NHN FORWARD

Unity Optimization
: Asset Dependency-Driven
: 에셋 의존성 주도 최적화

© 2021 NHN FORWARD. All rights reserved.

다룰 내용

1. 프로젝트 최적화
2. 성능 최적화
3. 최적화를 위한 개발
4. Assetmanagement의 활용
5. 정리

에셋 의존성 주도: 프로젝트 최적화

© 2021 NHN FORWARD. All rights reserved.

프로젝트 최적화

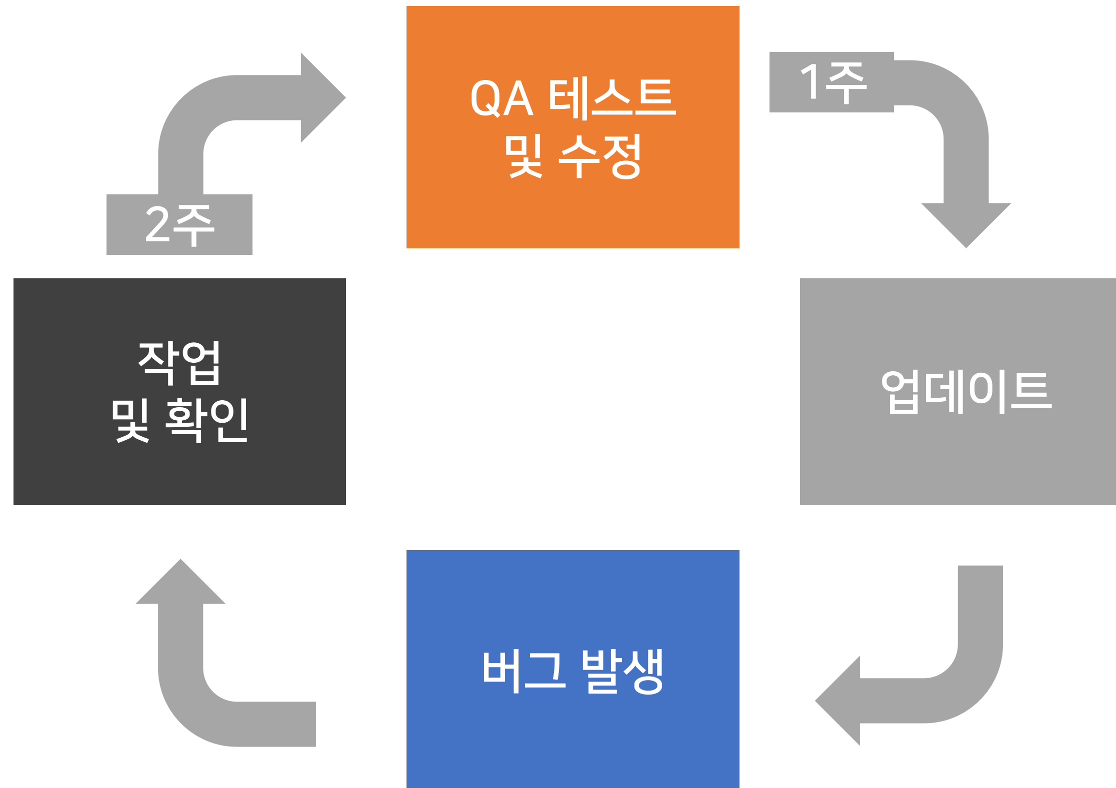
이 이야기는 실화를 바탕으로 제작되었습니다

프로젝트 최적화

단순한 작업이 고통이 되다

프로젝트 최적화

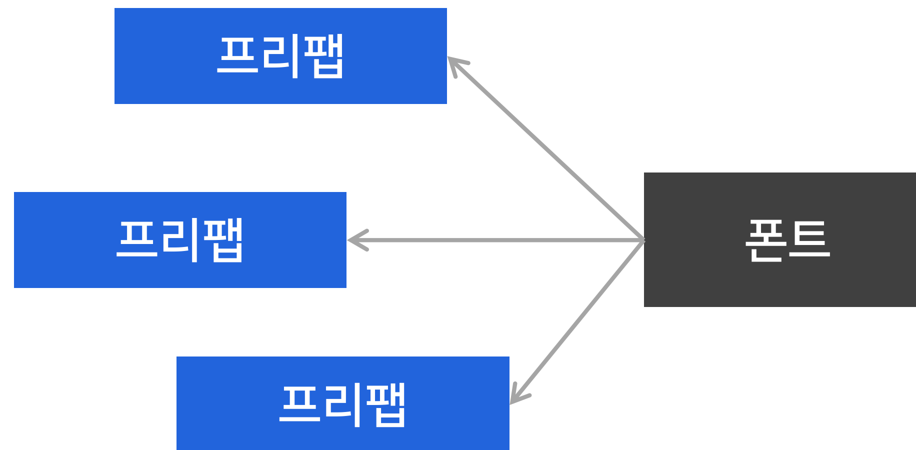
악순환이 반복되다



작업 속도도 오래 걸리고 재작업이 많이 나오는 악순환 발생

프로젝트 최적화

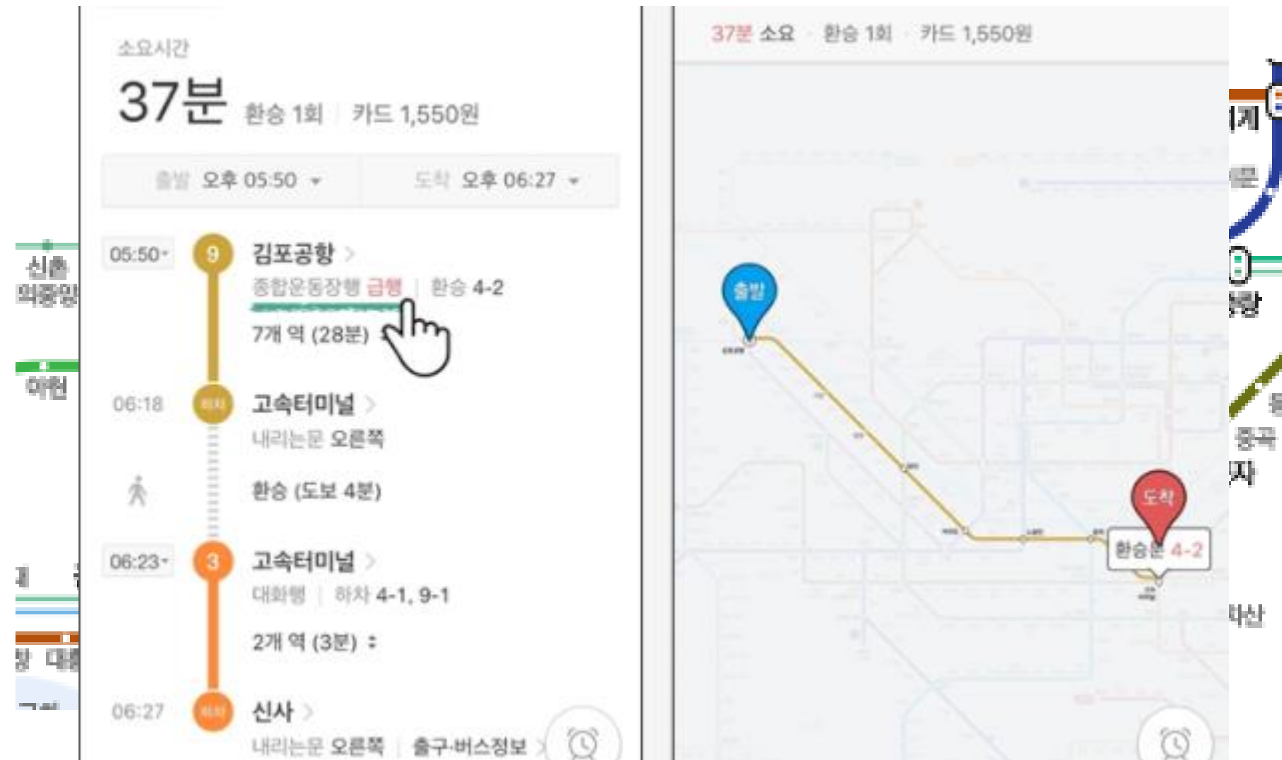
