
程序设计专题

主讲教师: 纪守领

学院: 计算机科学与技术

NESA Lab: <https://nesa.zju.edu.cn/>

Email: sji@zju.edu.cn

专题四：图形程序设计

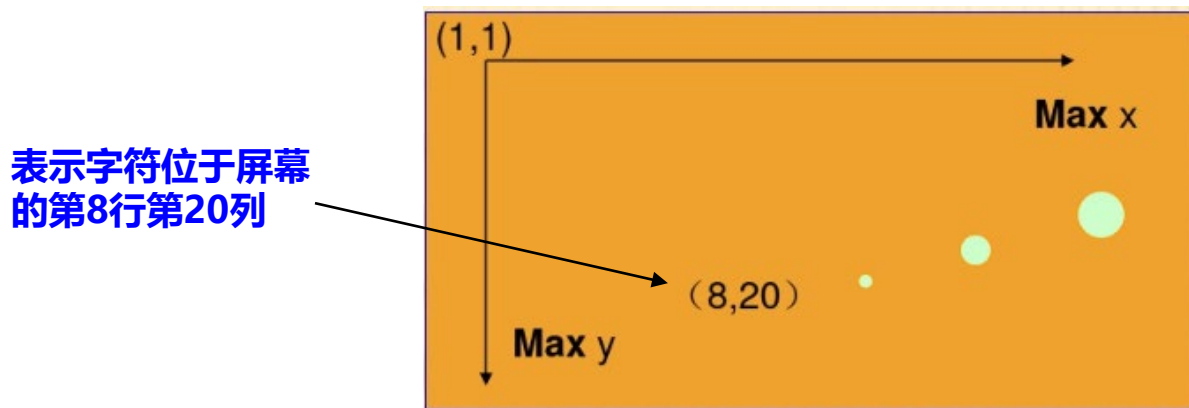
- 图形学基础
- 第三方图形库：libgraphics
- 交互式编程
- SimpleGUI

图形学基础

- 计算机图形学是一门实践性很强的课。
- 现在大多数高级程序设计语言都具有基本的绘图功能。
- 在屏幕上进行绘制图形，一般按照以下几个步骤进行：
 - 把屏幕设置为图形模式；
 - 选择背景与显示实体的颜色；
 - 计算图形显示坐标；
 - 调用绘图语句绘制实体。

文本模式与字符坐标系

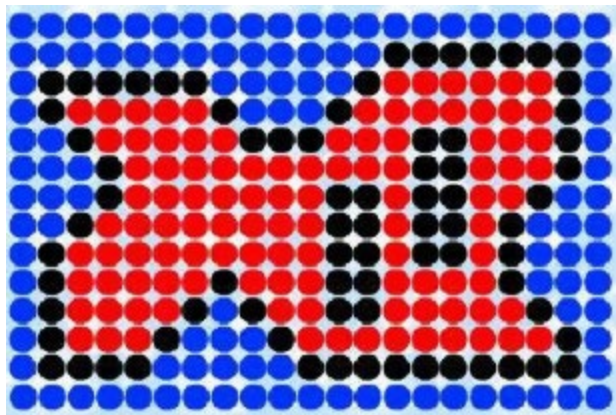
- 在屏幕上只能显示字符的方式称为**文本模式**。在文本模式下，屏幕上可以显示的最小单位是**字符**；为了能在指定的位置上显示每个字符，C语言提供了**字符坐标系**。
- 字符坐标系是以屏幕的左上角为坐标原点，水平方向为x轴，垂直方向为y轴。



注意：字符坐标系的原点为 $(1, 1)$ ，水平方向分为若干列，垂直方向分为若干行，用一对坐标可以指定屏幕上一个位置。

图形模式与点坐标系

- 在屏幕上显示图形的方式称为**图形模式**。在图形模式下，屏幕是由**像素点**组成的，像素点的多少决定了**屏幕的分辨率**。分辨率越高，显示图形越细致，质量越好。



