# Design Overview for Gibbous Tetris

Name: Trung Kien Nguyen

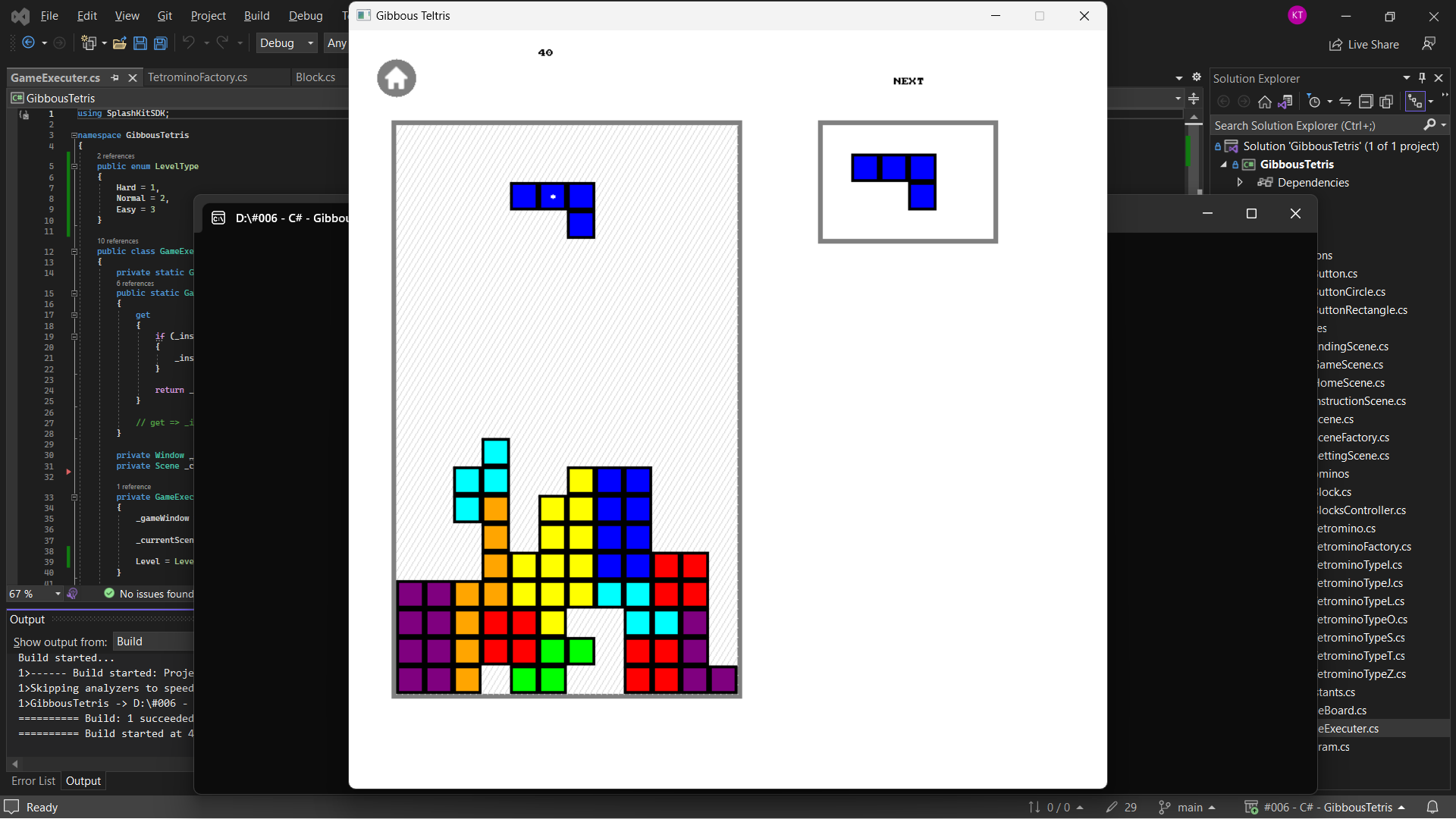
Student ID: 104053642

# Summary of Program

Describe what you want the program to do… one or two paragraphs. Include a sketch of sample output to illustrate your idea.

