Green Throttle SDK

Developer Guide

Version 1.2

2013-03-05

TABLE OF CONTENTS

Chapter 1 Welcome 1

Introducing the Green Throttle Experience 1

Audience 2

How to Use This Document 3

Document History 4

Chapter 2 Exploring the Green Throttle SDK 5

Exploring the Green Throttle Architecture 5

Exploring the Green Throttle SDK 7

Introducing the Atlas Controller 7

Chapter 3 Installing the Green Throttle SDK 8

Installation Overview 8

Checking System Requirements 9

Installing the Green Throttle SDK 9

Verifying Your Environment 14

Chapter 4 Integrating with Java Apps Using the Java Input Unifier 15

Introducing the Green Throttle Java Input Unifier 15

Accessing the API Reference Documentation 16

Using the Green Throttle Java Input Unifier 17

Chapter 5 Integrating with Unity-Based Games 20

Introducing the Green Throttle Unity Plugin 20

Using the Green Throttle Unity Plugin 21

Chapter 6 Integrating with Marmalade-Based Games 25

Introducing the Green Throttle Marmalade Extension 25

Using the Green Throttle Marmalade Extension 26

Chapter 7 Integrating with Corona Enterprise-Based Games 30

Introducing the Green Throttle Corona Enterprise Plugin 30

Exploring Green Throttle Constants 31

Preparing to Use the Green Throttle Corona Enterprise Plugin 33

Using the Green Throttle Corona Enterprise Plugin 34

Chapter 8 Integrating with Java Apps Using the Green Throttle Java API 37

Introducing the Green Throttle SDK for Java 37

Before Using the Green Throttle SDK 38

Accessing the API Reference Documentation 38

Configuring Your Project 39

Exploring the Green Throttle API 39

Using the Green Throttle Java API 42

Chapter 9 Integrating with Green Throttle Arena 45

Introducing Green Throttle Arena 45

Featuring Your Game in the Green Throttle Arena 47

Appendix A Using the GT Controller Test App 48

Using the GT Controller Test App 48

Appendix B Google Play and Green Throttle Arena Game Assets 51

Appendix C Frequently Asked Questions 55

Supported Engines and Devices 55

Development 55

Promotion 56

Appendix D Additional Resources 57

Green Throttle Resources 57

General Development Resources 57

Chapter 1
Welcome

Welcome to the Green Throttle SDK Developer Guide. This document contains everything you need to get started in integrating your Android-based, multiplayer game with the Green Throttle Atlas controller, creating an unparalleled big-screen television gaming experience using true analog wireless gaming controllers.

Using this guide, you will learn how to add analog controllers to your Android games, allowing you to deliver the precision and sophisticated control that users have come to expect only from dedicated console gaming systems.

This guide introduces you to the Green Throttle SDK and describes how to install the software. The guide then explains how to integrate the Green Throttle technology with your Java, Unity, Marmalade, and Corona Enterprise-based games. The guide further describes a series of best practices for integrating with Green Throttle Arena, the new premium online gaming hub.

Introducing the Green Throttle Experience

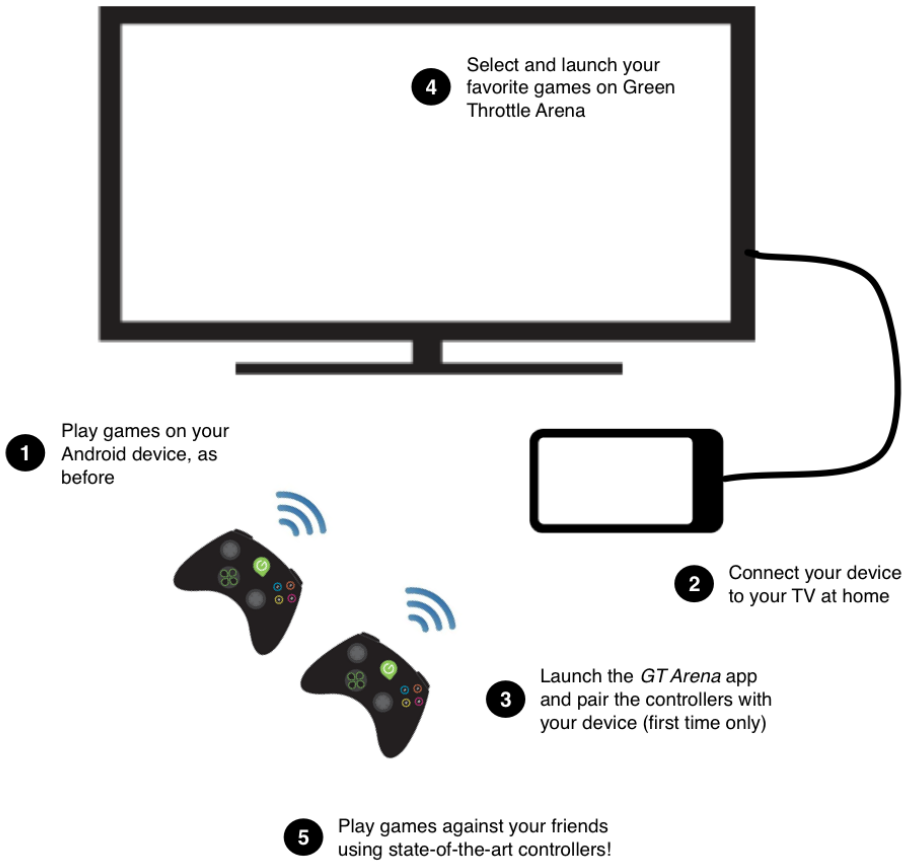
Green Throttle reimagines computer gaming by allowing you to take games that people enjoy playing on their mobile devices and turn them into a multiplayer, HDTV experience using state-of-the-art game controllers.

From a user's perspective, this rich new gaming experience works as follows:

1. Users play mobile games on their Android-based phones and tablets while on-the-go, as before.
2. After getting home (or while visiting a friend's home), users connect their Android device to a big-screen television using a Mobile HD TV Connector and HDMI cable.
3. Users launch the *GT Arena* app on their Android device, which serves as a portal into the Green Throttle Arena allowing users to discover their favorite Green Throttle-enabled games.

   Users also use the *GT Arena* app to pair Atlas controllers with their Android device (first time only).
4. Users select and launch a Green Throttle-enabled game, navigating on the big-screen TV using their Atlas controller.
5. Users play their favorite games against their friends using true, state-of-the-art wireless, gaming controllers.

The following figure illustrates the Green Throttle user experience:



The Green Throttle Experience

Green Throttle controllers feature full analog control sticks allowing players to have precise control and fluid motion for fast action games.

AUDIENCE

This guide is intended for developers who want to integrate multiplayer Android games with Green Throttle Atlas controllers for play on big-screen televisions.

To create Java-based games, this guide assumes that you are familiar with the Java programming language, Android application development, and how to use the Android SDK. To create Unity-based games, this guide assumes that you are familiar with the Unity game engine and associated technologies including the C# language.

Similarly, to create Marmalade-based games, this guide assumes that you are familiar with the Marmalade game engine and associated technologies including the C++ language. Finally, to create Corona Enterprise-based games, this guide assumes that you are familiar with Corona Enterprise and associated technologies including the Lua programming language.

This guide further assumes that you are familiar with an Integrated Development Environment (IDE), such as Eclipse or a similar environment.

How to Use This Document

This guide is organized as follows:

- *Chapter 1: Welcome* introduces this guide and describes the types of information available.
- *Chapter 2: Introducing the Green Throttle SDK* provides an overview of the Green Throttle architecture and describes the software development kit.
- *Chapter 3: Installing the Green Throttle SDK* describes the system requirements and shows how to install the software development kit in your environment.
- *Chapter 4: Integrating with Java Apps Using the Java Input Unifier* describes how to use the Java Input Unifier package to integrate Green Throttle technology with your Java-based game.
- *Chapter 5: Integrating with Unity-Based Games* describes how to integrate the Green Throttle system with your Unity-based game.
- *Chapter 6: Integrating with Marmalade-Based Games* describes how to integrate the Green Throttle system with your Marmalade-based game.
- *Chapter 7: Integrating with Corona Enterprise-Based Games* describes how to integrate the Green Throttle system with your Corona Enterprise-based game.
- *Chapter 8: Integrating with Java Apps Using the Green Throttle Java API* describes how to use the Java API to integrate the Green Throttle system with your Java-based game.
- *Chapter 9: Integrating with Green Throttle Arena* describes a series of best practices and recommendations for integrating with the Green Throttle Arena.
- *Appendix A: Using the GT Game Controller Test App* describes the bundled app that enables you to scan and connect game controllers to the Green Throttle service.
- *Appendix B: Google Play and Green Throttle Arena Game Assets* describes the game assets that you need to supply to support your games in both the Google Play market and the Green Throttle Arena game portal.
- *Appendix C: Frequently Asked Questions* lists the most frequently asked questions about the Green Throttle SDK and developing within the Green Throttle ecosystem.
- *Appendix D: Additional Resources* offers a range of additional resources that you can access to help you develop your Green Throttle games.

DOCUMENT HISTORY

The Green Throttle SDK Developer Guide is available directly from Green Throttle Games, Inc.

Version	Date	Description
1.0	December 12, 2012	Initial release.
1.1	February 26, 2013	Adds a new chapter about integrating Java apps using the Java Input Unifier, as well as new chapters about integrating Marmalade and Corona Enterprise-based games with Green Throttle technology. Adds details about how to integrate with (and supply assets for) Green Throttle Arena.
1.2	March 5, 2013	Adds a new section describing how to prepare to use the Green Throttle Corona Enterprise plugin. Corrects the package name for testing games in the Green Throttle Arena.

Chapter 2

Exploring the Green Throttle SDK

The Green Throttle SDK makes it easy to bring existing mobile games to the Green Throttle platform by integrating your Android-based, multiplayer games with the Green Throttle Atlas controller. Using the Green Throttle SDK, you can move beyond touch, and free your imagination from the constraints of mobile devices and small screens.

This chapter describes the Green Throttle architecture and introduces the Green Throttle SDK.

Exploring the Green Throttle Architecture

The Green Throttle system is designed to enable you to rapidly integrate advanced controller support into your Android-based games. The Green Throttle SDK supports the following application development environments:

