게임엔진프로그래밍응용

7. 2D 러너 2

청강문화산업대학교 게임콘텐츠스쿨

반 경 진

코로나 유의사항

코로나 유의 사항

- 2023학년도 1학기 방역 및 학사운영 방안
  https://www.ck.ac.kr/archives/193175
- 2023학년도 1학기 국가공휴일 및 대학 행사 수업 대체 일정 공지
  https://www.ck.ac.kr/archives/193109

온라인 수업 저작권 유의 사항

온라인 수업 저작권 유의 사항

온라인수업 저작권 유의사항 안내

강의 저작물을 다운로드, 캡처하여
교외로 유출하는 행위는
불 법 입 니 다

저작권자의 허락 없이 저작물을 복제, 공중송신 또는 배포하는 것은
저작권 침해에 해당하며 저작권법에 처벌받을 수 있습니다.

강의 동영상과 자료 일체는 교수 및 학교의 저작물로서 저작권이 보호됩니다.
수업자료를 무단 복제 또는 배포, 전송 시 민형사상 책임을 질 수 있습니다.

Index

UGUI

UGUI

- **UNITY UI**
  com.unity.ugui

  1. Unity UI는 게임 및 애플리케이션용 사용자 인터페이스를 개발하기 위한 UI 툴킷입니다. 구성 요소와 게임 보기를 사용하여 사용자 인터페이스를 정렬, 위치 지정 및 스타일 지정하는 GameObject 기반 UI 시스템입니다

  2. Unity UI를 사용하여 Unity 에디터에서 사용자 인터페이스를 생성하거나 변경할 수 없습니다

UGUI

- **Canvas**

1. Canvas 는 모든 UI 요소가 있어야 하는 영역입니다 . Canvas는 Canvas 구성 요소가 있는 게임 개체이며 모든 UI 요소는 이러한 Canvas의 자식이어야 합니다

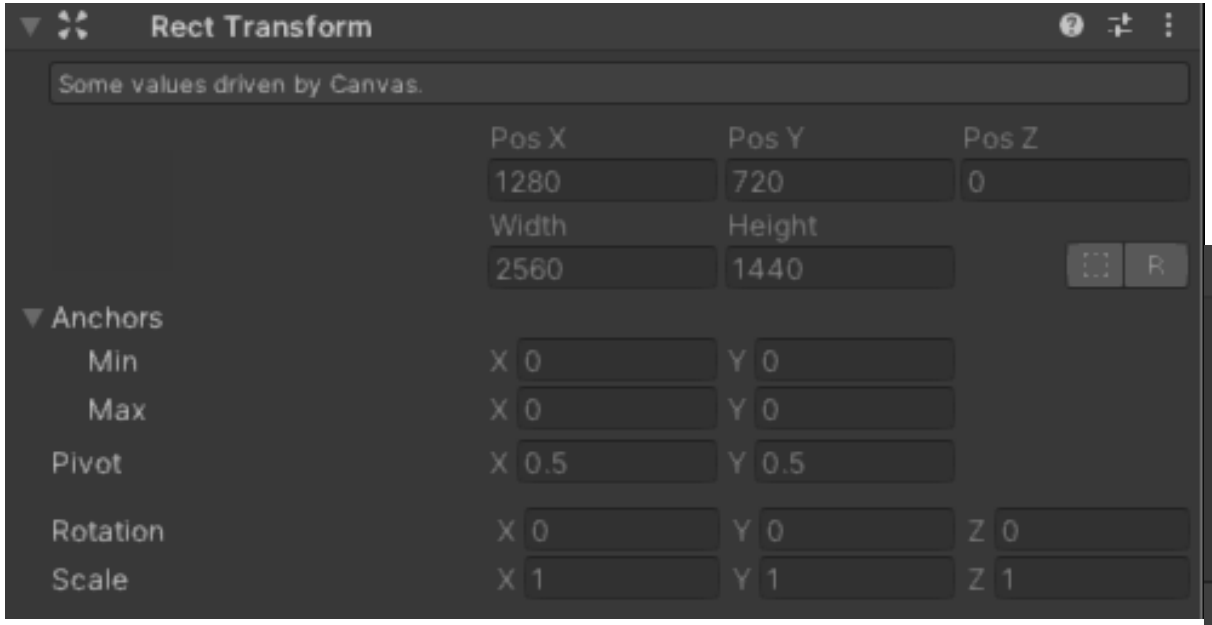
2. Canvas는 다른 게임 오브젝트와 동일한 방식으로 생성합니다. 계층 창에서 빈 공간을 마우스 오른쪽 단추를 클릭하거나 계층 창 맨 위의 Create 버튼을 클릭한 후 UI > Canvas를 선택하면 됩니다

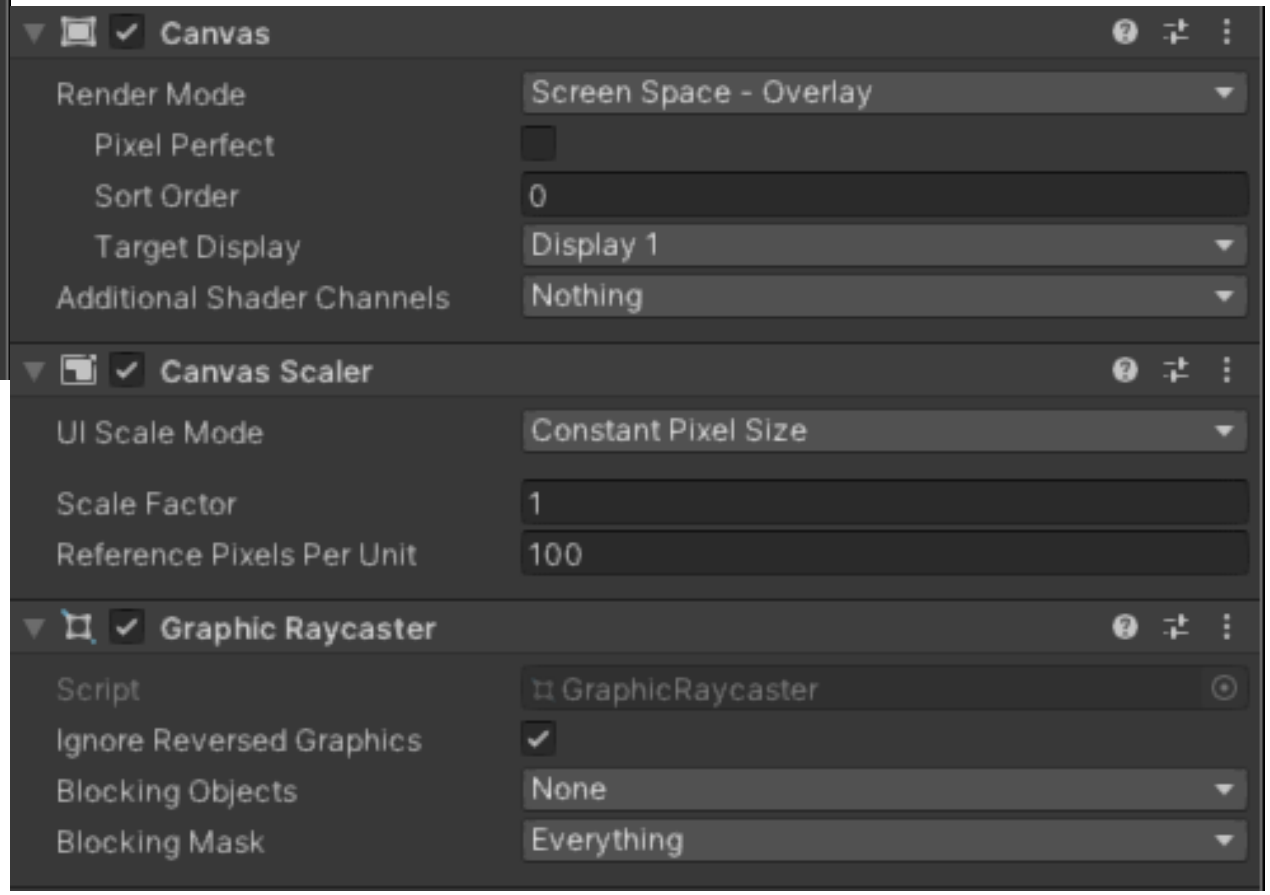
3. UI > Image 메뉴를 사용하여 Image와 같은 새 UI 요소를 생성하면 장면에 Canvas가 아직 없는 경우 자동으로 Canvas가 생성됩니다. UI 요소는 이 Canvas의 자식으로 생성됩니다

4. 캔버스 영역은 장면 보기에서 사각형으로 표시됩니다. 이렇게 하면 게임 보기를 항상 표시할 필요 없이 UI 요소를 쉽게 배치할 수 있습니다.

5. Canvas를 생성할 경우 씬에는 EventSystem이라고 하는 다른 게임 오브젝트도 함께 추가됩니다. 이 게임 오브젝트에는 UI를 사용한 이벤트 및 상호 작용(예: 마우스 클릭)을 처리하는 특수 컴포넌트가 포함되어 있습니다. EventSystem은 사용하지 않더라도 씬 안에 그대로 두어야 하며, 그러지 않으면 Canvas에서 경고를 기록합니다.

UGUI


Rect Transform
Some values driven by Canvas.
Pos X
Pos Y
Pos Z
1280
720
0
Width
Height
2560
1440
R
Anchors
Min
X 0
Y 0
Max
X 0
Y 0
Pivot
X 0.5
Y 0.5
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1


Canvas
Render Mode
Screen Space - Overlay
Pixel Perfect
Sort Order
0
Target Display
Display 1
Additional Shader Channels
Nothing
Canvas Scaler
UI Scale Mode
Constant Pixel Size
Scale Factor
1
Reference Pixels Per Unit
100
Graphic Raycaster
Script
GraphicRaycaster
Ignore Reversed Graphics
Blocking Objects
None
Blocking Mask
Everything

UGUI

- **Rect Transform**

1. Canvas 게임 오브젝트의 가장 큰 차이점은 다른 게임 오브젝트에서 볼 수 있는 Transform 대신에 Rect Transform 컴포넌트가 있다는 점입니다

2. Rect Transform은 역시 Transform에 해당하므로 스크립트에서 이를 Transform으로 활용할 수 있으나, 추가적 UI 데이터도 포함하고 있습니다. 특수한 종류의 Transform으로 간주하시면 됩니다.

3. Rect Transform 컴포넌트는 Transform 컴포넌트의 2D 레이아웃 버전입니다. Transform은 포인트 하나를 나타내지만, Rect Transform은 UI 요소를 안에 넣을 수 있는 Rect을 나타냅니다. Rect Transform의 상위 컴포넌트도 Rect Transform인 경우, Rect Transform이 상위 Rect에 대한 상대적인 포지션과 크기도 지정할 수 있습니다

4. 일부 RectTransform 계산은 프레임 전체에서 수행된 최신 변경 사항이 반영되도록 하기 위해 UI 버텍스를 계산하기 직전 프레임 끝에서 수행됩니다. 따라서 첫 Start() 콜백과 첫 Update() 콜백에서는 아직 계산되지 않은 상태입니다. 이 문제는 Start() 콜백을 만들고 여기에 Canvas.ForceUpdateCanvases() 메서드를 추가하여 우회할 수 있습니다. 그러면 캔버스가 프레임 끝에서 업데이트되지 않고 메서드가 호출될 때 업데이트됩니다.

UGUI

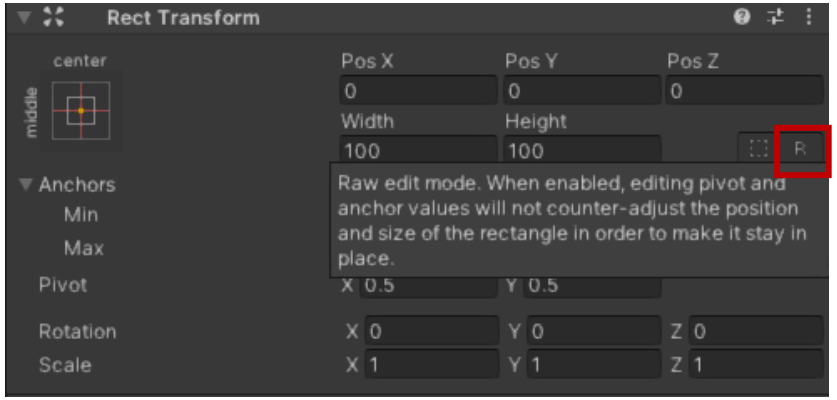
- Rect Transform Properties

Pos (X, Y, Z)	앵커를 기준으로 한 사각형의 피벗 포인트 포지션입니다. 피벗 포인트 주변으로 사각형이 회전합니다.
Width/Height	직사각형의 너비와 높이입니다.
Left, Top, Right, Bottom	앵커를 기준으로 한 사각형 에지의 상대적인 포지션으로, 앵커에 의해 정의되는 사각형 안의 패딩이라고 생각할 수 있습니다. 앵커가 분리된 경우에 *Pos* 와 *Width/Height* 대신 표시됩니다(아래 참조). 이 옵션에 액세스하려면 RectTransform 컴포넌트 왼쪽 상단에 있는 Anchor Presets 상자를 클릭하십시오.
Anchors	사각형 왼쪽 하단 모서리와 오른쪽 상단 모서리의 앵커 포인트입니다.
Min	상위 사각형 크기의 일부로 정의되는 사각형 왼쪽 하단 모서리의 앵커 포인트입니다. 0,0은 상위 사각형 왼쪽 하단 모서리의 앵커에 해당하고 1,1은 상위 사각형 오른쪽 상단 모서리의 앵커에 해당합니다.
Max	상위 사각형 크기의 일부로 정의되는 사각형 오른쪽 상단 모서리의 앵커 포인트입니다. 0,0은 상위 사각형 왼쪽 하단 모서리의 앵커에 해당하고 1,1은 상위 사각형 오른쪽 상단 모서리의 앵커에 해당합니다.

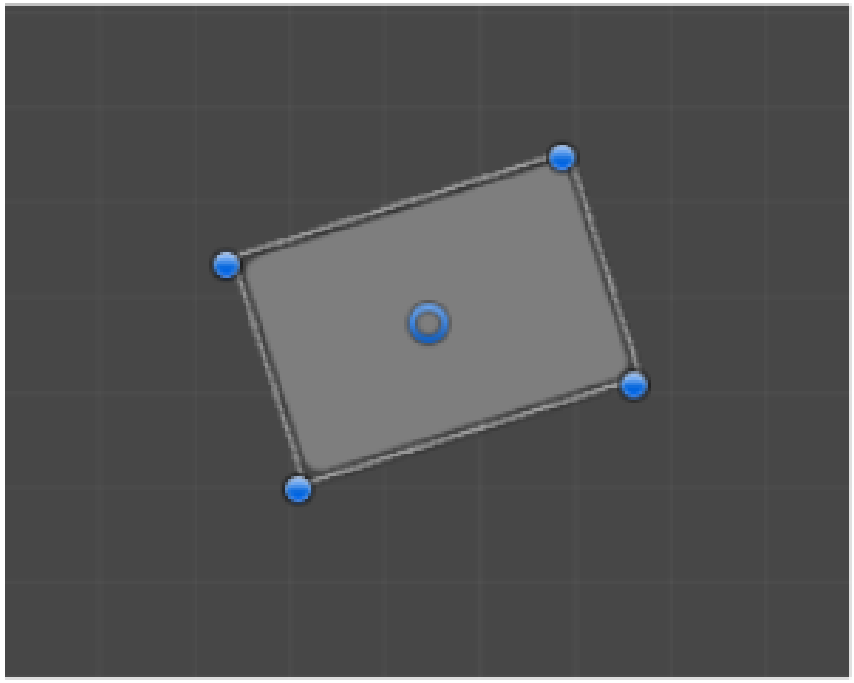
UGUI

- Rect Transform Properties

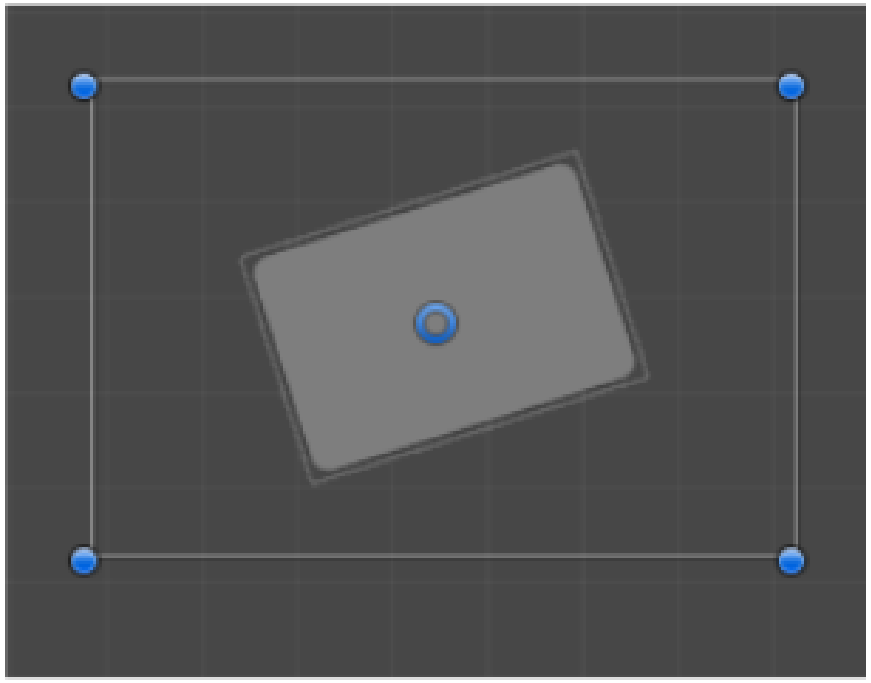
Pivot	피벗 포인트 주위로 사각형이 회전하며, 사각형 자체 크기의 일부로 정의됩니다. 0,0은 왼쪽 하단 모서리에 해당하고 1,1은 오른쪽 상단 모서리에 해당합니다.
Rotation	X, Y 및 Z 축을 따라 피벗 포인트를 중심으로 회전하는 오브젝트의 회전 각도(° 단위)입니다.
Scale	X, Y 및 Z 차원에서 오브젝트에 적용되는 스케일 팩터입니다.
Blueprint Mode	회전되고 스케일되지 않는 것처럼 RectTransform을 편집합니다. 이 경우 스내핑도 활성화됩니다.
Raw Edit Mode	활성화된 경우 피벗 및 앵커 값을 편집하면 사각형이 한 자리에 머무르도록 사각형의 포지션과 크기를 반대로 조정하지 않습니다.



