PROJEKTUJEMY GRY, TABLICE W PROCESSINGU

Agnieszka Borowiecka, Maciej Borowiecki
Ośrodek Edukacji Informatycznej i Zastosowań Komputerów
02-026 Warszawa, ul. Raszyńska 8/10
{agnieszka.borowiecka, maciej.borowiecki}@oeiizk.waw.pl

Abstract. While learning programming it is worthwhile to find the right examples to make students aware of their actions and to make correct intuitions. The world of simple logic and arcade games seems to be an inexhaustible source from which we should draw inspiration for interesting activities for students. Students will better remember how to use arrays, if they use that structure for prepare their own game.

1. Wstęp

Od roku szkolnego 2017/2018 nowa podstawa programowa zakłada naukę programowania na wszystkich etapach edukacyjnych. Nie będą to łatwe zagadnienia dla każdego ucznia i powinniśmy zastanowić się, w jaki sposób zachęcić uczniów do wytężonej pracy. Jednym ze sposobów może być wykorzystanie elementów rywalizacji i tworzenie różnorodnych gier. Uczniowie niejednokrotnie sami zaproponują modyfikacje i udoskonalanie opracowywanych przez siebie projektów, poszerzając tym samym wiedzę informatyczną. Będą szukali lepszych rozwiązań algorytmicznych, dążąc do tego, by tworzone gry były ciekawe i efektywne w działaniu. Łatwiej także będzie wdrożyć ich do pracy grupowej i wspomóc rozwijanie indywidualnych zdolności, zachęcając np. do opracowania grafiki gry, pisania funkcji łatwych do wykorzystania przez innych programistów, prawidłowego komentowania kodu, itd.

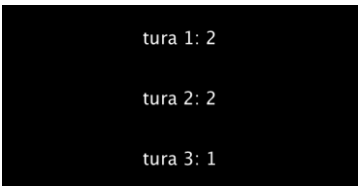
Równie istotnym zagadnieniem jest właściwy dobór języka programowania i środowiska, w którym będziemy pracować. Z młodszymi uczniami możemy tworzyć ciekawe projekty w Scratchu, jednak już w starszych klasach szkoły podstawowej czy w szkole ponadpodstawowej należałoby wybrać tekstowy język programowania. Zachęcamy do korzystania ze środowiska Processing. Jest to wygodne środowisko dostępne na różne systemy operacyjne, umożliwiające także tworzenie aplikacji mobilnych dla systemu Android. Język, jakim się posługujemy,

to uproszczona wersja Javy. W Processingu łatwo można uzyskać ciekawe efekty wizualne oraz zaprogramować interakcję z użytkownikiem.

Skupimy się na przygotowaniu kilku gier o różnym stopniu złożoności. Każda z nich będzie wykorzystywała złożone struktury danych – tablice, m.in. do przechowywania wyników osiągniętych przez gracza, pamiętania położenia ruchomych obiektów lub danych związanych z wyglądem planszy wyświetlanej podczas gry. Proponowane projekty można realizować ze starszymi uczniami, przygotowując z nimi prostsze lub bardziej złożone wersje. Każdą z gier łatwo rozbudowywać i ulepszyć, mogą one także stanowić podstawę do tworzenia własnych projektów uczniowskich.

2. Klikamy na czas

Zaczniemy od gry sprawdzającej refleks i sprawność naszych uczniów w posługiwaniu się myszką. Na ekranie wyświetlany jest upływający czas: kolejne sekundy i dziesiąte części sekundy. Użytkownik powinien klikać myszką w chwili, gdy wyświetlana jest pełna sekunda. Gra składa się z kilku prób, każda próba trwa 10 sekund – czyli gracz może w niej uzyskać maksymalnie 10 punktów. Wyniki kolejnych prób są pamiętane w tablicy, a po zakończeniu gry wyświetlane na ekranie.



Rysunek 1 Wyniki gry wyświetlane dla trzech prób

Do przechowywania wyników w pojedynczej turze gry wykorzystujemy zmienną **punkty**, wyniki kolejnych tur zapisujemy w tablicy **wyniki**. Pomiędzy turami będzie wyświetlana plansza informacyjna, dlatego dodamy do projektu zmienną logiczną **gra** informującą program o tym, czy w danej chwili wyświetlać informację dla gracza, czy upływający czas. Zmienna **tura** będzie przechowywać numer kolejnej próby, a zmienna **ms** zmieniające się dziesiąte sekundy.

```
int ms;
boolean gra = false;
int punkty;
int tura = 1;
int[] wyniki;
```

Rysunek 2 Deklarowane zmienne

Przygotowując grę z uczniami możemy zacząć od samego wyświetlania czasu, następnie dodać zdarzenia związane z obsługą myszki. Jeśli na ekranie wyświetlany jest czas i gracz kliknął myszką, to po sprawdzeniu, że wyświetlana jest pełna sekunda, zostanie powiększona liczba zdobytych punktów.

