E4UnityIntegration-MIT: An Open-Source Unity Plug-in for Collecting Physiological Data using Empatica E4 during Gameplay

Abstract—**Physiological measurement of player experience (PX) during gameplay has been of increasing interest within game research circles. A commonly-used non-invasive wearable device for physiological measurement is the Empatica E4 wristband, which offers multiple physiological metrics, ranging from electrodermal activity to heart rate. That said, the E4's integration with popular game engines such as Unity 3D presents certain challenges due to non-obvious critical bugs in the library and limited documentation applicability within the Unity context. In this paper, we present an open-source Unity plug-in designed to mitigate the challenges associated with integrating the E4 into Unity projects: E4UnityIntegration-MIT. The plug-in exposes the E4's API for interfacing with Unity C# scripts, thereby enabling realtime data collection and monitoring. E4UnityIntegration-MIT also provides the affordance of saving the E4 data into an external file for data analysis purposes.**

Index Terms—**Empatica E4, psychophysiological measures, electrodermal activity (EDA), heart rate (HR), affective gaming, Unity 3D**

I. Introduction

Assessing player experience (PX) is a matter of importance in game research. There are a range of approaches to measuring PX, from more subjective methods such as interviews or questionnaires to more objective techniques like in-game analytics and psychophysiological measurements. The latter offers advantages such as not needing to interrupt the experience, nor introducing memory bias (by not requiring of players to recall their experience after the fact) [1]. Moreover, the immediacy in the availability of the data also allows for the possibility of integrating the live readings as a real-time input into the game. This can be (and has been [2], [3]) exploited to craft physiological feedback loops, whereby game parameters are altered according to the player's state.

Common physiological measurements taken include electrodermal activity (EDA), heart rate (HR), interbeat interval (IBI), heart rate variability (HRV), blood volume pulse (BVP), temperature, electroencephalography (EEG), electromyography (EMG), respiratory rate (RR) and eye tracking [1]. These metrics are obtained through the use of a range of commercial or research devices.

One such commonly used device [2], [4], [5] is the Empatica E4. It is an enticing tool for this kind of research because of several factors:

- It comes in a non-invasive wearable wristband form factor.
- It simultaneously records several of the aforementioned measures, namely BVP (sampled at 64Hz), EDA (sampled at 4Hz), IBI (with a 1/64 second resolution), skin temperature (sampled at 4Hz), and HR (computed in spans of 10 seconds). (Additionally, while not a physiological sensor, it also includes a 3-axis accelerometer, sampled at 32Hz.) [6]
- It provides multiple methods of using or interfacing with it, ranging from out-of-the-box working cloud solutions to a streaming service program for Windows that can forward its data streams over TCP, or APIs for integrating it into an Android or iOS application.

The availability of an Android API, the E4 link library, is particularly interesting due to the plethora of devices which run on an Android or Android-based operating system, many of which are suitable platforms for games. Indeed, even some virtual reality (VR) headsets such as Meta's Quest offerings are Android-based. Thus, there is an increasing interest in and need for integrating the E4 wristband into game engines that support Android-based game releases. One such game engine that is commonly used and supported by the E4 API is the Unity 3D game engine (referred to as Unity).

However, it is not straightforward to use the E4 link library in Unity. While Unity provides ways of interfacing with Android libraries, and E4 link is a small and decently documented library (including an official example repository of a barebones Android application that uses it), the integration proves difficult in practice. There exists a non-obvious critical bug in the library that must be worked around, and the documentation is of limited applicability in the context of Unity. Additionally, the library cannot work within the Unity player. This means the application must be deployed (and debugged) on an Android device every time, making the development workflow slow and tedious.

These considerations motivate the present work, where we publish an open-source Unity plug-in for connecting and collecting data from the E4 wristband in Unity: E4UnityIntegration-MIT. The plug-in was designed as a minimal plug-in that exposes the API for interfacing in Unity C# scripts to enable game researchers to quickly integrate the E4 wristband into their Unity projects and to capture the various types of physiological data it provides. E4UnityIntegration-MIT allows for receiving the E4's data streams in real-time, thus enabling the creation of physiological feedback loops, while at the same providing the functionality to save the received data into a file.

The paper is structured as follows: first, we elaborate on our design considerations; second, we provide a technical description of E4UnityIntegration-MIT; lastly, we present a successful use case and potential applications.

