Unity Package Manager (UPM) 을 이용한 CustomPackage File 구성하기.

- 어떤 과정으로 진행하는 것이 좋을까?

1. Package Manager 에 대해 좀 더 이해하기. : Unity의 패키지 관리자 - Unity 매뉴얼 (unity3d.com)
2. 커스텀 패키지 생성 방법 알기
   - 커스텀 패키지 생성 - Unity 매뉴얼 (unity3d.com)
   - 공식은 좀 많이 복잡하고, 다양한 내용이 서술되어 있을 수 있음. 따라서 좀 더 쉽게 서술된 내용을 찾아 봄.
     - [Unity] Unity Custom Package (유니티 커스텀 패키지) (tistory.com)
     - Unity Custom Package 만들기 : 네이버 블로그 (naver.com)
     - 이 두 가지가 그래도 쉽게 서술되어 있는 듯.
     - (3) Creating Custom Packages in Unity! (Tutorial) - YouTube

---

Unity 의 UPM 관련 공식 튜토리얼

(3) Creating Custom Packages in Unity! (Tutorial) - YouTube

UPM 이란 뭘까?

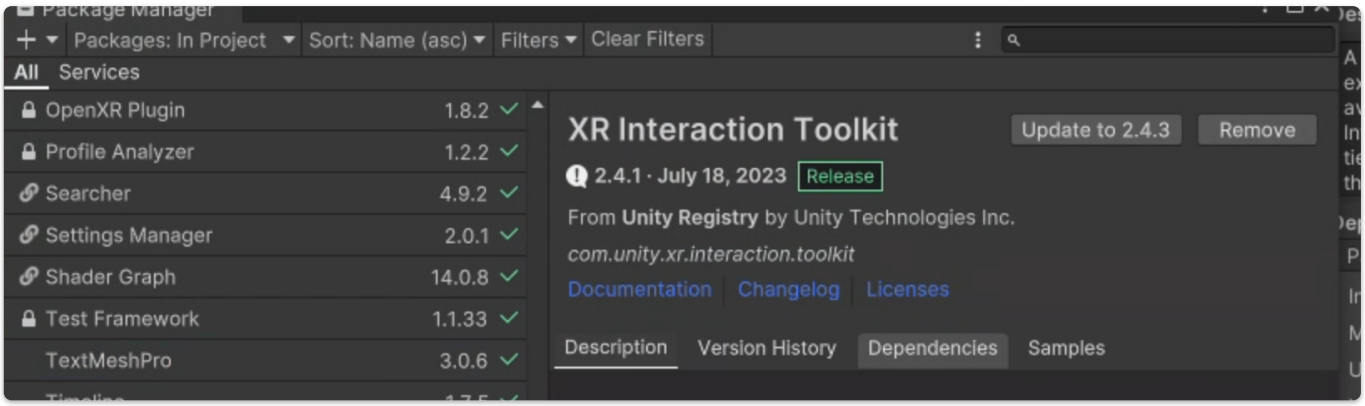
1. Unity 의 공식 Package Management System 이다.
2. Package Manager 란? : A modular and dynamic software and asset delivery system 이다.
   - 모듈화 (분산 및 독립) 적이고, 동적인 소프트웨어와 에셋 전달 시스템.
3. package dependency 를 찾아 동적으로 다운로드 받아줄 수 있도록 만들어 줄 수도 있다.
4. UPM Package는 특정 Layout 을 지키는 폴더이기도 하다. Unity Package Standard / Version Labling Format 등을 지키는 것이 중요하다.
5. Asset 을 유지하기 위한 모든 기능을 해당 Package Folder 에 모아두어야 한다.

실전 - 제작

실제로 Package 를 한번 만들어 볼 수 있도록 하자.

