# Diseño de Sistemas Interactivos

## Curso 2023 - 2024

**Profesor: José Manuel Velasco**

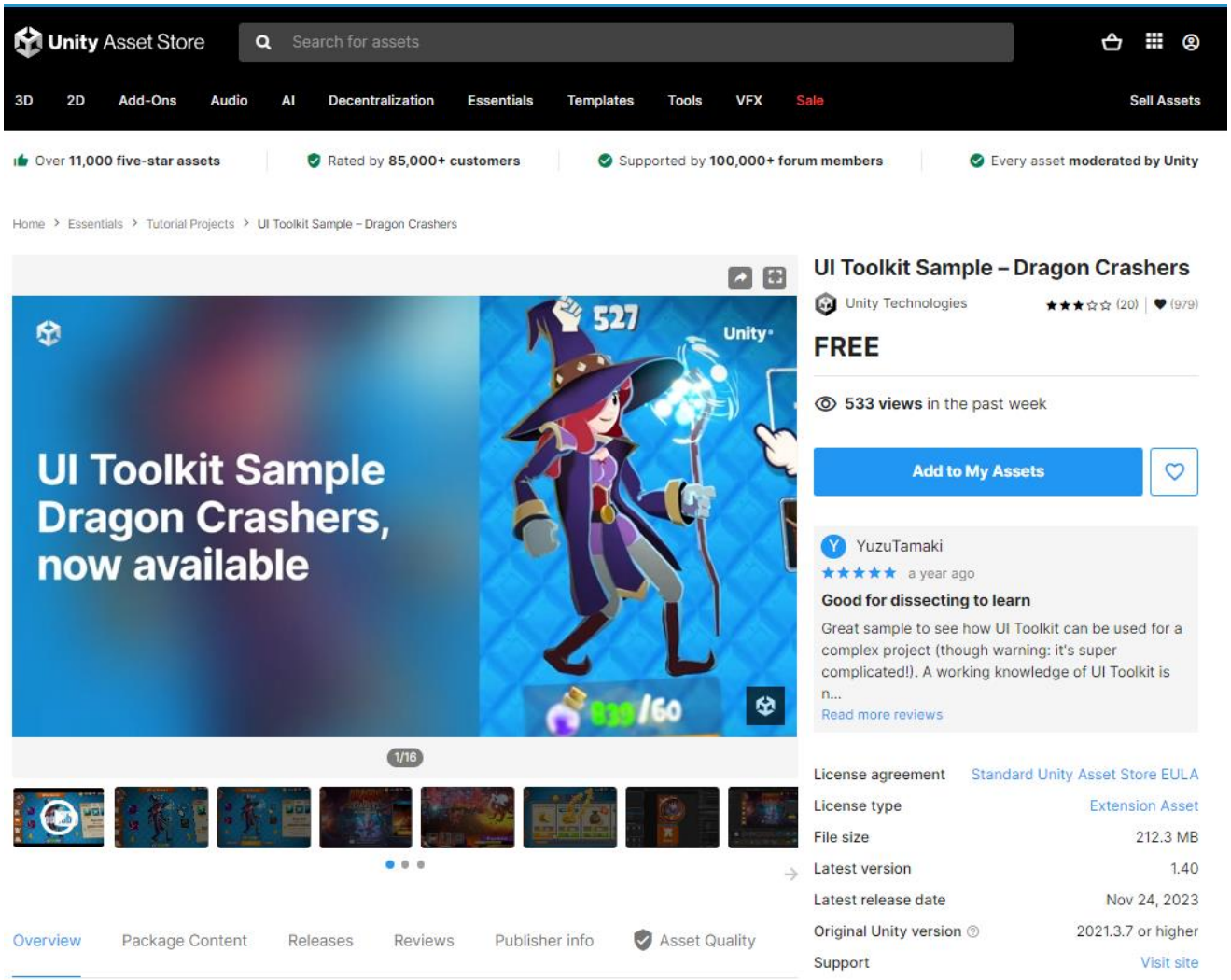
**Despacho 309, 3ª planta. Facultad de Informática.**

Laboratorio 7: Introducción a los UI Toolkit Project samples.

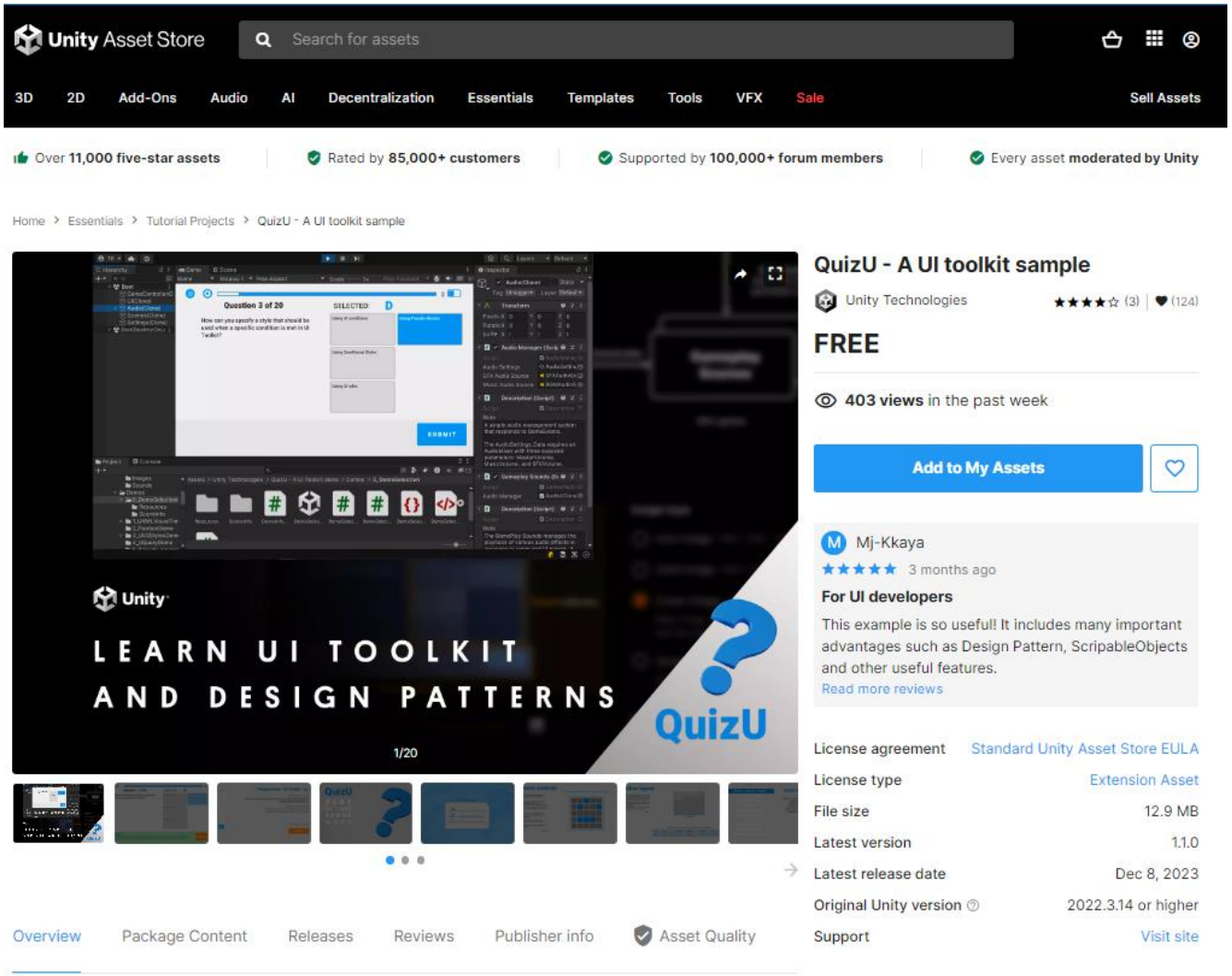
# Índice

- Project Samples
  - Dragon Crashers
    - Custom Controls
      - Tabbed menú
      - Radial progress bar
  - Quizu
    - Demos
      - CustomControls
      - Manipulators
    - Quiz
      - Data
      - Scripts
        - Coroutines
        - ScriptablesObjects
        - Actions
- Proyecto

# Dragon Crashers



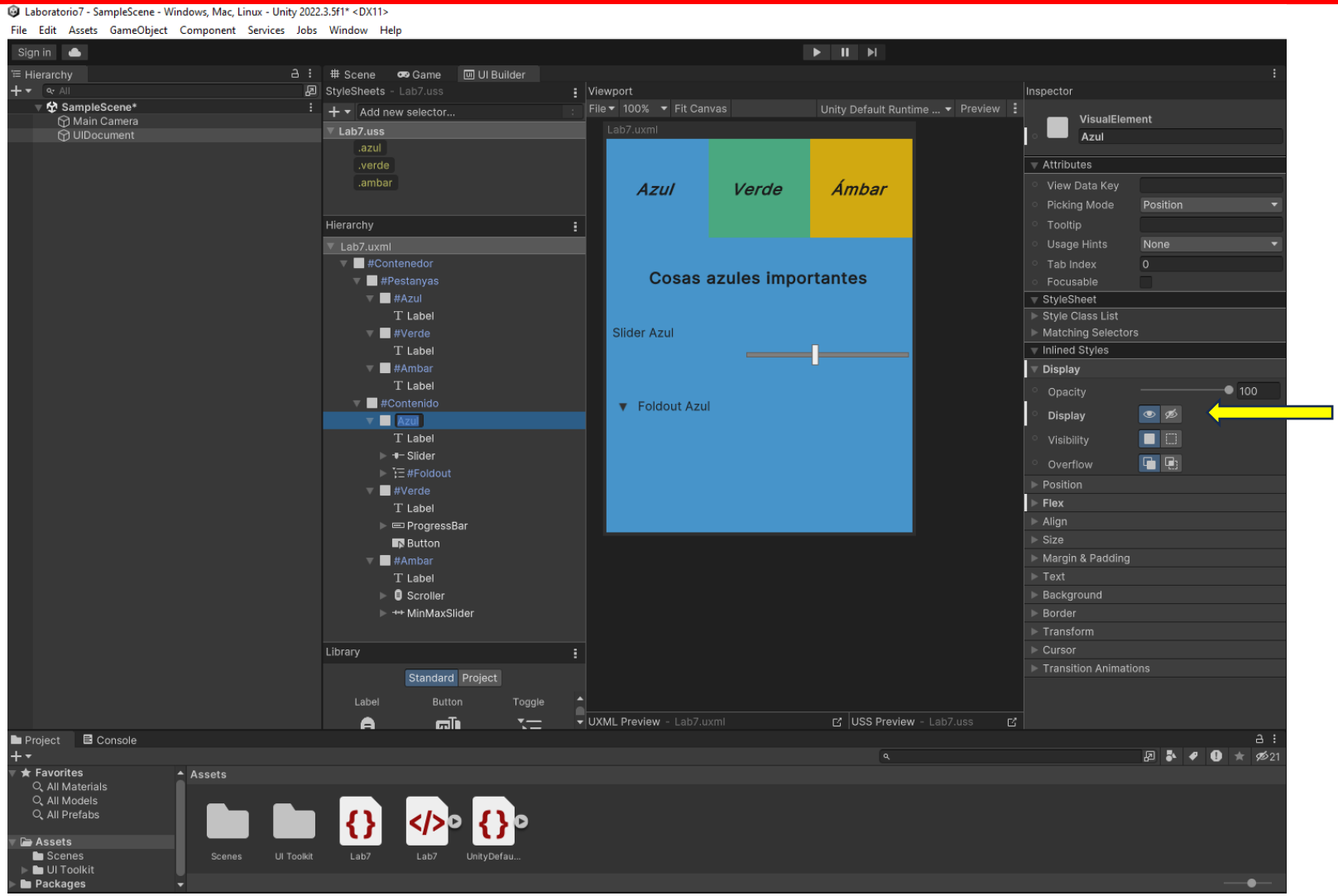
# QuizU


Unity Asset Store
Search for assets
3D
2D
Add-Ons
Audio
AI
Decentralization
Essentials
Templates
Tools
VFX
Sale
Sell Assets
Over 11,000 five-star assets
Rated by 85,000+ customers
Supported by 100,000+ forum members
Every asset moderated by Unity
Home > Essentials > Tutorial Projects > QuizU - A UI toolkit sample
Question 3 of 20
SELECTED:
SUBMIT
Unity
LEARN UI TOOLKIT
AND DESIGN PATTERNS
QuizU
1/20
QuizU - A UI toolkit sample
Unity Technologies
(3)
(124)
FREE
403 views in the past week
Add to My Assets
Mj-Kkaya
3 months ago
For UI developers
This example is so useful! It includes many important advantages such as Design Pattern, ScripableObjects and other useful features.
Read more reviews
License agreement
Standard Unity Asset Store EULA
License type
Extension Asset
File size
12.9 MB
Latest version
1.1.0
Latest release date
Dec 8, 2023
Original Unity version
2022.3.14 or higher
Support
Visit site
Overview
Package Content
Releases
Reviews
Publisher info
Asset Quality

# Índice

- Project Samples
  - **Dragon Crashers**
    - **Custom Controls**
      - **Tabbed menú**
      - **Radial progress bar**
  - Quizu
    - Demos
      - CustomControls
      - Manipulators
    - Quiz
      - Data
      - Scripts
        - Coroutines
        - ScriptablesObjects
        - Actions
- Proyecto

# Menú con pestañas muy sencillo



# Menú con pestañas

Lab7.cs

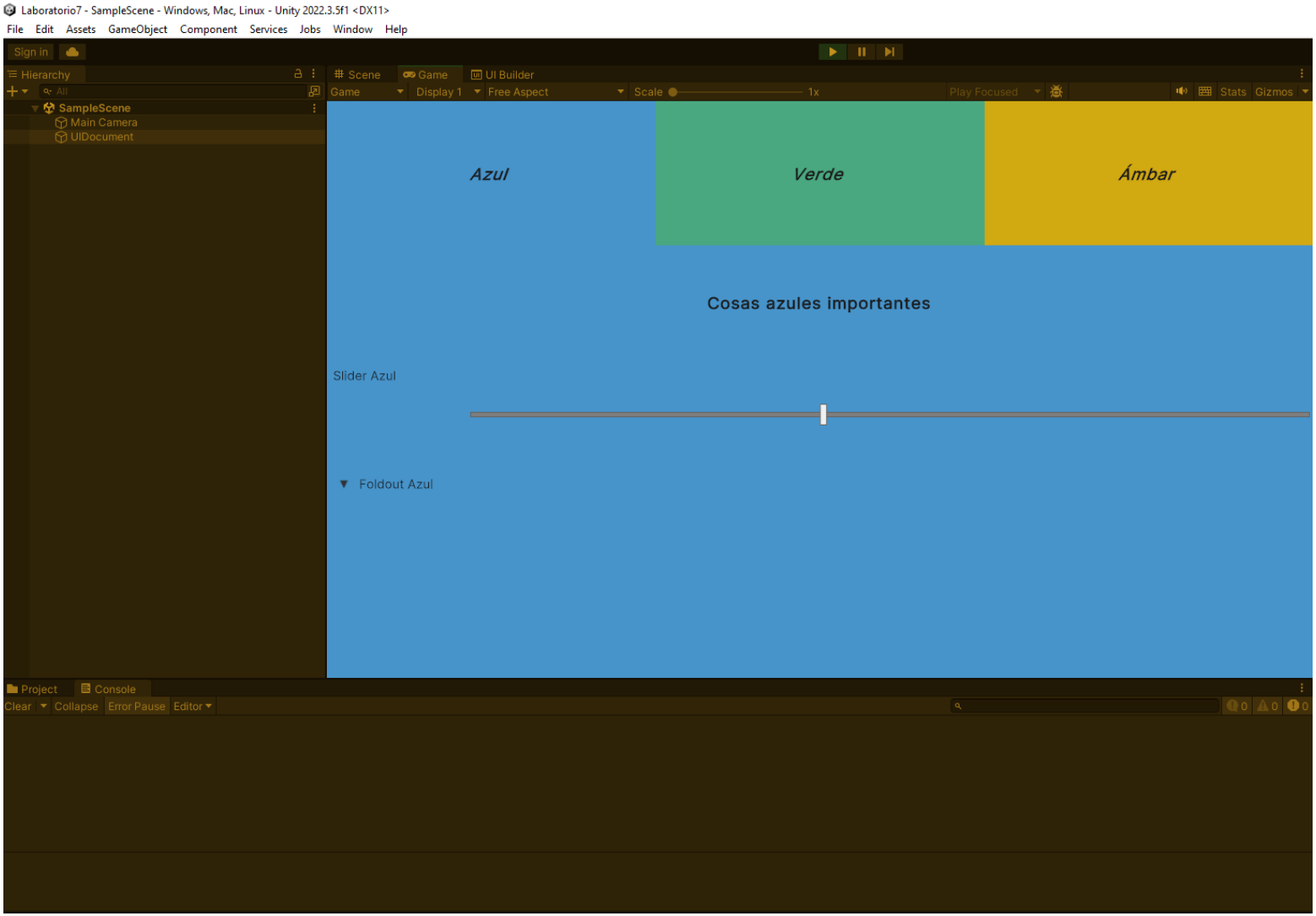
Assets > Lab7.cs > Lab7 > OnEnable

```
using UnityEngine;
using UnityEngine.UIElements;

0 references
public class Lab7 : MonoBehaviour
{
        3 references
    VisualElement contenido_azul;
        3 references
    VisualElement contenido_verde;
        3 references
    VisualElement contenido_ambar;
        2 references
    VisualElement pestanya_azul;
        2 references
    VisualElement pestanya_verde;
        2 references
    VisualElement pestanya_ambar;
    3 references
    private void NoContenido(){
        contenido_azul.style.display  = DisplayStyle.None;
        contenido_verde.style.display = DisplayStyle.None;
        contenido_ambar.style.display = DisplayStyle.None;
    }

```

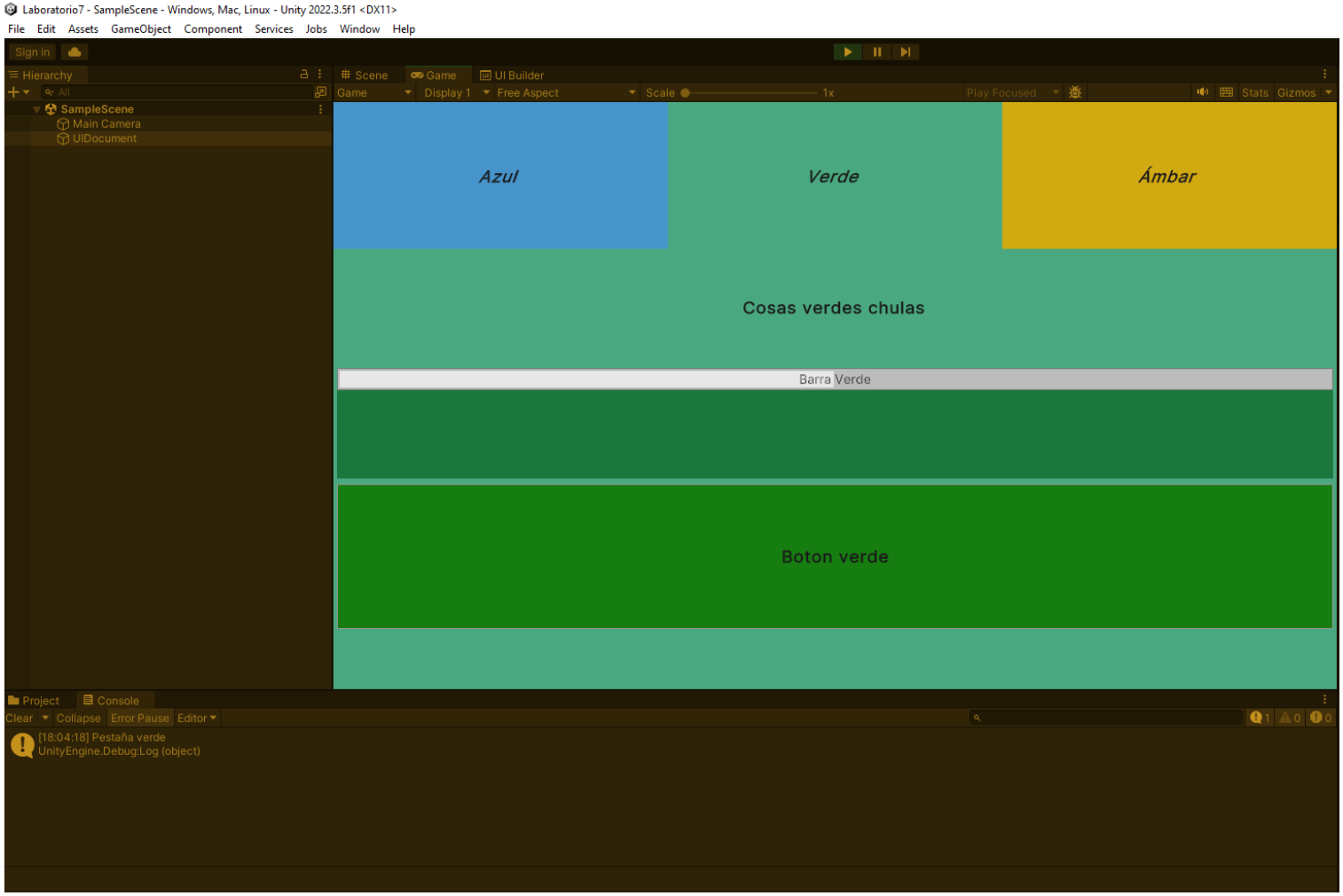
# Menú con pestañas

```
0 references
private void OnEnable() {
    UIDocument uidoc     = GetComponent<UIDocument>();
    VisualElement rootve = uidoc.rootVisualElement;

    VisualElement contenido = rootve.Q("Contenido");
    VisualElement pestanyas = rootve.Q("Pestanyas");

    pestanya_azul  = pestanyas.Q("Azul");
    pestanya_verde = pestanyas.Q("Verde");
    pestanya_ambar = pestanyas.Q("Ambar");

    contenido_azul  = contenido.Q("Azul");
    contenido_verde = contenido.Q("Verde");
    contenido_ambar = contenido.Q("Ambar");


    pestanya_azul.RegisterCallback<MouseDownEvent>(evt => {
        Debug.Log("Pestaña azul");
        NoContenido();
        contenido_azul.style.display  = DisplayStyle.Flex;
    });

    pestanya_verde.RegisterCallback<MouseDownEvent>(evt => {
                    Debug.Log("Pestaña verde");
        NoContenido();
        contenido_verde.style.display  = DisplayStyle.Flex;
    });

    pestanya_ambar.RegisterCallback<MouseDownEvent>(evt => {
        Debug.Log("Pestaña ambar");
        NoContenido();
        contenido_ambar.style.display  = DisplayStyle.Flex;
    });
}
}
```

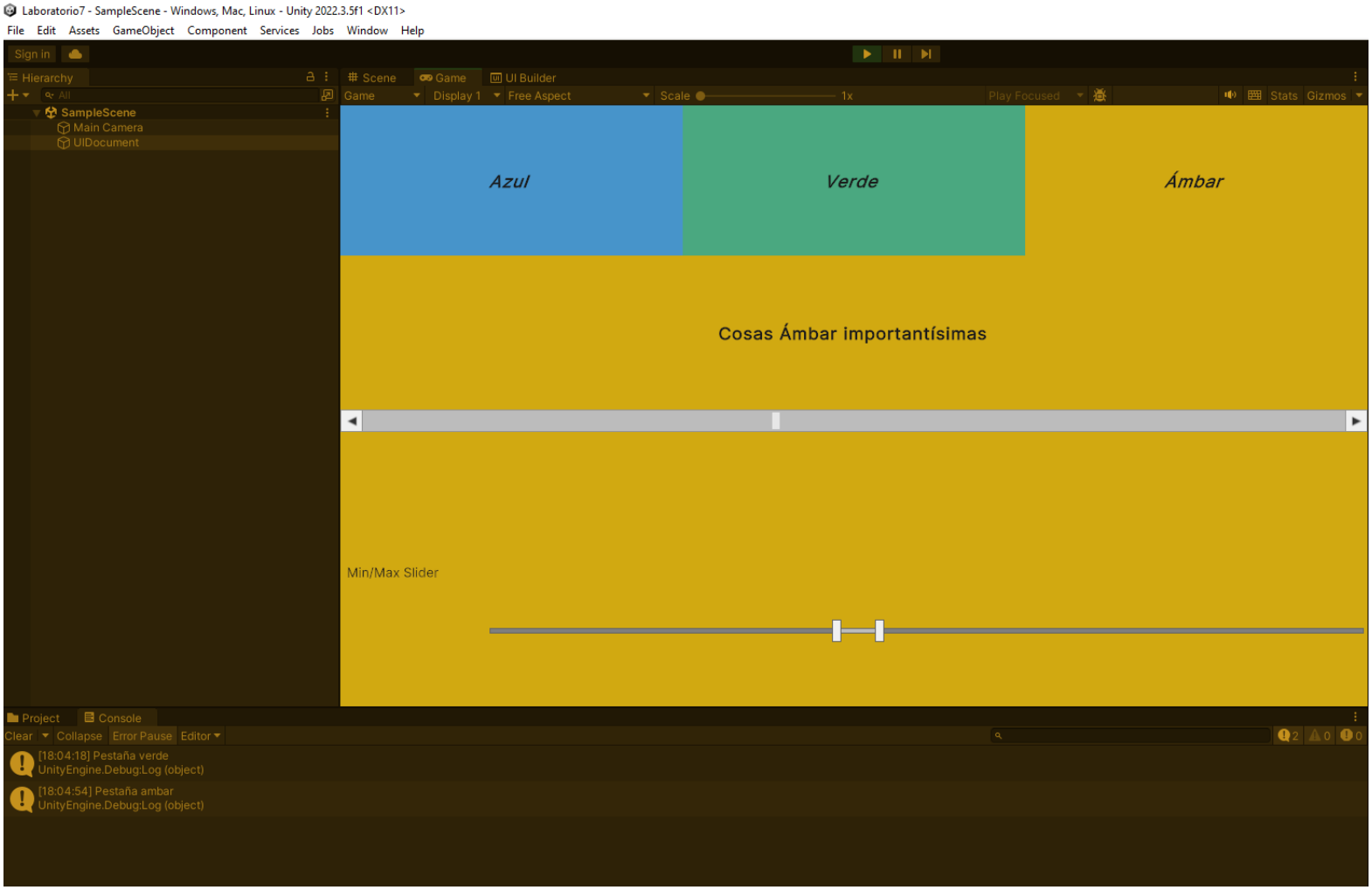
# Menú con pestañas



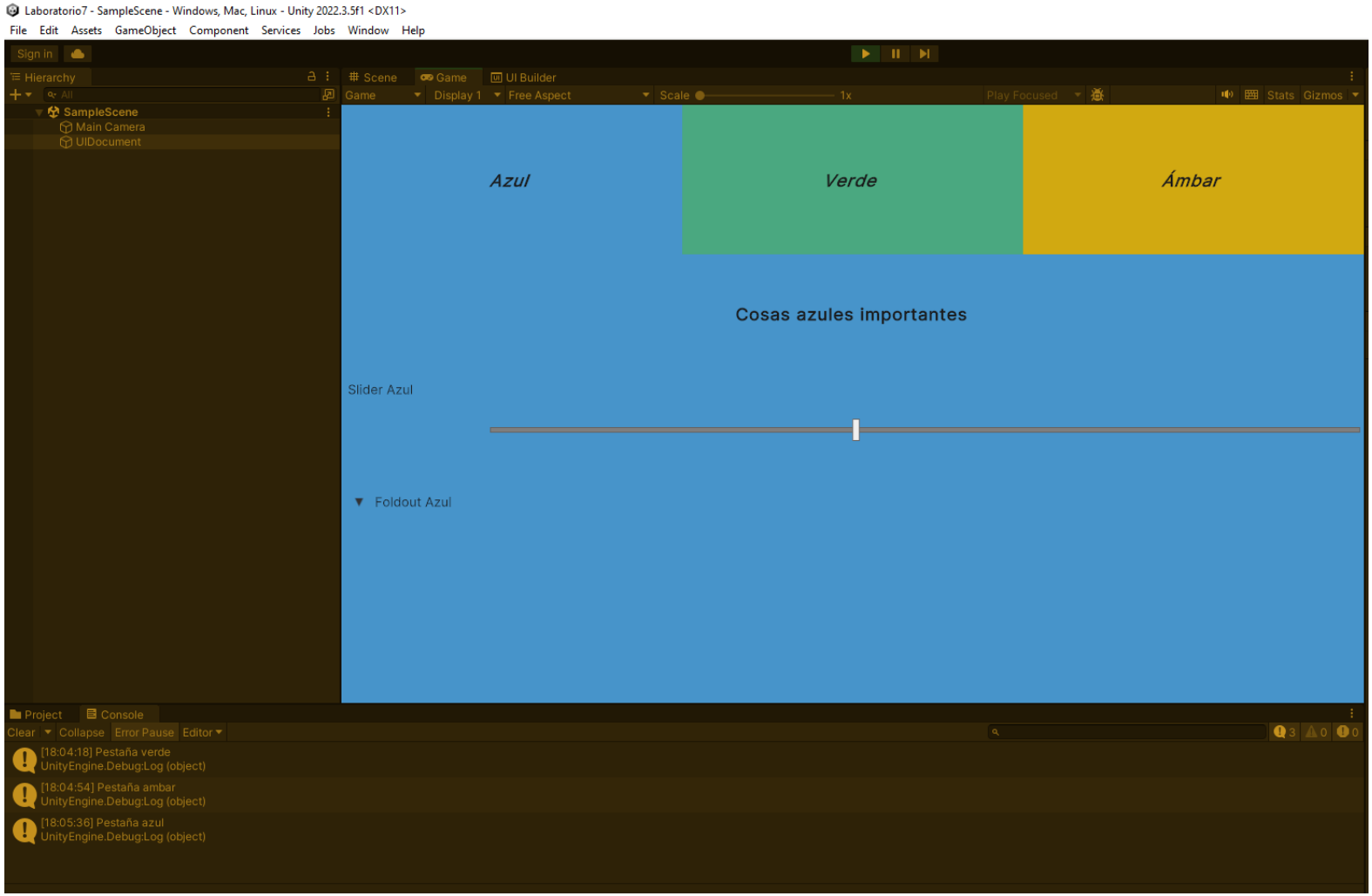
# Menú con pestañas



# Menú con pestañas



# Menú con pestañas



# Dragon Crashers → Tabbed menú

# Dragon Crashers → TabbedMenu (campus virtual)

TabbedMenuController.cs | TabbedMenu.cs

E: > PROYECTOS > UNITY > sample2 > Assets > Scripts > UI > Components > TabbedMenu.cs > ...

```
1   using UnityEngine;
2   using UnityEngine.UIElements;
3
4   namespace UIToolkitDemo
5   {
6       // adapted from: https://docs.unity3d.com/2021.2/Documentation/Manual/UIE-create-tabbed-menu-for-runtime.html
7
8       // This establishes a tabbed menu system for the UI document. This modifies the original to support multiple
9       // tabbed menus within the same document.
10
11      [System.Serializable]
            1 reference
12      public struct TabbedMenuIDs
13      {
14          // UXML selector for a clickable tab
                2 references
15          public string tabClassName;// = "tab";
16
17          // UXML selector for currently selected tab
                2 references
18          public string selectedTabClassName; //= "selected-tab";
19
20          // UXML selector for content to hide
                2 references
21          public string unselectedContentClassName; // = "unselected-content";
22
23          // use a basename to pair a tab with its content, e.g. 'name1-tab' matches 'name1-content'
24
25          // suffix naming convention for tabs
                2 references
26          public string tabNameSuffix;// = "-tab";
27
28          // suffix naming convention for content
                2 references
29          public string contentNameSuffix;// = "-content";
30
31      }
            0 references
```