Rysunek 3 Druga sekunda od rozpoczęcia próby, przy kliknięciu zostaną naliczone punkty

Wprowadzając tablicę do przechowywania wyników gry, pokazujemy uczniom sposób inicjowania tablicy, odwoływania się do jej poszczególnych elementów, zmieniania danych przechowywanych w tablicy. Zwracamy uwagę na sposób indeksowania poszczególnych pozycji w tablicy – dla tablicy n-elementowej pierwszy element ma indeks 0, ostatni n – 1. Ostrzegamy przed próbami odwołania się do elementów o indeksach większych niż rozmiar tablicy, czyli leżących poza tablicą.

```
// tworzenie 3-elementowej tablicy
// liczb całkowitych
wyniki = new int[3];

// zapamiętanie wyników tury
wyniki[tura-1] = punkty;

// pętla wypisująca wyniki 3 prób
for (int i = 0; i<3; i++)
      text("tura "+(i+1)+": "+wyniki[i],
            0, i*height/3, width, height/3);
```

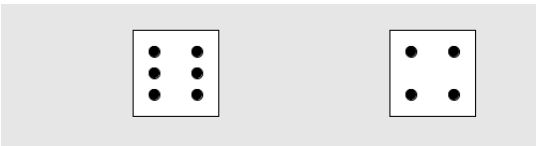
Rysunek 4 Przykładowe instrukcje związane z tablicą wyniki

Tworząc grę warto zacząć od mniejszych wartości danych (1-2 próby, czas próby np. 3 s.), by łatwiej było testować program. Można przeprowadzić z uczniami dyskusję na temat, jak najlepiej definiować rozmiar tablicy i odwoływać się do jej elementów, by można było w prosty sposób zmienić liczbę wykonywanych prób.

3. Rzucamy dwiema kostkami

Tablicę wyników możemy także wykorzystać do badania różnego typu zależności matematycznych. Proponujemy zastanowić się z uczniami nad zagadnieniem losowości. Jak zachowują się sumy oczek uzyskiwane przez gracza przy rzucie dwoma kostkami? Przygotowujemy z uczniami prostą symulację, w której na ekra-

nie pojawiają się dwie kostki o wylosowanej liczbie oczek. Możemy dodać element ruchu – kostki uciekają przed graczem, a jego zadaniem jest je złapać. Po złapaniu obu kostek sumowana jest liczba oczek i zwiększana odpowiednia pozycja w tablicy. Poruszające się kostki powinny pozostawać widoczne w oknie aplikacji (odbijanie się od krawędzi), mogą także się „toczyć", zanim wyświetlą wylosowaną liczbę oczek. Po zakończeniu gry i wypisaniu wyników, analizujemy je z uczniami.



Rysunek 5 Wylosowane kostki

Przygotowując tablicę do przechowywania informacji o uzyskanej sumie oczek, zastanawiamy się z uczniami, jaki powinien być jej zakres. Analizujemy możliwe do uzyskania sumy oczek na dwóch kostkach, zwracamy uwagę na sposób odwołania do elementów tablicy w Processingu. Podejmujemy decyzję, jak przechowywać uzyskane w rundzie gry wyniki. Możemy także wprowadzić funkcje zapisujące informacje w pliku lub wyświetlić zawartość tablicy w postaci wykresu słupkowego.

4. Złap wszystkie kółka

Wśród popularnych na urządzeniach mobilnych gier są takie, w których należy klikać w przesuwające się po ekranie obiekty. Przygotujemy grę, w której zadaniem będzie złapanie jak największej liczby spadających kółek. Po złapaniu każdego koła zwiększa się prędkość przemieszczania pozostałych kół. Jeśli obiektu nie udało się kliknąć, zanim w całości opuścił okno aplikacji, gracz traci życie. Kółka, które zostały złapane lub zniknęły z obszaru okna mają losowaną nową pozycję i ponownie pojawiają się na ekranie. Gra kończy się, gdy gracz utracił wszystkie życia.

