# 1.0 SYSTEM ARCHITECTURE

## 1.1 Functional Properties

#### User Accounts

## 1.2 Non-Functional Properties

#### Performance

## 1.3 Component Diagram

# 2.0 DESIGN

## 2.1 Ball

text

### 2.1.1 Structure

#### void function()

### 2.1.2 Design and Rationale

## 2.2 Board

The Board class is a major functional component that models the game state and the layout of the GridItems in the MainActivity. A cuustom grid coordinate system is used so that the game would look consistent on different sized phone screens when drawn with BoardView, and collision detection is simplified. It also handles the collision interactions between the Ball and GridItem classes.

### 2.2.1 Structure

#### ArrayList<ArrayList<GridItem>> grid

A 2D array that holds GridItems (blanks, Bricks, and Jails). Its dimensions are defined in the init function. This array is used to help the Board locate and notify GridItems that are potentially hit by the Ball.

#### void init(Context)

Obtains the dimensions of the screen in pixels. It uses this information to calculate appropriate grid sizes that would create a 16:9 board that has square shaped GridItems. It also calculates the vertical offset required to vertically center the board on phone screens that do not follow the 16:9 screen ratio.

#### void initBoard(Context, FrameLayout)

Currently uses a BrickFactory to randomly create 10 SquareBricks on each player’s section of the board and adds them to the provided FrameLayout.

#### Board.Boundaries getBoardBoundaries()

Returns a Boundaries object which has the y-axis pixel location of the BoardTop, OpponentTop, PlayerTop, and BoardBottom. This is needed for the Ball to know where the top and bottom of the board are, and for the Paddle to know along which y-value (OpponentTop and PlayerTop) the Paddle should move horizontally on.

#### Boolean isHit(float, float, float)

Given the Ball’s position and size in pixels, isHit calculates which GridItems are potentially hit by the ball and notifies them that they may have been hit. If they have been hit, the function will return True, and the Ball will reverse its direction.

#### float getGridItemSize()

Returns the GridItem size as calculated during the initialization of the Board object. Since the GridItem is square, its width and height are the same.

#### float getNumColumns(), float getNumRows()

Returns the number of columns and rows (respectively) on the Board.

### 2.2.2 Design & Rationale

The Board is a singleton because there should only be one game state. This prevents duplication. All of its parameters are private and only retrievable via getters, which makes it unmodifiable by other classes. Other classes access the getters via the Helper class, so that they do not have to have a copy of the Board object.

A fixed number of columns and rows have been carefully chosen so that each GridItem is a decently sized square and the Board follows a 16:9 ratio, which allows for a full-screen experience for most phones. In the case that the phone doesn’t have a 16:9 aspect ratio, vertical gaps will be added to ensure that the grid items remain square and the board is vertically centered.

Also, with our custom grid coordinate system, we can easily translate a Ball’s pixel position to a grid coordinate, and notify the GridItem in that grid coordinate that it may have been hit. In the worst case, only 4 GridItems will be notified of a potential hit. When communicating between phones of different screen sizes about the positions of objects, the grid coordinate makes it easy to calculate where the object should be drawn on their own screens.

The Board sections/background are drawn by the BoardView’s only function, onDraw. The reason for separating the BoardView from the Board is because view objects can be destroyed on the end of an activity. To ensure the state of the game and board can persist across different activities via Intent objects, we separated the model and the view of the board into those two classes.

To allow users to pause the game or to build the board, an enum will be added to the Board class to denote the state of the game, which will let the MainActivity change its behavior.

## 2.3 GridItems

The GridItems class is the parent of Jail (will be implemented) and Brick classes. Polymorphism allows the Board class to treat all GridItems the same, while each GridItem may handle the events differently. This decreases logical cohesion. A GridItem object that is neither a Jail nor a Brick is a blank spot, therefore, its onHit function is an empty one.

### 2.3.1 Structure

#### int row, int column

The grid coordinate position of the GridItem.

#### boolean onHit(ArrayList<int[]>)

An empty function, so that children are enforced to implement this function, and so that the Board can treat all GridItems the same. It accepts an array list of the pixel positions of the Ball’s bounding box’s 4 corners.

#### int[] getPosition()

Returns the x and y position of the GridItem, according to our custom grid coordinate system.

## 2.4 Brick (and its children classes)

The Brick class is an abstract one that has five children classes: Jail, Square, LeftUpperTriangle, LeftLowerTriangle, RightUpperTriangle, and RightLowerTriangle. The separation was required since they all handle onHit and onDraw differently. Brick objects inherit all parameters from GridItems.

### 2.4.1 Structure

#### int hp, int opacity

Depending on the number of times the brick has been hit, opacity will be zero so that it doesn’t show up on the board anymore. When the hp is 0, onHit will always return false, so the Ball doesn’t reverse direction, and the score in Board is not changed.

#### boolean onHit(ArrayList<int[]>)

#### Each type of Brick has a different shape, so they determine if they have been hit by the Ball differently. Even though the Ball may be in the same grid coordinate as them, the Ball may have missed them.

#### void onDraw(Canvas)

Draws the brick using shaders to ensure a gradient.

## 2.5 BrickFactory

Makes use of the Factory Method design pattern. It can create different types of Bricks, depending on the BrickType enum passed to it.

### 2.5.1 Structure

#### BrickFactory(Context, BrickType, float, float)

Given the BrickType, Context, and x-y positions, the factory returns a Brick of the specified BrickType.

## 2.6 Paddle

text

### 2.6.1 Structure

#### void function()

## 2.7 Helper

text

### 2.7.1 Structure

#### void function()

## 2.8 UML Diagram

# 3.0 General Coding Roles

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Karan Bhandari | Adil Mian | Zeyad Abdulghani | Eric Luo |  |  |  |  |
| Brick(s), BrickFactory | Ball, Helper | Paddle  Connecting two phones | Board, BoardView, GridItem  Image resource files, mockups |  |  |  |  |

Please feel free to view open and closed issues/pull requests on our Git repository to see past and ongoing progress.