Обновление плагина Brain Flow Unreal Engine 5

Аннотация. В статье рассмотрен процесс интеграции плагина Brain Flow в Unreal Engine 5. Рассмотрено существующее решение от Brain Flow, предложены улучшения и пример использования.

Общие сведения о Brain Flow SDK

Brain Flow SDK реализует базовое взаимодействие с COM типичными для BCI, большинство существующих неинвазивных нейронных интерфейсов [1] позволяют работать с данными через этот интерфейс, в том числе интерфейс позволяет эмулировать работу аппаратной части для тестирования системы на ранних стадиях разработки или если аппарат недоступен.

Решение предоставляет общий API для наиболее популярных языков программирования (Python, C++, Java, C#, R, MATLAB, Julia, Rust). Кроме базового взаимодействия в библиотеке есть средства для обработки сырых данных (фильтрация, трансформация, шумоподавление, простая регрессионная классификация и др.).

Качество документации Brain Flow

Раздел интеграции в игровые движки оказался неактуален, но на сайте есть примеры кода для всех поддерживаемых языков и они, при проверке, тоже оказались рабочими только частично.

В целом документация по библиотеке ограничивается интерфейсами и не предоставляет ни описаний сигнатуры, ни описаний внутренних алгоритмов. Частично эта проблема решается тем, что исходный код открыт, но то, что примеры кода не работают по умолчанию, сильно осложняет процесс использования библиотеки. На некоторые вопросы получилось найти ответы в авторском Slack канале, но это не улучшает качество документации.

Интеграция Brain Flow в игровые движки

В документации есть отдельный раздел [2] про интеграцию в Unity, Unreal Engine, CryEngine. На данный момент, наибольший интерес составляет работа с Unreal Engine т. к. он активно развивается, набирает популярность и предоставляет большое количество инструментов для разработки приложений, в том числе, с поддержкой VR.

Плагин для Unreal Engine распространяется в виде Github репозитория [3], а не готового пакета в Epic Game Store. Плагин был актуален на момент выхода UE 4.25, но при переходе на UE5 большое количество API Unreal были переработаны и плагин требует доработки. Также большим недостатком репозитория является отсутствие примеров кода.

Сама библиотека Brain Flow всех примерах используется в виде Unmanaged (C++) версии, которую требуется собирать самостоятельно с CMake для выбранной платформы. Для C# существует пре-компилированный контейнер NuGet, а для Python библиотека доступна для установки через PYPI.

Рассмотрим все этапы интеграции библиотеки в Unreal Engine начиная со сборки из исходного кода и заканчивая выводом данных.

Сборка CMake

Для сборки рекомендуется использовать последний стабильный релиз из официального репозитория Brain Flow [4], для работы на конкретной операционной системе (далее ОС), библиотека должна быть собрана именно на ней, т. к. на низком уровне используются системные команды, а они имеют различный интерфейс и реализацию в зависимости от ОС. Команда для запуска сборки CMake с помощью скрипта python:

*python tools\build.py*

*--clear-build-dir*

*--cmake-install-prefix %build\_dir%\x64\_dynamic*

*--msvc-runtime dynamic*

*--arch %version%*

Вместо *%build\_dir%* следует указать директорию, в которую собранные файлы будут скопированы, лучше использовать абсолютный путь до директории с исходными файлами плагина в проекте.

Последние 2 аргумента специфичны для ОС Windows, вместо *%version%* требуется указания разрядности системы: x32 и x64.

Восстанавливаем компилируемость проекта в UE5

Добавив плагин в директорию проекта, получаем ошибки компиляции, а именно ошибки UnrealBuildTool:

error CS0117: 'UnrealTargetPlatform' does not contain a definition for 'Win32'

error CS0103: The name 'PublicLibraryPaths' does not exist in the current context

О причине первой ошибки был найден пресс-релиз [5]:

*«We are officially deprecating our Win32 support. It will be included in Unreal Engine 4.25 and Unreal Engine 4.26, but support for 32-bit Windows operating systems will be removed in a future release.»*

Источник более недоступен на сайте unreal, но сохранился в web.archive.org [6].

Таким образом, блок для сборки плагина под Win32 является устаревшим и может быть полностью удален.

