



Unity 를 이용한 2D 게임프로그래밍

Lecture 4 User Interface로 퀴즈 게임 만들기

강영민

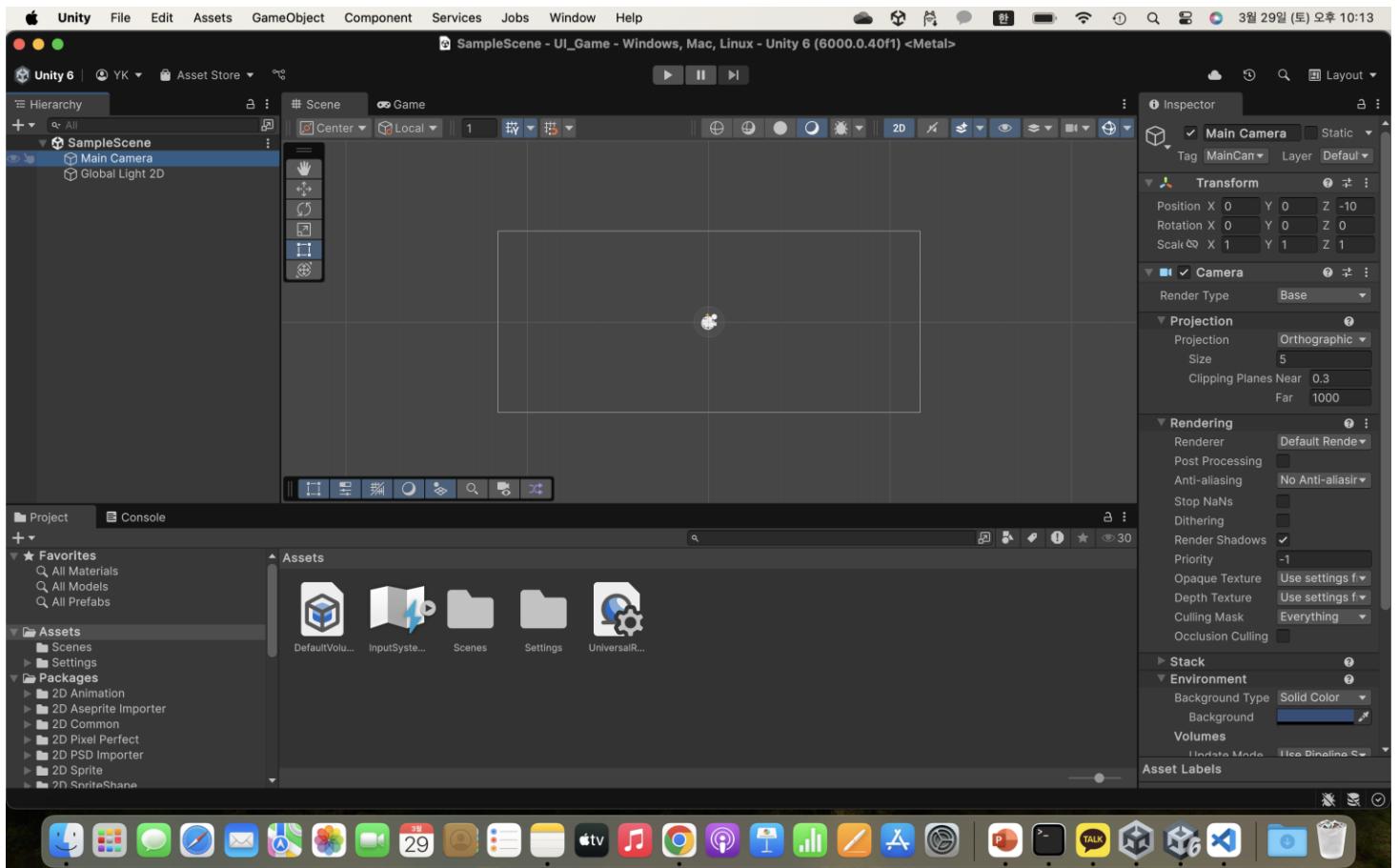
동명대학교 게임공학과

학습목표

- User Interface에 대해 이해한다.
- 캔버스의 개념을 이해하고 캔버스 공간과 월드 공간의 차이를 이해한다.
- 퀴즈 게임을 통해 사용자 인터페이스를 제어할 수 있다.

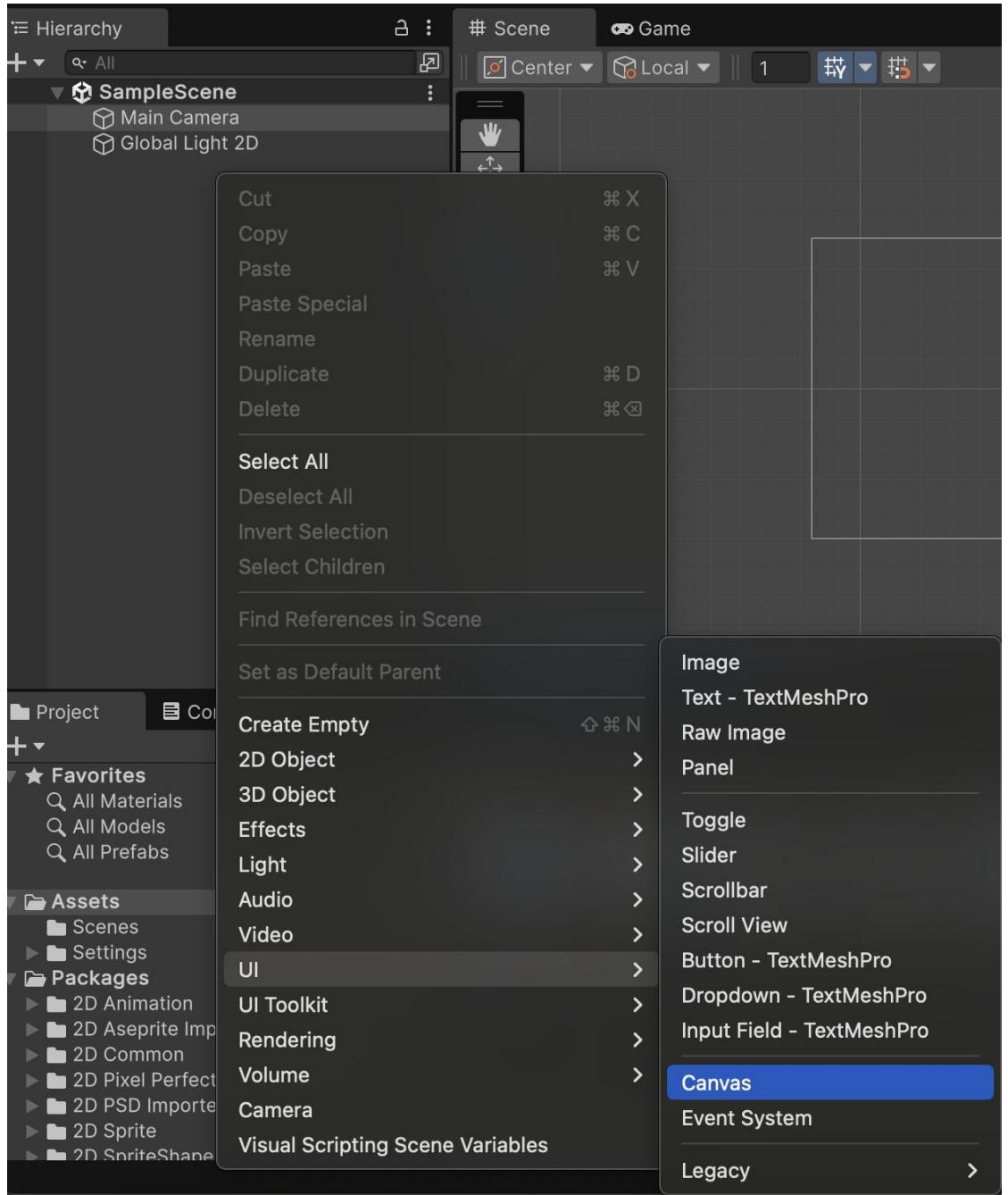
프로젝트를 생성하자

- 2차원 게임으로 생성하자



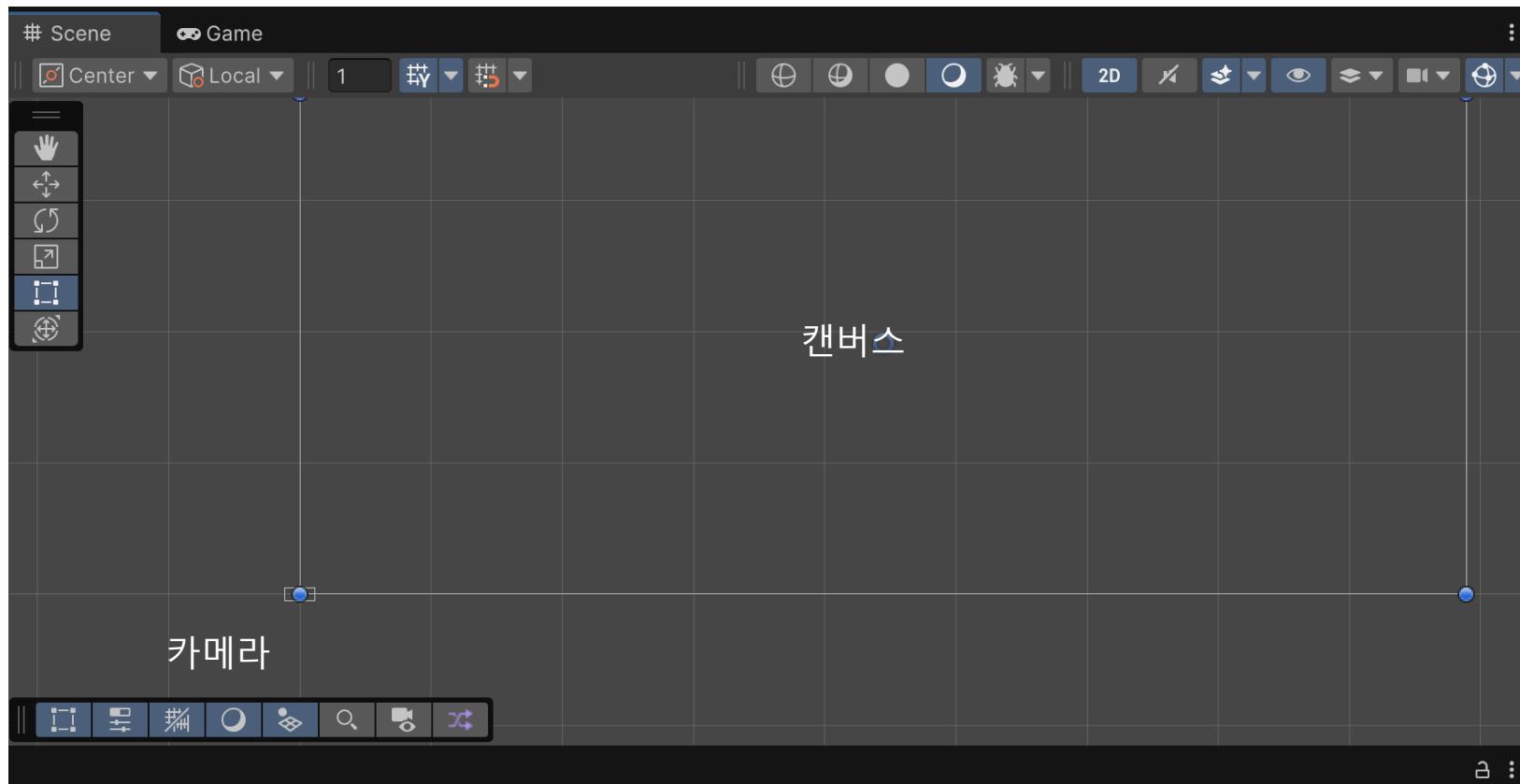
캔버스(Canvas)

- Hierarchy
 - Create → UI → Canvas
- UI 요소들이 캔버스에 실림
 - 캔버스는 스크린 공간에 존재
- 복수의 캔버스도 설치 가능

