LIVE – Gra w życie

Live jest jednym z pierwszych i najbardziej znanych tzw. automatów komórkowych. Został wymyślony w 1970 roku przez brytyjskiego matematyka Johna Conwaya. Co to takiego automat komórkowy? Mnóstwo sąsiadujących z sobą komórek. Każda z nich może być zapalona lub zgaszona zgodnie z wymyślonymi prostymi regułami. Na czym polega gra – symulacja? Ustalamy układ początkowy komórek, puszczamy cały układ w ruch i obserwujemy. Więcej na Wikipedii: https://pl.wikipedia.org/wiki/Gra_w_życie

LIVE w Pascalu (Delphi, Lazarus) – krok po kroku
*Podobnie jak w JavaScript, w tych wersjach Pascala mamy do czynienia z obiektem **canvas**, więc cała graficzna oprawa będzie bardzo podobna. Jedyna większa różnica, to sposób animacji – w Pascalu operujemy tzw. **timerami**.*

Tworzenie projektu
- wybierz z menu **Plik – Nowy – Aplikacja**
- zapisz projekt **Plik – Zapisz wszystko**
(najlepiej w nowym folderze i z domyślnymi nazwami plików)

Parametry początkowe formatki
- w oknie **Properties**, wybierz **Zdarzenia** dla komponentu **Form1**
- kliknij podwójnie w zdarzenie **OnCreate**
*utworzona zostanie procedura **FormCreate***
- wpisz do niej polecenia z ramki
- uruchom program poleceniem F9

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  caption:='LIVE';
  height:=620;
  width:=800;
  color:=clGray;
end;
```

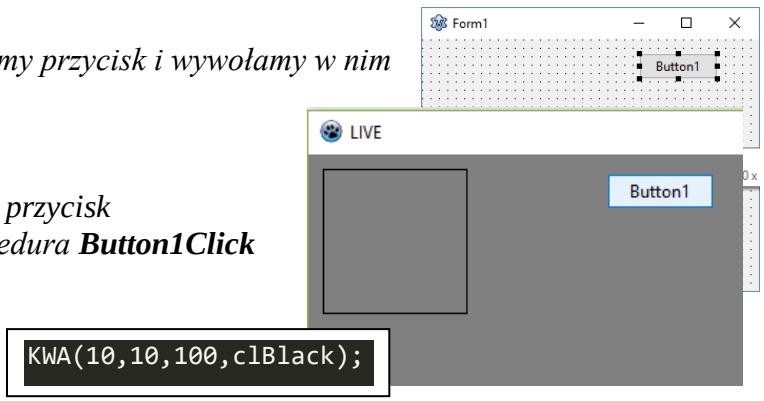
Jeśli wszystko poprawnie wpisane, to pojawi się szare okno o wymiarach 620x800 pikseli.
UWAGA – zawsze zakończ działanie programu przed kolejnym uruchomieniem
*UWAGA – własności każdego komponentu można również wpisywać bezpośrednio w oknie **Properties,** w zakładce **Zdarzenia**.*

KWADRAT
Procedura OnCreate pojawiła się automatycznie po podwójnym kliknięciu w zdarzeniach.
Procedurę rysującą kwadraty musimy przygotować samodzielnie.
- wklej do programu procedurę z ramki
parametry kwadratu: lewy, górny róg (x,y) oraz bok i kolor ramki.
Aby program działał poprawnie, nagłówek procedury powinien dodatkowo znaleźć się w miejscu, gdzie gromadzone są wszystkie metody obiektu Form1 – pod nagłówkiem procedury FormCreate
- wklej do programu procedurę KWA z ramki

```
procedure Tform1.KWA(x,y,bok,kolor:integer);
begin
   canvas.pen.color:=kolor;
   canvas.moveto(x,y);
   canvas.lineto(x+bok,y);
   canvas.lineto(x+bok,y+bok);
   canvas.lineto(x,y+bok);
   canvas.lineto(x,y);
end;
```

