1. 유니티 편집기(Unity Editor)

(1) 유니티 사용하기

⑩ 프리팹(Prefab) 시스템의 이해

유니티 프리팹 시스템은 게임 객체를 애셋으로 생성하고 관리할 수 있는 시스템이다. 프리팹 시스템은 게임 객체를 특별한 방법으로 재사용할 수 있는 애셋(프리팹 애셋)으로 생성할 수 있도록 지원한다. 프리팹 애셋은 새로운 프리팹 인스턴스(Instance, 게임 객체)를 생성할 수 있는 템플릿(Template)처럼 동작한다.

프리팹이 필요한 이유를 이해하기 위하여 다음의 두 가지 경우를 생각해보자.

- 경우 1

  큐브(Cube) 게임 객체를 10개 생성하자. 구(Sphere) 게임 객체를 2개 생성하자(구 게임 객체의 이름은 "Sphere01"과 "Sphere02"라고 가정하자). 그리고 다음과 같은 스크립트를 생성하여 모든 큐브 게임 객체에 연결하자. 각 큐브 게임 객체의 스크립트 컴포넌트의 "Target"에 "Sphere01"을 연결하자(어떻게 ?).

```
public class MoveTowardAngle : MonoBehaviour
{
   public Transform target;
   public float speed = 1.0f;

   void Update()
   {
      ...
   {
};
```

  어떤 성실 근면한 학생은 스크립트를 드래그하여 각 큐브 게임 객체에 연결하는 것을 10번 반복할 것이다. 그리고 "Sphere01" 게임 객체를 각 큐브 게임 객체의 스크립트 컴포넌트의 "Target"에 드래그하여 연결하는 것을 10번 반복할 것이다. 조금 꾀를 부리는 학생은 큐브 게임 객체를 1개("Cube01")를 먼저 생성한 다음에, "Sphere01" 게임 객체를 큐브 게임 객체의 스크립트 컴포넌트의 "Target"에 드래그하여 연결하고, "Cube01"을 9번 복제(Ctrl+D, 또는 Ctrl+C & Ctrl+V)할 것이다.

  그런데 스크립트 컴포넌트의 "Target"에 "Sphere01" 게임 객체가 아닌 "Sphere02" 게임 객체를 연결해야 함을 나중에 깨달았다. 그리고 스크립트 컴포넌트의 "Speed"의 값을 3.0으로 바꾸어야 한다. 이제 어떻게 할 것인가? 성실 근면한 학생이던 꾀돌이 학생이던 상관없이 10개의 모든 큐브 게임 객체에 대하여 작업을 반복해야 한다. 만약 큐브 게임 객체가 10개가 아니라 100개 또는 1000개라면 어떻게 할 것인가?

- 경우 2

큐브(Cube) 게임 객체를 생성하고 게임 카메라에 보이도록 배치하자. 구(Sphere) 게임 객체를 생성하고 게임 카메라의 가운데 보이도록 배치하자.

다음과 같은 "ExplosiveCube" 스크립트를 생성하여 큐브 게임 객체에 연결하자.

```
public class ExplosiveCube : MonoBehaviour
{
    Vector3 direction;

    void Start()
    {
        direction = Random.onUnitSphere;
        Renderer renderer = GetComponent<Renderer>();
        renderer.material.color = Random.ColorHSV(0f, 1f, 1f, 1f,
0.5f, 1f);
        Destroy(gameObject, 3.0f);
    }

    void Update()
    {
        transform.Translate(direction * 10.0f * Time.deltaTime);
        transform.Rotate(direction, 360.0f * Time.deltaTime);
    }
}
```

다음과 같은 "Explosion" 스크립트를 생성하여 구 게임 객체에 연결하고 인스펙터 뷰에서 "Explosion" 컴포넌트의 "Cube"의 값으로 큐브 게임 객체를 연결하자.

```
public class Explosion : MonoBehaviour
{
    public ExplosiveCube cube;

    void OnMouseDown()
    {
        for (int i = 0; i < 100; i++)
        {
            ExplosiveCube obj = Instantiate<ExplosiveCube>(cube,
transform.position, transform.rotation);
        }
    }
}
```

게임 프로젝트를 실행하고 게임 뷰에서 구 게임 객체를 마우스로 클릭해보라. 구 게임 객체를 3초 이내에 마우스로 클릭하면 폭발이 일어난다. 두 번째로 클릭하거나 3초가 지난 후에 클릭하면 다음과 같은 오류가 콘솔 뷰에 출력된다.

"MissingReferenceException: The object of type 'ExplosiveCube' has been destroyed but you are still trying to access it.
Your script should either check if it is null or you should not destroy the object."