# Dragon Crashers → TabbedMenu (campus virtual)

```
32      public class TabbedMenu : MonoBehaviour
            0 references
47          private void OnEnable()
48          {
49              if (m_Document == null)
50                  m_Document = GetComponent<UIDocument>();
51
52              VisualElement root = m_Document.rootVisualElement;
53              m_MenuElement = root.Q(m_MenuElementName);
54
55              // create a new TabbedMenuController for a specific element (fall back to the entire tree if unspecified)
56              m_Controller = (string.IsNullOrEmpty(m_MenuElementName) || m_MenuElement == null) ?
57                  new TabbedMenuController(root, m_TabbedMenuStrings) : new TabbedMenuController(m_MenuElement, m_TabbedMenuStrings);
58
59              // set up the click events on the tab
60              m_Controller.RegisterTabCallbacks();
61          }
62
63          // fill in default names for convenience - make these unique for each tabbed menu/UI
            0 references
64          void OnValidate()
65          {
66              if (string.IsNullOrEmpty(m_TabbedMenuStrings.tabClassName))
67              {
68                  m_TabbedMenuStrings.tabClassName = "tab";
69              }
70
71              if (string.IsNullOrEmpty(m_TabbedMenuStrings.selectedTabClassName))
72              {
73                  m_TabbedMenuStrings.selectedTabClassName = "selected-tab";
74              }
75
76              if (string.IsNullOrEmpty(m_TabbedMenuStrings.unselectedContentClassName))
77              {
78                  m_TabbedMenuStrings.unselectedContentClassName = "unselected-content";
79              }
80
81              if (string.IsNullOrEmpty(m_TabbedMenuStrings.tabNameSuffix))
82              {
83                  m_TabbedMenuStrings.tabNameSuffix = "-tab";
84              }
85
86              if (string.IsNullOrEmpty(m_TabbedMenuStrings.contentNameSuffix))
87              {
88                  m_TabbedMenuStrings.contentNameSuffix = "-content";
89              }
90          }
```

# Dragon Crashers → TabbedMenuController (campus virtual)

TabbedMenuController.cs

E: > PROYECTOS > UNITY > sample2 > Assets > Scripts > UI > Controllers > TabbedMenuController.cs > TabbedMenuController

```
using UnityEngine;
using UnityEngine.UIElements;
using System;

namespace UIToolkitDemo
{

    // adapted from: https://docs.unity3d.com/2021.2/Documentation/Manual/UIE-create-tabbed-menu-for-runtime.html

    1 reference
    public class TabbedMenuController
    {
        // event to notify other objects
        1 reference
        public static event Action TabSelected;

        // base VisualElement of the UI (e.g. MailScreen, CharScreen, ShopScreen)
        5 references
        readonly VisualElement m_Root;

        // strings to query for VisualElements
        12 references
        readonly TabbedMenuIDs m_IDs;

        // initialize the root Visual Element for reuse (constructor for non-Monobehaviour)
        0 references
        public TabbedMenuController(VisualElement root, TabbedMenuIDs ids)
        {
            m_Root = root;
            m_IDs = ids;
        }

        // set up click events for tab buttons
        0 references
        public void RegisterTabCallbacks()
        {
            // identify each tab
            UQueryBuilder<VisualElement> tabs = GetAllTabs();

            // register the ClickTab event handler for each Visual Element
            tabs.ForEach(
                (t) =>
                {
                    t.RegisterCallback<ClickEvent>(OnTabClick);
                });
        }
```

```
        // locate all VisualElements that have the tab class name
        4 references
        UQueryBuilder<VisualElement> GetAllTabs()
        {
            return m_Root.Query<VisualElement>(className: m_IDs.tabClassName);
        }
```

# Dragon Crashers → TabbedMenuController

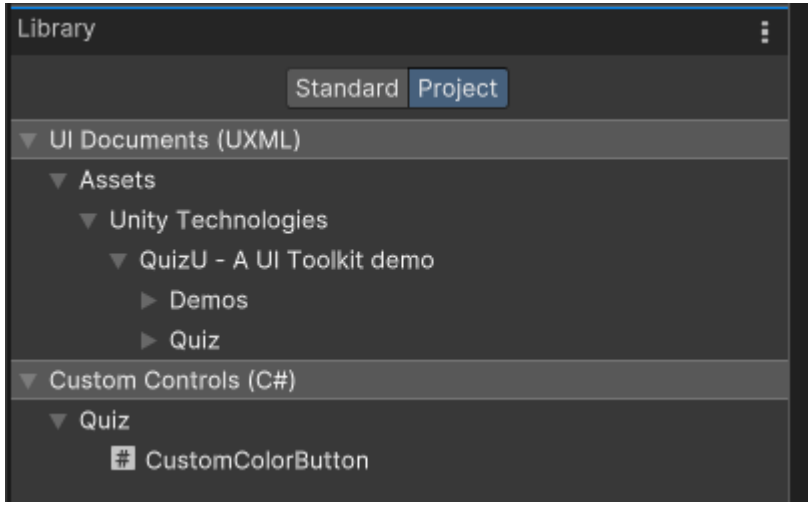
```
41
42      // process a click
        1 reference
43      void OnTabClick(ClickEvent evt)
44      {
45          VisualElement clickedTab = evt.currentTarget as VisualElement;
46
47          // if the clicked tab is not already selected, select the correct one
48          if (!IsTabSelected(clickedTab))
49          {
50              // de-select any other tabs that are currently active
51              GetAllTabs().Where(
52                  (tab) => tab != clickedTab && IsTabSelected(tab)
53                  ).ForEach(UnselectTab);
54
55              // select the clicked tab
56              SelectTab(clickedTab);
57              AudioManager.PlayDefaultButtonSound();
58          }
59      }
60
```

# Dragon Crashers → radial progress bar

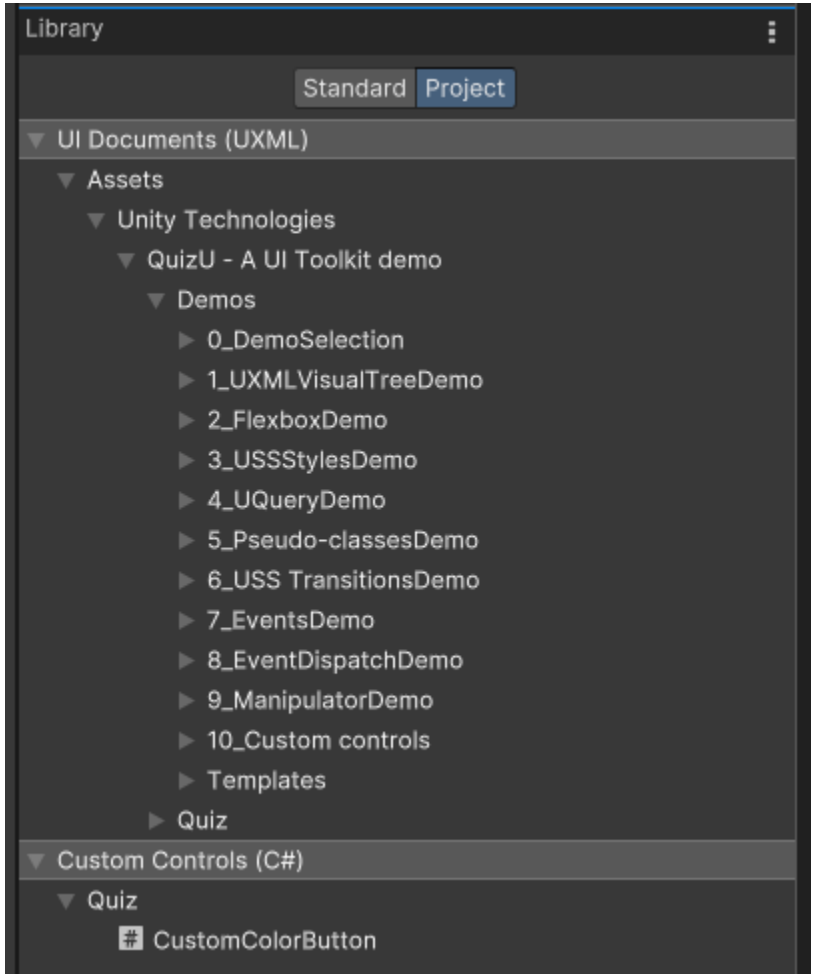
# Índice

- Project Samples
  - Dragon Crashers
    - Custom Controls
      - Tabbed menú
      - Radial progress bar
  - **Quizu**
    - **Demos**
      - **CustomControls**
      - **Manipulators**
    - Quiz
      - Data
      - Scripts
        - Coroutines
        - ScriptablesObjects
        - Actions
- Proyecto

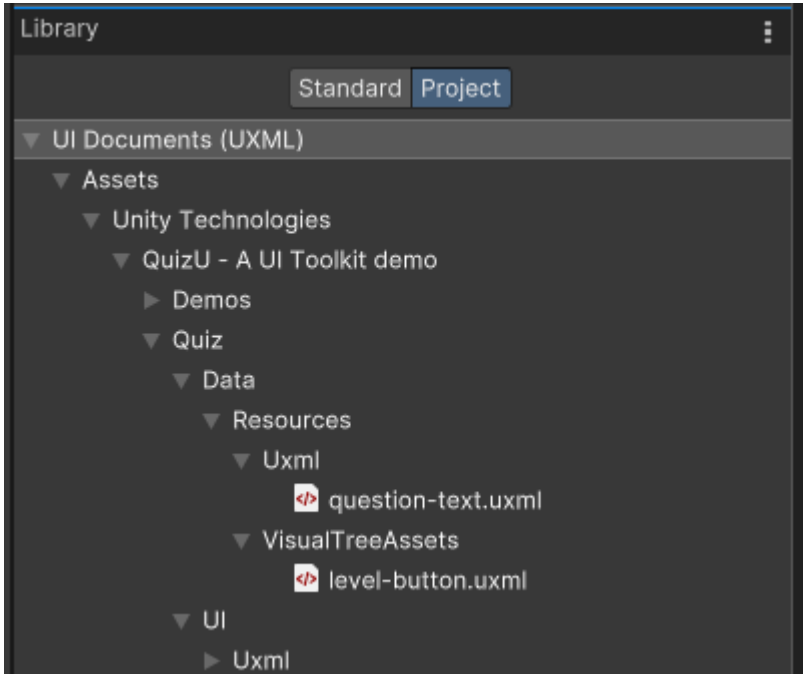
# QuizU

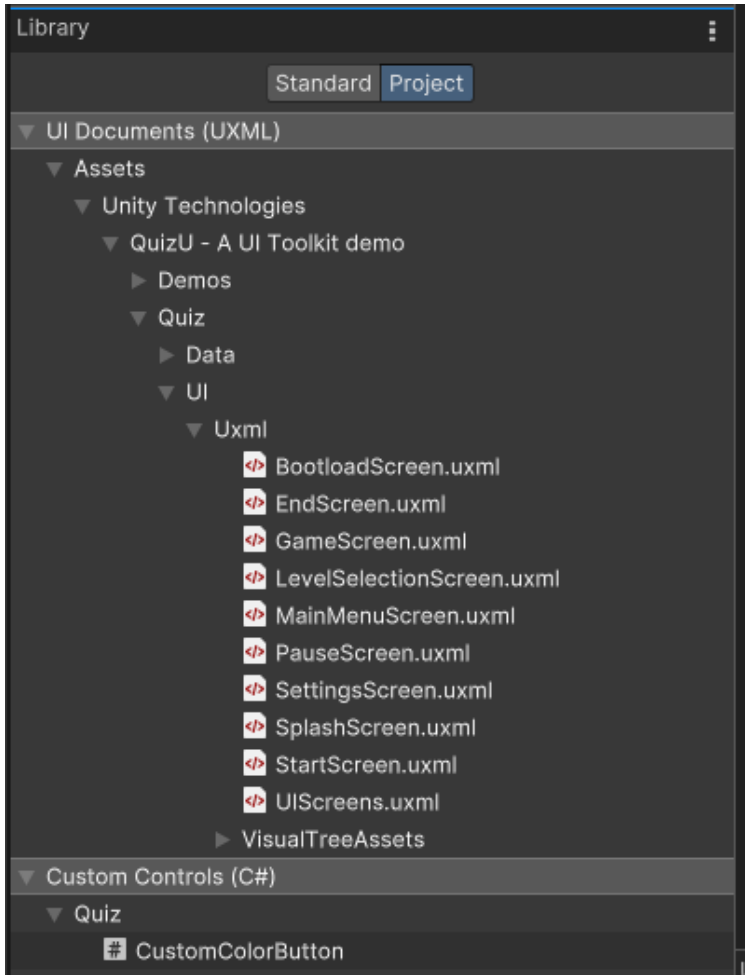

Library
Standard
Project
UI Documents (UXML)
Assets
Unity Technologies
QuizU - A UI Toolkit demo
Demos
Quiz
Custom Controls (C#)
Quiz
CustomColorButton

# QuizU - Demos


Library
Standard
Project
UI Documents (UXML)
Assets
Unity Technologies
QuizU - A UI Toolkit demo
Demos
0_DemoSelection
1_UXMLVisualTreeDemo
2_FlexboxDemo
3_USSStylesDemo
4_UQueryDemo
5_Pseudo-classesDemo
6_USS TransitionsDemo
7_EventsDemo
8_EventDispatchDemo
9_ManipulatorDemo
10_Custom controls
Templates
Quiz
Custom Controls (C#)
Quiz
CustomColorButton

# QuizU - Quiz


Library
Standard
Project
UI Documents (UXML)
Assets
Unity Technologies
QuizU - A UI Toolkit demo
Demos
Quiz
Data
Resources
Uxml
question-text.uxml
VisualTreeAssets
level-button.uxml
UI
Uxml


Library
Standard
Project
UI Documents (UXML)
Assets
Unity Technologies
QuizU - A UI Toolkit demo
Demos
Quiz
Data
UI
Uxml
BootloadScreen.uxml
EndScreen.uxml
GameScreen.uxml
LevelSelectionScreen.uxml
MainMenuScreen.uxml
PauseScreen.uxml
SettingsScreen.uxml
SplashScreen.uxml
StartScreen.uxml
UIScreens.uxml
VisualTreeAssets
Custom Controls (C#)
Quiz
CustomColorButton

# CustomControl → C# - USS transition

```
using UnityEngine;
using UnityEngine.UIElements;

namespace Quiz
{
    /// <summary>
    /// This button toggles between an active and inactive color, showing a simple custom control
    /// available in the UI Builder.
    /// </summary>
    3 references
    public class CustomColorButton : Button
    {
        // Properties to hold the active and inactive colors
        2 references
        Color activeColor = new Color(0f, 0.5f, 0.85f, 1f);
        3 references
        Color inactiveColor = new Color(0.5f, 0.5f, 0.5f, 1f);

        3 references
        bool m_IsActive = false;

        1 reference
        public new class UxmlFactory : UxmlFactory<CustomColorButton, UxmlTraits> { }

        // This exposes attributes in the UI Builder
        1 reference
        public new class UxmlTraits : Button.UxmlTraits
        {
            1 reference
            UxmlColorAttributeDescription m_AttrActiveColor =
                new UxmlColorAttributeDescription
                {
                    name = "active-color",
                    defaultValue = new Color(0f, 0.5f, 0.85f, 1f)
                };

            1 reference
            UxmlColorAttributeDescription m_AttrInactiveColor =
                new UxmlColorAttributeDescription
                {
                    name = "inactive-color",
                    defaultValue = new Color(0.5f, 0.5f, 0.5f, 1f)
                };
```

# CustomControl → C# - USS transition

```
0 references
public CustomColorButton()
{
    this.style.minWidth = 100;
    this.style.minHeight = 100;

    //// Optional: this shows the syntax for setting up a basic USS transition using the C# syntax
    //// More information: https://docs.unity3d.com/Manual/UIE-Transitions.html

    // Omitted for simplicitiy. Uncomment this block to add some "built-in" USS Transitions:


    // TransitionDuration needs a List of TimeValues

    List<TimeValue> durations = new List<TimeValue>();
    durations.Add(new TimeValue(0.5f, TimeUnit.Second));
    this.style.transitionDuration = new StyleList<TimeValue>(durations);

    // easingFunctions also needs a List

    List<EasingFunction> easingFunctions = new List<EasingFunction>();
    easingFunctions.Add(new EasingFunction(EasingMode.EaseInOut));

    this.style.transitionTimingFunction = new StyleList<EasingFunction>(easingFunctions);


    // Use the Clickable Manipulator to toggle between active and inactive color.
    // The clicked property is a shorthand way to subscribe to click events (versus RegisterCallback).
    this.clicked += OnClick;

}

// Note: CustomColorButton instance is tightly bound to its OnClick handler; when the button is destroyed, the OnClick handler will be
// be garbage collected alongside the button.
1 reference
private void OnClick()
{
    m_IsActive = !m_IsActive;
    this.style.backgroundColor = m_IsActive ? activeColor : inactiveColor;
}
}
}
```

# Manipulator → Dragger

```
21      public SimpleDragManipulator()
22      {
23          // This filter determines what condition starts the Manipulator (Left Mouse Button down)
24          activators.Add(new ManipulatorActivationFilter { button = MouseButton.LeftMouse });
25      }
26
27      // Event subscriptions
        0 references
28      protected override void RegisterCallbacksOnTarget()
29      {
30          target.RegisterCallback<MouseDownEvent>(OnMouseDown);
31          target.RegisterCallback<MouseMoveEvent>(OnMouseMove);
32          target.RegisterCallback<MouseUpEvent>(OnMouseUp);
33      }
34
35      // Event unsubscriptions
        0 references
36      protected override void UnregisterCallbacksFromTarget()
37      {
38          target.UnregisterCallback<MouseDownEvent>(OnMouseDown);
39          target.UnregisterCallback<MouseMoveEvent>(OnMouseMove);
40          target.UnregisterCallback<MouseUpEvent>(OnMouseUp);
41      }
42
43      // Check if we can click and drag an element and then initialize some values
        2 references
44      private void OnMouseDown(MouseDownEvent evt)
45      {
46          // Checks whether MouseEvent satisfies all of the ManipulatorActivationFilter requirements.
47          if (CanStartManipulation(evt))
48          {
49              // Save the mouse and element start positions
50              m_StartMousePosition = evt.mousePosition;
51              m_StartElementPosition = new Vector2(target.layout.x, target.layout.y);
52
53              // Flag that this is active and receive all mouse events, even if the mouse pointer leaves
54              m_IsDragging = true;
55              target.CaptureMouse();
56
57              // Limit the event to this element, do not send up or down the hierarchy
58              evt.StopPropagation();
59          }
60      }
```

# Manipulator → Dragger

```
61
62      // Offset the dragged element based on the difference from the start position
        2 references
63      private void OnMouseMove(MouseMoveEvent evt)
64      {
65          // Checks whether the MouseEvent is related to this Manipulator and dragging is active
66          if (CanStopManipulation(evt) && m_IsDragging)
67          {
68              // Use the difference in mouse position to offset the element as well
69              Vector3 mouseDelta = evt.mousePosition - m_StartMousePosition;
70
71              // Convert the pixel offset into a new left/top StyleLength
72              target.style.left = CreatePixelLength(m_StartElementPosition.x + mouseDelta.x);
73              target.style.top = CreatePixelLength(m_StartElementPosition.y + mouseDelta.y);
74
75              // Limit the event to this element, do not send up or down the hierarchy
76              evt.StopPropagation();
77          }
78      }
79
80      // Creates a new StyleLength representing a length in pixels.
        2 references
81      private StyleLength CreatePixelLength(float value)
82      {
83          return new StyleLength(new Length(value, LengthUnit.Pixel));
84      }
```

# Campus Virtual → Documentation

## QuizU documentation

Documentation for the QuizU - an official Unity sample demonstrating MVP, state pattern, managing menu screens and much more using UI Toolkit available on the Unity Asset Store.

Version 1.01 Nov 1st 2023

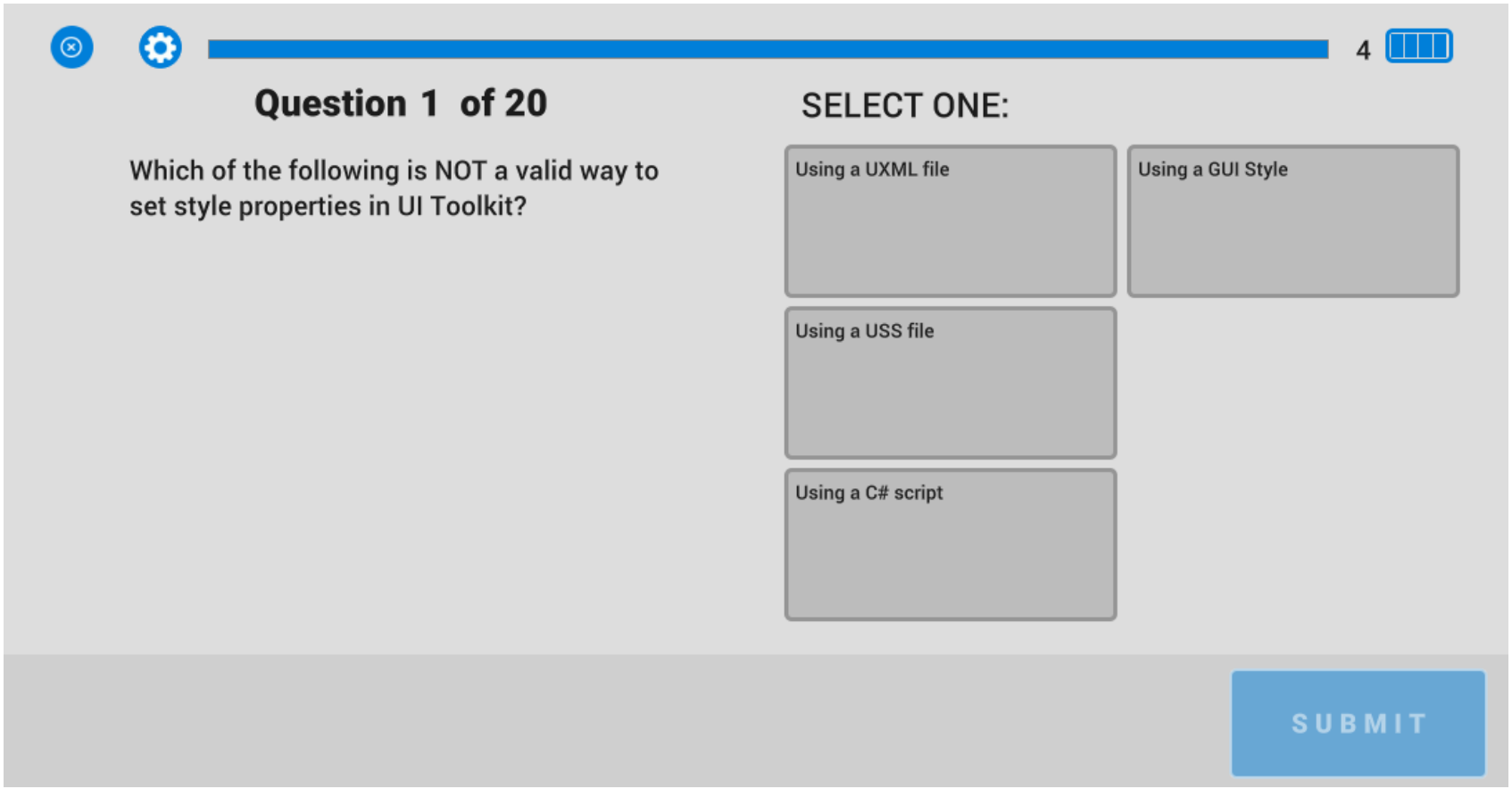
### Table of contents

# Índice

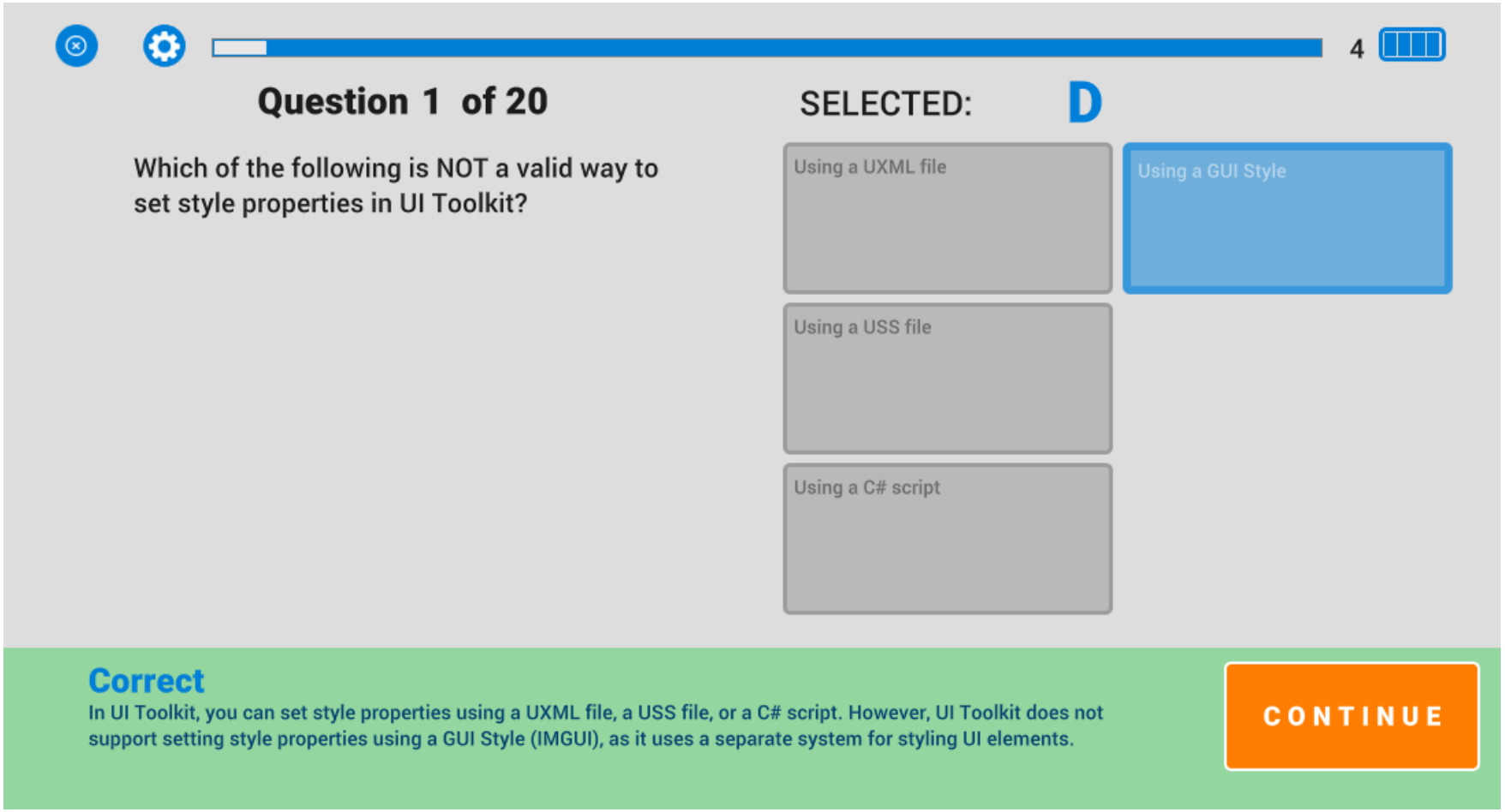
- Project Samples
  - Dragon Crashers
    - Custom Controls
      - Tabbed menú
      - Radial progress bar
  - Quizu
    - Demos
      - CustomControls
      - Manipulators
    - **Quiz**
      - **Data**
      - **Scripts**
        - **Coroutines**
        - **ScriptablesObjects**
        - **Actions**