```
TForm1 = class(TForm)
  procedure FormCreate(Sender: TObject);
  procedure KWA(x,y,bok,kolor:integer);
```

Sprawdzamy rysowanie kwadratów
Aby sprawdzić działanie procedury KWA utworzymy przycisk i wywołamy w nim rysowanie przykładowego kwadratu
- kliknij w formatkę
- wybierz z zakładki **Standard** przycisk **TButton**
- kliknij w dowolne miejsce formatki – *pojawi się przycisk*
- kliknij podwójnie w przycisk – *utworzy się procedura **Button1Click***
- wpisz do niej polecenie z ramki
- uruchom program F9 i kliknij w przycisk – *powinien pojawić się czarny kwadrat*




```
KWA(10,10,100,clBlack);
```

Parametry przycisku

Ustawimy przycisk w odpowiednim miejscu definiując jego własności w procedurze OnCreate (albo zapisując bezpośrednio w własnościach przycisku)

- wpisz do procedury OnCreate polecenia z ramki

Po uruchomieniu programu przycisk zawsze pojawi się w wyznaczonym miejscu

UWAGA – zamiast pisać kilka razy nazwę przycisku (Button1) można zastosować konstrukcję WITH …DO…

```
Button1.Height:=50;
Button1.Left:=620;
Button1.Width:=170;
Button1.Top:=10;
Button1.Caption:='START';
```

```
with Button1 do
begin
    Height:=50;
    Left:=620;
    Width:=170;
    Top:=10;
    Caption:='START';
end;
```

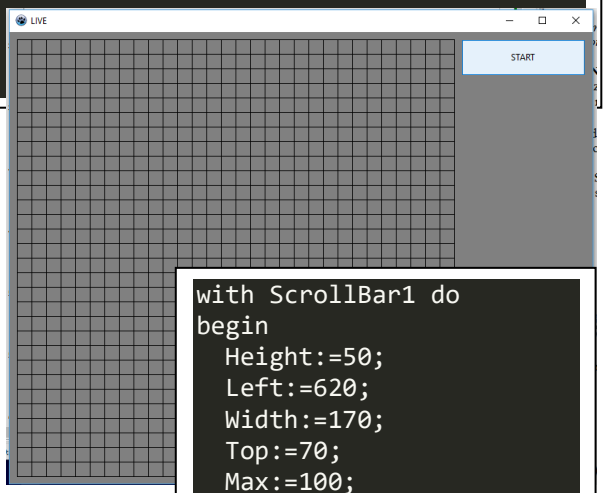
SZACHOWNICA

Szachownica zbudowana jest z kwadratów – taka sama ilość w pionie i w poziomie. Do narysowania szachownicy posłużymy się dwoma pętlami.

- wpisz procedurę SZACHY z ramki
- nagłówek procedury umieść w deklaracji procedur formatki
- w przycisku START umieść rysowanie przykładowej szachownicy

```
SZACHY(10,10,20,clBlack,30);
```

```
procedure TForm1.SZACHY(x,y,bok,kolor,ile:integer);
var
   i,j:integer;
   x1,y1:integer;
begin
  for i:=1 to ile do
  for j:=1 to ile do
  begin
     x1:=x+(i-1)*bok;
     y1:=y+(j-1)*bok;
     KWA(x1,y1,bok,kolor);
  end;
end;
```



SUWAK – rysujemy szachownice

Za pomocą suwaka będziemy ustawiać ilość kratek szachownicy. Bok kratki obliczymy dzieląc maksymalny rozmiar szachownicy (600 pikseli) przez ilość kratek). Na suwaku będziemy mogli zmieniać ilość kratek w zakresie 1 do 100.

- umieść komponent TScrollBar na formatce
- w procedurze OnCreate wpisz parametry suwaka ScrollBar1

