Arza

Design Document

Team 8
Ayush Vijaywargi
Noah Whaley
Eddie Gillott
Albert Huang
Brandon Xia
Ongun Savas

Table of Contents:

1. Purpose …………………………………………………………………………………...… 3
2. General Priorities…………………………………………………………………...…..… 5
3. Design Outline…………………………………………………………...…....…. 6
4. Design Issues……………………………………………………………………...…… 9
5. Design Details……………………………………………………………………….… 11
6. Mock Ups…………………………………………………………………………….… 14

Purpose:

There is a high demand for puzzle games on the Google play app store, and the purpose of our app is to fill in a niche in that market. We are going to make a clone of the classic puzzle game Puyo Puyo. There is currently no official version of the original game available in English for Android. This game is also unique, as it's a two player game where you are competing against the clock as well as your opponent. We will also put our own twist on the classic formula in the form of alternate game modes. Because of Puyo Puyo's simplicity, uniqueness, and two player element, we think a game similar to it will be a good fit for the Android platform.

Functional Requirements

- As a developer,
  - I need to implement a playing area where blocks land and stack on top of each other, and fall down if there is no block beneath it.
  - I need 2x1 pieces to fall down on their own, land, and then repeat the cycle with another piece of a different colors. The sequence of colored blocks must be the same for both players.
  - I need groups of four of the same color to be cleared. When this happens, garbage must be sent to the opponent in a queue, to be released once their next block lands. If the receiver clears a group when said block lands, it should negate the garbage by the same amount.
  - I need to implement a lose condition once a player's playing area is full.
- As a user,
  - I want to be able to move and rotate my pieces using the touch screen.
  - I want a guideline that shows where my piece will land if I swipe down.
  - I want to be able to see the next piece I am about to receive.
  - I want a summary screen when the match is over with various statistics along with the choice to quit or play again.
  - I want to play with other human opponents.
  - I want to view my opponent's playing area while in a game.
  - I want to play a single player game.
  - I want a straightforward UI for creating and joining games.
  - I would like a timer somewhere to see how long the match has been going.
- As a user,
  - I would like extra game modes that put a twist on the base game.
  - I want to be able to play against a computer opponent
  - I would like to automatically find people to start a game with.
  - I want to be able to save other players I meet online to a friends list so I can play with them again later.
  - I want to make an account that saves my statistics and friends list

    - I want to compare play statistics with my friends.

Non-Functional Requirements

- As a developer,
    - I would like to build a system that is scalable, so it is easy to add extra features in the future.
    - I would like to optimize game so that it doesn't consume a lot of battery life.
- As a user,
    - I want smooth, lagless gameplay and responsive controls.
    - I want a visually appealing backdrop
    - I want music to play while playing, and for it to respond to the gameplay
    - I want to see cool effects when I clear blocks?
    - I would like my account to be secure.
    - I would like the interface to be easy to understand and easy to use.
    - I want to be able to choose different backgrounds and music.

General Priorities

1. User Interface - Designing a user interface that is user friendly is one of the most important things in game design. Simplicity and ease of use would be our first consideration. So our game should be easy to use, engaging, should be great looking and responsive for all mobile devices. It should also have an interactive guide on how to play the game.

2. Performance - Optimized software will lead to gameplay without any crashes and lags. In order to achieve best performance, we need to use threads effectively that is doing networking calls on one thread and showing UI on other thread. We also have to choose better game development frameworks and services, so that game can be enjoyed on phones having low end hardware as well.

