Backbox Communication Protocol (BCP) Server for Unity

Developer's Guide

June 2015

Version 1.00

Table of Contents

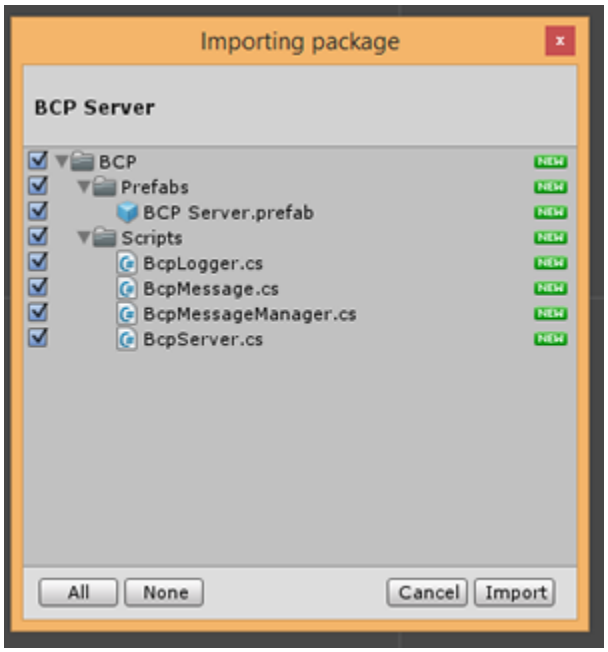
Overview

This Unity add-on was developed to provide the communication layer between a pinball controller and a Unity backbox application using the Backbox Communication Protocol (BCP). BCP is a simple, fast protocol for communications between an implementation of a pinball game controller and a multimedia controller. The BCP Specification was developed as part of the Mission Pinball Framework. This guide is designed to get you started building your own multimedia controller for your backbox using Unity. This document does not attempt to cover the fundamentals of developing applications in Unity. If you are new to Unity, there are numerous tutorials and a very active development community to help get you started. It should also be noted that this add-on only provides the communication layer for your project and does not contain any additional logic or functionality building blocks for a pinball backbox.

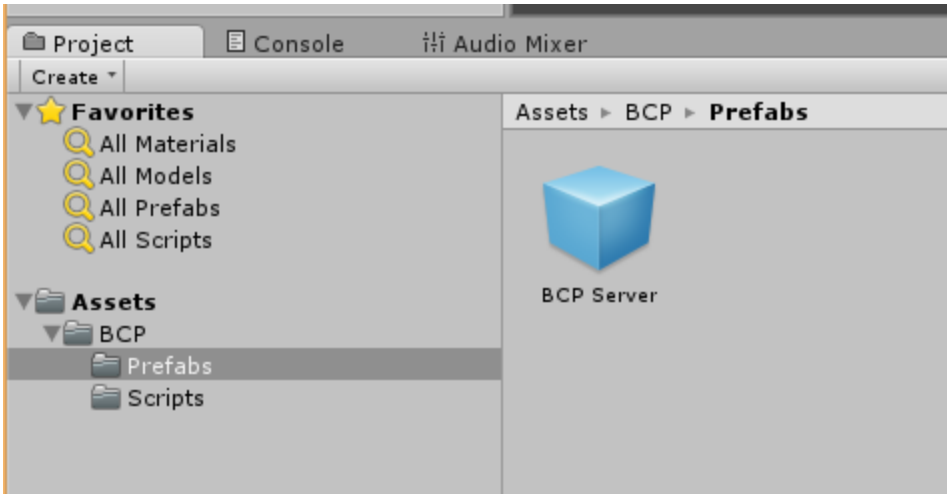
Quick Start: Setup Your Unity Project to Use the BCP Server

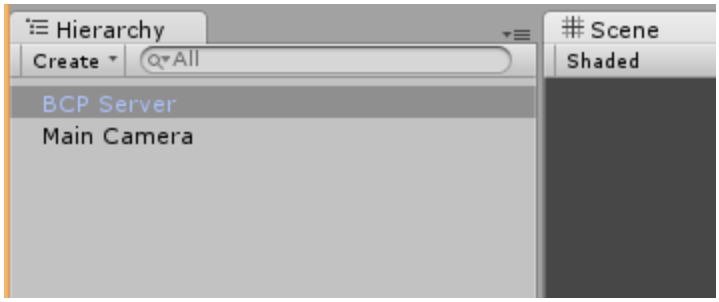
Ready to jump right in and get started using the BCP Server right away? This section will get the BCP server up and running. However, it will do nothing other than receive messages. In order to make Unity do something when receiving messages, additional work is required. If you run into problems, feel free to post in the Mission Pinball Framework BCP Specification forum (https://missionpinball.com/forum/f/bcp-spec/).

