Kasper Järvinen 275435
Vilhelmi Lehtonen 268600

Game

Pirkanmaa Conquest is a 2 player click-based strategy game where the Players conquer Pirkanmaa by spreading their territory and collecting resources from the rich lands. Clicking a tile on the map and building/assigning workers to it claims ownership of that tile and adds Player's own color to it. Player's resources and available actions on current tile are visible on the right panel. Resource labels and available buttons change according to the active tile. Active tile is highlighted in Player's own color. Here are the main objects of the game:

Tiles

- Grassland – Just plain grass. Good for food.
- Forest – Like grassland but taller grass and some trees. There is wood here.
- Water – Wet. Cannot do anything with this tile, as workers would drown, and buildings would get water damage.
- Lava – Build and assign workers here at your own risk. It's very hot.
- Rock – Rock. This is where the "gold" lies. Get ore and stone here.

Workers

- BasicWorker – Jack-of-all-trades. Put it anywhere and gain resources steadily.
- Miner – Most useful when placed on a rock tile with a Mine building on it.
- Teekkari – The god of the game. Can remove opponent's buildings and participate in a Teekkarifight. It comes at a price though, as it's the most expensive worker.

Buildings

- Headquarters
- Outpost
- Farm
- Mine – Can only be built on rock tile.
- TuniTower – Worshipping this with your Teekkari will earn you a great reward!

Starting the game

To start a game, choose New Game. First you will enter player names and choose desired colors. When ready, choose Start Game. Game map is drawn based on a random seed, and thus is different each time. To start performing actions in the game, select some tile from the game map.

NOTE: Load Game is not implemented. We ran out of time.

Playing the game

Player in turn is displayed on the top right. Add workers/buildings on tiles by selecting some tile that is not owned yet and select desired option from the panel on the right. When you are done and/or max amount of actions is reached, choose "End turn". This will calculate tile and worker actions and perform resource modifications. Now it's the other player's turn.

Winning the game

When 100 rounds have been played, the game will be ended, and an end dialog will be presented. Both Player's final resources are displayed, and a winner is calculated. See "Rules" for more details about how the winner is calculated.

Harjoitustyö – Loppudokumentti
Kasper Järvinen 275435
Vilhelmi Lehtonen 268600

Rules

-2 players
-Tiles have limits as to how many buildings and workers can be placed on the tile.
-Only 1 Teekkari at a time for each player.
-Only 2 actions (e.g. add building, add worker) per round.
-Water cannot be own, thus it cannot be built upon nor can any workers be assigned.
-Building/adding workers on lava will result in penalty (deduction of resources).
-Other Player's buildings may only be removed by placing Teekkari next to it.
-Other Player's cannot be removed (excluding Teekkari by winning Teekkarifight).
-Player can gain resources by worshipping TuniTower by having a Teekkari next to it.
-Game ends when 100 rounds are played.
-Game is won by the Player who manages to get the most resources in 100 rounds – but there is a catch: Each resource is weighed differently: money 1.5, food 1.1, wood 1.3, stone 1.1, ore 1.3. So total score differs from total combined resources.

Known issues

-Load Game is not implemented, we ran out of time. Clicking Load Game doesn't break anything, it just doesn't do anything.

-In start dialog, when player selects his/hers/others color with QColorDialog, a message prints in the application output: "*GtkDialog mapped without a transient parent. This is discouraged.*" This is a GIMP Toolkit warning that we cannot do anything about, since QColorDialog uses native color dialog. This warning is well documented e.g. in here:

https://www.linuxquestions.org/questions/debian-26/%5Bsolved%5D-gtk-message-gtkdialog-mapped-without-a-transient-parent-this-is-discouraged-4175550278/

-In TeekkariFight dialog, when reaction game is played, the button can be clicked when it is not yet enablet which results in very good score. We assume this is because disabled buttons still send the click event, and when the button comes back to life, it is instantaneously clicked. This won't be of harm, if the players agree not to cheat. We had no time to implement a fix to this.

-Some user selected colors may not be visible in the UI very good.