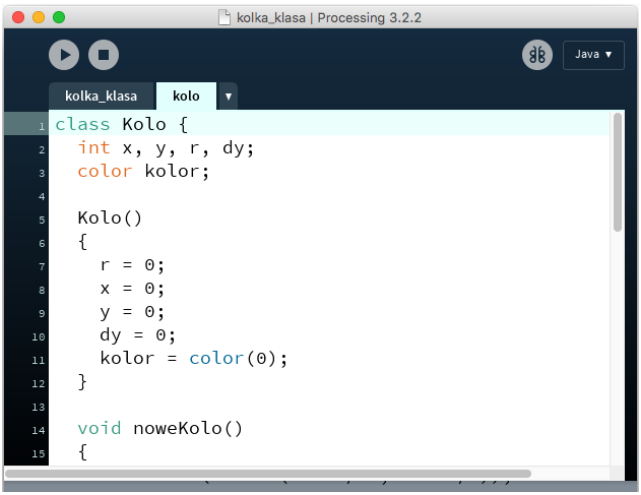
Rysunek 6 Spadające kółka – koniec gry

Projektując grę zauważamy, że jest ona trudniejsza, gdy na ekranie pojawia się większa liczba obiektów. Pamiętanie położenia wszystkich poruszających się obiektów na pojedynczych zmiennych jest niewygodne, w naturalny sposób wprowadzamy tablice do przechowywania współrzędnych x i y środków kół. Możemy także skorzystać z tablicy dwuwymiarowej. W najprostszej wersji gry wszystkie koła mają jednakową średnicę i poruszają się z taką samą prędkością. Interesującym zagadnieniem będzie sprawdzanie, które z kół zostało kliknięte przez gracza i odpowiednie zwiększenie liczby punktów.

```
void mousePressed()
{
  for (int i = 0; i < ile; i++)
    if (dist(kx[i], ky[i], mouseX, mouseY) < sz/2)
    {
      noweKolo(i);
      punkty++;
    }
}
```

Rysunek 7 Sprawdzamy, które koła zostały kliknięte przez gracza

Kolejnym etapem pracy z programem może być zróżnicowanie prędkości kół, ich rozmiarów i kolorów. W tym celu, zamiast definiować kolejne tablice do przechowywania potrzebnych danych, warto dodać do projektu klasę **Kolo**.



Rysunek 8 Dodawanie klasy Kolo do projektu

Po zdefiniowaniu wszystkich niezbędnych metod klasy **Kolo**, potrzebna będzie tablica do przechowywania obiektów tej klasy. Definiujemy ją w analogiczny sposób, jak tablice liczb czy wartości logicznych.

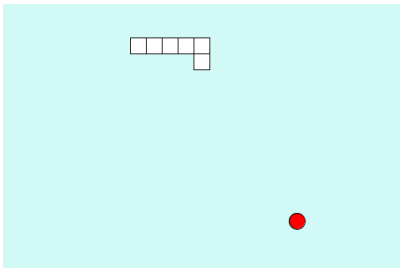
```
Kolo[] dane;

dane = new Kolo[ile];
```

Rysunek 9 Tablica przechowująca informacje o kołach

Uczniowie mogą zastanowić się, jakie metody należy dodać do klasy, by gra była bardziej interesująca. Dyskutujemy z nimi, co należałoby zmienić w programie, by po zniknięciu koła z okna aplikacji pojawiało się ono ponownie nie tylko na nowej pozycji, ale w innym kolorze i wielkości.

5. Snake, snake

Ostatnią grą, jaką chcemy przedstawić, jest popularna gra Snake. Pierwsza wersja została wydana w połowie lat siedemdziesiątych XX wieku. W grze poruszamy się postacią podobną do węża, zbierając jedzenie i różne przedmioty. Po zjedzeniu wąż się wydłuża o kolejne segmenty. Gracz zmienia kierunek ruchu węża za pomocą klawiszy ze strzałkami, pilnując by nie uderzyć w otaczające planszę ściany, ani nie trafić na część ciała węża. Opracowana przez nas wersja gry ograniczy pole ruchu węża do okna aplikacji, do zjedzenia będzie jeden przedmiot pojawiający się na losowej pozycji. Dla ułatwienia poszczególne segmenty ciała węża będziemy przedstawiali za pomocą kwadratów, a jedzeniem będzie kółko.



