Creating a Gamepad Application

Published 2014-10-27 | (Compatible with SDK 3.5,4.5,5.0,5.1 and 2012,2013,2014 models)

> This tutorial describes the use of the gamepad class of the External Interworking API. This class is needed to develop various game applications using different types of gamepads.

Contents

This tutorial describes the use of the gamepad API of the Hardware API. This API is needed to develop various game applications using different types of gamepads.

The application being developed in this tutorial demonstrates development of an application that uses a gamepad. The application connects to a USB HID gamepad device and gets data from the device through TV platform.

Programming of applications using gamepads is mostly based on the External Interworking API of the Hardware API.

Development Environment

Use Samsung Smart TV SDK to develop the application. The emulator provided along with the SDK can be used to debug and test the application before deploying it onto a TV. Refer Testing Your Application on a TV. Note that applications may perform better on a TV than on the emulator.

You can find general instructions for creating applications in Implementing Your Application Code.

Prerequisites

To develop TV applications, you need:

Samsung TV connected to the Internet

SDK **(Samsung Smart TV SDK is recommended)** or a text editor for creating HTML, JavaScript and CSS files

Source Files

> Note
>
> The files needed for the sample application are **here**.

The tutorial Gamepad Application is located under Tutorial_Gamepad.

Source files in directories are explained in the table:

Directory	Description
css	Contains the stylesheet file TutorialGamepad.css

Directory	Description
js	Contains the JavaScript file TutorialGamepad.js which does the following: - initializes the module - registers events - key handling - displays the results

Starting the Gamepad Application

To start the gamepad application,

1. Select **Open App** on the menu and select Gamepad_Tutorial.

   The following display on emulator indicates that the application has started successfully:

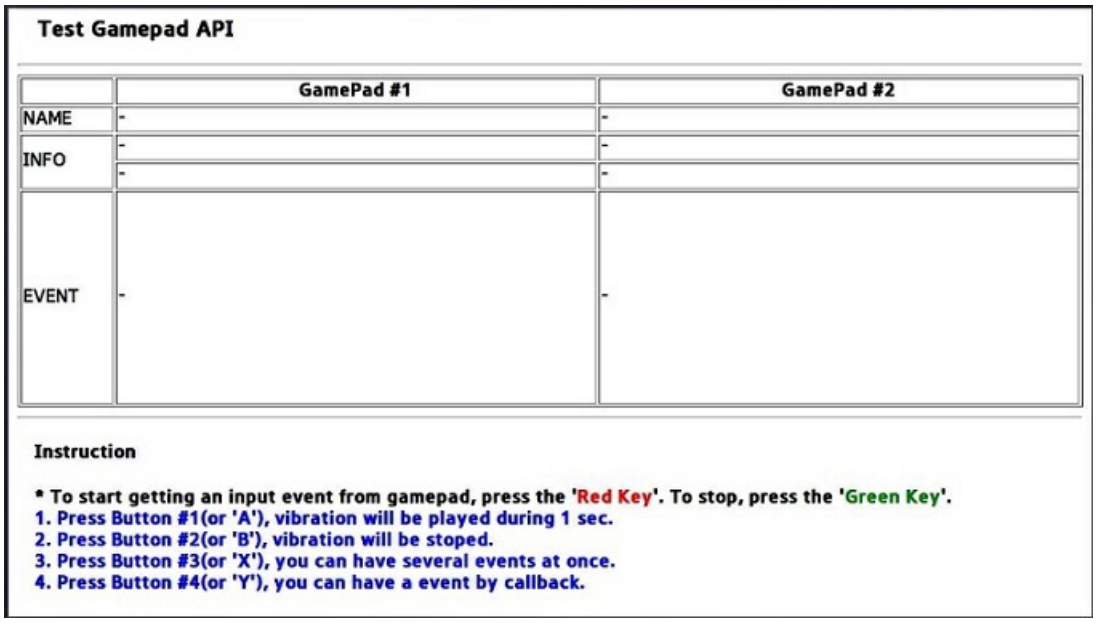

   Figure: Gamepad application started

2. Connect two gamepads and press the **red** key of the remote control; you should see the following display on emulator:

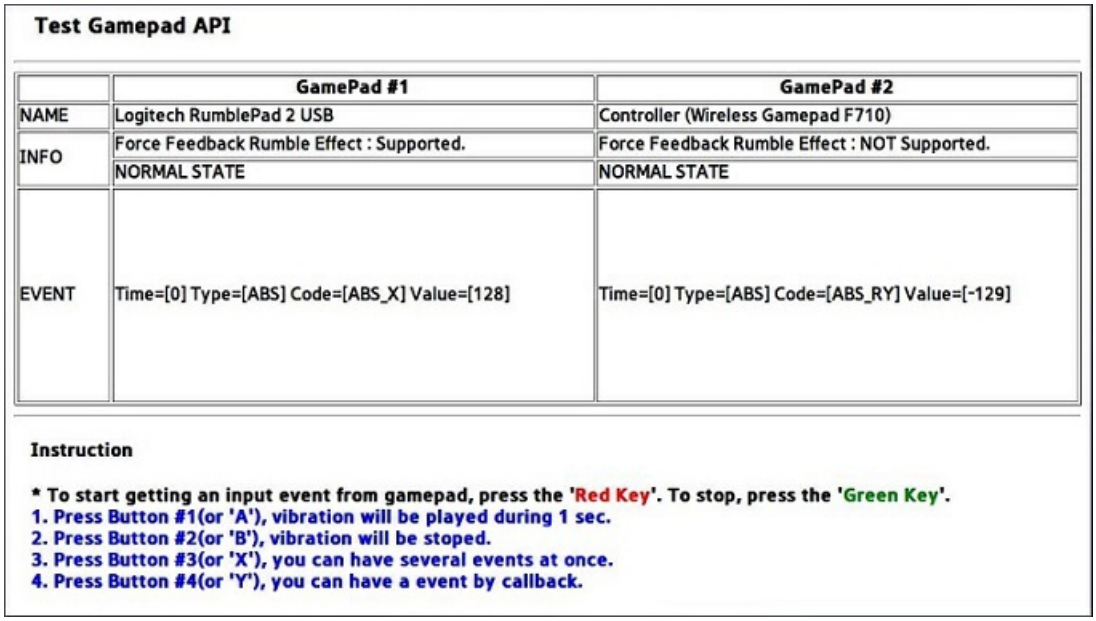

   Figure: Gamepads connected

3. Change operation mode of each gamepad to **CALLBACK STATE** or **MULTI-EVENT MODE** ; you should see the following display on emulator:

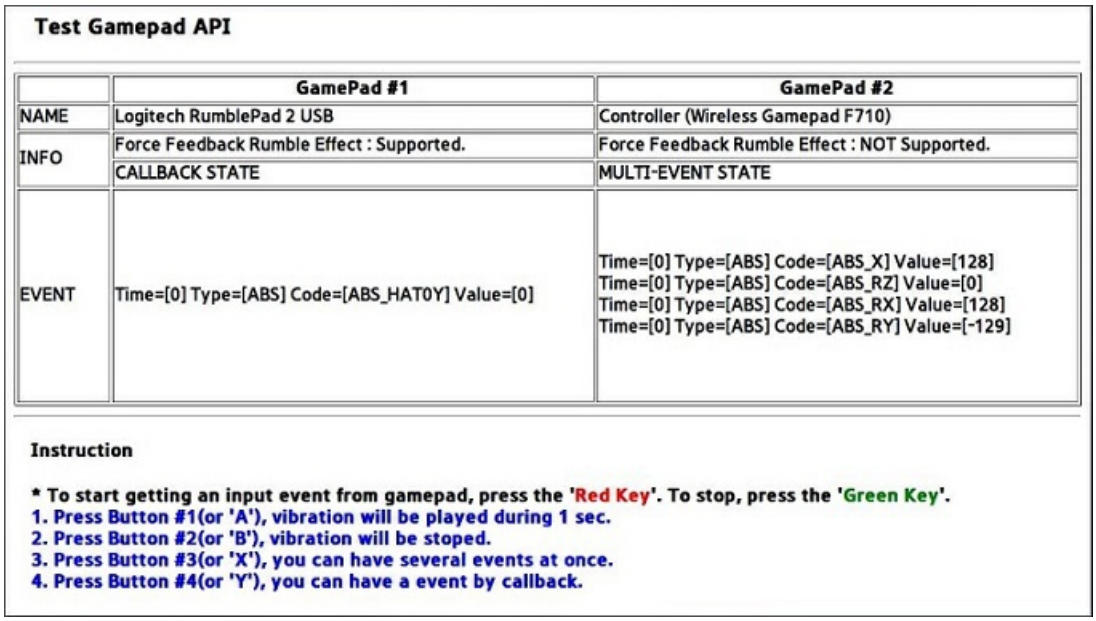

Figure: Changed operation state of Gamepads

Using the Gamepad

Following sections explain how to setup and use gamepads connected to the system. Code samples are based on the tutorial gamepad application. This tutorial obtains and displays input data from a gamepad. Observe how the displayed data changes in accordance with the movement and twisting of the stick and button pressing. In addition, it plays the force feedback effect depending on the button pressed.