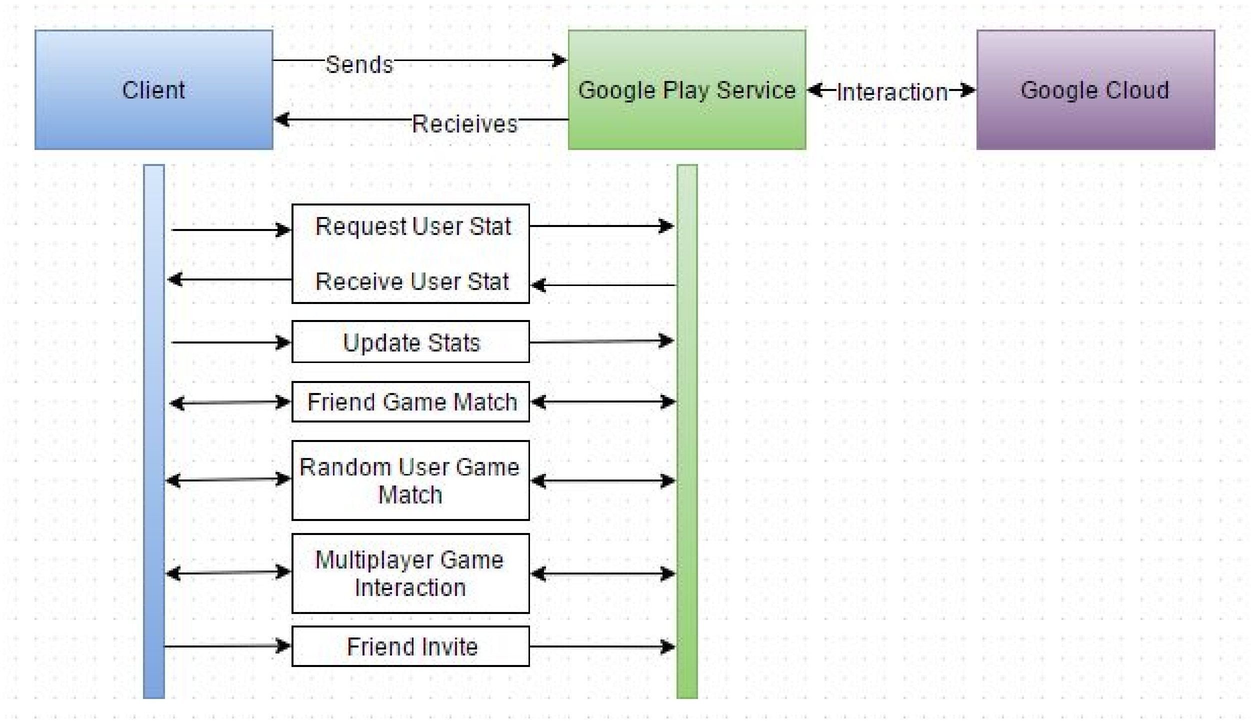
3. Gameplay - We are taking all of our gameplay inspiration from previous puzzle games. However we still want to make sure we have some of our own ideas through extra game modes. We won’t modify the original game mode that has been proven to be fun already, rather we will experiment with new ideas in extra game modes.

4. AI for Game - Good AI would be something that puts a good fight with user and in the end loses. So we need to make our AI keeping different types of gamers in mind - casual and competitive gamers. Hence we need an effective algorithm to decide where the blocks should be placed in the playing space automatically.

5. Visual style - Since no one on the team is an artist our main priority with the visuals will be to represent the information needed in a way that makes the most sense. In order to do this we will utilize open source art assets for Arza. So while we might not have the most unique looking game, it will still be functional.

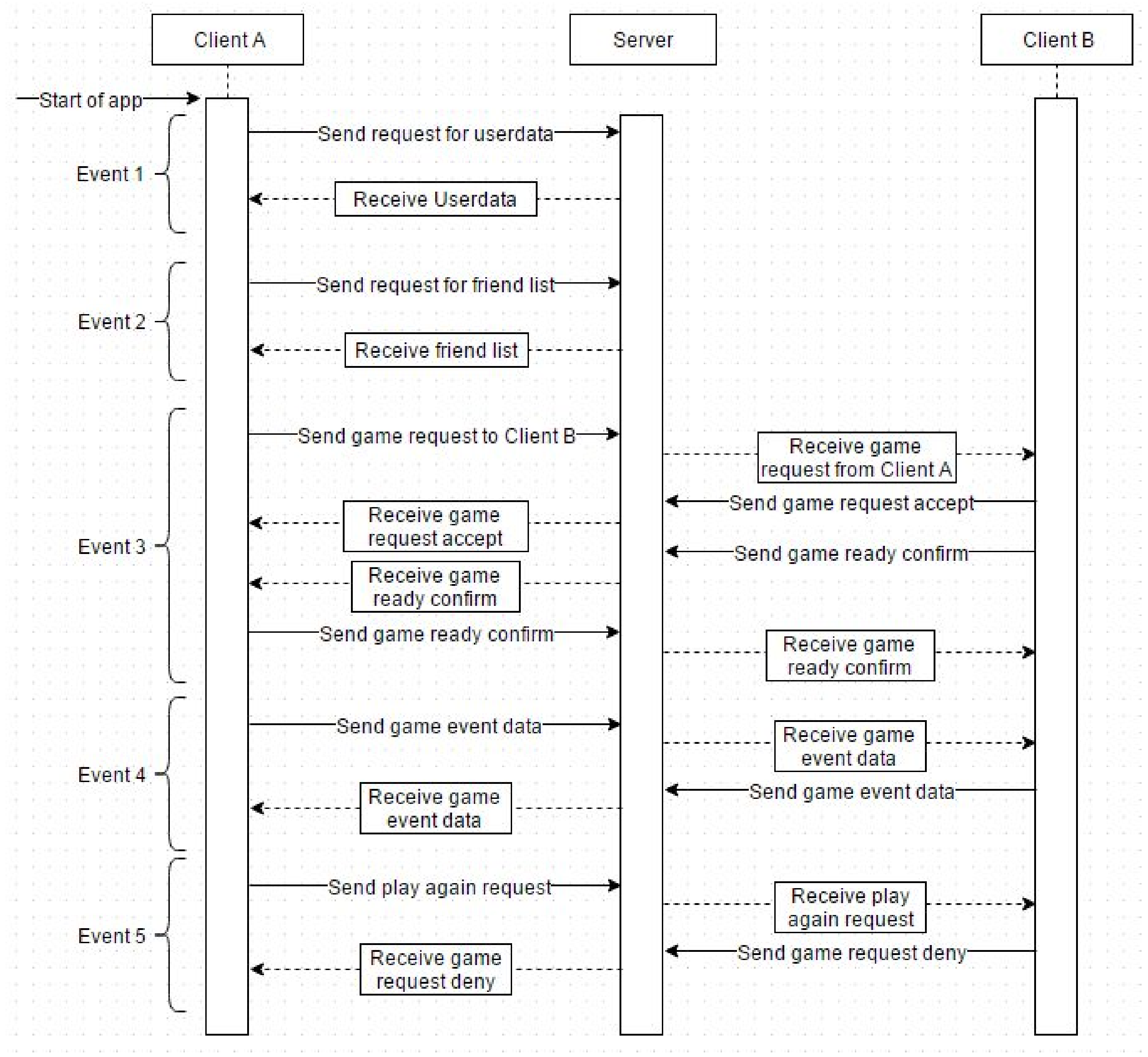
Design Outline:

Our project is a puzzle game app on android. This application will follow the Client-Server model where all online users will connect to the server. The app itself will follow the Model View Controller organization method. The following shows our model and the interactions between the client and server.



- Client - Android app
  - This is where the user would interact with the game.
  - The above diagram shows a list of network interactions the app would require.
- Server - Google Play Service
  - Decision and reasoning in Design Issue #1.
  - In order to play online, the user will connect to Google Play using their google account.
  - A user's game statistics and friends list will be stored here.
  - We will implement the user profile, friends, and multiplayer feature through it.

This figure shows a typical sequence of network events for a typical user.



In the figure, Client A and B are both users, but only relevant network activities from Client B is shown.

Event 1: Client A is trying to look at his profile. A request is sent for it, data is received, then information is shown in the app.

Event 2: Client A is trying to start a multiplayer game. A request for his friends list is send, data is received, friends list is shown on screen for Client A to make a selection.

Event 3: Client A has selected Client B to play with. A game request is sent to Client B, note that all activities goes through the server and not directly between users. Client B receives a request and sends an accept message. Before the game starts both Clients must tap a ready button, the app will start the game at the same time when both Clients are ready.
Event 4: This shows a sample of network activities during game play, information will be exchanged every time either player drops a block in order to keep the games in sync. Game event data will be sent in a random order determined by how fast the players drops blocks, not necessarily in the order shown here.
Event 5: The game ended, Client A would like to have a rematch. A play again request is send, and in this case a request denied is received, but in the case where Client B accepts it would go back to the second part of event 3 where both Clients taps ready before starting.

Design Issue

Design Issue #1 - How should we implement our back-end?

- Option 1: Google Play Service
- Option 2: Custom server
- Decision and reasoning: We choose option 1. Although a custom server gives extra flexibility, it creates more work and takes one or two people off the team for building the main game. After researching, we found out that Google Play Service will cover all our requirements, including all players' stats, friends list, and also multiplayer networking.

Design Issue #2 - Should the display be oriented portrait or landscape style for multiplayer?

- Option 1: Portrait
- Option 2: Landscape
- Decision and reasoning: We choose option 1. Playing in landscape mode would be counter-intuitive because only one part of the screen will be used and the usable area will be less as most of the screen will be covered by the user's' thumbs. We have designed the main game view so that single player and multiplayer mode will have minimal differences for usability. We were able to fit the extra views multiplayer mode required in portrait mode.

Design Issue #3 - Should users in multiplayer mode be able to customize the game settings?

- Option 1: Yes
- Option 2: No
- Decision and reasoning: We choose option 2. We first thought that a multiplayer game could be customized by the two parties in terms of gameplay to alter the difficulties and other aspects, but after consideration there was no options that made sense in multiplayer mode.

Design Issue #4 - Should there be achievements?

- Option 1: Yes
- Option 2: No
- Decision and reasoning: We choose option 1.We decided that adding achievements into the game would not take much effort and it will keep some gamers entertained. It could possibly be used to create a general summary of a player's skills on their profile and it could entertain gamers that enjoys working toward achievements.

Design Issue #5 - What factors should take priority in finding an opponent during random matchmaking (if any)?

- Option 1: Connect people with similar skills
- Option 2: Connect people based on connection
- Option 3: A combination of the two

- Decision and reasoning: We choose option 3. We want users to play against other users of similar skill level so both will enjoy the game. Connection may or may not be an issue for us. We will try to minimize network transfers between the users during game.