Rysunek 10 Tablica przechowująca informacje o kołach

Podstawowa trudność w programowaniu gry Snake polega na prawidłowym zdefiniowaniu struktur danych opisujących poruszającego się węża. Podobnie, jak w przypadku gry w łapanie spadających kółek, możemy zdefiniować dwie tablice **ogonX** i **ogonY** do pamiętania współrzędnych poszczególnych segmentów (a dokładnie lewego górnego narożnika rysowanego kwadratu), lub skorzystać z tablicy

dwuwymiarowej. Przyjmujemy długość boku segmentu np. 20, ustalamy także rozmiary okna aplikacji będące wielokrotnością tej liczby. Podczas trwania gry zmienia się długość węża, a tym samym rozmiar tablic potrzebnych do pamiętania jego położenia. Jednocześnie przesuwanie węża o jedną pozycję w danym kierunku powoduje, że pierwsza pozycja w tablicy zostaje zapomniana, wszystkie pozostałe należy „przesunąć" o jedną pozycję w lewo, a na ostatniej wstawić nowe współrzędne „głowy" węża. Gdy wąż coś zje, zamiast przeliczać zawartość tablicy dodajemy do niej nowe współrzędne, wydłużając go o jeden segment.

```
void przeliczOgon()
{
  for (int i=0; i<dl-1; i++)
  {
    ogonX[i] = ogonX[i+1];
    ogonY[i] = ogonY[i+1];
  }
  ogonX[dl-1] = x;
  ogonY[dl-1] = y;
}
```

Rysunek 11 Zmiana elementów tablic podczas przesuwania węża

Możemy zadeklarować tablice składające się na przykład ze 100 lub więcej elementów, bowiem jest mało prawdopodobne, by gracz uzyskał większą długość węża. Wtedy konieczne będzie wprowadzenie pomocniczej zmiennej pamiętającej, z ilu segmentów składa się w danym momencie wąż, czyli jaki fragment tablicy jest używany.

Problem zmiany długości węża możemy także rozwiązać wykorzystując dostępną w Processingu funkcję **append**. Pozwala ona utworzyć nową większą tablicę złożoną z tablicy już istniejącej, z dodanym na końcu nowym elementem. Początkowo w tablicy pamiętana jest jedynie głowa, w trakcie działania gry zwiększamy rozmiar tablicy.

```
void nowyOgon()
{
  ogonX = append(ogonX, jx);
  ogonY = append(ogonY, jy);
}
```

Rysunek 12 Wydłużamy węża

Do sprawdzania, czy należy wydłużyć węża, warto napisać pomocniczą funkcję logiczną, sprawdzającą odległość między głową węża a pozycją kółka symbolizującego jedzenie.

```
boolean zjadl()
{
  return dist(x, y, jx, jy) < 1;
}
```

Rysunek 13 Sprawdzanie, czy wąż spotkał jedzenie

Łatwo sobie wyobrazić kolejne rozszerzenia gry Snake. Po dodaniu warunków kończących grę uczniowie mogą zastanowić się nad zwiększeniem liczby pokarmu dla węża, dodaniem przeszkód powodujących jego skracanie, umożliwieniem rozpoczęcia nowej gry przez naciśnięcie klawisza lub rysowaniem wielobarwnego węża. Możemy także przedyskutować zmiany, jakie należałoby wprowadzić, by móc uruchamiać grę na urządzeniu mobilnym.

6. Podsumowanie

Ucząc wykorzystania struktur danych warto jest dobrać właściwe przykłady, by uzmysłowić uczniom ich działanie i wyrobić prawidłowe intuicje. Świat prostych gier logicznych i zręcznościowych wydaje się niewyczerpanym źródłem inspiracji, z którego powinniśmy czerpać pomysły na interesujące zajęcia dla uczniów. Na pewno lepiej zapamiętają, jak posługiwać się tablicą, do czego można ja wykorzystać w programie, z czym się wiąże próba wykroczenia poza jej zakres – jeśli przygotują własną grę, którą będą mogli pochwalić się wśród kolegów.

Literatura

1. Greenberg I., Xu D., Kumar D., *Processing Creative Coding and Generative Art in Processing 2*, APress Media, 2013.
2. Reas C., Fry B., *Make: Getting Started with Processing, Second Edition*, Maker Media, 2015.
3. Reas C., Fry B., *Processing: A Programming Handbook for Visual Designers, Second Edition*, The MIT Press, 2014.
4. Strona domowa Processingu, http://processing.org, ostatni dostęp 24.05.2017 roku.
5. Vantomme J., *Processing 2: Creative Programming Cookbook*, Packt Publishing, Birmingham 2012.