图形模式与点坐标系

- 在图形模式下，屏幕上各个像素的显示位置用点坐标来描述，点坐标是以屏幕左上角为坐标原点(0, 0)，水平方向为x轴，自左向右；垂直方向为y轴，自上而下。
- 水平方向和垂直方向的点数随着屏幕分辨率的不同而不同。



点坐标系坐标值的范围取决于所用的适配器/显示分辨率


















第三方图形库

- C语言本身不提供图形绘制功能。
- 借助于第三方提供的图形库，可实现图形的绘制。
- 图形库以C源码形式，或者以二进制目标码形式提供。
- 在应用第三方图形库时，不需要了解其具体的实现，只需了解其基本功能和图形绘制流程。
- 直接调用相关图形库函数来实现具体的图形绘制。
- 注意：图形库接口——头文件应当被包含到源文件中。
- 头文件包含了相关图形库函数的原型。

第三方图形库：libgraphics

- 本课程采用的第三方图形库封装了Windows Graphics Device Interface(图形设备接口)，提供了一套更为简单的绘图接口。

接口文件和实现

 boolean.h	2020/3/20 10:22
 exceptio.c	2017/6/11 16:13
 exception.h	2017/6/11 16:13
 extgraph.h	2019/2/27 17:06
 gcalloc.h	2017/6/11 16:13
 genlib.c	2017/6/11 16:13
 genlib.h	2020/4/1 16:03
 graphics.c	2019/3/29 17:27
 graphics.h	2020/3/20 10:36
 linkedlist.c	2020/3/19 22:01
 linkedlist.h	2020/3/20 10:22
 random.c	2017/6/11 16:13
 random.h	2017/6/11 16:13
 simpio.c	2017/6/11 16:13
 simpio.h	2017/6/11 16:13
 strlib.c	2017/6/11 16:13
 strlib.h	2017/6/11 16:13

graphics.c中包含的头文件

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include <conio.h>
#include <windows.h>
#include <time.h>
#include <wincon.h>
#include <Windows.h>

