Description

Intended User

Features

User Interface Mocks

Screen 1

Screen 2

Screen 3

Screen 4

Screen 5

Key Considerations

How will your app handle data persistence?

Describe any corner cases in the UX.

Describe any libraries you'll be using and share your reasoning for including them.

Describe how you will implement Google Play Services.

Next Steps: Required Tasks

Task 1: Project Setup (Already done)

Task 2: Implement UI for Each Activity and Fragment

Task 3: Join game UI events.

Task 4: Color picking.

Task 5: Improving user experience with multiplayer session.

GitHub Username: M187

TheGame

Description

Simple shooting game.
Contains local multiplayer game.

Intended User

All gamers who want to play simple game to relax.

Features

User can play single-player game with generated levels.
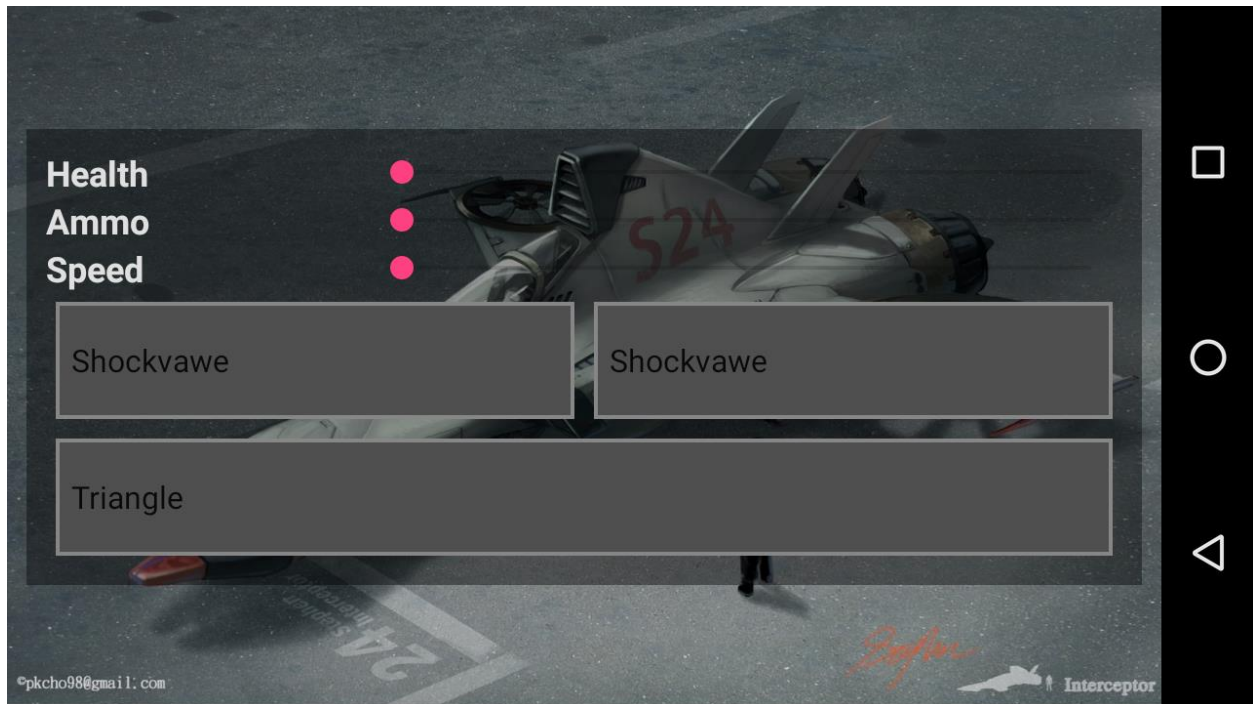User can play multiplayer game on local network.