- Proyecto

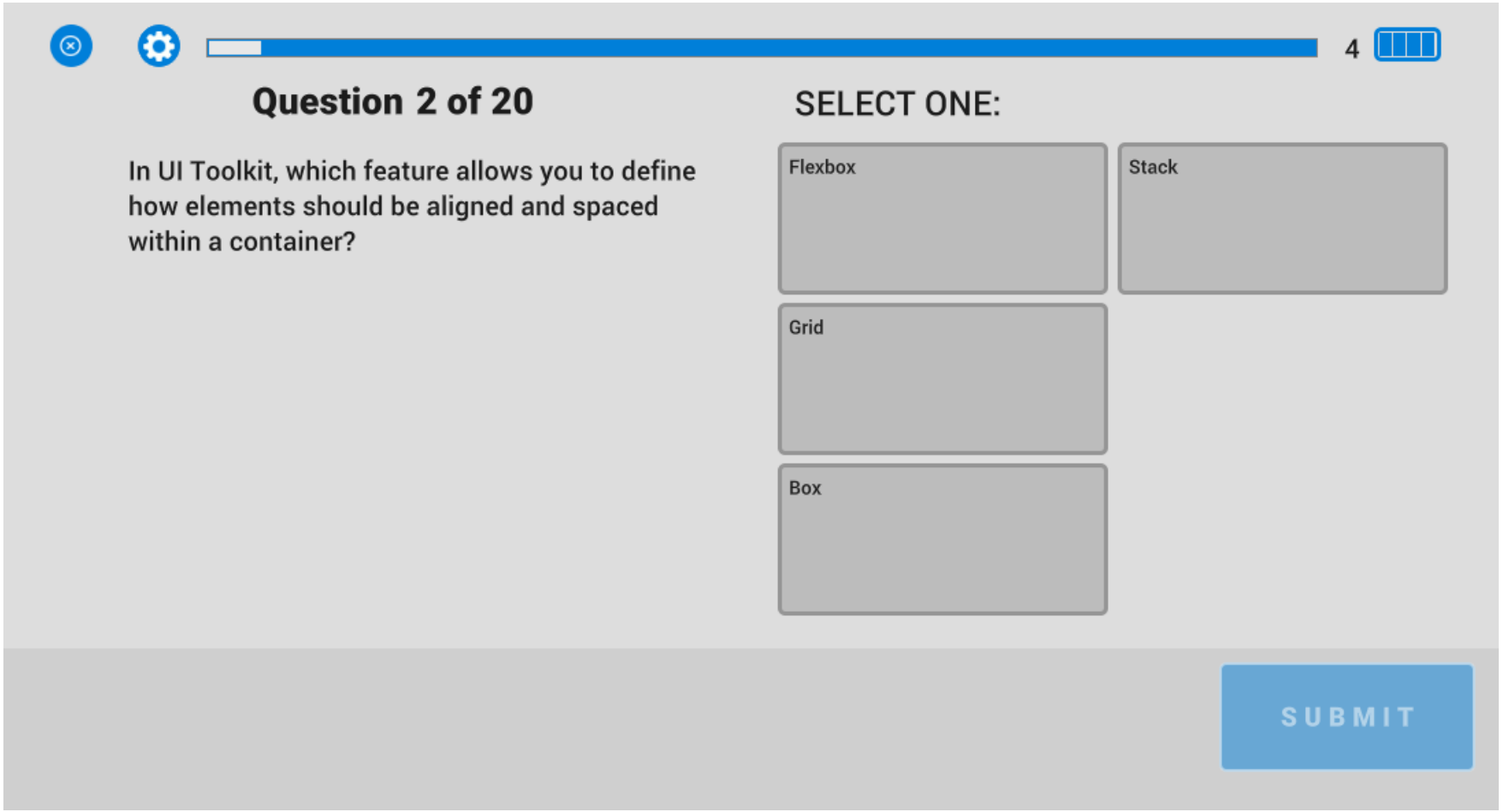
# Quiz → UI Toolkit



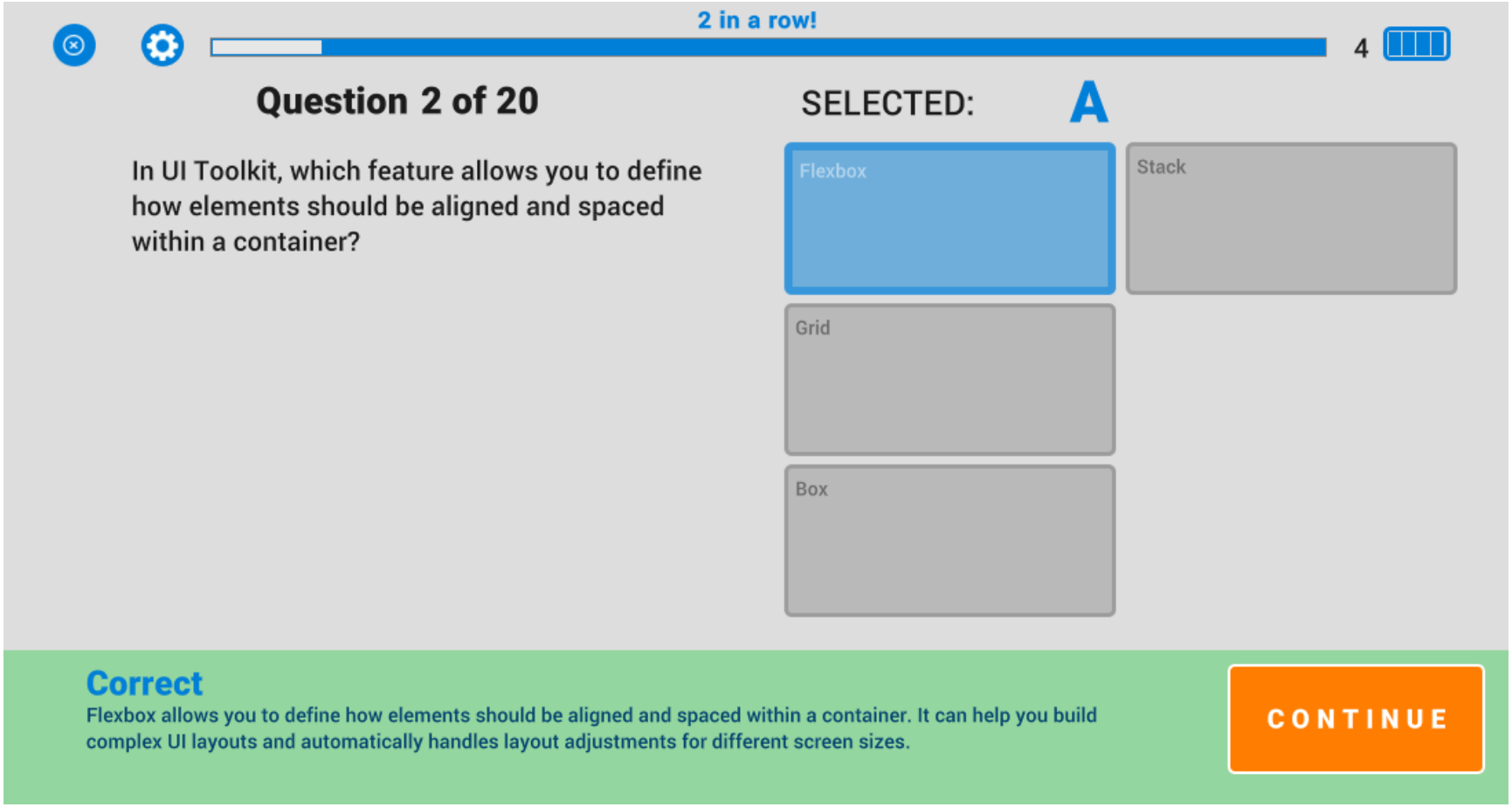
# Quiz → UI Toolkit



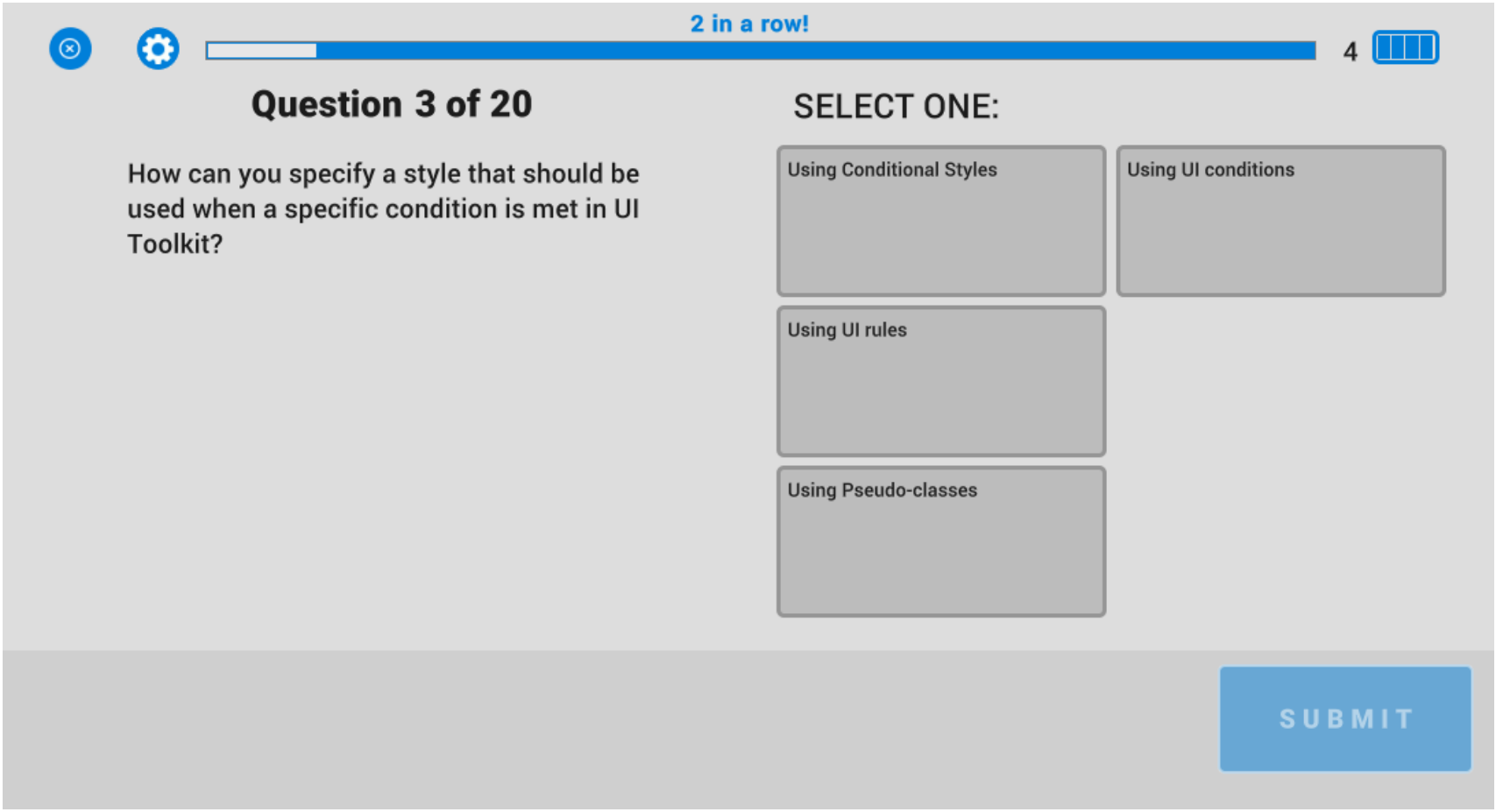
# Quiz → UI Toolkit



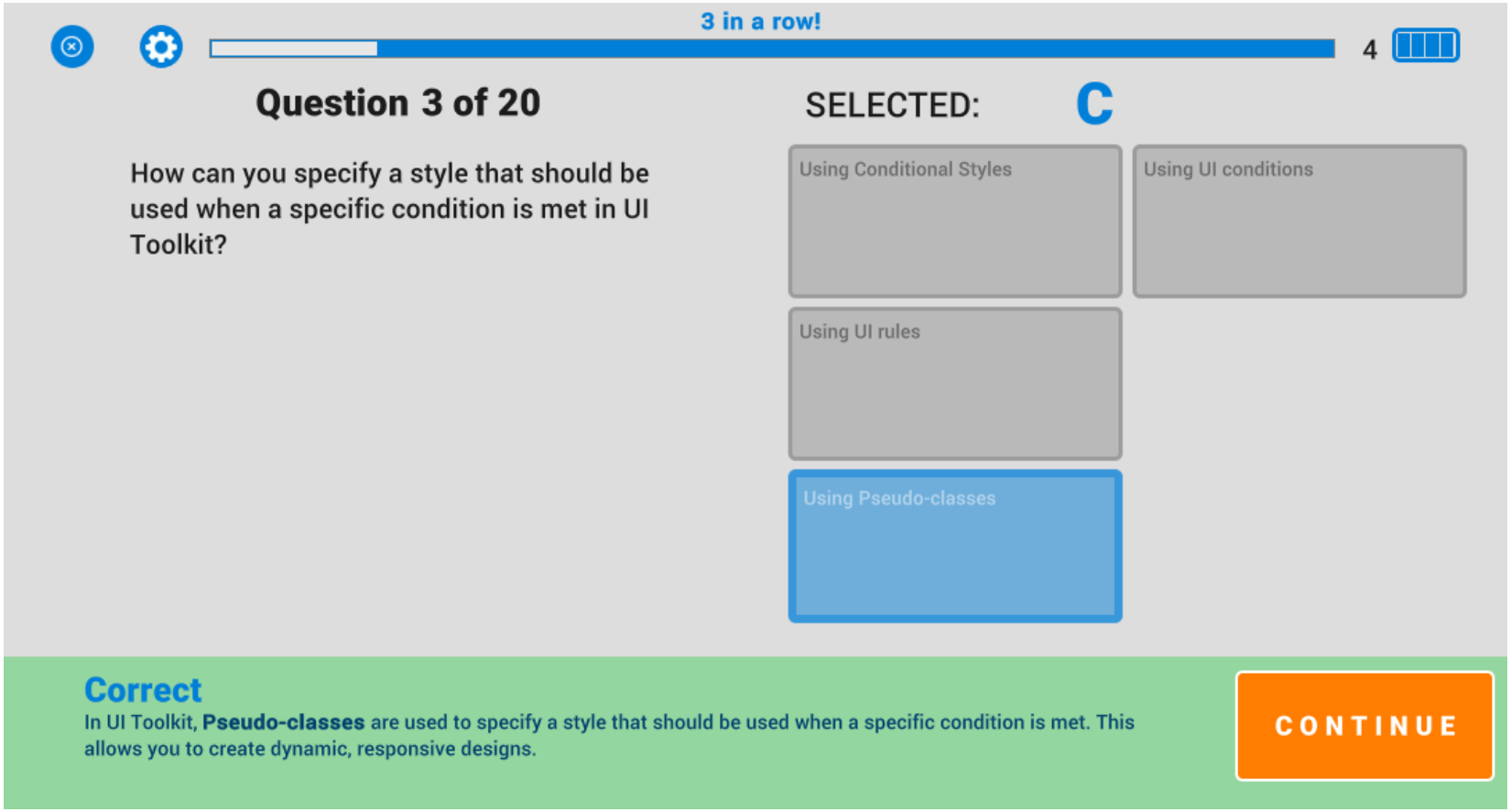
# Quiz → UI Toolkit



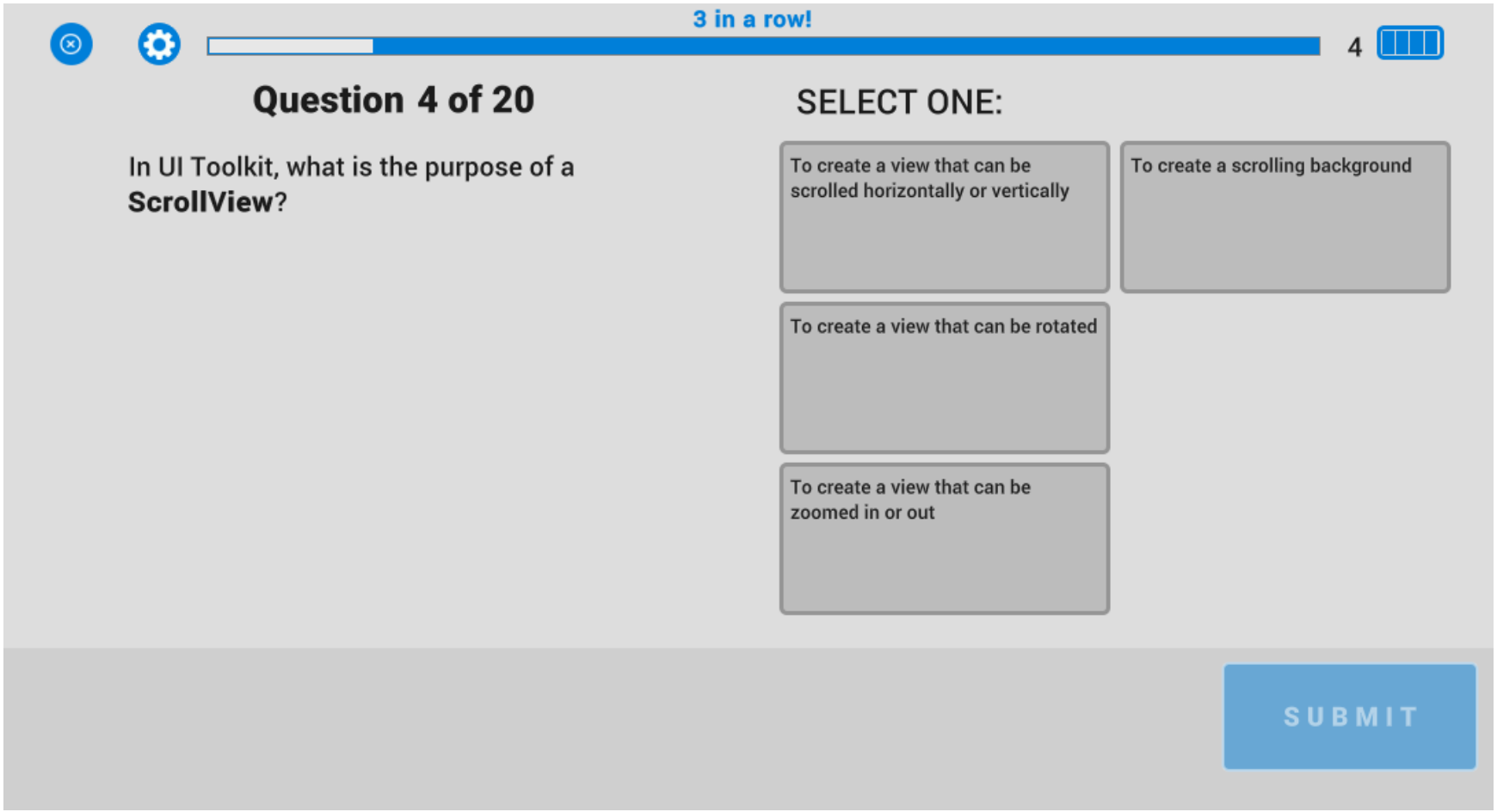
# Quiz → UI Toolkit



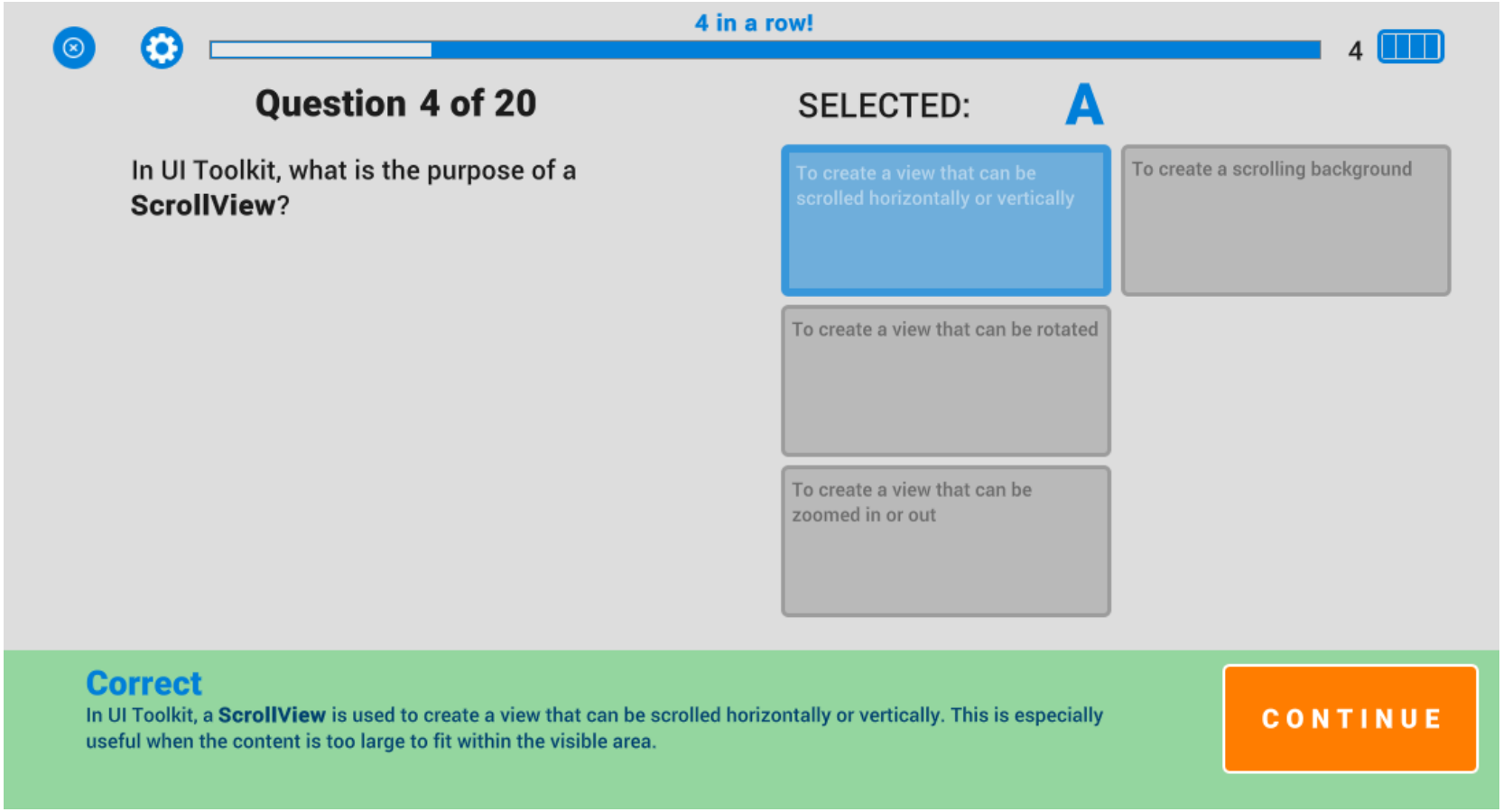
# Quiz → UI Toolkit



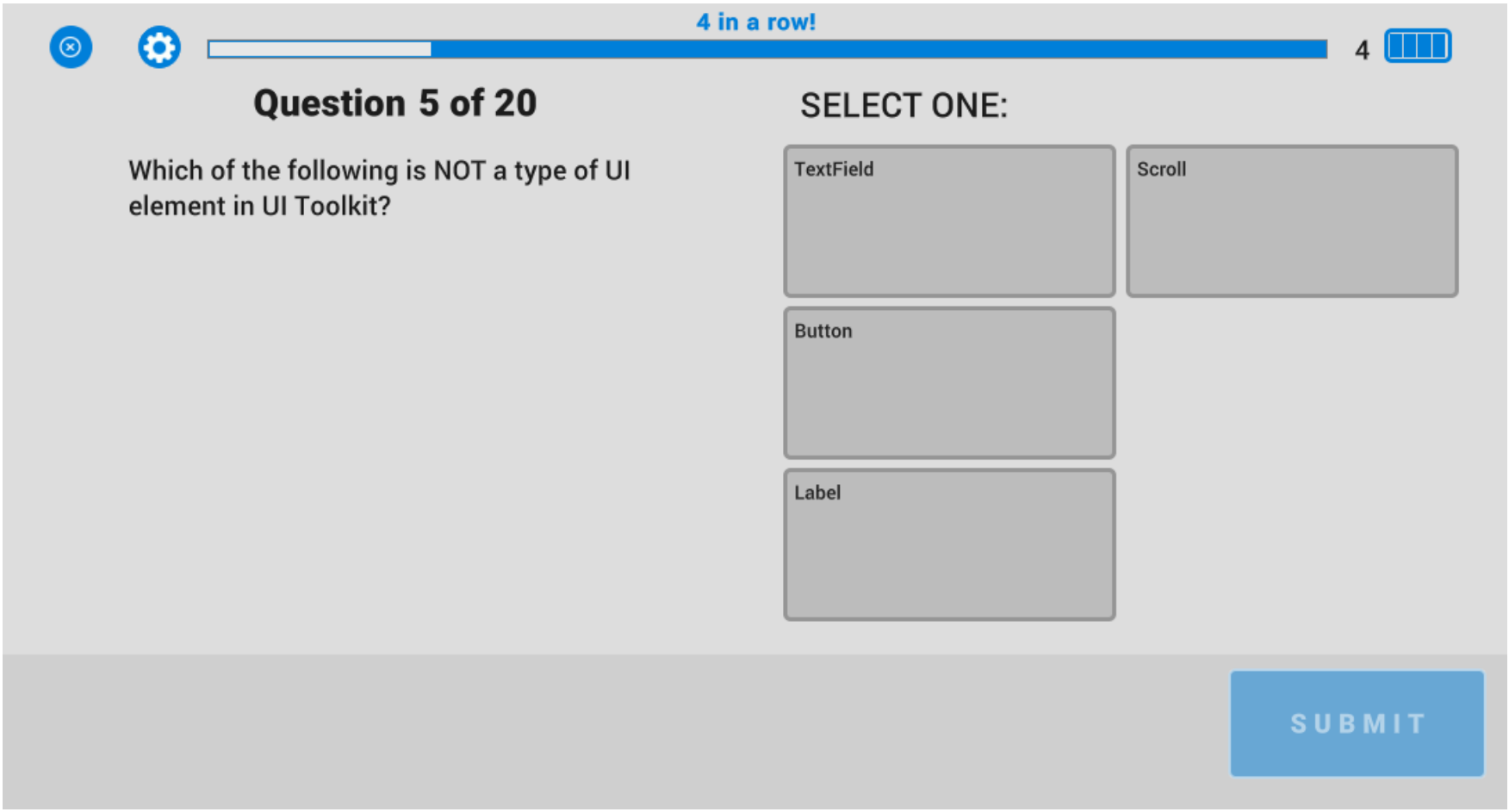
# Quiz → UI Toolkit



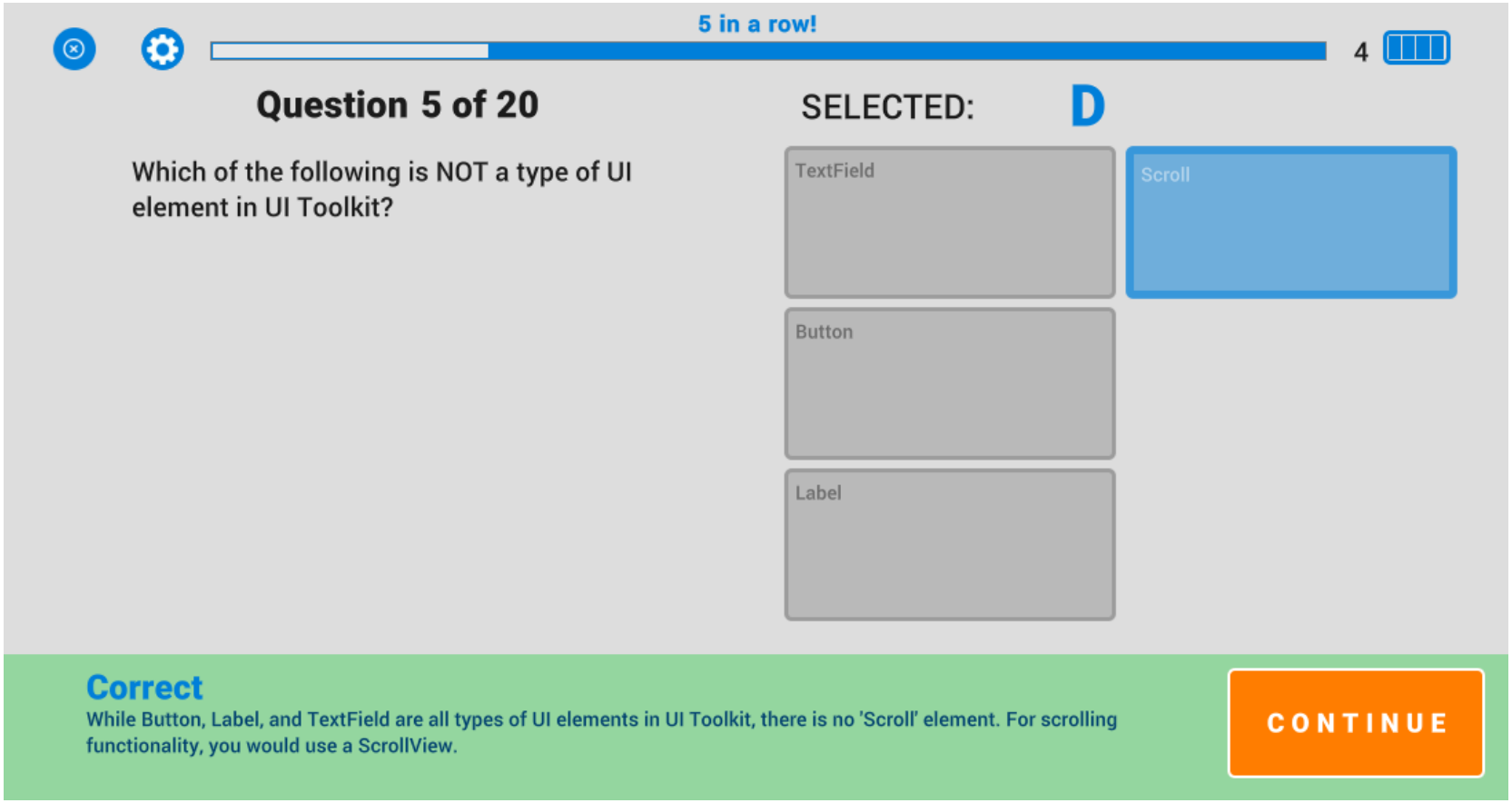
# Quiz → UI Toolkit



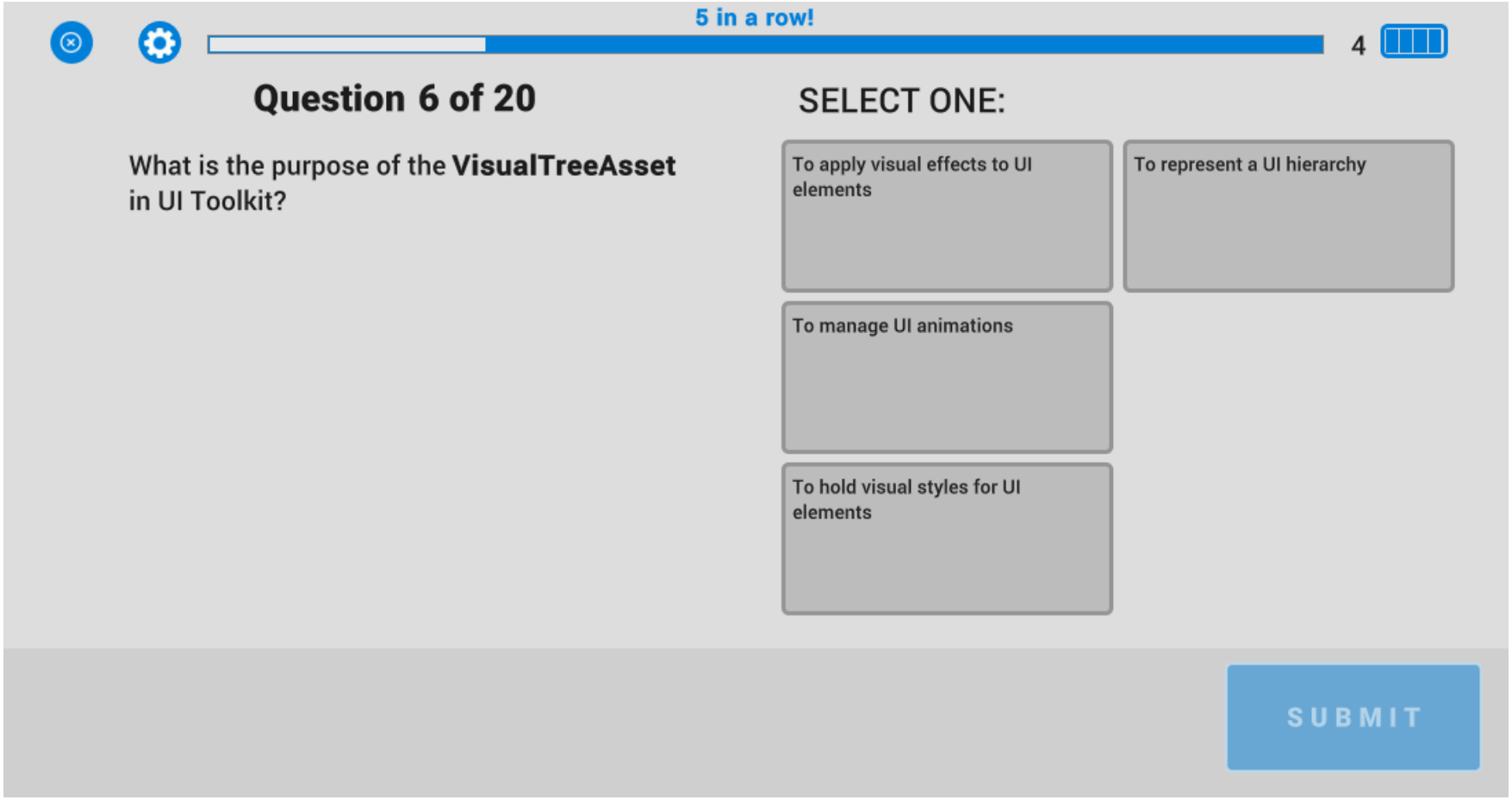
# Quiz → UI Toolkit



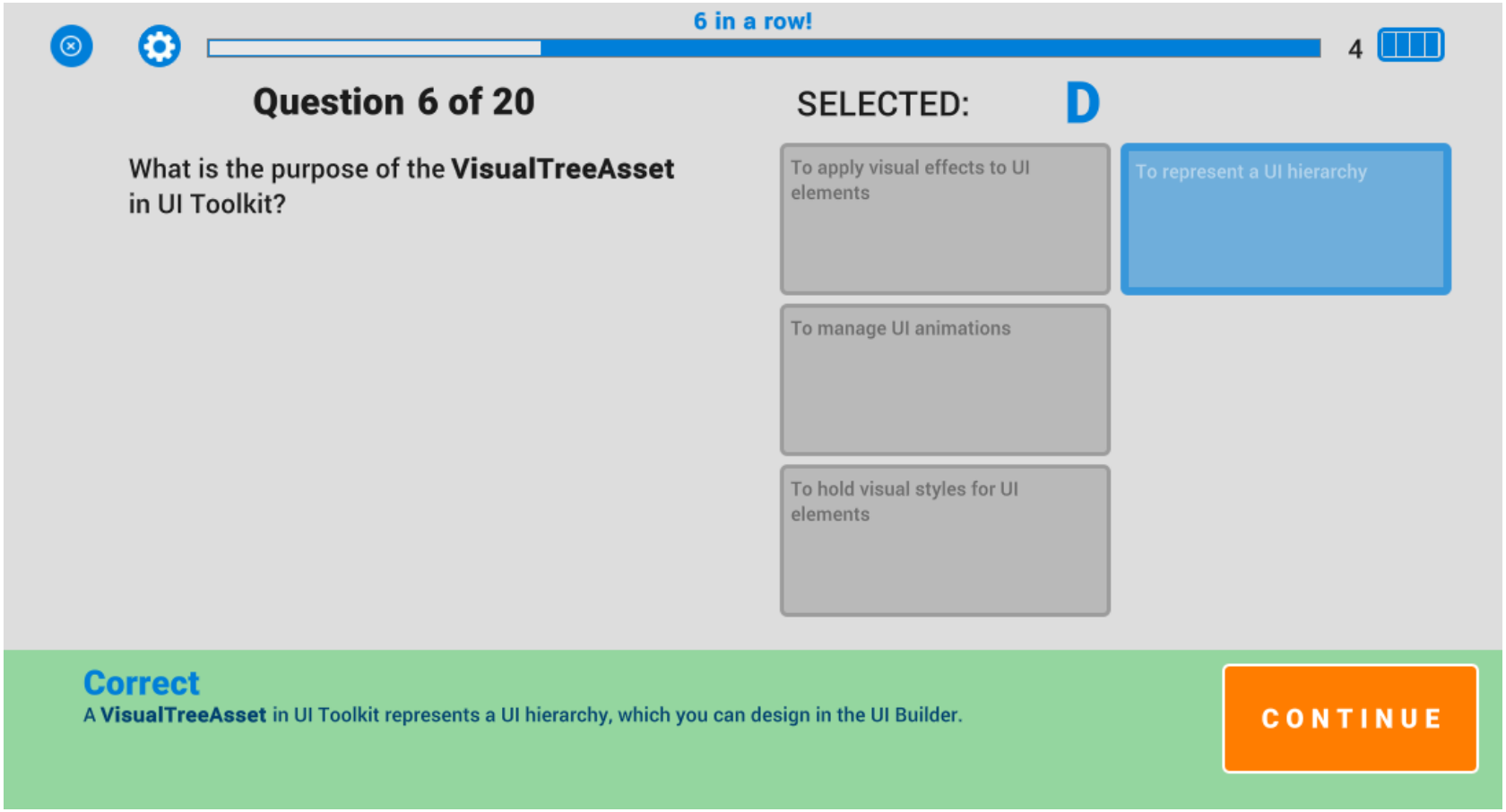
# Quiz → UI Toolkit



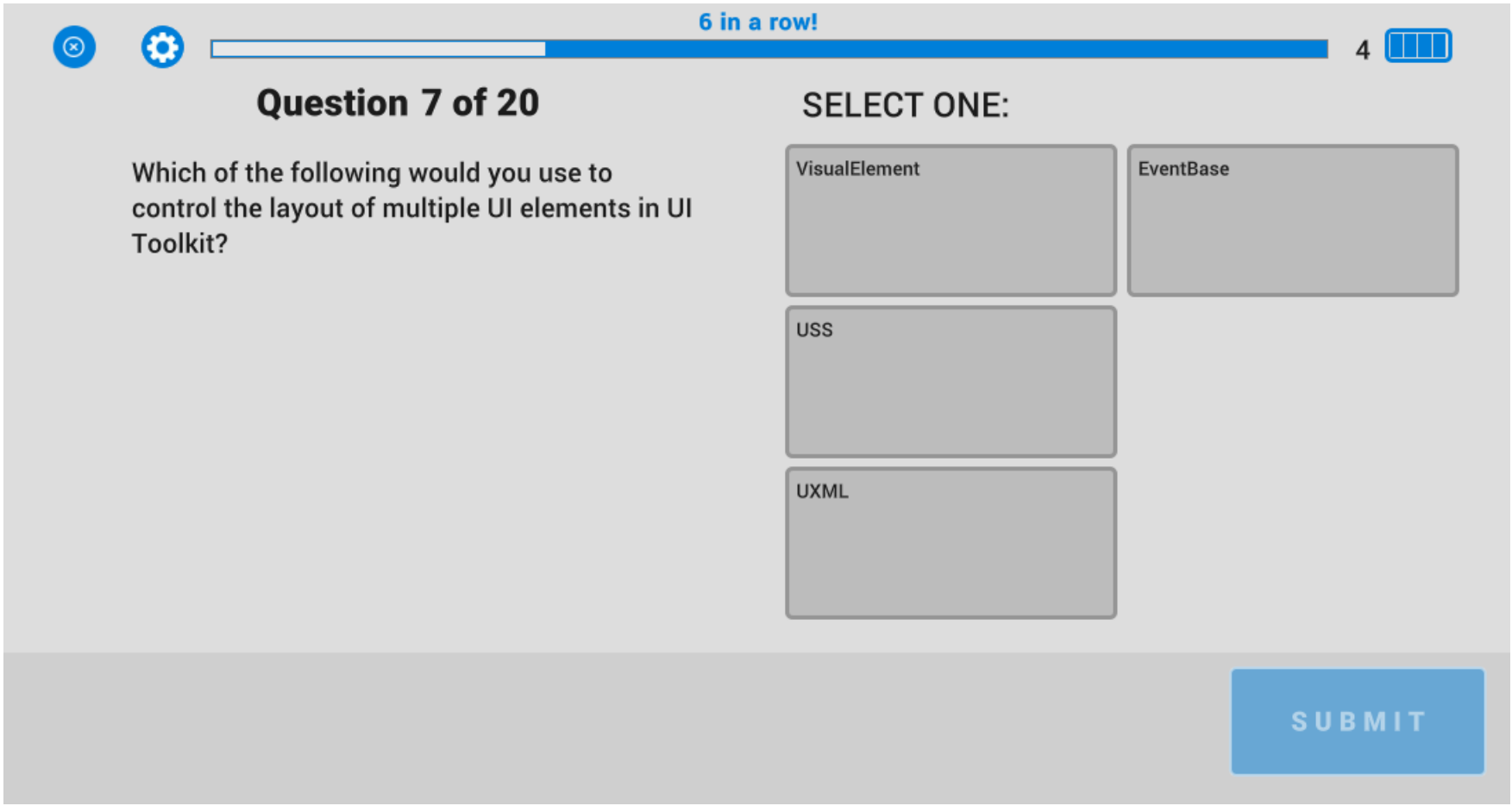
# Quiz → UI Toolkit



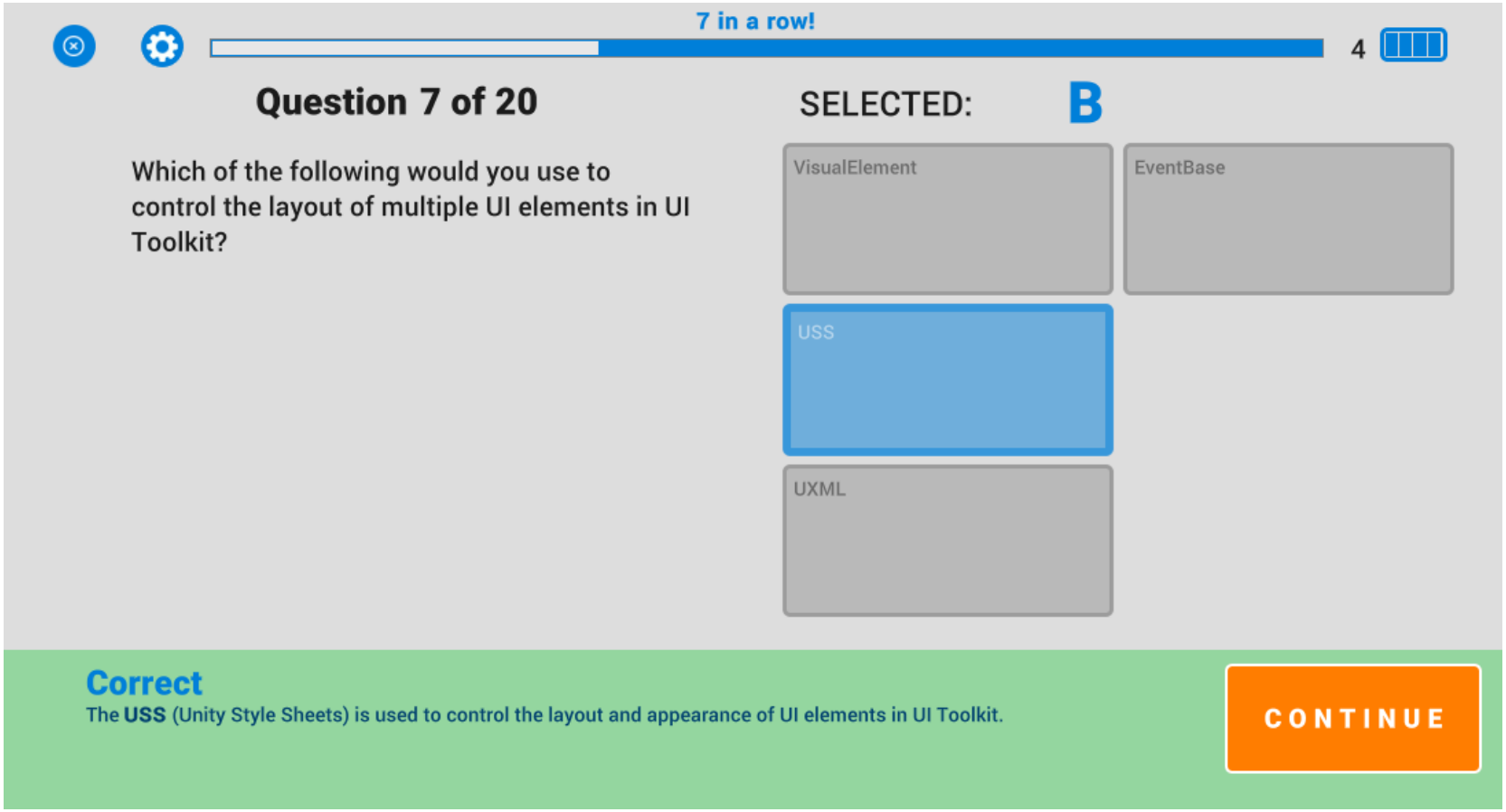
# Quiz → UI Toolkit



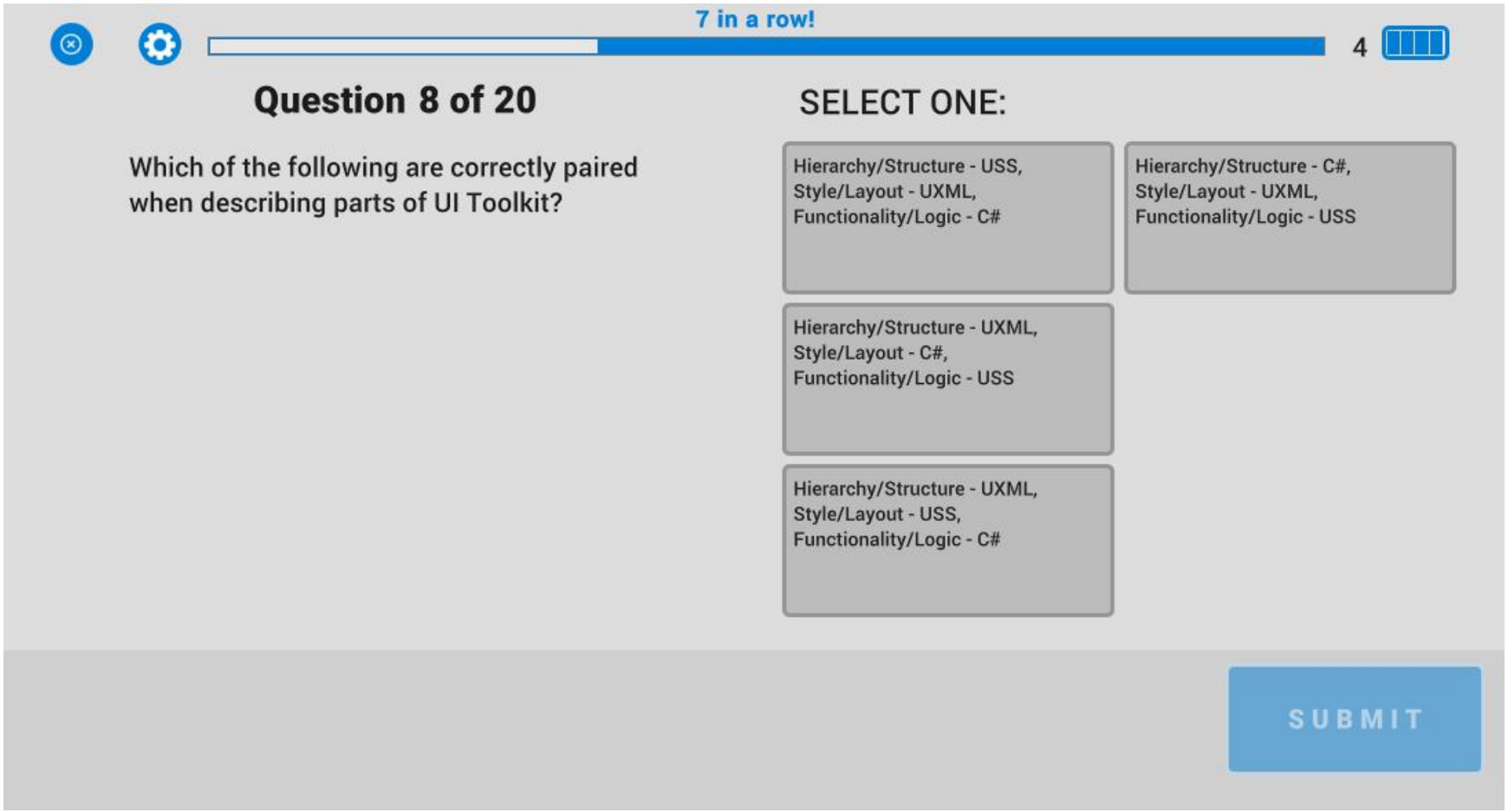
# Quiz → UI Toolkit



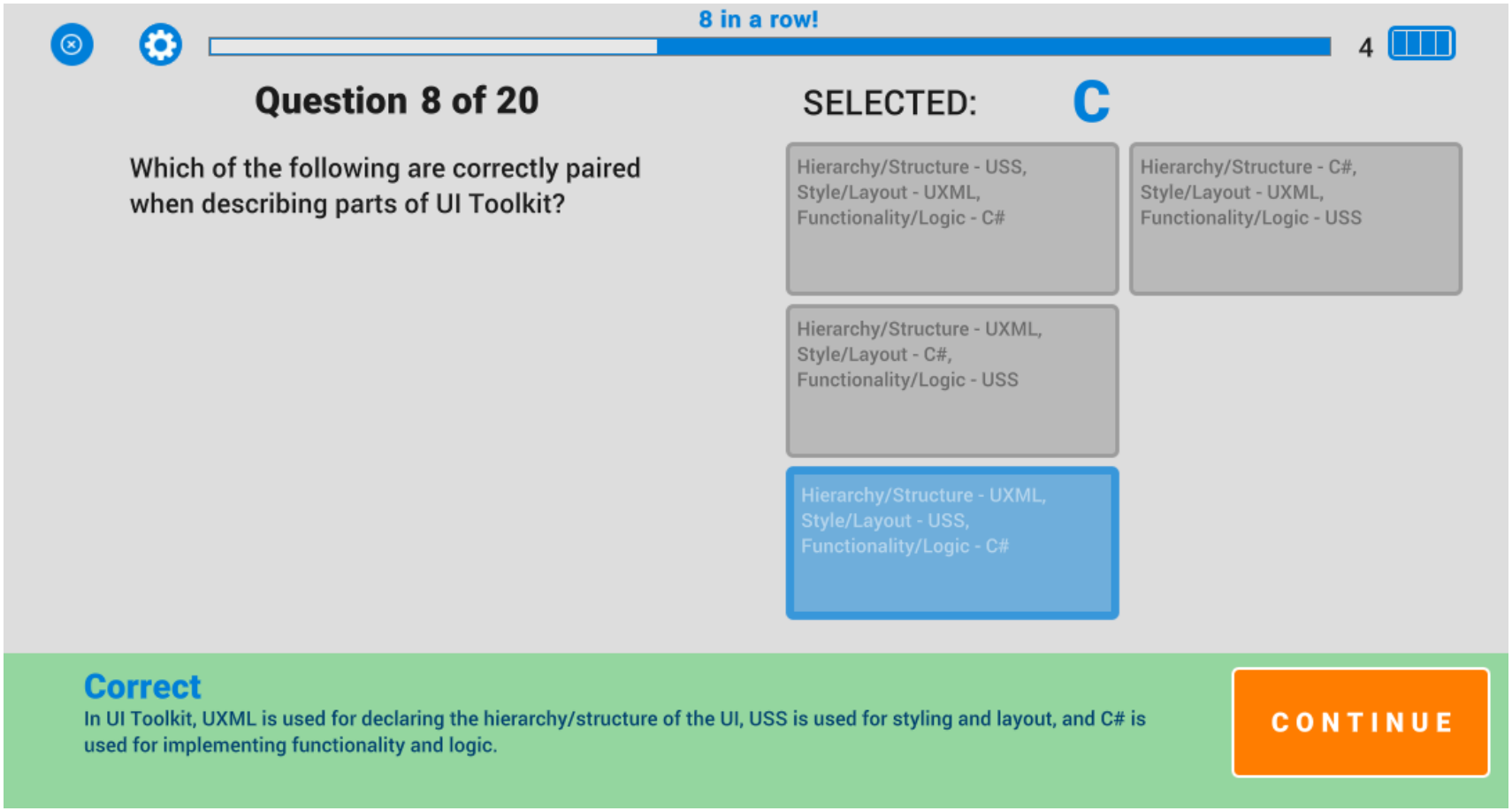
# Quiz → UI Toolkit



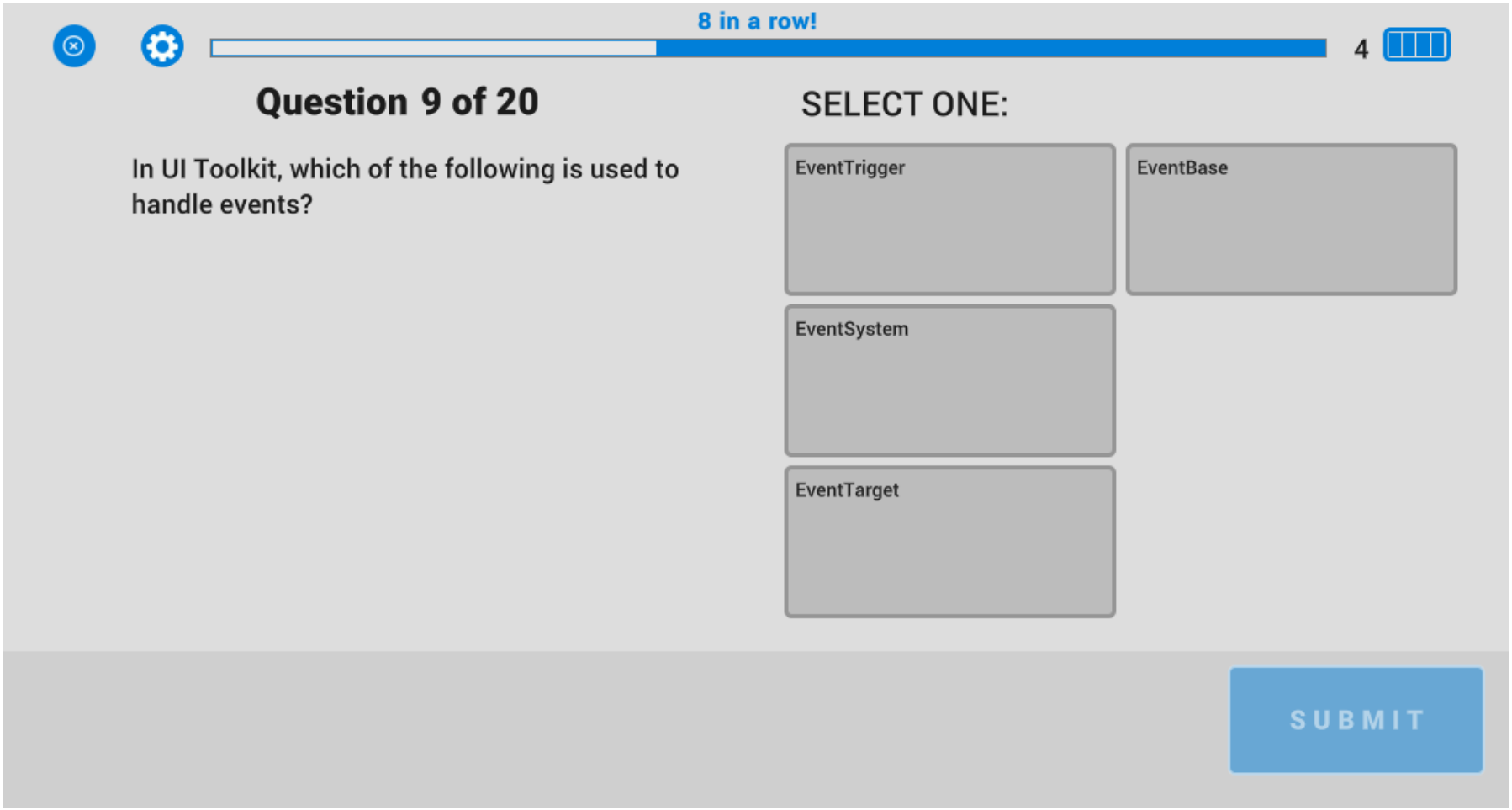
# Quiz → UI Toolkit



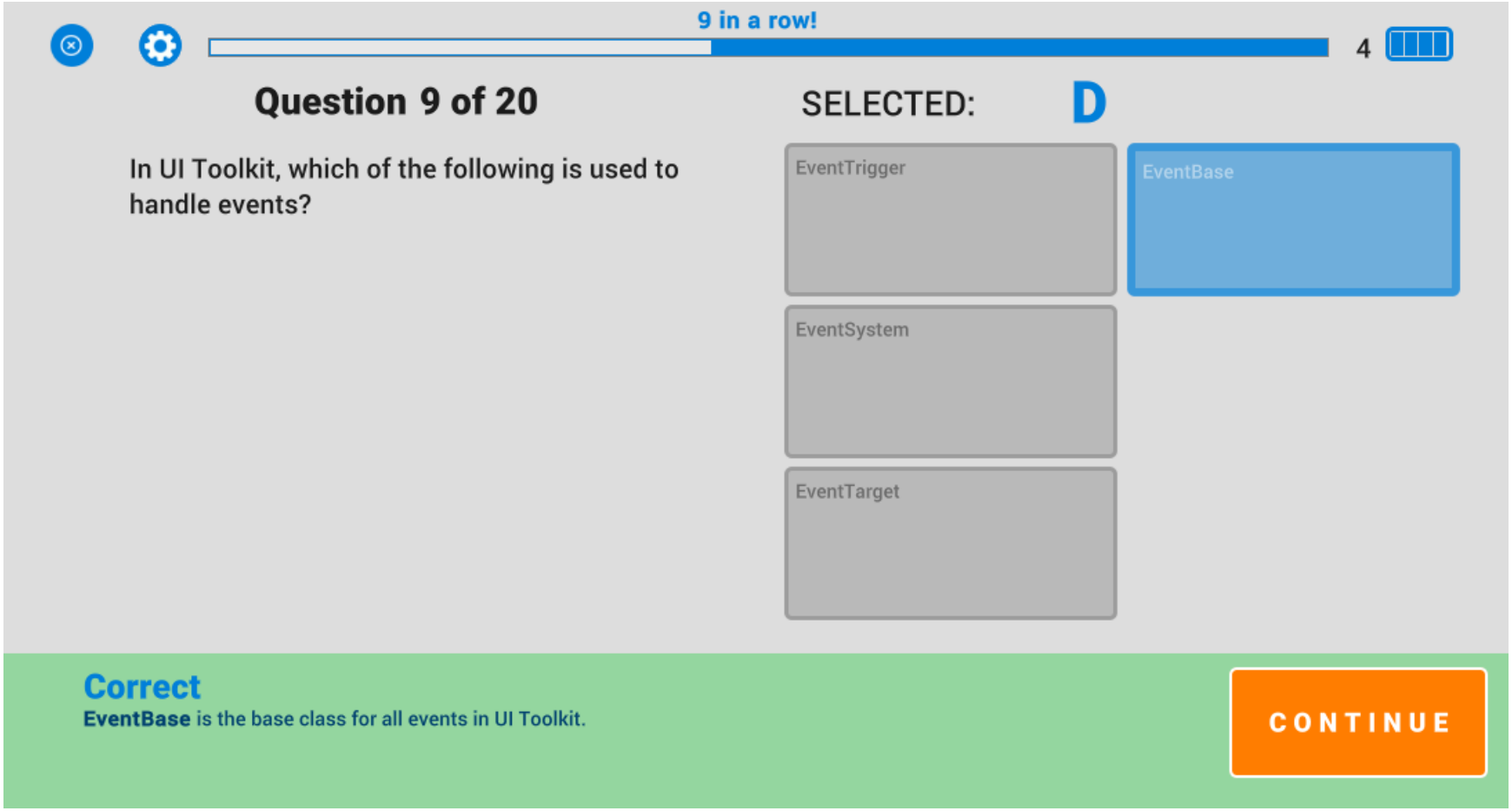
# Quiz → UI Toolkit



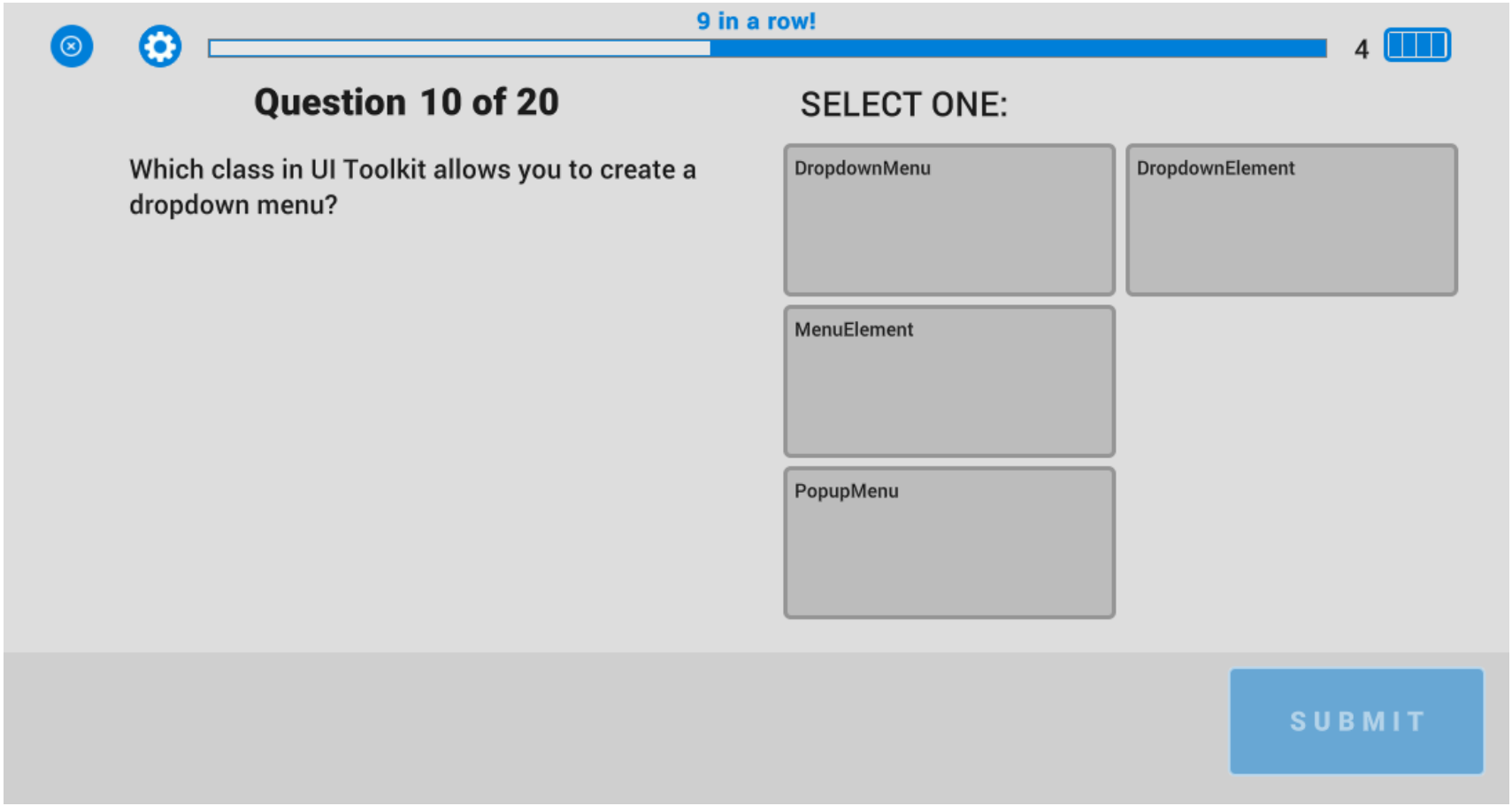
# Quiz → UI Toolkit



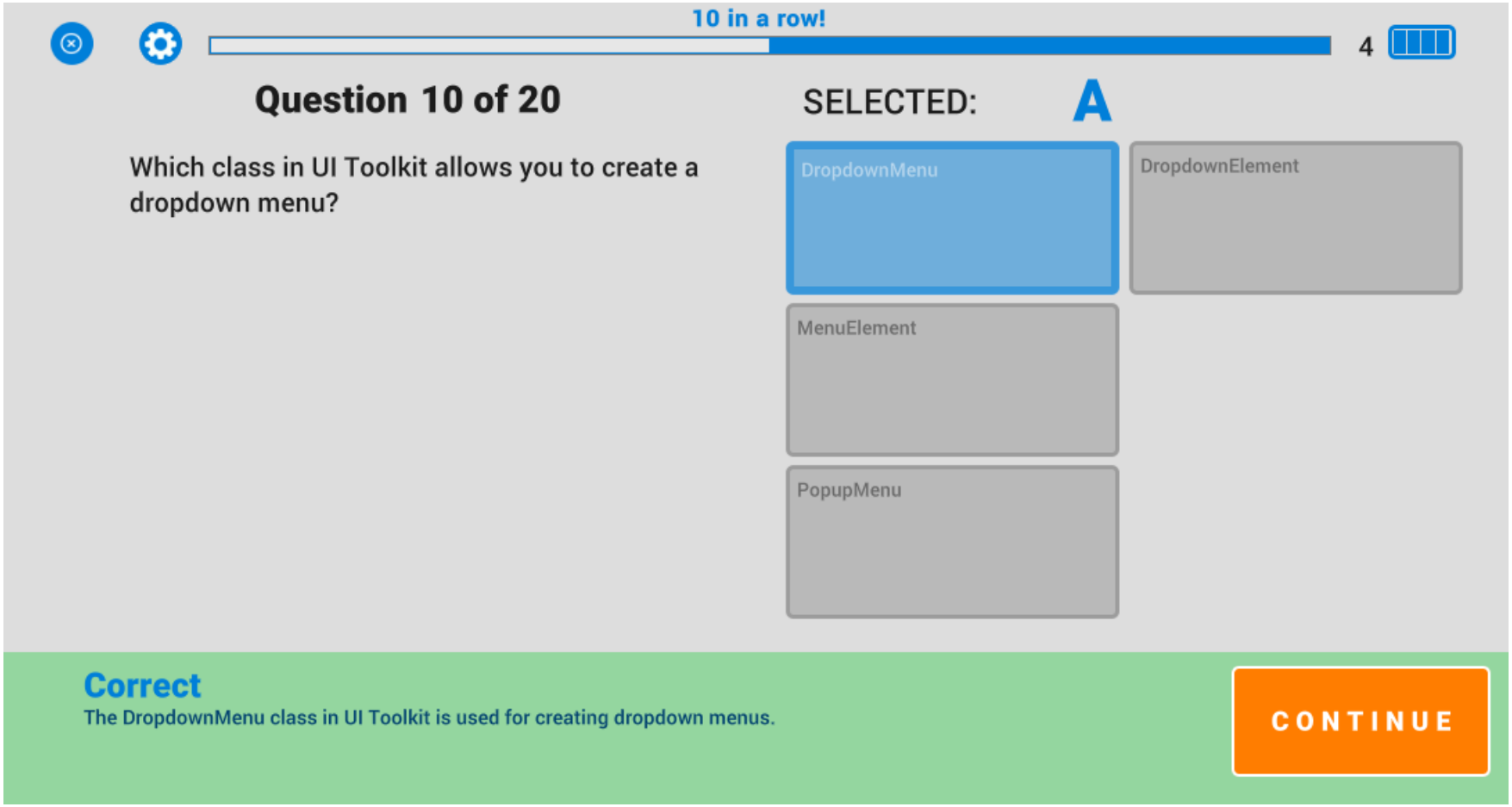
# Quiz → UI Toolkit



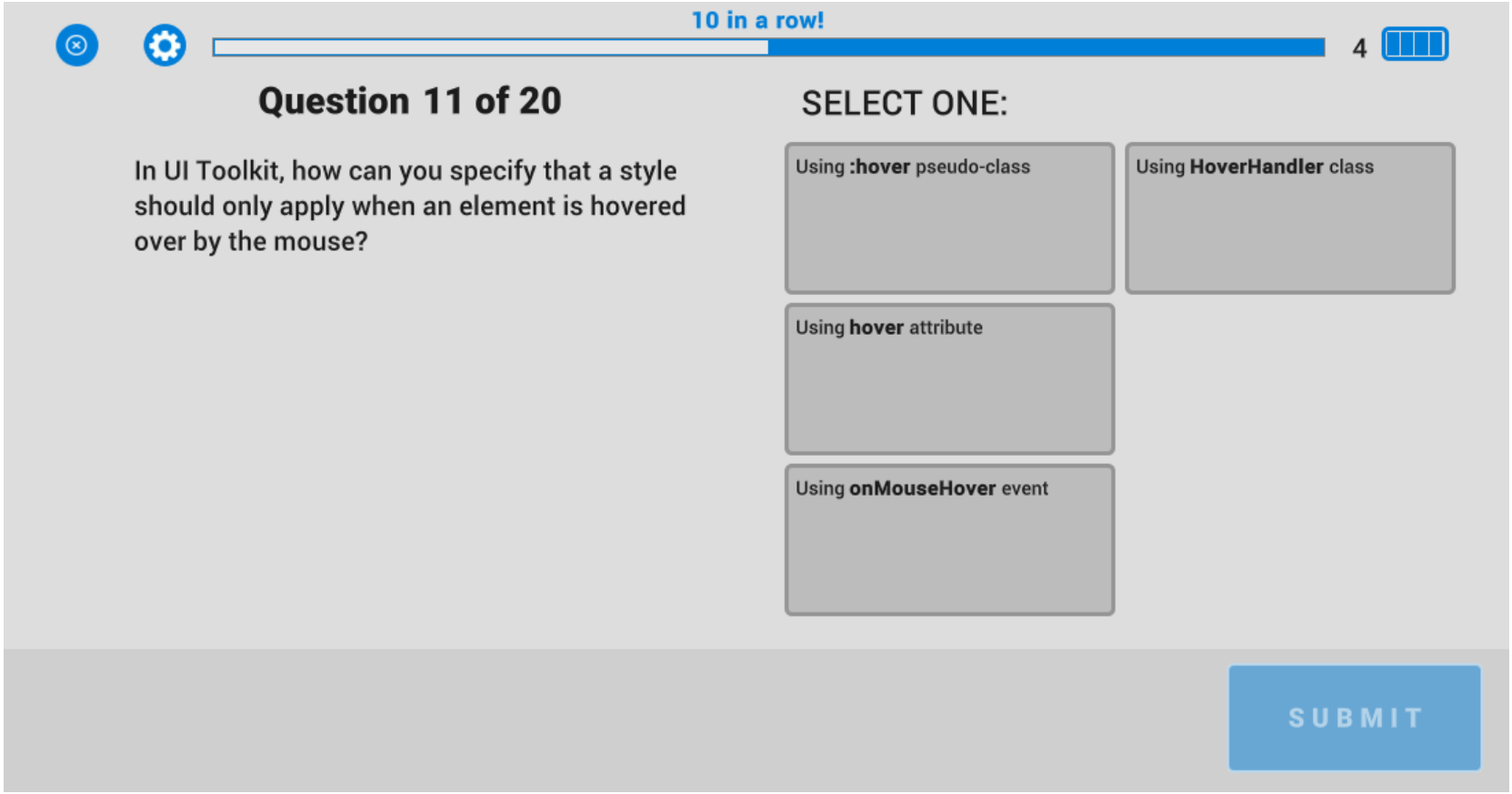
# Quiz → UI Toolkit



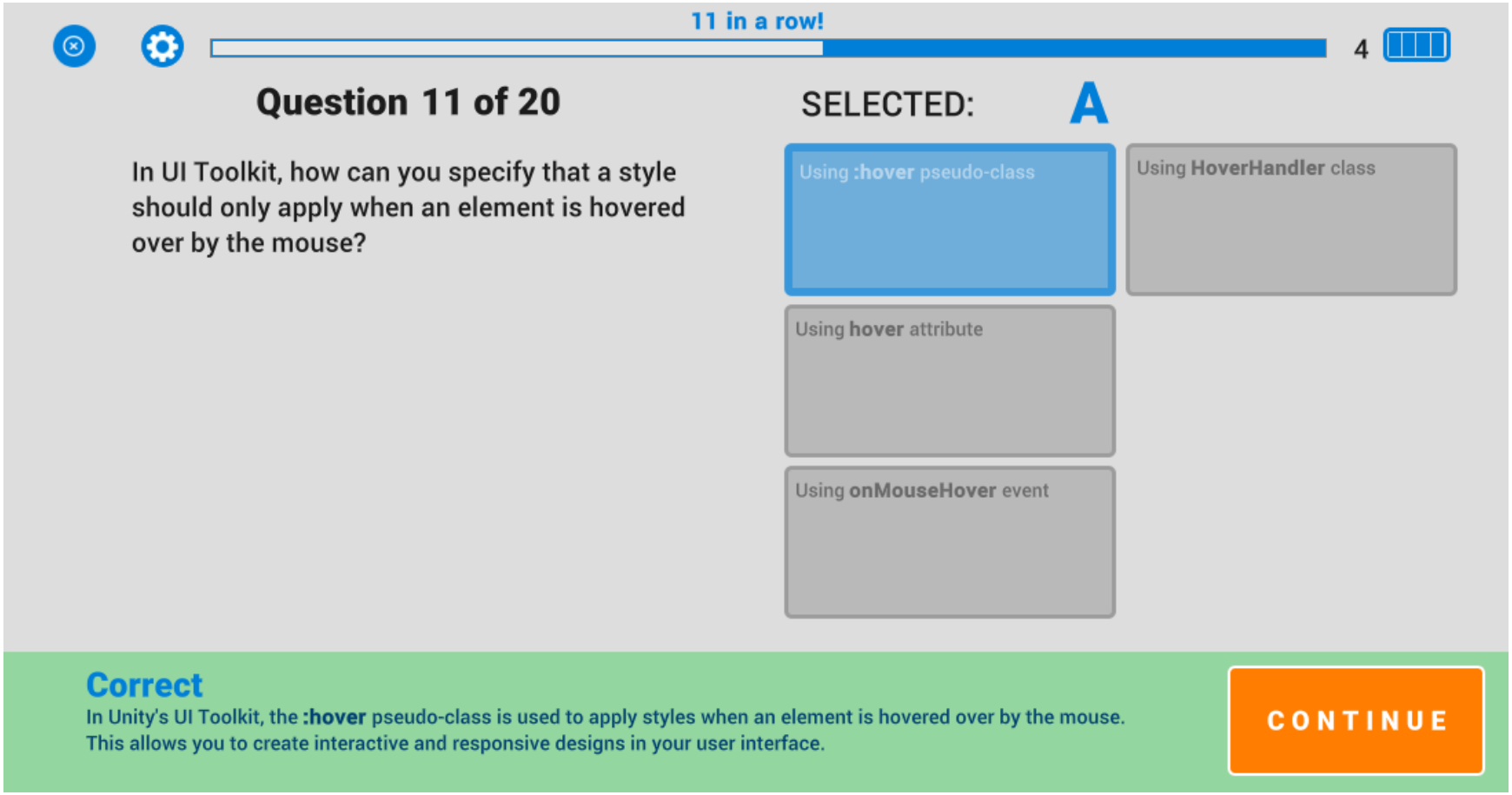
# Quiz → UI Toolkit



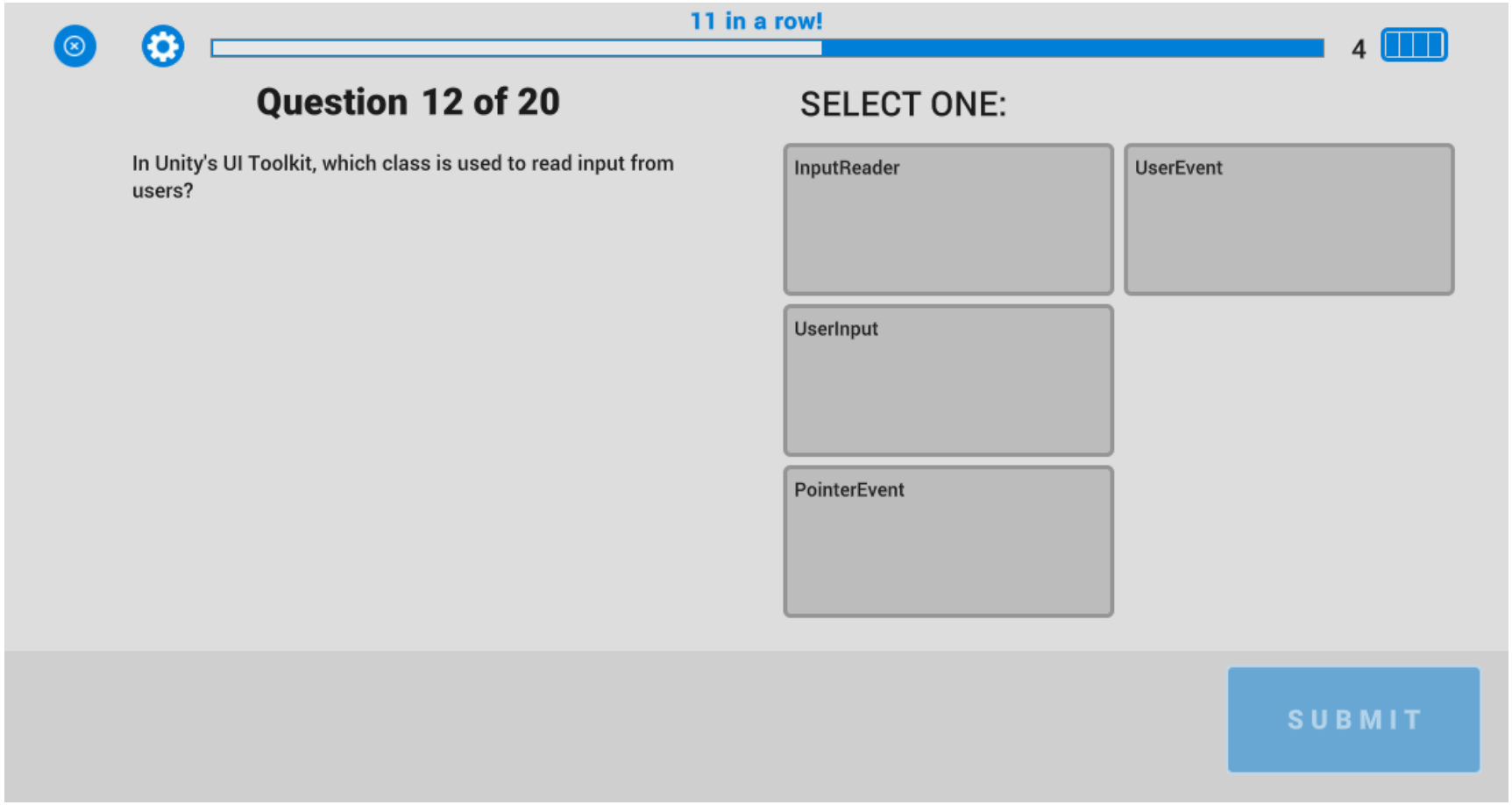
# Quiz → UI Toolkit



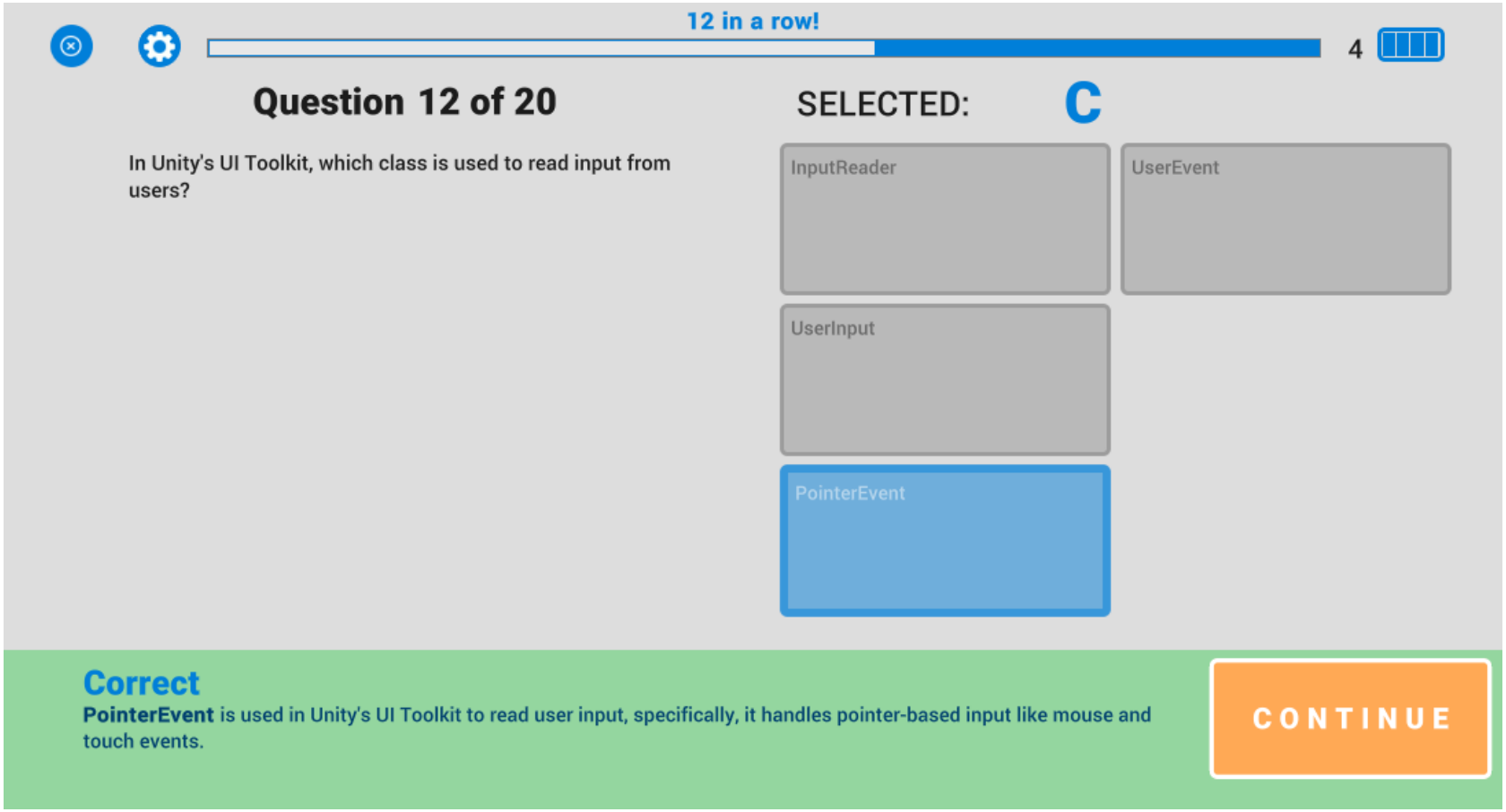
# Quiz → UI Toolkit



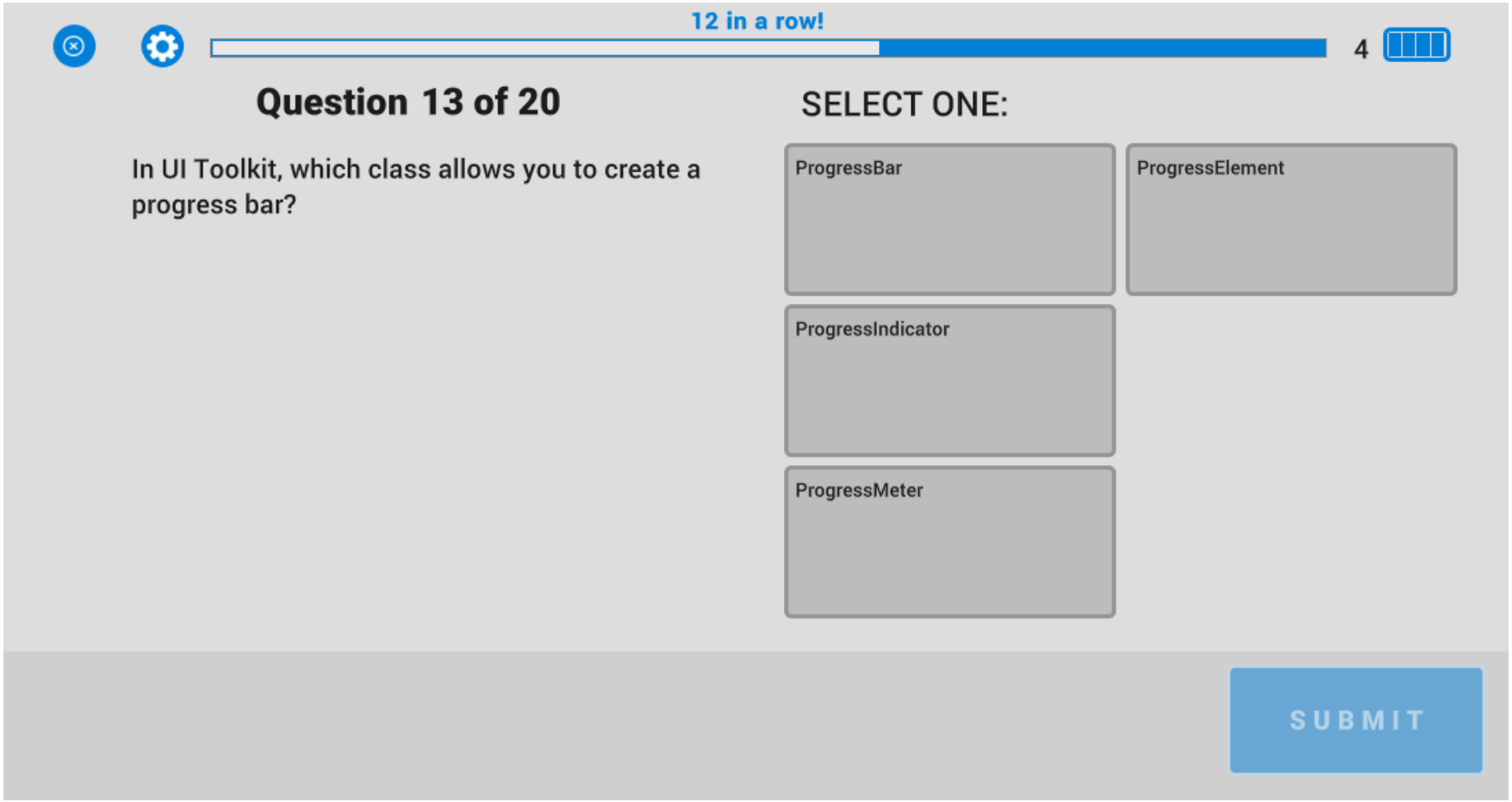
# Quiz → UI Toolkit



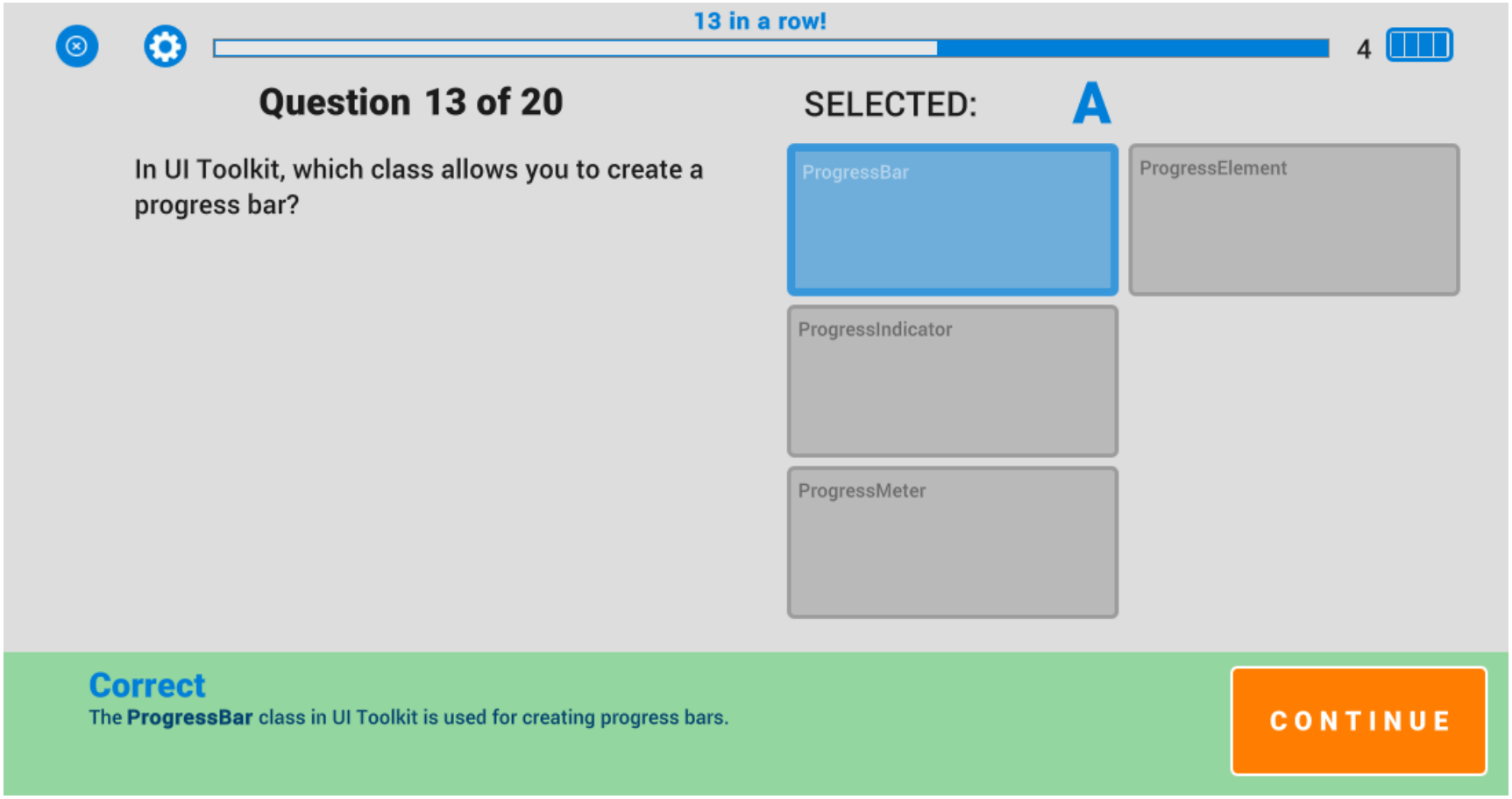
# Quiz → UI Toolkit



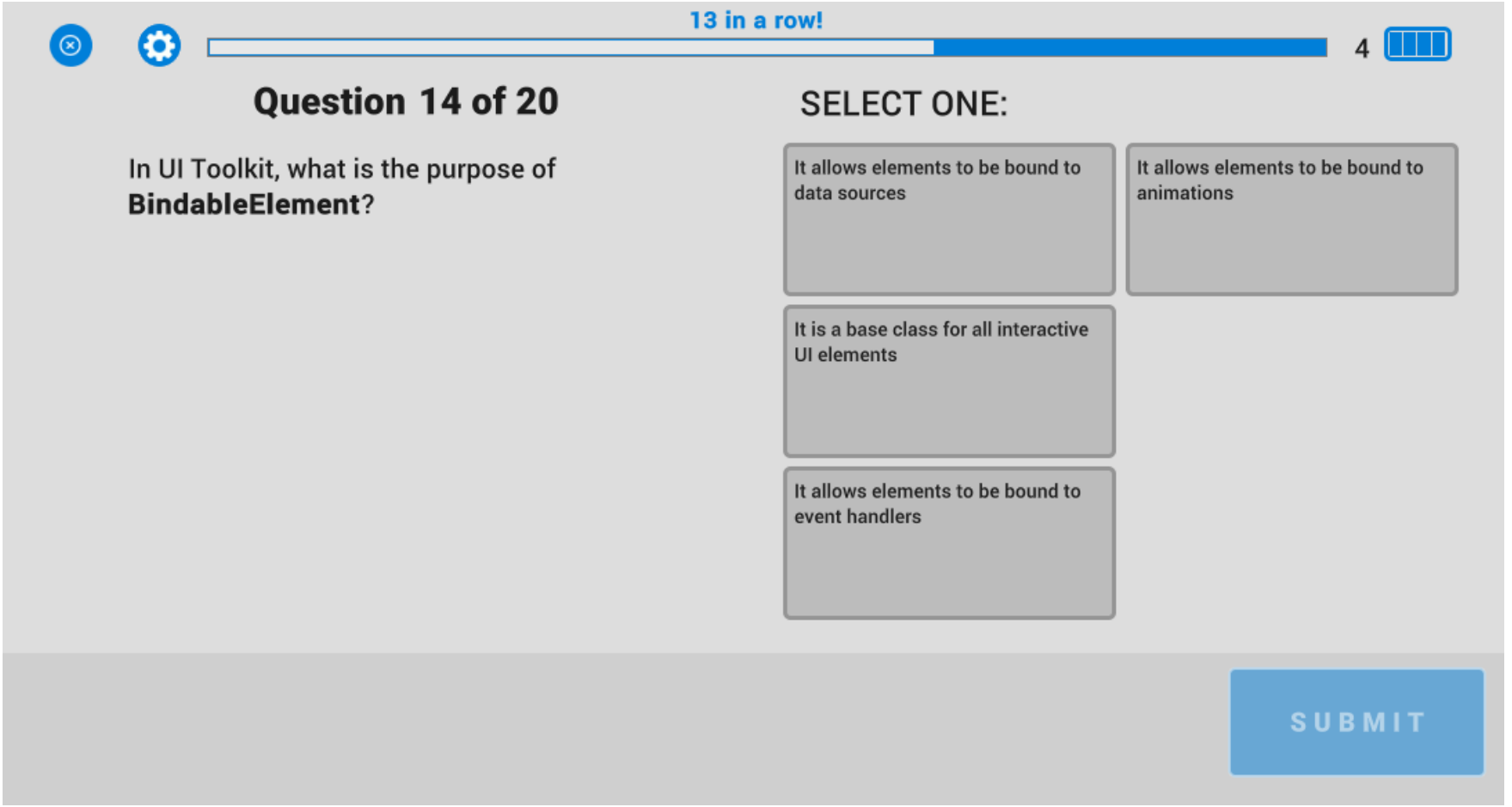
# Quiz → UI Toolkit



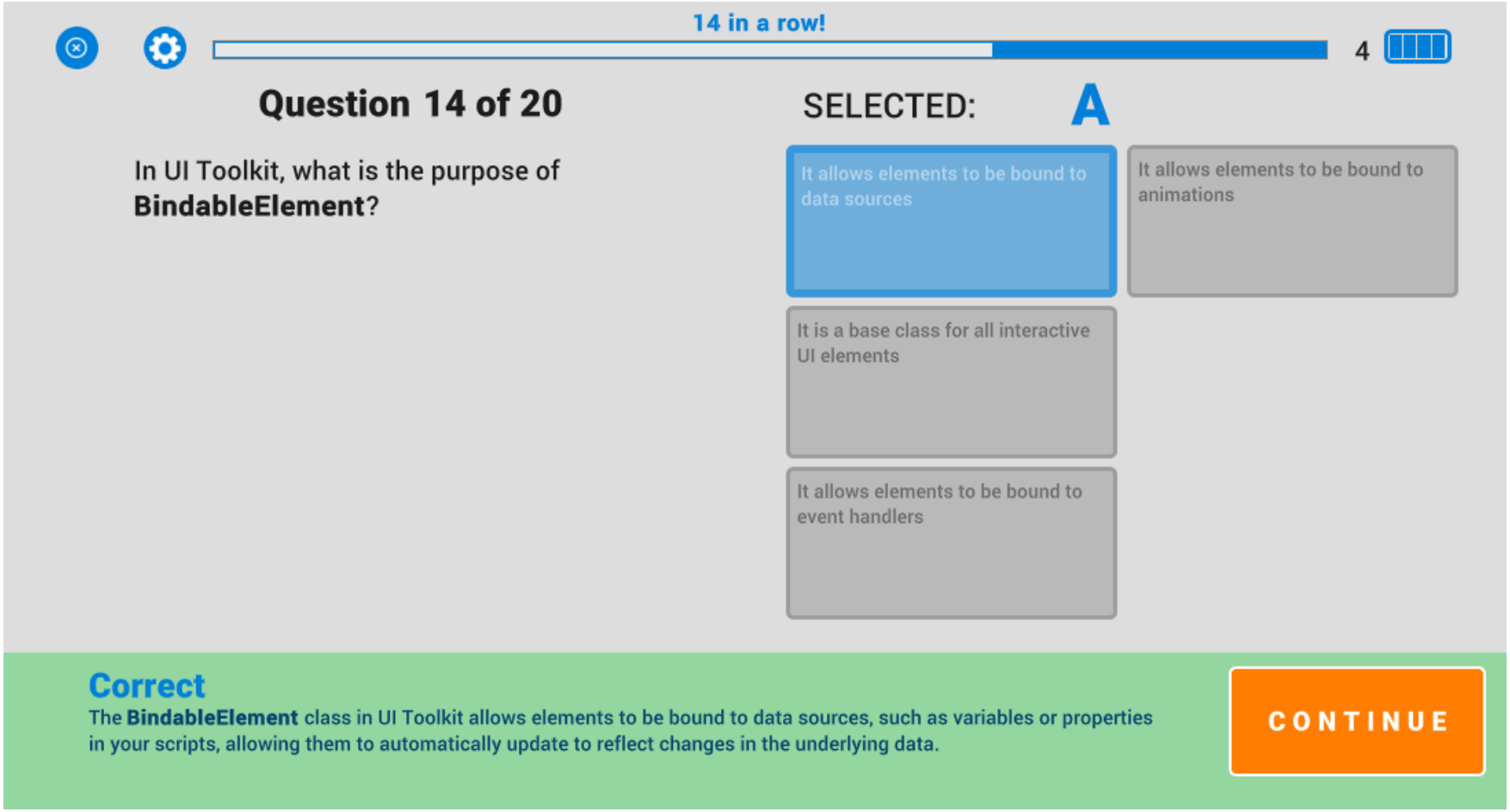
# Quiz → UI Toolkit



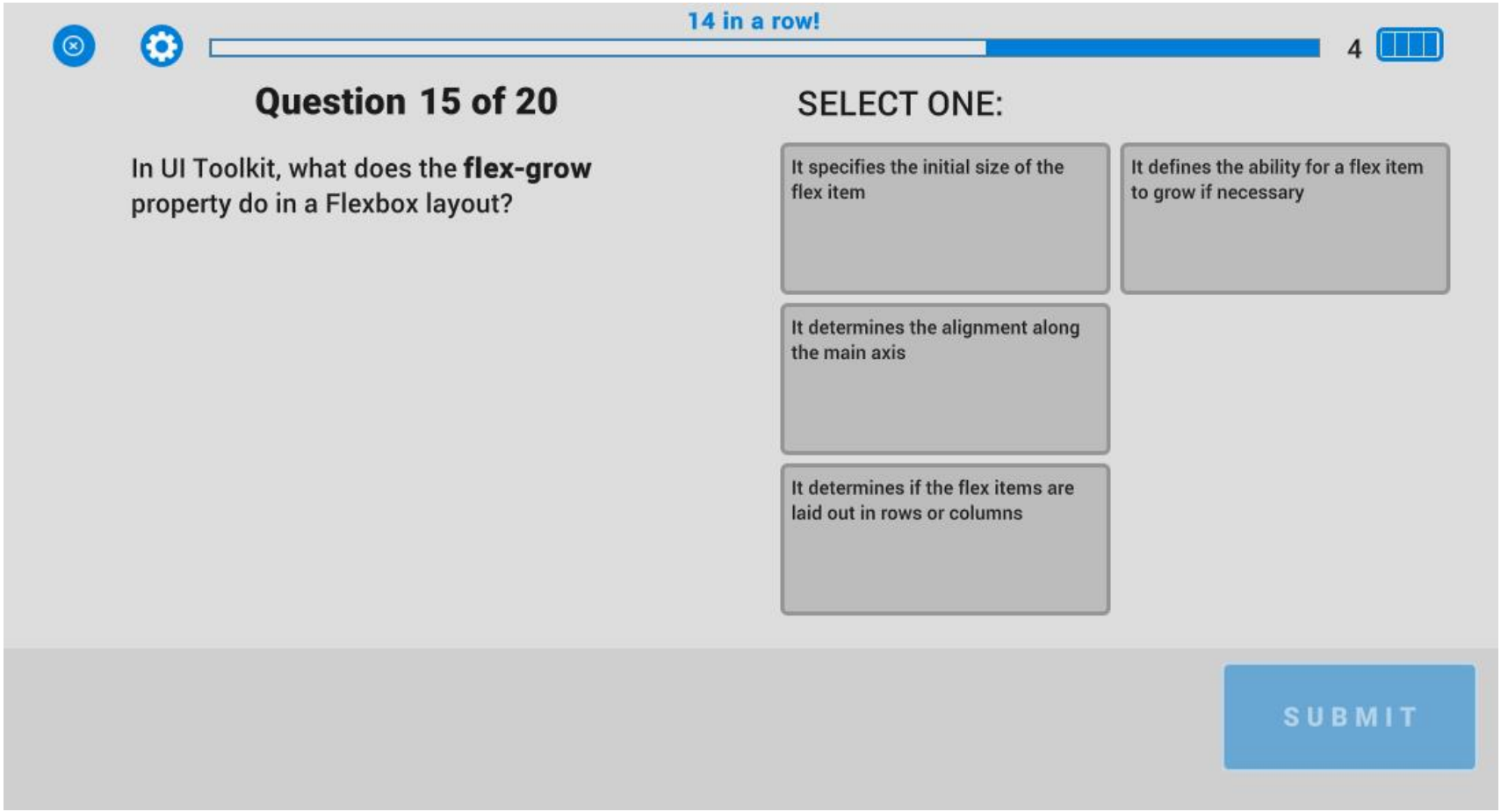
# Quiz → UI Toolkit



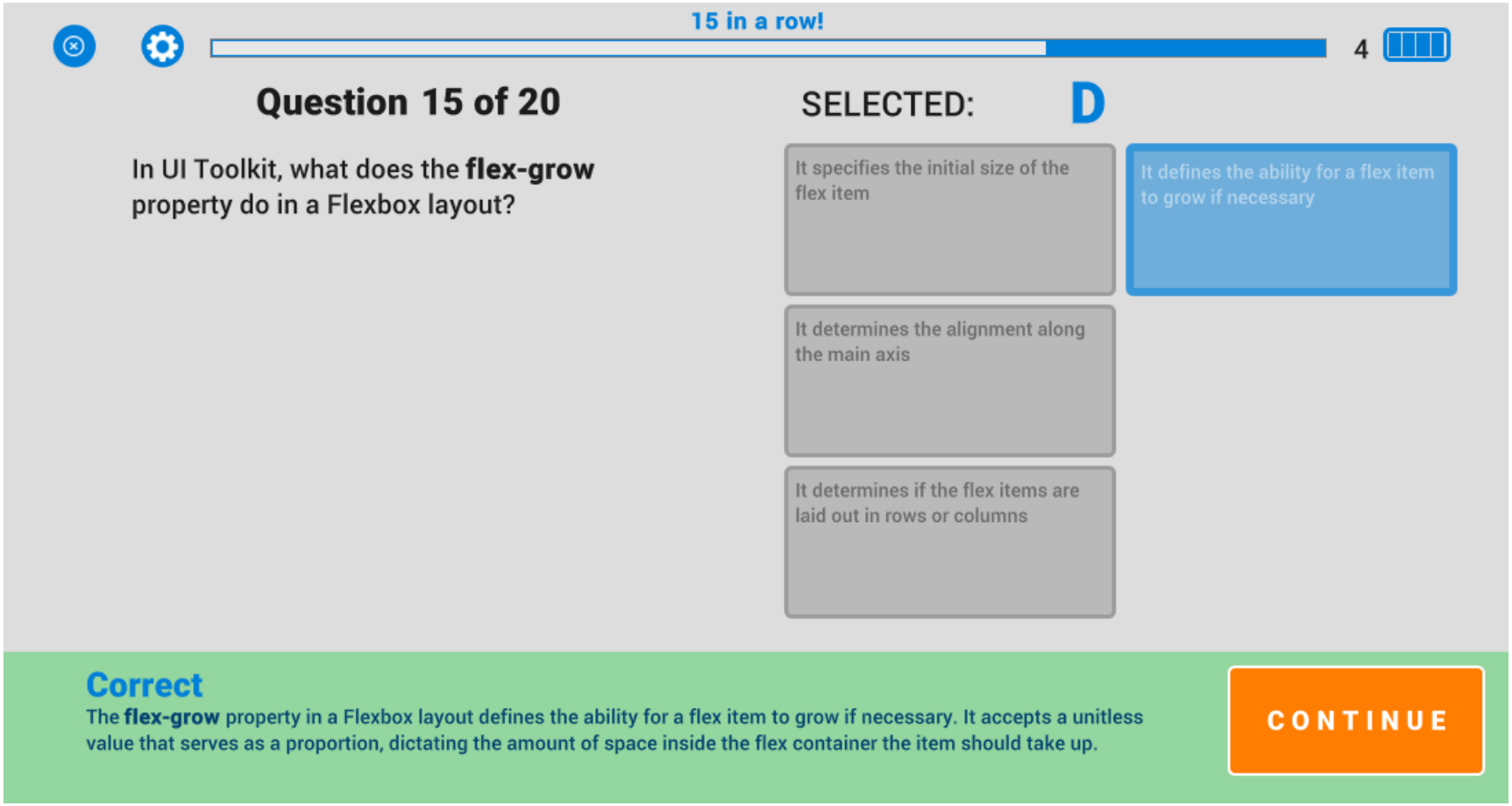
# Quiz → UI Toolkit



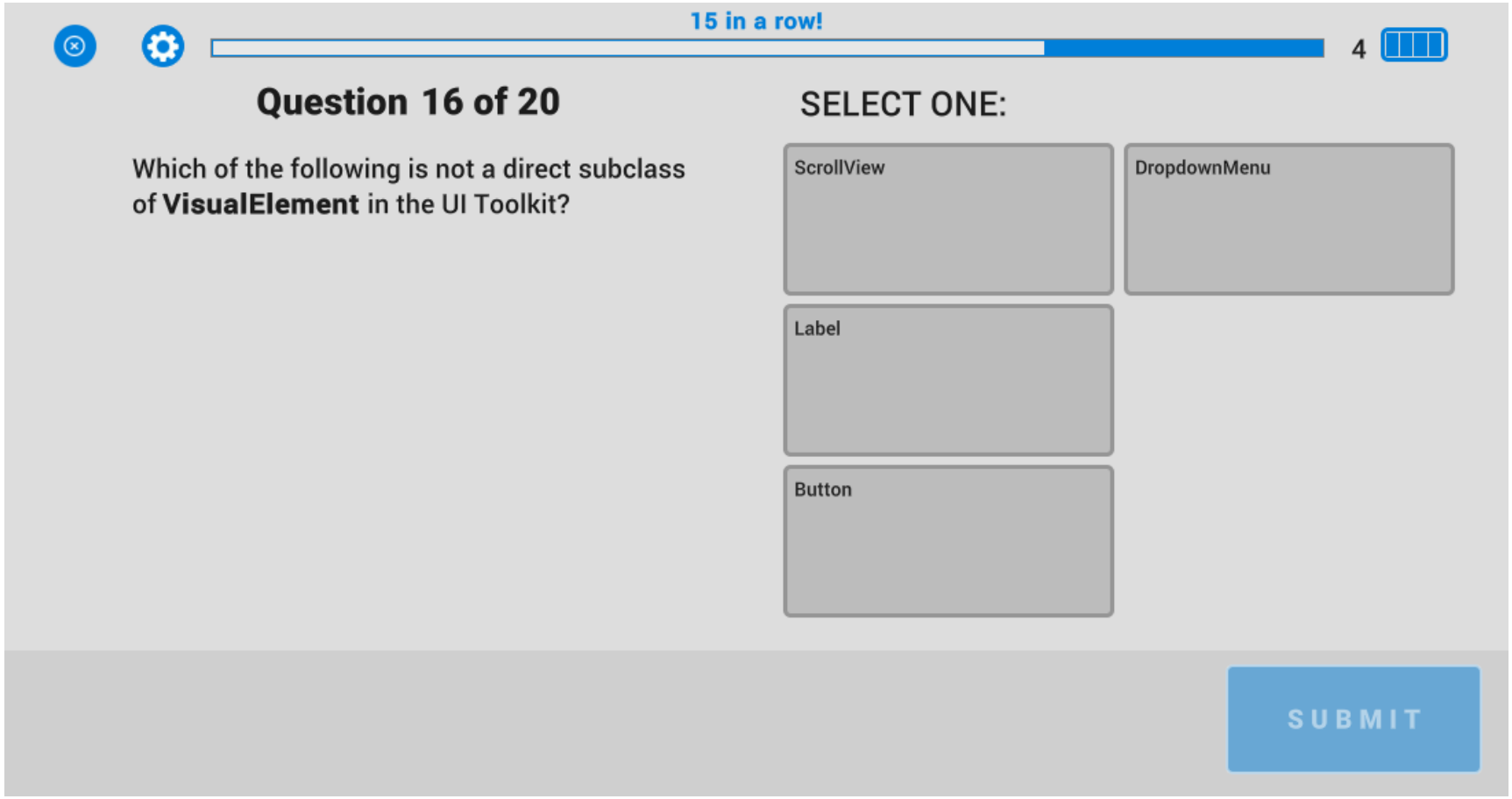
# Quiz → UI Toolkit



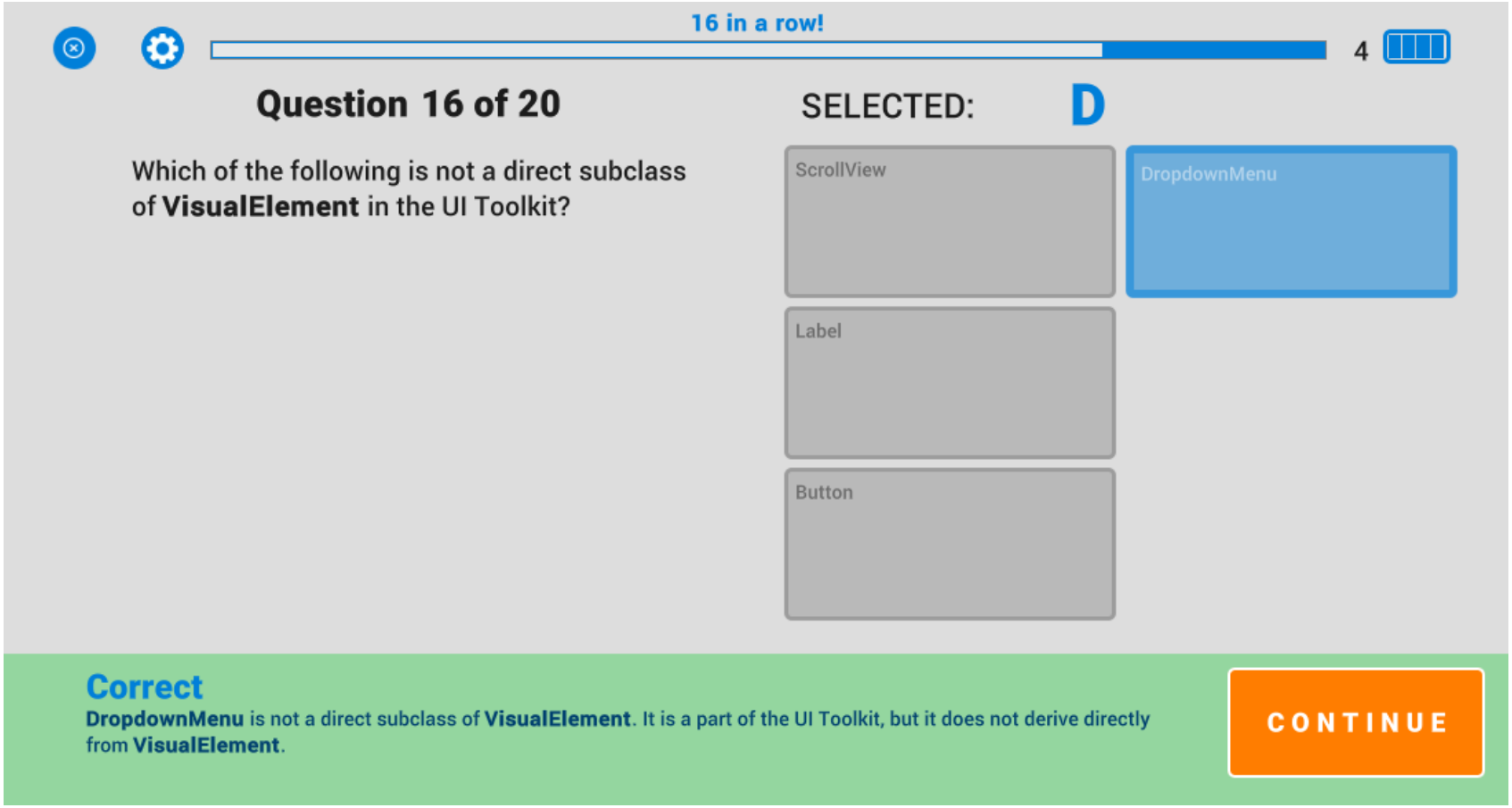
# Quiz → UI Toolkit



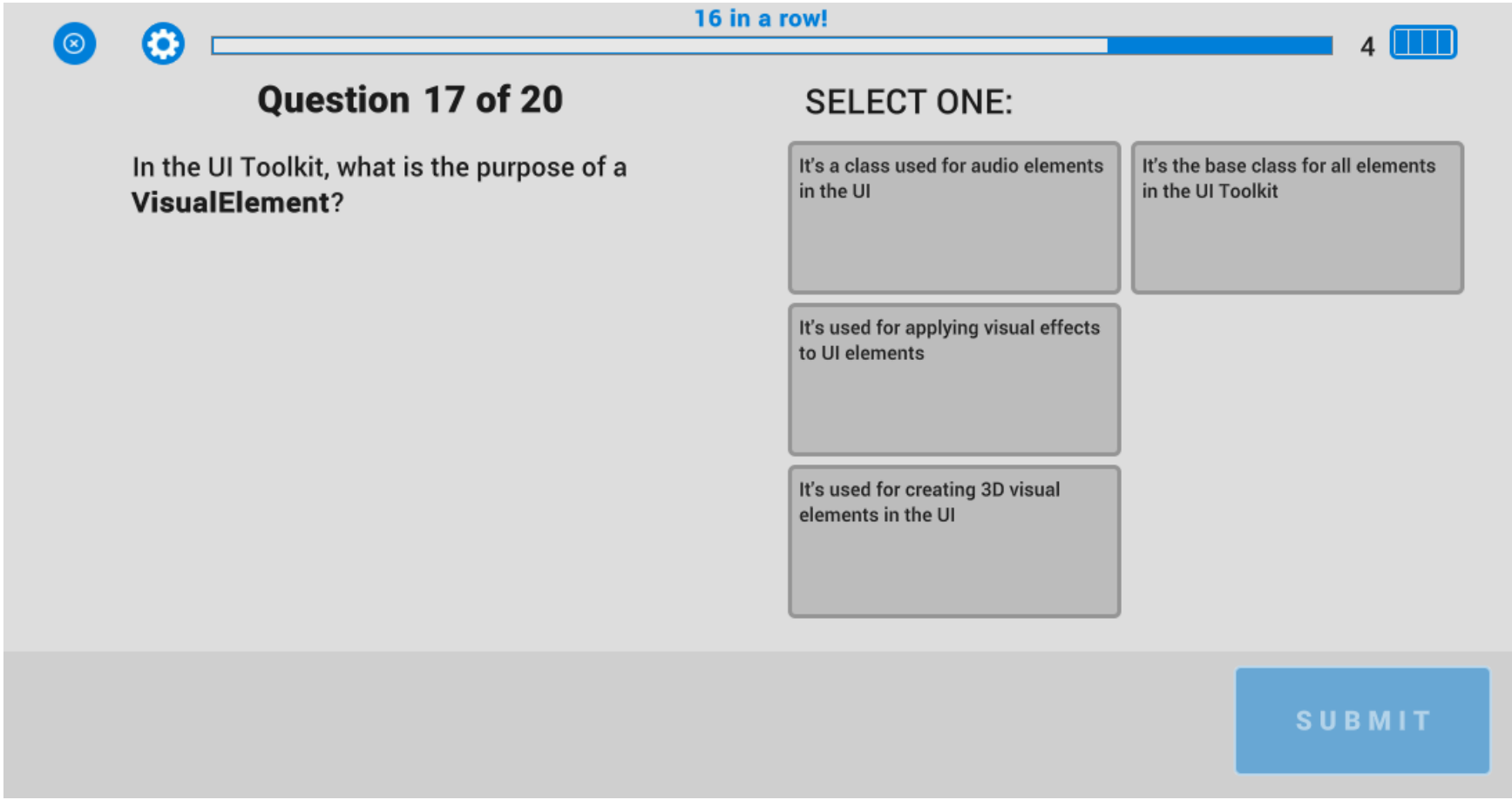
# Quiz → UI Toolkit



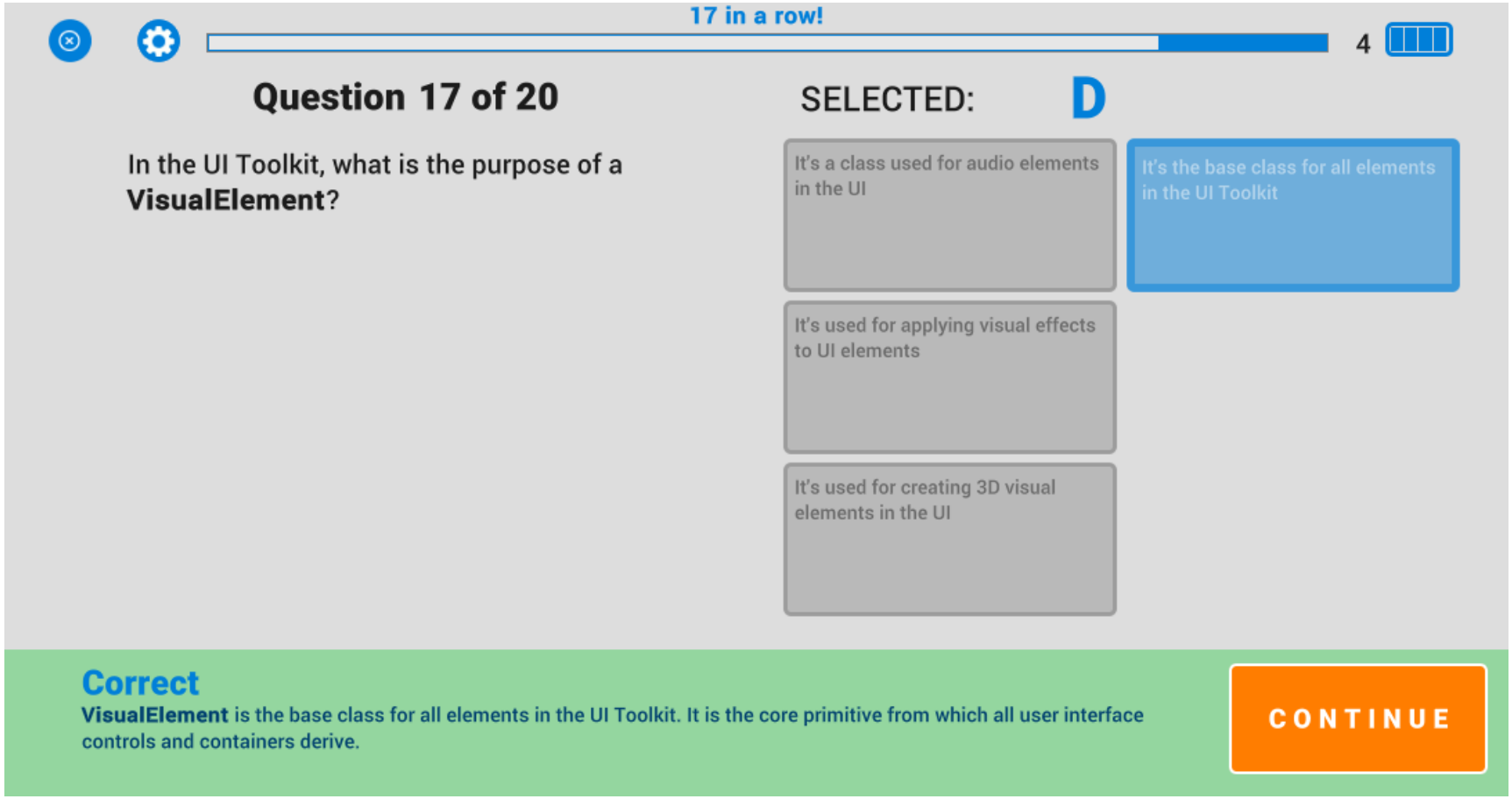
# Quiz → UI Toolkit



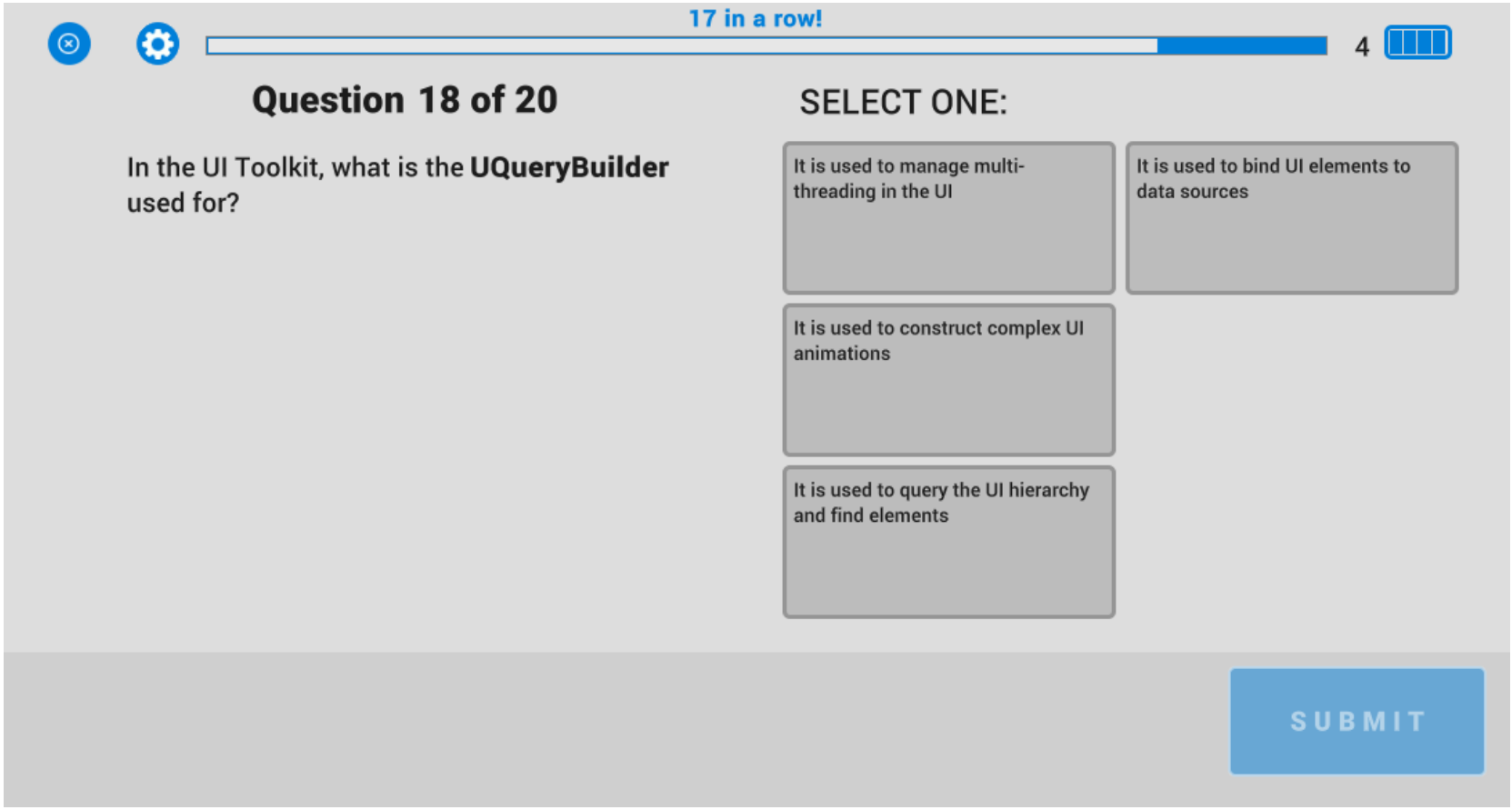
# Quiz → UI Toolkit



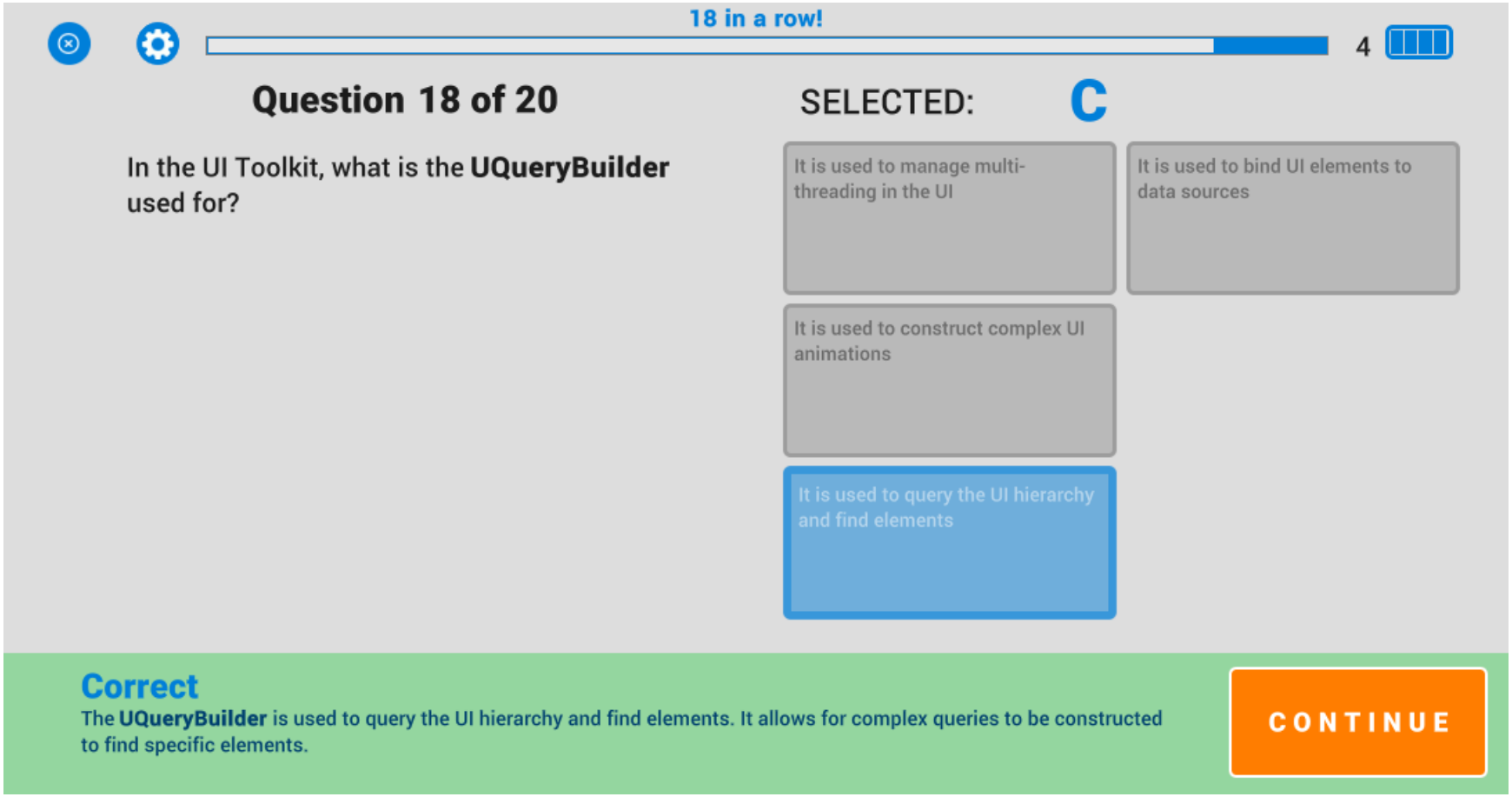
# Quiz → UI Toolkit



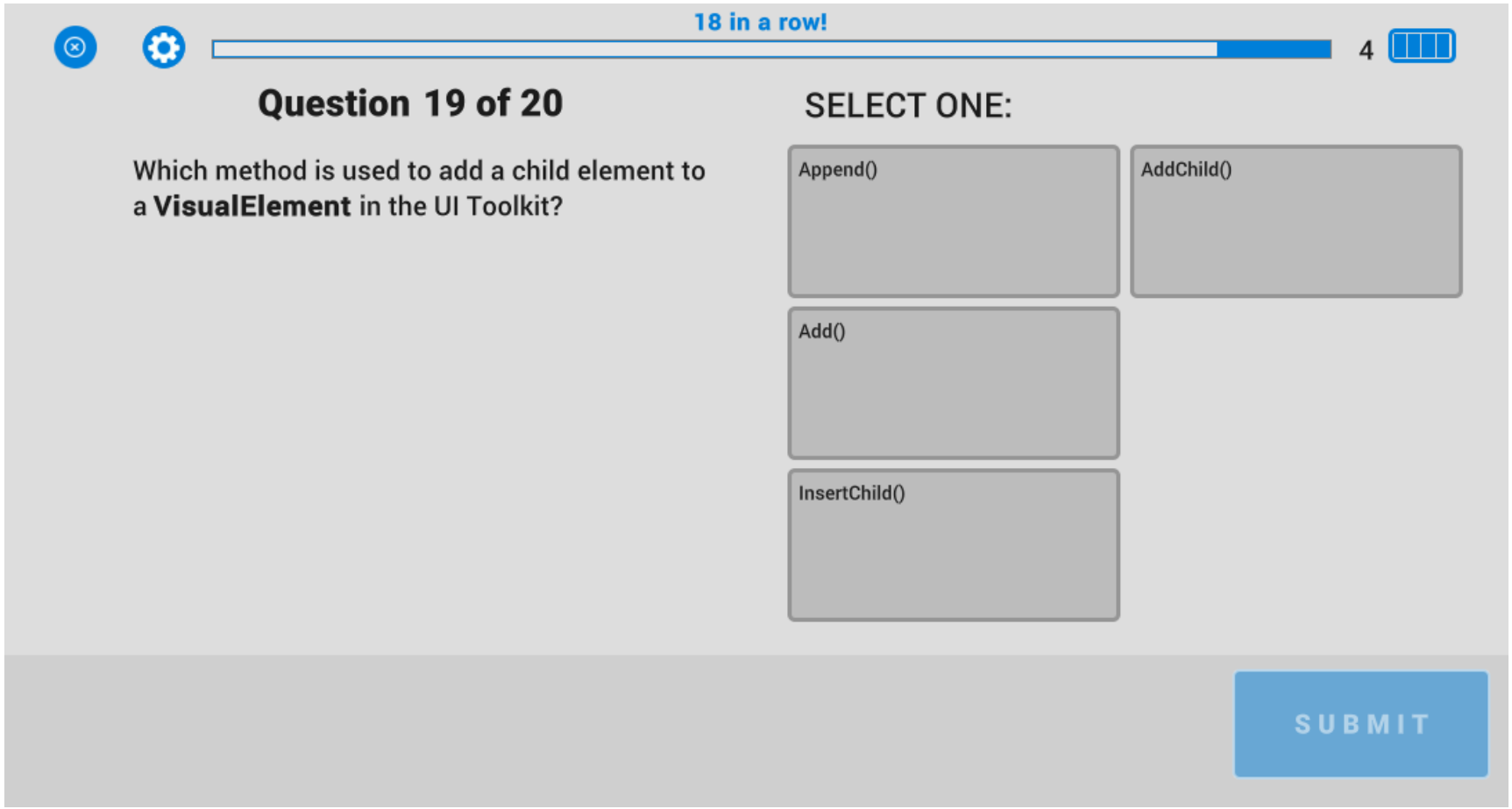
# Quiz → UI Toolkit



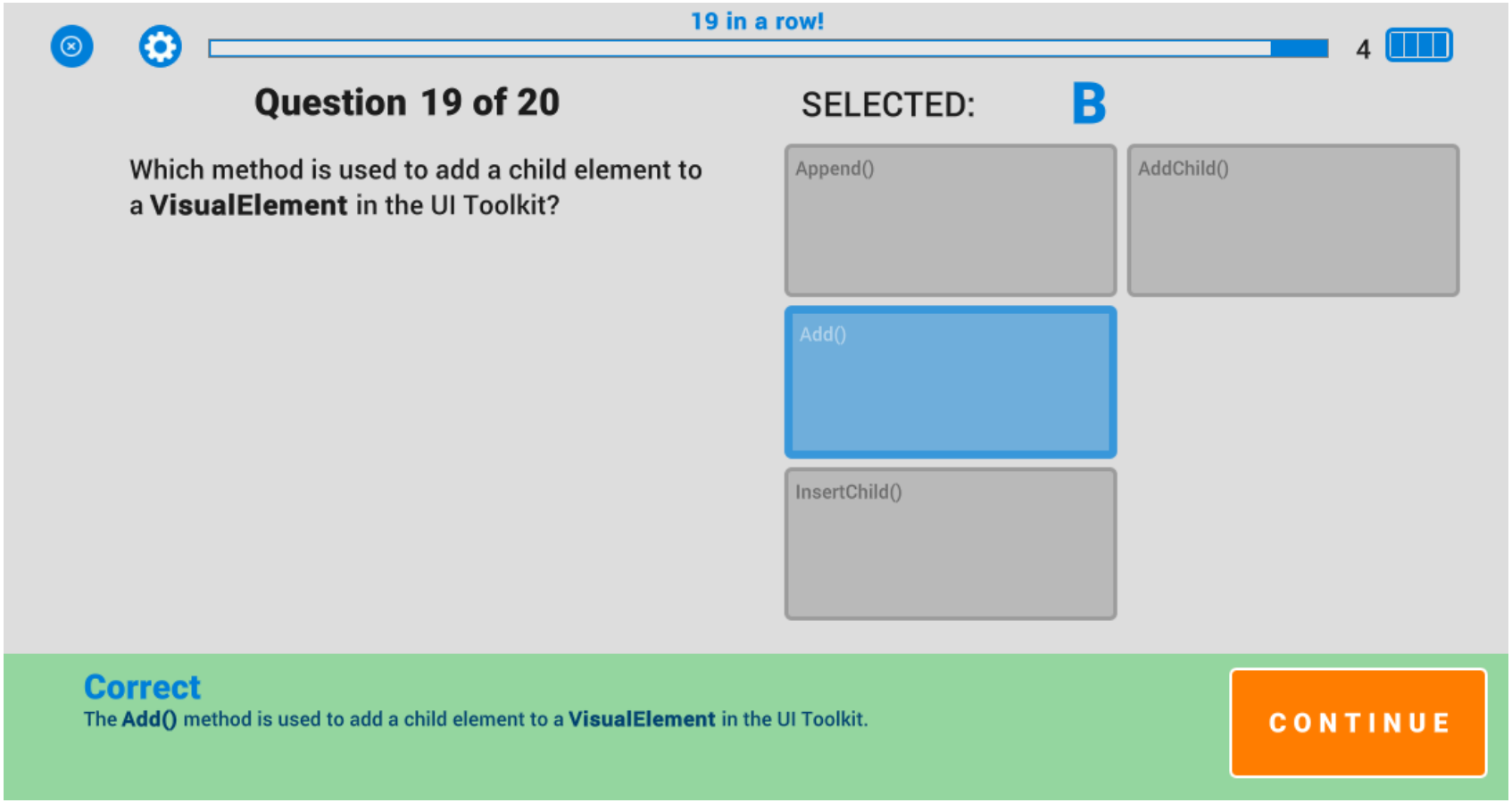
# Quiz → UI Toolkit



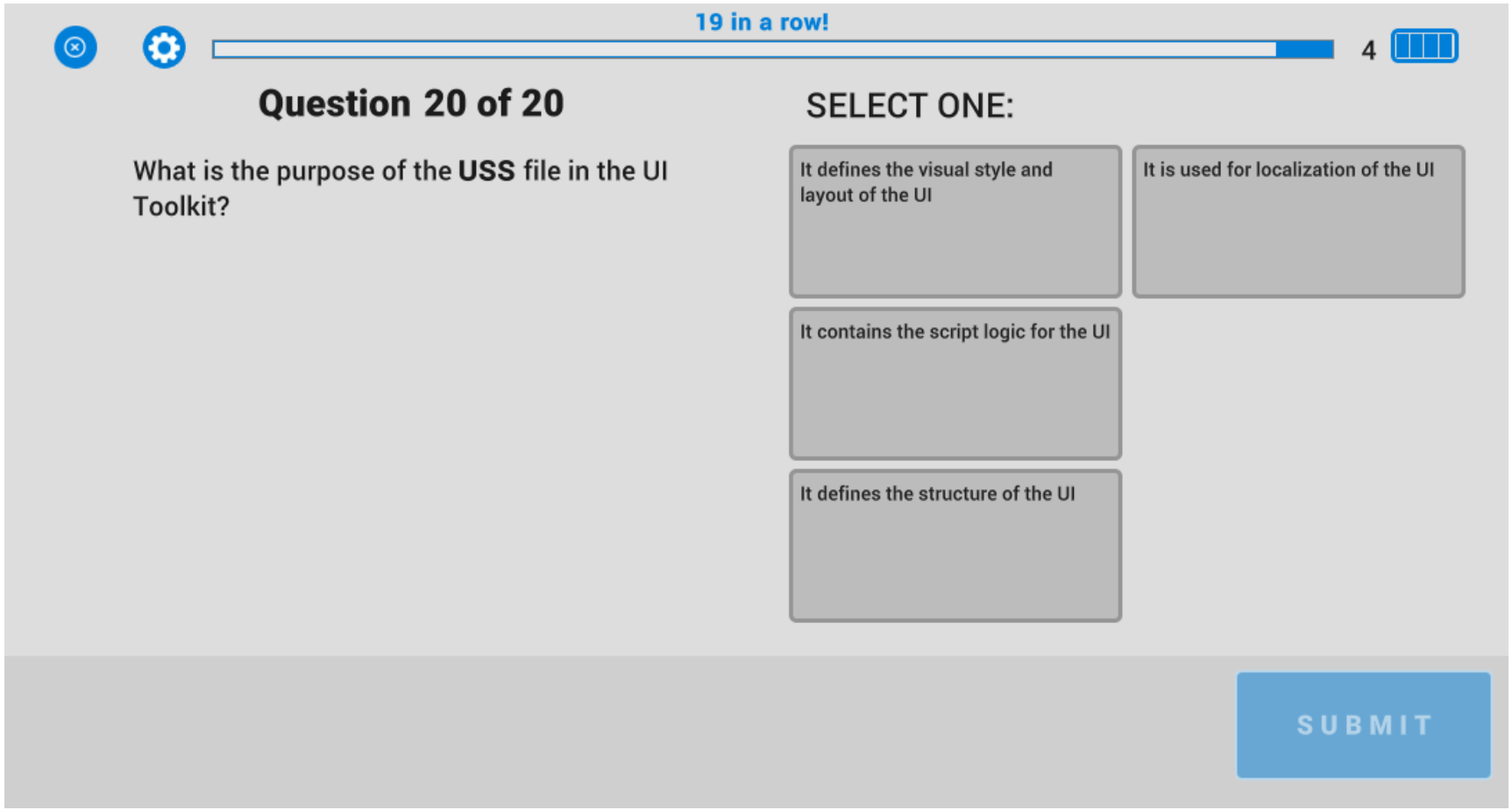
# Quiz → UI Toolkit



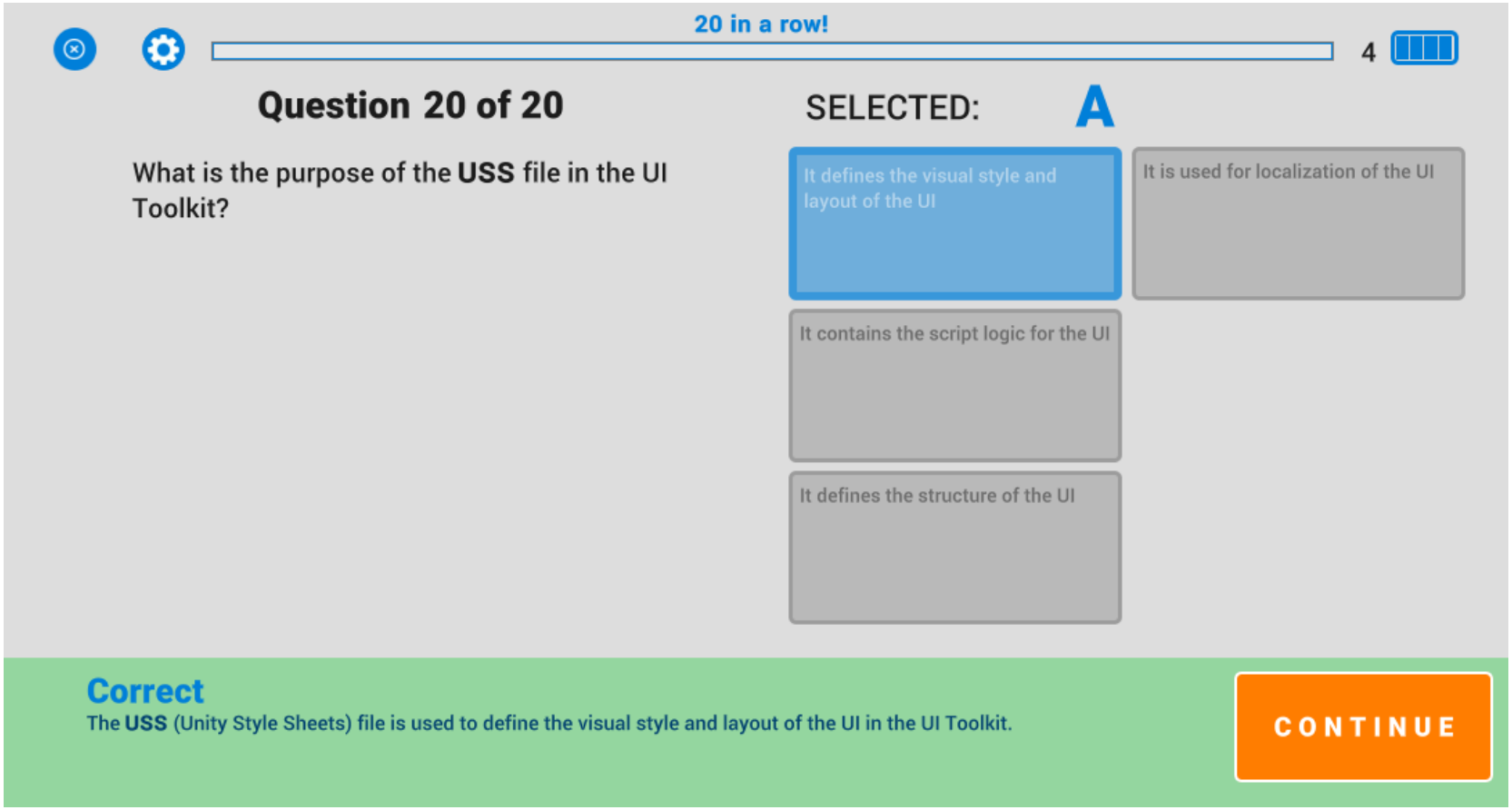
