**Computer game ‘Cultures evolution’**

by

**Sergei Ignatov**

**CAPSTONE PROJECT REPORT**

Submitted in partial fulfillment of the requirements

for the degree of

Master of Science in Software Engineering

International Technological University

San Jose, California

August 3, 2018

# Abstract

The idea of this project is a computer game similar to Europa Universalis, Crusader Kings and some others. The main difference is an approach to two aspects:

1. Emulation of the historical processes of the rise and fall of countries, people migration and the change of cultures
2. Customization of many characteristic including scripts for AI

This project contains the implementation of some ideas described above and others: the emergence of countries, migration of people, epidemics, climate change, cultural change, science development, customized scripts for AI. I made emphasis on AI players in the first version (for capstone project) and I am going to develop many functions for human player in the next versions. There are a lot of other ideas which will be very useful for the game. I will implement them later in the next versions.

Used technologies: Java 8, JavaFX, Spring Boot.

The game was tested on Windows 10. It should work on Mac OS X too (because all used technologies allow it), but I did not test it.

Working with this project allowed me to get experience with JavaFX and Spring Boot (I did not use them before). I actively used knowledges from many courses, e.g. Software Requirements Elicitation, Software Architecture, User Interface Design & Implementation, Data Structures.

**Table of Contents**

[**Abstract**](#_1pfjnrn7ojs9)2

[**List of Figures**](#_ls3ppowolchs)5

[**Introduction**](#_1w1l9axrlu4j)7

[**Primary ideas**](#_uqnk3lq1c3tu)8

[Emulation of the historical process](#_dtqmy7154wa0) 8

[The concept of "population culture".](#_w7ods24akmo7) 8

[Scientific level of provinces and population.](#_tu1olnowxhv6) 9

[The concept of “Country Focus”.](#_kj1a66tc2rsq) 9

[Customization](#_8kfqd6xxd879) 10

[**Development timeline**](#_53mv15hl5z6h)11

[**Project risks**](#_tvh46cbfr61a)13

[**Conceptual Model**](#_dzst85crb3wg)14

[**Use cases**](#_xv7524y09pa4)16

[**Functional requirements**](#_owwes8fm8bns)19

[**Non-functional requirements**](#_he3q0d9lwgv7)20

[**Technologies**](#_dzf6p3rz38o9)21

[**Architecture**](#_772qf9ha1prz)22

[**Diagrams**](#_b82lmbfmll8l)24

[**User Interface Design**](#_djsfc0bqbcu8)25

[**Source code**](#_gghvkcbvum19)27

[**Build scripts**](#_seo66wf6q2c2)28

[**Installation and running**](#_ogpqjqp4tdny)30

[**Implemented features / User Manual**](#_ligk4pvrz2cl)31

[Create new game](#_tqtsuqdcky0i) 31

[Save game](#_lus3sew2mdtp) 32

[Load game](#_8cqdw04u7xl7) 33

[Quick Save / Quick Load](#_bizjvl2kqgy2) 34

[Time management](#_ylaoluauk0h5) 34

[The map modes](#_tvr8bgoet43r) 34

[View several maps at the same time](#_xrih0pvm6o34) 38

[View global info](#_jpf3egwu0ici) 39

[View country info](#_yudmwjyr0ap) 40

[View province info](#_23chicpdm4m9) 40

[View province science level](#_bf49bxk22ugc) 40

[View army info](#_qu1sii8cm0x5) 41

[View event info](#_zfyhgdwou0wc) 41

[Population growth](#_k9ixo67ql7xe) 41

[Migration](#_bfptp2472zxb) 41

[Epidemic](#_hofq4zw5rxtl) 42

[Climate change](#_oujkfve3h9vu) 42

[Culture change](#_om4ycbqz7j2o) 42

[Science development](#_664m8hxa8e4) 43

[AI scripts](#_2ezmbrf64nuy) 43

[**Check of requirements**](#_gzbvvmfd8pcj)44

[**Conclusions**](#_h2jydnuc8ho8)45

[**Appendix. Source code**](#_4l8jvpu8a5nn)46

# List of Figures

|  |  |
| --- | --- |
| Figure 1. The application structure. | 22 |
| Figure 2. Layers of the application. | 22 |
| Figure 3. Major components of the application | 23 |
| Figure 4. Sequence diagram of new turn handling. | 24 |
| Figure 5. The file menu | 31 |
| Figure 6. The window for game creating | 31 |
| Figure 7. The main window after game creating | 32 |
| Figure 8. Save game dialog | 33 |
| Figure 9. The window with an error message | 34 |
| Figure 10. Time management pane | 34 |
| Figure 11. The map in geographical mode | 35 |
| Figure 12. The map in political mode | 36 |
| Figure 13. The map in population mode | 36 |
| Figure 14. The map in culture mode | 37 |
| Figure 15. The map in disease mode | 38 |
| Figure 16. The map in science mode | 38 |
| Figure 17. Two maps in different modes | 39 |
| Figure 18. Global info pane | 40 |
| Figure 19. Country info pane | 40 |
| Figure 20. Province info pane | 40 |
| Figure 21. Science info pane | 41 |
| Figure 22. Armies info pane | 41 |
| Figure 23. Events info pane | 41 |

# Introduction

The game is an interactive map divided into the provinces. Each of these provinces contribute to their country either positively or negatively. The gameplay requires the player to lead a nation by finding a balance of military, diplomacy and economy. The player does so through their choices as sovereign of their nation, and through the spending of resources available to them.

