COMPSYS 302 Final Report (10%)

1 Game Functionality

1.1 Random Level Generation

The game randomly generates a level for the player to play each time the player chooses to play the game or begin a new level. This random game generation occurs in BrickGroup.java. Each time a new level is started, the GameController.java class calls for a randomly generated level. This level is generated based on the current difficulty setting to customise the speed at which the bricks move downward and the number of bricks shown on the screen. The randomly generated levels are also symmetrically centred about the middle to give the game a great visual appearance. To change the difficulty of the game, and thus the randomly generated level, the player will need to go back to the menu screen and enter the setting screen to adjust the difficulty.

The random level generation works in the following way. A brick group is initialised when the game is first started. This creates a block of 8x7 bricks of the appropriate size with respect to the screen, each storing their size, location and visibility. The constructor for the random initialisation ends here. When the player wishes to play the game, the program iterates through the brick group to turn off certain bricks according to a random number generator and sortArray function. This way the random brick generation efficient as the only operation it needs to do is turn on and off bricks rather than create new bricks.

1.2 Move Bricks towards Ground

By moving the bricks towards ground, this function essentially adds a timer to the game. A search is conducted every cycle, from the last brick to the first to see whether a collision has occurred between any of the bricks and the paddle or ground. If so, the player loses a life and the level is restarted.

The move function works by iterating through the brick group, backwards (from the bottom most brick to the top most brick) every cycle and move the brick down. If the brick it is currently checking is in contact with the paddle or the ground, it immediately breaks from the loop, informs the player and ends the game. This way this search function is more efficient allowing the game to run faster.

1.3 Power Ups

Each time a brick is hit the game requests a random number to determine whether a power up will be dropped. If it will be dropped it determines what power up it is, change the colour and then move the power up towards the bottom of the screen at a predetermined speed. There are 2 interactions the power ups can have. One occurs when the power up touches ground, when this happens the power up disappears and becomes invalid. The other occurs when the power up drops onto a paddle causing the icon to disappear and the power up to take effect.

There are 3 power ups that can be caught:

1. Long paddle – lengthen the length of the paddle
2. Super ball – make the ball able to go through bricks without bouncing
3. Bomb – destroys the paddle causing the player to start again. A display is shown to inform the player.

2 Object-Orientated Design

2.1 MVC

The Entire code is created using model-view-controller method. This is done to assist with maintainability so that additional features can be added when required.

The main controller for the Game-Play section is the `GameController.java`. It controls all the interactions between objects and what to do about them. `GameController.java` mainly manages the following functions:

1. Running the game through `game_thread`
2. Creates and manages game component instances (ball, paddle, bricks and power ups)
3. Manages interactions between these objects
    a. Ball hits boundary
    b. Ball touch ground
    c. Ball hit blocks (drop power-ups)
4. Manages when objects move
    a. Move ball
    b. Move power up

2.2 Encapsulation

Every single class is programmed with encapsulation. All variables are private to the class in which they were created. Get-a and Set-a methods were created in order to handle those variables. For example, in the BrickGroup.java class has the following private attributes and get-a set-a methods.

```
private int brick_count;
private int individual_brick_width;
private int bricks_to_generate=18;//number of bricks that will be randomly

public int getBrickGroupX(){return brick_layout.length;}
public int getBrickCount(){return brick_count;}
public void setBricksToGenerate(int n){bricks_to_generate=n;}
```

This makes the code easier to understand.

2.3 Inheritance

Inheritance is used throughout the program to allow use the functions of the parent. These were used especially for user interface design as each UI screen in section 3 below all `extends JPanel`. The `Cardlayout` that manages screen displays is inherited from `JFrame` (`extends JFrame`).

This allow the interface to use functions such as

```
super.paint(g);
fillRect(x,y,width,height);
repaint();
g.setFont(new Font ("Bradley Hand ITC",Font.BOLD, 24));
menu_button.setOpaque(false);
```

et cetera to assist with the formatting of the screen.

3 User-Interface (UI) Design

3.1 Screens

The screens are all set out in a card layout that switches through the cards. There are 5 cards, each producing a different screen. These cards (screens) are explained in more detail below. These cards have been laid out in a way that makes it possible to switch to any card by calling the `Window_JFrame.switchCard()` function.

Screen 1: Loading Screen

This screen is shown when the game is first started. If the computer runs fast enough the user may not see this screen due to the speed at which the game finishes initialising and moves on to the menu screen.

Screen 2: Menu Screen

This screen gives the user several options:

1. Start the game by pressing the “Play Game” button with the mouse
2. Enter the setting screen to change the difficulty of the game by pressing “Setting”
3. Exit the game by pressing “Exit”

Screen 3: Play Screen

This screen shows the actual game. It shows the bricks at the top, the ball and the paddle. Power-ups are shown when they are triggered randomly. This screen allows the player to use their keyboard to control the paddle and play the game.

The Play screen is central to the game functionality. It governs how the game is played and the actions implemented when game objects (paddle, ball, brick, boundary, ground, and power-up) interact.

This screen has a “Menu” button that allows players to return to the menu screen (screen 2). It is defined in the `GameController.java`

Screen 4: End Screens

This screen is displayed when the player finishes their game. This can arise from pressing the “menu” button during the game. It displays the scores and lives after the game and allows the user to return to menu or exit.

Screen 5: Setting

This screen has 5 invisible buttons that form the slider. They allow the user to select their desired level between 1 and 5. There are additional 2 buttons on this `JPanel`: “return to menu” to return to the menu without making any changes and “save changes” to save the new level setting.

Screen 6: Instructions

This screen is shows the instructions of the game. By pressing any key or mouse click the player can enter the play screen (screen 3).

3.2 Graphics

The graphics in this game have been nicely designed to establish the mood of the game. They were created using JPanel (card layout), JFrame and JComponents. The labels, buttons, drawText, overlays have all been designed so that they adjust their size with relation to the screen so that if the screen size is changed the game layout will not be affected. However due to the 600x800 minimum requirement, the screen resize ability had been set to disabled.

Screens from above: