# Sprawozdanie z projektu
# Programowanie Komputerów

Temat: Gra w życie

Autor: Adam Ogiba
Semestr: piąty
Grpua: 2 (BDiS)

Prowadzący: mgr inż. Grzegorz Kwiatkowski

# 1. Analiza rozwiązania

Założeniem programu realizującego grę w życie jest stworzenie symulacji jednego z najbardziej popularnych automatu komórkowego, który został wymyślony przez brytyjskiego matematyka Johna Conwaya. Głównym zadaniem programu jest symulowanie automatu komórkowego oraz na sterowanie symulacją w wygodny sposób dla użytkownika, tak aby mógł mieć wpływ na działanie automatu. Program ma zostać zrealizowany używając konsoli oraz wydajnie przeprowadzać symulacje nawet dla około jednego miliona komórek.

Program uruchamiany jest zawsze w trybie interaktywnym, który na bieżąco wymaga interakcji z użytkownikiem. Wynika to z tego, że przeprowadzana symulacja nigdy się nie kończy i użytkownik może ją edytować w trakcie działania, dlatego nie ma możliwości aby program działał na zasadzie jego uruchomienia i zwrócenia gotowego wyniku. Do programu może zostać przekazany plik wejściowy w postaci ASCII, który po załadowaniu służy do przywrócenia stanu symulacji zapisanej w pliku. Praca z programem kończy się kiedy użytkownik zakończy symulacje i wyjdzie z programu. Program na bieżąco wyświetla wyniki symulacji dla użytkownika.

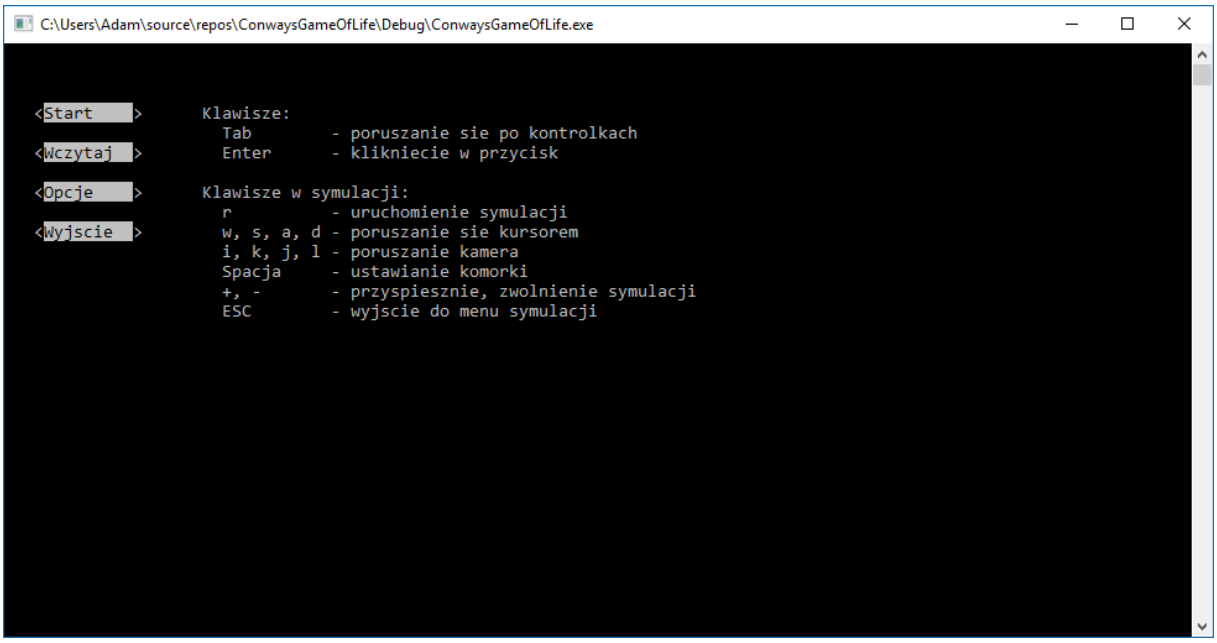
# 2. Specyfikacja zewnętrzna

Program należy uruchomić z linii poleceń (cmd.exe) w następujący sposób:

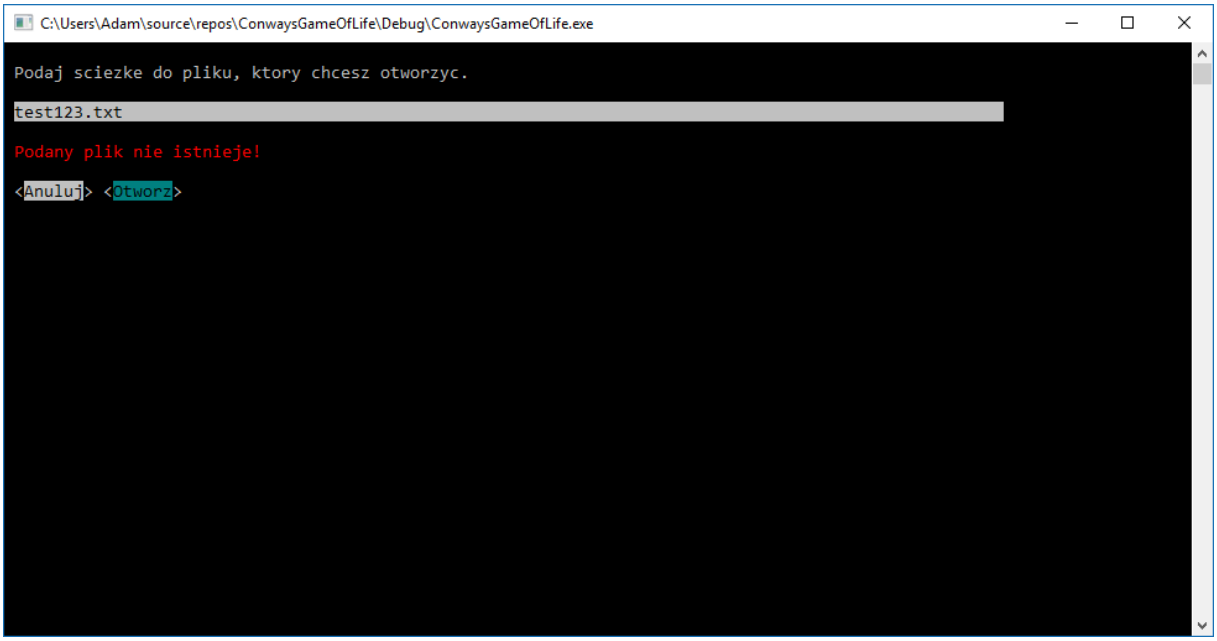
```
Uzycie:
  ConwaysGameOfLife.exe
  ConwaysGameOfLife.exe -h | --help
  ConwaysGameOfLife.exe <saved-simulation-file>

Argumenty:
  saved-simulation-file   Zapisany plik symulacji.

Opcje:
  -h --help               Pokazuje pomoc.
```

Obsługiwana jest tylko linia poleceń systemu Windows, ponieważ program został zaimplementowany używając wydajnych funkcji systemowych służących do bezpośredniej manipulacji konsolą. Program można uruchomić z dowolnego katalogu, lecz należy pamiętać, że kiedy będziemy podawać relatywną ścieżkę do pliku to jej początek będzie zależeć od katalogu, z którego został uruchomiony program.

Po uruchomieniu bez parametrów program wyświetli menu służące do poruszania się po opcjach programu. Program uruchomi się w trybie pełnoekranowym na systemie Windows 10. Na ekranie, po prawej zostanie ukazana instrukcja opisująca poszczególne klawisze na klawiaturze oraz ich przeznaczenie. Aby poruszać się po menu używamy klawisza `TAB` oraz `ENTER`. Klawisz `TAB` pozwala nam na przechodzenie pomiędzy kolejnymi opcjami w menu, a klawisz `ENTER` wybiera daną opcję, jeśli wybrana opcja na to pozwala. Dodatkowo podczas przechodzenia menu pojawiają się pola, w które można wpisywać tekst. Obsługiwane są wtedy wszystkie pozostałe klawisze służące do wprowadzania tekstu. Zaznaczone pole jest podświetlane poprzez rozjaśnienie lub zmianę koloru. Dodatkowo w instrukcji opisane są klawisze używane podczas symulacji automatu komórkowego.