1. Download and install Unity 5 (http://unity3d.com/get-unity). The full version of Unity 5 is available free (see site for more details and restrictions).
2. Download and install the Backbox Communication Protocol (BCP) Server for Unity software. There are two options for getting the BCP Server for Unity software:
    a. Download a Unity package that can easily be imported into your Unity project from GitHub releases (https://github.com/missionpinball/unity-bcp-server/releases). This is the easiest way to incorporate the BCP Server into your Unity project and the next couple of steps that follow will guide you through the package import process.
    b. As an alternative, you can download the latest source code from GitHub (https://github.com/missionpinball/unity-bcp-server). Feel free to fork or clone the repository and incorporate the source code into your Unity project.
3. Create a new Unity project (or open an existing project).
4. Import the BCP Server Unity Package into your project. Use the (Assets | Import Package | Custom Package menu item and select the package you downloaded in step 2a (**BCP Server.unitypackage**).

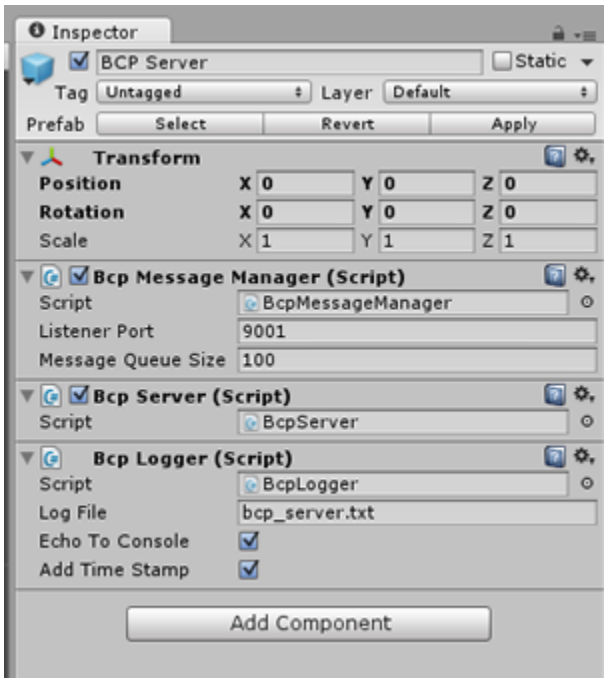


5. In the Project tab, navigate to the following path: Assets\MPF\Prefabs. Drag the BCP Server prefab to the project object hierarchy (a new GameObject will be created).

