Project Report

Mobile Computing - 2013/04

Course: MEIC

Campus: Alameda

Group: 1

Name: Daniel Bali Number: 79534 E-mail: janos.bali@tecnico.ulisboa.pt

Name: Gayana Chandrasekara Number: 79529 E-mail: gayana.withanage@ist.utl.pt

Name: Seçkin Savaşçı Number: 79503 E-mail: seckin.savasci@tecnico.ulisboa.pt

(PAGE LIMIT: 5 pages – including the cover)

1. Achievements

Feature	**Implemented (Fully / Partially / Not implemented)?**
Game scene	Fully Implemented.
Movement and life cycle of players	Fully Implemented.
Bomb drop off and explosion	Fully Implemented.
Movement and life cycle of robots	Fully Implemented.
Collision detection	Fully Implemented.
Score and game duration	Fully Implemented
Pausing / resuming the game	Fully Implemented.
Handling of relevant activity lifecycle events (e.g., pressing home button)	Fully Implemented.
Level selection	Not implemented. Players need to complete the current level to advance the next one
Multiplayer support	Partially Implemented
Clients leaving / joining the game	Not Implemented
Server hand-over	Not Implemented
Group merging (or group spitting)	Not Implemented
Extras : OpenGL graphics Custom game framework	Fully Implemented