# Quiz → UI Toolkit



# Quiz → UI Toolkit



# Quiz → UI Toolkit



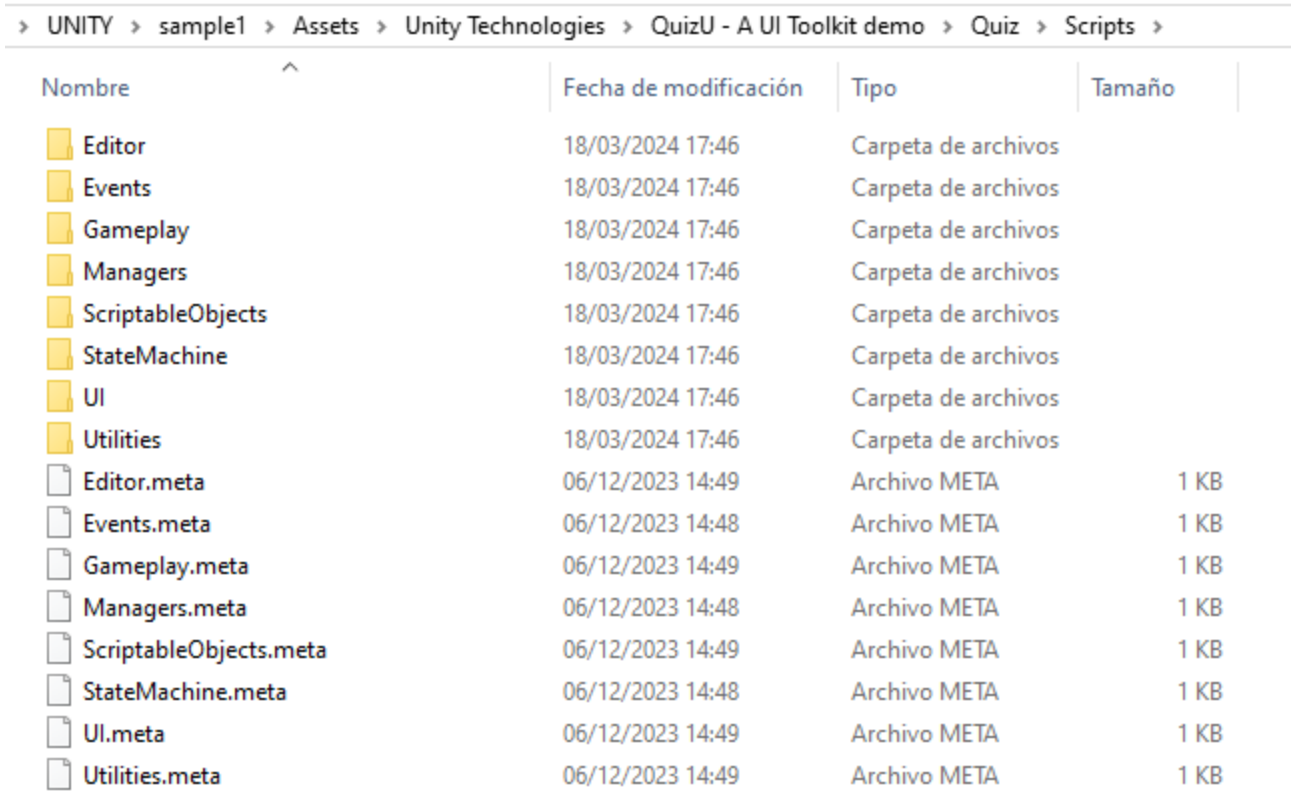
# Quiz


UNITY › sample1 › Assets › Unity Technologies › QuizU - A UI Toolkit demo › Quiz ›
Nombre
Fecha de modificación
Tipo
Tamaño
Data 18/03/2024 17:46 Carpeta de archivos
Prefabs 18/03/2024 17:46 Carpeta de archivos
Scripts 18/03/2024 17:46 Carpeta de archivos
Textures 18/03/2024 17:46 Carpeta de archivos
TutorialInfo 18/03/2024 17:46 Carpeta de archivos
UI 18/03/2024 17:46 Carpeta de archivos
Data.meta 06/12/2023 14:48 Archivo META 1 KB
Prefabs.meta 06/12/2023 14:48 Archivo META 1 KB
Scripts.meta 06/12/2023 14:48 Archivo META 1 KB
Textures.meta 06/12/2023 14:48 Archivo META 1 KB
TutorialInfo.meta 06/12/2023 14:49 Archivo META 1 KB
UI.meta 06/12/2023 14:50 Archivo META 1 KB

# Data → JSON


```json
{
    "text": "How many audio listeners should be present in a Unity scene?",
    "skills": "Implement audio in Unity, Produce customized results by correctly configuring audio in a scene",
    "shuffleOptions": true,
    "options": [
        {
            "text": "As many as there are sound producing objects",
            "isCorrect": false,
            "feedback": "An audio listener plays any audio sources that are within range in a scene. There is only ever one audio listener, generally on the main camera."
        },
        {
            "text": "Two",
            "isCorrect": false,
            "feedback": "An audio listener plays any audio sources that are within range in a scene. There is only ever one audio listener, generally on the main camera."
        },
        {
            "text": "One for each directly interactable object",
            "isCorrect": false,
            "feedback": "An audio listener plays any audio sources that are within range in a scene. There is only ever one audio listener, generally on the main camera."
        },
        {
            "text": "One",
            "isCorrect": true,
            "feedback": "There is only ever one audio listener, generally on the main camera."
        }
    ],
    "feedbackCorrect": "There is only ever one audio listener, generally on the main camera.",
    "feedbackIncorrect": ""
}
```

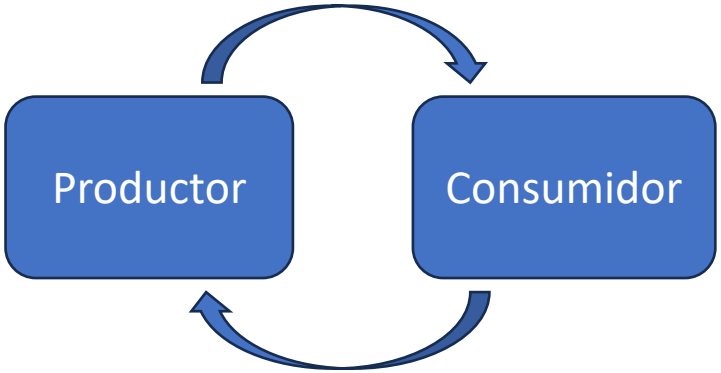
# Scripts

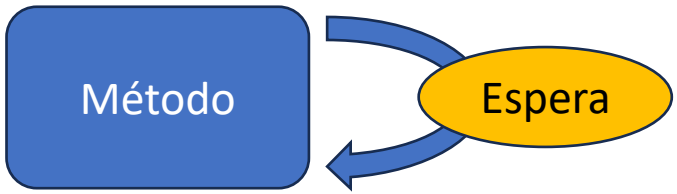

UNITY › sample1 › Assets › Unity Technologies › QuizU - A UI Toolkit demo › Quiz › Scripts ›
Nombre
Fecha de modificación
