Informatyka w Edukacji, XV
UMK Toruń, 2018

W POSZUKIWANIU WYJŚCIA ZABAWA Z PROCESSINGIEM

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. Providing lessons for programming at school raises some problems. The clue is to prepare interesting tasks that engage students and show them appliance of new knowledge in examples. Additionally, a reference to the world of games can raise student's motivation.

1. Wstęp

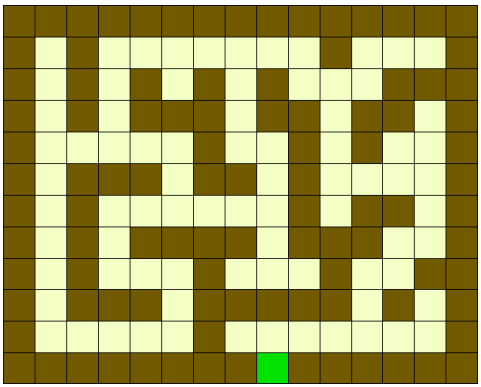
Obowiązująca od roku szkolnego 2017/2018 podstawa programowa zakłada naukę programowania na wszystkich etapach edukacyjnych. Uczniowie powinni rozwiązywać problemy i podejmować decyzje z wykorzystaniem komputera, stosować podejście algorytmiczne. Wielokrotnie podkreślane jest, że nie należy uczyć konkretnego narzędzia, ale korzystając z narzędzia rozwiązywać problemy. Zastanówmy się, jak powinniśmy uczyć programowania, wprowadzać kolejne konstrukcje programistyczne, by działo się to w sposób niewymuszony i wynikało z konkretnej potrzeby ucznia. Wydaje się, że najważniejszy jest właściwy dobór zadania, które chcemy rozwiązać. Warto jest sformułować je w postaci atrakcyjnej dla uczniów, na przykład nawiązując do popularnych gier komputerowych. Wszystkie pojęcia nagle okażą się proste i naturalne, a uczniowie sami będą wyszukiwać potrzebne informacje i dopytywać się o bardziej złożone konstrukcje.

Do realizacji opisanego przez nas projektu proponujemy wykorzystać Processing, wygodne środowisko dostępne na różne systemy operacyjne. Język, jakim będziemy się posługiwać to uproszczona wersja Javy. W Processingu łatwo można uzyskać animację oraz zaprogramować interakcję z użytkownikiem.

2. Projektujemy grę – labirynt

Jeden z klasycznych motywów gier komputerowych polega na wyprowadzaniu postaci z labiryntu. Przygotujemy prosty projekt, w którym będziemy sterować robo-

tem za pomocą wybranych klawiszy na klawiaturze. Z bardziej zaawansowanymi uczniami możemy przygotować wersję z robotem próbującym automatycznie odnaleźć drogę do wyjścia. Zaczynamy od przygotowania w edytorze graficznym planszy labiryntu. Dla ułatwienia możemy przyjąć, że labirynt jest zbudowany z kwadratów o boku długości 40 i w całości otoczony ścianą, za wyjątkiem wyjścia. Kolory ścian i ścieżek będziemy wykorzystywać w kodzie projektu, dlatego rysunek najlepiej zapisać w formacie bmp lub png.



Rysunek 1 Przykładowy labirynt

Pracę z projektem zaczynamy od wyświetlenia planszy labiryntu w oknie aplikacji i narysowania robota. Możemy przyjąć, że postać robota będzie symbolizowana za pomocą niewielkiego kółka. Uczniowie skupią się na prawidłowym przemieszczaniu robota w labiryncie, a nie na sposobie odtwarzania kolejnych klatek animacji. Do wyświetlenia labiryntu niezbędne jest zadeklarowanie zmiennej, wczytanie planszy do pamięci i wyświetlenie jej na ekranie. Należy także odpowiednio dobrać wielkość i położenie kółka symbolizującego postać robota.

```
PImage plansza;

void setup()
{
  size(600, 480);
  plansza = loadImage("plansza.png");
}

void draw()
{
  image(plansza, 0, 0);
  ellipse(60, 60, 30, 30);
}
```

Rysunek 2 Instrukcje wyświetlające labirynt i robota

W kolejnym kroku dodamy do projektu możliwość sterowania postacią za pomocą klawiatury. Uczniowie mogą zaproponować klawisze do wykorzystania – np. strzałki lub WSAD. Zauważamy, że skoro położenie robota będzie się zmieniać, niezbędne jest dodanie do projektu zmiennych określających jego położenie – środek koła. Dla ułatwienia zaprojektowany labirynt jest w całości otoczony ścianą. Dzięki temu nie musimy zastanawiać się, czy przesuwając robota nie opuszczamy labiryntu (badanie warunków brzegowych).

Do projektu dodajemy funkcję **keyPressed()** wywoływaną automatycznie przy naciśnięciu klawisza na klawiaturze. Znaki wprowadzane z klawiatury są pamiętane na zmiennej **key** (znaki alfanumeryczne) lub zmiennej **keyCode** (znaki specjalne).

```
void keyPressed()
{
  if (key == 'w') y = y - 40;
  if (key == 's') y = y + 40;
  if (key == 'a') x = x - 40;
  if (key == 'd') x = x + 40;
}
```

Rysunek 3 Sterowanie robotem za pomocą klawiszy WSAD

Uczniowie musza pamiętać o zainicjowaniu zmiennych opisujących robota, ustaleniu, o ile będzie się on przesuwał po naciśnięciu klawiszy i w którą stronę. Pewien problem może wynikać z faktu, że układ współrzędnych w Processingu jest odwrócony – współrzędna **y** rośnie w dół, a środek układu współrzędnych znajduje się w lewym górnym rogu okna aplikacji.

