图形界面开发(ofxGui, ofxUI)

【开始：两种方法新建使用插件的工程】

方法 1：新建 OF 工程时就加入扩展库

使用插件的最简单方法是在用 projectGenerator 新建工程时直接添加相应的插件。

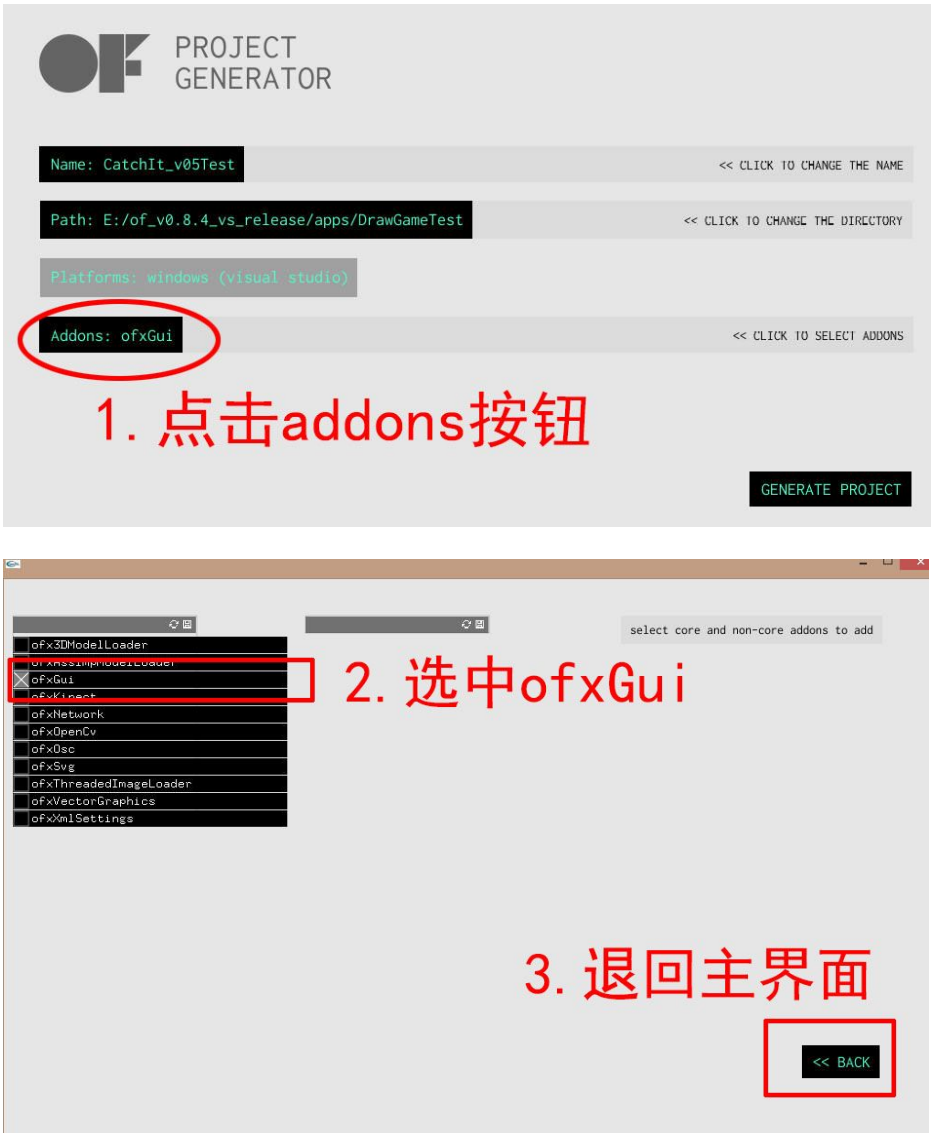


图 1 在新建工程时添加 ofxGui

选好 ofxGui 后，新建的工程中会加入 ofxGui 的所有代码文件。

方法 2：在已有工程中使用扩展库

最简单的办法就是直接从 addons\ofxGui\src 文件夹下将所有文件拷贝到当前工程文件夹下的 src 目录中，然后将所有文件添加到工程中来。但更通常的做法是直接通过工程设置来添加 ofxGui 的源码文件。首先，在 VS 的解决方案资源管理器中，选中当前工程，并选择右键菜单中的属性按钮，进入了当前工程的设置界面，然后在子菜单“c/c++->附加包含

目录”中，增加 ofxGui 的源文件目录，其操作过程如后图所示，最后，要在工设置界面点击“应用”并“确定”。为了让工程拷贝到另外电脑后也能运行，可以在添加 ofxGui 源码目录的时候将其从绝对目录位置的写法改为广义的写法” ..\..\..\addons\ofxGui\src “。

图 2 通过”右键菜单->属性”进入工程设置界面

图 3 选择“C/C++->附加包含目录”

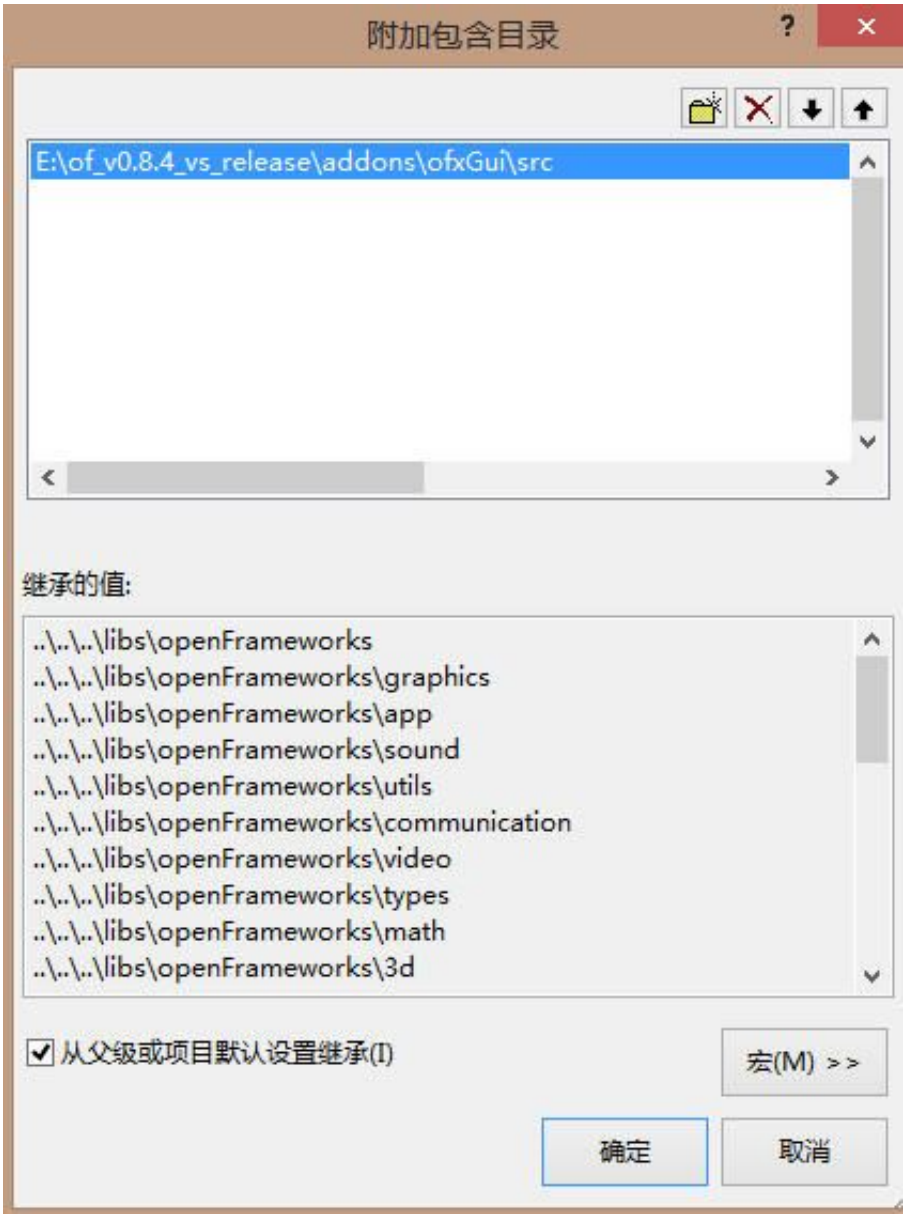


图 4 加入 ofxGui\src 文件夹

设置好后，ofxGui 的源码目录便成为本工程能够识别的目录，但还需要将所有代码添加到工程中。添加源文件的操作与新建源码文件的方法类似，只是要选择添加现有项，将 ofxGui\src 目录下的所有源码文件加入（图 5）。为了更好的整理代码文件，在解决方案资源管理器中，可在工程中新建一个目录（右键菜单->添加->新建筛选器），将 ofxGui 的所有代码文件放到此目录下，与之前的所有代码文件区分开（图 6）。

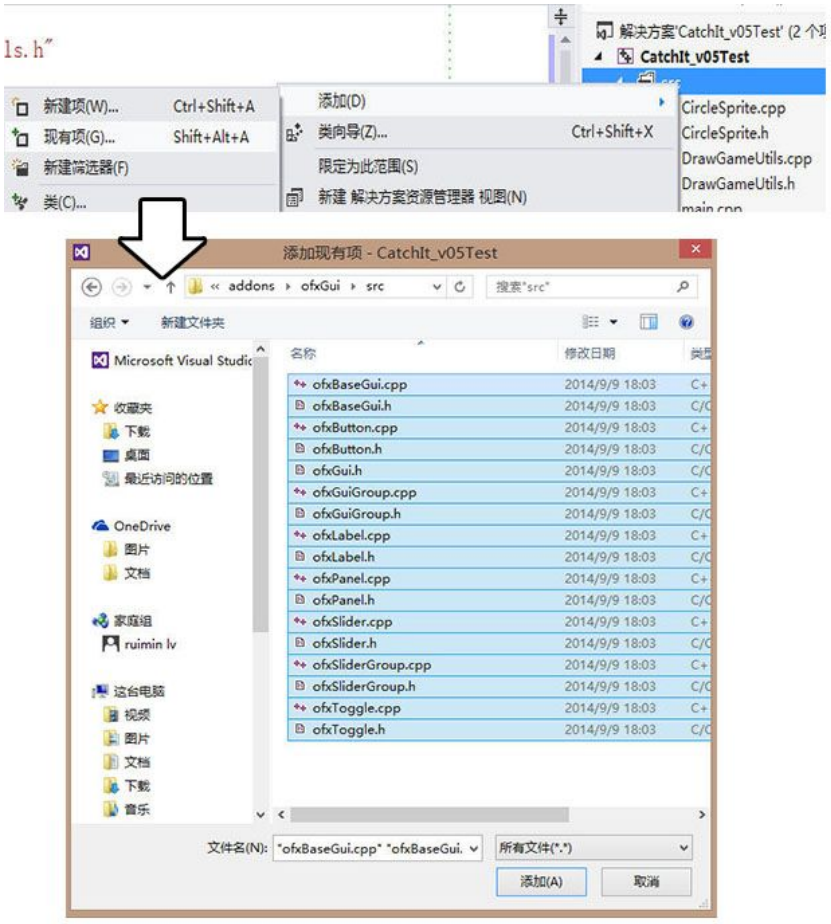


图 5 添加 ofxGui 的所有代码文件

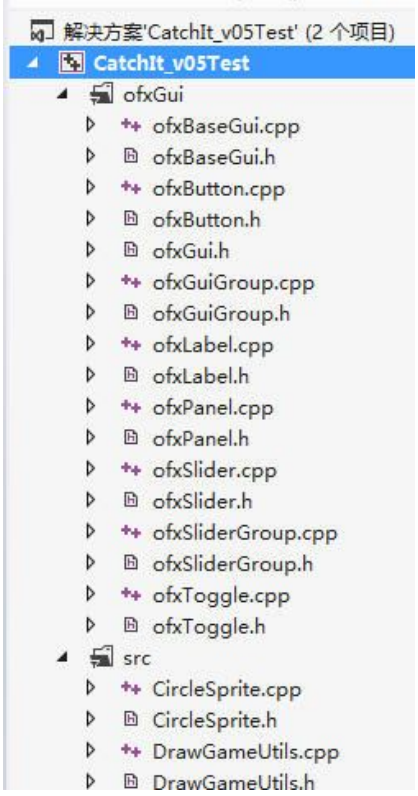


图 6 在工程中将 ofxGui 的文件放入新建的目录中

于是，ofxGui 扩展库便成功地添加到当前的工程中了。其中，头文件 ofxGui.h 包含了这个扩展库的所有接口定义，于是在之后的程序编写过程中，一旦需要用到 ofxGui 的功能，则需要在源码文件开头部分写上宏命令”#include “ofxGui.h”。

