UNITY 3D
-CHAPTER8-

SOULSEEK

목차

UGUI

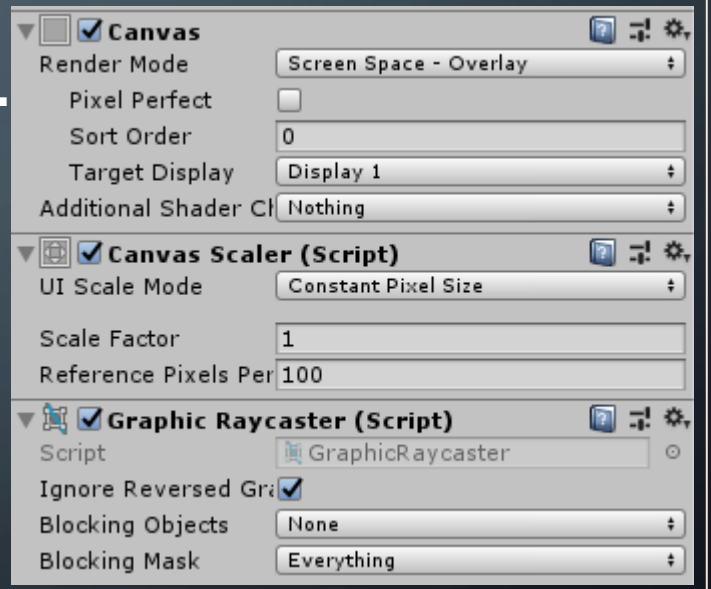
1. UGUI

- **UNITY**에서 제공하는 **UI** 오브젝트 관리 시스템**.**
- **NGUI**의 기능을 많이 가지고 있다**.**
- **NGUI**와 같이 파티클 렌더링 소트 문제가 있다**.**
- **Hierarchy**상의 계층 순서대로 **Depth**를 처리하기 때문에 **NGUI**보다 신경을 덜 쓸 수 있다**.**
- **Canvas, Image, Text**를 알아보자**.**
- **Script**에서 **UnityEngine.UI;**를 선언하고 **Code**에서 컨트롤 할 수 있다

1. UGUI

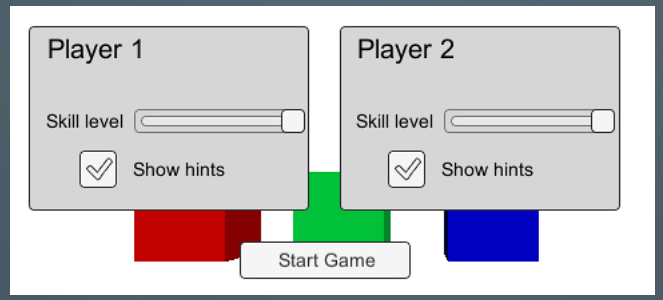
- **Canvas**
  - 모든 **UI** 요소를 배치하기 위한 영역**. Canvas** 요소와 함께 사용하는 게임 오브젝트로**,** 모든 **UI** 요소는 **Canvas**의 자식 요소여야 한다**.**
  - **UI**요소들을 그룹화하는 요소**, UI**요소들은 **Canvas**위에 존재해야 한다**.**
  - 한 씬에서 여러개의 **Canvas**가 존재할 수 있다**.**
  - **UI**요소들은 **Canvas**의 자식으로 존재한다**.**

- **Render Mode :**
  - **Screen Space Overlay -** 렌더링 되는 것들의 위에 **UI**가 옴**.** 강제로 화면 전체를 렉트로 채우기 때문에 **RectTransform**을 컨트롤 할 수 없다**.**
  - **Screen Space Camera -** 씬의 특정 카메라에 투영 시킬 때 사용한다**.**나머지는 **Overlay**랑 같다**.**
  - **World Space :** 씬 볼륨의 엘리먼트를 랜더링 한다**.** 다른 오브젝트 처럼 똑같이 관리한다**.**
- **Pixel Perfect :** 가장 가까운 픽셀로 조정**.** 렌더모드에서 **Screen Space**를 선택했을 때 나타난다**.**

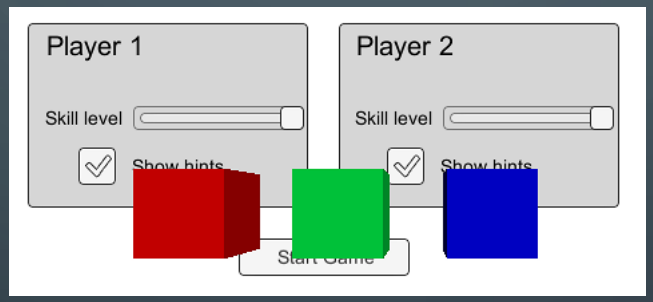


1. UGUI

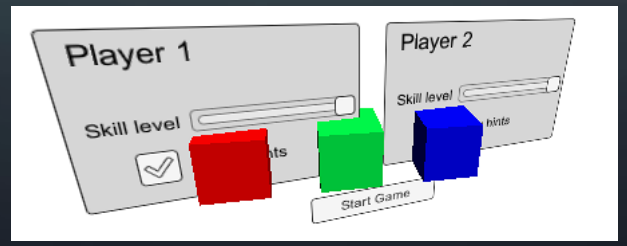
- **Screen Space – Overlay**



- **Screen Space – Camera**



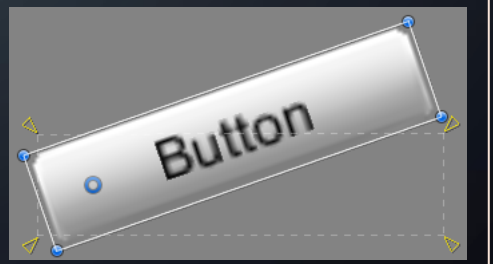
- **World Space**



1. UGUI

- **RectTransform**
  - **Transform**과 다른 직사각형 형태의 **UI**를 위한 **Transform Component**
  - **Pivot Point, Width, Height + Position, Rotate, Scale**
  - **Canvas**를 생성하게 되면 **RectTransform**은 자동으로 추가된다.
  - **Anchor**개념이 포함되어 있다.
  - **UI**요소들은 생성할 때마다 **RectTransform**을 가지고 있다.

