Bala: A 2D Bullet Heaven Game Engine

A Research Project
Presented to The Faculty of the
School of Computing
Holy Angel University

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Computer Science

Julian Mathew M. Guintu
Gabriel V. Pagcu
Aidan Zabdiel A. Serrano
Carlos Jose B. Ysais

October 2025

APPROVAL SHEET

This capstone entitled “**BALA: A 2D BULLET HEAVEN GAME ENGINE**”, prepared and submitted in partial fulfillment of the requirements for the degree Bachelor of Science in Information Technology with Area of Specialization in Web Development, has been examined and is recommended for acceptance and oral examination.

Dr. Raquel B. Rivera
Adviser

ORAL EXAMINATION

Approved by the committee of oral examiners on
September 10, 2025

Mr. Wilson D. Concepcion
Panel Chair

Ms. Carisma A. Caro
Panel Member

Mr. Christian Jericho Ngo
Panel Member

APPROVAL

Accepted and approved in partial fulfillment of the requirements for the degree Bachelor of Science in Computer Science

Mr. Ulysses Raymond F. Monsale
Program Coordinator,
Computer Science

Dr. Marlon I. Tayag
Dean, School of Computing

Acknowledgment

The researchers extend their sincerest appreciation to Almighty God for providing the wisdom, strength, and endurance necessary throughout the entire duration of this study.

Deepest gratitude is extended to the adviser, Dr. Raquel B. Rivera, for the invaluable guidance, constructive recommendations, and constant support, all of which played an integral role in the successful completion of this study.

The researchers also express their sincere appreciation to the panelists from the proposal and final oral defenses. Their insightful remarks and suggestions were crucial in refining and enhancing the study.

Special acknowledgement is offered to the experts who contributed their time, involvement, and professional insights. Their dedicated participation provided the necessary data that made this research feasible.

The School of Computing faculty and staff are thanked for their institutional support and for fostering an environment conducive to academic and research development.

Sincere gratitude is expressed to the friends and families of the researchers for their unwavering support, understanding, and encouragement throughout the most challenging phases of this research.

Finally, the researchers recognize their fellow researchers, whose dedication, commitment, and collaborative association enabled this research to be successfully conducted.

Laus Deo Semper!

Table of Contents

List of Tables

List of Figures

Bala: A 2D Bullet Heaven Game Engine

Julian Mathew M. Guintu, Gabriel V. Pagcu,
Aidan Zabriel A. Serrano, Carlos Jose B. Ysais

Holy Angel University
School of Computing

Abstract

This research explored the development of Bala: A 2D Bullet Heaven Game Engine, a tool designed to address the needs of novice game developers. Its primary goal was to simplify game creation within the "bullet heaven" genre by offering a user-friendly visual interface, thus lowering the barrier to entry for individuals with limited programming expertise. Flocking was utilized for behavior simulation, serving as the game's artificial intelligence. To generate dynamic environments and enemy spawn points, procedural generation algorithms, specifically Simplex and Perlin noise, were implemented. Collision detection was accomplished through a combination of circle-circle collision and axis-aligned bounding boxes. The researchers employed the Input-Process-Output (IPO) model to illustrate the development process, incorporating continuous feedback for iterative improvement. A quantitative research design was used to measure the engine's usability and performance. Usability was assessed using a System Usability Scale (SUS) questionnaire administered to game developers. Engine performance was measured by collecting frame time data during stress and performance tests. These frame times were used to calculate 28 paired t-test values to determine the statistical difference between the algorithms and 8 coefficients of variation to measure algorithmic stability. The test results indicated that PBnF yielded the most stable frame time. However, the SCnF combination achieved the lowest average frame time, procuring the best results across both categories.

Keywords: game engine, bullet heaven, procedural generation, Simplex noise, Perlin noise, collision detection, axis-aligned bounding boxes, circle-circle collision, game artificial intelligence, non-player character, behavior simulation, flocking, frame time, performance testing

Bala: A 2D Bullet Heaven Game Engine

Introduction

The development of video games, particularly those in the "bullet heaven" genre, frequently relies on complex programming logic and necessitates considerable expertise in established game engines, such as Unity. This complexity presents substantial challenges for students and novice developers, who often encounter difficulties in effectively implementing game mechanics and achieving desired gameplay experiences.

Bullet heaven, also known as reverse bullet hell or survivors-like, is a sub-genre of shoot 'em up (shmup) games inspired by the bullet hell genre. In bullet heaven games, players control a character capable of generating attacks, often characterized as an auto-shooter. The main objective is to eliminate waves of enemies while maneuvering strategically to avoid contact. Unlike traditional bullet hell games, where players primarily dodge incoming bullets, in the bullet heaven variant, the player must dodge enemies while generating projectiles to defeat them (Emelianov, 2024).

Bullet hell, or manic shooter, is itself a subgenre of shmup focusing intensely on evading a vast number of enemy projectiles, which gives the genre its descriptive name.

The game Vampire Survivors, an action roguelike developed by Poncle and released in December 2021, popularized the bullet heaven genre. In this game, players control characters that attack enemies automatically, requiring the player to focus on survival and strategic positioning within large enemy waves. The game shifted the traditional bullet hell mechanics by having players generate projectiles rather than solely avoiding them (Poncle, 2025).

Vampire Survivors achieved significant critical and commercial success, evidenced by its "Overwhelmingly Positive" Steam reviews (98.5% positive from 246,738 reviews; Poncle, 2022) and high Metacritic scores (87/100 from critics and 8.2/10 from players for the PC version; GameSensor, 2023). In light of this success and the genre's popularity, the bullet heaven format presents a promising opportunity to develop a specialized game engine, which can also contribute to the growth of the Philippine game development sector.

This research involved the development of a 2D game engine named "BALA" specifically designed for bullet heaven games. The engine was created to address the needs of game developers by providing a user-friendly and visual interface. The researchers selected the name "BALA" because it is the direct Filipino translation of the English word "bullet," signifying its connection to the genre. The decision to develop a 2D engine was based on the goal of targeting novice developers, as 3D engines are typically more complex and less beginner-friendly, and 2D development requires fewer computing resources. Furthermore, the defining game of the genre, Vampire Survivors, is also 2D. The term "bullet heaven" was used throughout this study over "survivors-like," as the former is more prevalent within the community” (Emelianov, 2024).

Review of Related Literature

Overview of Game Engines

Creating a game engine allows developers to focus on their ideas rather than the technical implementation. Game engines and frameworks provide features that reduce effort and time in video game development (Campos et al., 2022). Motivations for creating a custom game engine often include the pursuit of novel technology, specialization for a specific genre, a desire for independence, and learning objectives (Glaiel, 2021). Glaiel (2021) further noted that engines should be tailored to specific needs, given that commercial alternatives may contain features that are unnecessary for a particular project.

Unity Engine is a widely used platform for game development, offering a streamlined creative process through its interface components like the scene view, game view, hierarchy, project window, and inspector (Prakash et al., 2023). It can be used as a game engine for teaching game development due to its widespread use in the industry, cross-platform compatibility, rich features, and low barrier to entry. A computer game development course can be designed around Unity, enhancing students' knowledge and skills (Moiseienko et al., 2023). Commercially available game engines like Unreal Engine, Unity, CryEngine, and Godot provide efficient workflows but may not suit every developer's needs. Open-source game engines are gaining popularity due to their accessibility, customizability, and cost-effectiveness (Park & Baek, 2020).

Existing Game Engines and Frameworks

Unreal Engine is a game engine developed by Epic Games, first released in 1998. Initially designed for first-person shooters, Unreal Engine has evolved into a widely used engine in various industries, including game development, film production, and virtual reality

(Unreal Engine, 2025). Its high-quality rendering and extensive toolset have made it a go-to for large studios. Popular examples of games developed using it include Fortnite and Gears of War.

Unity, developed by Unity Technologies, is a cross-platform engine commonly used for both 2D and 3D development (Unity, 2024). Its accessibility and extensive asset store have made it popular for games such as Hollow Knight and Among Us.

CryEngine is a game engine developed by Crytek, known for its high-performance graphics and realistic environmental rendering (CryEngine, 2025). It was initially used in games like Far Cry but has since been adapted for other projects, including Amazon Lumberyard. Despite its powerful capabilities, CryEngine is often considered less beginner-friendly than other engines.

Godot is an open-source game engine known for its lightweight structure and flexibility, which makes it a popular choice for indie developers (Godot Engine, 2025). It supports both 2D and 3D game development and features its own scripting language, GDScript. Godot is free under the MIT license and has been used in games like Deponia and Carol Reed Mysteries.

Game engines are essential tools in game development, with popular options like Unreal Engine, Unity, and CryEngine enabling efficient workflows for developers. Recently, lightweight open-source engines have gained traction due to their accessibility, expandability, and cost-effectiveness. Open-source engines provide source code access, allowing developers to customize and enhance features for specific needs (Park & Baek, 2020).

Existing research, such as the work on Zetcil: Game Mechanic Framework for Unity Game Engine (Roedavan et al., 2020), has highlighted the academic need for simplified game development frameworks, especially those that facilitate visual game mechanics. This research indicates a trend away from purely code-based implementation toward a more intuitive, visual game design approach.

While existing game design frameworks such as Zetcil have shown promise in simplifying game development for Unity (Roedavan et al., 2020), a specialized framework tailored for the unique demands of Bullet Heaven games remains an unexplored area.

Custom Game Engines

Politowski et al. (2020) argued that a principal justification for developing custom game engines is to achieve greater control over optimization and workflow efficiency. Commercial engines, designed for broad genre compatibility, incorporate numerous general-purpose modules that can introduce abstraction overhead, potentially reducing raw performance in niche contexts. Conversely, a custom-built engine enables architects to streamline the system by removing irrelevant features and concentrating resources on domain-specific requirements. For instance, a dedicated Bullet Heaven engine can prioritize highly optimized rendering loops, lightweight collision detection, and object pooling to efficiently manage thousands of simultaneous projectiles, foregoing the need for full 3D physics (Politowski et al., 2020). The findings by Politowski et al. emphasized that while commercial engines facilitate rapid prototyping, custom engines provide higher freedom to fine-tune execution speed and design specialized workflows that precisely meet the demands of the intended genre.

Bullet Heaven Genre

The "bullet heaven" genre is closely associated with games like Vampire Survivors. Games similar to Vampire Survivors are sometimes miscategorized as bullet hell. A distinguishing feature of traditional bullet hell is the requirement for the player to evade a large volume of projectiles. Conversely, games in the bullet heaven genre reverse this mechanic, focusing on the player firing a substantial number of projectiles and generating attacks to defeat waves of enemies (Emelianov, 2024; Larsson, 2023). A game engine designed for this genre benefits from being tailored to these unique demands by providing a user-friendly and visual interface based on robust user interface design principles. These games require precise collision detection to accurately register player hits. Furthermore, performance optimization is critical for maintaining smooth gameplay when the screen is populated with a large number of entities (Davidson, 2020).

Procedural Content Generation

Procedural noise maps can be utilized to generate varied terrains by simulating height, moisture, and heat. Perlin noise, a type of gradient noise, is capable of generating complex patterns with minimal computational resources (Buckley, 2023). The implementation of Perlin noise involves generating a 2D float matrix with values normalized between 0.0f and 1.0f, which can be visualized as a grayscale image (Silveira, 2023). This method supports the creation of varied terrains and biomes by normalizing the noise map to define ground and empty points and by adjusting the scale parameter. Simplex noise is a later refinement of Perlin noise, developed to reduce computational complexity in higher dimensions and minimize directional artifacts (Xie, 2025). The performance advantage of Simplex noise becomes more pronounced as the number of dimensions increases.

Procedural textures are created using mathematical models and can simulate a variety of surface textures. These methods can be divided into structured and unstructured texture generation (Dong et al., 2020).

The advantages of procedural content generation methods can reduce development time and costs by up to 40% and can create products with higher replayability. More than 1,700 games on Steam feature content generation (Pereira et al., 2021).

Collision Detection Techniques

Collision detection algorithms depend on the types of shapes colliding, such as rectangle to rectangle or circle to circle. Common techniques include axis-aligned bounding boxes for simple rectangles and circle-circle collision for circles. For collision detection between any two circles, the circle-circle collision can be used by taking the center points of the two circles and ensuring the distance between the center points are less than the two radii added together (Mozilla Contributors, 2024). The bounding box for an object is typically determined by its 2D image and texture (Mileff, 2023). This can be calculated very simply when loading, since the width and height of the texture image will be in the case that the sprite is drawn correctly and does not contain unnecessary transparent pixels on the edges.

Flocking Algorithm in AI

The integration of artificial intelligence (AI) can make game testing more organic and allow the system to respond to unpredictable events by analyzing on-screen occurrences (Netlis-Galejs, 2023). One method is the Flocking algorithm, which simulates the collective behavior of birds by applying three core rules: separation (avoiding collision with neighbors), alignment (matching the movement of nearby entities), and cohesion (moving toward the average position of neighbors)(Garrigan, 2021).

Performance Metrics: Frame Rate vs Frame Time

While frame rate (FPS) is a common measure of performance, frame time provides a more accurate reflection of the player's experience by measuring the consistency of frame delivery (Bayley, 2025). Smooth and stable gameplay is generally prioritized over a high but inconsistent frame rate. Frame time reflects the stability of frame delivery for gameplay smoothness, whereas FPS gives only a numerical average. By analyzing frame time graphs alongside CPU and GPU utilization, it must be as close to a flat line as possible. Most players prefer a lower yet consistent frame rate over a higher but erratic one, making frame time a more reliable metric than FPS alone, providing insights into pacing, stability, and overall playability in gaming.

It's important to look at both frame rate and frame time when evaluating a game (Roach, 2022). Frame rate reflects the general performance of a game, whereas frame time provides insight into the actual gameplay experience. Frame time refers to the spacing between frames, and even at 60 FPS, uneven intervals from the ideal 16.6 ms can lead to noticeable stutter. A higher frame time values correspond to lower frame rates, while lower frame times indicate smoother performance.

The Philippines Gaming Industry

The gaming industry in the Philippines has the potential to be a major player in the entertainment industry, providing leisure, education, and career opportunities (Guevara, 2023). The local game development sector is expected to grow. The Philippines is recognized as a rapidly developing business with more than a billion players globally (CIIT College of Arts and Technology, 2022).

The Philippine game development sector has shown strong growth, attracting foreign companies to establish video game operation centers in the country. Philippine game development studios generated USD 9.20 million in sales revenue at Nordic Game 2024 (Department of Trade and Industry Philippines, 2024).

Online game addiction is recognized by the World Health Organization as a mental health condition. Research by Labana et al. (2020) found that online game addiction was positively correlated with depression among adolescents in Manila, with the participants reporting playing online games as a coping mechanism for emotional distress (Labana et al., 2020).

The stigma around gaming is seen as a waste of time, which hinders the industry's progress at household and institutional levels. This perception limits institutional support and funding and restricts understanding of the industry's potential for job creation and economic growth. Limited budget in purchasing gaming hardware is a common issue. The lack of institutional support by the Philippine government does not fully recognize the gaming industry, resulting in a lack of support and funding for gaming companies. Local esports teams struggle to acquire sponsorships compared to online influencers (Guevara, 2023).

There is also a limited market for video games in the Philippines due to the expense of video games, making them accessible only to the middle and upper classes (Cerda, 2021).

Limited funding opportunities also lead to a deficiency of local capital for video game projects (Ta-asan, 2024). While governments in other countries allocate budgets to support game development, the Philippines provides limited or no such funding, often resulting in small-scale, Filipino-made video games that are typically released as mobile games. This deficit also contributes to a lack of publishers in the local game development industry, complicating the search for funding for developers.

Furthermore, the dominance of the Western video game industry has constrained the creation of original Filipino video games, as many local developers instead outsource work to major international companies (Cerda, 2021).

Relatedly, studies have categorized homeworkers in the Philippines as industrial outworkers (assembling goods for factories) or online workers (providing services via telecommunications). Motivations for homework include childcare and family care, especially for women, while men have increasingly pursued online work due to its flexible schedule and higher pay (King-Dejardin, 2021).

Gamification in Education

Games can be used as a means of education or a source of information. A generation that mostly owns a cell phone a different source can be used as a learning environment. Mobile game-based learning can be effective in raising awareness (Caliston, 2025).

A study by Sison et al. (2023) developed a 2D game application for learning networks and security, with positive feedback from BSIT students who found the gamification effective for learning networking concepts and being aware of network threats. Interactive learning experiences in games can motivate students, give them control over their learning, and provide hands-on experiences.