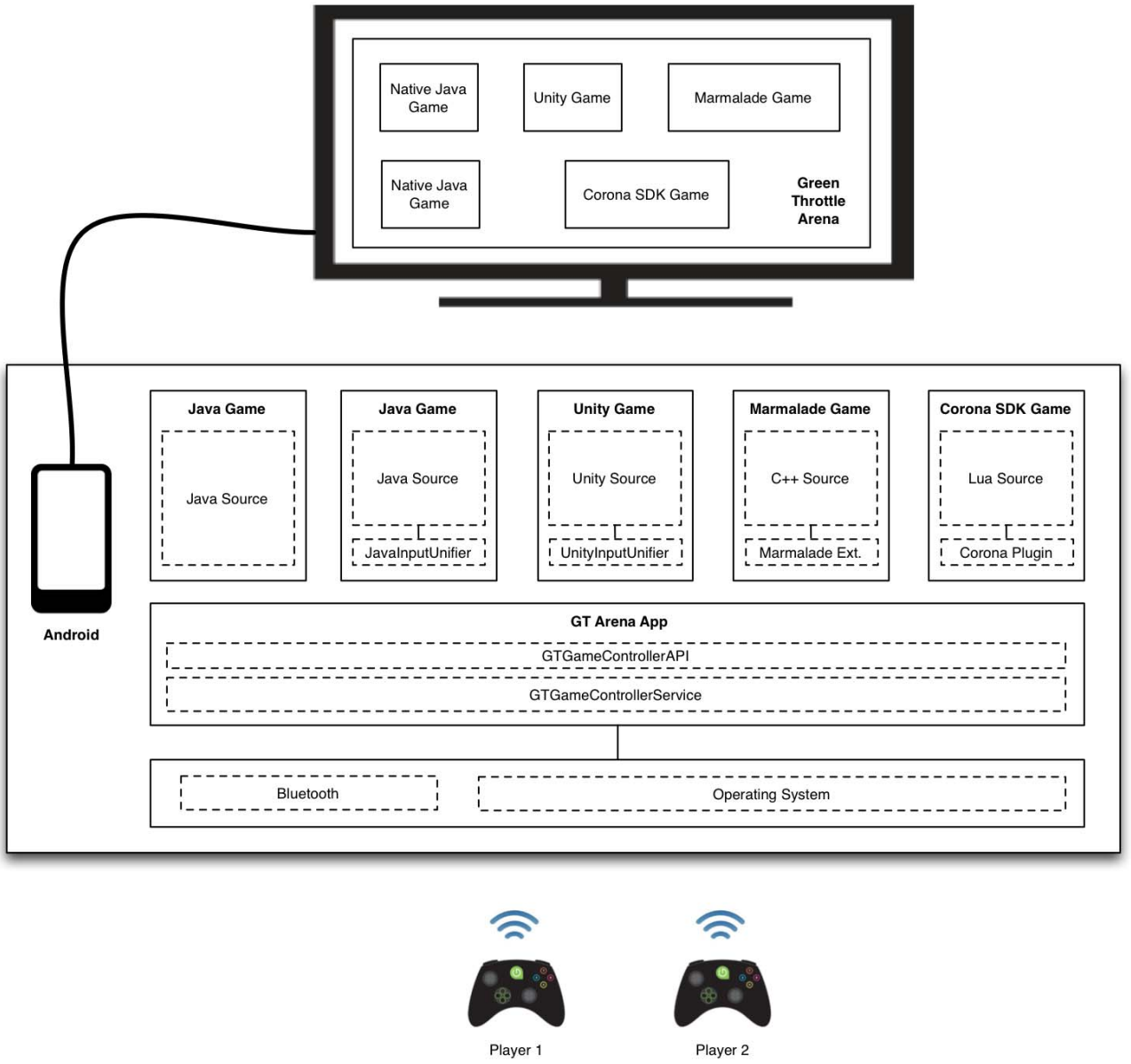
- Java
- Unity Game Engine
- Marmalade Game Engine
- Corona Enterprise

At the heart of the system is the Green Throttle Controller Service that manages all interactions between your game and Bluetooth-enabled Green Throttle Atlas controllers. To help integrate Java-based games, Green Throttle offers a Java Input Unifier package that automates the binding of controllers to your games and greatly simplifies the process of monitoring for controller status updates.

The Green Throttle SDK also includes an advanced Java API offering an extra level of control, allowing direct access to controller events (such as button presses and stick movements) and controller status events (such as controller connects and disconnects).

Integrating with Unity, Marmalade, or Corona Enterprise-based games is just as easy using the corresponding Green Throttle plugins or extensions, which automatically bind to the Green Throttle Controller Service and enables your game to access the current state of the game controllers either through a series of controller state structures (Unity) or using a simplified API (Marmalade and Corona Enterprise).

The following provides an overview of the Green Throttle architecture:



Green Throttle Architecture

NOTE: The Android phone or tablet requires a video-out port to connect to a television. Green Throttle supports up to four controllers connected to the same device.

Exploring the Green Throttle SDK

The Green Throttle SDK is available as a zip file on the Green Throttle web site (developer.greenthrottle.com). The Green Throttle SDK includes the following:

- Green Throttle API (zip file)
- GT Controller Test App (Android package)
- GT Java Input Unifier
- GT Unity Plugin
- GT Marmalade Extension
- GT Corona Enterprise Plugin
- GT Corona Simulator Plugin
- Java Input Test sample application (source code)
- Unity Input Test sample application (source code)
- Green Throttle SDK Developer Guide (this document)
- Green Throttle Design Standards Guide (PDF)

Introducing the Atlas Controller

With true analog control, Atlas controllers are the first Android Bluetooth controllers to provide console-level precision and accuracy. The following shows the layout of the Green Throttle Atlas controller:

Green Throttle Atlas Controller

Chapter 3
Installing the Green Throttle SDK

You need to install the Green Throttle SDK on your development computer before you can begin integrating your games with the Green Throttle controllers and ecosystem.

This chapter provides an overview of the installation process, and guides you through the steps to check the system requirements, install the SDK on your system, and verify that the installation completed successfully.

Installation Overview

This section provides an overview of the procedure to install the Green Throttle SDK on your system.

To install the Green Throttle SDK

1. Check the system requirements for the Green Throttle SDK.

   This involves verifying that the Android SDK and a compatible IDE are properly installed and configured on your system. For more information, see Checking System Requirements.

2. Install the Green Throttle SDK.

   This includes unzipping the Green Throttle SDK zip file and optionally installing the Green Throttle Controller Test app on your Android device. For more information, see Installing the Green Throttle SDK.

3. Download and install the *GT Arena* app using the *Google Play* market or the *Amazon Appstore for Android* (for Amazon Kindle Fire devices).

   You use the *GT Arena* app to connect controllers to the Green Throttle Controller Service (which is embedded in the app).

4. (Optional) Verify your development environment.

   You can verify your development environment and that you have successfully installed the Green Throttle SDK by building and running the demo application. For more information, see Verifying Your Environment.

Checking System Requirements

Before installing the Green Throttle SDK, verify that your environment matches the following system requirements:

Software	URL
Oracle Java SE 6 JDK 32-bit (not Java SE 7 SDK)	http://www.oracle.com/technetwork/java/javase/downloads/index.html
Google Android SDK Android 4.0 (Ice Cream Sandwich) Android 4.1 (Jelly Bean)	http://developer.android.com/sdk/index.html
Unity Game Engine (version 3.5 and 4.0)	http://unity3d.com/
Marmalade Game Engine (version 6.1)	http://www.madewithmarmalade.com
Corona Enterprise (current version)	https://www.coronalabs.com
Eclipse IDE – Juno (version 4.2 or higher)	http://www.eclipse.org/downloads/

You can use IDEs other than Eclipse, however, this guide describes how to integrate the Green Throttle SDK only on Eclipse.

Installing the Green Throttle SDK

You can install the Green Throttle SDK for the following environments:

- Java
- Unity
- Marmalade
- Corona Enterprise

INSTALLING THE SDK FOR JAVA DEVELOPMENT

The Green Throttle SDK is delivered as a zip file and includes software to help you integrate Green Throttle with Java-based games. You do not need to perform these steps if you are integrating Unity, Marmalade, or Corona Enterprise-based games.

To install the Green Throttle SDK for Java

1. Prepare your environment for Green Throttle Java application development.

   Option 1

   a. Download and install the Java SE 6 SDK (32-bit), if necessary.

      NOTE: The Android SDK does not currently support the Java SE 7 SDK.

   b. Download and install the ADT Bundle.

      The ADT Bundle includes everything you need to start developing apps, including a version of the Eclipse IDE with built-in ADT (Android Developer Tools). Use this option if you are creating a new development environment from scratch.

      For more information, see http://developer.android.com/sdk/installing/bundle.html.

   Option 2

   a. Download and install the Java SE 6 SDK (32-bit), if necessary.

      NOTE: The Android SDK does not currently support the Java SE 7 SDK.

   b. Download and install the Eclipse IDE, if necessary. Use either *Eclipse Classic* or *Eclipse for Java Developers*.

      For more information, see http://www.eclipse.org/downloads/.

   c. Download and install the Android SDK Tools for your development platform (Windows, Mac OS, or Linux).

      The SDK Tools package is not the complete SDK environment. It includes only the core SDK tools, which you can use to download the rest of the SDK packages, including the latest system image.

      Use this option if you have an existing development environment or feel more comfortable installing the components manually. On Windows, the download package is an executable file that launches an installer.

      For more information, see http://developer.android.com/sdk/installing/index.html.

   d. Install and configure the ADT Plugin.

      The Android Development Tools (ADT) Plugin extends the capabilities of Eclipse, allowing you to set up new Android projects, build an app user interface, debug your app, and export signed (or unsigned) app packages (APKs) for distribution.

      For more information, see http://developer.android.com/sdk/installing/installing-adt.html.

2. Download the Green Throttle SDK from the Green Throttle web site.

   Navigate to http://developer.greenthrottle.com, click the **Get Started** button, and click the **Download SDK** button. After accepting the SDK license agreement, the `GreenThrottleSDK<version>.zip` file downloads to your machine, (where <version> is the version number of the Green Throttle SDK).

3. Unzip the `GreenThrottleSDK<version>.zip` file.

   When developing using Java, make note of the following files in the directory that is created after unzipping:

   - `JavaInputUnifier.jar` — The Green Throttle Java Input Unifier package, which provides a high-level interface for integrating Atlas controllers.
   - `GTGameControllerApi.zip` — A zip file containing the Green Throttle Controller API.
   - `GTControllerTestApp<version>.apk` — The Green Throttle Test app, which enables you to display events in real-time as they are received from the connected controllers.

4. (Optional) Unzip the `GTGameControllerApi.zip` file.

   This is necessary only if you plan to use the low-level Green Throttle Controller API (instead of the higher-level Green Throttle Java Input Unifier).

The Green Throttle SDK for Java is now installed on your system.

Installing the SDK for Unity

The Green Throttle SDK is available as a Unity plugin accessible through the Unity Asset Store.

Note: You do not need to perform these steps if you are integrating Java-based games.

To install the Green Throttle SDK for Unity

1. Prepare your environment for Unity application development.

   For more information, refer to the Unity Technologies web site at http://unity3d.com.

2. Open the Green Throttle Controller Plugin in your Unity environment.

   Use the following URL:

   https://www.assetstore.unity3d.com/#/content/6853

   Alternatively, you can download the Green Throttle SDK from the Green Throttle web site by doing the following:

   a. Navigate to http://developer.greenthrottle.com, click the **Get Started** button, and click the **Download SDK** button.

      After accepting the SDK license agreement, the `GreenThrottleSDK<version>.zip` file downloads to your machine (where <version> is the version number of the Green Throttle SDK).

b. Unzip the GreenThrottleSDK<version>.zip file.

   When developing using the Unity game engine, make note of the following files in the directory that is created after unzipping:

   - `GTUnityPlugin<version>.zip` — A zip file containing the Green Throttle assets for Unity development.
   - `GTControllerTestApp<version>.apk` — The Green Throttle Controller Test app, which enables you to display events in real-time as they are received from the connected controllers.

c. Copy the `GTUnityPlugin<version>.zip` file to the root directory of your Unity project, and unzip the file.

   The contents of the `GTUnityPlugin<version>.zip` file expand to the corresponding folders in your Unity project, as follows:

   - Assets/Plugins/Android/AndroidManifest.xml
   - Assets/Plugins/Android/UnityInputUnifier.jar
   - Assets/Plugins/GreenThrottle/GreenThrottle.cs

3. (Optional) Install the `GTControllerTestApp<version>.apk` package on your Android device.

The Green Throttle SDK for Unity is now installed on your system.

INSTALLING THE SDK FOR MARMALADE

The Green Throttle SDK is delivered as a zip file and includes the Green Throttle Marmalade extension as well as the Green Throttle Controller Test app.

NOTE: You do not need to perform these steps if you are integrating Java-based games.

To install the Green Throttle SDK for Marmalade

1. Prepare your environment for Marmalade application development, as appropriate.

   For more information, refer to the Marmalade web site at http://www.madewithmarmalade.com.

2. Download the Green Throttle SDK from the Green Throttle web site.

   Navigate to http://developer.greenthrottle.com, click the **Get Started** button, and click the **Download SDK** button. After accepting the SDK license agreement, the `GreenThrottleSDK<version>.zip` file downloads to your machine (where <version> is the version number of the Green Throttle SDK).

3. Unzip the `GreenThrottleSDK<version>.zip` file.

   When developing using the Marmalade game engine, make note of the following files in the directory that is created after unzipping:

   - `GTMarmaladeExtension<version>.zip` — A zip file containing the Green Throttle Marmalade extension.
   - `GTControllerTestApp<version>.apk` — The Green Throttle Controller Test app, which enables you to display events in real-time as they are received from the connected controllers.

