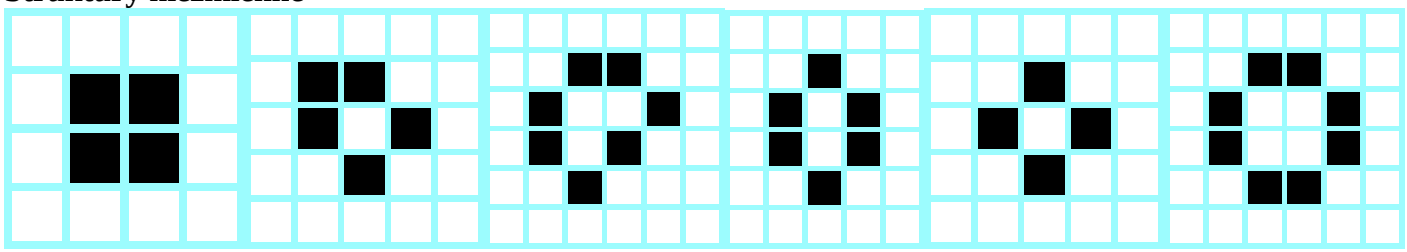
Gra w życie

To jeden z pierwszych przykładów tzw. automatu komórkowego wymyślonego w 1970 roku przez brytyjskiego matematyka Johna Conwaya. Dzięki kilku prostym regułom powstają skomplikowane struktury, co daje możliwość badania i symulowania dużo bardziej skomplikowanych układów z matematyki, fizyki, ekonomii, biologii.

Opis gry (za Wikipedią)
Gra toczy się na nieskończonej planszy (płaszczyźnie) podzielonej na kwadratowe komórki. Każda komórka ma ośmiu "sąsiadów", czyli komórki przylegające do niej bokami i rogami. Każda komórka może znajdować się w jednym z dwóch stanów: może być albo "żywa" (włączona), albo "martwa" (wyłączona). Stany komórek zmieniają się w pewnych jednostkach czasu. Stan wszystkich komórek w pewnej jednostce czasu jest używany do obliczenia stanu wszystkich komórek w następnej jednostce. Po obliczeniu wszystkie komórki zmieniają swój stan dokładnie w tym samym momencie. Stan komórki zależy tylko od liczby jej żywych sąsiadów. W grze w życie nie ma graczy w dosłownym tego słowa znaczeniu. Udział człowieka sprowadza się jedynie do ustalenia stanu początkowego komórek.

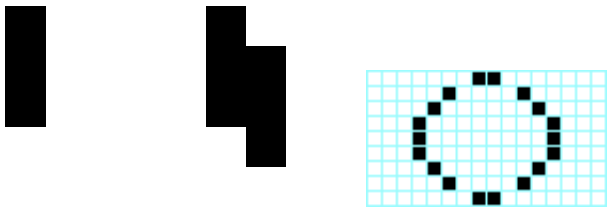
Reguły gry według Conwaya
Martwa komórka, która ma dokładnie 3 żywych sąsiadów, staje się żywa w następnej jednostce czasu (rodzi się). Żywa komórka z 2 albo 3 żywymi sąsiadami pozostaje nadal żywa; przy innej liczbie sąsiadów umiera (z "samotności" albo "zatłoczenia").

Struktury niezmienne



Klocek Łódź Bochenek Kryształ Koniczynka Staw

Oscylatory
Oscylatory zmieniają się okresowo, co pewien czas powracają do swojego stanu pierwotnego; najprostsza taka struktura składa się z trzech żywych komórek położonych w jednym rzędzie. Najprostsze z nich dość często pojawiają się jako produkty końcowe ewolucji struktur.