The experiences of Tuason (2020) illustrate many of the common struggles inherent in indie game development in the Philippines. Tuason's account detailed a personal journey through success and failure, highlighting numerous challenges. Throughout this process, Tuason faced difficulties in balancing work and family life, responding to the pressures of a shifting game market, and maintaining motivation. To overcome these hurdles, the developer experimented with various techniques, adopted strict work routines, and implemented significant lifestyle changes. External crises, such as a volcanic eruption and a nationwide quarantine, further disrupted work. Despite these hardships, Tuason's narrative ultimately serves as a testament to perseverance within the realities of indie game development.

System Usability Scale (SUS)

The GitLab Handbook underscores the importance of the System Usability Scale (SUS) as a key tool for assessing the long-term usability of their product. This 10-item standardized survey, featuring both positively and negatively worded statements on a five-point Likert scale, is regularly deployed to gather usability feedback from programs (DeSanto, 2024).

Some researchers have argued that removing the neutral midpoint forces participants to take a stance; however, evidence from usability research shows that the presence or absence of a neutral option generally has little impact on overall results. Large-scale studies, such as those by Lewis and Sauro (2023), demonstrate that mean scores and conclusions remain virtually unchanged whether a neutral response is available or not. What often gets overlooked, however, is that eliminating neutrality can actually introduce more confusion, particularly for respondents who genuinely lack a strong opinion. By forcing them to select "agree" or "disagree," surveys risk capturing artificial attitudes rather than authentic perceptions, thereby adding noise to the data rather than clarity. Therefore, retaining a neutral option preserves the integrity of participant choice and ensures the results accurately reflect the range of user experiences.

Previous research had effectively utilized procedural generation, flocking algorithms, and Separating Axis Theorem (SAT)-based collision detection, though these methods were largely studied in isolation. By integrating these components into a single game engine, the researchers recognized the potential to develop a 2D bullet-hell engine that was more adaptable and customizable. This approach offered both technical advancement and greater accessibility for developers by creating opportunities to balance accuracy and performance. However, a critical gap existed: there was no established performance benchmark for the specific combination of the Flocking algorithm, Axis-Aligned Bounding Boxes (AABB), Circle-Circle detection, Perlin Noise, and Simplex Noise when integrated into one system.

The study aimed to create a specialized game engine that allows novice game developers to create 2D bullet heaven games in top-down perspective while finding out what combination of procedural generation algorithms (Simplex or Perlin noise), collision detection algorithms (axis-aligned bounding boxes or circle-circle collision), and enablement or disablement of the flocking algorithms leads to the lowest frame time leading to higher frame rate (or frames per second). Specifically, this study benefits the following groups.

Novice Game Developers: They are able to experiment with and create 2D bullet-heaven games using the engine and its provided graphical user interface.

Novice Game Engine Developers: They can use the open-source code as a basis or reference for their own engines. Furthermore, seeing the performance data will allow them to design their own game engines with better frame time and frame rate stability.

Future Researchers: They can expand upon this study by using different procedural generation and collision detection algorithms, allowing for the use of 3D game assets, or enabling the use of programming languages other than Java through language bindings.

Objectives of the Study / Statement of the Problem

The primary goal of this study was to develop Bala, a 2D bullet heaven game engine designed to allow novice game developers to create 2D bullet heaven games from a top-down perspective. Additionally, the study aimed to test how different combinations of algorithms impacted the frame time of the developed 2D bullet heaven game. To achieve these goals, the specific objectives were as follows:

Creation of a Specialized Game Engine: A game engine capable of creating 2D bullet heaven games in a top-down perspective was developed through a simple graphical user interface implemented using Java and the Lightweight Java Game Library (LWJGL).

Application of Procedural Generation: Simplex noise and Perlin noise were implemented for the procedural generation of backgrounds and non-player character (NPC) spawn points.

Application of Collision Detection: Axis-Aligned Bounding Boxes (AABBs) and circle-circle collision were implemented to enable accurate interaction between the player, their projectiles, and NPCs.

Application of Game Artificial Intelligence (AI): A flocking algorithm was implemented to allow NPCs to simulate flock behavior, preventing NPCs from colliding with each other while still aligning toward a common direction.

Evaluation of Game Engine Performance: Frame times were collected and analyzed through stress and performance tests applied to the game engine while using different combinations of procedural generation and collision detection algorithms, both with and without the flocking feature enabled.

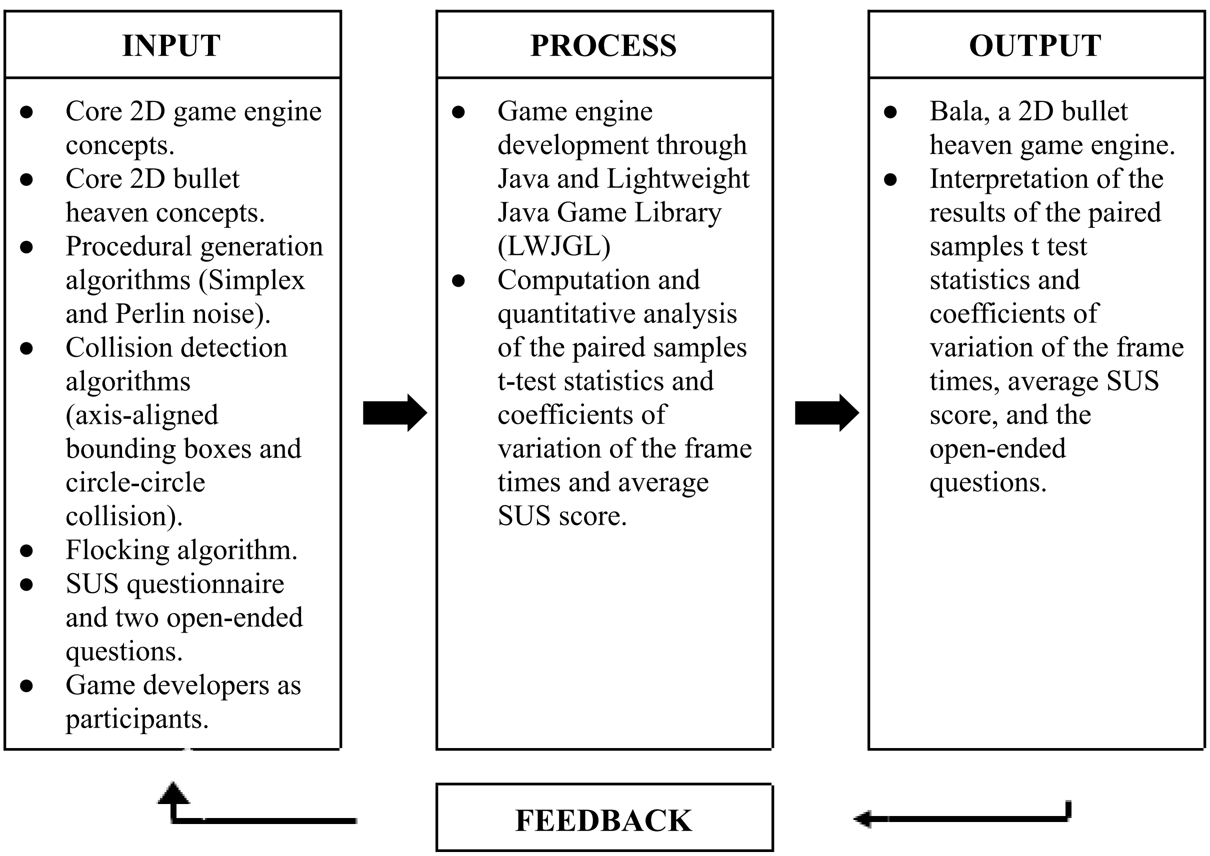
Assess the usability of the game engine. The average System Usability Scale (SUS) score and the responses to open-ended questions were calculated and interpreted from the SUS questionnaires administered to game developers who tested the engine.

Conceptual Framework

As illustrated in Figure 1, the Input-Process-Output (IPO) model is utilized to describe and display the flow of information (Adobe Experience Cloud Team, 2023) detailing the completion of this study. The IPO model is suitable for this research as it focuses on software development, which fundamentally relies on defining the necessary input before a program can be developed(IONOS Editorial Team, 2023).

Figure 1

IPO Model



The inputs of the study were composed of the core concepts and algorithms needed to develop a 2D bullet heaven game engine, as well as the participating game developers. These developers assessed the game engine by answering a System Usability Scale (SUS) questionnaire and two open-ended questions adapted from the GitLab SUS (DeSanto, 2024).

The actual development of the game engine, along with the computation and analysis of the frame times collected through stress and performance testing, the average SUS score, and the answers to the two open-ended questions, made up the process component of the IPO model. The output was the game engine itself, as well as the interpretations of the computed results. Finally, the IPO model includes a feedback component, as the research was developed using a Kanban workflow that implements a feedback loop through its testing or review phase before a task is deemed complete (Chervenkova, 2024).

Theoretical Framework

The development of Bala was anchored in two major theoretical foundations: Complexity Theory and Human-Computer Interaction (HCI) Theory. These theories guided the evaluation of algorithm efficiency and system usability, which were central to the study's objective of creating a specialized game engine for the bullet heaven genre.

Complexity Theory provides the basis for understanding the computational performance of the algorithms within the system. Games in the bullet heaven genre require the management of hundreds to thousands of on-screen entities; thus, algorithmic inefficiencies can lead to performance degradation. The study applied this theory in selecting and comparing different combinations of algorithms. For example, Circle-Circle collision is generally faster, while Axis-Aligned Bounding Boxes (AABBs) are more accurate. Similarly, Perlin and Simplex noise differ in implementation complexity and efficiency. Additionally, while the flocking algorithm enables emergent, lifelike group behaviors in non-player characters, it increases computational overhead, as each agent must process rules of separation, alignment, and cohesion in relation to others. This trade-off between realism and efficiency underscored how algorithmic choices directly affected the engine’s performance under heavy load.

Recent studies emphasize the importance of designing engines that are tailored to performance requirements. Politowski et al. (2020) reported that developers generally build game engines to "(a) better control the environment and source code, (b) learn about game engines, and (c) develop specific games." This highlights that specialized engines are often created to address performance and workflow needs better than general-purpose engines.

While performance ensures playability, usability determines whether the engine is effective for the users. Human Computer Interaction (HCI) Theory emphasizes usability as a function of effectiveness, efficiency, and user satisfaction, making it essential for a software designed for game development. The study adopted the System Usability Scale (SUS), which is widely validated in evaluating interactive systems (Schrepp et al., 2023). The study applied this framework to measure how intuitive and accessible the engine was, complementing the performance tests with a user-centered evaluation.

Together, complexity theory and HCI theory established the dual foundation of the study's theoretical framework. Complexity Theory addressed the algorithm and performance, while HCI Theory anchored the usability evaluation. This combination supported the study's objective of developing a specialized game engine for the bullet heaven genre.

Scope and Delimitations

The study focused on developing a game engine that solely creates 2D bullet heaven games in the top-down perspective, meaning it was not intended to be a general-purpose game engine for other genres such as visual novels or first-person shooters. The study was conducted during the first semester of the academic year 2025-2026. Participating game developers who assessed the engine were contacted exclusively online via text. The specific scopes (inclusions) and delimitations (exclusions) of the study were as follows:

Graphical User Interface (GUI). The game engine has a GUI with drag-and-drop capabilities to reduce extensive coding by allowing game developers to arrange objects via the mouse. The GUI of the game engine only handles terrain, enemy, player, and drop placement as well as procedurally generating terrain, generating a set amount of enemies, and playing, saving, clearing, and loading game scenes. The game engine also has a GUI for editing properties of game objects such as the editing of the position by numbers, and its colors through RGBA.

Procedural Generation. Built-in functions that implement Simplex noise and Perlin noise are available for game developers to use for quickly creating backgrounds and non-player character (NPC) spawn points.

Collision Detection. Built-in functions that implement Axis-Aligned Bounding Boxes (AABBs) and circle-circle collision are available for developers to call, ensuring proper interaction between objects such as the player and NPCs.

Game Artificial Intelligence. A built-in function implementing flocking is available for game developers to use for their NPCs so that they could interact properly.

Programming Language. The game engine was developed using Java and the Lightweight Java Game Library (LWJGL) which gives access to Open Graphics Library (OpenGL) for 2D graphics, Open Audio Library (OpenAL) for audio, and Graphics Library Framework (GLFW) for handling user input and creating windows (LWJGL, n.d). Therefore, game developers were only able to use Java to modify the game engine.

Minimum Software Requirements. The engine and games created with it require a computer running an operating system (Windows, Mac, or Linux) capable of installing and running Java Development Kit (JDK) (version 11 to 16) and Gradle (version 5.0 to 8.14) (Gradle, n.d.).

Minimum Hardware Requirements. Currently, the game engine has no set minimum requirements for the hardware but they should have a 64-bit CPU to be able to install and run JDK (version 11 to 16) and Gradle (version 5.0 to 8.14).

Game Assets Provided. The developers of the game engine used public domain assets during the development of the game engine that adheres to the Creative Commons license 0 which allows anyone to copy, modify, and distribute the assets without asking permission from the owners (Creative Commons, n.d.). The game developers can provide their own game assets, which include six spritesheets for the player, enemy, collidable terrain, non-collidable terrain, drops, and projectiles, as well as six audio assets consisting of background music, impact sound, shoot sound, pickup sound, enemy death sound, and player death sound, but the developers of the game engine are not liable for enforcing the legality of the assets provided and used by the game developers.

Online Capabilities. The game engine does not implement online capabilities such as online multiplayer.

Code Stepping in Debugger. The debugger in the game engine does not have code stepping which allows game developers to run the code line by line while debugging, though it does have basic debugging messages outputted in the terminal that contain information such as which function caused the error.

Algorithm Selection in GUI. The GUI of the game engine does not include an option to which algorithm the game engine uses. The GUI of the game engine does not handle editing of the settings of the engine.

GUI for editing bullet properties. The game engine does not have a GUI for editing projectile path, behavior, and properties.

Methods

Research Design

Descriptive research is used to identify why something happens, that is needed to understand how, when and where it happens (McCombes, 2023). The research design is developed so that the results are valid and reliable, specifically on survey research where it allows to gather large volumes of data that can be analyzed for frequencies, averages and patterns.

The research was descriptive because it aimed to describe the frame times produced by the various combinations of algorithms through the use of paired samples t-tests and coefficients of variation. The study did not attempt to explain the underlying causality for the observed frame times due to the lack of prior research examining the relational effects of these specific algorithms on frame time.

The study employed a quantitative research design to assess the performance and usability of the developed game engine, as this approach requires the collection, analysis, and interpretation of quantifiable data to test the research hypotheses (Ghanad, 2023). A Factorial design was also used, specifically a 2x2x2 design, because the study tested a combination of algorithms and configurations as independent variables to see which produced the least frame time as the dependent variable (Yang, 2023). The eight combinations were composed of procedural generation algorithms (Simplex or Perlin noise), collision detection algorithms (axis-aligned bounding boxes (AABBs) or circle-circle collision), and the flocking algorithm (enabled or disabled) are as follows.

Table 1

Factorial Design Table

Procedural Generation Algorithm	Collision Detection Algorithm	Flocking Algorithm (Enabled or Disabled)
Simplex Noise	AABBs	Enabled
		Disabled
	Circle-Circle Collision	Enabled
		Disabled
Perlin Noise	AABBs	Enabled
		Disabled
	Circle-Circle Collision	Enabled
		Disabled

Participants

The participants of the study consisted of six game developers who were knowledgeable in game engine development and fluent in English. Participants were selected using purposive sampling, specifically expert sampling, which is used when a research study requires participants with a specific set of knowledge regarding a topic (Nikolopoulou, 2023). In this study, the necessary expertise was in game development and game engines.

Although the study prioritized professional game developers from the Philippines due to accessibility, participation was not limited to Filipino developers. This decision was made considering the statement by the Game Developers Association of the Philippines (GDAP) president, Alvin Juban, who noted that professional Filipino game developers were "still few and far between" and that the industry required at least 2,000 more professionals in response to expansion driven by the COVID-19 pandemic (Rodriguez, 2020).

Research Instrument

For performance testing, the study utilized a built-in Java logger to measure the frame time of the game engine without influencing the results by recording the time provided by the GLFW clock. All performance tests were conducted on a single computer to eliminate the influence of extraneous factors, such as hardware differences, on the results.