에셋 탐색을 지원하다



이러한 의존성은 참조(reference)라고 불린다

프로젝트 최적화

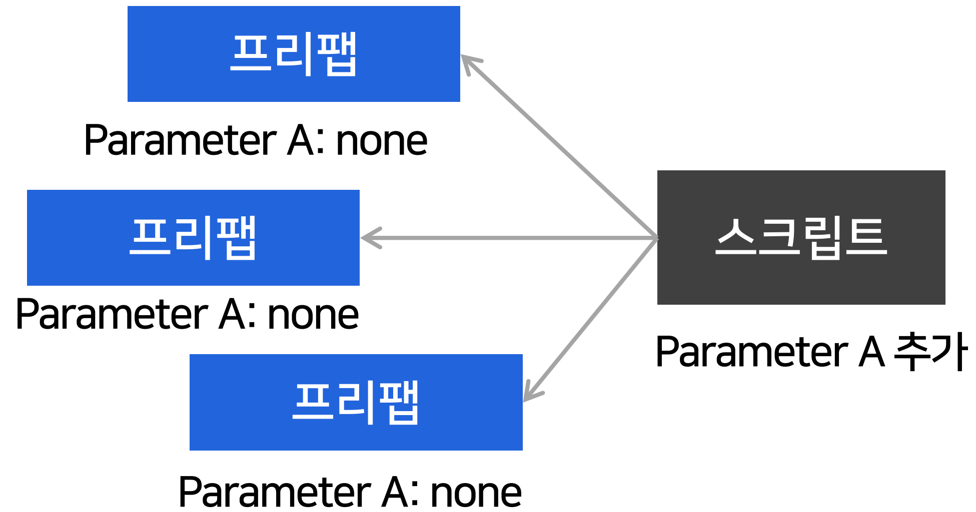
작업 단순화



출처 : 네이버 지도

프로젝트 최적화

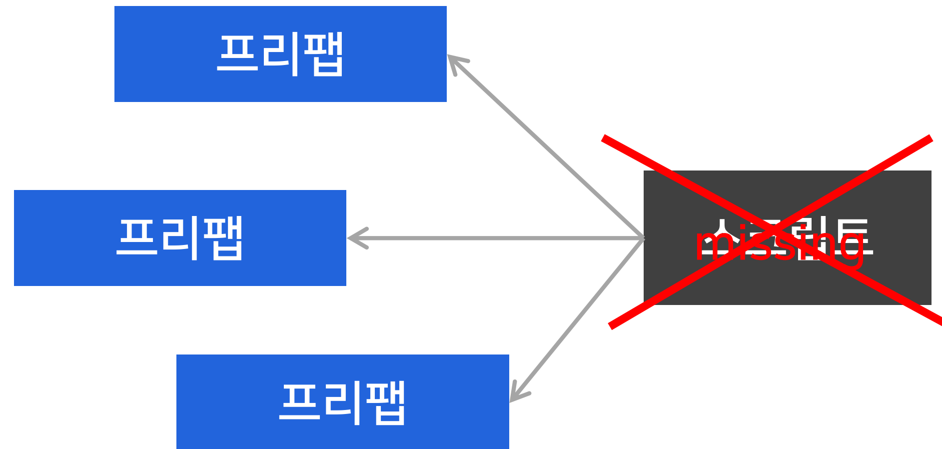
안정적인 작업



영향받는 에셋의 확인이 필요하다

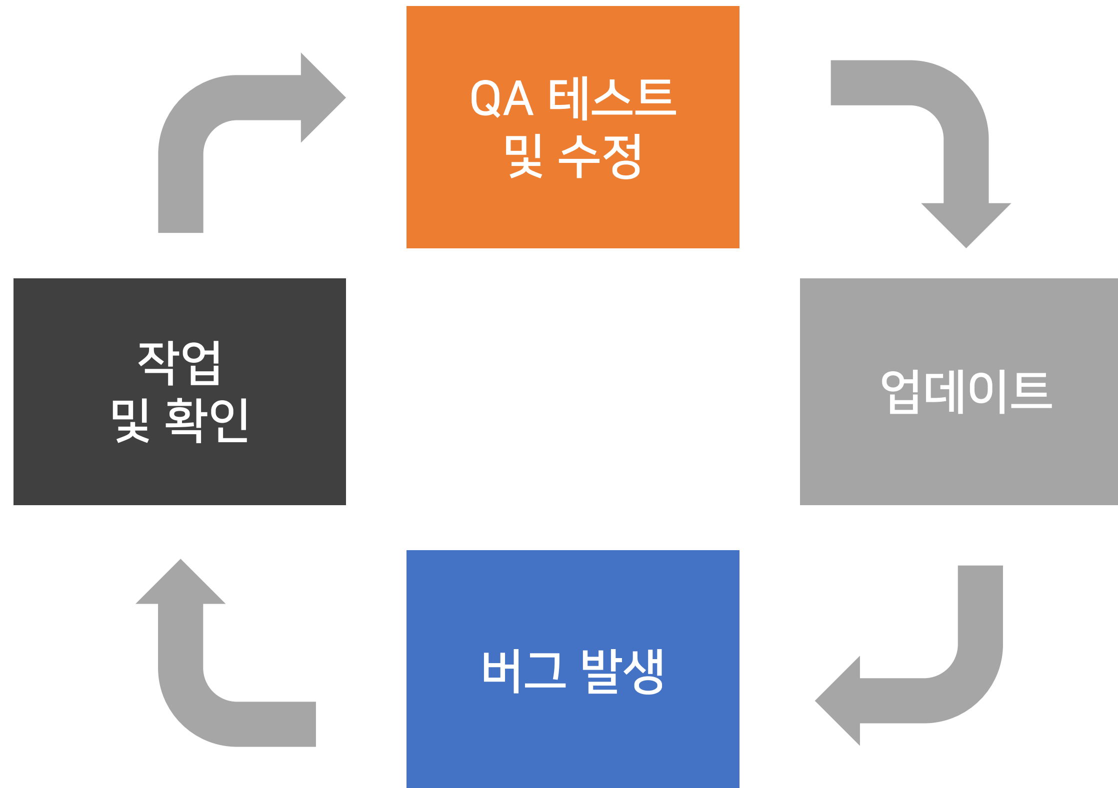
프로젝트 최적화

문제 예방



프로젝트 최적화

안정성은 생산성과 직결



문제가 적어지는 만큼 선순환 발생

에셋 의존성 주도: 성능 최적화

© 2021 NHN FORWARD. All rights reserved.

성능 최적화

에셋의 구조 최적화

프레임 성능 최적화

메모리, 용량 최적화

성능 최적화

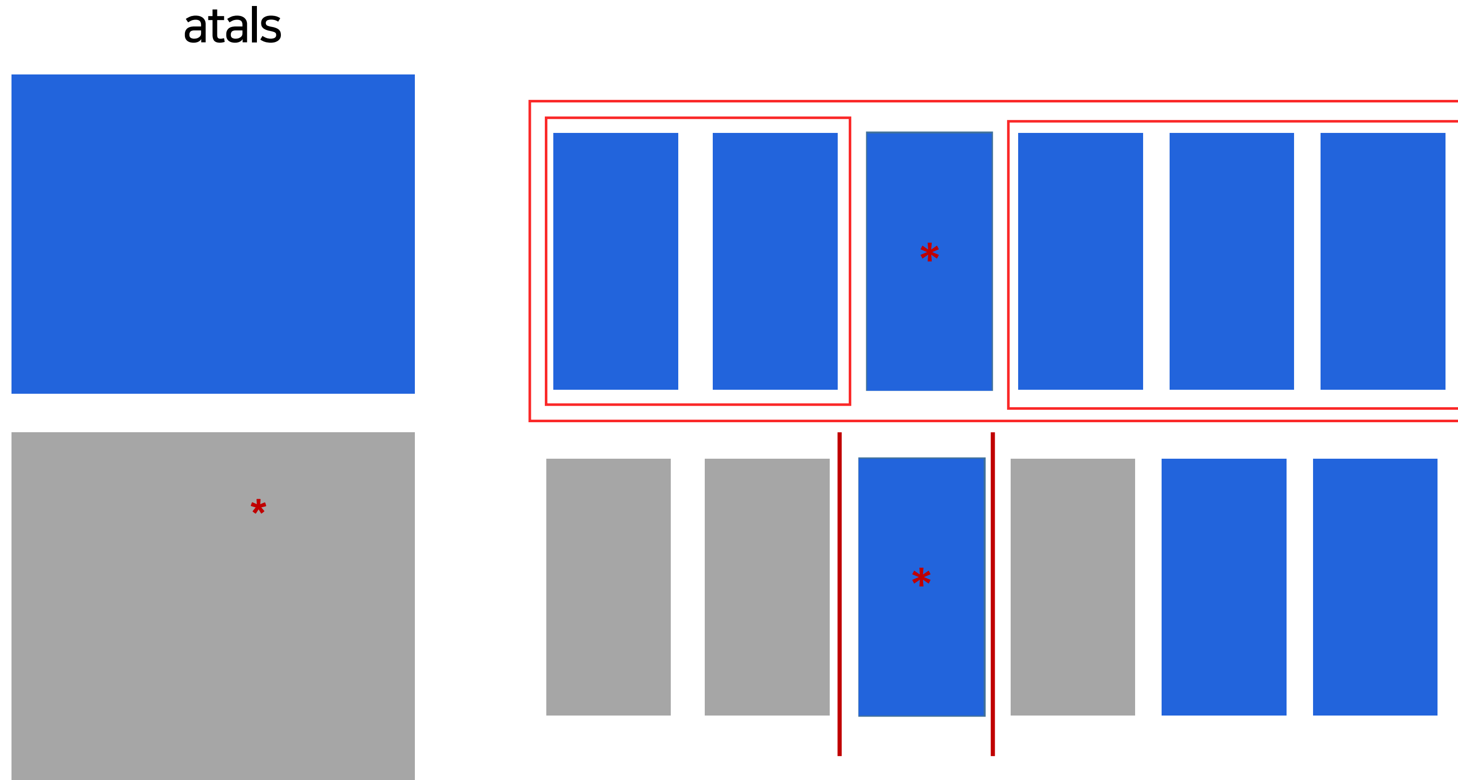
일괄 처리(batching)

