Josh UI 快速开发指南

--V 1.0.0

目录

简介 1
硬件和软件 1
屏幕连接 1
固件烧录 2
调试连接 3
API 文档 4
UI 框架使用介绍 5
Widget 基础控件 5
按钮 5
复选框 5
图片 5
文字 6
输入框 6
滚动列表框 6
滑动条 7
开关按钮 7
表格 8
下拉列表 8
输入键盘 9
Canvas 画布 9
Container 容器 12
布局 14
设置与父对象的对齐方式 14
Flex 内容布局 14
滚动条设置 15
字体设置 16
实践：创建第一个图形应用 17
IDE（版本号<1.2.20） 17
IDE（版本号>=1.2.20） 19

简介

本文档是对基于 JOSH VM 的 UI 框架，进行 UI 的快速开始开发指南。
JOSH UI 是 JOSH 公司自主研发的基础 UI 框架。 该模块允许创建基本的人机界面 （HMI），并在基于像素的屏幕上输出。

硬件和软件

硬件：

米尔显示屏：MY-TFT070CV2
米尔开发板：MYD-Y6ULY2-V2-256N256D-50-I

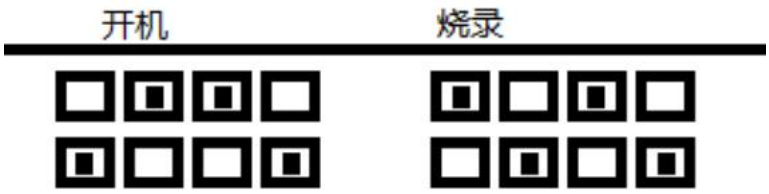
软件：

开发语言：Java
软件开发工具：进入 JOSH 官网，下滑网页找到 **JOSH Studio 下载**，点击下载。

屏幕连接

显示器和开发板中用异面的 50pin FPC 排线，按左图连接上显示屏一端，再按右图连接开发板端 J3，这样显示器和开发板就连接好了。

固件烧录

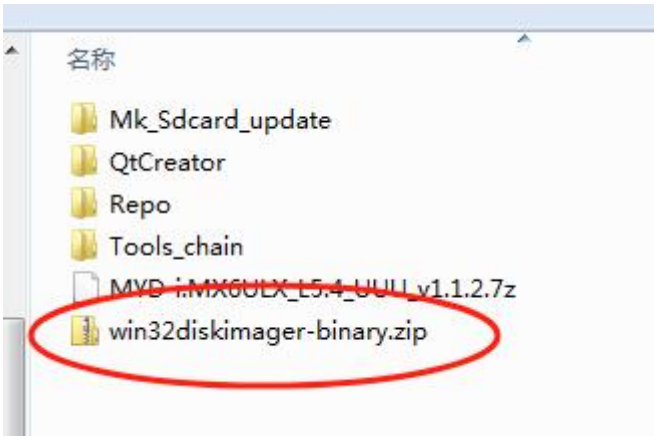


1. 从网站下载固件，写入到 SD 卡中。

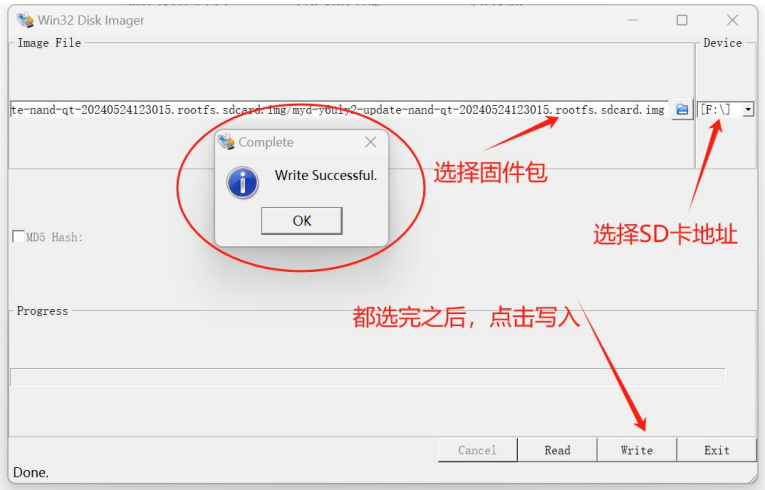
   下载地址：256N 版本　　4E 版本

   下载写入固件到 sd 卡的工具，进入米尔官网：http://down.myir-tech.com/MYD-Y6ULX/，选择 03Tools 下载。

   在下载的文件中找到 win32diskimager-binary.zip 这个工具运行起来。



   选择刚刚下载的固件压缩包解压后里面的 img 文件，烧录目标地址选择 SD 卡地址。注意 sd 卡地址千万不能选错，写入 img 会将整个盘覆盖写入。



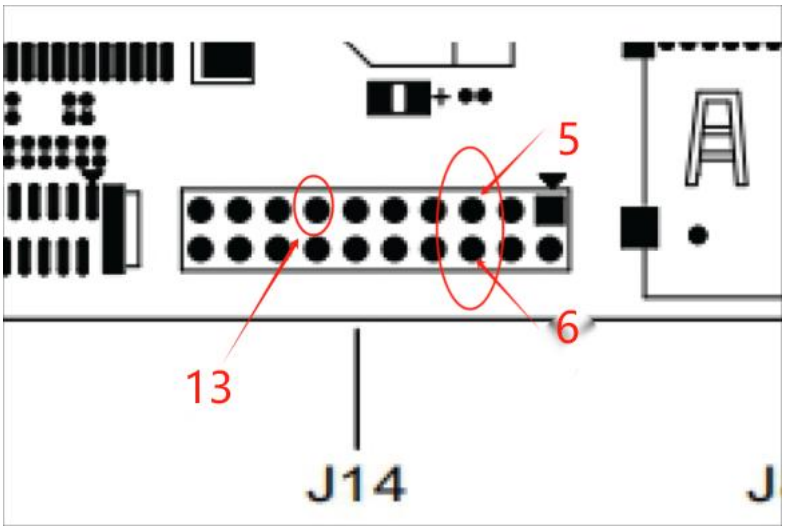
2. 断电，将网关拨成上图右边的烧录码
3. 插入装有写入固件的 SD 卡
4. 上电烧录（可连上串口，查看烧录信息，直到显示完成）

```
Updating...
Updating...
Updating...
Updating...
Updating...
Updating...
Updating...
Updating...
tar /dev/mtd4 okay!
Updating...