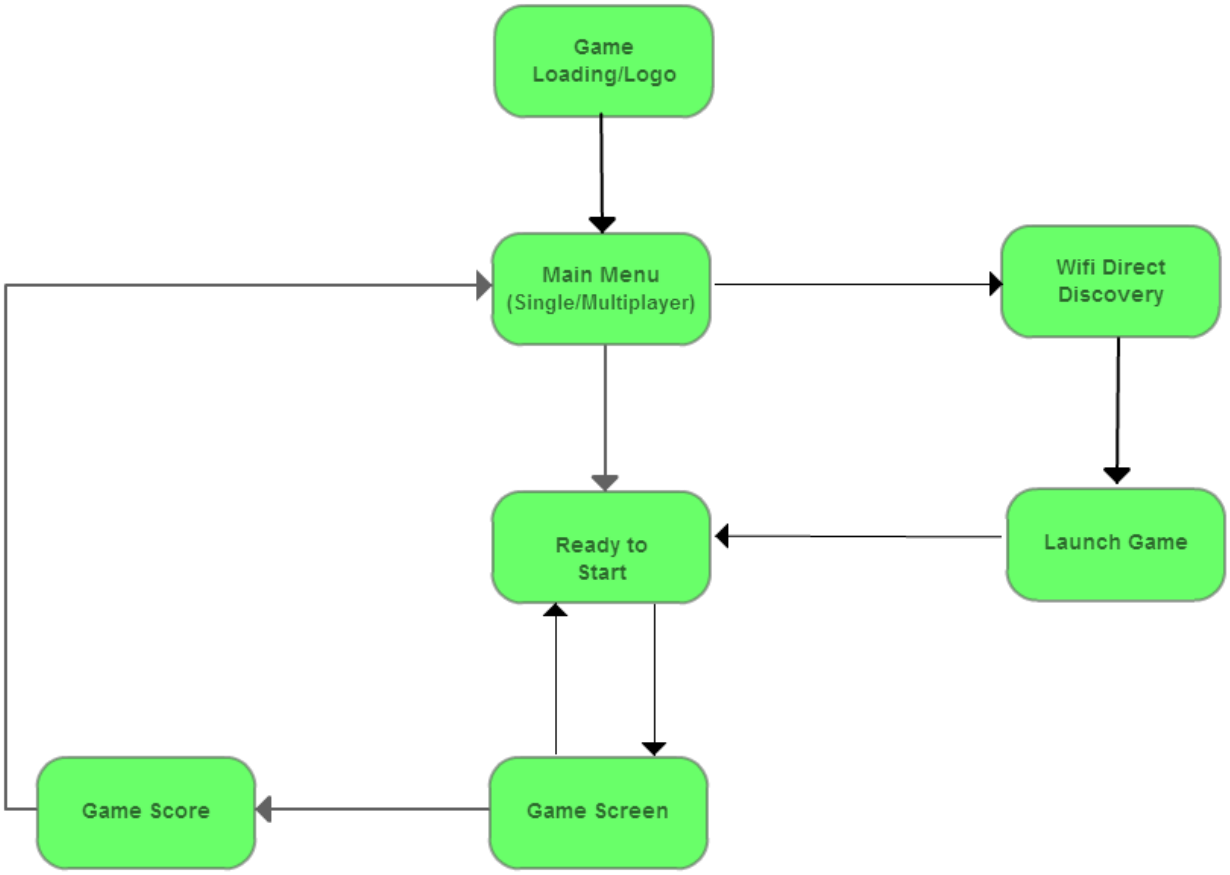
2. Specification

In this project, we have implemented a multiplayer bomberman clone which runs on a custom game framework we wrote during this project. We aim to achieve a solid and smooth gaming experience which led to implement our own game framework and use OpenGL for game graphics.

In our current implementation, we have four activities. SplashScreen activity which is the entry point of the application, responsible for loading and initializing general game resources. Main Menu is the activity where users chooses the game mode (single player/multiplayer) and also inputs the player name. If user selects the single player mode it directly loads the Ready to Start game view, but otherwise user will be redirected to the Wifi Direct discovery activity where the players who are willing to join will be negotiated and create a group with a group

owner (GO). Once the initial negotiations are over user will navigated to the Launch Game view to start the game. When a player is ready to play, Game Screen will be loaded which is the active game view where users interact most.

We have integrated the game levels so that when a user finishes a level he will automatically move to the next level and Ready to Start view will be displayed. This iteration will occur until the user reaches the level 3 which is the final level and once it is completed, Game score view will be displayed. This Game score view would also be displayed when a player failed to stay in the game board. Finally the main menu view will be loaded completing the whole life cycle of our game control flow.



2. Design

We have implemented our project in three phases as provided in the specification.

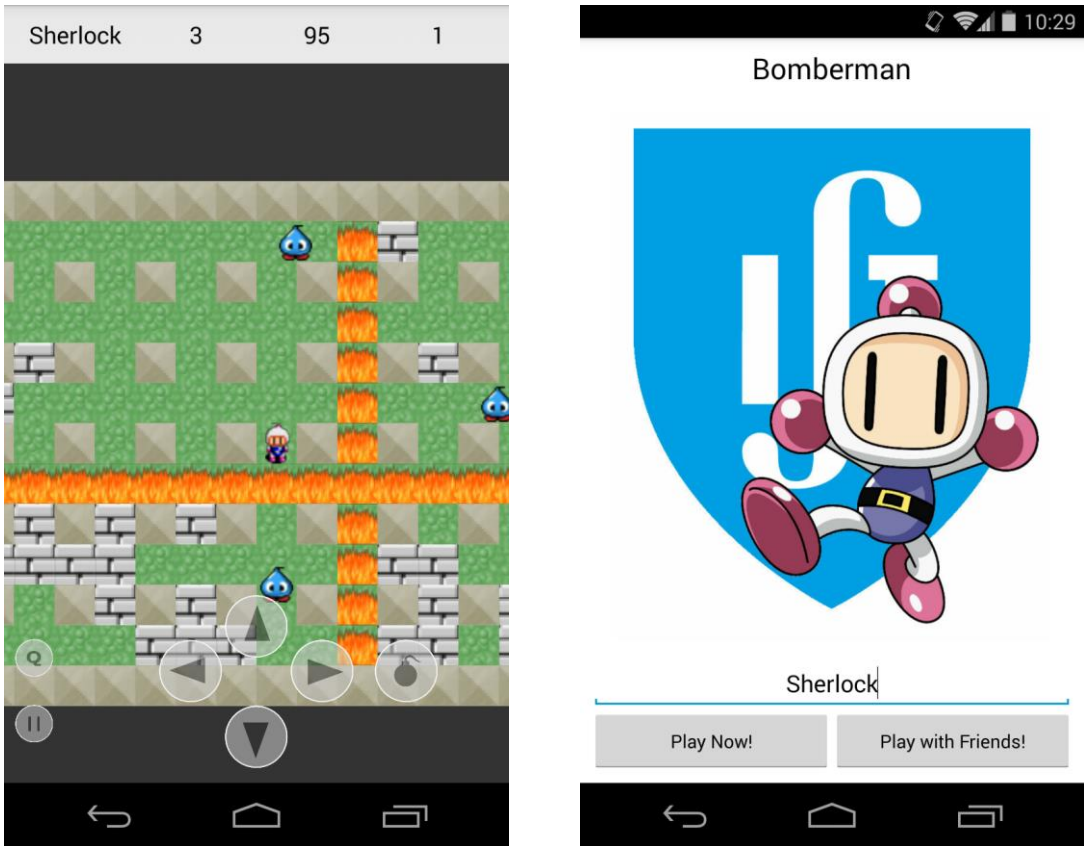
Our custom framework introduces “Screens” which are sub activities tailored for game development. Bombing activity contains several screens which are responsible for game logic and presentation flow. We have one main thread, one UI thread which is responsible for drawing and updating user interface, and another thread which is responsible for OpenGL and also network threads if we play multiplayer. Except from activity and screen transitions, our core game logic can be described as follows:

while True:
    call current screen's update method
    call current screen's present method

In this scenario, the screen's update method is responsible for updating the game state and handling game related inputs (Game unrelated inputs are handled by the encapsulating activity). The screen's present method draws the current game world using OpenGL.

In our first iteration (which is evaluated in project checkpoint and included as project Bomberman), we didn't have animations. Our game update code blocks were tightly coupled with code blocks presenting the updates. In addition to this, we didn't have clear separation between managing game resources and using them. We dropped its implementation to the point that we have finised implementing centralized multiplayer. For our server, we developed a simple server written in python. To solve the problems of this iteration, we implemented our custom game framework.

In earlier iterations using our framework, we have tried RelativeLayout and SurfaceView to present the game. Yet in both cases, we stuck 15-25 frames per second. This was unacceptable since we aimed to have smooth gaming experience. Finally we have switched to OpenGL ES 1.0. With OpenGL, our game runs around 60 frames per second. We assume we have done a good job while implementing the game framework, since 60 frames per second is the practical limit and we achieved it even with our custom framework overhead. We skipped implementing centralized multiplayer server and start directly to decentralize multiplayer. Yet until the checkpoint, we couldn't fully implement the decentralized version.



Screenshots from the game

3. Implementations Choices

At the beginning of the project we mainly worked with the Eclipse and used the real devices for the testing. When it becomes the multiple player development phase we need to find a fast and reliable emulator in order to make the development life easier and it is experienced fact that the android in built bundled emulator is too slow. As a solution we moved to GenyMotion which is fast and reliable emulator which matched exactly to our requirements.

Moreover, at the same time we move to new emulator, we moved to the Android Studio which has much better capabilities integrated as an Android development IDE. Integrating OpenGL as an enhancement for the gaming screen was a challenging task but we thought of implementing that because it was vital in the end user experience point of view when we develop a game. We should warn the future testers that default emulator has a bug in OpenGL side which makes our game unplayable. We advise to use real devices or Genymotion emulator.

At the final phase where the multiplayer connectivity was based on Wifi Direct, we initially started development using WDSim but sooner we realized it is much faster and easier to move to the real devices because real devices provided more flexibility in device discovery and connectivity rather than WDSim. Obviously, the testing was much faster with real devices rather than using emulators at this phase.

4. Conclusions

At the end of the successful completion of this project, we experienced vivid aspects of android development and mobile computing. Even though the game development for mobile devices becoming more trivial today with the help of support libraries, this project was an immense challenge because we had to use the basic android tools. But ultimately we could make a quality product which provides a smooth gaming experience.

We think that, this project has given more weight to the technical, android design and development aspects rather than research oriented, in a certain mobile computing field of interest such as location, hoarding etc. Therefore we would like to propose for future projects to have higher weight for the research oriented work.