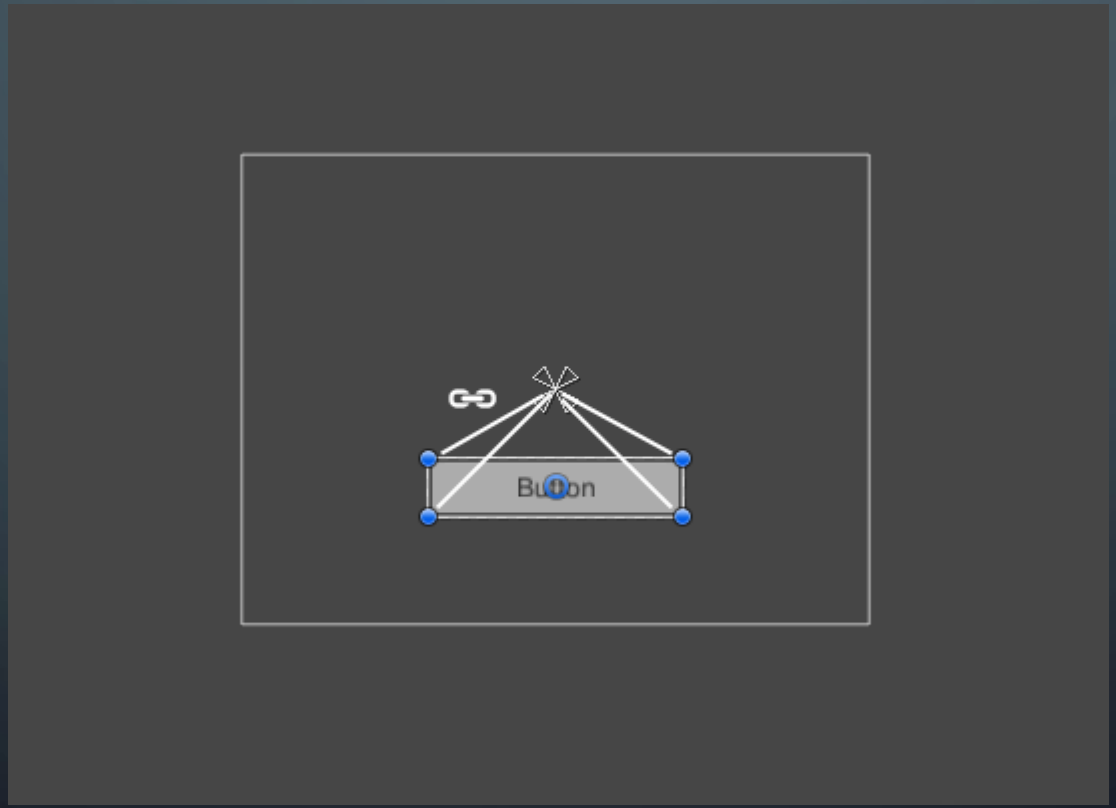
- **리사이징 VS 스케일링**
  - **Rect Tool**이 오브젝트의 크기 변경에 사용되는 경우, **2D** 시스템의 **Sprite**와 **3D** 오브젝트를 위해 일반적으로 오브젝트의 로컬 _**scale**_을 변경한다. 그러나 **Rect Transform** 컴포넌트가 연결된 오브젝트의 경우, 로컬 **scale**은 변경하지 않은 채 **width**와 **height**를 변경합니다. 이 리사이징은 글꼴 크기, 슬라이스 된 이미지의 경계선 등에 영향을 주지 않는다.



- **피벗(Pivot)**
  - 회전**(rotation)**, 크기**(size)**, 스케일**(scale)**의 수정은 피벗 주위에서 발생하므로 피벗의 위치는 회전, 리사이징 스케일링의 결과에 영향을 준다. 툴바의 **Pivot** 버튼이 **Pivot** 모드로 설정되어 있으면 **Rect Transform** 피벗은 **Scene View**에서 이동시킬 수 있다.

