Nazwa implementacji: Gra logiczna NIM

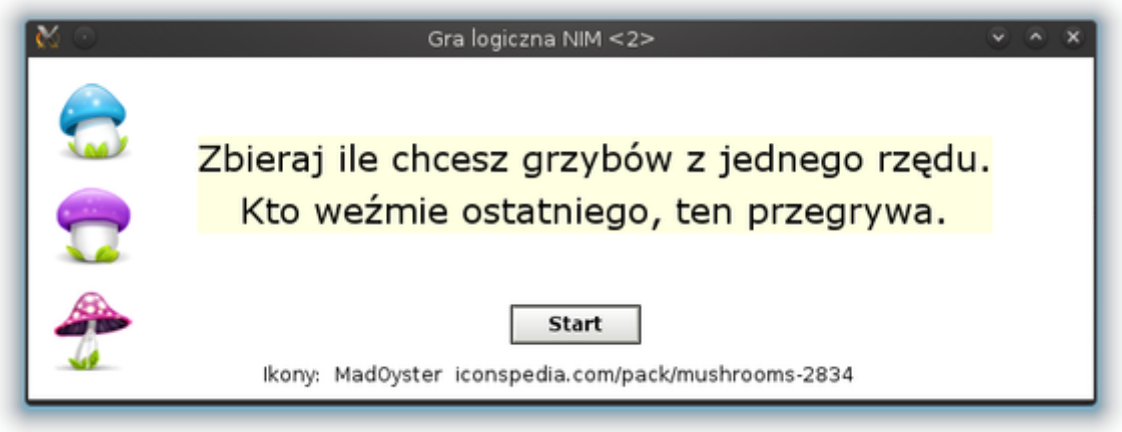
Autor: Stanisław Ubermanowicz Piotr Fiorek

Opis implementacji: Realizacja gry logicznej, w której chodzi o to, aby podczas naprzemiennego pobierania obiektów z jednego rzędu na planszy uniknąć konieczności zabrania obiektu ostatniego. Losowane są różne układy do 10 obiektów (np. grzybów) w każdym z trzech rzędów. Komputer ma zaprogramowaną strategię wygranej, dlatego rozpoczyna gracz, aby mieć szansę zwycięstwa.

Zaprojektuj grę logiczną NIM.

Program losuje w każdym rzędzie od 1 do 10 grzybów. Gracz rywalizuje z komputerem. Podczas ruchu można brać dowolną liczbę grzybów, ale tylko z jednego rzędu. Przegrywa ten, kto musi zabrać ostatniego grzyba. Strategia wygranej polega na tym, aby utrzymywać parzystość grup binarnych.

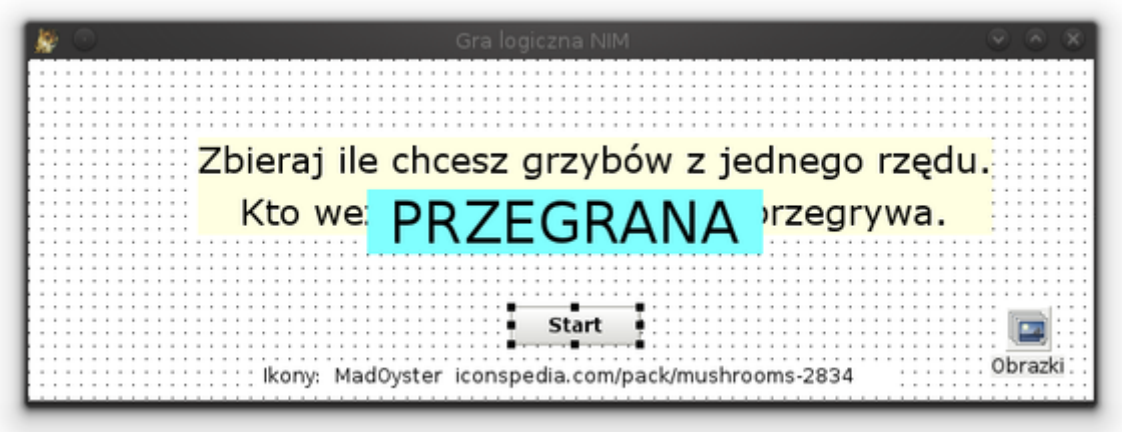
Widok po uruchomieniu



Widok w trakcie gry

Widok okna projektu

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
ORE
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY



Na formularzu należy umieścić tekst (TLabel), który będzie pojawiał się po uprzednim odpowiednim ustawieniu w przypadku wygranej lub przegranej gracza. Należy również dodać obiekt kontenera obrazków (TImageList) oraz obiekt guzika (TButton). Ponadto można dodać elementy tekstowe z instrukcją dla gracza.