**********************************************
********** System update successfully **********
********** System update successfully **********
********** System update successfully **********
**********************************************

/usr/bin/flash_nand.sh: line 227:   539 Terminated              update_start
Starting syslogd/klogd: done

Freescale i.MX Release Distro 4.1.15-2.0.1 myd-y6ull14x14 /dev/ttymxc0

myd-y6ull14x14 login:

U-Boot 2019.04-5.4.3-2.0.0+g6d601dd (Mar 11 2024 - 06:27:51 +0000)
```

5. 断电，拨成开机码。
6. 拔掉烧录的 SD 卡
7. 上电，查看系统启动是否正常

调试连接

6ul 开发板上，console 口管脚的位置在 J14，如下图位置，引脚定义如下图：



4.7.1. 引脚定义

位号	标识	功能	信号	
J14	1	输出电源 5V	VDD_5V	
	2	输出电源 3.3V	VDD_3V3	
	3	电源地	DGND	
	4	空脚，无连接	NC	
	5	UART3 发送	UART3_TX	
	6	UART3 接收	UART3_RX	
	7	UART3 接收就绪	UART3_RTS	
	8	UART3 发送清空	UART3_CTS	
	9	UART7 接收	UART7_RX	
	10	UART7 发送	UART7_TX	
	11	空脚，无连接	NC	
	12	空脚，无连接	NC	
	13	电源地	DGND	
	14	空脚，无连接	NC	
	15	SPI1 主输出从输入	SPI1_MOSI	

所以需要将接地线连接至 13 引脚，RX 连接 5 号引脚，TX 连接 6 号引脚。

API 文档

所有 JOSH UI 框架的 API 文档说明，见官网 API 文档。
官网 API 文档连接：API

UI 框架使用介绍

Widget 基础控件

JOSH UI 框架中基础的一些 UI 控件介绍：

按钮

按钮的实现类是 Button：

```
Button btn = new Button();
btn.setText("Button");//设置按钮文字
btn.setPos(300, 100);//设置按钮位置
btn.setChecked(false);//设置按钮是否被选中

btn.setClickedListener(new ClickedListener() {
    public void onClicked(UIObject target) {
        ((Button)target).setChecked(!((Button)target).isChecked());
        System.out.println("set checked to " + ((Button)target).isChecked());
    }
});
```