To assess the game developers' experience while using the engine, a System Usability Scale (SUS) questionnaire was disseminated online to the participating developers. This instrument was modeled after GitLab's SUS questionnaire (DeSanto, 2024) and was composed of ten 5-point Likert questions. The questionnaire used the following ten items, all of which were presented in a positively-oriented manner for ease of participant comfort.

1. Bala is frequently usable.
2. Bala is simple and straightforward.
3. Bala is easy to use.
4. Bala can be used without technical support.
5. The various functions in Bala are well integrated.
6. Bala is consistent in its design and features.
7. Most people would learn to use Bala very quickly.
8. Bala is convenient and efficient to use.
9. Using Bala inspires confidence.
10. Bala can be used effectively with minimal learning.

Responses ranged from 1 (strongly disagree) to 5 (strongly agree). For correct application of the SUS formula, the answers to the even-numbered questions were reversed during scoring, as these items were modified from their standard negative orientation to a positive one.

Following the Likert scale questions, participating game developers answered two open-ended questions designed to elicit qualitative feedback: "What problems or frustrations have been experienced while using Bala?" and "What recommendations can be given for improving Bala?" The complete questionnaire is presented in Appendix D.

The SUS was chosen over other user experience (UX) questionnaires because it is considered the "gold standard for industry norms" and offers a robust interpretation framework, having been freely available since the mid-1980s (Sauro & Lewis, 2024). Furthermore, the SUS is reliable for the study's target sample size of 3 to 6 professional game developers, unlike alternatives such as the Usability Metric for User Experience-Lite Version (UX-Lite), which requires a larger sample size (Hodrien & Fernando, 2021; Tullis & Stetson, 2004, as cited in Sauro & Lewis, 2024).

To minimize response bias, the study retained the neutral midpoint in its usability scale. Research has shown that removing the neutral option does not substantially change the overall results, but it can force undecided participants to provide artificial leanings, thereby reducing data validity (Lewis & Sauro, 2023).

Data Collection

The primary source of quantitative data was acquired through stress and performance tests of the game engine and the SUS questionnaires assessing the game developers' experience. Performance tests were used to evaluate system performance by measuring attributes such as response time (Cohen, 2024); the main attribute measured in this study was frame time, which shows the time required to process a single frame (Roach, 2022). Quantitative data from the participants were provided through the SUS scores collected via the ten 5-point Likert questions disseminated online using Google Forms (DeSanto, 2024).

Additional open-ended questions were included at the end of the survey to gather qualitative data regarding problems and recommendations for system improvement.

Data Analysis

To interpret the quantitative data effectively, the researchers used the following formulas:

1. The Paired Samples t-Test was used to check if the frame times between two combinations of algorithms were significantly different. Significance was determined by comparing the test statistic's equivalent p-value to the α=0.05 level of significance using Jamovi; a p-value less than 0.05 indicated a significant difference (Fransson & Hermansson, 2023; Kent State University, 2025). The researchers employed the two-tailed version of the test to identify any significant difference, whether positive or negative, between the paired combinations' frame times (Birkett, 2024).

Figure 2

Paired Samples t Test Formula

$$\boxed{t = \frac{\underline{x} - \underline{y}}{\sigma \div \sqrt{n}}}$$

Note: $\underline{x}$ is the mean of the frame times of the first combination of algorithms. $\underline{y}$ is the mean of the frame times of the second combination of algorithms. σ is the standard deviation of the differences of frame times between the two combinations of algorithms. n is the equal sample size of the frame times between the two combinations of algorithms.

2. The Combination formula was used to calculate the number of possible ways of pairing the algorithm combinations for the paired samples t-tests. A combination is a way of selecting items from a collection where the order of selection does not matter **(**BYJUS, 2020).

Figure 3

Combination Formula

$$nCr = \frac{n!}{(n-r)!r!}$$

Note:

n *is the total number of algorithm combinations* (8). r *is the number of chosen samples* (2), *repr*

3. The Frame Rate (FPS) formula was used to conveniently estimate the frames per second given the mean frame time. While this metric provides an interval between rendered frames, it was noted that real-world variance in frame time is inevitable, with high spikes resulting in stuttering and user dissatisfaction (Koski, 2024).

Figure 4

Frame Rate Formula

$$FPS = \frac{1}{Frame\ Time\ (seconds)}$$

4. The Coefficient of Variation (CV) formula was utilized for measuring the stability of the frame times of a specific algorithm combination. A high CV corresponds to a choppy frame time, whereas a low CV corresponds to a smooth frame time, which is the preferred outcome for consistent gameplay (Tundrowalker, 2020).

Figure 5

Coefficient of Variation Formula

$$CV = \frac{\sigma}{\mu}$$

Note: σ is the standard deviation of the frame times of a combination of algorithms. And μ is the mean of the frame times of a combination of algorithms.

4. The Average System Usability Score formula was applied to interpret the user experience (UX) of the participating game developers. The resulting average SUS score corresponds to a particular grade and adjective, as defined by established standards (DeSanto, 2024), which are presented in Table 2**.**

Figure 6

Average System Usability Score Formula

$$\underline{SUS\ Score} = \frac{\sum_{i=1}^{n}((x_i - y_i + 20) \times 2.5)}{n}$$

Note: n is the number of participating game developers. x is the total score of odd numbered questions for a game developer. And y is the total score of even numbered questions for a game developer.

Table 2

Average SUS Score Interpretation

Average SUS Score	Grade	Adjective
84.1 – 100	A+	Best imaginable
80.8 – 84.0	A	Excellent
78.9 – 80.7	A-	
77.2 – 78.8	B+	
74.1 – 77.1	B	
72.6 – 74.0	B-	
71.1 – 72.5	C+	Good
65.0 – 71.0	C	
62.7 – 64.9	C-	
51.7 – 62.6	D	OK
25.1 – 51.6	F	Poor
0 – 25.0	F	Worst Imaginable

Note: Grades from A- to B- are classified as *Good*, while grades from C to C- are classified as *OK*.

Ethical Considerations

The researchers ensured the privacy and confidentiality of the participants in accordance with the Data Privacy Act of 2012. This was accomplished by not disclosing personal details, such as names and contact information, to the public without explicit permission. Participants retained the right to withdraw from the study at any point and remove any data they had provided.

Research Procedure

Planning. In the planning phase, the researchers identified the objectives, scope, and delimitations of the study. It was also at this stage that the researchers identified and searched for potential game developers and prepared the System Usability Scale (SUS) questionnaire that would be distributed to them. The hardware and software required to create and evaluate the game engine were also considered.

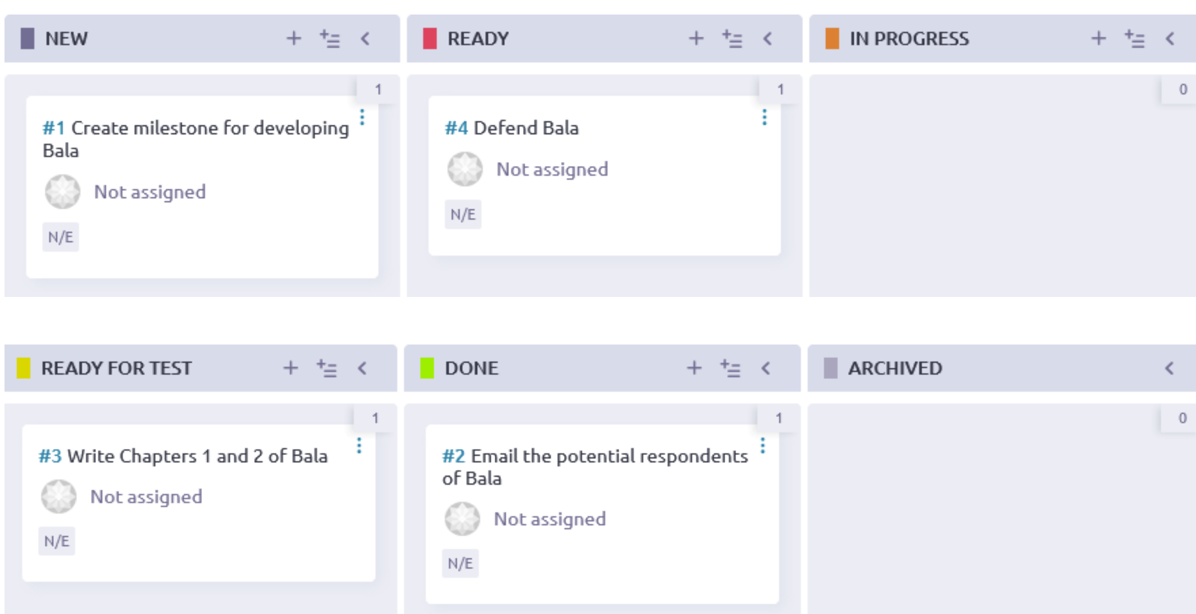
Designing. Guided by the objectives, scope, and delimitations identified in the planning phase, the researchers designed the initial structure of the game engine through a simple wireframe. It was during this stage that the graphical user interface (GUI) of the game engine was conceptualized.

Development and Implementation. In this phase, the game engine was developed and evaluated by the researchers using Kanban, an agile methodology that emphasizes visualizing workflows, limiting work-in-progress, and fostering continuous improvement (Shubhammodi1403, 2024). This methodology helped the researchers efficiently manage tasks and minimize inefficiencies. The researchers used Taiga, which provided a free and open-source platform for implementing Kanban online via a web browser (Taiga, n.d.). This platform allowed the researchers to collaborate online, and all actions were logged to accurately track progress.

The Kanban board utilized had six columns, as illustrated in Figure 6. The “NEW” column contained tasks that have been recently created while the “READY” column contained tasks that the researchers started. Meanwhile, the “IN PROGRESS” column contained tasks that were currently being done by the researchers. The “READY FOR TEST” column contained tasks that were currently finished but not yet checked. On the other hand, the “DONE” column contained tasks that have been checked and were finished. Lastly, the “ARCHIVED” column contained tasks that have been in the “DONE” column after the last major milestone. It could also contain discontinued tasks. Each task was assigned to a particular team member or the entire group. Taiga's logging of all actions done with the Kanban board allowed for accurate timekeeping.

Figure 7

Taiga Kanban Board



Data Treatment

Regarding the quantitative data collected from the System Usability Scale (SUS) questionnaire, all questions were set to required fields in Google Forms to prevent missing data. As mentioned in the statistical treatment section, the individual SUS scores were averaged to determine the overall usability result for the game engine.

The frame times collected from the performance tests were inputted into paired t-tests to check for significant differences among the eight algorithm combinations. To assess the influence of outliers, the researchers utilized the coefficient of variation (CV) to measure how closely the frame times clustered around the average, thereby indicating the stability of the frame time (Tundrowalker, 2020).

Algorithm Selection

The researchers selected a total of five algorithms for use in the game engine's development. Two were designated as procedural generation algorithms, two were for collision detection, and one was a behavior simulation algorithm intended to complement the Non-Player Characters (NPCs) in the bullet heaven game environment. These algorithms were as follows:

Circle-Circle Collision. This algorithm was chosen for its speed, as it calculates collision by comparing the distance between the two circular hitboxes' center points with the sum of their radii. A collision is detected if the distance is less than the sum of the radii (Mileff, 2023; Mozilla Contributors, 2024)

Axis-Aligned Bounding Boxes (AABBs). This algorithm was selected for its accuracy compared to Circle-Circle collision, as it allows for varying lengths and widths (rectangles) instead of fixed-radius circles. Collision is detected when no gap exists between all the sides of the two rectangles(Mileff, 2023; Mozilla Contributors, 2024).

Perlin noise. This algorithm was used for procedurally creating 2D backgrounds and spawn points. It was selected due to its simplicity (Shen, 2022) and minimal data storage requirements (Andrian et al., 2023).

Simplex noise. Similar to Perlin noise, this algorithm was used to procedurally create 2D backgrounds and spawn points. It was chosen because it is generally faster than Perlin noise and produces fewer artifacts (GarageFarm.NET, 2025; Peters, 2021), despite being more complex to implement.

Flocking. This algorithm was used to simulate the behavior of animal flocks, such as birds and fish (Failes, 2022). It was implemented to ensure that NPCs, particularly enemies, would not collide with or cover one another.

Algorithm Application

Collision detection. The Circle-Circle collision and AABBs were used to ensure the game engine recognized when objects collided, which was necessary to update player and enemy health and manage enemy positioning (complementing the Flocking algorithm). The Circle-Circle algorithm required circular hitboxes, which was less accurate than the rectangles used by AABBs. However, AABBs was theoretically slower because it requires four comparisons for two objects, while Circle-Circle collision requires only one comparison (Mozilla Contributors, 2024).

Procedural generation. The Perlin and simplex noise were used to procedurally create backgrounds which are natural looking (Andrian et al., 2023; Shen, 2022). It is also used unconventionally in real time by using them to create spawn points that could be for enemies or items that players could pick up.

Game artificial intelligence. The Flocking algorithm was implemented to simulate flock behavior. Specifically, the study implemented only the separation steering behavior, which prevents two NPCs from colliding (Failes, 2022; Garrigan, 2021). The remaining two steering behaviors, alignment and cohesion, were not implemented because NPCs in the bullet heaven genre typically maintain a constant target which is the player.

Testing

The main testing phase of the game engine consisted of two sequential parts:

Stress Testing. The game engine underwent stress testing in a total of eight tests for all combinations and configuration of algorithms presented in the research design by creating 75 enemies per second, which is equivalent to 1 enemy per frame based on the testing monitor's refresh rate. This was done until the game engine crashed or was unresponsive. The last number of enemies where the game engine still produced 30 frames per second are considered for next performance tests as it is often used as the preferred threshold because it represents the minimum level of acceptable playability in real-time applications (Needham, 2024). At this frame rate, the engine had approximately 33.33 milliseconds per frame to process all core tasks like logic, input, rendering, and updates. It served as a baseline standard for determining when performance degrades to the point of being noticeably disruptive (Elephantom, 2024) wherein the number of enemy units increases over time until the game engine can't handle it anymore. This is to determine the maximum number of enemy units the game engine can handle for each of the eight combinations and configuration of algorithms (Cohen, 2024). Afterwards, the maximum number of enemy units the game engine can handle for each combination and configuration of algorithms were used in the next test which is performance testing.

Performance Testing. The game engine underwent performance testing in a total of eight tests, one for each algorithm combination. Unlike in stress testing, the game engine immediately rendered the maximum number of enemy units identified during the stress test to accurately measure the frame time under peak load. The resulting frame times were then subjected to the paired samples t-test and coefficient of variation analysis, as mentioned in the statistical treatment section, to determine stability and significant differences between the combinations. Frame times were measured as averages over 2,000 frames (or fewer if the engine crashed or became unresponsive while handling the maximum stable enemy count). This process was conducted to check the consistency of Bala's performance under load.

Lastly, the game engine underwent bug testing during development by the researchers to find and fix bugs and also underwent user testing by the participating game developers to answer the provided system usability scale (SUS) questionnaire provided online through Google Forms.

Results

The Factorial Design was used to test the performance and usability of the game engine with the combination of algorithms and configurations on a 2x2x2x design. As a result, these independent variables were tested to produce the least frame time. Table 3 shows the computer specifications used in the stress and performance testing.

Table 3

Computer Specifications Used

Component	
CPU	AMD Ryzen 3 3100, 3.6 GHz
GPU	AMD Radeon RX 550 (Polaris 12/Lexa Pro), 4GB GDDR5
Memory	Kingston KHX2666C, 1x8GB DDR4, 2.6 GHz
Disk	HKVSN HS-SSD-C100, 240GB, 6.0GB/s
Monitor Resolution	1920 x 1080
Monitor Refresh Rate	75Hz
Operating System	Microsoft Windows 10 Professional (x64) Build 19045.6216
AMD Driver Version	31.0.12029.10015

The study conducted stress testing on each algorithm combination by generating 75 enemies per second (equivalent to one enemy per frame, given the testing monitor's refresh rate) until the engine froze or crashed. The results indicated that the combination of Simplex noise, Axis-Aligned Bounding Boxes (AABBs) collision detection, and flocking disabled (SBnF) yielded the best performance, successfully rendering 2,025 enemies while maintaining at least 30 frames per second (FPS), as shown in Table 4. This resulting maximum enemy count of 2,025 was subsequently used for the performance tests. Table 4 revealed that combinations where flocking was disabled yielded better results by at least a thousand enemies compared to those where flocking was enabled. Furthermore, combinations with flocking enabled showed the largest drops in frame rate between the last FPS measurement that was greater than or equal to 30 FPS and the next measured FPS, demonstrating greater instability under load.

