

# simpleGUI使用简介

刘新国

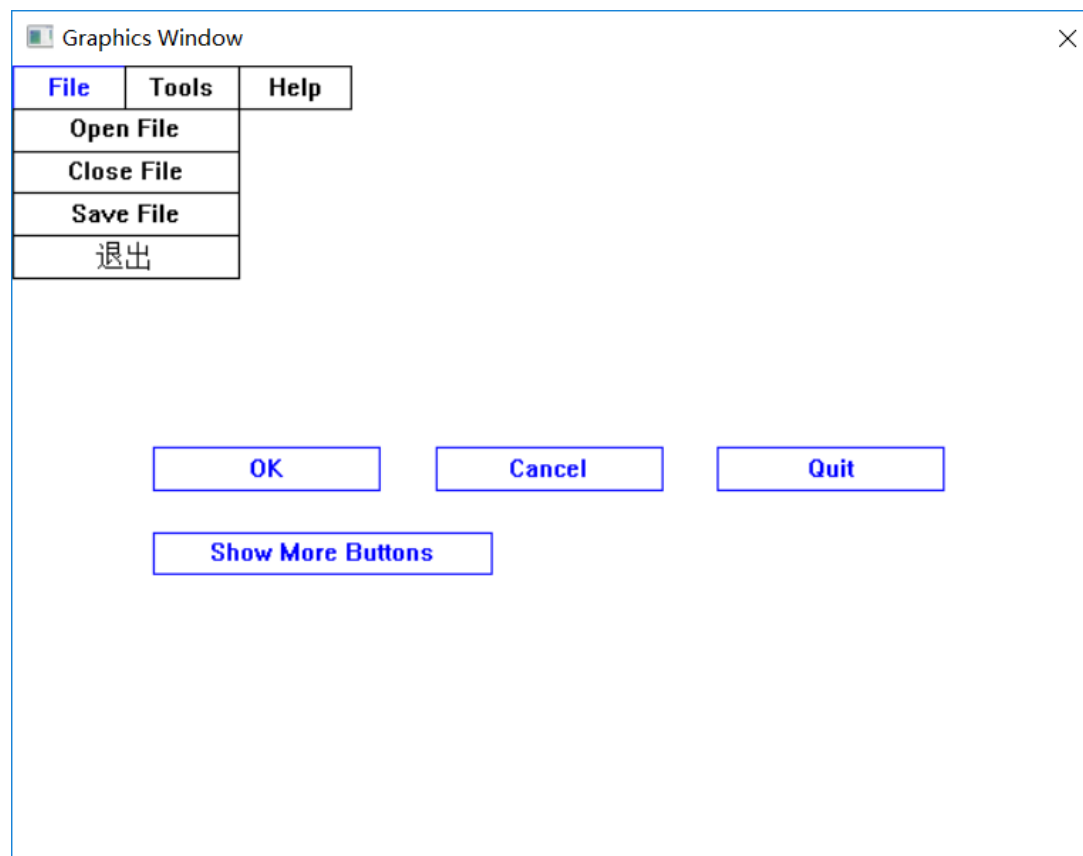
浙江大学计算机科学与技术学院

CAD&CG国家重点实验室

2019年2月

# 什么是simpleGUI

- 是一种简单的即时模式GUO
  - IMGUI – immediate mode graphics user interface
  - 适合高刷新率的应用程序
    - 屏幕总在实时刷新
- 目前只实现了3个控件
  1. button – 鼠标按钮
  2. menuList – 菜单列表
  3. textbox – 编辑字符串



# 如何使用simpleGUI

- 必须和libgraphics库一起使用
  - 如果和其他的图形库使用，需要做简单修改
- 在程序中包含头文件

```
#include "imgui.h"
```
- 将imgui.c加入程序工程中，或者在某个c文件中包含它

```
#include "imgui.c"
```
- 在鼠标处理函数中调用simpleGUI函数

```
guiGetMouse
```

记录鼠标状态

```
void MouseEventProcess(  
    int x, int y, int button, int event)  
{  
    /*擦除屏幕*/  
    DisplayClear();  
  
    /* 获取鼠标状态 */  
    uiGetMouse(x,y,button,event);  
  
    /* 调用显示函数显示内容 */  
    display();  
}
```

# 如何使用button控件

- 所有的控件的创建和响应都在display函数中完成

```
void display()
{
    int w = 80, h = 22, x = 0, y = winheight;
    button(GenUIID(0), x, y, w, h, "OK");
    button(GenUIID(0), x += 100, y, w, h, "Cancel");
    if (button(GenUIID(0), x += 100, y, w, h, "Quit"))
        exit(-1);
}
```