The game is a sandbox environment, and there is no strict rule on winning the game. A loss occurs when the player's nation is removed, or annexed, from the map.

Period of time: approximately from the Bronze Age to the beginning of the Middle Ages.

Motivation of the game:

1. Existing games poorly implement the real historical process of the rise and fall of countries, the change of cultures, the migration of peoples.
2. In my opinion, existing games do not provide enough opportunities for customization.

# Primary ideas

## Emulation of the historical process

Ideas:

* The concept of "population culture"
* Scientific level of provinces and population
* The concept of “focus of the country”

### The concept of "population culture".

The population has a certain culture, characterized by "culture DNA". Each province has its own DNA. Possible characteristics:

* Propensity to science. It affects the speed of scientific discoveries.
* Propensity to different kinds of sciences.
* Aggressiveness, propensity to war. Aggressiveness increases the likelihood of declaring war, raises the effectiveness of the army (morale). Reduces the effectiveness of trade with other states.
* Openness to change (open minded people). It increases the speed of spreading scientific knowledge and the rate of the cultural change under the influence of various factors.

DNA can be changed over time:

* Influence of neighboring provinces.
* Events
* Actions of the government, management style.

Each country initially has base DNA, to which the change of culture tends (if there are no other influences, DNA will tend to base). The base DNA can be changed very slowly towards the average culture in the country.

Culture can be changed in the capital province. It depends on the actions of the government.

Some provinces can become centers of cultural change (influential clans, people). If the culture is too different from the central province, then the provinces can stand out as a separate country. Similarly, if some remote province becomes too strong.

The game will have a competition between states and cultures, a kind of genetic algorithm for selecting the most successful cultures (of course, there are many other important factors: AI scripts, events).

### Scientific level of provinces and population.

Types of scientific developments:

* Agriculture. Affects the amount of food produced. The higher the efficiency, the more people can engage in other activities.
* Medicine. Affect resistance to diseases, and population growth.
* Administration (management). Affects effectiveness of government (taxes, culture changes).

### The concept of “Country Focus”.

To emulate stability of the country and the ability to develop, and to balance big and small countries, the game will use the concept of the focus. The focus is "spent" on:

* Administrative support of the government (the more provinces and the population, the more focus is required).
  + the complexity of managing a large state
  + evidence that as the state and its "strength" grow, people become more consumers and less interested in other things (science and others).
* The science. The effectiveness of studying science directly depends not only on money, but also on focusing on science.

The focus is a fixed number. The focus distribution can be set by player (or AI). It allows to choose development between "wide" and "tall" states. Wide country has many provinces and requires much focus for administrative support, so science will be developed slowly. Tall country has less provinces but they can be more effective (because they are more advanced in science). Maximum focus can be changed by events.

Note: it will be implemented in next versions (not in the capstone project).

## Customization

Ideas:

* Scripts/macroses for AI (strategic and tactical levels). The game shall allow users to change scripts.
* Map editor (in the future version, not capstone project)
* Module structure of application which will support flexible replacement of components (in the future version, not capstone project)

In fact, there are possible metagames. Users can create AI scripts, start game only with AI countries and watch whose scripts are more successful. The game can be platform for AI scripts competition.

# Development timeline

Table 1.

Development timeline

|  |  |
| --- | --- |
| Stages/steps | Date |
| Think and describe main ideas | 05/20/2018 |
| Requirements elicitation | 06/10/2018 |
| Describe Design | 06/20/2018 |
| Develop & Test: |  |
| Application framework, build scripts, mechanism for internationalization | 06/05/2018 |
| Base data model, world generation | 06/10/2018 |
| Map of the world (show, scroll, scale), modes: geographical, population, soil(fields) | 06/12/2018 |
| Save/load game | 06/12/2018 |
| Info panels (global, province) | 06/13/2018 |
| Time processing (start, pause, different speeds) | 06/15/2018 |
| Events (climate change, epidemics); map mode for epidemic; people migration | 06/18/2018 |
| Science (rise, fall, spread), info panel, map mode | 06/25/2018 |
| Infrastructure (rise, fall, spread), info panel, map mode | 06/28/2018 |
| Culture (spread, change, mutation, merging), info panel, map mode | 07/03/2018 |
| Countries, simple hardcoded algorithms for AI | 07/10/2018 |
| Armies (creating, movement), processing by AI | 07/13/2018 |
| Script engine for AI, replacement hardcoded parts to scripts | 07/16/2018 |
| Player countries, science and country focus management | 07/19/2018 |
| Player armies management | 07/21/2018 |
| Simple diplomatic interaction | 07/25/2018 |
| Integration tests, bug fix | 07/30/2018 |
| Performance analysis and optimization | 08/03/2018 |

# Project risks

Main risks:

1. Unfamiliar technologies. I use technologies which I have never used before: Spring Boot and JavaFX. Probably I will have time delays when some things will work in different way than I am used to.
2. Performance. One of the main goal is implementing AI logic through scripts. It can slow the game processing, so maybe I will have to implement some parts as hardcoded (java implementation).
3. Unpredictable complexity of AI logic. I do not have an experience with AI implementing. It can be more difficult than I suggest now. It can require significant time and efforts.

All risks seem to be solvable, but the development can require more time than I have for capstone project. In this case I will have to reject some features.

# Conceptual Model

There is a textual representation of conceptual model. Each entity is described with references to other entities.