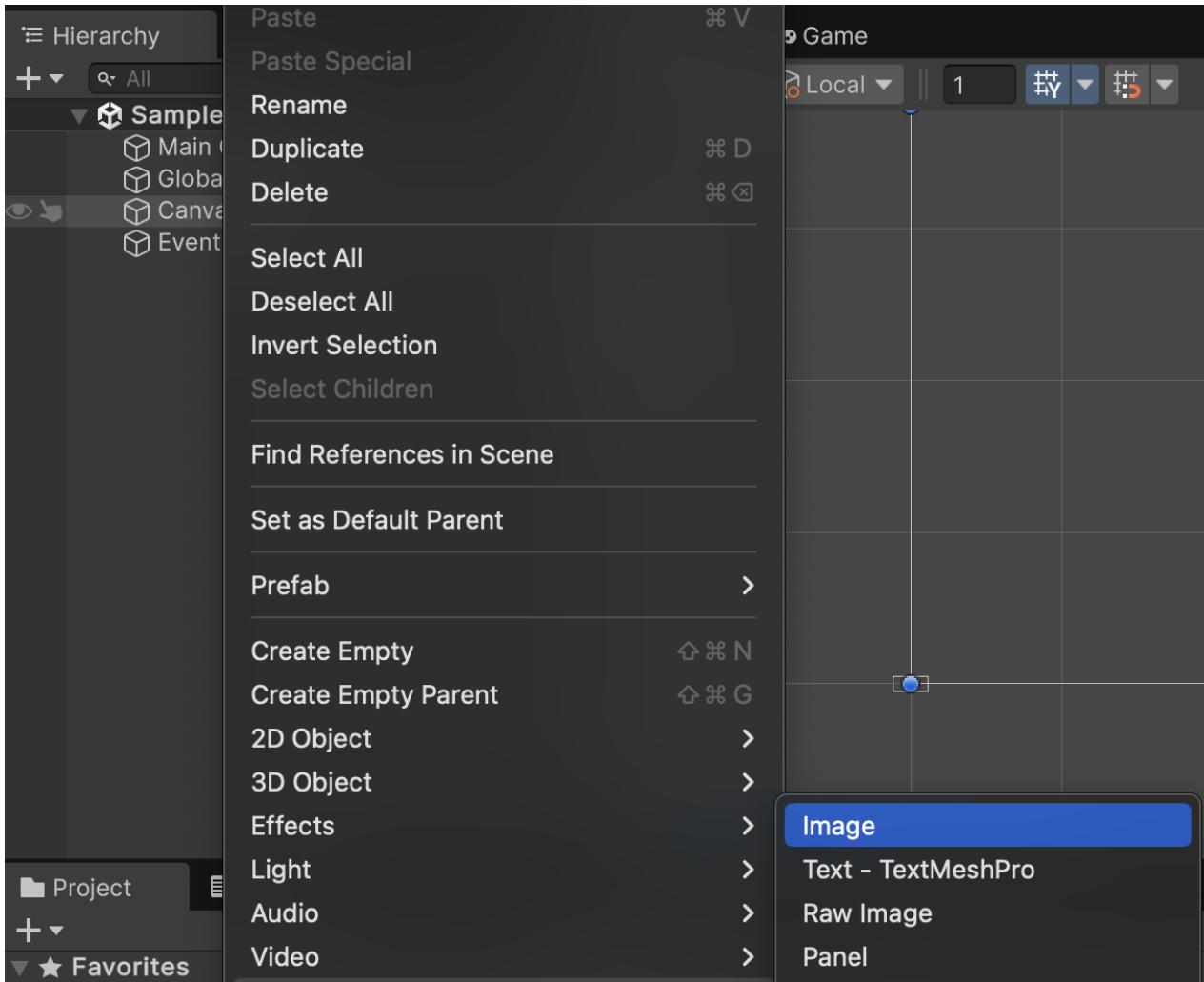


카메라 공간과 캔버스 공간

캔버스는
일반적 게임 객체의
변환과 다른 특성을 가짐

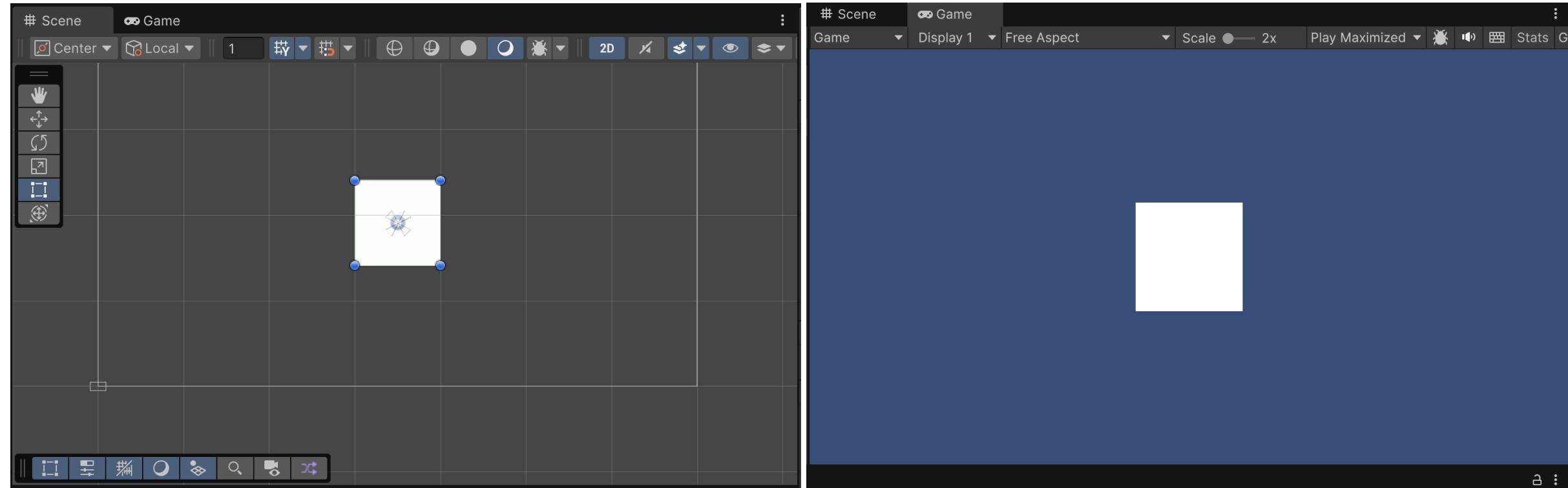


캔버스의 자식 객체로 UI Image 생성

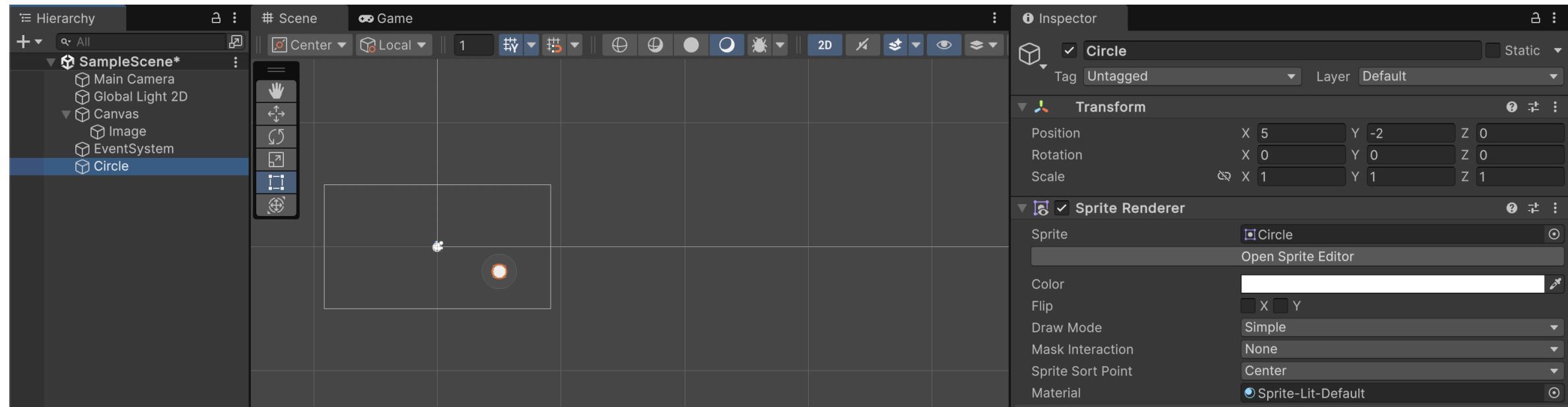


캔버스 공간에 이미지 존재 / 카메라 공간 밖

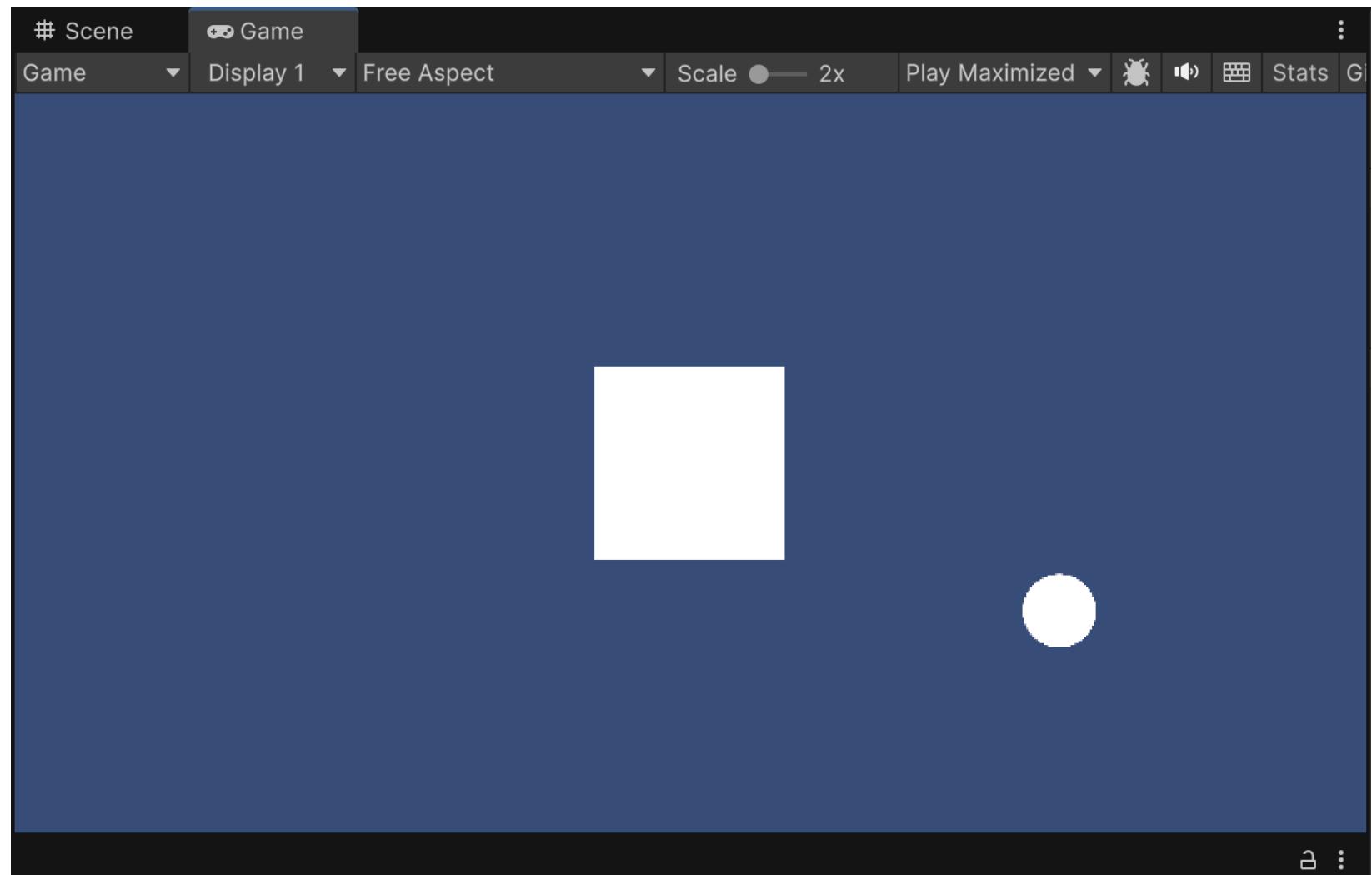
- 게임 화면을 보면 캔버스 공간이 보임



카메라 공간에 원을 하나 넣어 보자

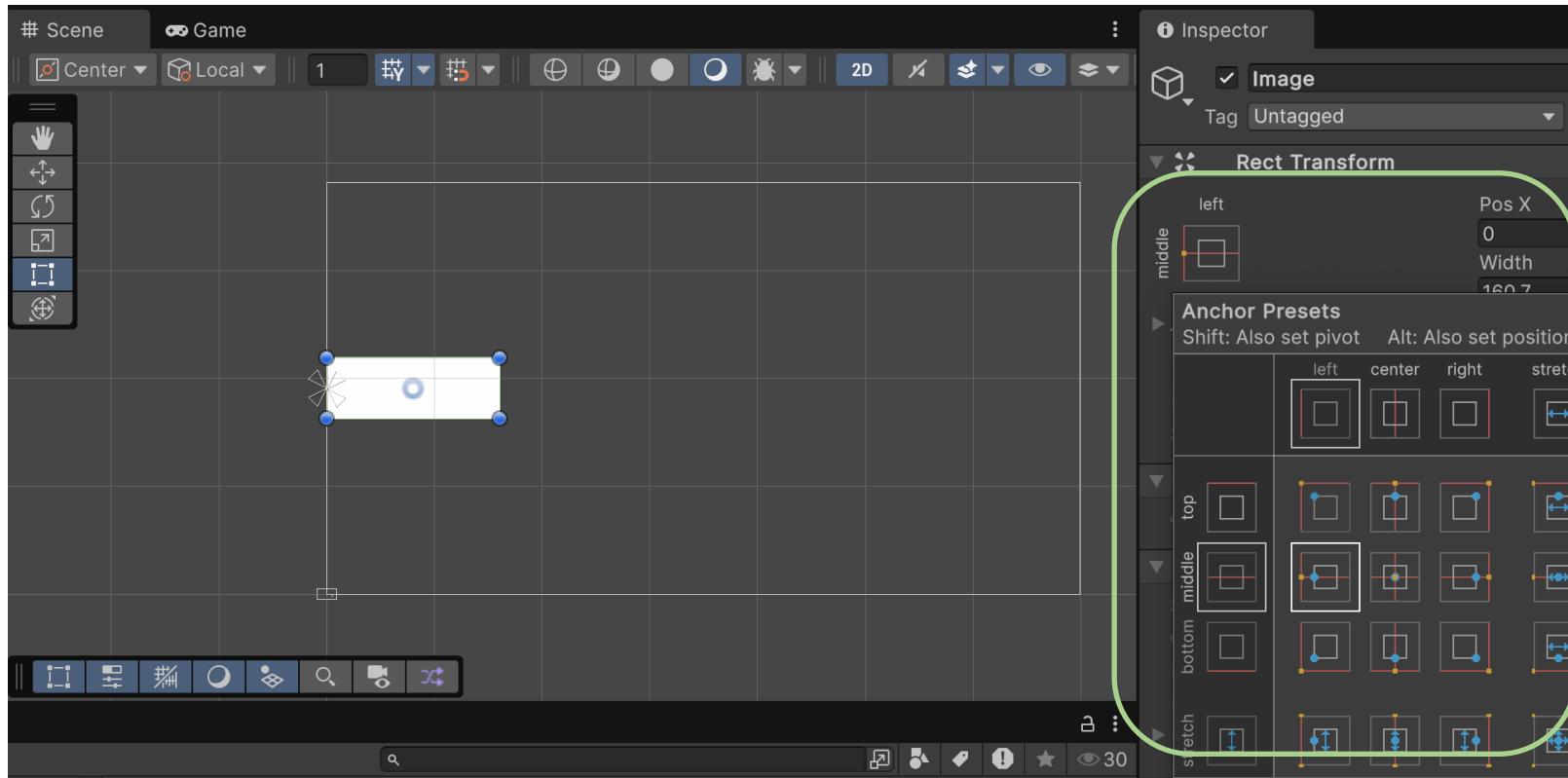


게임 화면은 캔버스 공간과 카메라 공간이 같이 보임

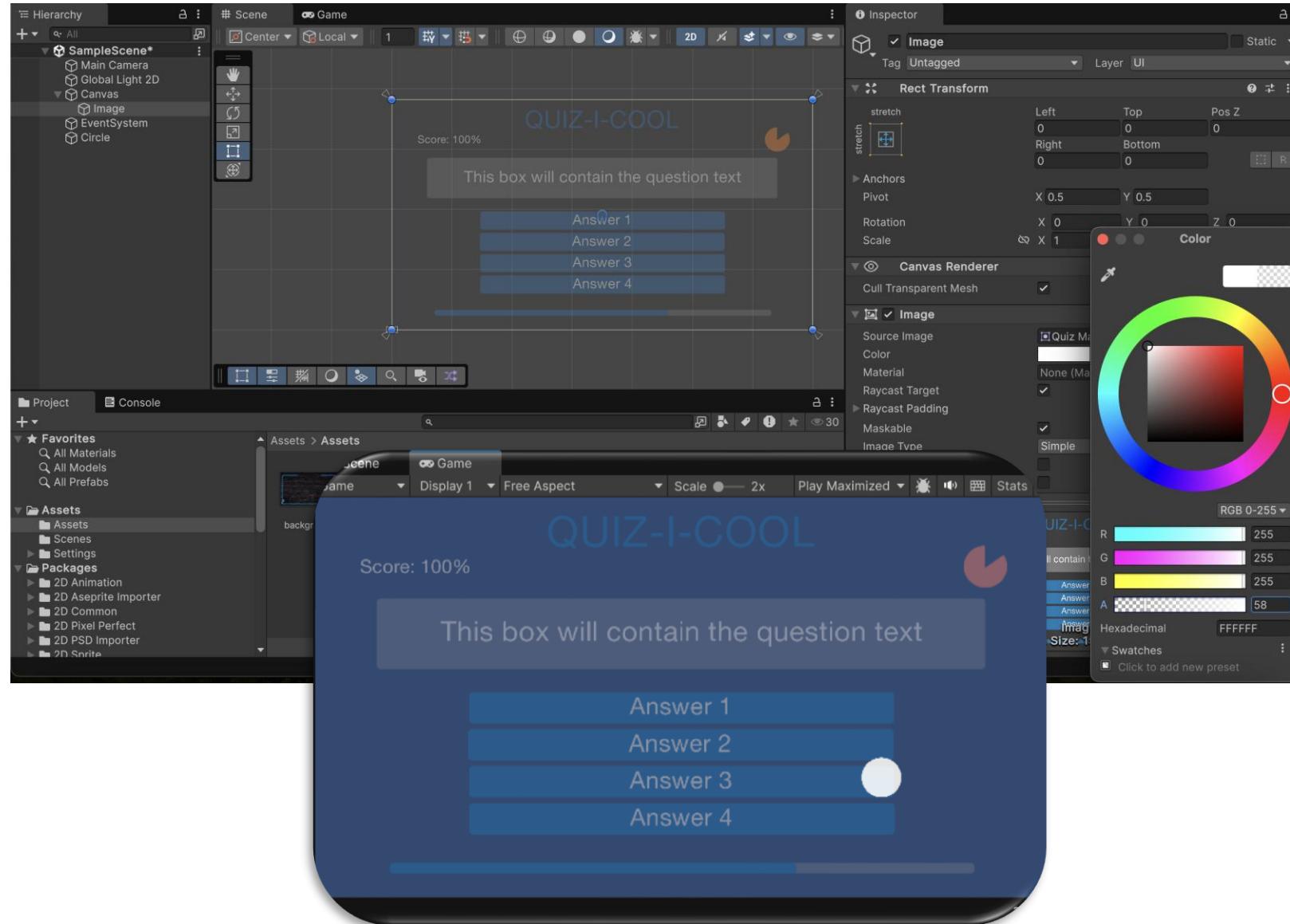


Anchor와 Pivot 테스트해 보기

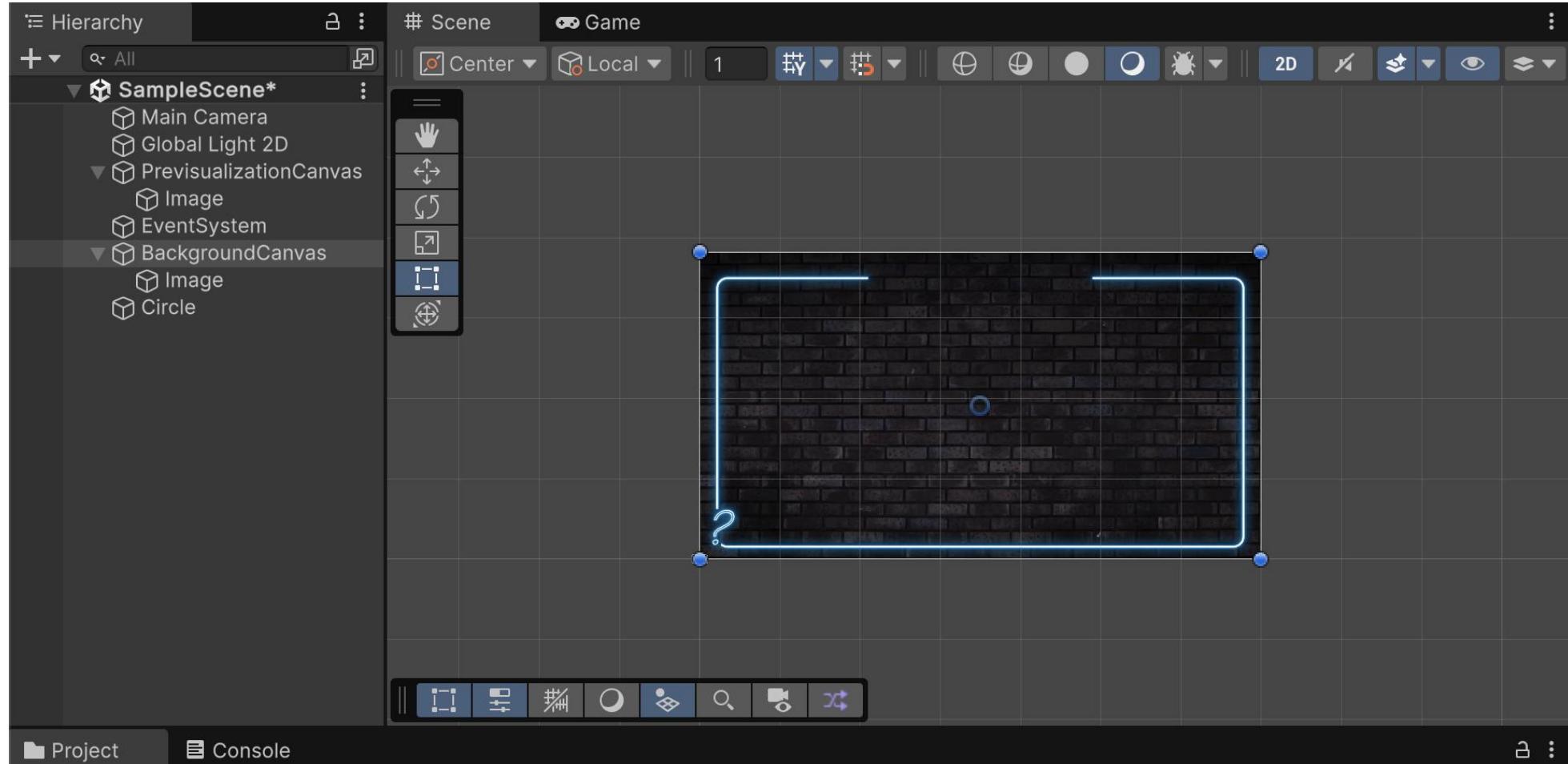
- Anchor Preset + (Shift / Alt)



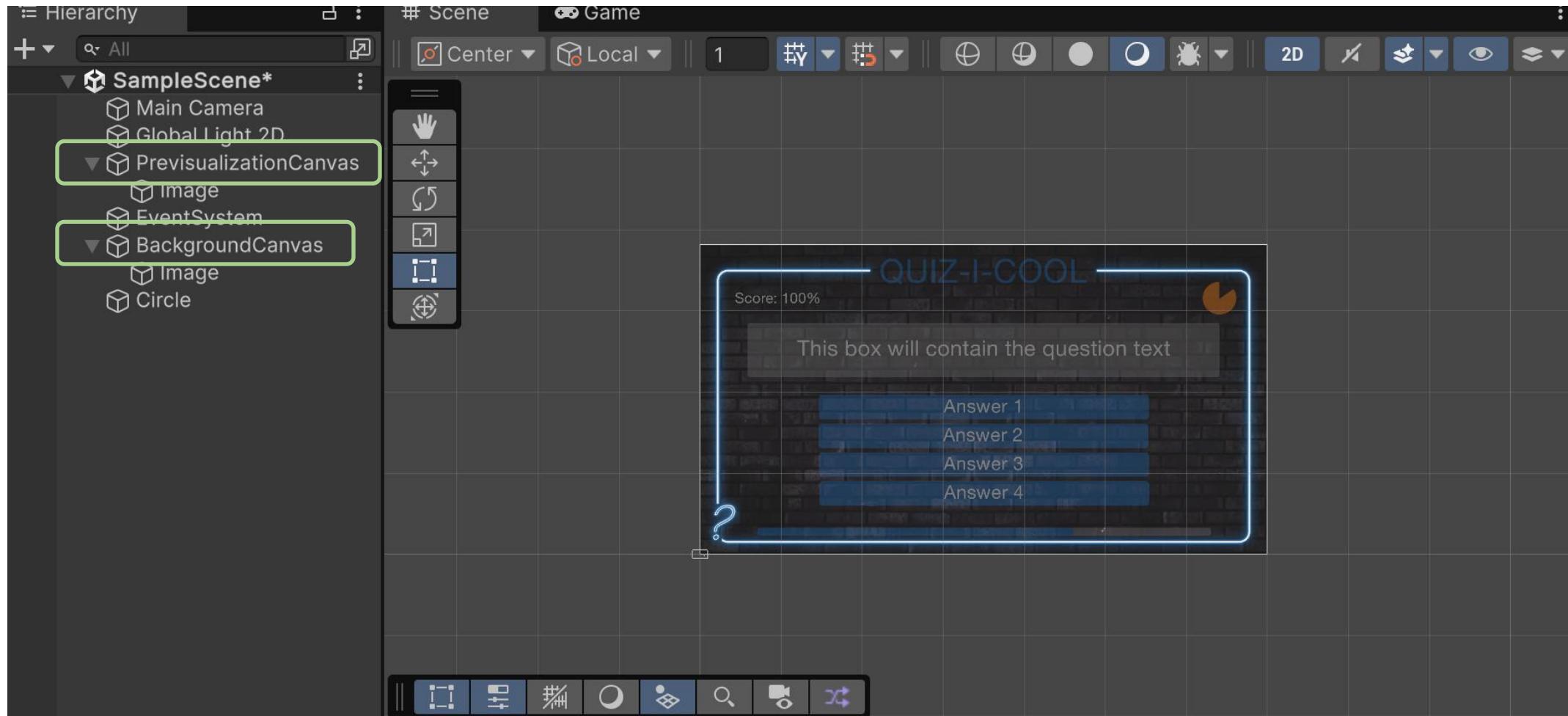
UI 캔버스에 놓인 이미지에 스프라이트 적용/알파 변경



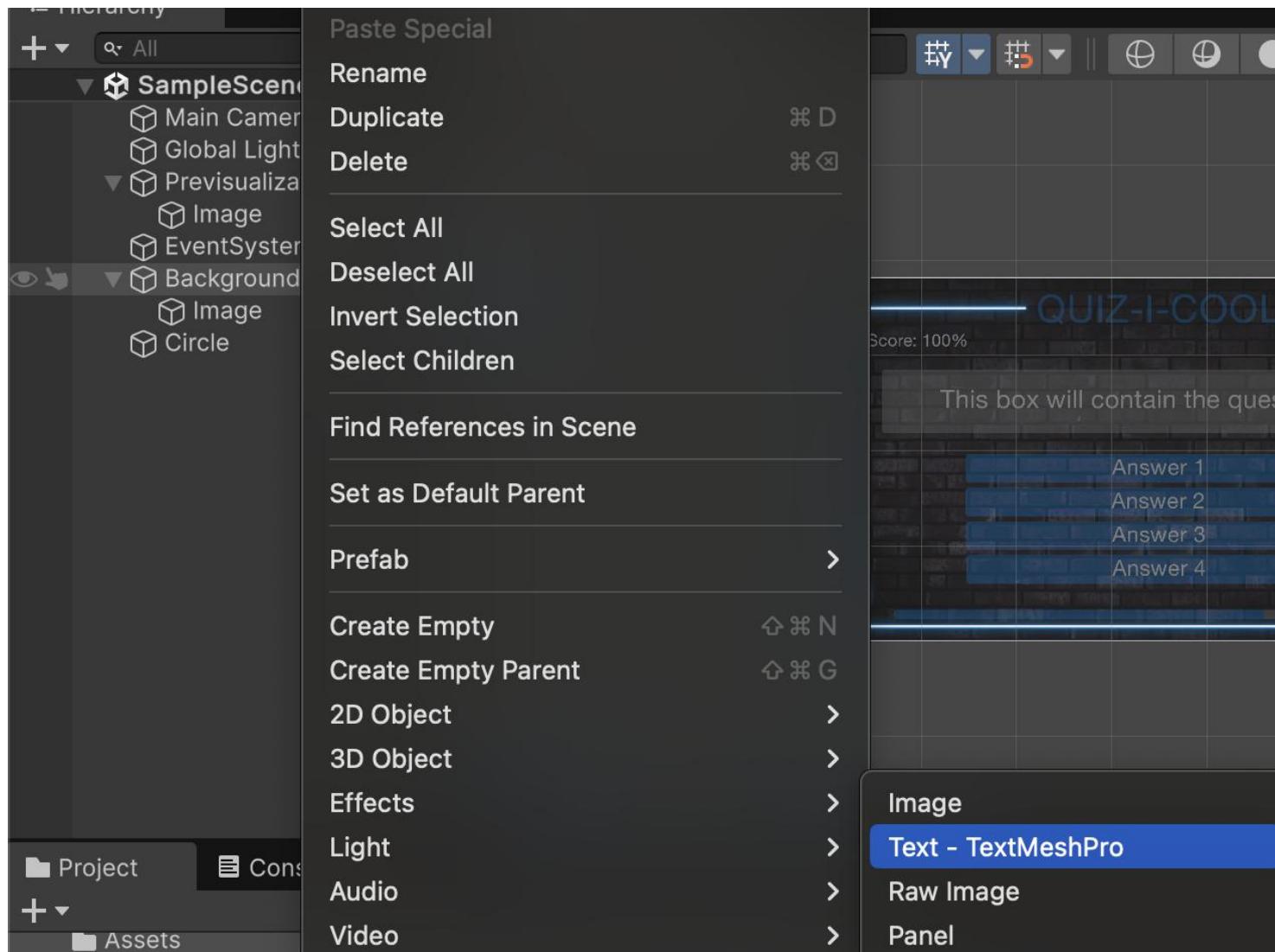
두 번째 캔버스 생성 (Background)



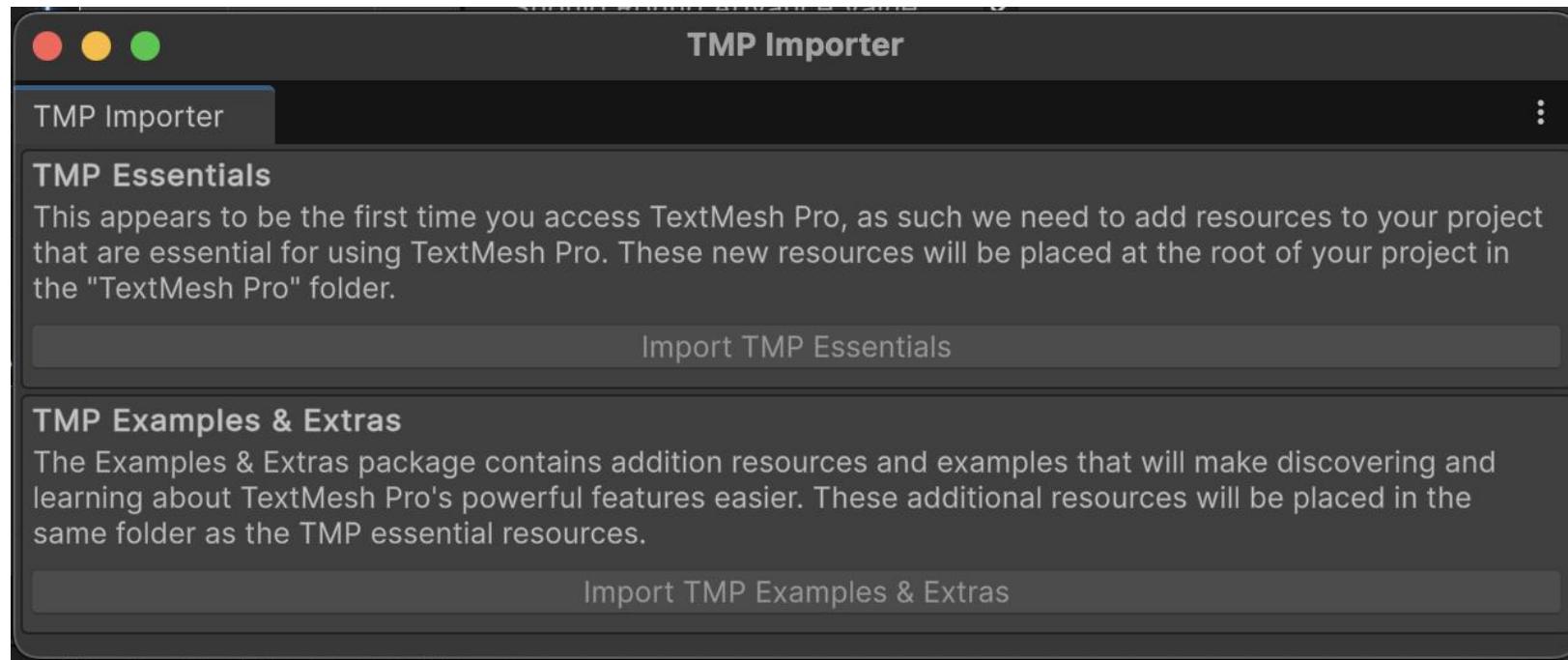
두 캔버스가 겹쳐서 보이게 만들어 보라



배경 캔버스에 텍스트 메시 프로를 추가한다

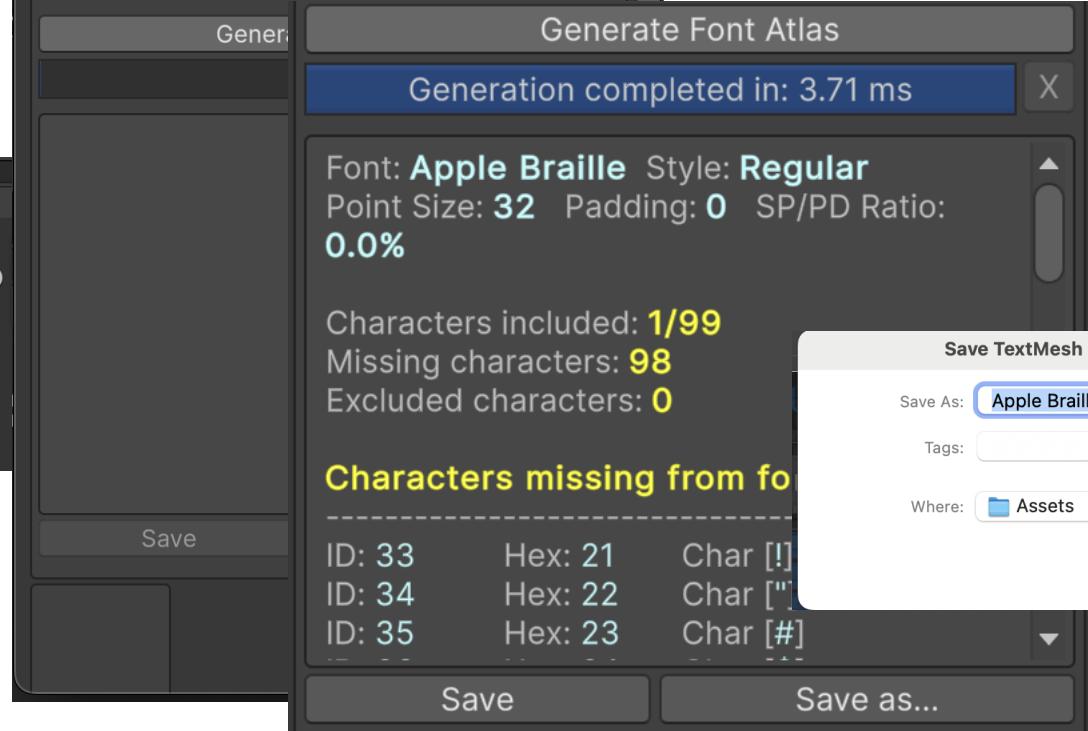
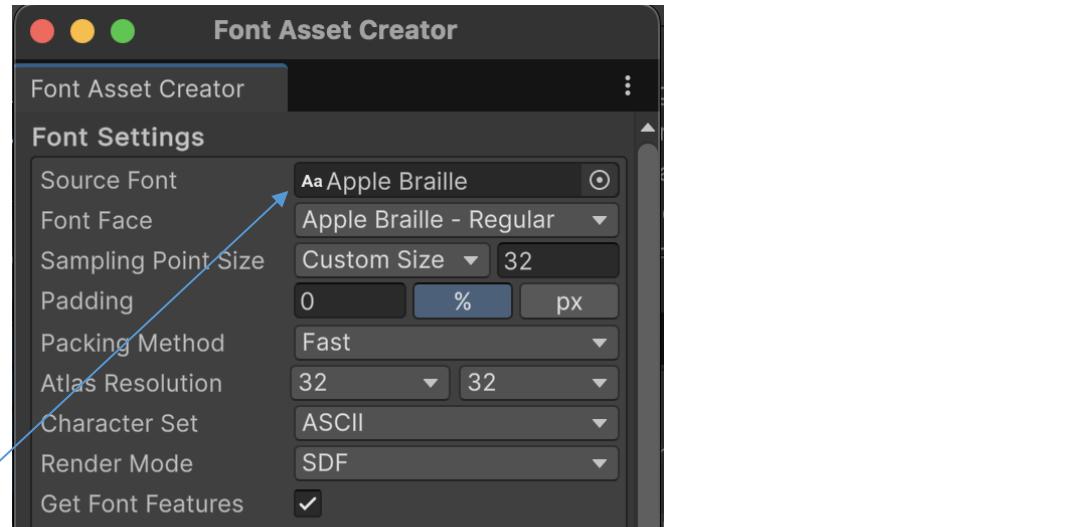
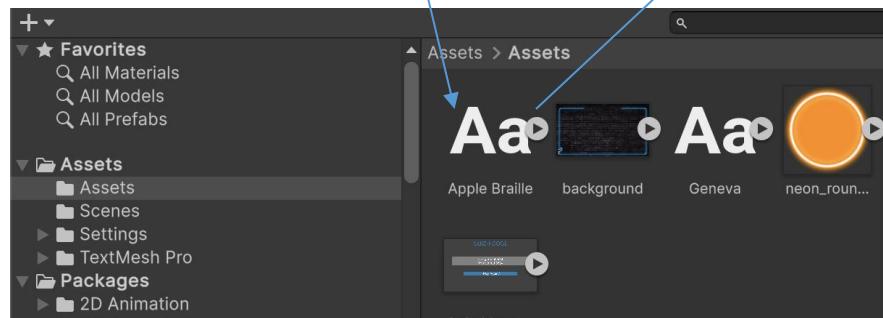