여러 개 따로 처리

묶어서 한 번에 처리

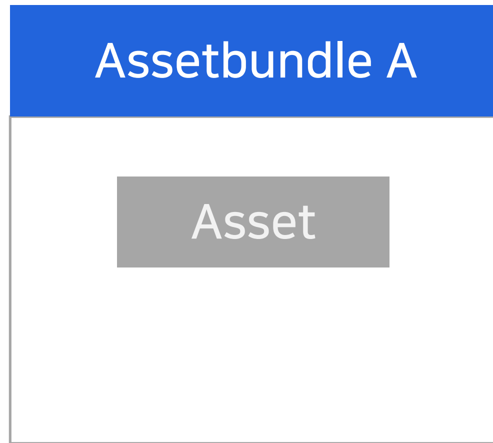
성능 최적화

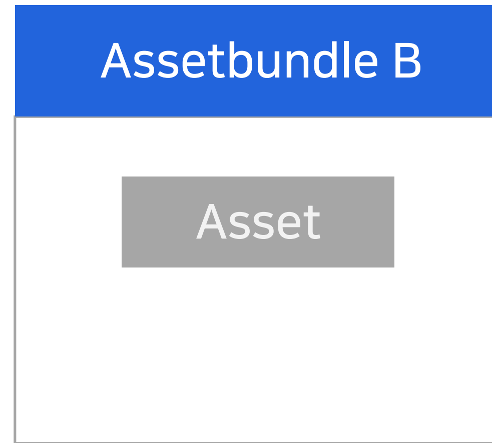
일괄 처리(batching)