Table 4

Stress Testing (Factorial Design Table)

Procedural Generation Algorithm	Collision Detection Algorithm	Flocking Algorithm (Enabled or Disabled)	Last FPS >= 30 FPS	Enemy Count of Last FPS >= 30 FPS	First FPS < 30 FPS	Enemy Count of First FPS < 30 FPS
Simplex Noise	AABBs	Enabled	34	300	17	375
		Disabled	30	2025	28	2100
	Circle-Circle Collision	Enabled	31	150	14	225
		Disabled	31	1950	29	2025
Perlin Noise	AABBs	Enabled	43	225	29	300
		Disabled	30	1875	28	1950

Circle-Circle Collision	Enabled	33	150	14	225
	Disabled	30	1800	29	1875

A Student's t-test was used to check for significant differences between the algorithm combinations because each had an equal number of samples (The VSNi Team, 2022). It was used to check if there were any significant differences between the combinations of algorithms with the frame times they produced.

Specifically, paired samples t-tests were conducted using Jamovi to compare the frame times produced by the different procedural noise measures under varying conditions of collision detection (AABBs or Circle-Circle collision) and flocking (enabled or disabled).The following naming convention was used for the algorithm combinations:

- P = Perlin noise
- S = Simplex noise
- B = Axis-Aligned Bounding Boxes (AABBs)
- C = Circle-Circle Collision
- nF = No Flocking
- F = Flocking

For example, PBnF referred to the combination of Perlin Noise, AABB collision detection, and no flocking, while SCF referred to Simplex Noise with Circle-Circle collision and flocking enabled.

The direction of the t-statistics indicated which condition produced higher frame times. Negative t-statistics suggested that the first combination had lower frame times than the second, whereas positive values suggested the opposite. A p-value less than 0.05

indicated a significant difference between their frame times. For example, PBnF produced significantly lower frame times than SCF ($t=57.900, p<.001$), as shown in Table 5.

As illustrated in Table 5, PBnF showed no significant difference in frame time only when compared to PCnF, with a t-value of −0.762. The PBnF combination exhibited the largest negative significant difference in frame time when compared with PCF ($t=-48.178$). It had the smallest negative significant difference when compared with PBF ($t=-5.315$), which was close to the t-value of −5.364 when compared to SBF.

The negative t-values with significant differences only occurred when PBnF was compared with combinations where flocking was enabled. This denoted that PBnF consistently produced a lower frame time (leading to a higher FPS) when compared to any combination that included flocking.

Conversely, when compared to other combinations where flocking was disabled, PBnF only had a lower frame time than PCnF, and this difference was not significant. This indicated that out of all combinations with flocking disabled, PBnF produced the highest frame time leading to lower frames per second.

Table 5

PBnF Paired Samples t Test Results

Paired Samples T-Test

			statistic	df	p
PBnF	**PBF**	**Student's t**	-5.315	99.0	<.001
	PCnF	**Student's t**	-0.762	1167.0	0.446
	PCF	**Student's t**	-48.178	84.0	<.001
	SBnF	**Student's t**	15.619	1999.0	<.001

SBF	**Student's t**	-5.364	100.0	<.001
SCnF	**Student's t**	13.574	1437.0	<.001
SCF	**Student's t**	-57.900	88.0	<.001

Note. Hₐ $\mu_{\text{Measure 1 - Measure 2}} \neq 0$

Table 6 demonstrated that the differences in frame times across algorithm configurations were largely significant. PCnF, SBnF, and SCnF all performed significantly better than PBF (p<.001). In contrast, SBF showed no meaningful difference (p=.229), while PCF and SCF performed significantly worse than PBF.

Table 6

PBF Paired Samples t Test Results

Paired Samples T-Test

			statistic	df	p
PBF	**PCnF**	**Student's t**	5.396	99.0	<.001
	PCF	**Student's t**	-3.100	84.0	0.003
	SBnF	**Student's t**	5.274	99.0	<.001
	SBF	**Student's t**	1.211	99.0	0.229
	SCnF	**Student's t**	5.468	99.0	<.001
	SCF	**Student's t**	-4.990	88.0	<.001

Note. Hₐ $\mu_{\text{Measure 1 - Measure 2}} \neq 0$

Table 7 showed that PCnF had significant differences when compared with all other algorithm combinations. Similar to earlier observations, PCnF consistently produced a lower frame time when compared to all combinations where flocking was enabled. Conversely, when compared to the remaining combinations where flocking was disabled, PCnF had a higher frame time, indicating a lower frames per second.

Table 7

PCnF Paired Samples t Test Results

Paired Samples T-Test

			statistic	df	p
PCnF	**PCF**	**Student's t**	-45.424	84.0	<.001
	SBnF	**Student's t**	2.283	1167.0	0.023
	SBF	**Student's t**	-5.432	100.0	<.001
	SCnF	**Student's t**	7.904	1167.0	<.001
	SCF	**Student's t**	-67.568	88.0	<.001

Note. H_a $\mu_{\text{Measure 1 - Measure 2}} \neq 0$

Table 8 indicated that PCF had significant differences in frame time when compared to the rest of the combinations. PCF was significantly slower even when compared to other combinations with flocking enabled, though comparisons to disabled-flocking combinations showed higher differences.

Table 8

PCF Paired Samples t Test Results

Paired Samples T-Test

			statistic	df	p
PCF	**SBnF**	**Student's t**	63.328	84.0	<.001
	SBF	**Student's t**	5.332	84.0	<.001
	SCnF	**Student's t**	40.399	84.0	<.001
	SCF	**Student's t**	2.054	84.0	0.043

Note. H_a $\mu_{\text{Measure 1 - Measure 2}} \neq 0$

Table 9 illustrated that SBnF had significantly lower frame times when compared to the remaining combinations where flocking was enabled. However, when compared to the remaining combination where flocking was disabled, SBnF had significantly higher frame times.

Table 9

SBnF Paired Samples t Test Results

Paired Samples T-Test

			statistic	df	p
SBnF	**SBF**	**Student's t**	-5.358	100.0	<.001
	SCnF	**Student's t**	6.047	1437.0	<.001
	SCF	**Student's t**	-42.202	88.0	<.001

Note. H_a $\mu_{\text{Measure 1 - Measure 2}} \neq 0$

Table 10 showed that SBF had no significant difference in frame times compared with SCF but did have a significantly higher frame time when compared to SCnF.

Table 10

SBF Paired Samples t Test Results

Paired Samples T-Test

			statistic	df	p
SBF	**SCnF**	**Student's t**	5.487	100.0	<.001
	SCF	**Student's t**	-1.503	88.0	0.136

Note. H_a $\mu_{\text{Measure 1 - Measure 2}} \neq 0$

Table 11 revealed that SCnF had a significantly lower frame time compared with SCF. Their corresponding test statistic was the largest among all pairs, powerfully illustrating the effect of enabling or disabling flocking.

Table 11

SCnF Paired Samples t Test Results

Paired Samples T-Test

			statistic	df	p
SCnF	**SCF**	**Student's t**	-83.762	88.0	<.001

Note. H_a $\mu_{\text{Measure 1 - Measure 2}} \neq 0$

Overall, the paired sample t-test revealed that the majority of the pairs showed statistically significant differences with p-values less than 0.001. Only 3 pairs were not significantly different which were PBnF vs PCnF (p = 0.446), PBF vs SBF (0.229), SBF vs SCF (p = 0.136).

As shown in Table 12, the highest variability was observed in the AABBs collision configurations with flocking enabled (CV>1.5 for both Simplex and Perlin noise). This indicated that AABBs combined with flocking produced highly unstable outcomes, with fluctuations that exceeded the mean values.

Lowest variability was observed in the non-flocking conditions with AABBs collision (CV ~0.52–0.57) and in Simplex noise with circle-circle collision with flocking enabled (CV = 0.53). These cases demonstrated stable and predictable outcomes.

Moderate Variability: Moderate variability occurred in the Perlin noise with Circle–Circle conditions (CV≈0.71–0.73), where outcomes were consistent regardless of flocking.

In the estimated FPS column, all combinations where flocking was disabled had at least 23 FPS, while all combinations where flocking was enabled at best only achieved 6 FPS.

Table 12

Coefficient of Variation (Factorial Design Table)

Procedural Generation	Collision Detection	Flocking Algorithm	Sample Size	Mean	Standard Deviation	Coefficient of Variation	Estimated FPS
Simplex Noise	AABBs	Enabled	101	196.83	323.41	1.64	5
		Disabled	2000	41.20	23.68	0.57	24
	Circle-Circle Collision	Enabled	89	163.55	86.09	0.53	6
		Disabled	1438	38.73	23.95	0.62	26
Perlin Noise	AABBs	Enabled	100	195.51	301.25	1.54	5
		Disabled	2000	43.14	22.51	0.52	23
	Circle-Circle Collision	Enabled	85	171.86	124.66	0.73	6
		Disabled	1168	39.05	27.77	0.71	26

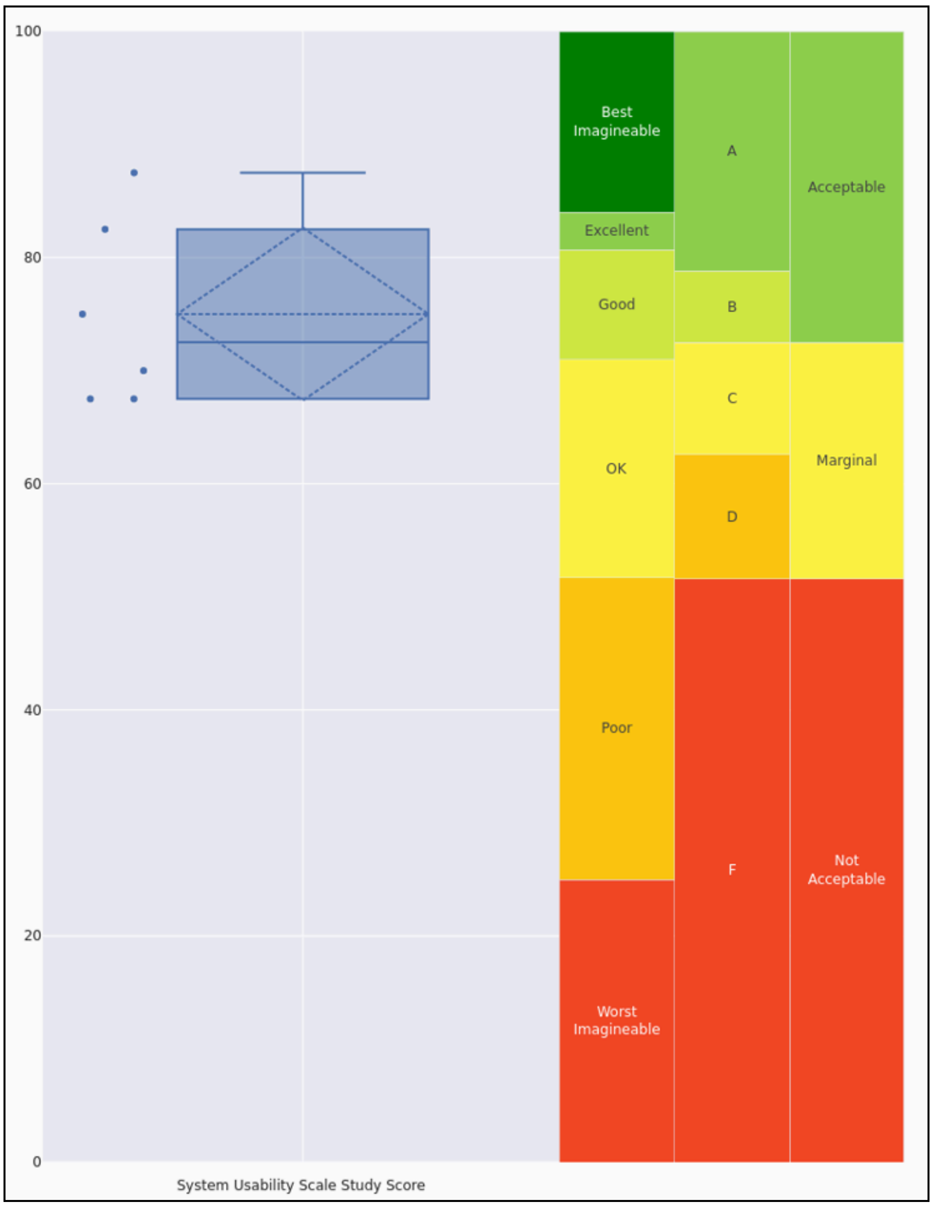
The System Usability Scale (SUS) questionnaire was distributed to evaluators using Google Forms to measure the usability of the game engine. After responses were gathered, the data was compiled and scored following standard SUS procedures.

The charts used in this study were generated through the web-based analysis toolkit for the System Usability Scale developed by Blattgerste et al. (2022) to present the SUS score, its corresponding percentile ranking, and the usability classification derived from the evaluators' feedback.

Figure 8 showed the distribution of individual participant scores around the median of 72.5, with most responses clustering in the acceptable range. The relatively compact distribution suggested consistent user experience across participants.

Figure 8

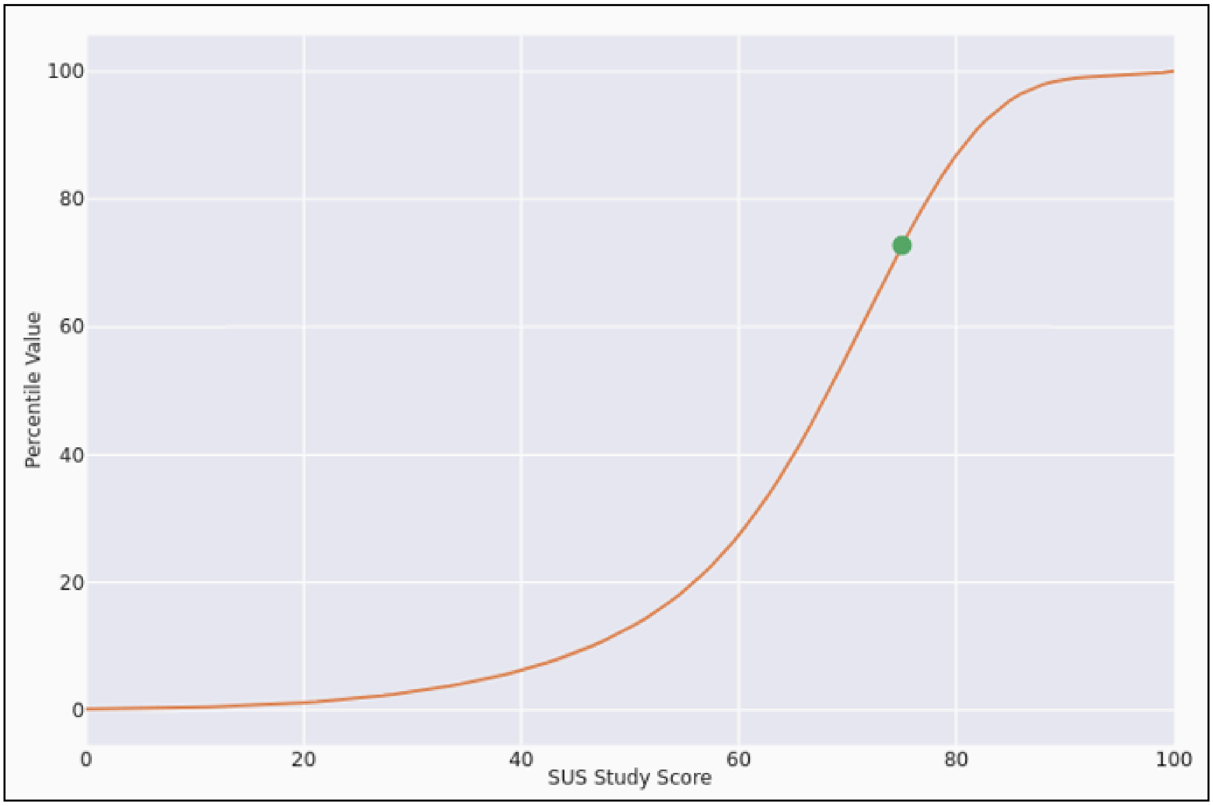
Box Plot



The overall usability surpassed a significant proportion of systems measured with the SUS, as the percentile plot placed the game engine above the 73rd percentile (see Figure 9). This result reinforced the finding that evaluators considered the system effective and reasonably user-friendly.