Text Mesh Pro를 처음 사용할 경우 필요한 자원 가져오기

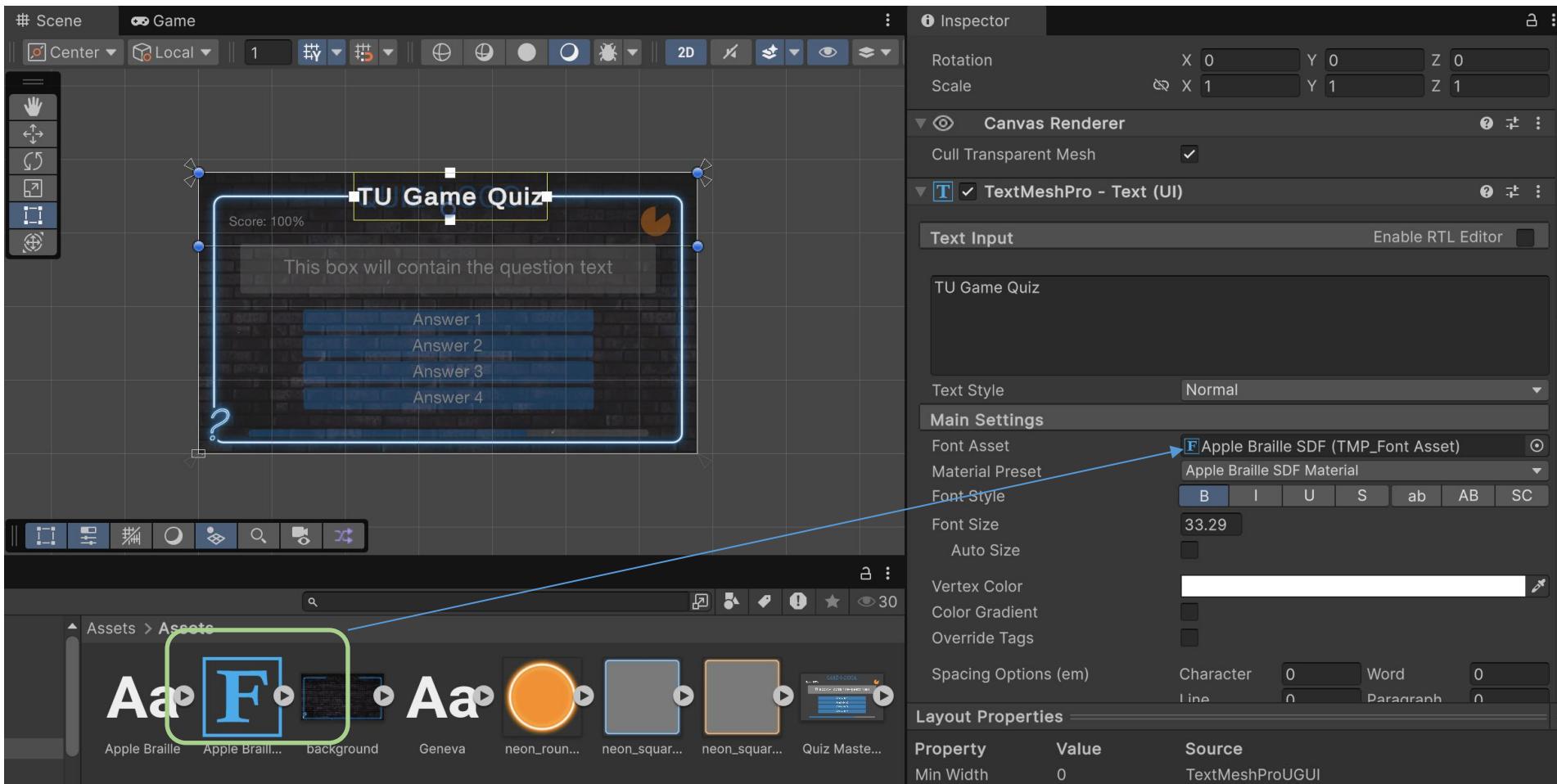


폰트 Asset 생성 필요

Font file

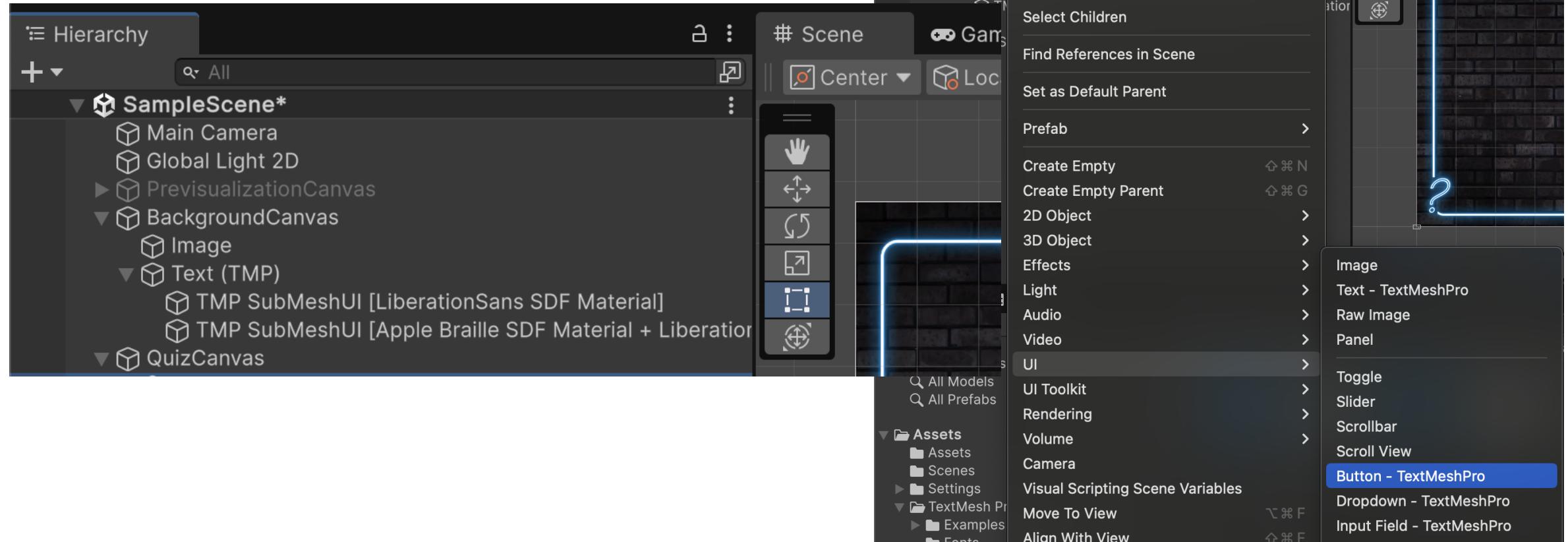


폰트를 적용하여 TMP 객체 다루기



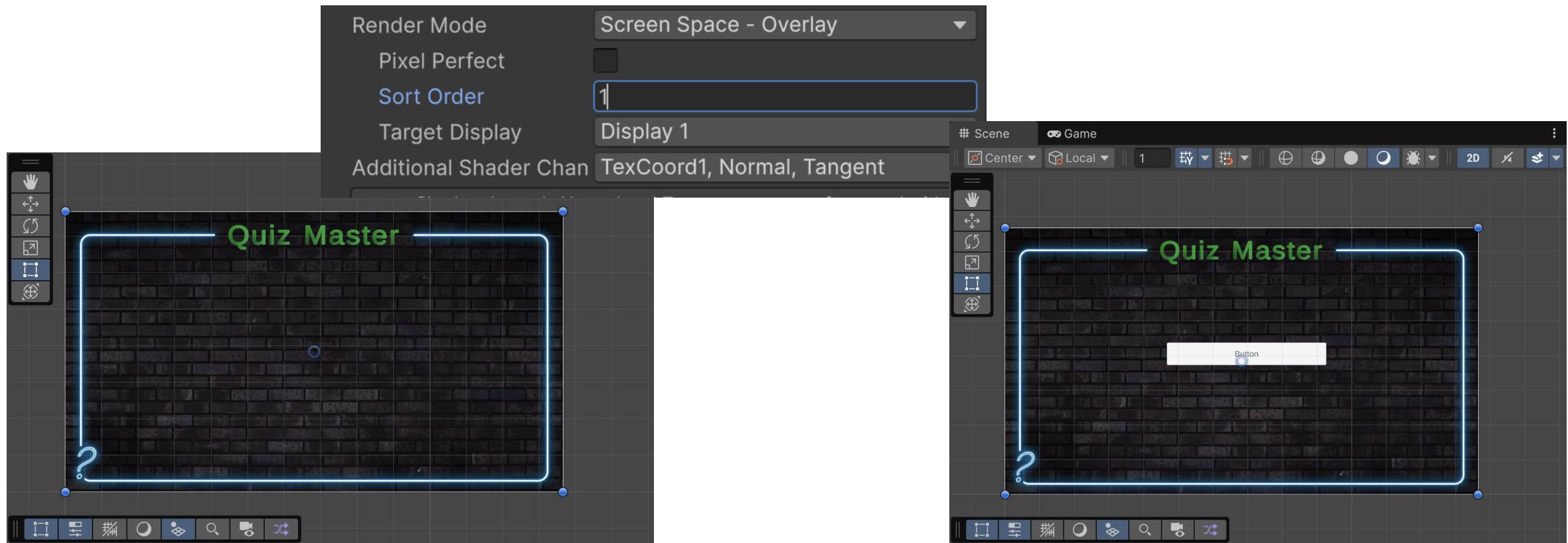
Canvas 추가하고 버튼 달아보기

- 캔버스 아래에 버튼 UI 객체들을 만든다
- UI → Button – TextMeshPro

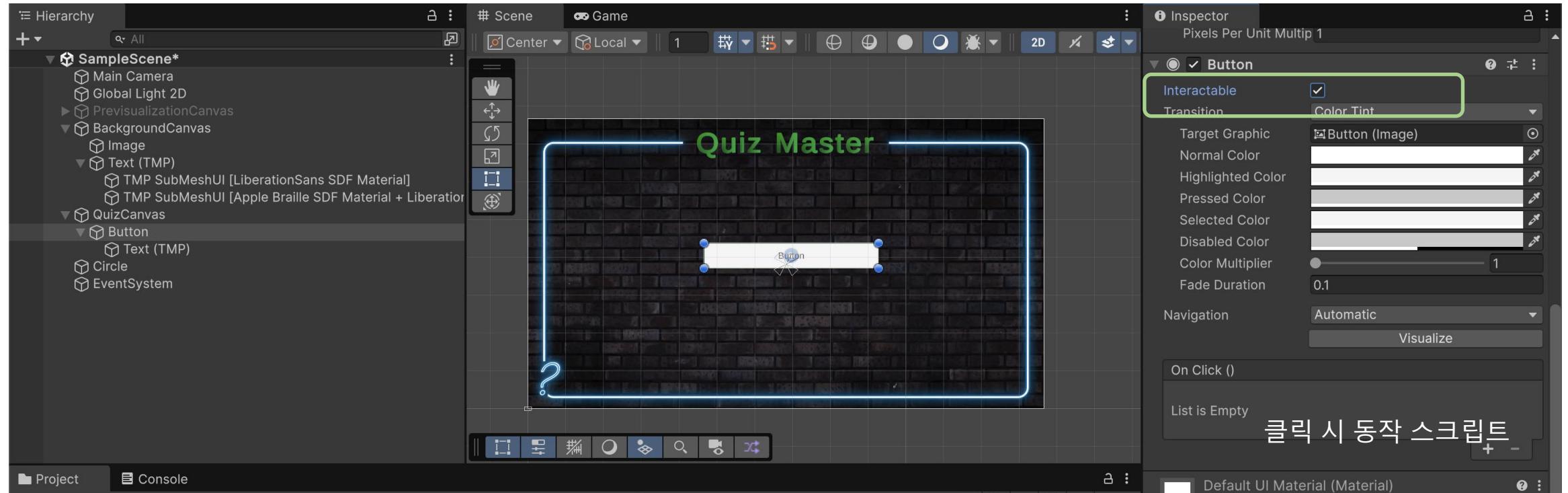


퀴즈 캔버스가 배경 캔버스 위에 보이게 하자

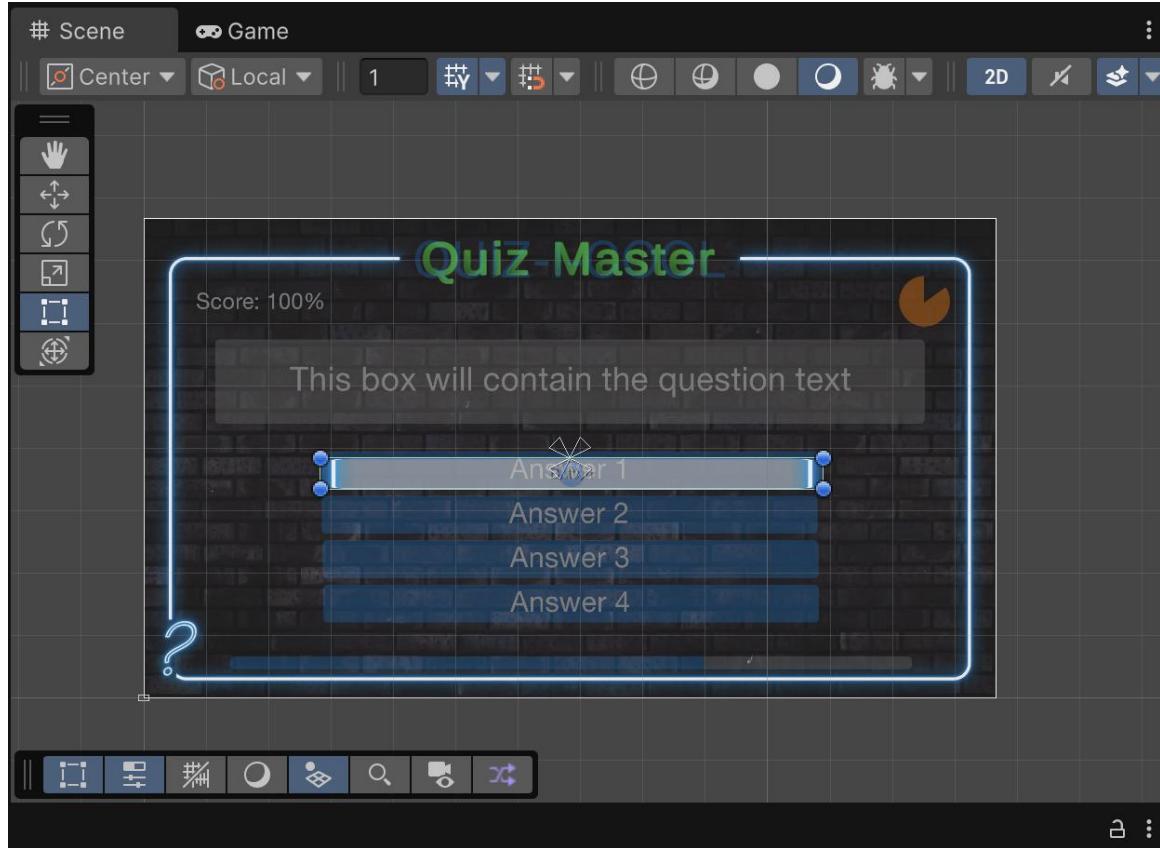
이것은 sort order를 통해 조정할 수 있다



버튼의 Interactivity



Button에 스프라이트 적용



테두리가 이상하게 나타남

좌우로 크기를 늘렸기 때문

버튼에 적용된 이미지를 그리는 방식 변경

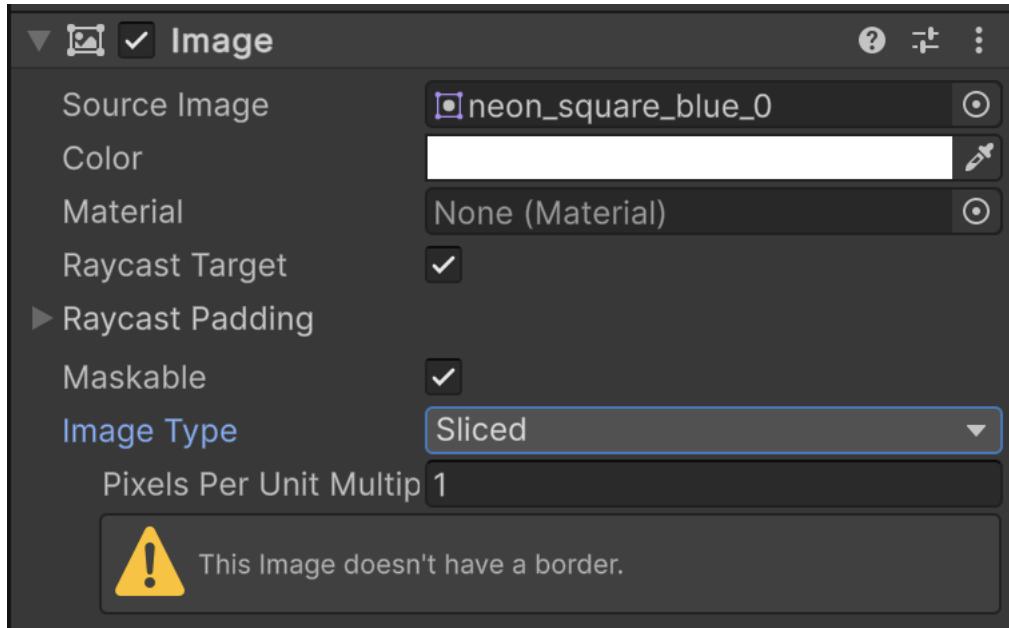
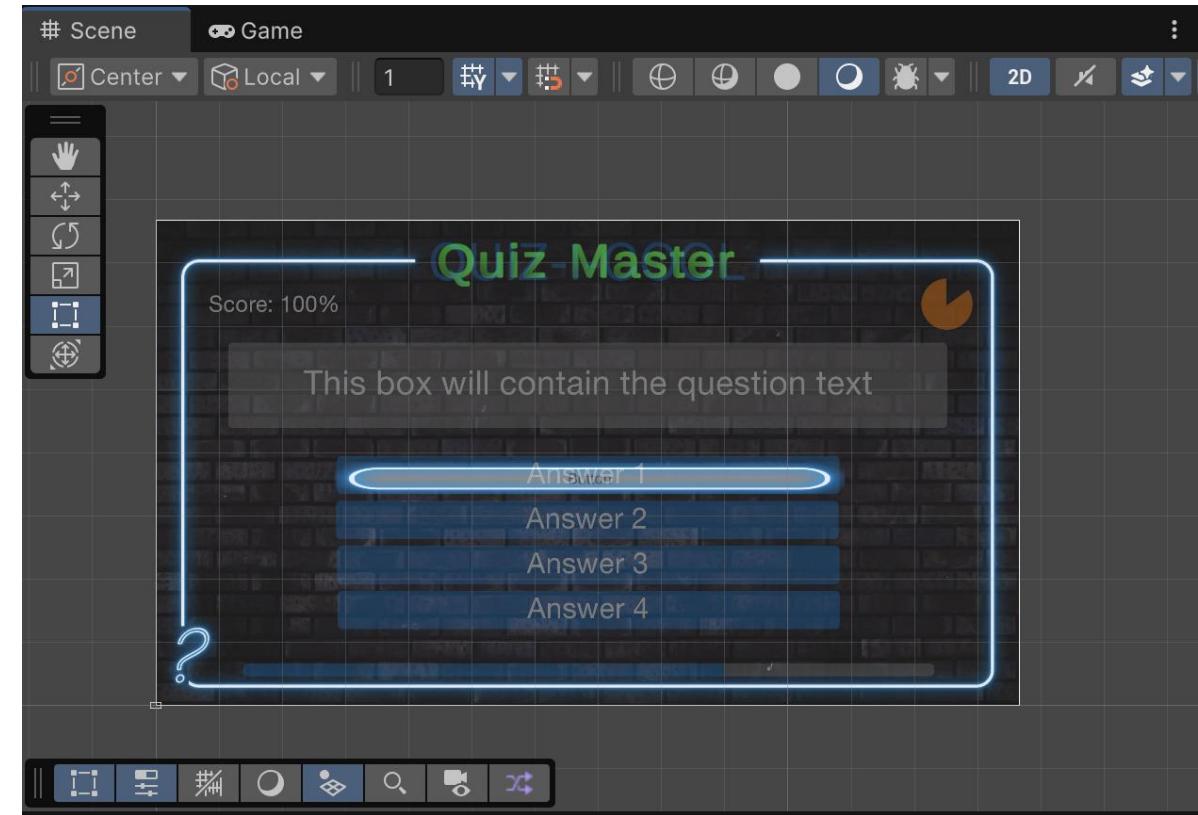
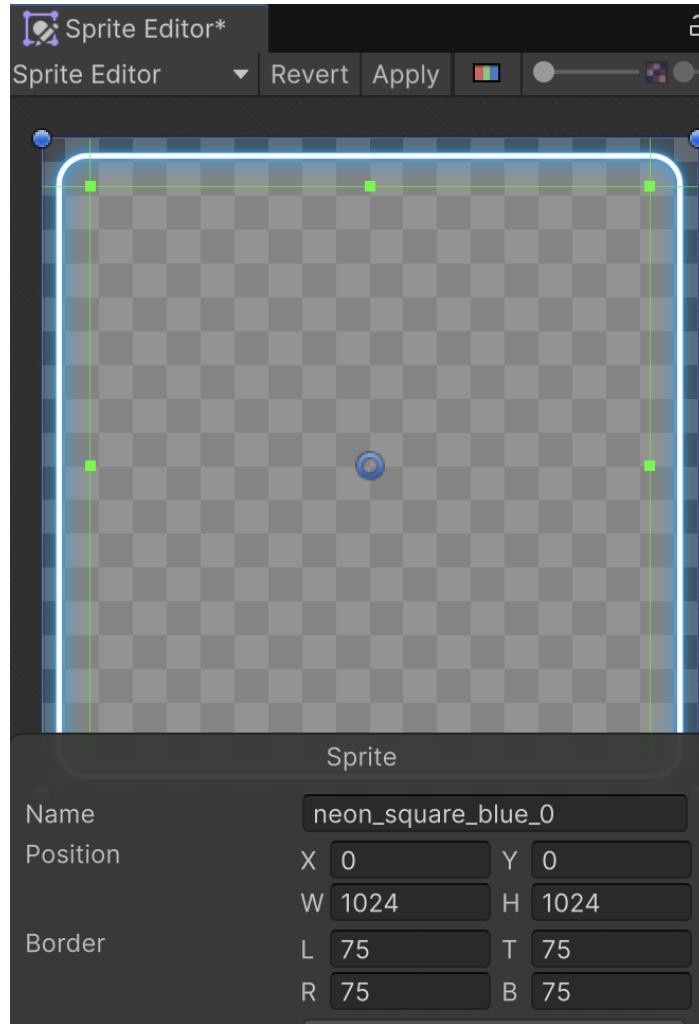