UGUI

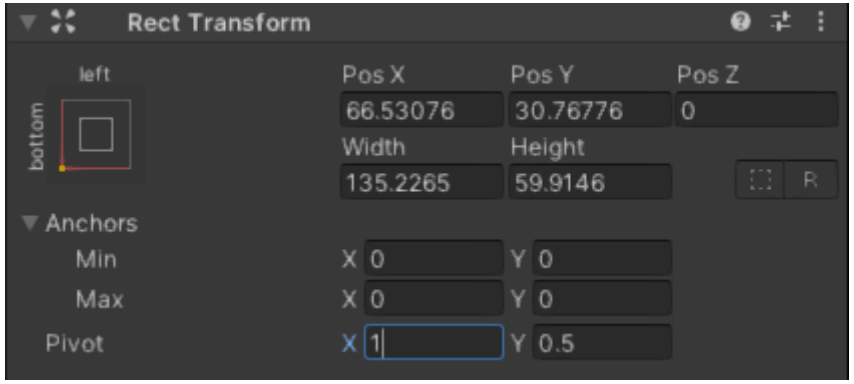


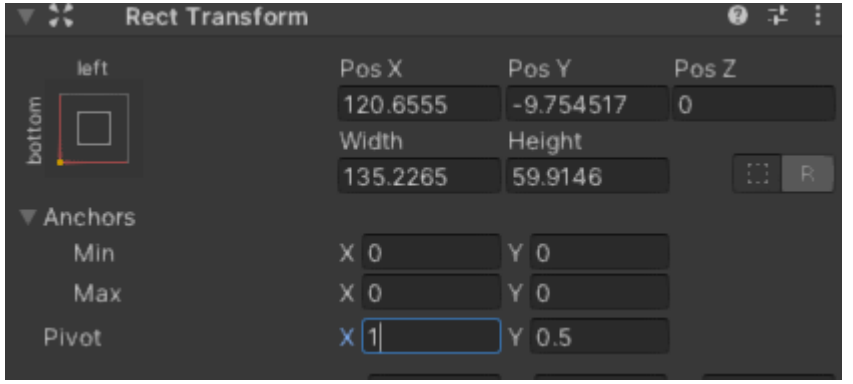
Blueprint Mode off

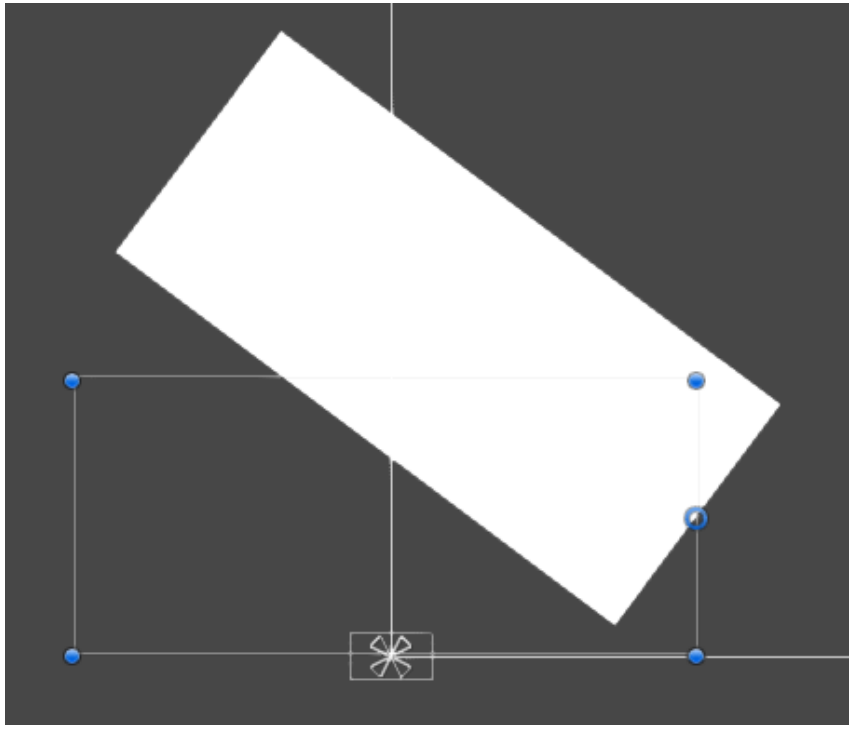


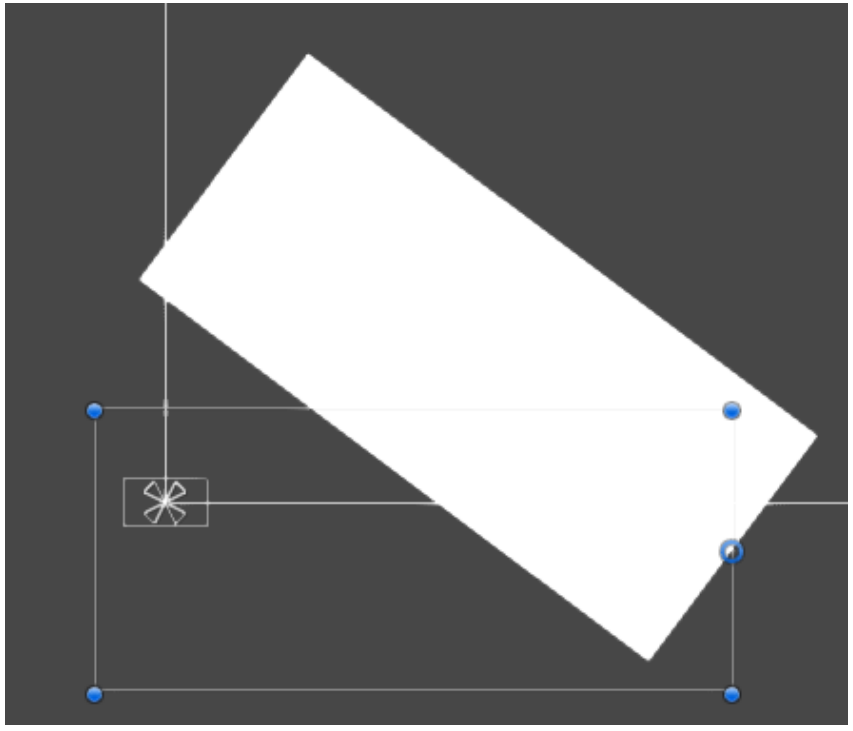
Blueprint Mode on

UGUI


Rect Transform
left
bottom
Pos X
Pos Y
Pos Z
66.53076
30.76776
0
Width
Height
135.2265
59.9146
R
Anchors
Min
X 0
Y 0
Max
X 0
Y 0
Pivot
X 1
Y 0.5


Rect Transform
left
bottom
Pos X
Pos Y
Pos Z
120.6555
-9.754517
0
Width
Height
135.2265
59.9146
R
Anchors
Min
X 0
Y 0
Max
X 0
Y 0
Pivot
X 1
Y 0.5





UGUI

- Render Mode
  1. Screen Space - Overlay
     기본 모드로, Unity에서 UI를 항상 게임 위에 표시합니다. 정보를 제공하려면 UI를 항상 위에 표시하는 편이 효과적이므로 대부분의 애플리케이션에서 이 모드를 사용합니다

  2. Screen Space - Camera
     카메라와 정렬을 이룬 평면 위에 UI를 표시합니다. 평면의 크기는 항상 화면을 가득 채우도록 지정되므로, 카메라를 움직이면 평면도 카메라를 따라 움직이면서 Overlay와 동일하게 나타납니다. 하지만, 평면은 화면의 위가 아니라 월드 안에 그려지므로 월드의 오브젝트가 UI 위에 그려질 수 있습니다

  3. World Space
     월드 안 임의의 위치에 평면을 그립니다. 예를 들어, 이 평면은 게임 내 컴퓨터의 화면, 월 등으로 사용하거나 캐릭터의 전면에서도 사용할 수 있습니다. 거리가 멀어질수록 UI의 크기가 작아지므로 3D 게임에서 더욱 유용한 모드입니다

UGUI

- ## Canvas Scaler

  화면 크기의 변화에 따른 UI의 확장 방식을 정의합니다. 게임을 실행하는 플레이어에 따라 해상도는 800x600 또는 1920x1080으로 맞출 수 있습니다. 모바일 앱에서는 가로 또는 세로 모드를 사용할 수 있습니다. 이러한 모든 옵션마다 서로 다른 화면 크기와 비율을 적용해야 합니다.

  1. ### Constant Size(Pixel 또는 Physical)

     화면의 크기나 형태와 관계없이 UI가 동일한 크기를 유지합니다. 어떤 화면에서든 UI를 읽을 수 있지만, 화면이 작을수록 UI가 비교적 더 많은 공간을 차지하게 됩니다. 따라서, 화면이 너무 작으면 여러 요소가 오버랩될 수 없습니다.

  2. ### Scale with Screen Size

     사용자가 레퍼런스 해상도(Reference Resolution)로 설정한 화면 크기에 따라 UI가 확장됩니다.
     예를 들어, 레퍼런스 해상도를 800x600으로 설정하고 화면 너비가 1600픽셀인 경우 UI는 두 배 더 크게 확장됩니다. 이와 같이 UI는 화면 크기와 관계 없이 화면에서 항상 동일한 비율을 차지합니다.
     이 모드의 단점은 UI의 기본 크기가 큰 게임을 작은 화면에서 플레이할 경우 UI가 너무 작아져 읽기 힘들 수도 있다는 것입니다. 또는, UI가 작은 화면에 맞게 조정된 게임을 큰 화면에서 플레이할 경우 UI가 흐릿해 보이거나 픽셀 단위로 보일 수도 있습니다.

- ## Graphic Raycaster

Canvas에 Raycaster를 하는 데 사용합니다. Raycaster는 Canvas의 모든 요소를 감시하여 그 중 하나에 충돌하였는지 여부를 결정합니다.

1. Graphic Raycaster Properties

Ignore Reversed Graphics	레이캐스터가 후면 그래픽스를 무시할지 여부입니다.
Blocked Objects	그래픽 레이캐스트를 막을 오브젝트 타입입니다.
Blocking Mask	그래픽 레이캐스트를 막을 오브젝트 타입입니다.

UGUI

- **Visual Components**

1. Text(- Text Mesh Pro)
   레이블이라고도 하며 표시할 텍스트를 입력, Rich Text 설정
2. Image
   Sprite
3. Raw Image
   Texture
4. Mask
   자식 요소를 부모의 형태대로 제약(즉 “마스킹”)합니다. 따라서 자식이 부모보다 클 경우 부모에 들어맞는 자식의 일부분만이 보일 수 있게 됩니다
5. Effects
   Shadow : Text 또는 Image 등의 그래픽 컴포넌트에 단순한 아웃라인 효과를 추가합니다. 그래픽 컴포넌트와 동일한 게임 오브젝트에 있어야 합니다.
   Outline : 텍스트와 이미지같은 그래픽 컴포넌트에 간단한 외곽선 효과를 추가합니다. 그래픽 컴포넌트와 동일한 게임 오브젝트에 있어야 합니다.
   Position as UV1 : 텍스트 및 이미지 그래픽스에 간단한 UV1 포지션 효과를 추가합니다.

UGUI

- **Interaction Components**
  1. Button(- Text Mesh Pro)
  2. Toggle
  3. Toggle Group
  4. Slider
  5. Scrollbar
  6. Dropdown(- Text Mesh Pro)
  7. Input Field(- Text Mesh Pro)
  8. Scroll Rect (Scroll View)

UGUI

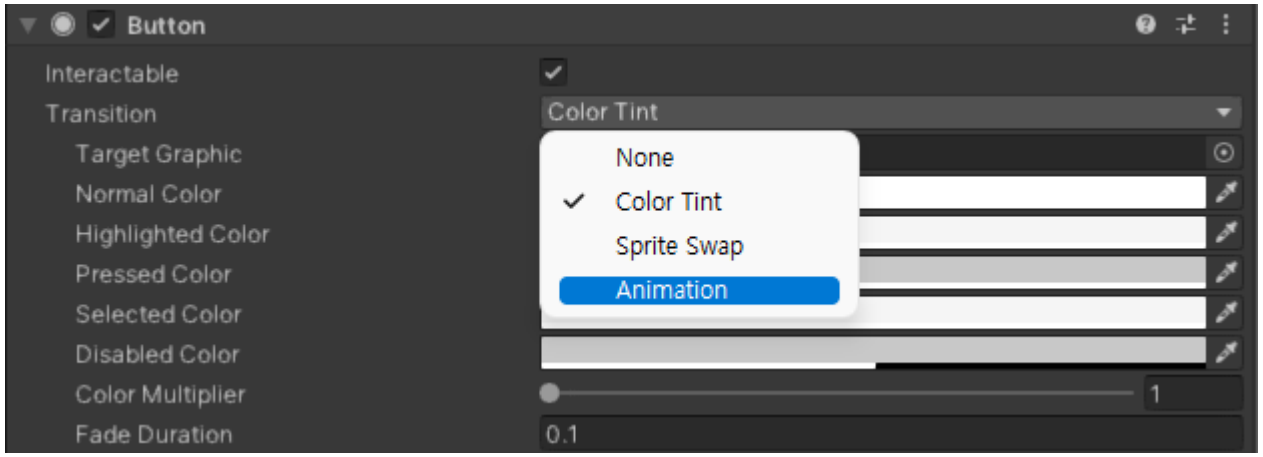
- **요소 그리기 순서**

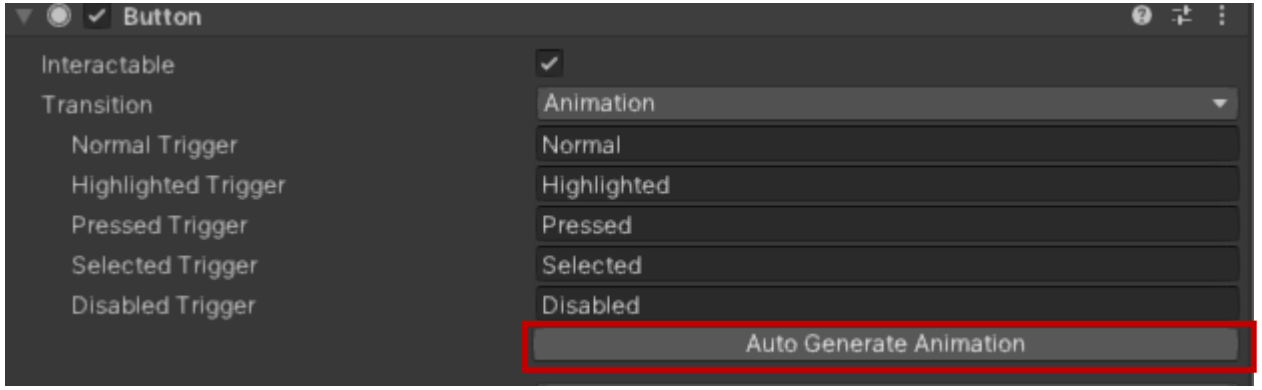
1. Canvas의 UI 요소는 계층 구조에 나타나는 것과 동일한 순서로 그려집니다. 첫 번째 자식이 먼저 그려지고 두 번째 자식이 다음에 그려지는 식입니다. 두 개의 UI 요소가 겹칠 경우 이전 요소 위에 나중에 표시되는 요소가 표시됩니다.

2. 다른 요소 위에 표시되는 요소를 변경하려면 계층 구조에서 요소를 끌어서 순서를 변경하기만 하면 됩니다.

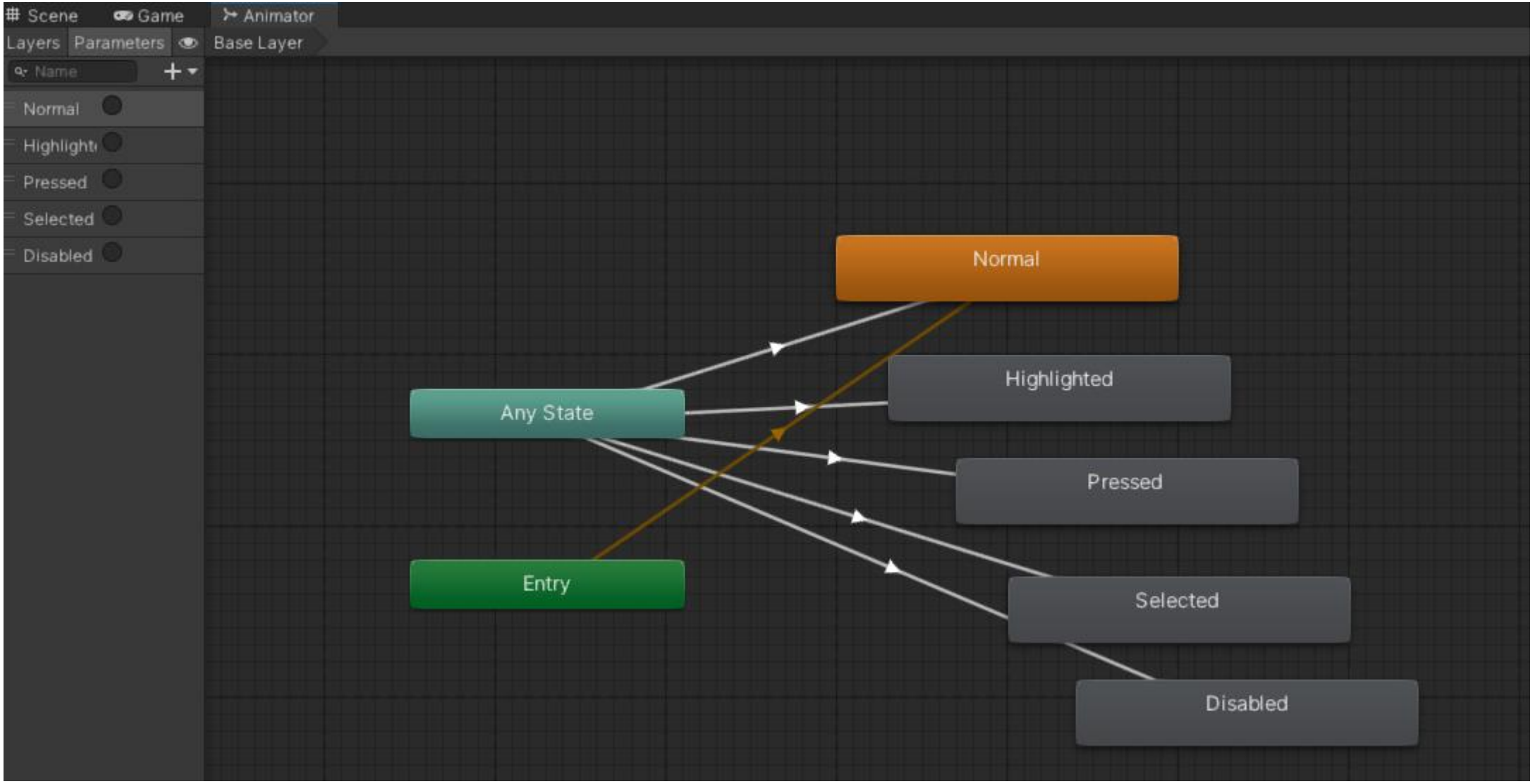
3. Transform 구성 요소에서 SetAsFirstSibling, SetAsLastSibling 및 SetSiblingIndex 메서드를 사용하여 스크립팅에서 순서를 제어할 수도 있습니다

- **Animation Integration**
  Unity의 애니메이션 시스템을 사용하여 컨트롤 상태의 각 전환을 완벽하게 애니메이션화할 수 있습니다





UGUI


Scene
Game
Animator
Layers
Parameters
Base Layer
Name
Normal
Highlight
Pressed
Selected
Disabled
Normal
Highlighted
Any State
Pressed
Entry
Selected
Disabled

UGUI


Console
Animation
Preview
0
Normal
Samples 60
Disabled
Highlighted
Normal
Pressed
Selected
Create New Clip...
roperty
0:00
0:05
0:10
0:15
0:20
0:25
0:30
0:35
0:40
Dopesheet
Curves

UGUI

- **Panel**
  Rect Transform, Image(Maskable)

  1. Layout Group
     Grid Layout Group, Horizontal Layout Group, Vertical Layout Group

  2. Layout Element
     Content Size Fitter, Aspect Ratio Fitter

SceneManagement

SceneManagement

- **UnityEngine.SceneManagement**

1. SceneManager
   런타임에 Scene 을 관리

   Static Properties

sceneCount	현재 로드된 총 Scene 수
sceneCountInBuildSettings	빌드 세팅에서 설정된 Scene 수

SceneManagement

Static Method

CreateScene	런타임에 지정된 이름으로 빈 새 Scene을 만듭니다.
GetActiveScene	현재 활성화된 Scene을 가져옵니다.
GetSceneAt	SceneManager의 로드된 Scene 목록에서 인덱스로 Scene을 가져옵니다
GetSceneByBuildIndex	빌드 인덱스로 Scene을 가져옵니다.
GetSceneByName	Scene 이름으로 로드된 Scene을 가져옵니다.
GetSceneByPath	지정된 파일 경로로 Scene을 가져옵니다.
LoadScene	이름 또는 인덱스로 빌드 설정에 세팅된 Scene을 로드합니다.
LoadSceneAsync	이름 또는 인덱스로 빌드 설정에 세팅된 Scene을 비동기(백그라운드)로 로드합니다.
MergeScenes	Scene을 병합합니다.(sourceScene, destinationScene)
MoveGameObjectToScene	게임 오브젝트를 현재 Scene에서 새 Scene으로 이동합니다.
SetActiveScene	Scene을 활성화하도록 설정합니다.
UnloadSceneAsync	주어진 Scene과 관련된 모든 게임오브젝트를 파괴하고 SceneManager의 로드된 Scene 목록 Scene을 제거합니다.

SceneManagement

Events

activeSceneChanged	활성 Scene이 변경될 때
sceneLoaded	Scene이 로드될 때
sceneUnloaded	Scene이 언로드될 때

SceneManagement

- **Scene**
  *.unity 파일의 런타임 데이터 구조체

  변수
  buildIndex
  isDirty - Scene이 수정되면 true를 반환
  isLoaded
  name
  path
  rootCount

  Public Method
  GetRootGameObjects
  IsValid

  Operator
  operator !=
  operator ==

SceneManagement

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ExampleClass : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        // Only specifying the sceneName or sceneBuildIndex will load the Scene with the
Single mode
        SceneManager.LoadScene("SampleScene", LoadSceneMode.Additive);
    }
}
```

SceneManagement

```
using System.Collections;
using UnityEngine;
using UnityEngine.SceneManagement;

public class LoadScene : MonoBehaviour
{
    private AssetBundle myLoadedAssetBundle;
    private string[] scenePaths;

    [SerializeField] private string m_Scene;
    [SerializeField] private GameObject m_MyGameObject;

    void Start()
    {
        myLoadedAssetBundle = AssetBundle.LoadFromFile("Assets/AssetBundles/scenes");
        scenePaths = myLoadedAssetBundle.GetAllScenePaths();
    }

    void OnGUI()
    {
```

SceneManagement

```
        if (GUI.Button(new Rect(10, 10, 100, 30), "Change Scene"))
        {
            Debug.Log("Scene2 loading: " + scenePaths[0]);
            SceneManager.LoadScene(scenePaths[0], LoadSceneMode.Single);
        }
    }

    IEnumerator LoadYourAsyncScene()
    {
        // Set the current Scene to be able to unload it later
        Scene currentScene = SceneManager.GetActiveScene();

        // The Application loads the Scene in the background
        // at the same time as the current Scene.
        AsyncOperation asyncLoad =
            SceneManager.LoadSceneAsync(m_Scene, LoadSceneMode.Additive);

        // Wait until the last operation fully loads to return anything
        while (!asyncLoad.isDone)
        {
```

SceneManagement

