1 Introduction

1.1 Design goals

The model should be decoupled from other parts of the program so that it could be reused in another context. Other parts of the program should be as independent as possible so that additions and changes doesn't require an understanding of unrelated parts of the application to be implemented.

1.2 Definitions, acronyms and abbreviations

Java, platform independent programming language.
GUI, graphical user interface.
MVC, a software architectural pattern to separate the model, view and controller of a GUI-application.
OpenGL, a cross-platform API for rendering vector graphics.
Overlay, buttons and similar components on top of the OpenGL surface view.

Player, a participant in a game.
Turn, a player can only act during his turn
Game Phase, phases, together making up one turn
Territory, a part of the world map that can be occupied by players.
Army, the game pieces used to occupy and conquer territories
Continent, a group of territories that, when all held by one player, grants bonus armies at the beginning of each turn.

2 System design

2.1 Overview

The core of the application is built as an MVC. Network functionality, OpenGL graphics and parts specific for Android is built around this core. The controller manipulates the model based on touch inputs.

2.1.1 Event paths

The controller manipulates the model.
The view observes the model and tells the graphics to change according to changes in the model.
All interaction with the overlay are passed through the main activity of the application before reaching the controller.
Touches on the OpenGL surface view representing interaction with territories are handled directly by the controller. Events from the Network is handled in a similar fashion in the controller.

2.2 Software decomposition

2.2.1 General

MainActivity is the entry class of the Android application.
model contains the classes representing the model, including the main "Risk" class.
graphics contains the OpenGL implementation
network contains all classes used exclusively for multiplayer games
Controller is the controller class manipulating the model based on user inputs

View is the view class that updates the graphics and the overlay when the model changes
Overlay is the class responsible for changing the graphical overlay

2.2.2 Decomposition into subsystems

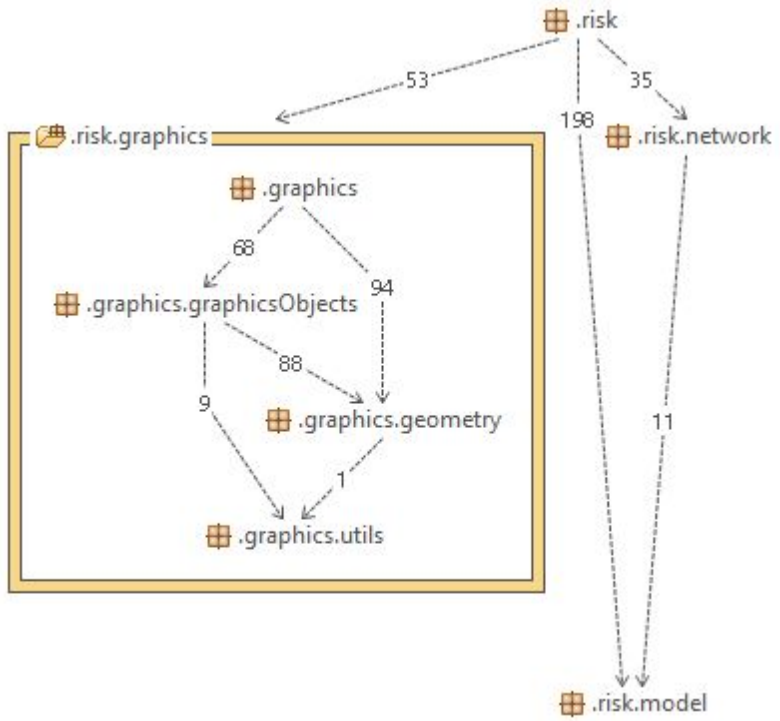
The application consists of the main system and two subsystems, Networking and Graphics.
The main system works completely independently of the Networking system. Changing or removing the networking capabilities would therefore be trivial. The Networking package acts as a wrapper around the google play networking api.

The Graphics system is also implemented as a subsystem. The graphics know nothing the application and functions completely independently. The graphics subsystem is responsible for loading in the correct world data and displaying as the application desires. The application interfaces with the graphics through two classes. One which is responsible for the initializing and displaying the graphics, called the RiskGlSurfaceView, to which the application supplies the correct SVG-file which represent the world. A second which is responsible for that class in the graphics package called the GraphcisManager.

2.2.3 Layering

Not Applicable.

2.2.4 Dependency analysis



2.3 Concurrency issues

Although the most of the application is executed sequentially a minor part is running on its own thread. The gpu-api, OpenGL, requires its own thread. Resources such as meshes and textures that the API can access needs to be initialized on the same thread. When a *GLObject* is created it is sent to the renderer which is calling the *GLObjects glInit* method from the correct thread. This method does the initialization that needs to be done on the GLThread. Removing

GLObjects from drawing is done in a similar fashion.

2.4 Persistent data management

The application stores the majority of its persistent data in an SVG-file and numerous XML files. This SVG file is storing how the world looks and which regions are connected to which. The XML files are responsible for how the ui looks. Additionally there exists a couple of images for the cards.

All these files are imported and loaded with the resource system provided within android runtime. All files are bundled into the application apk.

2.5 Access control and security

On startup the user has to log into its Google Play account. Its security is guaranteed by google. The application, as all android applications, is self contained and poses no threat to the underlying os.

2.6 Boundary conditions

Not applicable.

3 References

https://en.wikipedia.org/wiki/Model-view-controller
http://www.hasbro.com/common/instruct/risk.pdf
https://play.google.com/store/search?q=border%20risk