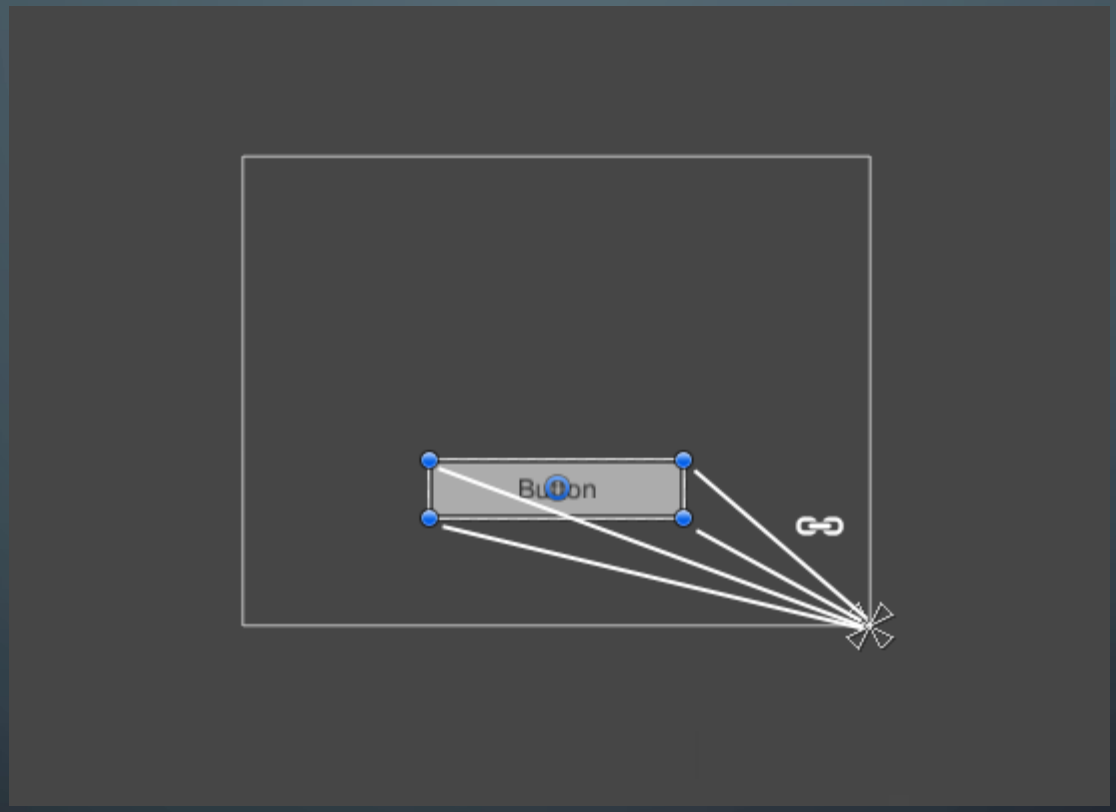
1. UGUI

- **Anchor**
  - **Rect Transform**은 **anchors**라는 레이아웃의 개념이 있다**.** 앵커는 **4**개의 작은 삼각형 핸들로 **Scene View**에 표시되고 앵커의 정보는 인스펙터에 표시된다**.**
  - **Rect Transform**의 부모도 **Rect Transform**이면 자식의 **Rect Transform**은 다양한 방법으로 부모의 **Rect Transform**에 고정할 수 있다**.** 예를 들어**,** 자식은 부모의 중심 또는 모서리 중 하나에 고정할 수 있다**.**



부모의 중심에 고정된 UI 요소. 요소는 중심에 대해 고정 오프셋을 유지하고 있다.

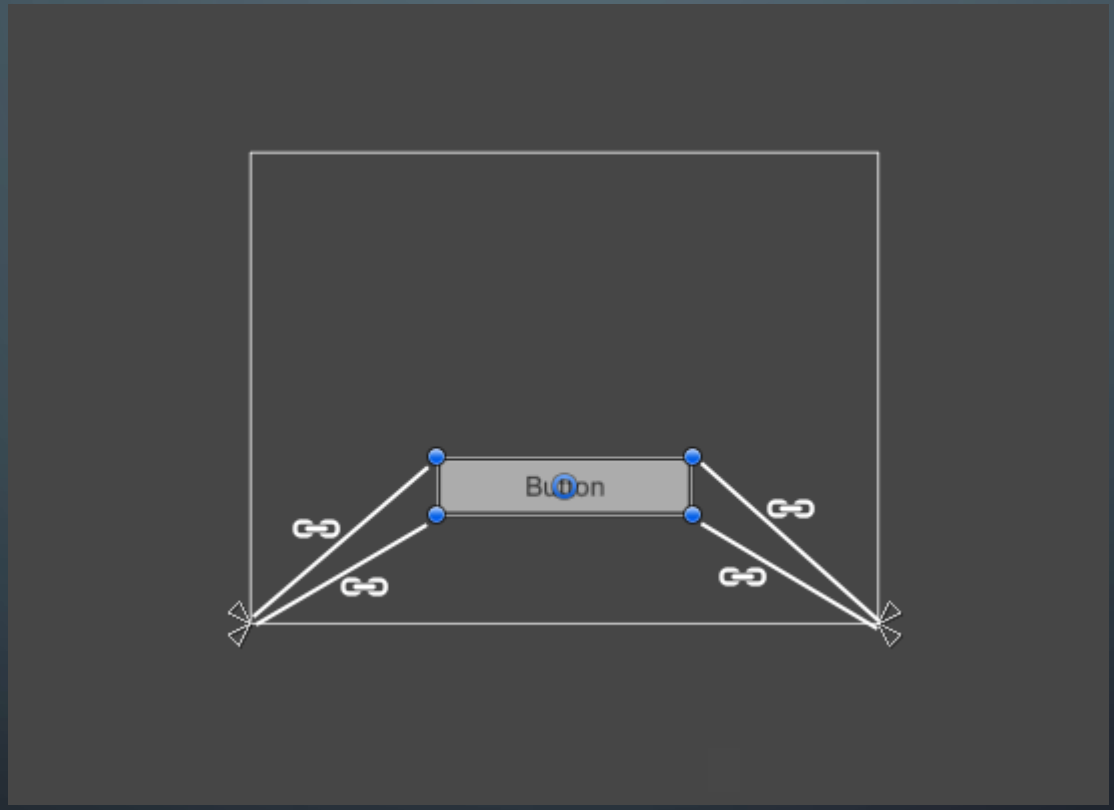
1. UGUI



부모의 오른쪽 하단에 고정된 UI 요소. 요소는 오른쪽 하단에 고정 오프셋을 유지하고 있다.

