*Informatyka w Edukacji, XV*
*UMK Toruń, 2018*

# PIĘKNO I RADOŚĆ PROGRAMOWANIA W SNAP! WARSZTAT

*Witold Kranas*
*Ośrodek Edukacji Informatycznej i Zastosowań Komputerów*
*02-006 Warszawa, Nowogrodzka 73*
*witek@oeiizk.waw.pl*

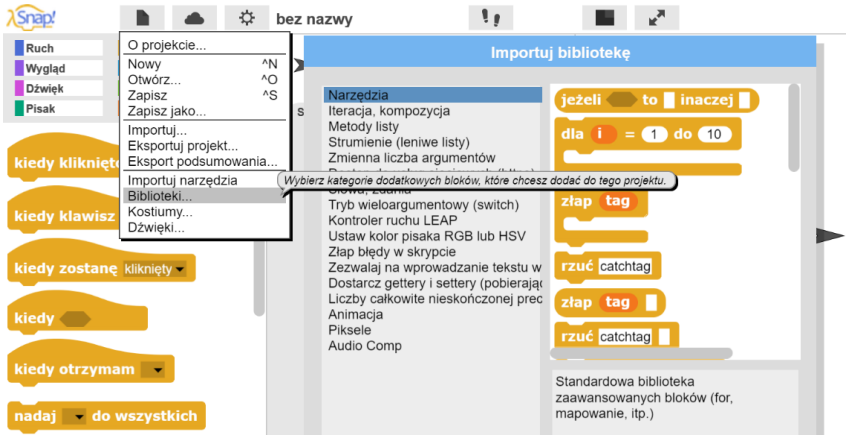
*Abstract. Some interesting examples of visual programming in Snap! taken from Advanced Placement Computer Science Principles course „The Beauty and Joy of Computing". Snap! is an environment developed at Berkeley University, combining Scratch interface with Scheme capabilities.*

## 1. O Snap!

Snap! jest klonem Scratcha stworzonym przez Briana Harveya i Jensa Möniga. W pierwszej wersji środowisko zostało nazwane BYOB (Build Your Own Blocks). Nazwa zawierała główny powód powstania środowiska – wprowadzenie możliwości tworzenia własnych bloków przez użytkownika gdyż w wersji Scratcha 1.4 nie było takiej możliwości. Twórcy zdecydowali się na zmianę nazwy ze względu na inne popularne rozwinięcie tego skrótu – bring your own bottle (przynieś swoją własną butelkę), dopisywane na zaproszeniach na imprezy. Tak piszą o tym w podręczniku: *Nazwa programu została zmieniona, ponieważ niektórzy nauczyciele nie mają poczucia humoru* [3].

Od stycznia 2018 roku mamy w pełni spolszczoną wersją środowiska w serwisie edukator.pl [6]. Oprócz polskich nazw bloków, jest też spolszczona pomoc do każdego bloku oraz podręcznik użytkownika. Scratch nie doczekał się jeszcze takiego pełnego spolszczenia. Snap! jest oprogramowany w HTML5 i można go uruchomiać w dowolnym systemie, na dowolnym urządzeniu. Tworzenie nowych bloków jest znacznie bardziej rozbudowane niż w Scratchu. Można wybrać paletę dla nowego bloku oraz określić, czy będzie on komendą (fragmentem układanki z wypustką), funkcją (blokiem zaokrąglonym), czy predykatem (blokiem sześciobocznym). Jest również możliwość prefiksowego wstawiania parametrów. Oprócz zestawu bloków niemal identycznego jak w Scratchu, Snap! zawiera kilkanaście rozszerzeń – bibliotek bloków dodatkowych. Jest tu również możliwość włączenia pracy krokowej – śledzenia wykonywania skryptów. Znacznie rozszerzona została

paleta bloków obsługujących listy, co za tym idzie, możliwości wykorzystywania tej struktury danych (lista, jako zmienna, listy list…) [4].




**Rysunek 1** Biblioteki dodatkowych bloków w Snap!

## 2. O kursie BJC

Amerykański kurs informatyki „The Beauty and Joy of Computing" (Piękno i radość programowania), nazywany tutaj w skrócie BJC jest przeznaczony dla uczniów w wieku 16-18 lat, czyli na poziomie naszego liceum. Należy on do kursów Advanced Placement, kończy się egzaminem i daje możliwość uzyskania punktów, liczących się przy przyjęciach na wyższe uczelnie [5].

