Bilkent University

Department of Computer Engineering

CS 319 - Object-Oriented Software Engineering

Analysis Report

Project Group 11 - LUDO

Ahmet Taha Albayrak - 2131440

Alp Pehlivanoğlu - 21202023

Asli Cengiz – 21301183

Mustafa Motani 21402995

Contents

[1 Introduction 1](#_Toc254263637)

[2 Overview 2](#_Toc254263638)

[3 Functional Requirements 5](#_Toc254263639)

[4 Non- Functional Requirements 7](#_Toc254263638)

[5 System Models 8](#_Toc254263639)

[6 References 9](#_Toc254263651)

Analysis Report

CS319 Project: Ludo

# Introduction

Ludo is a board game predominantly played in South-East Asia. It was originated in about 6th century and it has since evolved with several variations of game existing in present. It is very easy to play multi-player game where each player race their four tokens to the finish line and first one to finish wins the game. Number of places each player have to move is determined by a roll of a die.

This report contains detailed description of the game and our implementation segmented under easy to understand topics covering rules of the game, gameplay description, scenarios and user-controlled features. We have planned to include new features that are not found in traditional Ludo game e.g. random bonuses on boxes.

# Overview

This game is based on a game called Ludo which is an Asian board game for 2 to 4 players. In this game each player has 4 tokens and players race their tokens from start to finish based on the result of rolling a dice.

The game has “Cruciform”. The board is divided into 4 equal regions by a cross in the middle and each region belongs to a single player. At the beginning, all the tokens are staged in these regions which are called “houses” and they are out of play. Players need to roll a 6 on the die to get a token out of the house. Roll of the die can only be implemented on one token i.e. if the player rolls 6 he can’t move one token by 3 and other by 3. Tokens are moved in clockwise and placed on squares which are around the cross in the middle. If the place that a token is going to be put is already occupied by an opponent, the token that is occupying the place goes back to its house.

First player to successfully finish the race by reaching the finishing square wins the game and if there are more than 2 players in the game, they continue playing until there is only one player who couldn’t finish is left.

**2.1 Gameplay**

At the beginning, each of 2-4 players roll the die and the highest roller begins to play. After that, all the players take turns in clockwise direction and game continues in this direction.

Before the game starts, all the tokens are out of play and are staged in the regions that each 4 players have in different colors. When a player rolls a 6, he can enter a token into play from its staging area called a “house”.  There is a starting square for each player and a token enters the game from that square. If the player has no tokens in the game and can’t roll a 6, the turn passes to the next player in the clockwise direction.  If the player has at least one token in the game, he plays one of them (or the only one) according to the value on the die. Tokens are played forward in the clockwise direction along the squares around the cross on the board. Players must move a token according to the value of the die rolled and when there is no move possible the turns passes to the next player.

After entering at least one token to the play, players can decide between entering a new token or playing one of the ones in the game when they roll a 6. If the player rolls a 6 he or she gets an additional (bonus) roll. However, getting three consecutive 6s skips the turn of the player.

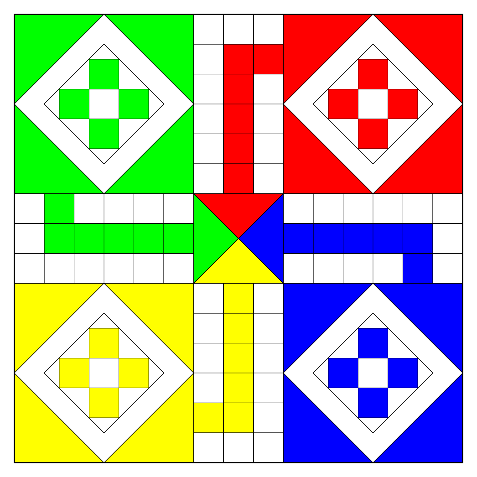
If a place that a token is going to be put is already occupied there are 2 possibilities: the token occupying the place either belongs to the player that is going to put the token there or to another player. In the first case (if the occupying token belongs to the player) player can move(stack) the token to be played on that square he already occupies. In the other case, player can move the token to the square occupied and the opponent’s token is returned back to the house of its owner. To enter that token back into the game, the owner must roll a 6 again.

The race continues until all 4 tokens of a player takes a turn around cross and are placed on the squares of the home column of that player. This column is colored in the same color with the tokens and the house of that player. When a token enters the game and  turns around the cross and reaches the home column, player moves the token along the home column and cannot continue moving it around the cross. These house columns are safe areas for their owners and other players are not allowed to put their tokens to the squares in that column. When all 4 tokens of a player are placed consecutively in that column, that player wins and the game ends for that player. However, the others  continue playing to determine second, third, and fourth-place finishers.

