**Asset Graph 매뉴얼**

**버전 1.4**

# **원본 사이트** –

[**https://docs.google.com/document/d/1yTkHZch5EaDBCria6I3Xy5UtqrfFx8n9jU1WHre-w0U/edit#**](https://docs.google.com/document/d/1yTkHZch5EaDBCria6I3Xy5UtqrfFx8n9jU1WHre-w0U/edit)

# **개요**

AssetGraph는 에셋의 처리, 변경함으로써 관련된 에섯 번들이나 사용자의 빌드에 관련된 각종 설정 및 스크립트 작업을 대폭 경감시키는 것을 목적으로 만들어진 툴 입니다..

AssetGraph를 활용하는 것으로, 에셋 번들의 작성뿐만 아니라 에셋의 작성, 편집, 설정에 관련된 제 작업을 그래픽 툴로 처리함으로써 게임 개발 도중에 발생하는 여러 가지 반복적인 작업으로부터 디자인 혹은 아티스트를 해방하는 것이 가능합니다.

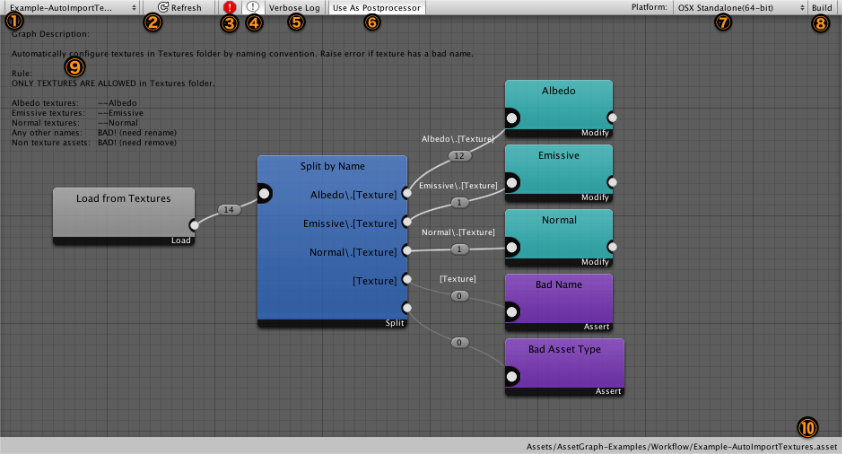
AssetGraph는 유니티 5.6.2f1이상 버전에서 동작합니다. 과거 버전에 대해서는 [여기](https://docs.google.com/document/d/1FxGn9Me12cb6jSRHceAl5fQqdCrGapmgr1q7aO0Nqco/edit)를 클릭해주세요.

P.S 이 툴을 사용할 경우 반드시 Assets 폴더 밑에 특정 폴더를 생성한 다음에 해당 폴더 밑에 AssetGraph 폴더를 넣어야 실행됩니다. Assets 폴더 바로 밑에 AssetGraph 폴더를 넣으면 실행이 되지 않습니다.

# **인터페이스**

## Graph Editor 윈도우

**Window-AssetGraph–Open Graph Editor**를 클릭하면 활성화되는 윈도우입니다.



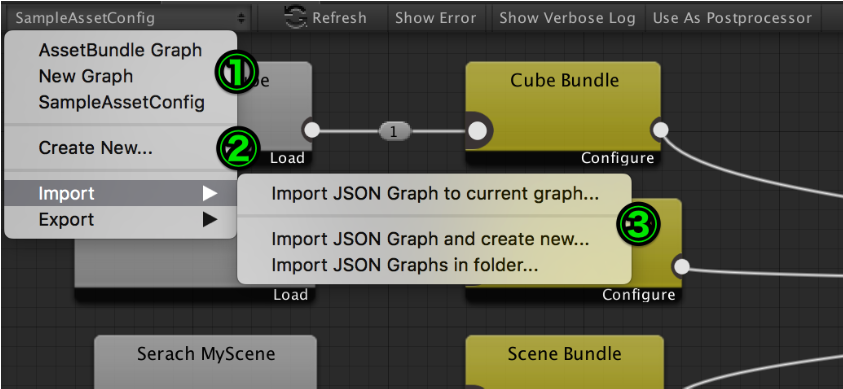
1. 열려있는 Graph 명을 표시합니다. 클릭으로 드롭 다운메뉴를 표시합니다.
2. Graph에 속한 에셋을 다시 불러옵니다.
3. Graph에서 발생한 에러를 볼 수 있습니다.
4. Graph의 설명문을 표시 혹은 비표시합니다.
5. Graph를 다시 불러오거나 빌드처리 때 상세한 메시지를 콘솔에 출력합니다. (디버그용)
6. On으로 해두면, 이 Graph를 [Asset Postprocessor](http://anchan828.github.io/editor-manual/web/assetpostprocessor.html)로써 사용할 수 있습니다.

Asset Postprocessor로써 Graph를 사용할 경우, 에셋을 갱신할 때 이 Graph 를 실행할 수 있습니다.

1. 현재 플랫폼. 클릭으로 플랫폼을 바꿀 수 있습니다.
2. 이 Graph를 빌드합니다.
3. 열려있는 Graph의 Assets이하의 경로를 표시합니다.

### **1 – 1. Graph Editor 메뉴**

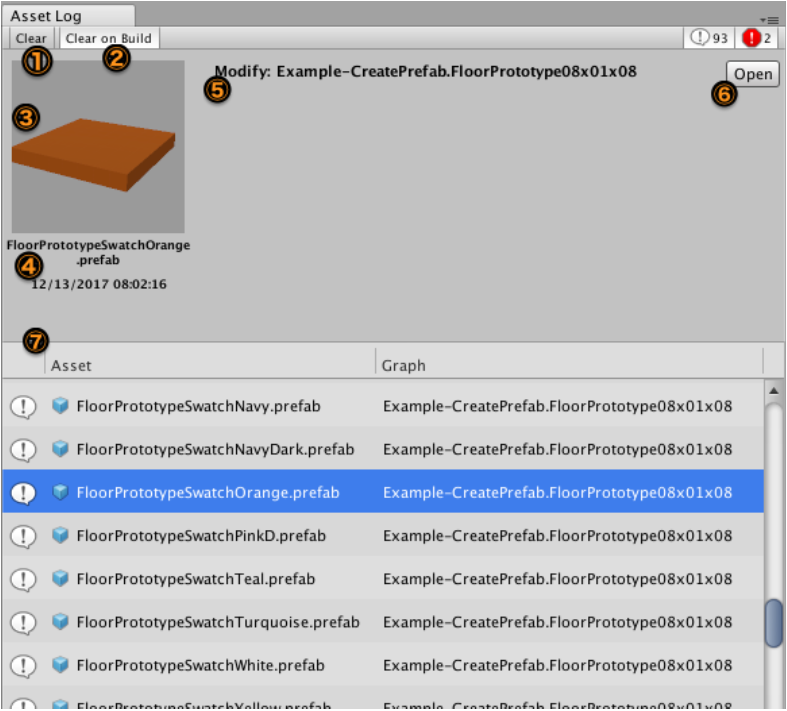
Graph 윈도우에서 1번 영역을 클릭하면 나오는 메뉴입니다.