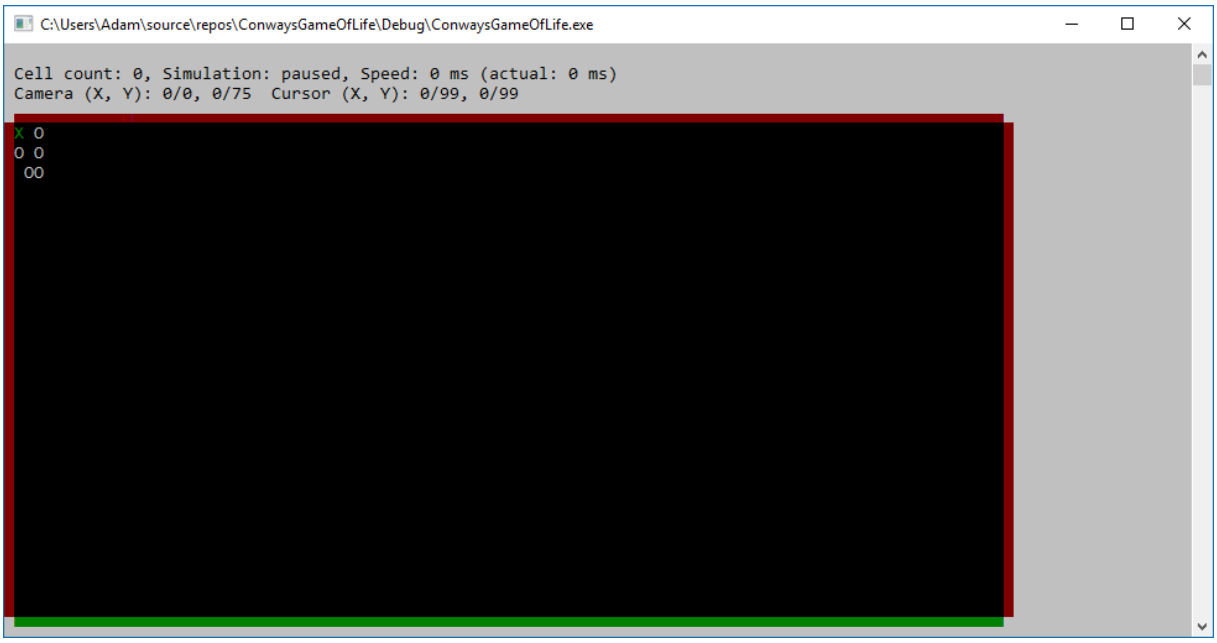



Program wyświetla komunikaty z błędami na ekranie w momencie wystąpienia danego błędu. Komunikaty są czytelne dla użytkownika i tłumaczą jaki błąd wystąpił.



Po wciśnięciu przycisku start, program przechodzi w widok wyświetlający symulacje automatu komórkowego, która wcześniej może zostać sparametryzowana w widoku opcji. W symulacji obsługiwane są dodatkowe klawisze, służące do sterowania symulacją, takie jak:

- Klawisz `R` - służy do zatrzymywania lub uruchamiania symulacji
- Klawisze `W S A D` – służą do poruszania kursorem oznaczony wielką literą X pozwalającym na dodawania lub usuwanie komórek w symulacji
- Klawisze `I K J L` – służą do poruszania kamerą, używane w momencie, kiedy wielkości siatki symulacji przekracza wielkość ekranu konsoli
- Klawisz `SPACJA` – pozwala na usunięcie lub dodanie komórki w miejscu, w którym umieszczony jest kursor
- Klawisze `+ -` – odpowiednio służą do przyśpieszania lub zwalniania symulacji, klawisze te nie są klawiszami umieszczonymi na klawiaturze numerycznej
- Klawisz `ESC` – służy do zatrzymania symulacji i przejścia do menu symulacji




Podczas symulacji zobaczymy na środku główny ekran obrazujący aktualny stan automatu komórkowego. Na czarnym tle za pomocą znaku wielkiego `O` wyświetlane są żywe komórki. Każde puste miejsce na czarnym tle jest martwą komórką. Na głównym ekranie zobaczymy również wielki znak `X` oznaczający aktualne położenie kursora. Może on przyjąć dwa stany oznaczane kolorem zielonym lub czerwonym. Kiedy kursor jest koloru zielonego, oznacza to, że pod nim znajduje się komórka martwa, a wciśnięcie klawisza `SPACJA` oznacza dodanie komórki (ożywienie jej) w tym miejscu. Kiedy kursor jest koloru zielonego to pod nim znajduje się komórka żywa i wciśnięcie klawisza `SPACJA` powoduje usunięcie tej komórki (uczynienie jej martwą).

