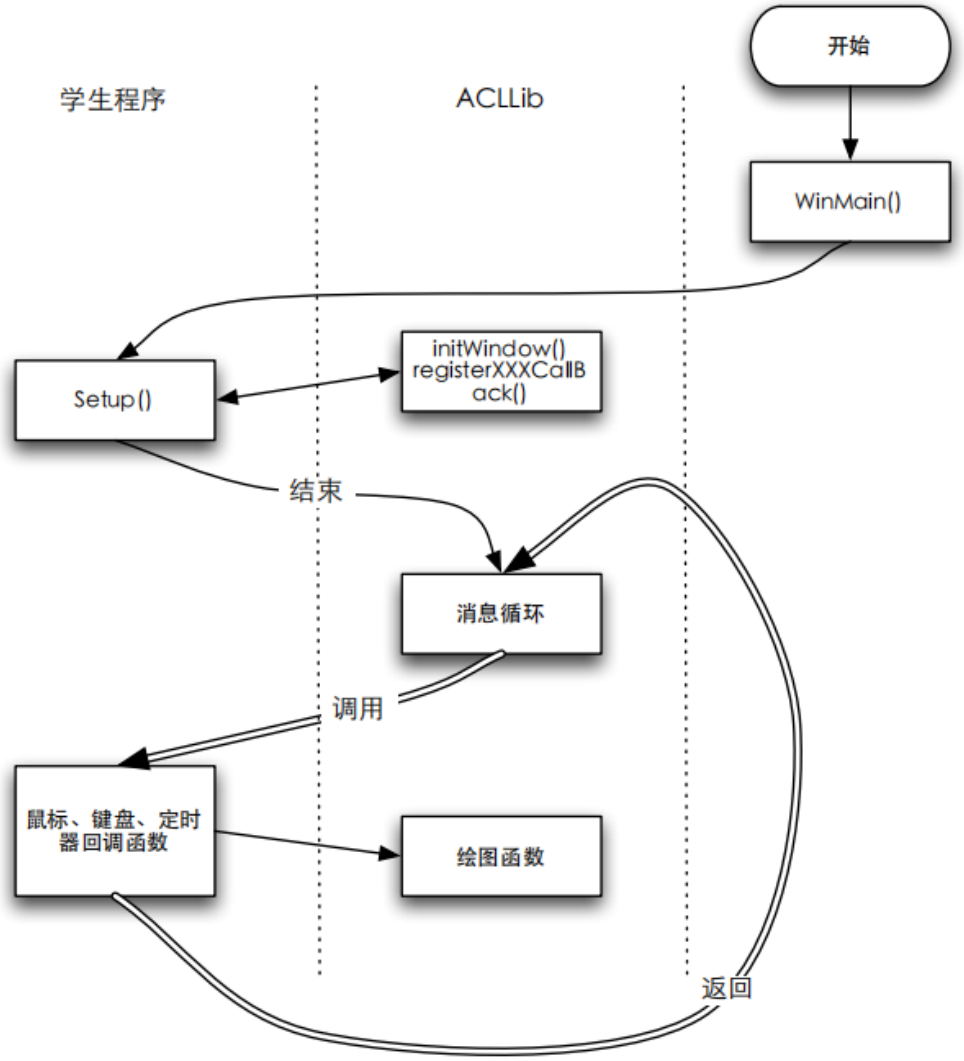
W7、8自学笔记

ACLLib图形库

ACLLib结构

下图是基本流程。



`Setup()`是用来初始化窗口，注册接受消息的回调函数的，`Setup()`的结束是程序的开始。

创建ACLLib程序

```
#include "acclib.h"
#include <stdlib.h>

int Setup(){
    initConsole();//为了可以使用printf和scanf
    printf("输入宽度:");
    int width;
    scanf("%d",&width);
    initWindow("test",100,100,width,width);//初始化一个窗口。
    /*所有画图都必须出现在beginPanit()和endPaint()之间*/
```

```
    beginPaint();
    line(20,20,width-20,width-20);
    endPaint();
    return 0;
}
```