This program will basically provide a grid-based playing field where various types of Tetrimino, which is a shape composed of four blocks (squares), fall from the top of the screen. The player's objective is to move and rotate the falling Tetriminos to create complete horizontal lines without any gaps. When a line is completed, it disappears, and the player earns points. As the game progresses, the playing field is gradually filled, challenging the player's reflexes and strategic thinking. The game continues until the stacked Tetriminos reach the top of the grid, at which point the game ends. The program should display the player's current score and provide an option to restart the game for continuous play.

In addition, the program has a total of 5-6 scenes, including the home scene (for choosing the difficulty level) main gameplay scene, instruction scene (for how-to-play), setting the scene (for advanced settings like the sound manager), ending scene for the result, and replaying, and also an intro scene (if possible).



# Required Roles

Describe each of the classes, interfaces, and any enumerations you will create. Use a different table to describe each role you will have, using the following table templates.

Tables : <<class>> details – duplicate

Game Executer

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Instance | GameExecuter | Singleton |
| Execute() | void | Loop for execute the Splashkit program |
| Update() | void | For all related objects to Update |
| Draw() | void | For all related objects to Draw |
| ChangeScene() | void | Change between scenes |

Block

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| X, Y | double | Position in screen |
| Xindex, Yindex | double | Position in the 2D-array |
| Draw() | void | For all related objects to Draw |
| CanMoveDown | bool |  |
| CanMoveLeft | Bool |  |
| CanMoveRight | bool |  |

Tetromino: Has 4 blocks each

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| \_tetrominoColor | (SplashKit.)Color |  |
| \_tetrominoAngle | (Tetromino.)Angle |  |
| TheTetromino | List<Block> | 4 blocks in a tetromino |
| Update() | void | For all related objects to Update |
| Draw() | void | For all related objects to Draw |
| Rotate() | void |  |
| MoveLeft() | void |  |
| MoveRight() | void |  |
| MoveDown() | void |  |
| MoveToBottom() | void | Move until there are blocks below |
| RotateIndexesClockwise | Point2DIndex | Rotate the block's offset clockwise |
| GetIndexOffsets() | Point2DIndex[] |  |
| TetrominoOffsets() | Point2DIndex[] | Offset of a block is the distance from the center one to it (in indexes) |
| TetrominoIndexPoints() | Point2DIndex[] | With the offsets of the four blocks, and the ceter point (in indexes), I can calculate the coordinates of those blocks (in indexes) |
| CheckIfEmpty | bool | if space for blocks are empty in the case of the assumed Angle of the tetromino |

BlocksController: Control all the blocks that have been terminated as a 2D-array

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Instance | BlocksController | Singleton |
| Update() | void | For all related objects to Update |
| Draw() | void | For all related objects to Draw |
| BlocksIndex | int[, ] | List of terminated blocks’ position (in indexes) |
| BlocksDisplay | List<Block> | List of terminated blocks |

TetrominoFactory: Factory Method Pattern, to generate Tetromino

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| CreateTetromino() | Tetromino |  |

Table : <<enumeration name>> details

(Tetromino.)Angle: for the current angle/direction of the tetrominos

|  |  |
| --- | --- |
| Value | Notes |
| Up | = 0 |
| Right | = 1 |
| Left | = 2 |
| Down | = 3 |

LevelType: difficulty of the game (index represents the amount of time for each drop of the tetromino)

|  |  |
| --- | --- |
| Value | Notes |
| Hard | = 1 (second) |
| Normal | = 2 |
| Easy | = 3 |

Table 3: <<struct name>> details

Point2Dindex: Similar as SplashKit.Point2D, but in integers

|  |  |
| --- | --- |
| Value | Notes |
| Xindex | = 0 |
| YIndex | = 1 |

Table 4: <<inteface name>> details

IObserver: for implementing simple Observer pattern

|  |  |  |
| --- | --- | --- |
| Value | Type Details | Notes |
| OnNotify | void |  |