---

ofxGui

【介绍】

ofxGui 库提供了一些最常见的图形界面类型，包括面板、触发按钮、开关按钮、标签、滑动条等，界面可以展开和收缩，下面通过 OF 自带的示例程序文件夹 (OFRoot)\examples\gui\下多个示例来学习 ofxGui

【例 1：guiExample】

学习按钮，滑动条，标签的使用。

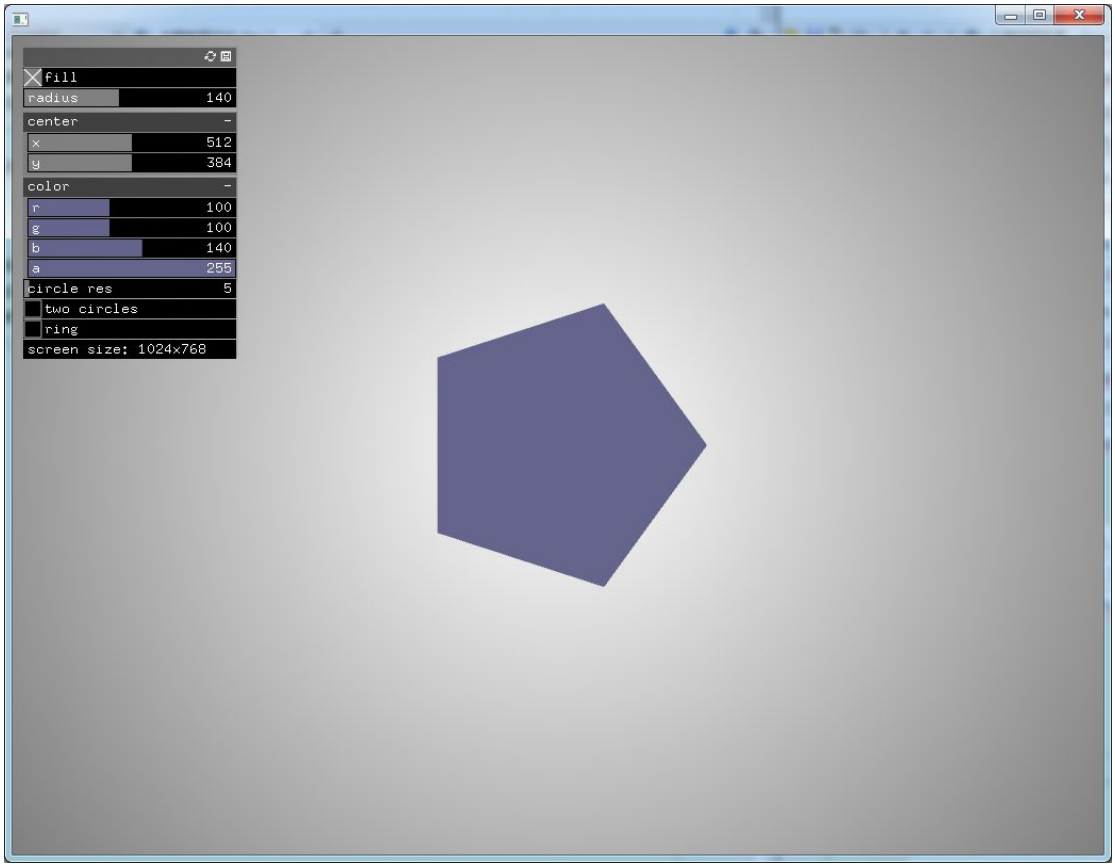


图 7 guiExample 程序运行图

先来介绍一下这个程序，第一个 fill 切换按钮可以选择是否填充形状；radius 滑动条可以调节形状的半径；x，y 滑动条可以移动形状的中心坐标；r，g，b，a 滑动条可以调节形状的 RGB 值及透明度；circle res 滑动条控制形状的边数；点下 two circles 按钮可以使形状分裂成两个相同的形状；点击 ring 按钮会发出铃声；screen size 标签显示程序窗口的大小；按 h 键可以隐藏或显示面板，s 键保存当前面板数据，l 载入之前保存的数据，空格键将颜色 rgba 值设为 255，255，255，255。

然后我们探索如何实现这些功能。

1.头文件添加

```
bool bHide;//控制是否显示面板

ofxFloatSlider radius;//float 类型的滑动条，控制形状的半径
ofxColorSlider color;//color 类型的滑动条，含 rgba，控制形状的颜色与透明值
ofxVec2Slider center;//vec2 类型的滑动条，含 x，y，控制形状的中心位置
ofxIntSlider circleResolution;//int 类型的滑动条，控制形状的边数
ofxToggle filled;//切换按钮，控制形状是否填色
ofxButton twoCircles;//按钮，点下分裂成两个形状，弹起恢复
ofxButton ringButton;//按钮，点击发出铃声
ofxLabel screenSize;//标签，显示程序窗口大小

ofxPanel gui;//总面板

ofSoundPlayer ring;//铃声
```

2.源文件实现

①setup 函数，更新程序前的初始化函数

```
void ofApp::setup(){
    //降低帧频率，在动作很快的时候会错位，去掉的话会影响更新，和 ofSetFramarate()差不多
    ofSetVerticalSync(true);

    //在开始前添加监听器，确保初始化圆的分辨率（边数）是正确的
    //当这两个变量有改动的话，就触发后面的函数
    circleResolution.addListener(this, &ofApp::circleResolutionChanged);
    ringButton.addListener(this,&ofApp::ringButtonPressed);

    //面板的初始化设置
    gui.setup(); //  基本上不需要设置名字
    //面板中添加 filled 切换按钮，命名为 fill，初始化为 true
    gui.add(filled.setup("fill", true));
    //面板中添加 radius 滑动条，命名为 radius，默认值 140，最小值 10，最大值 300
    gui.add(radius.setup( "radius", 140, 10, 300 ));
    //面板中添加 center 滑动条，命名为 center，默认窗口屏幕中心，最小左上角，最大
    //右下角
    gui.add(center.setup("center",ofVec2f(ofGetWidth()*.5,ofGetHeight()*.5),ofVec2
```

```
f(0,0),ofVec2f(ofGetWidth(),ofGetHeight())));
    //面板中添加 color 滑动条，命名为 color，默认 rgba 值为 100，100，140，255，
    //最小 rgba 值全为 0，最大 rgba 值全为 255
    gui.add(color.setup("color",ofColor(100,100,140),ofColor(0,0),ofColor(255,255)))
;
    //面板中添加 circleResolution 滑动条，命名 circle res，默认 5，最小 3，最大 90
    gui.add(circleResolution.setup("circle res", 5, 3, 90));
    //面板中添加 twoCircles 切换按钮，命名为 two circles
    gui.add(twoCircles.setup("two circles"));
    //面板中添加 ringButton 按钮，命名 ring
    gui.add(ringButton.setup("ring"));
    //面板中添加 screenSize 标签，命名 screen size，初始化为空
    gui.add(screenSize.setup("screen size", ""));

    //是否隐藏的布尔值为 true
    bHide = true;

    //加载 ring 铃声音频函数
    ring.loadSound("ring.wav");
}
```

②update 函数，按一定频率更新

```
void ofApp::update(){
    ofSetCircleResolution(circleResolution);//圆的分辨率状态声明，控制边数
}
```

③draw 函数

```
void ofApp::draw(){
    ofBackgroundGradient(ofColor::white, ofColor::gray);//背景是白到灰的梯度渐变

    if( filled ){
        ofFill();//选择填充模式
    }else{
        ofNoFill();//选择不填充模式
    }

    ofSetColor(color);//设置颜色
    if(twoCircles){//如果 twoCircles 为 true，则画两个圆
        ofCircle(center->x-radius*.5, center->y, radius );
        ofCircle(center->x+radius*.5, center->y, radius );
```

```
    }else{
        ofCircle((ofVec2f)center, radius );//画圆，center 为中心，radius 为半径
    }
    if( bHide ){
        gui.draw();//若 bHide 为 true，则画 gui 面板
    }
}
```

④keyPressed 函数，键盘按下事件

```
void ofApp::keyPressed(int key){
    if( key == 'h' ){
        bHide = !bHide;//按下 h，切换 gui 面板是否显示
    }
    if(key == 's') {
        gui.saveToFile("settings.xml");//按下 s，保存当前面板数据到 setting.xml 文件
    }
    if(key == 'l') {
        gui.loadFromFile("settings.xml");//按下 l，从 setting.xml 文件读取数据
    }
    if(key == ' '){
        color = ofColor(255);//按下空格，将 rgba 值设置全为 255
    }
}
```

⑤其他的函数

```
//用当前窗口大小设置 screenSize 标签
void ofApp::windowResized(int w, int h){
    screenSize = ofToString(w) + "x" + ofToString(h);
    }
//退出时删除 ringButton 的监听器
void ofApp::exit(){
    ringButton.removeListener(this,&ofApp::ringButtonPressed);
}
//监听函数，circleResolution 一改变就重新设定
void ofApp::circleResolutionChanged(int & circleResolution){
    ofSetCircleResolution(circleResolution);
}
//监听函数，ringButton 一按下就播放 ring 音频
void ofApp::ringButtonPressed(){
    ring.play();
```

}
通过例 1 学习了包括面板、触发按钮、开关按钮、标签、滑动条等组件的使用。

【例 2：guiFromParametersExample】
例 2 实现了与例 1 相同的程序，但例 2 使用了类模板 ofParameter 表达参数。
先来介绍一下类模板 ofParameter：OF 提供了一个模板类 ofParameter，可以方便地实现将参数显示到由 ofxGui 实现的图形界面上，并易于实现 xml 格式的数据保存与读取。ofParameter 是个类模板，它概括了“参数“的一些特色：设置/读取数值、取值范围、监听函数等。
1.头文件

```
bool bHide;
ofParameter<float> radius;
ofParameter<ofColor> color;
ofParameter<ofVec2f> center;
ofParameter<int> circleResolution;
ofParameter<bool> filled;
ofxButton twoCircles;
ofxButton ringButton;
ofParameter<string> screenSize;

ofxPanel gui;

ofSoundPlayer ring;
//可以看到面板中 radius，color，center，circleResolution，filled，screenSize 参数都使用了类模板 ofParameter
```

2.源文件与例 1 相同

例 2 示意了 ofParameter 与 ofxGui 联合使用的典型用法。在使用中，需注意以下要点：

- ofParameter 类模板支持绝大多数 C++原生的数据类型和 OF 提供的新数据类型；
- ofxPanel 是 ofxGui 库中的类型，使用时必须包含 ofxGui.h 头文件；
- ofxPanel 用于组织一系列控件，它能够识别多种 ofParameter 类型，自动运用匹配的 GUI 控件类型，但 ofxPanel 对象上能够识别并添加的 ofParameter 类型有限，只能识别上述程序中列出的数据类型；
- ofParameter 的在初始化时需要输入四个形式参数：名称，初始值，最小值和最大值；

【增加了 ofParameterGroup 的例 2】
例 2 使用了类模板 ofParameter 表达参数，也可以把参数放在参数组 ofParameterGroup 里，ofParameterGroup 用于组织一系列参数或子参数组。

1.头文件添加

```
ofParameterGroup Params;
```

2.将原来的 setup 函数修改

```
void ofApp::setup(){
    ofSetVerticalSync(true);

    circleResolution.addListener(this, &ofApp::circleResolutionChanged);
    ringButton.addListener(this,&ofApp::ringButtonPressed);

    Params.setName("parameters");
    Params.add(filled.set("bFill", true));
    Params.add(radius.set( "radius", 140, 10, 300 ));
    Params.add(center.set("center",ofVec2f(ofGetWidth()*.5,ofGetHeight()*.5),ofVe
c2f(0,0),ofVec2f(ofGetWidth(),ofGetHeight())));
    Params.add(color.set("color",ofColor(100,100,140),ofColor(0,0),ofColor(255,255)
));
    Params.add(circleResolution.set("circleRes", 5, 3, 90));
    Params.add(screenSize.set("screenSize", ""));

    gui.setup("panel");
    gui.add(Params);
    gui.add(twoCircles.setup("twoCircles"));
    gui.add(ringButton.setup("ring"));

    bHide = true;

    ring.loadSound("ring.wav");
}
```