Image Type: Simple → Sliced

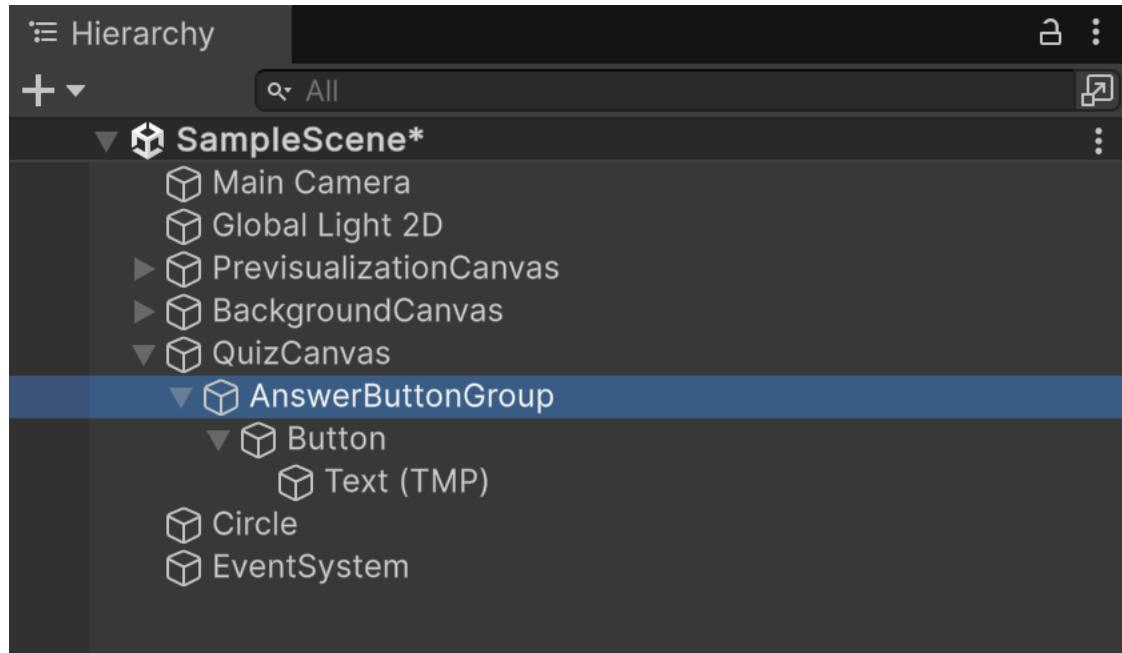
경고가 나타남 → 스프라이트 에디터로 이 스프라이트에 대해 경계 지정 필요

해당 스프라이트 선택하고 스프라이트 에디터 실행

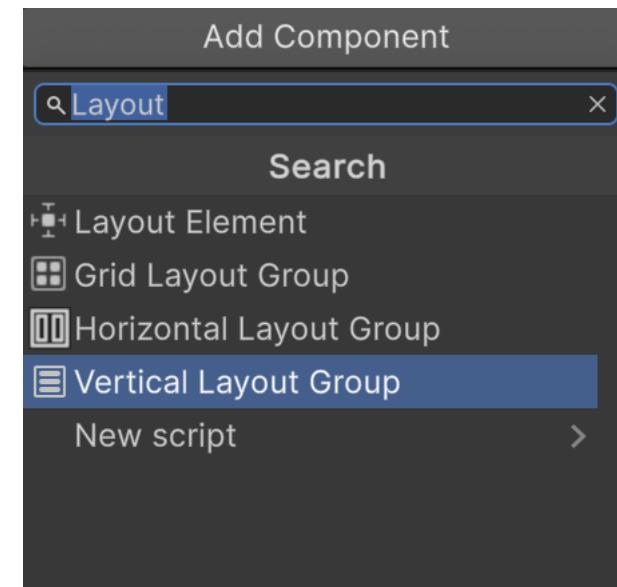


경계에 해당되는 픽셀 설정

여러 버튼을 담기 위해 빈 객체를 버튼 위에 생성

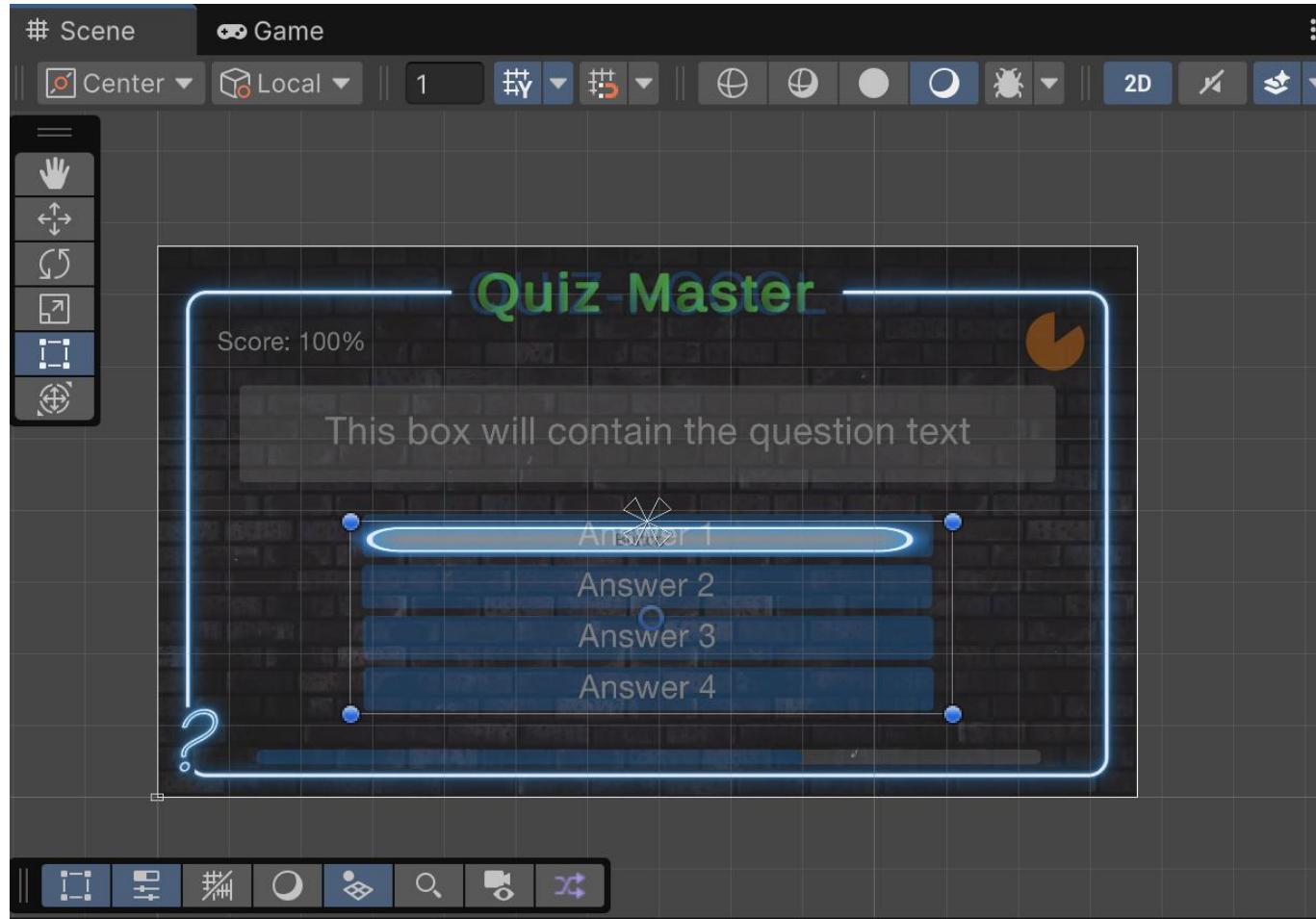


이름을 AnswerButtonGroup과
같이 설정



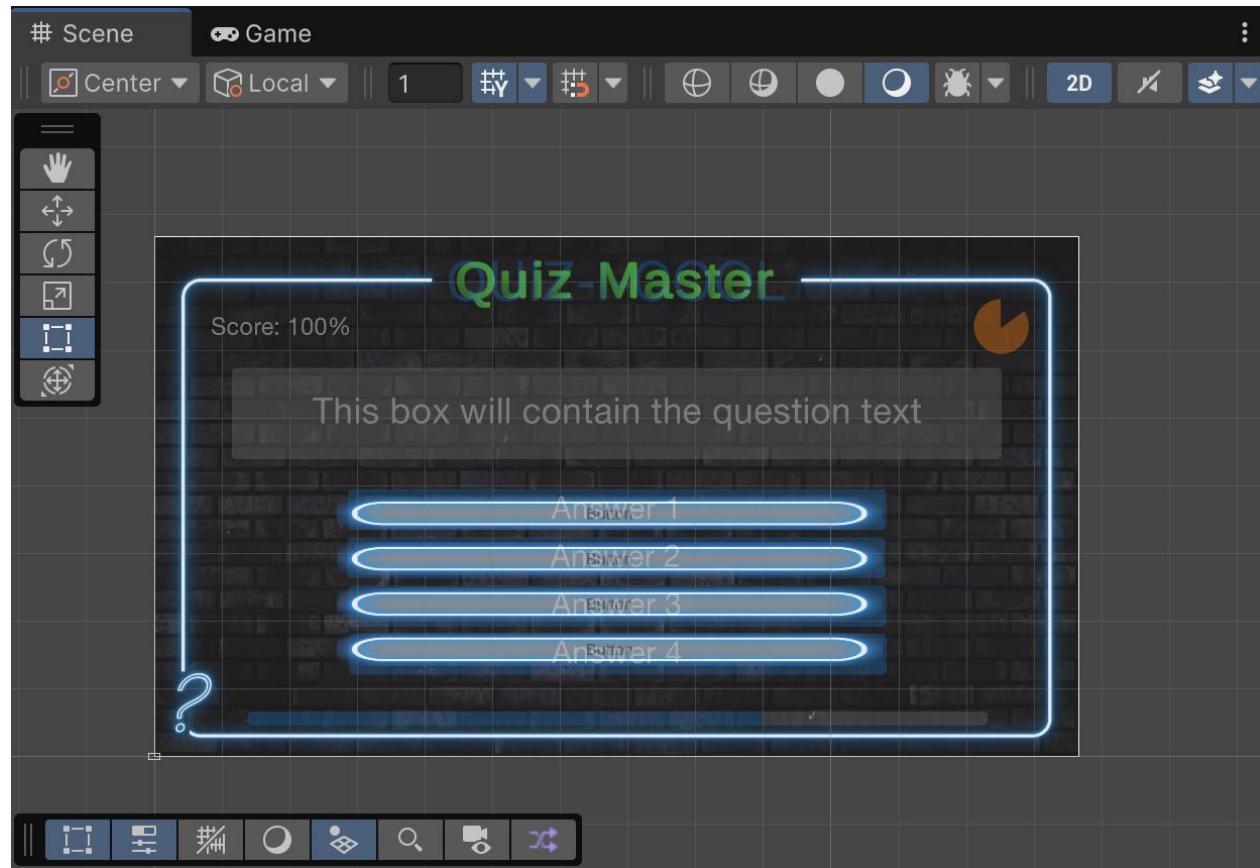
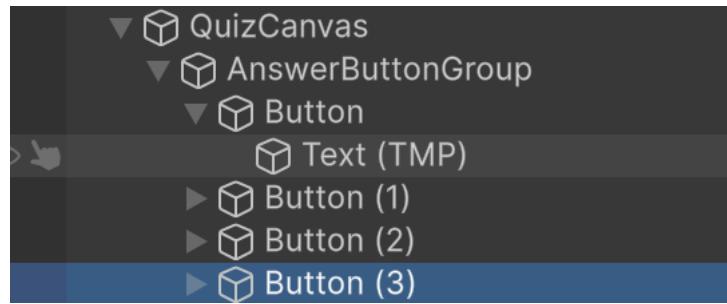
여기에 수직 레이아웃 그룹 컴포넌트 추가함
(버튼을 수직으로 배치할 수 있게 함)

버튼 그룹의 크기 조정



적당한 크기로 그룹 변형

버튼을 복사하여 4 개로 구성

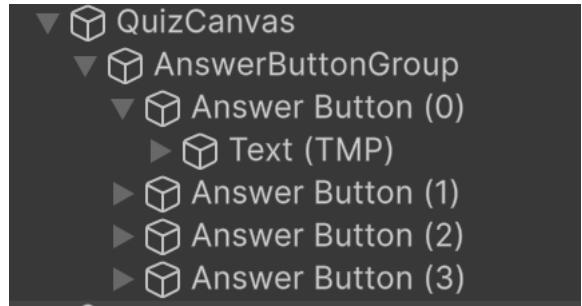


레이아웃에 따라 배치가 자동으로 됨

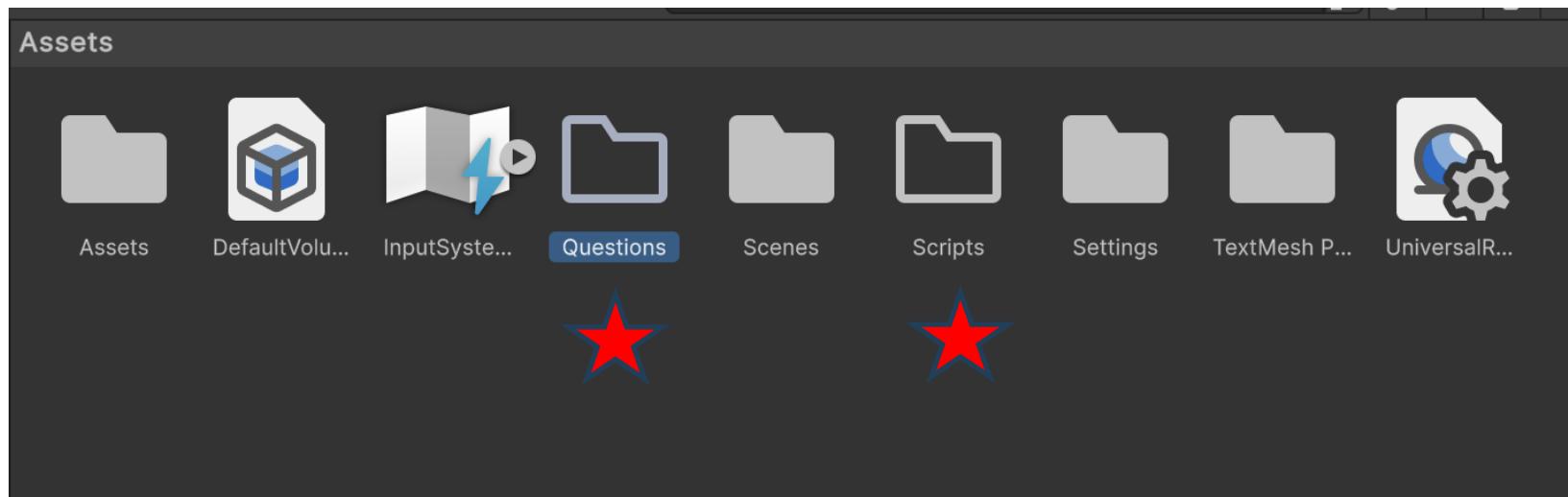
버튼 크기 조정



Button의 이름 정리



스크립터를 오브젝트를 위해 폴더 준비



스크립터블 객체(scriptable objects)

- 일종의 데이터 컨테이너
- 스크립트 밖에서 데이터를 유지
- 데이터를 한 곳에 저장함으로써 메모리 절약 가능
- 게임 객체에 연결할 필요 없음
- 경량이며 사용이 편리
- 일관성을 유지하기 위한 템플릿으로 기능
- 사용가능 예
 - RPG 게임에서의 무기 스탯
 - 카드 게임의 카드 정보 등

* 여기서는 질문 데이터

사용 방법의 개념

코드: quiz.cs

GetQuestionData()

[

]

DisplayQuestion()
CheckAnswer()

스크립터블 객체

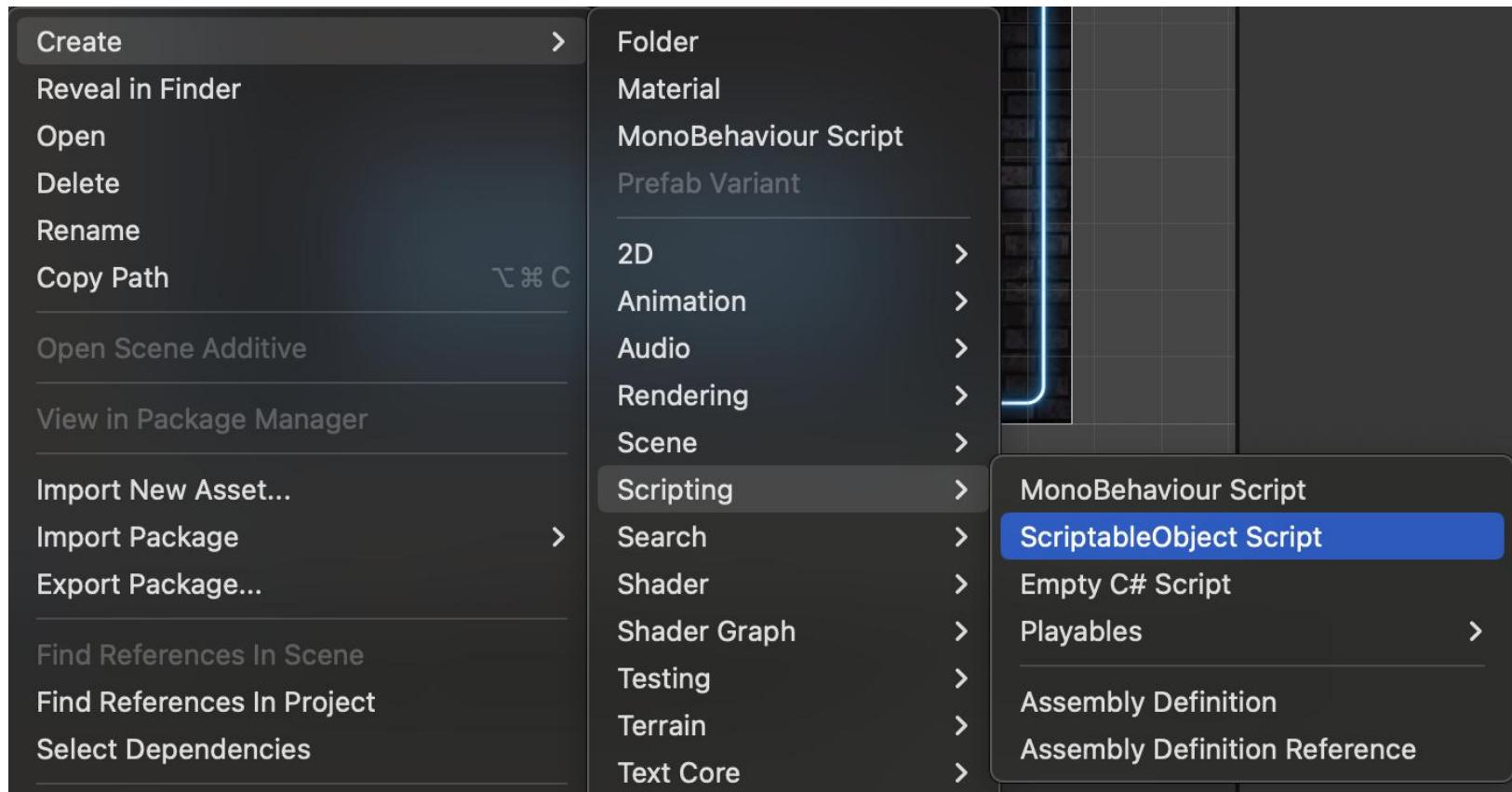
Question 1

Question 2

Question 3

Scriptable Object 코드 작성

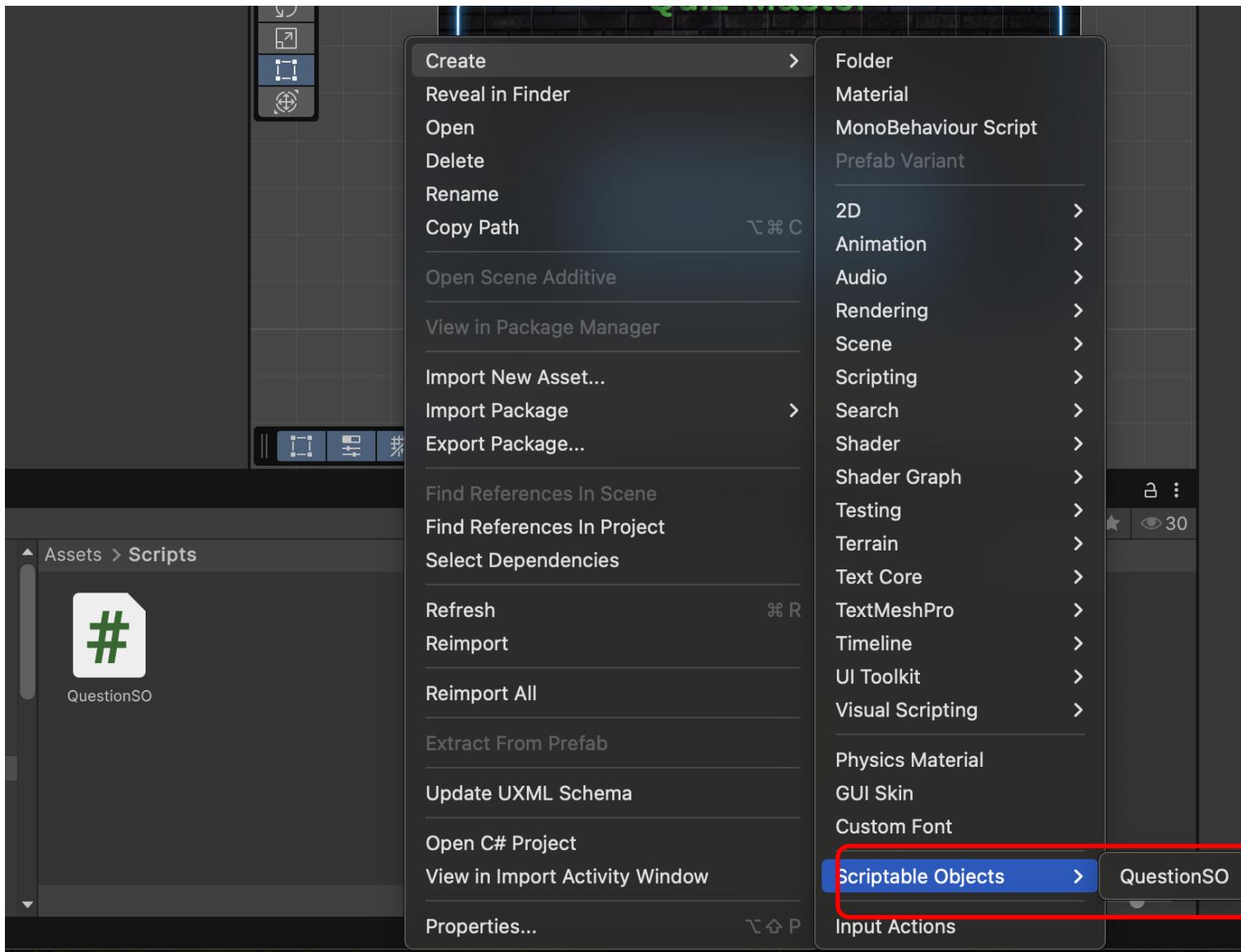
- Scripts 폴더 내에 스크립트 생성. (ScriptableObject Script로 생성)
 - QuestionSO.cs