```
with ScrollBar1 do
begin
  Height:=50;
  Left:=620;
  Width:=170;
  Top:=70;
  Max:=100;
  Min:=1;
end;
```

Jak zmieniać wymiary szachownicy? Za każdym razem gdy zmieniamy coś na suwaku generowane jest zdarzenie OnChange suwaka i właśnie w tej procedurze opiszemy rysowanie dowolnej szachownicy.

- kliknij podwójnie w suwak

pojawi się nowa procedura ScrollBar1Change

- wpisz do niej polecenia z ramki

```
procedure TForm1.ScrollBar1Change(Sender: TObject);
var bok,ile:integer;
begin
  ile:=Scrollbar1.position;
  bok:=round(600 / ile);
  Repaint;
  SZACHY(10,10,bok,clBlack,ile);
end;
```

*do zmiennej **ile** pobierana jest zawartość suwaka*

bok** kwadratu musi być liczbą całkowitą (ilość pikseli) dlatego zaokrąglamy funkcją **round

*procedura **Repaint** odświeża ekran przed narysowaniem kolejnej szachownicy*

KLIKANIE

Początkowy układ robaczków ustalamy klikając myszką w pola szachownicy. Formatka dysponuje oczywiście odpowiednim zdarzeniem związanym z klikaniem – OnMouseDown

- kliknij w dowolne miejsce na formatce (lub wybierz Form1:TForm1 na liście komponentów)
- w oknie Properties wybierz zakładkę Zdarzenia
- kliknij podwójnie w zdarzenie OnMuseDown - *pojawi się szkielet nowej procedury FormMouseDown*
- wpisz w procedurze polecenie z ramki

```
ShowMessage('X:'+IntToStr(X)+' '+'Y:'+IntToStr(Y));
```

Procedura FormMouseDown oferuje nam współrzędne kliknięcia w pikselach. Nam potrzebne są numery wiersza i kolumny kratek szachownicy. Jak to obliczyć? Współrzędną kliknięcia dzielimy całkowicie przez bok kwadratu. Uwzględniamy jeszcze przesunięcie początku szachownicy (o 10 pikseli) i powiększamy wszystko o jeden – i już!

- wpisz polecenia z ramki do procedury FormMouseDown

Zauważ, że znów musieliśmy pobierać zawartość suwaka i obliczać bok. Procedura ShowMessage tym razem pokazuje numer kratki na szachownicy.

```
var ile,bok,wie,kol:integer;
begin
   ile:=Scrollbar1.position;
   bok:=round(600 / ile);
   kol:=(X - 10) div bok +1;
   wie:=(Y - 10) div bok +1;
   ShowMessage('K:'+IntToStr(kol)+' '+'W:'+IntToStr(wie));
end;
```

WYPEŁNIANIE

Kratka została kliknięta – powinna wypełnić się białym kolorem. Ponowne kliknięcie maluje wnętrze kolorem szarym. Znamy współrzędne klikniętej kratki na szachownicy, ale nie znamy współrzędnych lewego górnego rogu kwadratu, który mamy wypełnić. Na podstawie numeru wiersza i kolumny (obliczonej przed chwilą) wyliczymy współrzędne początku kratki do wypełnienia.

```
var x1,y1:integer;

x1:=(kol-1)*bok+10;
y1:=(wie-1)*bok+10;
```

- dopisz do procedury FormMouseDown instrukcje z ramki

deklaracje zmiennych przed słowem begin, obliczanie współrzędnych x1 i y1 za instrukcjami wyliczającymi numer wiersza i kolumny.

*Aby narysować wypełniony kwadrat zmodyfikujemy napisaną wcześniej procedurę KWA, Pojawił się jeszcze jeden parametr **wnetrze** i dwie nowe instrukcje na początku.*

```
procedure TForm1.KWA(x,y,bok,kolor,wnetrze:integer);
begin
   canvas.brush.color :=wnetrze;
   canvas.fillrect(x,y,x+bok,y+bok);
   canvas.pen.color:=kolor;
   canvas.moveto(x,y);
   canvas.lineto(x+bok,y);
   canvas.lineto(x+bok,y+bok);
   canvas.lineto(x,y+bok);
   canvas.lineto(x,y);
end;
```

- zmień procedurę KWA

- zmień nagłówek procedury KWA w opisie klasy Form1