4. Unzip the `GTMarmaladeExtension<version>.zip` file, and copy the expanded contents to the `Extensions` folder in your Marmalade environment.

5. (Optional) Install the `GTControllerTestApp<version>.apk` package on your Android device.

The Green Throttle SDK is now available on your system.

INSTALLING THE SDK FOR CORONA ENTERPRISE

The Green Throttle SDK is delivered as a zip file and includes the Green Throttle Corona Enterprise plugin as well as the Green Throttle Controller Test app.

NOTE: You do not need to perform these steps if you are integrating Java-based games.

To install the Green Throttle SDK for Corona Enterprise

1. Prepare your environment for Corona Enterprise application development, as appropriate.

   For more information, refer to the Corona Enterprise web site at https://www.coronalabs.com.

2. Download the Green Throttle SDK from the Green Throttle web site.

   Navigate to http://developer.greenthrottle.com, click the **Get Started** button, and click the **Download SDK** button. After accepting the SDK license agreement, the `GreenThrottleSDK<version>.zip` file downloads to your machine (where <version> is the version number of the Green Throttle SDK).

3. Unzip the `GreenThrottleSDK<version>.zip` file.

   When developing using the Corona Enterprise game engine, make note of the following files in the directory that is created after unzipping:

   - `GTCoronaEnterprisePlugin<version>.zip` — A zip file containing the Green Throttle Corona Enterprise plugin.
   - `GTCoronaSimulatorPlugin<version>.zip` — A zip file containing the Green Throttle Corona Simulator plugin.
   - `GTControllerTestApp<version>.apk` — The Green Throttle Controller Test app, which enables you to display events in real-time as they are received from the connected controllers.

4. Unzip the `GTCoronaEnterprisePlugin<version>.zip` file, and copy the expanded contents to the `Plugins` folder in your Corona Enterprise environment.

5. (Optional) Unzip the `GTCoronaSimulatorPlugin<version>.zip` file, and copy the `plugin_greenthrottle.lua` file to the `/Library/Application/Support/Corona/Simulator/Plugins` folder.

   You can create this folder, if necessary.

6. (Optional) Install the `GTControllerTestApp<version>.apk` package on your Android device.

The Green Throttle SDK is now available on your system.

VERIFYING YOUR ENVIRONMENT

You can verify that you have successfully installed the Green Throttle SDK by importing and building the JavaInputTest project using your integrated development environment.

To build the JavaInputTest app

1. Using Eclipse, import the `JavaInputTest` project.

   Do the following:

   a. Choose **File > New > Project** from the main menu The *New Project* wizard launches.

   b. Select **Android > Android Project from Existing Code** in the *Wizards* list, and click **Next**. The *Import Projects* dialog appears.

   c. Click **Browse** and select the root directory of the `JavaInputTest` project.

   d. Click **Finish**.

   e. Click **OK**.

2. Build and install the app on your Android device.

Chapter 4

Integrating with Java Apps
Using the Java Input Unifier

The Green Throttle SDK includes a Java Input Unifier package that makes it easy to add support for advanced analog controllers to your Java-based games on Android phones and tablets.

This chapter introduces the Green Throttle Java Input Unifier and describes how to use the package to monitor controller status updates within your Java-based game.

NOTE: In addition to the Java Input Unifier, the Green Throttle SDK also offers direct access to the Green Throttle Java API enabling an extra level of control when adding analog controller support to your native Java app. In general, however, it is easier to use the Green Throttle Java Input Unifier. In either case, you need to install the Green Throttle SDK for Java, as described in Installing the SDK for Java Development.

INTRODUCING THE GREEN THROTTLE JAVA INPUT UNIFIER

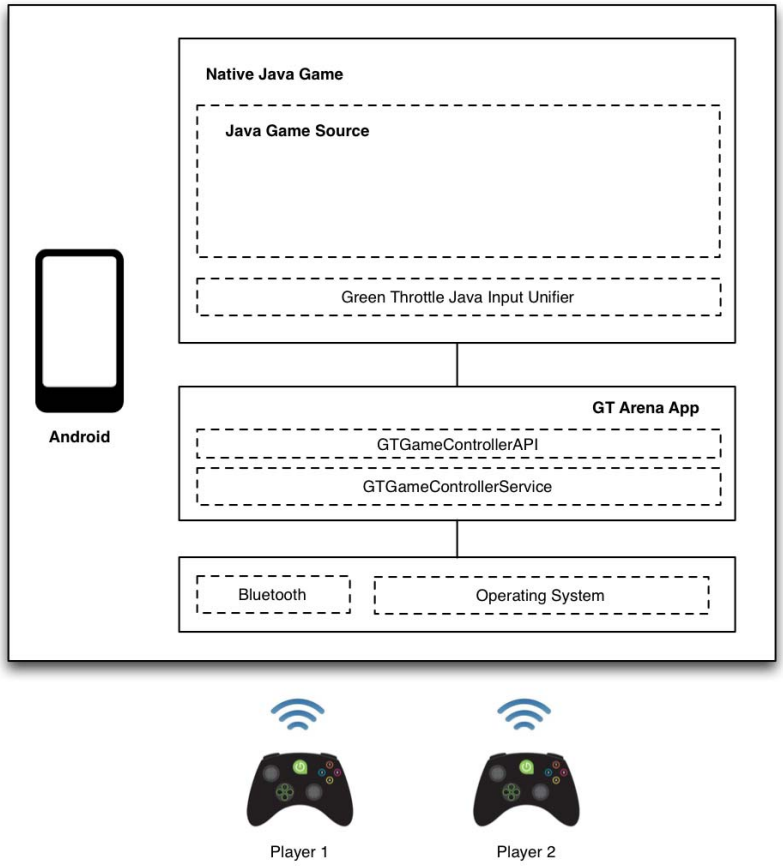
The Green Throttle SDK offers a Java Input Unifier package (`JavaInputUnifier.jar`) that automates the binding of controllers to your games and greatly simplifies the process of monitoring for controller status updates. The Java Input Unifier does this by including an abstract class (`GreenThrottleService`) that offers a set of callbacks that communicate the controller status to your game.

In this system, the Green Throttle Controller Service resides at the lowest level and manages all interactions with connected Atlas controllers. Within your code, you need to subclass and implement the `GreenThrottleService` abstract class (optionally by creating an inner class on the Android activity) and bind the activity to the Green Throttle Service. The service can then send button states and analog events to the application using the callback methods.

Alternatively, in more complex games (that require multiple activities), you can consider creating a singleton class that implements the abstract methods, allowing you to have the multiple activities in your application bind and unbind with the service, as required. In general, you can bind the lowest-level activity, as long as this activity does not call the `finish()` method.

NOTE: Green Throttle supports multiple bound activities to the Green Throttle Controller Service, but this scenario can significantly slow your application since the Green Throttle Controller Service sends a separate event to each activity.

The following provides an overview of the Green Throttle Java Input Unifier architecture:



Green Throttle Java Input Unifier Architecture

ACCESSING THE API REFERENCE DOCUMENTATION

The Green Throttle API documentation is available as Javadoc.

To access the API reference documentation

1. Expand the `gt-gamecontroller-doc<version>.zip` file.
2. Using a Web browser, open the `index.html` file in the `doc` folder.

USING THE GREEN THROTTLE JAVA INPUT UNIFIER

This section describes how to use the Green Throttle Java Input Unifier to monitor controller status updates within your Java-based game.

To use the Green Throttle Java Input Unifier

1. Add the `JavaInputUnifier.jar` file to your project.
2. Subclass the `GreenThrottleService` abstract class and implement the callback methods.

   The `GreenThrottleService` abstract class defines the following callback methods:

   - `ButtonAction()`
   - `AnalogEvent()`
   - `ControllerAction()`
   - `ServiceStatus()`

   NOTE: The Java Input Unifier also includes the `ControllerPlayer` class, which is a support class that serves as a wrapper for `controllerId`.

   The `ButtonAction` method is called when a button is pressed or released by a controller.

   ```
   protected abstract void ButtonAction(ControllerPlayer player,
                                        ControllerEvent.CommonCodes code,
                                        boolean pressed);
   ```

   Refer to the `ControllerEvent.CommonCodes` object in the Green Throttle API Javadoc for the list of possible button codes.

   The `AnalogEvent` method is called when an analog control sends a value update.

   ```
   protected abstract void AnalogEvent(ControllerPlayer player,
                                       ControllerEvent.CommonCodes code,
                                       float x, float y);
   ```

   Note that dual axis controls send both X and Y values while single axis controls only send X values (in this case, Y values are always zero). Refer to the `ControllerEvent.CommonCodes` object in the Green Throttle API Javadoc for the list of available values.

   The `ControllerAction` method is called when a controller connects or disconnects from the service.

   ```
   protected abstract void ControllerAction(ControllerPlayer player,
                                            boolean connected);
   ```

   The `ServiceStatus` method is called when the application connects or disconnects from the Green Throttle Controller Service.

   ```
   protected abstract void ServiceStatus(boolean isConnected);
   ```

3. Instantiate your implementation of the `GreenThrottleService` abstract class in the `onCreate()` method.

4. Bind the activity to the Green Throttle Controller Service.

   Use the following method:

   ```
   public void BindService(Activity activity)
   ```

   The `BindService()` method activates a connection to the Green Throttle Controller Service for the `activity` parameter. You should call the `BindService()` method in the `onCreate()` method of the activity (or soon after, as appropriate).

5. (Optional) Remap the left or right analog sticks to the D-pad or AYXB buttons respectively, as required.

   Use the following methods:

   ```
   void addLeftRemap()
   ```

   The left analog stick generates general analog data along with special analog D-pad events. The `addLeftRemap()` method remaps these analog D-pad events to regular D-pad events. This allows users to control menus using the D-pad or left analog stick without requiring your application to check every analog event.

   Note that if you remap the left analog stick, the original analog D-pad events are no longer sent.

   ```
   void removeLeftRemap()
   ```

   The `removeLeftRemap()` method removes the remapping between the left analog D-pad events and regular D-pad events.

   ```
   void addRightRemap()
   ```

   The `addRightRemap()` method remaps the right analog stick to the AYXB buttons (A is down, X is left, Y is up, and B is right). This allows users to generate button actions using the right analog stick (in the same cardinal directions as the buttons are laid out on the controller).

   Note that if you remap the right analog stick, the original analog buttons events are no longer sent.

   ```
   void removeRightRemap()
   ```

   The `removeRightRemap()` method removes the remapping between the right analog stick and the AYXB buttons.

6. (Optional) Determine the controller identifier for an event.

   The Java Input Unifier includes the `ControllerPlayer` class, which is a wrapper for the Bluetooth address and the controller identifier.

   Use the following method to determine the controller identifier for the event:

   ```
   public int getPlayerNumber()
   ```

   The `getPlayerNumber()` method returns a value between 1 and 4 corresponding to the light lit on the controller.

   Use the following method to determine the Bluetooth address of the controller sending the event:

   ```
   public String getControllerBtAddress()
   ```

   NOTE: You should rarely need to determine the Bluetooth address of the controller in your game.

7. (Optional) Hide the on-screen Android system user interface, if necessary.

   Use the following method:

   ```
   public void hideSystemUI(Activity activity)
   ```

   On-screen touch events typically display the Android system user interface. You can use the `hideSystemUI()` method to hide this interface, as required. In general, you should call the `hideSystemUI()` method in the `onCreate()` and `onResume()` methods.

8. Unbind the application from the Green Throttle Controller Service.

   Use the following method:

   ```
   public void UnbindService(Activity activity)
   ```

   You should call the `UnbindService()` method in the `onDestroy()` method of the activity to unbind from the Green Throttle Controller Service.

9. Build the Java project and deploy the package to the Android device.

10. Pair one or more game controllers with the Android device.

    Start the game while the controllers are searching for a connection (when all LEDs are flashing). The `JavaInputUnifier.jar` package binds to any searching controllers, assigning player numbers in the order that the controllers are bound.

Chapter 5

Integrating with Unity-Based Games

The Green Throttle SDK is compatible with the Unity Game Engine, making it easy to add support for advanced analog controllers to your Unity-based games on Android phones and tablets.