1. Graph 에디터에서 열 Graph를 선택합니다.
2. 새로운 Graph 에셋을 작성합니다.
3. Graph를 Json형식으로 Export하거나 Import를 할 수 있습니다.

## Asset Log 윈도우

**Window-AssetGraph–Open Asset Log Window**를 클릭하면 활성화되는 윈도우입니다.

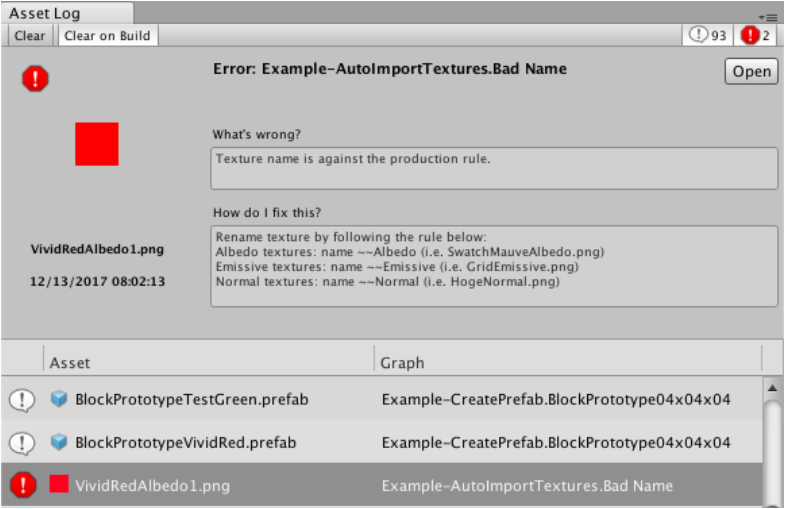


1. 로그를 초기화합니다.
2. Graph Editor로부터 명시적인 Graph를 실행할 때, 로그를 클리어 합니다.

* 유니티의 로그 콘솔에서 Clear On Play와 동일한 기능입니다.

1. 에셋의 프리 뷰를 표시합니다.
2. 에셋의 이름과 이 로그 이벤트의 발생시간을 표시합니다.
3. 이벤트 타입 및 Graph 명과 로그 이벤트가 발생한 노드 명을 표시합니다.
4. Graph를 열어 이 이벤트를 발생시킨 노드를 표시합니다.
5. 노드 이벤트 리스트 입니다.

## 2 – 1. 에러 표시

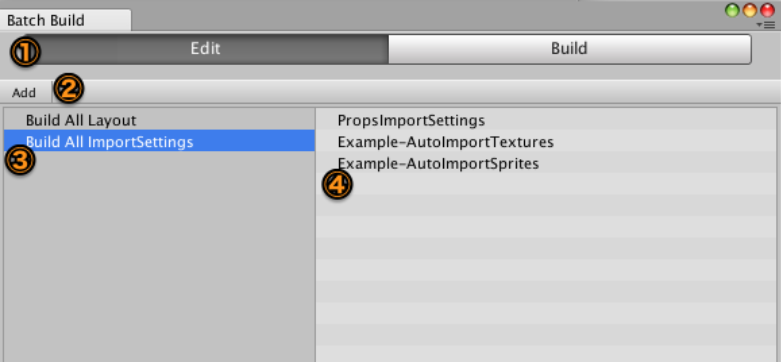


AssetGraph에서 에러가 발생했을 때, Asset Log에서 위와 같은 에러를 통지합니다. 에러에는 어떤게 발생했는지 뿐만 아니라 이용자가 자신에 대한 에러를 수정할 수 있도록, 수정방법의 가이드를 제시합니다.

## Batch Build 윈도우

**Window-AssetGraph–Open Batch Build Window**를 클릭하면 활성화되는 윈도우입니다. Batch Build 윈도우는 복수의 Graph 리스트를 복수의 플랫폼으로 대해서 한 번에 실행하는 것이 가능합니다.

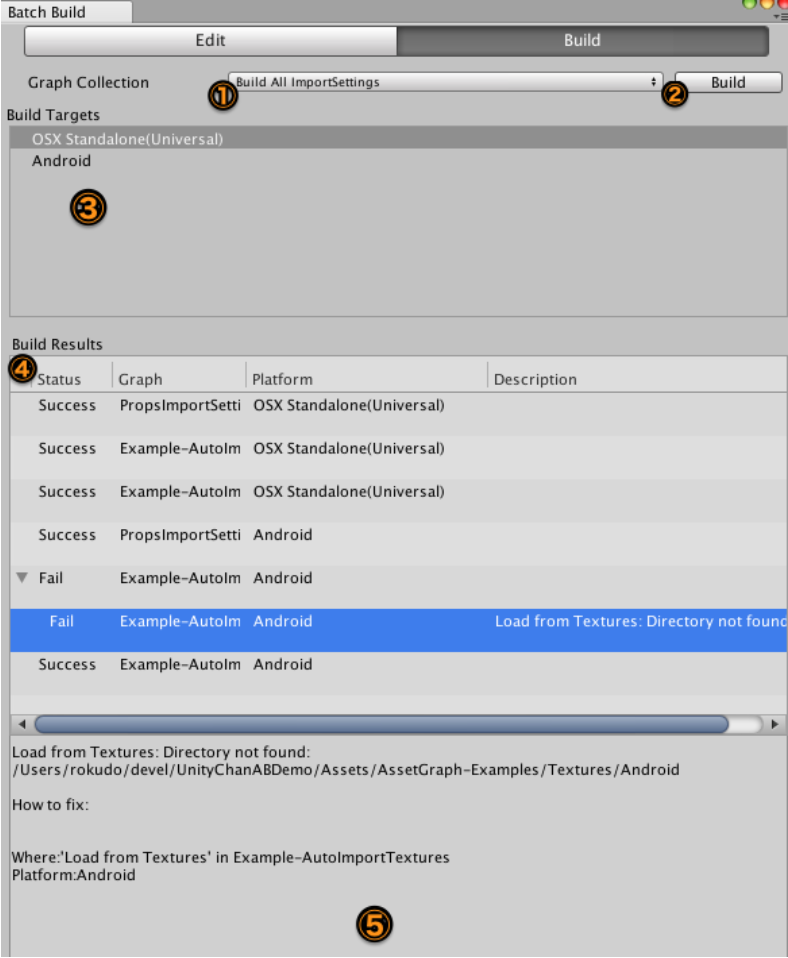
### **3 – 1. Edit 탭**



Edit 탭에서는 동시에 실행하는 Graph Collection을 편집합니다.