```
            yield return null;
        }

        // Move the GameObject (you attach this in the Inspector) to the newly loaded Scene
        SceneManager.MoveGameObjectToScene(m_MyGameObject,
            SceneManager.GetSceneByName(m_Scene));
        // Unload the previous Scene
        SceneManager.UnloadSceneAsync(currentScene);
    }
}
```

중간 평가 과제물

중간 평가 과제물

- 2D 러너
  - **Scene 구성**
    시작 Scene(이름, 학번), 게임 Scene,(결과 씬)
  - **한 개 이상 기능 추가 또는 기존 기능 변경**
    추가 및 변경 기능 설명 PPT
  - **제출물**
    프로젝트 폴더(Assets, Packages, ProjectSettings) + 추가 기능 설명 PPT
    이름(학번).zip
  - **이메일 제출**
    다음 강의 시간 (4월26일 9시) 전까지
    ibanho@gmail.com

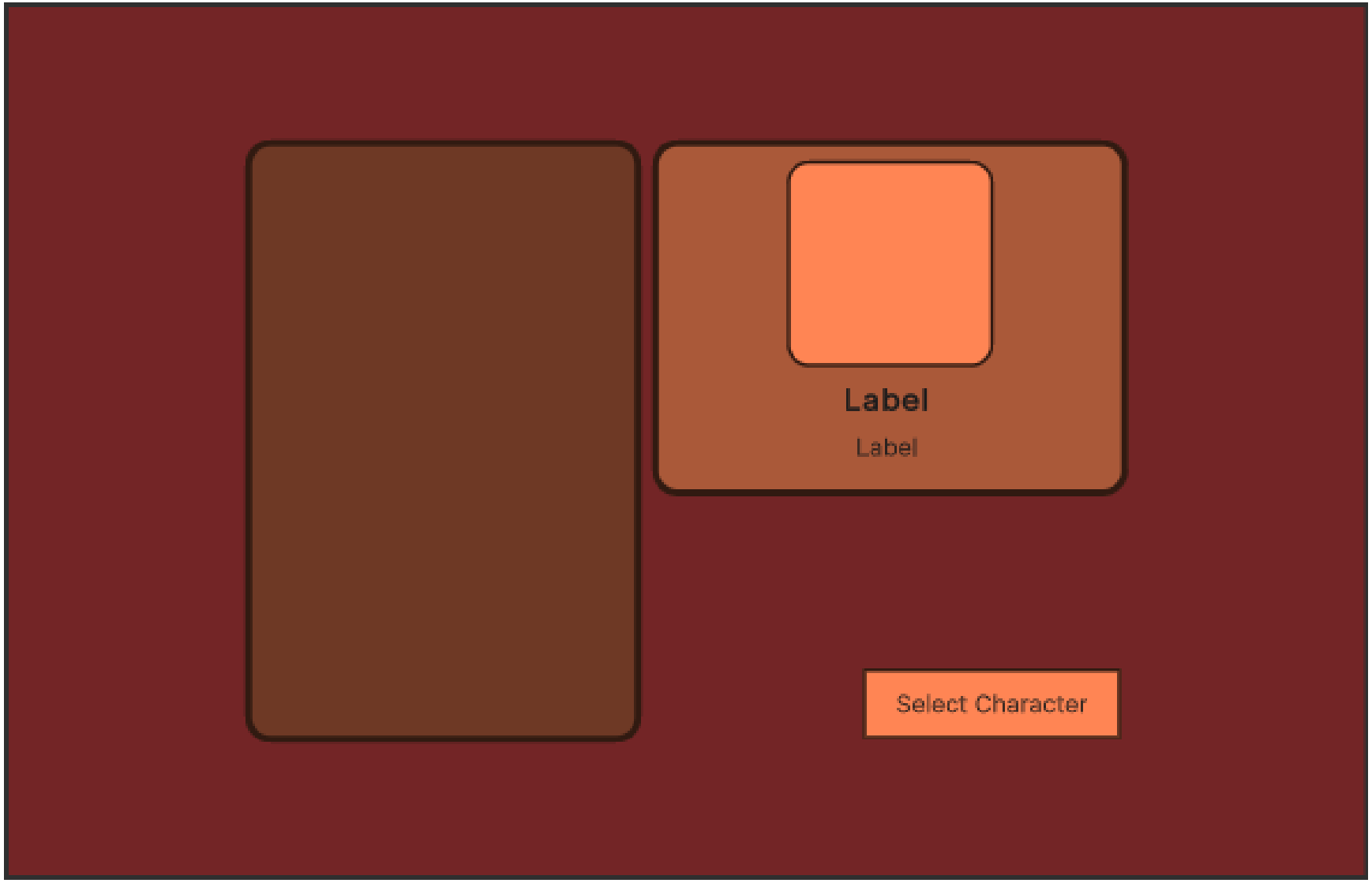
Q & A

Q & A

- UnityEngine Editor
- C# 프로그래밍
- 탄막 슈팅
- 2D 러너
- 중간 과제 제출

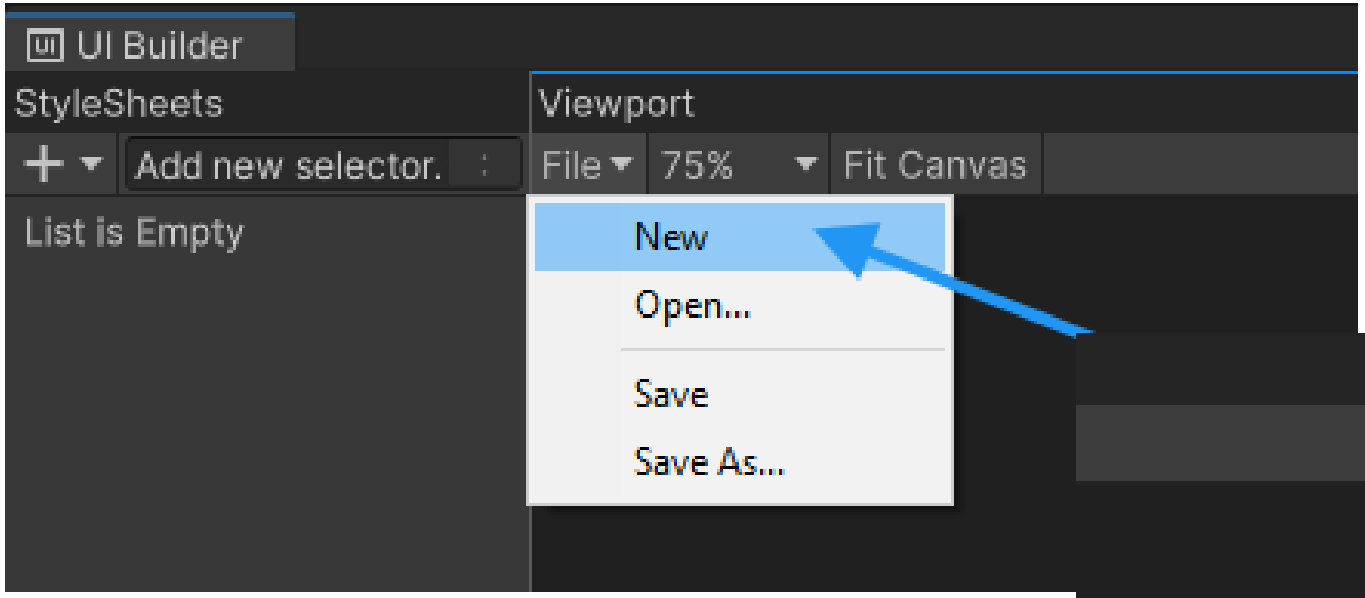
UI Toolkit

UI Toolkit

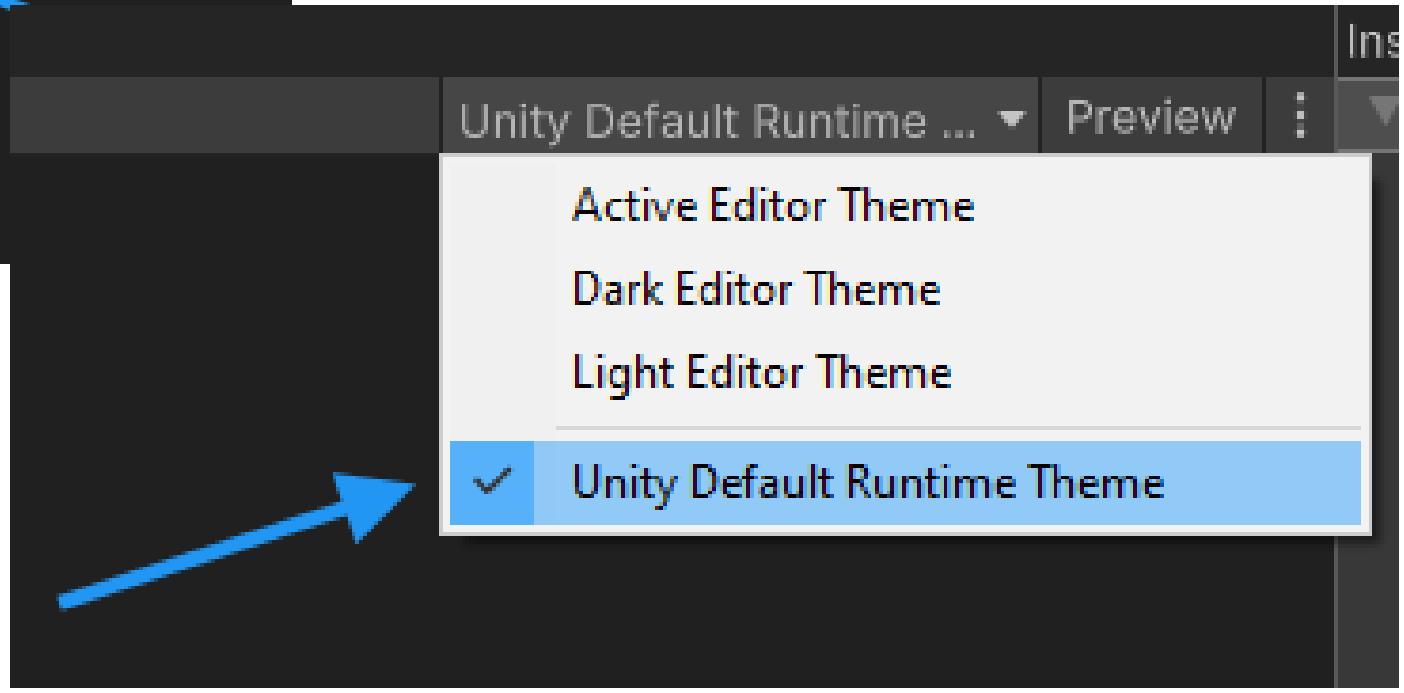


UI Toolkit

Window > UI Toolkit > UI Builder



뷰포트 왼쪽 상단에 있는 File > New 를 사용하여 새 UXML 문서를 생성

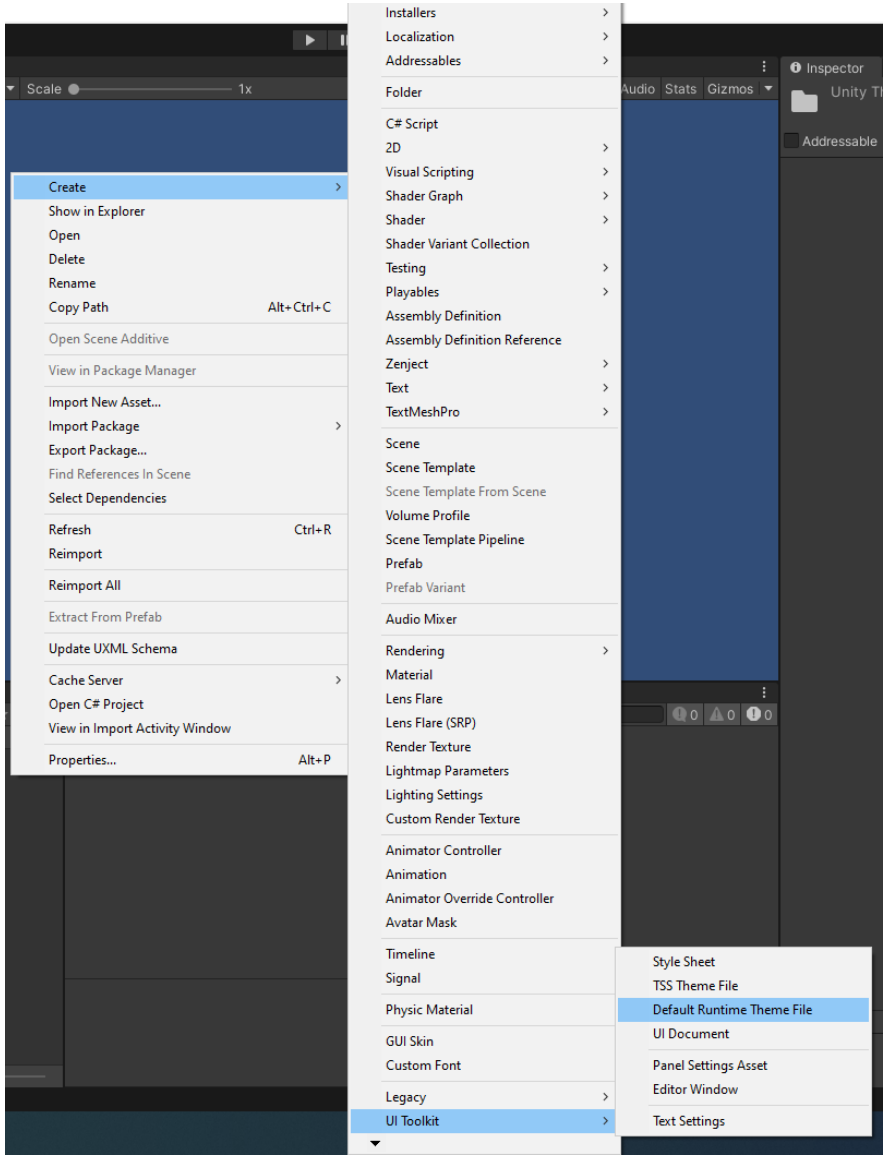


게임 UI를 개발할 때는 항상 UI 빌더 뷰포트 오른쪽 상단에 있는 Unity Default Runtime Theme을 선택.
에디터 및 런타임 테마의 기본 폰트 크기와 컬러는 서로 다르며, 이는 레이아웃에 영향을 미칩니다.

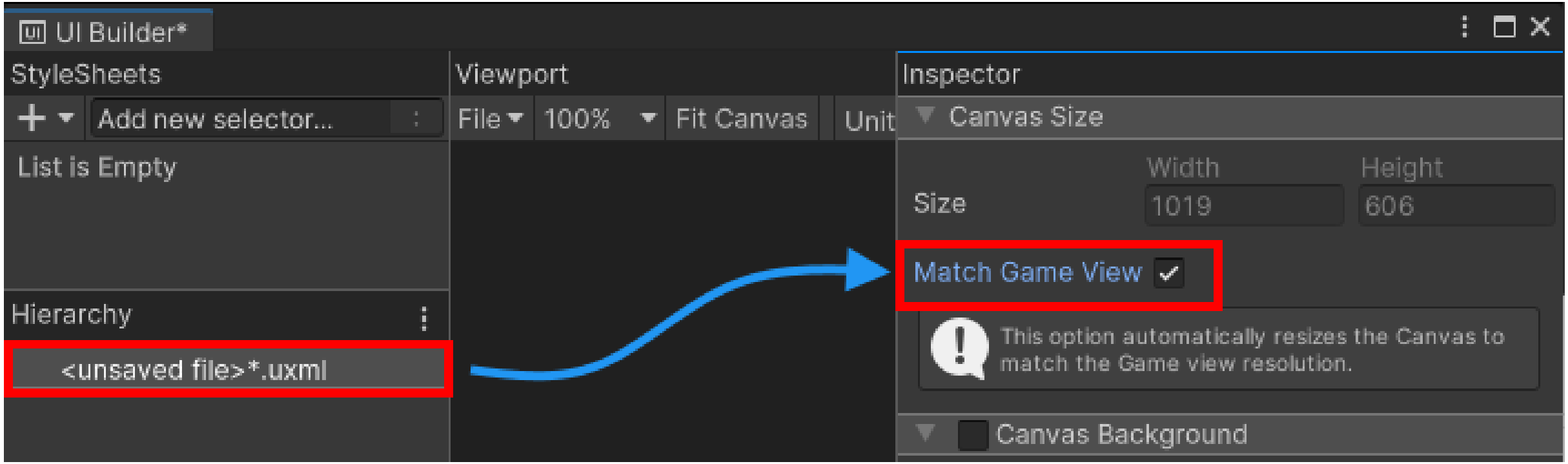
UI Toolkit

UnityDefaultRuntimeTheme

Project Window
Create > UI Toolkit > Default Runtime Theme File