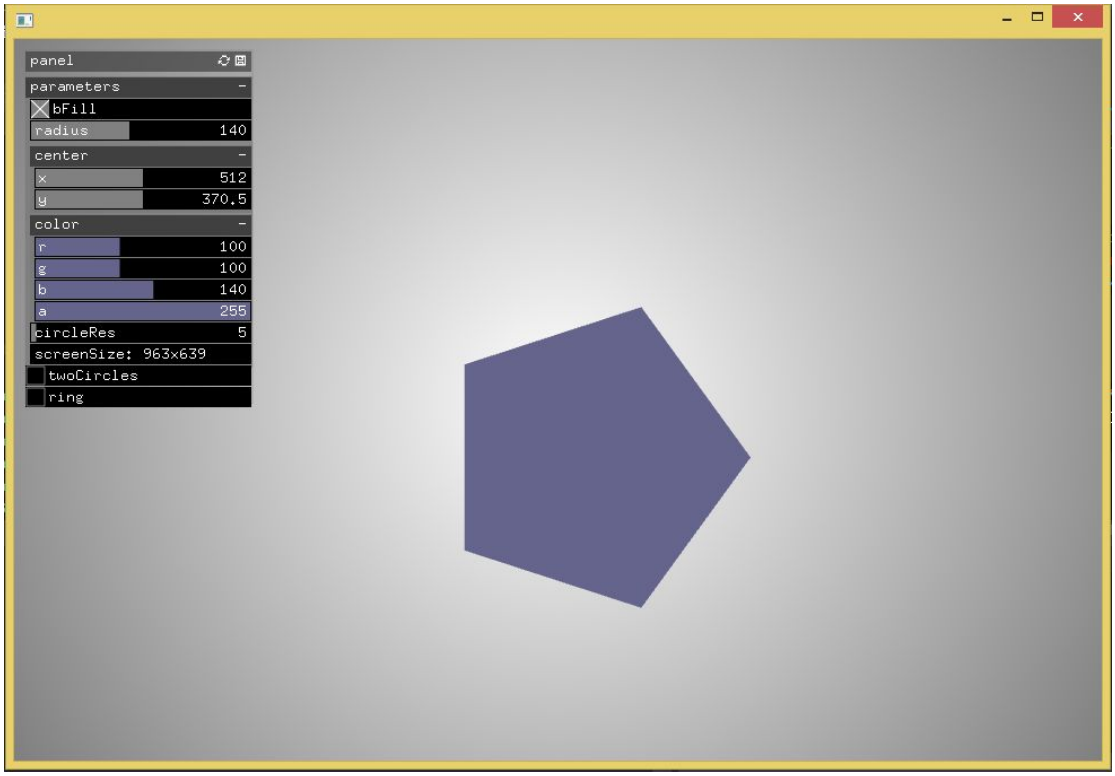


图 8 可以看到面板并没有什么不同，改变了一下参数的顺序。

```
<panel>
    <bFill>1</bFill>
    <radius>140</radius>
    <center>512, 384</center>
    <color>100, 100, 140, 255</color>
    <circleRes>5</circleRes>
    <screenSize>1024x768</screenSize>
<panel/>
        <parameters>
                <bFill>1</bFill>
                <radius>140</radius>
                <center>512, 370.5</center>
                <color>195, 100, 140, 255</color>
                <circleRes>5</circleRes>
                <screenSize>898x581</screenSize>
        </parameters>
</panel>
```

图 9 可以看到 Setting.xml 文件中增加了 params 的参数值

ofxUI

ofxUI 库：https://github.com/rezaali/ofxUI

【简介】

ofxUI 是 openFrameworkss 的一个插件，用户可以很容易地进行界面编辑。ofxUI 同时也提供布局，空间，字体载入，保存和载入，窗口部件回调功能。ofxUI 可以很容易的定制(颜色，字体，窗口大小，边距，布局等）。

ofxUI 包含了基本的图形用户界面的组件（按钮，触发器，图片按钮，标签按钮，按钮矩阵，

下拉菜单，滑动条，数据表，2D Pads，标签，FPS 标签，文字输入区）。

下面让我们通过实例来学习 ofxUI 的运用。

【例 1：example-AllWidgets】

学习各种控件的使用。

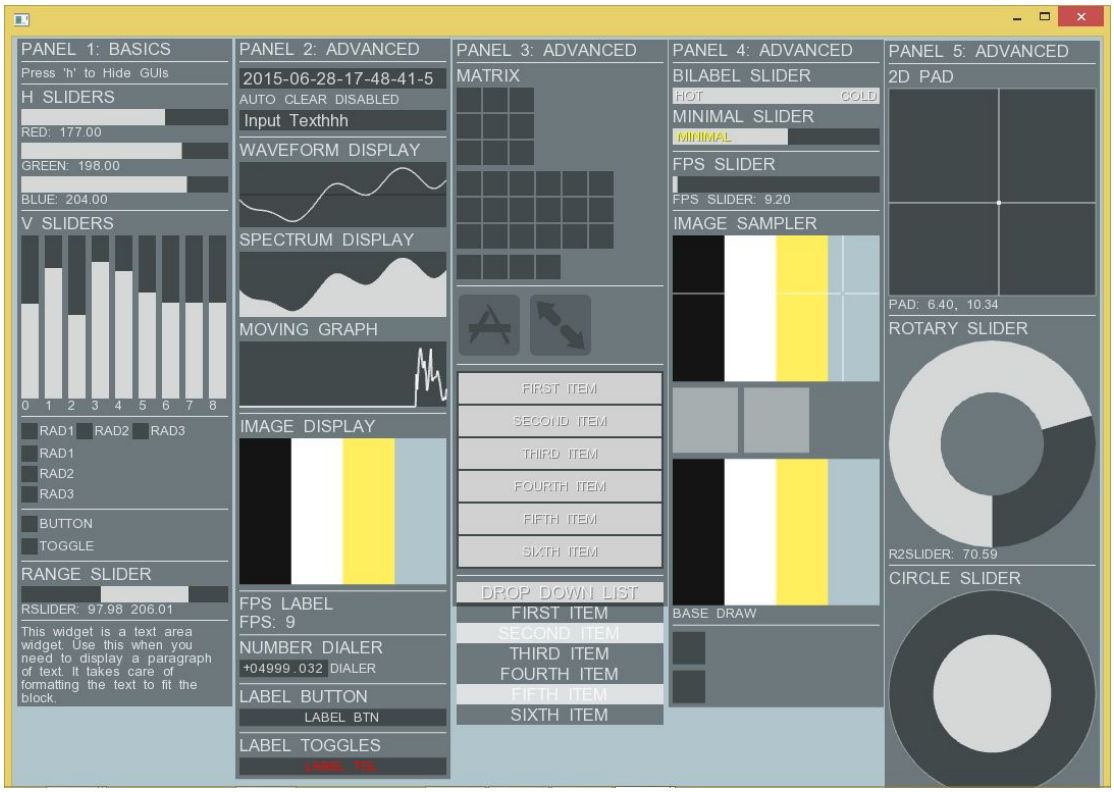


图 10 example-AllWidgets 的程序运行图

1.头文件

```
void exit();//退出函数
void drawGrid(float x, float y);//画网格函数
//设置面板 12345 函数
void setGUI1();
void setGUI2();
void setGUI3();
void setGUI4();
void setGUI5();
//分别对应 5 个面板
ofxUISuperCanvas *gui1;
ofxUISuperCanvas *gui2;
ofxUISuperCanvas *gui3;
ofxUISuperCanvas *gui4;
ofxUISuperCanvas *gui5;
ofxUITextInput *textInput; //文本输入区
```

```
bool hideGUI;//控制是否隐藏
float red, green, blue;//控制背景 rgb
bool bdrawGrid;//控制是否绘制网格
bool bdrawPadding;//控制是否显示衬底
void guiEvent(ofxUIEventArgs &e);//事件函数
ofxUIMovingGraph *mg;//移动曲线图
ofxUIDropDownList *ddl;//下拉菜单
ofxUIToggleMatrix *tm;//toggle 按钮矩阵
float *buffer;//缓冲
ofImage *img;//图片指针
```

