Programowanie obiektowe	Implementacja
Gra w Życie (Conway's Game of Life) i jej wariacje	
Wtorek, 12.15 **2012/2013**	**Dariusz Jędrzejczak** **Ewelina Otto**

Opis funkcjonalności

Program realizuje automat komórkowy znany jako Game of Life (Gra w Życie) – implementacja pozwala ponadto na realizację szerszej klasy automatów komórkowych.

Sterowanie odbywa się za pomocą interfejsu tekstowego (z konsoli) przy użyciu komend, o których będziemy dalej mówić jako o poleceniach języka skryptowego programu/gry. Opis języka znajduje się w dołączonym pliku readme.

Gra jest symulacją, interakcja z użytkownikiem (poza ustaleniem warunków początkowych i uruchomieniem) jest opcjonalna i polega na zmianie przez niego parametrów symulacji. Użytkownik może uruchamiać i zatrzymywać symulację dowolną ilość razy, za każdym razem na dowolną ilość kroków czasowych. Przez cały czas użytkownik może wpływać na stan planszy w grze za pomocą poleceń naszego języka.

Gra polega na obserwacji ewolucji zachodzącej na planszy zgodnie z zasadami. Gracz może obserwować rozmaite wzorce, z których w zwykłej wersji gry bardzo wiele jest nazwanych i sklasyfikowanych (http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life). W naszej wersji gracz może dodatkowo zmieniać zasady obowiązujące dla niektórych (dowolnie wybranych) komórek.

Zmiany

Funkcjonalność jest w większości taka, jaka była zakładana. Zaszły zmiany w strukturze klas, uwidocznione na diagramach 1 i 2 – ogólna struktura pozostała taka sama, ale zostały dodane lub usunięte niektóre pośrednie klasy oraz zostały dokonane modyfikacje, zgodnie z zaleceniami.

Konsekwencją jednej z modyfikacji jest możliwość dodawania kolejnych rodzajów komórek w grze, przez dodawanie nowych klas dziedziczących po Cell. Pozwala to na możliwość uogólnienia pierwotnego projektu do szerszej klasy automatów komórkowych niż Game of Life.

Dokumentacja techniczna - diagram klas

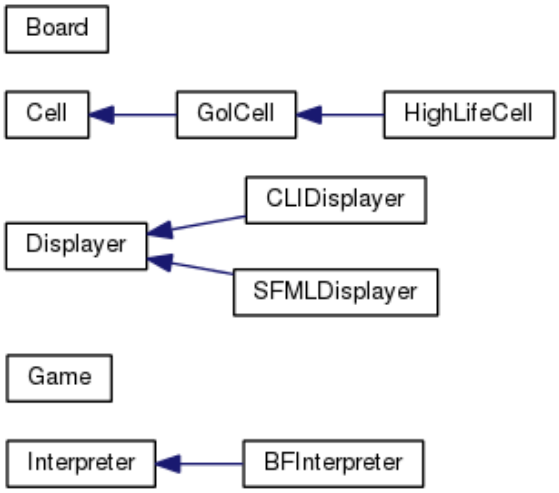

Diagram 1: ogólna struktura klas uwzględniająca dziedziczenie; Diagram 2 na stronie 3 zawiera szczegółową wersję

Klasa Game zawiera metody pozwalające na właściwą obsługę gry. Do obiektu klasy Game jest przypisana plansza (obiekt Board), na której operujemy, obiekt wyświetlający (dziedziczący po Displayer – nasza implementacja używa SFMLDisplayer) i obiekt interpretera (dziedziczący po Interpreter BFInterpreter).

Game zajmuje się aktualizacją stanu gry (metoda advance()) i wyświetlaniem (metoda display()). Wszystkie metody Game wywołują odpowiednie metody tych skojarzonych klas. Klasa Board reprezentuje planszę złożoną z komórek (tablica obiektów Cell), za pomocą metody update() aktualizuje swój stan, wywołując właściwą dla każdej komórki metodę. Plansza może zmieniać rozmiar metodą resize().