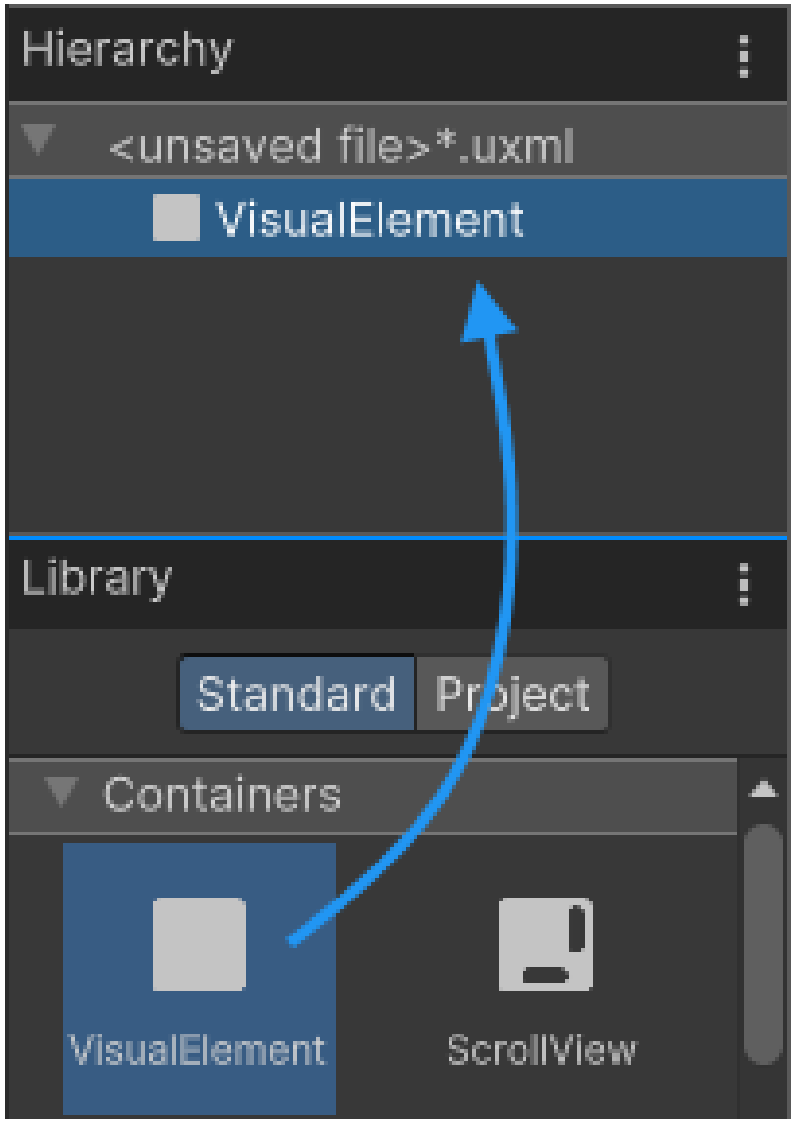


UI Toolkit

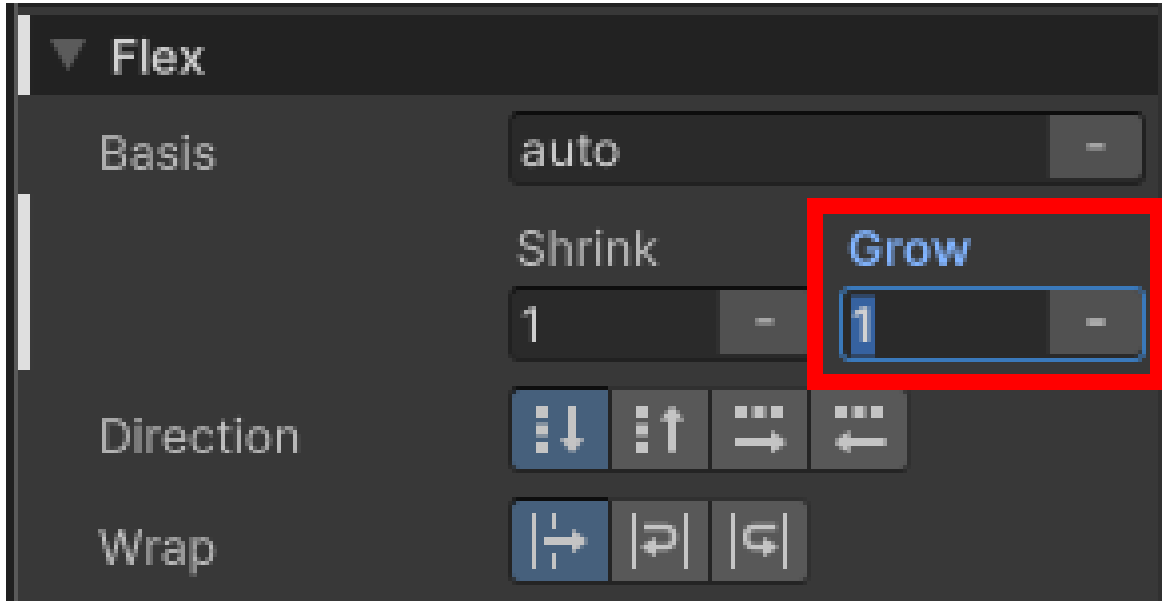


Hierarchy 에서 UXML 파일을 선택하고 Match Game View 체크박스를 활성화.
이미 설정되어 있지 않은 경우 Unity 에디터를 가로 해상도로 설정.

UI Toolkit



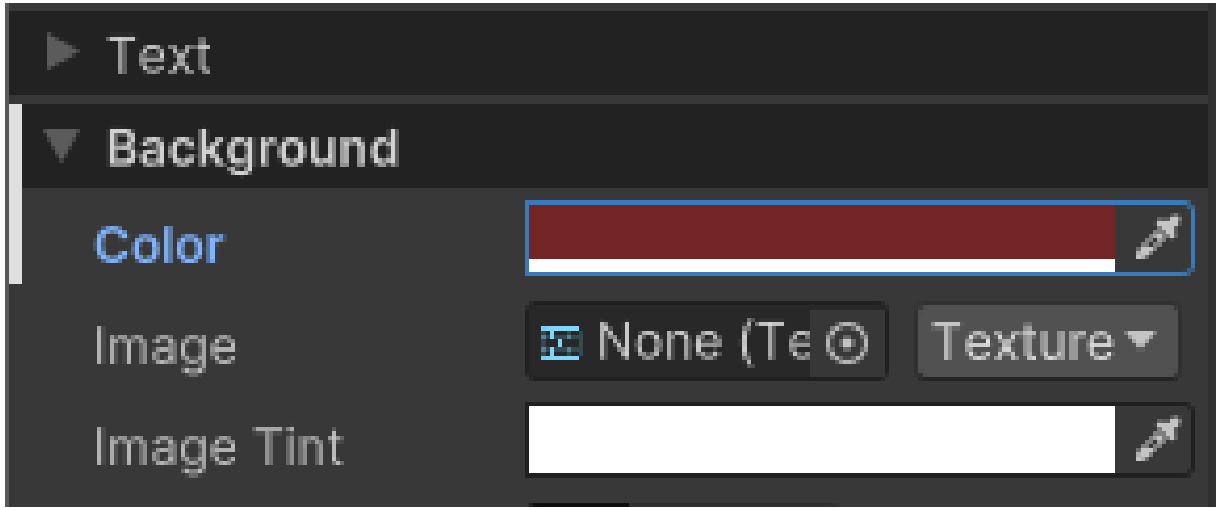
새 요소는 화면 전체를 커버해야 하므로, flex-grow 프로퍼티를 1로 설정해야 합니다. 계층 구조에서 요소를 선택하고 오른쪽 인스펙터 패널에서 Flex라는 레이블이 있는 폴드아웃을 찾으십시오. Grow의 값을 0에서 1로 변경하십시오.



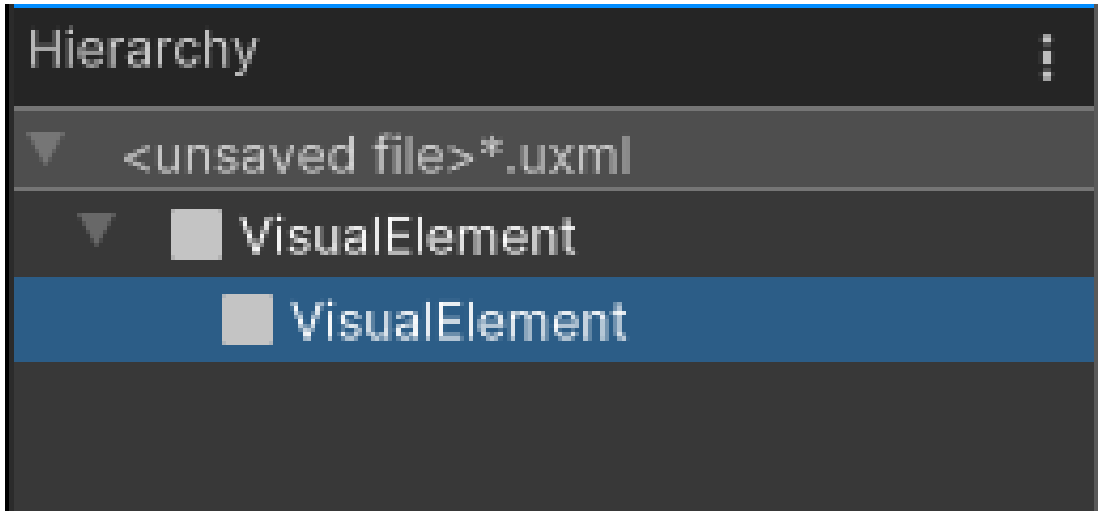
UI Toolkit

이 VisualElement의 모든 자식을 화면 중앙에 정렬하려면 VisualElement의 Align 프로퍼티를 변경하십시오. Align Items와 Justify Content를 모두 center로 설정해야 합니다.

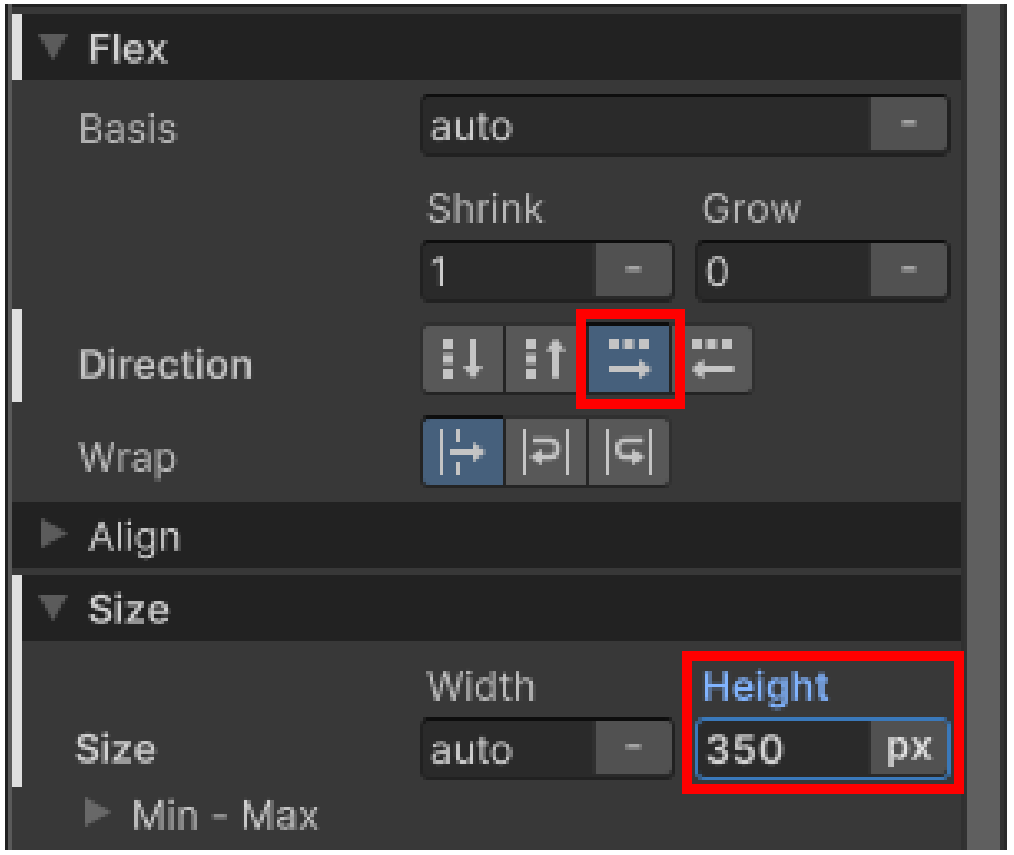
마지막으로 Background > Color에서 배경색을 선택할 수 있습니다. 이 단계는 선택 사항입니다. 이 예에서는 #732526을 컬러로 사용했습니다.



UI Toolkit

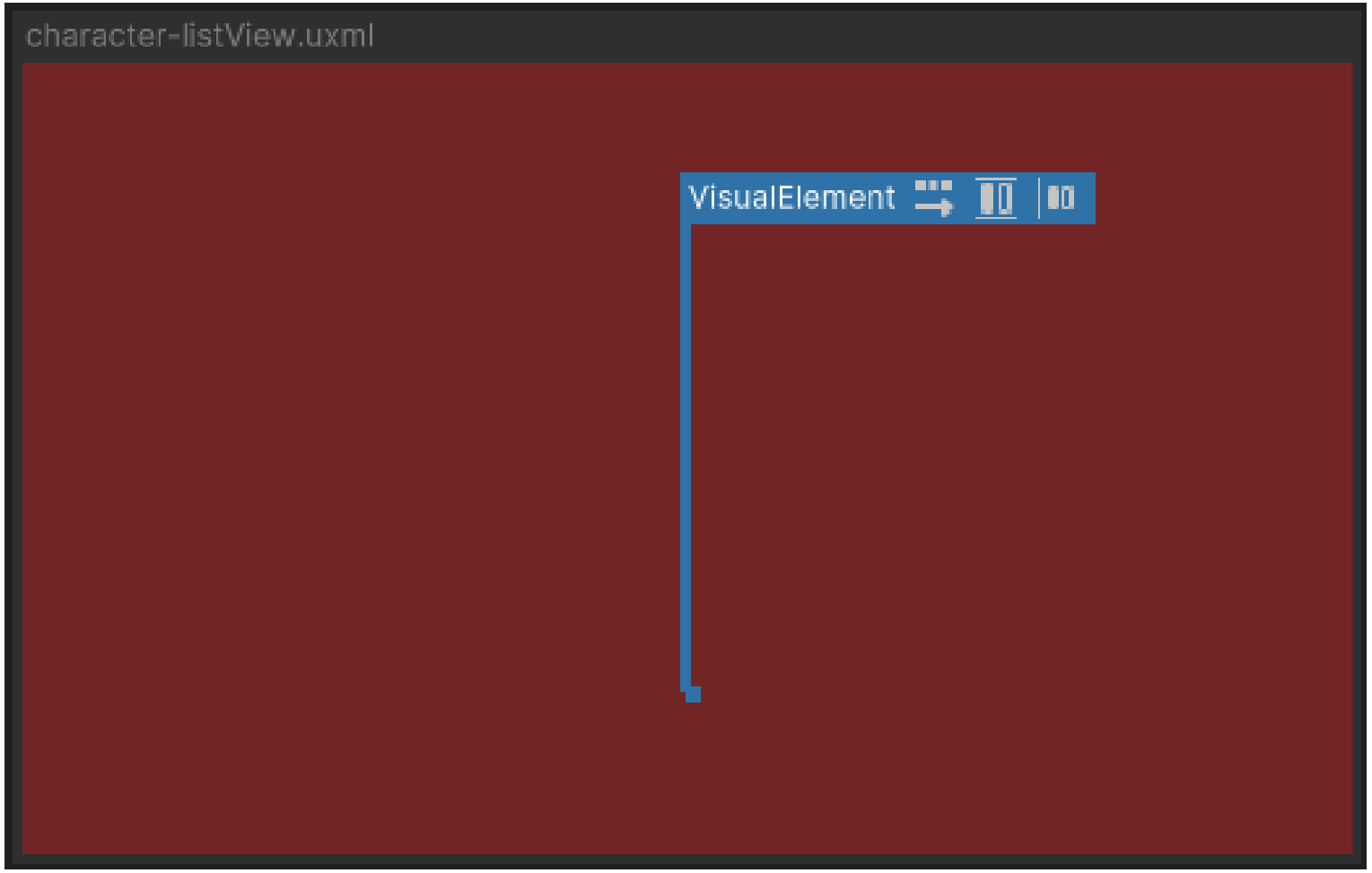


기존 VisualElement 아래에 새 VisualElement를 만드십시오. 이는 UI의 왼쪽 및 오른쪽 섹션을 위한 부모 컨테이너가 됩니다.

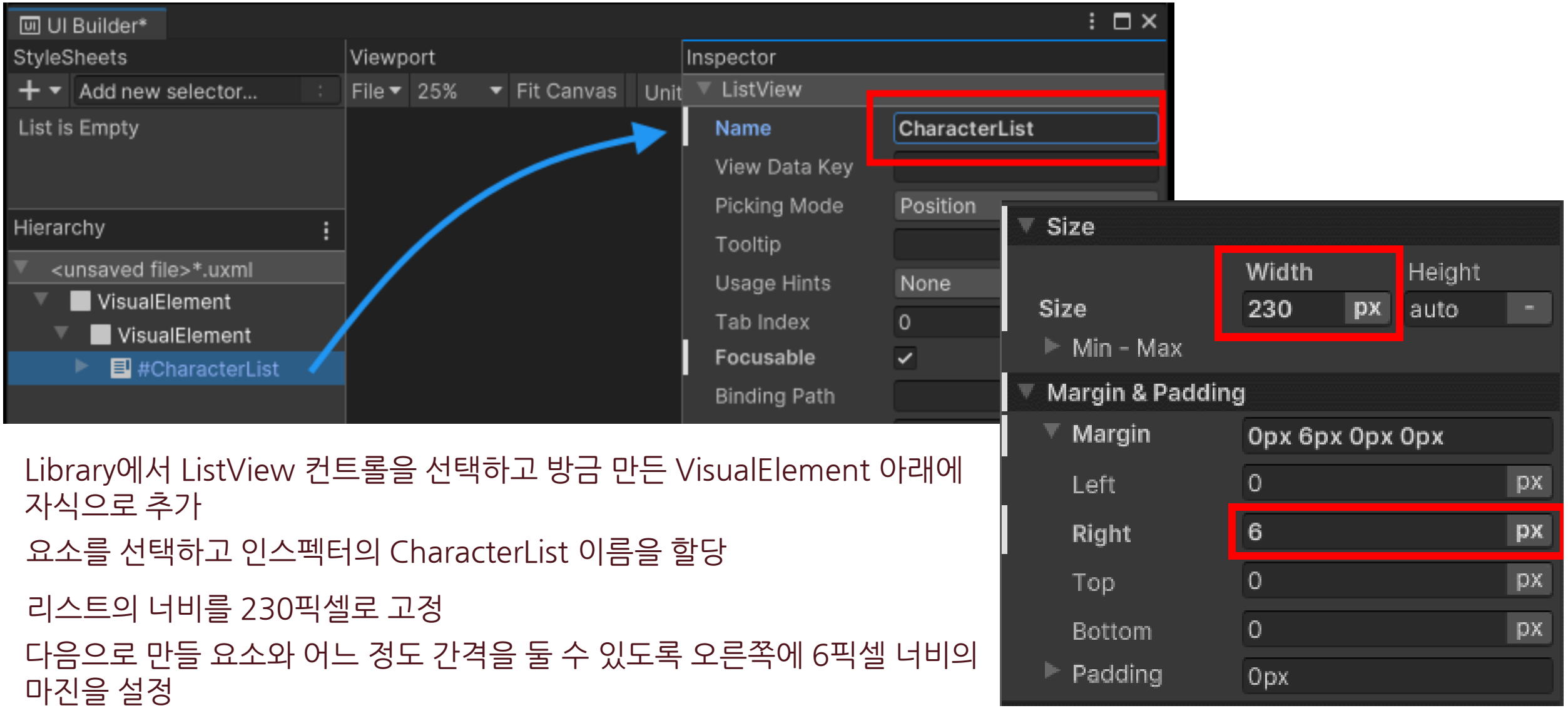


새 요소의 flex-direction 프로퍼티를 row로 설정하십시오 (기본값: column). 또한 픽셀 높이를 350픽셀로 고정해야 합니다.

UI Toolkit



UI Toolkit



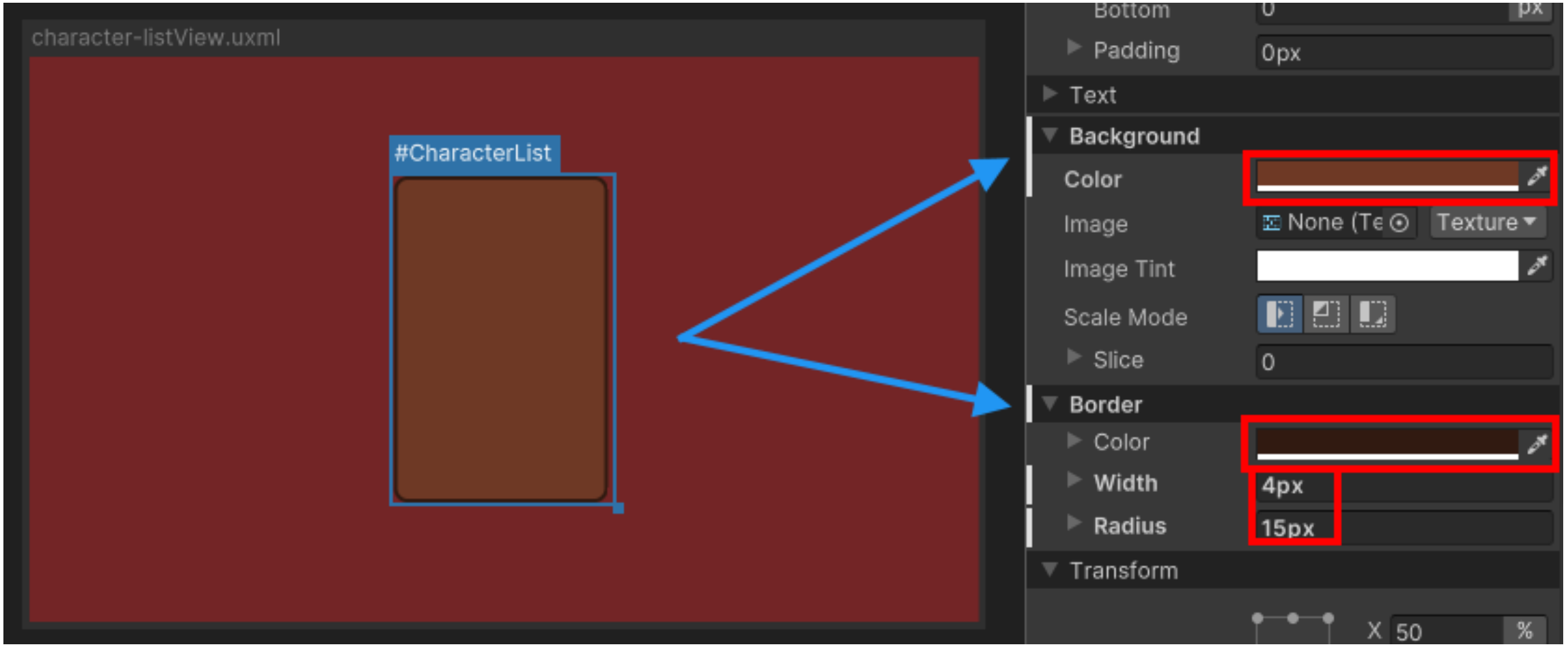
Library에서 ListView 컨트롤을 선택하고 방금 만든 VisualElement 아래에 자식으로 추가

요소를 선택하고 인스펙터의 CharacterList 이름을 할당

리스트의 너비를 230픽셀로 고정

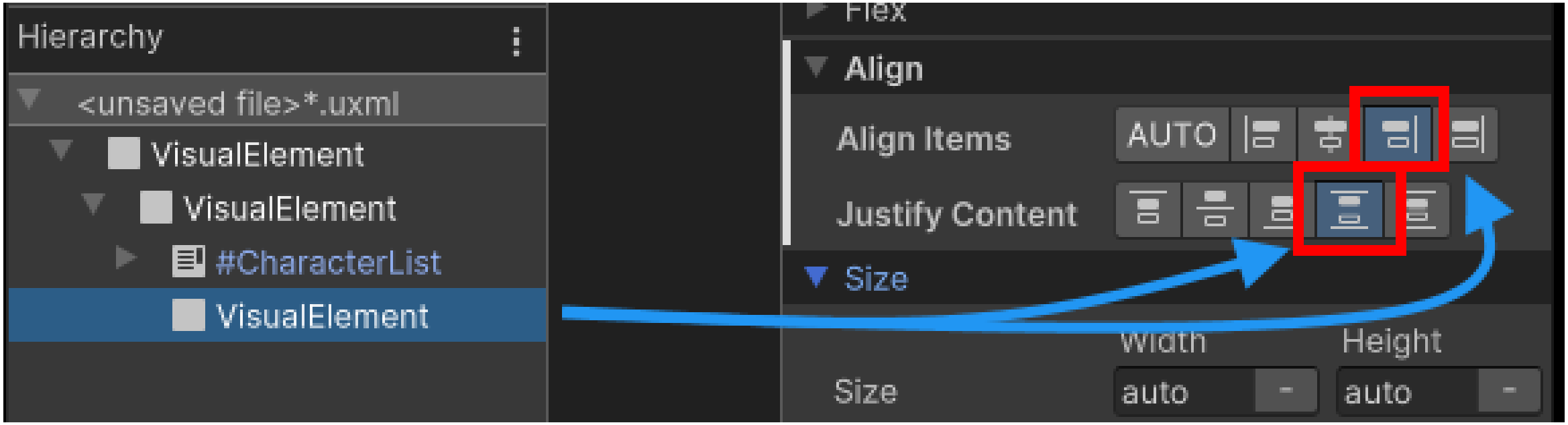
다음으로 만들 요소와 어느 정도 간격을 둘 수 있도록 오른쪽에 6픽셀 너비의 마진을 설정