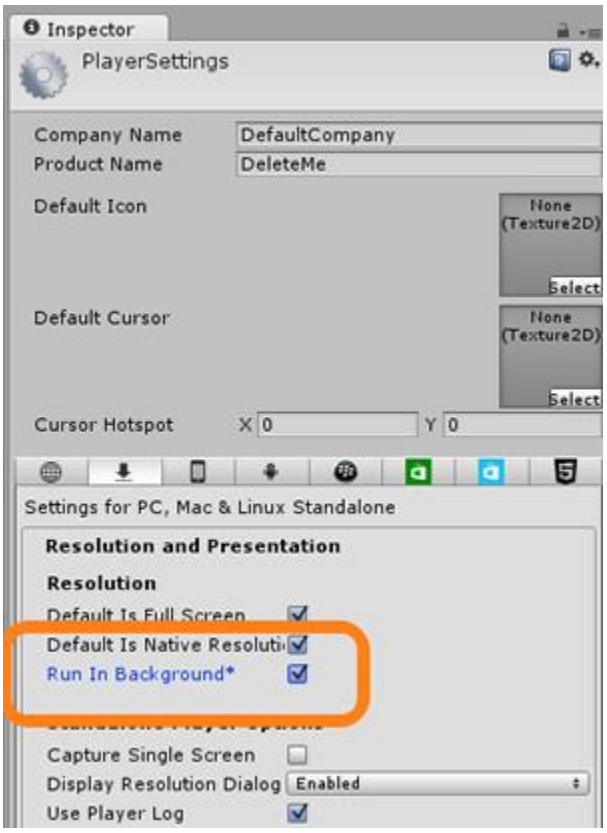




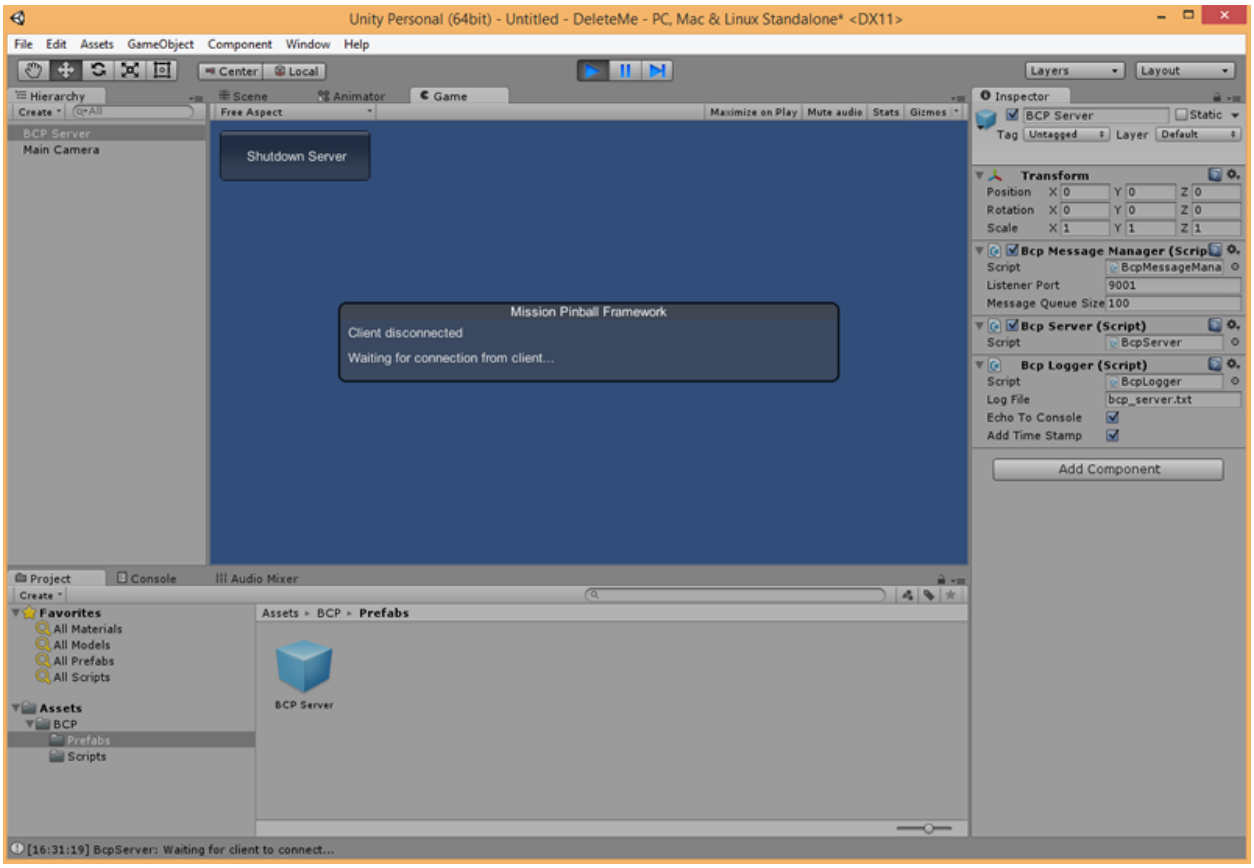
6. Select the BCP Server game object. In the Inspector, set the Listener Port to the port you wish to use to communicate with the pinball controller (default is 9001). You may have to change the BCP server configuration settings in the pinball controller to match.



7. By default, a Unity application will only run and update the screen when it has the focus. This can be a problem when working with multiple applications on the same CPU as frequently there are times when another window has the focus (such as your Python development environment). To resolve this issue, there is a setting to allow Unity to run as a background application. Select the Edit | Project Settings | Player menu item and check the Run in Background check box.