This chapter introduces the Green Throttle Unity plugin and describes how to use the plugin to monitor controller status updates within your Unity-based game.

NOTE: Before using the Green Throttle Unity plugin, you need to install the Green Throttle SDK for Unity as described in Installing the SDK for Unity.

INTRODUCING THE GREEN THROTTLE UNITY PLUGIN

The Green Throttle SDK offers a plugin for the Unity Game Engine that automates the binding of controllers to your games and greatly simplifies the process of monitoring for controller status updates.

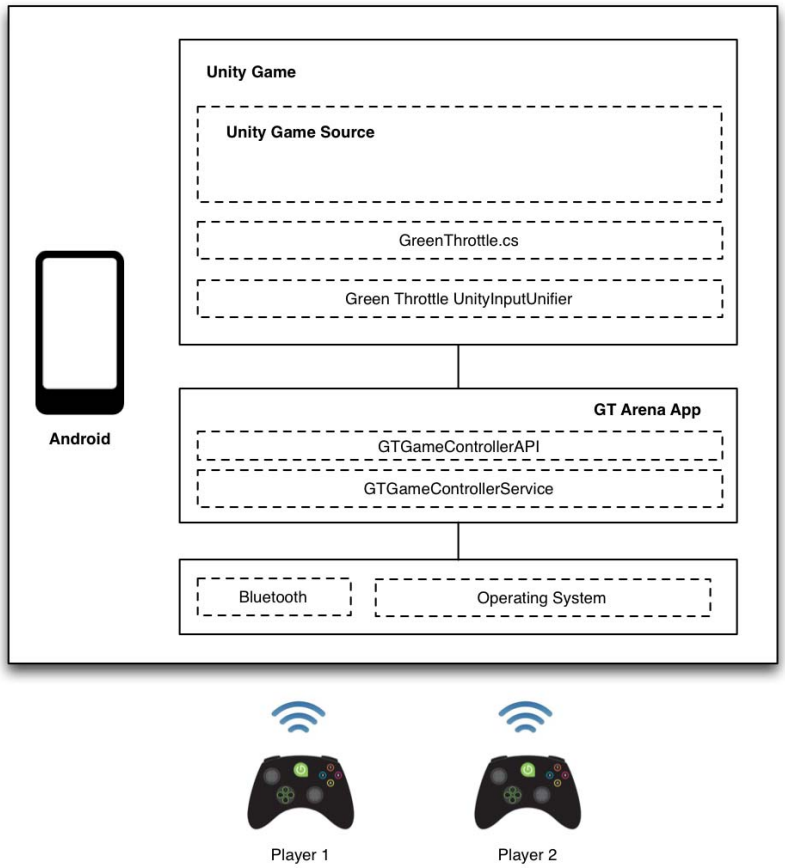
The Green Throttle Unity plugin consists of the following files that are added to your project when you extract the corresponding zip file:

- `Assets/Plugins/Android/AndroidManifest.xml`
- `Assets/Plugins/Android/UnityInputUnifier.jar`
- `Assets/Plugins/GreenThrottle/GreenThrottle.cs`

Similar to a purely Java-based environment, the Green Throttle Controller Service resides at the lowest level and manages all interactions with connected Atlas controllers. The `UnityInputUnifer`, in conjunction with the `AndroidManifest.xml` file, automatically starts the Unity system and binds your Unity game to the Green Throttle Controller Service. The `UnityInputUnifer` then sends messages to the `GreenThrottle` component (implemented as a singleton in the `GreenThrottle.cs` script) to store the current controller state. The `GreenThrottle` component provides a single point of access to all Green Throttle input.

Within your code, you need to call `GreenThrottle.Instance` in an `Awake` function early in the first scene of your game to ensure that you start capturing all controller state updates. After that initial call, subsequent calls to `GreenThrottle.Instance` allow you to access the input state of the controllers connected to the service through a set of controller state structures.

The following provides an overview of the Green Throttle Unity plugin architecture:



Green Throttle Unity Plugin Architecture

USING THE GREEN THROTTLE UNITY PLUGIN

This section describes how to use the Green Throttle Unity plugin to monitor controller status updates within your Unity-based game.

To use the Green Throttle Unity plugin

1. Call `GreenThrottle.Instance` in an `Awake` function early in the first scene of your game.

   This ensures that your game gets access to all controller state updates.

2. Poll the Green Throttle controller state structures in every frame of your game.

   `GreenThrottle.Instance` defines the following constants:

   - `BUTTON_<ID>` — Button indices defining each button type on the controller, for example, `BUTTON_A`, `BUTTON_B`, `BUTTON_Y`, `BUTTON_X`, `BUTTON_LEFT`, `BUTTON_RIGHT`, and so on.
   - `BUTTON_STATE_<ID>` — Button states, for example, `BUTTON_STATE_HELD` (button held but not released), `BUTTON_STATE_DOWN` (button not held, but was pressed this frame), and `BUTTON_STATE_UP` (button pressed and released this frame).
   - `ANALOG_<AXIS>` — Analog controls on the controller. These include `ANALOG_LEFT` (left analog stick dual axis control), `ANALOG_RIGHT` (right analog stick dual axis control), `ANALOG_L2` (left analog trigger single axis control), and `ANALOG_R2` (right analog trigger single axis control). Analog values can from -127 to 127 for dual axis controls, or 0 to 255 for single axis controls.
   - `CONTROLLER_<ID>` — Indices for controllers within the state structures, for example, `CONTROLLER_1`, `CONTROLLER_2`, `CONTROLLER_3`, and `CONTROLLER_4`.

   `GreenThrottle.Instance` defines the following state structures:

   - `bool buttonState[,,]` — State data for each button. The three-dimensional array uses the following indices: `[BUTTON_STATE_<ID>, CONTROLLER_<ID>, BUTTON_<ID>]`.
   - `float analogState[,,]` — State data for each analog control. The three-dimensional array uses the following indices: `[ANALOG_<DATA>, CONTROLLER_<ID>, ANALOG_<AXIS>]`.
   - `bool analogChanged[,]` — Specifies whether the analog control received an event in this frame. The two-dimensional array uses the following indices: `[CONTROLLER_ID, ANALOG_<AXIS>]`.
   - `bool connectedState[]` — Specifies whether the controller is connected or disconnected. The array uses the following index: `[CONTROLLER_ID]`.

   Examples:

   To determine whether `BUTTON_A` is being pressed on Controller 1:

```
if (GreenThrottle.Instance.buttonState[GreenThrottle.BUTTON_STATE_HELD,
    GreenThrottle.CONTROLLER_1, GreenThrottle.BUTTON_A]) {
   // do something
}
```

   To determine if `BUTTON_X` was released in this frame on Controller 2:

```
if (GreenThrottle.Instance.buttonState[GreenThrottle.BUTTON_STATE_UP,
    GreenThrottle.CONTROLLER_2, GreenThrottle.BUTTON_X]) {
   // do something
}
```

To retrieve the analog state, if the state has been updated for ANALOG_LEFT on Controller 1:

```
if (GreenThrottle.Instance.analogChanged[GreenThrottle.CONTROLLER_1,
    GreenThrottle.ANALOG_LEFT]) {
   float x = GreenThrottle.Instance.analogState[GreenThrottle.ANALOG_X,
      GreenThrottle.CONTROLLER_1, GreenThrottle.ANALOG_LEFT];
   float y = GreenThrottle.Instance.analogState[GreenThrottle.ANALOG_Y,
      GreenThrottle.CONTROLLER_1, GreenThrottle.ANALOG_LEFT];

   // do something with x and y
}
```

To determine whether Controller 3 or 4 have connected to the service:

```
if (GreenThrottle.Instance.connectedState[GreenThrottle.CONTROLLER_3]) {
   // do something
}

if (GreenThrottle.Instance.connectedState[GreenThrottle.CONTROLLER_4]) {
   // do something
}
```

3. (Optional) Check whether the Green Throttle Controller Service is active and sending data to your game.

   Check the `public bool isServiceActive` variable in `GreenThrottle.Instance`. You do not need to poll `GreenThrottle.Instance` in cases when the service is not sending data.

4. (Optional) Connect to the `ControllerConnectedStateChange` event in the opening scene of your game to receive events related to controllers connecting and disconnecting.

   Add the following code to your game:

   ```
   GreenThrottle.ControllerConnectedStateChange +=
       this.FunctionCallback;
   ```

   The signature of the `ControllerConnectedStateChange` event is as follows:

   ```
   public delegate void ControllerConnectedHandler(int controller,
                                                   bool isConnected);
   ```

   where `controller` indicates the controller for the event (between 1 and 4) and `isConnected` indicates whether the controller is connected or not.

   The signature of the `FunctionCallback` method is as follows:

   ```
   void FunctionCallback(int controller, bool isConnected)
   ```

   To remove the callback method (when the component is destroyed, for instance), add the following code to the `onDestroy()` method::

   ```
   GreenThrottle.ControllerConnectedStateChange -=
       this.FunctionCallback;
   ```

5. (Optional) Clear the state of a specific controller.

   You should do this after disconnecting from a game controller, clearing all the down button states and restoring the controller to a neutral state. Use the following method:

   ```
   public void ClearControllerState(int controller)
   ```

6. Build the Unity application package (.apk) and install the package on the Android device.

7. Pair one or more game controllers with the Android device.

   Start the game while the controllers are searching for a connection (when all LEDs are flashing). The `UnityInputUnifier` plugin binds to any searching controllers, assigning player numbers in the order that the controllers are bound.

Chapter 6

Integrating with Marmalade-Based Games

The Green Throttle SDK is compatible with the Marmalade Game Engine, making it easy to add support for advanced analog controllers to your Marmalade-based games on Android phones and tablets.

This chapter introduces the Green Throttle Marmalade extension and describes how to use the extension to monitor controller status updates within your Marmalade-based game.

NOTE: Before using the Green Throttle Marmalade extension, you need to install the Green Throttle SDK for Marmalade as described in Installing the SDK for Marmalade.

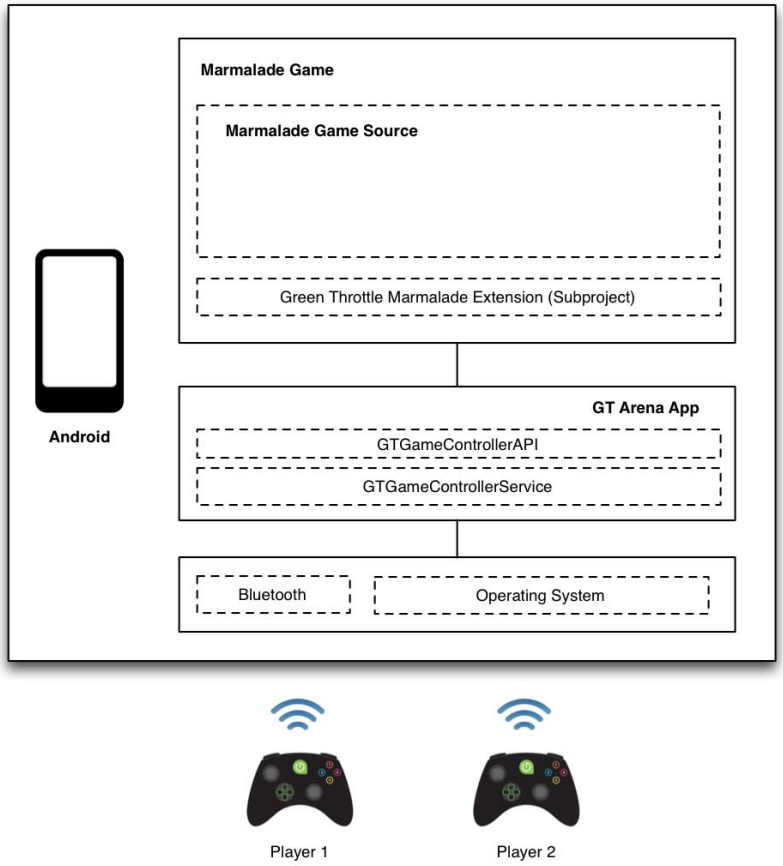
INTRODUCING THE GREEN THROTTLE MARMALADE EXTENSION

The Green Throttle SDK offers an extension for the Marmalade Game Engine that automates the binding of controllers to your games and greatly simplifies the process of monitoring for controller status updates.