Tipo
Tamaño
Editor 18/03/2024 17:46 Carpeta de archivos
Events 18/03/2024 17:46 Carpeta de archivos
Gameplay 18/03/2024 17:46 Carpeta de archivos
Managers 18/03/2024 17:46 Carpeta de archivos
ScriptableObjects 18/03/2024 17:46 Carpeta de archivos
StateMachine 18/03/2024 17:46 Carpeta de archivos
UI 18/03/2024 17:46 Carpeta de archivos
Utilities 18/03/2024 17:46 Carpeta de archivos
Editor.meta 06/12/2023 14:49 Archivo META 1 KB
Events.meta 06/12/2023 14:48 Archivo META 1 KB
Gameplay.meta 06/12/2023 14:49 Archivo META 1 KB
Managers.meta 06/12/2023 14:48 Archivo META 1 KB
ScriptableObjects.meta 06/12/2023 14:49 Archivo META 1 KB
StateMachine.meta 06/12/2023 14:48 Archivo META 1 KB
UI.meta 06/12/2023 14:49 Archivo META 1 KB
Utilities.meta 06/12/2023 14:49 Archivo META 1 KB

# State Machine → Coroutines

```
var q := new queue

coroutine produce
    loop
        while q is not full
            create some new items
            add the items to q
        yield to consume

coroutine consume
    loop
        while q is not empty
            remove some items from q
            use the items
        yield to produce

call produce
```

```
IEnumerator Fade()
{
    Color c = renderer.material.color;
    for (float alpha = 1f; alpha >= 0; alpha -= 0.1f)
    {
        c.a = alpha;
        renderer.material.color = c;
        yield return new WaitForSeconds(.1f);
    }
}
```



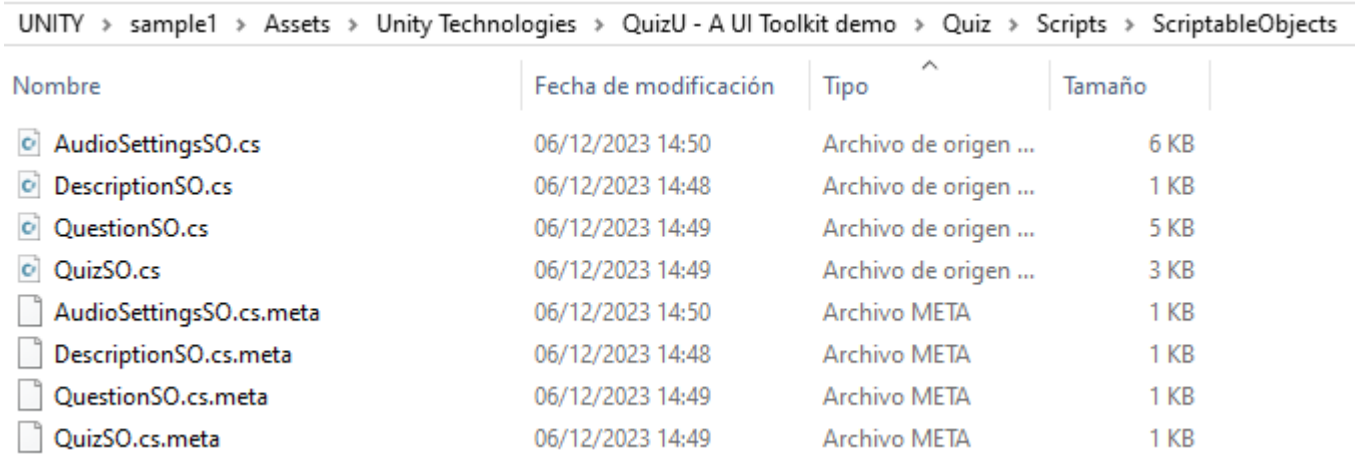



# State Machine → Coroutines

```
                1 reference
124             protected virtual IEnumerator Loop()
125             {
126                 while (true)
127                 {
128                     if (CurrentState != null && m_CurrentPlayCoroutine == null) //current state is done playing
129                     {
130                         if (CurrentState.ValidateLinks(out var nextState))
131                         {
132                             if (m_PlayLock)
133                             {
134                                 //finalize current state
135                                 CurrentState.Exit();
136                                 m_PlayLock = false;
137                             }
138
139                             CurrentState.DisableLinks();
140                             SetCurrentState(nextState);
141                             CurrentState.EnableLinks();
142                         }
143                     }
144