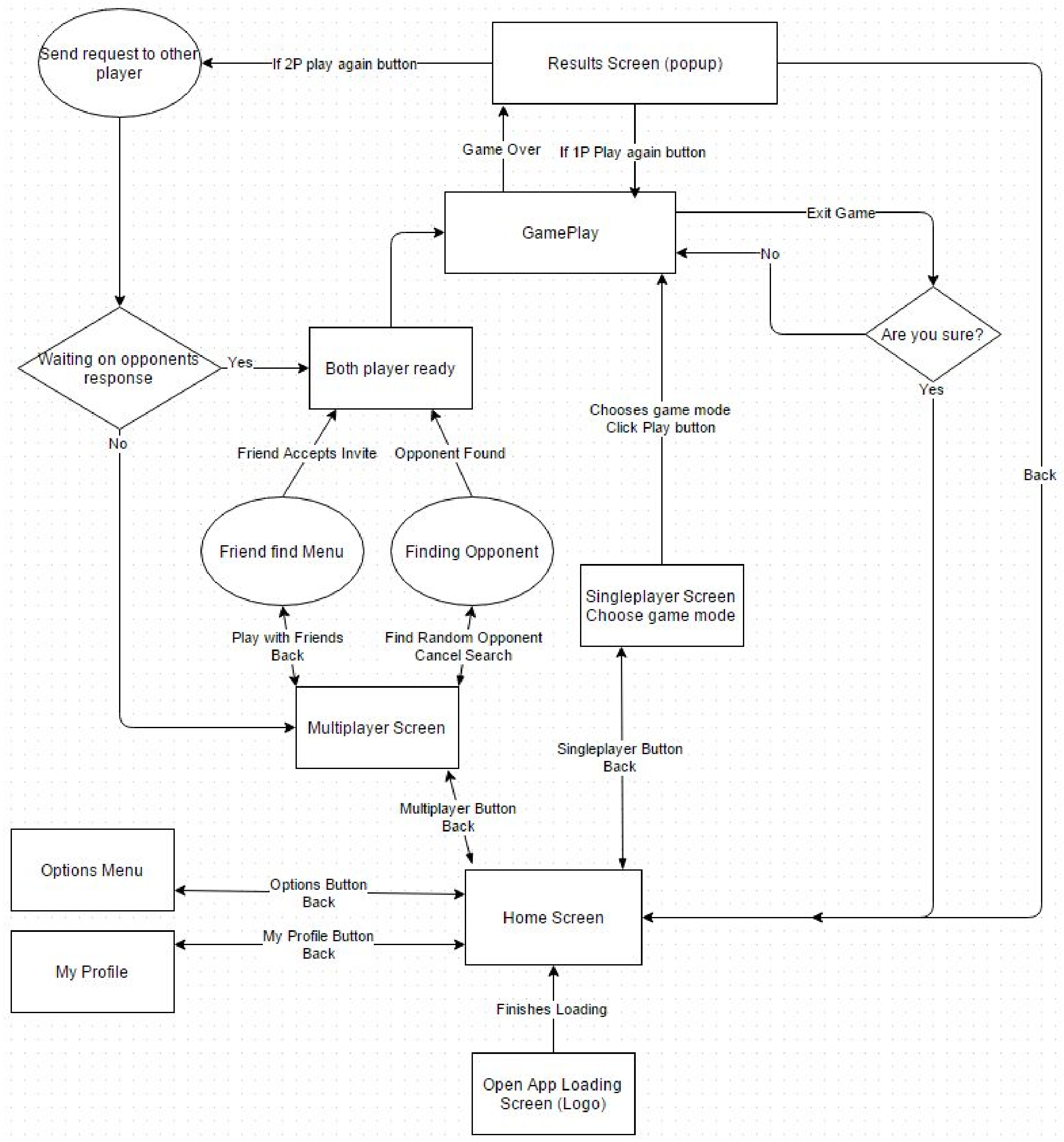
Design Issue #6 - Shall we give users option to play against their friends?
- Option 1: Yes
- Option 2: No
- Decision and reasoning: We choose option 1. Though adding this feature adds to complexity of application programmatically and usability from creating more steps to set up a game. A user would have to choose a friend to play with, send a request, wait for friend to reply, confirm that both parties are ready, then finally will the game start. We believe that this feature will add a great value to user experience and is heavily valued by users, therefore should be necessary.

Design Issue #7 - Should we include the "ready" button when using matchmaking?
- Option 1: Yes
- Option 2: No
- Decision and reasoning: We choose option 1. The sequence of events leading up to a multiplayer game will be choose method of matching with users, wait, match found. Because of the wait before the match is found the user might not be 100% focused on the game, so it would be good to implement a ready button that both parties will press before the game starts.