#include "genlib.h"
#include "gcalloc.h"
#include "strlib.h"
#include "extgraph.h"
```

内部实现细节可以参考官方文档中GDI相关的部分

<https://docs.microsoft.com/zh-cn/windows/win32/gdi/windows-gdi>

第三方图形库：libgraphics

- ❑ 在Win32API中，第一个C函数是WinMain()，而不是main()，且要遵循windows编程规范。windows API使用比较复杂，需要花很多时间去学习。
- ❑ 为了方便初学者使用，在libgraphics中，已实现了通用的WinMain()基本功能。而应用程序所要做的相关初始化工作只需写在Main()函数中即可。实际上，主程序的实际入口是graphics.c中的WinMain() 函数，它调用了用户实现的Main()函数：

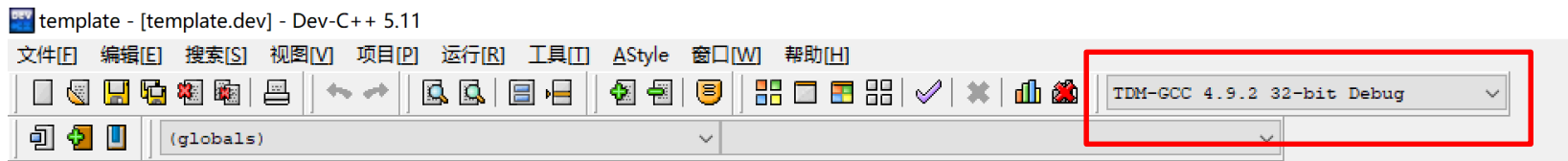
```
1924 int WINAPI WinMain (HINSTANCE hThisInstance,
1925                     HINSTANCE hPrevInstance,
1926                     LPSTR lpszArgument,
1927                     int nFunsterStil)
1928 {
1929     MSG messages;           /* Here messages to the application are saved */
1930
1931     Main();
1932
1933     /* Run the message loop. It will run until GetMessage() returns 0 */
1934     while (GetMessage (&messages, NULL, 0, 0))
1935     {
1936         /* Translate virtual-key messages into character messages */
1937         TranslateMessage(&messages);
1938         /* Send message to WindowProcedure */
1939         DispatchMessage(&messages);
1940     }
1941     FreeConsole();
1942     return messages.wParam;;
1943 }
1944
```

第三方图形库：libgraphics

- ❑ 在Win32API中，第一个C函数是WinMain()，而不是main()，且要遵循windows编程规范。windows API使用比较复杂，需要花很多时间去学习。
- ❑ 为了方便初学者使用，在libgraphics中，已实现了通用的WinMain()基本功能。而应用程序所要做的相关初始化工作只需写在Main()函数中即可。实际上，主程序的实际入口是graphics.c中的WinMain() 函数，它调用了用户实现的Main()函数。
- ❑ 在Main()函数中，首先要调用InitGraphics()来初始化窗口，以便绘制图形。
- ❑ libgraphics图形窗口坐标原点在左下角。

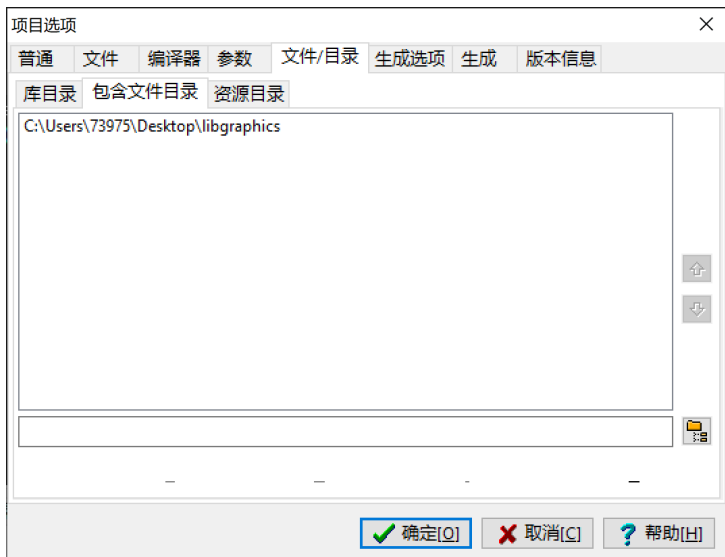
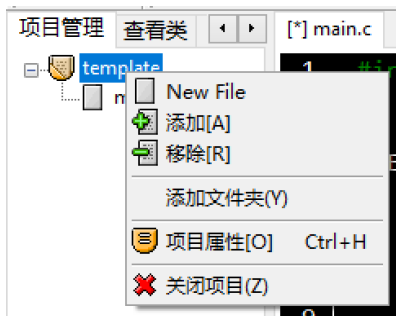
libgraphics环境配置示例 (Dev C++)

- 文件 -> 新建 -> 项目
- 选择Windows Application -> 输入项目名 -> 选择C项目 -> 选择项目保存路径
- 工具 -> 编译选项 -> 设定编译器配置 -> 设定为32位编译器
- 设定编译器为32位编译器



libgraphics环境配置示例 (Dev C++) (续)

- ❑ 打开工程所在目录，将libgraphics库拷贝到工程目录下
- ❑ 项目管理选项卡中右击项目根目录，设置项目属性
- ❑ 文件/目录选项卡 -> 包含文件目录选项卡
- ❑ 将libgraphics目录添加到路径中 (采用相对路径)



libgraphics环境配置示例 (Dev C++) (续)

- ❑ 项目管理选项卡中右击项目根目录，添加文件夹，命名为libgraphics
- ❑ 右击libgraphics文件夹，添加文件
- ❑ 将libgraphics库目录中的.c文件都添加到刚刚创建的libgraphics目录中，编辑main.c，实现显示一个空窗口。最后编译运行。

The screenshot displays the Dev C++ IDE interface. On the left, the 'Project Manager' (项目管理) pane shows a project named 'template' containing a folder 'libgraphics'. This folder contains several .c files: 'exceptio.c', 'genlib.c', 'graphics.c', 'linkedlist.c', 'random.c', 'simpio.c', 'strlib.c', and 'main.c'. The 'main.c' file is selected and its code is visible in the central editor. The code includes 'graphics.h' and 'extgraph.h', and defines a 'Main()' function that sets the window title to 'test' and calls 'InitGraphics()'. To the right of the editor, a small window titled 'test' is shown, representing the output of the program.