Figure 9

Percentile Plot



The Likert chart (Figure 10) indicated high agreement (67%–83%) for Questions 2, 4, and 6, which showed that users found the system easy to use and well-integrated. Questions 1, 3, 7, 9, and 10 received mixed responses, which suggested some inconsistency in user experience. The most concerning area was Question 8, which showed the most negative response (17% strongly disagree) and related to system complexity or the learning curve.

Figure 10

Likert Chart

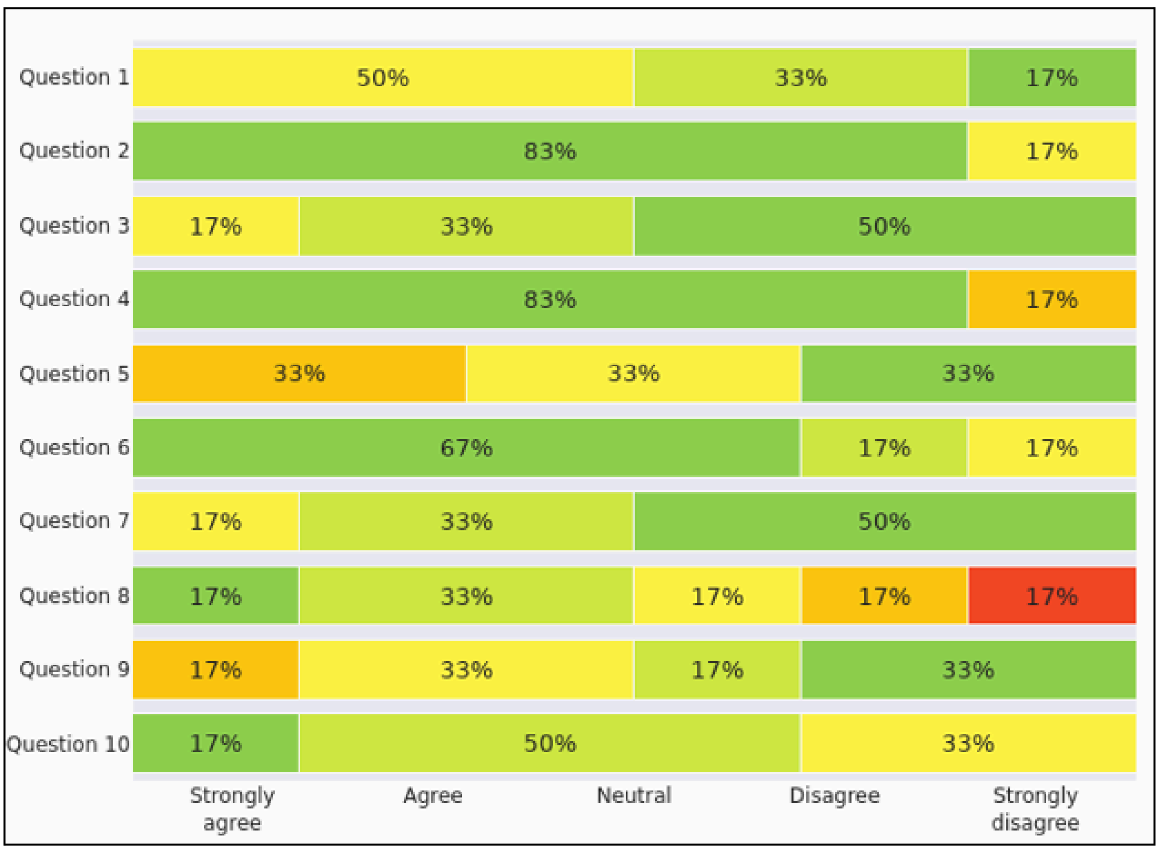
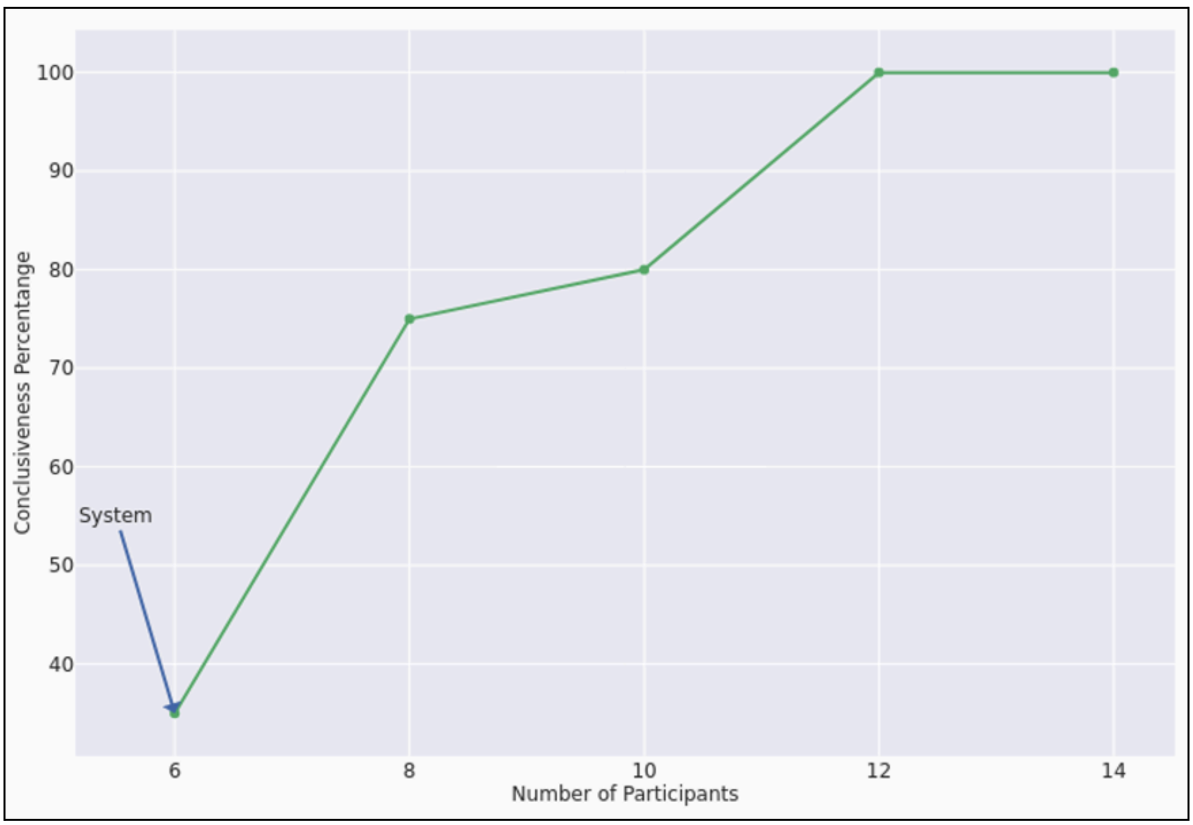


Figure 11 illustrated that conclusiveness of the results improved significantly with a higher number of participants, reaching nearly 100% conclusiveness with 12 or more participants. Although the evaluation involved only six participants, which was below the 12+ threshold commonly associated with near-100% conclusiveness, the results remain valid and provide meaningful insights into the system's usability. However, it is acknowledged that the relatively small sample size may introduce a higher degree of variability, and additional responses could further strengthen the conclusiveness of the findings.

Figure 11

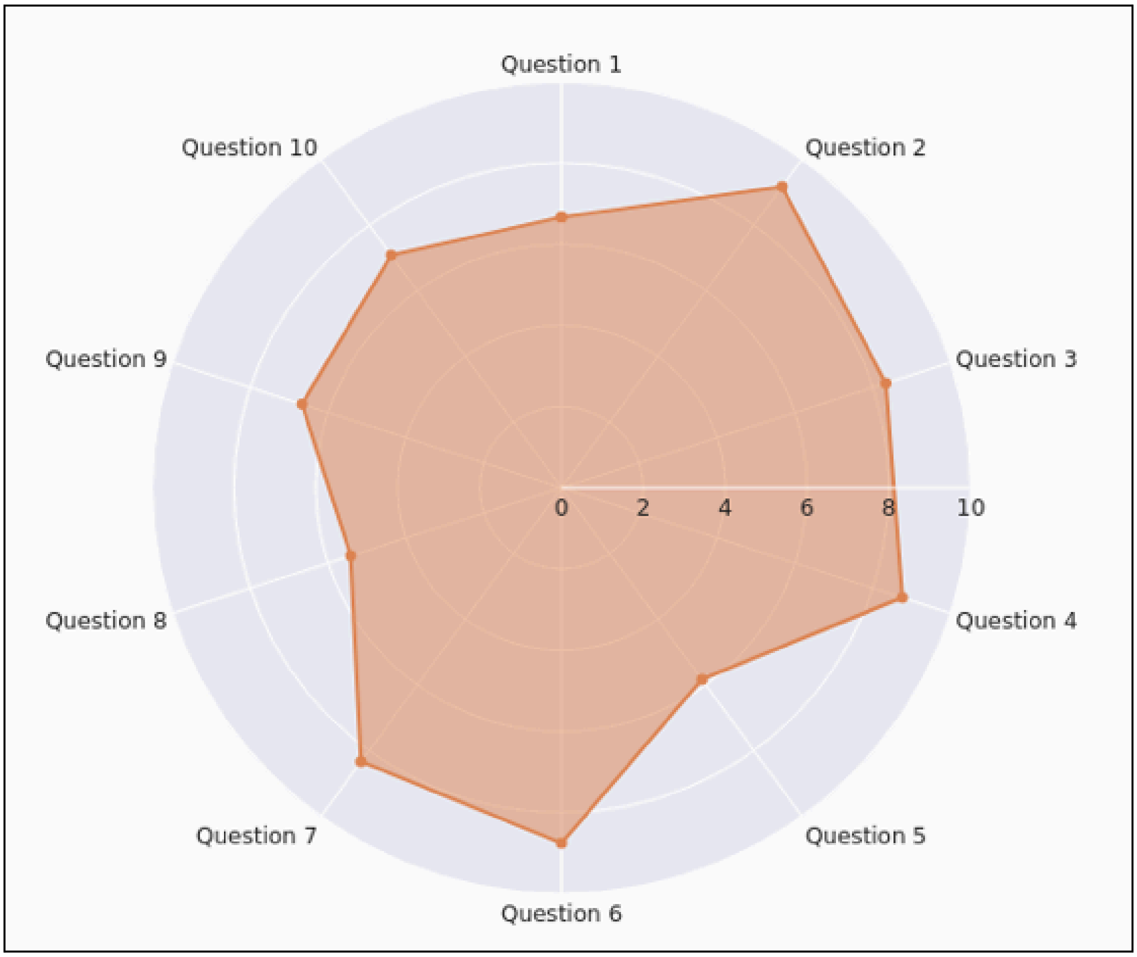
Conclusiveness Chart



The radar chart (Figure 12) provided a visual profile, showing that most questions scored in the 6-9 range. Questions 5 (Integration) and 8 (Convenience/Efficiency) were the weakest areas (around 5-6), while Questions 2 (Simplicity) and 4 (Technical Support) were the strongest (in the 8-9 range)..

Figure 12

Radar Chart



Discussion

The stress tests demonstrated that among all configurations, Simplex noise combined with Axis-Aligned Bounding Boxes (AABBs) and flocking disabled (SBnF) handled the test most effectively, managing to render 2,025 enemies while maintaining 30 frames per second (FPS). However, in the subsequent performance test, where 2,025 enemies were rendered immediately, the SBnF configuration was only able to achieve an estimated 24 FPS, 6 frames less than its sustained rate during the stress test. The coefficient of variation (CV) analysis revealed that disabling flocking consistently decreased the CV by at least 1 when AABBs was used, indicating greater frame time stability. When comparing collision detection methods with flocking disabled, AABBs produced lower CV values than Circle-Circle collisions in both Simplex and Perlin tests, confirming that AABBs was generally more stable and efficient under these conditions. Between the two procedural generation algorithms, Simplex and Perlin both performed reliably, but the lowest CV was observed in the Perlin noise using AABBs with flocking disabled ($CV=0.52$), establishing PBnF as the most stable configuration overall.

The analysis of the Paired Samples t-tests revealed that the majority of pairwise comparisons showed statistically significant differences ($p<.001$), indicating that most combinations of algorithms produced distinct performance results. For instance, PBnF differed significantly from PBF ($t=-5.315, p<.001$) and SCnF ($t=13.575, p<.001$). Similarly, SCnF and SCF demonstrated a very large, significant difference ($t=-83.8, p<.001$), underscoring the powerful impact of the flocking algorithm on performance.

A small number of comparisons did not yield significant differences. PBnF vs PCnF ($t = -0.762$, $p = 0.446$), PBF vs SBF ($t = -1.211$, $p = 0.229$), and SBF vs SCF ($t = -1.503$, $p = 0.136$) were statistically comparable. These cases suggest that in some conditions where both flocking is enabled or disabled, changing the collision detection algorithm or the procedural noise algorithm would not significantly alter the resulting frame time.

The System Usability Scale (SUS) results from six evaluators yielded an overall score of 75 and a grade of “B”. This places the game engine above the accepted benchmark of 68th Percentile, which represents average usability, as found by Jeff Sauro, PhD, and mentioned in the handbook by DeSanto (2024).

Looking at individual scores, Evaluator 6 rated the system highest at 87.5 ("Best Imaginable"), followed by Evaluator 2 at 82.5 ("Excellent"). This suggested that at least some users found the experience very smooth and efficient. Evaluators 1 and 3 gave scores of 75 and 70, respectively, which were above average but closer to the midline, indicating usable performance with minor points of friction. Evaluators 4 and 5 rated the system at 67.5, slightly below the average benchmark. This variation in scores highlighted that usability was not experienced uniformly across all evaluators.

The variation in scores indicates that while the system as a whole is performing well, certain aspects may not be equally intuitive for all users. It’s possible that differences in technical background, familiarity with game engines, or personal expectations influenced the evaluations.

Conclusion

The results of the paired samples t-tests demonstrate that procedural noise types, collision detection, and flocking yields statistically distinct outcomes. Perlin and Simplex noise, when combined with AABBs or Circle-Circle Detection and the presence or absence of flocking, produced significantly different results in most pairwise comparisons.

Only three pairings—PBnF vs PCnF, PBF vs SBF, and SBF vs SCF—showed no statistically significant differences, suggesting limited overlap between specific conditions. In contrast, the majority of comparisons revealed very strong differences ($p < .001$), indicating that these measures capture unique aspects of system behavior. Among them, the Simplex Circle with flocking (SCF) condition emerged as the most distinct, consistently producing results that diverged sharply from all other measures.

These findings confirm that the choice of procedural noise type and collision detection, as well as the presence of flocking, plays a decisive role in shaping outcomes.

The System Usability Scale evaluation of the game engine resulted in an overall System Usability Scale (SUS) score of 75, which is above the commonly cited benchmark of 68 for average usability (DeSanto, 2024). A score in the 70s generally suggests good usability, meaning evaluators were able to use the system effectively and with relatively low frustration, it also suggests that the system is generally usable and effective for its intended purpose. Individual evaluator scores, ranging from 67.5 to 87.5, indicate that while most users found the system easy to work with, there were variations in experience.

The higher scores demonstrate that the game engine has strong usability potential, with at least one evaluator rating it in the "Best Imaginable" range. The lower score points to specific areas where improvements can be made to ensure a more consistent user experience.

These findings highlight that while the current version of the engine is functional and well-received overall, refinements are needed to raise its usability into the excellent category and deliver a uniformly positive experience across different users.

The individual question results in the Likert chart reveal consistent strengths in ease of use, learnability, and user confidence. At the same time, questions regarding simplicity and the need for technical support received a higher rating. Questions about integration and design show more spread in responses, which indicates that some users found the game engine inconsistent or not fully cohesive in its features. Question 8, which addresses efficiency and convenience, stood out with mixed ratings, including 17% "Strongly Disagree," suggesting that while the system works, its workflows or interfaces may not feel optimal to all evaluators.

Recommendations

Based on the study's performance analysis and the usability feedback gathered from the evaluators, the following recommendations are proposed for future development of the Bala game engine.