2.源文件

①setup

```
void ofApp::setup(){
    ofSetCircleResolution(120);//设置圆的边数
    red = 233; blue = 233; green = 233;//设置初始化背景值
    hideGUI = false;//初始化界面可见
    bdrawGrid = false;//初始化网格不可见
    bdrawPadding = false;//初始化衬底不可见
    ddl = NULL;//下拉菜单指针初设为空
    textInput = NULL;//文本输入区指针初设为空
    img = new ofImage("nerd_me.png");//载入 nerd_me.png 图片到 img 中
    buffer = new float[256];//长度大小为 256 的 float 类型数组
    for(int i = 0; i < 256; i++) { buffer[i] = ofNoise(i/100.0); }

    setGUI1();//设置面板
    setGUI2();
    setGUI3();
    setGUI4();
    setGUI5();

    gui1->loadSettings("gui1Settings.xml");//从文件中分别载入 5 组界面设定
    gui2->loadSettings("gui2Settings.xml");
    gui3->loadSettings("gui3Settings.xml");
    gui4->loadSettings("gui4Settings.xml");
    gui5->loadSettings("gui5Settings.xml");
}
void ofApp::setGUI1()//基础运用
{
```

```
    vector<string> names;//动态 string 类型数组中添加了 3 个参数
    names.push_back("RAD1");
    names.push_back("RAD2");
    names.push_back("RAD3");

    gui1 = new ofxUISuperCanvas("PANEL 1: BASICS");
    gui1->addSpacer();
    gui1->addLabel("Press 'h' to Hide GUIs", OFX_UI_FONT_SMALL);

    gui1->addSpacer();
    gui1->addLabel("H SLIDERS");
    //设置了不同的触发类型
    gui1->addSlider("RED", 0.0, 255.0,
&red)->setTriggerType(OFX_UI_TRIGGER_ALL);
    gui1->addSlider("GREEN", 0.0, 255.0,
&green)->setTriggerType(OFX_UI_TRIGGER_BEGIN|OFX_UI_TRIGGER_CHANGE|OFX
_UI_TRIGGER_END);
    gui1->addSlider("BLUE", 0.0, 255.0,
&blue)->setTriggerType(OFX_UI_TRIGGER_BEGIN|OFX_UI_TRIGGER_CHANGE);

    gui1->addSpacer();
    gui1->addLabel("V SLIDERS");
    gui1->addSlider("0", 0.0, 255.0, 150, 17, 160);
    //设置窗口方向
    gui1->setWidgetPosition(OFX_UI_WIDGET_POSITION_RIGHT);
    gui1->addSlider("1", 0.0, 255.0, 150, 17, 160);
    gui1->addSlider("2", 0.0, 255.0, 150, 17, 160);
    gui1->addSlider("3", 0.0, 255.0, 150, 17, 160);
    gui1->addSlider("4", 0.0, 255.0, 150, 17, 160);
    gui1->addSlider("5", 0.0, 255.0, 150, 17, 160);
    gui1->addSlider("6", 0.0, 255.0, 150, 17, 160);
    gui1->addSlider("7", 0.0, 255.0, 150, 17, 160);
    gui1->addSlider("8", 0.0, 255.0, 150, 17, 160);
    gui1->setWidgetPosition(OFX_UI_WIDGET_POSITION_DOWN);

    gui1->addSpacer();
    //单选框，水平排列
    gui1->addRadio("RADIO HORIZONTAL", names,
```

```
OFX_UI_ORIENTATION_HORIZONTAL);
//单选框，垂直排列
    gui1->addRadio("RADIO VERTICAL", names,
OFX_UI_ORIENTATION_VERTICAL);

    gui1->addSpacer();
    //设置了字体大小为小
    gui1->setWidgetFontSize(OFX_UI_FONT_SMALL);
    //button 和 toggle
    gui1->addButton("BUTTON", false);
    gui1->addToggle( "TOGGLE", false);

    gui1->addSpacer();
    gui1->addLabel("RANGE SLIDER");
    //范围滑动条，最小值，最大值
    gui1->addRangeSlider("RSLIDER", 0.0, 255.0, 50.0, 100.0);

    string textString = "This widget is a text area widget. Use this when you need to
display a paragraph of text. It takes care of formatting the text to fit the block.";
    gui1->addSpacer();
    //文本区显示 textString 文本，小字体
    gui1->addTextArea("textarea", textString, OFX_UI_FONT_SMALL);

    gui1->autoSizeToFitWidgets();
    ofAddListener(gui1->newGUIEvent,this,&ofApp::guiEvent);
}
void ofApp::setGUI2()//高阶应用
{
    gui2 = new ofxUISuperCanvas("PANEL 2: ADVANCED");

    gui2->addSpacer();
    gui2->setWidgetFontSize(OFX_UI_FONT_MEDIUM);
    //文本输入区
    textInput = gui2->addTextInput("TEXT INPUT", "Input Text");
    //设置文本输入区自动聚焦
    textInput->setAutoUnfocus(false);
    gui2->addLabel("AUTO CLEAR DISABLED", OFX_UI_FONT_SMALL);
    //不会自动清除文本内容
```

```
gui2->addTextInput("TEXT INPUT2", "Input Text")->setAutoClear(false);
gui2->setWidgetFontSize(OFX_UI_FONT_MEDIUM);

gui2->addSpacer();
gui2->addLabel("WAVEFORM DISPLAY");
//波形表格，数据，数据大小，最小值，最大值
gui2->addWaveform("WAVEFORM", buffer, 256, 0.0, 1.0);
gui2->addLabel("SPECTRUM DISPLAY");
//频谱，数据，数据大小，最小值，最大值
gui2->addSpectrum("SPECTRUM", buffer, 256, 0.0, 1.0);

vector<float> buffer;
for(int i = 0; i < 256; i++)
{
    buffer.push_back(0.0);//动态数组初始化为 0
}

gui2->addLabel("MOVING GRAPH", OFX_UI_FONT_MEDIUM);
//移动曲线图，数据，数据大小，最小值，最大值
mg = gui2->addMovingGraph("MOVING", buffer, 256, 0.0, 1.0);

gui2->addSpacer();
gui2->addLabel("IMAGE DISPLAY");
gui2->addImage("IMAGE CAPTION", img);//载入了一张图片

gui2->addSpacer();
gui2->addLabel("FPS LABEL");
gui2->addFPS();//显示帧率

gui2->setWidgetFontSize(OFX_UI_FONT_SMALL);
gui2->addSpacer();
gui2->addLabel("NUMBER DIALER");
//数字拨号器，最小值，最大值，初始值，精度
gui2->addNumberDialer("DIALER", -10000, 10000, 5000, 3);

gui2->addSpacer();
gui2->addLabel("LABEL BUTTON", OFX_UI_FONT_MEDIUM);
gui2->addLabelButton("LABEL BTN", false);
```

```
    gui2->addSpacer();
    gui2->addLabel("LABEL TOGGLES", OFX_UI_FONT_MEDIUM);
    gui2->addLabelToggle("LABEL TGL",
false)->getLabelWidget()->setColorFill(ofColor(255, 0, 0));

    gui2->setPosition(212, 0);
    gui2->autoSizeToFitWidgets();

    ofAddListener(gui2->newGUIEvent,this,&ofApp::guiEvent);
}
void ofApp::setGUI3()//高级应用
{
    gui3 = new ofxUISuperCanvas("PANEL 3: ADVANCED");

    gui3->addSpacer();
    gui3->setGlobalButtonDimension(24);
    gui3->addLabel("MATRIX", OFX_UI_FONT_MEDIUM);
    gui3->addToggleMatrix("MATRIX1", 3, 3);
    tm = gui3->addToggleMatrix("MATRIX2", 3, 6);
    gui3->addToggleMatrix("MATRIX3", 1, 4);

    gui3->addSpacer();
    gui3->setGlobalButtonDimension(64);
    gui3->addImageButton("IMAGEBTN", "GUI/images/App.png", false);
    gui3->setWidgetPosition(OFX_UI_WIDGET_POSITION_RIGHT);
    gui3->addImageToggle("IMAGETGL", "GUI/images/Preview.png", false);
    gui3->setWidgetPosition(OFX_UI_WIDGET_POSITION_DOWN);

    gui3->addSpacer();
    vector<string> items;
    items.push_back("FIRST ITEM");
    items.push_back("SECOND ITEM");
    items.push_back("THIRD ITEM");
    items.push_back("FOURTH ITEM");
    items.push_back("FIFTH ITEM");
    items.push_back("SIXTH ITEM");
    gui3->setWidgetFontSize(OFX_UI_FONT_SMALL);
```

```
    gui3->addSortableList("SORTABLE LIST", items);//可排序列表

    gui3->addSpacer();
    gui3->setWidgetFontSize(OFX_UI_FONT_MEDIUM);
    ddl = gui3->addDropDownList("DROP DOWN LIST", items);//下拉菜单
    ddl->setAllowMultiple(true);//设置允许多触发

    gui3->setGlobalButtonDimension(OFX_UI_GLOBAL_BUTTON_DIMENSION);

    gui3->setPosition(212*2, 0);
    gui3->autoSizeToFitWidgets();

    ofAddListener(gui3->newGUIEvent,this,&ofApp::guiEvent);
}
void ofApp::setGUI4()//高阶应用
{
    gui4 = new ofxUISuperCanvas("PANEL 4: ADVANCED");

    gui4->addSpacer();
    gui4->addLabel("BILABEL SLIDER");
    //标签滑动条，左边标签 HOT，右边标签 COLD，最小 0，最大 100，初始化为 50
    gui4->addBiLabelSlider("BILABEL", "HOT", "COLD", 0, 100, 50);

    gui4->addLabel("MINIMAL SLIDER");
    //最低限度滑动条，MINIMAL 标签，最小 0，最大 100，初始 50，设置颜色黄色
    gui4->addMinimalSlider("MINIMAL", 0, 100,
50.0)->getLabelWidget()->setColorFill(ofColor(255, 255, 0));

    gui4->addSpacer();
    gui4->addLabel("FPS SLIDER", OFX_UI_FONT_MEDIUM);
    //FPS 滑动条
    gui4->addFPSSlider("FPS SLIDER");

    gui4->addSpacer();
    gui4->addLabel("IMAGE SAMPLER", OFX_UI_FONT_MEDIUM);
    //图片采样器
    gui4->addImageSampler("SAMPLER", img);
    gui4->setGlobalButtonDimension(64);
```

```
    gui4->addMultiImageButton("IMAGE BUTTON", "GUI/toggle.png", false);
    gui4->setWidgetPosition(OFX_UI_WIDGET_POSITION_RIGHT);
    gui4->addMultiImageToggle("IMAGE TOGGLE", "GUI/toggle.png", false);
    gui4->setWidgetPosition(OFX_UI_WIDGET_POSITION_DOWN);

    gui4->addBaseDraws("BASE DRAW", img, true);

    gui4->addSpacer();
    gui4->setGlobalButtonDimension(32);
    gui4->addButton("BTN", false)->setLabelVisible(false);
    gui4->addToggle("TGL", false)->setLabelVisible(false);

    gui4->setPosition(212*3,0);
    gui4->autoSizeToFitWidgets();

    ofAddListener(gui4->newGUIEvent,this,&ofApp::guiEvent);
}
void ofApp::setGUI5()//进阶应用
{
    gui5 = new ofxUISuperCanvas("PANEL 5: ADVANCED");
    gui5->addSpacer();

    gui5->addLabel("2D PAD");
    gui5->add2DPad("PAD", ofPoint(-100, 100), ofPoint(-100,100), ofPoint(0,0));

    gui5->addSpacer();
    gui5->addLabel("ROTARY SLIDER", OFX_UI_FONT_MEDIUM);
    //旋转滑动条，最小 0，最大 100，初始 50
    gui5->addRotarySlider("R2SLIDER", 0, 100, 50);

    gui5->addSpacer();
    gui5->addLabel("CIRCLE SLIDER", OFX_UI_FONT_MEDIUM);
    //圆形滑动条，最小 0，最大 100，初始 50
    gui5->addCircleSlider("NORTH SOUTH", 0, 100, 50.0);

    gui5->setPosition(212*4,0);
    gui5->autoSizeToFitWidgets();
```

```
    ofAddListener(gui5->newGUIEvent,this,&ofApp::guiEvent);
}
```
②update&draw
```
void ofApp::update(){
    mg->addPoint(buffer[0]);
    for(int i = 0; i < 256; i++) { buffer[i] = ofNoise(i/100.0, ofGetElapsedTimef()); }
}
void ofApp::draw(){
    ofBackground(red, green, blue, 255);//背景

    if(bdrawGrid)
    {
        ofSetColor(255, 255, 255, 25);//设置颜色画网格
        drawGrid(8,8);
    }
}
void ofApp::drawGrid(float x, float y)
{
    float w = ofGetWidth();
    float h = ofGetHeight();

    for(int i = 0; i < h; i+=y)
    {
        ofLine(0,i,w,i);
    }

    for(int j = 0; j < w; j+=x)
    {
        ofLine(j,0,j,h);
    }
}
```
③事件函数
```
void ofApp::guiEvent(ofxUIEventArgs &e)
{
    string name = e.getName();
    int kind = e.getKind();
    cout << "got event from: " << name << endl;
    if(kind == OFX_UI_WIDGET_NUMBERDIALER)
```

```
    {
        ofxUINumberDialer *n = (ofxUINumberDialer *) e.widget;
        cout << n->getValue() << endl;
    }

    if(name == "SAMPLER")
    {
        ofxUIImageSampler *is = (ofxUIImageSampler *) e.widget;
        ofColor clr = is->getColor();
        red = clr.r;
        blue = clr.b;
        green = clr.g;
    }
    else if(name == "BUTTON")
    {
        ofxUIButton *button = (ofxUIButton *) e.getButton();
        bdrawGrid = button->getValue();
    }
    else if(name == "TOGGLE")
    {
        ofxUIToggle *toggle = (ofxUIToggle *) e.getToggle();
        bdrawGrid = toggle->getValue();
        if(textInput != NULL)
        {
            textInput->setFocus(bdrawGrid);
        }
    }
    else if(name == "RADIO VERTICAL")
    {
        ofxUIRadio *radio = (ofxUIRadio *) e.widget;
        cout << radio->getName() << " value: " << radio->getValue() << " active
name: " << radio->getActiveName() << endl;
    }
    else if(name == "TEXT INPUT")
    {
        ofxUITextInput *ti = (ofxUITextInput *) e.widget;
        if(ti->getInputTriggerType() == OFX_UI_TEXTINPUT_ON_ENTER)
        {
```

```
            cout << "ON ENTER: ";
        }
        else if(ti->getInputTriggerType() == OFX_UI_TEXTINPUT_ON_FOCUS)
        {
            cout << "ON FOCUS: ";
        }
        else if(ti->getInputTriggerType() == OFX_UI_TEXTINPUT_ON_UNFOCUS)
        {
            cout << "ON BLUR: ";
        }
        string output = ti->getTextString();
        cout << output << endl;
    }
}
```

