Mobile Game Development - Individual Assignment

1 Introduction

This document specifies the requirements for the first assignment of the Mobile Game Development course. This assignment will develop your skills by allowing you to put into practice what has been taught to you in class during the first 5 weeks of the course.

This assignment requires you to design, implement, test, and debug a simple *Breakout*-style brick-busting game using the LibGDX framework. The work for this assignment is to be performed **individually** and will be **submitted by learnOnline** at the end of the first week of the mid-semester break: that being **due by 18 April 2019 11:00 PM**. Refer to the section Submission Details for specific instructions.

If any parts of this specification appear unclear, please ensure you seek clarification.

2 Task Description

The style of game being made for this assignment is a straight-forward brick-busting game, of which the classic example is name *Breakout*, originally released as an arcade machine by Atari in 1976. The game consists of a paddle controlled by the player at the bottom of the screen, a number of rows of squares across the top of the screen called the "bricks", and a ball that bounces off of every object it hits, destroying the object for points

2.1 Main Features

Main Menu When the game is launched there must be a starting screen showing two buttons: One marked "PLAY" to play the game, another marked "EXIT" to return to the OS. The main menu's background can be colored in or left black. This screen may have a name for the game displayed as well.

Game Screen For the purposes of this assignment a game screen of 640 by 480 pixels is assumed. When starting the game there needs to be four rows of 20 bricks going across the top of half of the screen, with a 16 pixel gap between the top of the topmost row of bricks and the top of the screen. Each row of bricks must cover the entire width of the screen, with no gap between them and the sides of the screen.

In the bottom half of the screen there needs to be a single paddle whose bottom is 32 pixels above the bottom of the screen. The screen background may be colored in or left black.

Gameplay Once the player has pressed the new game button in the main menu, the game shows the bricks and paddle but does not start until they have pressed the screen again. Once the player first presses the screen, the paddle snaps to the horizontal position of the player's finger on the screen and continues to match the player's horizontal movement as long as their finger remains depressed.

At the same time, the ball appears just above the paddle and moves diagonally upwards (can be a fixed or randomly determined angle, but the ball cannot move straight up or across). Whenever the ball hits a brick, the ball bounces directly off of the brick and causes the brick to disappear. If the ball hits the paddle, or the sides or top of the screen, it bounces off with no further effect. If the ball touches the bottom edge of the screen, the fail state is triggered. All collisions need to be visibly accurate to the assets used, inconsistencies will result in lost marks.

Once in motion, the ball should move at a velocity whereby it would take about two seconds (You don't have to be entirely precise) to reach the top of the screen from the bottom if moving straight up with no bricks in the way.

Fail State Should the ball touch the bottom of the screen, the player has lost. All interaction with the game is turned off but the game screen remains visible in its final fail state, with the words "GAME OVER" printed in a large font in the top half of the screen and buttons to "RETRY" and "QUIT" in the bottom half of the screen.

The Retry button resets the game to the state it was in when the "PLAY" button was pressed from the main menu, while the "QUIT" button closes the game.

Success State Should every brick be destroyed by the ball, the player has won. All interaction with the game is turned off but the game screen remains visible in its final fail state, with the words "YOU WIN" printed in a large font in the top half of the screen and buttons to "RETRY" and "QUIT" in the bottom half of the screen.

The Retry button resets the game to the state it was in when the "PLAY" button was pressed from the main menu, while the "QUIT" button closes the game.

2.2 Additional Features

For the last few features you have a choice of which additions you would like to make to complete the game:

- A pause feature
- Sounds
- Looping background music
- A scoring system with a high-score display
- A unique graphical style

Each addition is worth a number of marks (see the Marking Criteria section for details) and their combined total is more than the maximum number of marks available for this part. The listed marks indicate a high-standard of completion, where there is no visual clash with other elements on-screen and manipulating any new controls does not come at the cost of inconvenient movements on the player's part.

These new features may require some additional research by you to implement as the course will not have covered all of these possibilities yet. Basically this is a chance for you to show off and go above and beyond what is expected of you. New features added are still subject to the requirement for your game to run quickly and smoothly and so make sure you do not use an excessive amount of memory or write poorly optimized code, either of which could cause your game to hang or stutter.

Descriptions of each additional system are available below:

A Pause feature Some ability to pause and resume the game when it is in play. This should only be possible in the game screen, not when there is any sort of fail state. While paused the game should have text in the top half of the screen in large font reading "PAUSED" and a buttons in the bottom half marked "CONTINUE" to resume the game and "QUIT" to close the game.