基本绘图函数

```
void initWindow(const char title[],int left,int top,int width,int height);
void beginPaint();
void endPaint();
```

在Windows中，坐标是以像素点的数字来定义的。对于创建出来的窗口，左上角是（0,0），x轴由左向右增长，而y轴自上向下增长。

使用`scanf`和`printf`之前首先需要`initConsole();`

任何绘图函数的调用必须在`void beginPaint();`和`void endPaint();`这一对函数调用之间。

点

```
void putPixel(int x,int y,ACL_Color color);
ACL_Color getPixel(int x,int y);
```

颜色

RGB(r,g,b)，例如：红色→RGB（255,0,0）

预先定义好的符号：BLACK,RED,GREEN,BLUE,CYAN,MAGENTA,YELLOW,WHITE

线

```
void moveTo(int x,int y);
void moveRel(int dx,int dy);
void lint(int x0,int y0,int x1,int y1);
void lineTo(int x,int y);
void lineRel(int dx,int dy);
```

画笔

```
void setPenColor(ACL_Color color);
void setPenWidth(int width);
void setPenStlye(ACL_Pen_Style style);
      PEN_STYLE_SOLID
      PEN_STYLE_DASH              /*-------*/
      PEN_STYLE_DOT               /*.......*/
      PEN_STYLE_DASHDOT           /*_._._._*/
      PEN_STYLE_DASHDOTDOT        /*_.._.._*/
      PEN_STYLE_NULL
```

面、刷子

文字

```
void setTextColor(ACL_Color color);
void setTextBkColor(ACL_Color color);
void setTextSize(int size);
void setTextFont(char *pFontName);
void paintText(int x, int y, const char *pStr);
```

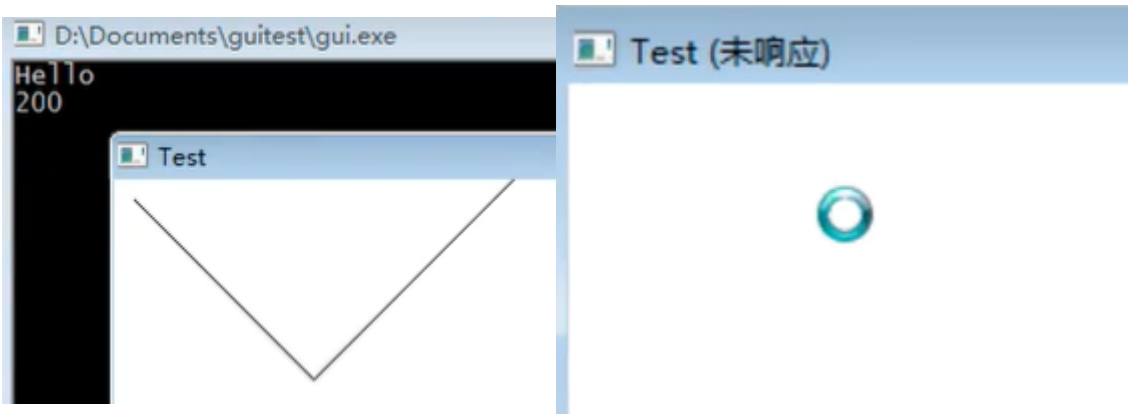
交互图形程序

终端输入输出

```
#include "acllib.h"
#include <stdio.h>

int main(){
    initWindow("Test",DEFAULT,DEFAULT,800,600);
    /*initConsole();
    printf("Hello\n");
    int x;*/
    beginPaint();
    line(10,10,100,100);
    /*scanf("%d",&x);
    line(100,100,x,0);*/
    endPaint();
    return 0;
}
```