1. Asset / Packages Folder 로 구분이 되어 있을 텐데, 이를 파일 탐색기로 열어서, Folder 를 하나 만들자.

2. Sub Folder 로 package.json file 을 하나 만들어서 붙여넣자.
    1. 이 package.json 파일은 우리가 위의 링크 및 모든 기타 자료에서 봤던 것과 같은 레이아웃을 지니고 있어야 한다.
    2. 보통 복사 하나 해서, 붙여 넣고, Unity 내부의 Inspector 에서 해당 package file을 수정할 수 있으므로, 유니티 내에서 수정하는 것이 좋다.
    3. https://docs.unity3d.com/Manual/upm-manifestPkg.html : 처음 package.json 파일을 사용하고자 하는 경우, 이 링크에 있는 내용을 사용할 수 있도록 하자.
3. 이후, 이 만들어둔 package 안에 우리가 넣고 싶은 파일들을 모두 집어넣으면 된다.
    1. 제시하고 있는 folder 구조 - package convention을 지키는 것이 권장된다.



- Dependencies 내용 채우고 싶을 때는, 직접 UPM 살펴보면 됨. 여기에, 해당 Package 의 실제 식별 이름 (com.?.?) 다 나와 있음.
- AssetStore 내에서 다운로드 후 Import 한 에셋의 경우. 사실 이건 Package Manager 에서 Package 처럼 보이고 있긴 하지만, package 에 저장되지는 않음. package 에 저장되는 것은 Unity 에서 개발한, Unity 에서 어떤 버전의 Unity 에서도 동작한다는 것을 인가를 받은 기능들.
    - 따라서, 해당 추가 내용들은, 에셋으로서 추가되는 친구들이므로, 폴더 내에 추가 첨부해두면 될 것으로 보이긴 한다.
    - 혹은, 자동으로 결국 해당 에셋을 최신화 시켜주는 것이 아니므로, 이를
- dependencies 부분에서 비워두면 자동으로 최신 다운로드 받게 될 줄 알았는데, 그냥 오류 뜨네.

> *manipest 파일이란 말이 계속 나오던데, manipest 파일이란 무엇일까? :*
>
> - *소프트웨어 패키지나 애플리케이션의 메타데이터를 담고 있는 설정 파일.*
> - *즉, 여기선 **package 에 대한 설정을 담고 있는 package.json 파일**이 manipest 파일이라고 생각하면 된다.*
> - *Unity - Manual: Package manifest (unity3d.com) : 여기서 package.json (manipest) 파일에 대해 담겨 있으므로, 궁금하면 더 자세히 살펴보면 된다.*

4. CustomPackage 가 Script 를 가질 경우, 관련되는 Script 에 대해 AssemblyDefinition 을 이용하여 연결시켜 두어야 한다.
    1. Assembly Definition 이 뭐냐? : AssemblyDefinition 파일은 .NET 시스템에서 C# 프로젝트에 해당하는 유니티 프로젝트입니다. => 이에 대해선 더 자세히 알아볼 필요 있을 듯. => Unity. Assembly Definition 참조.
5. Editor Specific / Runtime Specific 에 따라 각각 하나의 AsmDef 파일이 필요하다.
    1. asmdef 파일은 Unity Create 로 쉽게 생성하는 것이 가능함.
    2. naming convention : companyName.FeatureName(package name).FolderName(editor / runtime ...)
6. 만약 해당 Package 를 배포할 생각이라면, liesence.md 파일과 thirdpartynotices.md 파일을 제작할 것을 권유 함.
    1. Unity - Manual: Meeting legal requirements (unity3d.com) : 이 내용을 참고할 수 있도록.

실전 - 공유

1. Zip을 하여 공유한 후, 다른 사람이 package.json 파일을 통해 바로 추가하는 것이 가장 쉬움.
2. git repository 를 통한 공유가 가능함. 단, 이 경우 누구나 git url 이 있으면 사용할 수 있음.
    1. 이 방법에 대하여 좀 더 집중적으로 살펴보자.
    2. Github을 이용할 것.
    3. 새로운 repository 생성.
    4. repository 내에, packages folder 내에 들어가있던 세부적인 파일들을 모두 업로드 함. (이때, meta 파일도 포함되어 있어야 함.) (어떤 경로부터 시작인지 모호하게 느껴질 수도 있는데, package.json 파일이 존재하는 곳을 root 라고 생각하면 됨.)
    5. 이후 github에서 제공하는 url 을 바탕으로 UPM에서 git url 을 바탕으로 추가할 수 있음.
        1. 이때는, git 에 관한 내용이 local machine 상에 존재하여야 함.