1. Edit와 Build 탭을 교체할 수 있습니다.
2. Add 버턴을 누르면 새로운 빈 컬렉션을 작성할 수 있습니다. 컬렉션은 Asset Graph의 에셋을 Drag & Drop하는 것으로 작성할 수 있습니다.
3. Graph Collection의 리스트를 표시합니다.
4. 선택 중인 Graph Collection에 포함된 Graph를 표시합니다. 이 컬렉션 안에 있는 Graph는 위에서부터 순번대로 실행됩니다. 그래프의 순번은 Drag & Drop하는 것으로 변경할 수 있습니다.

### **3 – 2. Build 탭**



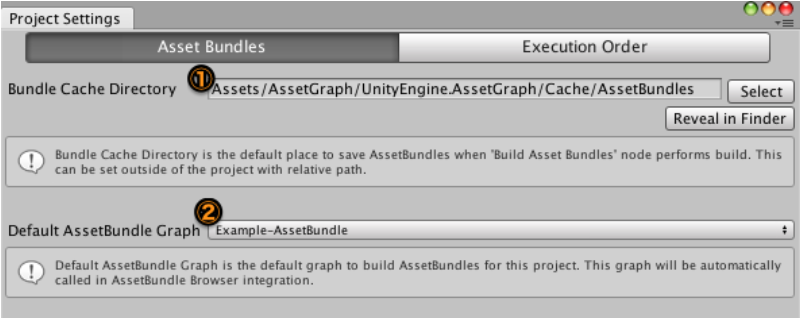
Build 탭에서는 선택한 Graph Collection을 지정한 플랫폼에 맞춰 빌드합니다.

1. 빌드할 Graph Collection를 선택합니다.
2. 선택한 Graph Collection를 빌드합니다.
3. 빌드할 타켓 플랫폼을 편집합니다. 오른쪽 클릭을 하면 플랫폼을 추가 삭제할 수 있습니다.
4. Graph Collection의 빌드 결과가 표기됩니다. 성공했을 경우 Success, 빌드가 실패했을 경우 Fail로 표시됩니다.
5. Graph 빌드 결과나 실패했을 때 실패의 이유를 표시합니다.

## 프로젝트 윈도우

AssetGraph에는 프로젝트 단위로 공유하는 설정이 있습니다. 프로젝트 윈도우에서는 AssetGraph를 이 프로젝트로 이용할 때 프로젝트 단위의 설정을 합니다. 프로젝트 설정을 변경을 할려면 Window-AssetGraph–Open Project Window를 선택하여 Project Settings 윈도우를 활성화합니다.

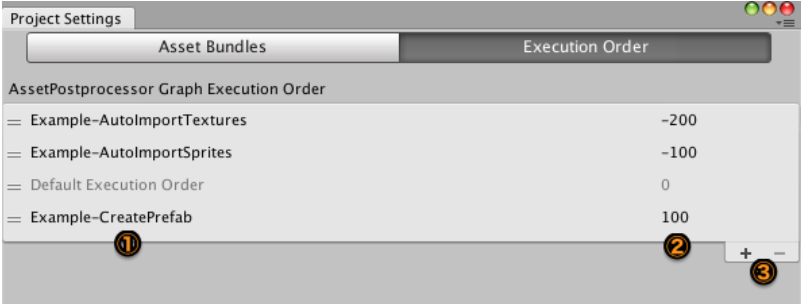
### **4 – 1. Asset Bundles 탭**



Asset Bundles 탭에서는 에셋 번들의 빌드에 관한 설정을 할 수 있습니다.

1. Build Asset Bundles 노드로 Build In Cache Directory를 선택할 때에 이용할 에셋번들의 캐시 전 폴더를 지정합니다.
2. AssetBundle Browser와 공유할 때에 이용할 에셋번들 빌드 용의 Graph입니다.

### **4 – 2. Execution Order 탭**



Execution Order 탭에서는, AssetPostprocessor와 이용할 Graph의 실행순서를 설정합니다.

1. Graph 명
2. Graph의 실행순서. 작은 값의 Graph일수록 먼저 실행됩니다. 특별히 지정하지 않은 Graph의 실행순서는 Default Execution Order의 값으로 사용됩니다.
3. 실행순서를 지정할 Graph를 추가, 삭제합니다.

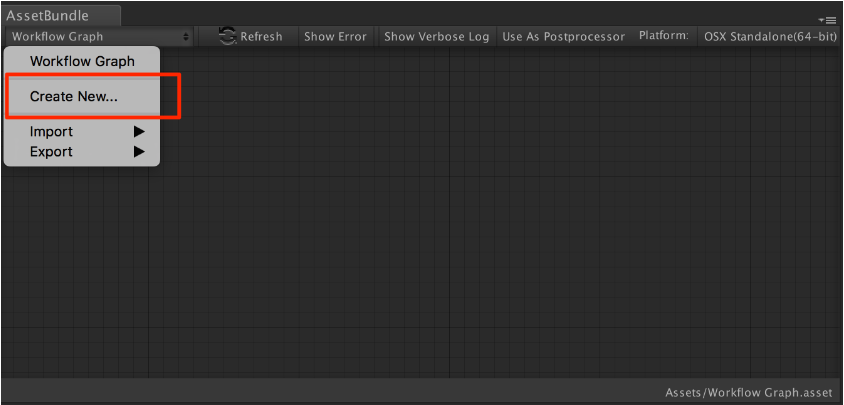
# **기본적인 사용법**

노드의 기본적인 용어와 기능에 대해서는 이 [링크](https://qiita.com/bonon0/items/f71a0c33766b67b9f891)를 참고해주세요.

AssetGraph는 전부 작업을 Graph 위에서 진행됩니다. Graph를 작성했으면, Graph 편집 윈도우로부터 노드를 추가할 수 있고, 노드를 연결해서 진행하는 것으로 에셋을 처리하는 흐름(파이프라인)으로 만들어 갈 수 있습니다. 에셋은 왼쪽부터 오른쪽으로 연결하여 처리됩니다. 각 각의 스테이지에서 분류랑 편집이 이루어지고 최종적으로 에셋번들에 포함해서 빌드를 하거나 특정 폴더에 복사하거나 합니다. AssetGraph의 작성된 노드랑 커스텀한 노드를 사용하여 이러한 흐름에 따라 많은 에셋에 일괄처리를 더해서 진행하는 것이 이 툴의 기본적인 사용법입니다.