145                     yield return null;
146                 }
147             }
148
                0 references
149             public bool IsRunning => m_LoopCoroutine != null;
150         }
151     }
```

# ScriptableObjects

- Objetos que son instancias de clases que heredan de MonoBehaviour (“MonoBehaviourObjects”) son almacenados en memoria y existen solo mientras se juega el juego.
- **ScriptableObjects** → son almacenados en el disco duro en forma serializada (existen durante y después del juego). Son usados como un contenedor/plantilla para almacenar datos.



# AudioSettingsSO

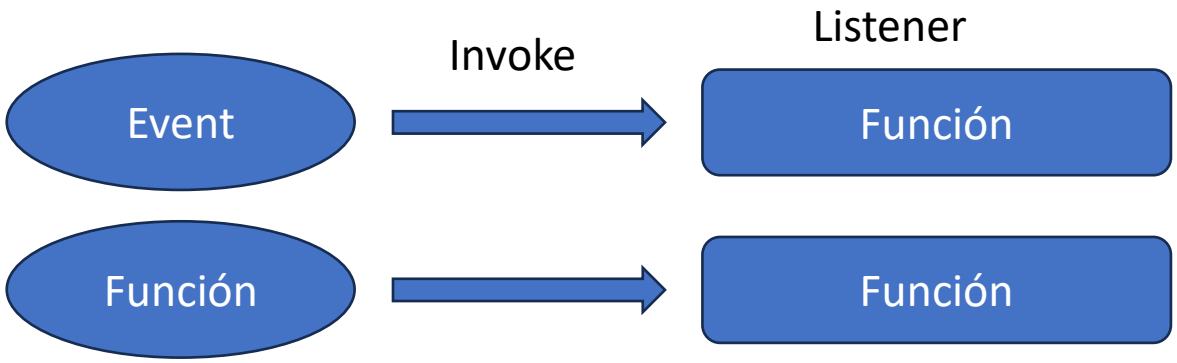
```
14      [CreateAssetMenu(fileName = "AudioSettings", menuName = "Quiz/AudioSettings", order = 1)]
        0 references
15      public class AudioSettingsSO : DescriptionSO
16      {
17          // Default ScriptableObject data
            1 reference
18          const float k_DefaultMasterVolume = 1f;
            1 reference
19          const float k_DefaultSFXVolume = 1f;
            1 reference
20          const float k_DefaultMusicVolume = 0f;
21
22          [Header("Mixer")]
23          [Tooltip("The AudioMixer that controls the audio levels for the game")]
            1 reference
24          [SerializeField] private AudioMixer m_AudioMixer;
25
26          [Header("Volume Settings")]
27          [Tooltip("The master volume level (0 to 1)")]
            2 references
28          [SerializeField] private float m_MasterVolume = k_DefaultMasterVolume;
29
30          [Tooltip("The sound effects volume level (0 to 1)")]
            2 references
31          [SerializeField] private float m_SoundEffectsVolume = k_DefaultSFXVolume;
32
33          [Tooltip("The music volume level (0 to 1)")]
            2 references
34          [SerializeField] private float m_MusicVolume = k_DefaultMusicVolume;
35
36          // Convert bool to 1 and 0 and multiply by MixerGroup (unused in this project, here for demo purposes)
37          [Header("Mute Settings")]
38          [Tooltip("Mute or unmute the master volume")]
            2 references
39          [SerializeField] private bool m_IsMasterMuted = false;
40
41          [Tooltip("Mute or unmute the sound effects volume")]
            2 references
42          [SerializeField] private bool m_IsSoundEffectsMuted = false;
43
44          [Tooltip("Mute or unmute the music volume")]
            2 references
