Czym są pętle i jak z nich korzystać? (while, for, foreach)

Wprowadzenie

Pętle to jedno z najważniejszych narzędzi w programowaniu. Pozwalają na **wielokrotne wykonywanie** tego samego kodu bez konieczności jego powtarzania. Dzięki pętlom możemy np. przeglądać listę obiektów, wykonywać obliczenia dla wielu wartości czy tworzyć sekwencje zdarzeń. W tej dokumentacji dowiesz się, czym są pętle, jak działają oraz jak używać ich w Unity.

Podstawowe rodzaje pętli

W C# (języku używanym w Unity) mamy trzy główne rodzaje pętli:

1. `while` – wykonuje kod, dopóki warunek jest spełniony.
2. `for` – wykonuje kod określoną liczbę razy.
3. `foreach` – wykonuje kod dla każdego elementu w kolekcji (np. liście).

Pętla `while`

Pętla `while` wykonuje kod **dopóki warunek jest prawdziwy**. Można ją porównać do pytania: "Czy mogę kontynuować?" – jeśli odpowiedź brzmi "tak", pętla działa dalej.

Składnia:

```
while (warunek)
{
    // Kod do wykonania
}
```

Przykład:

```
int licznik = 0;
while (licznik < 5)
{
    Debug.Log("Licznik: " + licznik);
    licznik++; // Zwiększ licznik o 1
}
```

Wyjaśnienie:

- Pętla działa, dopóki `licznik` jest mniejszy niż 5.
- W każdej iteracji wyświetla wartość `licznika` i zwiększa go o 1.
- Wynik:

```
Licznik: 0
Licznik: 1
Licznik: 2
```

```
Licznik: 3
Licznik: 4
```

Pętla `for`

Pętla `for` jest używana, gdy wiemy, **ile razy** chcemy wykonać kod. Składa się z trzech części:

1. **Inicjalizacja** – ustawienie początkowej wartości (np. `int i = 0`).
2. **Warunek** – określa, jak długo pętla ma działać (np. `i < 5`).
3. **Iteracja** – zmiana wartości po każdej pętli (np. `i++`).

Składnia:

```
for (inicjalizacja; warunek; iteracja)
{
  // Kod do wykonania
}
```

Przykład:

```
for (int i = 0; i < 5; i++)
{
  Debug.Log("Iteracja: " + i);
}
```

Wyjaśnienie:

- Pętla działa 5 razy (dla `i` od 0 do 4).
- W każdej iteracji wyświetla wartość `i`.
- Wynik:

```
Iteracja: 0
Iteracja: 1
Iteracja: 2
Iteracja: 3
Iteracja: 4
```

Pętla `foreach`

Pętla `foreach` jest używana do **przeglądania kolekcji** (np. list, tablic). Działa dla każdego elementu w kolekcji, bez konieczności ręcznego zarządzania indeksami.

Składnia:

```
foreach (typ element in kolekcja)
{
  // Kod do wykonania
}
```

Przykład:

```
string[] imiona = { "Anna", "Jan", "Maria" };
foreach (string imie in imiona)
{
  Debug.Log("Imię: " + imie);
}
```

Wyjaśnienie:

- Pętla przechodzi przez każdy element tablicy `imiona`.
- W każdej iteracji wyświetla wartość `imie`.
- Wynik:

```
Imię: Anna
Imię: Jan
Imię: Maria
```

Klasa `List<T>`

`List<T>` to specjalny typ kolekcji, który przechowuje elementy w formie listy. Można ją porównać do dynamicznej tablicy – można dodawać, usuwać i modyfikować elementy.

Jak działa `List<T>`?

- `T` oznacza typ danych, które przechowuje lista (np. `int`, `string`, `GameObject`).
- Lista automatycznie dostosowuje swój rozmiar, gdy dodajemy lub usuwamy elementy.

Przykład użycia:

```
using System.Collections.Generic; // Musimy dodać tę przestrzeń nazw

List<string> imiona = new List<string>();
imiona.Add("Anna");
imiona.Add("Jan");
imiona.Add("Maria");

foreach (string imie in imiona)
{
  Debug.Log("Imię: " + imie);
}
```

Wyjaśnienie:

- Tworzymy listę `imiona`, która przechowuje ciągi znaków (`string`).
- Dodajemy trzy imiona do listy za pomocą metody `Add`.
- Używamy pętli `foreach`, aby wyświetlić wszystkie imiona.

Przykład praktyczny w Unity

Użycie pętli i `List<T>` w skrypcie:

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

public class LoopExample : MonoBehaviour
{
  void Start()
  {
      // Przykład pętli for
      for (int i = 0; i < 3; i++)
      {
          Debug.Log("Pętla for: " + i);
      }

      // Przykład pętli foreach z List<T>
      List<string> przedmioty = new List<string>();
      przedmioty.Add("Miecz");
      przedmioty.Add("Tarcza");
      przedmioty.Add("Hełm");

      foreach (string przedmiot in przedmioty)
      {
          Debug.Log("Przedmiot: " + przedmiot);
      }

      // Przykład pętli while
      int zdrowie = 100;
      while (zdrowie > 0)
      {
          Debug.Log("Zdrowie: " + zdrowie);
          zdrowie -= 10; // Zmniejsz zdrowie o 10
      }
  }
}
```

Wyjaśnienie:

1. **Pętla `for`**:
    - Wykonuje się 3 razy, wyświetlając wartości od 0 do 2.
2. **Pętla `foreach`**:
    - Przechodzi przez listę `przedmioty` i wyświetla każdy przedmiot.
3. **Pętla `while`**:
    - Działa, dopóki `zdrowie` jest większe niż 0. W każdej iteracji zmniejsza `zdrowie` o 10.

Podsumowanie

Pętle:

- **`while`** – wykonuje kod, dopóki warunek jest spełniony.
- **`for`** – wykonuje kod określoną liczbę razy.
- **`foreach`** – wykonuje kod dla każdego elementu w kolekcji.

Klasa **`List<T>`**:

- To dynamiczna lista, która przechowuje elementy dowolnego typu.
- Można dodawać, usuwać i modyfikować elementy.

Pętle i listy są niezwykle przydatne w Unity, np. do zarządzania obiektami w grze, przeglądania listy przeciwników czy wykonywania powtarzalnych zadań. Zrozumienie ich działania to klucz do efektywnego programowania!