* Map is a main visible and most often usable object for a player. Map can work in different modes: geographical, political, population, soil quality, cultural, science and others. Map consists:
  + Province
* Province is a minimal unit of the interactive map. It has properties: name, type (e.g. grassland, ocean), soil fertility, area of fields (soil), infrastructure. It contains:
  + Population
  + Armies
* Country is a political unit ruled by Player or AI. It has:
  + Provinces
  + Armies
  + Science level
  + Culture
  + Country focus (can be used for country management and science development)
* Population. In fact, group of people. Provinces can have some amount of population. Population has:
  + Culture
  + Science level
* Army. Military unit. Effectiveness depends on amount of soldiers, science level and culture.
* Science influences on effectiveness of many aspects including: food production, protection from diseases, army capabilities and others. It is described in details in the chapter ‘Scientific level of provinces and population’.
* Culture. The population has a certain culture, characterized by "culture DNA". It is described in details in the chapter ‘The concept of "population culture".
* Event. Global and local events that affect the provinces, the population, and so on.

# Use cases

There are a few major uses cases (it is not a full list of all use cases) for human player.

The version of the game for capstone project made emphasis on AI player, so actually it does not have many use cases with the user as actor.

**Start new game**

Primary Actor: User

Goal: Start new game

Pre-conditions: User opened application

Main success scenario:

1. User invokes function “Start new game”
2. Application opens the window with editable game parameters
3. User modifies parameters if he wants and press ‘start game’
4. Application validates parameters
5. Application creates new game world
6. Application open map view with new world

Extensions:

4a - user parameters are not compatible with each other

4a1 - Application shows error message

4a2 - Application asked user to change parameters

**Save game**

Primary Actor: User

Goal: Save game

Pre-conditions: User opened application, the game is active

Main success scenario:

1. User invokes function “Save game”
2. Application opens Save file dialog
3. User selects folder and name of file
4. Application stores file

Extensions:

4a - error happened while saving

4a1 - Application shows error message

**Load game**

Primary Actor: User

Goal: Load game

Pre-conditions: User opened application

Main success scenario:

1. User invokes function “Load game”
2. Application opens Open file dialog
3. User selects folder and name of file
4. Application loads file

Extensions:

4a - error happened while loading

4a1 - Application shows error message

**Manage science areas priorities**

Primary Actor: User

Pre-conditions: User opened application, the game is active

Main success scenario:

1. User invokes function “Manage science”
2. Application opens the window “Manage science”
3. User configures science priorities, press ‘apply’ button
4. Application closes “Manage science” window and applies new parameters to the current game

**Move armies**

Primary Actor: User

Pre-conditions: User opened application, the game is active

Main success scenario:

1. User selects army on the map
2. User selects direction where army should be moved
3. Application remembers user decision and apply it when it is handing a new turn

# Functional requirements

FR-1. The game shall have a map with provinces. The map shall support zooming and scrolling.

FR-2. The game shall have information/management panels or windows for general/global data, province, science, diplomatic, military.

FR-3. The game shall support the movement of time at different speeds (1x - 10x). Base step = 1 year

FR-4. The game shall support functions: save game, load game, create new game, quick save and quick load.

FR-5. Provinces shall have properties: name, terrain type (e.g. grassland, ocean), soil fertility, fields area, infrastructure, population.

FR-6. The population shall have a "cultural DNA"

FR-7. Cultural DNA can be changed by others cultures (neighbours provinces).

FR-8. Countries shall have base DNA to which the change of culture tends (if there are no other influences, DNA will tend to base).

FR-9. The game shall support science development.

FR-10. Science shall be for several areas: Agriculture, Medicine, Administration. It is described in details in the chapter ‘Scientific level of provinces and population’.

FR-11. The game shall implement people migration (because of hunger or war).

FR-12. The game shall process random events (i.e, climate change, epidemic).

FR-13. The game shall support army building, moving and fighting.

FR-14. AI logic shall be implemented by script language.

FR-15. The game shall allow the user to modify AI scripts.

# Non-functional requirements

NFR-1. Operating system: Windows 10.

NFR-2. The implementation language and technologies should allow to port the game on another operating systems (Mac OS) in the future.

NFR-3. Save files shall have JSON format. It is human readable format that will ease development and testing, and also allows user to modify the game data.

NFR-4. One step of the time (one year) shall be processed not more than 1 sec in average on the map with 1000 provinces (Intel Core i7 7700HQ 2.8GHz, 4GB RAM).

NFR-5. The game shall support maps up to 2500 provinces.

NFR-6. The game shall support internationalization (many languages).

# Technologies

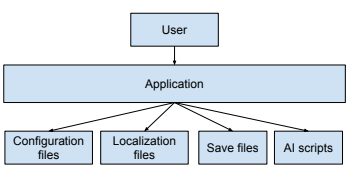
* Java 8
* GUI: JavaFX
* Framework: Spring, Spring Boot

These technologies allow implement reliable portable application (at least for Windows and Mac OS).

The developer (me) has an experience with Java, but did not work with JavaFX and Spring Boot in the past. JavaFX and Spring Boot are reliable modern technologies with a good reputation.

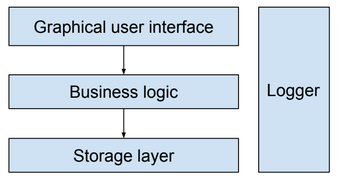
Possible risks are described in the chapter ‘Project risks’.

# Architecture



*Figure 1*. The application structure.