기본 코드의 모양이 Monobehaviour와 다름

```
1  using UnityEngine;
2
3  [CreateAssetMenu(fileName = "QuestionSO", menuName = "Scriptable Objects/QuestionSO")]
4  public class QuestionSO : ScriptableObject
5  {
6
7  }
8 |
```

스크립터블 객체를 생성할 수 있게 됨



현재로서는 특별한 정보가 담기지
않음

QuestionSO.cs 수정

The screenshot shows the Unity Editor interface. On the left, a code editor window displays the C# script `QuestionSO.cs`. The script contains the following code:

```
1  using UnityEngine;
2
3  [CreateAssetMenu(fileName = "QuestionSO", menuName = "Scriptable Objects/QuestionSO")]
4  public class QuestionSO : ScriptableObject
5  {
6      [TextArea(2, 6)]
7      string question = "Enter New Question Text Here";
8  }
9
10 }
```

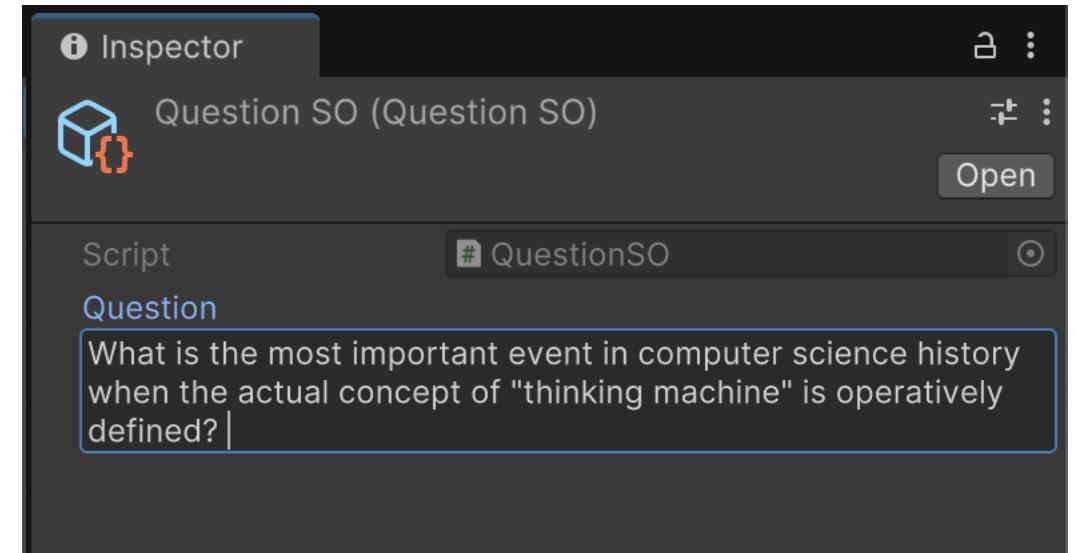
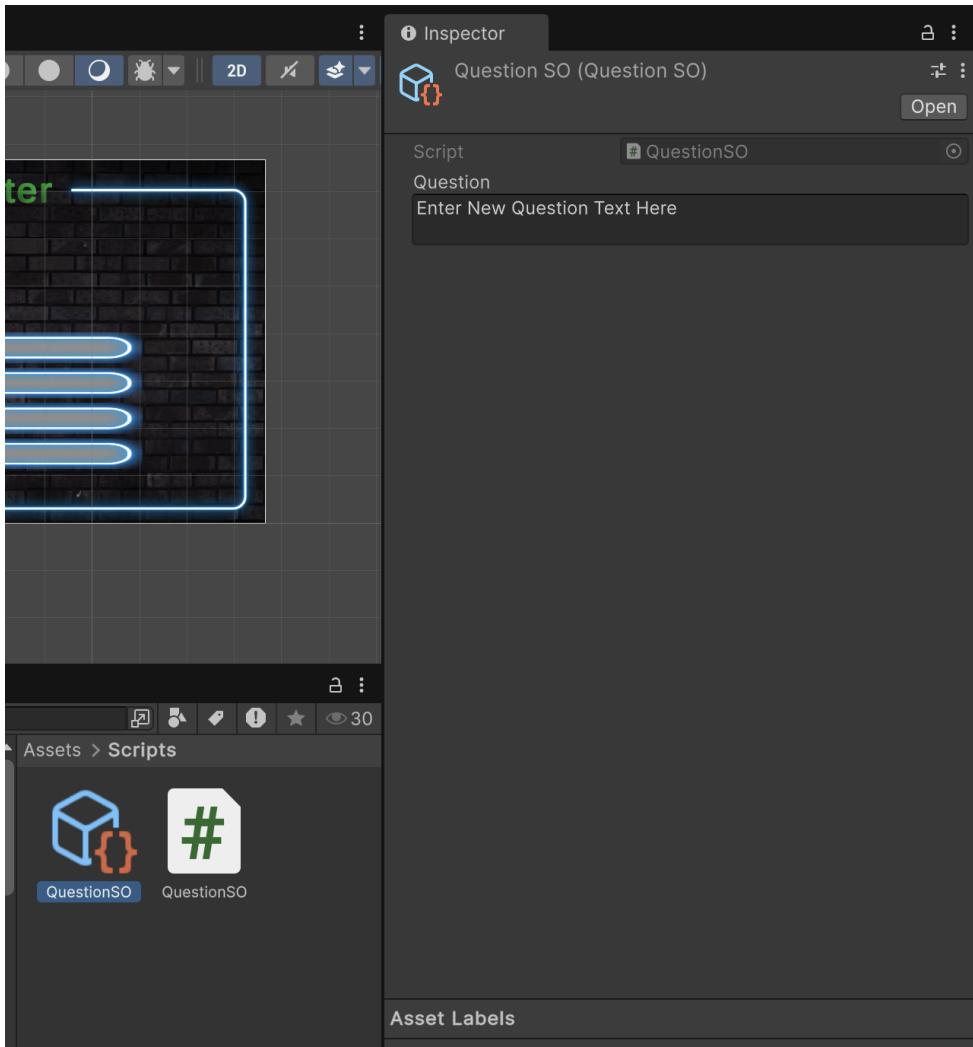
The script uses the `ScriptableObject` base class and includes a `TextArea` attribute for the `question` field. The `menuName` attribute in the `CreateAssetMenu` attribute specifies the name of the asset type in the Unity menu.

To the right of the code editor, a context menu is open over the script file. The menu is organized into several sections:

- Create**:
 - Folder
 - Material
 - MonoBehaviour Script
 - Prefab Variant
- 2D**:
 - Animation
 - Audio
 - Rendering
- Scene**:
 - Import New Asset...
 - Import Package
 - Export Package...
 - Find References In Scene
 - Find References In Project
 - Select Dependencies
- Shader**:
 - Shader Graph
 - Testing
 - Terrain
 - Text Core
 - TextMeshPro
 - Timeline
 - UI Toolkit
 - Visual Scripting
- Physics Material**:
 - GUI Skin
 - Custom Font
- Scriptable Objects**:
 - QuestionSO

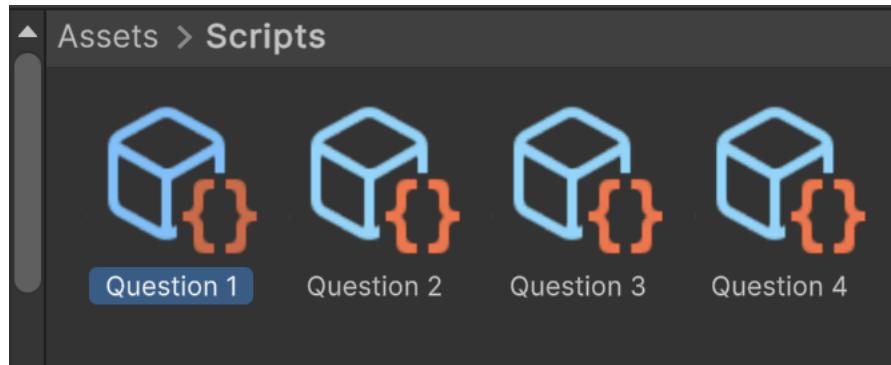
The `QuestionSO` item in the `Scriptable Objects` section is highlighted with a blue background, indicating it is the selected option.

Scriptable Object 생성



여섯 줄까지 늘어나는 TextArea 입력 공간에
question 데이터 채울 수 있음

여러 개의 질문 데이터를 생성



게터(getter)

- 읽기 전용인 변수에 대한 “읽기” 접근을 제공
- Private 데이터의 내용을 보호

```
using UnityEngine;

[CreateAssetMenu(fileName = "QuestionSO", menuName = "Scriptable Objects/QuestionSO")]
0 references
public class QuestionSO : ScriptableObject
{
    [TextArea(2, 6)]
    1 reference
    [SerializeField] string question = "Enter New Question Text Here";

    0 references
    public string GetQuestion()
    {
        return question;
    }
}
```

배열을 이용한 정답 저장

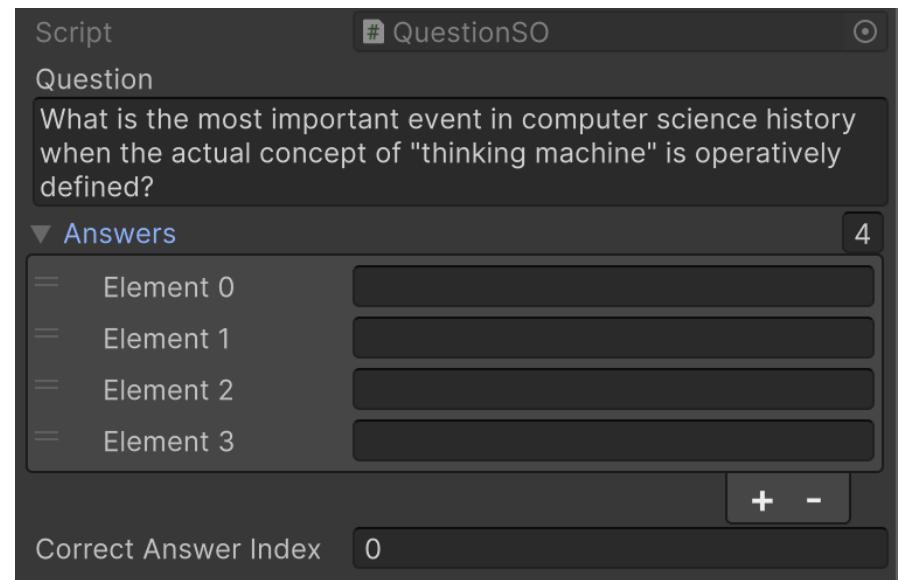
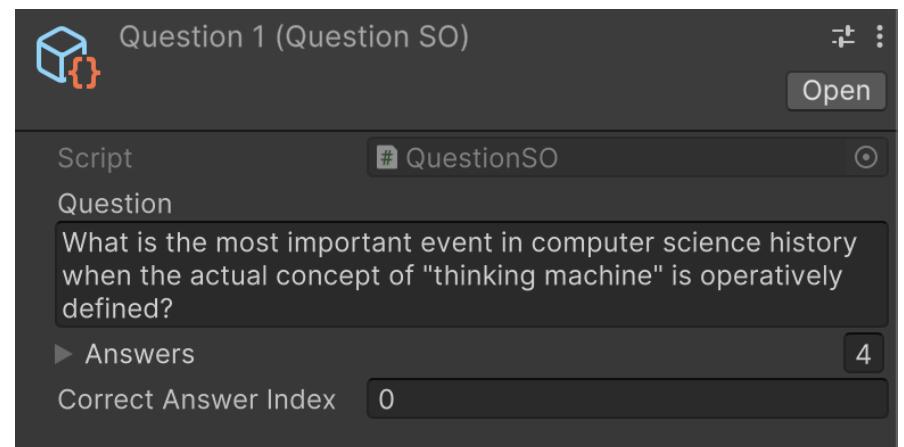
```
public class QuestionSO : ScriptableObject
{
    [TextArea(2, 6)]
    1 reference
    [SerializeField] string question = "Enter New Question Text Here";

    1 reference
    [SerializeField] string[] answers = new string[4];
    1 reference
    [SerializeField] int correctAnswerIndex;

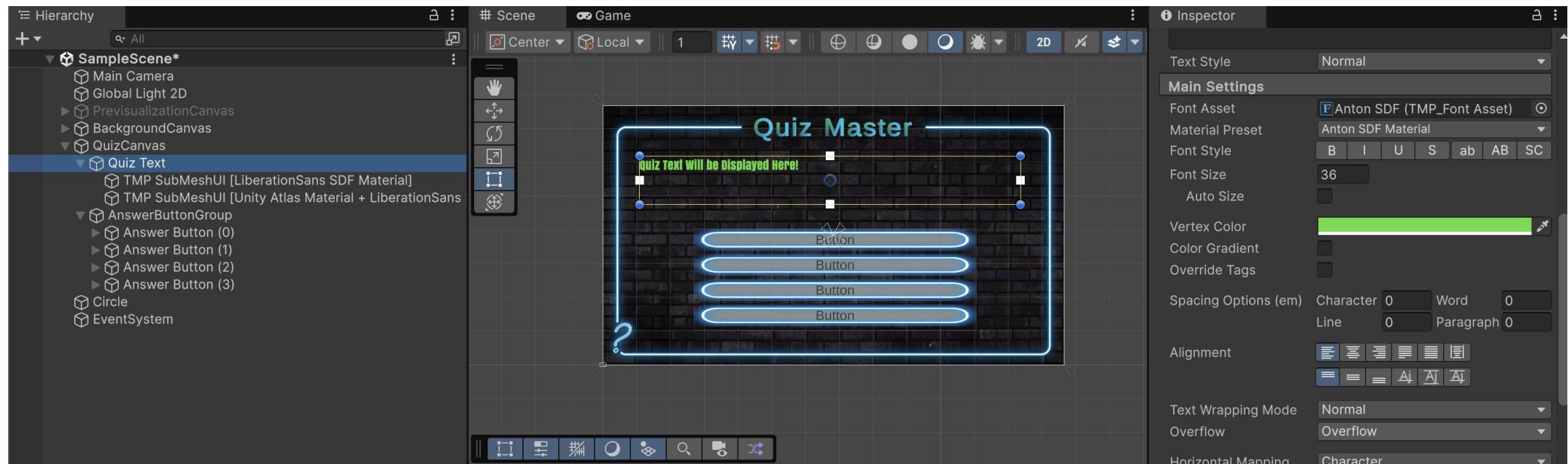
    0 references
    public string GetQuestion()
    {
        return question;
    }

    0 references
    public int GetCorrectAnswerIndex()
    {
        return correctAnswerIndex;
    }

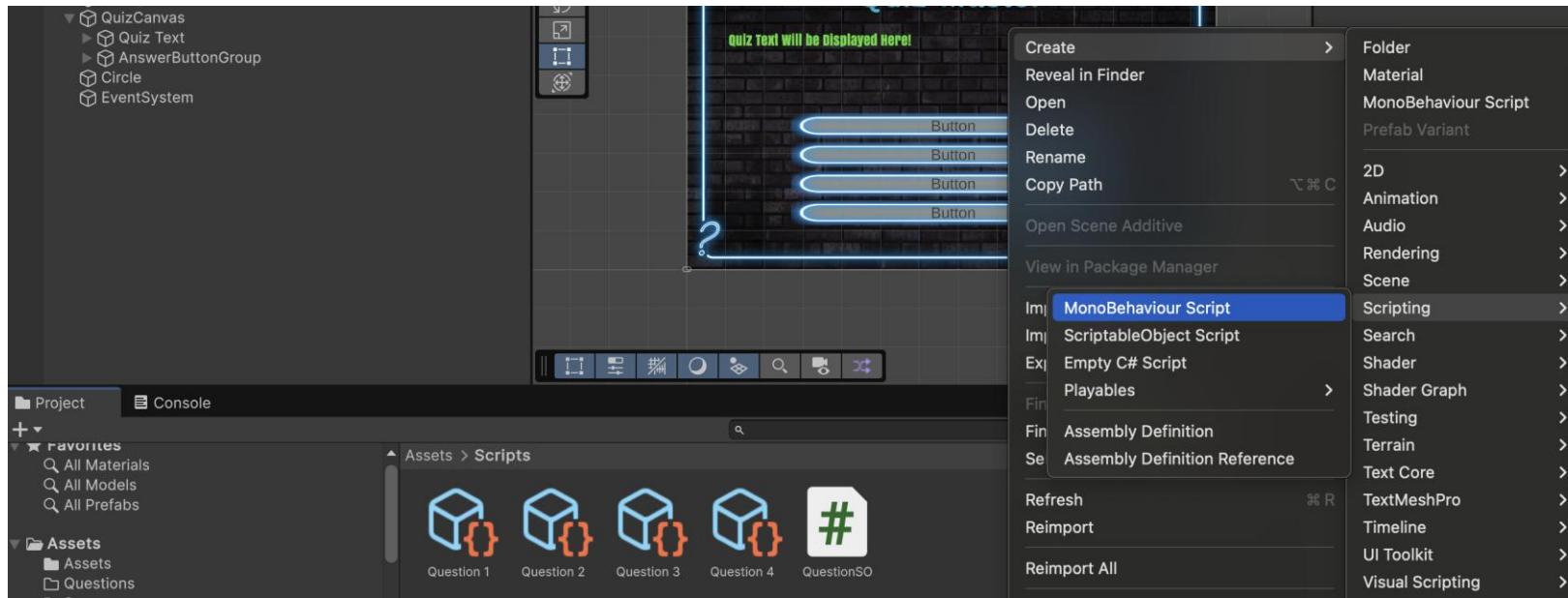
    0 references
    public string GetAnswer(int idx) {
        return answers[idx];
    }
}
```