④keyPressed

```
void ofApp::keyPressed(int key){
    if(gui2->hasKeyboardFocus())
    {
        return;
    }
    switch (key)
    {
        case 't'://t 键更新文件输入区的文本
        {
            if(textInput != NULL)
            {
                textInput->setTextString(ofGetTimestampString());
            }
        }
            break;

        case 'T':
        {
            if(tm != NULL)
            {
                int cols = tm->getColumnCount();
                int rows = tm->getRowCount();
                for(int row = 0; row < rows; row++)
```

```
                {
                    for(int col = 0; col < cols; col++)
                    {
                        cout << tm->getState(row, col) << "\t";
                    }
                    cout << endl;
                }
            }
        }
            break;

        case 'd':
        {
            if(ddl != NULL)
            {
                vector<ofxUIWidget *> selected = ddl->getSelected();
                for(vector<ofxUIWidget *>::iterator it = selected.begin(); it !=
selected.end(); ++it)
                {
                    ofxUILabelToggle *lt = (ofxUILabelToggle *) (*it);
                    cout << lt->getName() << endl;
                }
            }
        }
            break;

        case 'D':
        {
            if(ddl != NULL)
            {
                vector<string> names = ddl->getSelectedNames();
                for(vector<string>::iterator it = names.begin(); it != names.end();
++it)
                {
                    cout << (*it) << endl;
                }
            }
        }
```

```
        break;

    case 'r'://r 键聚焦文本输入区
    {
        if(textInput != NULL)
        {
            textInput->setFocus(!textInput->isFocused());
        }
    }
        break;

    case 'f'://控制全屏
        ofToggleFullscreen();
        break;

    case 'F':
    {
        if(tm != NULL)
        {
            tm->setDrawOutlineHighLight(!tm->getDrawOutlineHighLight());
//              tm->setDrawPaddingOutline(!tm->getDrawPaddingOutline());
        }
    }
        break;

    case 'h'://控制是否显示界面
        gui1->toggleVisible();
        gui2->toggleVisible();
        gui3->toggleVisible();
        gui4->toggleVisible();
        gui5->toggleVisible();
        break;

    case 'p'://p 键控制是否显示衬底
        bdrawPadding = !bdrawPadding;
        gui1->setDrawWidgetPaddingOutline(bdrawPadding);
        gui2->setDrawWidgetPaddingOutline(bdrawPadding);
        gui3->setDrawWidgetPaddingOutline(bdrawPadding);
```

```
        gui4->setDrawWidgetPaddingOutline(bdrawPadding);
        gui5->setDrawWidgetPaddingOutline(bdrawPadding);
        break;

    case '[':
        gui1->setDrawWidgetPadding(false);
        gui2->setDrawWidgetPadding(false);
        gui3->setDrawWidgetPadding(false);
        gui4->setDrawWidgetPadding(false);
        gui5->setDrawWidgetPadding(false);
        break;

    case ']':
        gui1->setDrawWidgetPadding(true);
        gui2->setDrawWidgetPadding(true);
        gui3->setDrawWidgetPadding(true);
        gui4->setDrawWidgetPadding(true);
        gui5->setDrawWidgetPadding(true);
        break;

    //12345 控制界面 12345 的显示隐藏
    case '1':
        gui1->toggleVisible();
        break;

    case '2':
        gui2->toggleVisible();
        break;

    case '3':
        gui3->toggleVisible();
        break;

    case '4':
        gui4->toggleVisible();
        break;

    case '5':
```

```
                gui5->toggleVisible();
                break;

            default:
                break;
        }
}
```

⑤exit

```
void ofApp::exit()//退出函数，保存界面设定到文件，清除内存
{
    gui1->saveSettings("gui1Settings.xml");
    gui2->saveSettings("gui2Settings.xml");
    gui3->saveSettings("gui3Settings.xml");
    gui4->saveSettings("gui4Settings.xml");
    gui5->saveSettings("gui5Settings.xml");

    delete gui1;
    delete gui2;
    delete gui3;
    delete gui4;
    delete gui5;
    delete[] buffer;
    delete img;
}
```

【例 2：example-ContextualMenu】

学习右键菜单的运用。

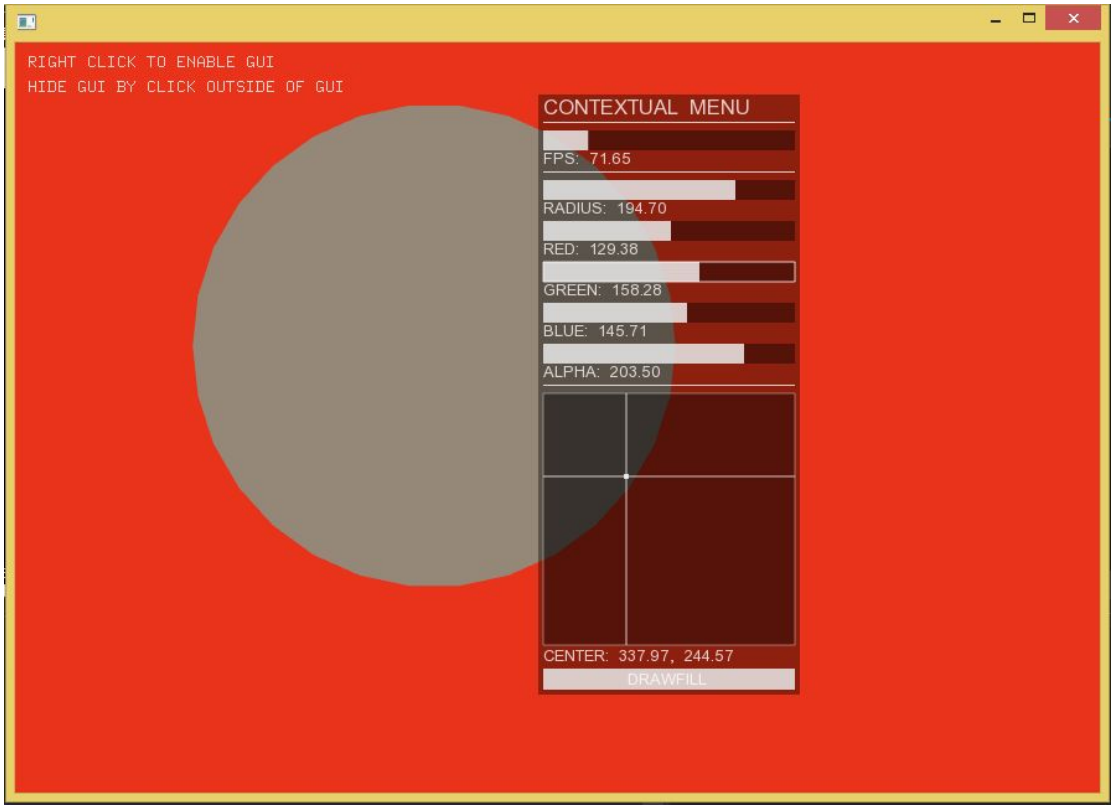


图 12 ContextualMenu 的运行结果

1.头文件添加

```
#include "ofxUI.h"//添加头文件
void exit();
ofxUICanvas *gui;//总面板
void guiEvent(ofxUIEventArgs &e);//事件函数
bool drawFill; //控制是否填充
float red, green, blue, alpha;//控制颜色

ofColor backgroundColor;//背影颜色
float radius;//控制圆的半径
int resolution;//控制圆的分辨率(边数)
ofPoint position;//中心位置
```

2.源文件

①setup

```
void ofApp::setup(){
    //初始化一些参数
    red = 233;      blue = 240;     green = 52;     alpha = 200;//圆的 RGBA 值
    radius = 150; //圆的半径
    drawFill = true;//是否填充
    backgroundColor = ofColor(233, 52, 27);//背景色
    resolution = 30;//边数
```

```
    position = ofPoint(ofGetWidth()*.5, ofGetHeight()*.5);//位置初始化为屏幕中心
    ofSetCircleResolution(resolution);
    //面板的初始化
    gui = new ofxUICanvas();
    gui->addLabel("CONTEXTUAL MENU");
    gui->addSpacer();
    gui->addFPSSlider("FPS");//FPS 滑动条，实时显示当前的帧率
    gui->addSpacer();
    gui->addSlider("RADIUS", 0.0, 255.0, &radius);//滑动条，调节半径
    gui->addSlider("RED", 0.0, 255.0, &red);//滑动条，调节 R
    gui->addSlider("GREEN", 0.0, 255.0, &green);//滑动条，调节 G
    gui->addSlider("BLUE", 0.0, 255.0, &blue);//滑动条，调节 B
    gui->addSlider("ALPHA", 0.0, 255.0, &alpha);//滑动条，调节 A
    gui->addSpacer();
    gui->add2DPad("CENTER", ofPoint(0,ofGetWidth()), ofPoint(0, ofGetHeight()),
&position);//2d 面板，命名 CENTER，xy 取值范围同窗口大小，圆中心位置为值
    gui->addLabelToggle("DRAWFILL", &drawFill);//toggle 按钮，控制是否填充
    gui->autoSizeToFitWidgets();//面板大小自动适应各组件
    ofAddListener(gui->newGUIEvent,this,&ofApp::guiEvent);//添加事件函数
    gui->loadSettings("GUI/guiSettings.xml");//从 guiSettings.xml 文件载入设定
    gui->setVisible(false); //面板初始化为不可见
}
```

②draw

```
void ofApp::draw(){
    ofBackground(backgroundColor);//设置背景色
    ofPushStyle();//在 ofPushStyle 和 ofPopStyle 之间画图
    ofEnableBlendMode(OF_BLENDMODE_ALPHA);//开启透明度模式
    ofSetColor(255,200);//设置颜色
    //在窗口 10，20 位置显示 RIGHT CLICK TO ENABLE GUI
    //在窗口 10，40 位置显示 HIDE GUI BY CLICK OUTSIDE OF GUI
    ofDrawBitmapString("RIGHT CLICK TO ENABLE GUI", 10,20);
    ofDrawBitmapString("HIDE GUI BY CLICK OUTSIDE OF GUI", 10,40);
    //设置圆的颜色
    ofSetColor(red, green, blue, alpha);
    //是否填充
    if(drawFill)
    {
        ofFill();
```

```
    }
    else
    {
        ofNoFill();
    }
    //在 x，y 处画半径为 radius 大小的圆
    ofCircle(position.x, position.y, radius);

    ofPopStyle();
}
```

③keyPressed

```
void ofApp::keyPressed(int key){
    switch (key)
    {
        //g 键控制菜单是否显示，有记忆性的
        case 'g':
        {
            gui->toggleVisible();
        }
            break;
        default:
            break;
    }
}
```

④mousePressed

```
void ofApp::mousePressed(int x, int y, int button){
     if(button == 2)//如果右键点下，将面板移到右键点下的位置，设置为可见
    {
        gui->setPosition(x, y);
        gui->setVisible(true);
        return;
    }

    if(!gui->isHit(x, y))//若面板外部被点击则将面板隐藏
    {
        gui->setVisible(false);
    }
}
```

⑤exit

```
void ofApp::exit()
{
    //退出前保存面板中的设置到 guiSetting.xml 文件，然后清除内存
     gui->saveSettings("GUI/guiSettings.xml");
    delete gui;
}
```

【例 3：button 的运用】

学习各种 button 的使用。(button,toggle,radio,label button,button matrix,image btn)

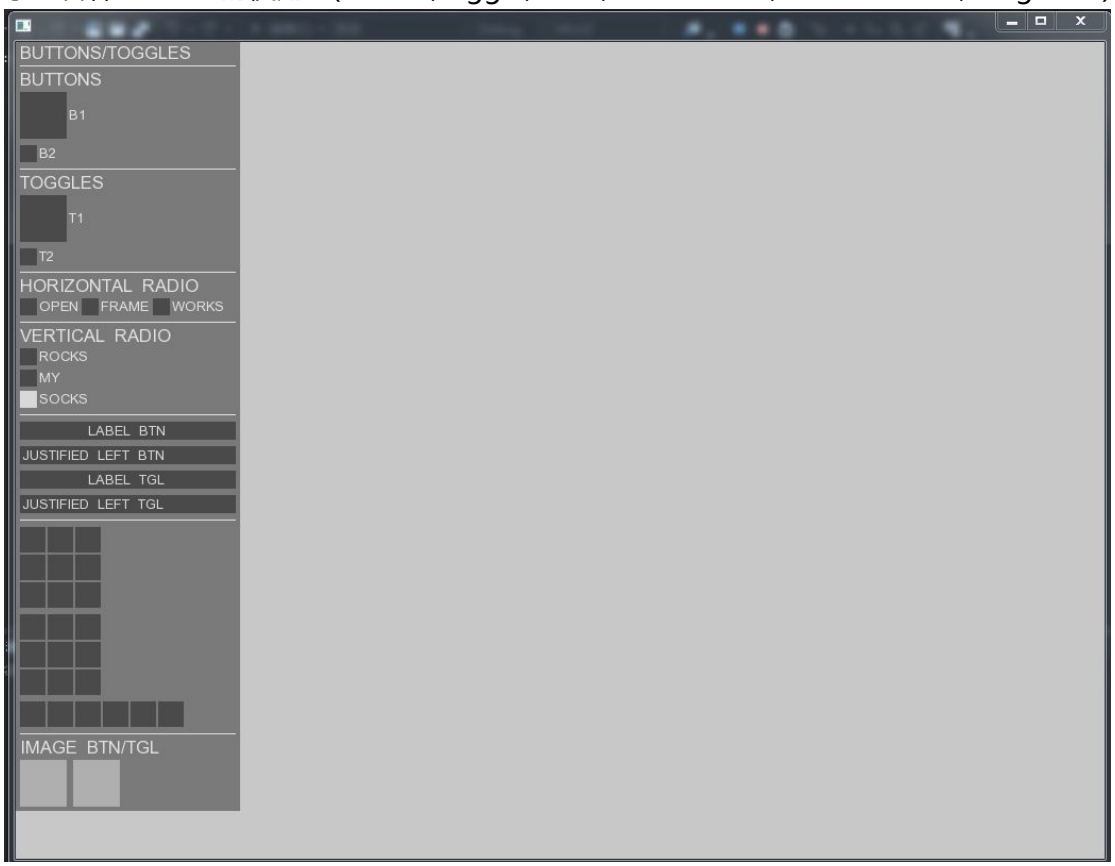


图 13 ofxUI buttons 程序运行图

1.头文件添加

```
//先包含 ofxUI 必要的头文件
#include “ofxUI.h”
//类中加入
void exit();//退出函数，清内存
ofxUISuperCanvas *gui;//总面板
void guiEvent(ofxUIEventArgs &e)//控制面板的事件函数;
bool drawPadding;//是否显示内边距
```