Similar to a purely Java-based environment, the Green Throttle Controller Service resides at the lowest level and manages all interactions with connected Atlas controllers. Within your code, you need to add the GreenThrottle extension as a subproject in your game project (MKB file) and call the `StartFrame()` method at the beginning of the game loop (which saves every button event to the state buffers).

Following this initial setup, you can use various methods on the GreenThrottle extension to determine the state of the service and controller, poll the controller button state (in every frame of your game), and respond to analog events, among other operations.

The following provides an overview of the Green Throttle Marmalade extension architecture:



Green Throttle Marmalade Extension Architecture

USING THE GREEN THROTTLE MARMALADE EXTENSION

This section describes how to use the Green Throttle Marmalade extension to monitor controller status updates within your Marmalade-based game.

To use the Green Throttle Marmalade extension

1. Add GreenThrottle as a subproject to your project.

   Edit the MKB file for the project, and add the following lines:

   ```
   subproject
   {
      GreenThrottle
   }
   ```

   Marmalade downloads the GreenThrottle project if the project is in the repository. Otherwise, ensure that the GreenThrottle project is in the `MARMALADE_HOME/extensions` directory.

   This ensures that your game gets access to all controller state updates.

2. Include the `GreenThrottle.h` file in your C++ source files.

   The `GreenThrottle.h` file defines all constants used with the Green Throttle Controller API.

3. Call the `StartFrame()` function inside, and at the beginning, of the game loop.

   The `StartFrame()` function saves every button event to the state buffers (since the last call) and manages the state based on `BUTTON_STATE_DOWN`, `BUTTON_STATE_UP` and `GetAnalogChanged()` events.

   NOTE: The game will not see any button state changes unless the `StartFrame()` function is called.

4. (Optional) Determine the state of the service and controllers.

   Use the following function to determine if the Green Throttle Controller Service has sent a connect message to the Marmalade extension:

   ```
   bool GetConnectedState(int controllerId)
   ```

   where `controllerId` is the identifier of the controller to query, as defined in the `GreenThrottle.h` file.

   The function returns `true` if the service has sent a connect message to the Marmalade extension; otherwise the function returns `false` if no connect message has been received or if a disconnect message has been received.

   Use the following function to determine if the Green Throttle Controller Service is connected:

   ```
   bool GetServiceConnected()
   ```

   The function returns `true` if the service is connected; `false` otherwise. The `GTMarmaladeActivity.onCreate()` function attempts to connect to the service automatically. You can, however, use this function if your project requires a separate activity.

   Use the following function to clear the state of the specified controller:

   ```
   void ClearControllerState(int controllerId)
   ```

   where `controllerId` is the identifier of the controller to query, as defined in the `GreenThrottle.h` file. You can use this function if the controller disconnects, thereby preventing the controller service from reporting events that might never be updated.

5. Poll the Green Throttle button state (on any of the controllers) in every frame of your game.

   Use the following function:

   ```
   bool GetButtonState(int buttonState, int controllerId, int button)
   ```

where

- `buttonState` indicates the button state, from among the following: `BUTTON_STATE_HELD` (button held down), `BUTTON_STATE_DOWN` (button pressed down this frame), or `BUTTON_STATE_UP` (button released this frame).
- `controllerId` is the identifier of the controller to query, as defined in the `GreenThrottle.h` file.
- `button` is the controller button, as defined in the `GreenThrottle.h` file.

6. Respond to analog controller events.

   Use the following function to determine whether there have been any analog control events for the frame:

   ```
   bool GetAnalogChanged(int controllerId, int analogControl)
   ```

   where

   - `controllerId` is the identifier of the controller to query, as defined in the `GreenThrottle.h` file.
   - `analogControl` is the identifier of the analog control, as defined in the `GreenThrottle.h` file.

   Use the following function to determine the state of the various analog controls on the controller:

   ```
   float GetAnalogState(int axis, int controllerId, int analogControl)
   ```

   where

   - `axis` is the X or Y axis constant, as defined the `GreenThrottle.h` file. Single axis controls have only an X value; in this case the Y value is always 0.
   - `controllerId` is the identifier of the controller to query, as defined in the `GreenThrottle.h` file.
   - `analogControl` is the identifier of the analog control, as defined in the `GreenThrottle.h` file.

7. (Optional) Remap the left or right analog sticks to the D-pad or AYXB buttons respectively, as required.

   Use the following functions:

   - `void AddLeftButtonRemap()` — Remaps the left analog stick to the D-pad buttons resulting in a D-pad pressed event from the left analog stick. The `GTMarmaladeActivity.onCreate()` function calls this function by default.
   - `void RemoveLeftButtonRemap()` — Removes the remapping between the left analog stick and the D-pad.

- `void AddRightButtonRemap()` — Remaps the right analog stick to the AYXB buttons (A is down, X is left, Y is up, and B is right). By default, this remapping is disabled.
- `void RemoveRightButtonRemap()` — Removes the remapping between the right analog stick and the AYXB buttons.

8. (Optional) Call the `HideAndroidSystemUI()` function to hide the on-screen Android system user interface.

   On-screen touch events typically display the Android system user interface. You can use the `HideAndroidSystemUI()` function to hide this interface, as required. By default, the `GTMarmaladeActivity.onCreate()` and `GTMarmaladeActivity.onResume()` functions call the `HideAndroidSystemUI()` function automatically.

9. Build the Marmalade project and deploy the package to the Android device.

10. Pair one or more game controllers with the Android device.

    Start the game while the controllers are searching for a connection (when all LEDs are flashing). The Green Throttle Marmalade extension binds to any searching controllers, assigning player numbers in the order that the controllers are bound.

Chapter 7

Integrating with Corona Enterprise-Based Games

The Green Throttle SDK is compatible with Corona Enterprise, making it easy to add support for advanced analog controllers to your Corona Enterprise-based games on Android phones and tablets.

This chapter introduces the Green Throttle Corona Enterprise plugin and describes how to use the plugin to monitor controller status updates within your Corona Enterprise-based game.

NOTE: Before using the Green Throttle Corona Enterprise plugin, you need to install the Green Throttle SDK for the Corona Enterprise as described in Installing the SDK for Corona Enterprise.

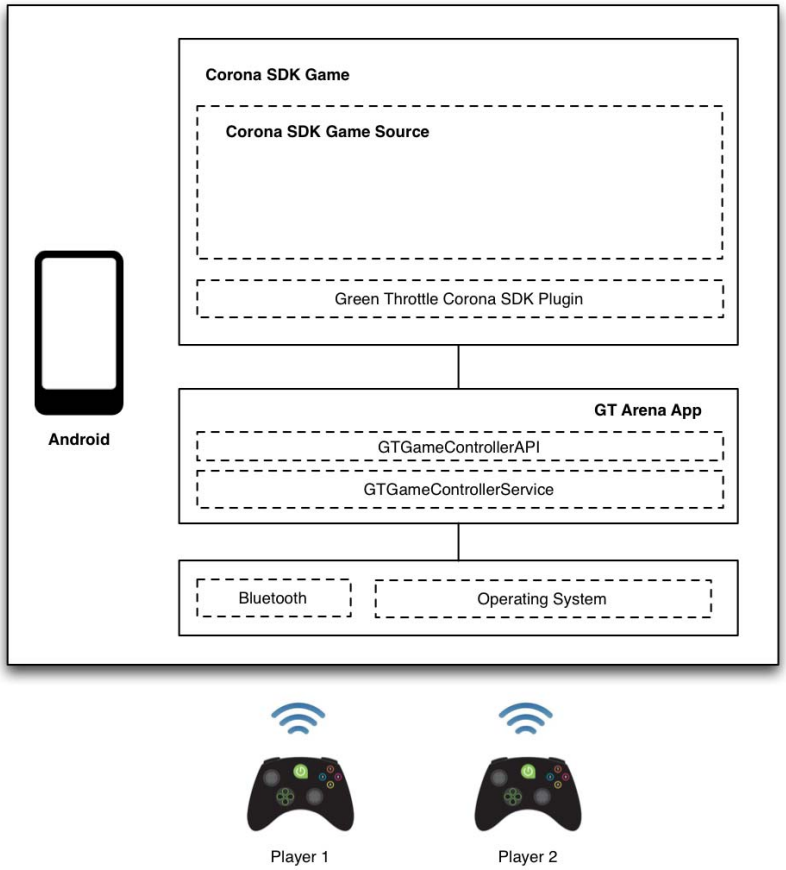
INTRODUCING THE GREEN THROTTLE CORONA ENTERPRISE PLUGIN

The Green Throttle SDK offers a plugin for Corona Enterprise that automates the binding of controllers to your games and greatly simplifies the process of monitoring for controller status updates.

Similar to a purely Java-based environment, the Green Throttle Controller Service resides at the lowest level and manages all interactions with connected Atlas controllers. Within your code, you need to add the GreenThrottle plugin to your project and call the `startFrame()` method at the beginning of the game loop (which saves every button event to the state buffers).

Following this initial setup, you can use various methods on the GreenThrottle extension to determine the state of the service and controller, poll the controller button state (in every frame of your game), and respond to analog events, among other operations.

The following provides an overview of the Green Throttle Corona Enterprise plugin architecture:



Green Throttle Corona Enterprise Plugin Architecture

EXPLORING GREEN THROTTLE CONSTANTS

Green Throttle defines a set of constants that are available for use with the Green Throttle SDK. These constants include the following:

- Controller numbers
- Buttons
- Button states
- Analog control
- Analog axis

Exploring the Controller Number Constants

Green Throttle defines the following constants, used with several functions, specifying the (potentially) available controllers:

- `CONTROLLER_1`
- `CONTROLLER_2`
- `CONTROLLER_3`
- `CONTROLLER_4`

Exploring the Button Constants

Green Throttle defines the following button constants used with the `getButtonState()` function:

- `BUTTON_A`
- `BUTTON_B`
- `BUTTON_Y`
- `BUTTON_X`
- `BUTTON_DOWN`
- `BUTTON_UP`
- `BUTTON_LEFT`
- `BUTTON_RIGHT`
- `BUTTON_L1`
- `BUTTON_L2`
- `BUTTON_L3`
- `BUTTON_R1`
- `BUTTON_R2`
- `BUTTON_R3`
- `BUTTON_ANALOG_LEFT_UP`
- `BUTTON_ANALOG_LEFT_DOWN`
- `BUTTON_ANALOG_LEFT_LEFT`
- `BUTTON_ANALOG_LEFT_RIGHT`
- `BUTTON_ANALOG_RIGHT_UP`
- `BUTTON_ANALOG_RIGHT_DOWN`
- `BUTTON_ANALOG_RIGHT_LEFT`
- `BUTTON_ANALOG_RIGHT_RIGHT`
- `BUTTON_BACK`
- `BUTTON_START`
- `BUTTON_GTHOME`

Exploring the Button State Constants

Green Throttle defines the following button state constants:

- `BUTTON_STATE_HELD` — The button is held down
- `BUTTON_STATE_DOWN` — The button was pressed down in this frame
- `BUTTON_STATE_UP` — The button was released in this frame

Exploring the Analog Control Constants

Green Throttle defines the following analog control constants used with the `getAnalogState()` and `getAnalogChanged()` functions:

- `ANALOG_LEFT` — Left analog stick (X and Y axis data: -127 to 127, 0 centered)
- `ANALOG_RIGHT` — Right analog stick (X and Y axis data: -127 to 127, 0 centered)
- `ANALOG_L2` — Left trigger (X axis data: 0 to 255, fully engaged)
- `ANALOG_R2` — Right trigger (X axis data: 0 to 255, fully engaged)

Exploring the Analog Axis Constants

Green Throttle defines the following analog axis constants used with the `getAnalogState()` function:

- `ANALOG_AXIS_X` — X data for dual axis controls (this is the only value for single axis control)
- `ANALOG_AXIS_Y` — Y data for dual axis controls (not used for single axis control)

Preparing to Use the Green Throttle Corona Enterprise Plugin

You need to complete the procedure described in this section before using the Green Throttle Corona Enterprise plugin.