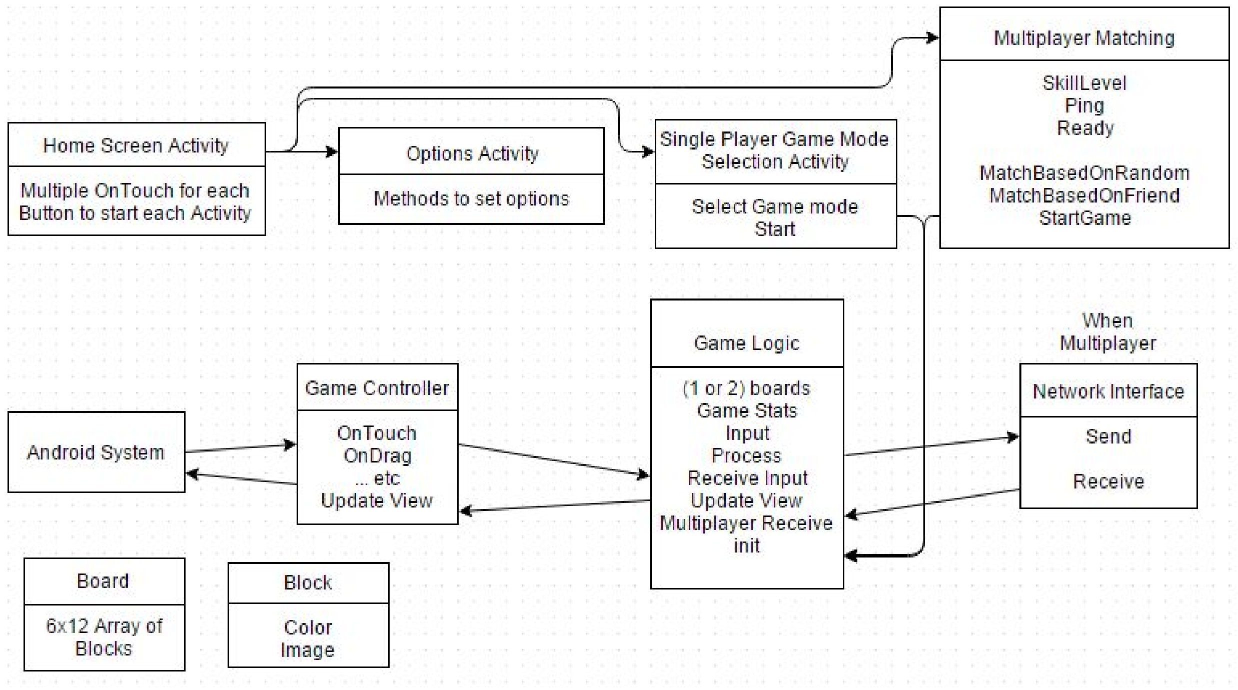
Design Issue #8 - Shall we have interactive guide which guides user through various steps on how to play the game or just provide instruction manual.
- Option 1: Interactive guide
- Option 2: Instruction manual
- Option 3: Both
- Decision and reasoning: We choose option 1. Having an interactive tutorial is fun. It is also the best and fastest way for a new user to learn how to play the game. If someone were to forget how to play, he or she could easily play through the tutorial again.

Design Details

This is the state diagram of our app, showing how each action and view transitions.



This figure shows the main classes of the android client.