По поводу второй ошибки найти официальную информацию не удалось ни на официальном сайте, ни с помощью web.archive.org, только упоминание в одном из пользовательских репозиториев [7]. Судя по оставленному автором комментариев, поддержка PublicLibraryPaths прекращена и для обратной совместимости был предоставлен аналог PublicSystemLibraryPaths, смена названия помогла исправить проблему. После этого набор ошибок изменился:

Source file '…x64\_dynamic\lib\gforce64.dll' does not exist

Source file '…x64\_dynamic\lib\gForceSDKWrapper.dll' does not exist

Ошибки связаны с изменением версии Brain Flow SDK. Список файлов соответствует версии версии Brain Flow 4.1.0, на момент написания статьи актуальная версия 5.8.1, обновим список загружаемых dll и lib файлов в соответствии с текущим списком:

Таблица 1. Список файлов brain flow при сборке из исходного кода

|  |  |
| --- | --- |
| **Lib’s** | **Dll’s** |
| BoardController.lib | BoardController.dll |
| BrainBitLib.lib | BrainBitLib.dll |
| Brainflow.lib | eego-SDK.dll |
| BrainFlowBluetooth.lib | BrainFlowBluetooth.dll |
| DataHandler.lib | DataHandler.dll |
| DSPFilters.lib | GanglionLib.dll |
| GanglionLib.lib | neurosdk-x64.dll |
| kissfft-double.lib | Unicorn.dll |
| MLModule.lib | MLModule.dll |
| MuseLib.lib | MuseLib.dll |
| WaveLib.lib |  |

Определение lib файлов через относительные пути с добавлением папки в PublicSystemLibraryPaths вызывает warning’и на каждый lib файл:

Library '%name%.lib' was not resolvable to a file when used in Module 'BrainFlowPlugin', assuming it is a filename and will search library paths for it. This is slow and dependency checking will not work for it. Please update reference to be fully qualified alternatively use PublicSystemLibraryPaths if you do intended to use this slow path to suppress this warning.

В теле ошибки есть рекомендация по отказу от практики использования PublicSystemLibraryPaths. Это позволит получить ускорение сборки и проверку зависимостей, а для сохранения функциональности вместо относительных путей требуется указывать абсолютный путь до файлов.

После всех исправлений вид BrainFlowPlugin.Build.cs для Win64:

string compiledDir = Path.Combine(ModuleDirectory, "Compiled", "x64\_dynamic");

string libDir = Path.Combine(compiledDir, "lib");

// lib’s

List<string> aFileLib = new List<string>();

aFileLib.AddRange(new string[] { "BoardController.lib", "BrainBitLib.lib", "Brainflow.lib", "BrainFlowBluetooth.lib", "DataHandler.lib", "DSPFilters.lib", "GanglionLib.lib", "kissfft-double.lib", "MLModule.lib", "MuseLib.lib", "WaveLib.lib" });

aFileLib.ForEach(s => PublicAdditionalLibraries.Add(Path.Combine(libDir, s)));

// dll’s

string copyDllTo = Path.Combine(PluginDirectory, "Binaries", "Win64");

List<string> aFileDll = new List<string>();

aFileDll.AddRange(new string[] { "BoardController.dll", "BrainBitLib.dll", "BrainFlowBluetooth.dll", "DataHandler.dll", "eego-SDK.dll", "GanglionLib.dll", "MLModule.dll", "MuseLib.dll", "neurosdk-x64.dll", "Unicorn.dll" });

aFileDll.ForEach(s => RuntimeDependencies.Add(Path.Combine(copyDllTo, s), Path.Combine(libDir, $"{s}")));

// includes folder

PublicIncludePaths.Add(Path.Combine(compiledDir, "inc"));

Использование LINQ семантики позволило сократить код и добиться лучшей читаемости, после указанных выше изменений проект собирается без ошибок, для поддержки других платформ потребуется только скорректировать набор файлов dll и lib, в рамках этой работы этого сделано не было из-за того, что целевой ОС является Windows, а развертывание окружения на Mac и Linux не имеет большого смысла так как Unreal Engine на них имеет ограниченную функциональность.

Использование библиотеки

Получение данных из Brain Flow обычно происходит с использованием COM-блока приемника и требует обеспечения единственного активного подключения, остальные будет неудачными. Такое требование может быть обеспечено с помощью паттерна синглтон, но в рамках предоставляемого unreal engine фреймворка более правильно решение — это использовать жизненный цикл объектов UESubsystem, например UGameIntance или UGameMode т. к. они порождаются при старте сеанса и автоматически уничтожаются после его окончания.

Подготовим компонент базового взаимодействия с библиотекой. В качестве примера реализуем считывание данных с контроллера. Для этого создадим класс-наследник от UActorComponent в составе плагина и произвольную структуру FMeasurementCollectedStruct для передачи данных в blueprint.

В классе определим 4 функции для вывода в blueprint:

*.h*

*UFUNCTION(BlueprintCallable, Category = "Persistence", meta = (DisplayName = "Start Session", Keywords = "Session")) bool StartSession(int32 BoardId);*

*UFUNCTION(BlueprintCallable, Category = "Persistence", meta = (DisplayName = "Release Session", Keywords = "Session")) bool ReleaseSession();*

*UFUNCTION(BlueprintCallable, Category = "Persistence", meta = (DisplayName = "Grab Series AUX And Stop Collection", Keywords = "Measurement")) void GrabMeasurementAndStop(FMeasurementCollectedStruct& Measurements);*

*UFUNCTION(BlueprintCallable, Category = "Persistence", meta = (DisplayName = "Start Collecting Series AUX for", Keywords = "Measurement")) void StartCollectingFor(float Second, float& SecondDelay);*

С реализацией в .cpp можно ознакомиться на Github [8] т.к. конкретная реализация не так важна, как возможность простой реализации вывода данных из C++ в Blueprints (Рисунок 1. Использование компонента вBlueprint).

