Playbook

Getting Started

Addr requires you to be familiar with Addressables. If you're new or at all unfamiliar you should check the Unity provided documentation at Getting Started. Make sure to select the correct version in the Version dropdown.

Bootstrapping

Bootstrapping an application refers to the process used to execute from a cold start to the point in which control is handed off to the player/user. In a Unity project, at least one non-addressable scene is required to be included in the build. **Addr: Scene Loader** (Addr) supports bootstrapping and provides a "Bootstrap" and "AddrMenu" scene to facilitate.

To bootstrap your Unity project, follow the steps below.

Use the provided bootstrapper

1. Convert the "AddrMenu" scene to an addressable asset.
2. Remove all pre-existing scenes from the **Build Settings**.
3. Add the "Bootstrap" scene to the **Build Settings**.
4. Open the "Bootstrap" scene and select he "Bootstrapper" object in your **Hierarchy Window**.
5. Check the **Load on Start** option if it is not enabled.
6. Add the "AddrMenu" scene to the **Scenes** collection of the **SceneLoader** component.
7. Run an addressable build.
8. Build and run your Unity project or press Play in the Unity Editor.

You should see the Bootstrap scene load, and additively load the scene(s) you added to the **SceneLoader** component.

Implement your own bootstrapper

1. Create and save an empty Unity scene.
2. Add an object to the scene and add a **SceneLoader** component to it.
3. Check the **Load on Start** options.
4. Add one or more addressable scenes to the **Scenes** collection of the **SceneLoader** component.
5. Remove all pre-existing scenes from the **Build Settings**.

6. Add your scene to the **Build Settings**.
7. Run an addressable build.
8. **Build and Run** your Unity project or press **Play** in the **Unity Editor**.

Use a loading screen

A loading screen is used to hide the unpleasantries of loading/unloading resources and provide a player/user of progress while they wait. **Addr** provides you with the ability to use an addressable scene as a loading scene. When provided, it will be shown/hidden automatically while you transition between gameplay/menu scenes.

To use a loading scene, follow the steps below.

Creating your loading screen

1. Create and save an empty Unity scene.
2. Make your scene an addressable asset.
3. Add a **Canvas** to your scene.
4. Add a **Canvas Group** component to your **Canvas** object.
5. Add a **Canvas Scaler** component to your **Canvas** object.
    - Set the **Reference Resolution** to 1920 x 1080.
    - Set the **Screen Match Mode** to **Expand**.
6. Add a **Loading Tracker** component to your **Canvas** object.
    - Assign the **Canvas Group** reference.
7. Delete the **EventSystem** object created automatically by Unity.
8. Add your scene to the **LoadingScene** reference on any **SceneLoader** component.
9. Run an addressable build.
10. **Build and Run** your Unity project or press **Play** in the **Unity Editor**.

Adding a progress bar

1. Complete steps 1-7 under **Create your loading screen**.
2. Add a **Slider** component to your **Canvas** object.
    a. Uncheck **Whole Numbers** if enabled.
    b. Set the **Min Value** to 0.
    c. Set the **Max Value** to 1.
3. Update the **Loading Tracker** component.
    a. Assign the **Progress Slider** reference.

Any Scene **Loader** using a progress bar will automatically update the **Slider** value as long as the references are set.

Implement an Options screen

Options screens are provided to allow a player/user to configure gameplay, graphics, audio settings to meet hardware requirements and tailor their playing experience. **Addr** does not provide a solution for adding an options screen, but it does provide you with all the tools necessary to use an addressable scene as an options screen.

To follow our recommended approach at implementing an options screen, follow the steps below.

Creating your options screen

1. Create and save an empty Unity scene.
2. Make your scene an addressable asset.
3. Add a **Canvas** to your scene.
4. Add a **Canvas Group** component to your **Canvas** object.
5. Add a **Canvas Scaler** component to your **Canvas** object.
    - Set the **Reference Resolution** to 1920 x 1080.
    - Set the **Screen Match Mode** to **Expand**.
6. Add any Ui objects to your **Canvas**.

Loading your options screen

1. Open or create a scene that will be used to show/hide the options screen.
2. Add a new object to your scene and add a **SceneLoader** component.
    - Add your addressable option scene to the **Scenes** collection.
    - Uncheck **Load on Start** if it is enabled.
    - Uncheck **Unload Previous** if enabled.
    - Uncheck **Destroy on Unload** if enabled.
3. If your scene contains an **Event System**:
    - Set **EventSystem** *SetActive* component to **false** in **OnLoadStart**.
    - Set **EventSystem** *SetActive* component to **true** in **OnUnloadEnd**.
4. If you have a button you want to select after hiding your option screen:
    - Invoke **Button** *Select* in **OnUnloadEnd**.

Unloading your options screen

You’ll need to program a way to invoke the **Unload** method of the **SceneLoader** component at this point. If you’re using the new **Input System** you can perform the steps below.

1. Create a new object in your scene.
2. Add a **Player Input** component.
    - Set the **Default Map** to **UI**.
3. Expand **Events**.
4. Expand **UI**.
5. Set **SceneLoader** *Unload* in **Cancel**.

Extending event data

Each C# event is a partial class. If there is additional data you want to include with an event, follow the steps below.

1. Create your own definition for the event class you want to add supplemental data to.
    - Add any additional fields.
2. Create your own **SceneLoader** class and inhert the **Addr SceneLoader** class.
3. Override the corresponding factory methods for the event you’re augmenting, returning an instance of the event data with your new fields populated.
    - CreateOnLoadEndData
    - CreateOnLoadingTickData
    - CreateOnLoadStartData
    - CreateOnUnloadEndData
    - CreateOnUnloadingTickData
    - CreateOnUnloadStartData

Support

If you require further assistance, want to submit a feature request, or are looking for custom development, kindly send an email to support@over-one.studio.