성능 최적화

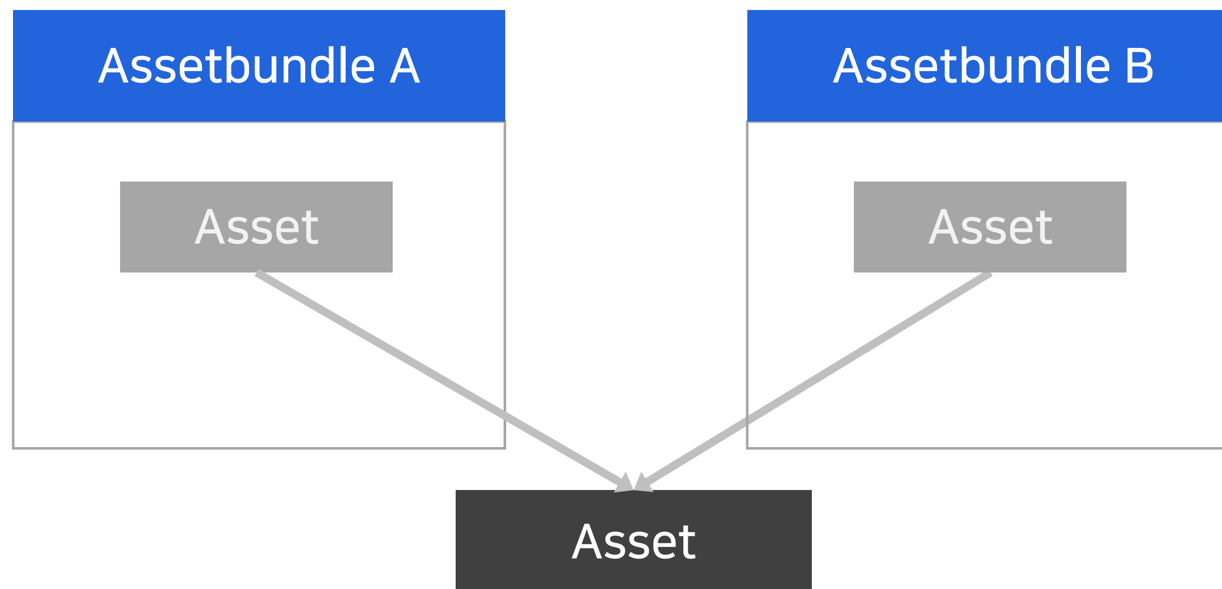
중복 제거





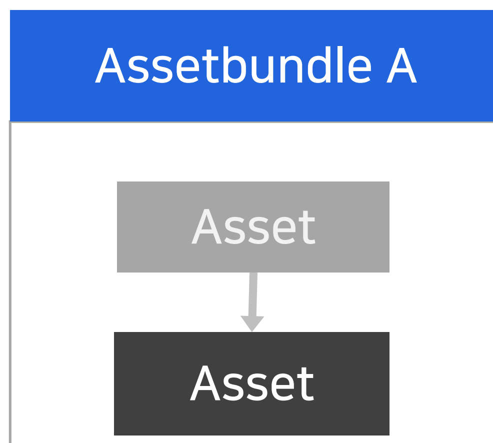
성능 최적화

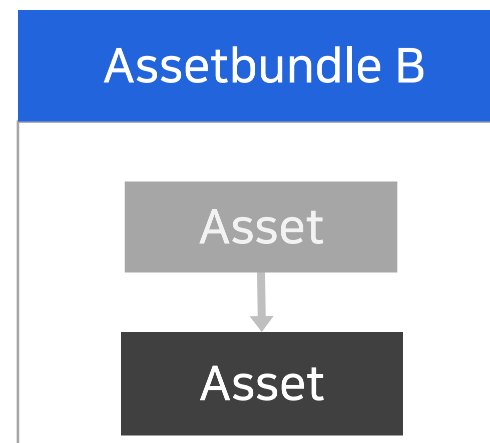
중복 제거



성능 최적화

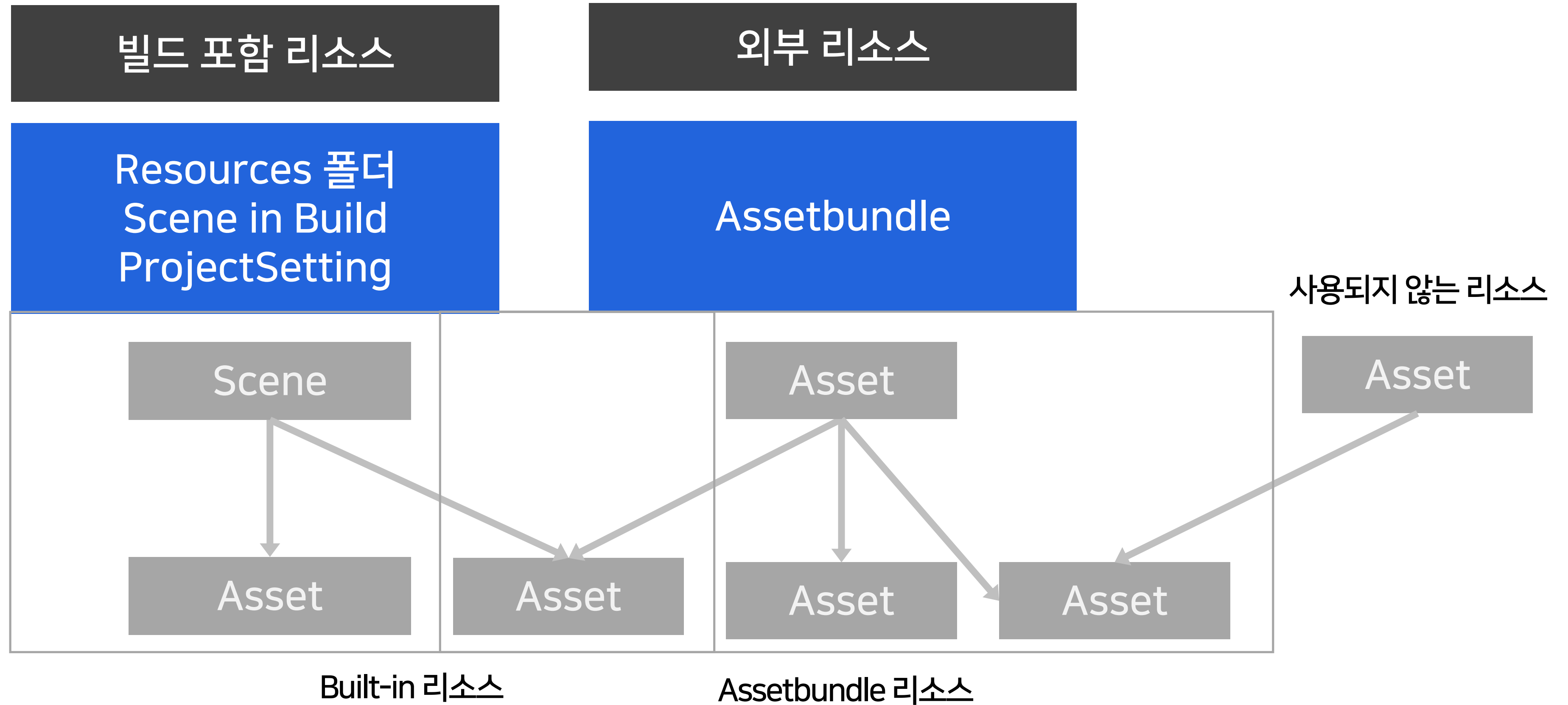
중복 제거





성능 최적화

중복 제거



성능 최적화

최적화가 어려운 이유

- 구조를 알기 쉽지 않다
- 수정된 것을 확인하려면 시간이 걸린다

최적화를 위한 개발

최적화를 위해 필요한 것

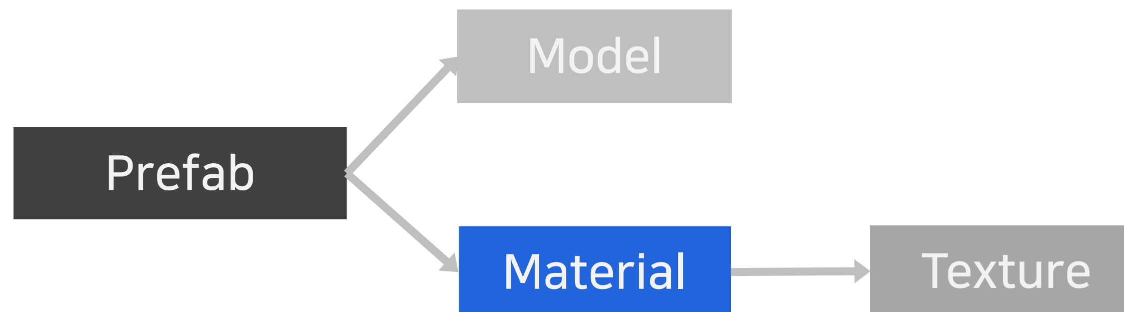
최적화를 위한 개발

© 2021 NHN FORWARD. All rights reserved.

최적화를 위한 개발

유니티의 의존성

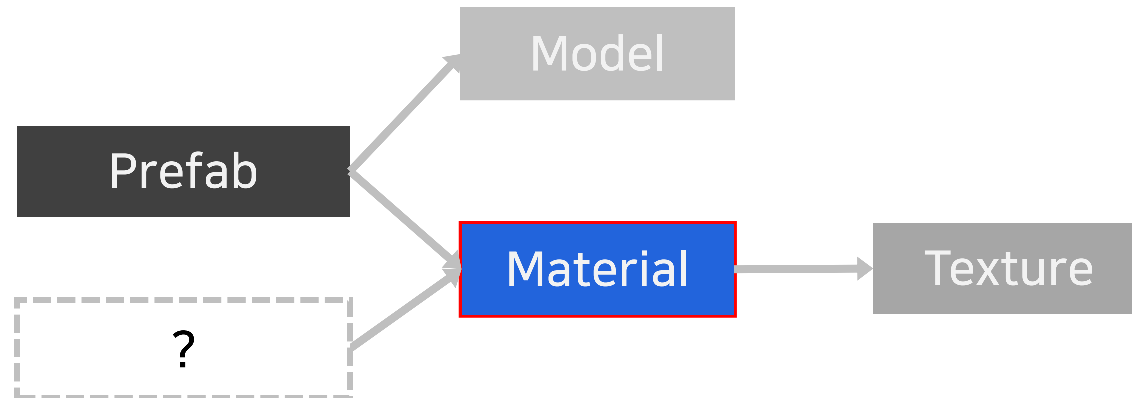
- 포함하는 에셋의 의존성을 지원한다.
  - AssetDatabase.GetDependencies(path)
  - EditorUtility .CollectDependencies(object)



최적화를 위한 개발

유니티의 의존성

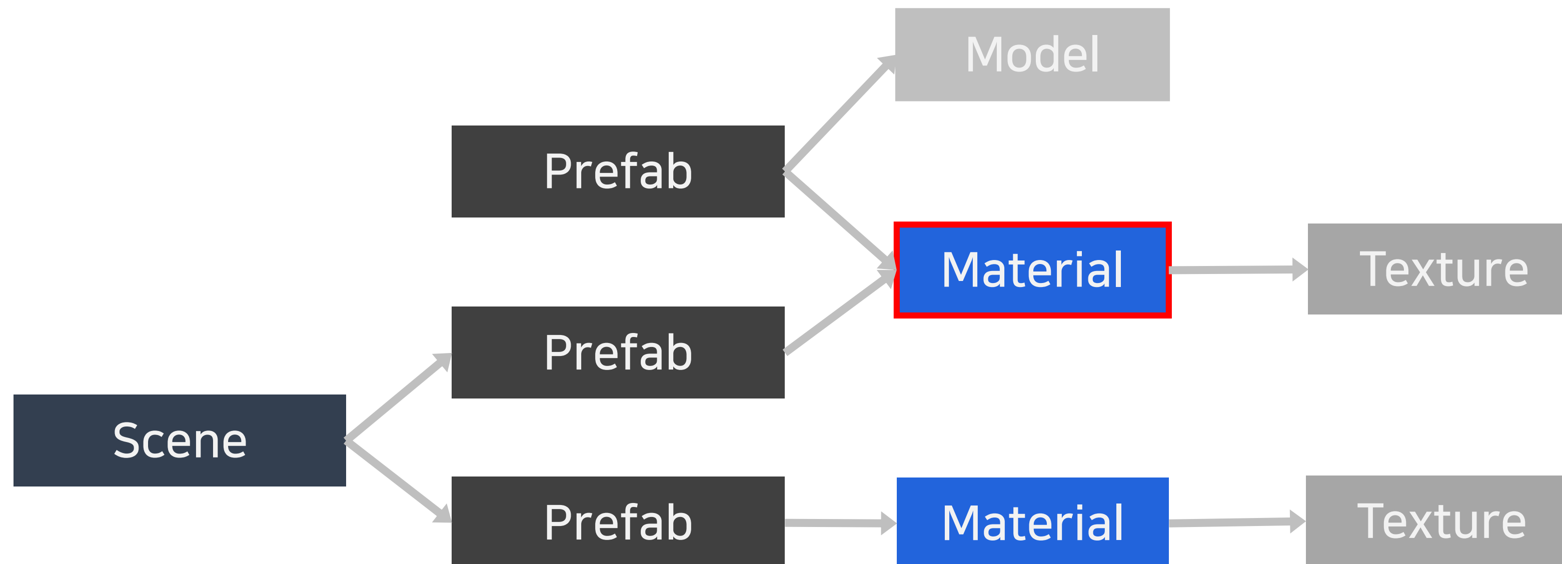
- 포함하는 에셋의 의존성을 지원한다.
  - AssetDatabase.GetDependencies(path)
  - EditorUtility .CollectDependencies(object)
- 참조되는 에셋은 알 수 없다