Odwołanie do kursu znajduje się na głównej stronie Snap! (snap.berkeley.edu, bezpośredni adres: bjc.berkeley.edu) [1, 2]. Kurs zawiera szczegółowe materiały w formie umożliwiającej uczniowi samodzielną pracę. Kurs składa się z 6 części obowiązkowych:

1: Wprowadzenie do programowania
2: Abstrakcja
3: Przetwarzanie danych i listy
4: Jak działa Internet

Zadania badawcze (do zaliczenia)

5: Algorytmy i symulacje

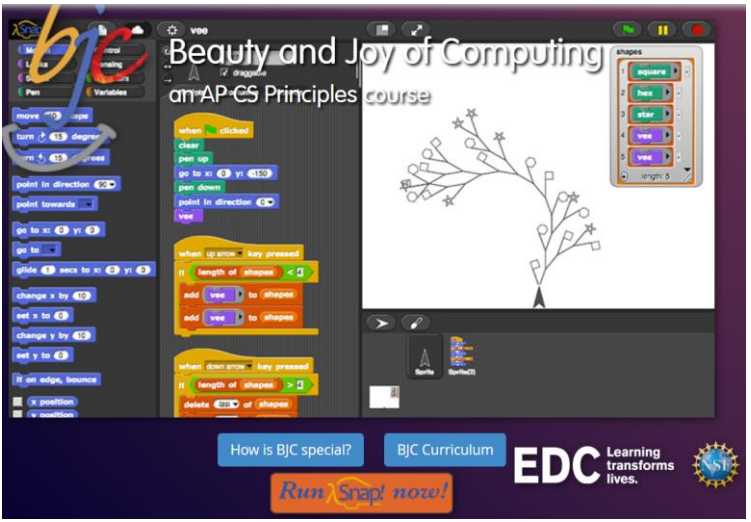
Zadania twórcze (do zaliczenia)

6: Jak działają komputery

oraz dwóch części dodatkowych, do przerobienia po egzaminie:

7: Fraktale i rekurencja
8: Funkcje rekurencyjne

Autorzy piszą we wstępie (tłumaczenie autora): *W tym kursie będziesz tworzyć programy używając języka programowania Snap!, poznasz największe idee informatyki, będziesz pracować twórczo, będziesz dyskutować społeczne implikacje informatyki i myśleć poważnie o tym, jak osobiście promować korzyści i zmniejszać możliwe szkody związane z rozwojem informatyki* [1].



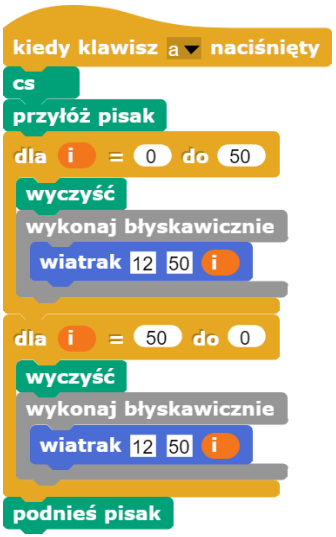

**Rysunek 2** Ekran startowy kursu BJC

## 3. Wielokąt, wiatrak i pętla dla

Pierwszy przykład nawiązuje do programowania w Logo. Jak narysować wielokąt? Możemy utworzyć nowy blok np. w palecie Ruch i w edytorze bloków zbudować skrypt komendy (czyli nowy blok będzie miał kolor niebieski i wypustkę układanki). Dodajmy wielokątowi trzeci parametr – **wystaje** i blok, który spowoduje, że odcinek o takiej długości będzie wystawać z każdego boku. Nazwijmy ten twór wiatrakiem. Wielokąt uzyskamy w skrajnym przypadku, gdy parametr **wystaje** będzie równy zeru. A jaką figurę uzyskamy, gdy będzie on równy rozmiarowi?


**Rysunek 3** Skrypt bloku wiatrak

Teraz z menu **Plik** zaimportujmy narzędzia. Na dole w palecie **Kontrola** pojawi się blok **dla** – wygodna pętla iteracyjna. W tej samej palecie znajdziemy nie występujący w Scratchu blok **wykonaj błyskawicznie**, powodujący włączenie trybu turbo tylko dla bloków w nim zawartych. Możemy już zbudować skrypt z animacją wiatraka, pokazującą zmiany rysunku, gdy zmienia się wartość parametru **wystaje**.

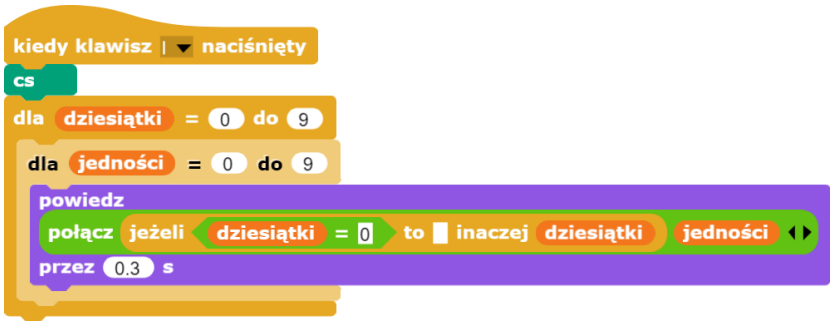



**Rysunek 4** Skrypt animacji wiatraka

Za pomocą pętli **dla** można wykonać wiele interesujących rysunków. Oto dwa przykłady do wypróbowania (Rys. 5).