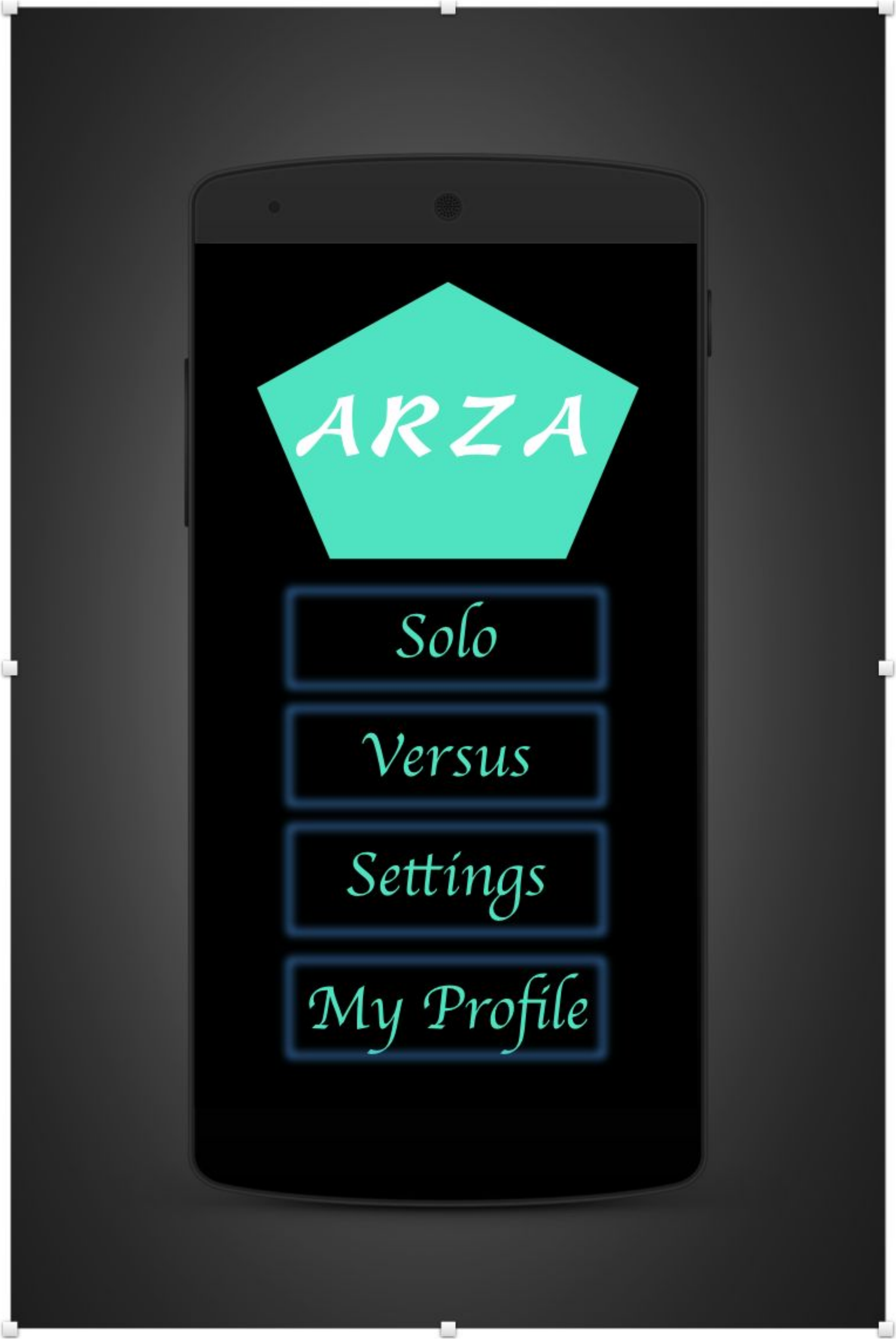


- Home Screen Activity
  - Selection to start Options, SinglePlayer, Multiplayer, and Profile Activity.
- Options Activity
  - Interface to change game settings.
- Single Player Game Mode Selection Activity
  - Selection for game mode.
- Multiplayer Matching Activity
  - Finding a match based on random player or friends list.
  - Game ready confirmation.
- Game Logic
  - Main part of the game.
  - Keeps a record of all data for the current game.
  - Receives player input.
  - Sends and receives game events in multiplayer games.