复选框

复选框的实现类是 CheckBox:

```
CheckBox cb = new CheckBox();
cb.setText("Option1");//设置复选框标题
cb.setPos(300, 50);//设置复选框位置
```

CheckBox 是复选框类。

图片

图片的实现类是 Image：

```
Image image = new Image();
image.setRadius(30);//设置圆角
image.enableClipCorner(true);//允许在圆角上剪裁溢出的内容
image.setBackgroundColor(new Color(0xFF00FF00));//设置背景色
image.setPos(20, 20);//设置位置
```

```
        image.setSize(300, 300);//设置大小

        try {
            image.setSource("/image/default.jpt");//设置图片位置比如 res 下的 image 目录下的
default.jpg
        } catch (Exception e1) {
            e1.printStackTrace();
        }
```

文字

文字的类的实现类是 Label：

```
        Label l = new Label();
        l.setPos(300, 100);//位置
        l.setText("文字测试");//设置文字
        l.setTextColor(Color.RED);//设置颜色
        l.setTextLetterSpace(3);//设置文字间隙
        l.setTextLineSpace(10);//设置行间距
        l.setTextAlign(UIObject.ALIGN_CENTER);//文字对齐
        l.setLongTextMode(Label.LONG_TEXT_MODE_SCROLL_CIRCULAR);//长文本显示模式设置
```

输入框

输入框的实现类是 Textarea：

```
        Textarea ta = new Textarea();
        ta.setText("This is a textarea.");//设置输入框里的内容
        ta.setPos(200, 300);//设置位置
        ta.enableMode(Textarea.MODE_PASSWORD);//设置输入框的模式，默认是多行，这里设置的是密
码模式
```

滚动列表框

滚动列表框的实现类是 Roller：

```
        Roller roller = new Roller();
        roller.setPos(200, 200);
        roller.addOption("First");//设置滚动选项
```

```
roller.addOption("Second");
roller.addOption("Third");
roller.addOption("Fourth");
roller.addOption("Fifth");
roller.addOption("Sixth");
roller.addOption("Seventh");
roller.setSelected(5);//设置选中的选项
```

滑动条

滑动条的实现类是 Slider:

```
Slider slider = new Slider();
slider.setRange(50, 200);//设置滑动范围
slider.setPos(150, 300);//设置位置
slider.setValue(100);//设置当前值
slider.setMode(Slider.MODE_RANGE);//设置滑动条的样式
slider.setStartValue(80);//设置开始值
```

开关按钮

开关按钮的实现类是 switch：

```
Switch sw = new Switch();
sw.setPos(400, 200);//开关位置
sw.setSize(150, 50);//开关大小
sw.setDisabled(false);//是否可用

sw.setValueChangedListener(new ValueChangedListener() {

    public void onValueChanged(final UIObject target) {
        if (((Switch)target).isChecked()) {
            //TODO
        } else {
            //TODO
        }
    }
});
```

表格

表格的实现类是 Table:

```
Table tb = new Table(2, 3);//设置表格 2 行，3 列
tb.setPos(100, 200);//设置位置
tb.setColumnWidth(1, 50);//每列宽度
tb.setCellText(0, 0, "00");//设置第一行第一列文字
tb.setCellText(1, 0, "10");
tb.setCellText(0, 1, "01");
tb.setCellText(1, 1, "11");
tb.setCellText(0, 2, "02");
tb.setCellText(1, 2, "12");
tb.setTextAlign(UIObject.ALIGN_CENTER);//设置文字对齐方式
```

下拉列表

下拉列表的实现类是 DropdownList：

```
DropdownList dropdown = new DropdownList();
dropdown.setPos(200, 100);//设置位置
dropdown.addOption("A");//设置下拉选项
dropdown.addOption("B");
dropdown.addOption("C");
dropdown.addOption("D");
dropdown.addOption("E");
dropdown.addOption("F");
dropdown.addOption("G");
dropdown.setValueChangedListener(new ValueChangedListener() {////设置值的变化监听器

    public void onValueChanged(UIObject target) {
        System.out.println("selected index:"
                + ((DropdownList) target).getSelected());
        String[] options = ((DropdownList) target).getOptions();
        System.out.println("selected str:"
                + options[((DropdownList) target).getSelected()]);
    }
});
```