최적화를 위한 개발

유니티의 의존성

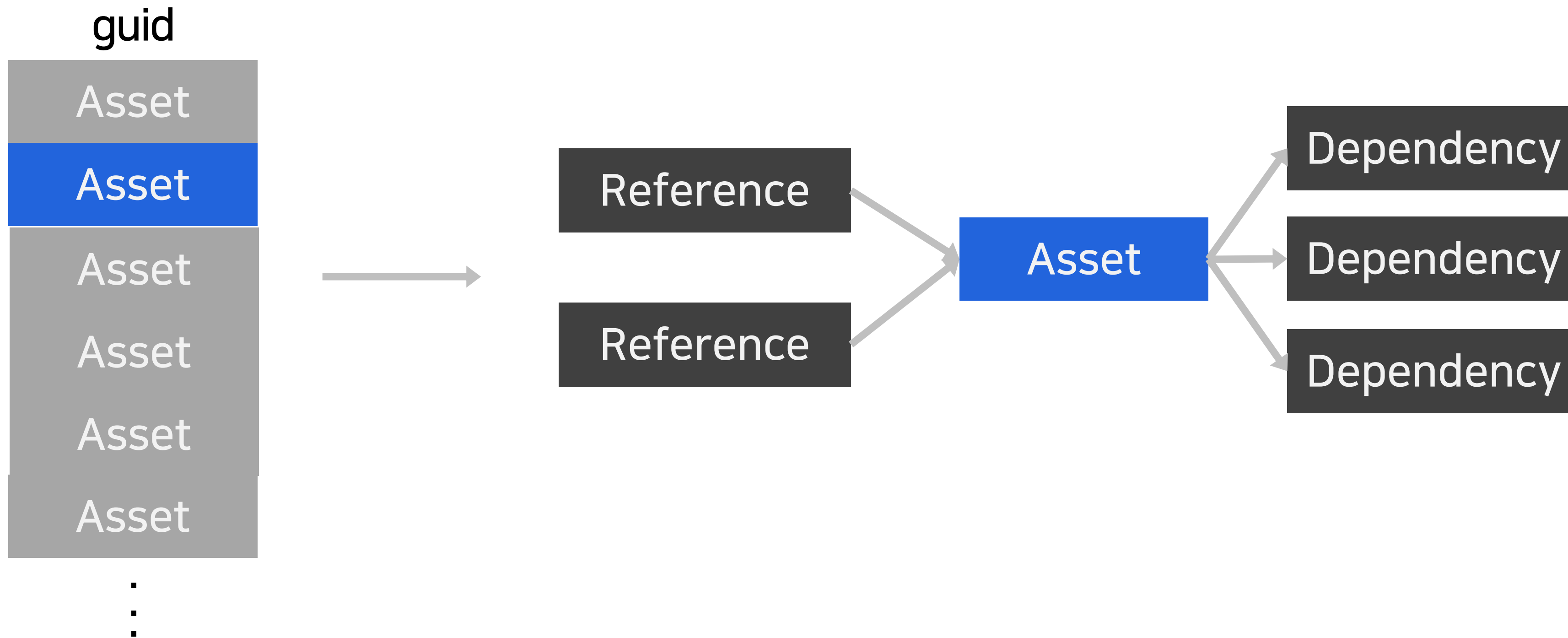
- 모든 에셋의 의존성을 검사하면 참조를 알 수 있다.



최적화를 위한 개발

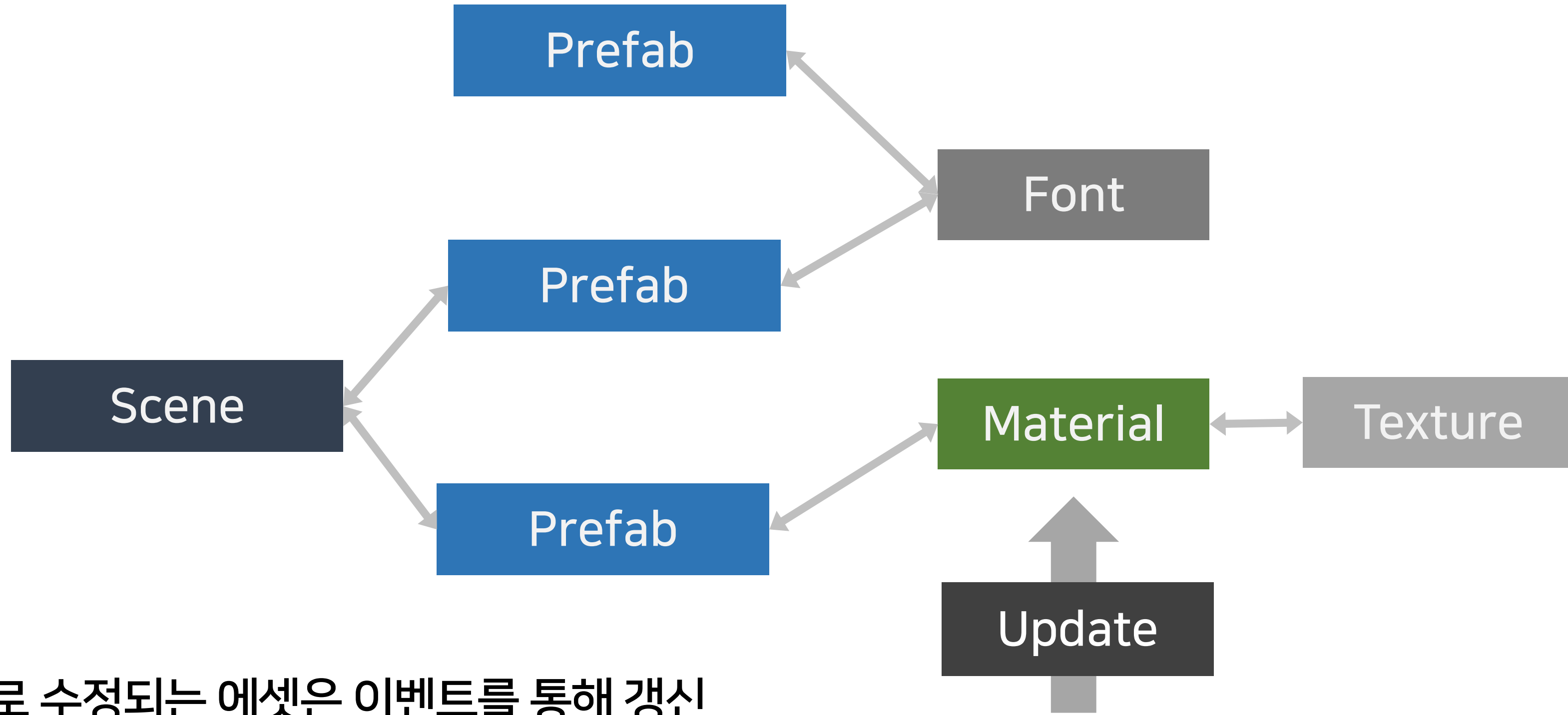
반복적으로 사용할 수 있어야 한다

- 재사용 할 수 있도록 캐싱
- 쉽게 접근이 쉽도록 관리
- 갱신이 쉬운 구조



최적화를 위한 개발

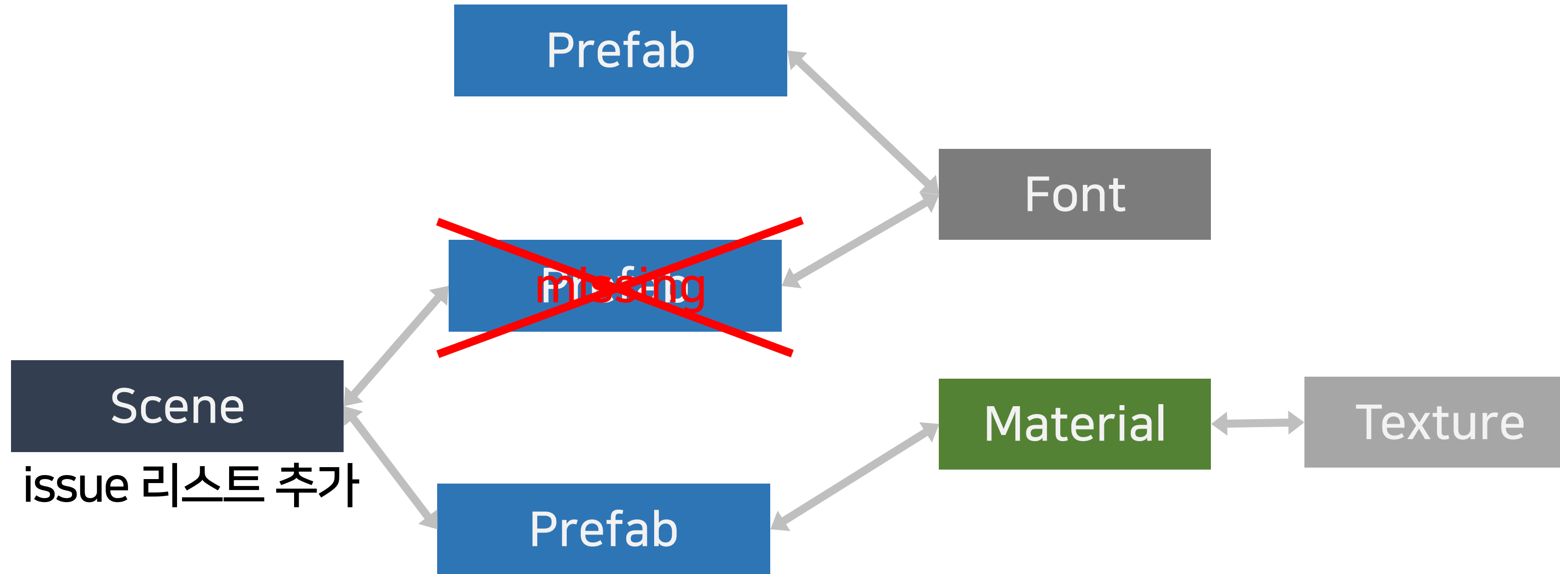
1. 모든 에셋의 의존성을 그래프 구조로 캐싱한다.