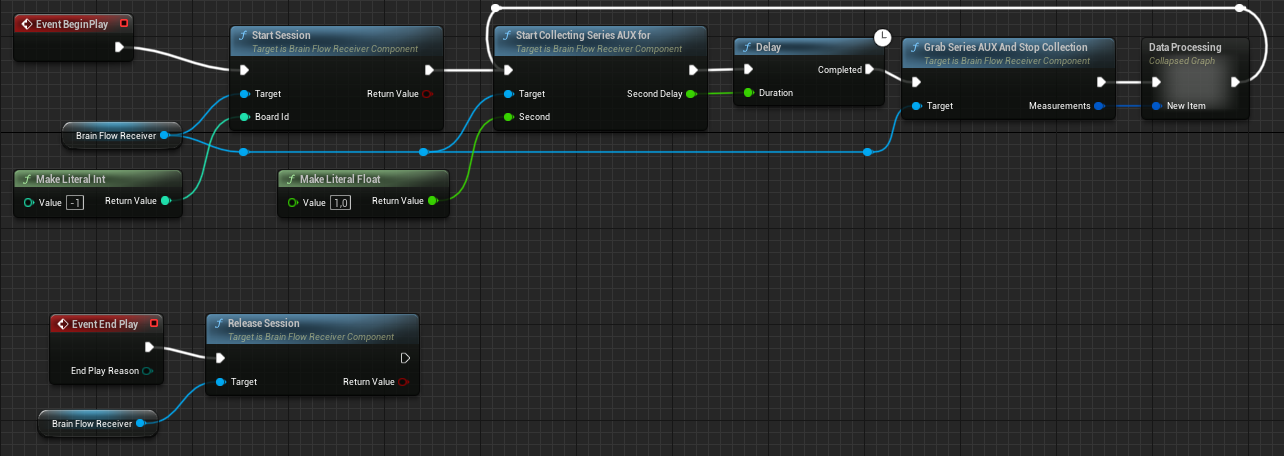


Рисунок . Использование компонента вBlueprint

Заключение

В статье приведен процесс интеграции плагина Brain Flow, внесены изменения в существующий код для устранения ошибок компиляции, приведены изменения в API и подготовлен пример кода, демонстрирующий базовое взаимодействие Brain Flow и Unreal Engine.

Исходный код полученного проекта был выложен в качестве открытого репозитория Github [8].

Список литературы

x

|  |  |
| --- | --- |
| 1. | Brain Flow Suported Boards // Brain Flows Docs. URL: https:/​/​brainflow.readthedocs.io/​en/​stable/​SupportedBoards.html (дата обращения: 04.03.2023). |
| 2. | Brain Flow Integration with Game Engines // Brain Flows Docs. URL: https:/​/​brainflow.readthedocs.io/​en/​stable/​GameEngines.html (дата обращения: 04.03.2023). |
| 3. | BrainFlowUnrealPlugin repository // Github. URL: https:/​/​github.com/​brainflow-dev/​BrainFlowUnrealPlugin (дата обращения: 04.03.2023). |
| 4. | brainflow [Электронный ресурс] // Github: [сайт]. [2023]. URL: https:/​/​github.com/​brainflow-dev/​brainflow (дата обращения: 19.06.2023). |
| 5. | Unreal Engine 4.25 Release Notes [Электронный ресурс] // Docs Unrealengine: [сайт]. [2020]. URL: https:/​/​docs.unrealengine.com/​en-US/​Support/​Builds/​ReleaseNotes/​4\_25/​index.html (дата обращения: 08.07.2020). |
| 6. | Unreal Engine 4.25 Release Notes [Электронный ресурс] // Web Archive Snapshot of Docs UnrealEnngine: [сайт]. [2020]. URL: http:/​/​web.archive.org/​web/​20200708173204/​https:/​/​docs.unrealengine.com/​en-US/​Support/​Builds/​ReleaseNotes/​4\_25/​index.html (дата обращения: 19.06.2023). |
| 7. | ue4cli: 4.24 compat: substitute PublicSystemLibraryPaths for PublicLibraryPaths [Электронный ресурс] // Github commit: [сайт]. [2019]. URL: https:/​/​github.com/​adamrehn/​ue4cli/​commit/​6229232297e897ac3decc4db02e442dcbbc3e592 (дата обращения: 19.06.2023). |
| 8. | Unreal Engine 5 plugin for BrainFlow [Электронный ресурс] // Github: [сайт]. [2023]. URL: https:/​/​github.com/​dmji/​BrainFlowUnrealEngine5Plugin (дата обращения: 22.06.2023). |

x