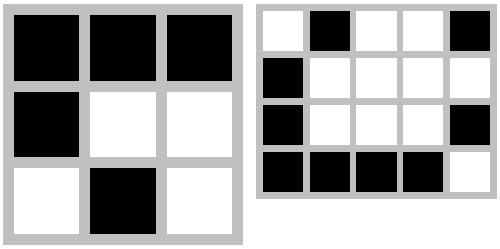


Blinker Żabka Krokodyl

Układy nieśmiertelne

Po okresie przejściowym tworzą podobne (takie same?) "dymiące lokomotywy" zostawiające zygzakowatą linię kwadratów i układ nieśmiertelny w jednej linii

Statki
Tzw. "statki" zwykle zmieniają się okresowo – choć okresy nie przekraczają jednak najczęściej kilkunastu kroków czasowych – ale wraz z każdym cyklem przesuwają się o stałą liczbę pól po planszy w określonym kierunku. Glider (szybowiec) i LWSS (Dakota)



LEKCJA 01 **SKALOWANA SZACHOWNICA**

Szachownica rysowana jest na formatce w obszarze CANVAS za pomocą linii. Tyle samo linii (pól-1) w pionie i poziomie.
X,Y - lewy górny róg szachownicy
O - odstęp pomiędzy liniami
P - ilość pól szachownicy

Na formatce mamy możliwość ustawienia ilości pól szachownicy za pomocą paska SCROLLBAR.
Chcemy żeby szachownica miała zawsze 600 pikseli, dlatego odstęp między liniami obliczymy za pomocą wzoru: O=600 div P

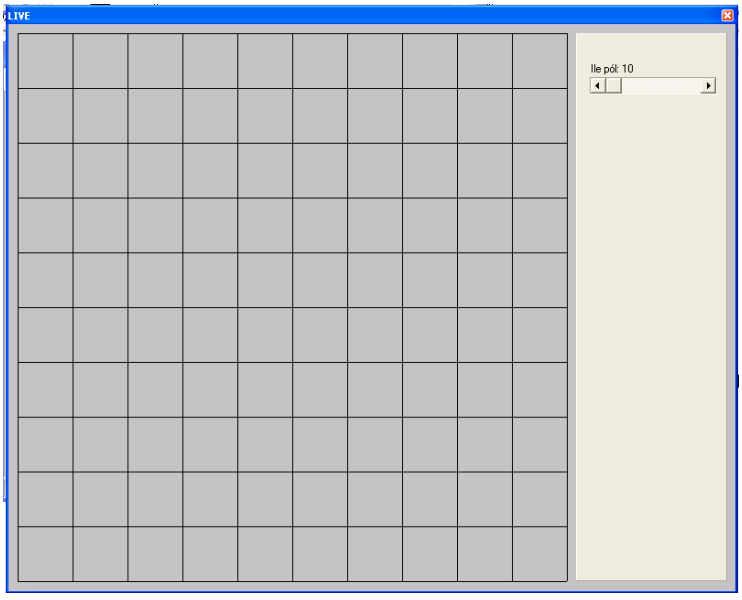
```
procedure TForm1.Szachownica(x,y,o,p,k:integer);
var i:integer;
begin
  with canvas do
  begin
    pen.color:=K;
    for i:=0 to P do
    begin
      MoveTo(X+i*O,Y);
      LineTo(X+i*O,Y+P*O);
      MoveTo(X,Y+i*O);
      LineTo(X+P*O,Y+i*O);
    end;
  end;
end;
```

```
procedure TForm1.FormCreate
          (Sender: TObject);
begin
  SzX:=10;
  SzY:=10;
  SzI:=10;
  SzO:=600 div SzI;
  SzK:=clBlack;
  Form1.Height:=645;
  Form1.Width:=800;
  Form1.BorderStyle:=bsToolWindow;
  Form1.Color:=clSilver;
  Form1.Caption:='LIVE';
  Panel1.Caption:='';
  Panel1.Left:=620;
  Panel1.Top:=10;
  Panel1.Width:=165;
  Panel1.Height:=600;
```