```
procedure KWA(x,y,bok,kolor,wnetrze:integer);
```

- zmień procedurę SZACHY

```
KWA(x1,y1,bok,kolor,clGray);
```

- zmień procedurę Button1Click

```
KWA(10,10,600,clBlack,clGray);
```

- dopisz do procedury FormMouseDown

```
KWA(10,10,600,clBlack,clWhite);
```

TABLICE

Klikanie w szachownicę będzie „zapalało" kratki szachownicy na biały kolor. Co zrobić, żeby ponowne kliknięcie „gasiło" obszar na kolor szarego tła? Można sprawdzać kolor kliknięty piksela: jeśli był szary to zapal go na biały, w przeciwnym razie – zapal go na szary kolor. Można i tak, ale po pierwsze jest to nieefektywne, a po drugie – aby zagrać w Live musimy pamiętać (program musi pamiętać), które piksele są zapalone, a które zgaszone, aby obliczać gdzie zapalą się lub zgaszą robaczki w kolejnych pokoleniach. W tym celu wykorzystamy ***tablice*** *– możesz sobie wyobrazić, że komputer będzie miał do dyspozycji szufladę z mnóstwem przegródek w formie szachownicy właśnie. Do każdej przegródki można wsadzić jakieś informacje. Łatwo je zapisać i łatwo je odczytać za pomocą indeksowania – numerowania, jak w grze w statki.*

Na początku założyliśmy, że tablica będzie miała maksymalny wymiar 100x100, więc i taka tablica będzie nam potrzebna. Uprzedzając nieco zadeklarujemy ją nieco większą, aby łatwiej było analizować brzegi.

- zadeklaruj tablicę ROBACZKI w głównej części programu
poniżej instrukcji var Form1:TForm1;
w każdej komórce tablicy będziemy wpisywać 0 – brak robaczka lub 1 – jest robaczek
tablica powinna być wyzerowana na początku działania programu

```
ROBACZKI:array[0..101,0..101] of byte;
```

- wpisz procedurę ROBACZKI_ZERUJ
- umieść wywołanie procedury ROBACZKI_ZERUJ(); na końcu procedury FormCreate
ponieważ ta procedura nie ma nic wspólnego z rysowaniem i obiektami komponenty Form1, dlatego też jej nagłówka nie musimy umieszczać w opisie klasy

```
procedure ROBACZKI_ZERUJ;
var w,k:byte;
begin
  for w:=0 to 101 do
  for k:=0 to 101 do
    ROBACZKI[k,w]:=0;
end;
```

ZAPALANIE-GASZENIE

Gdy klikamy w szachownicę pole zapala się na biało i dodatkowo do odpowiedniego pola tablicy zapisujemy jedynkę. Gdy gasimy (na szaro) kratkę – do tablicy zapisujemy zero. Odwróćmy jednak tok rozumowania. Klikamy w szachownicę, zaglądamy do tablicy i jeśli w polu jest zero, to rysujemy białą kratkę i zapisujemy jedynkę. W przeciwnym razie postępujemy odwrotnie – rysujemy szare pole i do tablicy zapisujemy zero. Cała instrukcja warunkowa będzie umieszczona w procedurze FormMouseDown

- w procedurze FormMouseDown usuń instrukcję rysującą biały kwadrat
- wpisz polecenia z ramki