Improvement of the debugging tools and the addition of error message pop-ups and error handling are recommended. The GUI should be enhanced with the addition of tooltips or specific properties editing of the assets, and the selection of algorithms must be made available through the interface/GUI instead of implementing changes inside the source code. Furthermore, developers should implement a game user interface for clear visual indicators and assign keybinds for intuitive and customizable controls to enhance overall usability. Additionally, an option for the game engine to be used as a library tool or framework should be implemented to be accessed easily by game developers.

Regarding future research and documentation, the conclusiveness chart suggested that with more participants, conclusiveness can reach nearly 100% with 12 or more participants. Although the number of participants was below the 12+ threshold, the results remain valid and provide meaningful insights into the system's usability. It is acknowledged that the relatively small sample size may introduce a higher degree of variability, and additional responses could further strengthen the conclusiveness of the findings.

Based on the recommendations of the evaluators, providing documentation, a tutorial video, or a guide can reduce the need for technical assistance and help users learn how to effectively use the Game Engine. Lastly, giving users the ability to export/build the game as an Executable and incorporating Quality of Life changes (such as tools to make the users' experience more efficient) are recommended.

References

Adobe Experience Cloud Team. (2023, July 19). *Learn how to use the input-process-output (IPO) model*. https://business.adobe.com/blog/basics/learn-about-the-input-output-model

Andrian, R., Permana, A. A., & Kusnadi, A. (2023). Design adventure role playing game with procedural content generation using Perlin noise algorithm. *Journal of Theoretical and Applied Information Technology*, *101*(9), 3653–3665.

Bayley, D. (2025). *Forget FPS. Here's why Frame Time is the PC gaming stat that matters most.* https://www.pcworld.com/article/2685112/forget-fps-heres-why-frame-time-is-the-pc-gaming-stat-that-matters-most.html

Birkett, A. (2024, June 13). One-tailed vs. Two-tailed tests (does it matter?). CXL. https://cxl.com/blog/one-tailed-vs-two-tailed-tests/

Blattgerste, J., Behrends, J., & Pfeiffer, T. (2022). A web-based analysis toolkit for the system usability scale. In *Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments* (pp. 237–246). Association for Computing Machinery. https://doi.org/10.1145/3529190.3529216

Buckley, D. (2023, November 3). *A guide to procedural generation*. GameDev Academy. https://gamedevacademy.org/procedural-2d-maps-unity-tutorial/

BYJUS. (2020). *Permutation and Combination Questions* https://byjus.com/maths/permutation-and-combination/

Caliston, N. P. (2025). Evaluating the effectiveness of mobile game-based learning for raising adolescent health awareness: The case of "AHlam Na 2.0." *Indonesian Journal of*

E-Learning and Multimedia, *4*(1), 11–20. https://doi.org/10.48550/ARXIV.2501.15047

Campos, S. A. E., Morales, B. A. M., & Nunez, A. A. V. (2022). Open-source game engine & framework for 2D game development. *2022 IEEE Engineering International Research Conference (EIRCON)*, 1–4. https://doi.org/10.1109/EIRCON56026.2022.9934816

Cerda, C. M. C. (2021). Towards a “Filipino” video game: Teaching Filipino culture and identity for video game development. *Transactions of the Digital Games Research Association*, *5*(3). https://doi.org/10.26503/todigra.v5i3.122

Chervenkova, M. (2024, August 13). *Feedback loops: How to do it the agile way*. Businessmap. https://businessmap.io/blog/feedback-loops

CIIT College of Arts and Technology. (2022, October 6). *Is game development a good career: Weighing the pros and cons*. https://www.ciit.edu.ph/is-game-development-a-good-career-pros-and-cons/

Cohen, N. (2024, March 19). *Performance testing vs. Load testing vs. Stress testing*. BlazeMeter by Perforce. https://www.blazemeter.com/blog/performance-testing-vs-load-testing-vs-stress-testin g

Creative Commons. (n.d.). CC0 1.0 Universal Deed. Retrieved September 3, 2025, from https://creativecommons.org/publicdomain/zero/1.0/

CryEngine. (2025). *Features*. https://www.cryengine.com/features

Davidson, E. (Director). (2020, June 2). *How to make a bullet hell game in Godot [P1]—Basic collisions* [Video recording].

https://www.youtube.com/watch?v=NMNl9Ix6ng8

Department of Trade and Industry Philippines. (2024, July 9). *Philippine game development studios generate USD 9.20 million in sales revenue at Nordic Game 2024*. https://www.dti.gov.ph/archives/news-archives/philippine-game-development-studios-generate-usd-9-20-million-sales-revenue-nordic-game-2024/

DeSanto, D. (2024, November 27). *System Usability Scale*. The GitLab Handbook. https://handbook.gitlab.com/handbook/product/ux/performance-indicators/system-usability-scale/

Dong, J., Liu, J., Yao, K., Chantler, M., Qi, L., Yu, H., & Jian, M. (2020). Survey of procedural methods for two-dimensional texture generation. *Sensors*, *20*(4), 1135. https://doi.org/10.3390/s20041135

Elephantom. (2024, January 21). How the framerate affects gameplay. https://elephantom.pl/en/2024/01/21/how-the-framerate-affects-gameplay

Emelianov, A. (2024, January 13). *Report: Bullet heaven as a separate genre*. Medium. https://medium.com/@anton.emelianov/bullet-heaven-fest-report-f337b7f1cfbd

Failes, I. (2022, April 7). *A history of CG bird flocking*. Befores & Afters. https://beforesandafters.com/2022/04/07/a-history-of-cg-bird-flocking/

Fransson, E., & Hermansson, J. (2023). *Performance comparison of WebGPU and WebGL in the Godot game engine* [Master's thesis, Blekinge Institute of Technology]. https://www.diva-portal.org/smash/record.jsf?pid=diva2:1762429

GameSensor. (2023, April 2). *Vampire Survivors sales soar as game generates almost $7 million in the first month on Steam*. https://gamesensor.info/news/vampire_survivors

GarageFarm.NET. (2025, March 3). *Perlin noise: Implementation, procedural generation,* *and Simplex noise*. https://garagefarm.net/blog/perlin-noise-implementation-procedural-generation-and-simplex-noise

Garrigan. (2021, August 13). *The flocking algorithm*. Codeheir. https://codeheir.com/2021/03/27/the-flocking-algorithm/

Ghanad, A. (2023). An Overview of Quantitative Research Methods. *INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH AND ANALYSIS*, *06*(08). https://doi.org/10.47191/ijmra/v6-i8-52

Glaiel, T. (2021, November 12). *How to make your own game engine (and why)*. Medium. https://medium.com/geekculture/how-to-make-your-own-game-engine-and-why-ddf0acbc5f3

Godot Engine. (2025). *Features*. https://godotengine.org/features/

Gradle. (n.d.). *Compatibility Matrix*. https://docs.gradle.org/current/userguide/compatibility.html

Guevara, R. (2023, February 27). *Gaming industry in the Philippines: Challenges & opportunities*. M2.0 Communications. https://m2comms.com/2023/02/27/gaming-industry-in-the-philippines-challenges-opportunities/

Gürler, F., & Onbaşioğlu, E. (2022). Applying perlin noise on 3D hexagonal tiled maps. *2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 670–673. https://doi.org/10.1109/ISMSIT56059.2022.9932712

Hodrien, A., & Fernando, T. (2021). A review of post-study and post-task subjective questionnaires to guide assessment of system usability. *Journal of User Experience*, *16*(3), 203–232.

IONOS editorial team. (2023, November 1). *What is the IPO model?* IONOS. https://www.ionos.com/digitalguide/server/know-how/ipo-model/

Kent State University. (2025, March 18). *SPSS tutorials: Paired samples t test*. https://libguides.library.kent.edu/spss/pairedsamplesttest

King-Dejardin, A. M. (2021). *Homeworking in the Philippines: Bad job? Good job?* International Labour Organization. https://hdl.handle.net/10419/263091

Koski, V. (2024). Benchmarking and reduction of open-source HTML5 game engine performance [Bachelor's Thesis, Lappeenranta-Lahti University of Technology LUT]. https://urn.fi/URN:NBN:fi-fe20241212101575

Labana, R. V., Hadjisaid, J. L., Imperial, A. R., Jumawid, K. E., Lupague, M. J. M., & Malicdem, D. C. (2020). Online game addiction and the level of depression among adolescents in Manila, Philippines. *Central Asian Journal of Global Health*, *9*(1). https://doi.org/10.5195/cajgh.2020.369

Larsson, A. (2023). *Defining a new game genre: Ontological approach to identify and define a new genre of games*. https://urn.kb.se/resolve?urn=urn%3Anbn%3Ase%3Ahis%3Adiva-23227

Lewis, J. R., & Sauro, J. (2023, April 3). Does removing the neutral response option affect rating behavior? MeasuringU. https://measuringu.com/removing-the-neutral-response-option/

LWJGL. (n.d.). *Lightweight Java game library*. Retrieved March 25, 2025, from https://www.lwjgl.org/

McCombes, S. (2019, May 15). Descriptive Research | Definition, Types, Methods &

Examples. *Scribbr*. https://www.scribbr.com/methodology/descriptive-research/

Mileff, P. (2023). Collision detection in 2D games. *Production Systems and Information Engineering*, *11*(3), 10–26.

Moiseienko, N. V., Moiseienko, M. V., Kuznetsov, V. S., Rostalny, B. A., & Kiv, A. E. (2023). *Teaching computer game development with Unity engine: A case study*. CEUR Workshop Proceedings. https://doi.org/10.31812/123456789/8486

Mozilla Contributors. (2024, December 19). *2D collision detection—Game development*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection

Needham, J. (2024, January 30). 30 FPS vs 60 FPS: Frame rate options in console gaming explained. Stuff. https://www.stuff.tv/features/30-fps-vs-60-fps-frame-rate-options-in-console-gaming-explained/

Netlis-Galejs, E. (2023, August 28). *Possible ways to automate in-game movements using AI & python*. TestDevLab. https://www.testdevlab.com/blog/possible-ways-to-automate-in-game-movements-using-ai

Nikolopoulou, K. (2023, June 22). *What is purposive sampling? | Definition & examples*. Scribbr. https://www.scribbr.com/methodology/purposive-sampling/

Park, H., & Baek, N. (2020). Developing an open-source lightweight game engine with DNN support. *Electronics*, *9*(9), 1421. https://doi.org/10.3390/electronics9091421

Pereira, L. T., Viana, B. M. F., & Toledo, C. F. M. (2021). Procedural enemy generation through parallel evolutionary algorithm. *2021 20th Brazilian Symposium on*

Computer Games and Digital Entertainment (SBGames), 126–135. https://doi.org/10.1109/SBGames54170.2021.00024

Peters. (2021, July 17). *Perlin vs. Simplex*. BIT-101 [2017-2023]. https://bit-101.com/2017/2021/07/perlin-vs-simplex/

Politowski, C., Petrillo, F., Montandon, J. E., Valente, M. T., & Guéhéneuc, Y.-G. (2020). Are Game Engines Software Frameworks? A Three-Perspective Study. arXiv. https://arxiv.org/abs/2004.05705

Poncle. (2022, October 21). *Vampire Survivors*. Steam. https://store.steampowered.com/app/1794680/Vampire_Survivors/

Poncle. (2025). *Vampire Survivors*. https://poncle.games/vampire-survivors

Prakash, S., Kumaraguru, G., Dharanidhar, A., & Kabilan, V. (2023). 2D game development using Unity engine "Minotscape." *International Journal of Scientific Research in Engineering and Management*, *07*(08). https://doi.org/10.55041/IJSREM25405

R Core Team (2025). R: A Language and environment for statistical computing. (Version 4.5) [Computer software]. Retrieved from https://cran.r-project.org. (R packages retrieved from CRAN snapshot 2025-05-25)

Roach, J. (2022, October 27). *What is frame time, and why is it so important in games?* Digital Trends. https://www.digitaltrends.com/computing/what-is-frame-time/

Rodriguez, B. (2020, August 4). *PH video game developers 'desperate' to hire more as industry levels up despite virus*. ABS-CBN News. https://www.abs-cbn.com/business/08/04/20/ph-video-game-developers-desperate-to-hire-more-as-industry-levels-up-despite-virus

Roedavan, R., Pratondo, A., Utoro, R. K., & Sujana, A. P. (2020). Zetcil: Game mechanic framework for Unity game engine. *IJAIT (International Journal of Applied Information Technology)*, 96. https://doi.org/10.25124/ijait.v3i02.2779

Sauro, J., & Lewis, J. (2024, March 19). *Is the SUS too antiquated?* MeasuringU. https://measuringu.com/is-the-sus-too-antiquated/

Schrepp, M., Kollmorgen, J., & Thomaschewski, J. (2023). A comparison of SUS, UMUX-LITE, and UEQ-S. *Journal of User Experience*, *18*(2), 86–104.

Shen, Z. (2022). Procedural generation in games: Focusing on dungeons. *SHS Web of Conferences*, *144*, 02005. https://doi.org/10.1051/shsconf/202214402005

Shubhammodi1403. (2024, May 15). *Kanban – Agile methodology*. GeeksforGeeks. https://www.geeksforgeeks.org/kanban-agile-methodology/

Silveira, G. (2023, April 12). *How I developed my own 2D procedural world generation*. Blog Ateliware. https://blog.ateliware.com/how-i-developed-my-own-2d-procedural-world-generation

Sison, A. A. R. C., Centeno, C. J., Mercado, M. A. S., Dapitan, L. R., Farparan, L. Z., & Gonzales, R. C. V. (2023). Development of a 2D game application for learning networks and security. *World Journal of Advanced Research and Reviews*, *18*(3), 1283–1289. https://doi.org/10.30574/wjarr.2023.18.3.1191

Ta-asan, K. B. (2024, January 30). *Struggling Philippine indie programmers looking for backers to step up their game*. BusinessWorld. https://www.bworldonline.com/top-stories/2024/01/30/572022/struggling-philippine-indie-programmers-looking-for-backers-to-step-up-their-game/

Taiga. (n.d.). *About us*. Retrieved March 25, 2025, from https://taiga.io/about-us/

The jamovi project (2025). jamovi. (Version 2.7) [Computer Software]. Retrieved from https://www.jamovi.org

The VSNi Team. (2022, June 13). What t-test to use: Student's or Welch's? VSNi. https://vsni.co.uk/what-t-test-to-use-students-or-welchs/

Tuason, N. (2020, July 7). *My 10 year game development journey*. Game Developer. https://www.gamedeveloper.com/business/my-10-year-game-development-journey

Tundrowalker. (2020, March 11). *Measuring FPS performance (beyond % lows)*. Mod DB. https://www.moddb.com/tutorials/measuring-fps-performance-beyond-lows

Unity. (2024). *Unity use cases*. https://unity.com/use-cases

Unreal Engine. (2025). *Features*. https://www.unrealengine.com/en-US/features

Xie, J. (2025). Analysis of Procedural Content Generation and Stylisation Techniques in Developing Video Games. Transactions on Computer Science and Intelligent Systems Research, 9, 450–455. https://doi.org/10.62051/0swzbh75

Yang, Y. L. (2023, January 4). *3.1: Factorial designs*. Statistics LibreTexts. https://stats.libretexts.org/Courses/Kansas_State_University/EDCEP_917%3A_Experimental_Design_(Yang)/03%3A_Between-Subjects_Factorial_Design/3.01%3A_Setting_Up_a_Factorial_Experiment

Appendices

Appendix A
Cover Letter

HOLY ANGEL UNIVERSITY SCHOOL OF COMPUTING

February 18, 2025

ALESSANDRA BETTINA S. GARCIA
Game Developer
Iterative Games

Dear Ms. Garcia:

Greetings of peace!
A contingent part of the requirements for the Bachelor of Science in Computer Science is 6CISTUDY1 – Independent Study Course. Students enrolled in the course are required to create a research paper that will lead towards the development of their Undergraduate Thesis Project: "*Bala: A 2D Bullet Heaven Game Engine*". Included in the course is the requirement to identify respondents for their study, including data gathering through interviews and surveys. The objective of the course is to provide the students with the opportunity to create a research paper that could help in the area they have chosen for their field of study.

The following students have expressed their intention to consider your prestigious institution for their project proposal as respondents:

Julian Mathew M. Guintu
Gabriel V. Pagcu
Aidan Zabdiel A. Serrano
Carlos Jose B. Ysais

Through this letter, I would like to seek your approval for the above undertaking. Rest assured that the information that will be given to them will be for academic purposes only and will be treated with utmost confidentiality. Thank you very much.