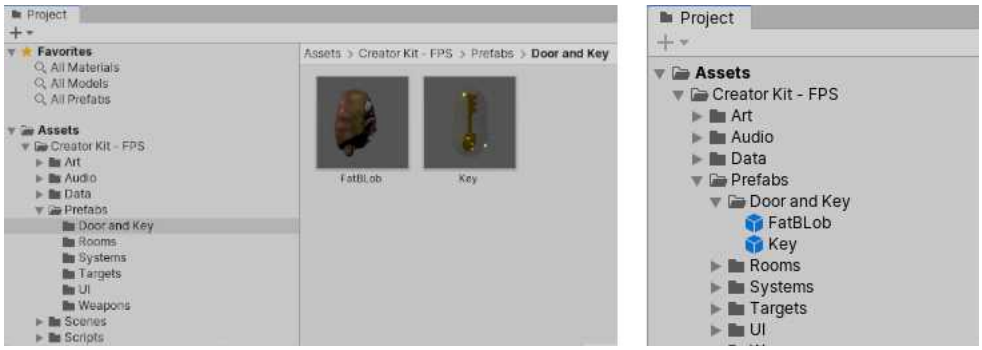
이러한 상황은 총알을 발사하거나 게임 객체를 리스폰(Respawn)할 때 발생할 수 있다.

위의 두 가지 상황을 손쉽게 해결하는 방법은 프리팹을 사용하는 것이다.

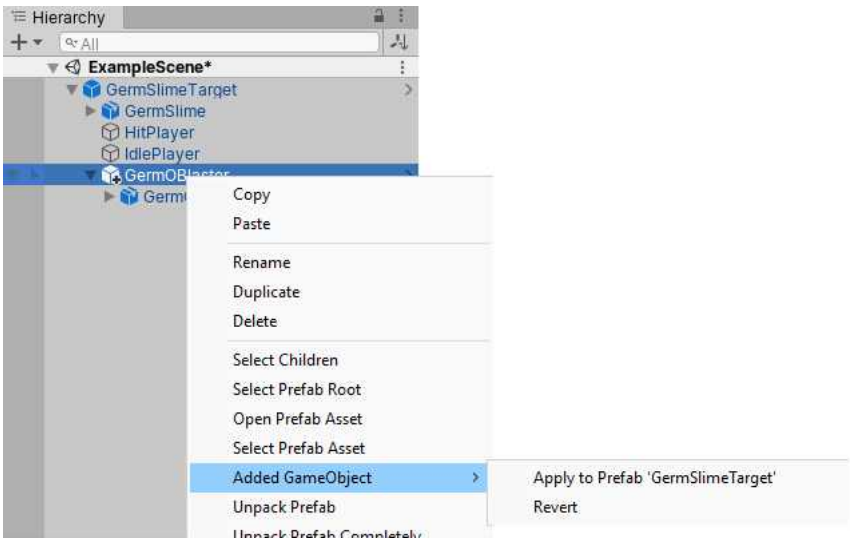
⑪ 프리팹 애셋(Prefab Asset)의 생성

- 프리팹 애셋의 생성

  계층 뷰에서 게임 객체를 선택하여 프로젝트 뷰로 드래그&드랍을 하면 프리팹 애셋(간단히 프리팹이라고 부른다)이 생성된다. 게임 객체(연결된 모든 컴포넌트와 자손 게임 객체들을 포함하여)가 재사용할 수 있는 애셋이 된다(게임 객체를 완전히 복사하여 애셋으로 생성한다). 애셋은 게임 객체가 아님에 유의하라. 프리팹 애셋은 게임 객체를 특별한 방법으로 재사용할 수 있는 애셋으로 만든 것이다. 프리팹 애셋은 다음과 같이 프로젝트 뷰에서 섬네일 뷰(Thumbnail view) 또는 파란색 큐브 프리팹 아이콘으로 표시된다. 프리팹을 생성한 다음에 원래의 게임 객체는 필요가 없다면 제거하여도 된다.

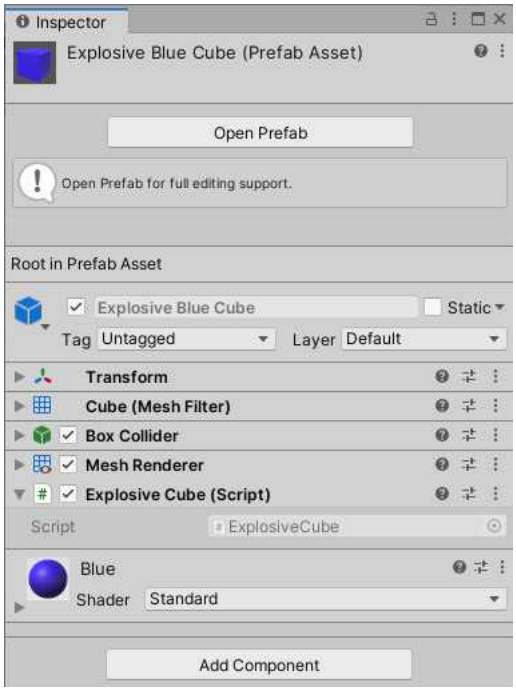


게임 객체에서 프리팹 애셋을 생성하면 원래의 게임 객체의 이름과 자식 게임 객체는 파란색 글씨로 표시된다. 루트 게임 객체는 파란색 큐브 프리팹 아이콘으로 표시된다. 이것은 원래의 게임 객체가 프리팹 애셋의 인스턴스(Instance)임을 표시하는 것이다. 프리팹 애셋의 인스턴스 또는 간단히 프리팹 인스턴스라고 한다. 프리팹 인스턴스는 프리팹 애셋에서 게임 객체를 생성한 것을 의미한다.