Getting an Available Gamepad

If gamepads are connected, the window.webapis.gamepad.getGamepads() function will return a Gamepad class object for each of those gamepads. You can check if the force feedback rumble effect is supported with the Gamepad.isForceFeedbackSupported() function.

```
var gamepad = window.webapis.gamepad || {};

var Main = {
    oWidgetAPI: null,
    oTVKey: null,
    nInterval: 30, //30ms
    oGamepad: new Array(3),
    bForceFeedback: new Array(3),
    bRegisterCB: false,
};

Main.keyDown = function () {

    var keyCode = event.keyCode;

    switch (keyCode) {
        case Main.oTVKey.KEY_RED:
            gamepad.getGamepads(Main.onGamepadObtained);
            if (Main.bRegisterCB == false) {
                gamepad.registerManagerCallback(Main.onEventGamepadManager);
                Main.bRegisterCB = true;
            }
            break;

        case Main.oTVKey.KEY_GREEN:
            Main.oGamepad[0] = null;
```

```
            Main.oGamepad[1] = null;
            Main.oGamepad[2] = null;
            break;

        case Main.oTVKey.KEY_YELLOW:
            Main.getABSValueRange(0);
            Main.getABSValueRange(1);
            Main.getABSValueRange(2);
            break;

        default:
            break;
    }
    return;
}

Main.onGamepadObtained = function (gamepads) {
    if (gamepads.length > 0) {
        for (var i = 0; i < gamepads.length; i++) {
            if (gamepads[i] != null) {
                Main.oGamepad[i] = gamepads[i];
                Main.bForceFeedback[i] = Main
                    .oGamepad[i]
                    .isForceFeedbackSupported();
                if (Main.bForceFeedback[i] == true) {
                    // ...
                } else {
                    // ...
                }
                Main.getABSValueRange(i);
                Main.handleInputEvent(i);
            }
        }
    } else {
        alert("[Gamepad]: no gamepads found");
    }
    return;
}
```

Registering a Callback Function for a Device Connection Event

To receive a device connection event, register your callback function with the window.webapis.gamepad.registerManagerCallback() function. Once the callback registration is done, callback functions are called at connecting or disconnecting a gamepad. The callback function can get a window.webapis.gamepad.ManagerEvent class object including event type and gamepad name as an input parameter.

```
Main.keyDown = function () {
    var keyCode = event.keyCode;
    switch (keyCode) {
    case Main.oTVKey.KEY_RED:
        gamepad.getGamepads(Main.onGamepadObtained);
        if (Main.bRegisterCB == false) {
            gamepad.registerManagerCallback(Main.onEventGamepadManager);
            Main.bRegisterCB = true;
        }
        break;
    }
    return;
}

Main.onEventGamepadManager = function (devEvInfo) {
    switch (devEvInfo.eventType) {

        case gamepad.MGR_EVENT_DEV_DISCONNECT:
            for (var i = 0; i < 3; i++) {
                if (Main.oGamepad[i].getUniqueID() == devEvInfo.UID) {
                    Main.oGamepad[i] = null;
                    Main.bForceFeedback[i] = false;
                }
            }
            break;

        default:
            break;
    }

    return;
}
```

Getting Input Data from a Gamepad

To get input data from a gamepad, call the Gamepad.getInputEvent() function. This function returns a GamepadEvent class object; event time, type, code, value. The event time is always zero. To get events from gamepads continuously, polling must be done on this function.

```
Main.handleInputEvent = function (index) {
    if (Main.oGamepad[index] != null) {
        var effectStatus = 0;
        var inputEvent = Main.oGamepad[index].getInputEvent();
        if (inputEvent != null) {
            Main.showEventString(index, inputEvent.time, inputEvent.type, inputEvent.code, inputEvent.value);

            switch (inputEvent.type) {
                case gamepad.EV_KEY:
                    switch (inputEvent.code) {
                    case gamepad.BTN_1:
                        effectStatus = 1;
                        break;
                    case gamepad.BTN_2:
                        effectStatus = -1;
                        break;
                    default:
                        effectStatus = 0;
                        break;
                    }
                    //...
                    break;
                default:
                    break;
            }
        }
        setTimeout(funcName, Main.nInterval);
    } else {
        //...
    }
    return;
}
```

