Design Document

Six Men’s Morris

T02 Team 1

Contents

[1 Introduction 1](#_Toc446523504)

[2 Architecture and Module Decomposition 1](#_Toc446523505)

[2.1 Architecture 1](#_Toc446523506)

[2.2 Module Decomposition 1](#_Toc446523507)

[3 Module Guide 2](#_Toc446523508)

[3.1 Game 2](#_Toc446523509)

[3.2 CustomGame 2](#_Toc446523510)

[3.3 Menu 3](#_Toc446523511)

[3.4 GamePane 3](#_Toc446523512)

[3.5 MenuPane 5](#_Toc446523513)

[3.6 GameModel 5](#_Toc446523514)

[3.7 PieceType 5](#_Toc446523515)

[4 Uses Relationship 6](#_Toc446523516)

[5 Trace to Requirements 7](#_Toc446523517)

[6 Anticipated Changes 8](#_Toc446523518)

[7 Internal Review 8](#_Toc446523519)

[8 Test Plan/Report 8](#_Toc446523520)

[8.1 Test Plan 8](#_Toc446523521)

[8.2 Test Report 8](#_Toc446523522)

# Introduction

This document provides a detailed description of the design for an application that implements the Six Men’s Morris board game.

# Architecture and Module Decomposition

## Architecture

The model-view-controller (MVC) architecture was used in the design of the Six Men’s Morris application. This application was well suited for the MVC architecture since it could easily be decomposed into modules that make up the model of the game, a view, and a controller.

## Module Decomposition

The module decomposition for the application is shown in . The way in which the decomposed modules fit into the MVC architecture is also captured in this figure.

mvc.png

Figure : Module decomposition under MVC architecture

# Module Guide

## Game

**Class: Game**

Package morris

Provides a window for the Six Men’s Morris game. Handles input capture and processing.

*Interface*

*Uses*

GameModel, GamePane

*Type*

none

*Access Programs*

run: Environment

Runs the six men’s morris game within the GUI

*Implementation*

*Uses*

GameModel, GamePane

*Variables*

*Access Programs*

run: Environment

Runs the six men’s morris game within the GUI

## CustomGame

**Class: CustomGame**

Package morris

Provides a window for the Six Men’s Morris game. Handles input capture and processing.

*Interface*

*Uses*

GameModel, GamePane

*Type*

none

*Access Programs*

run: Environment

Runs the six men’s morris game within the GUI

*Implementation*

*Uses*

GameModel, GamePane

*Variables*

*Access Programs*

run: Environment

Runs the six men’s morris game within the GUI

## Menu

**Class: Menu**

Package morris

Provides a window for the Six Men’s Morris main menu. Handles input capture and processing.

*Interface*

*Uses*

MenuPane

*Type*

Environment

*Access Programs*

run: Environment

Runs the six men’s morris game within the GUI

## GamePane

**Class: GamePane**

Package morris

Provides the on-screen view for the Six Men’s Morris game where the game is rendered.

*Interface*

*Uses*

GameModel

*Type*

none

*Access Programs*

clickInTray(Point, Integer): Boolean

clickInBoard(Point, Integer): Boolean

clickInMenu(Point, Integer): Boolean

highlightMenu(Point)

*Implementation*

*Uses*

GameModel

*Variables*

gameModel: GameModel

Stores a reference to the game model that is being rendered.

winX: Integer

The width of the rendered output in pixels.

winY: Integer

The height of the rendered output in pixels.

gapFraction: Real

segmentLen: Integer

Length of the line segments that connect the board spaces in pixels.

pieceDiameter: Integer

Diameter of the pieces displayed on the board in pixels.