To prepare to use the Corona Enterprise plugin

1. Edit the `project.properties` file in the `android` directory in your Corona Enterprise project.

   Add the following line to the `project.properties` file:

   ```
   android.library.reference.<number>=${CoronaEnterpriseDir}/Plugins/
   GreenThrottle/android
   ```

   Replace `<number>` with the next number in the sequence in your file. In many cases, this will be the number 2.

2. Edit the `build.sh` file in the `android` directory in your Corona Enterprise project.

   Add the following line to the appropriate section of the `build.sh` file:

   ```
   "$SDK_PATH/tools/android" update lib-project -p
   "$CORONA_PATH/Plugins/GreenThrottle/android"
   checkError
   ```

USING THE GREEN THROTTLE CORONA ENTERPRISE PLUGIN

This section describes how to use the Green Throttle Corona Enterprise plugin to monitor controller status updates within your Corona Enterprise-based game.

To use the Green Throttle Corona Enterprise plugin

1. Add the GreenThrottle plugin to your project.

   Edit your project file, and add the following line near the top of the project:

   ```
   local greenthrottle = require("plugin.greenthrottle")
   ```

   This ensures that your game gets access to all controller state updates.

2. Call the `startFrame()` function inside, and at the beginning, of the game loop.

   The `startFrame()` function saves every button event to the state buffers (since the last call) and manages the state based on `BUTTON_STATE_DOWN`, `BUTTON_STATE_UP` and `getAnalogChanged()` events.

   NOTE: The game will not see any button state changes unless the `startFrame()` function is called.

3. (Optional) Determine the state of the service and controllers.

   Use the following function to determine if the Green Throttle Controller Service has sent a connect message to the Corona Enterprise plugin:

   ```
   getConnectedState(controllerId)
   ```

   where `controllerId` is the identifier (number) of the controller to query.

   The function returns `true` if the service has sent a connect message to the Corona Enterprise plugin; otherwise the function returns `false` if no connect message has been received or if a disconnect message has been received.

   Use the following function to determine if the Green Throttle Controller Service is connected:

   ```
   getServiceConnected()
   ```

   The function returns `true` if the service is connected; `false` if the service has failed or is not installed. The Corona Enterprise startup attempts to connect to the service automatically.

   Use the following function to clear the state of the specified controller:

   ```
   clearControllerState(controllerId)
   ```

   where `controllerId` is the identifier (number) of the controller to query. You can use this function if the controller disconnects, thereby preventing the controller service from reporting events that might never be updated.

4. Poll the Green Throttle button state (on any of the controllers) in every frame of your game.

   Use the following function:

   ```
   getButtonState(buttonState, controllerId, button)
   ```

   where

   - `buttonState` indicates the button state, from among the following: `BUTTON_STATE_HELD` (button held down), `BUTTON_STATE_DOWN` (button pressed down this frame), or `BUTTON_STATE_UP` (button released this frame).
   - `controllerId` is the identifier (number) of the controller to query.
   - `button` is the controller button (number).

5. Respond to analog controller events.

   Use the following function to determine whether there have been any analog control events for the frame:

   ```
   getAnalogChanged(controllerId, analogControl)
   ```

   where

   - `controllerId` is the identifier (number) of the controller to query
   - `analogControl` is the identifier (number) of the analog control

   The function returns `true` if there have been analog control events for the frame.

   Use the following function to determine the state of the various analog controls on the controller:

   ```
   getAnalogState(axis, controllerId, analogControl)
   ```

   where

   - `axis` is the X or Y axis constant (number). Single axis controls have only an X value; in this case the Y value always returns 0.
   - `controllerId` is the identifier (number) of the controller to query.
   - `analogControl` is the identifier (number) of the analog control.

   The function returns a number between -127.0 and 127.0 or 0.0 and 255.0 (based on the axis).

6. (Optional) Remap the left or right analog sticks to the D-pad or AYXB button respectively, as required.

   Use the following functions:

   - `addLeftButtonRemap()` — Remaps the left analog stick to the D-pad buttons resulting in a D-pad pressed event from the left analog stick. This function is called on startup by default.
   - `removeLeftButtonRemap()` — Removes the remapping between the left analog stick and the D-pad.
   - `addRightButtonRemap()` — Remaps the right analog stick to the AYXB buttons (A is down, X is left, Y is up, and B is right). By default, this remapping is disabled.
   - `removeRightButtonRemap()` — Removes the remapping between the right analog stick and the AYXB buttons.

7. (Optional) Call the `hideAndroidSystemUI()` function to hide the on-screen Android system user interface.

   On-screen touch events typically display the Android system user interface. You can use the `hideAndroidSystemUI()` function to hide this interface, as required. By default, the `startup()` and `onResume()` functions call the `hideAndroidSystemUI()` function automatically.

8. Build the Corona Enterprise project and deploy the package on the Android device.

9. Pair one or more game controllers with the Android device.

   Start the game while the controllers are searching for a connection (when all LEDs are flashing). The Green Throttle Corona Enterprise plugin binds to any searching controllers, assigning player numbers in the order that the controllers are bound.

Chapter 8

Integrating with Java Apps
Using the Green Throttle Java API

The Green Throttle SDK includes an advanced Java API offering an extra level of control when adding analog controller support to native Java games on Android phones and tablets.

In most cases, it is easier to use the Green Throttle Java Input Unifier, as described in Using the Green Throttle Java Input Unifier, to integrate with your Java-based games. However, Green Throttle provides the Java API to allow developers to create custom input implementations if special circumstances require it.

This chapter introduces the Green Throttle Java API and describes how to use the API to monitor controller status updates within your Java-based game.

NOTE: Before using the Green Throttle Java API, you need to install the Green Throttle SDK for Java as described in Installing the SDK for Java Development.

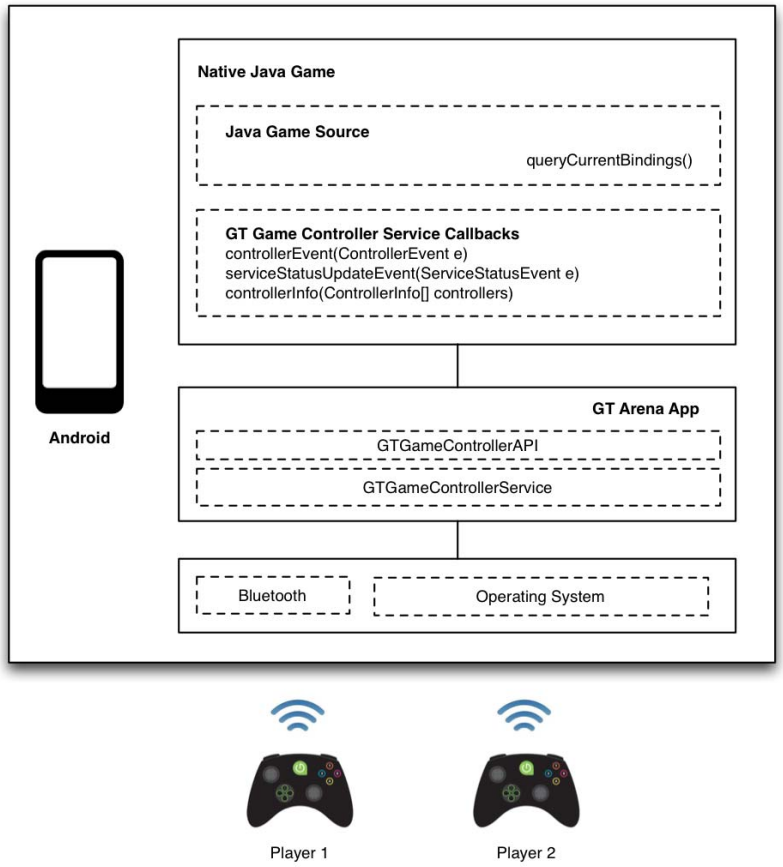
INTRODUCING THE GREEN THROTTLE SDK FOR JAVA

The Green Throttle environment offers a simple yet comprehensive system that allows your games to bind to Green Throttle Atlas controllers and monitor controller status updates. At the lowest layer, the Green Throttle Controller Service manages all interactions between the your game and the Bluetooth-enabled Atlas controllers.

When integrating your Java-based game, you begin by binding to the Green Throttle Controller Service and querying the service to request information about the game controllers currently connected. You can also request that the service automatically connect game controllers (previously discovered and bound) as they are powered on.

The core of your interactions with the service then involves a set of callbacks that provide access to controller events, including button presses and stick movements, and controller status events, such as when a controller connects or disconnects from the system.

The following provides an overview of the Green Throttle SDK for Java:



Green Throttle Architecture for Java

BEFORE USING THE GREEN THROTTLE SDK

This chapter assumes that you are familiar with the following Android concepts and systems:

- How Android service binding and unbinding works
- How to use background tasks, handlers, and the runnable interface

ACCESSING THE API REFERENCE DOCUMENTATION

The Green Throttle API documentation is available as Javadoc.

To access the API reference documentation

1. Expand the `gt-gamecontroller-doc<version>.zip` file.
2. Using a Web browser, open the `index.html` file in the `doc` folder.

CONFIGURING YOUR PROJECT

When developing your application, you need to configure your project to add a reference to the Green Throttle API.

To configure your project

1. Using your IDE, import the `GTGameControllerApi` project.

   Using Eclipse, for example, do the following:

   a. Choose **File > New > Project** from the main menu The *New Project* wizard launches.

   b. Select **Android > Android Project from Existing Code** in the *Wizards* list, and click **Next**. The *Import Projects* dialog appears.

   c. Click **Browse** and select the root directory of the `GTGameControllerApi` project.

   d. Click **Finish**.

2. In the Package Explorer, select your project and choose **Project > Properties** from the main menu. The *Properties* dialog appears.

3. Select *Android*, and click **Add** in the *Library* section. The *Project Selection* dialog appears.

4. Select `GTGameControllerApi` in the list and click **OK**.

5. Click **OK**.

EXPLORING THE GREEN THROTTLE API

The Green Throttle API defines a set of objects that enables you to monitor controller status updates using a high-level interface.

EXPLORING THE SERVICE CALLBACKS

The Green Throttle API offers the following callbacks that the service uses to send events to your client application:

- `controllerEvent()`
- `serviceStatusUpdateEvent()`
- `controllerInfo()`

controllerEvent

The `controllerEvent()` method is the main callback from the service. The service sends one call per event generated by the game controller.

```
void controllerEvent(in ControllerEvent e);
```

serviceStatusUpdateEvent

The `serviceStatusUpdateEvent()` method is called from the service for events that report on the state of the game controllers and the service itself. The principal events sent are:

- `QUERY_BOUND_CONTROLLERS_VALUE` — Indicates that a game controller has connected, containing a string with the following format:

  `bluetooth_address=player_<n>`

  where <n> is the number of the player. The index starts at 1.

- `CONTROLLER_DISCONNECTED` — Indicates that a game controller has disconnected, containing a string with the Bluetooth address of the corresponding controller

```
void serviceStatusUpdateEvent(in ServiceStatusEvent e);
```

controllerInfo

The `controllerInfo()` method is called with information about connected controllers after your application calls the `queryCurrentBindings()` method.

```
void controllerInfo(in ControllerInfo[] ia);
```

Exploring the ControllerEvent Class

The `ControllerEvent` class defines the input codes and data structure that comprise an event from the Green Throttle service, encapsulating the payload of each event as received from the game controller.

Exploring the Controller Button and Joystick Codes

The `ControllerEvent` class includes the `CommonCodes` enum, which defines all input codes available from the game controller (along with some extra codes).

```
enum ControllerEvent.CommonCodes
```

Examples of common codes include `A_BUTTON`, `B_BUTTON`, `X_BUTTON`, `Y_BUTTON`, `BACK_BUTTON`, `R1_BUTTON`, `L1_BUTTON`, `DPAD_RIGHT`, `DPAD_LEFT`, `RIGHT_ANALOG_AS_DPAD_RIGHT`, and `RIGHT_ANALOG_AS_DPAD_LEFT`, among others.