Extra functionalities (for bonus points)

Cool backstory

See Backstory.txt in our group's repo under Documentation.

Multiple dialogs

The game features 5 dialogs in addition to the one required. We thought that a total of 6 dialogs surely must be worth something 😊(?).

Styles.cpp

I mean look at the complexity of that thing! Kasper has put his heart and soul into drawing sick representations of workers, tiles and buildings using the most masochistic methods imaginable. Might as well mention it, since it took most of Kasper's time.

Teekkarifight

A game inside a game. Who would have thought? The teekkarifight reaction minigame settles the debate of who's the Pirkanmaa reaction king in the game. Loser loses his Teekkari worker from the game in addition to his honor.

Teekkarifight can be initiated when Players have Teekkaris as each other's neighbours. Clicking the appearing "Fight" button will launch an epic reaction game. Click "Begin" to initiate the reaction game. Player 1 goes first. To indicate that he's ready, he clicks the "Start" button. When the "Stop" button becomes active, he must quickly click it in order to get a good reaction time. After that, it is the second player's turn.

Color picker

We're not sure whether or not this could count as additional part, since it wasn't too hard to implement. Still, we decided to mention it because we thought this was a neat idea.

Classes and responsibilities

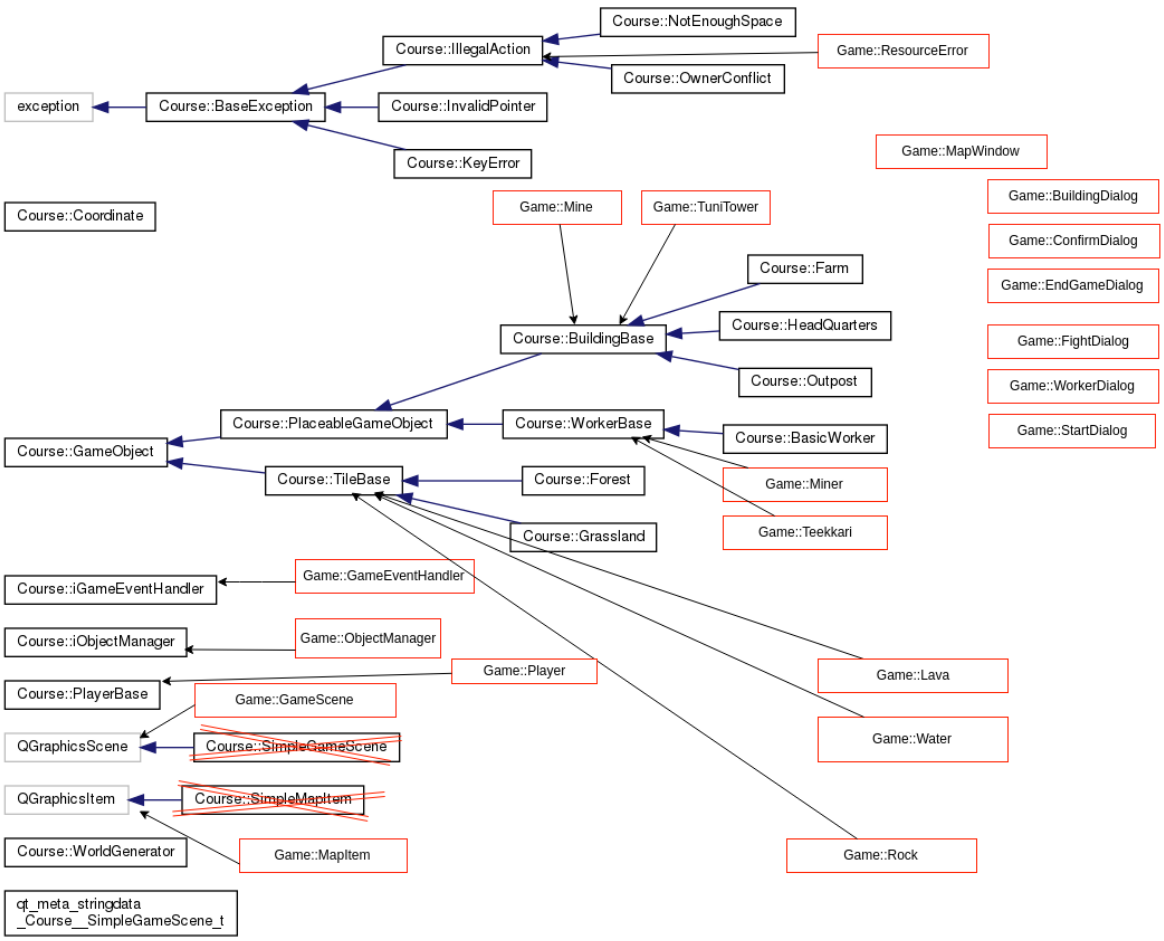

Figure 1: Class diagram

Kasper Järvinen 275435
Vilhelmi Lehtonen 268600

Class diagram shown in figure 1. Red classes were implemented by Vilhelmi and Kasper, black ones by course staff. More detailed information may be found in respective classes' documentation in the code.

MapWindow

Main class in the Game. Acts as a controller of sorts between the UI and GameEventhandler and ObjectManager. MapWindow receives signals from game actions performed by the Player's. The most important functionality is handling tile clicks of the scene based on signal emitted from the scene. Different tiles require different actions and the logic is mainly stored in GameEventHandler. MapWindow makes requests to the GameEventhandler and ObjectManager and communicates these responses to the GameScene and UI.

MapItem

QGraphicsitem class that is responsible of single GameObject's graphical element. Uses mainly the styles.cpp file to create the graphics.

GameScene

QGraphicsScene that shows the rendering of the game map. Draws the rect and is responsible for adding and removing items to and from scene using MapItem class. Handles the highlight of the clicked tile.

GameEventHandler