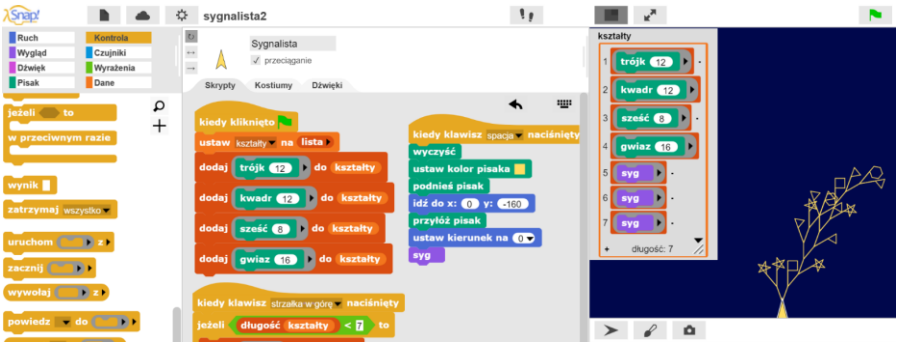

**Rysunek 5** Jakie obrazki powstaną?

Jeszcze jeden przykład wykorzystania pętli **dla** – odliczanie od 0 do 99. W skrypcie wykorzystujemy funkcję **jeżeli**, która pozawala nie wypisywać zera dziesiątek na początku odliczania.




**Rysunek 6** Skrypt odliczania od 0 do 99

## 4. Sygnały z listy




**Rysunek 7** Projekt Sygnalista

Kolejny przykład został przygotowany przez autorów Snap!, aby pokazać możliwości tego środowiska [2]. Zaczynamy niewinnie od kliku nowych bloków rysujących proste figury.


**Rysunek 8** Trójkąt i gwiazdka z parametrem

Teraz tworzymy zmienną kształty i nadajemy jej wartość pustej listy. Następnie na tej liście umieszczamy utworzone bloki tak, aby można je było wykonać, czyli w szarej obwiedni, która znajduje się w palecie **Wyrażenia**. Można to zrobić za pomocą bloku **dodaj … do …**, tak jak w skrypcie zielonej flagi na Rys. 9.


**Rysunek 9** Skrypt zielonej flagi tworzący listę bloków do wykonania

Pora na utworzenie głównego bloku, który będzie rysował sygnał złożony z dwóch flag o losowych kształtach.


**Rysunek 10** Blok rysujący sygnał

Uruchomienie tego bloku spowoduje narysowanie dwóch flag. Ale jak można uzyskać obrazek taki jak na Rys. 7, gdzie tych flag jest znacznie więcej?

Wystarczy umieścić na liście kilka bloków **syg** w szarych obwiedniach. Dodawać je i usuwać można za np. pomocą skryptów pokazanych Rys. 11.


**Rysunek 11** Dodawanie bloku syg do listy i usuwanie go

No i mamy rekurencję bez przypadku bazowego i co za tym idzie, bez warunku zatrzymania!

## 5. Sufit

Na koniec warto zauważyć, że Snap! nie jest zabawką dla dzieci. Został on stworzony, aby pomóc w nauce programowania rozumianego jako tworzenie i implementowanie algorytmów. I choć podłoga, podobnie jak w Scratchu, jest dość nisko, to sufit jest znacznie wyżej. W naszych realiach można go sensownie używać na poziomie klas 4-6, a w sposób pełniejszy na poziomie 7-8, choć można sobie wyobrazić kurs informatyki dla liceum bazujący na Snapie, jako głównym środowisku programistycznym [5]. Z pewnością warto spróbować pracy w tym środowisku.

## Literatura

1. Głowna strona kursu BJC, bjc.berkeley.edu, ostatni dostęp 11.06.2018 roku.
2. Główna strona Snap (Berkeley), snap.berkeley.edu, ostatni dostęp 11.06.2018 roku.
3. Harvey B., Mönig J., *Snap! Podręcznik użytkownika*, do pobrania ze strony Snap!, w wersji polskiej ze strony Snap! w serwisie Edukator: www.edukator.pl/tik_edukator/Snap/snap.html, ostatni dostęp 11.06.2018 roku.
4. Kranas W., *Piękno i radość programowania w Snap!*, W cyfrowej szkole, Nr 2/2018, OEliZK, 2018 (w przygotowaniu).
5. Kranas W., *Snap! cudowne dziecko Scratcha*, W cyfrowej szkole, Nr 1/2018, OEliZK, 2018.
6. Polski serwer Snap! w serwisie Edukator, www.edukator.pl/tik_edukator/Snap/snap.html, ostatni dostęp 11.06.2018 roku.