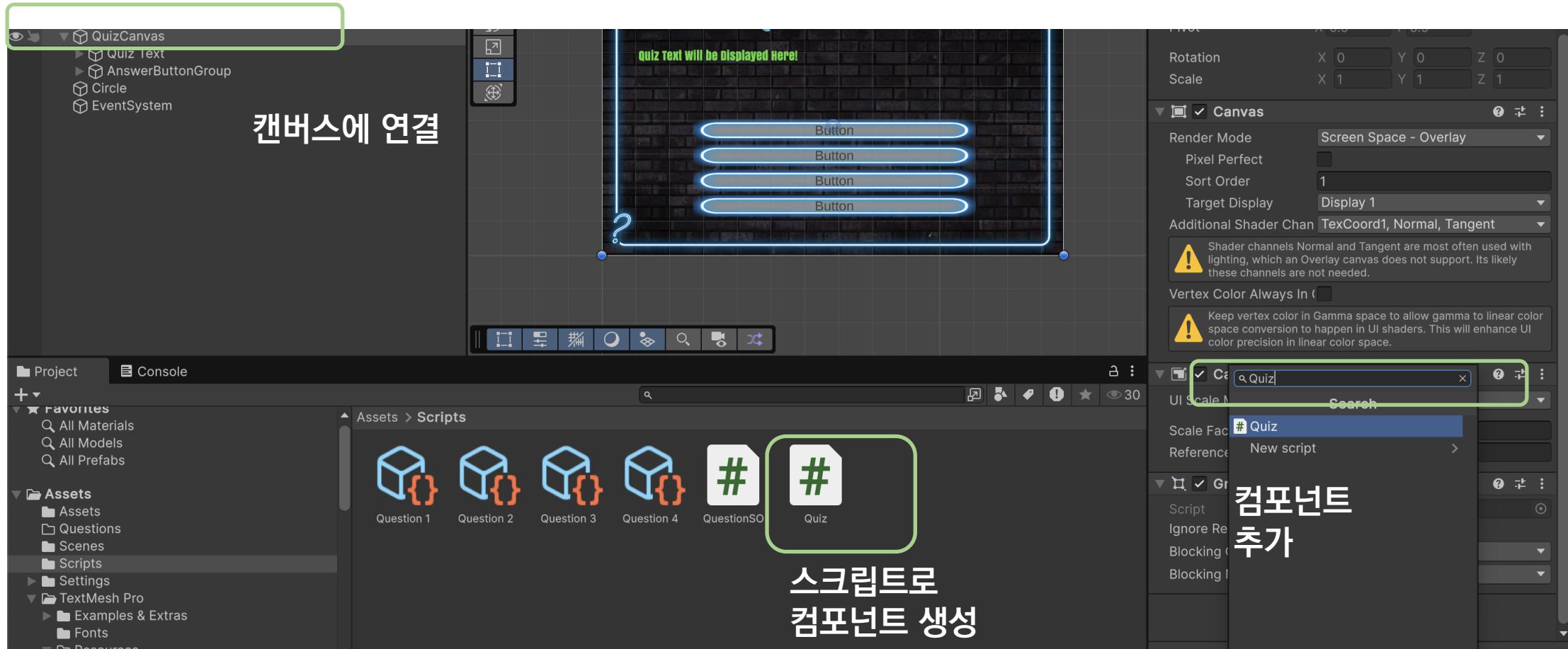
Quiz 텍스트가 나타날 공간에 TextMeshPro 배치



Quiz 캔버스에 스크립트를 설치하여 문제 다루기

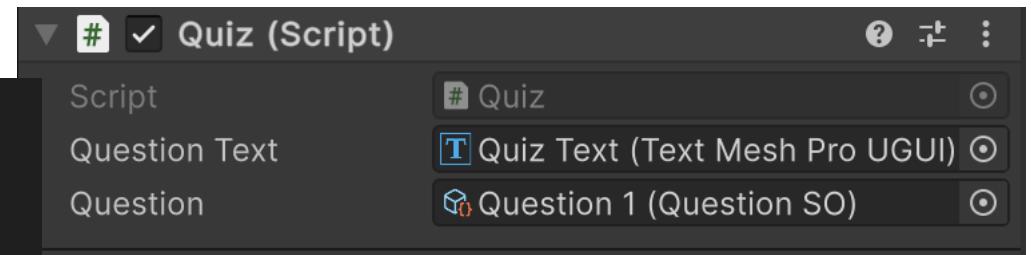


Quiz 캔버스에 스크립트를 설치하여 문제 다루기

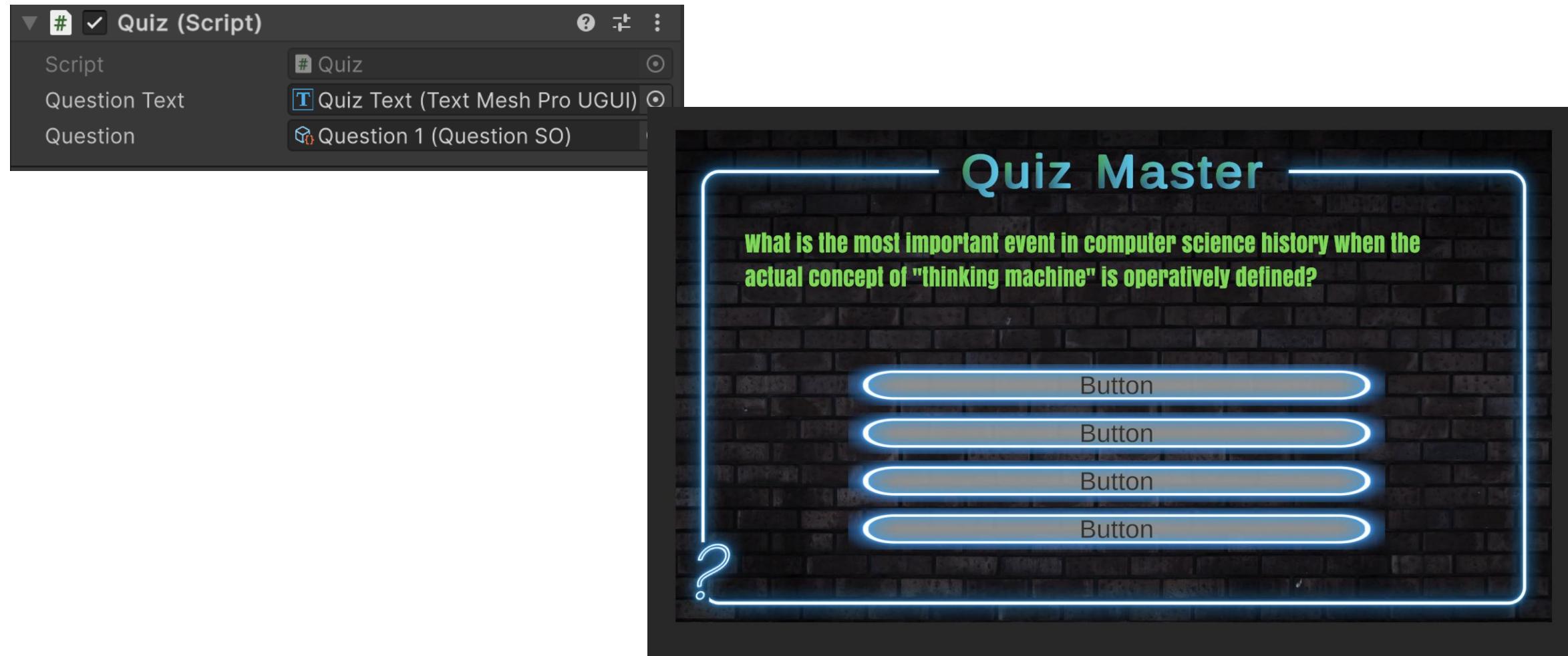


Quiz 스크립트 설정

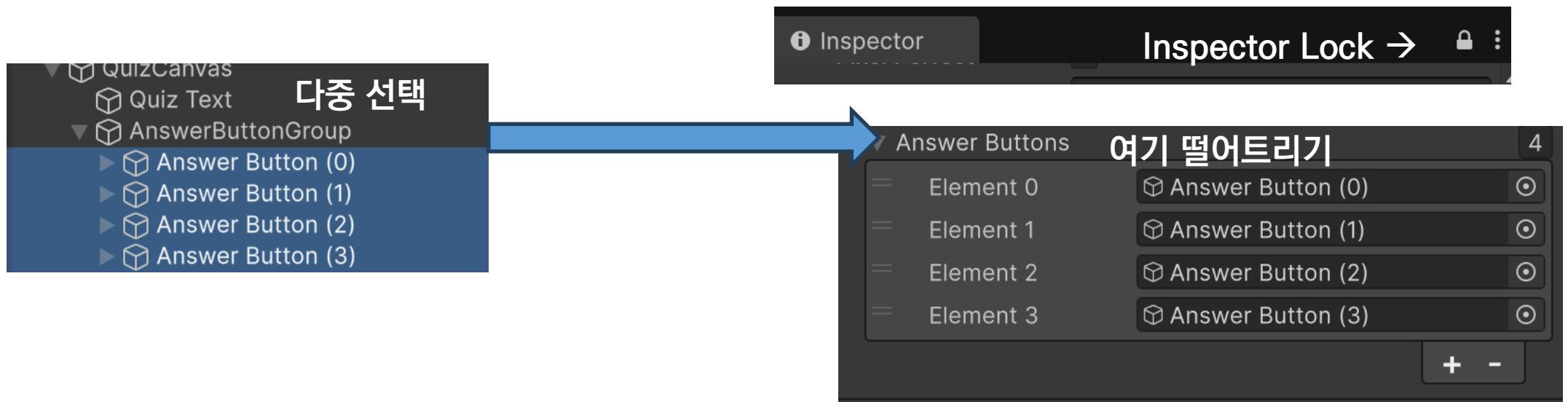
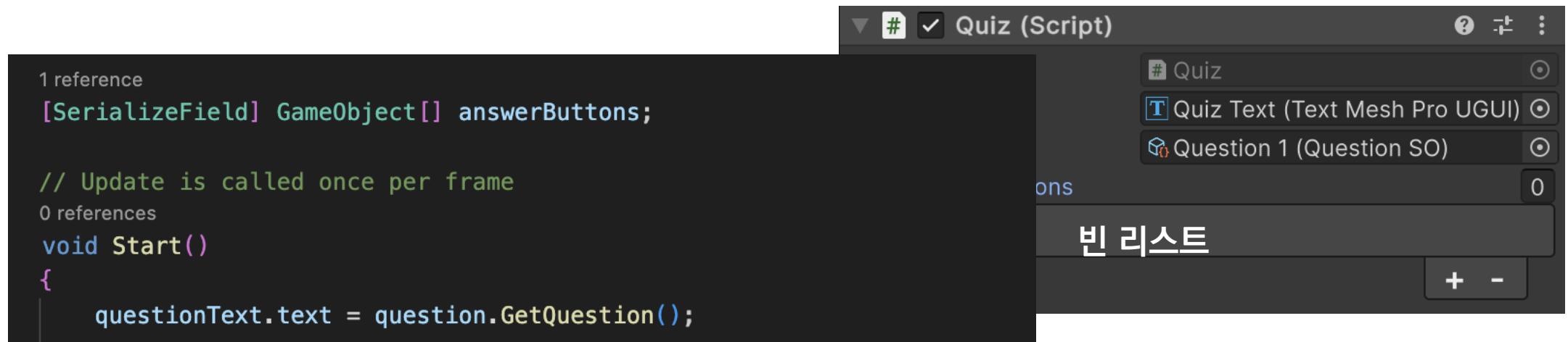
```
1  using UnityEngine;
2  using TMPro;
3  0 references
4  public class Quiz : MonoBehaviour
5  {
6      1 reference
7      [SerializeField] TextMeshProUGUI questionText;
8      1 reference
9      [SerializeField] QuestionSO question;
10
11     // Update is called once per frame
12     0 references
13     void Start()
14     {
15         questionText.text = question.GetQuestion();
16     }
17 }
```



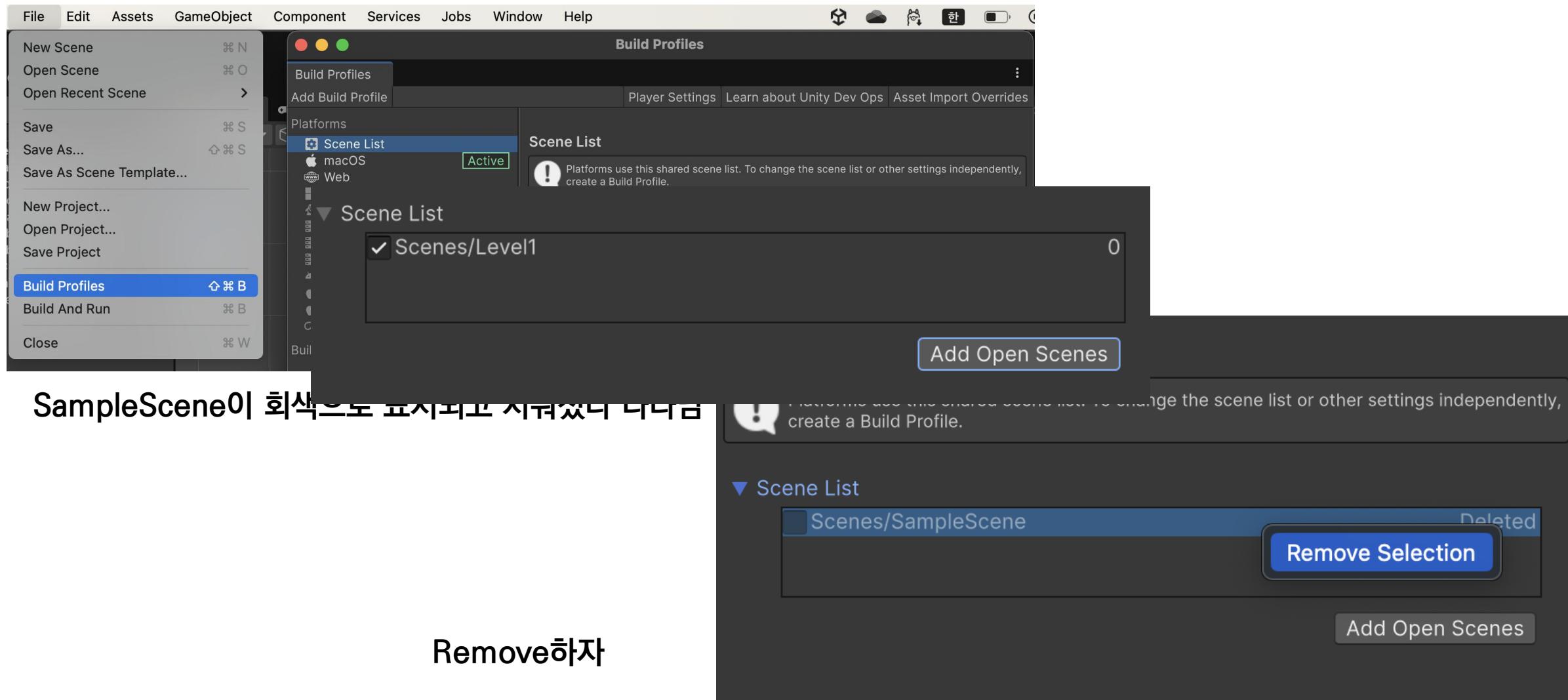
Canvas의 Inspector에서 설정



답변 가져오기



Build Profile 다시 확인



답변 버튼에 저장된 답변 출력하기 (1개만)

```
public class Quiz : MonoBehaviour
{
    1 reference
    [SerializeField] TextMeshProUGUI questionText;
    2 references
    [SerializeField] QuestionSO question;

    1 reference
    [SerializeField] GameObject[] answerButtons;

    // Update is called once per frame
    0 references
    void Start()
    {
        questionText.text = question.GetQuestion();

        TextMeshProUGUI buttonText =
            answerButtons[0].GetComponentInChildren<TextMeshProUGUI>();

        buttonText.text = question.GetAnswer(0);
    }
}
```

- Quiz Master -

Important event in computer science history when the "thinking machine" is operatively defined?

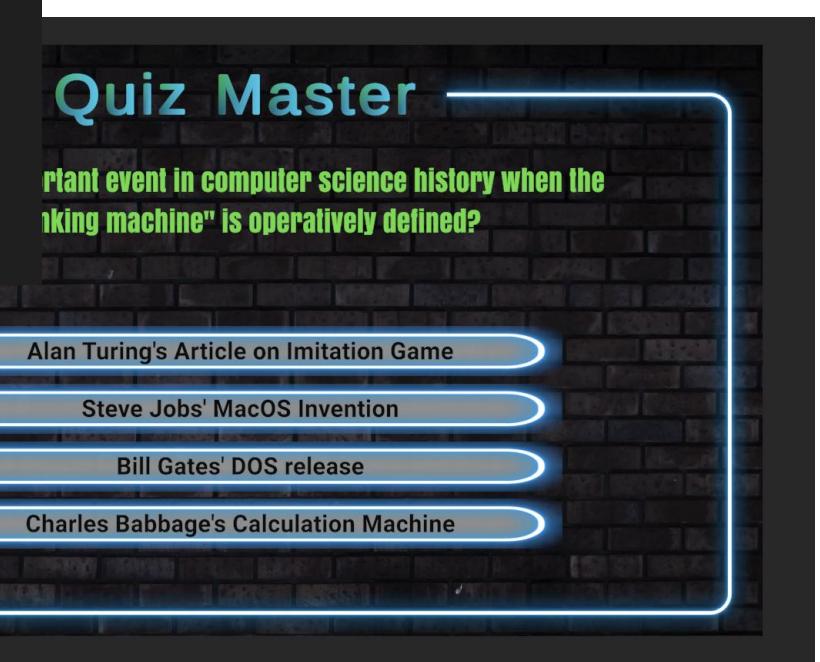
Turing's Article on Imitation Game
Button
Button
Button

For 문을 이용한 전체 답변 표시

```
void Start()
{
    questionText.text = question.GetQuestion();

    for(int i = 0; i < answerButtons.Length; i++)
    {
        TextMeshProUGUI buttonText =
        answerButtons[i].GetComponentInChildren<TextMeshProUGUI>();

        buttonText.text = question.GetAnswer(i);
    }
}
```



다음으로 해야 할 일은?

답을 하기 위해 버튼을 누르면 정답이 맞는지 확인하기

하나의 문제에 답을 하면 다음 문제로 넘어가기

Quiz Script를 수정하자

```
using UnityEngine.UI;

0 references
public class Quiz : MonoBehaviour
{
    3 references
    [SerializeField] TextMeshProUGUI questionText;
    3 references
    [SerializeField] QuestionSO question;

    3 references
    [SerializeField] GameObject[] answerButtons;

    0 references
    int correctAnswerIndex;
    0 references
    [SerializeField] Sprite defaultAnswerSprite;
    1 reference
    [SerializeField] Sprite correctAnswerSprite;
```

```
void Start()
{
    questionText.text = question.GetQuestion();

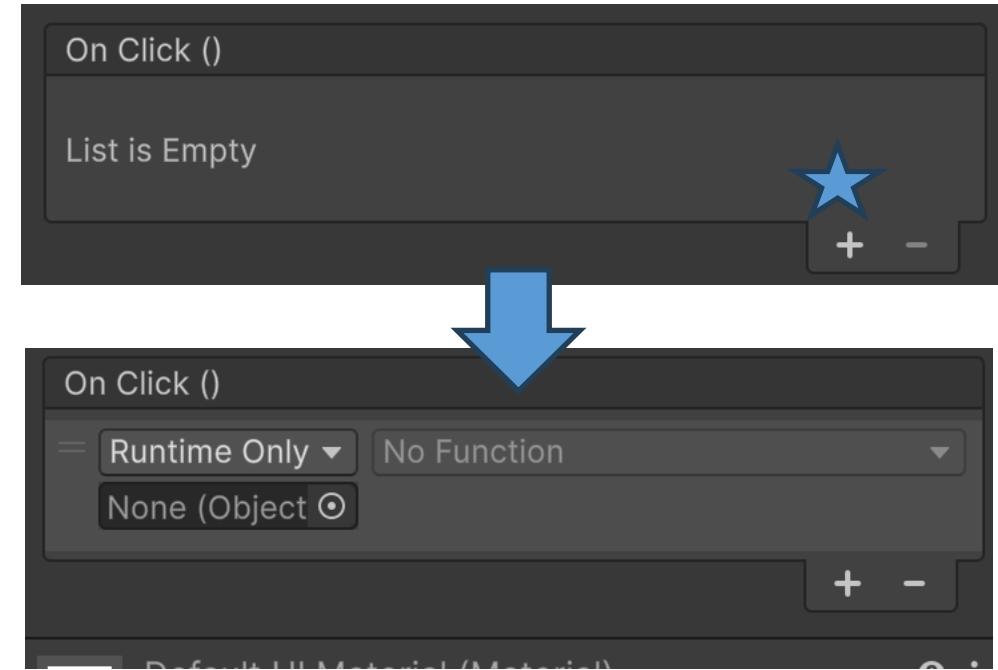
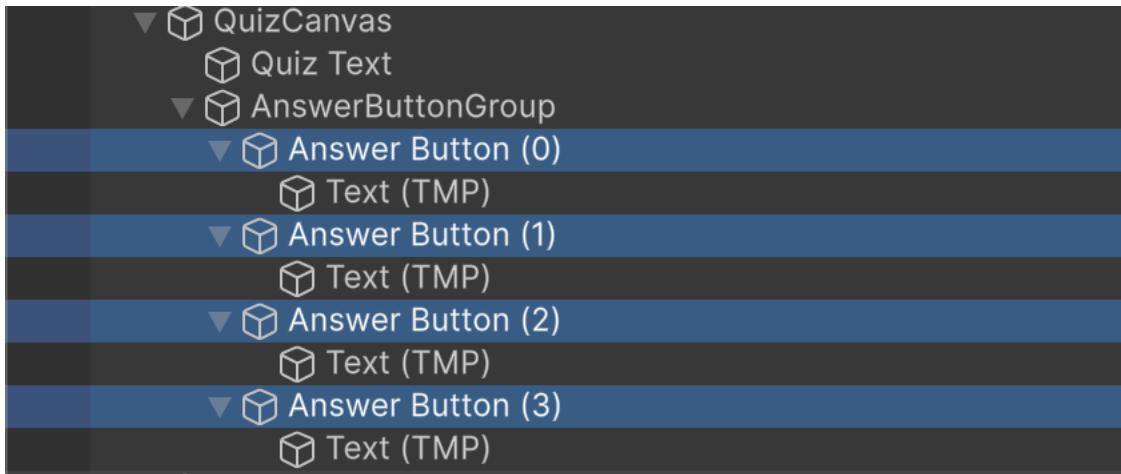
    for(int i = 0; i < answerButtons.Length; i++)
    {
        TextMeshProUGUI buttonText =
        answerButtons[i].GetComponentInChildren<TextMeshProUGUI>();

        buttonText.text = question.GetAnswer(i);
    }
}

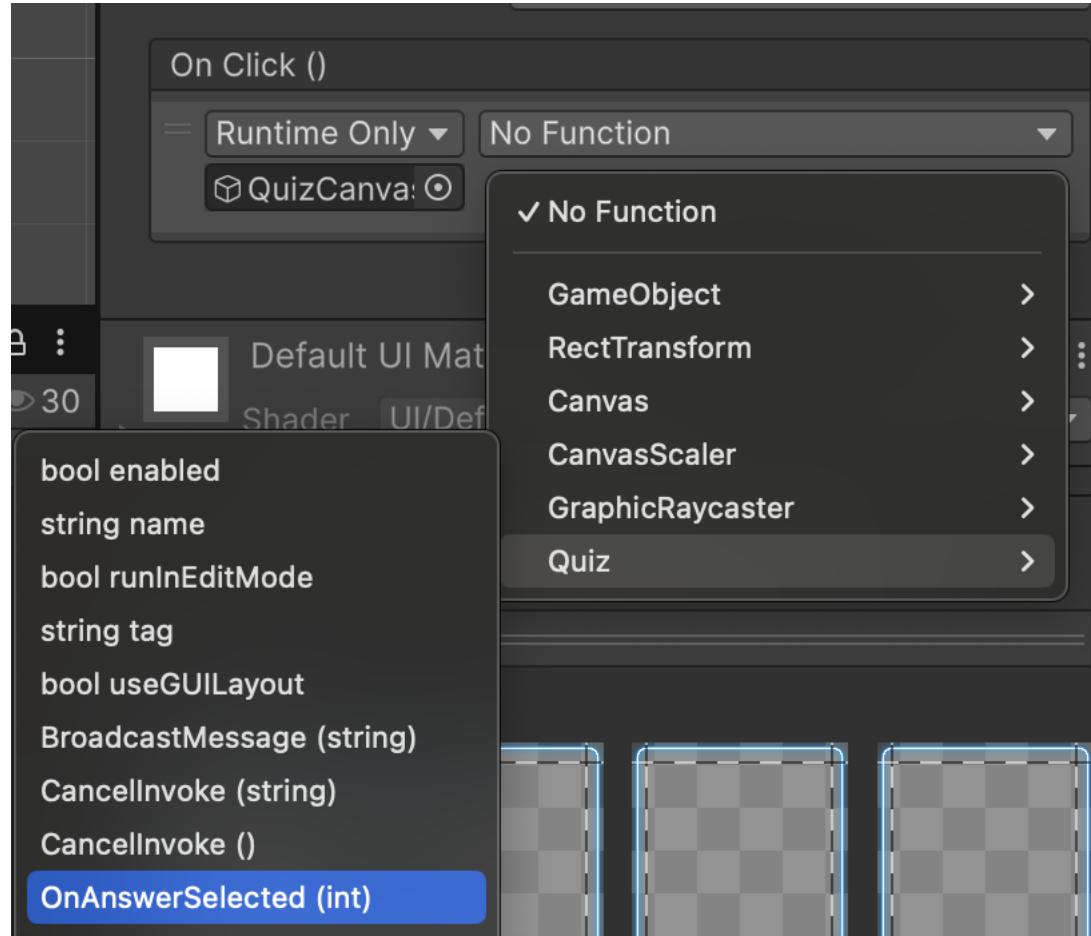
0 references
public void OnAnswerSelected(int index)
{
    if(index == question.GetCorrectAnswerIndex())
    {
        questionText.text = "Correct!";
        Image buttonImage = answerButtons[index].GetComponent<Image>();
        buttonImage.sprite = correctAnswerSprite;
    }
    else {
        questionText.text = "Guess Again!";
    }
}
```

이벤트 처리기 설정

버튼들을 다중 선택 (윈도: ctrl+선택. / MacOS: cmd+선택)