계층 뷰에서 새로운 게임 객체를 선택하여 프리팹으로 드래그&드랍하면 프리팹의 내용에 해당하는 게임 객체를 교체할 수 있다.

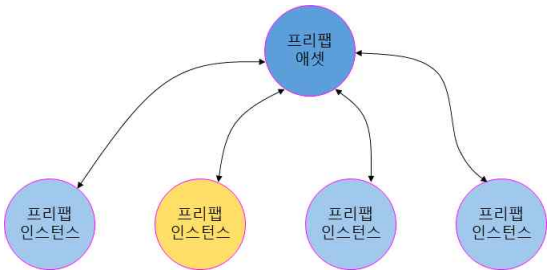
프로젝트 뷰에서 프리팹 애셋을 선택하면 인스펙터 뷰에 다음 그림과 같이 표시된다. 이것을 보면 프리팹 애셋은 게임 객체의 속성을 가지고 있음을 알 수 있다. 게임 객체처럼 프리팹 애셋에 새로운 컴포넌트를 추가하거나 컴포넌트를 수정할 수 있으며, 필요 없는 컴포넌트를 제거할 수도 있다.



- 프리팹 인스턴스(Instance)의 생성

  프로젝트 뷰에서 프리팹 애셋을 선택하여 계층 뷰 또는 씬 뷰로 드래그&드랍을 하면 프리팹 인스턴스(게임 객체)를 생성할 수 있다. 스크립트에서 Instantiate() 함수를 사용하여 동적으로 프리팹 인스턴스를 생성할 수 있다.

프리팹 인스턴스가 생성되면 생성된 모든 프리팹 인스턴스들은 다음 그림과 같이 프리팹 애셋과 연결되어 관리된다.



프리팹 시스템은 프리팹 애셋에서 생성된 게임 객체(프리팹 인스턴스)들을 관리한다. 프리팹 애셋의 내용이 바뀌면(편집이 되면) 연결된 모든 프리팹 인스턴스들의 해당 내용이 한꺼번에 자동적으로 바뀌게 된다.

프리팹 애셋의 루트 변환(Transform) 컴포넌트의 위치(Position)와 회전(Rotation) 속성의 수정은 프리팹 인스턴스로 반영되지 않는다. 이것이 반영되면 모든 프리팹 인스턴스들의 위치와 방향이 같아지기 때문이다. 2D 게임 객체들의 사각형 변환 컴포넌트(Rect Transform)의 "Width, Height, Margins, Anchors, Pivot) 속성도 반영되지 않는다. 자손 게임 객체의 위치와 방향은 부모 게임 객체에 상대적인 위치와 방향으로 표현되므로 프리팹 애셋에서 수정되면 반영된다.

- 프리팹 인스턴스 오버라이드(Instance Override)

  프리팹 인스턴스(게임 객체)의 어떤 컴포넌트 속성이 프리팹 애셋 또는 다른 프리팹 인스턴스와 다를 필요가 있다. 예를 들어, 같은 프리팹 애셋으로 만들어진 NPC 게임 객체들이 많을 때, 각 NPC 게임 객체들의 속력이 다를 수 있다. 이러한 이유로 프리팹 인스턴스들의 속성을 직접 수정할 필요가 있다.

  프리팹 인스턴스의 컴포넌트 속성을 직접 수정하면 프리팹 애셋은 아무런 변화가 없다. 그리고 수정된 값은 그 프리팹 인스턴스에서만 유지된다. 즉, 프리팹 인스턴스의 내용이 프리팹 애셋의 내용을 오버라이드(Instance override)한다.

  인스턴스 오버라이드의 유형은 다음과 같다.
  - 프리팹 인스턴스의 속성 값을 프리팹 애셋과 다르게 수정(오버라이드)하기
    자손 게임 객체를 비활성화(Inactive)하기
  - 새로운 컴포넌트를 추가하기
  - 컴포넌트를 제거하기
  - 자식 게임 객체를 추가하기

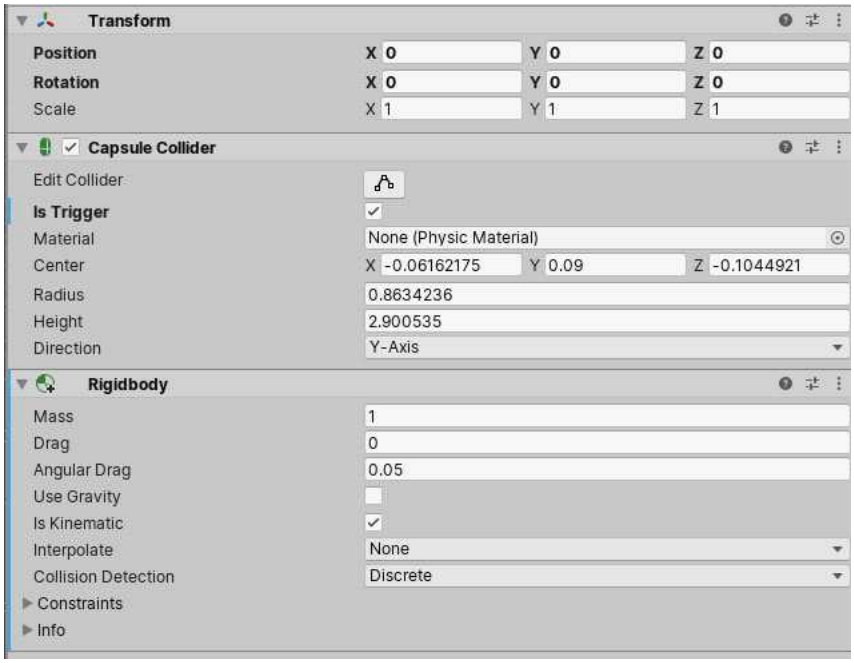
  오버라이드가 된 속성은 프리팹 인스턴스의 인스펙터 뷰에서 다음 그림과 같이 굵은 글씨(Bold)로 표시되고, 왼쪽에 파란색 선이 나타난다.