45          [SerializeField] private bool m_IsMusicMuted = false;
46
```

# QuestionSO

```
33      public class QuestionSO : ScriptableObject
34      {
35          [Tooltip("List of question text, supporting different font sizes.")]
            2 references
36          [SerializeField] List<QuestionText> m_QuestionText;
37          [Tooltip("Optional image associated with the question.")]
            1 reference
38          [SerializeField] [Optional] Sprite m_QuestionImage;
39          [TextArea(3, 5)]
40          [Tooltip("Skills tested or used in the question.")]
            2 references
41          [SerializeField] string m_Skills;
42          [Tooltip("Should the answers should be shuffled?")]
            2 references
43          [SerializeField] bool m_ShuffleAnswers;
44          [Tooltip("Additional information about the correct/incorrect responses")]
            6 references
45          [SerializeField] List<Answer> m_Answers;
46          [TextArea(3, 5)]
47          [Tooltip("Additional information provided when the correct answer is chosen.")]
            2 references
48          [SerializeField] string m_FeedbackCorrect;
49          [TextArea(3, 5)]
50          [Tooltip("Additional information provided when an incorrect answer is chosen.")]
            2 references
51          [SerializeField] string m_FeedbackIncorrect;
52          [Tooltip("Optional link to the documentation for more information.")]
            0 references
53          [SerializeField][Optional] string m_FeedbackLink;
54
```

# Actions

- En clase hemos visto Delegates, events y callbacks. Resumiendo:
  - Delegates en C# son punteros a funciones. Los Delegates nos permiten usar las funciones como parámetros de entrada de otras funciones. Esto nos permite crear Callbacks.
  - En los Callbacks, un evento de sistema dispara una función.
  - En las Actions, el final de una función dispara otra función.
  - Observer Pattern:



# Events

```
15      public class AudioSettingsSO : DescriptionSO
76
77          // Event subscriptions
            0 references
78          private void OnEnable()
79          {
80              SettingsEvents.MasterVolumeChanged += SettingsEvents_MasterVolumeChanged;
81              SettingsEvents.MusicVolumeChanged += SettingsEvents_MusicVolumeChanged;
82              SettingsEvents.SFXVolumeChanged += SettingsEvents_SFXVolumeChanged;
83