Exploring the Controller Actions

The `ControllerEvent` class also includes the `Action` enum, which defines the action for each input.

`enum ControllerEvent.Action`

Examples of game controller actions include `ACTION_UP`, `ACTION_DOWN`, and `UPDATE` (for analog events).

Exploring the Methods

The `ControllerEvent` class defines the following methods (among others):

- `controllerEvent.id()`— Returns the Bluetooth address of the controller that generated the event
- `controllerEvent.action()`— Returns the type of action (as an `Action` object) that the user performed (such as `ACTION_DOWN` for a button down event, `ACTION_UP` for a button up event, and `UPDATE` for analog stick/trigger movement)
- `controllerEvent.cCode()`—Returns a value of type `CommonCodes` identifying the button, stick, or trigger that generated the event
- `controllerEvent.analogData()` — Returns a hashmap of `AnalogDataType` to `Double` for the axes that generated the event

Exploring the ServiceStatusEvent Class

The `ServiceStatusEvent` class defines callback methods used to send service status information (that is, status not specifically related to game controller input).

Exploring the Service Status Codes

The `ServiceStatusEvent` class includes the `StatusCode` enum, which defines the status codes used by the Green Throttle Game Controller Service.

`enum ServiceStatusEvent.StatusCode`

Examples of service status event values include `QUERY_BOUND_CONTROLLERS_VALUE` and `CONTROLLER_DISCONNECTED`, among others.

Exploring the Methods

The `ServiceStatusEvent` class defines the following methods (among others):

- `StatusCode code()` — Returns the status code, such as `QUERY_BOUND_CONTROLLERS_VALUE` and `CONTROLLER_DISCONNECTED`, among others
- `String encodedData()` — Returns the payload, specific to the type of event

USING THE GREEN THROTTLE JAVA API

The Green Throttle Game Controller API enables your application to retrieve input and other status information from game controllers connected to your application. This section describes how to use the Green Throttle Game Controller API.

NOTE: When testing your applications, you need to connect the game controllers using the *GT Controller Test* app prior to launching your game.

To use the Green Throttle API

1. Bind the application to the Green Throttle Service.

   Use a call similar to the following:

   ```
   bindService(new Intent(IGTController.class.getName()), mHandler,
               Context.BIND_AUTO_CREATE);
   ```

   where

   `mHandler` is an instance of an object that extends `IGTControllerCallback.Stub`.

   The `mHandler` instance receives the following Android service-related calls:

   - `onServiceConnected()` — Called when the game is connected to the service
   - `onServiceDisconnected()` — Called when the game is disconnected from the service

   For more information about Android services, see http://developer.android.com/guide/components/services.html.

   In addition, the `mHandler` instance receives the following calls from `GTGameControllerService`:

   - `controllerEvent(ControllerEvent e)` — Called when the user generates an event (moves the joystick or presses a button) on the game controller
   - `serviceStatusUpdateEvent(ServiceStatusEvent e)` — Called when a game controller is connected or disconnected
   - `controllerInfo(ControllerInfo[] controllers)` — Called with information about connected controllers after the application calls the `queryCurrentBindings()` method

2. Call the `queryCurrentBindings()` method to request information about the game controllers currently connected.

   `GTGameControllerService` calls the `controllerInfo(ControllerInfo[] controllers)` callback method in the application with information about the game controllers (passed as an array of `ControllerInfo` objects).

3. Call the `startAutoconnect()` method to request that the `GTGameControllerService` automatically connect game controllers (that have been previously discovered and bound) as they are powered on.

4. Respond to events when controllers are connected or disconnected.

   The `serviceStatusUpdateEvent(in ServiceStatusEvent e)` method is called from the service for events that report on the state of the game controllers. The principal events sent are:

   - `QUERY_BOUND_CONTROLLERS_VALUE` — Indicates that a game controller has connected, containing a string with the following format:

     `bluetooth_address=player_<n>`

     where <n> is the number of the player. The index starts at 1.

   - `CONTROLLER_DISCONNECTED` — Indicates that a game controller has disconnected, containing a string with the Bluetooth address of the corresponding controller

5. Respond to controller input events.

   The `controllerEvent()` method is the main callback from the service. The service sends one call per event generated by the game controller.

   ```
   void controllerEvent(in ControllerEvent e);
   ```

   You can use the following methods to retrieve details about the event:

   - `controllerEvent.id()`— Returns the Bluetooth address of the controller that generated the event
   - `controllerEvent.action()`— Returns the type of action (as an `Action` object) that the user performed (such as `ACTION_DOWN` for a button down event, `ACTION_UP` for a button up event, and `UPDATE` for analog stick/trigger movement)
   - `controllerEvent.cCode()`— Returns a value of type `CommonCodes` identifying the button, stick, or trigger that generated the event

6. Respond to analog input events.

   Analog input differs from regular button input in that the action (`Action` object) is always `UPDATE` with the payload containing X and Y analog information. You can use the following method to retrieve details about analog events:

   - `controllerEvent.analogData()` — Returns a hashmap of `AnalogDataType` to `Double` for the axes that generated the event

   The `AnalogDataType` enum (on the `ControllerEvent` object) includes the following:

   - `X_DATA` — Used for all analog sticks and triggers. For triggers, the value is between 0.0 and 255.0. For analog sticks, the value is between -127 and 127.
   - `Y_DATA` — Used only for analog sticks, with a value between -127 to 127.

   NOTE: The axis polarity follows the Cartesian plane where positive values represent upper right and negative values represent lower left.

The `CommonCodes` enum includes the following:

- `LEFT_ANALOG` — Left dual analog stick
- `RIGHT_ANALOG` — Right dual analog stick
- `L2_ANALOG` — L2 analog trigger
- `R2_ANALOG` — R2 analog trigger

NOTE: Analog sticks also send events that translate to D-pad-style events, for example, `LEFT_ANALOG_AS_DPAD_UP` and `RIGHT_ANALOG_AS_DPAD_UP`.

7. Build the Java project and deploy the package to the Android device.
8. Pair one or more game controllers with the Android device.

   Start the game while the controllers are searching for a connection (when all LEDs are flashing). The Green Throttle Service binds to any searching controllers, assigning player numbers in the order that the controllers are bound.

Chapter 9

Integrating with Green Throttle Arena

Green Throttle Arena is an enhanced HDTV user experience that provides a central hub and game portal allowing users to find and interact with games developed by Green Throttle and other independent developers and publishers.

This chapter introduces Green Throttle Arena, describes the benefits of Arena, and explains how to have your game featured on the game portal.

Introducing Green Throttle Arena

Green Throttle Arena (*GT Arena*) is an Android-based app that enables users to do the following:

- Pair their Atlas controllers with their Android device
- Discover and interact with GT-compatible games
- Follow shortcuts to *Google Play* or the *Amazon Appstore for Android* (for Amazon Kindle Fire devices) to purchase and download games

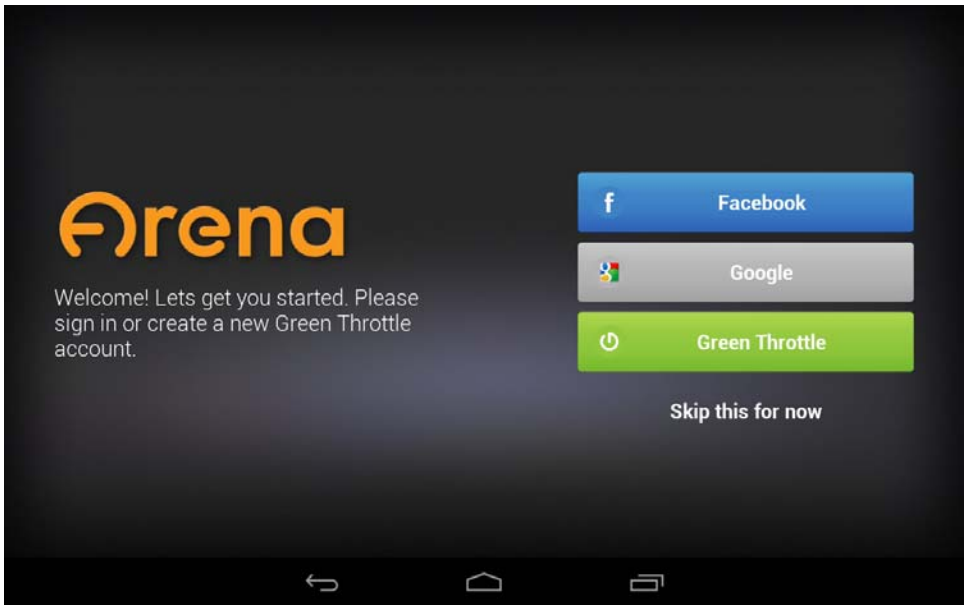
Exploring the Benefits of Green Throttle Arena

As a developer, there are several advantages and benefits of having your game in the Green Throttle Arena. These include:

- Green Throttle Arena is a fun and easy-to-use interactive space through which consumers can easily find their favorite games from the comfort of their couch using advanced analog controllers
- The system makes it easy to feature and highlight your game directly to consumers
- Green Throttle Arena is the go-to place for Green Throttle users, especially since all new controllers are paired using the *GT Arena* app.

EXPLORING THE GT ARENA APP

The first time users launch the *GT Arena* app, they are presented with the opportunity to sign in to Arena using their Facebook, Google, or Green Throttle credentials. The following shows the *GT Arena* sign in screen:



Green Throttle Arena: Sign In Screen

After signing in, users can pair their Green Throttle Atlas controllers to their Android device (and then connect their device to their TV). Following this one-time setup, users can then find and interact with games (or manage their controllers) using the following areas:

- Home — Displays recently-played games and the top five featured games
- My Games — Displays your Green Throttle-compatible games
- Recommended — Displays recommended Green Throttle-compatible games available on *Google Play* or on the *Amazon Appstore for Android*
- Settings — Shows Green Throttle Arena and Green Throttle Controller Service settings, allowing users to check for *GT Arena* app updates and manage paired controllers.

NOTE: The *GT Arena* app supports touch control and does not assume or depend on a controller being present. This allows users to use the *GT Arena* app to find, download, and play games in cases when they do not have a controller available.

Featuring Your Game in the Green Throttle Arena

This section describes the steps you need to complete to have your game featured in the Green Throttle Arena.

To feature a game in the Arena

1. Verify that the game matches the Green Throttle design standards.

   See the *Green Throttle Design Standards Guide* for more information.

2. Create a single APK file that supports both touch and controller-based user interfaces.

3. (Optional) Test the game in the Green Throttle Arena.

   See Testing Games in the Green Throttle Arena for more information.

4. Publish the game on either *Google Play* or on the *Amazon Appstore for Android*.

   For more information, refer to the publishing guidelines for the respective markets.

5. Submit the game to Green Throttle to have it appear in the Green Throttle Arena.

   Complete the following:

   a. Supply Green Throttle with the graphic assets for the game.

   b. Submit shortcuts to the game (in *Google Play* or the *Amazon Appstore for Android*) to Green Throttle.

   Note: Green Throttle Arena displays a high-resolution icon for your app only after you publish your app with Green Throttle. Arena uses the standard Android app icon for apps that are not published with Green Throttle.

Testing Games in the Green Throttle Arena

To have your games appear in the Arena, you need to submit your games to Green Throttle for review, following the Green Throttle publishing guidelines. You can, however, test your games in the Arena by naming your package using the following convention:

```
com.greenthrottle.<your_app_name>
```

For example, if your app is named *MyGreatGame*, you can use the following package name to have the game appear in the Green Throttle Arena:

```
com.greenthrottle.mygreatgame
```

Note: This method is only for testing. Green Throttle does not recommend using this method when you publish your games.

Appendix A

Using the GT Controller Test App

The Green Throttle SDK includes a simple but useful Android app called *GT Controller Test* (`GTControllerTestApp<version>.apk`) that enables you to scan and connect game controllers to the Green Throttle Controller Service, and then display events in real time as they are received from the connected controllers.