  프리팹 인스턴스에서 새로운 컴포넌트를 추가하면 컴포넌트 전체에 파란색 선이 나타나고 컴포넌트 아이콘에 + 표시가 나타난다. 계층 뷰에서 이 프리팹 인스턴스의 아이콘에 + 표시가 나타난다. 프리팹 인스턴스에서 컴포넌트를 제거하면 컴포넌트 아이콘에 - 표시가 나타난다.

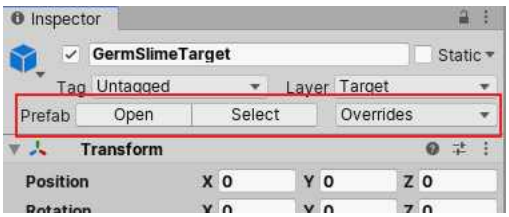
  프리팹 인스턴스에서 프리팹의 일부분인 게임 객체를 제거할 수 없다. 프리팹 인스턴스에서 프리팹의 일부분인 게임 객체를 다른 게임 객체의 자식이 되게 할 수 없다.

  프리팹 인스턴스에서 오버라이드를 한 속성의 값이 프리팹 애셋의 값보다 항상 우선

한다. 즉, 프리팹 애셋의 값을 수정해도 프리팹 인스턴스에서 오버라이드 한 속성의 값은 변하지 않는다.



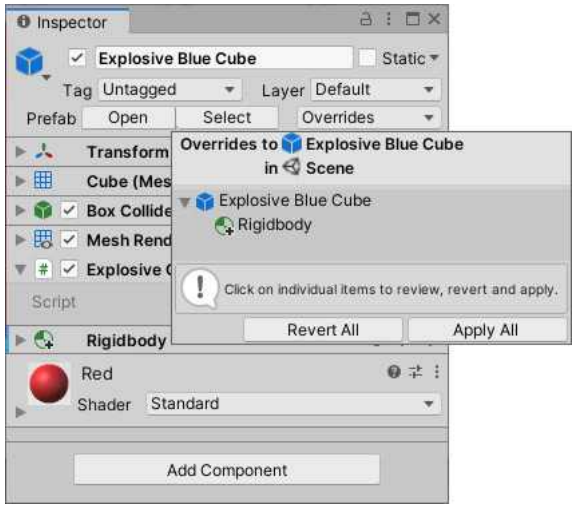
프리팹 인스턴스를 선택하면 인스펙터 뷰에 다음과 같이 3개의 추가적인 버튼이 표시된다(“Open”, “Select”, “Overrides”).



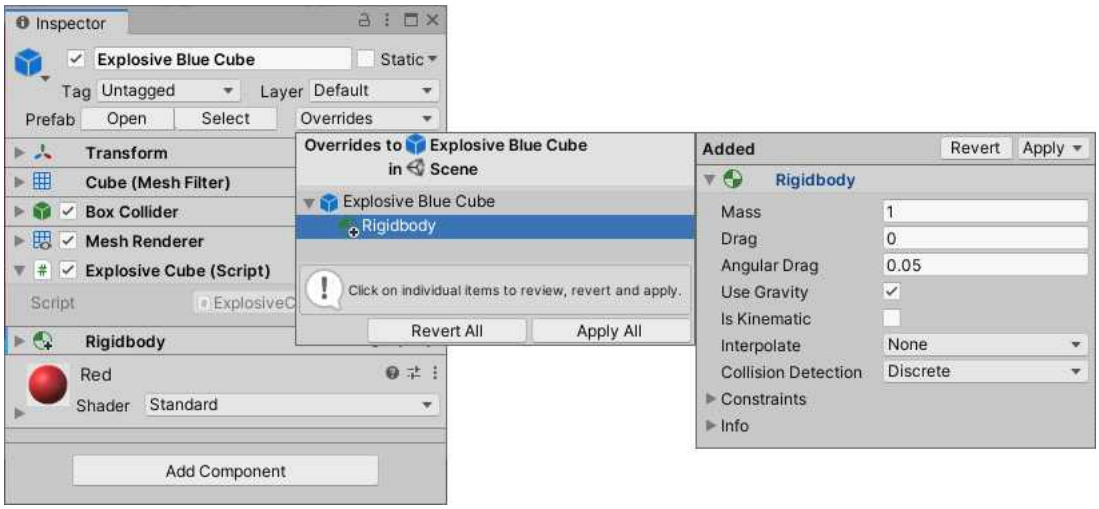
“Open” 버튼을 선택하면 현재 프리팹 인스턴스를 생성한 프리팹 애셋을 프리팹 모드에서 편집할 수 있다.