Na bokach głównego ekranu zobaczymy paski w kolorach czerwonym lub zielonym. Kolor czerwony mówi nam, że to jest już krawędź siatki symulacji i nie możemy przesunąć kamery dalej w tamtą stronę. Kolor zielony oznacza, że w tą stronę siatka symulacji nie jest rysowana w pełni z powodu ograniczenia wielkości konsoli, dlatego możemy przesunąć kamerę w daną stronę

Na górnym pasku informacyjnym nad ekranem głównym zobaczymy dodatkowe informacje dla użytkownika takie jak:

- Cell Count – ilość aktualnie żywych komórek w symulacji
- Simulation: paused/running – mówi nam, czy aktualnie symulacja jest uruchomiona, czy wstrzymana
- Speed: 1000 ms (actual: 1000 ms) – określa jaka jest aktualna prędkość symulacji w milisekundach (mówi co ile milisekund następuje następny krok symulacji), wartość podana w nawiasie pokazuje rzeczywisty czas pomiędzy krokami symulacji, jest ona wyświetlana aby w momencie wolniejszego działania programu ujrzeć rzeczywisty czas z jakim jest odświeżana symulacja
- Camera (X, Y): 4/10, 3/20 – mówi nam w jakim położeniu X, Y aktualnie znajduje się kamera, wartości podane po ukośniku są wartościami maksymalnymi przemieszczenia kamery
- Cursor (X, Y): 0/20, 0/10 - mówi nam w jakim położeniu X, Y aktualnie znajduje się kursor, wartości podane po ukośniku są wartościami maksymalnymi przemieszczenia kursora

Uruchomienie programu z parametrem zapisanego pliku symulacji, program odczyta plik i od razu uruchomi symulację.

# 3. Specyfikacja wewnętrzna

Program został stworzony używając C++11, który pozwala na łatwiejsze zarządzanie pamięcią poprzez użycie `unique_ptr<T>, shared_ptr<T>` oraz `weak_ptr<T>`. Zarządzanie pamięcią zostało oparte głównie na klasie `shared_ptr<T>`, która dodatkowo została opakowana w klasę `Ref<T>` aby ułatwić pracę z wskaźnikami współdzielonymi. Dało to efekt stworzenia kodu C++, który przypominał kod zarządzany z innych języków wyższego poziomu. Podczas implementacji należało, głównie pamiętać oraz uważać aby nie stworzyć tzw. *reference cycle*, które spowodowało by uniemożliwienie zniszczenia danego obiektu. W momencie planowania klas, należało zawsze zastano wić się czy nie tworzymy cyklu referencji, a jeśli tak to wtedy należało wykorzystać `weak_ptr<T>` opakowany w klasę `WeakRef<T>`. Wykorzystane zostały również też inne techniki wykorzystywane w językach wyższego poziomu ułatwiające prace z klasą `Ref<T>`. Sposób ten nie jest w pełni optymalny, ale zapewnia bardzo dużą wygodę oraz pozwala na pracę z językiem C++, jak z językiem jeszcze wyższego poziomu. Podczas pisania kodu nie trzeba było martwić się zupełnie o zwalnianie pamięci, ponieważ zawsze było robione to automatycznie. Należało tylko bardzo dobrze zaplanować swoje klasy aby nie stworzyć cyklu referencji.

Specyfikacja wewnętrzna została wygenerowana przez program Doxygen. Została ona umieszczona wraz z plikami źródłowymi projektu w folderze html.

# 4. Testowanie

Testowanie aplikacji odbywało się poprzez sprawdzanie trybu interaktywnego i poprawności symulacji automatu komórkowego. Nie były przeprowadzane testy jednostkowe dla konkretnych metod. Testowanie nie mogło odbywać się dla konkretnych przypadków, gdyż program sam w sobie nie przyjmuje danych z zewnątrz oprócz pliku zapisanej wcześniej symulacji.