With the respect to requirements the game can be implemented as a monolithic application with a classical structure of three levels.

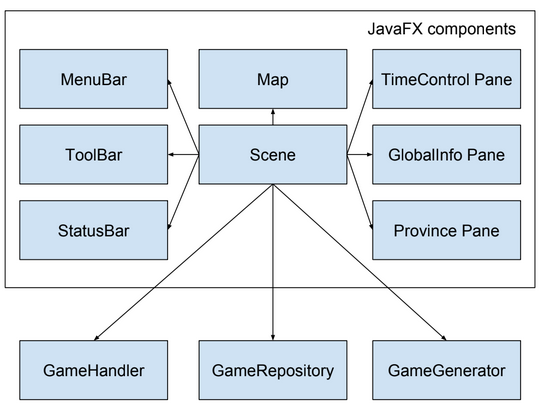


*Figure 2*. Layers of the application.

Layers:

* Storage. Components for saving and loading data. Implementation as Spring Components.
* Business logic. Components for processing game logic. Implementation as Spring Components.
* GUI. Components for interaction the user with the application. Implementation as JavaFX components.

Logger is a cross-layers component for logging info/error/etc messages.

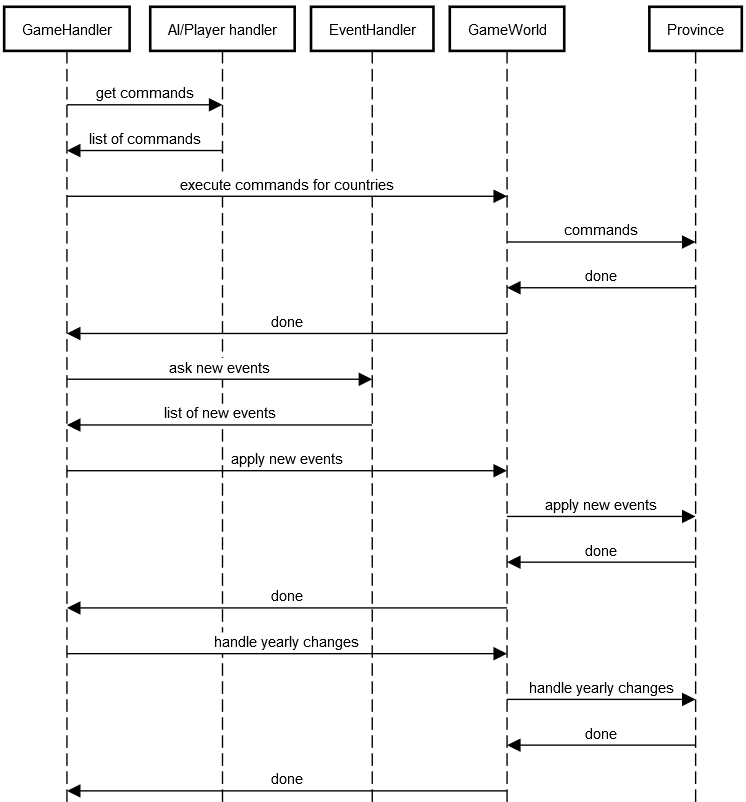


*Figure 3*. Major components of the application

* GameHandler. Handles the flow of time, applies all the game rules to the model of the world.
* GameRepository. Save/load games.
* GameGenerator. Create new game based on configured parameters.
* Scene. Central JavaFX component for GUI. It contains and manages all embedded elements: Map, Menu, Info panels, etc
* Map. UI component for working with a world map.
* TimeControl Pane. Pane with elements for managing time flow (start, stop, etc).
* GlobalInfo Pane. Information about the world.
* Province Pane. Information about the selected province.
* MenuBar, ToolBar, StatusBar - standard UI elements.

All components are implemented as Spring Components.

# Diagrams



*Figure 4*. Sequence diagram of new turn handling.

# User Interface Design

Examples of UI pages are described in the chapter ‘Implemented features’.

**Patterns for page layout**

* Visual Framework. Each window uses the same basic layout, colors, and stylistic elements.
* Center Stage. The main UI element is a map, it is the largest part of main window.
* Accordion. All info panels are placed on the right side and they can be opened and closed independently of each other. If the panel is closed, then it will show tooltip with the information when the mouse cursor is pointing to the panel title
* Liquid Layout. As the user resizes the window, the main window content resizes along with it so that the page is constantly filled.
* Right/Left Alignment. All info panels have right/left alignment for their elements.
* Diagonal Balance. Some windows use Diagonal Balance pattern, e.g. Create New Game dialog.

**Patterns for navigation**

* Clear Entry Points. The main entry points is a menu File with functions create new game and open game.
* Modal Panel. Several windows use this pattern: save/load dialogs, create new game dialog.

**Primary navigation model**

Hub and spoke. Hub is a main window with a map. Spokes - additional windows.

**Pattern for information architecture**

Alternative Views. The user is able to see a map in different modes. There is a special button panel below the map to choose a mode. Also the user is able to open many windows with the map in different modes in the same time. To do this he/she should click right mouse button on mode button.

# Source code

Public repository with source code: <https://github.com/ignatovsni/cw-main>

Also source code is included to Appendices.

**Main class**: com.cwsni.world.client.desktop.MainWindow

Packages:

**com.cwsni.world.model** - classes for data model

**com.cwsni.world.model.data** - base data model (for saving and loading files)

**com.cwsni.world.model.player** - data model which is used by player handler (AI and human). These classes isolate internal model from inappropriate using by client side.

