Colonizing Pirkanmaa Game Documentation

By: Team Flatmate

Team members:

Tran Hoanh Le – 281721 – hoanh.le@tuni.fi

Ben Kavanagh – 281654 – benjamin.kavanagh@tuni.fi

1. Introduction:

Colonizing-Pirkanmaa game is a course project given in TIE-02408. The game is based on an existing board game and the course staffs provided base classes for building, tile and worker as well as other core functionalities. The task of our team is to implement the graphical user interface, defines details of game play and implement them.

Our team has implemented the minimum and intermediate requirements, as well as two additional features and some extra works qualified for bonus points. Section 2 describes the software architecture and details the classes implemented by our team. Section 3 describes the implementation of extra features in our game and extra works. Section 4 provides information about the workload division. Sections 5 details the gameplay implement by our team. Section 6 concludes the documentations.

2. Software Architecture:

The game is implemented by both our team and course staffs. Details workload implemented by course staff can be found via the following link:

All the implementations by the course side are in the namespace **Course**. Our team implementation resides in two main namespaces **Student** and **Ui**.

THE PROJECT STRUCTURE OF CLASSES IMPLEMENTED BY COURSE STAFF

- **Building**
  - **BuildingBase**: represents base-class for different buildings in the game
  - **Farm**: represent farm production buildings in the game, which create resources for the player.
  - **Outpost**: represent outpost building, which the player can buy to claim ownership of neighborhood tiles.
- **Core**
  - **PlayerBase**: represents base class for Player class.
  - **PlaceableGameObject**: represent Gameobject that can be placed on Tile

    - **Gameobject**: a base-class that contain's general information on different objects in the game
    - **BasicResource**: implement resource operations and structure in the game
- **Exception**
    - Implementation of the following exception in the game: illegalaction, invalidpointer, keyerror, noteenoughspace, ownerconflict
- **Tile**
    - **Forest**: represent Forest Tile in the gameworld.
    - **Grassland**: represent Grassland Tile in the gameworld.
    - **TileBase** represent base-class for all different tile-classes in the game.
- **Workers**
    - **WorkerBase**: represent abstract base-class for all worker-objects.

THE PROJECT STRUCTURE OF CLASSES IMPLEMENTED BY US

- **Building**
    - **HousingBase** (and other derived classes from **HousingBase** – **ApartmentBlock, LargeHouse, SmallHouse, Skyscraper**): represents housing building in the game, which add different number of workers to the player.
    - **Mine** and **Sawmill**: represent production buildings in the game, which create resources for the player.
- **Core**
    - **GameEventHandler**: handles events happening in the Mapwindow such as create players in the beginning, assign workers from the player to a tile, check if a player has won.
    - **Gamescene**:
    - **ObjectManager**:
    - **Player**: represents a player in the game, which is used to store and access NewBasicWorker object and Tile object and keep track the Resources and Workers the player has in the game.
- **Exception**
    - NoOwner:
- **Graphics**
    - **AssignDialog**: implement a dialog to take the number of workers the player wants to assign to the selected Tile.
    - **HighScoreDialog**: implement a dialog to show the top 5 scores of all time. Score is the number of turns the player takes to get 5000 resources total and 50 workers in total.
    - **Mapitem**:
    - **Mapwindow**: implement the main GUI of the game, where the players do actions on. This class gets the handler
    - **RulesDialog**: implement a dialog to show the players instruction on how to play the game.

- **SetPlayerDialog**: implement a dialog to take the number of players in the game.
- **StartDialog**: implement a dialog to show in the beginning of the game, where the players can go to RuleDialog to see the instructions or start the game and go to Mapwindow.
- **UnAssignDialog**: implement a dialog to take the number of workers the player wants to unassign from the selected Tile.
- **WinDialog**: implement a dialog to display winning message after a player has won, asking if the players want to play again or exit the game.

• **Tile**
- **Rock**: represent Rock Tile in the gameworld.
- **Sand**: represent Sand Tile in the gameworld.
- **Water**: represent Water Tile in the gameworld.

• **Workers**
- **NewBasicWorker**: represents new basic worker in the game, which the players can only acquire by buying housing-type building.
- **Farmer**, **Logger** and **Miner**: represent farmer, logger and miner worker in the game, each has different efficiency and production-focus.