```
1  #include "graphics.h"
2  #include "extgraph.h"
3
4
5  void Main()
6  {
7      SetWindowTitle("test");
8      InitGraphics();
9  }
```

常用的绘制函数

InitGraphics()	Initialize the graphics package, open the window for rendering
MovePen(x, y)	Move the pen to an absolute position
DrawLine(dx, dy)	Draw a line from the current position to a relative coordinates
DrawArc(r, start, sweep)	Draw an arc specified by a radius and two angles
GetWindowWidth()	Return the width of the graphics window
GetWindowHeight()	Return the height of the graphics window
GetCurrentX()	Return the current x-coordinate of the pen
GetCurrentY()	Return the current y-coordinate of the pen

Example: Drawline

- ❑ 画直线函数**Drawline(dx, dy)**
- ❑ dx和dy是相对于画笔当前位置的偏移量
- ❑ 功能：从当前位置到dx, dy位置绘制一条直线
- ❑ 画完直线后，画笔移动到(x+dx, y+dy)

Example: Drawline

```
#include "graphics.h"
#include "extgraph.h"

void Main()
{
    SetWindowTitle("DrawLine");
    InitGraphics();

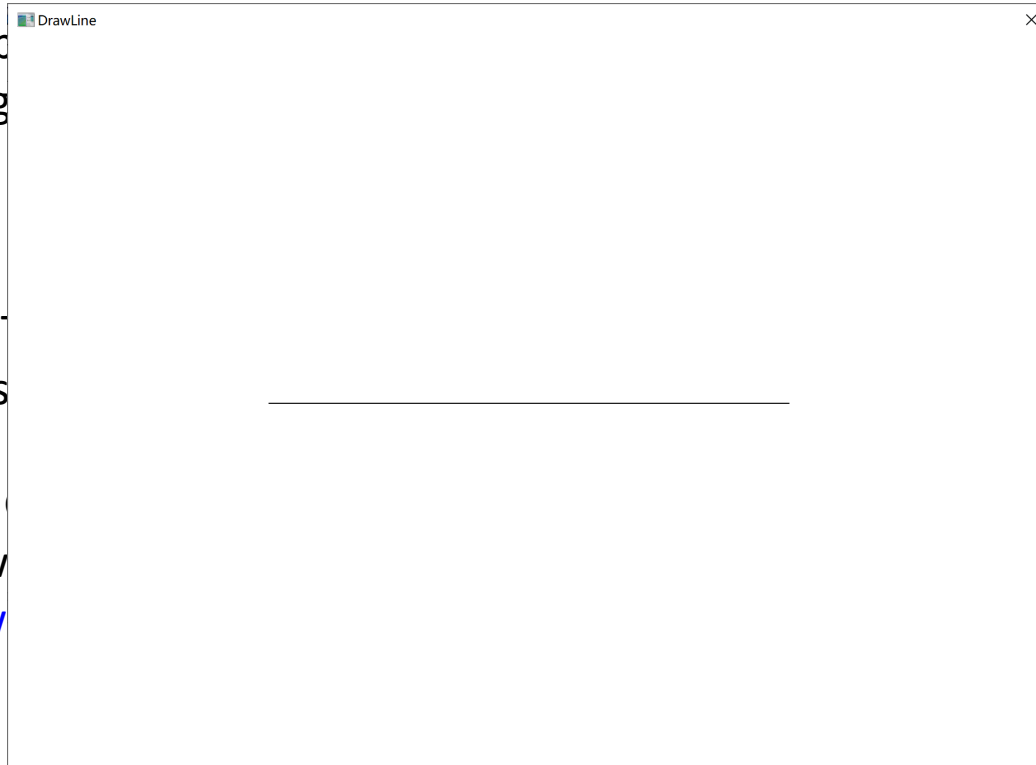
    double w = GetWindowWidth(), h = GetWindowHeight();
    MovePen(w / 4, h / 2);
    DrawLine(w / 2, 0.);
}
```


Example: Drawline