**com.cwsni.world.services** - services for creating game, saving/loading, handling new turns

**com.cwsni.world.services.algorithms** - services with algorithms like searching of shortest way in graph

**com.cwsni.world.game.ai** - AI implementation including script engine

**com.cwsni.world.game.commands** - data model of commands. AI/Human player handler do not manipulate with data model of the world directly, it creates commands which are executed by ‘server side’.

**com.cwsni.world.client.desktop** - client side of the game (User Interface)

**com.cwsni.world.client.desktop.game** - classes for UI (windows, dialogs, etc)

**com.cwsni.world.client.desktop.game.map** - map implementation

# Build scripts

Apache Maven is used for building of the application.

Command “mvn package” creates target artifact: \target\cw-main-0.0.1-SNAPSHOT.jar - all executable files and configuration are packed to this .jar file.

Configuration (pom.xml):

<**project xmlns="http://maven.apache.org/POM/4.0.0"**

**xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**

**xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>

<**modelVersion**>4.0.0</**modelVersion**>

<**groupId**>com.cwsni.world</**groupId**>

<**artifactId**>cw-main</**artifactId**>

<**version**>0.0.1-SNAPSHOT</**version**>

<**packaging**>jar</**packaging**>

<**name**>cw-main</**name**>

<**parent**>

<**groupId**>org.springframework.boot</**groupId**>

<**artifactId**>spring-boot-starter-parent</**artifactId**>

<**version**>2.0.2.RELEASE</**version**>

</**parent**>

<**properties**>

<**java.version**>1.8</**java.version**>

<**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>

</**properties**>

<**dependencies**>

<**dependency**>

<**groupId**>org.springframework.boot</**groupId**>

<**artifactId**>spring-boot-starter-test</**artifactId**>

</**dependency**>

<**dependency**>

<**groupId**>com.fasterxml.jackson.core</**groupId**>

<**artifactId**>jackson-databind</**artifactId**>

</**dependency**>

<**dependency**>

<**groupId**>org.codehaus.groovy</**groupId**>

<**artifactId**>groovy</**artifactId**>

<**version**>2.5.1</**version**>

</**dependency**>

</**dependencies**>

<**build**>

<**plugins**>

<**plugin**>

<**groupId**>org.springframework.boot</**groupId**>

<**artifactId**>spring-boot-maven-plugin</**artifactId**>

</**plugin**>

</**plugins**>

</**build**>

</**project**>

# Installation and running

To install and run:

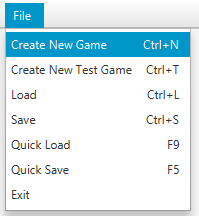
1. The user should have installed Java 8.
2. Copy cw-main-0.0.1-SNAPSHOT.jar to any directory
3. “java -jar cw-main-0.0.1-SNAPSHOT.jar” - for starting with EN locale

“java -jar cw-main-0.0.1-SNAPSHOT.jar --app.locale=RU” - for starting with RU locale

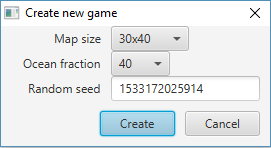
# Implemented features / User Manual

## Create new game

The user should click the menu item ‘Create New Game’ (or press “Ctrl+N”)

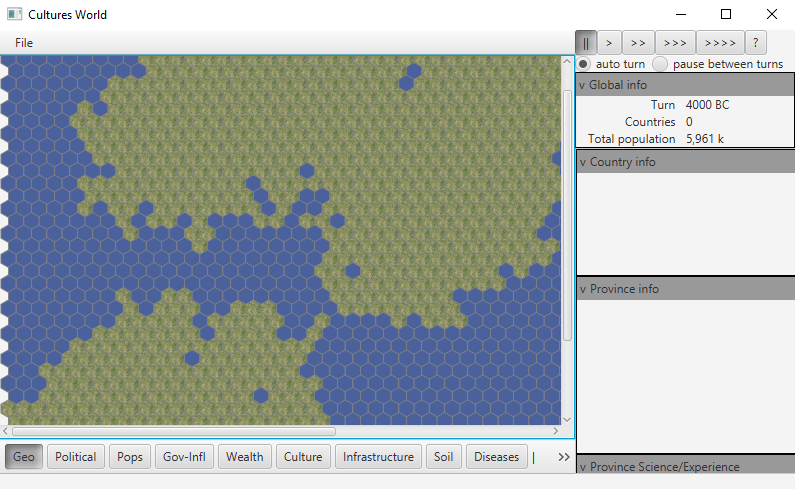


*Figure 5*. The file menu



*Figure 6*. The window for game creating

The application creates new game world and shows it:

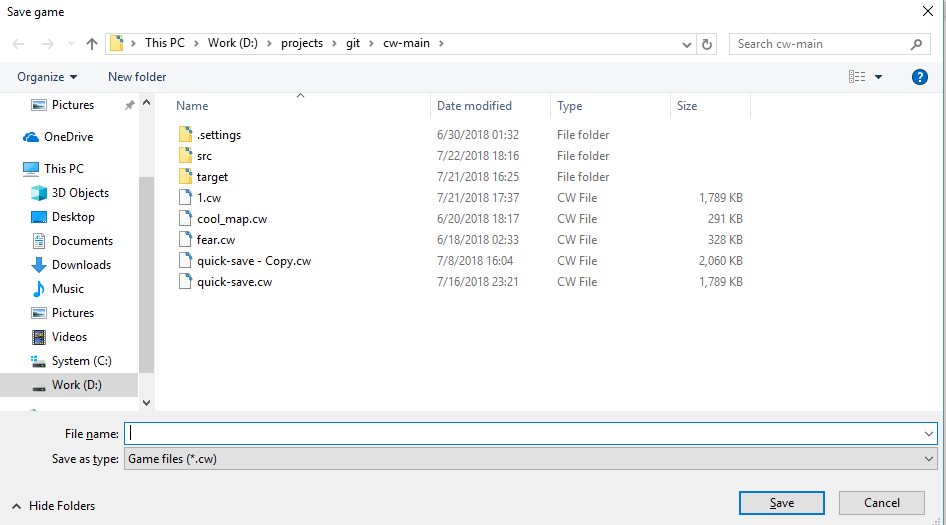


*Figure 7*. The main window after game creating

## Save game

The user should click the menu item ‘Save’ (or press “Ctrl+S”)

The application shows standard dialog for saving files like:



*Figure 8*. Save game dialog

The user can input the file name and the application saves the game state to the file after the user clicks the button ‘Save’

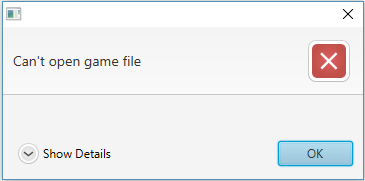
## Load game

The user should click the menu item ‘Load’ (or press “Ctrl+L”)

The application shows standard dialog for loading files (similar to saving)

The user can choose the file and the application loads the game state from the file after the user click the button ‘Open’

If some error happens, the application will show a window with error message:

  
*Figure 9*. The window with an error message

If the user clicks “Show Details”, then the application will show detailed information about the error.

## Quick Save / Quick Load

The user should click the menu item ‘Quick Load’ (or press “F9”) to load the game or the menu item ‘Quick Save’ (or press “F5”) to save the game.

The application will load or save game with default name.

## Time management

The user can manage time speed with the help of time management panel.



*Figure 10*. Time management pane

There are buttons:

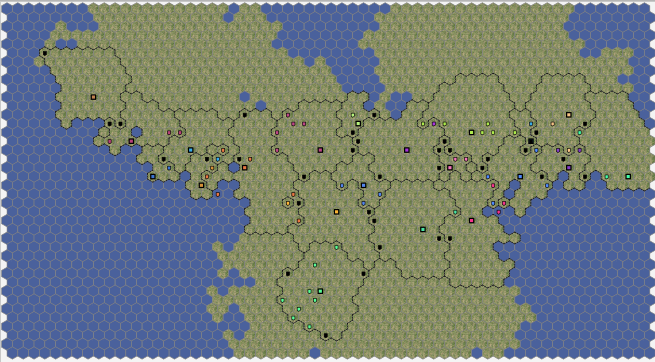
1. Pause/Stop
2. Run with regular speed x1
3. Run with speed x2
4. Run with speed x5
5. Run with speed x10
6. “auto turn” - if it is enabled, then the application will process step by step with choosed speed. If it is disabled, then the application will stop the game after each step.
7. “pause between turns” - if it is enabled, the application will pause the game on small period of time after each step

## The map modes

The user can choose which map mode he wants to see:

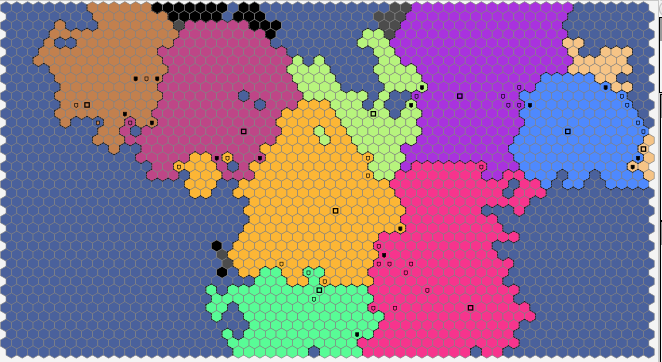


**Geo** - geographical map



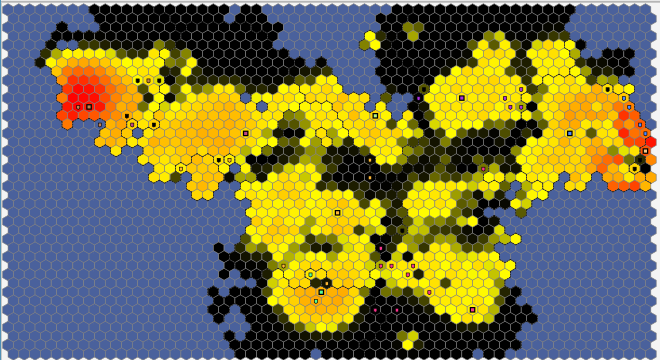
*Figure 11*. The map in geographical mode

