DODATEK SPECJALNY NR 26

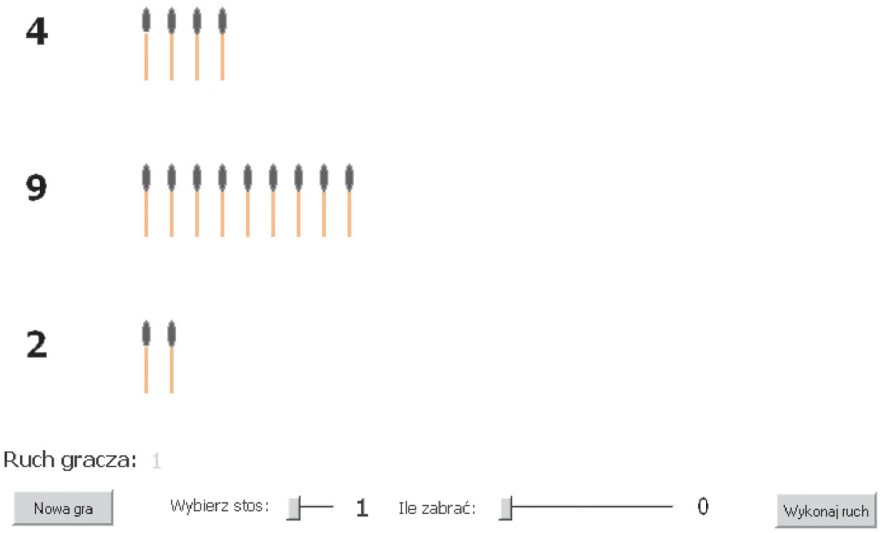
Nauka programowania od PODSTAW

W niniejszej części „Nauki programowania od podstaw" przedstawimy wersję starej gry chińskiej w zbieranie kamieni, znanej pod nazwą NIM. Podobnie, jak w omawianej w tym numerze Świata Matematyki Grze żetonami istnieje strategia wygrywająca. W niniejszym artykule zostanie przedstawiona wersja gry dla dwóch graczy, którzy na przemian wykonują ruchy. Rola komputera w tym przypadku ograniczy się więc do sterowania grą.

ZASADY GRY NIM

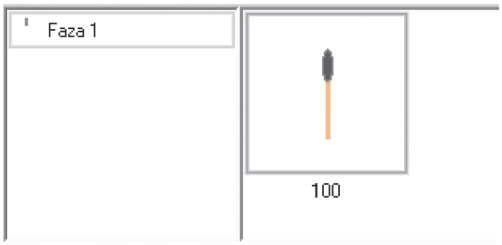
Gra polega na zbieraniu kamieni (pionków) z kilku grup (stosów) w ten sposób, że można zebrać dowolną liczbę kamieni, ale w jednym ruchu tylko z jednej grupy. Wygrywa ten gracz, który weźmie ostatni kamień lub kamienie. W grze NIM, rolę kamieni lub pionków pełnią często zapałki, w związku tym w implementacji gry użyjemy także zapałek.

W grze z dwoma stosami dosyć łatwo jest podać strategię wygrywającą. Rozpoczynający gracz powinien zebrać ze stosu z większą liczbą zapałek tyle, aby oba stosy składały się z tej samej liczby zapałek. Jeśli na starcie oba stosy są równoliczne, to gracz rozpoczynający grę przegra – strategię wygrywającą posiada drugi gracz, o ile nie popełni błędu. Ciekawsza gra staje się przy większej liczbie stosów, przedstawimy implementację dla trzech.



PRZYGOTOWUJEMY OBRAZ DO GRY NIM

W naszej wersji gry będą trzy rządki zapałek, od 1 do 15 sztuk w rządku. Utworzymy w Edytorze postaci obraz złożony z 1 fazy, przedstawiający pojedynczy pion do gry w **NIM**. Może to być dowolny rysunek, w naszym przykładzie jest to zapałka o wymiarach 7x57.