# **Graph를 작성**

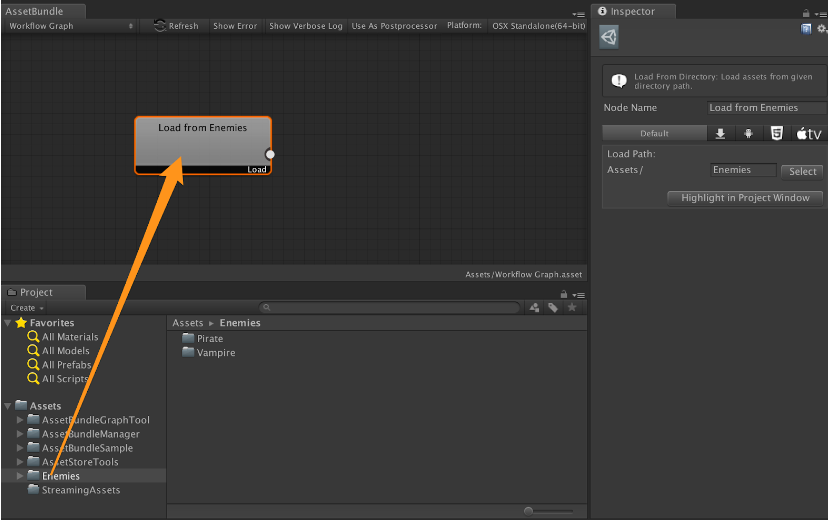
AssetGraph의 Graph를 작성하려면 Project View의 “Create” 메뉴, 혹은 오른쪽 클릭 메뉴로부터 “Asset Graph”를 선택합니다. 혹은 메뉴-Window-AssetGraph-Open Graph Editor에서 Asset Graph윈도우를 열고 “Create…”버튼을 누르면 만들 수 있습니다.



AssetGraph 에셋을 작성하고, 더블 클릭을 하면 Graph편집 윈도우가 활성화되고 해당 Graph를 편집을 할 수 있습니다. 새로운 노드는 윈도우에서 오른쪽 클릭을 한 다음, 만들고자 하는 노드를 선택을 하면 추가할 수 있습니다.

# **에셋 불러오기**

에셋에 특정 처리를 하기 위해서는 특정 작업과 연관된 에셋을 불러와야 합니다. 에셋을 불러오려면, Graph를 오른쪽 클릭한 다음 Load Assets밑의 메뉴를 선택하여 노드를 만들 수 있습니다. 여기서 [Load From Directory](#_Load_From_Directory)를 선택하면, 특정의 폴더로부터 에셋을 불러옵니다. Load From Directory노드는 Project윈도우로부터 폴더를 Graph 에디터에 드래그하는 것으로 만들 수 있습니다.



노드가 만들어지면 노드를 선택해서 Inspector를 엽니다. Load From Directory밑의 Sel ect 버튼을 누르면 임의의 디렉토리를 선택할 수 있으므로, 에셋이 존재하는 디렉토리를 자유롭게 선택할 수 있습니다.

이것으로 에셋이 불러와졌습니다. 한번 더 특정 노드를 만들어서 노드간의 연결하여 접속시키면 접속한 노드 간 불러올 에셋을 넘겨받을 수 있는 것을 확인할 수 있습니다. 이 연결을 클릭하면 Inspector에서 넘겨받을 수 있는 에셋의 일람을 볼 수 있습니다.

로드만을 위한 노드는 하나의 Graph의 가운데 몇 개 있어도 상관없습니다. 최종적으 로 에셋번들을 만들 때, 하나의 에셋이 복수의 에셋번들에 넣도록 지정했다면, 빌드 때 에러가 발생하므로 주의해주세요.

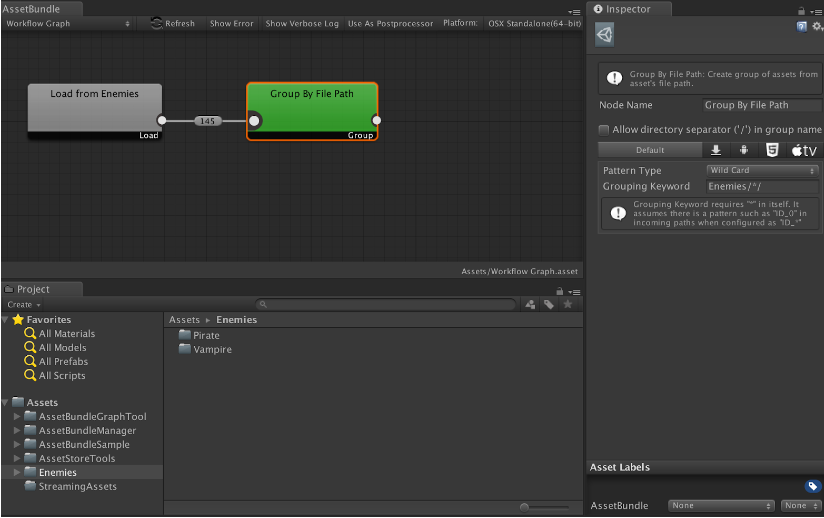
P.S 노드에서 연결되는 노드가 있고 연결이 되지 않는 노드가 있습니다. 연결한다고 해서 다 되는 것이 아닙니다.

## 대량의 에셋을 불러오는 것은 무거운 가요? AssetGraph에서 에셋의 로드

AssetGraph는 수 만개의 에셋에 대응할 수 있도록 만들어진 툴입니다. AssetGraph 에서 에셋의 로드란, 실제로는 에셋의 파일 경로라던지, 에셋의 타입 정보만을 읽어오는 것을 말합니다. 노드에 의해 에셋의 실 데이터를 사용하지만 그런 경우에는 사용되는 직전까지 읽지 않기에 또는 읽어 들인 실 데이터는 불필요해지면 즉시 해방하도록 만들어져 있습니다.

# **에셋을 그룹으로 묶기**

AssetGraph는 에셋을 그룹으로 정리하는 것이 가능합니다. 에셋번들에는 복수의 에셋 을 포함할려고 할 때나 에셋의 주체로부터 프리팹을 작성할 때 그릅화해서 사용하면 용이합니다. 에셋은 최초에 불러와졌을 때 “0”이라는 하나의 그룹에 속해집니다. 그룹에 속해지는 방식은 자유롭게 설정할 수 있습니다. 시험 삼아 Graph를 오른쪽 클릭을 한 다음 Group Assets 밑의 메뉴로부터 [Group By File Path](#_Group_By_File)를 선택해서 노드를 만들어 봅시다.

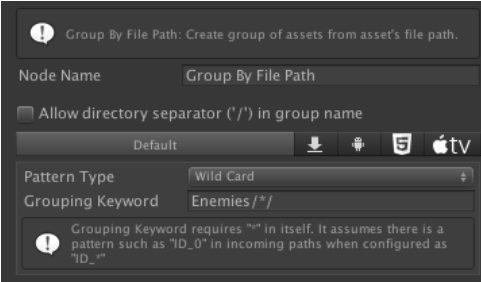


Group By File Path는 파일 경로명으로부터 그룹을 정리한 노드입니다. 예를 들면, 적 캐릭터의 에셋이 Assets/Enemies폴더 밑의 “Pirate”이라던지 “Vampire”라던지 이런 타입이 들어있을 경우, 각 각의 폴더 밑의 에셋을 “Pirate”, “Vampire”의 2개의 그룹으로 묶으려고 합니다.