UI Toolkit



배경색을 #6E3925로, 테두리 컬러를 #311A11로 설정했으며, 테두리 너비는 4픽셀로, 반지름은 15픽셀로 설정

UI Toolkit

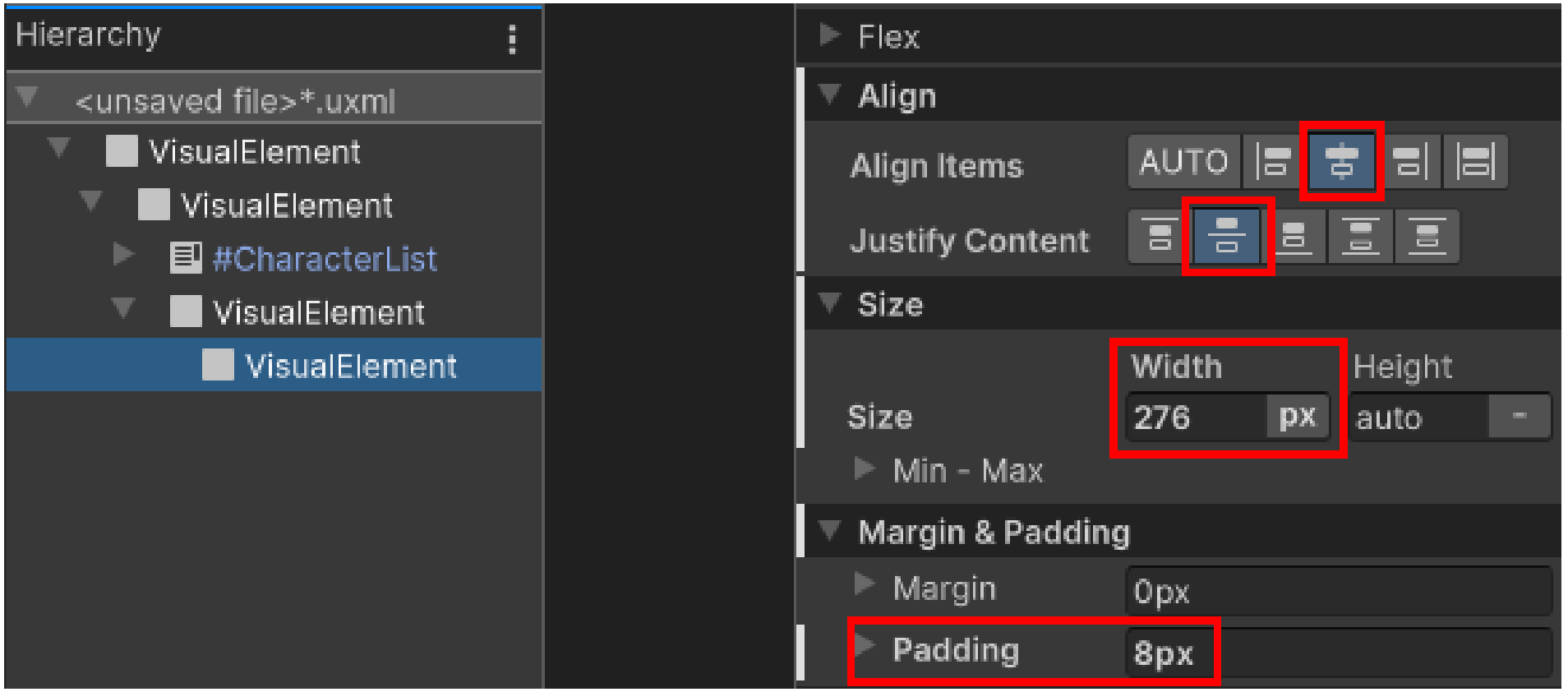


CharacterList와 같은 부모 아래에 새 VisualElement를 추가

이 VisualElement는 캐릭터 세부 사항 패널과 버튼을 포함

Align 폴드아웃 아래에서 Align Items 설정을 flex-end로, Justify Content를 space-between으로 변경

UI Toolkit



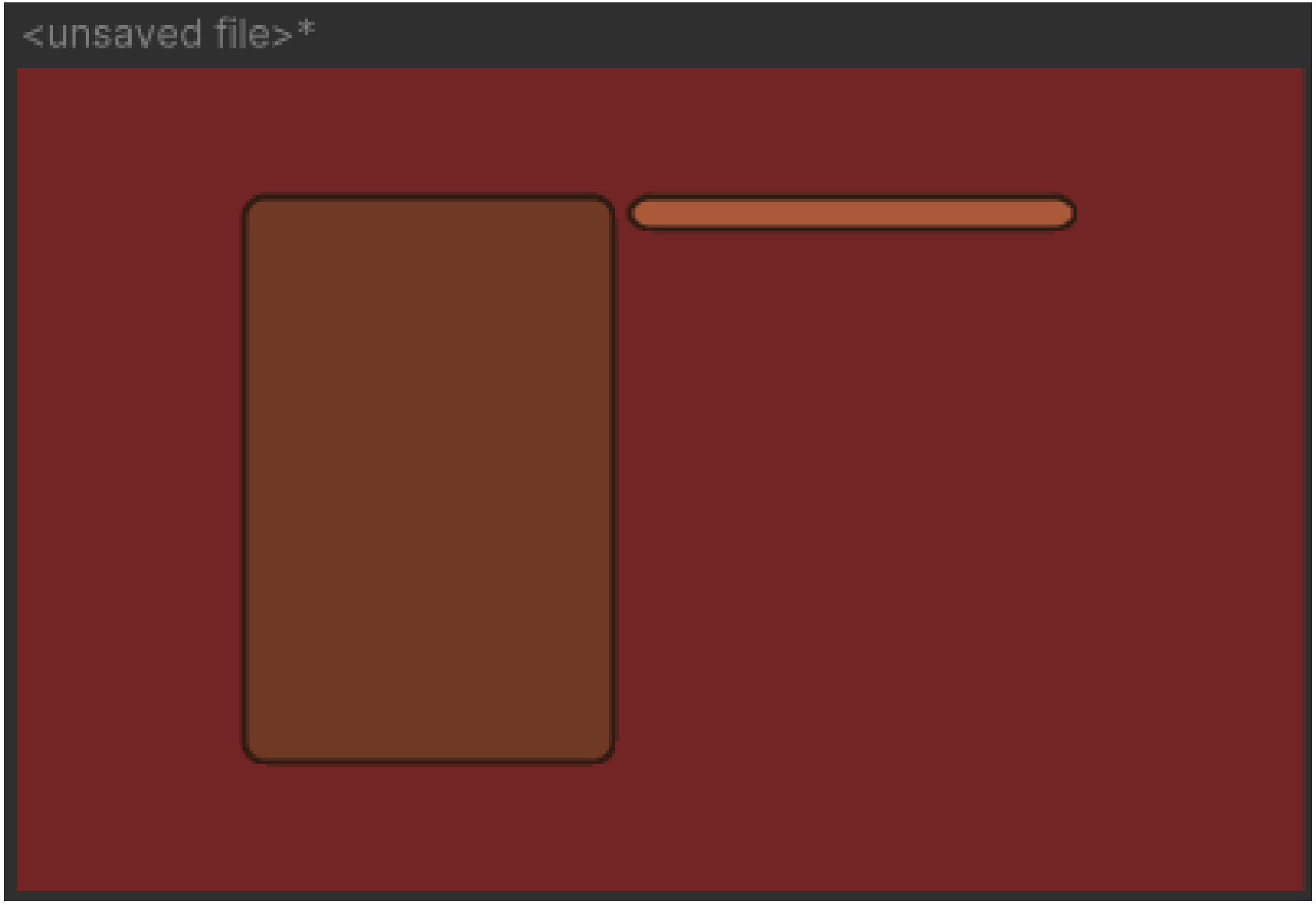
이 새 컨테이너에 새로운 VisualElement를 추가. 이 VisualElement는 캐릭터 세부 사항 패널

사용자가 왼쪽 리스트에서 캐릭터를 선택하면 이 패널은 캐릭터의 초상, 이름과 클래스를 표시

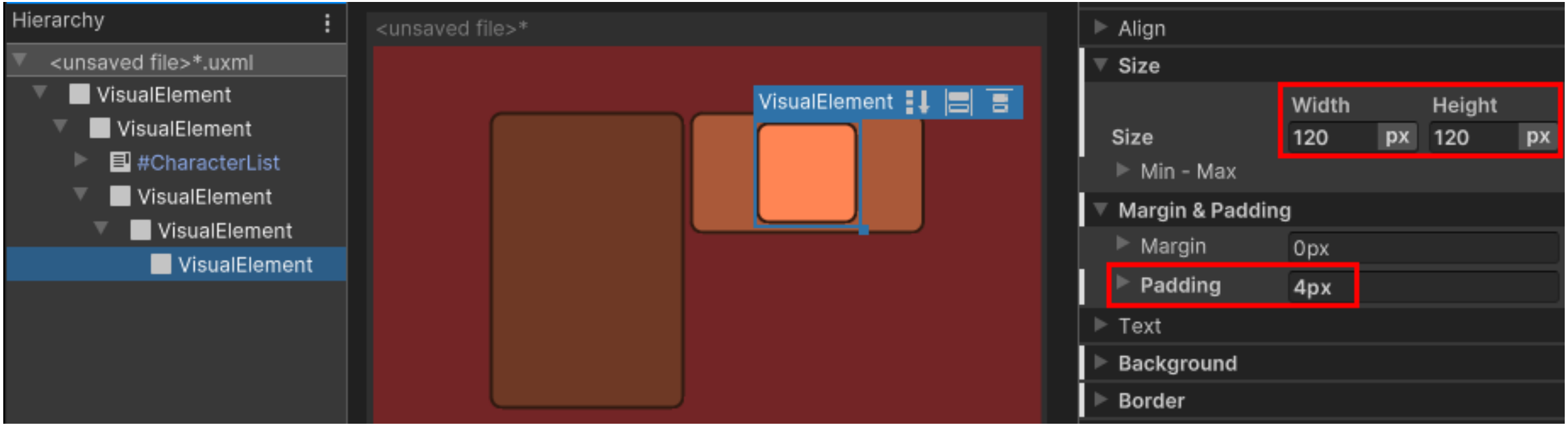
너비를 276픽셀로 고정하고, Align Items와 Justify Content를 center로 전환. 8픽셀 너비의 패딩을 추가

배경색을 #AA5939로, 테두리 컬러를 #311A11　너비와 반지름을 각각 4px와 15px로 설정

UI Toolkit



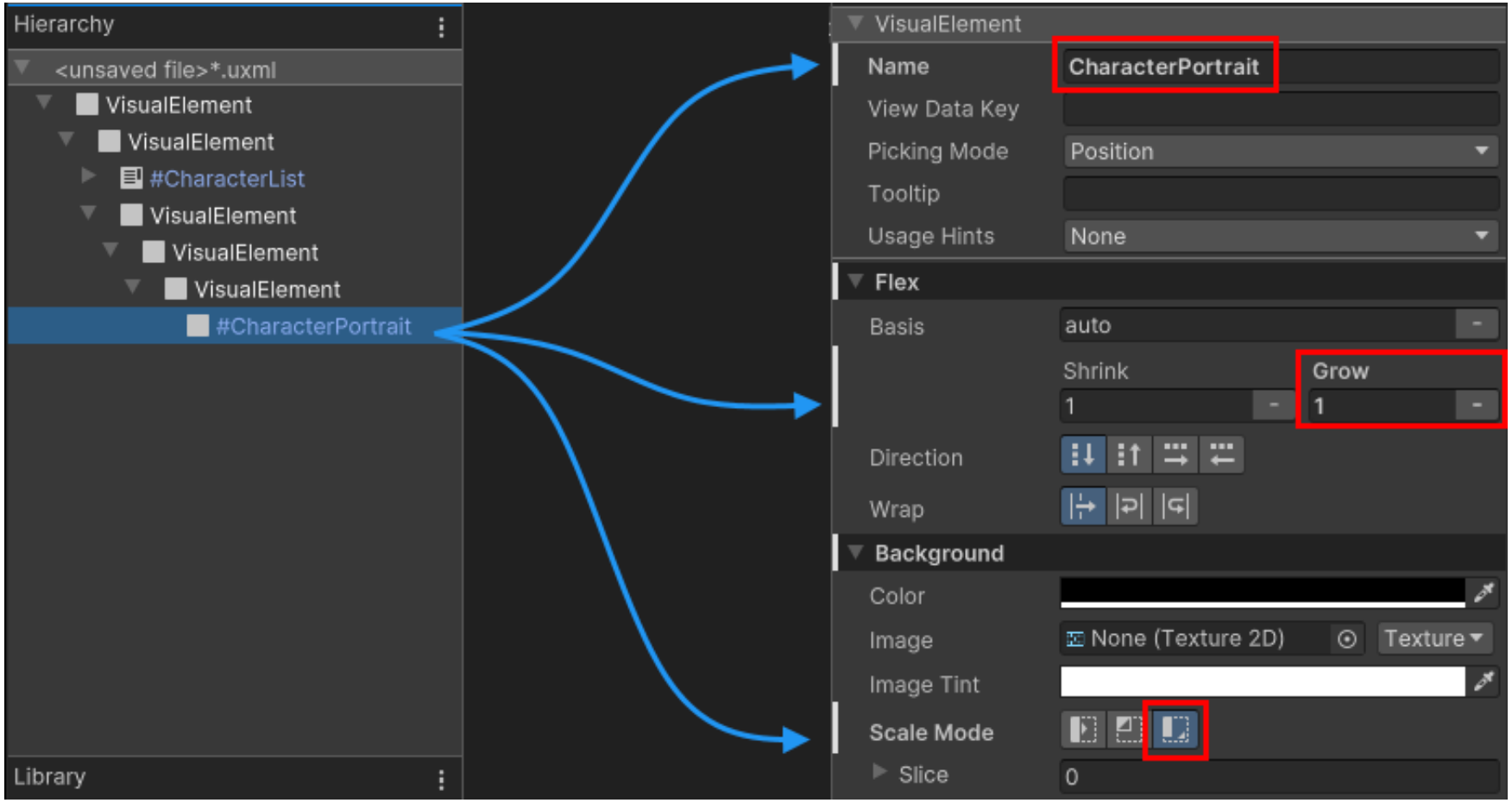
UI Toolkit



새 VisualElement를 추가 120x120픽셀의 고정된 크기를 할당 4픽셀의 패딩을 설정

테두리가 #311A11이고 배경색이 #FF8554이며, 너비와 반지름이 각각 2픽셀, 15픽셀

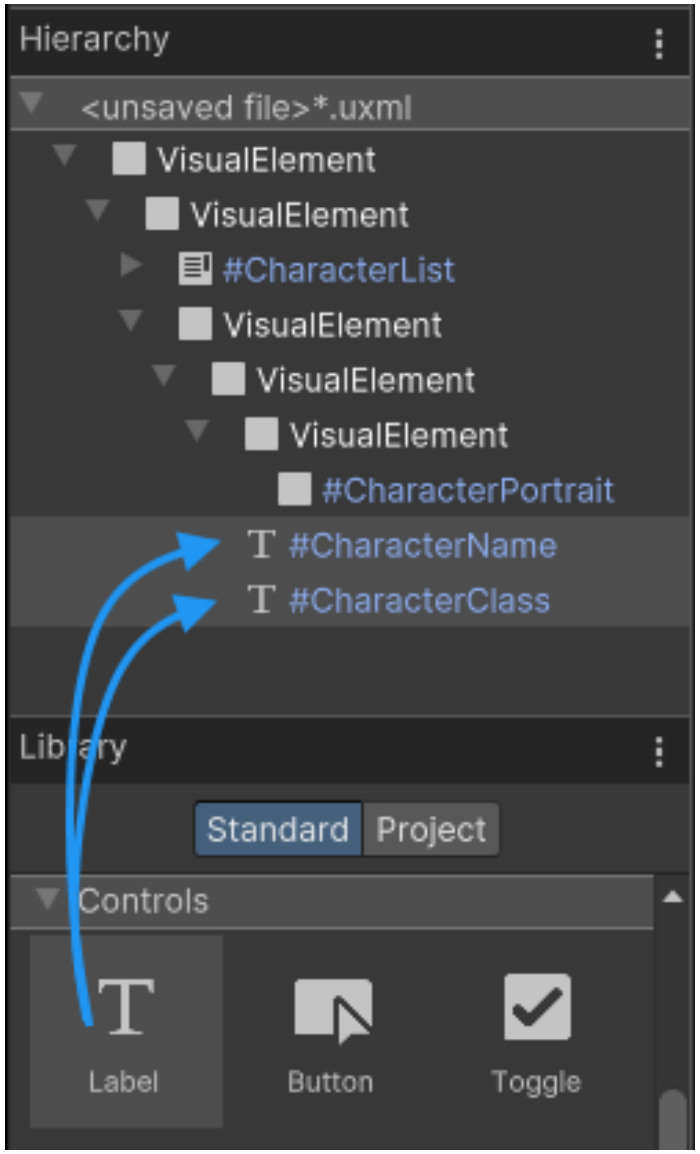
UI Toolkit

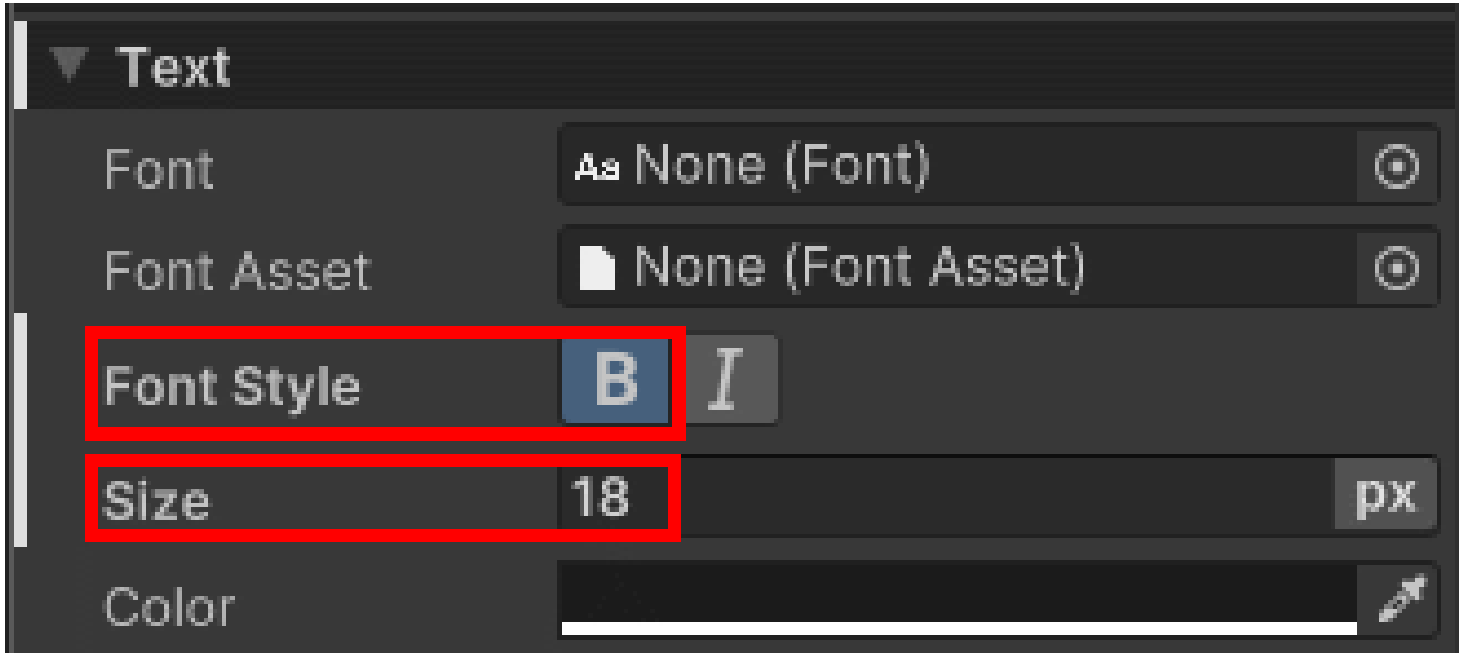


새 VisualElement를 자식으로 추가. 이름을 CharacterPortrait로 지정

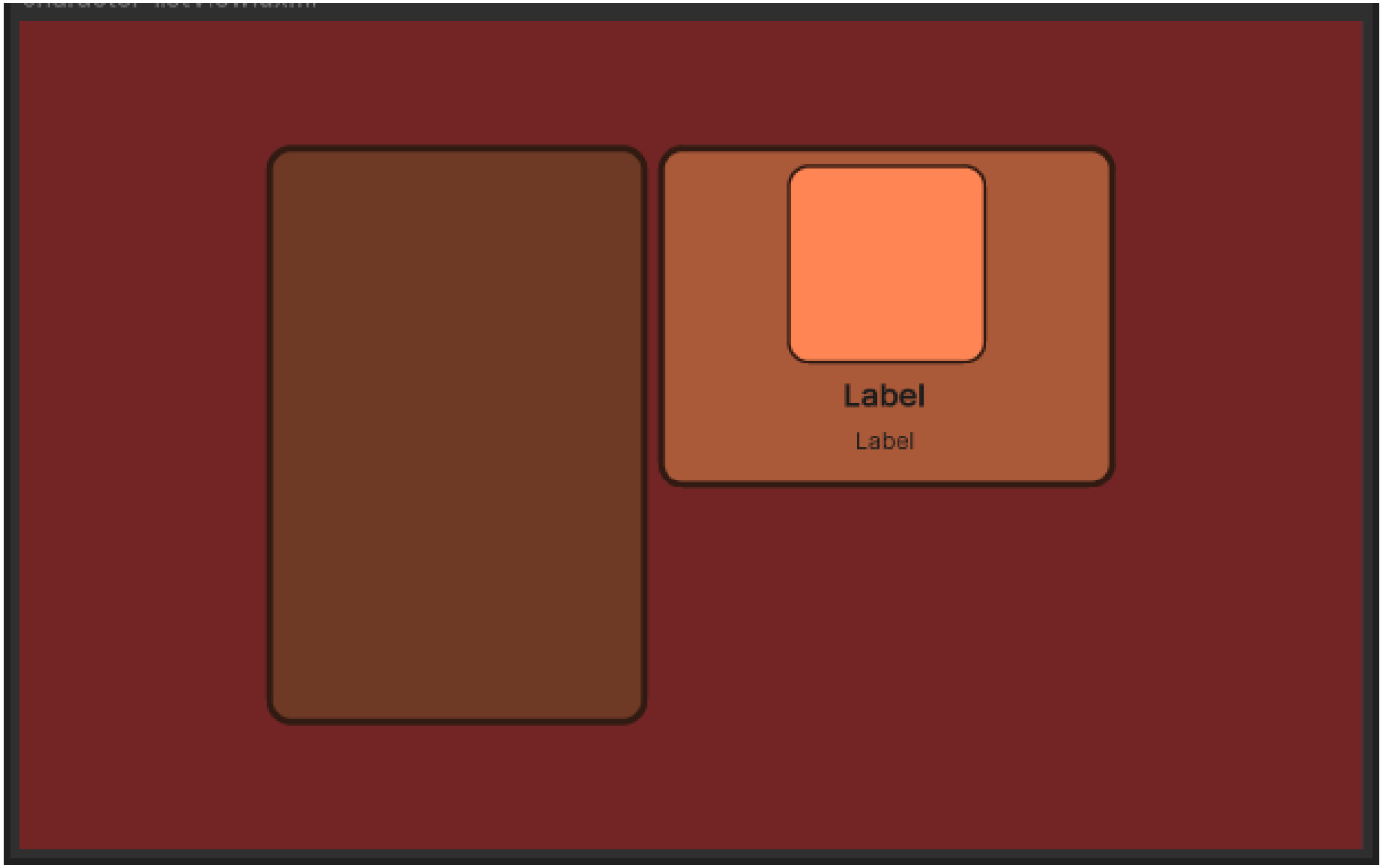
Flex > Grow를 1로 설정. Background > Scale Mode에서 확대/축소 모드를 scale-to-fit으로 변경

UI Toolkit

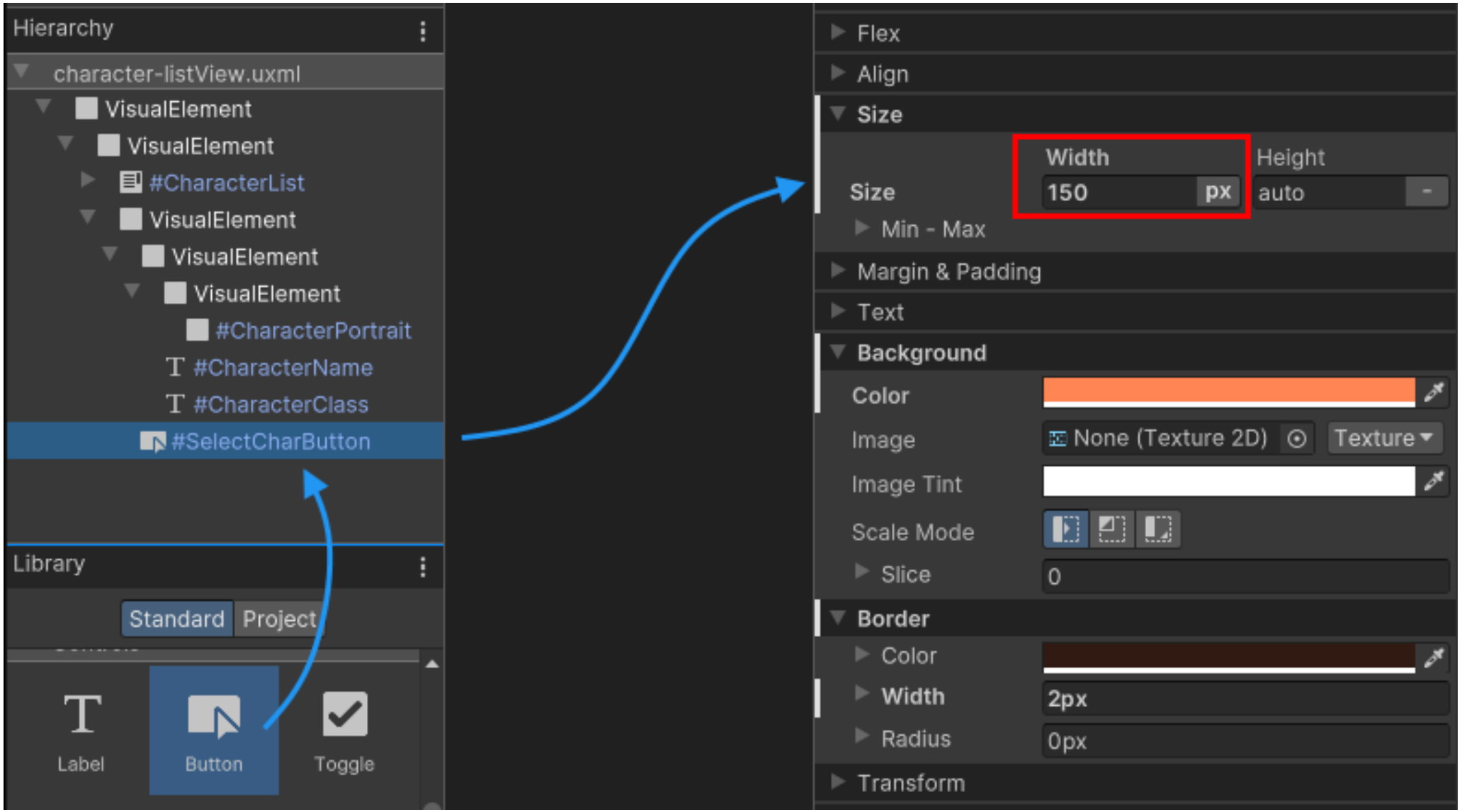

Hierarchy
<unsaved file>*.uxml
VisualElement
VisualElement
#CharacterList
VisualElement
VisualElement
VisualElement
