

Benjamin Hoang

Daniel Burris

Irene Kasian

Zachary Haynes

Team 8 - CS 321

Ultimate Tic-Tac-Toe

# Contents

[Contents 1](#_Toc467427970)

[Project Description 2](#_Toc467427971)

[Project Management 3](#_Toc467427972)

[Use Cases 3](#_Toc467427973)

[Requirements 4](#_Toc467427974)

[Use Cases 4](#_Toc467427975)

[Implementation 4](#_Toc467427976)

[Discussion 4](#_Toc467427977)

# Project Description

Our project is to create ultimate Tic-Tac-Toe. Some of the functionality for this game will be: player vs. player, simple computer AI, database management for features such as: highscore, time elapsed, leader board, etc., menu with options to play, change settings, close application, etc.

# Project Management

History

[Github Commits](https://github.com/BenjHoang/Ultimate_tictac) [Github Repository](https://github.com/Dburris13/UltimateTicTacToe)

Personnel

Daniel Burris - Senior in Computer Engineering, 3 years experience with C++ / OOP.

Ben Hoang - Senior in Computer Science

Irene Kasian - Junior in Computer Science

Zach ~~~~

Effort

6 hours a week



# Use Cases

# Requirements

Overview

This game has to address problems such as File based database management, dealing with GUI design and animations, working with java.

Defined requirements

GUI - our application is GUI-based, allowing the user to play the game, change settings, load profiles, and check the high scores all from a GUI.

Text formatting and processing - We use HTML to format a couple of jLabels, we also use group layout to organize text.

Graphics - our main menu features animation, our game scene is created and manipulated through our Game hierarchy classes.

Storage and retrieval of information - we have a user profile system as well as a high score system that is all file based information.

Editing and configuring the software product - we have a settings menu that allows the user to customize the software product.

Project specific requirements

Maybe AI

Maybe Loading/Saving game state.

Maybe

Future modification and extensions

AI difficulty settings.

More customization.

Loading/Saving a game state.

Indexed summary list

Software with provide a basic user interface for menu navigation.

Software will feature a fully functioning Ultimate Tic-Tac-Toe game.

Software will be configurable based on user settings.

User settings as well as game statistics can be saved / loaded into user profiles.

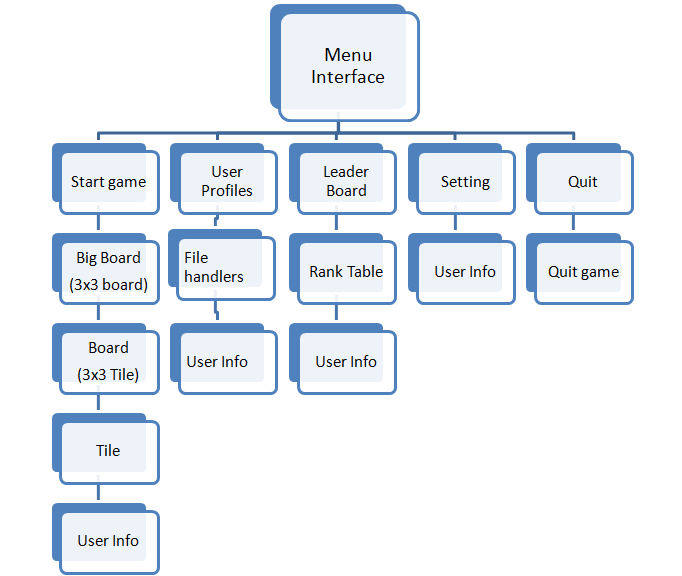
Software will feature a high score system.

Software will feature animation.

Associated tests

|  |  |
| --- | --- |
| GUI | Enter into all menu branches and return to main menu. |
| Text Processing | Visually verify that all menus look correct. |
| Graphics | Visually verify that game scene looks correct and main menu animation is correct. |
| Storage and Retrieval of Information | Load user profiles, change user profiles, load high score table, complete game, load high score table again. |
| Editing and Configuring the Software Product | Change settings with user profiles and without user profiles and enter game scene. |
| Artificial Intelligence | Play several games against the AI. |

# Design



# Implementation

Packages and classes

We have four packages: GUIStuff, MenuScenes, resources and Group8. The GUIStuff package contains all the miscellaneous classes and images/gifs used with the GUI. The MenuScenes package contains all of the JFrames used in this project, such as the Menu, the settings menu, and the actual game board.

\*\*Resources probably needs to either be deleted, and the stuff in it moved to GUIStuff, or add more images/gifs/icons.

\*\*Group8 needs to be renamed/reorganized. Mostly contains all model related things.

Application program interfaces

Classes

\*\* reference Javadocs somehow

1. AnimatedPanel -- defines a JPanel that holds BackgroundChars and moves them.
2. BackgroundChars -- an abstract class of objects that will be “animated”
3. XChar -- implementation of BackgroundChars; the object that is animated is the string “X”.
4. YChar -- implementation of BackgroundChars; the object that is animated is the string “Y”.
5. BigBoard -- encapsulates 9 Boards to create the Ultimate tic tac toe board
6. Board -- encapsulates 9 Tiles to represent 1 regular tic tac toe board
7. FileHandler --
8. Game -- represents the game scene. It implements the BigBoard class, and updates user profiles.
9. Player --
10. Settings -- jframe that allows the user to change game settings. Allows player to change the color schemes used, as well as the resolution of the game.
11. Tile -- an class that extends the JButton
12. UserInfo -- a class that defines player information such as name, scores, and settings preferences
13. MenuGUI -- a jframe that shows the game menu. User can navigate to all parts of the game from here (such as settings, leaderboard, etc.)
14. Rank\_table
15. UserProfiles --

Tests

Test plan

Tested functionality

Untested functionality

Requirement satisfaction

# Discussion

As with most of the projects in this class, the majority of our trade-offs revolve around schedules and time consuming. Do we want to spend time writing a complicated and creative computer opponent (a project specific requirement) or spend the same amount of time creating an animated login screen (a project requirement). Our code is certainly not the most abstract / generic is can be, but we tried our best to create general methods and organize private and public variables so that a potential future team would be able to look at our code and figure it out. There is some complicated logic in some areas of our code that could be simplified / rewritten, but we tried to balance out the complication with detailed javadocs.