```
if ROBACZKI[kol,wie]=0 then
  begin
    ROBACZKI[kol,wie]:=1;
    KWA(x1,y1,bok,clBlack,clWhite);
  end else
  begin
    ROBACZKI[kol,wie]:=0;
    KWA(x1,y1,bok,clBlack,clGray);
  end;
```

Teraz możemy ustawić początkowe położenie robaczków. Robaczki są rysowane lub wymazywane i dodatkowo wpisywane są jedynki i zera do tablicy ROBACZKI.

PRZERYSOWANIE SZACHOWNICY

Gra w życie polega na ciągłym odradzaniu się i umieraniu robaczków. Program będzie musiał przerysowywać całą szachownicę zgodnie z tym co jest w tablicy ROBACZKI – procedura RYSUJ_ROBACZKI.

- wpisz procedurę RYSUJ_ROBACZKI
- umieść nagłówek procedury w opisie klasy Form1
- w procedurze ScrollBar1Change wpisz instrukcję RYSUJ_ROBACZKI(); i usuń instrukcję SZACHY(…)

```
procedure TForm1.RYSUJ_ROBACZKI();
var
  k,w:integer;
  ile,bok:integer;
  x1,y1:integer;
begin
  ile:=Scrollbar1.position;
  bok:=round(600 / ile);
  for k:=1 to ile do
  for w:=1 to ile do
  begin
    x1:=(k-1)*bok+10;
    y1:=(w-1)*bok+10;
    if ROBACZKI[k,w]=1
      then KWA(x1,y1,bok,clBlack,clWhite)
      else KWA(x1,y1,bok,clBlack,clGray);
  end;
end;
```

Po uruchomieniu programu i ustawieniu początkowego stany robaczków będą zapamiętywane i odtwarzane gdy zmienimy suwakiem wielkość szachownicy.

STAN POCZĄTKOWY

Po uruchomieniu programu ustawiamy suwak na 10 i rysujemy takąż pustą szachownicę

- do FormCreate wstaw instrukcję dla ScrollBar1

```
ScrollBar1.Position:=10;
```

Okazuje się, że wstawienie instrukcji RYSUJ_ROBACZKI do FormCreate nie da oczekiwanego rezultatu – FormCreate wykonuje się jeszcze przed utworzeniem formatki z obszarem canvas. Procedurę rysowania musimy umieścić w podobnej procedurze FormActivate.

-kliknij podwójnie w zdarzenie OnActivate formatki
- wpisz instrukcje z ramki

```
repaint;
RYSUJ_ROBACZKI();
```

Po uruchomieniu programu mamy szachownicę o 10 kolumnach i wierszach

POKOLENIA

Potrafimy zmieniać szachownicę, ustawiać początkowe położenie robaczków i je zapamiętywać w tablicy. Aby „puścić w ruch" naszą grę należy wykonać jeszcze dwie czynności:
- wyliczyć ile robaczków znajduje się wokół każdego pola
- usunąć lub wstawić robaczki na podstawie ilości sąsiadów
Ilość sąsiadów zapiszemy (jak należy się domyślać) w kolejnej tablicy - SASIEDZI. Każde pole ma 8 sąsiadujących pól, więc w tablicy będą znajdować się liczby od 0 do 8.

- zadeklaruj tablicę SASIEDZI (pod deklaracją tablicy ROBACZKI)

```
SASIEDZI:array[0..101,0..101] of byte;
```

- zmodyfikuj procedurę ZERUJ_ROBACZKI
zwróć uwagę na konieczność zastosowania instrukcji bloku (Begin ... end)

```
procedure ROBACZKI_ZERUJ();
var w,k:byte;
begin
  for w:=0 to 101 do
  for k:=0 to 101 do
  begin
    ROBACZKI[k,w]:=0;
    SASIEDZI[k,w]:=0;
  end;
end;
```

Jak policzyć ilość zajętych pól wokół wybranego pola? – należy „rozejrzeć się" wokół, w ośmiu kierunkach. W tym celu napiszemy osobną funkcję, która policzy ile jest robaczków wokół pola

- wpisz definicję funkcji WYLICZ_ROBACZKI
do funkcji „wrzucamy" numer kolumny i wiersza a otrzymujemy (ostatnia instrukcja funkcji) – ilość zajętych pól wokół kratki (k,w)

