# Gra w życie -
# automat komórkowy

# 1. Wstęp teoretyczny

Gra w życie jest przykładem automatu komórkowego. Została ona wymyślona w 1970 roku przez brytyjskiego matematyka - Johna Hortona Conwaya. Odbywa się ona na kwadratowej siatce komórek, gdzie każda komórka może być albo żywa, albo martwa.

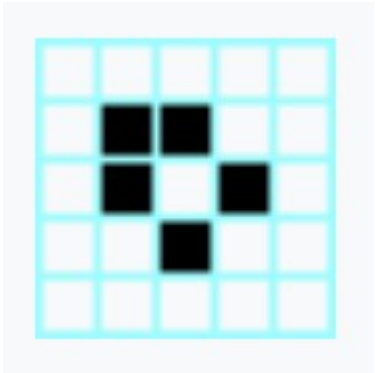
Ich zachowaniem kierują następujące reguły:

- martwa komórka ożywa, gdy ma dokładnie trzech żywych sąsiadów

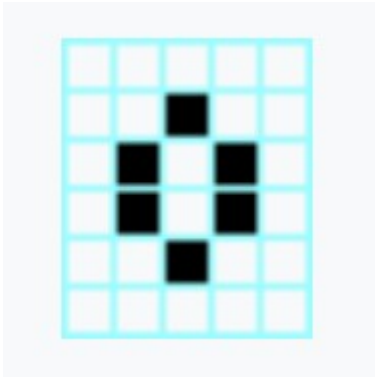
- żywa komórka przeżywa, gdy ma dwóch lub trzech sąsiadów żywych

- żywa komórka umiera, jeśli ma więcej niż trzech sąsiadów (z przeludnienia) lub gdy ma mniej niż dwóch sąsiadów (z samotności)

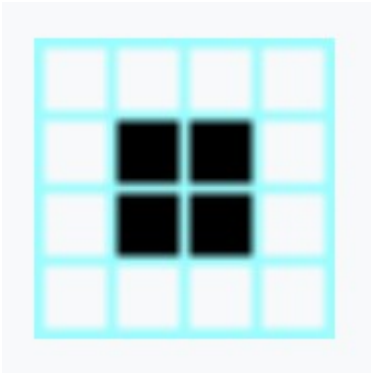
W „Grze w życie" występuje wiele różnych struktur niezmiennych, takich jak:



*Rysunek 2 - łódź*

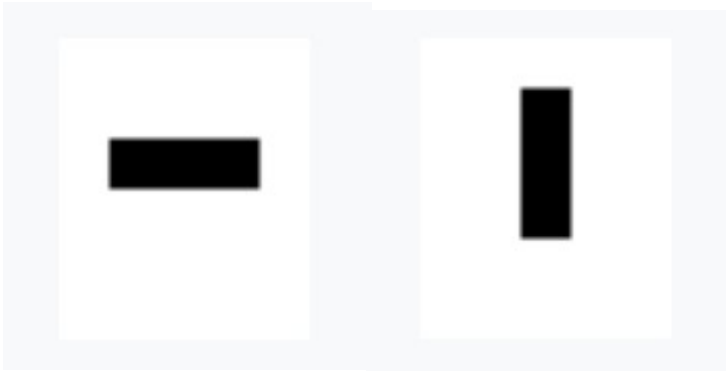


*Rysunek 3 - kryształ*



*Rysunek 1 - Blok*

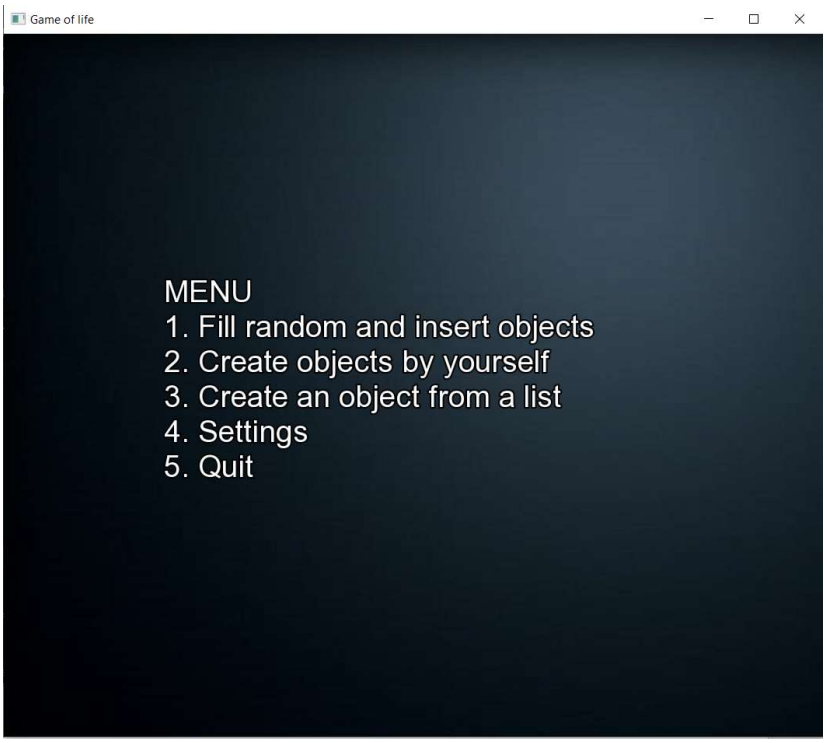
Poza obiektami stałymi można wyróżnić jeszcze oscylatory: są to wzory zmieniające się okresowo, co pewien czas powracające do stanu pierwotnego. Okresy oscylatorów najczęściej przyjmują wartości 2, 3, 4, 6 lub 8, choć w *grze w życie* znaleziono i takie, których okres wynosi prawie 150000. Przykładem oscylatora może być Blinker, który występuje w następujących fazach:



Jak widać, okres Blinkera wynosi 2, ponieważ po upłynięciu jednego kwantu czasu wraca on do swojej pozycji startowej.

## 2. Opis działania programu

Program został napisany w języku C++ używając biblioteki SFML. Jest to prosta biblioteka stworzona do projektowania wydajnych gier 2D, dostarczając przy tym niezwykle prosty i intuicyjny interfejs. Menu główne programu:

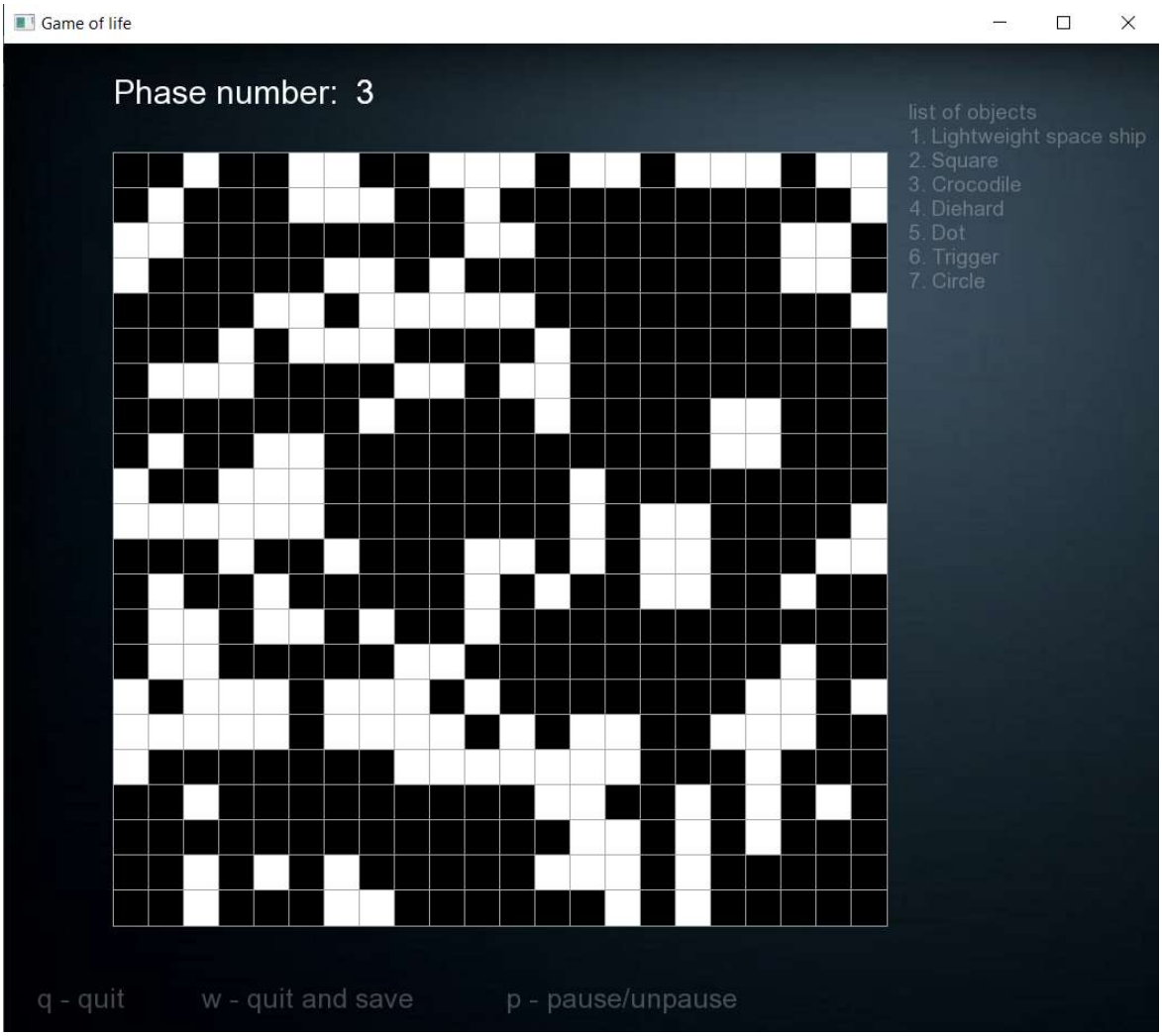



Opis poszczególnych sekcji:

1. „Fill random and insert objects” – zostaje utworzona nowa plansza z losowo wygenerowanymi komórkami. Dodatkowo, istnieje możliwość wstawienia wbudowanych obiektów do planszy.
2. „Create objects by yourself” – zostaje utworzona czysta plansza, w której użytkownik może tworzyć własne obiekty myszą. Utworzony obiekt może zostać zapisany do pliku, by później otworzyć go z sekcji „Create an object from a list”, lub można rozpocząć symulację nowo utworzonego obiektu.
3. „Create an object from a list” – użytkownik zostaje przeniesiony do drugiego menu, w którym to może wybrać który z wcześniej przygotowanych obiektów go interesuje – m.in. statek, pulsar.. Z listy można również wybrać odtworzenie wcześniej narysowanego obiektu
4. Settings – użytkownik zostaje przeniesiony do menu, w którym ma do wyboru opcje zmiany czasu pomiędzy każdą turą, jak i opcje do zmiany reguł Gry w życie.
5. Quit – wyjście.