W procedurze FORMCREATE ustawiamy wartości początkowe dla formatki i program

- lewy górny róg szachownicy, zawsze w tym samym miejscu

- ile pól na początku (potem można zmienić)
- początkowa szerokość jednego pola (cała szachownica 600)
- kolor linii
- wysokość całej formatki (szachownica + pasek)
- szerokość formatki (szachownica + panel sterowania)
- można przesuwać, bez skalowania, wąski pasek tytułu)
- kolor tła formatki
- tytuł formatki
- brak tytułu na panelu sterującym
- pozycja panelu sterującego



Na panelu sterującym umieszczamy pasek SCROLLBAR – ustawianie ilości pól i dwa pola LABEL. W pierwszym napis „Ile pól”, w drugim ilości pól wybrana na pasku ScrollBar.

W procedurze FORMRESIZE umieszczamy przerysowywanie szachownicy ze zmiennymi globalnymi

```
procedure TForm1.FormResize(Sender: TObject);
begin
  Szachownica(SzX,SzY,SzO,SzI,SzK);
end;
```

Po zmianie na pasku SCROLLBAR zostaną obliczone nowe wartości zmiennych globalnych i szachownica będzie przerysowana.
- wymazanie starej szachownicy - kolor tła
- nowe ilość pól do pola LABEL
- nowa szerokość pola
- rysowanie nowej szachownicy

```
procedure TForm1.ScrollBar1Change(Sender: TObject);
begin
  Szachownica(SzX,SzY,SzO,SzI,clSilver);
  Label2.Caption:=IntToStr(ScrollBar1.Position);
  SzI:=ScrollBar1.Position;
  SzO:=600 div SzI;
  Szachownica(SzX,SzY,SzO,SzI,SzK);
end;
```

LEKCJA 02 - **MYSZKĄ W POLA**

Kliknięcie w pole narysowanej szachownicy spowoduje zamalowanie pola. Zamalowanie wnętrza pola za pomocą linii, o jeden piksel mniej, aby pozostały widoczne ramki.
Pozycję myszy w momencie kliknięcia odczytamy za pomocą zdarzenia ONMOUSEDOWN i odpowiedniej dla niej procedury.

```
procedure TForm1.Pole
(x,y,bok,kolor:integer);
var i:integer;
begin
  with canvas do
  begin
    pen.color:=kolor;
    for i:=1 to bok-1 do
    begin
      MoveTo(x+i,y+1);
      LineTo(x+i,y+bok);
    end;
  end;
end;
```

Zamalowanie pola – pionowe linie o długości bok
x,y – lewy górny róg pola
bok – ile pikseli w pionie (i w poziomie)
kolor – kolor linii, którymi zamalowujemy

Które pole zostało wybrane - kliknięte
Za pomocą zdarzenia ONMOUSEDOWN pobierzemy współrzędne punktu na formatce, który został kliknięty. Przeliczymy te współrzędne na numer wiersza i kolumny na szachownicy, a następnie wyliczone zostaną współrzędne początkowe pola do zamalowania.

```
procedure TForm1.FormMouseDown
        (Sender: TObject; Button: TMouseButton;
        Shift: TShiftState; X, Y: Integer);
var MyK,MyX,MyY,Xsz,Ysz:integer;
begin
  MyX:=((X-10) div SzO)+1;
  MyY:=((Y-10) div SzO)+1;
  Xsz:=10+SzO*(MyX-1);
  Ysz:=10+SzO*(MyY-1);
  pole(Xsz,Ysz,SzO,SzP);
end;
```

MyX, MyY – numer kolumny i wiersza

Xsz, Ysz – lewy górny róg pola do zamalowania

malujemy pole

SzP – dodatkowa zmienna globalna, kolor pola po zamalowaniu

Jeślibyśmy chcieli, aby ponowne kliknięcie zamalowywało w innym kolorze (lub kolorze tła) powinniśmy rozpoznawać kolor kliknięty piksela. Jest to możliwe **kolor:=canvas.Pixels[X,Y];** jednak w prawdziwych grach stosuje się raczej tablice i w nich zapisuje się aktualny stan pól gry.