2. 실시간으로 수정되는 에셋은 이벤트를 통해 갱신

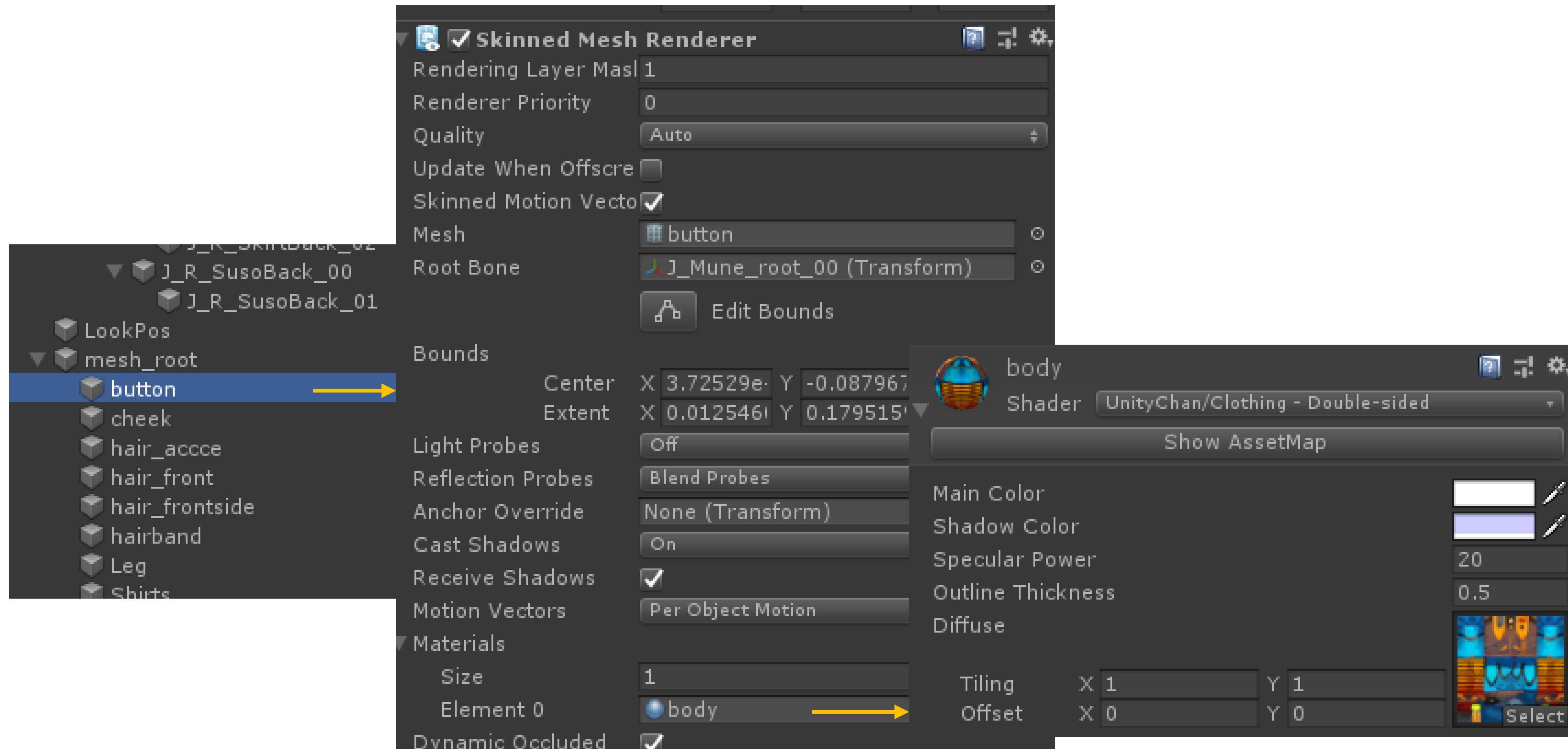
최적화를 위한 개발

문제 예방



최적화를 위한 개발

에셋 내부의 오브젝트



최적화를 위한 개발


```
--- !u!137 &137400926127224090
SkinnedMeshRenderer:
  m_ObjectHideFlags: 1
  m_CorrespondingSourceObject: {fileID: 0}
  m_PrefabInternal: {fileID: 100100000}
  m_GameObject: {fileID: 1800512846348982}
  m_Enabled: 1
  m_CastShadows: 0
  m_ReceiveShadows: 0
  m_DynamicOccludee: 1
  m_ReflectionProbeUsage: 1
  m_RenderingLayerMask: 4294967295
  m_Materials:
    - {fileID: 2100000, guid: eaa601280af69c143a19ea01108d48d8, type: 2}
  m_StaticBatchInfo:    firstSubMesh: 0    subMeshCount: 0
```

최적화를 위한 개발

에셋 내부의 오브젝트

```
--- !u!1 &100036
GameObject:
  m_ObjectHideFlags: 0
  m_CorrespondingSourceObject: {fileID: 0}
  m_PrefabInstance: {fileID: 0}
  m_PrefabAsset: {fileID: 0}
  serializedVersion: 6
  m_Component:
  - component: {fileID: 400036}
  - component: {fileID: 13700002}
  m_Layer: 0
  m_Name: button
  m_TagString: Untagged
  m_Icon: {fileID: 0}
  m_NavMeshLayer: 0
  m_StaticEditorFlags: 0
  m_IsActive: 1
```

Skinned Mesh Renderer

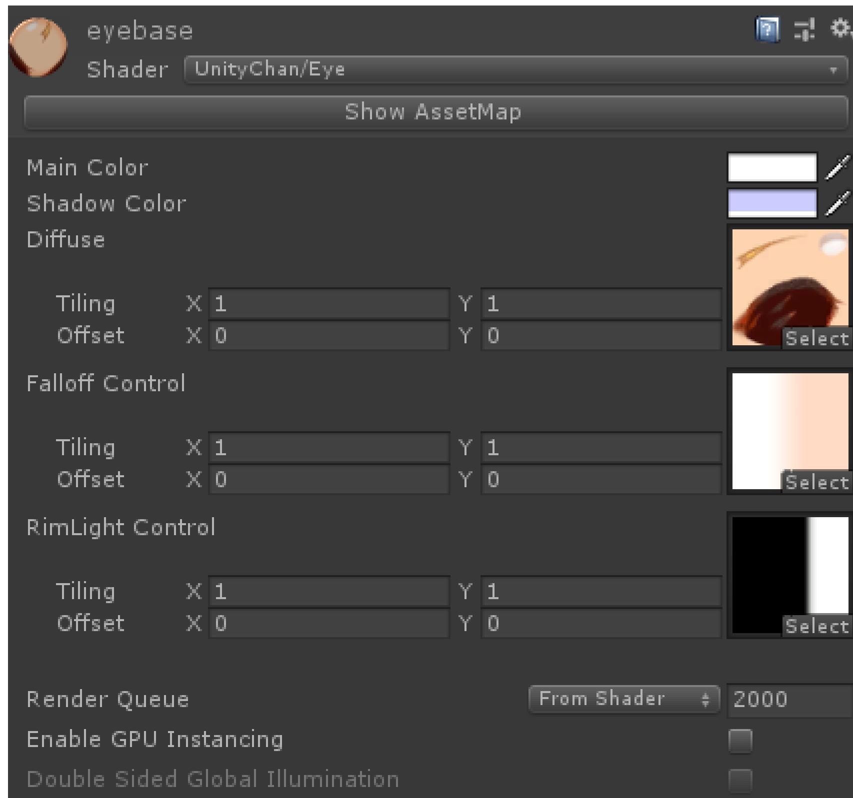
```
--- !u!137 &13700002
SkinnedMeshRenderer:
  m_ObjectHideFlags: 0
  m_CorrespondingSourceObject: {fileID: 0}
  m_PrefabInstance: {fileID: 0}
  m_PrefabAsset: {fileID: 0}
  m_GameObject: {fileID: 100036}
  m_Enabled: 1
  m_CastShadows: 1
  m_ReceiveShadows: 1
  m_DynamicOccludee: 1
  m_MotionVectors: 1
  m_LightProbeUsage: 0
  m_ReflectionProbeUsage: 1
  m_RenderingLayerMask: 1
  m_RendererPriority: 0
  m_Materials:
  - {fileID: 2100000, guid: bb570020a8c
  m_StaticBatchInfo:
    firstSubMesh: 0
```