1. UGUI

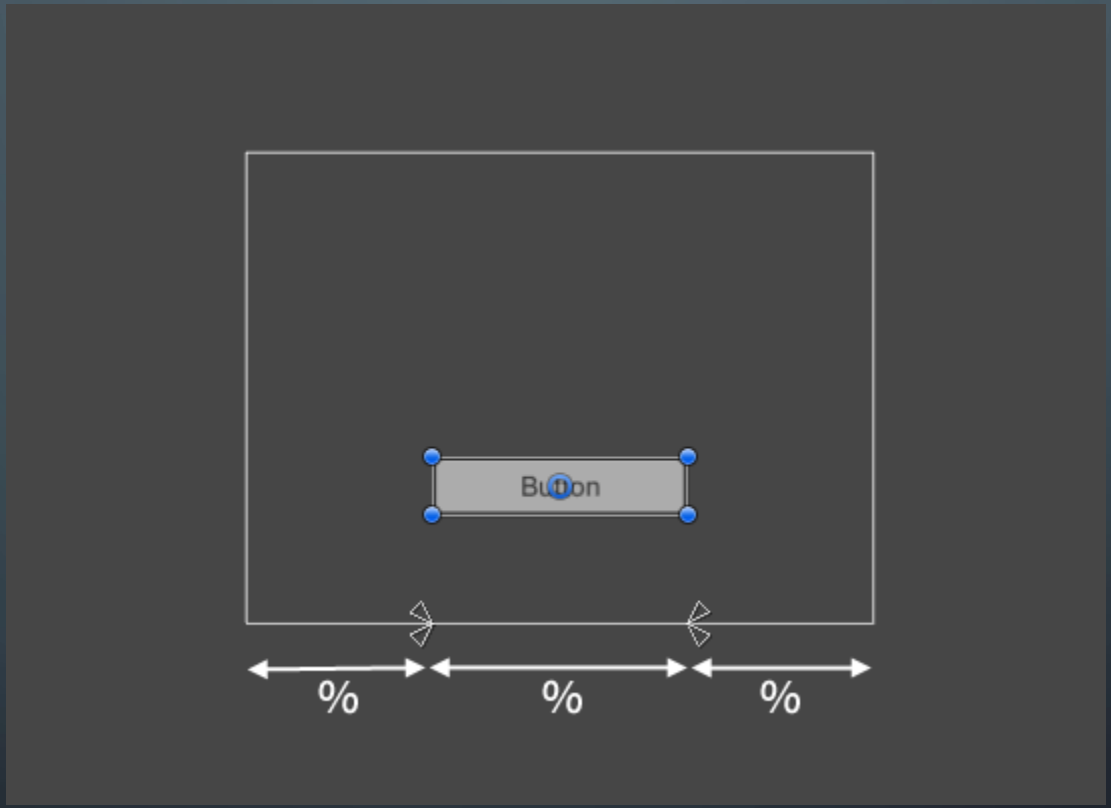
- 고정**(anchoring)**은 부모의 **width** 또는 **height**와 함께 자식을 늘리는 것을 허용한다**.** 사각형의 각 모서리는 해당 앵커에 고정 오프셋을 가지고 있다**.** 즉**,** 구형의 왼쪽 상단 모서리는 왼쪽 상단 앵커에 고정 오프셋을 갖는 것이다**.** 이와 같이**,** 사각형의 다른 모서리를 부모 사각형의 다른 점에 고정할 수 있다**.**



왼쪽 모서리를 부모 사각형의 왼쪽 아래 오른쪽 모서리가 오른쪽 하단 모서리에 고정된 UI 요소. 요소의 모서리는 각각의 앵커에 고정된 오프셋을 유지합니다.

1. UGUI

- 앵커의 위치는 부모 구형의 **width**와 **height**를 분수(또는 퍼센트)로 정의한 것이다**. 0.0(0 %)**은 왼쪽 또는 하단에**, 0.5(50 %)**은 중간에**,** 그리고 **1.0(100 %)**은 오른쪽 또는 상단에 대응하고 있다**.** 그러나 앵커는 끝이나 중간에 제한되는 것은 아니다**.** 그들은 부모 사각형 내의 어떠한 지점에도 고정할 수 있다**.**

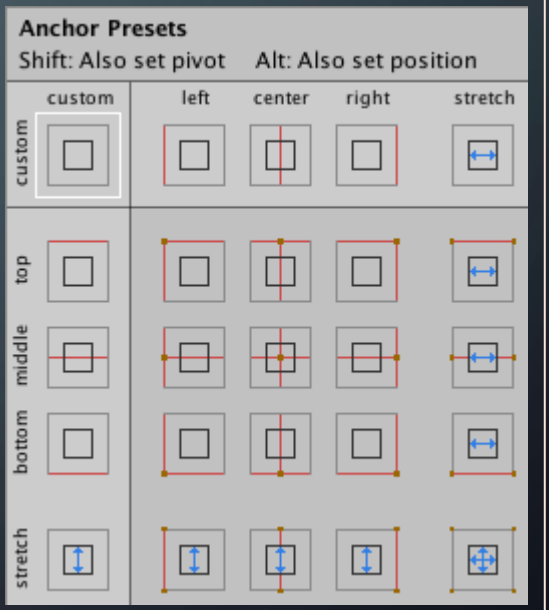


왼쪽 모서리를 부모 사각형의 왼쪽에서 특정 백분율만큼 떨어뜨려 고정하고, 오른쪽 모서리를 부모 사각형의 오른쪽 맨 끝에서 특정 백분율만큼 떨어뜨려 고정한 UI 요소.