User Interface Mocks

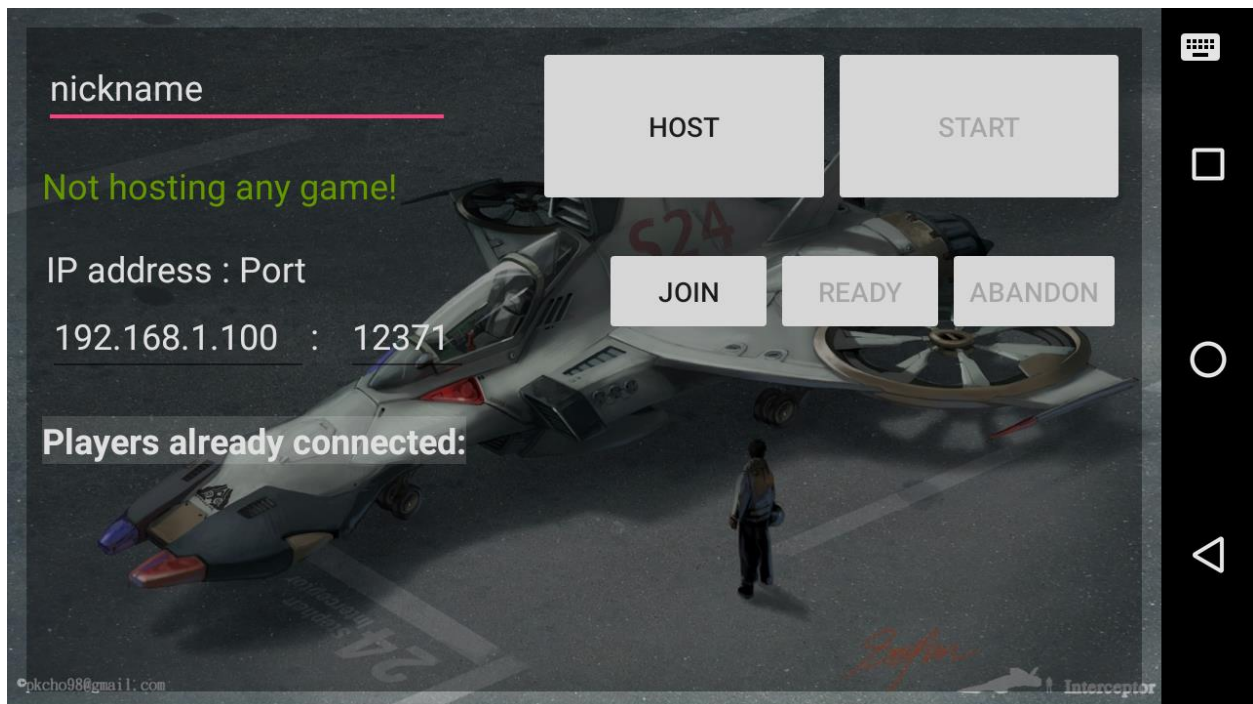### Screen 1

Main screen of a game

Screen 2

Option screen of a game.

Player can set up health, speed and ammo bonuses. Choose actions for buttons and sprite for a ship.

Screen 3

Multiplayer lobby.