84              // Note: we disable validating the Sound Effects AudioClips here, instead we
85              // validate it from the GameplaySounds class
86
87              // NullRefChecker.Validate(this);
88          }
89
90          // Event unsubscriptions
            0 references
91          private void OnDisable()
92          {
93              SettingsEvents.MasterVolumeChanged -= SettingsEvents_MasterVolumeChanged;
94              SettingsEvents.MusicVolumeChanged -= SettingsEvents_MusicVolumeChanged;
95              SettingsEvents.SFXVolumeChanged -= SettingsEvents_SFXVolumeChanged;
96          }
97
```

# Game events → Actions

```
0 references
public static class GameEvents
{
    #region Gameplay events

    0 references
    public static Action<int> IndexUpdated;

    // Refreshed the current Question with new ScriptableObject data
    0 references
    public static Action<QuestionSO> QuestionUpdated;

    // Selected one or more user selected responses
    0 references
    public static Action<List<string>> AnswerSelected;

    // Submitted the selected answer
    0 references
    public static Action AnswerSubmitted;

    // Passes feedback for correct/incorrect responses
    0 references
    public static Action<string> CorrectlyAnswered;
    0 references
    public static Action<string> IncorrectlyAnswered;
```

# Índice

- Project Samples
  - Dragon Crashers
    - Custom Controls
      - Tabbed menú
      - Radial progress bar
  - Quizu
    - Demos
      - CustomControls
      - Manipulators
    - Quiz
      - Data
      - Scripts
        - Coroutines
        - ScriptablesObjects
        - Actions

- **Proyecto**

# Proyecto

- El proyecto se hace individual o en grupos de 2-3 personas.
- Tiene que recoger más o menos todo lo que hemos ido viendo en las prácticas.
- Crearos una cuenta en Github (si no la tenéis ya).
- En el campus virtual hay dos enlaces:
  - 1. PDF con el prototipo en Figma.
  - 2. Fichero de texto con el enlace a github para el proyecto.

Último día presentación → Proyecto y prácticas

| Mayo 2024 | | | | | | |
|---|---|---|---|---|---|---|
| 29 | 30 | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |

Examen (si no proyecto)

| | | |
|---|---|---|
| Miércoles 29 | 9:00 | 2ºDSI |
| | 13:00 | |
| de mayo | 17:00 | CSV |
| Jueves 30 | 9:00 | IW |
| | 13:00 | |
| de mayo | 17:00 | EIRV |
| Viernes 31 | 9:00 | 2ºP2 |
| | 13:00 | |
| de mayo | 17:00 | 3ºP3 |
| | | Gr. Videojuegos |