Następnie utworzymy w projekcie zmienną globalną **bierka**, której wartością będzie stworzony obraz. W tym celu można edycyjnie zmienić postać standardowego żółwia (przycisk **Ustal postać** dialogu **Zmień mnie**), wówczas do utworzenia zmiennej wystarczy polecenie:

przypisz "bierka postać

RYSUJEMY RZĄD BIEREK

Utworzymy kilku przedstawicieli klasy **Żółw**, które będą miały postać taką jak zmienna **bierka**. Każdy kolejny żółw powinien być przesunięty w prawo. Na przykład dla trzech żółwi nasze polecenie mogłoby mieć postać:

powtórz 3 [nowy "żółw [postać :bierka pozx (-300+npw*20) pozy 200]]

Pamiętajmy, że mamy przygotować trzy rządki bierek, jeden pod drugim, liczba zapałek w rządku niech będzie losowa z zakresu od 1 do 15. Powinniśmy także zapamiętać na pomocniczej zmiennej, ile bierek leży w każdym z rządków – będziemy potrzebować tej informacji podczas gry. Napiszmy funkcję **zrobStos**, która narysuje taki rządek i jako wynik zwróci informację, ile wyświetlono bierek. Ponieważ gra polega na zabieraniu kilku bierek z konkretnego stosu, wygodnie byłoby znać nazwy żółwi z konkretnego rzędu. Dla ułatwienia przyjmijmy, że nazwy żółwi-bierek są liczbowe i żółwie w I (górnym) rządku są ponumerowane od 1 do 15, w II rządku – od 16 do 30, zaś w III – od 31 do 45. Funkcja **zrobStos** będzie miała dwa parametry: **n** – numer rysowanego rzędu oraz **y** – współrzędna pionowa bierek.

```
oto zrobStos :n :y
  niech "pom 1+losowa 15
  powtórz :pom [nowy "żółw [ postać :bierka
                                nazwa (npw+(:n-1)*15)
                                pozx (-300+npw*20)
                                pozy (:y)]]
  wy :pom
już
```

WYŚWIETLAMY LICZBY BIEREK, DODAJEMY PRZYCISK NOWA GRA

Dodamy teraz do projektu trzy pola tekstowe, które będą wyświetlały liczby bierek w poszczególnych rzędach. Zmieniamy nazwy tych pól odpowiednio na **stos1**, **stos2** i **stos3**. Pola tekstowe powinny być położone na lewo od stosu, który reprezentują.

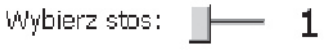
Następnie zdefiniujemy procedurę **Start** tworzącą planszę do gry w NIM. Podczas gry żółwie nie będą zmieniały swojego położenia, będą usuwane z planszy. W trakcie gry musimy oczywiście pamiętać, ile w danym momencie jest bierek w kolejnych rzędach. Te liczby będą pamiętane na liście **stos** i wyświetlane w dodanych polach tekstowych.

```
oto Start
  usuńObiekt wszystkie
  przypisz "stos []
  powtórz 3 [niech "ile zrobStos npw 320-120*npw
             proszę słowo "stos npw [ustalzawartość :ile]
             przypisz "stos nak :ile :stos]
już
```

Procedura **Start** usuwa na początku wszystkie żółwie-bierki oraz tworzy pustą listę, na której będą pamiętane liczby zapałek w poszczególnych stosach. Następnie w pętli powtarzanej trzy razy tworzy stosy (wywołuje funkcję **zrobStos** z parametrami określającymi numerem stosu oraz położenie na ekranie), wypełnia odpowiednie pole edycyjne oraz dopisuje wylosowaną liczbę bierek do listy **stos**. Dodajmy jeszcze przycisk **Nowa gra** wywołujący procedurę **Start**.