SFMLDisplayer wykorzystuje bibliotekę SFML do wyświetlania stanu planszy w formie graficznej (metoda display()). BFInterpreter zajmuje się interpretowaniem prostego języka, opartego na języku Brainfuck (metoda run() z kodem jako argument), który służy do modyfikowania stanu planszy. Opis języka znajduje się w dołączonym pliku readme.

Komórki na planszy reprezentowane są przez obiekty klas dziedziczących po Cell – GolCell (zwykła komórka, obsługująca standardowe zasady Gry w Życie) i HighLifeCell (komórka działająca na lekko zmodyfikowanych zasadach). Komórki zajmują się aktualizowaniem swojego stanu (metodą update()) na żądanie planszy. Zarządzaniem wszystkimi klasami, przyjmowaniem wejścia od użytkownika, wczytywaniem skryptów i interpretowaniem wyżejpoziomowej warstwy języka zajmuje się funkcja main() programu.

Dokumentacja techniczna - wybrane scenariusze działania

- Po uruchomienu tworzone są obiekty klas Board, BFInterpreter i Game (który wiążemy z poprzednimi dwoma i nowym obiektem klasy SFMLDisplayer). Tymi obiektami manipuluje funkcja main(), zawierająca pętlę główną gry, gdzie przyjmowane i interpretowane jest wejście od użytkownika. Wyświetlanie i aktualizacja stanu/manipulacja planszą odbywa się przez wywoływanie metod obiektu Game. Interpretacją kodu I modyfikowaniem stanu planszy zajmuje obiekt BFInterpreter, przy użyciu metody run(). Opis języka używanego do kontrolowania programu znajduje się w pliku readme.

Dokumentacja techniczna - inne elementy nie opisane powyżej

Do wyświetlania grafiki została użyta biblioteka SFML (http://www.sfml-dev.org/) oraz do obsługi interfejsu tekstowego w konsoli dodatkowo popularna biblioteka ncurses (http://www.gnu.org/software/ncurses/).

Skrypty języka używanego do obsługi gry są wczytywane z plików tekstowych. Można również podać kod skryptu jako argument(y) do programu oraz wpisywać skrypty podczas działania gry.

Do kodu źródłowego dołączamy: dokumentację (prosty wykaz klas, funkcji i zależności między nimi również przedstawionymi na diagramach) wygenerowaną za pomocą programu Doxygen + GraphViz; przykładowe skrypty; plik readme z dodatkowymi szczegółami i opisami (m.in. opis komend naszego języka).

Programowanie obiektowe	Implementacja

Dokumentacja użytkownika

Grę można uruchomić, jako argument w linii komend podając kod źródłowy skryptu, który zainicjuje stan gry (jest to opcjonalne). Po uruchomieniu, gra wyświetla okno graficzne ze stanem planszy i oczekuje na wejście od użytkownika . Jest ono traktowane jako treść skryptu, którego komendy są interpretowane jedna po drugiej, po zatwierdzeniu klawiszem enter (gra działa jednak bez przerwy – żeby to zapewnić użyto biblioteki ncurses, m.in. do wyłączenia buforowania standardowego wejścia). Komenda 'import' powoduje wczytywanie skryptów z pliku. Symulację na określoną ilość kroków czasowych uruchamia komenda 'advance'. Przez cały czas działania użytkownik może dowolnie przeplatać uruchamianie skryptów z plików z uruchamianiem bezpośrednio oraz zatrzymywać i uruchamiać symulację. Gra jest zakończona kiedy użytkownik wciśnie klawisz escape lub wpisze komendę 'quit'.

Dodatkowe informacje w dołączonym pliku readme.