```
function WYLICZ_ROBACZKI(k,w:byte):byte;
var suma:byte;
begin
  suma:=0;
  suma:=suma+ROBACZKI[k-1,w-1];
  suma:=suma+ROBACZKI[k-1,w-0];
  suma:=suma+ROBACZKI[k-1,w+1];
  suma:=suma+ROBACZKI[k-0,w-1];
  suma:=suma+ROBACZKI[k-0,w+1];
  suma:=suma+ROBACZKI[k+1,w-1];
  suma:=suma+ROBACZKI[k+1,w-0];
  suma:=suma+ROBACZKI[k+1,w+1];
  WYLICZ_ROBACZKI:=suma;
end;
```

Potrafimy policzyć ilość robaczków wokół jednego pola. Kolejna procedura policzy ilość robaczków wokół wszystkich pól – wyniki zapisujemy w nowej tablicy SASIEDZI

-wpisz w dowolnym miejscy procedurę POLICZ_ROBACZKI

```
procedure POLICZ_ROBACZKI();
var w,k:byte;
begin
  for w:=0 to 101 do
  for k:=0 to 101 do
    SASIEDZI[k,w]:=WYLICZ_ROBACZKI(k,w);
end;
```

REGUŁY LIVE

Pozostała ostatnia (prawie) procedura – na podstawie ustawienia robaczków w tablicy ROBACZKI oraz ilości robaczków wokół pola obliczonej w tablicy SASIEDZI, należy wyliczyć jakie będzie nowe pokolenie robaczków: w których polach narodzą się nowe, a w których polach zginą robaczki. Wynik zapiszemy powrotem w tablicy ROBACZKI. Powtarzając w koło wyliczanie sąsiadów, wyliczanie nowego pokolenia i rysowanie szachownicy będziemy mogli „wprawić w ruch" naszą symulację.

Jakie są reguły życia i śmierci?
Narodziny – nowy robaczek rodzi się na pustym polu i musi mieć dokładnie trzech sąsiadów
Śmierć – robaczek umiera gdy ma mniej niż dwóch sąsiadów (samotność) lub gdy ma więcej niż trzech sąsiadów (brak pożywienia)
Wystarczy te warunki zapisać w postaci matematycznej.

```
procedure NOWE_POKOLENIE();
var
   k,w,sum,rob:byte;
begin
  for w:=1 to 100 do
  for k:=1 to 100 do
  begin
    rob:=ROBACZKI[k,w];
    sum:=SASIEDZI[k,w];
    if (rob=0) and ( (sum=3) or (sum=3))
      then ROBACZKI[k,w]:=1;
    if (rob=1) and ((sum<2) or (sum>=4))
      then ROBACZKI[k,w]:=0;
  end;
end;
```

- wpisz procedurę NOWE_POKOLENIE

sprawdzimy jak działa nasza gra krok po kroku – klikając w przycisk START

- wpisz do procedury Button1Click nowe instrukcje

```
POLICZ_ROBACZKI();
NOWE_POKOLENIE();
RYSUJ_ROBACZKI();
```

Narysuj układ początkowy i klikaj w przycisk START, a będziesz mógł śledzić, jak zmieniają się kolejne pokolenia robaczków.

ANIMACJA

I to już ostatni element programu – chcemy aby proces odradzania i umierania odbywał się automatycznie. W opisywanych odmianach Pascala (Delphi i Lazarus) mamy możliwość zastosowania komponentu Timer, który cyklicznie potrafi wywoływać określone instrukcje.

- wstaw na formatkę komponent TTimer z zakładki System
w dowolnym miejscu – jest niewidoczny po uruchomieniu programu
- kliknij podwójnie w komponent TTimer na formatce
pojawi się szkielet procedury Timer1Timer
- wpisz do procedury trzy instrukcje