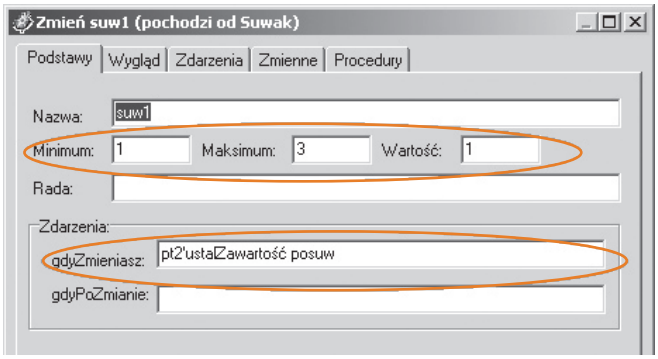
DODAJEMY SUWAKI I PRZYCISK WYKONAJ RUCH

Zadaniem każdego z graczy jest wskazanie stosu i określenie liczby bierek, jakie chce z niego zdjąć. Dodamy w związku z tym do projektu suwak i dwa pola tekstowe do wyboru rzędu bierek.



Pierwsze pole tekstowe wyświetla informację dla gracza, drugie – który stos został przez niego wybrany. Musimy jeszcze dostosować suwak do warunków gry – ustawiamy minimum na 1 i maksimum na 3 (liczba rzędów z bierkami) oraz dopisujemy polecenie, aby zmiana pozycji suwadła na suwaku była wyświetlana w polu tekstowym **pt2**.

pt2'ustalZawartość posuw



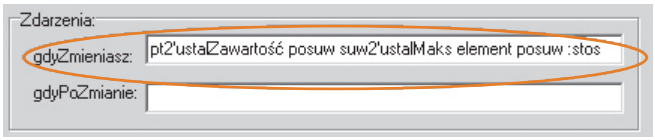
Następnie dodamy do projektu kolejny suwak z dwoma polami tekstowymi do wyboru liczby bierek.

Pierwsze pole tekstowe wyświetla informację dla gracza, drugie – liczbę bierek do zabrania. Należy także zdefiniować zdarzenie **gdyZmieniasz** dla drugiego suwaka, aby przy zmianie pozycji suwadła odpowiedni tekst był wyświetlany w polu tekstowym **pt4**.

pt4'ustalZawartość posuw

Zakres liczb, jakie możemy wybierać na drugim suwaku, zależy od tego, który stos został wybrany przez gracza. Powinniśmy zatem poprawić zdarzenie **gdyZmieniasz** dla suwaka **suw1** tak, aby przy zmianie pozycji suwadła zmieniał się zakres dopuszczalnych wartości na suwaku **suw2** (maksymalna liczba zapałek w danym stosie, element o numerze stosu z listy **stos**).

pt2'ustalZawartość posuw suw2'ustalMaks element posuw :stos



Należy teraz dodać przycisk **Wykonaj ruch** realizujący ruch gracza – z wybranego stosu usuwający podaną liczbę bierek oraz uaktualniający wszystkie wyświetlane i pamiętane informacje: pola tekstowe wyświetlające liczby bierek w rzędach, zmienną **stos** oraz zakres wartości, jakie można wybrać za pomocą suwaka **suw2**. W tym celu dopiszemy procedurę **Ruch** (wywoływaną na zdarzenie **gdyWciśnięty** dla tego przycisku), która usuwa bierki ze wskazanego stosu, modyfikuje na liście **stos** liczbę bierek dla tego stosu, wyświetla w odpowiednim polu tekstowym aktualną liczbę bierek oraz zmienia maksimum dla suwaka **suw2**:

```
oto Ruch
 jeśli suw2>0 [ usuńBierki suw1 suw2
                niech "n (element suw1 :stos) - suw2
                przypisz "stos zastąp suw1 :stos :n
                proszę słowo "stos suw1 [ustalzawartość :n]
                suw2'ustalMaks :n ]
już
```