LEKCJA03 - **ZMIANA KOLORÓW – TABLICE**

```
TABLICA: array[1..200,1..200] of integer;
```

Wszystkie potrzebne dane zapamiętamy w tablicy. Jest to najwygodniejszy sposób. Zawsze możemy odwołać się do konkretnej komórki niezależnie od tego, co na ekranie.
W tablicy zapamiętamy kolor punktu:
0 – kolor tła - clSilver
1 – kolor pola - clWhite
Za każdym razem jak klikniemy w pole, to zmieniamy zawartość tablicy i przerysowujemy.

```
procedure TForm1.ZerowanieTablicy;
var i,j:integer;
begin
  for i:=1 to 200 do
    for j:=1 to 200 do
      Tablica[i,j]:=0;
end;
```

Deklaracja tablicy i procedura zerowania tablicy – kolor tła na początku.

Przeliczanie współrzędnych kliknięgo punktu na współrzędne lewego górnego roku pola na szachownicy będziemy dość często wykonywać, dlatego zapiszemy je w postaci funkcji

WSPnaTAB - współrzędne myszki na numery (wiersza lub kolumny) pól szachownicy
TABnaWSP – numer wiersza (lub kolumny) przeliczamy na współrzędne lewego górnego rogu pola szachownicy

```
function TForm1.WSPnaTAB(x:integer):integer;
begin
  WSPnaTAB:=((x-10) div SzO)+1;
end;

function TForm1.TABnaWSP(x:integer):integer;
begin
  TABnaWSP:=10+SzO*(x-1);
end;
```

Zmiany w procedurze do zdarzenia **ONMOUSEDOWN**
- wyliczanie współrzędnych

- sprawdzamy tablicę po każdym kliknięciu
- jeżeli komórka tablicy miała wartość 0 (tło) to malujemy pole na biało i zmieniamy wartość komórki na 1

- w przeciwnym razie (gdy komórka miała wartość 1) malujemy pole na kolor tła i komórka – wartość 0

```
var My[illegible]
begin
  TaX:=WSPnaTAB(x);
  TaY:=WSPnaTAB(y);
  Xsz:=TABnaWSP(TaX);
  Ysz:=TABnaWSP(TaY);
  if Tablica[TaX,TaY]=0
  then begin
     pole(Xsz,Ysz,SzO,clWhite); Tablica[TaX,TaY]:=1;
    end
  else begin
     pole(Xsz,Ysz,SzO,clSilver); Tablica[TaX,TaY]:=0;
    end;
end;
```

Takie działanie wystarczyłoby, gdyby użytkownik nie zmieniał szachownicy (ilość pól). Po każdej zmianie należy jeszcze raz przerysować całą szachownicę i wszystkie pola zgodnie z zawartością TABLICY – procedura **RYSOWANIETABLICY**.

```
procedure TForm1.RysowanieTablicy;
var i,j:integer;
begin
  for i:=1 to SzI do
    for j:=1 to SzI do
      if Tablica[i,j]=0
        then pole(TABnaWSP(i),TABnaWSP(j),SzO,SzT)
        else pole(TABnaWSP(i),TABnaWSP(j),SzO,SzP);
end;
```

Procedurę wstawiamy do SCROLLBAR1CHANGE na końcu. i na początku jeszcze jedna instrukcja: **FORM1.REPAINT**, która powoduje wyczyszczenie całej formatki – na czystej można rysować od początku.

LEKCJA04 - **ROBACZKI**

Gra toczy się w turach. Co to znaczy? Na ekranie widać plemię robaczków. Dokładnie taka sama sytuacja odwzorowana jest w tablicy. Kolejne pokolenie pojawi się po wyliczeniu położenia nowego pokolenia robaczków i narysowaniu na nowo całej tablicy. Jak tworzy się nowe pokolenie?

Badany jest teren w koło każdego robaczka (pola), tzn. 8 sąsiednich pól wokół.

