

目录

简介	2
配置开发环境.....	2
Android.....	2
J2se.....	3
构建实例.....	3
Android.....	3
J2se.....	6
模块使用说明.....	10
1. Config.....	10
2. Arena.....	10
3. Scene.....	10
4. Panel.....	11
5. Util.....	11
6. DrawableHelper.....	11
7. Control.....	11
8. Component.....	11
9. 其它.....	11
配置执行环境.....	12
Android.....	12
Windows.....	12
Linux	12
执行效率测试.....	12
测试内容.....	12
Android.....	12
Windows.....	13
Linux	13

简介

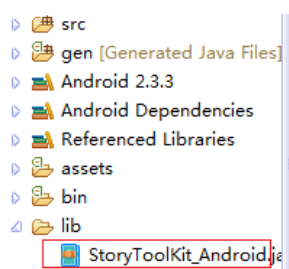
《StoryToolKit》（以下简称 STK）是一款用 java 语言开发的工具包，将其导入到您的 java 项目构建路径中，您就可以方便地构建一个“以指定刷新率对窗口进行绘图”的应用（通常指游戏）。当前版本（v2.10）包含 Android 和 J2se，其中 Android 版本可以应用在 Android 2.1 及以上版本系统中，更低的版本没有进行测试所以不推荐使用；J2se 版本可以执行在 Windows 和 Linux 系统的 JVM 中，其他系统的 JVM（如 OS X）未测试，不推荐使用。

其他平台版本（如 WP）正在开发中。

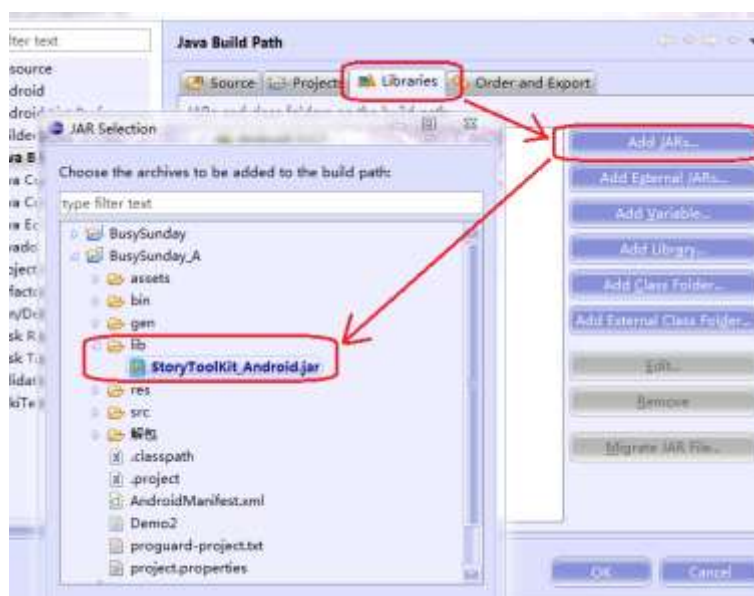
配置开发环境

Android

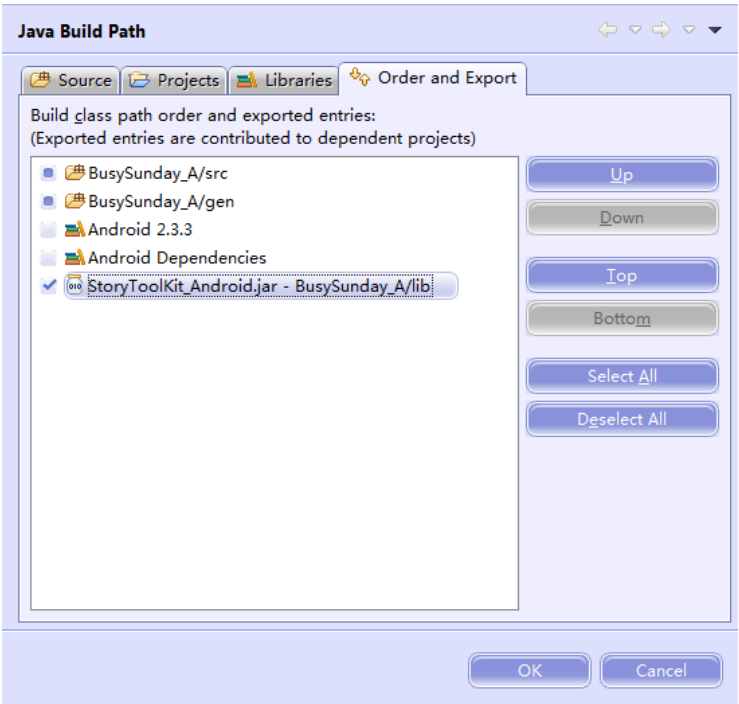
与通常在 Android 项目中引用第三方工具包的方式一样，或者可以参照下面的例子：