- 调用button函数创建一个按钮
- 根据返回值判断用户是否点击了该按钮，并进行相应处理。

# 关于宏 GenUIID

```
#define GenUIID(N) ( ((__LINE__<<16) | ( N & 0xFFFF))^((long)&__FILE__) )
```

- GenUIID(k), 在编译时计算生成一个**唯一号**。计算时使用
  - 参数k
  - 宏调用所在的 **文件名**
  - 宏调用所在的 **行号**
  - 宏调用时的**参数 k**
- 用法 1: GenUIID(0)
  - 如果一行代码只产生一个唯一ID

- 用法 2: GenUIID(k)
  - 如果需要在同一行代码产生多个不同的唯一ID。例如:

```
for( k = 0; k<3; k++ )  
{  
    button(GenUIID(k), x, y-k*40, w, h, names[k]);  
}
```

用for循环创建三个按钮，纵向排列，标签为names[k]

# 如何使用menuList控件

- 在display函数中完成menu控件的创建和响应

1. 定义菜单选项字符串

```
char * menuListFile[ ] = {"File",  
"Open  | Ctrl-O",  
"Close",  
"Exit  | Ctrl-E" };
```

2. 绘制和处理菜单

```
selection = menuList(GenUIID(0), x, y, w, wsub, h, menuListFile,  
    sizeof(menuListFile)/sizeof(menuListFile[0]));  
if( selection==3 ) // choose to the menu of exit  
    exit(-1); // act on the selection
```

- 用户可以用鼠标选择菜单，也可以用快捷键
  - 快捷键在选项字符串中给出
  - 快捷键必须是Ctrl-X形式，而且位于字符串的结尾部分

# 如何使用menuList控件 - continued

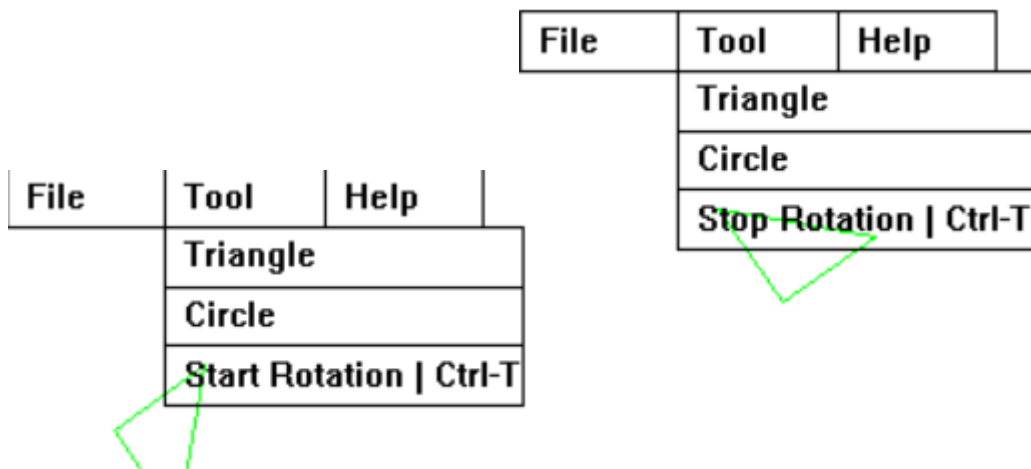
- 控件menuList介绍

- x, y - 菜单左上角坐标
- w - 类别标签的显示宽度
- wlist - 菜单选项的显示宽度
- h - 菜单项的显示高度
- labels - labels[0]是菜单的类别名
  - labels[1.....n-1]是该类别菜单选项标签
  - 其中可以包含快捷键
- n - labels中标签字符串的个数

```
int menuList(int id,  
             double x,  
             double y,  
             double w,  
             double wlist,  
             double h,  
             char *labels[],  
             int n);
```

# 如何使用menuList控件 - continued

- 设置动态可变菜单标签



- 根据实际情况设置合适的标签

```
static int show_more_buttons = 0;
char * menuListTool[] = {"Tool",
    "Triangle",
    "Circle",
    "Stop Rotation | Ctrl-T"};
// 设置动态可变菜单标签
menuListTool[3] = enable_rotation ?
    "Stop Rotation | Ctrl-T" :
    "Start Rotation | Ctrl-T";

selection = menuList(GenUIID(0), x+w, y, w,wlist,h,
    menuListTool,sizeof(menuListTool)/
    sizeof(menuListTool[0]));

if( selection==3 )
    enable_rotation = ! enable_rotation;
```