**Political** - countries



*Figure 12*. The map in political mode

**Pops** - population density

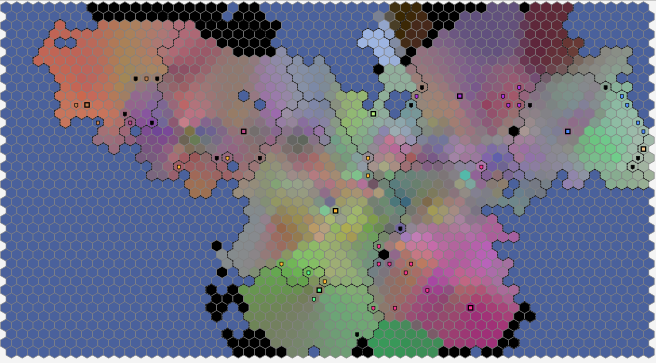


*Figure 13*. The map in population mode

**Gov-Infl** - the government influence - how effective government actions in the provinces. The same style as population density

**Wealth** - wealth of provinces. The same style as population density

**Culture** - Different cultures have different colors. More similar colors mean more close cultures.

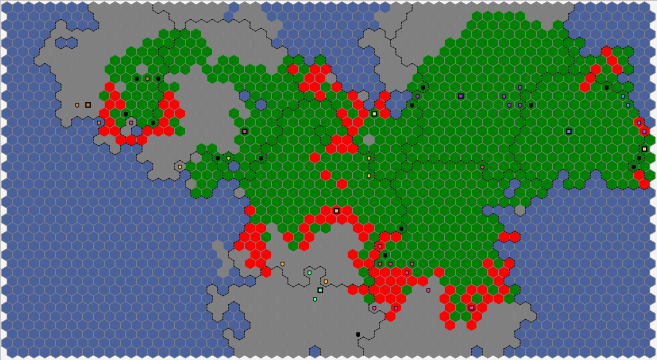


*Figure 14*. The map in culture mode

**Infrastructure** - the level of infrastructure (affects how effective people work in the province). The same style as population density

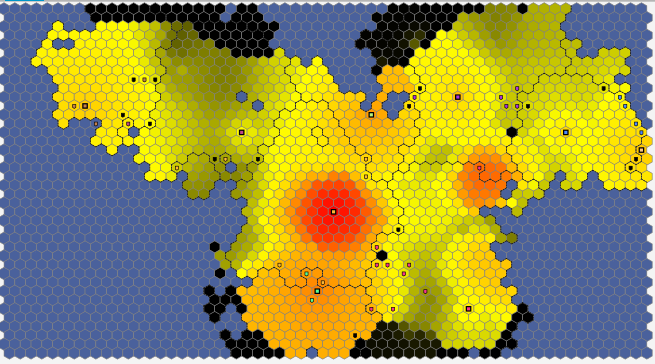
**Soil** - soil fertility (affects how much people can live in the province). The same style as population density

**Diseases** - red color for provinces with epidemic, green color means the province had the epidemic recently, and now it has temporary immunity



*Figure 15*. The map in disease mode

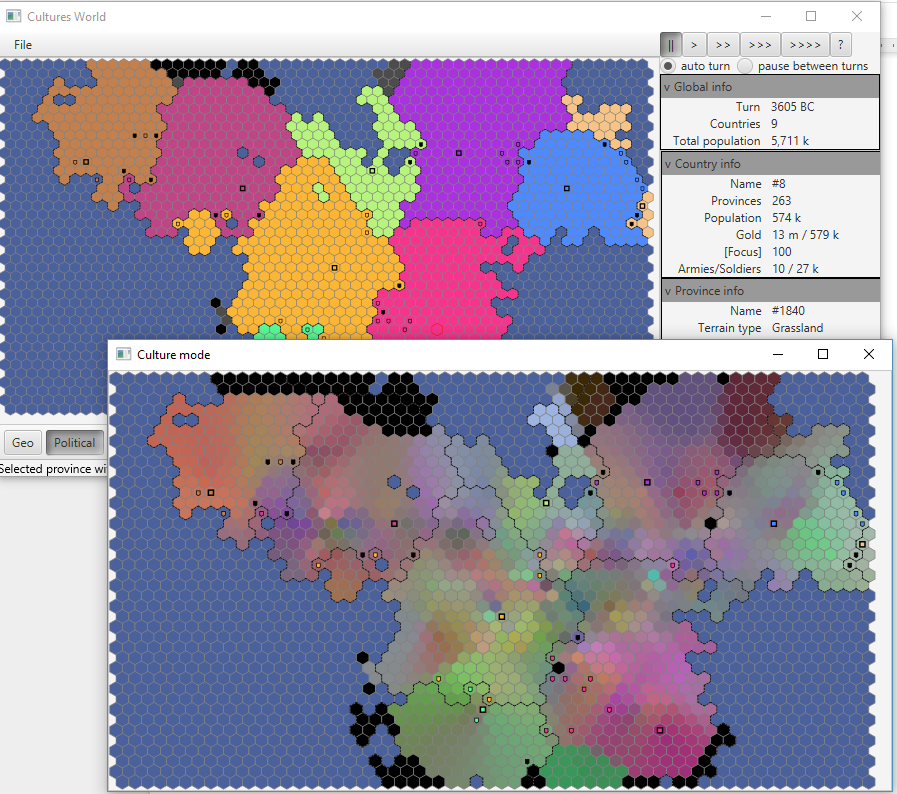
**Science** - the user is able to look at map with science level for each kind of science



*Figure 16*. The map in science mode

## View several maps at the same time

The user is able to see several maps with different modes at the same time. The user should click right button mouse on the buttons with map modes.

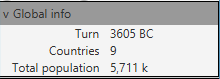


*Figure 17*. Two maps in different modes

There are two windows: 1) common windows with political mode; 2) window with culture mode.