Po uruchomieniu programu zauważamy, że nie działa tak, jak oczekiwaliśmy – robot jak duch przenika przez ściany. Konieczne jest dodanie warunku sprawdzającego, czy robot może wykonać ruch w kierunku wskazanym przez gracza. Modyfikujemy kod dodając funkcję **ruch()**, która sprawdza czy można przejść na pole określone przez parametry, jeśli tak, to zmienia położenie robota.

```
void keyPressed() {
  if (key == 'w') ruch(x, y-40);
  if (key == 's') ruch(x, y+40);
  if (key == 'a') ruch(x-40, y);
  if (key == 'd') ruch(x+40, y);
}

void ruch(int a, int b)
{
  if (plansza.get(a, b) != mur)
  {
    x = a;
    y = b;
  }
}
```

Rysunek 4 Funkcja wyliczająca nowe współrzędne robota

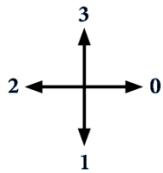
Ostatnią modyfikacją będzie dodanie warunku, badającego czy udało się dotrzeć do celu – postać robota znajduje się w kwadracie symbolizującym wyjście. Możemy wyświetlić na ekranie tekst z gratulacjami dla gracza.

3. Automatyczne wychodzenie z labiryntu

Możemy omówić z uczniami sposób, w jaki robot mógłby samodzielnie wyszukać drogę do wyjścia z labiryntu. Proponujemy zaimplementować algorytm, w którym poruszamy się po labiryncie cały czas trzymając się lewą ręką ściany:

```
dopóki nie doszedłeś do wyjścia
  jeśli po lewej stronie jest przejście to skręć w lewo
  jeśli możesz iść przed siebie to idź
    w przeciwnym razie skręć w prawo
```

Zauważamy, że robot będzie zachowywał się w inny sposób niż dotychczas – zamiast czekać na nasze polecenie „Przesuń się o jeden krok w określonym kierunku", musi pamiętać kierunek, w którym podąża. Dodajemy zmienną całkowitą **kierunek** i przypisujemy wartości liczbowe do kierunków, w jakich może poruszać się robot:



Rysunek 5 Kierunki ruchu robota

Następnie modyfikujemy funkcję **ruch()** sterującą ruchem robota. Najpierw sprawdzamy, czy możliwy jest ruch w lewą stronę. Jeśli tak, to odpowiednio zmieniamy wartość zmiennej **kierunek**. Do wyliczeń wykorzystujemy resztę z dzielenia przez 4 (cztery możliwe kierunki ruchu) oraz fakt, że obrotowi w lewo odpowiadają trzy obroty w prawą stronę: **(kierunek + 3) % 4**. Następnie próbujemy przesunąć robota w kierunku, w którym patrzy. Gdy ruch nie jest możliwy, obracamy robota w prawo.

Animację ruchu robota rozpoczynamy po naciśnięciu klawisza na klawiaturze. Ponowne naciśnięcia tego klawisza zatrzymują i wznawiają animację. By ułatwić obserwowanie kolejnych ruchów robota, można zmniejszyć szybkość animacji za pomocą funkcji **frameRate()**.

Po sprawdzeniu działania projektu warto omówić z uczniami zachowanie robota. Możemy zadać następujące pytania:

- Jakie poprawki należałoby wprowadzić w programie, by robot trzymał się ściany po prawej stronie?
- Czy miejsce startu robota ma znaczenie?
- Czy robot zawsze trafi do wyjścia, a jeśli nie to dlaczego?
- Czy zaproponowany algorytm jest zatem poprawny?

Z uczniami zdolnymi ten projekt może być wstępem do poszukiwania algorytmu znajdującego najkrótszą drogę wyjścia z labiryntu (a ogólniej, najkrótszej ścieżki pomiędzy dwoma polami).

4. Podsumowanie

W opisanym projekcie uczniowie zapoznają się z zasadami projektowania prostych gier, mogą przy tym wykorzystać swoją wiedzę praktyczną o różnego typu grach komputerowych. Przygotowują interfejs użytkownika najpierw planując jego logikę (jak sterujemy robotem za pomocą klawiszy), a następnie odpowiednio go programują. W sposób naturalny wprowadzamy pojęcie zmiennej, instrukcję warunkową, definiowanie własnych funkcji. Każdy uczeń może szybko uzyskać satysfakcjonujący go efekt, a sam projekt może stanowić punkt wyjścia dla wielu interesujących modyfikacji.

Literatura

1. Greenberg I., Xu D., Kumar D., *Processing Creative Coding and Generative Art in Processing 2*, APress Media, 2013.
2. Nyhoff J. L., Nyhoff L. R., *Processing: An Introduction to Programming,* CRC Press, 2017.
3. Reas C., Fry B., *Make: Getting Started with Processing, Second Edition,* Maker Media, 2015.
4. Reas C., Fry B., *Processing: A Programming Handbook for Visual Designers,* Second Edition, The MIT Press, 2014.
5. Strona domowa Processingu, http://processing.org, ostatni dostęp 10.06.2018 roku.