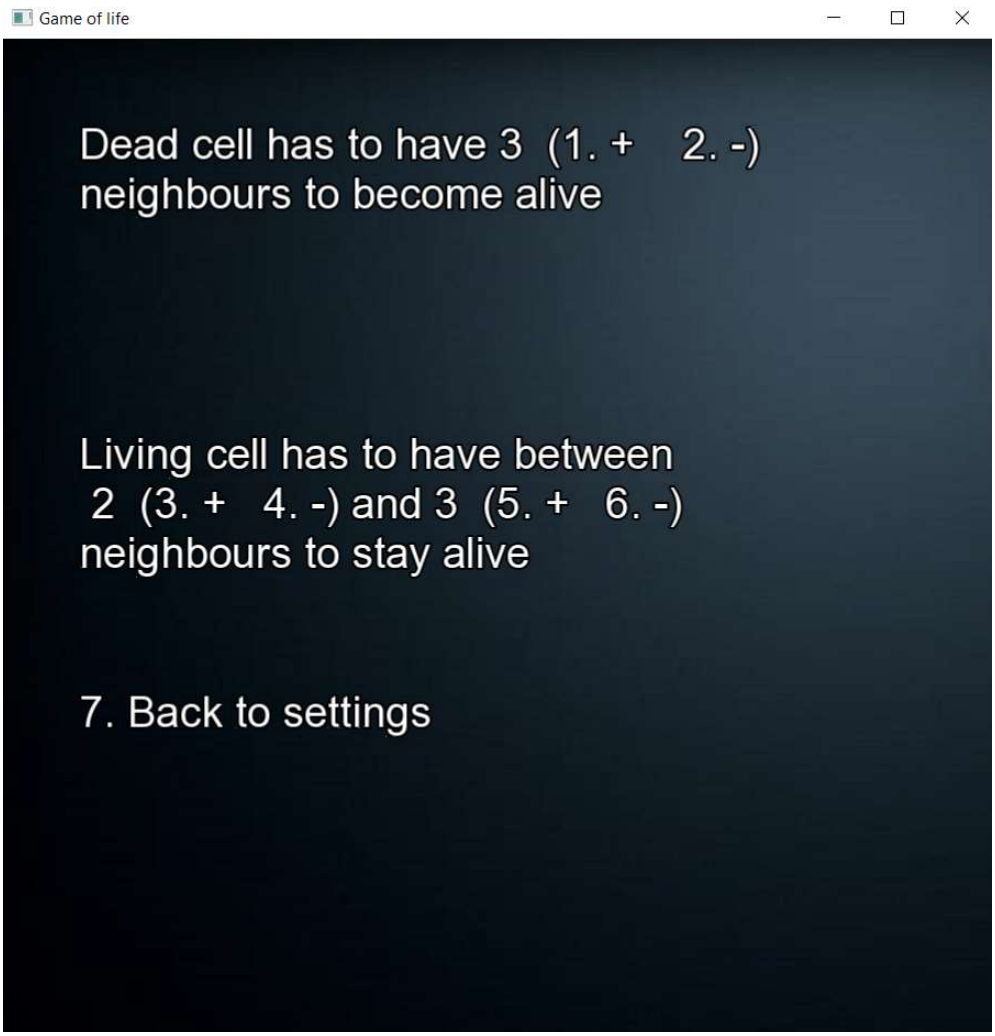
## 3. Dodatkowy opis funkcjonalności

Program pozwala na uruchomienie wybranego modułu, np. Fill random, spauzowanie go, następnie wyjście z niego, zmianę reguł/szybkości i powrót do momentu, w którym użytkownik opuścił dany moduł. Opuszczenie planszy przyciskiem **q** skutkuje zresetowaniem stanu sekcji, a opuszczenie przyciskiem **w** spowoduje zapamiętanie ostatnio wykonanego ruchu. Działa to do wszystkich modłów.




Jak widać na załączonym zrzucie ekranu, podczas trwania gry w życie istnieje możliwość dodania wbudowanych obiektów takich jak kwadrat, statek, kółko itd.

Przechodząc do ustawień zasad gry w życie, użytkownik może przyciskami **1** i **2** odpowiednio zwiększyć i zmniejszyć liczbę sąsiadów zmarłej komórki do ożywienia jej. Podobna sytuacja dzieje się w drugiej części; dla żywej komórki, jednak teraz użytkownik do wyboru ma zakres od jakiej ilości sąsiadów żywa komórka umrze z przeludnienia bądź samotności.




## 4. Opis klas, metod i funkcji

Program składa się z 3 klas.

- Game – to tutaj są przeprowadzane wszystkie mechanizmy logiczne: obliczanie i ustalanie ilości komórek w następnej turze, jak i ożywianie komórek za pomocą przycisków. W funkcji znajduje się wiele metod typu GET/SET, dlatego zostaną one pominięte w dalszej części opisu kodu.

- UI – klasa która nie może posiadać żadnego obiektu, wszystkie jej metody są statyczne i służy głównie w celach komunikacji z użytkownikiem: wyświetlanie menu, ustawień, tekstu, itd.

- Objects – klasa składająca się z wbudowanych do niej wcześniej obiektów. Są one wykorzystywane np. w module nr 1, lub module nr 3.

W klasie Game znajdują się następujące publiczne metody:

- void set_blank() – zeruje planszę

- void fill_random() – losowo wypełnia planszę

- void print(sf::RenderWindow& window, Game& g) – rysuje plansze

- void calc_next_phase() – oblicza ilość żywych sąsiadów poszczególnej komórki

- void do_next_phase() – na podstawie poprzedniej metody ustawia komórkę na żywą lub martwą

- void set_alive_by_mouse(sf::RenderWindow& window, bool leftClick) – umożliwia ożywianie komórek za pomocą myszki

- void save_to_file() – pozwala na zapis stworzonej planszy do pliku

- void read_from_file() – zezwala na odczytanie planszy z pliku do programu

W klasie UI znajdują się następujące metody do komunikacji z użytkownikiem:

static void print_main_menu(sf::RenderWindow& window);

static void print_list(sf::RenderWindow& window);

static void print_drawing(sf::RenderWindow& window);

static void print_quit(sf::RenderWindow& window);

static void print_background(sf::RenderWindow& window);

static void print_number_of_phases(sf::RenderWindow& window, Game& g1);

static void print_settings(sf::RenderWindow& window, Game& g1);

static void print_speed_settings(sf::RenderWindow& window, int SLEEP_TIME_MS);

static void print_rules_settings(sf::RenderWindow& window, Game& g1);

static void print_file_saved(sf::RenderWindow& window);

static void print_add_objects(sf::RenderWindow& window, Game& g1);