#CharacterPortrait
#CharacterName
#CharacterClass
Library
Standard
Project
Controls
Label
Button
Toggle


Text
Font
None (Font)
Font Asset
None (Font Asset)
Font Style
B
I
Size
18
px
Color

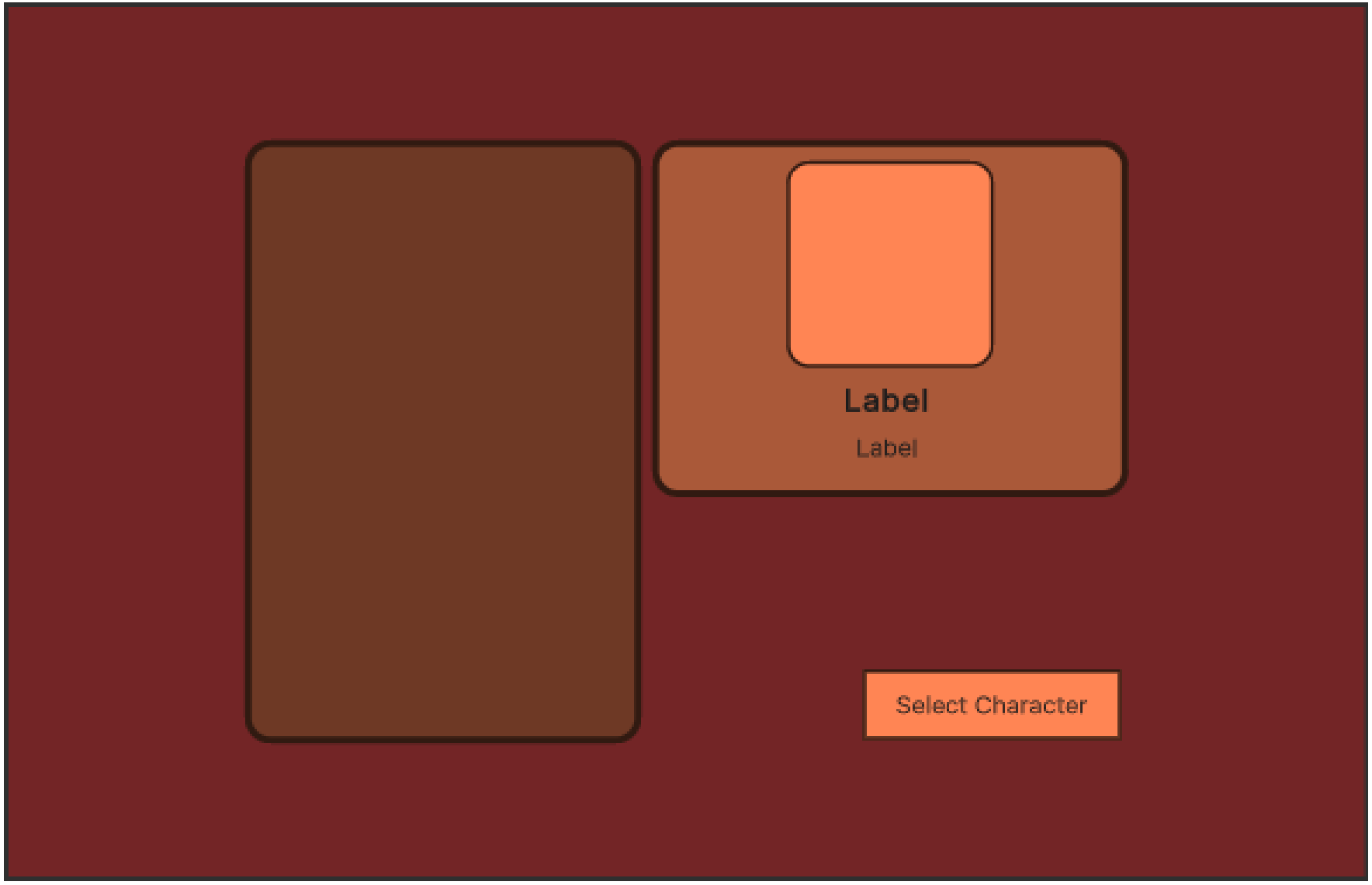
UI Toolkit



UI Toolkit

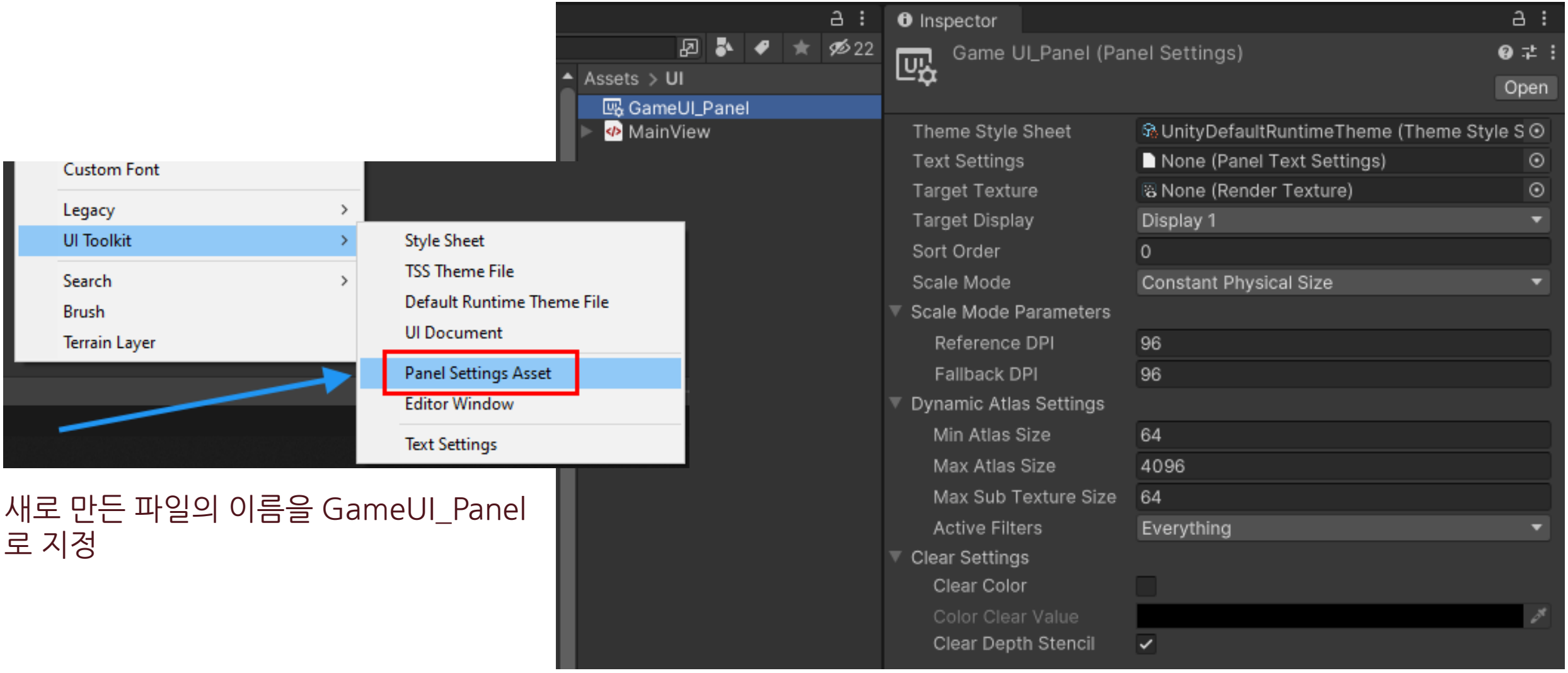

Hierarchy
character-listView.uxml
VisualElement
VisualElement
#CharacterList
VisualElement
VisualElement
VisualElement
#CharacterPortrait
#CharacterName
#CharacterClass
#SelectCharButton
Library
Standard
Project
Label
Button
Toggle
Flex
Align
Size
Width
Height
Size
150
px
auto
Min - Max
Margin & Padding
Text
Background
Color
Image
None (Texture 2D)
Texture
Image Tint
Scale Mode
Slice
0
Border
Color
Width
2px
Radius
0px
Transform

UI Toolkit



UI Toolkit

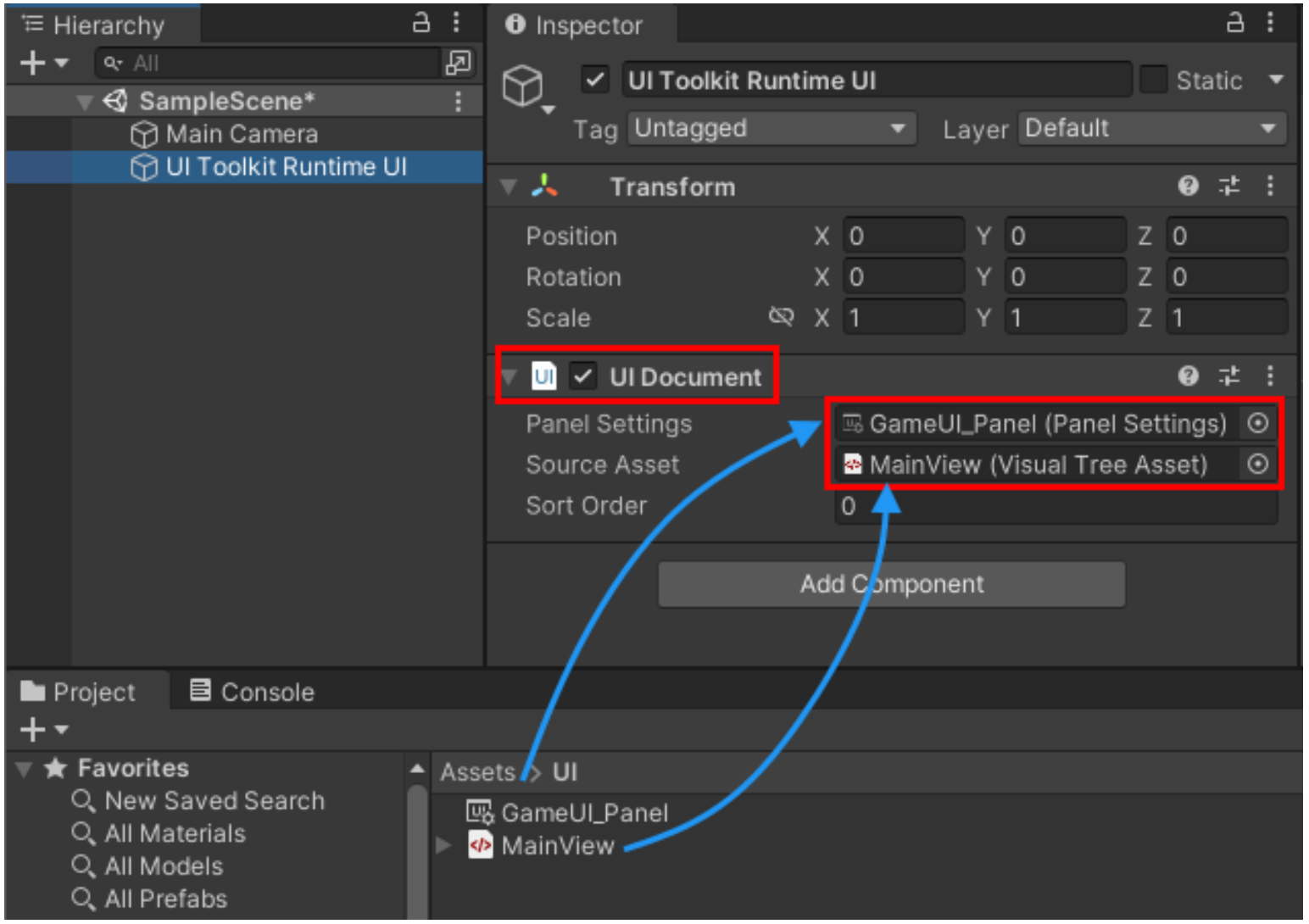
Create > UI Toolkit > Panel Settings Asset



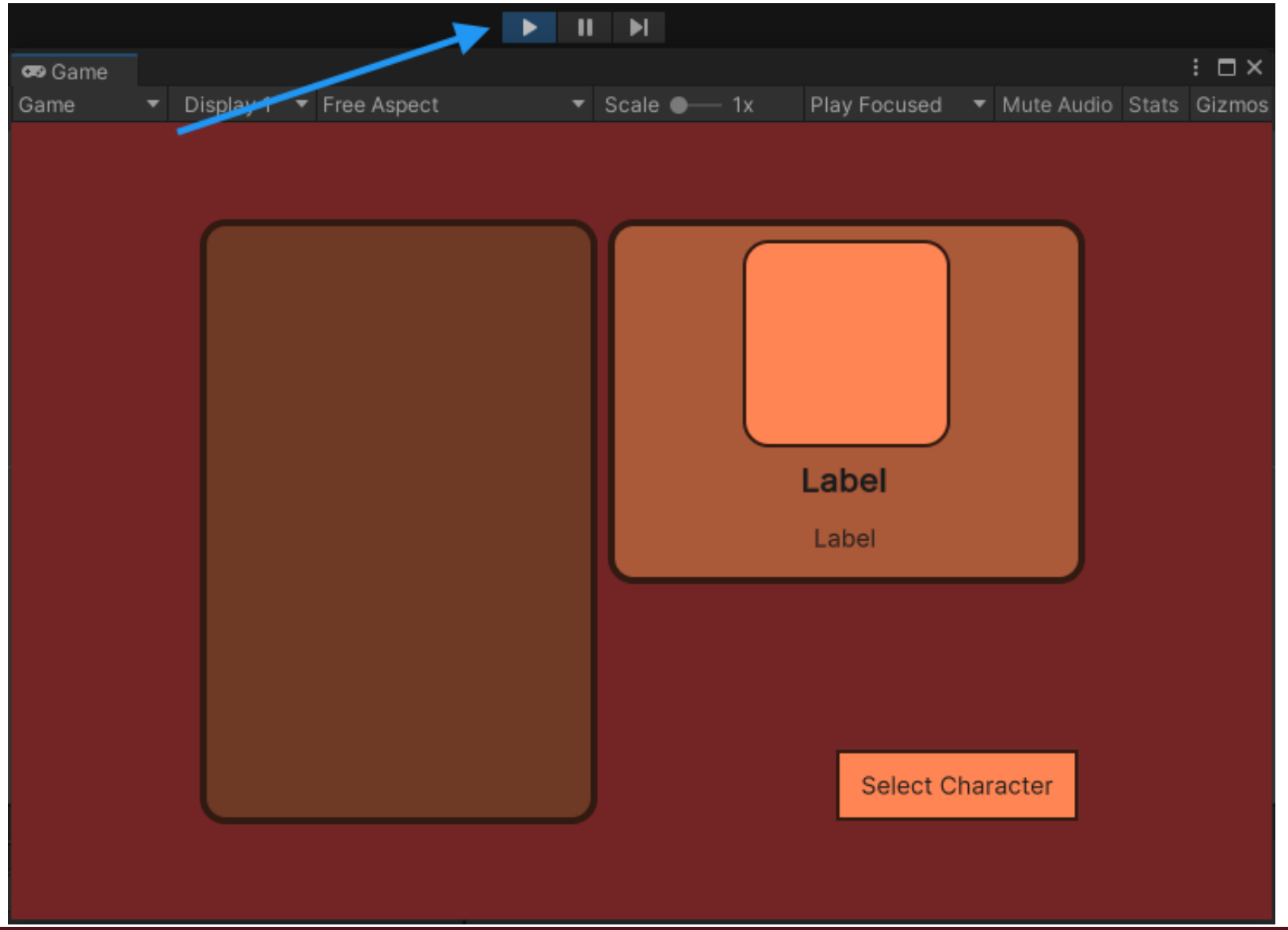
새로 만든 파일의 이름을 GameUI_Panel 로 지정

UI Toolkit

게임 오브젝트를 만들어
 해당 게임 오브젝트에
UIDocument 컴포넌트를 부착



UI Toolkit


Game
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Mute Audio
Stats
Gizmos
Label
Label
Select Character

UI Toolkit

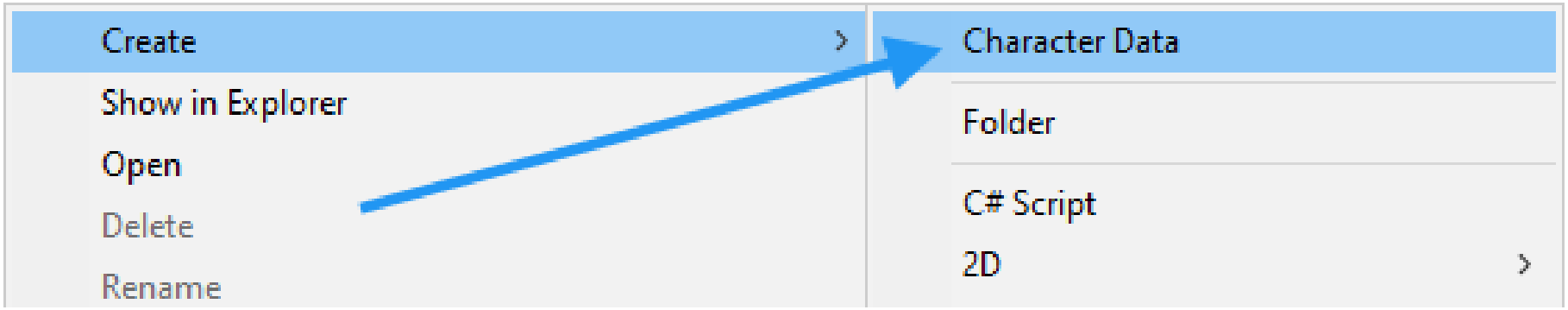
Assets/Scripts/CharacterData.cs

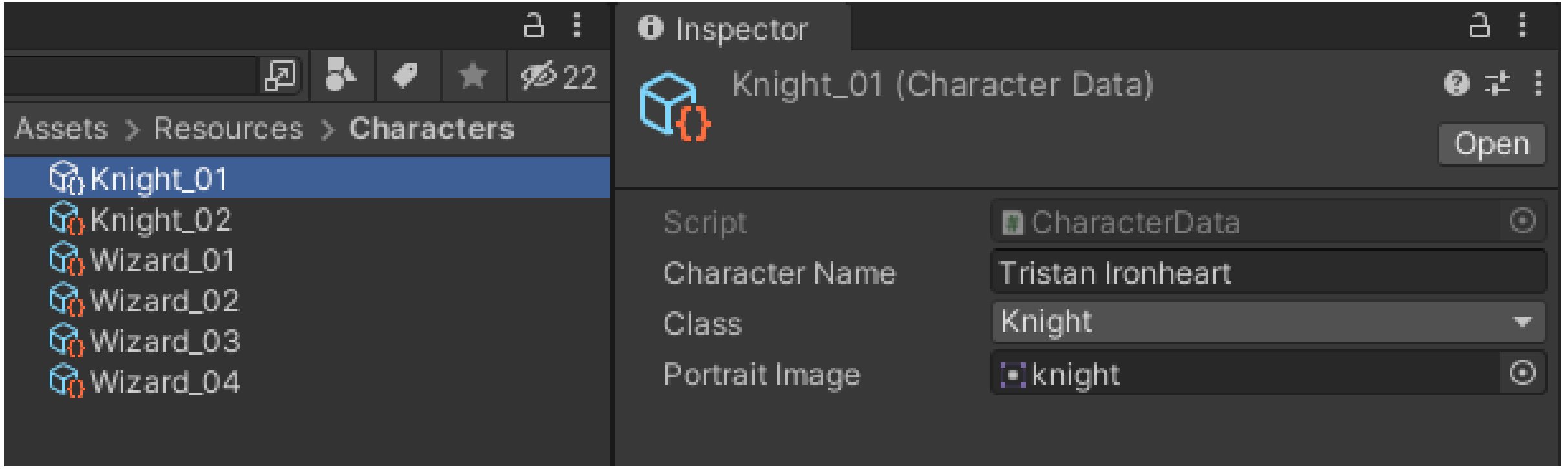
```
using UnityEngine;