W oknie „Object Inspector” można również ustawić nazwy poszczególnych elementów.

W tej implementacji używamy nazw:

„Link” dla obiektu typu TLabel prezentującego adres internetowy; „Opis” dla obiektu typu TLabel przedstawiającego opis gry; „Info” dla obiektu typu TLabel prezentującego informację o wygranej lub przegranej; „Obrazki” dla obiektu typu TImageList przechowującego obrazki kolejnych grzybów; „Start” dla obiektu typu TButton będącego przyciskiem rozpoczynającym grę;

Kod:

```
unit gra;
{$mode objfpc}{$H+}
interface
uses
 Classes, SysUtils, FileUtil, LResources, Forms, Controls, Graphics, Dialogs,
 StdCtrls, ExtCtrls;
type
 { TOkno}
 TOkno = class(TForm)
   Link: TLabel;           {Zmienne reprezentujące obiekty na formularzu}
   Start: TButton;
   Opis: TLabel;
   Obrazki: TImageList;
   Info: TLabel;
   procedure StartClick(Sender: TObject);        {Procedury obsługujące grę}
   procedure FormCreate(Sender: TObject);
   procedure Rzad1Click(Sender: TObject);
   procedure Rzad2Click(Sender: TObject);
   procedure Rzad3Click(Sender: TObject);
   procedure Rzad1Action(grzyb: Integer);
   procedure Rzad2Action(grzyb: Integer);
   procedure Rzad3Action(grzyb: Integer);
```

```
  procedure SprawdzKoniec();
  procedure PCMove();

 private
  {private declarations}
  ile1, ile2, ile3: Integer;         {Zmienne wskazują, ile grzybów jest w danej chwili w rzędzie}
  tura: Integer;    {Zmienna, która określa liczbę tur i to, czyj był ruch}
  koniec: Boolean;  {Zmienna określająca czy nastąpił koniec gry}
  Grzyb1: array[1..10] of TImage;   {Lista obiektów pierwszego rzędu grzybów}
  Grzyb2: array[1..10] of TImage;   {Lista obiektów drugiego rzędu grzybów}
  Grzyb3: array[1..10] of TImage;   {Lista obiektów trzeciego rzędu grzybów}
 public
  {public declarations}
 end;
var
 Okno: TOkno;
 i: Integer;                    {Zmienna pomocnicza, iteracyjna}
implementation
{TOkno}         {Procedura wywoływana tylko raz podczas tworzenia okna}
procedure TOkno.FormCreate(Sender: TObject);
begin
 for i:= 1 to 10 do                 {Twórz wszystkie elementy tablicy}
 begin
  Grzyb1[i]:= TImage.Create(self);          {stwórz element}
  Grzyb1[i].Parent:= self;            {przypisz go do okna}
  Grzyb1[i].Visible:= False;          {ukryj go}
  Obrazki.GetBitmap(0, Grzyb1[i].Picture.Bitmap); {przypisz mu obrazek}
  Grzyb1[i].Top:= 16;                       {określ jego położenie od góry}
  Grzyb1[i].Left:= (64*i)-48;               {określ jego położenie od lewej}
  Grzyb1[i].Tag:= i;                        {przypisz mu numer porządkowy}
  Grzyb1[i].OnClick:= @Rzad1Click;          {określ procedurę obsługującą kliknięcie}
 end;
 for i:= 1 to 10 do                  {Drugi rząd – analogicznie jak powyżej}
 begin
  Grzyb2[i]:= TImage.Create(self);
  Grzyb2[i].Parent:= self;
  Grzyb2[i].Visible:= False;
  Obrazki.GetBitmap(1, Grzyb2[i].Picture.Bitmap);
  Grzyb2[i].Top:= 80;
  Grzyb2[i].Left:= (64*i)-48;
  Grzyb2[i].Tag:= i;
   Grzyb2[i].OnClick:= @Rzad2Click;
 end;
 for i:= 1 to 10 do                  {Trzeci rząd – analogicznie jak powyżej}
 begin
  Grzyb3[i]:= TImage.Create(self);
  Grzyb3[i].Parent:= self;
  Grzyb3[i].Visible:= False;
  Obrazki.GetBitmap(2, Grzyb3[i].Picture.Bitmap);