1. UGUI

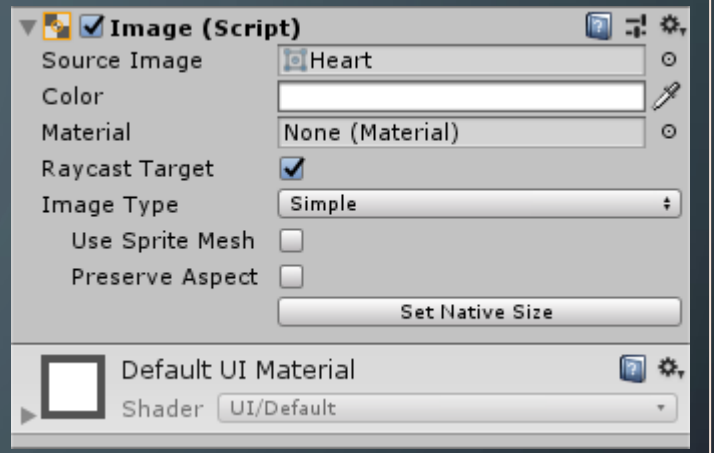
- 각각의 앵커를 개별적으로 드래그 할 수 있고 만약 함께 있으면, 그 중간을 클릭하고 드래그하여 함께 드래그 할 수 있다. 앵커를 Shift 키를 누른 상태로 드래그하면 사각형에 대응하는 코너는 앵커와 함께 움직인다.

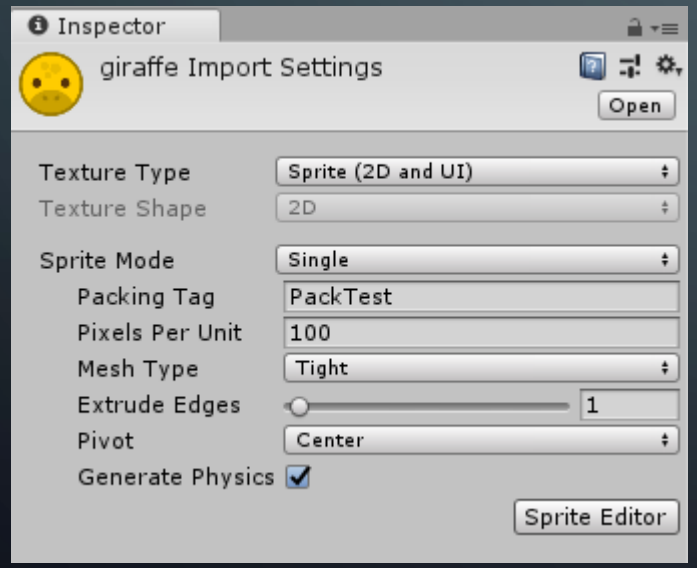
- **Anchor presets**
  - 인스펙터에서는 **Anchor Preset** 버튼이 **Rect Transform** 컴포넌트의 왼쪽 위의 모서리에 있다**.** 버튼을 클릭하면**,** 앵커 프리셋이 표시된다**.** 여기에서 바로 몇 가지 가장 일반적인 앵커 옵션 중 하나를 선택할 수 있고 **UI** 요소를 부모의 가장자리 또는 중간에 고정하거나 부모의 크기에 따라 늘릴 수 있다**.** 수평 및 수직 고정은 별도**.**
  - **Anchor Presets** 버튼은 선택되어 있는 것이 하나 있다면**,** 현재 선택되어 있는 프리셋 옵션을 표시한다**.** 수평 또는 수직 축에서 앵커가 프리셋 중 하나와 다른 위치에 설정되어 있는 경우**,** 사용자 지정 옵션이 표시된다**.**



1. UGUI

- **Image**
- **Source Image : 2DUI Sprite** 타입의 **Image**를 링크한다.
- **Color :** 지정한 컬러 값을 **Image**에 준다.
- **Material :** 특정한 셰이더를 추가하고 싶을 때 쓴다.
- **Image Type : Simple** – 원래 형식 그대로.
  - **Sliced - 9**등분한 곳 중 가운데만 크기만큼 늘려주고 싶을 때.
  - **Tiled - Image**를 타일배열로 크기만큼 채워준다.
  - **Filed -** 이미지를 수치에 따라 채워주고 싶을 때.

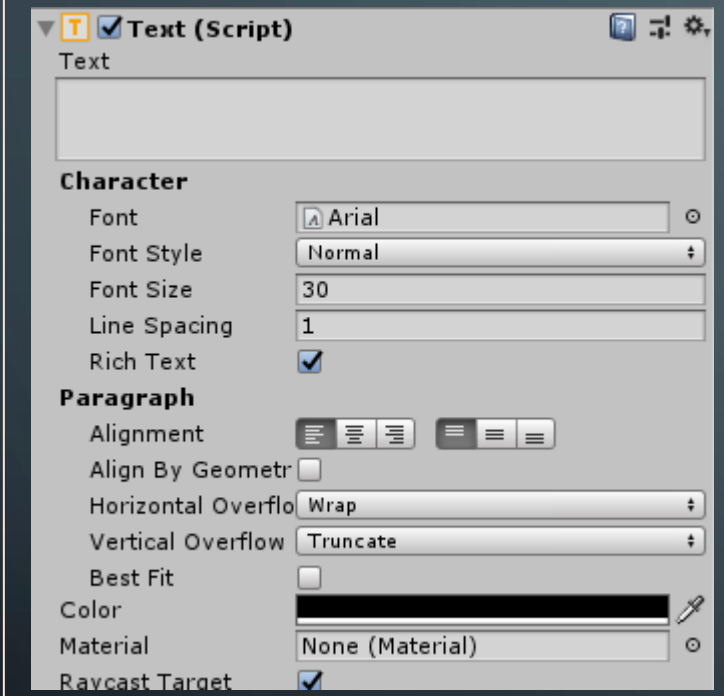




Image들은 저장된 형태에 따라 인식되는 Type이 있는데 Sprite형식으로 저장되어 있지 않을 것입니다. UNITY Sprite로 쓸 수 있는 형식으로 변경하는 것이 중요하다. Texture Type를 눌러서 보이는 것같이 Sprite(2D and UI)로 변경 시켜줘야 한다.