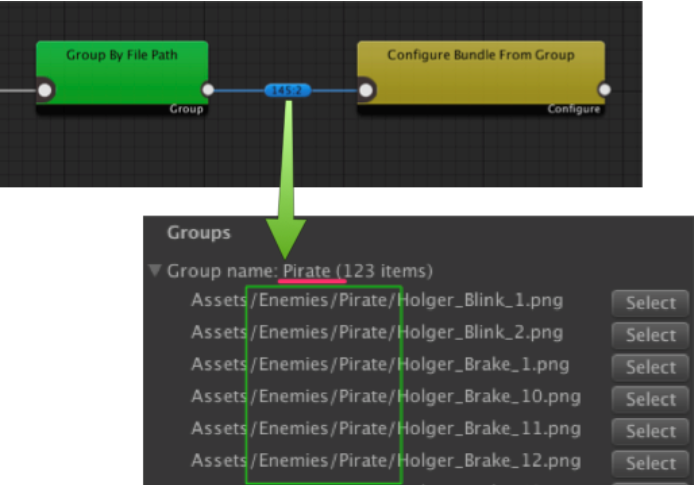
폴더마다 캐릭터의 에셋이 들어있으므로, 이것을 그룹으로 묶으려고 합니다.

Group By File Path는 파일 경로를 [패턴 매칭](https://ja.wikipedia.org/wiki/パターンマッチング)으로 그룹으로 묶습니다. 위와 같은 경우에는 [와일드 카드](http://terms.naver.com/entry.nhn?docId=161571&cid=50331&categoryId=50331)를 사용한 패턴 매칭이 편리합니다. Pattern Type을 “Wild Card”으로 설정하고, Grouping Keyword로는 “Enemies/\*/”라고 기재합니다.



폴더마다 캐릭터의 에셋이 들어있으므로, 이것을 그룹으로 묶으려고 합니다.

한 번 더 다른 노드를 만들어서 Group By File Path의 출력과 만든 노드의 입력과 연결을 합니다.(밑의 그림 참고) Inspector를 표시해서, Pirate폴더랑 Vampire폴더 밑의 에셋이 각 각 “Pirate”그룹이랑 “Vampire”그룹에 분할되어있는 것을 확인할 수 있습니다.

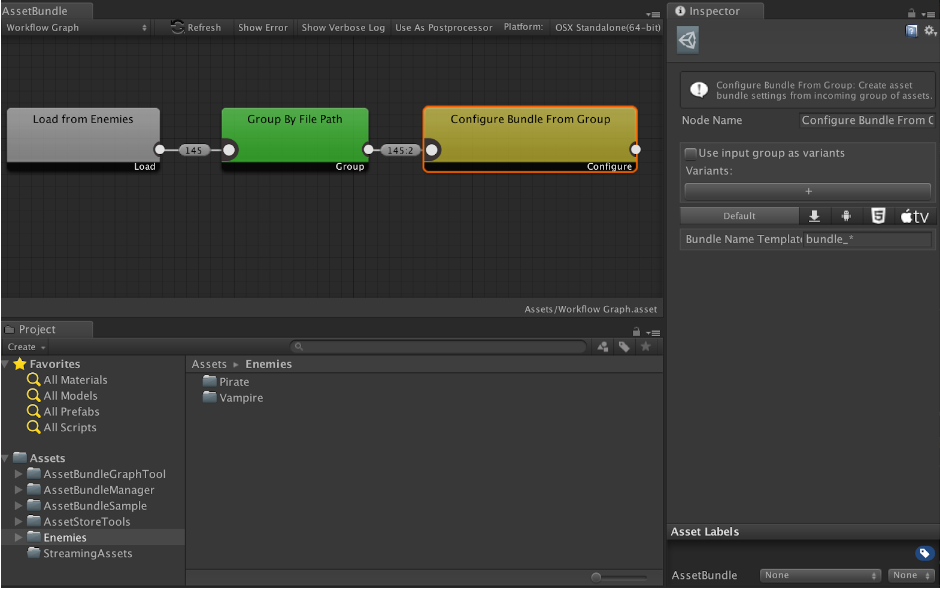


에셋의 파일 경로 안에 입력 패턴 “Enemies/\*/”와 매칭한 부분을 사용해서 그룹화가 됩니다.

# **에셋번들 설정을 구축하기**

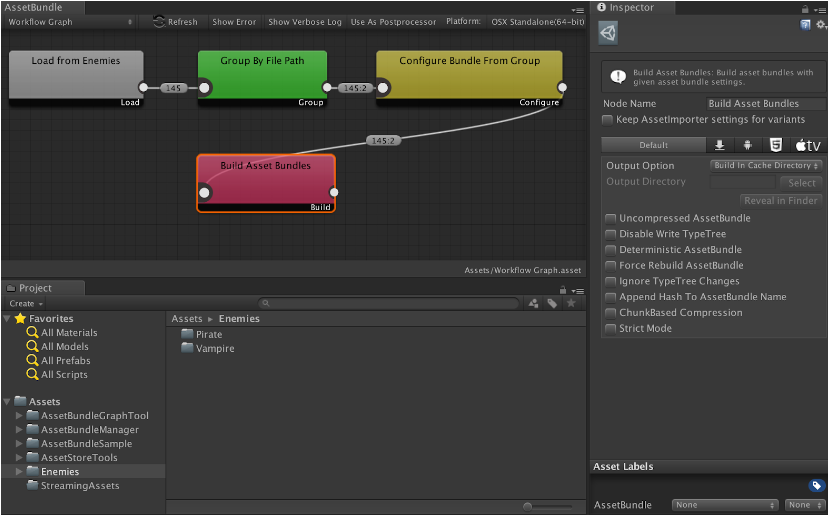
에셋을 그룹으로 묶었으면, 시험 삼아 이 그룹을 에셋번들로 만들어봅니다. 그룹을 에셋번들로 하려면, 에셋번들 설정을 구축합니다. 에셋번들 설정이란 그룹을 실제로 어떤 이름의 에셋번들로 할 것인지, Variant를 어떤 설정으로 할 지와 같은 것을 말합니다. 에셋번들을 실제로 빌드하기 전에 에셋번들 설정을 구축하기 위한 노드를 통해서, 그룹을 에셋번들 설정에 변환합니다.

에셋번들 설정을 구축할려면 Graph를 오른쪽 클릭을 한 다음 Configure Bundle 밑의 메뉴에서 [Configure Bundle From Group](#_Configure_Bundle_From)을 선택합니다. Configure Bundle From Group 노드가 만들어지면, Group By File Path의 출력을 이 노드와 연결하면 됩니다.