Playing and Stopping the Force Feedback Rumble Effect

If the gamepad supports the rumble effect of force feedback, it can be played by playForceFeedback() or stopped by stopForceFeedback().

```
Main.handleInputEvent = function (index) {
    if (Main.oGamepad[index] != null) {
        var effectStatus = 0;
        var inputEvent = Main.oGamepad[index]
            .getInputEvent();
        if (inputEvent != null) {
            Main.showEventString(index, inputEvent.time, inputEvent.type, inputEvent.code, inputEvent.value);
            switch (inputEvent.type) {
                case gamepad.EV_KEY:
                    {
                        switch (inputEvent.code) {
                            case gamepad.BTN_1:
                                effectStatus = 1;
                                break;

                            case gamepad.BTN_2:
                                effectStatus = -1;
                                break;

                            default:
                                effectStatus = 0;
                                break;
                        }

                        if (inputEvent.value == gamepad.KEY_PRESSED) {
                            if ((Main.bForceFeedback[index] == true) && (effectStatus == 1)) {
                                if (Main.oGamepad[index]
                                    .playForceFeedback() == false) {
                                    alert("[Gamepad]: ERROR! playForceFeedback()");
                                }
                            } else if ((Main.bForceFeedback[index] == true) && (effectStatus == -1)) {
                                if (Main.oGamepad[index]
                                    .stopForceFeedback() == false) {
                                    alert("[Gamepad]: ERROR! stopForceFeedback()");
                                }
                            }
                        }
                        break;
                    }

                default:
                    break;
            }
        }
        //...
        return;
    }
}
```

Set Callback State to Receive Input Event

To get the input date from a gamepad by a callback function like Device Connection Event, call the Gamepad.setActive() function after registration of your callback function with the Gamepad.registerDeviceCallback() function. Then, the Gamepad.getInputEvent() and Gamepad.getInputEventEx(count) functions will not be available.

> Note
>
> This API is only supported on Samsung Smart TV models for 2013 and later. Available for Samsung SDK 4.0 or higher.

```
Main.onGamepadObtained = function (gamepads) {
    if (gamepads.length > 0) {
        for (var i = 0; i < gamepads.length; i++) {
            if (gamepads[i] != null) {
                Main.oGamepad[i] = gamepads[i];
                Main.oGamepad[i].registerDeviceCallback(Main.onEventGamepad);
                //...
                Main.handleInputEvent(i);
            }
        }
    } else {
        alert("[Gamepad]: no gamepads found");
    }
    return;
}

Main.onEventGamepad = function (index, inputEvent) {
    var txtEventID = "txtEvent" + index;
    if (inputEvent != null) {
        Main.showEventString(index, inputEvent.time, inputEvent.type, inputEvent.code, inputEvent.value, false);
        Main.processSpecificEvent(index, inputEvent.time, inputEvent.type, inputEvent.code, inputEvent.value);
    } else {
        document.getElementById(txtEventID).innerHTML = "-";
    }
    return;
}

Main.processSpecificEvent = function (index, evTime, evType, evCode, evValue) {
    if ((evType == gamepad.EV_KEY) && (evValue == gamepad.KEY_PRESSED)) {
        if (evCode == gamepad.BTN_4) {
            if (Main.nGamepadStatus[index] != STATUS_CALLBACK) {
                if (Main.oGamepad[index].setActive() == true) {
                    alert("[Gamepad]: Set Active!");
                    Main.nGamepadStatus[index] = STATUS_CALLBACK;
                    var txtModeID = "txtMode" + index;
                    document.getElementById(txtModeID).innerHTML = "CALLBACK STATE";
                    return;
                }
            }
        } else if (evCode == gamepad.BTN_3) {
            //...
        }
        //...
    }
    return;
}
```

Getting Several Input Data from a Gamepad

To get several input data from a gamepad at once, call the Gamepad.getInputEventEx(count) function. This function returns a

array of GamepadEvent objects. The **first element** of this array is **the count of events**.

> Note
>
> This API is only supported on Samsung Smart TV models for 2013 and later. Available for Samsung SDK 4.0 or higher.

```
Main.handleMultiInputEvents = function (index) {
    var txtEventID = "txtEvent" + index;
    var funcName = "Main.handleMultiInputEvents(" + index + ")";
    if (Main.oGamepad[index] != null) {
        //event count : 5
        var inputEventArray = Main.oGamepad[index].getInputEventEx(5);
        if (inputEventArray != null) {
            //alert("inputEventArray[0] : " + inputEventArray[0]);
            var evCnt = inputEventArray[0];
            var bContinue = false;
            for (var i = 0; i < evCnt; i++) {
                if (i == 0) {
                    bContinue = false;
                } else {
                    bContinue = true;
                }
                Main.showEventString(index, inputEventArray[i + 1].time, inputEventArray[i + 1].type, inputEventArray[i + 1].co
de, inputEventArray[i + 1].value, bContinue);
                Main.processSpecificEvent(index, inputEventArray[i + 1].time, inputEventArray[i + 1].type, inputEventArray[i +
1].code, inputEventArray[i + 1].value);
            }
        }
        if (Main.nGamepadStatus[index] != STATUS_MULTI) {
            alert("[Controller] Stop handleMultiInputEvents(" + index + ")");
        } else {
            setTimeout(funcName, Main.nInterval);
        }
    } else {
        document.getElementById(txtEventID).innerHTML = "-";
    }
    return;
}
```