1. UGUI

- **TEXT**



- **Text :** 표시할 문자열
- **Font :** 설정하고 싶은 문자열의 폰트, 추가폰트는 **Asset**에 있어야 한다.
- **FontStyle :** 폰트스타일을 지정한다.
- **Line Spacing :** 줄 간격.
- **Font Size : Font** 크기
- **Rich Text :** 서식이 있는 문자열 지원 **exhello** 굵은 글씨 **<color =#ff0000ff>**내가**</color>** 색깔변경.
- **Alignment :** 정렬 형태.
- **Horizontal Overflow :** 수평으로 넘어 갈 때, 어떻게 처리할 것인지 설정. **Wrap** – 다음 줄로, **Overflow** 넘어가게 둔다.
- **Vertical Overflow :** 라인을 넘어 갔을 때, 어떻게 처리할 것인지 설정.
- **Truncate -** 영역을 넘어가면 보이지 않게, **Overflow -** 영역을 넘어간 글자도 보이도록 할 때

NGUI(NEXT - GEN USER INTERFACE)

2. NGUI

유니티 엔진의 강력한 UI 미들웨어

유니티가 지원하는 기본 GUI 시스템으로는 UI를 구성하기가 까다롭고 성능 면에서도 좋지 않다.
그래서 유니티를 이용하는 개발자들이 가장 많이 찾는 미들웨어이다.
사용하기 편리하고, 많은 기능들을 제공한다. C#으로 구성되어 있기 때문에 C#을 알고 있다면 이해하기 쉽고 필요에 따라서는 커스텀해서 쓸 수도 있다.
하지만 NUGI는 유료구매를 해야 하고 $95(10만원 정도)하고 있다. 현재는 3.x 버전이 나와있고 업데이트가 꾸준히 진행되고 있으며 2.7버전이 무료로 배포되고 있다 NGUI의 제작사인 Tasharen(http://www.tasharen.com)이나 유니티 AssetStore에서 구매가 가능하다.

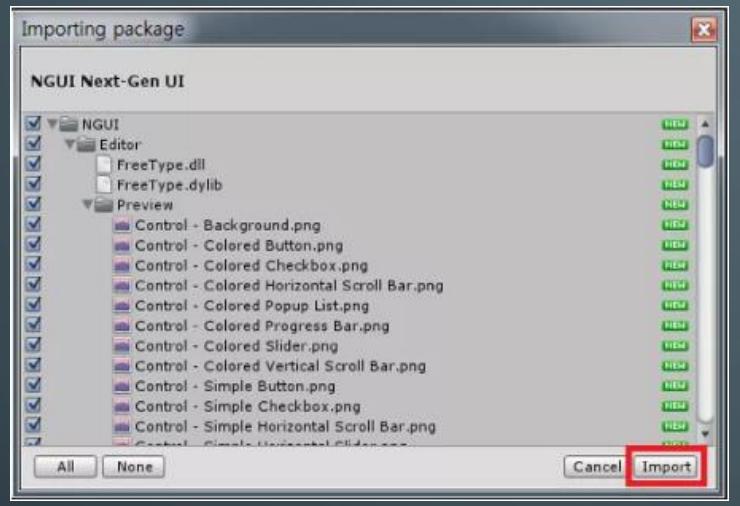
UI 미들웨어라는걸 항상 인지하자.

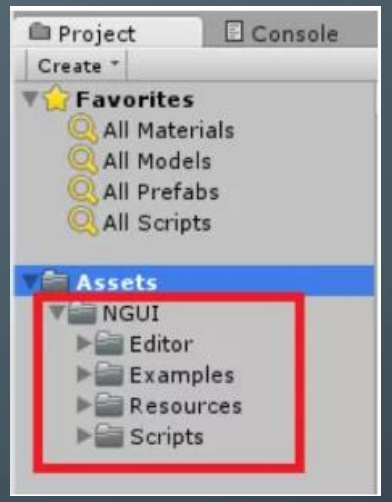
NGUI의 풀 네임을 보면 **User Interface**라는 말이 있다. 말 그대로 **UI.** 그렇기 때문에 **UI**를 구성하라고 만들어 진 것이다. 하지만 **2D**게임을 개발하는 개발자들의 대부분이 **UI**이외의 구성요소에도 이를 적용시켜 많은 성능적 저하나 **Render Sort**에서 고생하는 실수를 많이 일으킨다 **UI**는 **UI**로만 사용하자.