2.源文件

①setup 函数

```
void ofApp::setup(){
```

```
ofSetVerticalSync(true);
//使线条流畅
ofEnableSmoothing();
//初始化为 false，不显示内边距

drawPadding = false;
//新建了一个面板，命名为 BUTTONS/TOGGLES
gui = new ofxUISuperCanvas("BUTTONS/TOGGLES");
gui->addSpacer();//增加一个间隔条，就是图片里的那个白横条
//增加一个标签，命名 BUTTONS
gui->addLabel("BUTTONS");
//增加了两个按钮，B1 初始值为 false 大小 44*44，B2 初始值为 false 默认大小
gui->addButton("B1", false, 44, 44);
gui->addButton("B2", false);
gui->addSpacer();
//增加了一个标签，命名 TOGGLES
gui->addLabel("TOGGLES");
//增加了两个按钮，T1 初始值为 false 大小 44*44，T2 初始值为 false 默认大小
gui->addToggle("T1", false, 44, 44);
gui->addToggle("T2", false);
gui->addSpacer();
//hnames 是动态 string 类型数组
vector<string> hnames;
//push 了三个变量到数组中，OPEN FRAME WORKS
hnames.push_back("OPEN");
hnames.push_back("FRAME");
hnames.push_back("WORKS");
//添加标签，命名 HORIZONTAL RADIO
gui->addLabel("HORIZONTAL RADIO");
//添加水平单选框，命名 HR
gui->addRadio("HR", hnames, OFX_UI_ORIENTATION_HORIZONTAL);

gui->addSpacer();

vector<string> vnames;
vnames.push_back("ROCKS");
vnames.push_back("MY");
vnames.push_back("SOCKS");
```

```
    gui->addLabel("VERTICAL RADIO", OFX_UI_FONT_MEDIUM);
    ofxUIRadio *radio = gui->addRadio("VR", vnames,
    OFX_UI_ORIENTATION_VERTICAL);
    //和上面一样添加垂直单选框，命名 VR，使 SOCKS 处于激活状态
    radio->activateToggle("SOCKS");
    gui->addSpacer();
    //面板中添加标签按钮，初始化为 false，第三个参数设置是否左对齐
    gui->addLabelButton("LABEL BTN", false);
    gui->addLabelButton("JUSTIFIED LEFT BTN", false, true);
    //面板中添加标签触发按钮，初始化为 false，第三个参数设置是否左对齐
    gui->addLabelToggle("LABEL TGL", false);
    gui->addLabelToggle("JUSTIFIED LEFT TGL", false, true);
    //设置全局按钮尺寸为 24
    gui->setGlobalButtonDimension(24);
    gui->addSpacer();
    //3*3 的 toggle 按钮矩阵
    gui->addToggleMatrix("3X3 MATRIX", 3, 3);
    gui->addToggleMatrix("DISABLE MULTIPLE", 3, 3);

    ofxUIToggleMatrix* mtx = (ofxUIToggleMatrix *) gui->getWidget("DISABLE
MULTIPLE");
    //将 DISABLE MULTIPLE 按钮矩阵设置为不允许多个按钮激活
    mtx->setAllowMultiple(false);
    //1 行 6 列 toggle 按钮矩阵，命名 CUSTOM SIZE
    gui->addToggleMatrix("CUSTOM SIZE", 1,6);

    gui->addSpacer();
    gui->addLabel("IMAGE BTN/TGL");

    gui->setGlobalButtonDimension(44);
    //添加图片按钮，命名 BUTTON,图片路径 GUI/toggle.png，初始化值为 false
    gui->addMultiImageButton("BUTTON", "GUI/toggle.png", false);
    gui->setWidgetPosition(OFX_UI_WIDGET_POSITION_RIGHT);
    gui->addMultiImageToggle("TOGGLE", "GUI/toggle.png", false);
    gui->setWidgetPosition(OFX_UI_WIDGET_POSITION_DOWN);
    //控制面板大小自动适应各个组件
    gui->autoSizeToFitWidgets();
    //给面板添加事件函数
```

```
    ofAddListener(gui->newGUIEvent,this,&ofApp::guiEvent);
}
```

注意：

·button 和 toggle 的不同在于，button 没有记忆性而 toggle 是有记忆性的。试试就能马上 get 到他们的不同之处。

②事件函数

```
void ofApp::guiEvent(ofxUIEventArgs &e)
{
    //控件名
    string name = e.widget->getName();
    //获得控件种类
    int kind = e.widget->getKind();
    //判断是什么类型的按钮，输出名字和值
    if(kind == OFX_UI_WIDGET_BUTTON)
    {
        ofxUIButton *button = (ofxUIButton *) e.widget;
        cout << name << "\t value: " << button->getValue() << endl;
    }
    else if(kind == OFX_UI_WIDGET_TOGGLE)
    {
        ofxUIToggle *toggle = (ofxUIToggle *) e.widget;
        cout << name << "\t value: " << toggle->getValue() << endl;
    }
    else if(kind == OFX_UI_WIDGET_IMAGEBUTTON)
    {
        ofxUIImageButton *button = (ofxUIImageButton *) e.widget;
        cout << name << "\t value: " << button->getValue() << endl;
    }
    else if(kind == OFX_UI_WIDGET_IMAGETOGGLE)
    {
        ofxUIImageToggle *toggle = (ofxUIImageToggle *) e.widget;
        cout << name << "\t value: " << toggle->getValue() << endl;
    }
    else if(kind == OFX_UI_WIDGET_LABELBUTTON)
    {
        ofxUILabelButton *button = (ofxUILabelButton *) e.widget;
        cout << name << "\t value: " << button->getValue() << endl;
```

```
    }
    else if(kind == OFX_UI_WIDGET_LABELTOGGLE)
    {
        ofxUILabelToggle *toggle = (ofxUILabelToggle *) e.widget;
        cout << name << "\t value: " << toggle->getValue() << endl;
    }
    else if(name == "B1")
    {
        ofxUIButton *button = (ofxUIButton *) e.widget;
        cout << "value: " << button->getValue() << endl;
    }
    else if(name == "B2")
    {
        ofxUIButton *button = (ofxUIButton *) e.widget;
        cout << "value: " << button->getValue() << endl;
    }
    else if(name == "T1")
    {
        ofxUIToggle *toggle = (ofxUIToggle *) e.widget;
        cout << "value: " << toggle->getValue() << endl;
    }
    else if(name == "T2")
    {
        ofxUIToggle *toggle = (ofxUIToggle *) e.widget;
        cout << "value: " << toggle->getValue() << endl;
    }
}
```

③keyPressed 函数

```
void ofApp::keyPressed(int key){
    switch (key)
    {
        case 'p'://p 键控制衬底
        {
            drawPadding = !drawPadding;
            gui->setDrawWidgetPadding(drawPadding);
        }
            break;
```

```
            case 'g'://g 键控制面板是否显示
            {
                gui->toggleVisible();
            }
                break;
            default:
                break;
        }
}
```

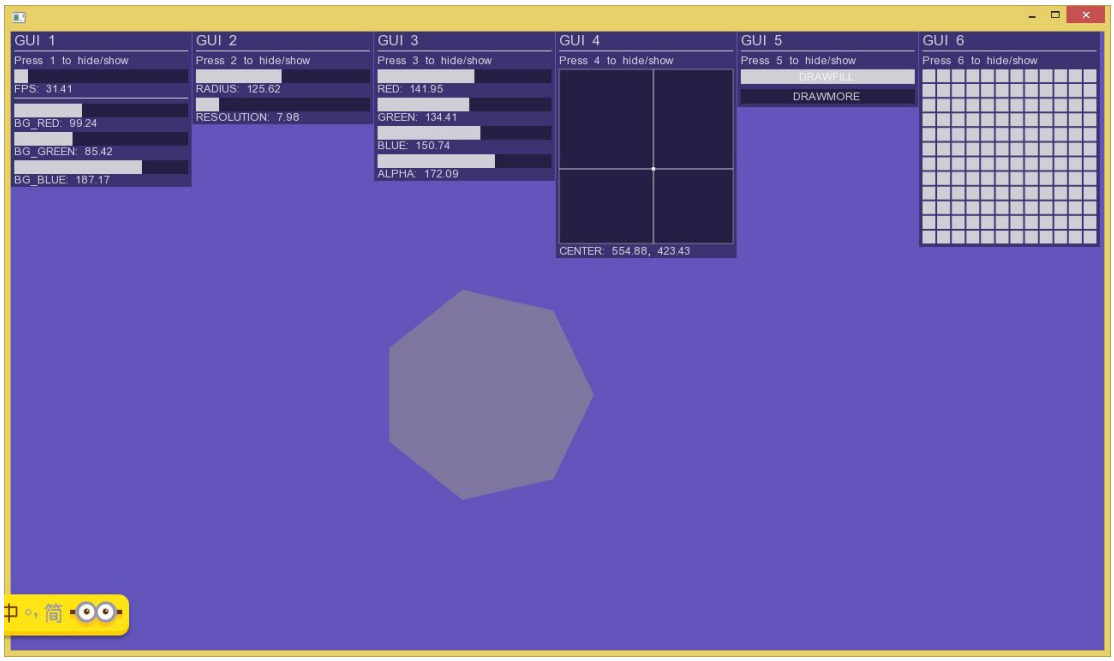
④else

```
void ofApp::exit()
{
    delete gui;//退出时清除内存
}
```