Sincerely,

MR. ULYSSES RAYMOND F. MONSALE
Adviser (6CISTUDY1)

ALESSANDRA BETTINA S. GARCIA
February 19, 2025

Noted by:

MRS. MA. LOUELLA M. SALENGA
Chairperson
Computer Science Program

DR. MARLON I. TAYAG
Dean
School of Computing

#1 HOLY ANGEL AVENUE, STO. ROSARIO, ANGELES CITY, PHILIPPINES 2009
TEL. NOS.: (045) 888-8691; 888-2902; 887-5748; 887-2455; 624-5277; 625-9619 | FAX: (045) 888-1754; 888-2514
EMAIL: HAU@HAU.EDU.PH | WWW.HAU.EDU.PH

HOLY ANGEL UNIVERSITY SCHOOL OF COMPUTING

July 15, 2025

JOHN ANTHONY SILVA
Game Developer
RLUX Studios

Dear Mr. Silva:

Greetings of peace!
A contingent part of the requirements for the Bachelor of Science in Computer Science is 6CISTUDY1 – Independent Study Course. Students enrolled in the course are required to create a research paper that will lead towards the development of their Undergraduate Thesis Project: "*Bala: A 2D Bullet Heaven Game Engine*". Included in the course is the requirement to identify respondents for their study, including data gathering through interviews and surveys. The objective of the course is to provide the students with the opportunity to create a research paper that could help in the area they have chosen for their field of study.

The following students have expressed their intention to consider your prestigious institution for their project proposal as respondents:

Julian Mathew M. Guintu
Gabriel V. Pagcu
Aidan Zabdiel A. Serrano
Carlos Jose B. Ysais

Through this letter, I would like to seek your approval for the above undertaking. Rest assured that the information that will be given to them will be for academic purposes only and will be treated with utmost confidentiality. Thank you very much.

Sincerely,

DR. RAQUEL B. RIVERA
Adviser (6CISTUDY2)

Noted by:

MR. ULYSSES RAYMOND F. MONSALE
Coordinator
Computer Science Program

John Anthony A. Silva
Game Developer
RLUX Studios

DR. MARLON I. TAYAG
Dean
School of Computing

#1 HOLY ANGEL AVENUE, STO. ROSARIO, ANGELES CITY, PHILIPPINES 2009
TEL. NOS.: (045) 888-8691; 888-2902; 887-5748; 887-2455; 624-5277; 625-9619 | FAX: (045) 888-1754; 888-2514
EMAIL: HAU@HAU.EDU.PH | WWW.HAU.EDU.PH

HOLY ANGEL UNIVERSITY SCHOOL OF COMPUTING

August 26, 2025

JOAQUIN PAOLO S. ANDAL
Front End Mobile Developer
WeMove Technology

Dear Mr. Andal:

Greetings of peace!
A contingent part of the requirements for the Bachelor of Science in Computer Science is 6CISTUDY2 – Independent Study Course. Students enrolled in the course are required to create a research paper that will lead towards the development of their Undergraduate Thesis Project: "*Bala: A 2D Bullet Heaven Game Engine*". Included in the course is the requirement to identify respondents for their study, including data gathering through interviews and surveys. The objective of the course is to provide the students with the opportunity to create a research paper that could help in the area they have chosen for their field of study.

The following students have expressed their intention to consider your prestigious institution for their project proposal as respondents:

Julian Mathew M. Guintu
Gabriel V. Pagcu
Aidan Zabdiel A. Serrano
Carlos Jose B. Ysais

Through this letter, I would like to seek your approval for the above undertaking. Rest assured that the information that will be given to them will be for academic purposes only and will be treated with utmost confidentiality. Thank you very much.

Sincerely,

DR. RAQUEL B. RIVERA
Adviser (6CISTUDY2)

Noted by:

MR. ULYSSES RAYMOND F. MONSALE
Coordinator
Computer Science Program

8-27-2025
DR. MARLON I. TAYAG
Dean
School of Computing

Joaquin Paolo S. Andal

#1 Holy Angel Avenue, Sto. Rosario, Angeles City, Philippines 2009
Tel. Nos.: (045) 888-8691; 888-2902; 887-5748; 887-2455; 624-5277; 625-9619 | Fax: (045) 888-1754; 888-2514
Email: hau@hau.edu.ph | www.hau.edu.ph

HOLY ANGEL UNIVERSITY SCHOOL OF COMPUTING

August 26, 2025

ARON JAKE S. RADAM
Game Developer

Dear Mr. Radam:

Greetings of peace!
A contingent part of the requirements for the Bachelor of Science in Computer Science is 6CISTUDY2 – Independent Study Course. Students enrolled in the course are required to create a research paper that will lead towards the development of their Undergraduate Thesis Project: "*Bala: A 2D Bullet Heaven Game Engine*". Included in the course is the requirement to identify respondents for their study, including data gathering through interviews and surveys. The objective of the course is to provide the students with the opportunity to create a research paper that could help in the area they have chosen for their field of study.

The following students have expressed their intention to consider your prestigious institution for their project proposal as respondents:

Julian Mathew M. Guintu
Gabriel V. Pageu
Aidan Zabdiel A. Serrano
Carlos Jose B. Ysais

Through this letter, I would like to seek your approval for the above undertaking. Rest assured that the information that will be given to them will be for academic purposes only and will be treated with utmost confidentiality. Thank you very much.

Sincerely,

DR. RAQUEL B. RIVERA
Adviser (6CISTUDY2)

Noted by:

MR. ULYSSES RAYMOND F. MONSALE
Coordinator
Computer Science Program

for 8-27-2025
DR. MARLON I. TAYAG
Dean
School of Computing

#1 Holy Angel Avenue, Sto. Rosario, Angeles City, Philippines 2009
Tel. Nos.: (045) 888-8691; 888-2902; 887-5748; 887-2455; 624-5277; 625-9619 | Fax: (045) 888-1754; 888-2514
Email: hau@hau.edu.ph | www.hau.edu.ph

HOLY ANGEL UNIVERSITY SCHOOL OF COMPUTING

August 26, 2025

CHLOE APACIBLE
Associate Software Engineer

Dear Ms. Apacible:

Greetings of peace!
A contingent part of the requirements for the Bachelor of Science in Computer Science is 6CISTUDY2 – Independent Study Course. Students enrolled in the course are required to create a research paper that will lead towards the development of their Undergraduate Thesis Project: "*Bala: A 2D Bullet Heaven Game Engine*". Included in the course is the requirement to identify respondents for their study, including data gathering through interviews and surveys. The objective of the course is to provide the students with the opportunity to create a research paper that could help in the area they have chosen for their field of study.

The following students have expressed their intention to consider your prestigious institution for their project proposal as respondents:

Julian Mathew M. Guintu
Gabriel V. Pagcu
Aidan Zabdiel A. Serrano
Carlos Jose B. Ysais

Through this letter, I would like to seek your approval for the above undertaking. Rest assured that the information that will be given to them will be for academic purposes only and will be treated with utmost confidentiality. Thank you very much.

Sincerely,

DR. RAQUEL B. RIVERA
Adviser (6CISTUDY2)

Noted by:

MR. ULYSSES RAYMOND F. MONSALE
Coordinator
Computer Science Program

for 8-27-2025
DR. MARLON I. TAYAG
Dean
School of Computing

#1 Holy Angel Avenue, Sto. Rosario, Angeles City, Philippines 2009
Tel. Nos.: (045) 888-8691; 888-2902; 887-5748; 887-2455; 624-5277; 625-9619 | Fax: (045) 888-1754; 888-2514
Email: HAU@HAU.EDU.PH | WWW.HAU.EDU.PH

HOLY ANGEL UNIVERSITY SCHOOL OF COMPUTING

August 26, 2025

ROBIN GIBBS
Freelance Game Development Contractor

Dear Mr. Gibbs:

Greetings of peace!
A contingent part of the requirements for the Bachelor of Science in Computer Science is 6CISTUDY2 – Independent Study Course. Students enrolled in the course are required to create a research paper that will lead towards the development of their Undergraduate Thesis Project: "*Bala: A 2D Bullet Heaven Game Engine*". Included in the course is the requirement to identify respondents for their study, including data gathering through interviews and surveys. The objective of the course is to provide the students with the opportunity to create a research paper that could help in the area they have chosen for their field of study.

The following students have expressed their intention to consider your prestigious institution for their project proposal as respondents:

Julian Mathew M. Guintu
Gabriel V. Pageu
Aidan Zabdiel A. Serrano
Carlos Jose B. Ysais

Through this letter, I would like to seek your approval for the above undertaking. Rest assured that the information that will be given to them will be for academic purposes only and will be treated with utmost confidentiality. Thank you very much.

Sincerely,

DR. RAQUEL B. RIVERA
Adviser (6CISTUDY2)

Noted by:

MR. ULYSSES RAYMOND F. MONSALE
Coordinator
Computer Science Program

for 8-27-2025
DR. MARLON I. TAYAG
Dean
School of Computing

Robin Gibbs

#1 Holy Angel Avenue, Sto. Rosario, Angeles City, Philippines 2009
Tel. Nos.: (045) 888-8691; 888-2902; 887-5748; 887-2455; 624-5277; 625-9619 | Fax: (045) 888-1754; 888-2514
Email: hau@hau.edu.ph | www.hau.edu.ph

Appendix B
Sample Instrument

Bala: A 2D Bullet Heaven Game Engine

Greetings!

We are **SYPG**, a group of 4th Year Computer Science students from Holy Angel University:

Serrano, Aidan Zabdiel A.

Ysais, Carlos Jose B.

Pagcu, Gabriel V.

Guintu, Julian Mathew M.

We are conducting this assessment as part of our study. The survey will take approximately **5-8 minutes** to complete. There are no right or wrong answers — please respond honestly based on your own thoughts and experiences.

All responses will be treated with strict **confidentiality** and **anonymity**.

Data Privacy Notice:

Pursuant to R.A. 10173, or the Data Privacy Act, the researchers are committed to the protection of the personal data you provide and respect your value as a data subject in accordance with the requirements of the law and its implementing rules and regulations. In line with this, the researchers implement reasonable safeguards to maintain the confidentiality, integrity, and availability of your personal data. By accessing the form, you are giving your full consent to the researchers, whether manually or electronically, for the allowable period under the law for any personal data you provide for legitimate research purposes only. You are advised that whatever personal data is required here is necessary for conducting data collection in line with the researcher's study. The researchers give notice that the privacy notice may change from time to time in keeping with updates under the law and its implementing rules and regulations. Your continued use and access to the form, after having read this privacy notice, signifies that you have understood the same and its implications under existing laws, rules, and regulations. Thank you very much for your significant contribution in realizing the researcher's goal of making this study a success.

Switch account

Draft saved

* Indicates required question

Email *

☐ Record as the email to be included with my response

Are you willing to be part of this study? *

○ Yes

○ No

Bala Usability Survey

Name *

Your answer

Occupation *

Your answer

Please select the response that best indicates your response to the statements below. If you are unsure about an answer select the middle (neutral)

The response scale for each question is a 5-point Likert agreement scale:

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
1	2	3	4	5

Bala is frequently usable. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Bala is simple and straightforward. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Bala is easy to use. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Bala can be used without technical support. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

The various functions in Bala are well integrated. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Bala is consistent in its design and features. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Most people would learn to use Bala very quickly. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Bala is convenient and efficient to use. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Using Bala inspires confidence. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Bala can be used effectively with minimal learning. *

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

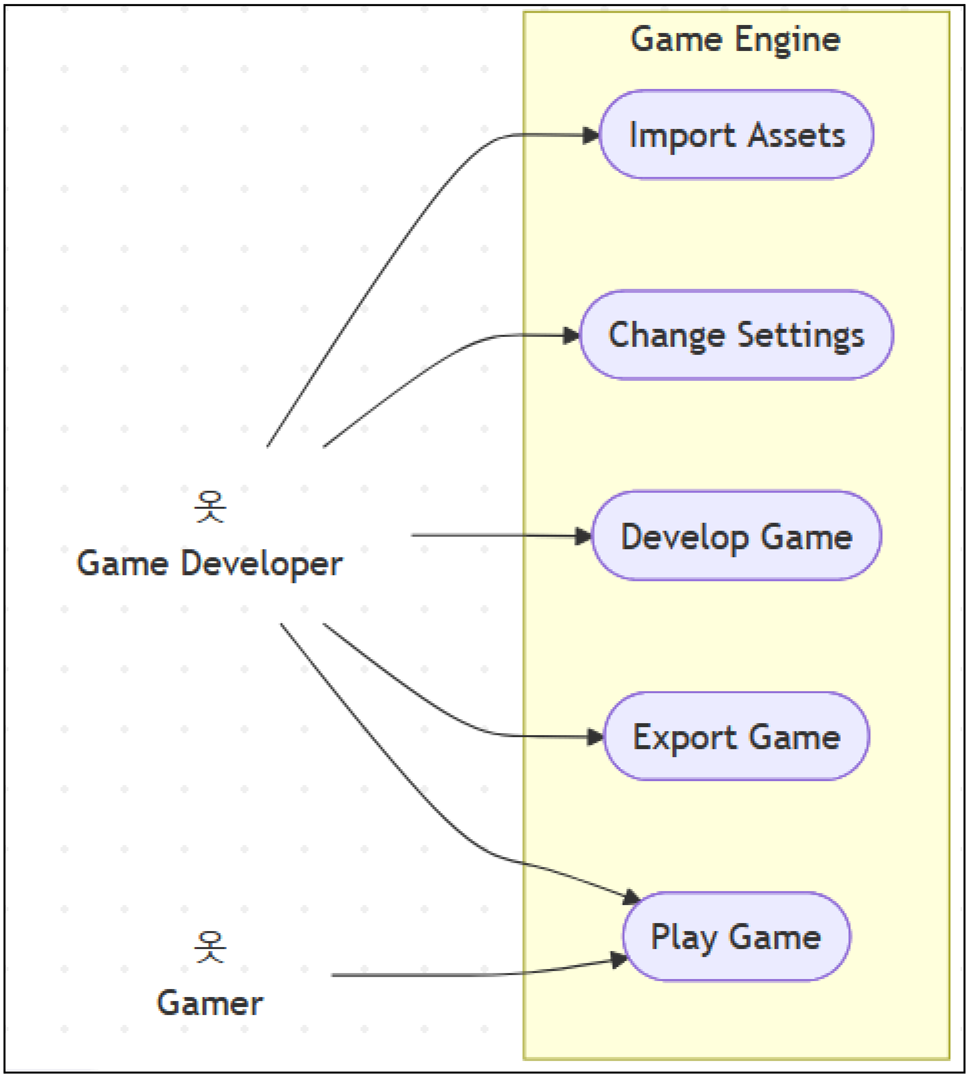
What problems or frustrations have been experienced while using Bala? *