**2.2 Board**

In Ludo, players’ movement based on board. Every player has different starting point, finish position and house. House contain tokens which are not released. Also, tokens can get powers up from random block which is on the path. Different boards might differ due to different variances of game. Some examples of maps are :

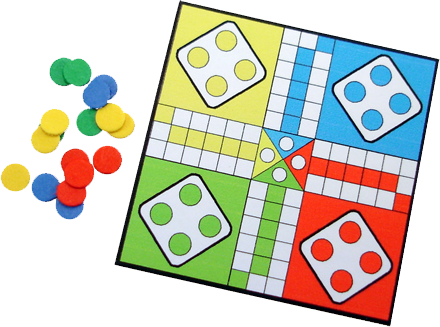
1. Classic Ludo:



(Figure 1 - Classic Ludo)

Link: <https://upload.wikimedia.org/wikipedia/commons/thumb/0/03/Ludo_board.svg/2000px-Ludo_board.svg.png> [2]

1. Classic Ludo-2



(Figure 2 - Classic Ludo-2)

Link : <http://www.mazegames.biz/images/Ludo.png> [3]

**2.3 Tokens**

Each player in the game has 4 tokens. Each player picks a color and the tokens are colored in that color. The color may be green, blue, yellow or red as they are in the classical game. In addition to these, this game offers other colors.

**2.4 Dice**

Using number of dice depends upon the players’ mutual choice. For our implementation we would roll a single die.

# Functional Requirements

## 3.1 Play Game

Ludo is basic strategy board game. The main purpose of game is to bring all token of player to finishing position. This game can be played by 2 to 4 players at the same time. Every player must roll six to enter his token to enter starting position. If player doesn’t roll six, player cannot enter his token to starting point. Player can only race with the tokens which are outside the house. Player can send back other players’ token to their houses by stepping into their occupied positions on the board. To reach the finish position, player must cross all the map. If player enter all tokens to finishing position first, he or she wins.

--Power Ups

## 3.2 Change Settings

Some default settings can be changed by Players demands. There are these settings can be changed by player:

* Player Id
* Token’s color
* Sound On/Off
* Map Type
* Color blind
* Automatic die roll

Before game starting, Player can choose any nickname he wants and his tokens color. In addition that, map can be changed by all player decision. Ludo have lots of variance related to types of map. All player should select the map which they will play. If they don’t select any map, game will create default map. This setting can be changed before starting game. There is only one setting can be changed during game. This setting is mute. Player can mute the game any time they want.

## Leaderboard

Player can see most winner player among all player. Leaderboard will show only first 3 players in selected classmen. This encourage to players win to show their name in scoreboard.

## 3.4 View Help

Player find all the essential information about gameplay and game.

This information includes:

* Gameplay information
  1. Rules of game
  2. Control and token selection
  3. Map Type
* Settings information
  1. Changing settings
  2. Useful hints about settings

This part help newbie player to learn game quickly. Also experienced people can find useful information about all map which game support. Gameplay can show difference. This documentation is encouraged to be read before playing. This includes not only gameplay information but also some information about how settings can be changed.

## View Credits

Players can see information all the developer, designer of program. We will also put our email address so that players can notify us in case there is as a bug in our game thus, allowing us to modify our game making it bug free. Moreover players can give us their valuable feedbacks on how to further improve our game.

# Non-functional Requirements

## 4.1 Performance

In order to minimize delay for the users, control mechanisms of the game will have low response time. Since the game includes graphical interfaces and sounds, we’re going to try to keep the speed of the game as high as possible. And, we will try to reduce system requirements in order to make it playable for almost all computers.

## 4.2 Graphics

Graphical interfaces will be an important issue for our project. Even if our game will be board game, we would like to make graphics smooth and good looking so that our game might excite attention of users. Since, our game is not that complicated one, our interfaces will be user-friendly. So that user can easily play the game.

## 4.3 OS

We consider making our game to be executable in every operating system except for mobile systems.

## 4.4 Pseudo Function Requirements

Our game does not require any installation or setup process. It will be implemented in Java.

# System Models

## Scenarios

* The player starts a new 2 player game. She selects red a house.. She rolls a 2 on the die. Her turn finishes since she didn’t get a 6. Each player will roll the die again and again without doing anything until one of them gets 6. Even though her opponent starts the game first, the player to roll 6 first enters the game first. Therefore, player with the first roll has a higher chance to get 6 first.
* The player gets annoyed of the game sounds because he or she is already losing, he can mute the sound by accessing it through options. Moreover, he or she can also forfeit the game.
* The player rolls a 4. He has two movable tokens. He can choose one of them to move complete four steps. He can’t move 1 on one token and 3 on other.
* The player rolls 3. He again has two movable tokens. With one of the token he can kill the opponent’s token. However, it is not necessary to do so. If the player wants to move the other token he can. If the player decides to kill the token, he or she moves his or her token at the occupied box sending opponent’s token back to his or her house.
* Player moves his token and a random bonus is triggered e.g. invulnerability. If the opponent’s token attempts to move at this square. It won’t kill player’s token but will just stack on that square with both tokens side by side. Similar thing would happen if the player wants to move his token the square which is already occupied by one of his or her other token.

# References

1. Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.

[2] Classic Ludo-1 <https://upload.wikimedia.org/wikipedia/commons/thumb/0/03/Ludo_board.svg/2000px-Ludo_board.svg.png>

[3] Classic Ludo-2 <http://www.mazegames.biz/images/Ludo.png>