## View global info

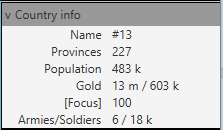
Global info panel is always available in the top right corner.



*Figure 18*. Global info pane

## View country info

The user should click on the province to look at the country info. The application will show the country info panel on the right side.

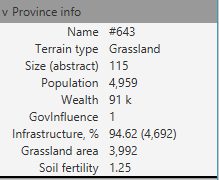


*Figure 19*. Country info pane

If the user clicked on the province, that does not belong to any country, then this panel will be empty.

## View province info

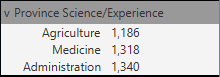
The user should click on the province to look at the province info. The application will show the province info panel on the right side.



*Figure 20*. Province info pane

## View province science level

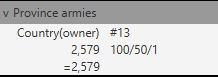
The user should click on the province to look at the province science info. The application will show this info panel on the right side.



*Figure 21*. Science info pane

## View army info

The user should click on the province to look at the armies info. The application will show this info panel on the right side.



*Figure 22*. Armies info pane

There are: the army owner, number of soldiers, organization/training/equipment level

## View event info

The user should click on the province to look at the events info. The application will show this info panel on the right side.



*Figure 23*. Events info pane

Example of the panel with the information about climate change.

## Population growth

The population is growing each turn. The base growth is 1% per year, and it is slowly increasing with medicine level. The maximum of population depends on province size and soil fertility.

The user can check this feature:

* select province
* check the population
* let the game process one or more turns
* compare the current population with the population in previous turn

## Migration

Part of population migrate each turn to neighbours provinces. If the province is overpopulated, then the more people will migrate.

The user can check this feature:

* Select empty province nearby populated province
* let the game process one or more turns
* compare the current population with the population in previous turn

## Epidemic

Epidemic can appear in any province with a small chance. It lasts a few years and kills people each turn. Medicine decreases rate of death.

The user can check this feature:

* select province with epidemic
* check the population
* let the game process one or more turns
* compare the current population with the population in previous turn

## Climate change

Climate can change with small chance every turn. It can decrease or increase soil fertility, and in this way it influences maximum of possible population.

The user can check this feature:

* Select any province when climate is changing
* check soil fertility
* let the game process one or more turns
* compare current soil fertility with the soil fertility in previous turn

## Culture change

Population culture can be changed by:

* migration of people from provinces with different culture
* government influence - each province is slowly changing the culture towards country culture

The user can check this feature:

* turn on culture map mode
* let the game process one or more turns
* observe how the culture is changing

## Science development

Science level is growing each turn:

* very slowly in each province
* in capitals of countries speed depends on finances

The user can check this feature:

* select province
* check the science level
* let the game process one or more turns
* compare the current science level with the science level in previous turn

## AI scripts

AI is implemented as groovy scripts. Base scripts are packed into .jar game file (cw-main-x.jar\BOOT-INF\classes\ai-scripts\). The user is able to modify it and to create new scripts. He/she should:

1. Create the folder 'ai-scripts' in the directory with .jar game file
2. Create new groovy script or modify default script (default.groovy).
3. Restart the game.

# Check of requirements

**Functional requirements**

Their implementation is described in the chapter ‘Implemented features’.

**Non-functional requirements**

* NFR-1. Operating system: Windows 10.

The game was developed and tested on Windows 10.

* NFR-2. The implementation language and technologies should allow to port the game on another operating systems (Mac OS) in the future.

All used technologies (Java 8, JavaFX) support Mac OS.

* NFR-3. Save files shall have JSON format. It is human readable format that will ease development and testing, and also allows user to modify the game data.

Save files have JSON format. It is implemented with the library ‘jackson’.

* NFR-4. One step of the time (one year) shall be processed not more than 1 sec in average on the map with 1000 provinces (Intel Core i7 7700HQ 2.8GHz, 4GB RAM).

The average time on the map with 1200 provinces is 0.05 seconds.

* NFR-5. The game shall support maps up to 2,500 provinces.

The largest allowed map is 2,600 provinces. The internal game engine (‘server side’) is able to process 10,000 provinces and more, but current ‘client side’ (JavaFX) has visual errors if the map has more than 4,000 provinces.

* NFR-6. The game shall support internationalization (many languages).

The game supports english and russian languages at this moment. It is able to work with many languages. To add new language the developer (or even regular user) should:

1. create new file ‘messages\_<...>.properties’ with text and place it to directory ‘.jar\BOOT-INF\classes\messages\’
2. Start the game with argument --app.locale=<...>

For example: java -jar cw-main-1.0.0.jar --app.locale=RU

# Conclusions

This project contains the implementation of some ideas for the game: countries evolution, migration of people, epidemics, climate change, cultural change, science development, customized scripts for AI. I made emphasis on AI players in the first version (for capstone project) and I am going to develop many functions for human player in the next versions.

Working with this project allowed me to get experience with Java 8, JavaFX and Spring Boot. I actively used knowledges from many courses, e.g. Software Requirements Elicitation, Software Architecture, User Interface Design & Implementation, Data Structures.

The game is not finished yet. Remaining tasks and ideas:

* To improve human player interaction with the game. It is implemented on minimum level now.
* To add new science types (e.g. warfare).
* To add new terrain type (mountains, deserts, etc).
* To add diplomacy.
* To improve AI scripts for smarter AI behavior.
* To improve cultures influence on countries and people.
* Probably I need to replace current implementation of map (JavaFX components) by another implementation (OpenGL?).

# Appendix. Source code

<deleted>