Handles most of the heavy lifting done in the game. Performs actions based on user actions, which are communicated by MapWindow. Returns answers to MapWindow to be displayed to the Player's and keeps track of the flow of the game. Most actions revolve around knowing the current Player and the currently highlighted active tile. Handles resource modification, keeps track of Player's, keeps track of actions per round, and checks certain game situations by analyzing the current active tile in conjunction with ObjectManager.

ObjectManager

Used mainly as a storage for all game tiles. Tiles are added here. Doesn't contain too much logic but handles requests to access tiles with their coordinates and ID's. Also checks and keeps count of dull tiles (tiles that cannot be performed any actions on).

Player

Holds all necessary information about a Player in the game, such as name, resources and color. Inherits Course side playerbase, but Player's objects are not used and not stored in the class. Class is used by GameEventHandler.

Styles

Forms the tiles, workers and buildings. The items graphics are made by creating polygons, brushes and pens and drawing them in certain locations depending on the type of the item to be drawn, and the number of other items on the same space (tile). Uses several own functions, although there was no need for an individual function for each object type. Is used only by MapItem class.

Kasper Järvinen 275435
Vilhelmi Lehtonen 268600

Group member responsibilities

Planned

In the initial plans we discussed that Kasper would do more stuff on the visual side of things and Vilhelmi would do more of the “back-end” work. Trello would be utilized to split the work into smaller pieces and see what the other member was doing. Initially Kasper started to look at GameItem and GameScene classes and Vilhelmi started to look at GameEventhadler and ObjectManager. Overall Kasper was to look at more of the visual side of the Game and finding issues and vulnerabilities and Vilhelmi would focus first on implementing the necessary interface methods needed by the course side. A couple of get-togethers were arranged at the but due to the fact that we both were working while doing the project, most of the communication would happen via Discord.

How it turned out

As the project progressed, our roles further strengthened so that Kasper was fully concentrating in the visual representations of the tiles, workers and buildings while Vilhelmi was wrestling with inheriting classes and implementing new features. Even though our timetables were rather challenging we managed to do some progress. Trello table was not used quite as heavily as we initially expected, since our communication was quite good otherwise. There is only two of us after all and the program is not that massive. Discord was heavily utilized during the project. Extension of the deadline helped out quite a bit. After a while found a programmatical junction point when Kasper needed some “back-end” functionalities to help with his styling.