```
#include "graph.h"
#include "extg.h"

void Main()
{
    SetWindow
    InitGraphics

    double w =
    MovePen(w
    DrawLine(w
}
```



Example: DrawArc

- 绘制圆弧函数 `DrawArc(r, start, sweep)`
- 以画笔当前位置作为圆弧所在圆的x轴上右起点，画一段圆弧。
- 圆弧半径为 r ，起始角度为 $start$ （角度制），弧度为 $sweep$ （角度制），逆时针方向为正。
- 画圆： $sweep$ 设置为360

Example: DrawArc

```
#include "graphics.h"
#include "extgraph.h"

void Main()
{
    SetWindowTitle("DrawArc");
    InitGraphics();

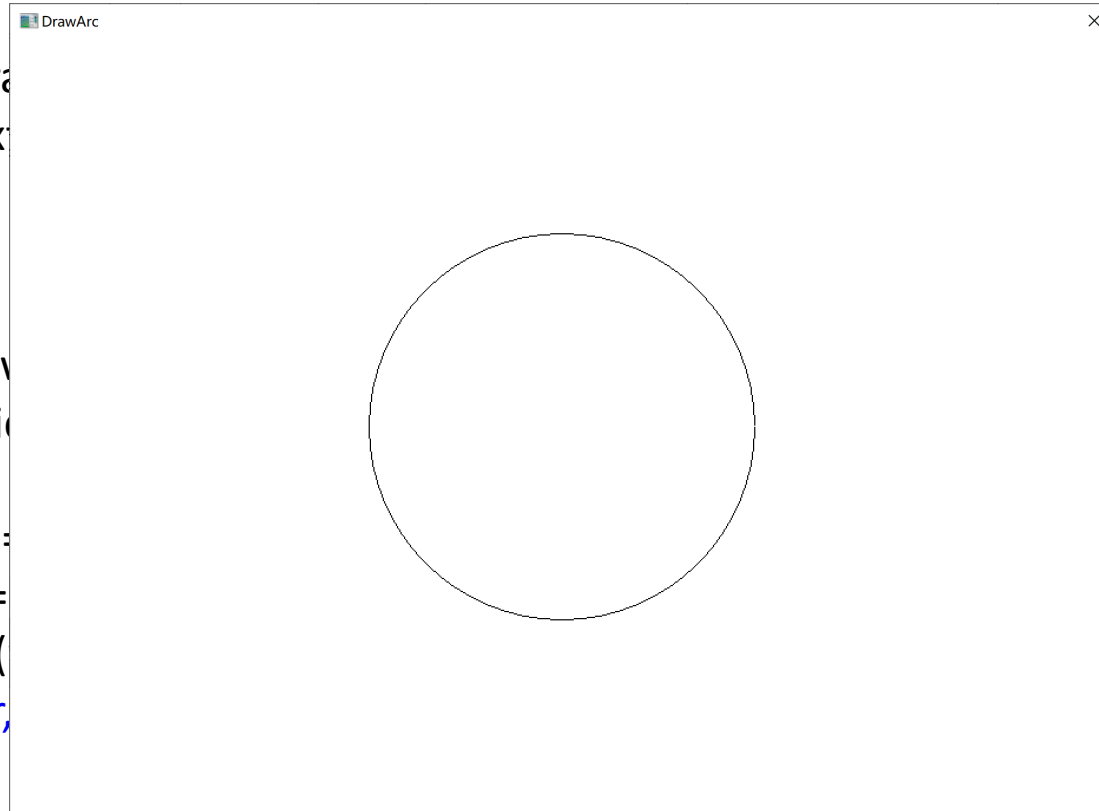
    double w = GetWindowWidth(), h = GetWindowHeight();
    double r = (w < h ? w : h) / 4;
    MovePen(w / 2 + r, h / 2);
    DrawArc(r, 0, 360);
}
```

Example: DrawArc

