图形库

基本概念

- 坐标单位是inch
- 坐标系和正常使用的坐标系一样，原点在左下角

基本函数

main及初始化

```
void Main()
{
  InitGraphics();
  OpenConsole();//如果要输入字符
  CloseConsole();
}
```

绘图函数

- 获取屏幕中央坐标

`cx = GetWindowWidth()/2;`

`cy = GetWindowHeight()/2;`

- 获取当前坐标

```
x = GetCurrentX();
y = GetCurrentY();
```

- 移动画笔

    - 把画笔移到(x,y)处
    - 不画线

```
MovePen(x, y);
```

- 画线
    - 从当前位置开始画
    - 参数是偏移量，单位是inch
    - 新的画笔位置是线末端

```
DrawLine(dx, dy);
```

- 画圆
    - 始终从当前位置开始画
    - 第二个参数
        - 如果是0，则从🕒开始画
        - 如果是90，则从🕒开始画
        - 如果是180，则从🕒开始画
        - 如果是-90，则从🕒开始画
    - 第三个参数：圆的角度，单位是度

```
DrawArc(r, 0, 360);
```

- 中心圆

```
//自己写一个画中心圆的函数!
void DrawCenterCircle(double x,double y,double r) {
  MovePen(x+r,y);
  DrawArc(r,0,360);
}
```

计时器

- 计时器函数原型

```
void TimerEvent(int timerID);//timerID是计时器编号
```

- 主程序中注册回调函数

`registerTimerEvent(TimerEvent);//注册回调函数`

- 打开和关闭计时器

`startTimer(timerID,interval);//每隔interval启动计时器，单位ms`

`cancelTimer(timerID);`

键盘

- 键盘函数原型

`void KeyboardEventProcess(int key,int event);`

> `key`是按键的虚拟码，如`VS_ESCAPE`
>
> `event`只有两种，`KEY_DOWN`和`KEY_UP`

- 注册键盘回调函数

`registerKeyboardEvent(KeyboardEventProcess);`

鼠标

- 鼠标函数原型

- 注册鼠标回调函数

应用举例

闪烁 - SetEraseMode 和 TimerEvent

```
static bool isDisplayCircle = true;
if (timerID == TIMER_BLINK100) {
  bool erasemode = GetEraseMode();//初始化擦除函数
  SetEraseMode(isDisplayCircle);//设置是否擦除
  DrawCenterCircle(cx,cy,radius);
  SetEraseMode(erasemode);
  isDisplayCircle=!isDisplayCircle;
}
```

- 函数功能：*Flash a circle drawn in the center of a window once every 500 milliseconds. The ESCAPE key is used as a switch to toggle the blink.*

```
#include <windows.h>
#include "genlib.h"
#include "graphics.h"
#define TIMERB 1
const int mseconds = 500;
static double ccx = 1.0, ccy = 1.0; static double radius = 1.0;
static bool isB = FALSE;
static bool isD = TRUE;
```

```
void DrawCenteredCircle(double x, double y, double r); void KeyboardEventProcess(int
key,int event);
void TimerEventProcess(int timerID);
void Main()
{
  InitGraphics();
  registerKeyboardEvent(KeyboardEventProcess);
  registerTimerEvent(TimerEventProcess);
  ccx = GetWindowWidth()/2;
  ccy = GetWindowHeight()/2;
  DrawCenteredCircle(ccx, ccy, radius);
  if(isB) startTimer(TIMERB, mseconds);
}

void DrawCenteredCircle(double x, double y, double r) {
  MovePen(x+r, y);
  DrawArc(r, 0.0, 360.0); }
void KeyboardEventProcess(int key,int event) { if(event == KEY_DOWN && key ==
VK_ESCAPE) {
  isB = !isB;
  if (isB) {
    startTimer(TIMERB, mseconds);
  }
  else {
    cancelTimer(TIMERB);
    DrawCenteredCircle(ccx, ccy, radius);
  }

void TimerEventProcess(int timerID)
{
  if(timerID == TIMERB)
  {
    bool erasemode = GetEraseMode();
    SetEraseMode(isD); //true，则画笔设置为背景色；false，则画笔为正常颜色
    DrawCenteredCircle(ccx, ccy, radius);
    SetEraseMode(erasemode);
    isD=!isD;
  }
}
```

分形树

```
void FractalTree(int n, double x, double y, double length, double theta) {
  if (n > 0) {//分形次数，也即递归树深度
  double radians = theta / 180.0 * PI;//角度转弧度
  int dx = length * cos(radians);//计算x方向偏移量
  int dy = length * sin(radians);//计算y方向偏移量
```

```
  MovePen(x, y);
  DrawLine(dx, dy);//画第一条竖线
  FractalTree(n-1, x+dx, y+dy, length*0.75, theta + 15); //右树
  FractalTree(n-1, x+dx, y+dy, length*0.75, theta - 15);//左树
  }
}
void Main() {
  int n;
  double length;
  InitGraphics();
  OpenConsole();
  n = GetInteger();
  length = GetReal();
  CloseConsole();
  FractalTree(n, GetWindowWidth()/2.0, 0, length, 90);
  return;
}
```

等边三角形分形

```
#include "graphics.h"
#include <math.h>
#define LEN 6.0
#define PI 3.14159
#define EPS 0.05
typedef struct {
double x, y;
} VERTEX;
VERTEX MidPoint(VERTEX A, VERTEX B);
void DrawTriangle(VERTEX A, VERTEX B, VERTEX C);
void FraTriangle(VERTEX A, VERTEX B, VERTEX C);
void Main()
{
  VERTEX A, B, C;
  double cx, cy;
  InitGraphics();
  cx = GetWindowWidth()/2;
  cy = GetWindowHeight()/2;
  A.x = cx;
  A.y = cy + LEN/2*sin(PI/3);
  B.x = cx - LEN/2;
  B.y = cy - LEN/2*sin(PI/3);
  C.x = cx + LEN/2;
  C.y = B.y;
  FraTriangle(A, B, C);
}
void DrawTriangle(VERTEX A, VERTEX B, VERTEX C)/*Draw △ABC*/
{
  MovePen(A.x, A.y);
```

```
  DrawLine(B.x-A.x, B.y-A.y);
  DrawLine(C.x-B.x, C.y-B.y);
  DrawLine(A.x-C.x, A.y-C.y);
}
VERTEX MidPoint(VERTEX A, VERTEX B) {
  VERTEX mAB;
  mAB.x = (A.x + B.x) / 2;
  mAB.y = (A.y + B.y) / 2;
  return ___(1)___ ;//mAB
}
void FraTriangle(VERTEX A, VERTEX B, VERTEX C) {
  VERTEX mAB, mBC, mCA;
  if (fabs(A.x-B.x) < EPS) return;
  ___(2)___;//DrawTriangle(A,B,C);
  mAB = MidPoint(A, B); mBC = MidPoint(B, C); mCA = MidPoint(C, A);
  ___(3)___;//FracTriangle(A,mAB,mCA);
  ___(4)___;//FracTriangle(mAB,B,mBC);
  ___(5)___;//FracTriangle(mCA,mBC,C);

}
```

- 总结以上两个程序，都是在Main里直接调用递归函数
- 将n次分形，拆解成1 + (n-1)次分形。即先画出第一步，后面由递归实现
- 分形树的 **第一步** 是画出树干
- 三角形的 **第一步** 是画出大三角形