```
POLICZ_ROBACZKI();
NOWE_POKOLENIE();
RYSUJ_ROBACZKI();
```

Aby uruchomić lub zatrzymać animację wykorzystamy przycisk START
- usuń z procedury Button1Click trzy instrukcje ręcznej animacji
- wstaw do procedury Button1Click instrukcje z ramki
- wpisz do procedury FormCreate początkowe ustawienie Timera – wyłączony oraz szybkość animacji (milisekundy

```
Timer1.Enabled:=not(Timer1.Enabled);
if Timer1.Enabled
  then Button1.Caption:='STOP'
  else Button1.Caption:='START';
```

```
Timer1.Enabled:=false;
Timer1.Interval:=1;
```

W dowolnym momencie możesz uruchomić i zatrzymać animację. Jeśli przebiega zbyt szybko możesz zmienić parametr Interval (1000 milisekund to 1 sekunda)

O różnych interesujących aspektach LIVE dowiesz się na przykład z Wikipedii. Interesujące są układy cykliczne, nieśmiertelne, samoloty, pulsujące i wiele innych.
Jeśli chcesz poeksperymentować, spróbuj zmienić zasady narodzin i umierania robaczków w procedurze NOWE_POKOLENIE

CAŁY PROGRAM

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
      Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls,
                  ExtCtrls;

type

                  { TForm1 }

      TForm1 = class(TForm)
            Button1: TButton;
            ScrollBar1: TScrollBar;
            Timer1: TTimer;
            procedure Button1Click(Sender: TObject);
            procedure FormActivate(Sender: TObject);
            procedure FormCreate(Sender: TObject);
            procedure FormMouseDown(Sender: TObject; Button: TMouseButton;
                              Shift: TShiftState; X, Y: Integer);
            procedure KWA(x,y,bok,kolor,wnetrze:integer);
            procedure ScrollBar1Change(Sender: TObject);
            procedure SZACHY(x,y,bok,kolor,ile:integer);
            procedure RYSUJ_ROBACZKI();
            procedure Timer1Timer(Sender: TObject);


      private

      public

      end;

var
      Form1: TForm1;
      ROBACZKI:array[0..101,0..101] of byte;
      SASIEDZI:array[0..101,0..101] of byte;

implementation

{$R *.lfm}

{ TForm1 }

procedure ROBACZKI_ZERUJ();
var w,k:byte;
begin
  for w:=0 to 101 do
  for k:=0 to 101 do
  begin
    ROBACZKI[k,w]:=0;
    SASIEDZI[k,w]:=0;
      end;
end;

function WYLICZ_ROBACZKI(k,w:byte):byte;
var suma:byte;
begin
  suma:=0;
  //gorny wiersz
  suma:=suma+ROBACZKI[k-1,w-1];
  suma:=suma+ROBACZKI[k-1,w-0];
```

```
  suma:=suma+ROBACZKI[k-1,w+1];
  //srodkowy wiersz
  suma:=suma+ROBACZKI[k-0,w-1];
  //samego siebie nie wliczamy do sumy
  suma:=suma+ROBACZKI[k-0,w+1];
  //dolny wiersz
  suma:=suma+ROBACZKI[k+1,w-1];
  suma:=suma+ROBACZKI[k+1,w-0];
  suma:=suma+ROBACZKI[k+1,w+1];
  WYLICZ_ROBACZKI:=suma;
end;

procedure POLICZ_ROBACZKI();
var
   k,w:byte;
begin
  for w:=1 to 100 do
  for k:=1 to 100 do
      SASIEDZI[k,w]:=WYLICZ_ROBACZKI(k,w);
end;

procedure NOWE_POKOLENIE();
var
   k,w,sum,rob:byte;
begin
  for w:=1 to 100 do
  for k:=1 to 100 do
  begin
      rob:=ROBACZKI[k,w];//pole wolne czy zajete