输入键盘

输入键盘的实现类是 keyboard：

```
private void Keyboard_1(Container container) {
    Textarea ta = new Textarea();
    ta.enableMode(Textarea.MODE_ONELINE);
    ta.setText("This is a textarea.");
    final Keyboard kb = new Keyboard();
    kb.setTextarea(ta);//设置对应的输入框
    container.addWidget(ta);//需要添加控件到父界面
    container.addWidget(kb);//需要添加控件到父界面
    ta.setAlign(UIObject.ALIGN_CENTER, UIObject.ALIGN_TOP);//设置在父 UI 里的对齐位置
    ta.setY(50);//基于对齐位置后调整 Y
    ta.setSize(300, 200);//设置输入框大小
    kb.setSize(600, 200);//设置键盘大小
    ta.setReadyListener(new ReadyListener() {
        public void onReady(UIObject target) {
            System.out.println(
                    "current text:" + ((Textarea) target).getText());
        }
    });
    ta.setClickedListener(new ClickedListener() {//设置点击事件
        public void onClicked(UIObject target) {
            kb.setHidden(false);
        }
    });
    kb.setReadyListener(new ReadyListener() {//设置 UIObject 就绪通知
        public void onReady(UIObject target) {
            kb.setHidden(true);
        }
    });
    kb.setCancelListener(new CancelListener() {//设置 UIObject 的取消通知
        public void onCancel(UIObject target) {
            kb.setHidden(true);
        }
    });
}
```

Canvas 画布

画布可以实现自由绘制 UI 界面，同时它也是一个 widget 控件，可用于绘制线条、矩形、文

本、图像等。

- drawLine(int, int, int, int) : void
- drawLine(PointGroup) : void
- drawPolygon(PointGroup) : void
- drawRect(int, int, int, int) : void
- drawArc(int, int, int, int, int) : void
- drawText(String, int, int) : void
- drawText(String, int, int, int) : void
- drawImage(String, int, int) : void
- drawImage(byte[], int, int, int, int) : void

下面是一些绘制示例代码：

```
//示例 1 设置背景色后修改像素颜色
Canvas canvas = new Canvas(50, 50);
 canvas.setPos(300, 200);
 canvas.setSize(100, 100);
 canvas.fill(new Color(0x7F007F00));
 canvas.setBackgroundColor(new Color(0xFFFF0000));
 for (short i = 0; i < 25; i++) {
     for (short j = 0; j < 40; j++) {
         canvas.setPixel(i, j, new Color(0xFFFFFFFF));
     }
 }

//示例 2 绘制虚线
 Canvas canvas = new Canvas(150, 150);
 canvas.setPos(300, 200);
 canvas.setBackgroundColor(new Color(0xFFFF0000));
 PointGroup pg = new PointGroup();
 pg.addPoint(10, 20);
 pg.addPoint(10, 120);
 pg.addPoint(140, 120);
 canvas.setLineColorOfDraw(Color.GREEN);
 canvas.setLineWidthOfDraw(2);//设置线的宽度
 canvas.setLineDashWidthOfDraw(10);//设置线段长度
 canvas.setLineDashGapOfDraw(10);//设置间隔
 canvas.drawLine(pg);

//示例 3 绘制实线
Canvas canvas = new Canvas(50, 50);
canvas.setPos(300, 200);
canvas.setSize(75, 75);
canvas.setBackgroundColor(new Color(0xFFFF0000));
PointGroup pg = new PointGroup();
```

```
pg.addPoint(10, -10);
pg.addPoint(10, 10);
pg.addPoint(40, 0);
pg.addPoint(30, 30);
pg.addPoint(10, 20);
pg.addPoint(100, 100);
pg.addPoint(10, 10);
canvas.setLineColorOfDraw(Color.PURPLE);
canvas.drawLine(pg);

//示例 4 绘制多边形
Canvas canvas = new Canvas(100, 100);
canvas.setPos(300, 200);
canvas.setBackgroundColor(new Color(0xFFFF0000));
PointGroup pg = new PointGroup();
pg.addPoint(20, 20);
pg.addPoint(50, 20);
pg.addPoint(50, 50);
pg.addPoint(20, 50);
pg.addPoint(-10, 35);
canvas.setBackgroundColorOfDraw(Color.BLUE);
canvas.drawPolygon(pg);

//示例 5 绘制带阴影的矩形
Canvas canvas = new Canvas(100, 100);
canvas.setPos(300, 200);
canvas.setBackgroundColor(new Color(0xFFFF0000));
canvas.setBackgroundColorOfDraw(Color.WHITE);//背景色
canvas.setLineColorOfDraw(Color.GREEN);//边框颜色
canvas.setLineWidthOfDraw(3);
canvas.setRectRadiusOfDraw(5);
 //阴影设置
canvas.setShadowColorOfDraw(Color.GRAY);
canvas.setShawdowWidthOfDraw(3);
canvas.setShadowOffsetXOfDraw(-5);
canvas.setShadowOffsetYOfDraw(-5);
canvas.drawRect(20, 20, 70, 30);

 //示例 6 绘制圆弧
Canvas canvas = new Canvas(100, 100);
canvas.setPos(300, 200);
canvas.setBackgroundColor(new Color(0xFFFF0000));
canvas.setLineColorOfDraw(Color.BLUE);
canvas.setLineWidthOfDraw(3);
```

```
canvas.drawArc(50, 50, 40, 0, 270);

//示例 7 绘制文本
Canvas canvas = new Canvas(100, 100);
canvas.setPos(300, 200);
canvas.setBackgroundColor(new Color(0xFFFF0000));
canvas.setTextColorOfDraw(Color.GREEN);
canvas.setTextLetterSpaceOfDraw(20);
canvas.setTextLineSpaceOfDraw(20);
canvas.drawText("你好\nWorld", 10, 10);


//示例 8  绘制图片
Canvas canvas = new Canvas(200, 200);
canvas.setPos(300, 100);
canvas.setBackgroundColor(new Color(0xFF0000FF));
try {
    canvas.drawImage("/package.png", 50, 20);
} catch (IOException e) {
    e.printStackTrace();
}
canvas.setImageAngleOfDraw(1800);//设置图像绘制角度
canvas.setImagePivotOfDraw(25, 25);//旋转轴心点
try {
    canvas.drawImage("/package.png", 120, 20);
} catch (IOException e) {
    e.printStackTrace();
}
canvas.resetPropertyValuesOfDraw();
canvas.setImageRecolorOfDraw(Color.GREEN);//图像重新上色
canvas.setImageZoomOfDraw(1.5);//放大
try {
    canvas.drawImage("/package.png", 100, 100);
} catch (IOException e) {
    e.printStackTrace();
}
```

