

Mikrobitowcy

Autorzy: Dawid Osuchowski, Filip Kłębczyk

Lekcja 6:

Micro:bit - pętla i animacje

Podczas zajęć uczniowie w praktyce zapoznają się z poszczególnymi funkcjami edytora. Kolejnym etapem będzie wykorzystanie wyświetlacza diodowego do wyświetlania obrazków, a następnie tworzenia na ich bazie animacji. Do tego drugiego celu zostanie wykorzystana pętla.

Cele lekcji:

Uczeń powinien:

- Znać pojęcia: pętla, zmienna, opóźnienie
- Wiedzieć, jak stworzyć pętlę nieskończoną w programie
- Wiedzieć, jak wprowadzać opóźnienia czasowe między wykonywaniem instrukcji kodu
- Wiedzieć, jak stworzyć animację z wykorzystaniem pętli
- Umieć zapisać swojej pracy do pliku
- Wiedzieć, jak weryfikować kodu z wykorzystaniem edytora Mu

Materiały pomocnicze:

- <http://microbit.org/>
- <https://bbcmicrobitmicropython.readthedocs.io/>
- <https://github.com/MicrobitPolska/FunWithMicrobit>

Pojęcia kluczowe:

→ obrazek → pętla → opóźnienie → zmienna → animacja

Czas realizacji: 45 min.

Metody pracy:

- wykład,
- dyskusja,
- ćwiczenia praktyczne przy komputerze,
- prezentowanie efektów pracy,
- szukanie rozwiązań na postawiony problem.

Treści programowe:

Podstawa programowa kształcenia ogólnego dla szkół podstawowych – II etap edukacyjny – klasy VII-VIII, informatyka:

- I. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń:
- 3) przedstawia sposoby reprezentowania w komputerze warto-

ści logicznych, liczb naturalnych (system binarny), znaków (kody ASCII) i tekstów;

- 4) rozwija znajomość algorytmów i wykonuje eksperymenty z algorytmami, korzystając z pomocy dydaktycznych lub dostępnego oprogramowania do demonstracji działania algorytmów;
- 5) prezentuje przykłady zastosowań informatyki w innych dziedzinach, w zakresie pojęć, obiektów oraz algorytmów.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. Uczeń:

- 1) projektuje, tworzy i testuje programy w procesie rozwiązywania problemów. W programach stosuje: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje oraz zmienne i tablice;
- 2) projektuje, tworzy i testuje oprogramowanie sterujące robotem lub innym obiektem na ekranie lub w rzeczywistości;
- 4) zapisuje efekty swojej pracy w różnych formatach i przygotowuje wydruki;

III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi. Uczeń:

- 2) rozwija umiejętności korzystania z różnych urządzeń do tworzenia elektronicznych wersji tekstów, obrazów, dźwięków, filmów i animacji;
- 3) poprawnie posługuje się terminologią związaną z informatyką i technologią.

IV. Rozwijanie kompetencji społecznych. Uczeń:

- 1) bierze udział w różnych formach współpracy, jak: programowanie w parach lub w zespole, realizacja projektów, uczestnictwo w zorganizowanej grupie uczących się, projektuje, tworzy i prezentuje efekty wspólnej pracy;

Wprowadzenie w tematykę i integracja grupy (10 min.)

Nauczyciel pyta uczniów, czy spotkali się z pojęciem pętli w programowaniu (jeśli nauczyciel przeprowadzał lekcje 1-4, przypomina zakres lekcji dotyczącej pętli). Jeżeli uczniowie nie potrafią wytłumaczyć działania pętli lub mają z tym problemy, nauczyciel wyjaśnia pojęcie oraz stara się je wytłumaczyć na łatwo przystępnych przykładach (np. sygnalizator świetlny dla przechodniów, który powtarza sekwencję trzech rodzajów świateł - czerwone, zielone, migające zielone).

Część zasadnicza (30 min.)