      sum:=SASIEDZI[k,w];//ile robaczkow wokolo

      if (rob=0) and ( (sum=3) or (sum=3)) then ROBACZKI[k,w]:=1;//rodzi sie nowy
robaczek
      if (rob=1) and ((sum<2) or (sum>=4)) then ROBACZKI[k,w]:=0;//umiera
      end;
end;


procedure TForm1.FormCreate(Sender: TObject);
begin
  caption:='LIVE';
  height:=620;
  width:=800;
  color:=clGray;
  with Button1 do
  begin
    Height:=50;
    Left:=620;
    Width:=170;
    Top:=10;
    Caption:='START';
      end;
  ROBACZKI_ZERUJ();
  ScrollBar1.Position:=10;
  Timer1.Enabled:=false;
  Timer1.Interval:=1;
end;

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
                  Shift: TShiftState; X, Y: Integer);
var
  ile,bok,wie,kol:integer;
  x1,y1:integer;
begin
   ile:=Scrollbar1.position;
   bok:=round(600 / ile);
   kol:=(X - 10) div bok +1;
```

```
   wie:=(Y - 10) div bok +1;
   x1:=(kol-1)*bok+10;
   y1:=(wie-1)*bok+10;

   if ROBACZKI[kol,wie]=0 then
   begin
     ROBACZKI[kol,wie]:=1;
     KWA(x1,y1,bok,clBlack,clWhite);
       end else
   begin
     ROBACZKI[kol,wie]:=0;
     KWA(x1,y1,bok,clBlack,clGray);
       end;

   //KWA(x1,y1,bok,clBlack,clWhite);
   //ShowMessage('X:'+IntToStr(X)+' '+'Y:'+IntToStr(Y));
   //ShowMessage('K:'+IntToStr(kol)+' '+'W:'+IntToStr(wie));
end;


procedure TForm1.Button1Click(Sender: TObject);
begin
  //KWA(10,10,600,clBlack,clGray);
  //SZACHY(10,10,20,clBlack,30);
  //POLICZ_ROBACZKI();
  //NOWE_POKOLENIE();
  //RYSUJ_ROBACZKI();
  Timer1.Enabled:=not(Timer1.Enabled);
  if Timer1.Enabled
     then Button1.Caption:='STOP'
     else Button1.Caption:='START';
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
  repaint;
  RYSUJ_ROBACZKI();
end;

procedure TForm1.KWA(x,y,bok,kolor,wnetrze:integer);
begin
   canvas.brush.color :=wnetrze;
   canvas.fillrect(x,y,x+bok,y+bok);
   canvas.pen.color:=kolor;
   canvas.moveto(x,y);
   canvas.lineto(x+bok,y);
   canvas.lineto(x+bok,y+bok);
   canvas.lineto(x,y+bok);
   canvas.lineto(x,y);
end;

procedure TForm1.ScrollBar1Change(Sender: TObject);
var bok,ile:integer;
begin
  ile:=Scrollbar1.position;
  bok:=round(600 / ile);
  repaint;
  //canvas.Clear;
  RYSUJ_ROBACZKI();
  //SZACHY(10,10,bok,clBlack,ile);
end;

procedure TForm1.SZACHY(x,y,bok,kolor,ile:integer);
var
   i,j:integer;
   x1,y1:integer;
begin
```

```
  for i:=1 to ile do
  for j:=1 to ile do
  begin
     x1:=x+(i-1)*bok;
     y1:=y+(j-1)*bok;
     KWA(x1,y1,bok,kolor,clGray);
  end;
end;

procedure TForm1.RYSUJ_ROBACZKI();
var
  k,w:integer;
  ile,bok:integer;
  x1,y1:integer;
begin
  ile:=Scrollbar1.position;
  bok:=round(600 / ile);

  for k:=1 to ile do
  for w:=1 to ile do
  begin
    x1:=(k-1)*bok+10;
    y1:=(w-1)*bok+10;

    if ROBACZKI[k,w]=1
      then KWA(x1,y1,bok,clBlack,clWhite)
          else KWA(x1,y1,bok,clBlack,clGray);
  end;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  POLICZ_ROBACZKI();
  NOWE_POKOLENIE();
  RYSUJ_ROBACZKI();
end;

end.
```