# **에셋번들 만들기**

무사히, 에셋번들의 설정을 완료했습니다. 위 설정에서 에셋번들을 실제로 빌드하기 위해서 준비를 갖추었습니다. 완료된 에셋번들의 설정에 따라 에셋번들을 빌드할려면, Graph를 오른쪽 클릭을 한 다음 Build/Build Asset Bundles를 선택하여, 새로운 노드를 작성합니다. 새롭게 Graph에 추가된 Build Asset Bundles 노드의 입력에 Configure Bundle From Group노드의 출력을 연결합니다. 압축방법이라던지, 에셋번들의 빌드 설정을 변경하고 싶은 경우, [Build Asset Bundles](#_Build_Asset_Bundles)노드를 선택해서 Inspector로부터 체크 포커스를 선택하면 됩니다. 이것으로 에셋번들을 빌드하는 준비를 끝났습니다.



# **그래프 빌드하기**

에셋을 불러오는 것으로부터 에셋번들의 빌드까지 대략적인 작업을 설정했으므로, 실제로 Graph로부터 빌드해서 동작을 살펴보겠습니다. Graph를 빌드할려면, Asset Bundle 윈도우의 툴바의 오른쪽 옆에 있는 “Build”버튼을 누릅니다.

프로그래스 바를 표시하는 다이얼 로그박스가 사라지면 빌드가 완료됩니다. Build Asset Bundles에 지정된 폴더를 열어서 보면, 실제로 파일이 출력되어 있는지 없는지 확인해봅니다. 캐쉬 폴더에 빌드하도록 설정했을 경우, Assets의 AssetGraph의 인스톨 폴더 밑의 Cache폴더에 빌드 된 에셋번들이 배치되어있습니다.

# **복수 그래프 빌드하기**

프로젝트가 진행되서 Graph가 몇 개정도 만들어지면, 복수의 그래프를 한 번에 빌드하고 싶어지는 경우가 생깁니다. 이런 경우 메뉴-> Window-> Asset Bundle Graph-> Open-> Batch Build Window를 선택해서, [Batch Build 윈도우](#_Batch_Build_윈도우)를 활성화합니다.

Batch Build 윈도우에서는, 프로젝트에 있는 복수의 Graph를 선택해서 임의의 그래프를 한 번에 빌드할 수 있도록 해줍니다. Batch Build 윈도우의 상세한 사용법은 [인터페이스 설명](#_Batch_Build_윈도우)을 참고해주세요.

# **조금 응용해서 사용하기**

기본적인 사용법의 섹션에서는 에셋을 읽어와서 에셋번들을 만들어서 특정 폴더에 복사하는 것으로 배웠습니다. 이 섹션에서는 Graph 툴에서 가능한 범위에서 응용하는 방법을 배웁니다.

## 에셋을 분할한다.

폴더랑 라벨로 에셋들을 불러왔을 때, 넘겨지는 에셋의 타입이 보통 하나라고 볼 수는 없습니다. 에셋을 이름이랑 타입이랑 구분해서 다음 노드에 넘겨주려고 할 때, Split Assets/Split By Filter를 사용하는 것으로 가능합니다. [Split By Filter](#_Split_By_Filter)는 이름과 타입을 사용해서 에셋을 분할할 수 있지만, 분할 방법은 스스로 Filter를 추가하는 것으로 늘릴 수 있습니다.(고도 토픽 : [커스텀 필터를 추가하기](#_커스텀_필드를_추가하기)를 참조) 넘겨진 에셋이 복수의 필터 조건에 일치할 경우 최초 일치하는 조건의 아웃풋에 에셋을 출력합니다. 즉, Split By Filter의 조건의 순번에는 강한 의미가 있으므로 주의가 필요합니다.(이에 대한 상세한 정보는 이 [링크](https://www.evernote.com/shard/s722/sh/46f4becf-ac02-4968-bb54-ce448eb2b697/f292e8d812e8a6dd)를 참고해주세요.)

## 에셋을 작성하기

AssetGraph는 에셋을 새롭게 작성하기 위해서도 이용할 수 있습니다. 예를 들면, 아티스트가 추가한 텍스처랑 모델로부터, 자동적으로 실제의 게임에서 사용되는 Prefab을 작성하고 싶다거나 하는 경우가 있을지도 모릅니다. Graph를 오른쪽 클릭을 한 다음 Create Assets/Create Prefab From Group을 선택하면, 그룹으로부터 Prefab을 작성할 노드를 만들 수 있습니다. [Create Prefab From Group](#_Create_Prefab_From)에서 Prefab을 작성하려면 PrefabBuilder 스크립트를 새롭게 작성해서 프로젝트에 추가할 필요가 있습니다. PrefabBuilder를 만드는 방법은 고도 토픽의 [Prefab을 자동적으로 작성하기](#_Prefab을_자동적으로_작성하기) 를 참조해주세요. Create Prefab From Group에서 Prefab작성에 사용할 그룹을 넘겨주면 자동적으로 Prefab을 작성해서 출력에 추가해줍니다.

## 에셋을 편집하기

에셋을 에셋번들에 추가할 때, 혹은 단지 패치 처리의 일환으로써, 에셋을 일괄 편집을 해야할 때가 있을 지도 모릅니다. AssetGraph를 사용하는 것으로, 임포트 설정을 일괄로 변경하거나, Importer를 경유하지 않는 에셋(Material이나 RenderTexture 등)을 직접 변경하거나 할 수 있습니다. 텍스처랑 모델의 임포트 설정을 변경할 경우, Graph를 왼쪽 클릭을 해서 Modify Assets/Overwrite Import Setting을 선택해주세요.

[Overwrite Import Setting](#_Overwrite_Import_Setting)노드는 텍스처나, 오디오 혹은 비디오의 임포트 설정을 변경할 수 있습니다. 실제 사용법은 빌드 인 노드->Overwrite Import Setting를 참조해주세요. 작성된 노드에 에셋을 넘겨주면, 빌드 때 에셋의 설정이 변경됩니다.

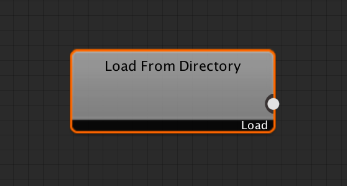
아직 임포타가 없는 에셋을 직접 편집 할 경우에는 [Modify Assets Directly](#_Modify_Asset_Directly)노드를 사용합니다. 실제 사용법은 빌드 인 노드의 Modify Assets Directly를 참조해주세요.

## 에셋번들 설정을 체크하기

에셋번들을 작성할 때, 실수로 원하지 않은 에셋이 에셋번들에 들어갈 수도 있습니다. 에셋번들에 원하지 않는 데이터가 들어가지 않았는지 체크를 하려면 Assertion노드를 사용하면 됩니다. 빌드 인 노드의 [Assert Unwanted Assets In Bundle](#_Assert_Unwanted_Assets)를 사용하면 특정 폴더 밑의 에셋만 포함되도록 허가하거나 특정 폴더 밑의 에셋이 포함되도록 금지하거나 할 수 있습니다. 원하지 않는 에셋이 검출되면 노드가 에러를 보고하므로 빌드 전에 제시해서 구체적으로 어떤 파일이 포함되었는지 알 수 있습니다. 상세한 사용법은 빌드 인 노드의 Assert Unwanted Assets In Bundle을 참고해주세요.