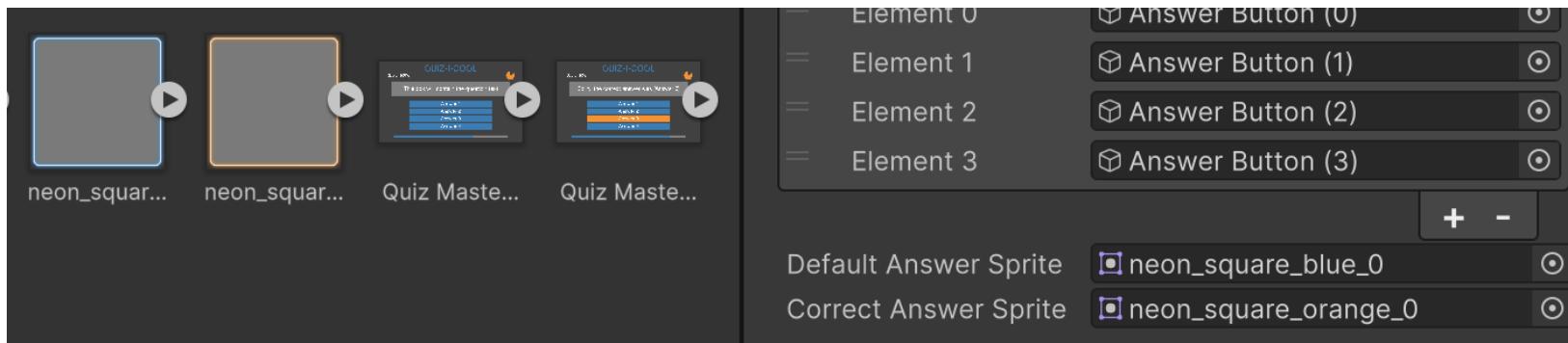


이벤트를 처리할 함수를 지정하고 각 버튼별로 인자 설정



스프라이트 바꾸기

Canvas에 두 개의 스프라이트 지정

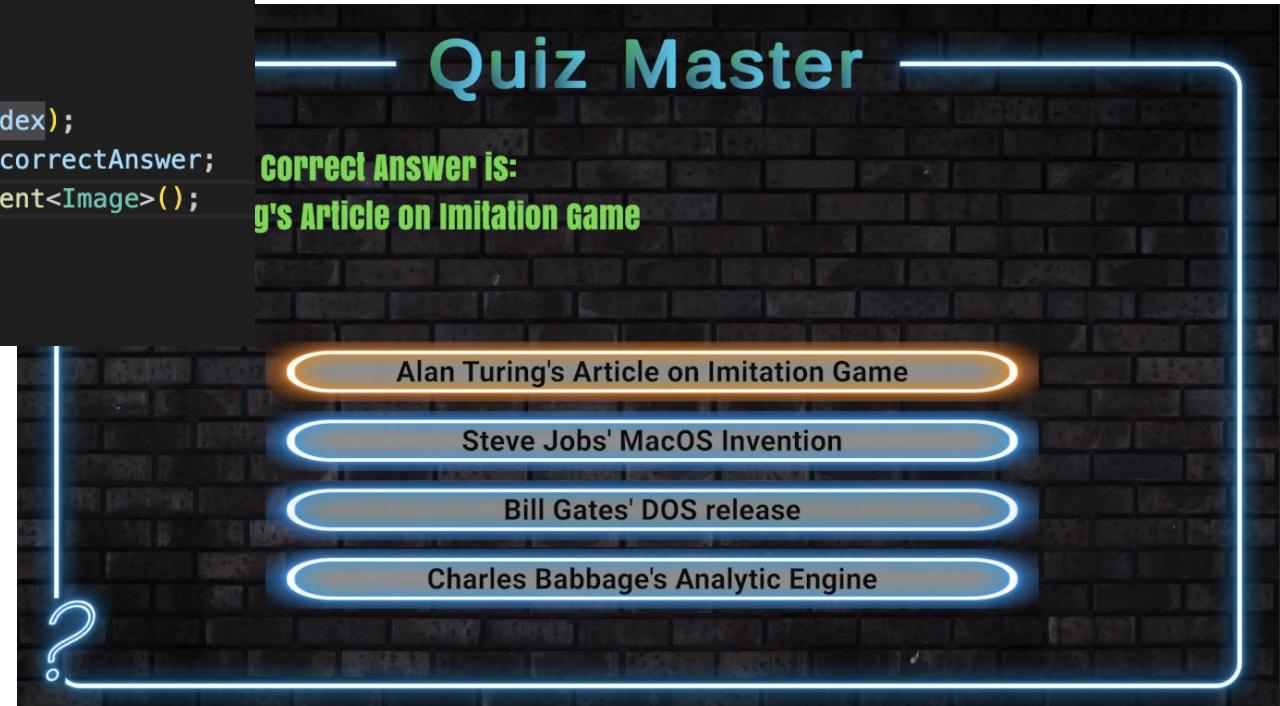


동작 개선

```
public void OnAnswerSelected(int index)
{
    correctAnswerIndex = question.GetCorrectAnswerIndex();
    Image buttonImage;

    if(index == correctAnswerIndex)
    {
        questionText.text = "Correct!";
        buttonImage = answerButtons[index].GetComponent<Image>();

    }
    else {
        string correctAnswer = question.GetAnswer(correctAnswerIndex);
        questionText.text = "Sorry, the Correct Answer is: \n" + correctAnswer;
        buttonImage = answerButtons[correctAnswerIndex].GetComponent<Image>();
    }
    buttonImage.sprite = correctAnswerSprite;
}
```



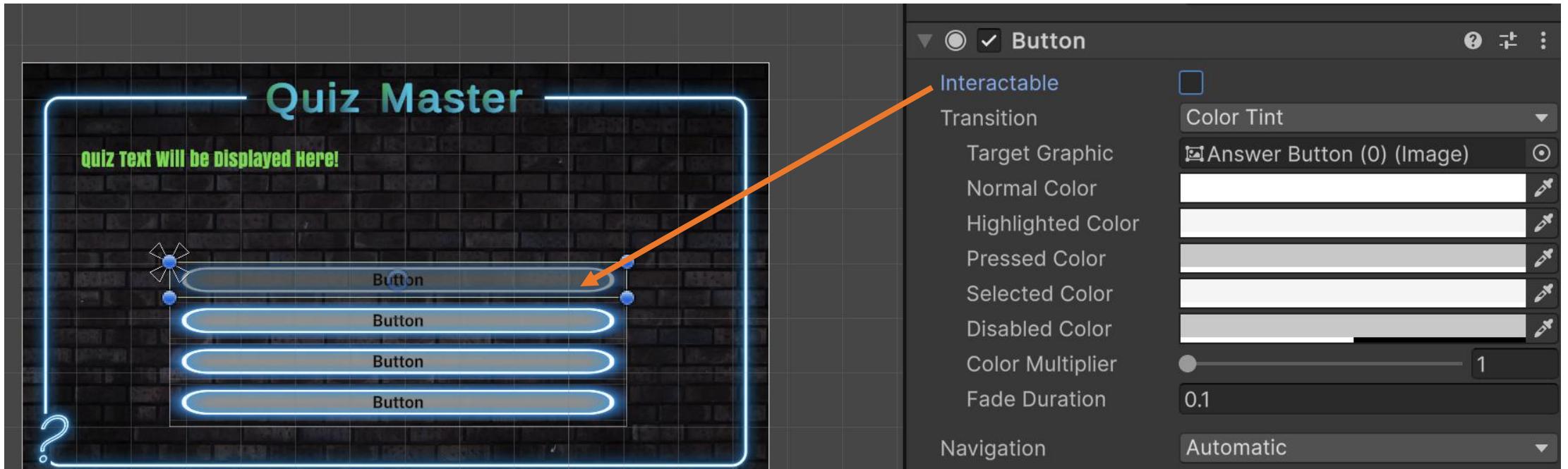
부정행위 방지

버튼을 한 번 선택하면 바꿀 수 없게 해야 함
전체적인 게임의 흐름



버튼의 활성화/비활성화

Interactable option



코드 수정 – 질문 표시를 별도의 메소드로

```
void Start()
{
    DisplayQuestion();
}

1 reference
void DisplayQuestion()
{
    questionText.text = question.GetQuestion();

    for(int i = 0; i < answerButtons.Length; i++)
    {
        TextMeshProUGUI buttonText =
        answerButtons[i].GetComponentInChildren<TextMeshProUGUI>();

        buttonText.text = question.GetAnswer(i);
    }
}
```

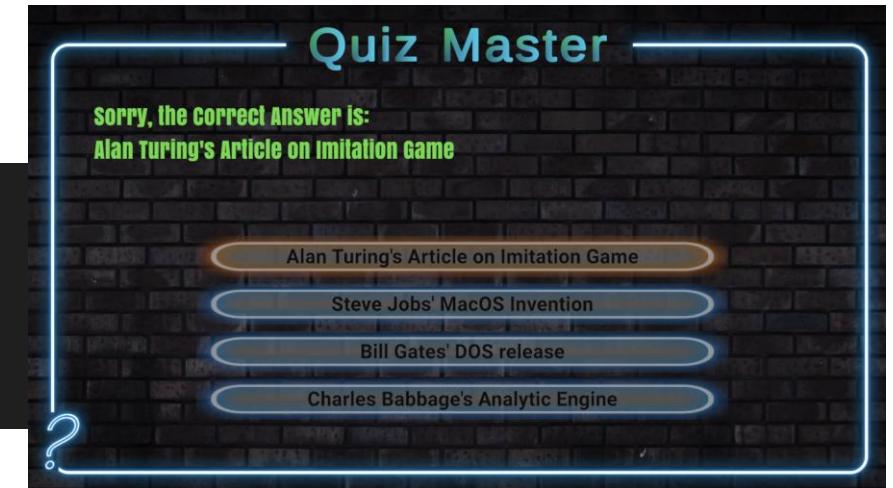
동일한 동작을 보여 줌 (당연)

버튼 상태 설정 메소드 추가

SetButtonState

```
1 reference
void SetButtonState(bool state)
{
    for(int i = 0; i < answerButtons.Length; i++)
    {
        Button button = answerButtons[i].GetComponent<Button>();
        button.interactable = state;
    }
}
```

```
public void OnAnswerSelected(int index)
{
    ...
    SetButtonState(false);
}
```



답변 선택 후에는 버튼이 비활성화 됨

새로운 문제 출력

모든 버튼을 활성화

모든 버튼에 디폴트 스프라이트 설정

- 모든 버튼을 돌며 이미지 컴포넌트 얻기
- 이미지 컴포넌트를 디폴트 스프라이트로 전환

```
1 reference
void GetNextQuestion()
{
    SetButtonState(true);
    SetDefaultButtonSprites();
    DisplayQuestion();
}

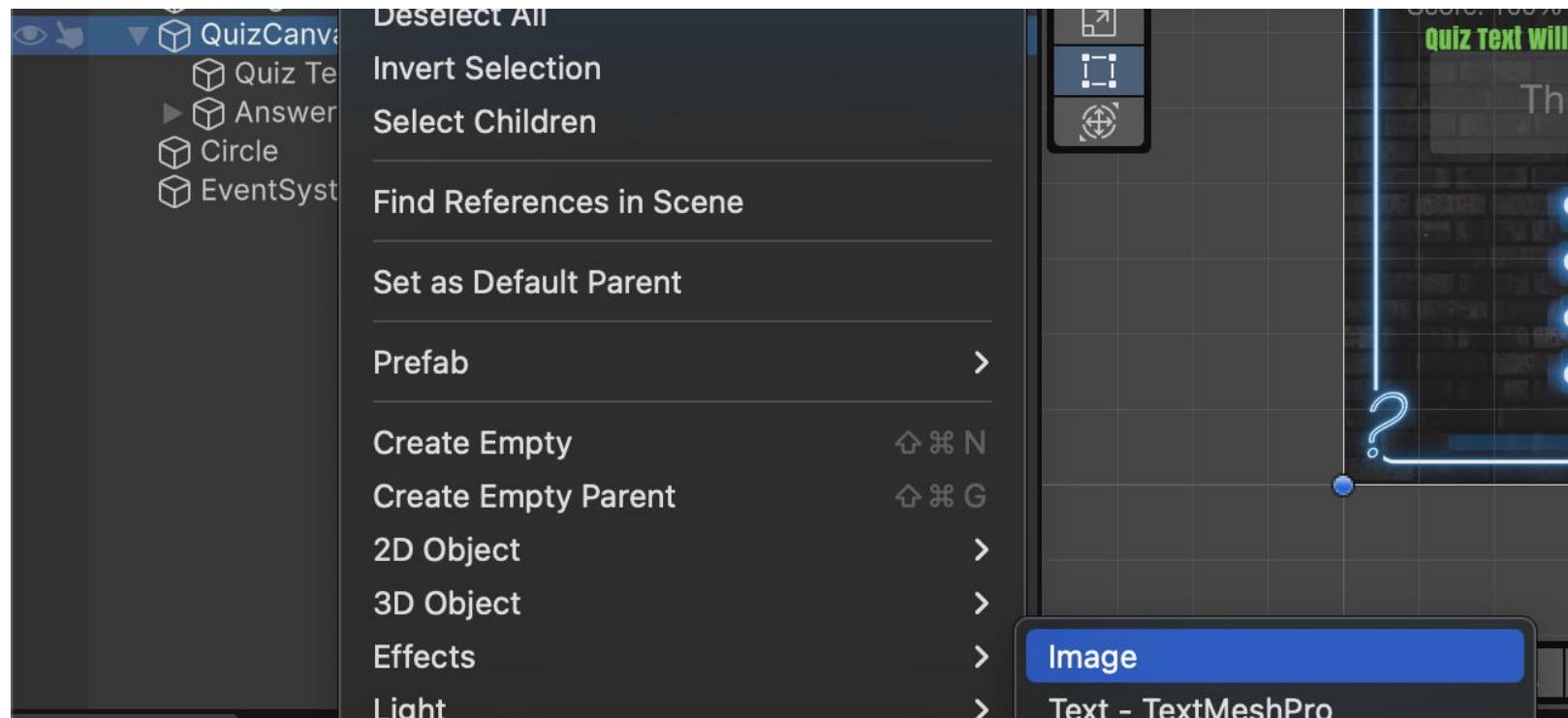
1 reference
void SetDefaultButtonSprites()
{
    for(int i = 0; i < answerButtons.Length; i++)
    {
        Image buttonImage = answerButtons[i].GetComponent<Image>();
        buttonImage.sprite = defaultAnswerSprite;
    }
}
```

```
void Start()
{
    //DisplayQuestion();
    GetNextQuestion();
}
```

다음 문제로 넘어가기 전에

타이머 다루기

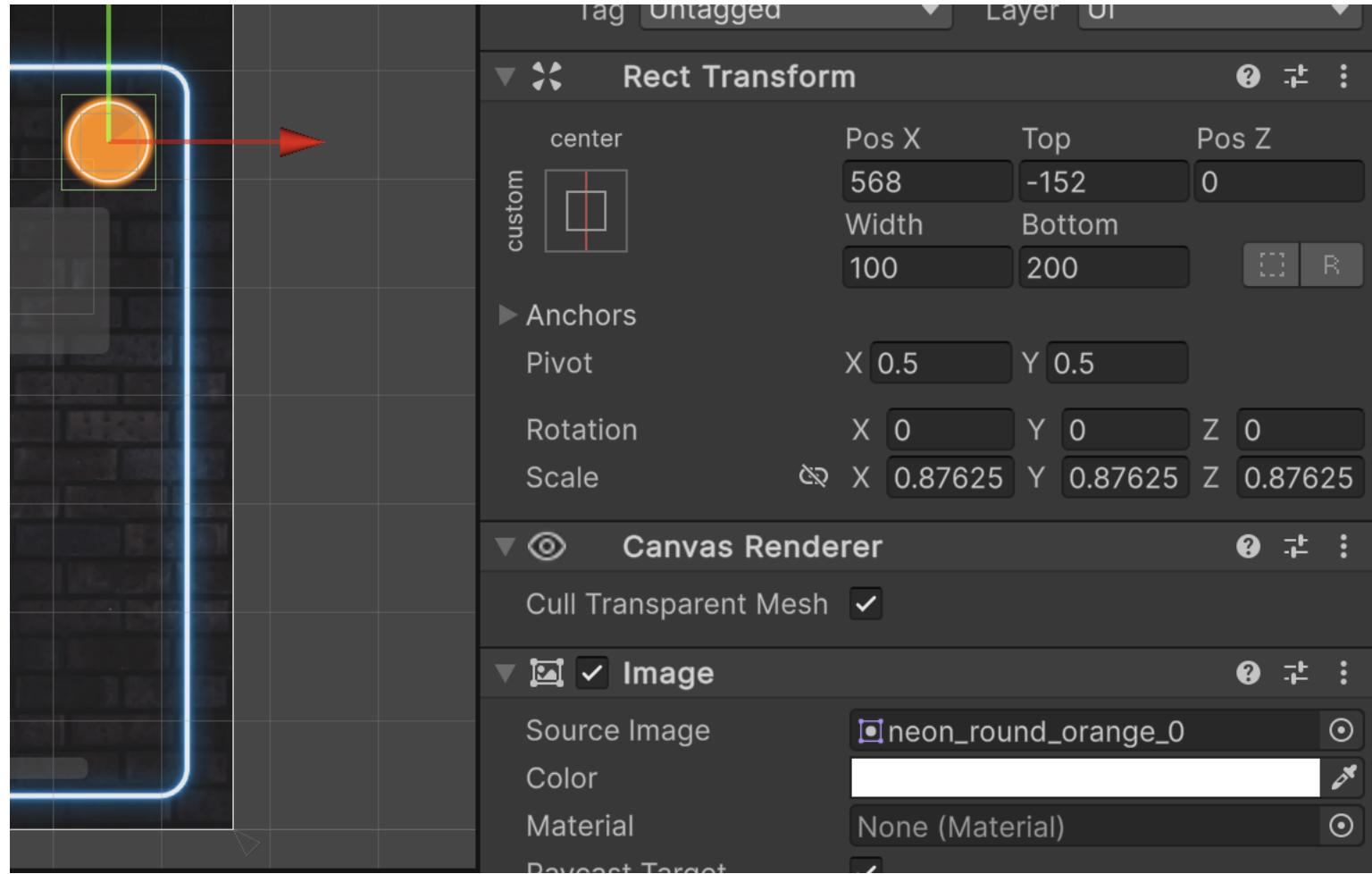
- 문제를 보여주고 답변 제한 시간 설정
- 답변 제한 시간이 지나면 → 정답 보여 주기
- 정답 보여 주기 시간 설정 → 정답 보여주기 시간이 지나면 → 다음 문제 보이기



Timer 상태를 보일
이미지를 캔버스에 추가

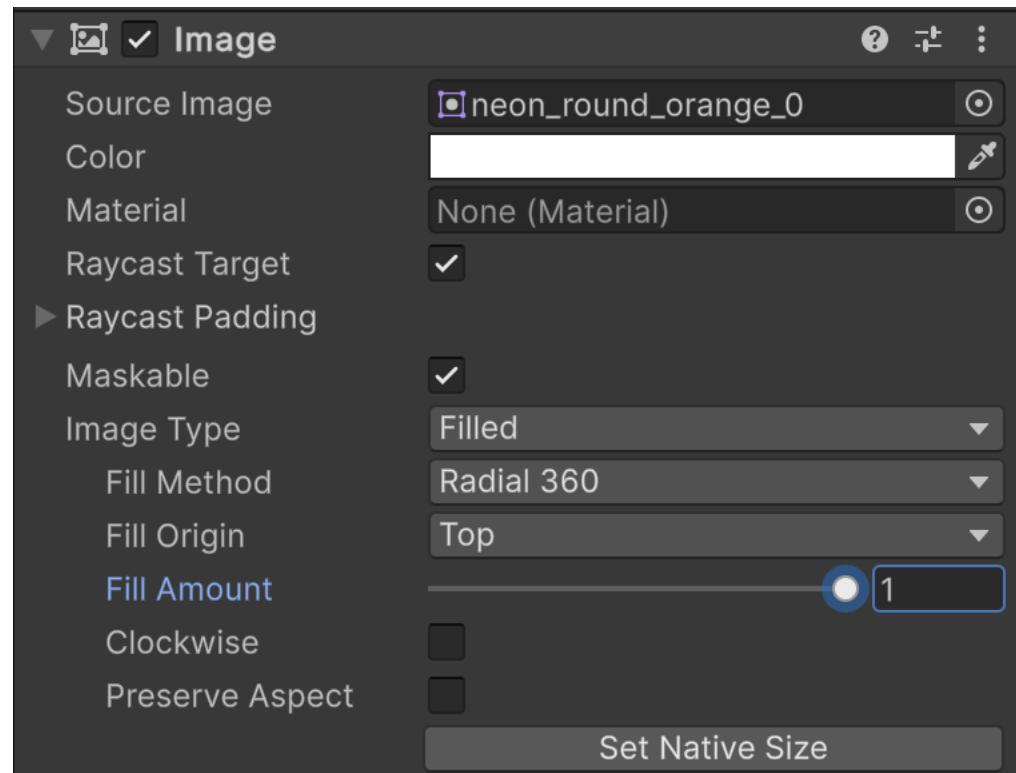
타이머 이미지

스프라이트 설정

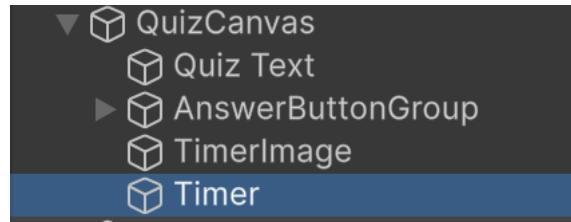


타이머 이미지를 “filled type”으로 변경

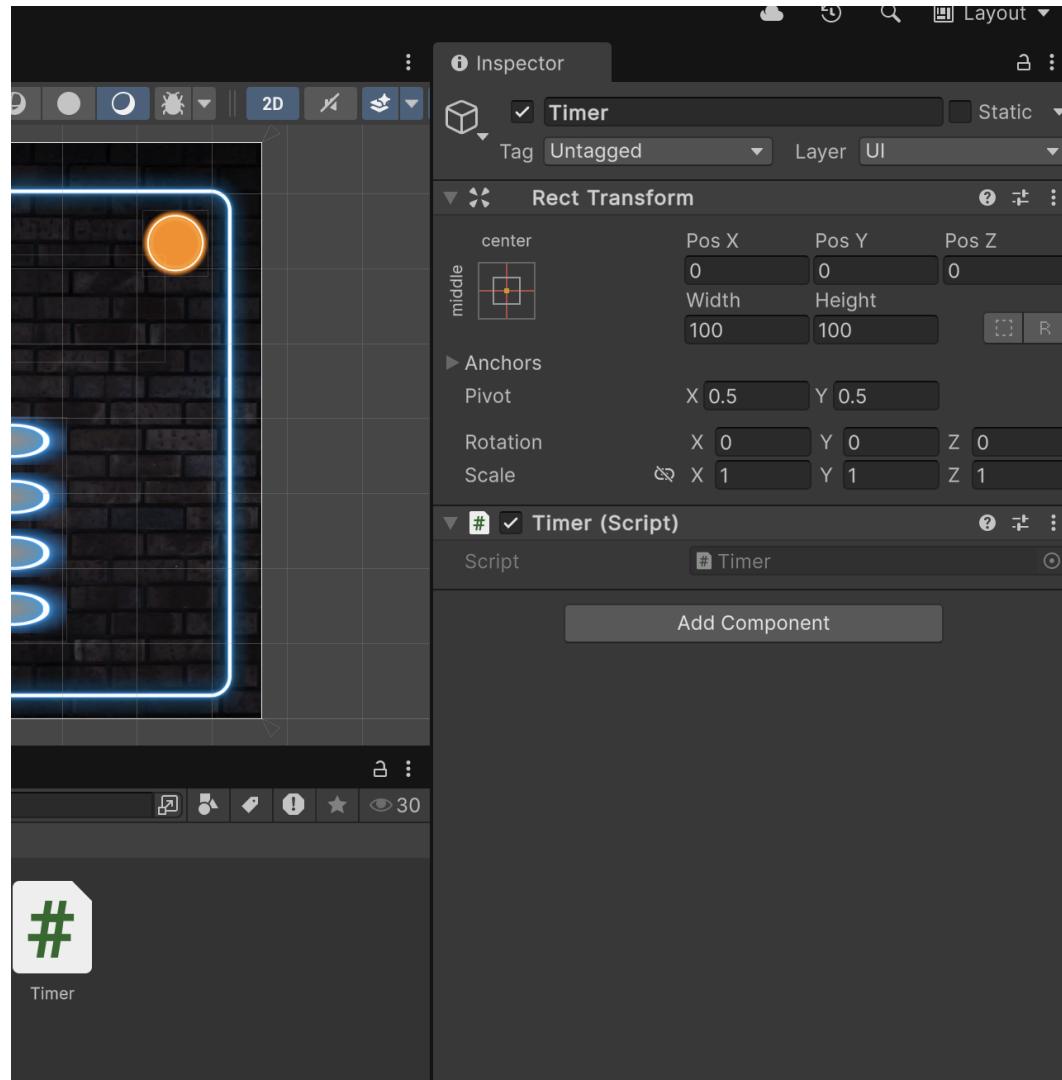
Fill Amount로 모양 조절 가능



타이머의 동작을 위해 타이머 객체 생성 / 스크립트 생성



Empty Object “Timer”