“Select” 버튼을 선택하면 현재 프리팹 인스턴스를 생성한 프리팹 애셋을 프로젝트(애셋) 뷰에서 선택한다.

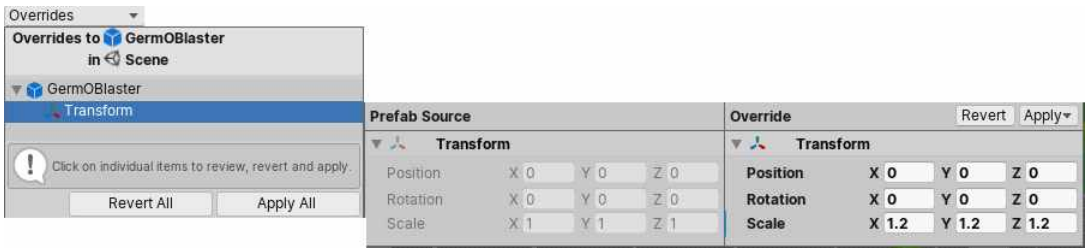
“Overrides” 버튼을 선택하면 현재 프리팹 인스턴스에서 오버라이드 한 내용이 다음 그림과 같이 나타난다.



오버라이드 한 내용을 클릭하면 수정된 내용을 옆에 별도의 윈도우로 표시한다. 다음의 그림은 새로운 컴포넌트를 추가하여 오버라이드 한 것을 보여준다.



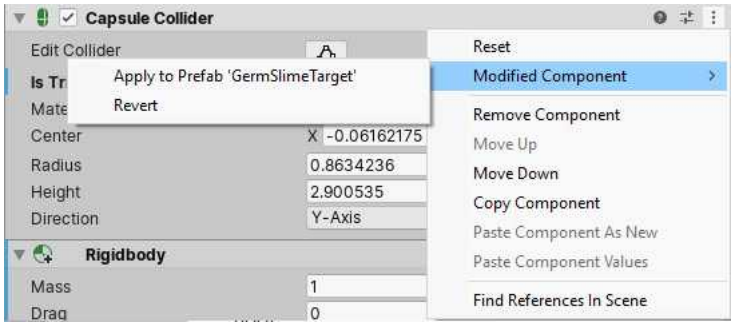
프리팹 애셋의 속성 값을 오버라이드한 경우에는 프리팹 애셋의 값과 오버라이드 한 값을 비교할 수 있도록 나란히 보여준다. “Override” 윈도우에서 값을 수정할 수 있다.



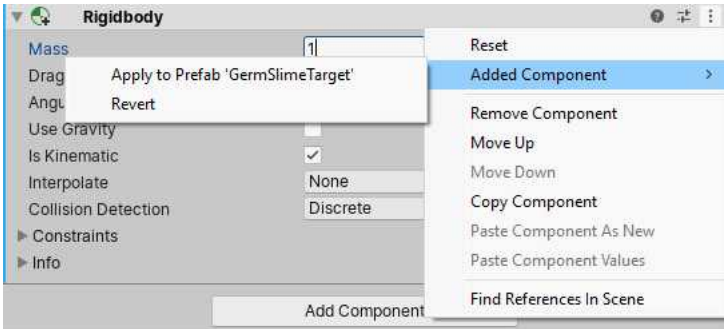
“Revert”는 현재의 오버라이드를 취소하고 프리팹 인스턴스를 프리팹 애셋과 같아지도록 되돌리는 것을 의미한다. “Apply”는 현재 오버라이드로 프리팹 애셋을 수정하는 것을 의미한다. “Apply”를 선택하면 프리팹 애셋이 수정되므로 프리팹 애셋에 연결된 모든 프리팹 인스턴스에 현재의 오버라이드가 반영된다.

“RevertAll”은 모든 오버라이드를 취소하고 프리팹 인스턴스를 프리팹 애셋과 완전히 같아지도록 되돌리는 것을 의미한다. “ApplyAll”은 모든 오버라이드로 프리팹 애셋을 수정하는 것을 의미한다. “Apply”를 선택하면 프리팹 애셋이 수정되므로 프리팹 애셋에 연결된 모든 프리팹 인스턴스에 모든 오버라이드가 반영된다. 프리팹이 중첩(프리팹이 다른 프리팹을 자식으로 갖는 경우)되면 “ApplyAll”은 항상 가장 바깥쪽 프리팹(루트 프리팹)에 적용된다.

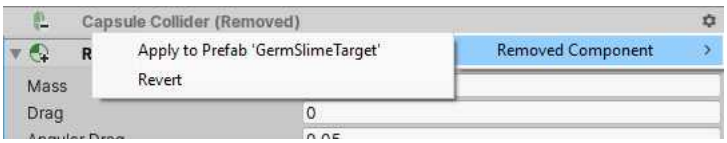
프리팹 인스턴스의 인스펙터 뷰에서 오버라이드 컨텍스트 메뉴를 사용할 수 있다. 오버라이드 한 컴포넌트의 이름 오른쪽에 있는 “⋮” 버튼을 누르거나 컴포넌트 이름 부분을 오른쪽 마우스 버튼으로 선택하면 오버라이드의 유형에 따라 컨텍스트 메뉴가 나타난다. 다음 그림은 속성의 값을 수정하여 오버라이드한 컴포넌트의 경우이다. “Modified Component” 메뉴에서 “Apply” 또는 “Revert”를 선택할 수 있다.