```
#include "gra  
#include "ex
```

```
void Main()  
{  
    SetWindow  
    InitGraphic
```

```
    double w =  
    double r =  
    MovePen(  
    DrawArc(r,  
}
```



Example: DrawTextString

- ❑ 绘制文本**DrawTextString(string)**
- ❑ 常用的printf用于标准输出（控制台窗口）输出格式化数据，不能用于在图形窗口输出文本。
- ❑ 图形库提供了专门用于图形窗口输出文本的函数。
- ❑ 从当前位置开始输出文本字符串string。
- ❑ string是字符串指针。
- ❑ 可以用sprintf格式化字符串。

Example: DrawTextString

```
#include "graphics.h"
#include "extgraph.h"
#include <time.h>
void Main()
{
    SetWindowTitle("DrawText");
    InitGraphics();
    SetFont("Times");
    SetPointSize(50);
    struct tm *local; time_t t;
    t = time(NULL);
    local = localtime(&t);
    string text = (string)malloc(sizeof(char)*64);
    sprintf(text, "C Programming %d", local->tm_year+1900);
    double w = GetWindowWidth(), h = GetWindowHeight();
    MovePen(w / 2 - TextStringWidth(text) / 2, h / 2 - GetFontAscent() / 2);
    DrawTextString(text);
}
```

libgraphics/genlib.h:
typedef char *string;

Example: DrawTextString

```
#include "graphics.h"
```

```
#include "extgraph.h"
```

```
#include <time.h>
```

```
void Main()
```

```
{
```

```
    SetWindowTitle("DrawText")
```

```
    InitGraphics();
```

```
    SetFont("Times")
```

```
    SetPointSize(50);
```

```
    struct tm *local;
```

```
    t = time(NULL);
```

```
    local = localtime(&t);
```

```
    string text = (string)0;
```

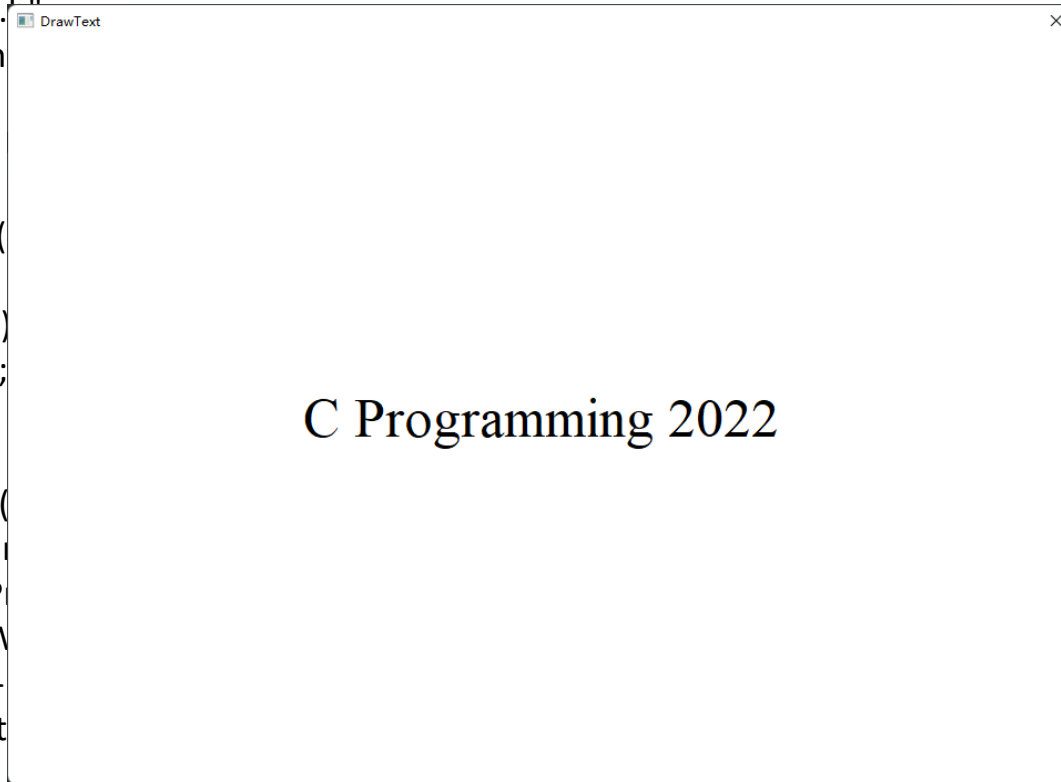
```
    sprintf(text, "C P
```