要注意，如果想要让这个程序与用户交互（比如接受输入），可以用`printf`，但在这个程序中不能直接用，需要加入`initConsole();`但需要在`beginPaint();`之前。



在上述这种情况下，只有在输入数据以后，显示图形的窗口才有响应。因为当程序在做`scanf`时，程序是停下来的，因此无法在图形界面响应。只有在用户终端窗口输入以后，图形界面才有反应。

函数指针

定义：如果在程序中定义了一个函数，那么在编译时系统就会为这个函数代码分配一段存储空间，这段存储空间的首地址称为这个函数的地址。而且函数名表示的就是这个地址。既然是地址我们就可以定义一个指针变量来存放，这个指针变量就叫作函数指针变量，简称函数指针。

函数指针的定义方式

```
returnType (*pointerName)(param list);//函数返回值类型(* 指针变量名)(函数参数列表);
```

注：`( )`的优先级高于`*`，第一个括号不能省略,如果写作`returnType *pointerName(param list);`就成了函数原型，它表明函数的返回值类型为`returnType *`。

“函数返回值类型”表示该指针变量可以指向具有什么返回值类型的函数；“函数参数列表”表示该指针变量可以指向具有什么参数列表的函数。这个参数列表中只需要写函数的参数类型即可。

首先看变量名前面有没有“”，*如果有*“说明是指针变量；其次看变量名的后面有没有带有形参类型的圆括号，如果有就是指向函数的指针变量，即函数指针，如果没有就是指向变量的指针变量。

```
#include <stdio.h>

void f(void){
    printf("in f()\n");
}
int main(void){
    int i=0;
    int *p=&i;
    *p=20;
     int a[]={1,2};
    //f;在程序中只写一个f，编译时会给warning，但不是错误，表达式的结果没有被使用，单独的f是一个地址。
    //p=f;如果直接赋值，编译时会给warning，因为类型不匹配，但并不意味着这种赋值不被允许。
    void (*pf)(void)=f;//这是正确的写法，定义一个变量，变量名是pf，类型是指向函数的指针。
    //*pf=0;错误的写法。
    f();//调用f函数。
    (*pf)();//调用了f函数。
    printf("%p\n",main);
    //printf("%p\n",f);
    printf("%p\n",pf);
    printf("%p\n",a);
}
```

输出结果：

```
in f()
in f()
0x10d8a3ea0
0x10d8a3e80
0x7fff5235cce0
[Finished in 0.0s]
```

如何判断指针变量是指向变量的指针还是指向函数的指针变量?

首先看变量名前面有没有`*`，如果有`*`说明是指针变量；其次看变量名的后面有没有带有形参类型的圆括号，如果有就是指向函数的指针变量，即函数指针，如果没有就是指向变量的指针变量。

注：最后需要注意的是，指向函数的指针变量没有 ++ 和 -- 运算。

函数指针的使用

简单示例：

```
int Func(int x);   /*声明一个函数*/
int (*p) (int x);  /*定义一个函数指针*/
p = Func;          /*将Func函数的首地址赋给指针变量p*/
```

注：赋值时函数 Func 不带括号，也不带参数。由于函数名 Func 代表函数的首地址，因此经过赋值以后，指针变量 p 就指向函数 Func() 代码的首地址了。

```c
#include <stdio.h>

void f(int i){
    printf("in f(),%d\n",i);
}
int main(void){
    int i=0;
    int *p=&i;
    *p=20;
    int a[]={1,2};
    void (*pf)(int)=f;
    f(20);
    (*pf)(10);//函数指针，括号中是参数。
    printf("%p\n",main);
    printf("%p\n",pf);
    printf("%p\n",a);
}
```

函数指针数组的实用之处：当我们需要判断大量条件的时候，并且在每一个条件都有相应的处理函数，这时使用 `switch` 和 `case` 的代码量会很大，并且效率会比较低，这个时候就可以使用函数指针数组来解决这个问题了。