다음 그림은 새로운 컴포넌트를 추가하여 오버라이드한 컴포넌트의 경우이다. “Added Component” 메뉴에서 “Apply” 또는 “Revert”를 선택할 수 있다.

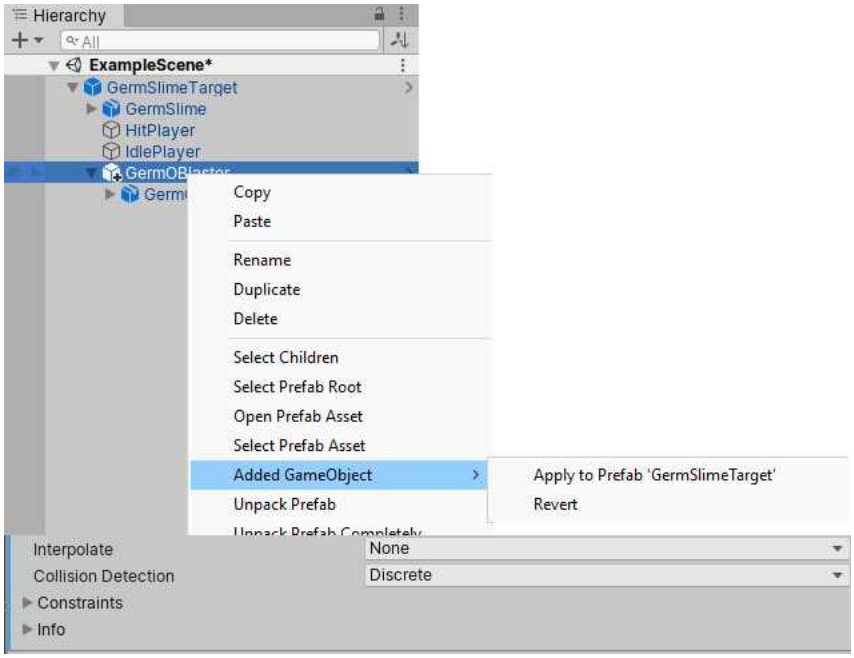


다음 그림은 컴포넌트를 삭제하여 오버라이드한 컴포넌트의 경우이다. “Removed Component” 메뉴에서 “Apply” 또는 “Revert”를 선택할 수 있다.



다음 그림은 자식 게임 객체를 추가하여 오버라이드한 경우이다. 계층 뷰에서 게임 객체의 이름 옆 아이콘에 “+”가 표시된다. 오른쪽 마우스 버튼을 누르면 컨텍스트 메뉴

가 나타나고 “Added GameObject” 메뉴에서 “Apply” 또는 “Revert”를 선택할 수 있다.



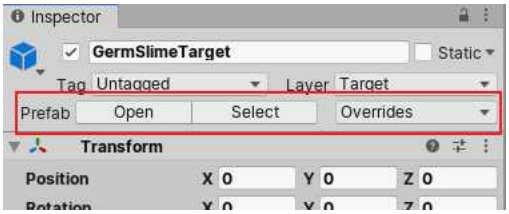
- 프리팹 애셋의 편집(Editing Prefab)

  프리팹 애셋을 편집하려면 프리팹 애셋을 프리팹 모드(Prefab Mode)로 오픈한다. 프리팹 모드는 프리팹 애셋의 내용을 별도의 윈도우에 나타나게 하고 편집할 수 있도록 한다. 프리팹 모드에서 수정한 모든 내용은 연결된 모든 프리팹 인스턴스에 반영된다.

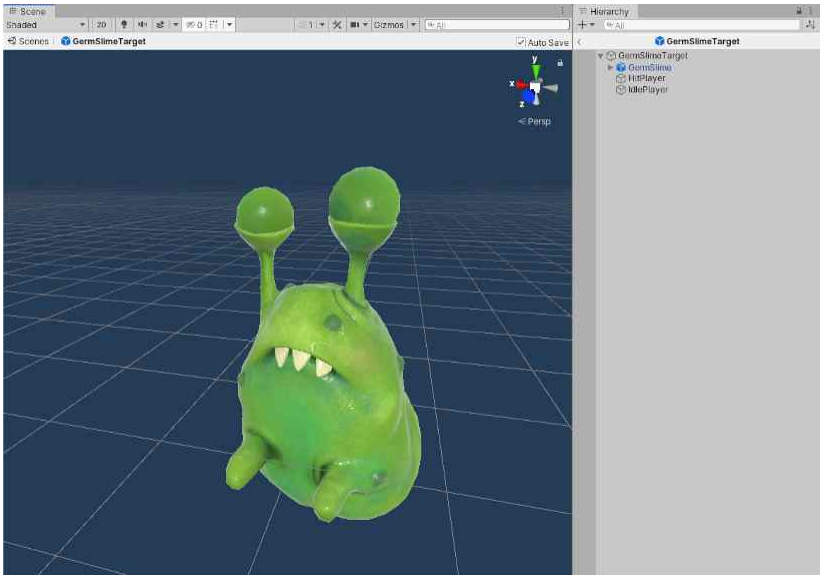
  프리팹 모드는 다음과 같은 방법 중 하나로 선택할 수 있다.