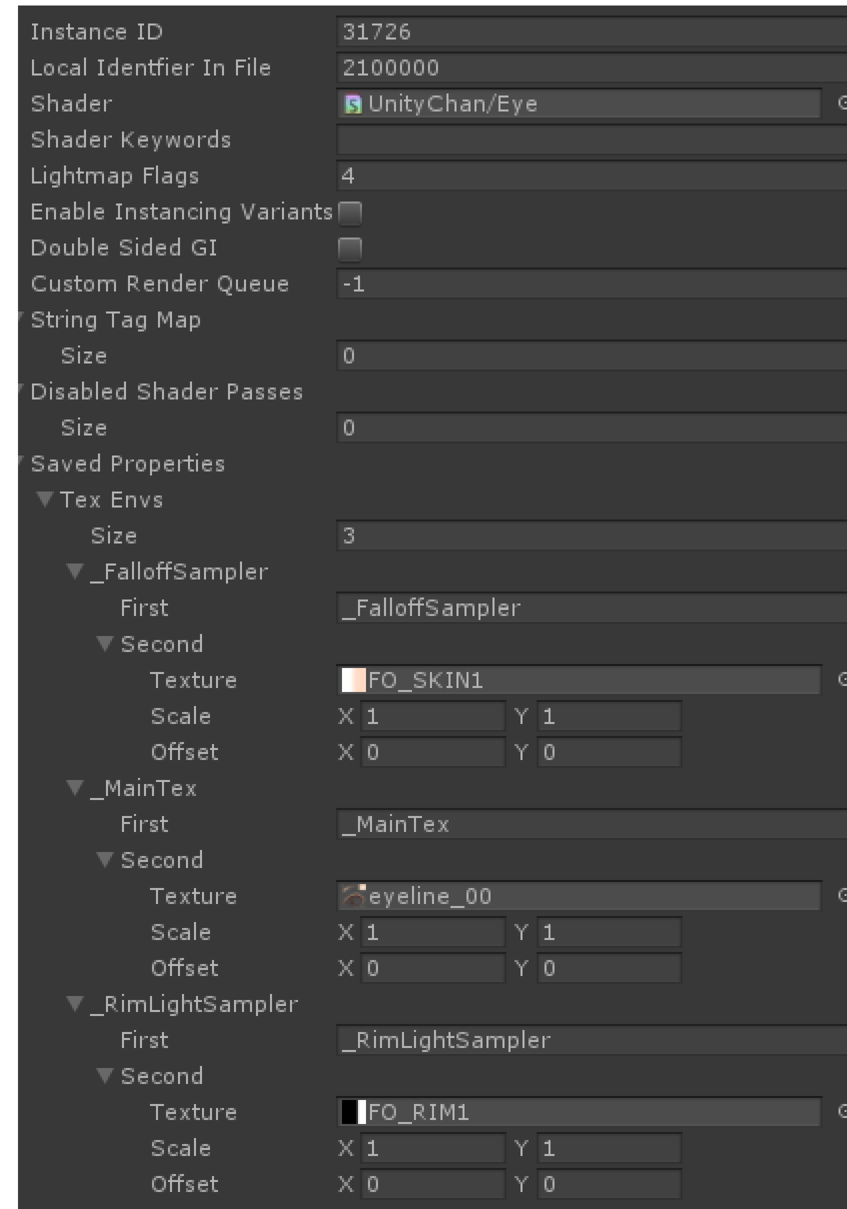
```
--- !u!21 &2100000
Material:
  serializedVersion: 3
  m_ObjectHideFlags: 0
  m_PrefabParentObject: {fileID: 0}
  m_PrefabInternal: {fileID: 0}
  m_Name: body
  m_Shader: {fileID: 4800000, guid: 96d05de60c5f7474491f9f94568c
  m_ShaderKeywords: []
  m_CustomRenderQueue: -1
  m_SavedProperties:
    serializedVersion: 2
    m_TexEnvs:
```

최적화를 위한 개발

에셋 내부의 오브젝트

- 직렬화 구조로 되어있다
  - SerializedObject와 SerializedProperty로 관리





최적화를 위한 개발


```
SerializedObject so = new SerializedObject(object);
SerializedProperty sp = so.GetIterator();

while (sp.NextVisible(true))
{
    if (sp.propertyType == SerializedPropertyType.ObjectReference)
    {
        // sp.objectReferenceValue;
    }
}
```

최적화를 위한 개발

자유롭게 요리하자

- 직관적으로 시각화
- 에셋 조건 탐색, 교체
- 프로젝트 이슈 예방
- 불필요한 프로젝트 정리

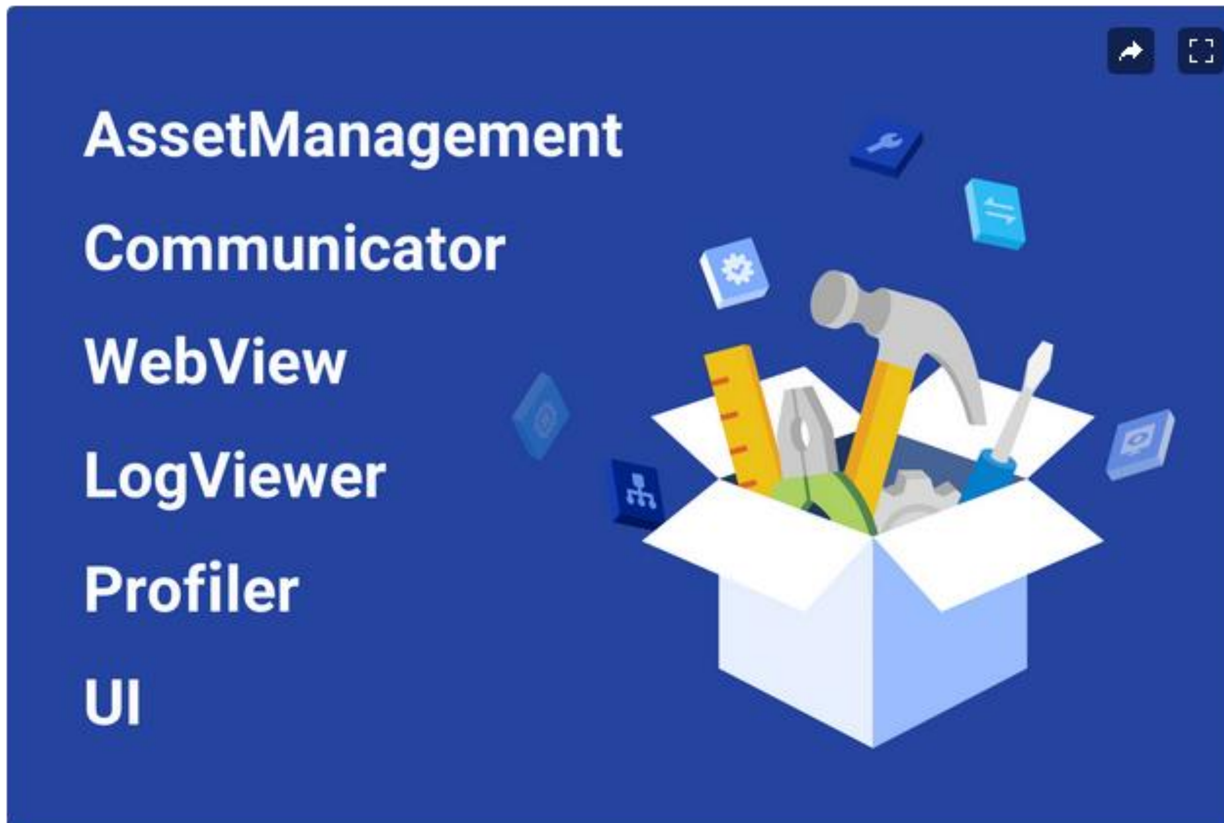
AssetManagement의 활용

© 2021 NHN FORWARD. All rights reserved.