2. NGUI

Import

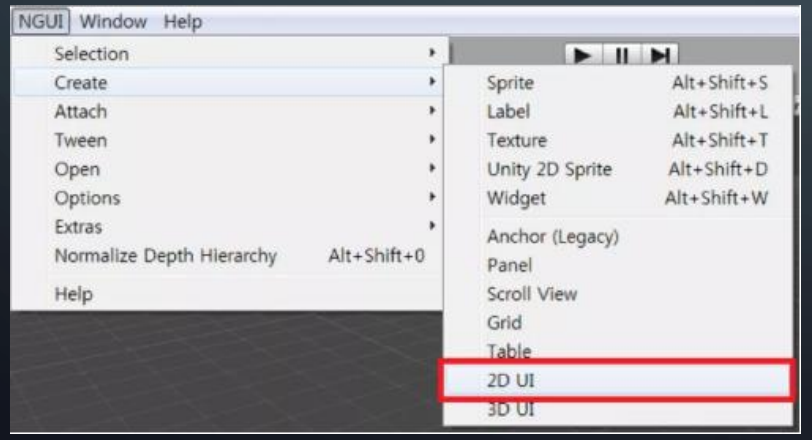
Asset Store에서 구입절차를 끝낸 후 **Import**하면 **package Import**창이 뜬다. 거기서 **Import**버튼을 누르면 **NGUI**가 설치된다. **NGUI**폴더 자체를 해당 프로젝트 폴더로 복 붙해도 알아서 인식하고 설치하므로 한번 받았다면 이 방법을 이용해도 된다.

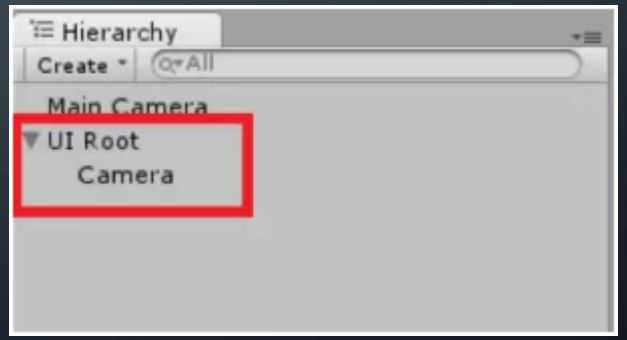




UI 생성

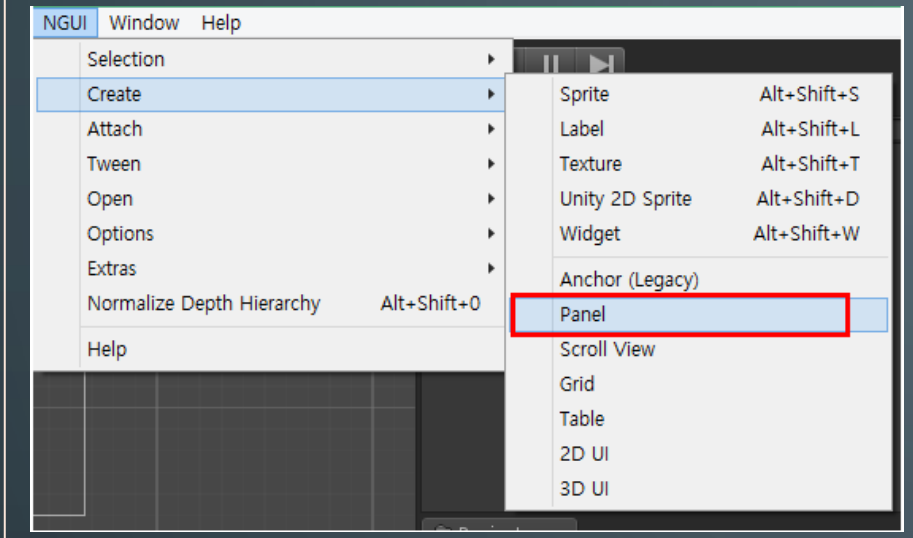
UI를 기본 구조를 생성해서 프로젝트 폴더에 배치해준다.

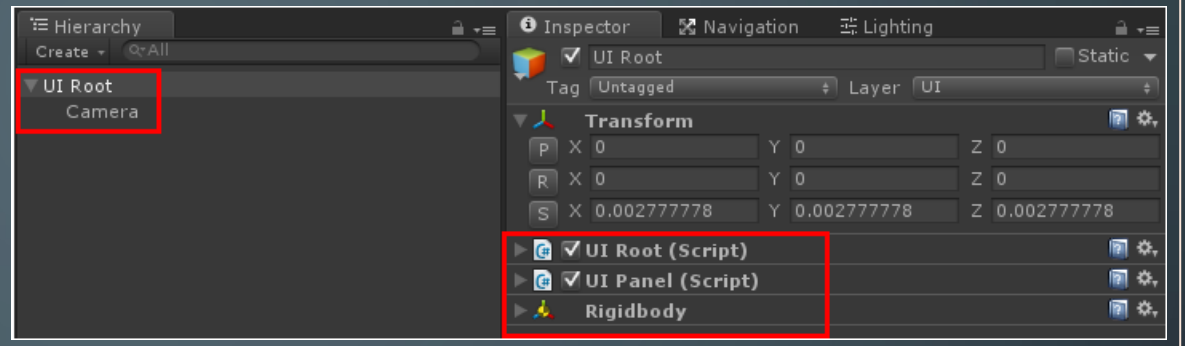




2. NGUI

Panel





NGUI의 기본은 Panel

- **NGUI**를 구성하는 **Component**들은 모두 **Panel**의 자식 **Object**들로 존재하기 때문에 **Panel**이 없으면 그려질 수 없다. 모든 **NGUI**의 작업이전에는 **Panel**을 생성해야 한다.