（图 1：在工程目录中新建“lib”文件夹，并将 STK 的 jar 文件拷贝放入）



（图 2：刷新项目后配置构建路径，导 STK 的库）



(图 3: **重要**，在“排序和导出”中选中 STK 库文件)

J2se

与通常在 Java 项目中引用第三方工具包的方式一样，在“导出”中无须选中 STK 库，示例略。

构建实例

Android

新建类或使用您在新建项目时建立的主 Activity 类，假设其命名为“MainActivity.java”。首先让我们来看一下通常情况下此 Activity 类的代码：

```
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

(图 4: 通常情况下新创建的 Activity 主类)

@ Wally

现在让我们对其作一些修改：继承类由 `Activity` 改为继承 `STK` 类。如果您使用 Eclipse 作为 IDE，这时会提示有必须实现的方法（`init()`：游戏初始化各种参数的方法，`perform()`：游戏逻辑代码的执行方法）。加入后我们需要的主执行类就改造好了：

```
public class MainActivity extends STK {
    // /** Called when the activity is first created. */
    // @Override
    // public void onCreate(Bundle savedInstanceState) {
    //     super.onCreate(savedInstanceState);
    //     setContentView(R.layout.main);
    // }

    @Override
    public void init() {
        // TODO Auto-generated method stub
    }

    @Override
    public void perform() {
        // TODO Auto-generated method stub
    }
}
```

（图 5：改造后的 Activity 主类）

一切准备完毕，让我们来写一些测试代码：

```
public class MainActivity extends STK {

    @Override
    public void init() {
        // 定义游戏尺寸
        Config.setGameSize(640, 480);
        // 定义游戏资源文件的根目录
        Config.setRootPath("/sdcard/wafly/demo_3/data");
        // 游戏窗口显示模式
        Config.setDisplayMode(Config.DisplayMode.STRETCH);
        // 设置游戏背景色为紫色
        Config.setBackgroundColor(0xffff00ff);
        // 开启调试模式
        Config.enableDebug();
        // 设置游戏字体大小
        Config.setGlobalFontSize(30);
        // 设置游戏最大帧数
        Config.setMaxFPS(30);
    }

    @Override
    public void perform() {
        // 初始化一个图片
        DrawableHelper background = new DrawableHelper("map.jpg");
        // 设置当前游戏背景
        Arena.getScene().setBackground(background, 0, 0, null);
    }
}
```

(图 7: 测试代码, 基于改造后的 Activity 主类)

执行效果:



(图 8: 将图片资源放置到 SD 卡正确位置后, 在模拟器中执行测试代码)

J2se

新建一个包含 main 函数的执行类, 并继承 STK 类, 加入游戏初始化方法和执行方法。然后在 main 函数中对执行类进行初始化:

```
public class MainActivity {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

(图 9: 新建的主执行类)