  - 프로젝트 뷰에서 프리팹 애셋을 더블 클릭한다.
  - 인스펙터 뷰에서 “Open Prefab” 버튼을 선택한다.
  - 계층 뷰에서 프리팹 인스턴스 이름 옆의 화살표를 선택한다.

  - 프리팹 인스턴스의 인스펙터 뷰에서 “Open” 버튼을 선택한다.



  프리팹 모드에서 씬 뷰와 계층 뷰에는 기본적으로 다음 그림과 같이 프리팹 애셋만 표시된다. 씬의 다른 게임 객체들은 표시되지 않는다.



프리팹 모드일 때, 씬 뷰의 툴바 아래에 현재 편집 중인 프리팹 애셋의 이름이 다음과 같이 표시된다. “Scenes”를 선택하면 프리팹 모드가 종료된다(씬 뷰로 되돌아 간다).

프리팹 모드일 때, 계층 뷰에 현재 편집 중인 프리팹 애셋의 이름이 다음과 같이 표시되고 이름 왼쪽에 화살표가 표시된다. 화살표를 선택하면 프리팹 모드가 종료된다(씬 뷰로 되돌아 간다).

프리팹 모드일 때, 씬 뷰의 툴바 오른쪽에 “Auto Save” 체크 박스가 표시된다. “Auto Save”가 활성화되면 프리팹 모드에서 편집한 모든 내용이 자동적으로 저장된다. 프리팹 모드에서 수정한 내용은 “Edit ▶ Undo” 또는 “Ctrl+Z”로 취소할 수 있다.

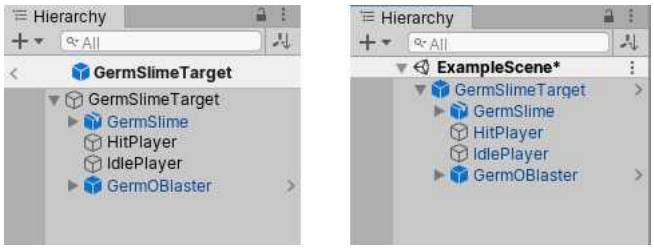
프리팹 모드에서 씬 뷰에 편집 배경(Editing environment)으로 씬을 설정할 수 있다.

“Edit ▶ Project Settings ▶ Editor ▶ Prefab Editing Environments”

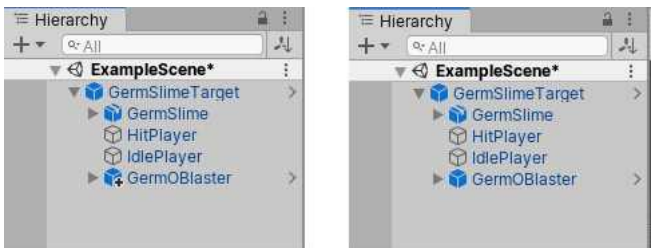
- 프리팹의 중첩(Nested Prefabs)

  프리팹 모드에서, 프리팹 애셋을 선택하여 계층 뷰 또는 씬 뷰의 다른 프리팹의 인스턴스로 드래그&드랍을 하면 자식(게임 객체)으로 만들 수 있다. 이것을 프리팹 중첩이라고 한다. 중첩된 프리팹 인스턴스(자식 게임 객체)는 자신의 프리팹 애셋과 연결을 유지하면서 다른 프리팹 인스턴스의 자식 게임 객체가 된다.

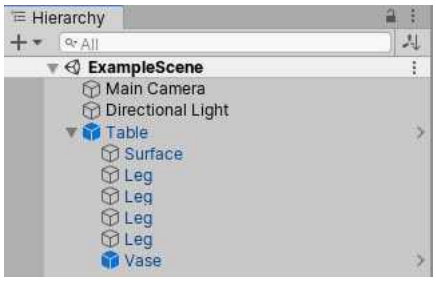
  계층 뷰에서 프리팹의 루트 게임 객체는 파란색으로 표시되지만, 프리팹 모드에서 루트 게임 객체는 다음 왼쪽 그림과 같이 파란색으로 표시되지 않는다.



