**CS102 Game Project Proposal:**

**Matrix 110**

**Group Members:**

**Shumirai Gunzo**

**Lehlohonolo Theletsane**

**Siphesihle Ndlovu**

**Lungelo Mngomezulu**

**Siyabonga Mgwenya**

**1. Game Title: Brick Breaker Matrix v1.10**

**2. Background and Motivation**

***Brick Breaker*** is a ***Breakout*** clone in which the player must smash a wall of bricks by deflecting a bouncing ball with a paddle (Wikipedia, 2024). The paddle may move horizontally and is controlled with the computer’s mouse or the touch of a finger (in the case of touchscreen). The player gets three lives to start with; a life is lost if the ball hits the bottom of the screen. When all the bricks have been destroyed, the player advances to a new, harder level. There are 34 levels. Many levels have unbreakable silver bricks. If all lives are lost, the game is over (Wikipedia, 2024).

There are many versions of ***Brick Breaker***, some in which players can shoot flaming fireballs or play with more than one ball if the player gets a power-up (Wikipedia, 2024). Our version of the game will have a split-screen multiplayer mode, where the player with the highest score when all the lives of both players deplete wins. It will also include timed power-ups which the player will get when they destroy certain bricks. These power-ups include more balls, a longer paddle, and a stronger ball that can destroy a brick with one hit (the normal ball will take 2 hits to destroy a brick). We will implement a scoreboard and a life counter to allow the player to keep track of their progress and make the game more fun. The aim of this version of *Brick Breaker* is to destroy as many bricks as the player can before the player’s lives deplete. We will also implement sound effects and background music.

**3. Problem Statement**

The aim of this version of *Brick Breaker* is to destroy as many bricks as the player can before the player’s lives deplete, the game’s setting and the sprites we will use, will be different from other versions of it. This game has a moderate complexity, but features will be added in an attempt to increase its complexity, attractiveness, and depth for casual and competitive play.

**4. Approach**

This project aims to implement a Brick Breaker game in Java with the following milestones:

* **Milestone 1**: Code an array that can contain the layout of bricks and the paddle.
* **Milestone 2**: Implement the ball’s movement and collision detection with bricks, paddle, and screen boundaries.
* **Milestone 3**: Add power-ups and their effects on gameplay.
* **Milestone 4**: Implement a split-screen multiplayer mode.
* **Milestone 5**: Create a scoreboard and timer to track player progress.
* **Milestone 6**: Add sound effects and background music to enhance the gaming experience.
* **Milestone 7**: Develop an easy-to-use GUI to improve the attractiveness of the game.
* **Milestone 8**: Present the work for assessment in the form of a GitHub code repository: <https://github.com/GoldenOwl24/Matrix110.git> and Project Document, which will include the revision and extension of this proposal. The extensions will include a walkthrough of the steps through the SDLC that our group followed, and a Visual Guide that shows the game in action. We will do this through ‘storyboards’ with an explanation of each frame, that will show the start of the game, playing example of the game, and the end of the game.

**5. Technologies and Concepts**

* **Object-Oriented Programming (OOP):** OOP principles such as inheritance, encapsulation, and polymorphism make it easier to manage complex game projects and develop more sophisticated game mechanics (Bello, 2024). We will use OOP principles to implement the game. Classes and interfaces will be used to represent game entities and their interactions.
* **Collection Framework**: Java Collection Framework defines several classes and interfaces to represent a group of objects as a single unit (Vypirailenko, 2023). We will use Java’s Collection Framework to represent lists of bricks and power-ups.
* **Java Exceptions**: Java Exception mechanism simplifies bug catching in programs (Vypirailenko, 2023). Exception handling will be implemented to manage errors and ensure the game runs smoothly.
* **Input/Output Streams**: Java performs Input and Output operations via Streams. A stream seems to be a continuous flow of data (Vypirailenko, 2023). We will use I/O streams to handle file operations, such as saving and loading game states.
* **Multi-threading**: the concurrent execution of two or more processes with the most efficient use of a CPU (Selawsky, 2019). Parts of the game will run concurrently to improve performance and responsiveness.
* **GitHub**: We will use GitHub to track changes, coordinate work, and manage development. This is the link to our GitHub repository: <https://github.com/GoldenOwl24/Matrix110.git>
* **Java Swing**: We will use Java Swing to create the game’s user interface, including JFrame, JPanel, and JButton.
* **LibGDX**: libGDX is a framework for developing cross-platform games (Vypirailenko, 2023). We will learn this framework and use it to make our game work on both Android and Windows platforms.

**6.Group Member Responsibilities:**

* **Shumirai Gunzo:** Graphics Programmer – Optimizes the visual rendering pipeline to achieve stunning graphics and high performance (Baker College,2024).
* **Lehlohonolo Theletsane:** Systems Designer – Focuses on designing the game mechanics and underlying systems driving the gameplay (Baker College,2024).
* **Siphesihle Ndlovu:** Audio Programmer – Responsible for integrating sound effects, music, and voiceovers into the game (Baker College,2024).
* **Lungelo Mngomezulu:** UI/UX Programmer – Focuses on the development of user interfaces and user experiences, ensuring the game is accessible and enjoyable for players (Baker College,2024).
* **Siyabonga Mgwenya:** Level Designer – Responsible for designing and building game levels, including layout, pacing, and player progression (Baker College,2024).

**7. Timeline**

| **Milestone** | **Date** |
| --- | --- |
| Proposal | August 14 |
| Design | August 31 |
| Milestone 1: Code a data structure for bricks and paddle | Week 1 |
| Milestone 2: Implement ball’s movement and collision detection | Week 2 |
| Milestone 3: Add power-ups and their effects | Week 3 |
| Milestone 4: Implement split-screen multiplayer mode | Week 4 |
| Milestone 5: Create scoreboard and timer | Week 5 |
| Milestone 6: Add sound effects and background music | Week 6 |
| Milestone 7: Develop an easy-to-use GUI | Week 7 |
| Milestone 8: Present the work for assessment | Week 8 |
| Implementation | October 6 |
| Presentation | During the week that starts on October 7 |

**8. References**

* Wikipedia contributors. (2024, May 29). Brick Breaker. In Wikipedia, The Free Encyclopaedia. Retrieved 20:08, August 8, 2024, from <https://en.wikipedia.org/w/index.php?title=Brick_Breaker&oldid=1226287109>
* Baker College. (2024). *Game Developer Jobs: Where Can Your Passion for Gaming Take You?.* Retrieved from <https://www.baker.edu/about/get-to-know-us/blog/game-developer-jobs-where-can-your-passion-for-gaming-take-you/>
* Vypirailenko, A. (2023). *Java Game Programming For Beginners: Where to Start*. CodeGym. Retrieved from <https://codegym.cc/groups/posts/182-java-game-programming-for-beginners-where-to-start>
* Selawsky, J. (2019). *The Secrets to Learning Java Game Development with Beginner Coding Skills*. Level Up Coding. Retrieved from <https://levelup.gitconnected.com/the-secrets-to-learning-java-game-development-with-beginner-coding-skills-e5341eca74f8>
* Bello, A. (2024). *Game Development with Java: A Comprehensive Guide*. AmorServ. Retrieved from <https://amorserv.com/insights/game-development-with-java-a-comprehensive-guide>