Player can host game or join game.
Hosting game is automatically performed on port 12371.
To Join game, you need to know IP of an hosting device. Port should not be changed and maybe this input test field will be removed.
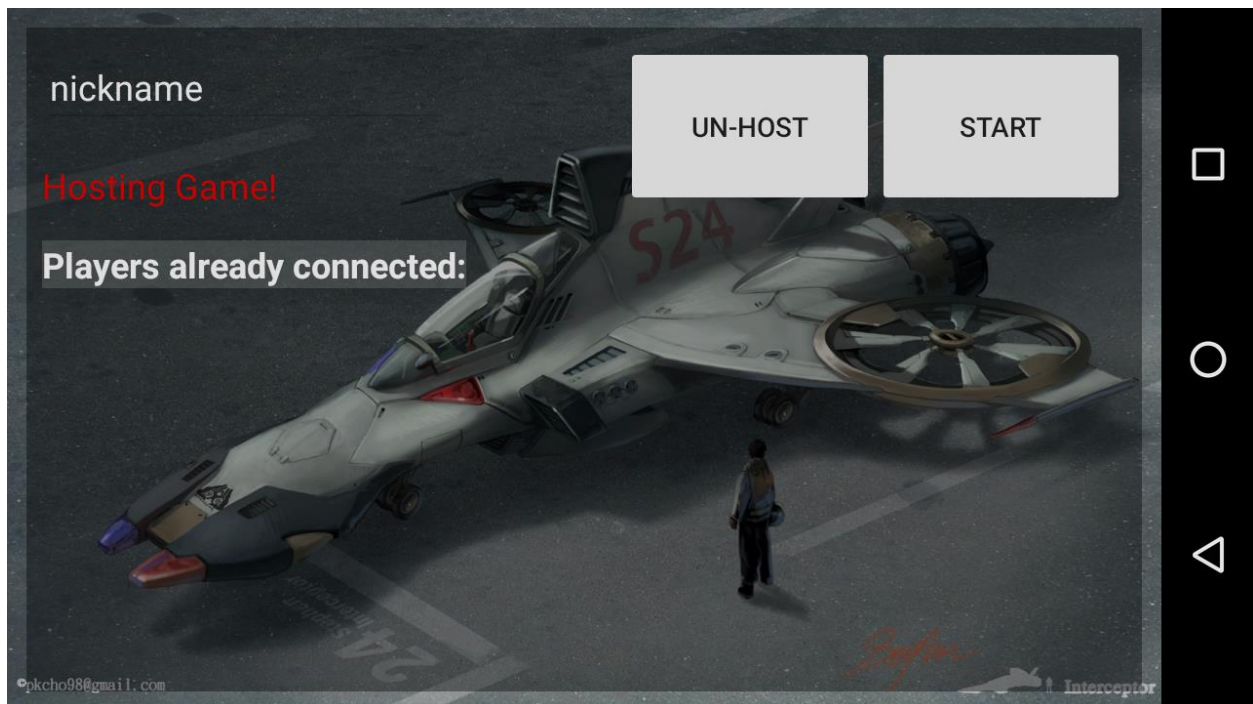
For host to be able to start game, each joined player must click ready button.

Future considerations:
Improve UI when join game is clicked. – Host buttons are not disappearing now, think of something….
Add color picker to pick color, and implement color messages/logic into game.

Screen 4

When hosting game, join row will disappear.
Nickname not editable. (Though this can be changed)

Screen 5

Actual gameplay screenshot.

Ui can be observed. Two joystick – for moving and shooting are present. 2 buttons to execute defined skills are present on the left side.

Two touch is supported. 3 fingers will not work.

Status bar is on the bottom, with Health and Ammo indicator.

Main base is the circle in the middle.

Player ship is red triangle.

In single-player, there are nests on the map that spawn enemies periodically and they engage player.

Future considerations:
Use native android functionality to draw this Interface.

Key Considerations

How will your app handle data persistence?

So far only Preferences are used to store player options.

Describe any corner cases in the UX.

Each new screen spawn new activity.

When back button is pressed, Activity is closed.
For multiplayer instances, when back button is pressed, Threads handling networking are terminated also.

Describe any libraries you'll be using and share your reasoning for including them.

Butterknife – I like it ☺

Describe how you will implement Google Play Services.

I will probably not. Don't have idea how to use them in this game for now.

Next Steps: Required Tasks

Task 1: Project Setup (Already done)

- Implement main game thread.
- Implement GameObjects.
    1. Players
    2. Enemies
    3. Spells
    4. Obstacles
    5. Collectibles
- Implement SAT collision detection (2d).
- Implement C# pathfinding.
- Implement in-game UI.
- Implement game menu screens.
- Implement networking.
- Implement map generation.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Build UI for PlayerOptions
- Build UI for MultiplayerLobbyActivity
- Build UI for GameActivity

Task 3: Join game UI events.

Improve Join game UI in MultiplayerLobby.
- Make join row disappear and change actions/text of Host / Start buttons to Abandon/Ready.
- If clicked abandon, put UI into starting state.

Task 4: Color picking.

Add color-picker to multiplayer lobby, so that player can pick color. This color will be later used to distinguish between players.
- Add color-picker.
- Add network message for picking color.
- Add network message for host to inform other players about color.
- Add member to Player class to hold color information.

Task 5: Improving user experience with multiplayer session.

Debug multiplayer session.
- Finish end game events for Victor.
- Try out with more than 2 players.
- Handle lost packets/messages.
- Add timeout for non-responsive player.
- Add logic to drop this player.
- Debug… massive debug…
- Debug… massive debug…
- Debug… massive debug…