Miejsce na uwagi prowadzącego

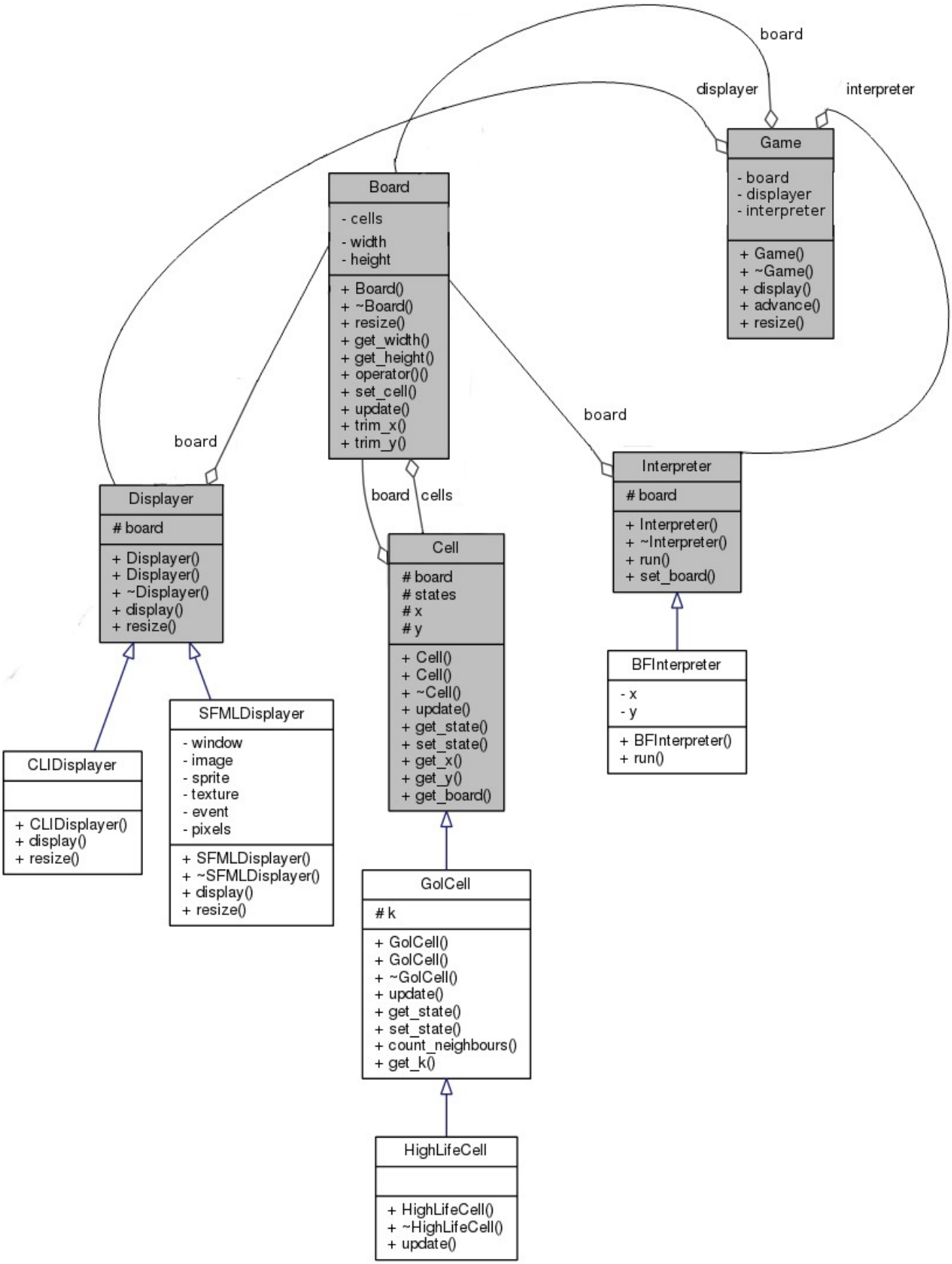

Diagram 2: powiązania między klasami (na podstawie diagramów Doxygen + GraphViz); bardziej szczegółowe informacje (jak typy danych i argumenty metod) w załączonej do kodu dokumentacji wygenerowanej za pomocą narzędzia Doxygen (dostępna w dokumentacja/html/index.html)