II. Design

A. Design goals

The desired functionality was to support the E4 within Unity, exposing the API to Unity C# scripts, with special attention being paid to allowing the data streams to be received and acted upon in real-time, as well as be saved to a file.

Additionally, the plug-in was designed to have a simple installation and minimal dependencies. The justification is that when choosing which sensors to use for a playtesting session or user study, it is often important to have proof of the feasibility of using a candidate device. In this case, researchers considering using the E4 may wish to quickly check if the E4 can be satisfactorily integrated into their Unity project. Thus, E4UnityIntegration-MIT was made so that it can be installed by merely dropping the files into the project's plug-ins folder. Moreover, the plug-in has no dependencies besides Empatica's E4 link.

B. Considered alternatives

Several of the ways to interface with the E4 provided by Empatica do not provide enough functionality. Indeed, the out-of-the-box cloud solution (E4 connect) is inadequate for our purposes because there is no way to programmatically connect with Empatica's cloud web portal. The E4 manager program (available for Windows and Mac) is unsuitable, too, because it can only retrieve data after the completion of a recording session, and thus has no real-time data streaming potential.

On the other hand, the E4 streaming server (available for Windows), is capable of connecting to multiple E4 devices and allows forwarding their data streams to clients through TCP. Multiple device functionality is not necessary in our case, but the TCP forwarding feature would have been sufficient, as it could have been set up to stream the data to a Unity game. However, this method has several drawbacks. First, it introduces the need for an intermediary Windows machine. Additionally, the E4 streaming server requires a low energy bluetooth USB dongle to function. While affordable, such a dongle may not always be available. This restricts how quickly and easily game researchers and developers can ascertain whether the E4 can be satisfactorily integrated into their Unity project, which is undesirable. E4UnityIntegration-MIT avoids these downsides by connecting the E4 to the Android device running the game directly over Bluetooth, and as such is more convenient to use.

III. Technical Description

A. Installation instructions

It is simple to install E4UnityIntegration-MIT. All its files must be placed in the Assets/Plugins/Android directory of a given Unity project. Then the following steps should be followed:

- Enable Custom Main Manifest in the project settings. If it is already enabled and a custom main manifest is already in use, then the plug-in must be merged into the existing one, by copying over the lines related to permissions and feature requirements.
- Enable Custom Main Gradle Template in the project settings. If it is already enabled and a custom main gradle template is already in use, then the plug-in must be merged into the existing one, by copying over the *okhttp* dependency (required by Empatica's E4 link).

Lastly, needless to state, Empatica's E4 link is also needed. It is available for download through Empatica's E4 connect web portal in the developer area [7].

B. Plug-in overview

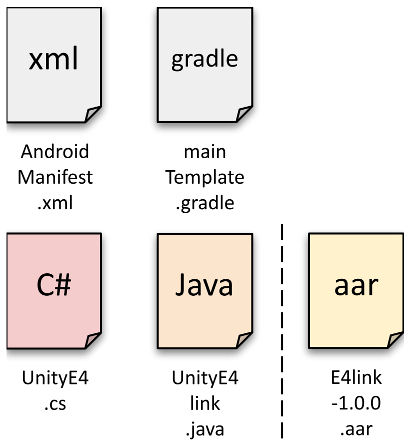

Fig. 1. E4UnityIntegration-MIT files and dependencies. At the top are the build and configuration files that must be used or merged with existing ones. At the bottom, left of the dashed line, are the source files that may be edited by users to achieve desired functionality. To the right of the dashed line is Empatica's E4 link library; it is not distributed with the plug-in and must be obtained directly from Empatica.

The plug-in is available for download online [8]. It is provided in the form of open-source code files; users can use the plug-in by editing them directly. The provided code's behavior is to connect to the first E4 it encounters, which we expect is a useful default. Following is an overview of each file's purpose and content.

1) `UnityE4.cs`*:* Unity C# script containing the UnityE4 C# class. Attach the script to an empty GameObject to use the plug-in.

Inside, the Empatica API is exposed by providing two inner classes, `UnityE4linkDataDelegateCallback` and `UnityE4linkStatusDelegateCallback`, which implement the E4 link interfaces for receiving data and status updates, respectively. Users wishing to integrate the plug-in to their use cases should alter the implementation of these classes' callback methods accordingly.

