

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 4](#_Toc467427973)

[Requirements 5](#_Toc467427974)

[Use Cases 7](#_Toc467427975)

[Implementation 7](#_Toc467427976)

[Discussion 8](#_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, leader board, and user profiles, menu with options to play the game, change settings (such as color schemes), load a user profile, and close the application.

# Project Management

## History

Throughout the semester, we have worked to develop an application that will allow a user to play Ultimate Tic Tac Toe against a friend or against a very simple AI. We first began with creating a GUI prototype of a menu and a game board. After creating the prototypes, we set to implement the main controls of the game, such as enabling/disabling pressed tiles and available boards, as well as checks for a game winner. During this stage, we also added a few settings options. Next, we worked on calculating highscores. While working on that, we realized that it may be useful to create user profiles that will all for easier storage of settings, and the amount of games played, etc. The links below lead to our Github repository for more details.

[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, 4 years practice C++/Java, Algorithms.

**Irene Kasian** - Junior in Computer Science, 2 years experience with C++

**Zach Haynes** -

## Effort

On average, approximate 6 hours per week were dedicated to this project between all 4 of us.



# Use Cases

## Main use scenario

## Load player profile scenario

# Requirements

## Overview

This game has to address problems such as file based database management, dealing with GUI design and simple animations, and working with java. In addition, this game needs to appropriately disable taken tiles, and enable/disable the correct boards a player can play on. The game also needs to be able to implement the rules of basic Tic Tac Toe, and Ultimate Tic Tac Toe.

It is assumed that the user knows how to play the game. The program will allow a user to play as many game of ultimate tic tac toe as they desire against another player or against a very simple AI.

## 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 few JLabels; we also use group layout to organize text.

**Graphics** – We have a main menu that features a simple background animation, and our game scene is created and manipulated through our Game hierarchy classes.

**Storage and retrieval of information** - we have a file based user profile system that creates a file for each user as well as a high score system that is also stored as a file.

**Editing and configuring the software product** - We have a settings menu that allows the user to customize the color scheme and resolution of the game board.

## Project specific requirements

1. have a basic AI to play against the user
2. The user is able to play the game with a friend or against the AI
   1. This including having correct enabling and disabling of the mini tic tac toe boards and not allowing a user to overwrite a previously played on tile.

## Future modification and extensions

1. AI difficulty settings
2. Saving/Loading an unfinished game
3. More customization options, such as color schemes, and fonts.
4. Randomly selecting which player goes first/which character they are (X or O)

## 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 basic animation.

## Associated tests

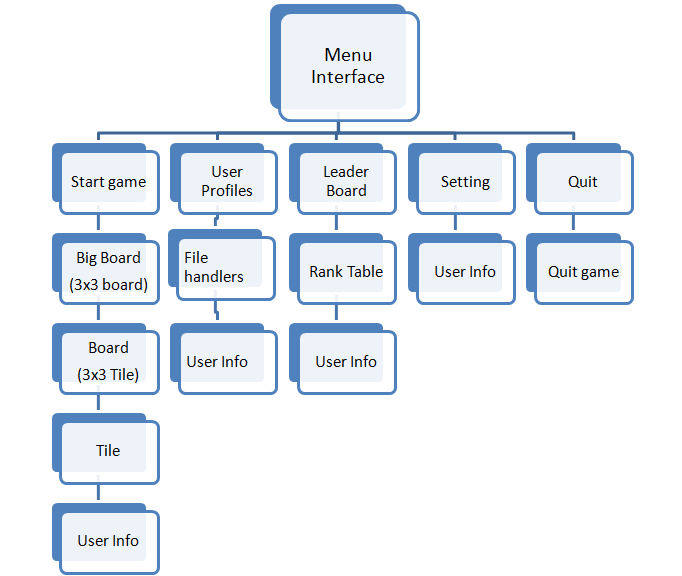
|  |  |
| --- | --- |
| GUI | Enter into all menu branches and return to main menu. |
| Text Processing | Visually verify that all menus look correct. |
| Graphics | Visually verify that game scene looks correct and main menu animation is correct. |
| Storage and Retrieval of Information | Load user 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

## Overview

\*\*Basic deciption of how the model works. Include CRC cards?

## Data Model: Object and State Design



## Views and Controls

??????

# Implementation

## Packages and classes

1. **Resources package** – this package contains the gifs used for the dialogs; this would be a good play to put images that can be used to further customize game appearance
2. **MenuScenes Package** – this package contains Jframes used in the application (except for the game board)
   1. MenuGUI
   2. Rank\_Table
   3. UserProfiles
   4. Settings
   5. Dialog
      1. WinDialog
      2. TieDialog
3. **GameFiles Package** – this package contains all the classes that define the game GUI and the game logic
   1. BigBoard
   2. Board
   3. Tile
   4. Game
4. **PlayerInfo Package** – this package contains all classes related to the player
   1. UserInfo
   2. Player
   3. FileProfile
   4. FileManager(?)
5. **GUIMisc Package** – this package contains all miscilanous classes that can be used with a GUI
   1. AnimatedPanel
   2. BackgroundChars
      1. XChar
      2. YChar

For more information about each class see Classes, or Javadocs

## 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.