【另外的例子】

基本上所有的控件都在 allwidgets 里介绍过了，可以自己探索后边有趣的例子。

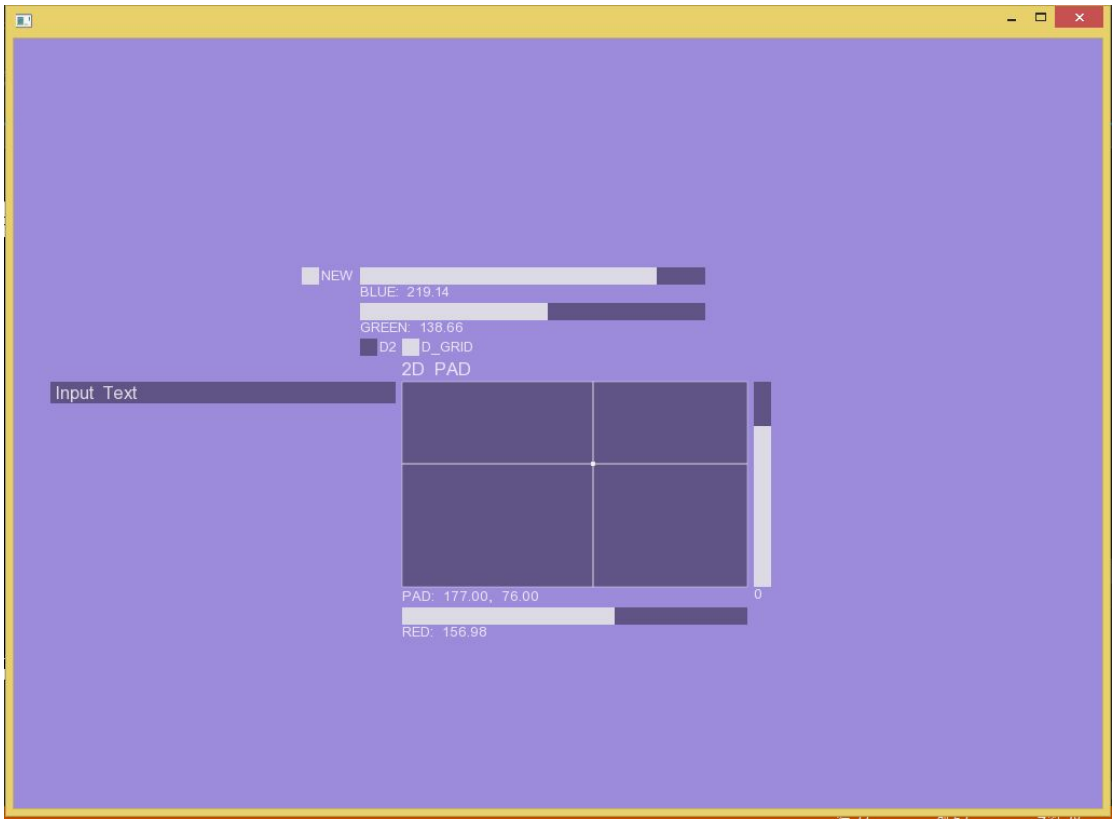
example-CanvasMapVector



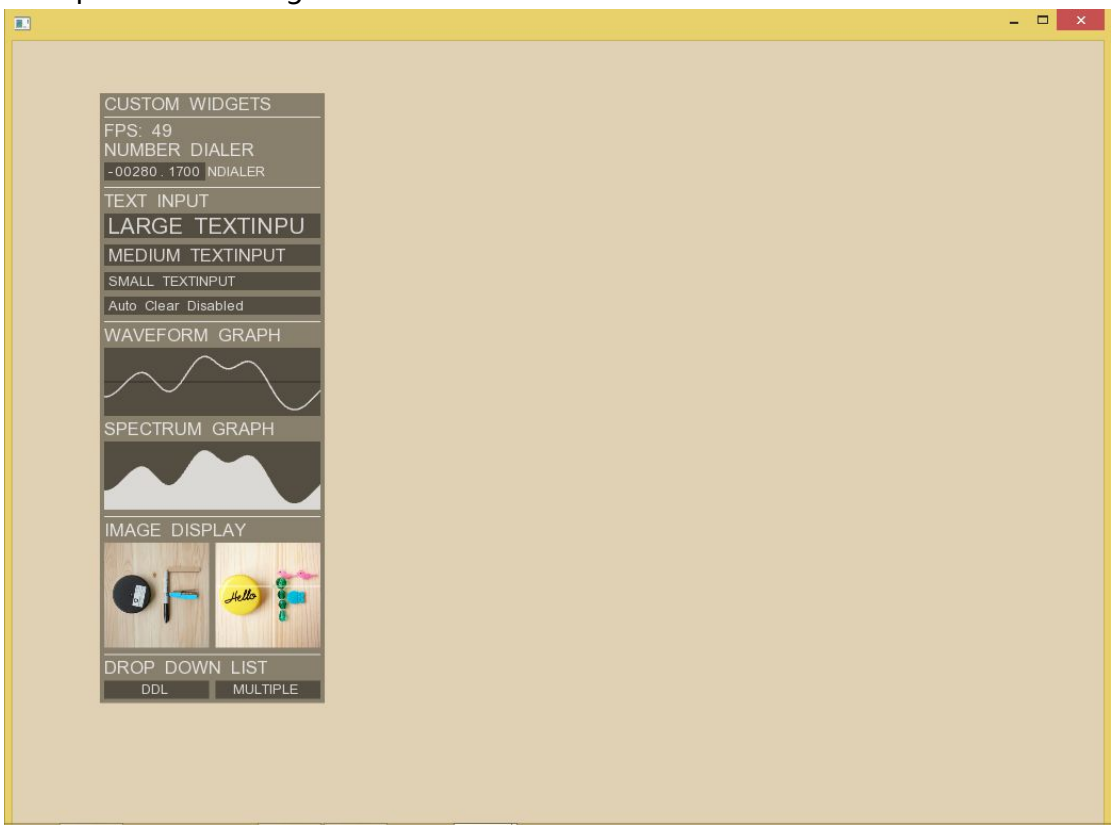
example-CustomFont

example-Customization

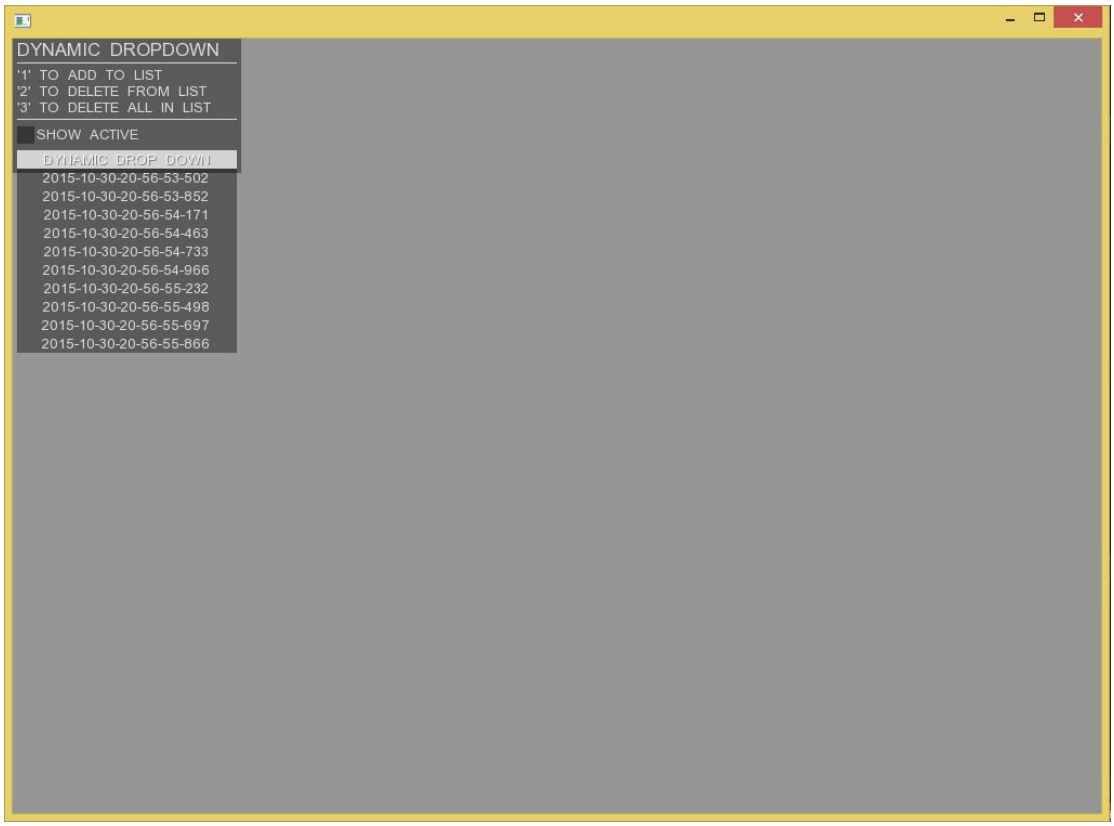
example-CustomPlacement



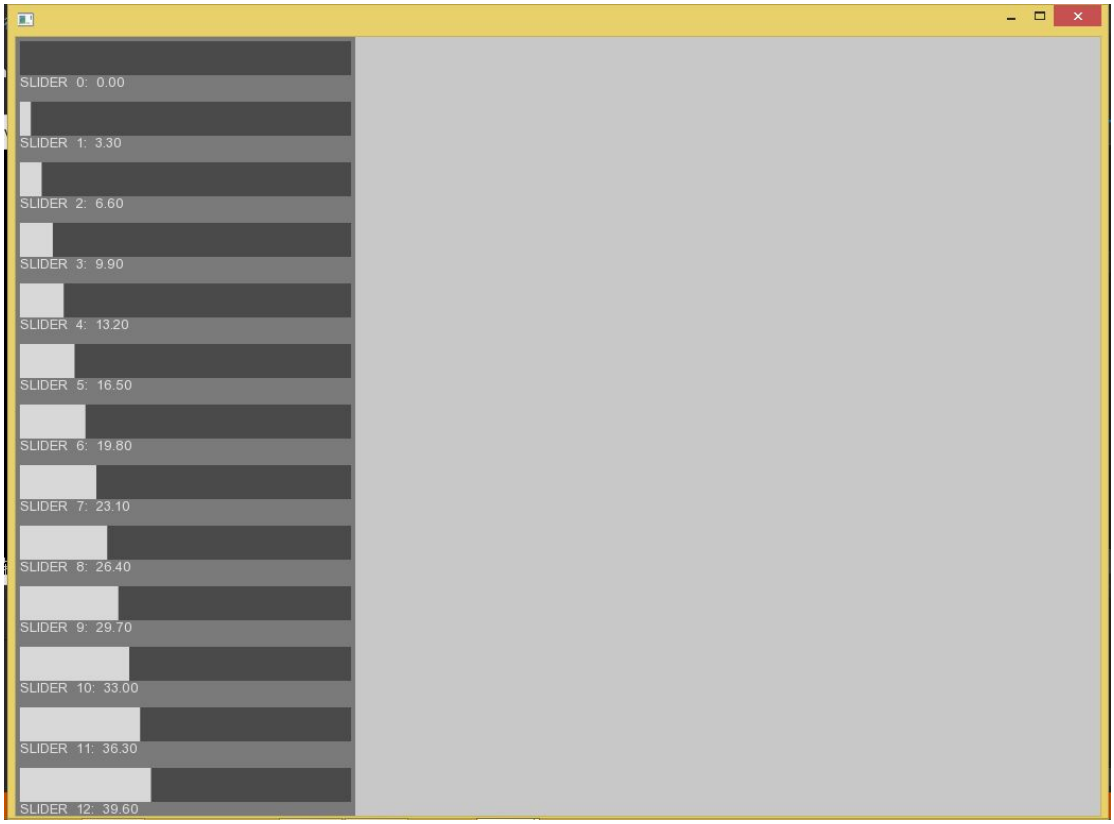
example-CustomWidgets



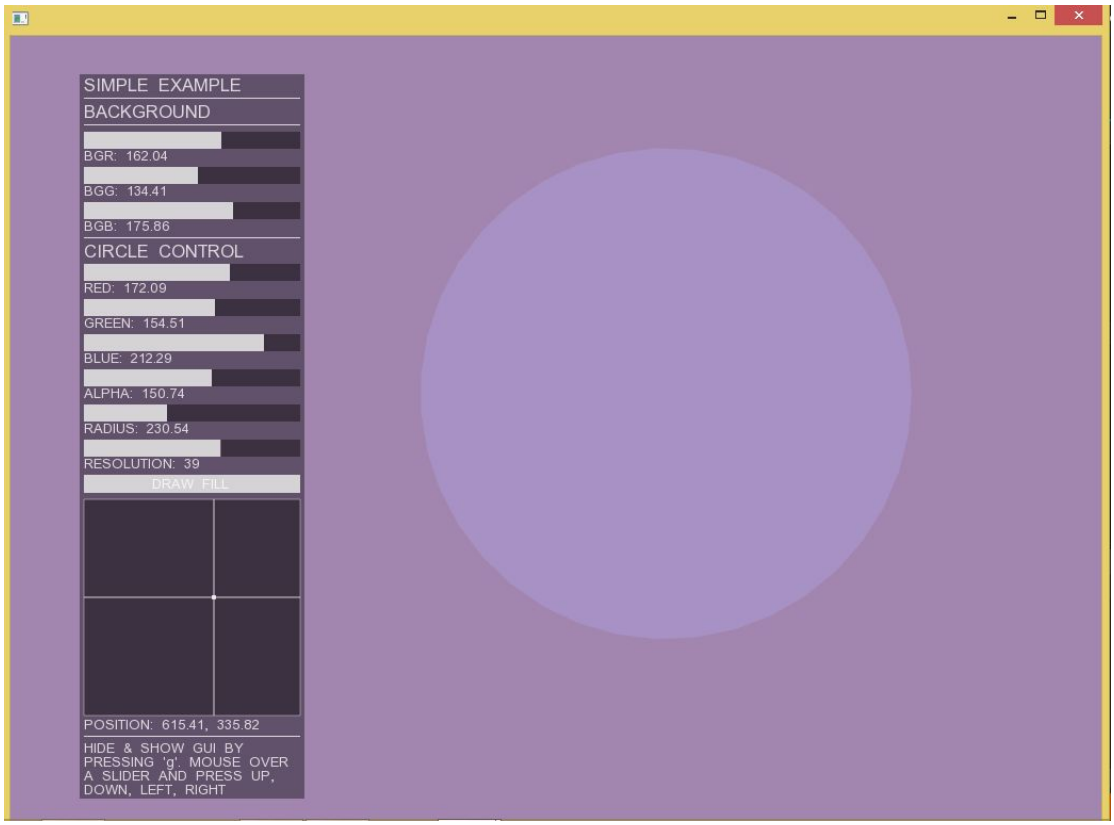