Container 容器

在 UI 中，当多个控件需要组合的时候，可以使用 UI 容器放置他们，同时也可以添加子容器。

```
Container subContainer = new Container();//子容器
container.addContainer(subContainer);//添加子容器到父容器

subContainer.setBackgroundColor(new Color(0x6400FF00));//背景色
subContainer.setHeight(200);//高度
subContainer.setWidth(300);//宽度
subContainer.setX(250);//x 的位置
subContainer.setY(100);//y 的位置
subContainer.setBorderWidth(5);//设置边框宽度
subContainer.setBorderColor(new Color(0xFF777700));//边框颜色
subContainer.setRadius(100);//设置圆角
subContainer.setShadowWidth(10);//阴影宽度
subContainer.setShadowColor(new Color(0xFF007777));//阴影颜色
subContainer.setShadowOffsetX(20);//阴影 x 的偏移量
subContainer.setShadowOffsetY(20);//阴影 y 的偏移量
//创建一个按钮
Button btn = new Button("Button");
btn.setX(100);
subContainer.addWidget(btn);
Button btn2 = new Button("Button2");
btn2.setAlign(UIObject.ALIGN_LEFT, UIObject.ALIGN_CENTER);
btn2.setX(50);
btn2.setTranslateX(50);
subContainer.addWidget(btn2);//把按钮控件放进子容器

Area area = subContainer.getCoordinates();//获取子容器的区域坐标
System.out.println(area.getX1() + "," + area.getY1() + ","
        + area.getX2() + "," + area.getY2());
area = btn.getCoordinates();
System.out.println(area.getX1() + "," + area.getY1() + ","
        + area.getX2() + "," + area.getY2());
area = btn2.getCoordinates();
System.out.println(area.getX1() + "," + area.getY1() + ","
        + area.getX2() + "," + area.getY2());
```

布局

设置与父对象的对齐方式

给控件设置布局的时候用的是 setAlign(int arg0, int arg1)方法，比如：

```
Label l = new Label();
l.setText("Center");
l.setAlign(Label.ALIGN_CENTER, Label.ALIGN_CENTER);//左右居中,上下居中
```