public enum ECharacterClass
{
    Knight, Ranger, Wizard
}

[CreateAssetMenu]
public class CharacterData : ScriptableObject
{
    public string m_CharacterName;
    public ECharacterClass m_Class;
    public Sprite m_PortraitImage;
}
```

UI Toolkit

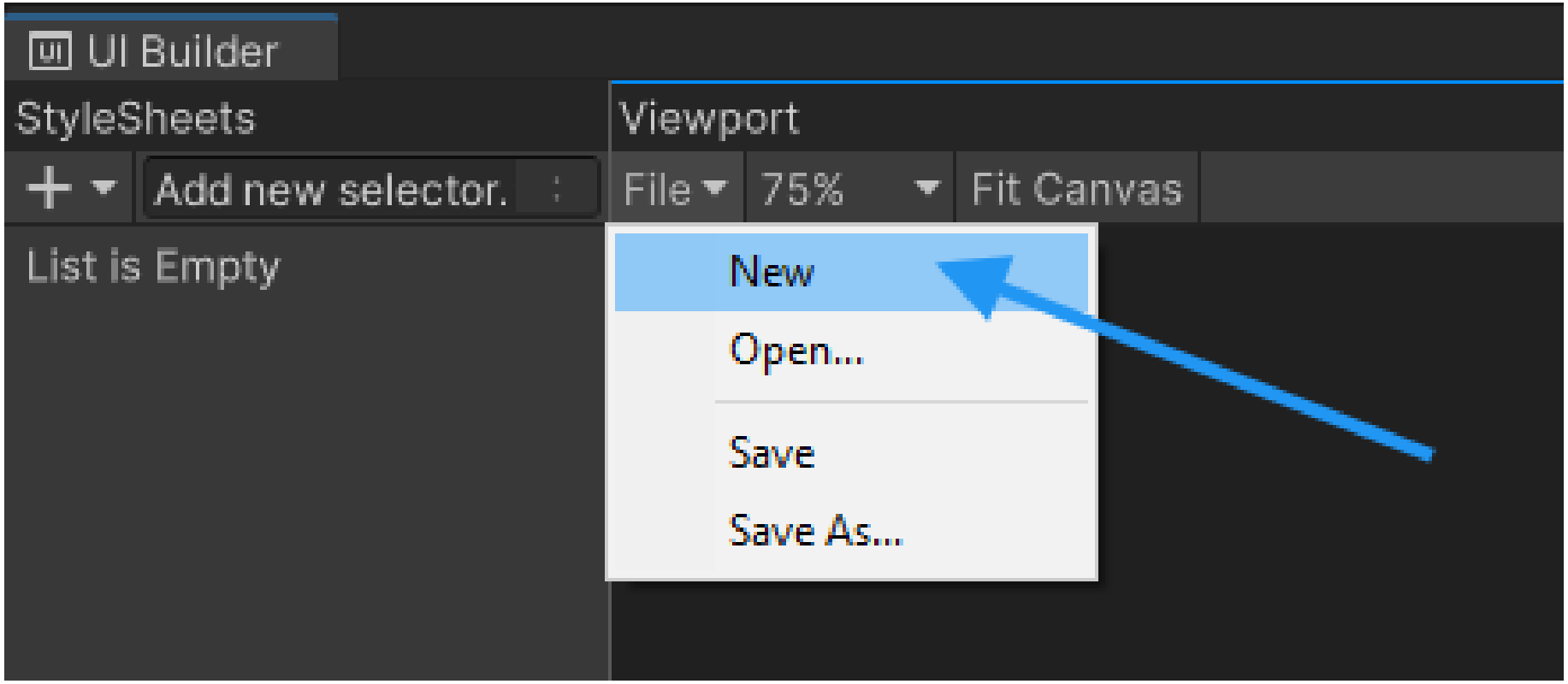

Create
Show in Explorer
Open
Delete
Rename
Character Data
Folder
C# Script
2D


22
Assets > Resources > Characters
Knight_01
Knight_02
Wizard_01
Wizard_02
Wizard_03
Wizard_04
Inspector
Knight_01 (Character Data)
Open
Script
CharacterData
Character Name
Tristan Ironheart
Class
Knight
Portrait Image
knight

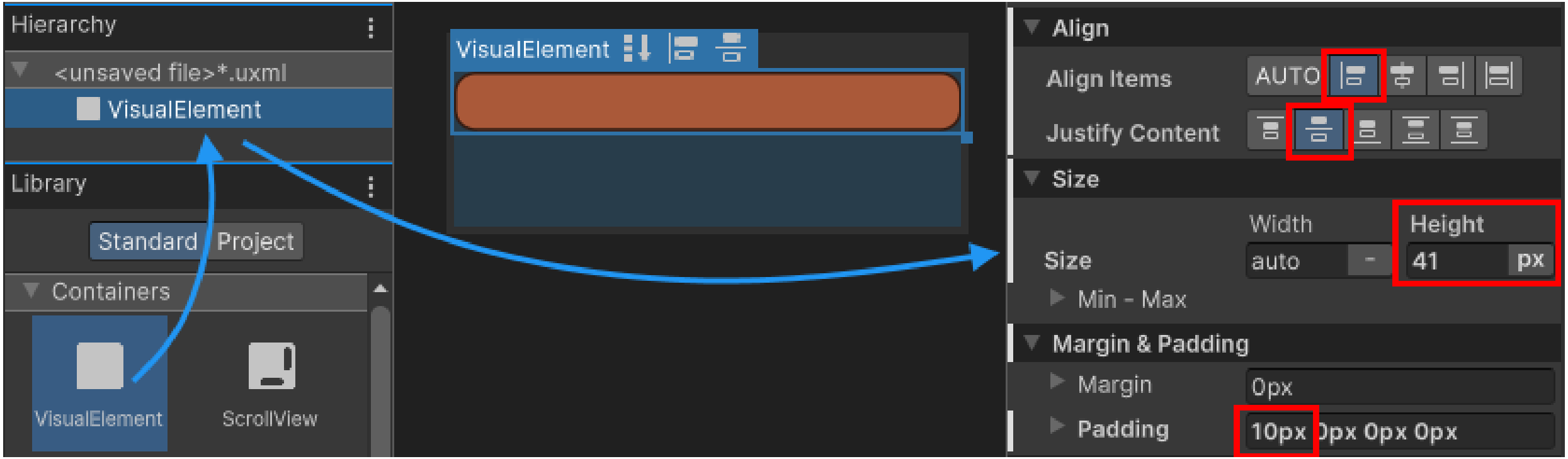
UI Toolkit

리스트 엔트리 UI 템플릿 만들기

Label



UI Toolkit



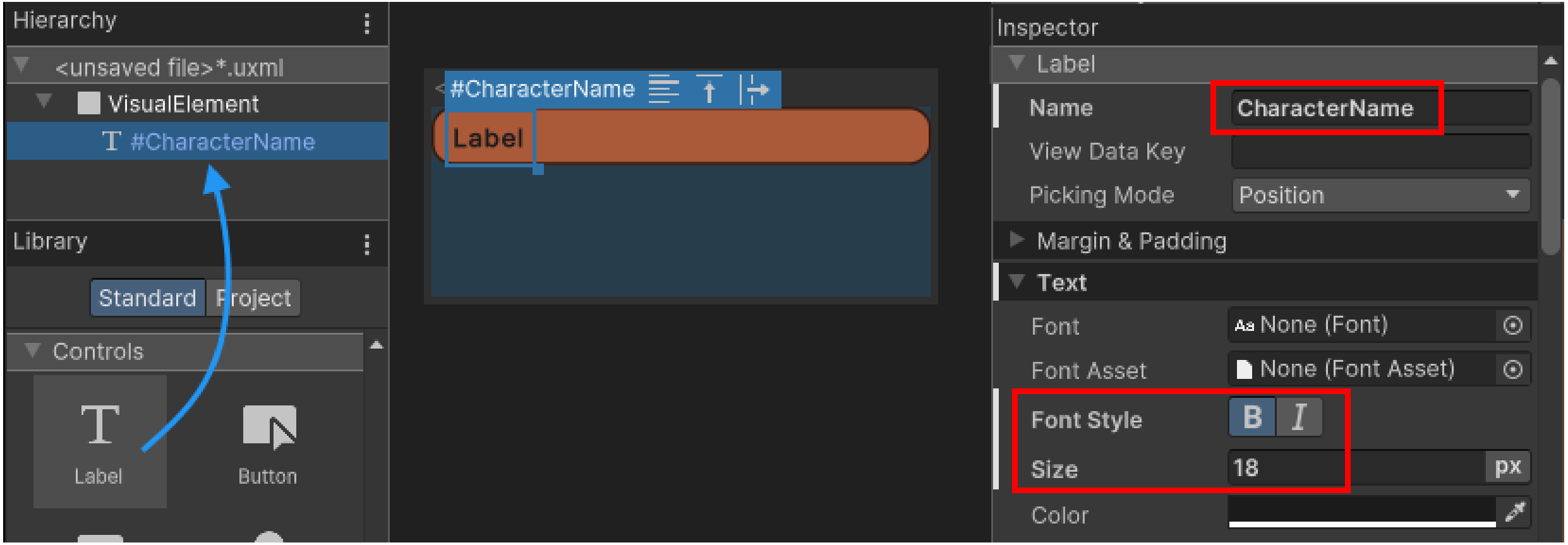
배경을 위한 VisualElement를 추가하고, 높이를 41픽셀로 고정

Align 폴드아웃을 열고 Align Items를 left로, Justify Content를 center로 설정

10픽셀의 왼쪽 패딩을 설정

배경색을 #AA5939로 설정하고 너비가 2픽셀, 반지름이 15픽셀이며 컬러가 #311A11로 설정된 테두리를 추가

UI Toolkit



VisualElement의 자식으로 레이블을 추가 이름을 CharacterName으로 지정

Font Style을 bold로 설정하고 폰트 크기는 18로 설정

Assets/UI/ListEntry.uxml로 저장

UI Toolkit

Assets/Scripts/UI/CharacterListEntryController.cs

```
using UnityEngine.UIElements;

public class CharacterListEntryController
{
    Label m_NameLabel;

    public void SetVisualElement(VisualElement visualElement)
    {
        m_NameLabel = visualElement.Q<Label>("CharacterName");
    }

    public void SetCharacterData(CharacterData characterData)
    {
        m_NameLabel.text = characterData.m_CharacterName;
    }
}
```

UI Toolkit

Assets/Scripts/UI/CharacterListController.cs

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UIElements;

public class CharacterListController
{
    public void InitializeCharacterList(
      VisualElement root,
      VisualTreeAsset listElementTemplate)
    {
    }
}
```

UI Toolkit

Assets/Scripts/UI/MainView.cs

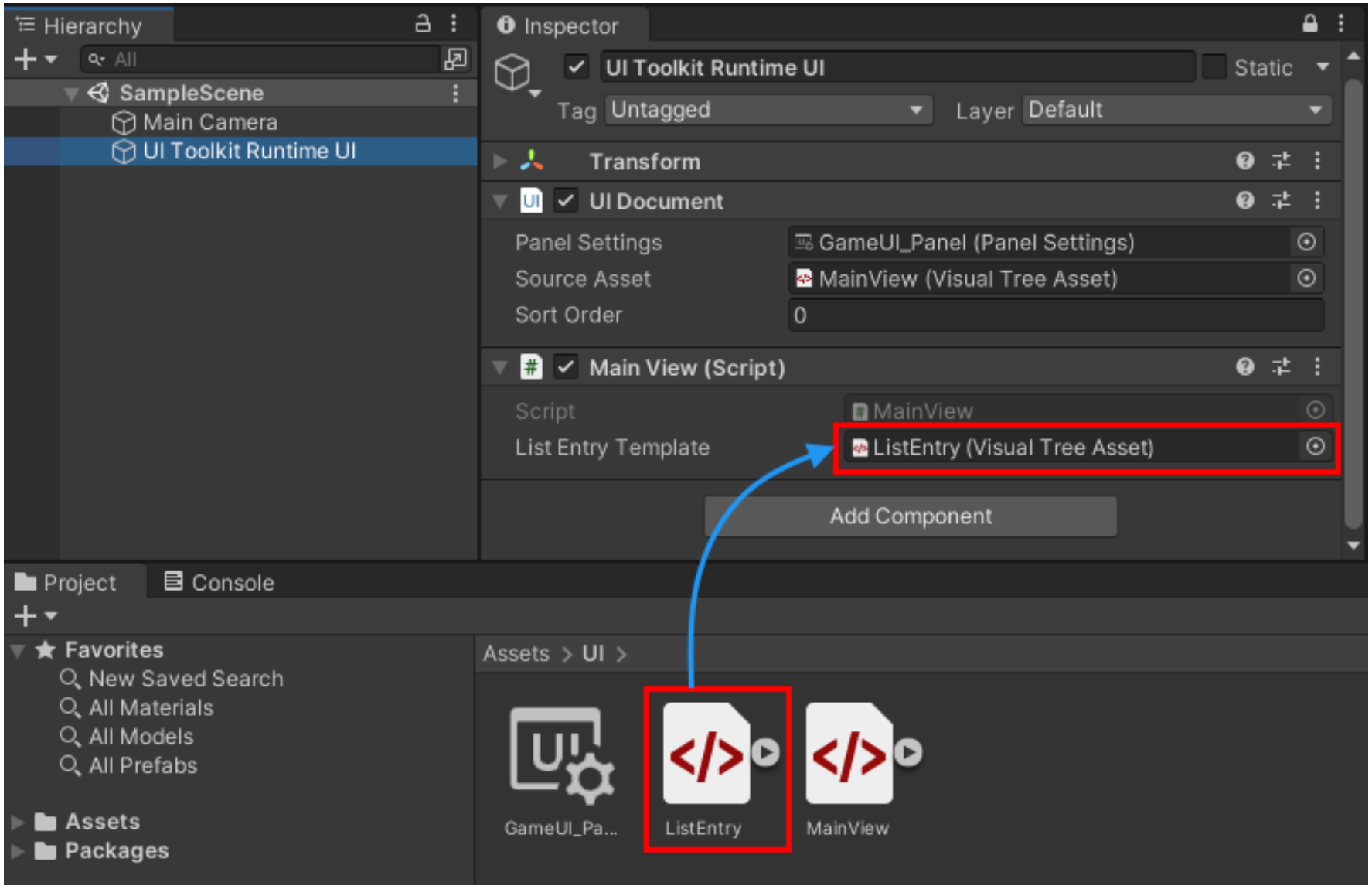
```
using UnityEngine;
using UnityEngine.UIElements;

public class MainView : MonoBehaviour
{
    [SerializeField]
    VisualTreeAsset m_ListEntryTemplate;

    void OnEnable()
    {
        // The UXML is already instantiated by the UIDocument component
        var uiDocument = GetComponent<UIDocument>();

        // Initialize the character list controller
        var characterListController = new CharacterListController();
        characterListController.InitializeCharacterList(
            uiDocument.rootVisualElement,
            m_ListEntryTemplate);
    }
}
```

UI Toolkit


Hierarchy
All
SampleScene
Main Camera
UI Toolkit Runtime UI
Inspector
UI Toolkit Runtime UI
Static
Tag
Untagged
Layer
Default
Transform
UI Document
Panel Settings
GameUI_Panel (Panel Settings)
Source Asset
MainView (Visual Tree Asset)
Sort Order
0
Main View (Script)
Script
MainView
List Entry Template
ListEntry (Visual Tree Asset)
Add Component
Project
Console
Favorites
New Saved Search
All Materials
All Models
All Prefabs
Assets
Packages
Assets > UI >
GameUI_Pa...
ListEntry
MainView

UI Toolkit

CharacterListController .cs

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UIElements;

public class CharacterListController
{
    List<CharacterData> m_AllCharacters;

    // UXML template for list entries
    VisualTreeAsset m_ListEntryTemplate;

    // UI element references
    ListView m_CharacterList;
    Label m_CharClassLabel;
    Label m_CharNameLabel;
    VisualElement m_CharPortrait;
    Button m_SelectCharButton;
```

UI Toolkit

```
    void EnumerateAllCharacters()
    {
        m_AllCharacters = new List<CharacterData>();
        m_AllCharacters.AddRange(Resources.LoadAll<CharacterData>("Characters"));
    }

    public void InitializeCharacterList(VisualElement root, VisualTreeAsset
listElementTemplate)
    {
        EnumerateAllCharacters();

        // Store a reference to the template for the list entries
        m_ListEntryTemplate = listElementTemplate;

        // Store a reference to the character list element
        m_CharacterList = root.Q<ListView>("CharacterList");

        // Store references to the selected character info elements
        m_CharClassLabel = root.Q<Label>("CharacterClass");
        m_CharNameLabel = root.Q<Label>("CharacterName");
```

UI Toolkit

```
        m_CharPortrait = root.Q<VisualElement>("CharacterPortrait");

        // Store a reference to the select button
        m_SelectCharButton = root.Q<Button>("SelectCharButton");

        FillCharacterList();

        // Register to get a callback when an item is selected
        m_CharacterList.onSelectionChange += OnCharacterSelected;
    }

    void FillCharacterList()
    {
        // Set up a make item function for a list entry
        m_CharacterList.makeItem = () =>
        {
            // Instantiate the UXML template for the entry
            var newListEntry = m_ListEntryTemplate.Instantiate();
```

UI Toolkit

```
            // Instantiate a controller for the data
            var newListEntryLogic = new CharacterListEntryController();

            // Assign the controller script to the visual element
            newListEntry.userData = newListEntryLogic;

            // Initialize the controller script
            newListEntryLogic.SetVisualElement(newListEntry);

            // Return the root of the instantiated visual tree
            return newListEntry;
        };

        // Set up bind function for a specific list entry
        m_CharacterList.bindItem = (item, index) =>
        {
            (item.userData as
CharacterListEntryController).SetCharacterData(m_AllCharacters[index]);
        };
```

UI Toolkit

```
        // Set a fixed item height
        m_CharacterList.fixedItemHeight = 45;

        // Set the actual item's source list/array
        m_CharacterList.itemsSource = m_AllCharacters;
    }

    void OnCharacterSelected(IEnumerable<object> selectedItems)
    {
        // Get the currently selected item directly from the ListView
        var selectedCharacter = m_CharacterList.selectedItem as CharacterData;

        // Handle none-selection (Escape to deselect everything)
        if (selectedCharacter == null)
        {
            // Clear
            m_CharClassLabel.text = "";
            m_CharNameLabel.text = "";
            m_CharPortrait.style.backgroundImage = null;
```

UI Toolkit

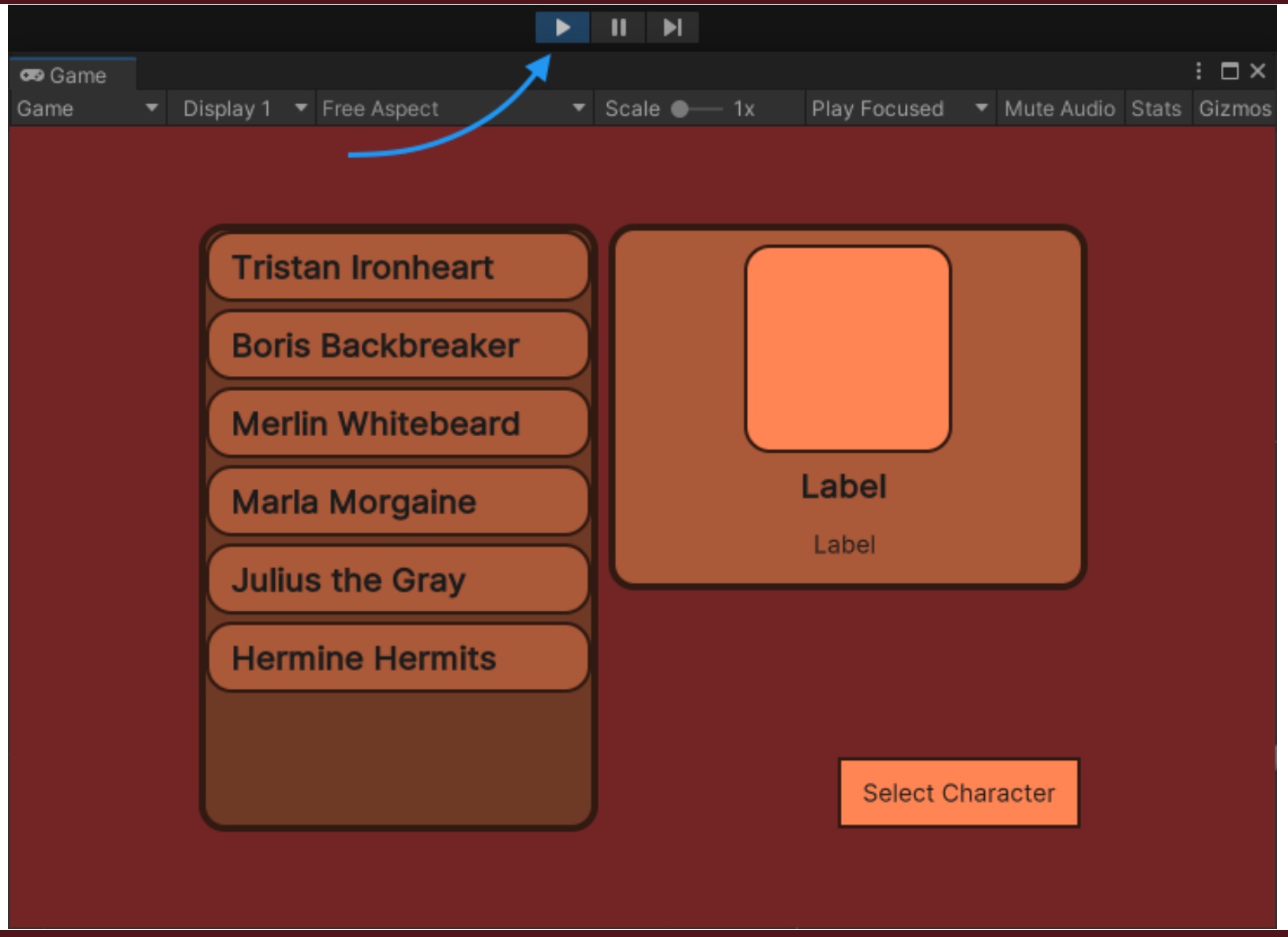
```
            // Disable the select button
            m_SelectCharButton.SetEnabled(false);

            return;
        }

        // Fill in character details
        m_CharClassLabel.text = selectedCharacter.m_Class.ToString();
        m_CharNameLabel.text = selectedCharacter.m_CharacterName;
        m_CharPortrait.style.backgroundImage =
          new StyleBackground(selectedCharacter.m_PortraitImage);

        // Enable the select button
        m_SelectCharButton.SetEnabled(true);
    }
}
```

UI Toolkit


Game
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Mute Audio
Stats
Gizmos
Tristan Ironheart
Boris Backbreaker
Merlin Whitebeard
Marla Morgaine
Julius the Gray
Hermine Hermits
Label
Label
Select Character