Game Of Life
Aplikacje Mobilne - Projekt Zespołowy

Artur Bednarczyk, Dawid Grajewski, Tomasz Januszek
Politechnika Śląska
Wydział Matematyki Stosowanej
Informatyka, semestr VI

7 kwietnia 2019

Spis treści

1 O projekcie

1.1 Zespół

Osoba	Główna odpowiedzialność
Artur Bednarczyk	Organizacja, dokumentacja, film, projekty graficzne, code review
Dawid Grajewski	Struktura aplikacji, implementacja
Tomasz Januszek	Algorytm gry w życie, „nie nagrywaj mnie“

1.2 Temat

Gra w życie Conwaya Wizualizacja ciągła i krokowa (zmienna szybkość), możliwość odczytu i zapisu planszy, różne rozmiary planszy, dostosowywanie planszy do różnych ekranów urządzeń mobilnych.

Bonus: konfigurowalne reguły gry z uwzględnieniem wersji kolorystycznych.

2 Projekt

2.1 UI/UX

2.1.1 Zawartość

UI programu będzie złożone z kilku elementów:

- Splash Screen - ekran powitalny
- Menu - pozwoli na przejście do konkretnych opcji aplikacji.
- Ustawienia - w tym miejscu użytkownik może zmienić reguły gry oraz kolorystykę.
- O Projekcie - Informacje o projekcie oraz krótka instrukcja.
- Wczytywanie - Lista zapisanych stanów gry.
- Gra - Widok planszy oraz ustawień prędkości. Tutaj również użytkownik może zapisać stan gry.

2.1.2 Projekty UI

3 Algorytmy

3.1 Aktualizacja gry

Z każdym kolejnym krokiem krokiem aktualizowane są wszystkie komórki.

```
for (x in 0..(gameBoardSize - 1)) {
```

```
    for (y in 0..(gameBoardSize - 1)) {
        updateCellLife(x, y)
    }
}
```

Przy aktualizacji sprawdzany jest stan komórki, w zależności od którego sprawdzane są odpowiednie zasady, które są porównywane z stanem sąsiadów komórki.

```
if (gameBoard[x][y] == 1) {
    if (gameRules.neighboursToDie.contains(
       cellNeighbours)) {
        conwayTransitionGameBoard[x][y] = 0
    }
} else {
    if (gameRules.neighboursToBorn.contains(
       cellNeighbours)) {
        conwayTransitionGameBoard[x][y] = 1
    }
}
```

Gdzie stan sąsiadów, to liczba żywych komórek w otoczeniu sprawdzanej:

```
if (x > 0 && gameBoard[x - 1][y] == 1) {
    liveNeighbouringCellsCounter += 1
}
if (x < (gameBoardSize - 1) && gameBoard[x + 1][y] ==
   1) {
    liveNeighbouringCellsCounter += 1
}
if (y > 0 && gameBoard[x][y - 1] == 1) {
    liveNeighbouringCellsCounter += 1
}
if (y < (gameBoardSize - 1) && gameBoard[x][y + 1] ==
   1) {
    liveNeighbouringCellsCounter += 1
}
if (x > 0 && y > 0 && gameBoard[x - 1][y - 1] == 1) {
    liveNeighbouringCellsCounter += 1
}
if (x > 0 && y < (gameBoardSize - 1) && gameBoard[x -
   1][y + 1] == 1) {
    liveNeighbouringCellsCounter += 1
```

```
}
if (x < (gameBoardSize - 1) && y > 0 && gameBoard[x +
    1][y - 1] == 1) {
     liveNeighbouringCellsCounter += 1
}
if (x < (gameBoardSize - 1) && y < (gameBoardSize - 1)
   && gameBoard[x + 1][y + 1] == 1) {
     liveNeighbouringCellsCounter += 1
}
```

4 Narzędzia

4.1 Kontrola wersji

Do zarządzania kodem i wersjami projektu wykorzystujemy narzędzie Git. Korzystamy z platformy GitHub jako repozytorium dostępnego online. Dobór narzędzi służących do korzystania z repozytorium to sprawa indywidualna każdego członka zespołu, ponieważ nie ma ona wpływu na sam projekt.

4.2 Zarzadzanie zespołem

Trello - Kanban Board - to tutaj rozpisujemy zadania i przydzielamy je sobie, określamy również terminy i planujemy.

Przy współpracy, aby dane zmiany zostały wprowadzone muszą zostać akceptowane przez innego członka zespołu.