Jeśli wokół pustego pola znajduje się dokładnie 3 robaczki, to rodzi się na tym polu nowy robaczek.

Jeśli wokół pola z robaczkiem jest dokładnie 2 lub 3 robaczki – to ten robaczek przeżywa.

W każdym innym przypadku, gdy wokół pola z robaczkiem jest 0,1,4,5,6,7,8 innych – robaczek ginie (z samotności lub z zatłoczenia).

Funkcja zliczająca robaczki wokół wybranego pola – wokół komórki tablicy **ILEROBACZKOW**

Jako parametry podajemy współrzędne pola

W zmiennej ILE przechowujemy ilość robaczków wokół pola, a skoro komórka z robaczkiem zawiera jedynkę, a pole puste zero, wystarczy podsumować odpowiednie komórki. Dodatkowo sprawdzamy, czy kolejna komórka nie jest poza brzegiem tablicy.

```
function TForm1.IleRobaczkow(x,y:integer):integer;
var ile:integer;
begin
  ile:=0;
    if (x>0)                then ile:=ile+tablica[x-1,y   ];
    if (x>0) and (y>0)      then ile:=ile+tablica[x-1,y-1];
    if (y>0)                then ile:=ile+tablica[x   ,y-1 ];
    if (x<SzI) and (y>0)    then ile:=ile+tablica[x+1,y-1];
    if (x<SzI)              then ile:=ile+tablica[x+1,y   ];
    if (x<SzI) and (y<SzI) then ile:=ile+tablica[x+1,y+1];
    if (y<SzI)              then ile:=ile+tablica[x   ,y+1];
    if (x>0) and (y<SzI)    then ile:=ile+tablica[x-1,y+1];
  IleRobaczkow:=ile;
end;
```

Myszką klikamy w pola i ustawiamy początkowe położenie robaczków. Tworzymy dodatkowy przycisk i po kliknięciu w niego będzie wyliczona kolejna tura i narysowane nowe pokolenie robaczków.

Sprawdzamy całą tablicę TABLICA (wszystkie komórki), a nowe pokolenie zapisujemy chwilowo do nowej tablicy T1 (zadeklarować).

- ilość robaczków wokół pola (X,Y)
- pole puste i 3 robaczki wokół – rodzi się
- pole zajęte i 2 lub 3 wokół – bez zmian
- pole zajęte i za mało lub za dużo wokół – robaczek umiera
- po sprawdzeniu całej tablicy przepisujemy nowe pokolenie na tablicę główną
- i rysujemy nowe pokolenie

```
procedure TForm1.Button1Click(Sender: TObject);
var i,j,ile:integer;
begin
  t1:=tablica;
  for i:=1 to SzI do
    for j:=1 to SzI do
    begin
      ile:=IleRobaczkow(i,j);
      if (tablica[i,j]=0) and (ile=3) then t1[i,j]:=1;
      if (tablica[i,j]=1) and (ile in [2,3]) then t1[i,j]:=1;
      if (tablica[i,j]=1) and (ile in [0,1,4,5,6,7,8]) then t1[i,j]:=0;
    end;
    tablica:=t1;
    RysowanieRablicy;
end;
```

Automatyzacja

- wstawić przycisk z nazwą START
- wstawić TIMER
- w procedurze FORMCREATE wyłączyć na początku timera - **Timer1.Enabled:=false;**
- w zdarzeniu ONTIMER ustawiamy procedurę BUTTON1CLICK – nowe pokolenie. Gdy włączymy timera, to co 1 sekundę będzie wyliczane i rysowane nowe pokolenie.
- w procedurze BUTTON2CLICK dla nowego przycisku włączanie i wyłączanie timera oraz zmiana nazwy przycisku
- nowy przycisk z nazw KONIEC i instrukcją **close;**

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  if Timer1.Enabled=false
  then begin
    Button2.caption:='STOP';
    Timer1.Enabled:=true;
  end
  else begin
    Button2.caption:='START';
    Timer1.Enabled:=false;
  end;
end;
```