Student name:	Veronika Hrytsyk					
Student number:	**3128619**					
Faculty:	**Computing Science**					
Course:	**BSCH/BSCO/EXCH**			**Stage/year:**	**2**	
Subject:	**Software Development 2**					
Study Mode:	Full time			Part-time		
Lecturer Name:	**Gemma Deery**					
Assignment Title:	**Report 3**					
Date due:	**29/04/2025**					
Date submitted:	**29/04/2025**					

Plagiarism disclaimer:

I understand that plagiarism is a serious offence and have read and understood the college policy on plagiarism. I also understand that I may receive a mark of zero if I have not identified and properly attributed sources which have been used, referred to, or have in any way influenced the preparation of this assignment, or if I have knowingly allowed others to plagiarise my work in this way.

I hereby certify that this assignment is my own work, based on my personal study and/or research, and that I have acknowledged all material and sources used in its preparation. I also certify that the assignment has not previously been submitted for assessment and that I have not copied in part or whole or otherwise plagiarised the work of anyone else, including other students.

Signed: Veronika Hrytsyk **Date: 29/04/2025**

Project Report: Player Movement, Collision Logic, and Testing System

In the course of this project, my primary responsibility entailed the implementation, and validation of the player movement system, with a particular focus on accurate collision detection. I implemented and tested the logic that governs how the player navigates the game world, ensuring they respond naturally to physical constraints such as solid tiles, walls, and gravity. These mechanics are fundamental to player experience and are tightly integrated with both the rendering and game loop systems.

I concentrated on the detailed development of Java-based systems that manage player physics, including gravity-induced movement, directional controls, and boundary adherence. Central to this was the integration of hitbox logic and level data analysis to detect and prevent invalid movements. To guarantee the robustness of these systems, I also created an extensive suite of JUnit tests targeting the most critical elements of the collision and movement systems.

Player Movement and Gravity Simulation

To simulate natural movement, I implemented continuous vertical displacement driven by gravity, allowing the player to fall when not grounded. This behavior was embedded into the Player.update() method, which calculates and applies vertical movement frame by frame. For testing purposes, the player is initially spawned mid-air and expected to descend until stable contact with the ground is detected.

This gravitational system was verified using the PlayerPositionTest class, particularly in the testPlayerStartsAboveGround() and testPlayerStopsAtGround() methods. These tests confirmed that the player moves downward as expected and eventually stabilizes upon reaching the ground, without falling through or getting stuck.

Level Data and Collision Constraints

To ensure meaningful player interaction with the world, I implemented a collision system based on level data—represented as a 2D array (lvlData) where tile values determine solidity. Walkable tiles are marked with specific integers, while solid tiles represent barriers such as the ground and walls.

A custom test grid was created in the setUp() method of PlayerPositionTest to simulate this environment. The bottom row and outermost columns of the level were marked as solid, effectively bounding the playable area. This setup enabled consistent testing of both gravity and collision behaviors and was validated in testLevelDataInitialization().

JUnit Testing Strategy

A comprehensive suite of unit tests was developed using JUnit 5 and placed within a dedicated testcases package inside the main source directory. This organization ensures compatibility across multiple devices and IDE setups, enabling all team members to contribute to testing without technical barriers.

The PlayerPositionTest class specifically tests dynamic player behaviors under various conditions, such as:

- **Falling due to gravity**
- **Halting on solid ground**
- **Valid initial spawn location**
- **Correct level data setup**

Assertions with a tolerance margin were used to accommodate floating-point imprecision, ensuring reliable and flexible validation of physical positions during simulation.

Modularity and Scalability

The system was developed with modularity in mind. Core responsibilities are distributed across classes:

- Entity: for shared behavior and properties like hitboxes.
- Player: for custom movement and gravity logic.
- HelpMethods: for tile-based collision validation (used in other test classes).

This modular design not only makes the system easier to maintain and extend but also enables efficient unit testing and reuse of logic across game entities. Future enhancements such as new player abilities, AI characters, or tile interactions can be integrated with minimal restructuring.

Conclusion

In summary, I developed and tested a robust movement and collision system that governs player behavior within the game environment. Through careful integration of level data, gravity, and hitbox logic, combined with extensive unit testing using JUnit, I ensured both functionality and reliability. This groundwork forms a scalable and maintainable foundation for ongoing development, providing a stable base upon which future gameplay features can be built.