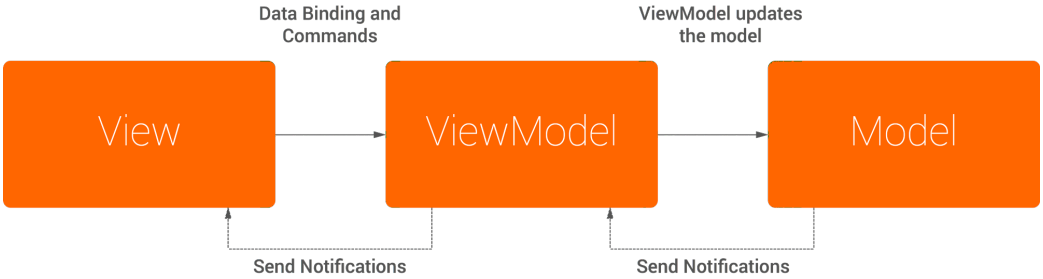
4.3 Środowisko

Android Studio - Kotlin

5 Aplikacja

5.1 Architektura

Aplikacja została utworzona zgodnie z architekturą MVVM (Model-View-ViewModel)



Dzięki „Data binding“ dane przechowywane przez model są automatycznie odświeżanie w widoku. View, jako warstwa widoku zawiera interfejs użytkownika oraz odpowiada za interakcję z użytkownikiem. Viewmodel jest warstwą modelu widoku, która udostępnia dane i odpowiada za wymianę informacji z modelem. Przechowuje również referencje do modelu. Model zawiera logikę aplikacji.

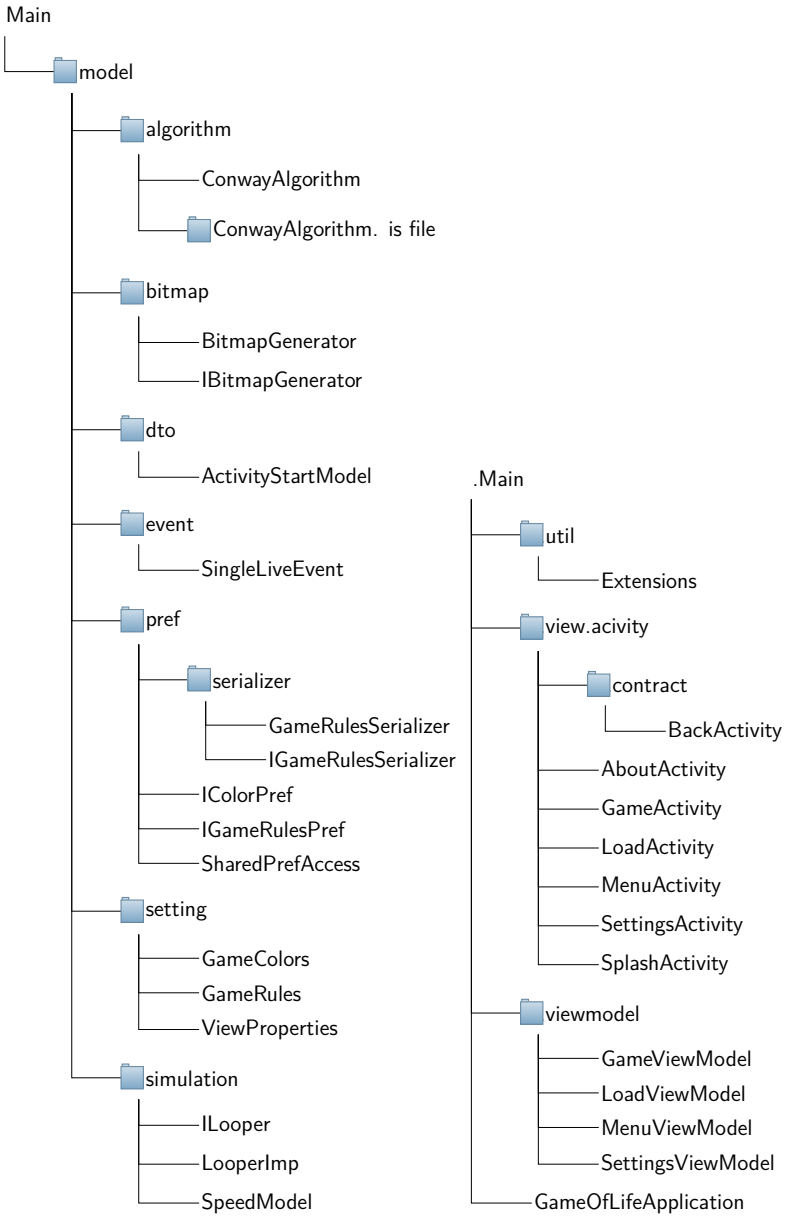
5.2 Struktury danych

Reguły gry w życie przechowujemy jako ciąg liczb określających liczbę sąsiadów, oddzielany średnikiem. Przykład:
”0;1;2;3;”
Kolory są w formacie ARGB.
Zapis stanu do pliku w formacie JSON, przykład:

```
{
        x: 3,
        y: 3,
        birth: "1;2;3;",
        death: "4;5;",
        aliveColor: -1,
        deadColor: -16777216,
        gameState:      [1,0,0,
                         1,0,1,
                         1,0,1,]
}
```

5.3 Schemat graficzny struktury systemu

5.4 Podział na pliki



5.5 Biblioteki

W projekcie zostały wykorzystane następujące biblioteki:

- Lifecycle library - zarządzanie aktywnościami i cyklem życia
- Koin - wstrzykiwanie zależności
- Dexter - zarządzanie uprawnieniami

- QuadFlask:colorpicker - wybieranie kolorów
- Timber - logowanie

5.6 Testowanie

Aplikacja jest testowana testami jednostkowymi oraz manualnie. Jednostkowo została przetestowana klasa „GameRulesSerializer“