  Grzyb3[i].Top:= 144;
  Grzyb3[i].Left:= (64*i)-48;
  Grzyb3[i].Tag:= i;
  Grzyb3[i].OnClick:= @Rzad3Click;
 end;
 Grzyb1[1].Visible:=True;    {Pokaż pierwsze grzyby z każdego rzędu}
```

```
 Grzyb2[1].Visible:=True;
 Grzyb3[1].Visible:=True;
end;
procedure TOkno.StartClick(Sender: TObject);        {Obsługa kliknięcia przycisku START}
begin
 Start.Visible:= False;                 {ukryj przycisk}
 Link.Visible:= False;                 {ukryj adres autora ikon}
 Opis.Visible:= False;                 {ukryj opis gry}
 Info.Visible:= False;                 {ukryj Info o wygranej lub przegranej}
 koniec:= False;         {ustaw zmienną oznaczającą koniec gry na Fałsz}
 tura:= 1;                          {ustaw licznik tur na zero}
 Randomize;                          {ustaw pozycję startową generatora liczb losowych}
 ile1:= Random(10)+1;                  {wylosuj liczbę kulek w pierwszym rzędzie}
 for i:= 1 to ile1 do                  {pętla przebiega przez elementy od pierwszego do wylosowanego}
   Grzyb1[i].Visible:= True;          {i ustawia, aby elementy były widoczne}
 ile2:= Random(10)+1;                  {to samo dla drugiego rzędu}
 for i:= 1 to ile2 do
   Grzyb2[i].Visible:= True;
 ile3:= Random(10)+1;                  {to samo dla trzeciego rzędu}
 for i:= 1 to ile3 do
   Grzyb3[i].Visible:= True;
end;
       {Procedura obsługująca kliknięcie w element z pierwszego rzędu}
procedure TOkno.Rzad1Click(Sender: TObject);
 {Sender jest ogólną reprezentacją obiektu, który został kliknięty.
 Przekształcamy go na konkretny obiekt TImage, pobieramy z niego wartość Tag i przekazujemy do procedury Rzad1Action}
begin
Rzad1Action((Sender as TImage).Tag);
         {sprawdź czy zmienna 'koniec' została ustawiona na Fałsz, jeśli tak, to zrealizuj ruch komputera}
 if koniec = False then PCMove();
end;
procedure TOkno.Rzad2Click(Sender: TObject); {jak wyżej dla drugiego rzędu}
begin
 Rzad2Action((Sender as TImage).Tag);
 if koniec = False then PCMove();
end;
procedure TOkno.Rzad3Click(Sender: TObject); {jak wyżej dla trzeciego rzędu}
begin
 Rzad3Action((Sender as TImage).Tag);
 if koniec = False then PCMove();
end;

procedure TOkno.Rzad1Action(grzyb: Integer); {Procedura obsługująca zmiany stanów rzędu 1}
begin
 tura:= tura+1;                   {zwiększ licznik tur}
 for i:= grzyb to ile1 do         {wykonaj ruch od grzyba kliknietego do ostatniego}
   Grzyb1[i].Visible:= False;     {ukryj grzyba}
 ile1:= grzyb-1;                  {ustaw nową wartość pozostałych grzybów}
 SprawdzKoniec;                   {sprawdź czy nastąpił koniec gry}
end;
procedure TOkno.Rzad2Action(grzyb: Integer);         {to samo dla rzędu drugiego}
begin
 tura:= tura+1;
 for i:= grzyb to ile2 do
   Grzyb2[i].Visible:= False;
```

```
 ile2:= grzyb-1;
 SprawdzKoniec;
end;
procedure TOkno.Rzad3Action(grzyb: Integer);         {to samo dla rzędu trzeciego}
begin
 tura:= tura+1;
 for i:= grzyb to ile3 do
   Grzyb3[i].Visible:= False;
 ile3:= grzyb-1;
 SprawdzKoniec;
end;

procedure TOkno.SprawdzKoniec(); {Sprawdzaj, czy wszystkie grzyby zostały zabrane}
begin
 if ile1+ile2+ile3 < 2 then          {Sprawdź, czy grzybów jest mniej niż dwa}
 begin
   if (tura mod 2) = 1 then          {Jeśli była tura gracza}