参数：

arg0-在水平方向上对齐。

ALIGN_DEFAULT 或 ALIGN_LEFT 或 ALIGN_CENTER 或 ALIGN_RIGHT

arg1-在垂直方向上对齐。

ALIGN_DEFAULT 或 ALIGN_TOP 或 ALIGN_CENTER 或 ALIGN_BOTTOM

Flex 内容布局

Flex 布局类，可以将 UI 上的内容对象排列成行或列，调整行和列之间的间距，处理增长以使项目填充最小/最大宽度和高度的剩余空间。

- 使用 forceNewTrack，Flex 布局将把一个对象放入新行。
- Flex Grow 如果被设置，将增长以填充列上的剩余空间。可用空间将根据 Grow 值在对象之间分配（价值越大意味着空间越大），可以使用 setFlexGrow 设置 Grow 值。
- setRowPadding 和 setColumnPadding 可用于修改行和列之间的空间或轨道上的项。

属性	释义
FLEX_ALIGN_CENTER	简单居中
FLEX_ALIGN_END	意思是水平在右边，垂直在底部
FLEX_ALIGN_SPACE_AROUND	对象在轨道上均匀分布，周围空间相等。
FLEX_ALIGN_SPACE_BETWEEN	子对象在轨道上均匀分布：第一个子对象在开始行，最后一个子对象在结束行。
FLEX_ALIGN_SPACE_EVENLY	内容被分布，使得任意两个子对象的间距（以及到边缘的空间）相等。

FLEX_ALIGN_START	表示水平向左，垂直向上
FLEX_FLOW_TYPE_COLUMN	将子 UI 对象放在一列（轨道）中，不包装
FLEX_FLOW_TYPE_COLUMN_WRAP	将子 UI 对象放在带包装的列（轨道）中
FLEX_FLOW_TYPE_ROW	将子 UI 对象排成一排（轨道），不要包裹
FLEX_FLOW_TYPE_ROW_WRAP	将子 UI 对象用包裹物排成一排（轨道）

```
Container c1 = new Container();//UI 容器
c1.setPadding(5);
c1.setSize(400, 200);
c1.setPos(200, 200);
for (int i = 0; i < 5; i++) {
    Button btn = new Button("Button" + i);
    c1.addWidget(btn);
}
Flex layout = new Flex();
layout.setFlowType(Flex.FLEX_FLOW_TYPE_ROW_WRAP);//设置自动填充主方向类型
layout.setMainAlign(Flex.FLEX_ALIGN_SPACE_EVENLY);//设置主要主方向上的对齐类型
layout.setCrossAlign(Flex.FLEX_ALIGN_CENTER);//设置主方向横向对齐类型
layout.setTrackCrossAlign(Flex.FLEX_ALIGN_SPACE_EVENLY);//设置轨道横向对齐类型
c1.updateLayout(layout);
```

滚动条设置

AbstractUIObject 类拥有设置滚动条的方法，它的子类 Container、Widget 也属于用于 UI 绘制的类，因此他们也可以进行滚动条设置：

```
Container c1 = new Container();
c1.setSize(400, 200);
c1.setAlign(UIObject.ALIGN_CENTER, UIObject.ALIGN_CENTER);
for (int i = 0; i < 10; i++) {
    Button btn = new Button("Button" + i);
    c1.addWidget(btn);
}
Flex layout = new Flex();
layout.setFlowType(Flex.FLEX_FLOW_TYPE_ROW);
c1.updateLayout(layout);

c1.setScrollbarMode(AbstractUIObject.SCROLLBAR_MODE_OFF);//设置滚动条展示样式
```

```
        c1.scrollToView();//滚动到显示出来
        c1.scrollToView(false);//参数代表是否立即展示，false 代表带动画滚动展示
        c1.scrollToX(0);//滚动 X 位置
        c1.scrollToX(0, false);//滚动到 x 位置，false 代表带动画滚动
        c1.scrollToY(0);//滚动到 Y 位置
        c1.scrollToY(0, false);//滚动到 Y 位置，第二个参数是代表是否立刻展示，false 表示会带着动画滚动展示
        c1.setScrollDirection(AbstractUIObject.SCROLL_DIR_ALL);//设置滚动方向
        c1.setScrollable(false);//是否可以滚动
        c1.setScrollListener(new ScrollListener() {//设置滚动监听

            public void onScrolling(UIObject uiobject) {
                // TODO Auto-generated method stub
            }

            public void onScrollEnd(UIObject uiobject) {
                // TODO Auto-generated method stub
            }

            public void onScrollBegin(UIObject uiobject) {
                // TODO Auto-generated method stub
            }
        });
```