Pierwszy krok lekcji to stworzenie prostej animacji bijącego serca z wykorzystaniem **Image.HEART**, **Image.HEART_SMALL**, funkcji opóźnienia **sleep()** oraz pętli nieskończonej. Należy zwrócić uwagę uczniom, że funkcję **sleep()** (oznaczającą «czekaj») trzeba wywoływać po każdym obrazku, gdyż w przeciwnym razie obrazek od razu się zmieni na kolejny i nie będzie można go zobaczyć.

```
1 from microbit import *
2
3 while True:
4     display.show(Image.HEART)
5     sleep(500)
6     display.show(Image.HEART_SMALL)
7     sleep(500)
```

Nauczyciel tłumacząc działanie pętli **while** zwraca uwagę, że pętla wykonuje się, dopóki warunek jest prawdą (w przypadku powyższej pętli wykorzystywana jest wartość logiczna **True**, przez co warunek jest zawsze spełniony i pętla nigdy się nie kończy). Dodatkowo szczególną uwagę należy poświęcić na kwestię wcięć w powyższym kodzie. W języku Python, w przeciwieństwie do wielu innych języków programowania, wcięcia nie są tylko elementem stylu kodowania, ale służą przede wszystkim do definiowania bloków kodu np. w ramach pętli, instrukcji warunkowych, funkcji i klas. Wcięcie w prawo rozpoczyna nowy blok kodu, natomiast powrót z tego wcięcia na ten sam poziom kończy blok kodu (np. blok kodu pętli). Do tworzenia wcięć zalecane jest używanie 4 znaków spacji lub tabulatora, przy czym należy konsekwentnie trzymać się jednej konwencji w celu zachowania spójności i przejrzystości kodu. W przypadku zagnieżdżonych bloków kodu, odpowiednio dokładamy kolejne poziomy wcięcie w prawo, aby oznaczać kolejne poziomy zagnieżdżenia. Każdy nowy blok zaczynamy znakiem dwukropka.

```
Blok 1:
  Blok 2:
    Blok 3:
      Blok 2
    Blok 1
```

Uczniowie mogą modyfikować w dowolny sposób kod pierwszego programu, w szczególności zmniejszając i zwiększając opóźnienia, zmieniając obrazki, czy dodając kolejne obrazki do pętli (tworząc dłuższą i bardziej rozbudowaną animację).

Aby ułatwić dodawanie opóźnień przy kolejnych obrazkach warto zdefiniować sobie zmienną (w naszym programie **delay**). Umożliwi to zmianę opóźnienia dla całej animacji bez potrzeby zmiany czasów w każdym wywołaniu funkcji **sleep**.

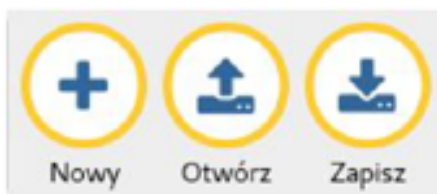
```
1 from microbit import *
2
3 delay = 500
4
5 while True:
6     display.show(Image.HEART)
7     sleep(delay)
8     display.show(Image.HEART_SMALL)
9     sleep(delay)
```

Pojęcie **zmiennej** możemy zobrazować uczniom jako oznaczone nazwą pudełko do przechowywania wartości, które potem możemy wykorzystywać i modyfikować (zmieniać wartość).

W ramach tej lekcji warto zachęcać uczniów do korzystania z funkcji **Sprawdź** edytora, przed wysłaniem programu na urządzenie.



Funkcja ta sprawdza nasz kod w poszukiwaniu błędów i informuje o nich bez konieczności flesztowania i odczytywania błędów na wyświetlaczu Micro:bita.



Na tym etapie warto też zachęcać uczniów do zapisywania różnych wersji swoich programów i pracy w wielu kartach. Warto zauważyć, że opcję "Zapisz jako" uzyskuje się poprzez dwukrotne kliknięcie w nazwę pliku na karcie, co jest w obecnej wersji edytora Mu nieoczywiste.

Pomysły zadań zachęcające uczniów do modyfikacji kodu i samodzielnego myślenia mogą obejmować np. dodanie kolejnych obrazków do animacji, zmiana opóźnień między wyświetlaniem kolejnych obrazków.

Podsumowanie (5 min.)

Na końcu lekcji uczniowie dyskutują, kto zrobił najciekawszą animację i zastanawiają się, jakie inne pomysły można zrealizować z wykorzystaniem wiedzy nabytej podczas zajęć.