详见demoGuiMenu.c或demoGuiALL.c



# 如何使用menuList控件 - continued

- 菜单的快捷键
  - 在标签结尾设置，例如

File	Tool	Help
	Triangle	
	Circle	
	Stop Rotation   Ctrl-T	



右侧菜单列表的选项“Stop Rotation”的快捷键是

Ctrl-T (同时按下Control键和字符键 t

那么 我们将Ctrl-T添加到标签的末尾

- 注意必须在末尾

```
static int show_more_buttons = 0;  
char * menuListTool[] = {"Tool",  
    "Triangle",  
    "Circle",  
    "Stop Rotation | Ctrl-T"};
```

- 为了使用快捷键，还需要调用simpleGUI的getKeyboard函数
  - 具体见有关textbox的讲解

# 如何使用textbox控件，编辑字符串

- 首先记录鼠标和键盘输入
  - 编写鼠标事件回调函数
    - MouseEventProcess
    - 调用 uiGetMouse
  - 编写键盘事件回调函数
    - KeyboardEventProcess
    - 调用 uiGetKeyboard
  - 编写字符串事件回调函数
    - CharEventProcess
    - 调用 uiGetChar

```
void CharEventProcess(char ch)
{
    uiGetChar(ch); /*获取字符*/
    display(); /*更新显示*/
}

void KeyboardEventProcess(int key, int event)
{
    uiGetKeyboard(key, event); /*获取键盘*/
    display(); /*更新显示*/
}


void MouseEventProcess(int x, int y,
                       int button, int event)
{
    uiGetMouse(x, y, button, event); /*获取鼠标*/
    display(); /*更新显示*/
}
```

# 如何使用textbox控件， 编辑字符串 - I

- 在display函数中完成textbox控件的创建和编辑

```
static char str[80] = "Click and Edit";
```

```
textbox(GenUIID(0), x, y, w, h, str, sizeof(str));
```

- 运行效果：
- 如果由多个textbox， 用户可以用Tab和Shift+Tab在他们之间轮转
- 还可以根据textbox返回值判断用户是否进行了编辑

# 如何使用textbox控件， 编辑字符串 - II

- 在display函数中完成textbox控件的创建和编辑

姓名

Text edit result is: Xing+Liu

```
static char firstName[80] = "Xinguo";
static char lastName[80] = "Liu";
static char results[256] = "";

if( textbox(GenUIID(0), x+20, y, w, h, firstName, sizeof(firstName)) )
    sprintf(results, "%Text edit result is: %s+%s", firstName, lastName);

if( textbox(GenUIID(0), x+20+5+w, y, 30, h, lastName, sizeof(lastName)) )
    sprintf(results, "%Text edit result is: %s+%s", firstName, lastName);

SetPenColor("Red");
drawLabel(x, y-20, results); // 显示结果
```

# simpleGUI控件的颜色设置

- 调用下面的函数，使用预定义的颜色组合

```
void usePredefinedColors(int k);  
void usePredefinedButtonColors(int k);  
void usePredefinedMenuColors(int k);  
void usePredefinedTextBoxColors(int k);
```

- 函数usePredefinedColors会对button/menu/textbox三种类型全部进行设置
- 而其他的三个函数对button/menu/textbox分别进行设置

# 设置自定义的颜色组合

```
void setButtonColors (char *frame, char*label, char *hotFrame, char *hotLabel, int fillflag);  
void setMenuColors   (char *frame, char*label, char *hotFrame, char *hotLabel, int fillflag);  
void setTextBoxColors(char *frame, char*label, char *hotFrame, char *hotLabel, int fillflag);
```

- 功能

1. setButtonColors - 设置按钮颜色
2. setMenuColors - 设置菜单颜色
3. setTextBoxColors - 设置编辑框颜色

- 参数

1. frame/label - 控件框/文字标签的颜色
2. hotFrame/hotLabel - 鼠标划过时，控件框/文字标签的颜色
3. fillflag - 是否填充背景。0 - 不填充，1 - 填充

- 当某个参数字符串为空时，对应的颜色不做改变
- 颜色设置是状态变量，会影响之后绘制的控件

# simpleGUI其他辅助画图函数 - 1

- 画一个矩形 (x,y,w,h)

```
void drawRectangle(double x, double y, double w, double h,  
                  int fillflag);
```

fillflag 是填充与否的标志

1 - 填充

0 - 不填充

# simpleGUI其他辅助画图函数 - 2

- 同时画矩形和标签字符串

```
void drawBox(double x, double y,  
             double w, double h, int fillflag,  
             char *label, char xalignment,  
             char *labelColor);
```

- xalignment – 指定标签和矩形的对齐方式
  - 'L' - 靠左
  - 'R' - 靠右
  - 其他- 居中
- labelColor – 指定标签的颜色名



# 结束

下载 LibGraphics2019New, 运行demo程序