Arcweave Unity Plugin

Arcweave Unity Plugin is a boilerplate code for importing Arcweave Projects from arcweave.com in Unity and running them.

Getting Started

These instructions will get you a sample UI and Arcweave project running inside your Unity project.

Prerequisites

- An Unity3d installation (we tested this on Unity 2017, ping us if you got a version with issues)
- The Arcweave Unity Plugin package, which can be downloaded here (arcweave-unity-plugin-1.0.unitypackage)
- (Optional) Your own Arcweave exported project
- (Optional) Good will

Installing

Open Unity and import arcweave-unity-plugin-1.0.unitypackage.

```
Assets -> Import Package -> Custom Package...
```

The Sample

What does the sample do?

The sample contains an example UI and controls the Runner that plays the flow in your project. You can choose which connections to play and navigate your design to get a look-and-feel of your flow. On the right side, you can view the Components that are present in the current Element.

What must I do to have it?

Go to Unity's Project tab and open up the sample scene

```
Assets/Arcweave/Samples/sample_scene.unity
```

The initial plugin does not have any project loaded. So we'll need to setup one. To do so, open up the Arcweave Utility Panel.

```
Component -> Arcweave -> Settings
```

The panel should appear next to Unity's Inspector tab. You'll need to select an Arcweave exported project to load. Click the 'Setup' button in the panel and select the folder containing the sample Arcweave project

```
Assets/Arcweave/Sample/Data
```

The loader will generate the Arcweave objects that will be used by Unity at runtime. After loading finishes, click Play to start the sample.

Running your own project

First, export from Arcweave...

Export your project from Arcweave by using the 'Export Project in JSON' option (top-right). It will export an archive.

Unpack this archive and place it anywhere inside Unity's Assets folder. Note: This folder must be placed inside the "Assets" folder, so we can load sprites in Unity.

Open up the Arcweave Utility Panel

```
Component -> Arcweave -> Settings
```

Click 'Setup' and select your newly added folder to generate the runtime for your own project.

After generation is done, you can run your own project inside the sample.

Boards and Root Elements

There's no way to mark what board should be the starting board inside Arcweave. That's why, the Arcweave Utility Plugin has a second section, below the project path selection. You can choose there which board you want to be played when you Play the sample.

Same rule applies for selecting the starting element inside a board. Since there's no way to mark the starting point directly into Arcweave, a second dropdown list allows you to select the starting node.

Arcweave Classes

So, what gets generated when you import a project?

First of all, a Project object (extending Unity's ScriptableObject). This is the main container, and it keeps references to all other Arcweave objects.

Boards, which also extend ScriptableObject are created. One for each Board you've previously created in Arcweave. Components, also extending ScriptableObjects, are created. They are references by all the elements in your Project.

Elements, Connections and Attributes are created as regular classes, serialized and kept by the Project. They are referenced by UDID.

Notes, Component Folders and Board Folders are kept by the Project, but they have no intended use for now in the sample.

The Runner

This package has a ProjectRunner class that takes care of navigating the graph created in your boards.

It has a basic functionality and it's the class you'll mostly interact with. You create it like this:

```
MonoBehaviour monoRunner = someSceneObject;
ProjectRunner runner = new ProjectRunner(project, monoRunner);
```

passing the Project object you want to use and a runner on which to start Coroutines under the hood.

Start running the project using a callback that will receive the element which is currently in Play.

Advance the runner by choosing which transition to play from the currently active Element. (this is usually taken from user input)

```
ProjectRunner runner;

public void OnElementActive(Element element) {
    // Populate the view with this element
    view.Populate(element);
}
...
void Awake() {
    runner = new ProjectRunner(project, this);
    runner.Play(OnElementActive);
}
...
void Update() {
    // This should be a more ample piece of code
    // And check what transitions are available
    // in the current element
    if (Input.GetKeyDown(KeyCode.Alpha1))
        runner.ChooseTransition(0);
    else if (Input.GetKeyDown(KeyCode.Alpha2))
        runner.ChooseTransition(1);
    // etc...
}
```

Note: At the moment, Coroutines are only use to delay the callback to the next frame. So, ChooseTransition will advance the runner, but the OnElementActive callback will be received next frame.

Sample.cs and SampleViewController.cs from the sample found in this package are both good examples on how to hack your own UI and mechanics in Arcweave flows.

Additional Notes

To parse the JSON file exported by Arcweave, we make use of a JSON parser called SimpleJSON. It is often used in Unity projects, so we've left it intentionally outside Arcweave's namespace (AW). In case you already have this JSON parser in your project, in the global namespace, you can safely delete the one in the sample.

JSON parsing is only used at edit time, when importing a new project.

Authors

- Andrei Bogdan

License

This project is licensed under the MIT License - see the LICENSE.md file for details

Support

If you encounter problems in using this plugin, want to give us feedback on it or simply say hi, Write us here: info@email.arcweave.com -or- Join our Discord here: https://discord.gg/atVXxAK