example-DynamicDropDown



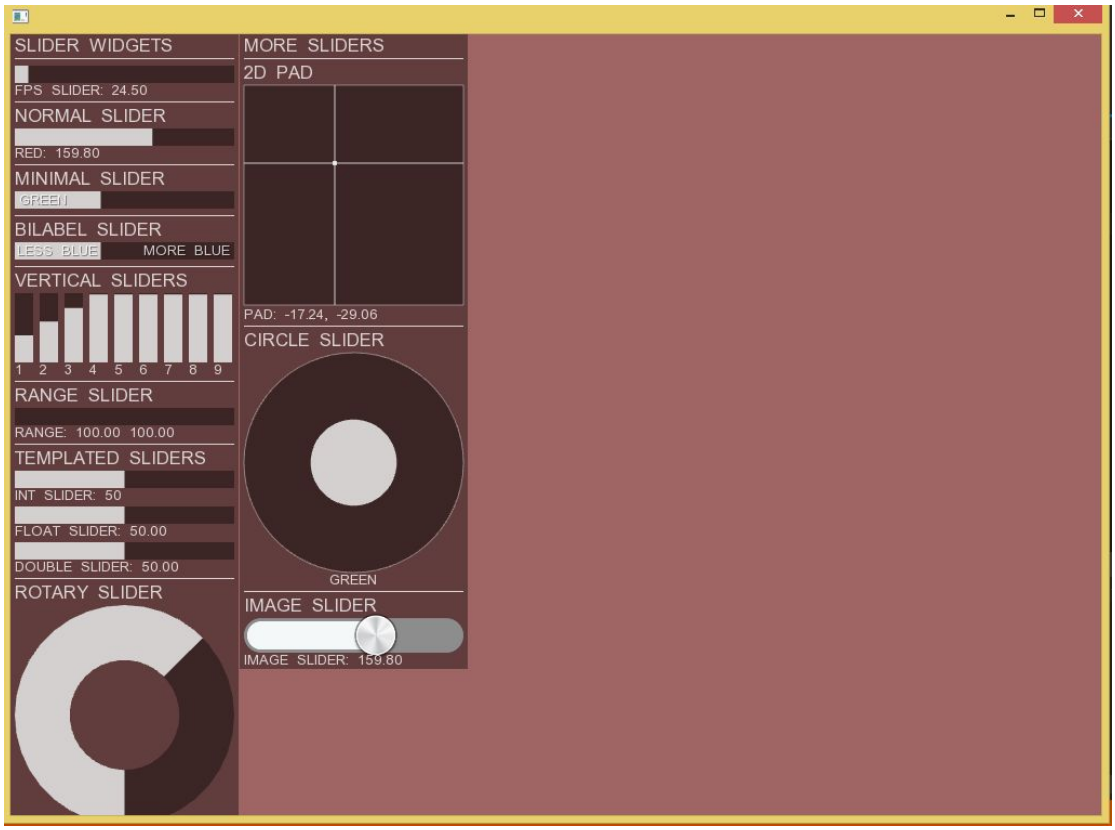
example-Scrollable



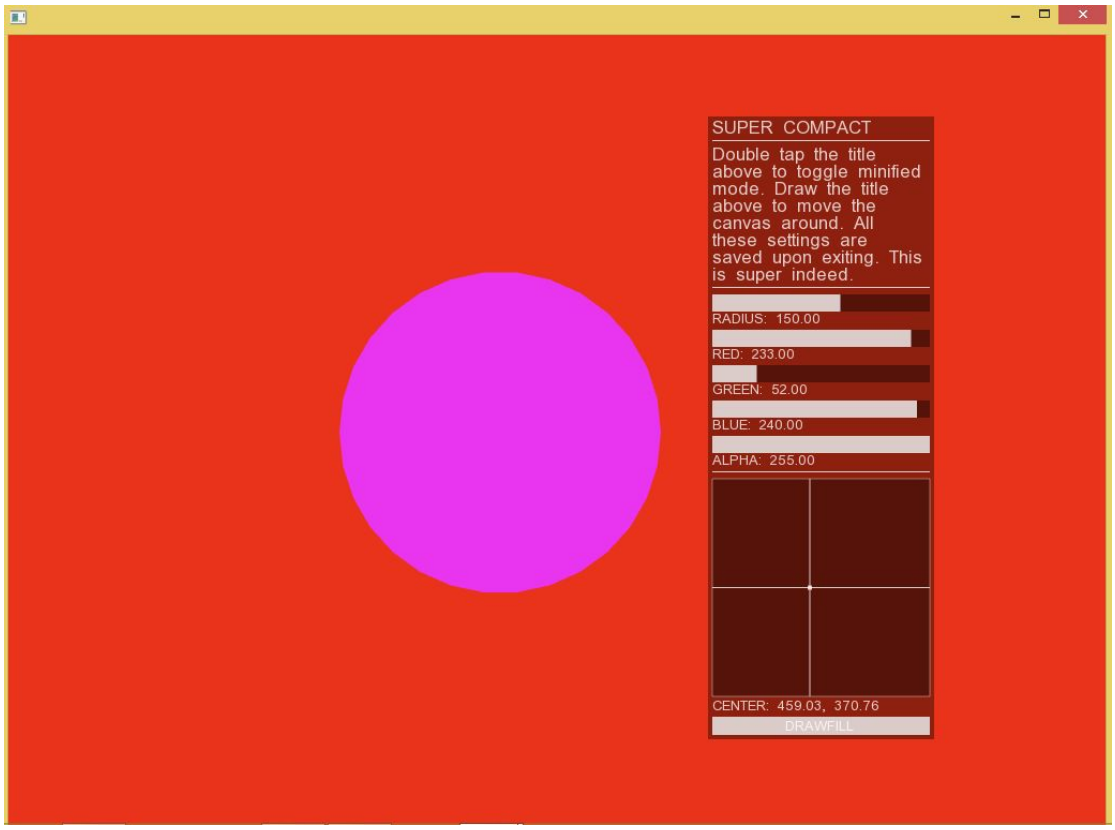
example-SimpleExample



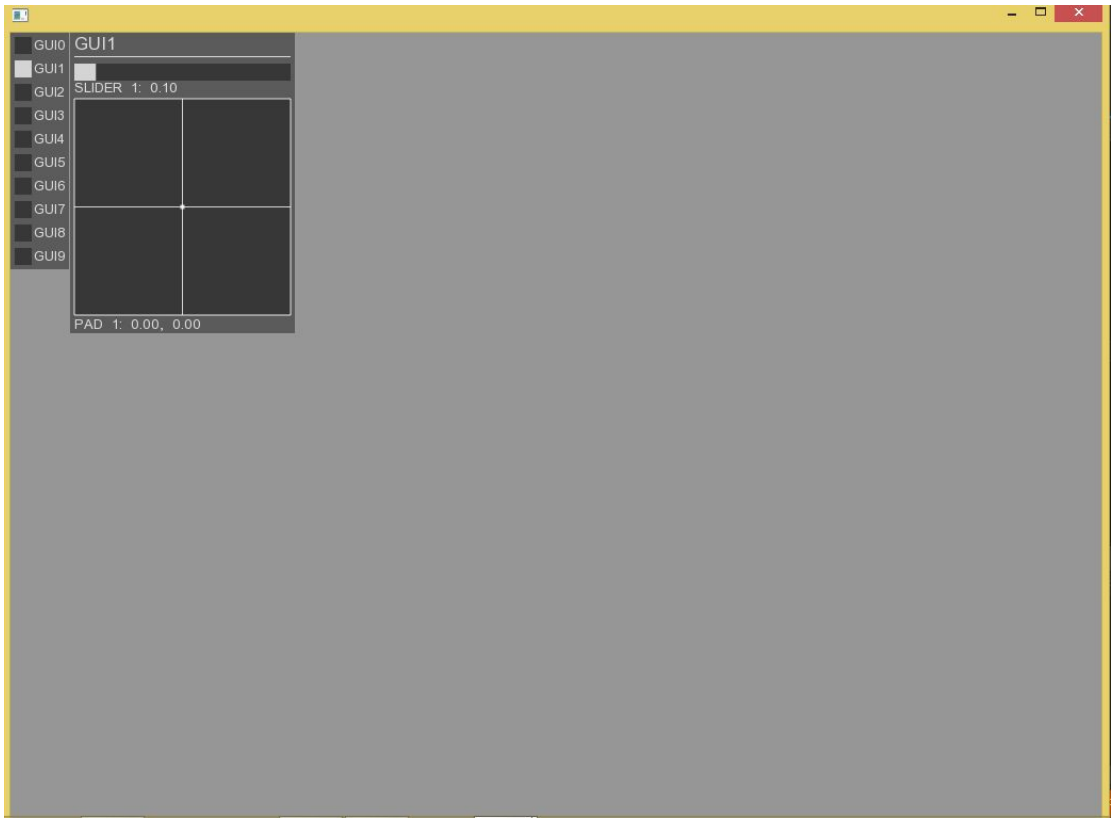
example-Sliders



example-SuperCanvas

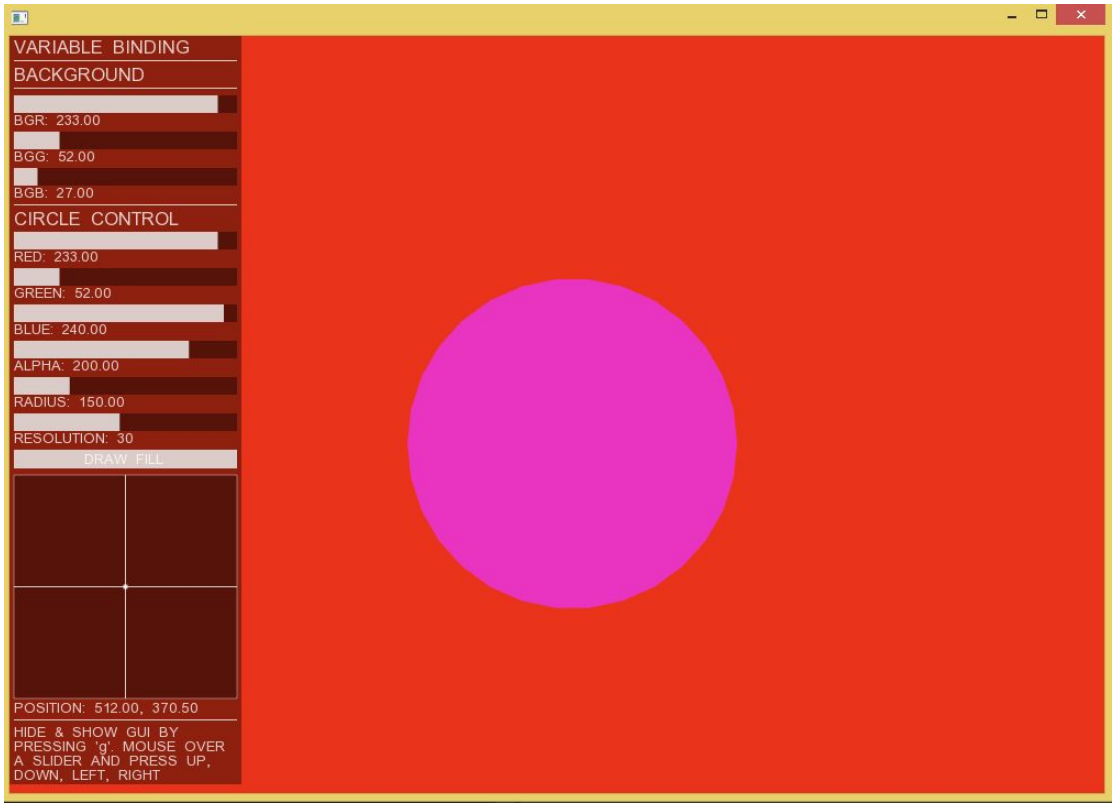


example-TabBar



example-Tutorial

example-VariableBinding



【参考】《游戏规则增强--图形界面与数据存储》律老师