```
    double w = GetW
```

```
    MovePen(w / 2 -
```

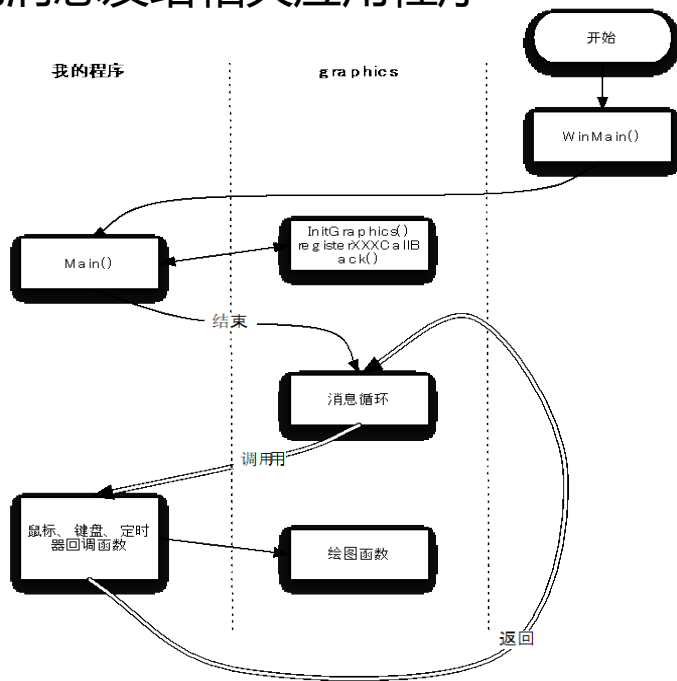
```
    DrawTextString(t
```

```
}
```



交互式编程：基于事件驱动的编程模型

- 以往的程序：当程序有需要时等待用户的输入
- 事件驱动程序：当有用户输入时就响应，其他时间不会阻塞等待
- 系统会捕获事件并把消息发给相关应用程序



graphics.h中定义的四种回调函数类型

□ 键盘消息回调函数

- **void KeyboardEventProcess(int key, int event);**

- key表示哪个按键，event表示按下或松开事件

□ 字符消息回调函数

- **void CharEventProcess(char c);**

- c表示按键的ASCII码

□ 鼠标消息回调函数

- **void MouseEventProcess(int x, int y, int button, int event);**

- x, y位置坐标，button表示哪个键，event表示按下/松开/移动事件

□ 定时器消息回调函数

- **void TimerEventProcess(int timerID);**

- timerID表示定时器号，哪个定时器触发了消息

Example: Simple Clock

- 编写一个时钟，实时显示当前时间。刷新频率为每秒30帧。



Example: Simple Clock

```
typedef enum {  
    FLUSH,  
    TICK  
}MyTimer;  
void mytimer(int timerID)  
{  
    time_t t;  
    switch(timerID)  
    {  
        case FLUSH:  
            display();  
            break;  
        case TICK:  
            t = time(NULL);  
            local = localtime(&t);  
            break;  
        default:  
            break;  
    }  
}
```