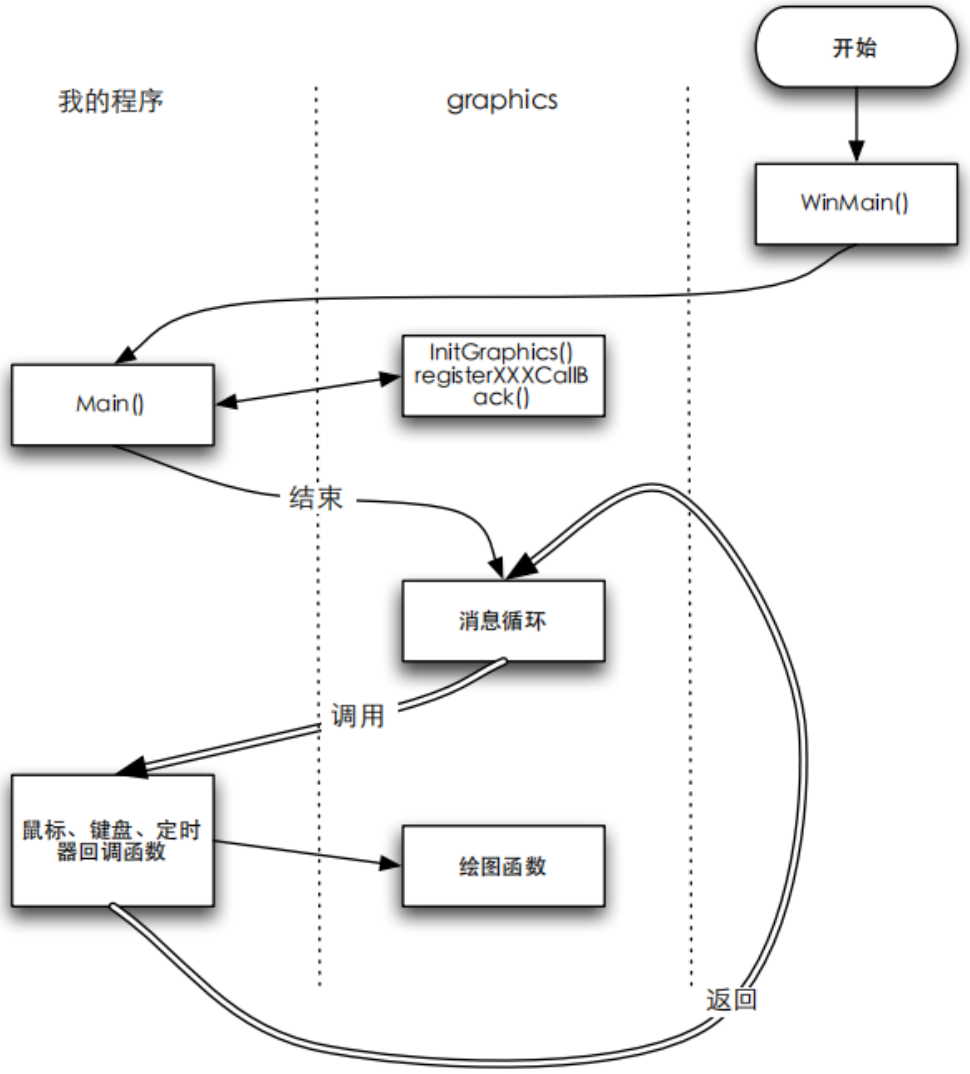
```c
#include <stdio.h>
void f(int i){printf("in f(),%d\n",i);}
void g(int i){printf("in f(),%d\n",i);}
void h(int i){printf("in f(),%d\n",i);}
//void k(int i){printf("in f(),%d\n",i);}
int main(void){
    int i=0;
    void(*fa[])(int)={f,g,h/*,k*/};
    scanf("%d",&i);
    /*switch(i){
        case 0:f(0);break;
        case 1:g(0);break;
        case 2:h(0);break;
    }如果用switch结构，需要不断的加入函数，但是用函数指针可以解决*/
    if(i>=0&&i<sizeof(fa)/sizeof(fa[0])){
        (*fa[i])(0);
    }//通过这样的方式，从scanf到这里的代码是与有多少函数被调用无关。如果要就，只需要在开头加入
void k(int i)，在数组中加入k;
}
```

```c
#include <stdio.h>
int plus(int a,int b){return a+b;}
int minus(int a,int b){return a-b;}
void cal(int (*f)(int,int)){
    printf("%d",(*f)(2,3));
}//此时cal函数无需任何改变，传递给它怎样的函数，cal函数就会有什么样的改变。
int main(void){
    cal(plus);
    cal(minus);
    return 0;
}
```

```
5-1[Finished in 0.0s]
```