topGap: Integer

The number of pixels that separate the

leftGap: Integer

Number of pixels that separate the left side of the

trayGap: Integer

traySpace: Integer

inputMapBoard: Rectangle[ ][ ]

inputMapTray: Rectangle[ ]

inputMapMenu: Rectangle[ ]

f: Font

fm: FontMetrics

highlight: Integer

menuItems: String[ ]

errorString: String

*Access Programs*

run: Environment

## MenuPane

**Class: MenuPane**

Package morris

Defines a GUI instance of the six men’s morris game. Contains access programs to

*Interface*

*Uses*

none

*Type*

none

*Access Programs*

run: Environment

Runs the six men’s morris game within the GUI

## GameModel

**Class: GameModel**

Package morris

Defines a GUI instance of the six men’s morris game. Contains access programs to

*Interface*

*Uses*

none

*Type*

none

*Access Programs*

run: Environment

Runs the six men’s morris game within the GUI

## PieceType

**Enum: PieceType**

Package morris

Defines a GUI instance of the six men’s morris game. Contains access programs to

*Interface*

*Uses*

none

*Type*

PieceType

*Access Programs*

run: Environment

Runs the six men’s morris game within the GUI

# Uses Relationship

The uses diagram is shown in Figure 2.

2me3uses.png

Figure 2: Uses Diagram

# Trace to Requirements

The requirements of the application have been adapted from the problem description and are restated here for convenience:

R1: The user shall be able to choose to start a new game.

R2: At the start of a new game the board should include two different kinds of discs (blue and red), one for each player, initially placed on either side of an empty board.

R3: At the start of a new game there should be no discs already entered on the board.

R4: The order of play (blue first or red first) shall be determined randomly when starting a new game.

R5: The user shall be able to choose to create a custom game.

R6: When creating a custom game, the user can choose to enter discs to represent the current state of a game by placing different coloured discs in the frame.

R7: When creating a custom game, once all the discs the user wants to place have been placed in the frame (the end of this phase being indicated in some way by the user), the system should analyze whether the current state is possible or not and all the errors shall be highlighted in some way on the screen.

R8: The user shall be able to save a game currently in progress.

R9: The user shall be able to load a previously saved game.

R10: During game play, the user shall be able to make successions of legal moves as described in the rules at <https://en.wikipedia.org/wiki/Nine_Men's_Morris> until a point at which the game has been won.

R11: During game play, the application must be able to detect when the game has been won.

R12: During game play, the result of the game must be displayed at all times.

In Table 1, each requirement is traced to the modules that it is implemented by.

Table 1: Trace of Requirements to Modules

|  |  |
| --- | --- |
| **Requirement** | **Modules** |
| R1 | Menu, MenuPane |
| R2 | GameModel |
| R3 | GameModel |
| R4 | GameModel |
| R5 | Menu, MenuPane |
| R6 | CustomGame, GameModel |
| R7 | CustomGame, |
| R8 | Game, GamePane, GameModel |
| R9 | Menu, MenuPane, GameModel |
| R10 | GameModel |
| R11 | GameModel |
| R12 | GamePane |

# Anticipated Changes

The following changes are likely to occur:

AC1: An additional mode of play will be added in which the user may play against the computer.

# Internal Review

The MVC architecture combined with the application of the information hiding principle were used in the development of the Six Men’s Morris application. The resulting design is highly flexible and can easily and efficiently handle changes or additions to the application requirements.

For example, the implementation of AC1 can be achieved entirely through the modification of the model since the view and controller are separated from the model (so long as no existing public interfaces in the model are altered).

# Test Plan/Report

## Test Plan

Testing for adherence to requirements will be done via manual testing. For each requirement, the game will be run in a way such that it can be determined if the requirement is met.

## Test Report

The results of the manual requirements testing are given in Table 2.

Table : Results of requirements testing

|  |  |
| --- | --- |
| **Requirement Tested** | **Result** |
| R1 | PASS |
| R2 | PASS |
| R3 | PASS |
| R4 | PASS |
| R5 | PASS |
| R6 | PASS |
| R7 | PASS |
| R8 | PASS |
| R9 | PASS |
| R10 | PASS |
| R11 | PASS |
| R12 | PASS |