Please note that callback code may not execute Unity API calls. Unity limits API calls to running on its main thread. Instead, the callbacks are called by the E4 link library's code, which runs on a different thread. To react to received data or status updates, store the information and trigger the desired behavior on the following Unity Update.

Additionally, there is also code to record the received data at an interval (default is every .25 seconds) and store it in a file.

2) `UnityE4link.java`: Auxiliary Java file containing the UnityE4link class. It serves as a necessary bridge between E4 link and `UnityE4.cs`.

Having to work around a bug in the Empatica library has also shaped the plug-in's design. E4 link exposes two java interfaces, one to receive the E4's status updates and another to receive its data updates. Both are essential for the usage of the library. However, there exists a bug whereby a method from one interface is called on an object whose class implements the other interface[1]. The bug is critical in that it causes the application to instantly crash if allowed to happen. The workaround is to have one same class implement both interfaces. However, the C# class that Unity provides to implement java interfaces (AndroidJavaProxy) is only able to implement one at a time. As such, it is unfortunately not possible to have all of the plug-in's code in a single C# script. We have therefore opted to implement the workaround in this separate file.

Introducing an intermediary Java file is not without advantages, however. The present implementation allows programming E4 link related functionalities directly in Java, which may result in simpler code considering the alternative is having to call Java's utilities from C#. For example, the provided default behavior of connecting to the first E4 encountered is contained in this file.

3) `AndroidManifest.xml`: The plug-in's custom main manifest. Specifies the need for Bluetooth, internet, and location permissions, as well as the Bluetooth low energy feature.

4) `mainTemplate.gradle`: The plug-in's custom main gradle template. Specifies E4 link's okhttp dependency.

IV. Use Case and Applications

The intended user base for this plug-in are game researchers and developers who use Unity for game development and research. Its potential applications include physiological data gathering, as in [5], or the creation of physiological feedback loops, as in [2]. Both of these studies employed the E4, demonstrating the interest and suitability of the device for these uses.

V. Conclusion

E4UnityIntegration-MIT is an open-source Unity plug-in designed for game researchers and developers to integrate Empatica's E4 wristband into Unity projects. Its intended use cases include physiological data collection and storage as well as development of physiological feedback loops in games. We have developed the plug-in in response to the difficulty in integrating Empatica's Android E4 link library into Unity and have published it open-source to enable the game research community to easily integrate it into their projects.

[1]This bug was reported (on September 2020) and acknowledged by Empatica (on November 2020) [9] in the E4 link sample Android project repository. The issue was closed as completed on November 2020. Nevertheless, the latest version of E4 link available for download is from February 2020, before the report, and the bug is still present.

References

[1] L. E. Nacke, "An introduction to physiological player metrics for evaluating games," *Game Analytics: Maximizing the value of player data*, pp. 585–619, 2013.

[2] H. Polman, "The impact of changing moods based on real-time biometric measurements on player experience," in *Gaming, Simulation and Innovations: Challenges and Opportunities: 52nd International Simulation and Gaming Association Conference, ISAGA 2021, Indore, India, September 6–10, 2021, Revised Selected Papers*. Springer, 2022, pp. 171–181.

[3] F. Chiossi, R. Welsch, S. Villa, L. Chuang, and S. Mayer, "Virtual reality adaptation using electrodermal activity to support the user experience," *Big Data and Cognitive Computing*, vol. 6, no. 2, p. 55, 2022.

[4] E. S. Siqueira, M. C. Fleury, M. V. Lamar, A. Drachen, C. D. Castanho, and R. P. Jacobi, "An automated approach to estimate player experience in game events from psychophysiological data," *Multimedia Tools and Applications*, pp. 1–32, 2022.

[5] A. F. Bulagang, J. Mountstephens, and J. Teo, "Multiclass emotion prediction using heart rate and virtual reality stimuli," *Journal of Big Data*, vol. 8, pp. 1–12, 2021.

[6] Data export and formatting from e4 connect. [Online]. Available: https://support.empatica.com/hc/en-us/articles/201608896-Data-export-and-formatting-from-E4-connect-

[7] E4 for developer. [Online]. Available: https://e4.empatica.com/connect/developer.php

[8] An android plugin for e4 wristband integration in unity. [Online]. Available: https://osf.io/v9whk/?view_only=dc43354770044134a45c0a74c312514f

[9] Native code expects "didupdateonwriststatus" in "empadatadelegate". [Online]. Available: https://github.com/empatica/e4link-sample-project-android/issues/2