Unity – przypisanie EventHandlerów z tablicy zewnętrznych obiektów (sheadovas/poradniki/goto/unity-przypisanie-eventhandlerow-z-tablicy-zewnetrznych-obiektow/)

Lip 09, 2016 / goto (sheadovas/category/poradniki/goto/)

(oczywiście przy użyciu C#)

Ostatnio zdarzyło mi się powrócić do Unity (przynajmniej na chwilę) i udało mi się natrafić na pewien problem, którego rozwiązanie – przynajmniej na pierwszy rzut oka – nie jest zbyt oczywiste.

Intro

A więc załóżmy, że mamy skrypt, w którym posiadamy 2 niepowiązane ze sobą tablice:

1. tablica przycisków (buttons)
2. tablica obiektów jakiejś zewnętrznej klasy, której metodę chcemy wywołać na zdarzenie *OnClick* (listeners)

Listing 1 C#

```
public class Controller : MonoBehaviour
{
    public void Toggle() { ... }
}


// -------------------


public class MenuUiManager : MonoBehaviour
{
    public Button[] buttons;
    public Controller[] controllers;

    ...
}
```

To co mamy w chwili startowej to niepowiązane ze sobą obiekty przycisków (buttons) oraz kontrolerów (controllers), same kontrolery mogą być podczepione do innego zewnętrznego obiektu z racji większej wygody i chęci odseparowania pewnej logiki od samego przycisku (i tak jest w tym przypadku).

W naszym przypadku rozmiar obu tablic jest identyczny oraz same obiekty są w relacji 1-1, tzn 1-szy button po zdarzeniu *OnClick* powinien wywołać metodę `Toggle()` 1-szego kontrolera, itd.

Rozwiązanie

Pierwsze podejście

Rozwiązanie wydaje się oczywiste:

Listing 2 C#

```
1  void Start()
2  {
3      if (controllers.Length != buttons.Length)
4          return; // różne rozmiary tablic
5
6      for (int i = 0; i < controllers.Length; ++i)
7      {
8          buttons[i].onClick.AddListener(() => controllers[i].Toggle());
9      }
10 }
```

Jednakże to rozwiązanie , które na pozór powinno przypisać i-temu przyciskowi – i-ty Toggle tego nie robi. W momencie wygenerowania zdarzenia OnClick (używając dowolnego przycisku) otrzymujemy informację o błędzie (crash):

> IndexOutOfRangeException: Array index is out of range.

Ze wskazaniem na linię dodającą listener do przycisku (w powyższym listingu to linia 8).

Drugie podejście

Na pierwszy rzut oka nie widać nigdzie wyjścia poza zakres tablicy (a to się dzieje gdy spróbujemy kliknąć dowolny przycisk), jak się okazuje wyrażenia lambda używają wszędzie najnowszej wartości i, która przy takim zapisie jest zawsze identyczna i wynosi rozmiar tablicy (n), czyli mamy oczywiste wyjście poza zakres tablicy.

> **Przykład:** jeżeli zastosowalibyśmy tablice o rozmiarze 5 (controllers[5]), to za każdym razem próbowalibyśmy przypisać Toggle z obiektu controllers[5], a jak wiadomo element o takim indeksie nie istnieje.

Rozwiązanie okazuje się trywialne i wymagające utworzenia zmiennej wewnątrz pętli będącą kopią licznika pętli oraz wrappera na funkcję, co także nieco ułatwi (zwiększy czytelność) bardziej złożone wyrażenia.

C#

```
1  void Start()
2  {
3      if (controllers.Length != buttons.Length)
4          return; // różne rozmiary tablic
5      else
6      {
7          int idx = 0;
8          foreach (Button b in buttons)
9          {
10             int i = idx;
11             b.onClick.AddListener(() => ToggleWrapper(i));
12             idx++;
13         }
14     }
15 }
16
17
18 void ToggleWrapper(int idx)
19 {
20     controllers[idx].Toggle();
21 }
```

Taka konstrukcja zapewnia poprawne podłączenie event handlerów do przycisków.

Code ON!