Timer 객체에 Timer.cs
스크립트 생성해 연결

타이머가 할 일

게임이 어떤 상태인지 확인

- 답변을 기다리는 상태
- 정답을 보여주는 상태

타이머가 시간 측정을 하도록 함

타이머 이미지의 상태를 타이머 상태에 맞춰 변경

타이머가 다 돌았으면 게임 상태 변경

Timer.cs

```
using UnityEngine;
using UnityEngine.UIElements;

0 references
public class Timer : MonoBehaviour
{
    2 references
    [SerializeField] float timeToCompletQuestion = 30f;
    2 references
    [SerializeField] float timeToShowCorrectAnswer = 10f;

    1 reference
    public bool loadNextQuestion;
    2 references
    public float fillFraction;
    3 references
    public bool isAnsweringQuestion;

    8 references
    float timerValue;

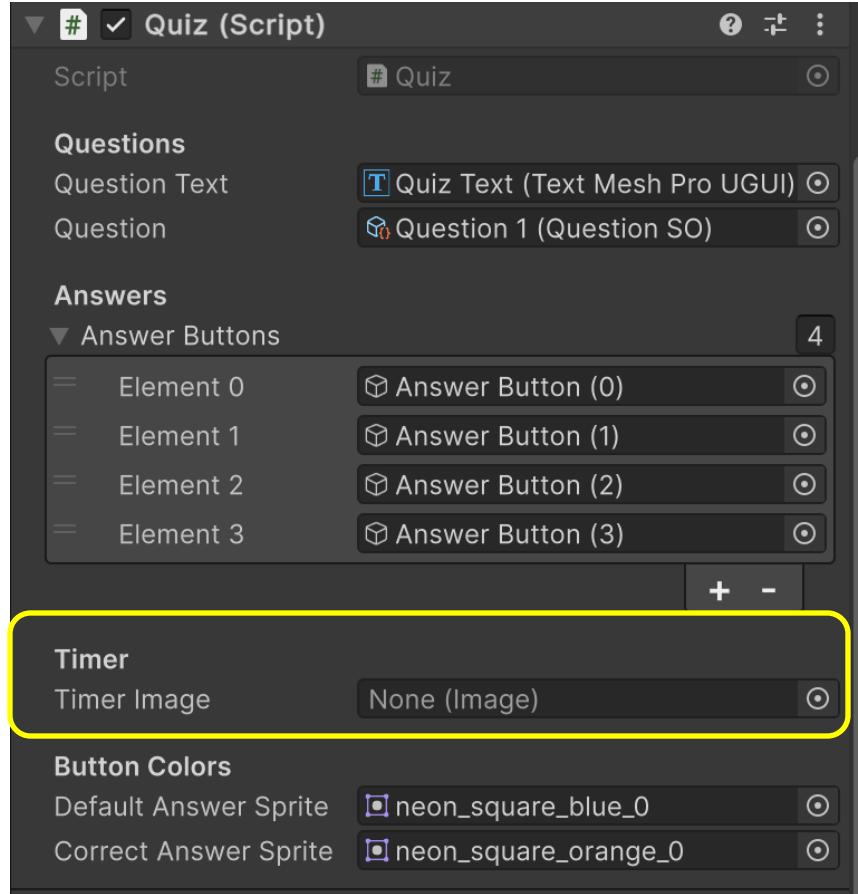
    0 references
    void Update()
    {
        UpdateTimer();
    }

    0 references
    public void CancelTimer()
    {
        timerValue = 0;
    }
}
```

```
public void UpdateTimer()
{
    timerValue -= Time.deltaTime;

    if(isAnsweringQuestion)
    {
        if(timerValue > 0 )
        {
            fillFraction = timerValue / timeToCompletQuestion;
        }
        else
        {
            isAnsweringQuestion = false;
            timerValue = timeToShowCorrectAnswer;
        }
    }
    else // showing correct answer
    {
        if (timerValue > 0 )
        {
            fillFraction = timerValue / timeToShowCorrectAnswer;
        }
        else
        {
            isAnsweringQuestion = true;
            timerValue = timeToCompletQuestion;
            loadNextQuestion = true;
        }
    }
}
```

Quiz script 변경



```
[Header("Questions")]
3 references
[Serializable] TextMeshProUGUI questionText;
4 references
[Serializable] QuestionSO question;

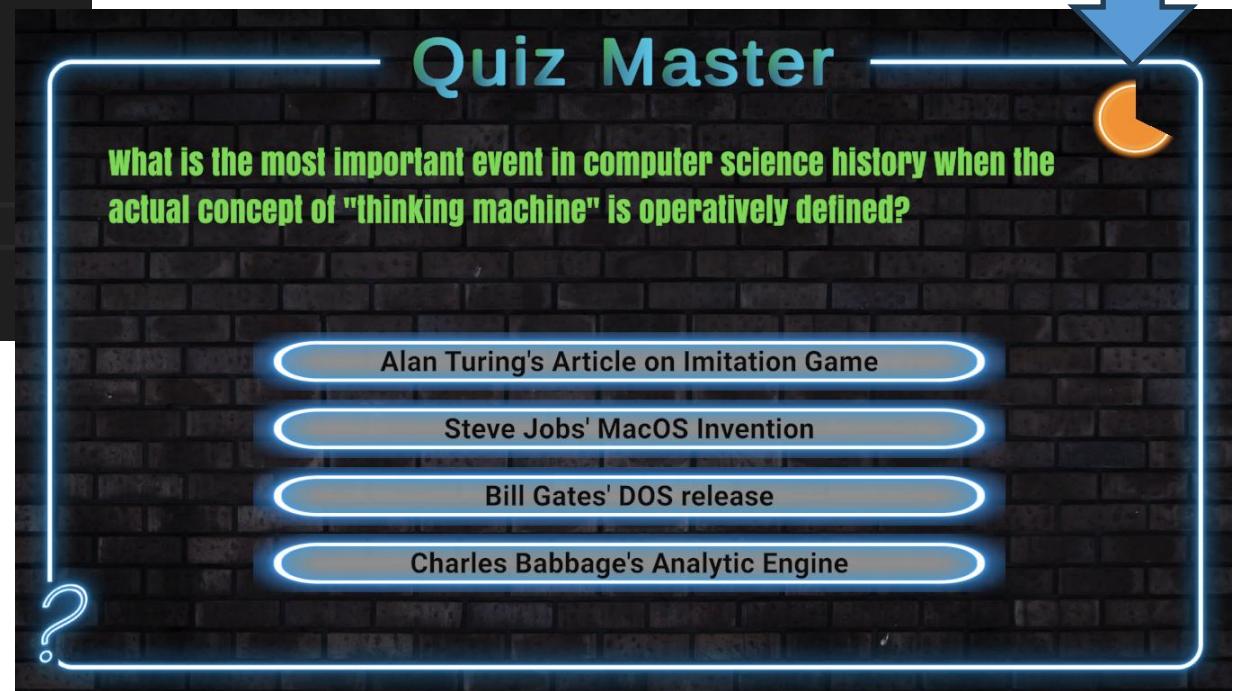
[Header("Answers")]
8 references
[Serializable] GameObject[] answerButtons;
4 references
int correctAnswerIndex;

[Header("Timer")]
0 references
[Serializable] Image timerImage;
0 references
Timer ...;

[Header("Button Colors")]
1 reference
[Serializable] Sprite defaultAnswerSprite;
1 reference
[Serializable] Sprite correctAnswerSprite;
```

Quiz와 Timer 스크립트 연결

```
0 references
void Start()
{
    //DisplayQuestion();
    GetNextQuestion();
    timer = FindFirstObjectByType<Timer>();
}
0 references
void Update()
{
    timerImage.fillAmount = timer.fillFraction;
}
```



일찍 답을 했을 때

```
public void OnAnswerSelected(int index)
{
    correctAnswerIndex = question.GetCorrectAnswerIndex();
    Image buttonImage;

    if(index == correctAnswerIndex)
    {
        questionText.text = "Correct!";
        buttonImage = answerButtons[index].GetComponent<Image>();
    }
    else {
        string correctAnswer = question.GetAnswer(correctAnswerIndex);
        questionText.text = "Sorry, the Correct Answer is: \n" + correctAnswer;
        buttonImage = answerButtons[correctAnswerIndex].GetComponent<Image>();
    }
    buttonImage.sprite = correctAnswerSprite;

    timer.process_Answer(); // Answer is given, the timer should be changed

    SetButtonState(false);
}
```

답이 주어졌으므로
정답을 확인하는 단계로
타이머 전환

Timer의 변화

```
public void UpdateTimer()
{
    timerValue -= Time.deltaTime;

    if(isAnsweringQuestion)
    {
        if(timerValue > 0 )
        {
            fillFraction = timerValue / timeToCompleterQuestion;

        }
        else
        {
            process_Answer();
        }
    }
}
```

2 references

```
public void process_Answer()
{
    isAnsweringQuestion = false;
    timerValue = timeToShowCorrectAnswer;
}
```

정답 보여주기가 끝나면 문제 다시 로드 – Quiz.cs

```
void Start()
{
    //DisplayQuestion();
    GetNextQuestion();
    timer = FindFirstObjectByType<Timer>();
    timer.loadNextQuestion = false;
}
0 references
void Update()
{
    timerImage.fillAmount = timer.fillFraction;
    if (timer.loadNextQuestion)
    {
        GetNextQuestion();
        timer.loadNextQuestion = false;
    }
}
```

같은 문제만
반복해서 보이게 됨
→ 문제를 바꾸기

List 사용하기

배열

→ `Int[] oddNumbers = new int[5]`

리스트

→ `List<int> oddNumbers = new List<int>()`

List의 유용한 메소드

아이템 개수 체크: List.Count

특정 아이템 존재 확인: List.Contains(item)

아이템 추가: List.Add(item)

아이템 삭제: List.Remove(item)

특정 위치 아이템 삭제: List.RemoveAt(idx)

리스트 모두 삭제: List.Clear()

여러 질문에서 하나를 다루게 변경

여러 질문은 리스트로 담고, `SerializeField`로 설정

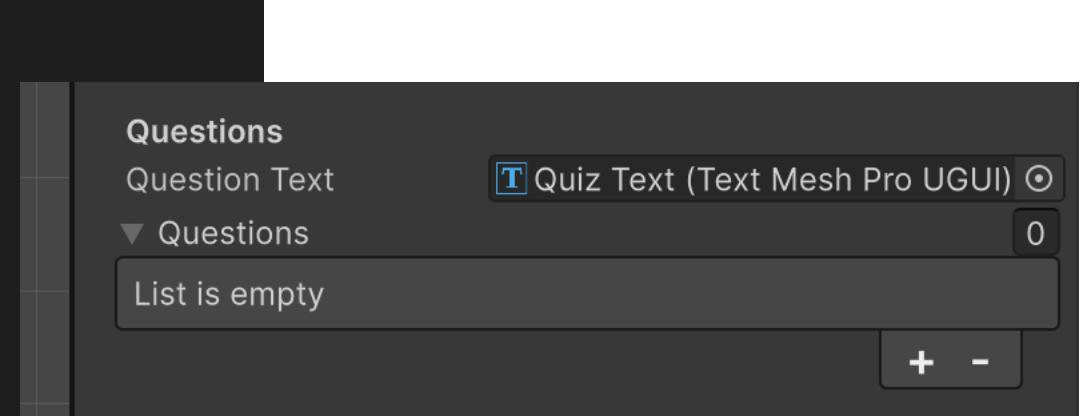
```
public class Quiz : MonoBehaviour
{
    [Header("Questions")]
    3 references
    [SerializeField] TextMeshProUGUI questionText;
    0 references
    [SerializeField] List<QuestionSO> questions = new List<QuestionSO>();
    4 references
    QuestionSO question;
```

다음 질문을 랜덤하게 가져오기

리스트에서 선택

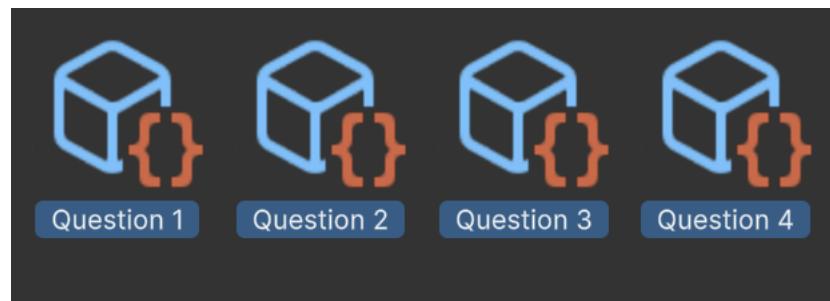
```
void GetNextQuestion()
{
    SetButtonState(true);
    SetDefaultButtonSprites();
    GetRandomQuestion();
    DisplayQuestion();
}

1 reference
void GetRandomQuestion()
{
    int index = Random.Range(0, questions.Count);
    question = questions[index];
    questions.Remove(question);
}
```



현재 질문 리스트는 비어
있음

리스트 채우기



Inspector

Canvas Scaler

- UI Scale Mode: Constant Pixel Size
- Scale Factor: 1
- Reference Pixels Per Unit: 100

Graphic Raycaster

- Script: GraphicRaycaster
- Ignore Reversed Gravity:
- Blocking Objects: None
- Blocking Mask: Everything

Quiz (Script)

- Script: Quiz

Questions

Question Text: Quiz Text (Text Mesh Pro UGUI)

Questions (Count: 4)

- = Element 0: Question 1 (Question SO)
- = Element 1: Question 2 (Question SO)
- = Element 2: Question 3 (Question SO)
- = Element 3: Question 4 (Question SO)

+ -

첫 시작에서 두 문제 가져오는 버그 해결

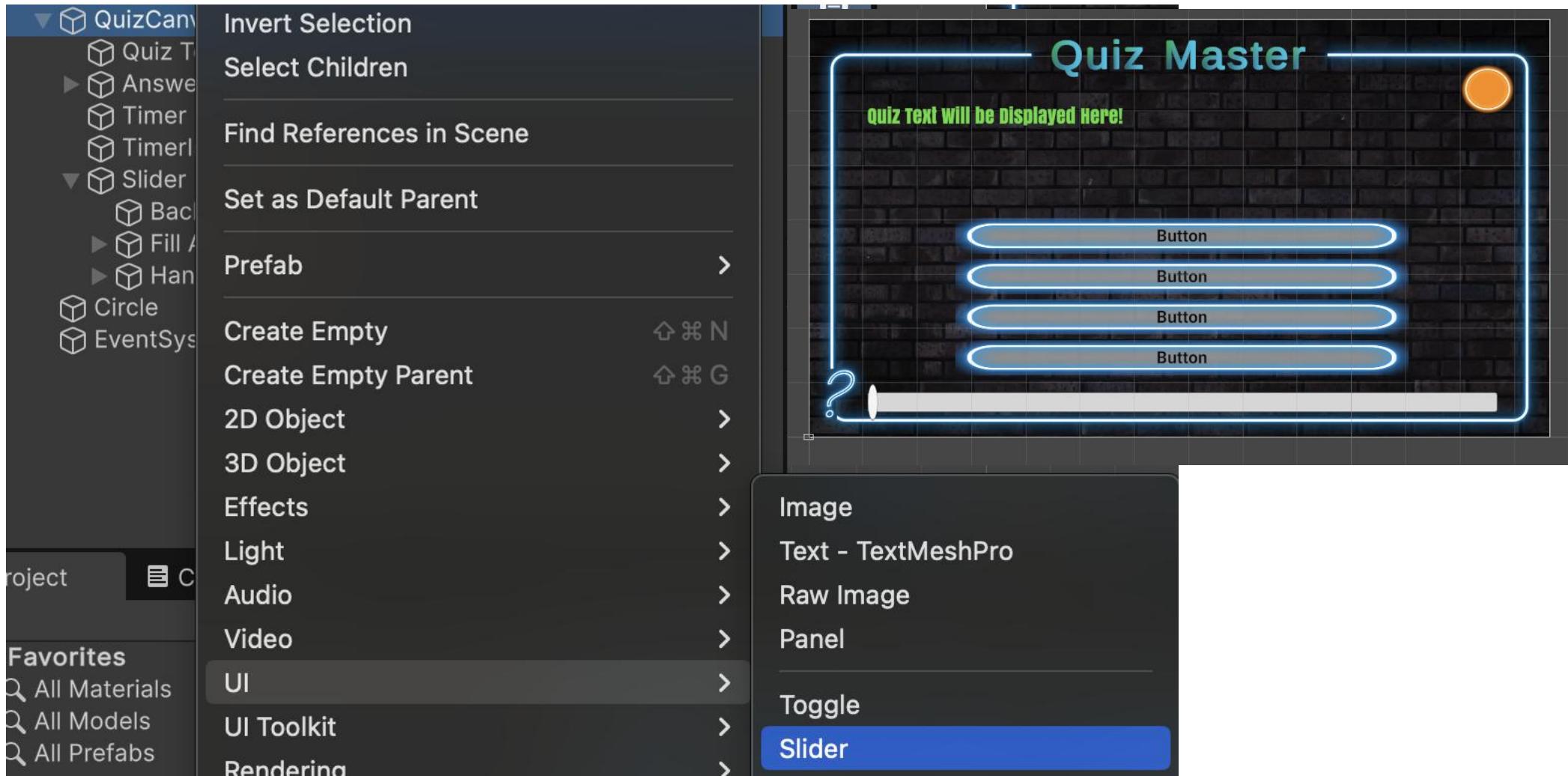
```
void Start()
{
    //DisplayQuestion();
    //GetNextQuestion();
    timer = FindFirstObjectByType<Timer>();
    timer.loadNextQuestion = true;
}

0 references

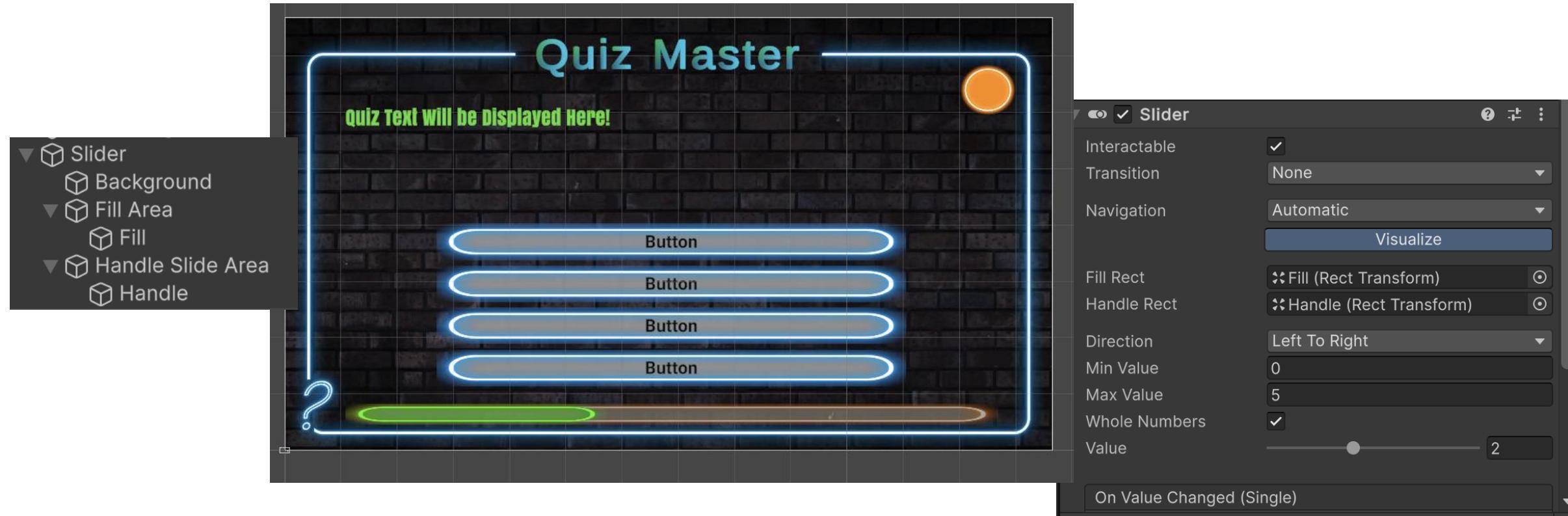
void Update()
{
    timerImage.fillAmount = timer.fillFraction;
    if (timer.loadNextQuestion)
    {
        GetNextQuestion();
        timer.loadNextQuestion = false;
    }
}
```

문제를 처음에 가져온 뒤
바로 Update에서 가져오는
문제 해결

진행상황 보여주기 – Progress Bar

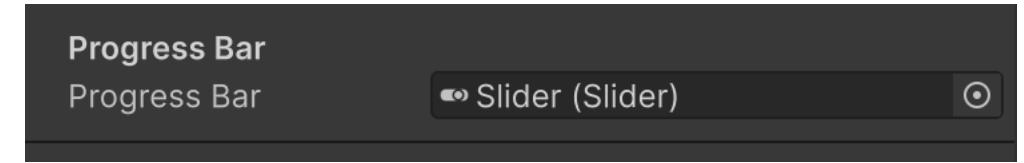


슬라이더 각 요소에 스프라이트 설정



Quiz.cs 고쳐서 progressBar를 인스펙터에서 지정

```
[Header("Progress Bar")]
0 references
[SerializeField] Slider progressBar;
```



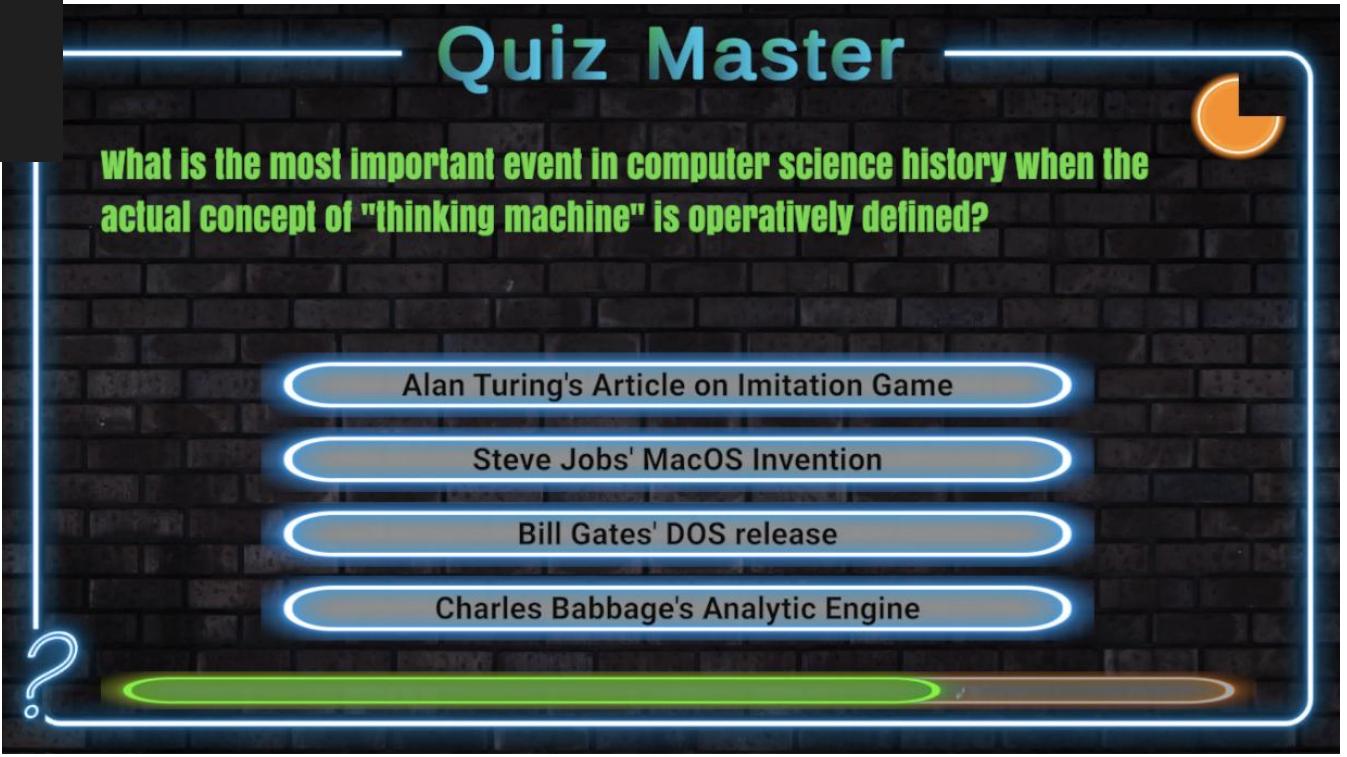
코드 수정하여 슬라이더 동작 완성

```
void Start()
{
    //DisplayQuestion();
    //GetNextQuestion();
    timer = FindFirstObjectByType<Timer>();
    timer.loadNextQuestion = true;

    progressBar.MaxValue = questions.Count;
    progressBar.value = 0;
}
```

```
void GetNextQuestion()
{
    if (questions.Count > 0)
    {
        SetButtonState(true);
        SetDefaultButtonSprites();
        GetRandomQuestion();
        DisplayQuestion();
        progressBar.value++;
    }
}
```

문제점: 슬라이더를 조작할 수 있다
해결법: Interactable flag



도전 과제 1

화면에 점수를 출력하자

도전 과제 2

모든 문제를 풀면 최종
점수를 출력하고,
다시 도전할지를 묻는
버튼을 달아 보자



이제 UI를 이용해
게임을 만들 수 있나요?