Specifically, the *GT Controller Test* app allows you to do the following:

- Scan and connect to available game controllers
- Search for new controllers
- Reset the service and clear the controller cache
- Display received controller events in real-time

NOTE: The *GT Arena* app needs to be installed on the Android device to use the *GT Controller Test* app.

USING THE GT CONTROLLER TEST APP

The *GT Controller Test* app features a basic interface that displays all actions and information on a single screen.

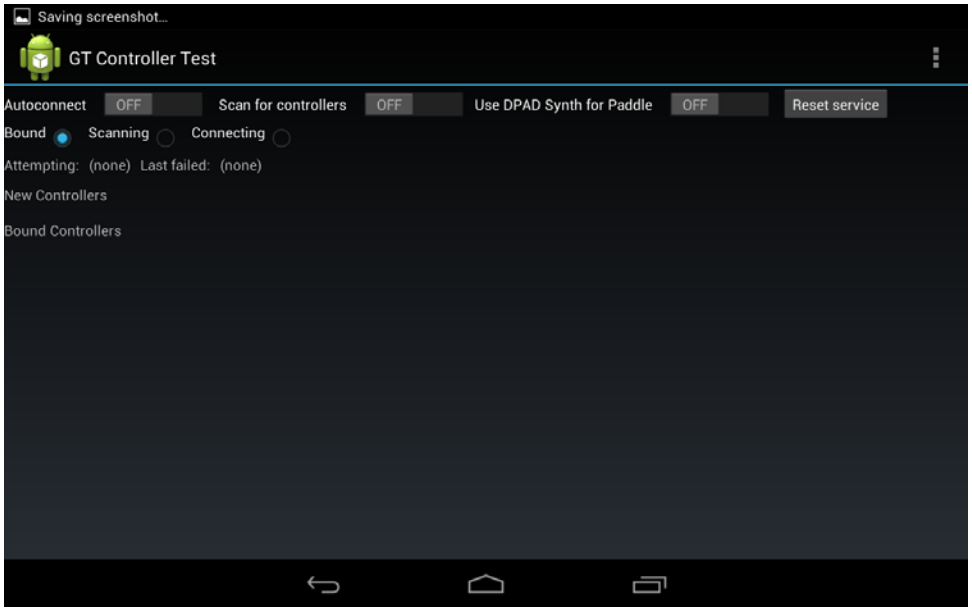
To use the GT Controller Test app

1. Install the `GTControllerTestApp<version>.apk` file on your Android device.

   The file is located in the `GreenThrottleSDK` folder. After installing, the application appears on the Apps screen.

2. Tap the `GT Controller Test` icon  on the Apps screen.

   The app opens displaying the following screen:

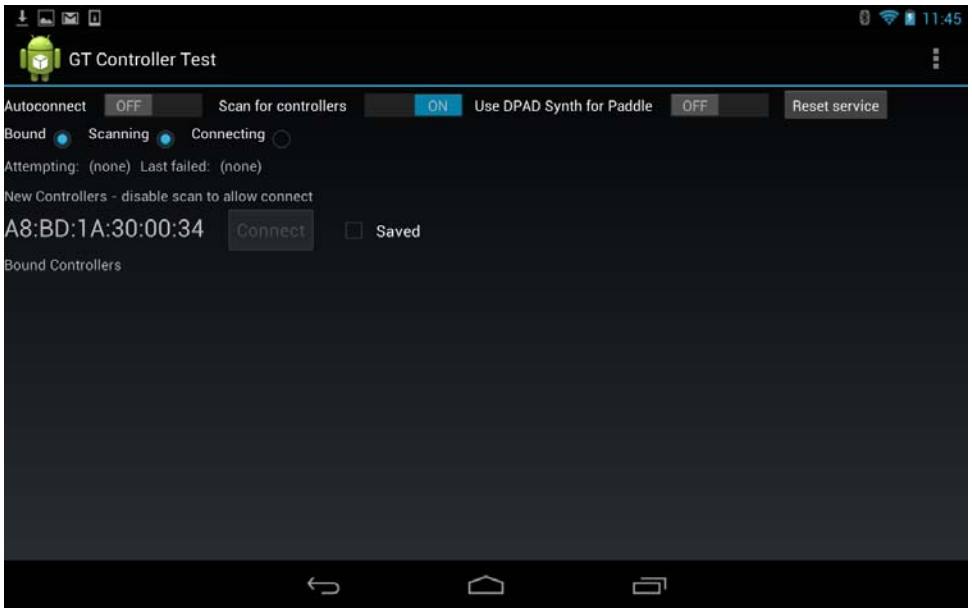


GT Controller Test: Main Screen

3. Tap **Scan for controllers** to turn the switch on. The app scans for available Green Throttle controllers.

4. Press and hold the Green Throttle button on the Atlas controller for several seconds. The LED buttons flash.

   The app displays the Bluetooth address of discovered controllers.



GT Controller Test: Controller Found

5. Tap **Scan for controllers** again to turn the switch off. The **Connect** button is enabled.

6. Tap **Connect** to connect the controller to the Green Throttle Controller Service.

   The Android system asks for confirmation to pair the controller using Bluetooth in cases when the devices have never been paired before. Tap **Pair** to complete the Bluetooth pairing.

7. Interact with the controller by pressing buttons and moving the sticks. The app displays the controller state in real time.

GT Controller Test: Real Time State Display

8. Tap **Reset service** to clear the application controller cache and return the app to the initial state.

Appendix B

Google Play and Green Throttle Arena Game Assets

Game assets—typically icons, thumbnails, banners, and videos, among others—are essential for highlighting and promoting your game to users.

This appendix describes the game assets that you need to supply to support your games in both the Google Play market and the Green Throttle Arena game portal.

NOTE: Contact Green Throttle at devsupport@greenthrottle.com for more information about how to have your apps promoted in the featured games section of the Green Throttle Arena.

SUPPLYING GOOGLE PLAY ASSETS

You need to supply the following game assets for Google Play:

- Game icon
- Featured graphic

GAME ICON

The game icon you supply for Google Play needs to match the following requirements:

- 512 x 512 pixels
- Include the Green Throttle badge (optional)

The following illustrates a variety of game icons with several styles of Green Throttle badges (not actual size):



Featured Graphic

The featured graphic you supply for Google Play needs to match the following requirements:

- 1024 x 500 pixels
- Embedded player count icon (optional)

The following shows a couple of featured graphics with two styles of Green Throttle badges (with embedded player counts, but not actual size):

Supplying Green Throttle Arena Assets

You need to supply the following game assets for the Green Throttle Arena:

- Game icon
- Game thumbnail
- Game banner

You can also optionally supply the following game assets for the Arena:

- Video trailer
- Recommended tab banner

Game Icon

The game icon you supply for Green Throttle Arena needs to match the following requirements:

- 512 x 512 pixels
- Include the Green Throttle badge (optional)

The following illustrates a variety of game icons with several styles of Green Throttle badges (not actual size):



Game Thumbnail

The game thumbnail you supply for Green Throttle Arena needs to match the following requirement:

- 436 x 206 pixels

The following shows a sample Green Throttle thumbnail (not actual size):

Game Banner

The game banner you supply for Green Throttle Arena needs to match the following requirements:

- 1024 x 500 pixels
- Embedded player count icon (optional)

Note: These specifications are the same as the specifications for the Google Play featured graphic.

The following shows a couple of game banners with two styles of embedded player count icons (not actual size):

Video Trailer

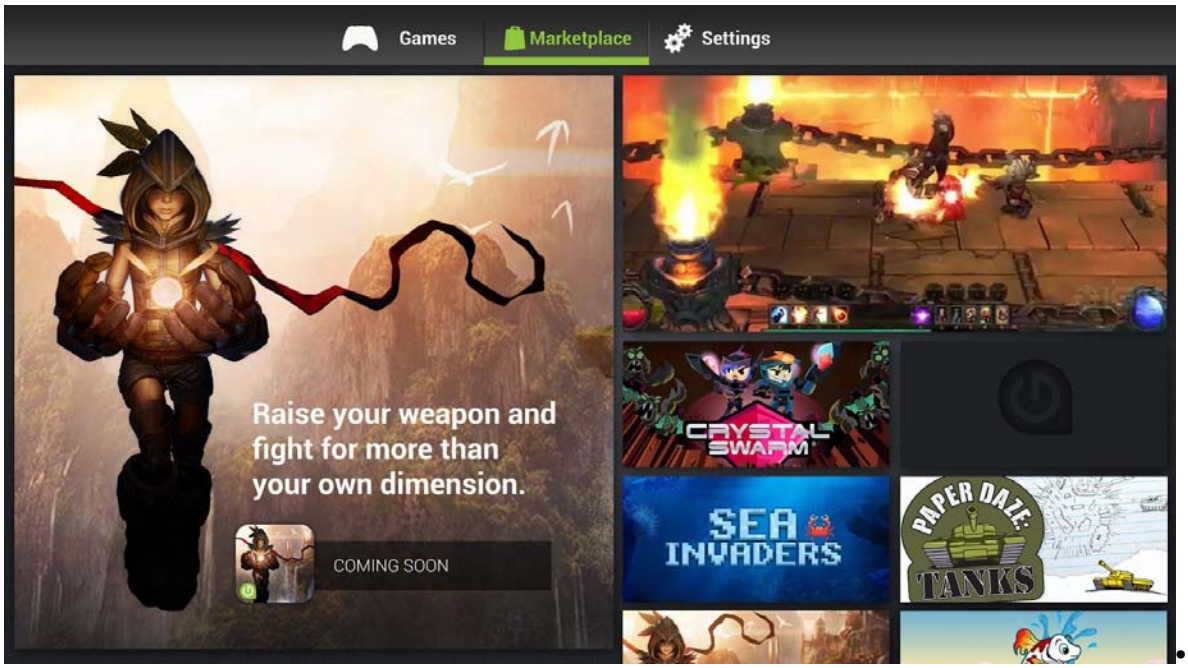
You can optionally supply a video trailer for use in the Green Throttle Arena. Host the video trailer on YouTube, preferably in HD format to ensure the highest quality.

Recommended Tab Banners

You can optionally supply *Recommended* tab banners for Green Throttle Arena, matching the following requirements:

- Half size: 980 x 940 pixels
- Quarter size: 890 x 420 pixels

The following shows sample half- and quarter-size *Recommended* tab banners (not actual size):



Appendix C
Frequently Asked Questions

This appendix lists the most frequently asked questions about the Green Throttle SDK and developing within the Green Throttle ecosystem.

Supported Engines and Devices

What devices does Green Throttle support?

The supported device list is constantly growing. For a complete list of supported devices and operating systems, see the list of supported devices.

How many controllers does Green Throttle support for local multiplayer?

Green Throttle supports up to four controllers connected to the same Android device.

Which game engines does Green Throttle support?

Green Throttle is compatible with Unity 3.5 (Unity Technologies) apps for Android, but we have plans to expand our service to support other game engines. If there's a specific engine or language you would like supported, let us know by sending an email to DevSupport@GreenThrottle.com.

Development

Do you support targeted phones or devices in the SDK?

Not at this time.

Do you have an SDK for iOS developers?

We are working to support iOS developers. Be sure to sign up for updates regarding iOS and other developments.

PROMOTION

How can I get my app promoted in the Green Throttle Arena?

Green Throttle offers several ways to promote your app in the featured games section of the Arena. Contact Green Throttle (devsupport@greenthrottle.com) for more information.

Appendix D
Additional Resources

This appendix offers a range of additional resources that you can access to help you develop your Green Throttle games.

Green Throttle Resources

This section lists the additional developer resources offered by Green Throttle Games.

Green Throttle Forums

Visit our Developer Forums at www.greenthrottleforums.com for help, tips and community news.

Technical Support

Got questions about the SDK, API, or game controllers? Contact us anytime at devsupport@greenthrottle.com.

General Development Resources

This section lists general developer resources available from a range of third parties.

- For more information about developing for the Google Android platform, see the Android developer site.
- For more information about the Eclipse IDE, see the Eclipse home page.
- For more information about the Unity game development environment, see the Unity home page.