AssetManagement의 활용

개발기술 기반 구축 시간 단축

- 필요 기술 조건이 전제되는 기술
- 에셋 스토어의 Game Package Manager(GPM) 서비스를 통해 기능 제공



Game Package Manager

NHN Corp. ★★★★★ 5 | 26 Reviews

FREE

Update

License	Extension Asset
File size	234.5 KB
Latest version	2.0.3
Latest release date	May 28, 2021
Supported Unity versions	2018.4.0 or higher
Support	Visit site

AssetManagement의 활용

의존성 주도로 개발하기 위한 도구

- 의존성의 즉각적인 상호작용
- 직관적으로 시각화 관리
- 에셋의 탐색, 변경
- 이슈 관리
- 불필요한 에셋 제거

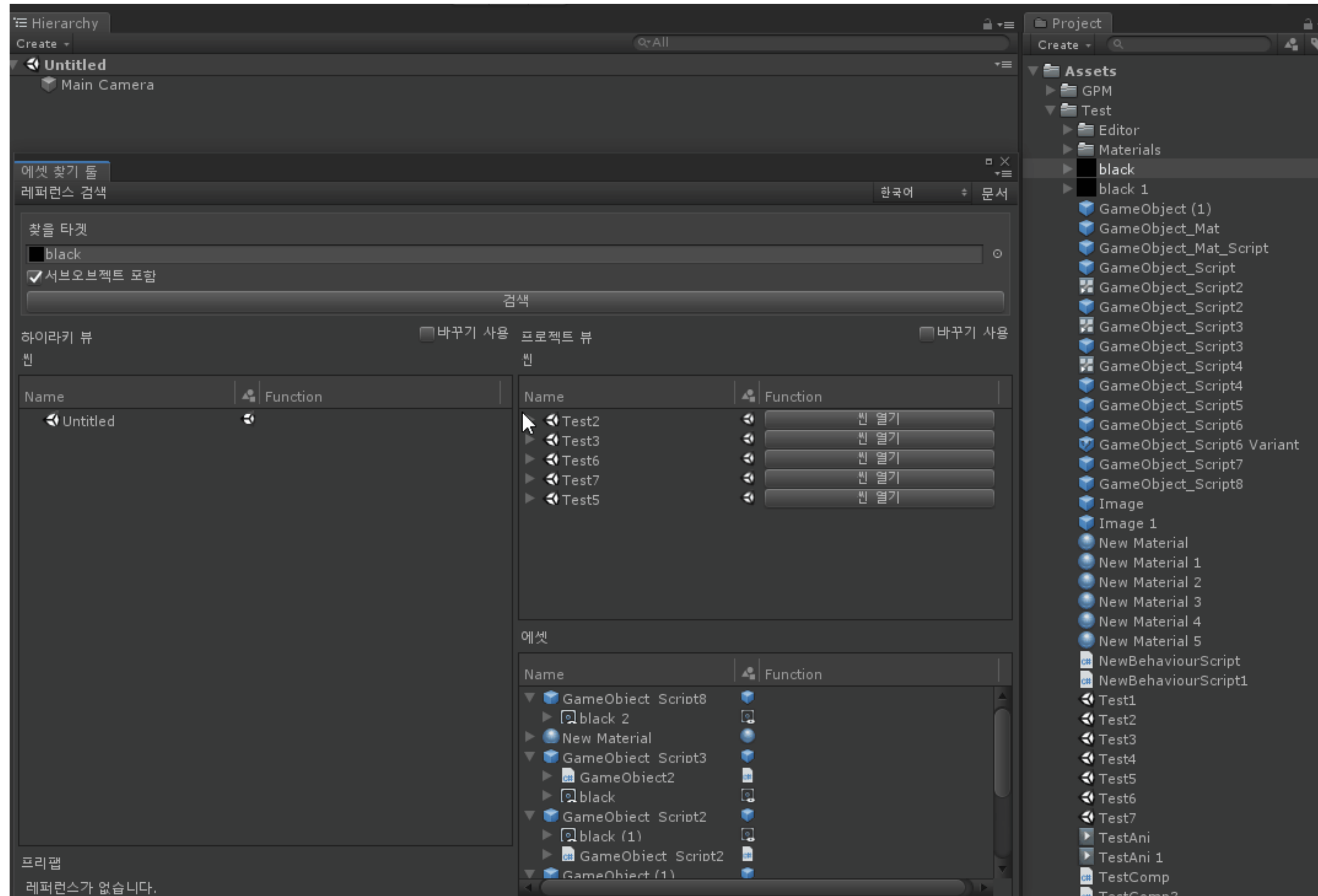
AssetManagement의 활용

의존성의 시각화

AssetManagement의 활용

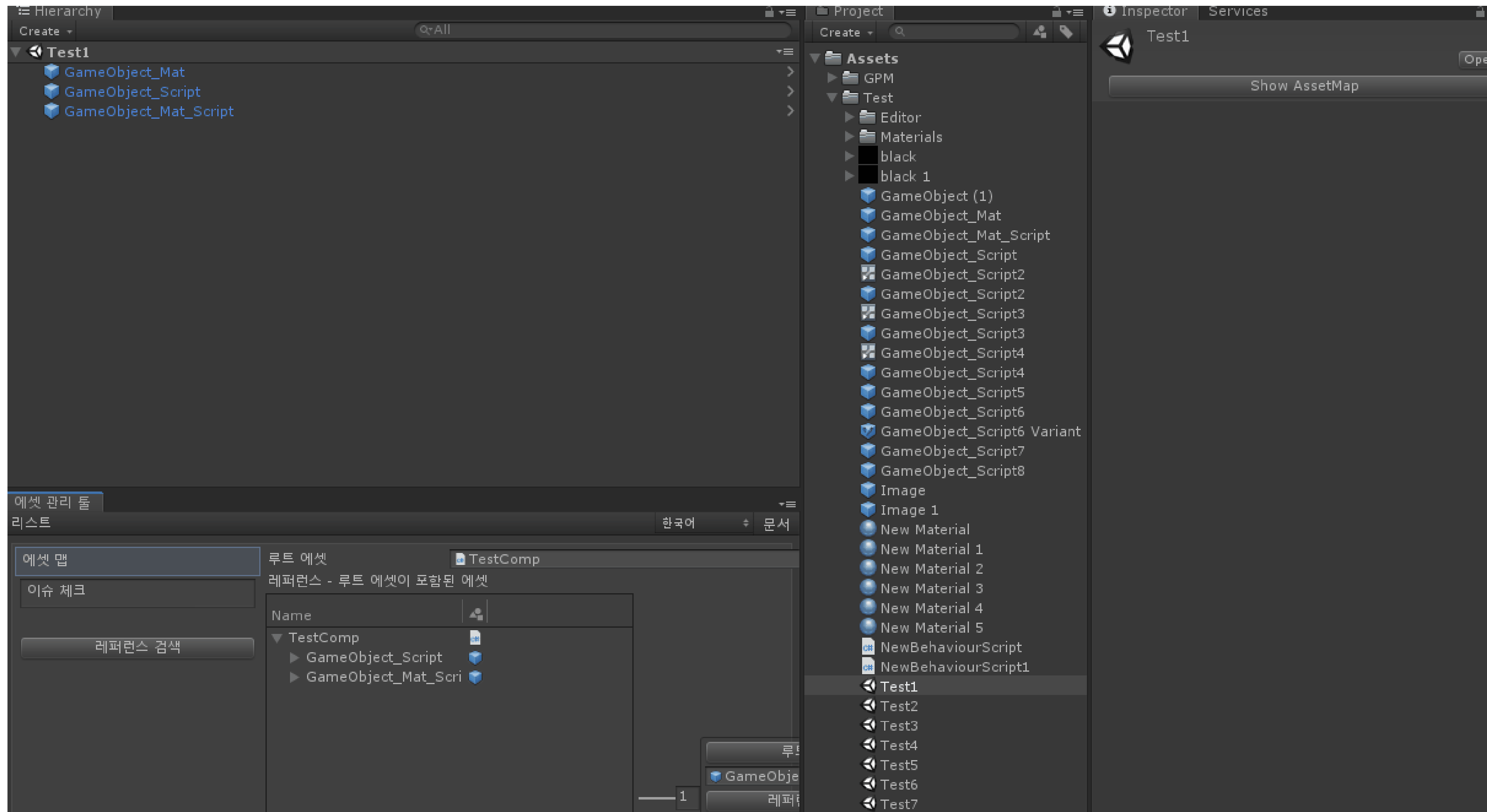
NHN FORWARD

에셋 탐색, 교체



AssetManagement의 활용

이슈 해결



정리

© 2021 NHN FORWARD. All rights reserved.

정리

프로젝트를 개선하자

- 프로젝트 개발 효율 향상
- 프로젝트의 안정성 향상
- 인적 관리 효율성 향상

정리

불필요한 부분을 없애자

- 중복제거
- 일괄 처리

정리

의존성 주도로 개발 하려면 필수

- 직관적인 분석
- 즉각적인 피드백

정리

기반 구축은 AssetMangement를 활용하자!

- 필요 기술 조건이 전제되는 기술
- Assetmanagemanet 사용을 통해 기반 구축 시간을 단축하자
- 에셋 스토어의 Game Package Manager(GPM) 서비스를 통해 기능 제공

정리

널리 세상을 이롭게 하자

- 많은 사용 부탁드립니다.
- 주변에 기술 공유 부탁드립니다.
- 적극적인 피드백 부탁드립니다.

정리

참고

- Unity 의존성을 통한 에셋 관리로 생산성 향상
  - MeetUp: https://meetup.toast.com/posts/263

Unity 의존성

Unity 의존성을 통한 에셋 관리로 생산성 향상 : NHN Cloud Meetup

NHN FORWARD