```
static struct tm *local;  
  
void Main()  
{  
    SetWindowTitle("Simple Clock");  
    InitGraphics();  
    SetFont("Times");  
    SetPointSize(50);  
  
    time_t t;  
    t = time(NULL);  
    local = localtime(&t);  
  
    registerTimerEvent(mytimer);  
    startTimer(FLUSH, 1000/30);  
    startTimer(TICK, 1000);  
  
    display();  
}
```

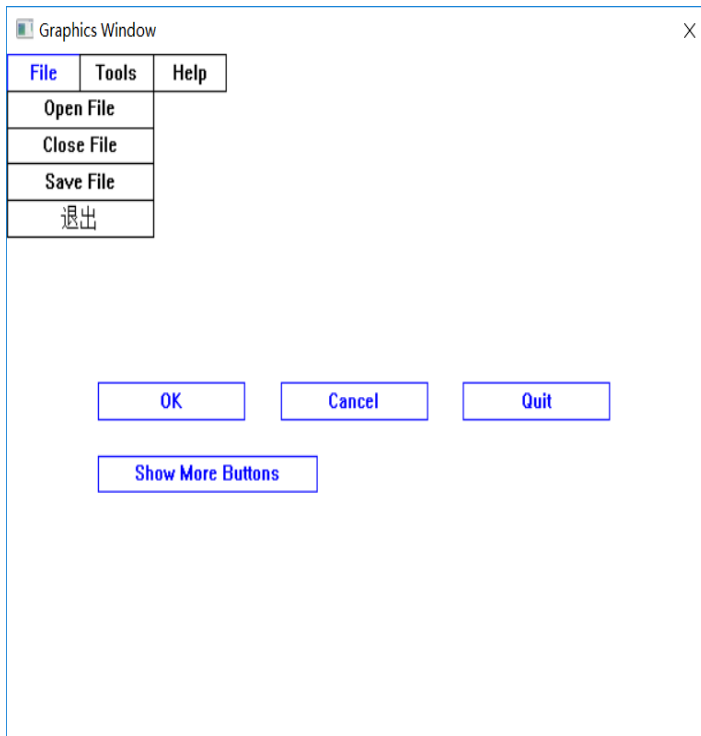
Example: Simple Clock

```
void display()
{
    static char buffer[128];
    DisplayClear();
    sprintf(buffer, "%04d/%02d/%02d %02d:%02d:%02d",
        local->tm_year+1900,
        local->tm_mon+1,
        local->tm_mday,
        local->tm_hour,
        local->tm_min,
        local->tm_sec);

    double w = GetWindowWidth(), h = GetWindowHeight();
    MovePen(w / 2 - TextStringWidth(buffer) / 2, h / 2 - GetFontAscent() / 2);
    DrawTextString(buffer);
}
```

SimpleGUI

- ❑ 一种简单的即时模式GUI
 - ❑ immediate mode graphics user interface
 - ❑ 适合高刷新率的应用程序
- ❑ 目前只实现了3个控件
 - ❑ button: 鼠标按钮
 - ❑ menulist: 菜单列表
 - ❑ textbox: 编辑字符串
- ❑ 课后结合文档和代码自行学习



如何深入学习

□ 参考资料（已上传学在浙大）

- doclibgraphics：介绍libgraphics的基本用法以及simpleGUI的基本用法
- docproject：介绍基本的工程管理知识
- libgraphics：libgraphics库文件，包括接口文件.h和源文件.c
- simpleGUI：基于libgraphics实现的简单控件库
- tutorialcode：各个tutorial工程的源代码目录
- tutorialDevC：各个tutorial的DevC工程
- tutorialVS2010：各个tutorial的VS工程
- readme：简介

□ 学习方法：

- 结合tutorial示例掌握libgraphics库的绘图逻辑以及常用的函数接口
- 阅读源代码中的注释，了解各函数功能及其接口格式
- 如果学有余力，可以尝试阅读其源代码，了解windows GDI

谢 谢