@ Wally

J2se 的代码除了 main 函数，其他部分和 Android 版本的完全一样。也就是说，J2se 版本的代码在测试完毕后可以拿到 Android 项目中少量修改后使用。现在让我们将 Android 版本的测试代码复制过来：

```
public class MainActivity extends STK {

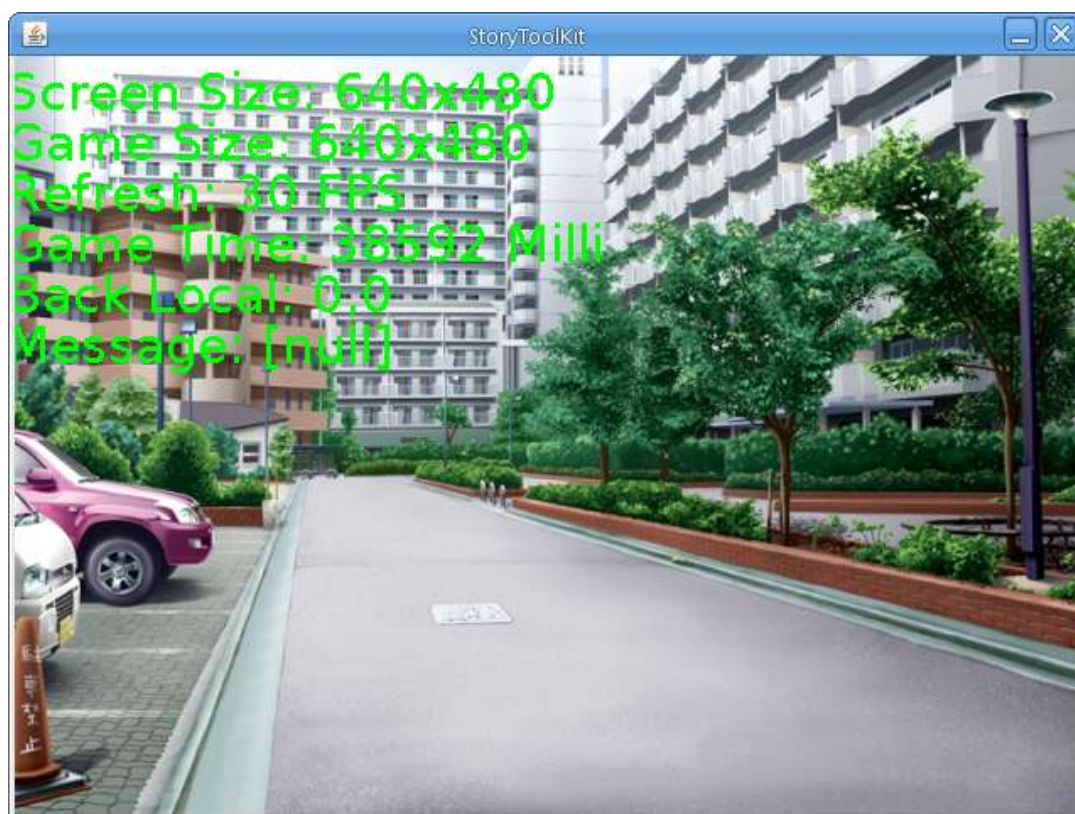
    @Override
    public void init() {
        // 定义游戏尺寸
        Config.setGameSize(640, 480);
        // 定义游戏资源文件的根目录
        Config.setRootPath("/sdcard/wafly/demo_3/data");
        // 游戏窗口显示模式
        Config.setDisplayMode(Config.DisplayMode.WINDOW);
        // 设置游戏背景色为紫色
        Config.setBackgroundCoLor(0xffff00ff);
        // 开启调试模式
        Config.enableDebug();
        // 设置游戏字体大小
        Config.setGlobalFontSize(30);
        // 设置游戏最大帧数
        Config.setMaxFPS(30);
    }

    @Override
    public void perform() {
        // 初始化一个图片
        DrawableHelper background = new DrawableHelper("map.jpg");
        // 设置当前游戏背景
        Arena.getScene().setBackground(background, 0, 0, null);
    }

    public static void main(String[] args) {
        new MainActivity();
    }
}
```

（图 10：测试代码，可以与图 7 比较一下）

执行效果：



（图 11：Linux 下的执行效果，可以与图 8 比较一下）

在 windows 下时路径有些不同，让我们修改一下代码：（当然您也可以使用相对路径）


```
public class MainActivity extends STK {

    @Override
    public void init() {
        // 定义游戏尺寸
        Config.setGameSize(640, 480);
        // 定义游戏资源文件的根目录, 修改为Windows格式
        Config.setRootPath("D:\\wafly\\demo_3\\data");
        // 游戏窗口显示模式
        Config.setDisplayMode(Config.DisplayMode.WINDOW);
        // 设置游戏背景色为紫色
        Config.setBackgroundColor(0xffff00ff);
        // 开启调试模式
        Config.enableDebug();
        // 设置游戏字体大小
        Config.setGlobalFontSize(30);
        // 设置游戏最大帧数
        Config.setMaxFPS(30);
    }

    @Override
    public void perform() {
        // 初始化一个图片
        DrawableHelper background = new DrawableHelper("map.jpg");
        // 设置当前游戏背景
        Arena.getScene().setBackground(background, 0, 0, null);
    }

    public static void main(String[] args) {
        new MainActivity();
    }
}
```

(图 12: 修改为 Windows 路径的代码)

执行效果:



(图 13: Windows 下的执行效果, 可以与图 8、图 11 比较一下)

