

Programming Project

Theme: *Game Brick Breaker*

**COMP1020: OOP and Data Structures Spring 2022**

Final Report

1. **Product Description**

* **Name: Brick Breaker**
* Brick Breaker is a type of game in which the player must smash a wall of bricks by deflecting a bouncing ball with a paddle. The paddle could move horizontally and be controlled with the computer's mouse. Whenever the ball touches a brick, it will disappear, and the player wins when breaking all the bricks or loses when failing to catch the ball with the paddle.
* There are several versions of Brick Breaker that has been developed throughout years (around half of a century). The main difference between the 2 most common type is that one ball and multiple balls. In our game, we use the “one ball” version.
* Although this game is so popular in the world, we believe that we have updated further functionalities to make it more unique and interesting, details will be discussed later in this report.

1. **Project Deliverable Issue** (Is there any expected outcomes that cannot be delivered? If YES, please justify the reason(s) why it can’t be delivered, e.g., technical difficulty, time management issue, team conflict, etc.)

The game is finally successful, but it has been quite a tough journey. However, we were able to anticipate such difficulties, all problems are solved in the end.

1. Due to a huge number of references on the internet about this game, we encountered an ironic situation when we must come up with different things to distinguish our game from others. We assembled all the coolest features of this game and integrated them to our final product. Other uniqueness is about the design part when we build the main interface, game maps, and backgrounds by ourselves.
2. It is the first time for the whole team to develop a complete game from scratch, so we had to get familiarized with including definitions, terminologies, and implementation of many techniques like GUI, awt and swing packages, etc. Thanks to this project, we could be able to widen our knowledge in this exciting field.
3. As all 4 members contributed to the mutual work, we must split the tasks reasonably and then gather the code. However, each of us has our own “style” and the code is really complex, we faced a lot of difficulties when finding ways to run all code smoothly. Fortunately, we knew how to use GitHub effectively and after some meetings to explain our ideas, we could finalize all the game.
4. To be honest, internal conflict is something inevitable when doing teamwork projects. There is no perfect “dream team” but we have to sit down together and figure out the problems, as well as agree on an appropriate solution. In the first weeks, we were a little independently and subjective. Moreover, sometimes we couldn’t deal well with the task allocation since there are too many details. After all, we had several discussions and were able to find mutual voice, as not much time left so we prioritized finishing the game first. We have learnt many things about time management and conflict management skills.
5. **Product Functionality** (Describe the main functions/features in your product)

***Side functions:***

* When users open the game, it will display the introductory interface and users need to type in their name to continue.
* Background music plays right in the beginning of the game and only stops when users adjust it. Also, there is a short sound when the ball collides with the brick to notify users.
* In the “Settings” part, users could modify the Paddle color & Ball color, turn on or off the music, and choose the map that they want.
* In the “Scoreboard” part, top 10 highest scores ever will be displayed. The name column is saved at first when users enter their names, and the scores are automatically saved every time users lose.

***Game features:***

* There are 2 modes of the game: Classic and Super, which are not much different except the dropping items from broken bricks in Super mode.
* Users could choose one of the 7 maps; all will have the exactly same features but in diferent styles.
* Basic rules are clearly stated in the Description, users have to move the paddle and catch the ball until all bricks are broken.
* Users can pause the game and resume whenever they want.
* Score is calculated by a counter and displayed on the up left corner of the screen.
* The hardness of the brick will be divided into 4 levels with 4 different colors: Blue - 1 hit (get 2 points). Green - 2 hits (get 4 points). Yellow - 3 hits (get 6 points). Red 4 hits: (get 8 points). When each brick is hit, it will change color to the color with the previous hardness (e.g. when hitting a red brick, it will change to yellow).
* Some bricks contain special items inside that will drop when bricks are broken. Users must move the paddle to “eat” such items and get the functions. However, it is just limited to a certain amount of time (10 seconds), after that it will return back to the initial status. The functions of the item might include:
* Multiply 3 balls: the ball splits itself into 3, ending when 2 of the 3 balls fall out
* Enlarge: lengthen the paddle further

1. **Architecture** (Architecture model, technology, software, and hardware used)

Programming language: 100% JAVA

IDE: Eclipse

Code management: GitHub – store code and collaborate together

Package: swing, awt, util, etc.

* “resources”

Is used to store the necessary files for the game: audio sounds, background images for interfaces of main game, game icons, and a “scores.txt” files

* Main game

We organize the projects and split the main function into 3 packages for different purposes:

* + Manager: Saved 2 classes “Constants” and “RoundBtn” which are the constant variable such as font, color, size frame that are used for the main game and UI.
  + Maps: Stored the design for our maps of our game
  + main: This is where all the algorithms and the UI design is located
* Maps

1 “Map” file in “main”: call for the maps when users choose, define the color for each number using nested list, method hitBrick() used to make the bricks disappear when touching the ball, method winCheck() used to check if there are any bricks left on map to end the game

7 files in “map” package: set the background and the number of row and column, imagine 2 coordinate axes and “draw” – assign number to each coordinate

* Game UI
  + For the UI of the menu and game, we use JFrame, JButton and JTextField for taking input, Jlabel, and JPanel to organize elements in the frame.
  + Classes are extended class PressingKeys and overrides its methods to listen to the events from users.
  + Use JPanel(), Insets(), GridBagLayout() methods to design the background for buttons
  + Size 640 x 480, awt, JFrame; design & position of Ball & Paddle is defined in their own files and also an update() method to call when users get special items; fixed font & size & color of the game is stored in “Constants” file in “Manager” package
  + Utilized encapsulation through making the class variables of these objects *private*, only accessible through getter and setter methods
* Settings:

Both uxt and swing to develop UX and UI for the “Settings” window that enables users to customize the game: Paddle Color (yellow, blue, red), Ball Color (yellow, blue, red), Music (on, off), Map (7 maps)

JRadioButton & ButtonGroup to ensure that users can choose only 1 option at a time

Use addActionListener & actionPerform methods to implement the fuction when users choose options

* Scoreboard:

Utilized BufferedWriter to write and store the scores into a notepad file after the game is over

Utilized BufferedReader to read the score from that file for highscore management

Used a LinkedHashMap and a simple Comparator to store the name and corresponding score in descending order of score values

Used two lists to split the hashmap into a list of names and a list of score for simplicity when drawing the scoreboard

* Game algorithms classes
* HUD & Score

Calculate the total score each game, also a “MULTIPLIER” number, and display them on the up left corner of the screen; everytime the ball breaks a brick, it will be automatically added 1 point, use simple awt commands

* Power Ups

Utilize polymorphism to create different powerup types and corresponding interactions with the paddle/ball

Import javax.swing to add items’ icons

The timer and feature for each item are already set in the Ball & Paddle file

* Sound

Utilize javax.sound.sampled in 2 file “Sound” & “CollisionSound” to create a background music and collision sounds: AudioInputStream, Clip, AudioSystem

Users are able to turn on or off the background music in “Settings” window

* GamePanel

Control all main activities of the game: checkCollisions() – use nested if-else to implement PowerUps functions and hitBrick(), addScore(); drawWin() & drawLose() to display on the screen when the game ends

Implements “KeyListener” interface to respond when users press any key, including method keyPressed(KeyEvent e) and getKeyCode()

Implements MouseMotionListener to controll the paddle when users move the mouse

1. **Design** (GUI design-main screenshots, UML-activity diagram)
2. **Task distribution**