SOFTWARE DEVELOPMENT 

OTHELLO PROGRAM

### PREPARED BY

Reinaldo Michael Hasian

Sept 09, 2019

### CBS Techno Corp

3-10-17 ITmeieki Bld.No.2, Meieki,

Nishi-ku Nagoya-shi, Aichi,

451-0045 Japan

# Table of Contents

1. Preliminary
2. Algorithm Design
3. Module Design
4. Man Hour

# 1. Preliminary

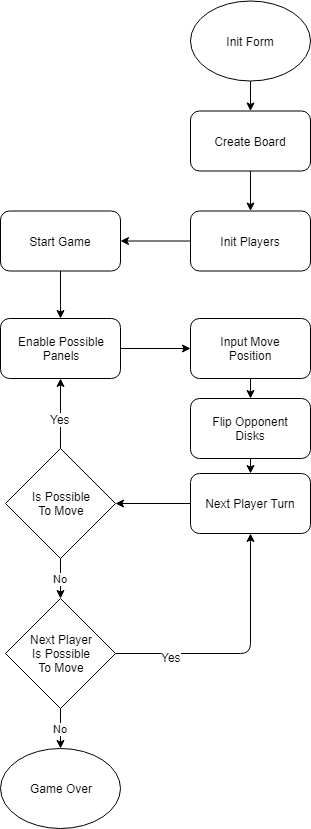
Othello or reversi is a [strategy](https://en.wikipedia.org/wiki/Abstract_strategy_game) [board game](https://en.wikipedia.org/wiki/Board_game) for two players, played on an 8×8 uncheckered board. There are sixty-four identical game pieces called *disks* (often spelled "discs"), which are light on one side and dark on the other. Players take turns placing disks on the board with their assigned color facing up. During a play, any disks of the opponent's color that are in a straight line and bounded by the disk just placed and another disk of the current player's color are turned over to the current player's color.

The object of the game is to have the majority of disks turned to display your color when the last playable empty square is filled.

This program is Othello board game implemented as windows form application with C# programming languange.

# 2. Algorithm Design

Below is flow chart of algorithm design that used to develop this Othello program.



Explanation for each process :

1. Init Form

Create windows form to be shown on screen.

1. Create Board

Add Othello board to windows from

1. Init Players

Create White Player and Black Player. Set their initial disks on board and set Black Player as first turn.

1. Start Game

Done initialization, execute the game logic.

1. Enable Possible Panels

Based on current Player turn, enable all panels that possible for player to put disk.

1. Input Move Position

Set chosen panel as Player’s panel.

1. Flip Opponent Disks

Flip all enemy disks that flanked by new disk from step 6.

1. Next Player Turn

Set current turn to the next player then check if that player can move or not.  
If that player can move then back to step 5. If that player can not move, program will check if previous player can move or not. If previous player can move, then current turn will be previous player. But if previous player also can not move then game is over.

1. Game Over

Calculate each player disk and decide who is the winner. The winner shown at message box with each player score.

# 3. Module Design

[Any technical obstacles like integration between different systems, as well as mitigation strategies]

# 4. Man Hour

[Any industry or market-related risks]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Milestone** | **Tasks** | **Reporting** | **Hrs** | **Date** |
| **1 - Design** | | | | |
| 1.1 | Analysis and design stage, gather data and create system mockup | None | 20 | 20/01/15 |
| 1.2 | Architecture design | None | 4 | 01/02/15 |
| 1.3 | Design work plan (distribution of tasks to development teams) | Client meeting to review work plan | 10 | 07/02/15 |
| **2 - Development** | | | | |
| 2.1 | Create database | None | 5 | 14/02/15 |
| 2.2 | Import existing client data | None | 5 | 21/02/15 |
| 2.3 | Clean data | None | 5 | 28/02/15 |
| 2.4 | Create GUI | Client meeting to review GUI | 30 | 01/04/15 |
| 2.5 | Integration with PaperlessOffice.net | None | 10 | 14/04/15 |
| 2.6 | Integration with smartphone network | Email report | 10 | 21/04/15 |
| **3 - Testing** | | | | |
| 3.1 | Alpha testing desktop application (Closed) | Email report | 25 | 07/05/15 |
| 3.2 | Alpha testing smartphone application (Closed) | None | 25 | 14/05/15 |
| 3.3 | Open Beta (volunteer employees) | Client meeting | 22 | 21/05/15 |
| 3.4 | Finalise documentation | None | 20 | 28/05/15 |