You may do this with a button on the screen though that will take up space needed to play, so you would want to make it somehow translucent. Do not make the pause button move around the screen if you choose to use a button.

Another option is to activate the pause feature by pressing back key on an Android phone (simulate this with the backspace key on the keyboard for the desktop module). This would allow you to keep the play area free of distractions.

Sounds Adding sound will give the game much more atmosphere. Ensure you have a complete suite of effects including button presses, the ball touching different surfaces and win/lose conditions. Remember to only use sounds you have the rights to use, freesound.org is a good start, as is BFXR.

Looping Background Music Adding a nice soundtrack can also improve the play experience. If you do this make sure the song loops continuously. Finding appropriate royalty free music can take some digging around, many people swear by incompetech for their music needs.

Scoring System with High-Score Display The exact method of scoring is up to you, remember that simply counting the number of blocks is not reliable as every successful playthrough will result in the same score, so it is better to include some sort of time-constraint in the scoring system.

When going in this direction, the user's score needs to be displayed on screen during the game as well as in the win/lose condition states. To get the second half of the points for this feature you will also need to add a third button on the main menu screen populated with the high scores for the current session. You do not need to store the score data on the phone.

Unique Graphical Style Instead of using the provided assets, you may choose to replace or modify some or all of the assets to make them more striking and unique. If you want to get the full marks for this feature, you must ensure all of the assets added look like they're part of a consistent art style and they must all appear consistently scaled (no mixing resolutions). You must also add background images to each screen that do not distract from the gameplay or obscure the interactive elements displayed.

2.3 Assets

You will be provided public domain assets for all of the assets required to make the main features of the game. You must also ensure any new assets used are either made by yourself or licensed under a permissive license for use in an academic project. Use of public domain resources is recommended.

This really should go without saying, but make sure that any assets you add to the game are safe for work.

2.4 Documentation

Use Javadoc-style documentation on methods and constructors. Documentation can be omitted for simple methods if their functionality is obvious and they have no side-effects.

Code comments should be used sparingly, aim to comment blocks of code rather than individual statements unless the outcomes of the statements are not obvious.

2.5 Code Style

You must write your code in a consistent style. Keep your style close to Java guidelines and if you choose to deviate from it, please do so consistently. Note that you are not specifically required to follow Object-Oriented design, you nearly need to make sure your code is legible.

2.6 Version Control

You must use a git repository to track your implementation progress and to keep a record of your work to show you have not committed plagiarism. You may choose to exclude the majority of the project from being tracked to keep your file version history compact and make it easier to switch between computers, but you must at least include the source directory of the `core` module. If you make any changes to the source in the `android` or `desktop` modules then be sure they are included in version control as well.

3 Submission Details

You must submit your complete Android Studio project folder in a single zip archive file. The zip archive must include all source files and asset files required to compile and run your project in both android and desktop modes. The submitted archive must also include your version control directory (there should be a `.git` folder inside the project).

Once you have created the zip file according to the specifications, you are required to upload it to the **Individual Assignment Submission** link via the course website on *LearnOnline*. The deadline for the submission is **18 April 2019 11:00 PM**, after which time any further submissions will be considered late and attract a penalty of 10% of their final mark for every day past the due date.

4 Marking Criteria

Criteria	Mark
Main Features	(45)
Main Menu with Play and Exit buttons	5
Control of paddle with touch/cursor controls	5
Collision response of ball with paddle	10
Collision response of ball with bricks	10
Collision response of ball with screen edges	5
Fail and Success States, including Retry button	10
Stability*	(25)
Desktop module compiles and runs without errors or crashes	10
Android module compiles and runs without errors or crashes	5
Desktop module runs smoothly without hangs or stuttering	10
Code Quality	(15)
Documentation and Commenting	5
Consistent and Readable Code Style	5
Correct use of Version Control	5
Additional Features†	(15)
Ability to pause the game	5
Sounds	5
Looping Background Music	5
Scoring and separate High-Score screen	10
Unique Graphical Style	5
Total Possible Marks	**100**

* The specific hardware target used to measure performance and stability is the computers used for the practicals in the university mac labs

† Additional Features are worth a maximum of 15 points in total

5 Academic Misconduct

This is an individual assignment: your submitted files will be checked against other student's code, and other sources, for instances of plagiarism.

Students are reminded that they should be aware of the academic misconduct guidelines available from the University of South Australia website.

Deliberate academic misconduct such as plagiarism is subject to penalties. Information about Academic integrity can be found in Section 9 of the Assessment policies and procedures manual at: http://www.unisa.edu.au/policies/manual/