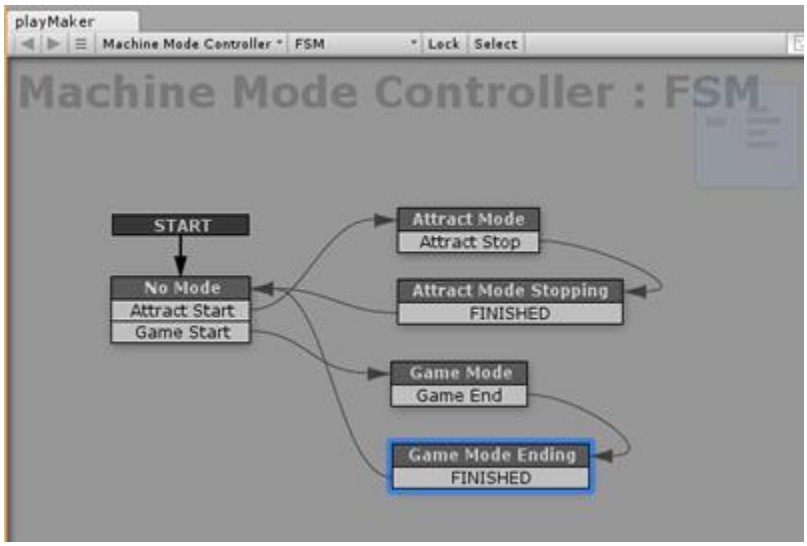


8. You can now run the application in the Unity Editor and the BCP Server will launch and wait for a connection. You can view log information in the console.



playMaker Integration

A Unity package is also available that contains many custom actions to integrate the BCP Server with playMaker without writing any additional code. playMaker (http://www.hutonggames.com), a visual scripting environment, is one of the most popular extensions on the Unity Asset Store. If you are already familiar with Unity development chances are you already know and use playMaker. An entire pinball backbox can be created in Unity without writing a single line of code using playMaker!



The BCP Server playMaker custom actions are available at GitHub as a Unity package (https://github.com/missionpinball/unity-bcp-server/releases). Simply import the **BCP Server – PlayMaker Integration.unitypackage** file (Assets | Import Package | Custom Package) and the actions will appear in the playMaker Action Browser in a new section titled BCP.

The following playMaker BCP actions are currently available:

- Get BCP Attract Start
- Get BCP Attract Stop
- Get BCP Ball End
- Get BCP Ball Start
- Get BCP Config
- Get BCP Game End
- Get BCP Game Start
- Get BCP Goodbye
- Get BCP Mode Start
- Get BCP Mode Stop
- Get BCP Player Added
- Get BCP Player Score
- Get BCP Player Turn Start
- Get BCP Player Variable
- Get BCP Switch Active
- Get BCP Switch Inactive
- Get BCP Timer Complete
- Get BCP Timer Paused
- Get BCP Timer Started

- Get BCP Timer Stopped
- Get BCP Timer Tick
- Get BCP Timer Time Added
- Get BCP Timer Time Subtracted
- Get BCP Trigger
- Send BCP Switch
- Send BCP Trigger
- Shutdown BCP Server

Technical Overview

The Backbox Communication Protocol (BCP) Server for Unity is written in C# and implements the 1.0 Draft revision of the BCP Specification (https://missionpinball.com/docs/programming-guide/bcp1-0-spec/). BCP is a simple, fast protocol for communications between an implementation of a pinball game controller and a multimedia controller. BCP communications are transmitted using TCP sockets between the client and server and are most typically co-located on the same machine.

In this Unity implementation, the BCP Server is implemented as a singleton class that is automatically created when attached to a Unity GameObject. A separate listener thread is spawned by the server to receive incoming messages. All Unity SDK calls must be made on the main Unity thread. All incoming messages are converted to BcpMessage objects and are inserted into a thread-safe queue. The BcpMessageManager (another singleton object) runs on the main Unity thread and checks the thread-safe queue for messages every frame. If there are any messages in the queue, it processes them. The BCPMessageManager object fires a specific event for each BCP message received. Event handlers can be added to any Unity GameObject via scripting to receive those events. BCP messages from the Multimedia Controller to the Pinball Controller are sent from the main Unity thread during frame processing.

Programmer's Guide

Coming soon…

Until this section is written, you can look at the custom playMaker action classes (Assets\BCP\Scripts\PlayMaker) for examples of adding BCP event handlers for each BCP command.