模块使用说明

1. Config

全局静态类, 用于配置各类游戏参数。

使用方法: 直接调用其中 API 即可。

2. Arena

STK 的基础建筑, 承重于各种控件、模块之下, 并附带一些全局方法, 如播放视频、播放背景音乐等。

使用方法: 直接调用其中 API 即可。

3. Scene

用于对特定游戏场景的展示。其内包含对场景的各类操作方法, 如切换背景、承载模块等。您也可以继承此类重构自己的 Scene 类型, 用以满足您对不同游戏类型的需求。(可以参照 STK 自带的特殊场景“地图行走”、“剧本对话”)

使用方法: 通过构造一个 Scene 对象来使用。得到 Scene 实例后, 使用 Arena 的 Arena.setScene(Scene) 方法来切换当前场景。每个时刻在 Arena 中只存在一个 Scene 实例。

4. Panel

一个面板，在其上放置各类控件，并在当前场景中显示。可用于显示角色状态、存档画面等。

使用方法：通过构造一个 Panel 对象使用。得到实例后，使用 panel.show()方法在当前场景中显示。可以同时存在多个 Panel。当关闭一个 Panel 时会自动显示其下方未关闭的 Panel，直到所有 Panel 均被关闭。可以只用 scene 对象的 scene.closeAllPanel()方法关闭当前场景的所有 Panel。

5. Util

通用工具类，其内包含一些程序的常用方法，如记录日志、获取游戏参数等。可以在任意地方使用。

使用方法：直接调用其中 API 即可。

6. DrawableHelper

绘图助手，用于创建一幅图片或在这个图片上绘制其他东西。

使用方法：（示例）

```
DrawableHelper img = new DrawableHlper("C:\\1.jpg"); //通过图片路径构建一个图片  
img.drawLine(0,0,100,100); //在这个图片上再画一条直线
```

7. Control

一个包含指定位置和大小显示对象的概念，拥有点击（触控）事件。包括按钮（Button）、文字标签（Label）、图片框（PictureBox）.....等，各自功能不同。可以加入 Panel 组中一起显示，或个体直接在当前场景中显示。

使用方法：构建实例后通过 scene 的 addControl()或 panel 的 addControl()添加；通过重载控件并实现指定的方法可以添加控件事件。

8. Component

一个包含多种组合的显示对象的概念，包括对话框（TalkBox）、天气（weather）、精灵（Spirit）.....等。

使用方法：构建实例后通过 scene 的 addComponent()添加。

9. 其它

请参考 API 索引文档

配置执行环境

Android

在构建路径“排序和导出”中将 STK 类库勾选上，无须特别配制，直接导出 apk 安装包即可

Windows

在项目下载地址（<http://code.google.com/p/storytoolkit/downloads/list>）找到“执行环境 (Windows).zip”并下载解包，参考其内的配置说明。也可以自行将游戏 jar 包和 STK 类库转换为可执行程序。

Linux

在项目下载地址（<http://code.google.com/p/storytoolkit/downloads/list>）找到“执行环境 (Linux).zip”并下载解包，参考其内的配置说明。也可以自行将游戏 jar 包和 STK 类库打包为 Linux 格式安装程序。

执行效率测试

以下数据为各个平台真实的实测数据，会受到其系统内部各个应用程序的影响而存在一定的偏差，因此数据仅供参考，不作为发布标准。

测试内容

游戏尺寸：800 x 600

游戏内容：切换背景，切换对话场景，切换精灵行走场景

Android

测试机型：Moto Millstone（里程碑）

系统版本：MIUI2.3

帧率：20 FPS - 26 FPS

测试机型：中兴 N760

系统版本：2.3

帧率：10 FPS - 15 FPS

测试机型：联想 A68e

系统版本: 2.2

帧率: 18 FPS – 22 FPS

Windows

测试机型: 东芝 M400

系统版本: Windows XP SP3

CPU: 双核 1.83 G

内存: 1 GB x 2

帧率: 最大帧 60 FPS, 满速

测试机型: 联想 Z370

系统版本: Windows 7 Home Premium

CPU: 双核 2.3 G

内存: 2 GB + 4 GB

帧率: 最大帧 60 FPS, 满速

Linux

测试机型: 东芝 M400

系统版本: Deepin Linux 11.12

CPU: 双核 1.83 G

内存: 1 GB x 2

帧率: 最大帧 60 FPS, 满速

测试机型: 虚拟机

系统版本: Ubuntu 8.04

CPU: 双核 2.3 G

内存: 2 GB

帧率: 最大帧 60 FPS, 满速