# **빌드 인 노드**

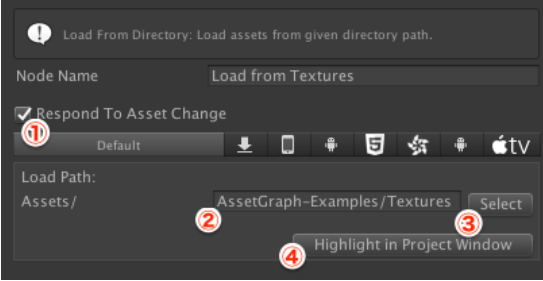
## Load From Directory



**개요**

Load From Directory는 지정 폴더 밑의 에셋을 불러옵니다.

**속성**



1. AssetPostprocessor로써 이용할 시, Load할 에셋의 변경에 응답할지 말지 결정합니다. 이 체크가 되어있을 경우 이 Loader에 의해 불러올 에셋은 AssetPostprocessor의 이벤트를 호출합니다.
2. 폴더 경로 지정
3. 폴더 선택 다이얼로그 팝업 창을 활성화
4. Project View에 지정한 폴더를 하이라이트 한다.

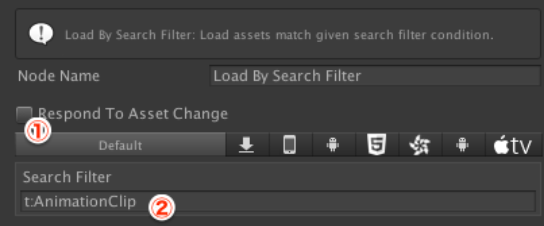
## Load By Search Filter



**개요**

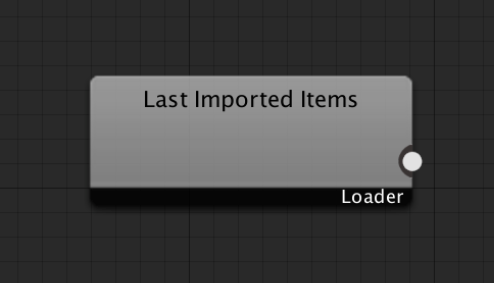
Load By Search Filter는 지정한 검색 필터를 사용해서 에셋을 불러옵니다. 검색 필터의 형식은 Project윈도우의 검색형식과 같습니다. 상세한 것은 [Project윈도우의 문서](#_프로젝트_윈도우)를 참조해주세요.

**속성**



1. AssetPostprocessor로써 이용할 시, Load할 에셋의 변경에 응답할지 말지 결정합니다. 이 체크가 되어있을 경우 이 Loader에 의해 불러올 에셋은 AssetPostprocessor의 이벤트를 호출합니다.
2. 검색 필터

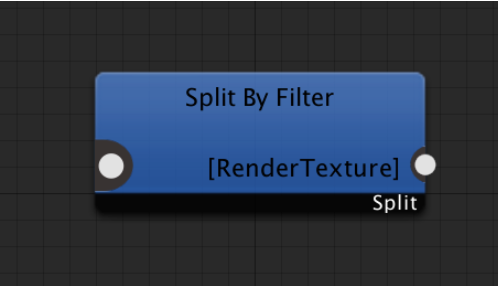
## Last Imported Items



**개요**

Last Imported Items는 마지막으로 임포트된 에셋을 불러옵니다. 이 노드는 Graph를 Postprocessor로써 사용될 때만 준비되어있습니다.

## Split By Filter

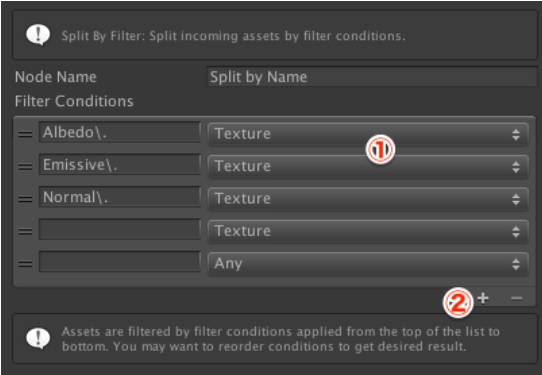


**개요**

Split By Filter는 입력된 에셋을 재 조건에 의해 분배합니다. 디폴트에 분배 조건으로 사용될 수 있는 것은 파일 경로와 에셋 타입입니다. 커스텀 필터를 프로젝트에 추가하는 것으로, Split By Filter의 분배방법을 확장할 수 있습니다.

Split By Filter의 에셋의 분배는 위의 조건으로부터 순서대로 진행됩니다. 혹시 어떤 에셋이 복수의 필터 조건에 매칭할 경우 그 에셋은 필터 조건의 처음 만나는 조건에 해당되고 그 후의 조건에는 포함되지 않습니다. 추가한 필터는 리스트의 왼쪽 옆을 드래그하는 것으로 순서를 교체할 수 있습니다.

**속성**



1. 필터 조건. 파일 패스의 지정에는 정규표현을 사용할 수 있습니다.
2. 새로운 필터 조건을 추가합니다. 커스텀 필터가 설치되어있다면, +버튼을 눌렀을 때 추가할 필터를 선택하기 위한 드롭 다운 메뉴가 표시됩니다.

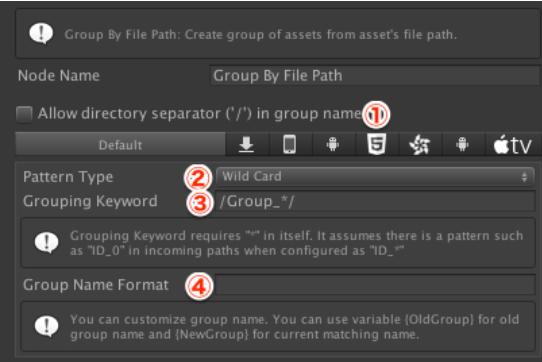
## Group By File Path



개요

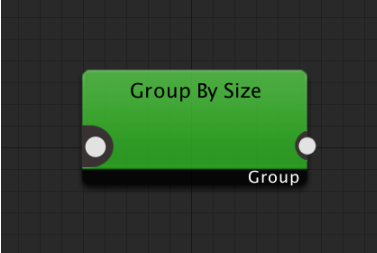
파일 경로를 사용해서 그룹을 작성합니다.