Table 5: <<abstract class name>> details

ObservableSubject: for implementing simple Observer pattern

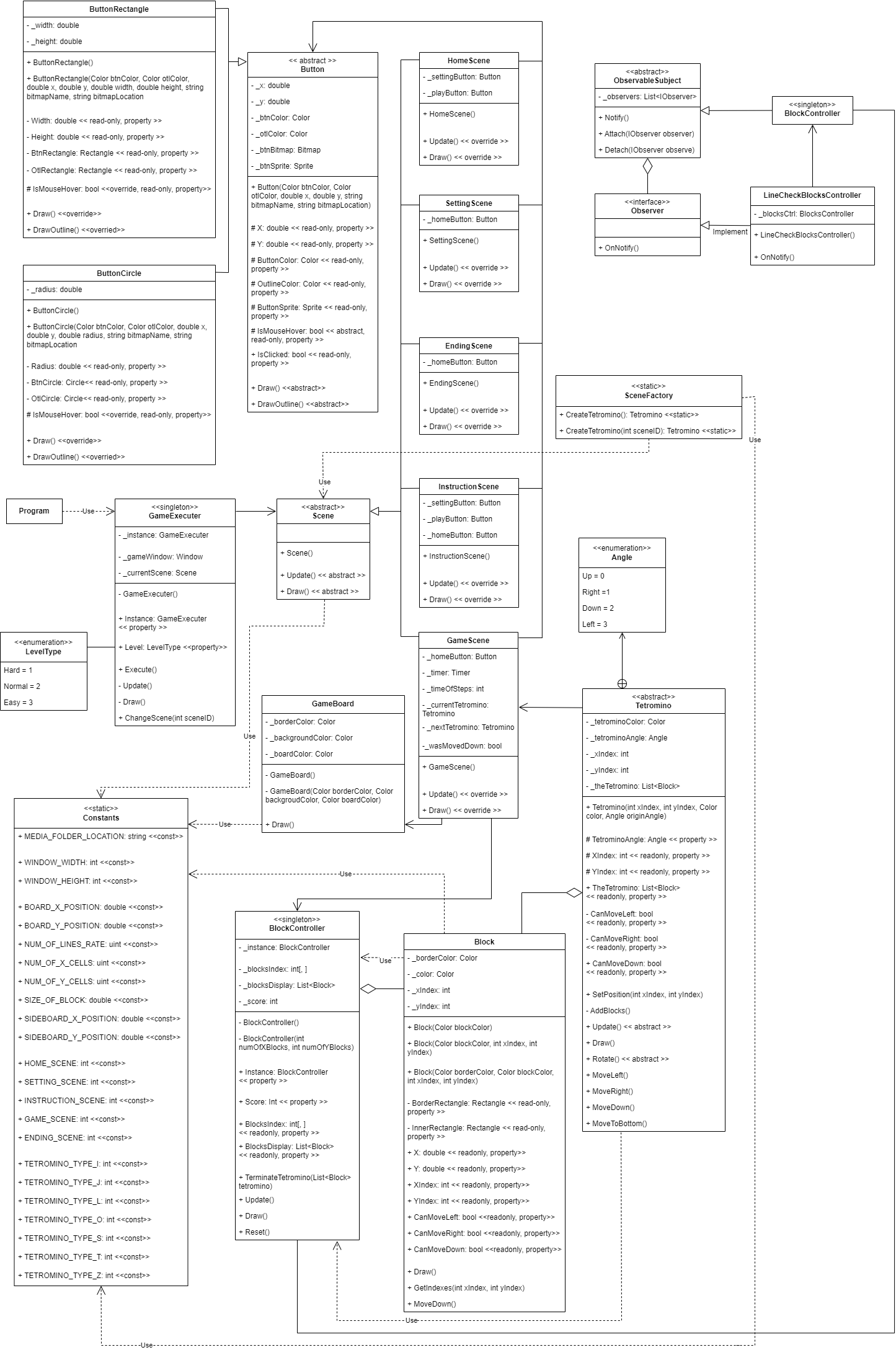
|  |  |  |
| --- | --- | --- |
| Value | Type Details | Notes |
| \_observers | List<IObserver> |  |
| Notify() | void |  |
| Attach() | void |  |
| Detach() | void |  |

Scene

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Update() | void | For all related objects to Update |
| Draw() | void | For all related objects to Draw |

# Class Diagram

Provide an initial design for your program in the form of a class diagram.



# Sequence Diagram

Provide a sequence diagram showing how your proposed classes will interact to achieve a specific piece of functionality in your program.