字体设置

字体的实现类是 Font，下面是字体设置的示例，包括自定义字体设置方式：

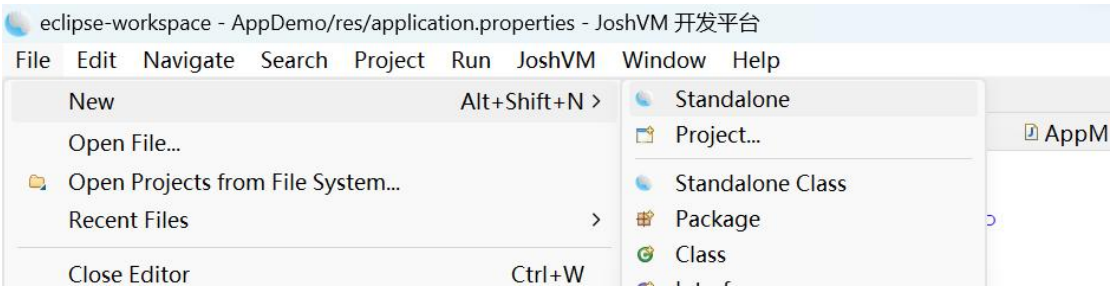
```
        Label l = new Label();
        l.setPos(100, 100);
        l.setPadding(10);
        l.setText("Hello World!\n" + "树 木");
        l.setBackgroundColor(Color.GREEN);
        Font font = Font.getFont(16);//设置字体的大小为 16，字体默认
        //      font = Font.getFont(24, Font.STYLE_ITALIC);//大小 24，意大利字体
        //      font = Font.getFont(32, Font.STYLE_BOLD | Font.STYLE_ITALIC);//大小 32，加粗意大利字体
            try {
             font = Font.createFont("/test.ttf", 24);//使用自定义字体库
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
1. setTextFont(font);//给文字设置字体
```