속성



1. 그룹 명에 폴더 구분자(‘/’)를 포함할지 말지를 체크합니다.
2. 그룹핑 키워드, **Grouping Keyword**의 포맷을 지정합니다. Wild Card의 경우, 파일 경로의 일부가 \*과 매칭할 때, 이 부분이 그룹 명으로 됩니다. Regular Exporession을 사용할 경우, 그룹핑 키워드는 정규표현으로 ()을 사용해 그룹화할 부분을 그룹 명으로 됩니다.
3. 파일 경로로부터 그룹 명을 취득하기 위해 사용할 매칭 조건을 지정합니다. 매칭 방법은 Pattern Type로 지정합니다.
4. 그룹 명의 포맷을 지정합니다. 이 노드에 의해 Grouping Keyword에 매칭하는 키워드는{NewGroup}, 이 노드에 건내질 때 사용되고 있던 그룹명은 {OldGroup}로 지정할 수 있습니다.

## Group By Size



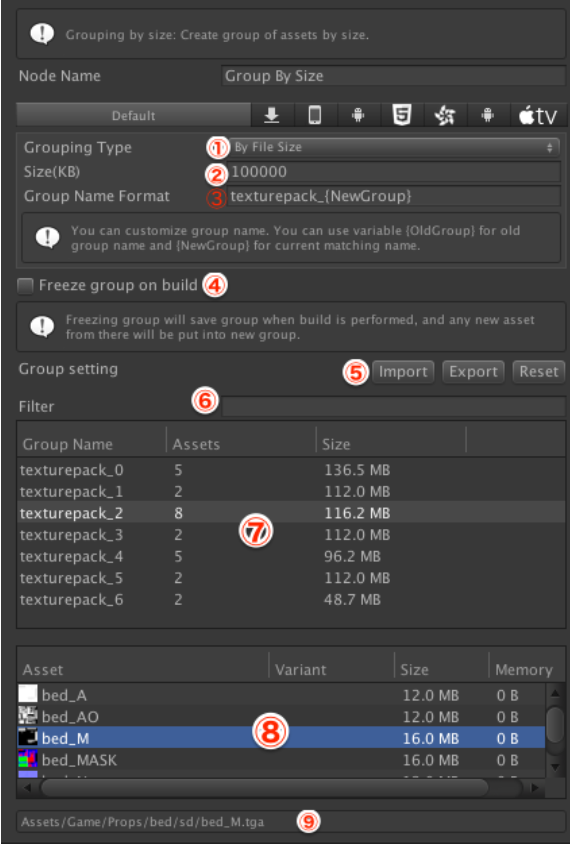
**개요**

Group By Size는 입력되는 에셋을 크기로 그룹화합니다. 예를 들면, **[파일 크기를 1MB** **씩]** 와 같은 설정을 할 경우, 에셋의 파일크기를 체크하면서 그룹에 순서대로 추가해서, 합계가 1MB에 도달하면 다음 에셋을 새롭게 그룹에 추가합니다. 단일 에셋이 지정 크기보다 클 경우 해당 파일은 단독으로 하나의 그룹으로 됩니다. 이 노도는 그룹의 크기를 반드시 지정 크기 이하로 담아두기 위한 것이 아닙니다.

**그룹핑의 유지**

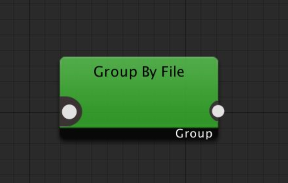
Group By Size의 그룹핑은 크기에 의해 진행되므로, 해당 크기에 따라 에셋이 속할 그룹을 결정하게 됩니다. 이것은 AssetBundle의 크기를 고려하여 이 노드를 사용하려고 할 때, 더 많은 에셋번들 파일의 갱신이 필요하다는 점에서 운용면에서 좋지 않은 효과를 초래합니다.(CND, DLC와 같이 파일을 하나의 크기로 묶어서 다운로드를 받는 것이 아니라 파일이 크면 해당 파일을 분리해서 다운로드 받는 것이 더 좋다.) 이러한 문제를 해소하기 위해 Group By Size에는 빌드 시 그룹 설정을 기억해둔 다음, 이것을 유지하기 위해 기능을 갖추고 있습니다.

**속성**



1. 그룹으로 사용할 방법을 지정합니다. By File Size의 경우 에셋의 파일 크기를 입력하고 By Runtime Memory Size이면 로드 후에 메모리 사용량을 봐서 그룹을 처리합니다. By Runtime Memory Size는 모든 에셋의 대해 사용되는 것이 아니므로 주의해주세요.
2. 그룹으로 사용할 크기의 정도(KB 단위)
3. 그룹 명의 포맷을 지정합니다. 이 노드에 의한 Grouping Keyword에 매칭되는 키워드는 {NewGroup}, 이 노드에 건내질 때 사용되는 그룹 명은 {OldGroup}으로 지정할 수 있습니다. 아무것도 입력하지 않는 경우, {NewGroup}과 같은 값이 지정됩니다.(이 부분은 테스트를 해보면서 재해석을 해야할 듯. 7번과 연관성이 있을 듯)
4. 빌드 시 그룹 설정을 기억할 것인지 아닌지를 지정합니다.
5. 현재 그룹을 json형식으로 [임포트/익스포트]합니다. Reset 버튼을 지정하면, 현재 기억하고 있는 그룹설정을 초기화합니다.
6. 7번의 표시된 에셋 목록들 중에서 특정 에셋만 표시하기 위한 필터입니다. 그룹 자체에 영향은 없습니다.
7. 현재 그룹 구성에서 보여지는 그룹 명, 에셋의 수, 추정되는 파일 크기를 표시합니다.
8. 7에서 선택된 그룹에 포함하는 에셋의 목록을 표시합니다.
9. 8에서 선택된 에셋의 프로젝트 내 패스를 표시합니다.

## Group By File



개요

Group By File은 입력된 에셋을 하나씩 별도로 그룹에 추가합니다. 예를 들면 Group By File에 3개의 에셋을 끌어 놓으면, Group By File은 Group Name Fomat의 지정에

## Create Prefab From Group

## Overwrite Import Setting

## Generate Asset

## Modify Asset Directly

## Label Assets

## Configure Bundle From Group

## Extract Shared Assets

## Assert Unwanted Assets In Bundle

## Error

## Build Asset Bundles

## Export To Directory

## Mirror Directory

# **고도 토픽**

## AssetGraph에 의한 AssetBundle의 설정

## 커스텀 노드를 추가하기

## 커스텀 필터를 추가하기

## 커스텀 에셋에 대응하기

## 에셋을 자신의 방법으로 편집하기.

## Prefab을 자동적으로 작성하기

## PrefabBuilder 스크립트 변경 시 Prefab을 재 작성하기

## 파생 에셋을 자동적으로 작성하기

## 빌드 후 사후처리를 진행하기

## 스크립트로부터 Graph를 빌드하기

## 코맨도 라인 툴을 사용해서 실행

## AssetBundle Browser와 합쳐서 사용하기