回调函数

回调函数：通过函数指针调用的函数。如果你把函数的指针（地址）作为参数传递给另一个函数，当这个指针被用来调用其所指向的函数时，我们就说这是回调函数。回调函数不是由该函数的实现方直接调用，而是在特定的事件或条件发生时由另外的一方调用的，用于对该事件或条件进行响应。



register将某个函数指针交给某个地方保存下来。

```
#include "acllib.h"
#include <stdio.h>

void mouseListener(int x,int y,int button,int event){
    printf("x=%d,y=%d,button=%d,event=%d");
}
int main(){
    initWindow("Test",DEFAULT,DEFAULT,800,600);
    initConsole();
    printf("Hello\n");
    int x;
    registerMouseEvent(mouseListener);//每次鼠标一动，函数就被调用，在本程序中即会有输出。这就是回调函数。
    beginPaint();
```

```
    line(10,10,100,100);
    endPaint();
    return 0;
}
```

图形交互消息

```
The Callbacks
typedef void(*KeyboardEventCallback)(const char key);//可以知道一些特殊功能键，对于每个功能键有按下和抬起两种状态。
typedef void(*CharEventCallback)(int key);
typedef void(*MouseEventCallback)(int x,int y,int button,int status);//可以看到鼠标的移动，按下和弹起。
typedef void(*TimerEventCallback)(int timerID);//是一个定时器。
```

```
#include "acllib.h"
#include <stdio.h>

void mouseListener(int x,int y,int button,int event){
    static int ox=0;
    static int oy=0;
    printf("x=%d,y=%d,button=%d,event=%d");
    beginPaint();
    lineTo(ox,oy);//如果直接lineTo(x,y)且没有开头static，程序运行的结果是每次鼠标移动的时候，从原点到鼠标处划线。
    endPaint();
    ox=x;oy=y;
}
void keyListener(int key,int event){
    printf("key=%d,event=%d\n",key,event);
}
void timerListener(int id){
    static int cnt=0;
    printf("id=%d\n",id);
    if(id==0){
        cnt++;
        if(cnt==5) cancelTimer(0);
    }
}
int main(){
    initWindow("Test",DEFAULT,DEFAULT,800,600);
    initConsole();
    printf("Hello\n");
    int x;
    registerMouseEvent(mouseListener);
    registerKeyboardEvent(keyListener);
    registerTimerEvent(timerListener);
    startTimer(id,500);
    startTimer(id,1000);
    beginPaint();
    line(10,10,100,100);
    endPaint();
    return 0;
}
```

MVC设计模式

MVC是一种设计模式，如何安排程序各个部分之间的关系。

View读取数据（当需要画东西的时候，从Model取数据），数据位于Model。Ctrl告诉Model什么数据怎么改，Model通知View数据经过改动。在这个模式中，Ctrl不会直接影响View（鼠标的动作不会直接引起图象的改变，会引起数据的改变）。

游戏设计思路

①bs for objects

②timer（要做的事情在间隔内做完）

③scan（扫描数据结构）move

④hit，crash

⑤refresh，draw

TC，KC，Paint