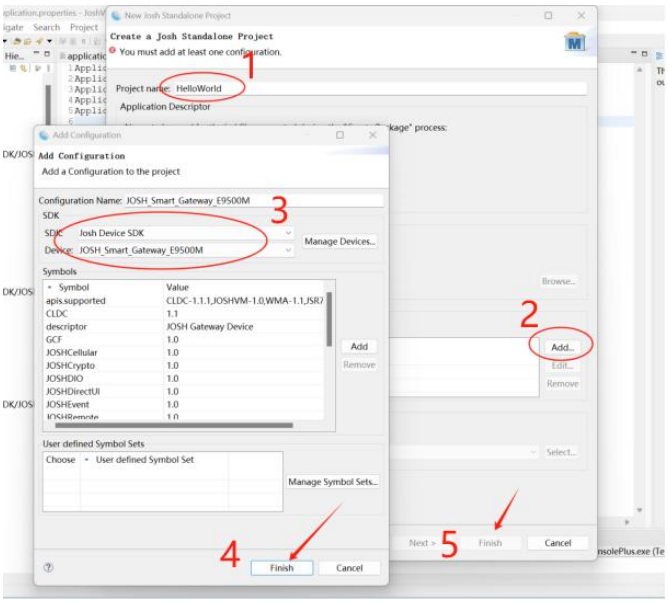
实践：创建第一个图形应用

IDE（版本号<1.2.20）

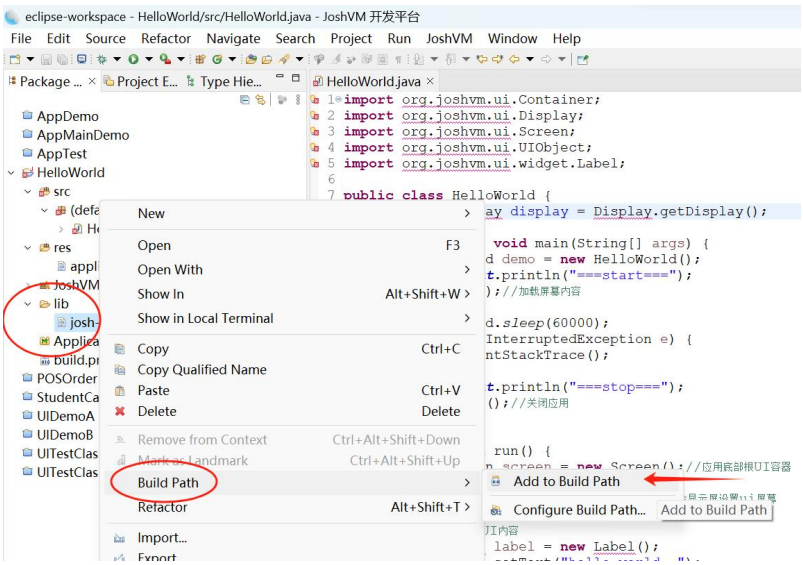
1.点击 IDE 的 File->New->Standalone



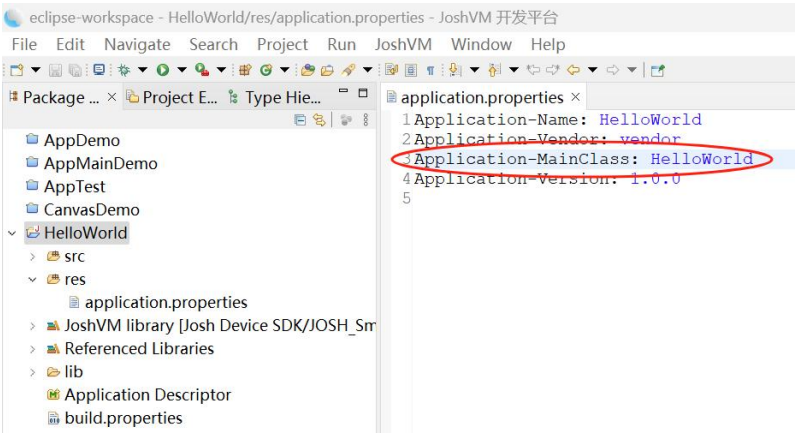
2.根据下图步骤，依次填写项目名称，点击 add 添加设备，选择 E9500M，点击 4 Finish，再点击 5 Finish，创建一个新项目。



3. 在创建的应用中新建 lib 文件夹，复制入需要 UI 框架 jar 包 josh-ui.jar，在 jar 包上右键选择 Build Path->Add to Build Path 导入。



4. 指定启动的主类，通过主类的 main 方法作为程序开始入口。



主类 HelloWorld.java 代码如下：

```
import org.joshvm.ui.Container;
import org.joshvm.ui.Display;
import org.joshvm.ui.Screen;
import org.joshvm.ui.UIObject;
import org.joshvm.ui.widget.Label;

public class HelloWorld {
    private Display display = Display.getDisplay();//获取显示屏对象
    public static void main(String[] args) {
        HelloWorld demo = new HelloWorld();
        System.out.println("===start===");
        demo.run();//加载屏幕内容
        try {
```

```
            Thread.sleep(60000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("===stop===");
        demo.stop();//关闭应用
    }

     private void run() {
            Screen screen = new Screen();//应用底部根 UI 容器
            Container container = screen;
            display.setScreen(screen);//给显示屏设置 ui 屏幕

            //设置 UI 内容
            Label label = new Label();
            label.setText("hello world ~");
            label.setAlign(UIObject.ALIGN_CENTER, UIObject.ALIGN_CENTER);
            container.addWidget(label);
        }

    private void stop() {
        display.stop();
    }}
```

5. 将程序下载到设备上运行起来，至此第一个 UI 程序已经创建完毕。

IDE（版本号>=1.2.20）

JOSH Studio 从 1.2.20 版本开始已经支持 GUI 设备，可以直接在 add 添加设备的时候，选择 JOSH_GUI_imx6ull，选择这个设备之后，**就不用再导入 UI 框架 jar 包 josh-ui.jar 到项目的 lib 目录下了**，可以直接使用 GUI 的 API。其他步骤同旧版本的 IDE。

同时我们为您提供了更多的官方 UI 控件使用 demo，期待您的下载体验。

基础功能 demo：

普通 UI 控件 demo：Widget

画布 demo ： UIDemoA.zip

UI 测试 demo：UITest.zip

App UI 界面 Demo：

1. 充电桩
2. 学生卡
3. Pos 点餐系统