主函数  int main() {  initialize();  \_getch();  multitrans();  \_getch();  closegraph();  return 0;  } |