     Info.Caption:= 'Przegrana'        {wpisz informację o przegranej}
   else if ile1+ile2+ile3 = 1 then   {Jeśli pozostał ostatni grzyb}
     Info.Caption:= 'Wygrana'          {wpisz informację o wygranej}
   else                              {a jeśli był ruch komputera}
     Info.Caption:= 'Przegrana';       {wpisz informację o przegranej}
   koniec:= True;                    {Ustaw zmienną oznaczającą koniec gry}
   Info.Visible:= True;              {Pokaż napis z informacją o wyniku gry}
   Start.Visible:= True;             {Pokaż przycisk START, dla uruchomienia nowej gry}
 end;                                {po wykonanym ruchu wyjdź z funkcji}
end;

procedure TOkno.PCMove();                {Procedura realizująca strategię komputera – sztuczny intelekt}
begin
 if (ile1=ile2) and (ile1<2) and (ile3>1) then  {gdy w rzędach 1 i 2 jest 0 lub 1 grzyb,}
 begin                          {to pozostaw jednego grzyba w rzędzie 3.}
  Rzad3Action(2);               {kliknij grzyba nr 2 w rzędzie 3.}
  Exit;
 end
 else if (ile1=ile3) and (ile1<2) and (ile2>1) then {gdy w rzędach 1 i 3 są <2 grzyby,}
 begin                          {to pozostaw 1 grzyba w rzędzie 2.}
   Rzad2Action(2);              {kliknij grzyba nr 2 w rzędzie 2.}
   Exit;
 end
 else if (ile2=ile3) and (ile2<2) and (ile1>1) then {gdy w rzędach 2 i 3 są <2 grzyby,}
 begin                          {to pozostaw 1 grzyba w rzędzie 1.}
   Rzad1Action(2);              {kliknij grzyba nr 2 w rzędzie 1.}
   Exit();
 end;
 If (ile1+ile2)=1 then          {gdy w rzędach 1 i 2 pozostał łącznie jeden grzyb}
 begin
   Rzad3Action(1);              {usuń wszystkie z rzędu 3}
   Exit();
 end
 else if (ile1+ile3)=1 then           {analogicznie jak powyżej}
 begin
   Rzad2Action(1);
   Exit();
 end
```

```
 else if (ile2+ile3)=1 then          {analogicznie jak powyżej}

 begin
   Rzad1Action(1);
   Exit();
 end;

{Sprawdzenie parzystości binarnej w środkowej fazie gry. Szukanie układu dającego parzystość grup.
Jeśli dla danej wartości ‘i’ poniższe wyrażenia logiczne z operatorem XOR się zerują,
to jest to poszukiwany układ parzystości, dający szansę na wygraną.}
for i:= 1 to 10 do
 begin
   if (ile1>=i) and (((ile1-i) xor ile2 xor ile3) = 0) then
   begin
     Rzad1Action(ile1 - i + 1);              {doprowadź układ do parzystości}
     Exit();                                {po wykonanym ruchu wyjdź z funkcji}
   end
   else if (ile2>=i) and ((ile1 xor (ile2-i) xor ile3) = 0) then
   begin
     Rzad2Action(ile2 - i + 1);              {doprowadź układ do parzystości}
     Exit();
   end
   else if (ile3>=i) and ((ile1 xor ile2 xor (ile3-i)) = 0) then
   begin
     Rzad3Action(ile3 - i + 1);              {doprowadź układ do parzystości}
     Exit();
   end;
 end;

{Losowy wybór posunięcia, jeśli strategię wykorzystał gracz i nie ma ruchu wygrywającego}
 if (ile1>=ile2) and (ile1>=ile3) and ((Random(10)/10)>0.5) then
   Rzad1Action(Random(ile1) + 1)
 else if (ile2>=ile1) and (ile2>=ile3) then
   Rzad2Action(Random(ile2) + 1)
 else
   Rzad3Action(ile3);
end;
initialization
 {$I gra.lrs}
end.
```

UWAGA: Podczas zapisywania projektu nie wolno zapisywać plików ".pas" (plik z kodem) oraz ".lpi" (plik projektu) pod tymi samymi nazwami.