PIRATE SPADES README

Contents

1. Project Overview incl. system requirements 1

2. Production of your system analysis 2

3. Production of your system design 2

4. Production and demonstration of the working system 2

5. Final Project Overview 4

6. Instructions - installing and in game 5

7. Version control repository 6

8. Reflection 6

1. Project Overview incl. system requirements

We started out with discussing, what our project should be. Andreas had a fun game he suggested, and we all agreed on making this. We handed in the following overview and requirements.

**Authors:**

» Helena Charlotte Lyn Krüger (hclk@itu.dk)

» Morten Chabert Eskesen (mche@itu.dk)

» Andreas Hallberg Kjeldsen (ahal@itu.dk)

**Abstract:**

Card game for two players and up. You can play for fun, for money or for the glory. A typical game consists of 20 rounds, first round each player is dealt 10 cards, next round each player is dealt 9 cards and so forth until 1 card is being dealt. Then you go from 1 card, to 2 cards and up to 10 cards again, then the game is over. Each round you have to make a prediction as to how many tricks you will get during the round. When everyone has made a prediction, they're revealed. When all cards have been played, each player is given 10 points plus one additional point per trick if they guessed correctly, otherwise they get subtracted one point per trick they're wrong. The player with the most points at the end of the game is the winner. Ace is highest, 2 is lowest. You have to follow suit when possible. Spades are trump (Ace of Spades is a sure trick etc.).

**Requirements (Mandatory):**

» Must model the game "Pirat Bridge".

» Must involve at least two players.

» Must have an interactive user interface.

» Must store player state between games.

» Must accurately model players and non-player/non-card entities relevant to the game.

**Requirements (Secondary):**

» Should be playable over network.

» Should support different rule sets (only minor modifications to the main rule set).

2. Production of your system analysis

When we started hacking game logic and the networking the coupling was too high. Everything depended on each other and made it impossible to edit one of the classes without the other one crashing. When we made the choice to make this differently everything started to get better including the coupling. We didn't analyze the XNA code in the beginning, as it wasn't made. We had to make the pictures for the game and some of the game logic and networking to make it.

3. Production of your system design

In the beginning we wrote a game logic. We choose to make it different because the design wasn't good enough. It was too difficult to code game logic and networking at once as said at the point above. We did this part of the system once again from scratch. We didn't include this bon file as we didn't think it was important. The This part of the project is a .bon file, which is on the CD as well.

4. Production and demonstration of the working system

The following are screenshots of the game. These are shots of the most important parts of the final game.



The main menu where you can create or join a game, edit settings or read the rules.

****

When wanting to join a game, the program search the network for created games, which hasn't begun.



When you have joined a game and the host hasn't yet started the game (perhaps waiting for more players) this is where you'll wait.



Here we see a screenshot in the game. To the right we see the tricks and bets. In the bottom the cards, and the rest of the shot shows you(from the back) and your opponent(s).



Here the scoreboard function is shown. This works by holding down the tab button

5. Final Project Overview

In the beginning we wrote one game logic - this is not the one committed in the final project. The design of the game logic, networking and collections has been edited once through the project process. The XNA part on the other hand has only been made once. We have parted the XNA into small folders - GameModes, InGameFunc, Misc, Music and settings. The GameMode folder contains 4 classes, which implements the interface GameMode, as all 4 classes need a Load-, Update- and Draw method. In IngameFunc we have the code for the which table to load (there are a table for each possible number of players - that is 4 different tables). In the Misc folder we have all the textboxes, buttons etc.

The cyclomatic complexity has been good the whole time as well as the maintenance index. The analysis' of the final project shows some fine numbers, which we are satisfied with. We have analyzed pirateSpades by among other things getting NDepend to calculate the code metrics for us, which mostly shows a high maintainability index, which is good. The class coupling is a bit high in some of the classes - for example pirateHost, but it is a big project, and a lot of com­ponents have to corroborate (All the numbers can be seen in the excel document on the CD called pirateSpadesMetric). The cyclomatic complexity is good in most of the methods. It is mostly under 10, but some methods has a complexity higher than 10. As seen in the other excel document called pirateSpadesGame­Metrics, which is the XNA part of the project, the maintainability index is a bit lower than pirateSpades as it's here the project is being built and the interface is getting coupled. The cyclomatic complexity of pirateSpadesGame is as pirateSpades good and mostly under 10.

the requirements has been the same throughout the hole process even though there is one requirement, that haven't been fulfilled and another requirement we should have added (Check point 8 page 6).

6. Instructions - installing and in game

**Installing:**

**In game:**

1. Create a new game

- Push the create game button

- Enter your name and press 'change'

- Enter a server name (the name of your game)

- With the keyboard arrows, enter the number of max. players

- Press create game

- In game:

- Press the black box in the right bottom corner and use the keyboard arrows to set your bet and press the bet button.

- It will be displayed in the middle of the table, which player gets the trick.

- As the game consists of 20 rounds, you can press down the tab and see the score of each player in each round.

- In the right upper corner you can see you and your opponents bet and number of tricks.

2. Join a game

- Press the join game button

- Enter your name and press the 'change' button

- If there is no games try to press 'refresh'

- If a game appears click on the game and press 'join game'

3. Read the rules

- Press the 'rules' button

- When finished press the 'back' button

4. Edit settings

- Press the button 'Setting'

- Editing your name, press the old map besides 'player name' and type the wanted name

- Setting the volume of the music, click on the black box and use the keyboard arrows, up and down

- Press apply to use the edited settings or cancel

5. Exit

- Press the 'exit' button

7. Version control repository

The following link is the link to our version control repository: https://github.com/ahkjeldsen/piratespades

8. Reflections

We have fulfilled almost all of our requirements including the secondary's. The only requirement we haven't fulfilled is the requirement with supporting different rule sets - for different occasions (normal mode, party mode etc.) We are overall happy with the program. The game logic, collection and networking part of the system is validated with pex generated unit tests and handwritten unit tests which covers 81 % of the system. The coverage rapport is be to found at \_\_\_\_. The XNA part of the system is validated by playing the game and checking that everything works as it intended to. One thing we didn't get to implement was safety against cheating. We wrote contracts in the code, but one is able to create a client, start a game with only him in it and crash the system, or the dealer is able to give himself the highest spade. This is some improvements that could be done, if we had more time.