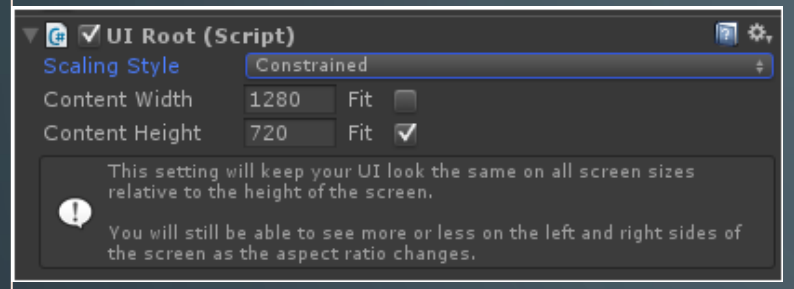
UI Root

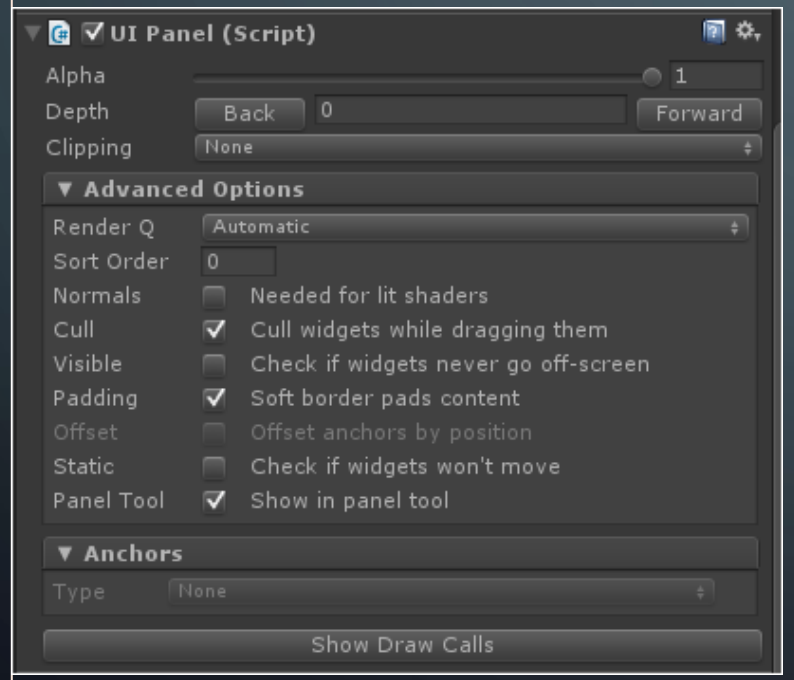
- **UI** 생성을 통해 만들어두지 않았더라도 **NGUI** 구성요소를 생성하게 되면 기본적으로 **Root**와 **Camera**가 필요하기 때문에 자동적으로 생성된다.

기본 구성요소

UI Root, UI Panel, Rigidbody + Camera

2. NGUI





- **Flexible :** 항상 같은 픽셀 수 유지.

- **Constrained :** 지정한 해상도로 보여준다.
- **ConstrainedOnMobiles : PC**에선 **Flexible,** 모바일에선 **Constrained**로.

- **Alpha :** 패널 전체의 투명도를 설정 할 수 있다. 속한 모든 구성요소에 영향을 준다.
- **Depth :** 패널간의 **Sort** 순서를 알려준다.
- **Clipping :** 패널범위가 클리핑 영역이 된다.
- **RenderQ : RenderQ** 값의 적용 형태를 설정한다.
- **Normals : UI**에 사용하는 **Mesh Normal** 값을 계산한다.
- **Cull :** 패널이 드래그되는 동안 패널에 속한 위젯들의
- 렌더링을 끌 수 있다.
- **Visible :** 패널에 속한 **UI**가 화면 밖에 벗어나지 않도록 할 때 사용
- **Static :** 패널에 속한 위젯이 움직이지 않는 경우에만 사용
- **Panel Tool : Scene**에 있는 모든 패널을 확인하고 선택할 수있는 패널 관리 도구.

2. NGUI

UICamera

- 각 **UI**위젯에서 발생하는 이벤트와 관련된 메시지를 보내는 역할을 한다**. OnClick / OnHover**라는 이벤트를 자주 사용한다**.** 두개 이상의 카메라를 이용해서 **UI**를 구성할 수도 있다**.**

- **Event Type :** 타입설정에 따라 **UI**혹은 월드상의 오브젝트와 상호작용할 것인지를 설정**.**
- **Event Mask :** 이벤트를 받아들일 레이어를 지정한다**.**
- **Event go to.. :** 이벤트를 전달할 컴포넌트를 설정**.**
- **Debug :** 디버그 모드의 사용 여부를 설정**.**
- **Allow Multi Touch :** 멀티터치 입력의 허용 여부를 설정한다**.**
- **Stick Tooltip : Sticky Tooltip**의 사용 여부를 설정**.**
- **Tooltip Delay :** 툴팁이 표시되기 위한 시간을 지정한다**.**
- **Raycast Range :** 카메라에서 이벤트를 감지할 때 사용되는 **Raycast**의 범위를 지정**.**
- **Event Sources : NGUI**가 이벤트를 처리 할 입력소스를 지정**.**
- **Threshould :** 특정이벤트가 발생하기 위한 **threshould**값 즉**,** 최소값을 설정한다**.**
- **Axes and Keys :** 축과 키보드를 **NGUI**의 입력시스템으로 인식하는데 사용

2. NGUI

Atlas

Texture들을 모아놓은 커다란 Texture

수많은 **Texture**들을 사용하는데 개별적으로 사용하고 관리하는 것은 성능적으로나 관리적으로나 비효율적이다**.** 따라서 **Texture**들을 모아서 하나의 **Atlas**로 만든다**.** 하나의 **Meterial**만 사용 할 수 있다**.** 드로우 콜과 **Batch**를 줄이고 성능 향상에 도움을 준다**.**