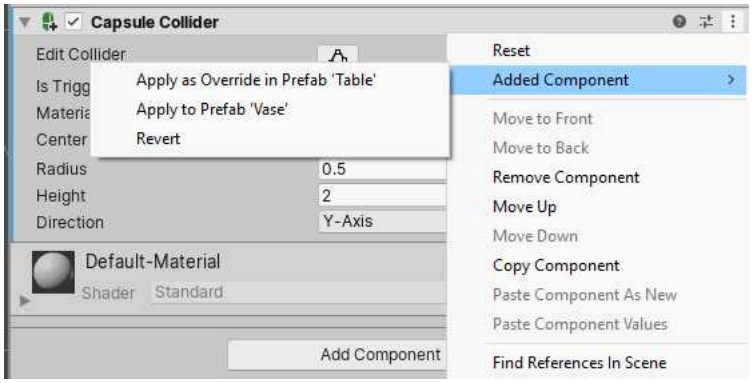
  프리팹 모드가 아닐 때, 프리팹 애셋 또는 다른 프리팹 인스턴스를 선택하여 계층 뷰 또는 씬 뷰의 다른 프리팹의 인스턴스로 드래그&드랍을 하면 자식(게임 객체)으로 만들 수 있다(프리팹 인스턴스 오버라이드). 계층 뷰에서 자식 게임 객체의 아이콘에 “+” 표시가 나타난다(아래 왼쪽 그림). 오버라이드된 프리팹 인스턴스를 “Apply”하면 프리팹이 중첩되고 “+” 표시가 사라진다(아래 오른쪽 그림).



  프리팹이 중첩될 때, 오버라이드가 여러 단계에서 발생할 수 있고 같은 오버라이드가 여러 프리팹에 적용(“Apply”)될 수 있다. 다음 그림과 같이 프리팹 “Table”안에 프리팹 “Vase”가 중첩되었다고 가정하자. “Vase”의 속성을 오버라이드하면 오버라이드를 반영(“Apply”)할 대상은 프리팹“ Table”과 프리팹 “Vase”이다. “Apply All” 버튼은 오버라이드를 “Table”에 적용한다.



프리팹 인스턴스의 인스펙터 뷰에서 오버라이드 컨텍스트 메뉴 “Apply”를 사용할 때 다음 그림과 같이 타겟을 선택할 수 있다. “Apply to Prefab ‘Vase’”를 선택하면 프리팹 “Vase”에 오버라이드가 적용되고, 프리팹 “Vase”의 모든 인스턴스에 반영된다. “Apply as Override in Prefab ‘Table’”를 선택하면 프리팹 “Table”안에 있는 프리팹 “Vase”에 오버라이드가 적용된다.

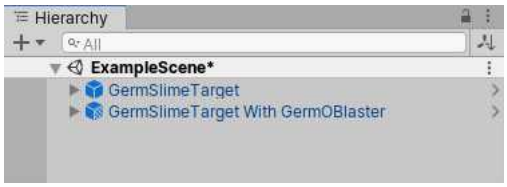


- 프리팹 변종(Prefab Variants)

  프로젝트 뷰에서 오른쪽 마우스 버튼으로 프리팹 애셋을 선택하고 “Create ▶ Prefab Variant”를 선택하거나 또는 메뉴 “Assets ▶ Create ▶ Prefab Variant”를 선택하여 프리팹 변종을 생성할 수 있다.

  또는 계층 뷰에서 프리팹 인스턴스를 선택하여 프로젝트 뷰로 드래그&드랍하면 프리팹 변종을 생성할 것인가를 묻는 대화상자가 다음과 같이 나타난다. “Prefab Variant” 버튼을 선택하면 프리팹 변종을 생성한다.

프리팹 변종이 생성되면 다음 그림과 같이 계층 뷰에서 선택한 프리팹 인스턴스와 생성된 프리팹 변종의 아이콘에 빗금 표시가 나타난다.



프리팹 변종은 하나의 프리팹 애셋에서 다른 프리팹 애셋을 생성하는 것이다. 개념적으로는 C++ 언어의 클래스 상속(파생 클래스 생성)과 유사한다. 여러 개의 프리팹 애셋들의 공통적이고 기본적인 내용으로 기반 프리팹 애셋(Base Prefab)을 생성하고 이를 기반으로 다른 프리팹 애셋을 생성하는 것이다. 프리팹 변종은 기반 프리팹 애셋의 내

용을 상속받는다. 프리팹 변종을 오버라이드하여 기반 프리팹 애셋과 다르게 구성할 수 있다.

계층 뷰를 통하여 프리팹 모드에서 프리팹 변종을 편집할 때, 다음 그림과 같이 루트에 기반 프리팹이 표시된다. 프리팹 변종에서 편집한 내용은 기반 프리팹을 오버라이드한 것이다.

- 프리팹 인스턴스의 프리팹 연결 끊기(Unpacking Prefab Instance)

  계층 뷰에서 프리팹 인스턴스를 오른쪽 마우스 버튼으로 선택하여 “Unpack Prefab”을 선택하면 프리팹 인스턴스와 프리팹 애셋의 연결을 끊을 수 있다. 프리팹 인스턴스와 프리팹 애셋의 연결이 끊어지면 프리팹 인스턴스는 보통의 게임 객체가 되고 프리팹 애셋의 수정에 영향을 받지 않는다.

  “Unpack Prefab Completely“를 선택하면 그 프리팹 인스턴스가 포함하고 있는 모든 프리팹 인스턴스와 프리팹 애셋의 연결을 끊는다.

- 프리팹을 사용하여 경우 2를 해결

  “ExplosiveCube” 스크립트를 포함하는 큐브 게임 객체를 프리팹 애셋으로 생성하고, 큐브 게임 객체는 제거한다. 구 게임 객체에 연결하고 인스펙터 뷰에서 “Explosion” 컴포넌트의 “Cube”의 값으로 큐브 프리팹 애셋을 연결하자.