  - Processes game 60 times a second.
  - Sends out view update.
- Game Controller
  - Controls input and output.
  - Receives input from the Android system and sends it to Game Logic class.
  - Receives update information from Game Logic and determine how the views should be updated.
- Network Interface
  - Transfers game events between two players in a multiplayer game.
  - running on a separate thread.

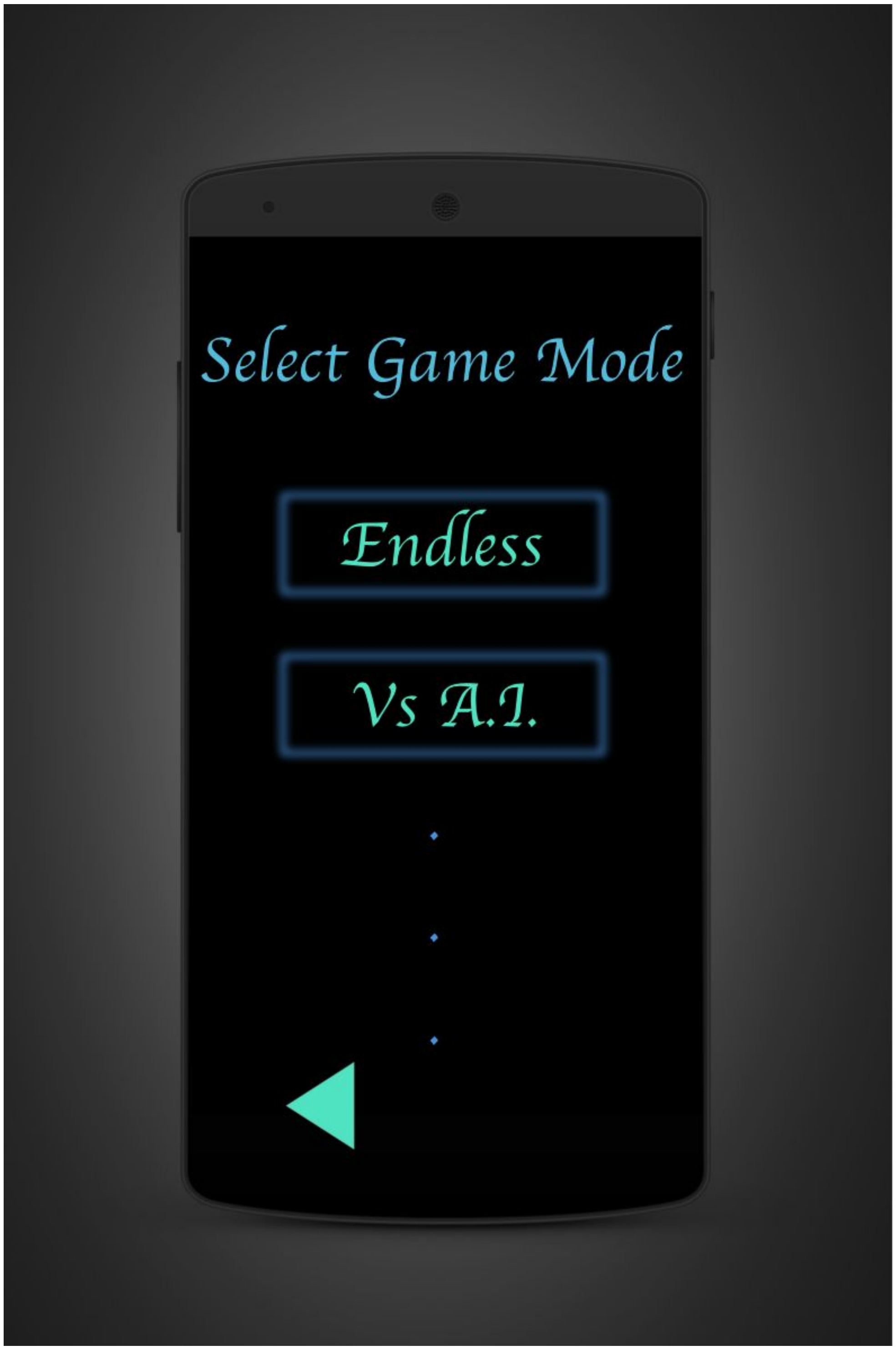
- Board
    - Contains information on the current state of the game board.
- Block
    - Contains information on a certain block.