Musimy jeszcze zdefiniować usuwanie bierek ze wskazanego rzędu (pomocnicza procedura **usuńBierki** wywoływana w procedurze **Ruch**). Bierki usuwane są od prawej do lewej, dzięki temu jeśli nie usuwamy wszystkiego z danego rzędu, pozostałe piony są wyświetlane po kolei przy polu tekstowym opisującym stan gry:

```
oto usuńBierki :n :ile
 powtórz :ile [usuńObiekt 15*(:n-1)+(element :n :stos)+1-npw]
już
```

DODAJEMY GRACZY I BADANIE STANU GRY

W naszej grze brakuje jeszcze informacji, który gracz wykonuje bieżący ruch. Jak pamiętamy, wygrywa ten z graczy, który jako ostatni weźmie pionki z planszy. Powinniśmy zatem dopisać procedurę badającą stan gry i wyświetlającą planszę z jej wynikiem. Wprowadzimy w związku z tym pomocniczą zmienną **nr_gracza** do przechowywania numeru aktualnego gracza i dodamy pole tekstowe wyświetlające go na ekranie. Dla ułatwienia zmieniamy nazwę tego pola na **gracz**. Dodamy też pole tekstowe wyświetlające gratulacje dla wygrywającego – zmieńmy jego nazwę na **koniec**. Modyfikujemy wielkość czcionki i rozmiary pola tekstowego **koniec** tak, aby przesłoniło ono cały obszar gry.

Modyfikujemy teraz procedurę **Start**:

```
oto Start
  usuńObiekt wszystkie
  przypisz "nr_gracza 1
  gracz'ustalZawartość :nr_gracza
  koniec'ustalWidoczny "fałsz
  przypisz "stos []
  powtórz 3 [niech "ile zrobStos npw 320-120*npw
                proszę słowo "stos npw [ustalzawartość :ile]
                przypisz "stos nak :ile :stos]
  suw1'ustalPosuw 1
  suw2'ustalPosuw 0
  suw2'ustalMaks pierw :stos
już
```

Musimy także dodać nową procedurę sprawdzającą, czy gra została już zakończona. Jeśli tak (wszystkie stosy są puste, czyli lista **stos** ma wartość [0 0 0]) – wyświetlimy planszę gratulacyjną, jeśli nie – kolejny ruch będzie wykonywał drugi z graczy:

```
oto koniecGry
  jeżeli :stos=[0 0 0]
        [ koniec'ustalzawartość słowo "|Koniec gry!! Wygrał gracz | :nr_gracza
          koniec'ustalwidoczny "prawda]
        [ przypisz "nr_gracza 3-:nr_gracza
          gracz'ustalZawartość :kto]
już
```

Zawartością pola tekstowego jest zawsze słowo. Słowo wstawiane do pola tekstowego koniec budujemy z napisu **Koniec gry!! Wygrał gracz** oraz wartości zmiennej **nr_gracza** przy pomocy funkcji pierwotnej **słowo**. Jeśli napis ze spacjami chcemy potraktować jak pojedyncze słowo, należy go umieścić pomiędzy znakami |.

Procedurę **koniecGry** wywołujemy po każdym ruchu, czyli najwygodniej na końcu procedury **Ruch**.

```
oto Ruch
  jeśli suw2>0 [ usuńBierki suw1 suw2
                 niech "n (element suw1 :stos) - suw2
                 przypisz "stos zastąp suw1 :stos :n
                 proszę słowo "stos suw1 [ustalzawartość :n]
                 suw2'ustalMaks :n
                 koniecGry ]
już
```

Pozostaje, drogi Czytelniku, zagrać w grę i odkryć strategię wygrywającą. W następnym numerze Świata Matematyki przedstawimy wersję gry z komputerem, w której komputer realizuje strategię wygrywającą.

Maciej Borowiecki
Wicedyrektor ds. edukacyjnych Ośrodka Edukacji
Informatycznej i Zastosowań Komputerów w Warszawie
www.oeiizk.waw.pl