UI Mock Ups

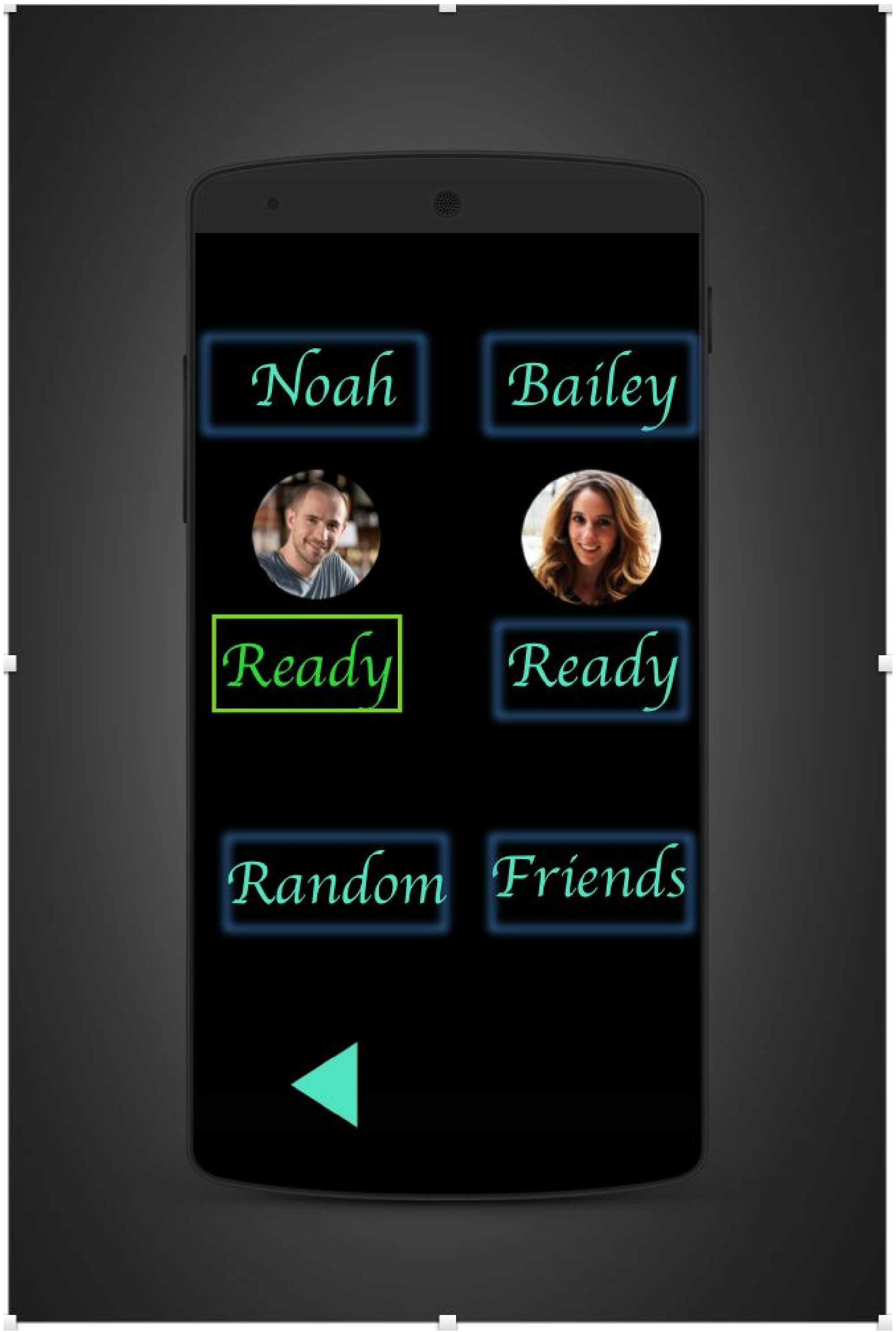
Home Screen



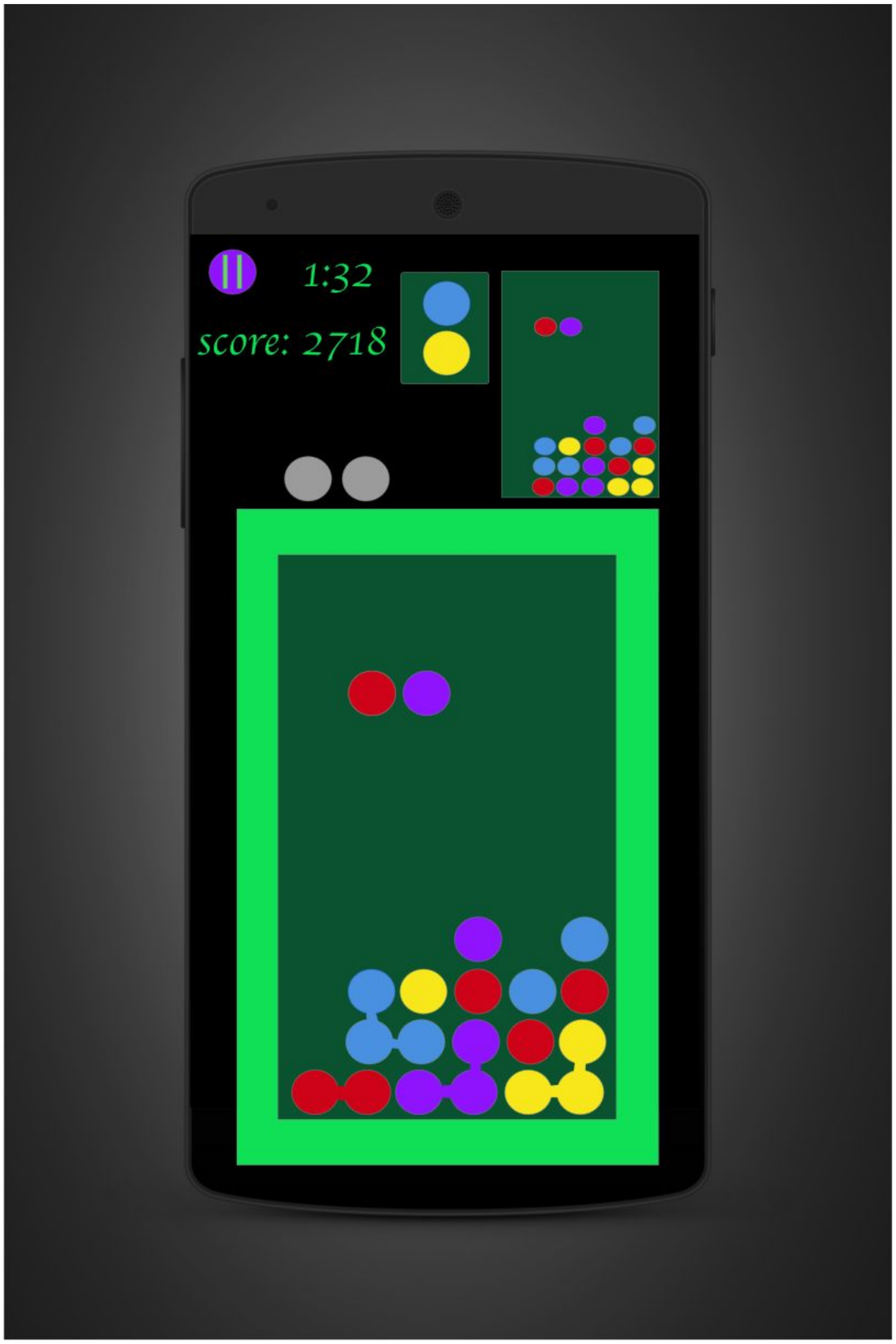
Single player game mode selection screen



Multiplayer opponent finding screen



Multiplayer in game screen.


1:32
score: 2718