Your answer

What recommendations can be given for improving Bala? *

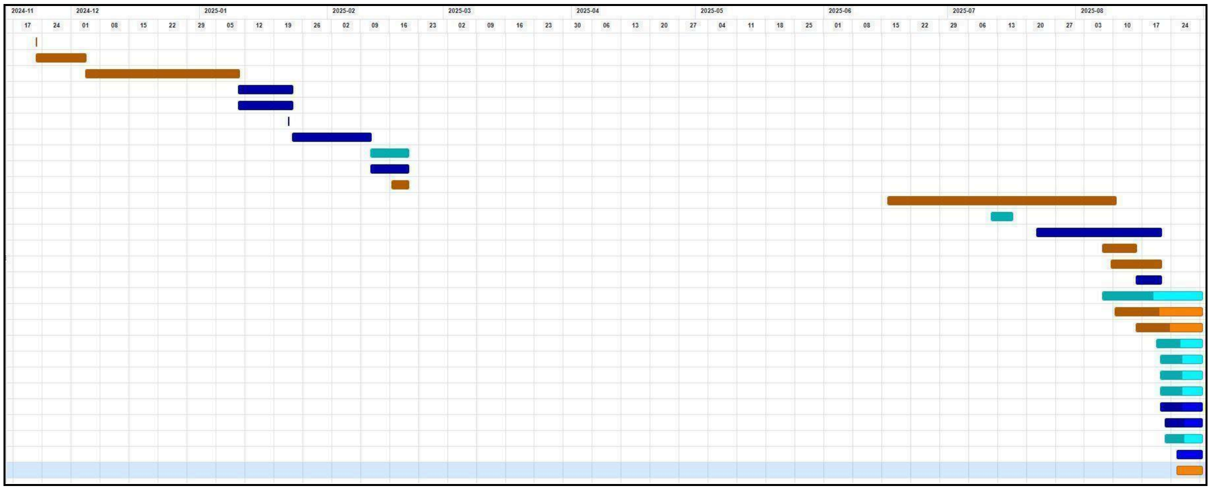
Your answer

Appendix C
Use Case Diagram

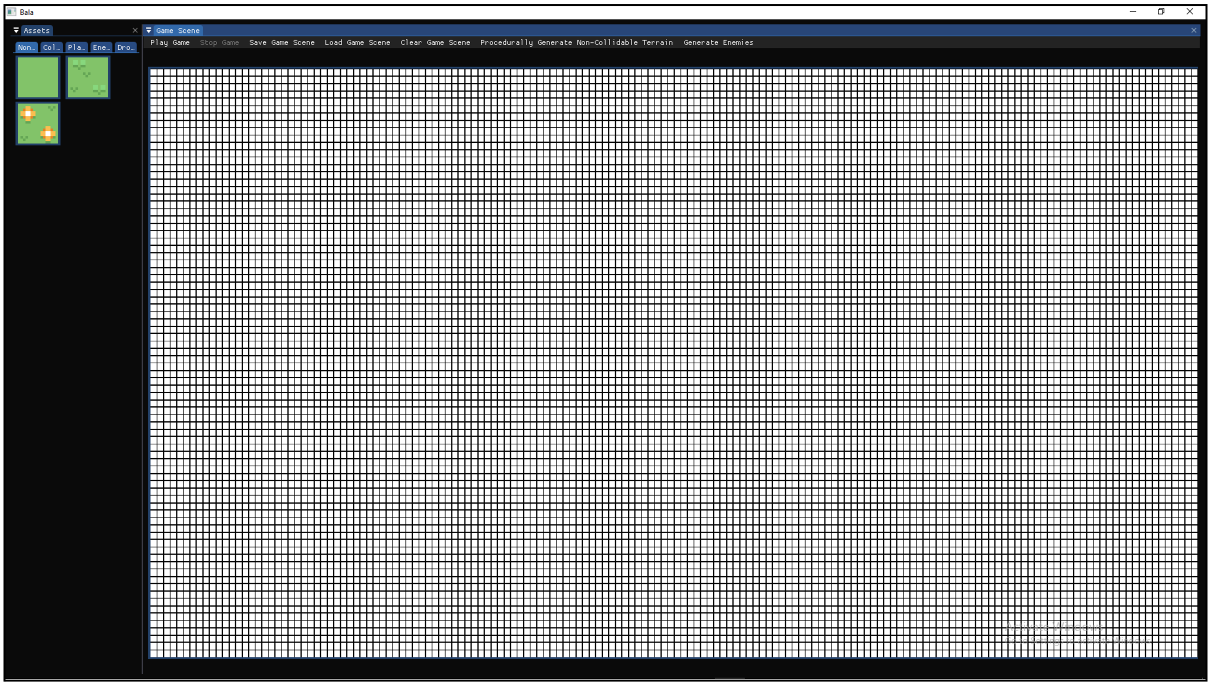


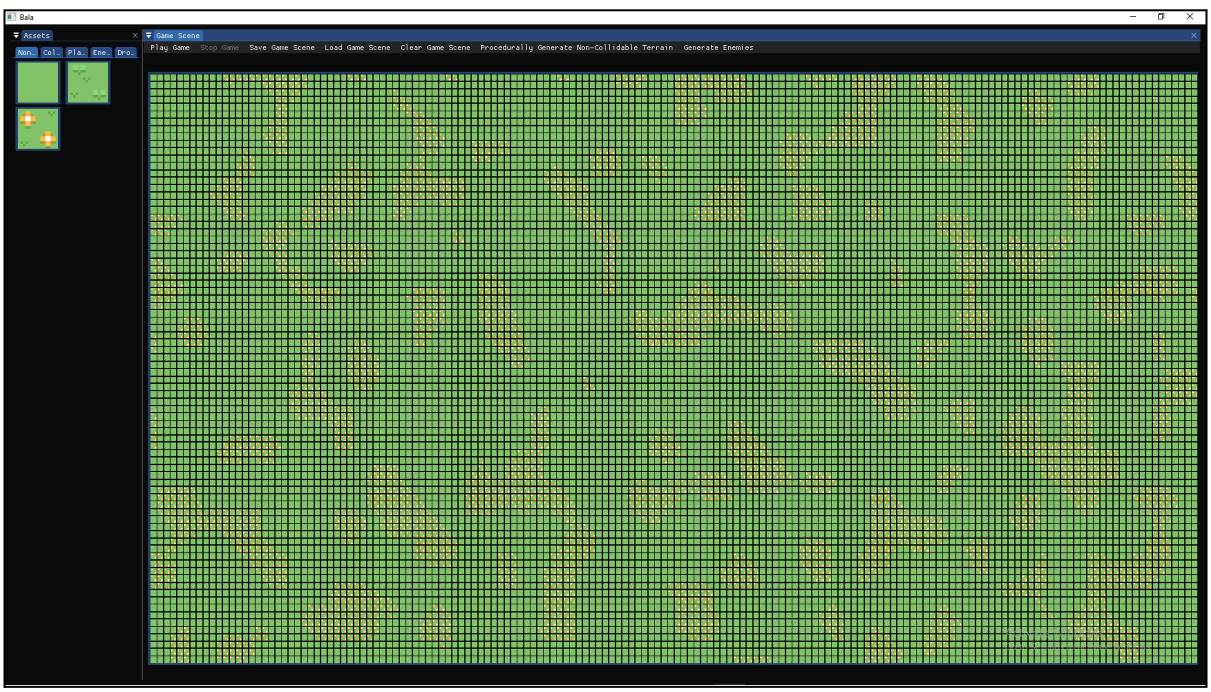
Appendix D
Gantt Chart

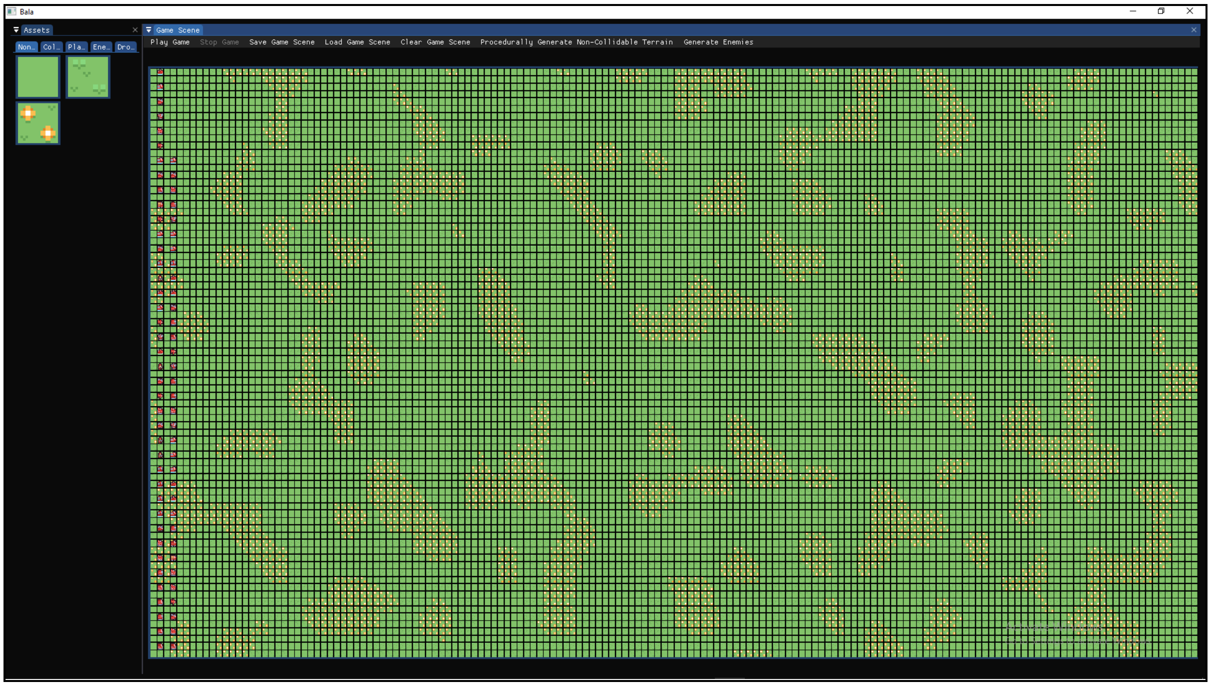
ID	Task Name	Start	End	Duration
19	Form Group	2024-11-22	2024-11-22	1 day
20	Brainstorm Titles	2024-11-22	2024-12-04	13 days
21	Write Title Proposals	2024-12-04	2025-01-10	38 days
22	Finished Chapter One	2025-01-10	2025-01-23	14 days
23	Write Chapter Two until Sources of Data	2025-01-10	2025-01-23	14 days
24	Visit Libraries of other Schools	2025-01-22	2025-01-22	1 day
25	Finished Chapter Two	2025-01-23	2025-02-11	20 days
26	Communicate with Potential Respondents	2025-02-11	2025-02-20	10 days
27	Revised Chapters One and Two	2025-02-11	2025-02-20	10 days
28	Created Slides and Videos	2025-02-16	2025-02-20	5 days
1	LWJGL Group Study	2025-06-16	2025-08-10	56 days
2	Silva - Correspondence Letter	2025-07-11	2025-07-16	6 days
3	Quantitative Research Design	2025-07-22	2025-08-21	31 days
4	3 minutes Video Presentation (DLSU GAME L...	2025-08-07	2025-08-15	9 days
5	Game Engine Wireframe	2025-08-09	2025-08-21	13 days
6	Survey Questionnaire Changes (Google Form...	2025-08-15	2025-08-21	7 days
7	Del Gallego - Correspondence Letter	2025-08-07	2025-08-31	25 days
8	Game Engine Coding	2025-08-10	2025-08-31	22 days
9	Presentable Game Engine	2025-08-15	2025-08-31	17 days
10	Find Additional Respondent for Backup	2025-08-20	2025-08-31	12 days
11	Radam - Correspondence Letter	2025-08-21	2025-08-31	11 days
12	Andal - Correspondence Letter	2025-08-21	2025-08-31	11 days
13	Apacible - Correspondence Letter	2025-08-21	2025-08-31	11 days
14	Additional RRL on why 2D game engine is ne...	2025-08-21	2025-08-31	11 days
15	Methodology Update	2025-08-22	2025-08-31	10 days
16	Gibbs - Correspondence Letter	2025-08-22	2025-08-31	10 days
17	Scope and Delimitations Update	2025-08-25	2025-08-31	7 days
18	Game Engine Testing	2025-08-25	2025-08-31	7 days

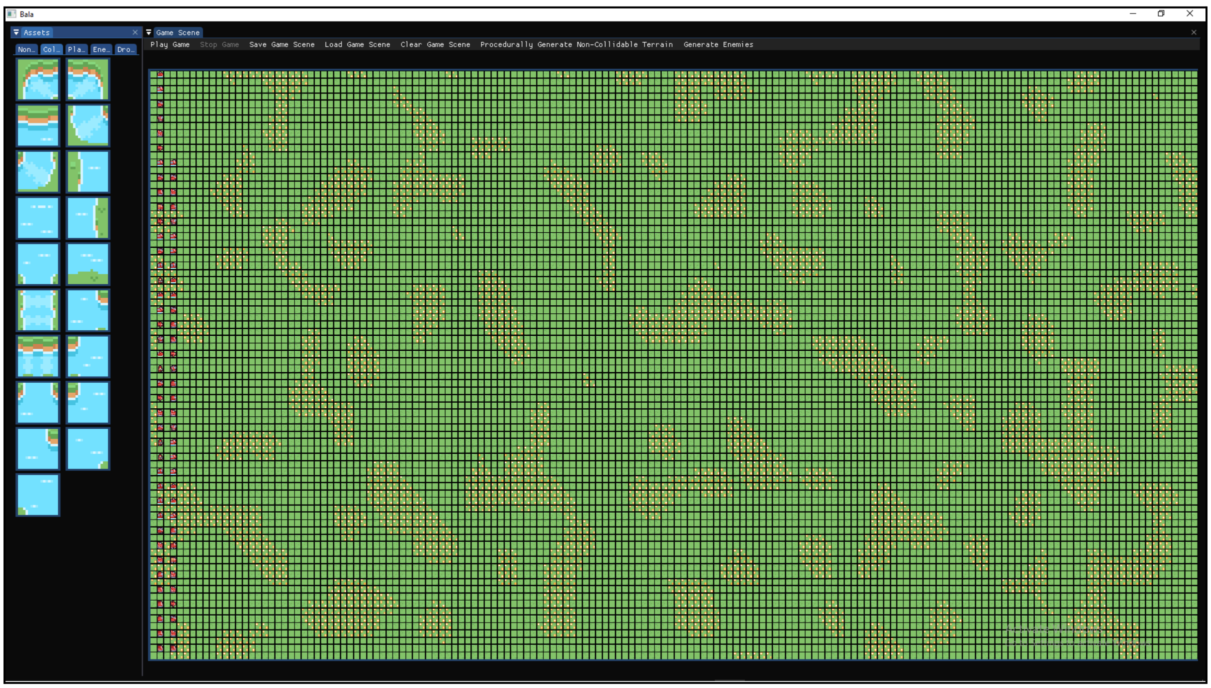

2024-11
2024-12
2025-01
2025-02
2025-03
2025-04
2025-05
2025-06
2025-07
2025-08
17 24 01 08 15 22 29 05 12 19 26 02 09 16 23 02 09 16 23 30 06 13 20 27 04 11 18 25 01 08 15 22 29 06 13 20 27 03 10 17 24

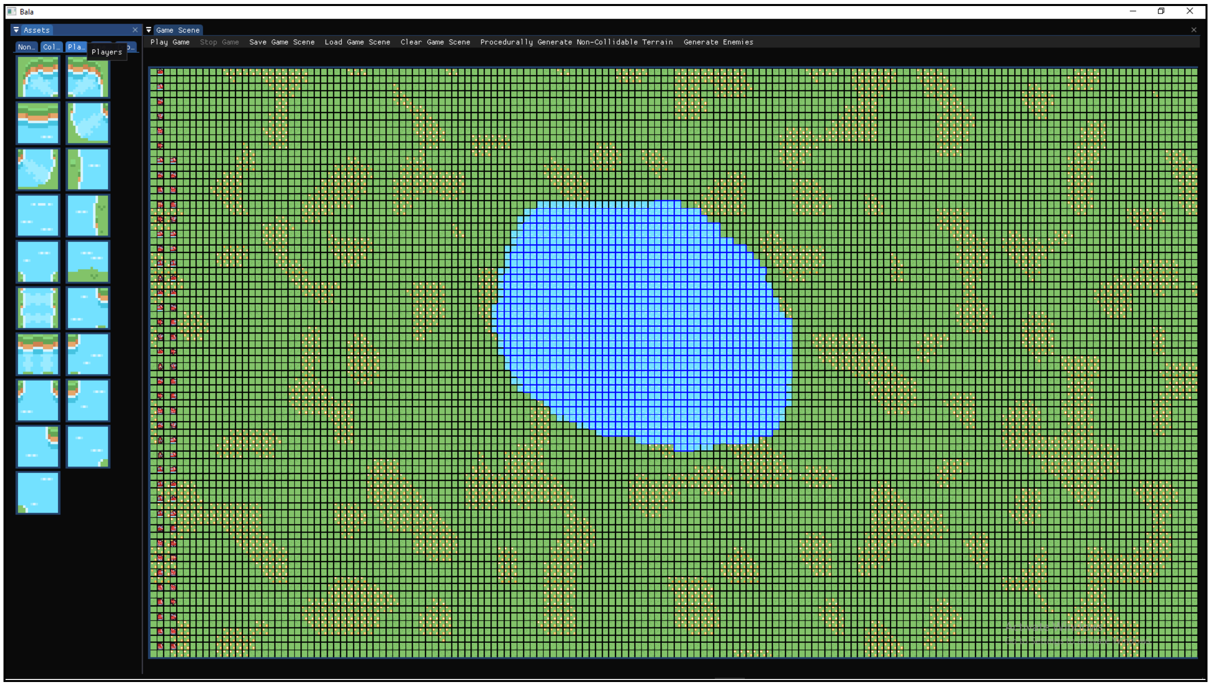
Appendix E
Screenshots of Application

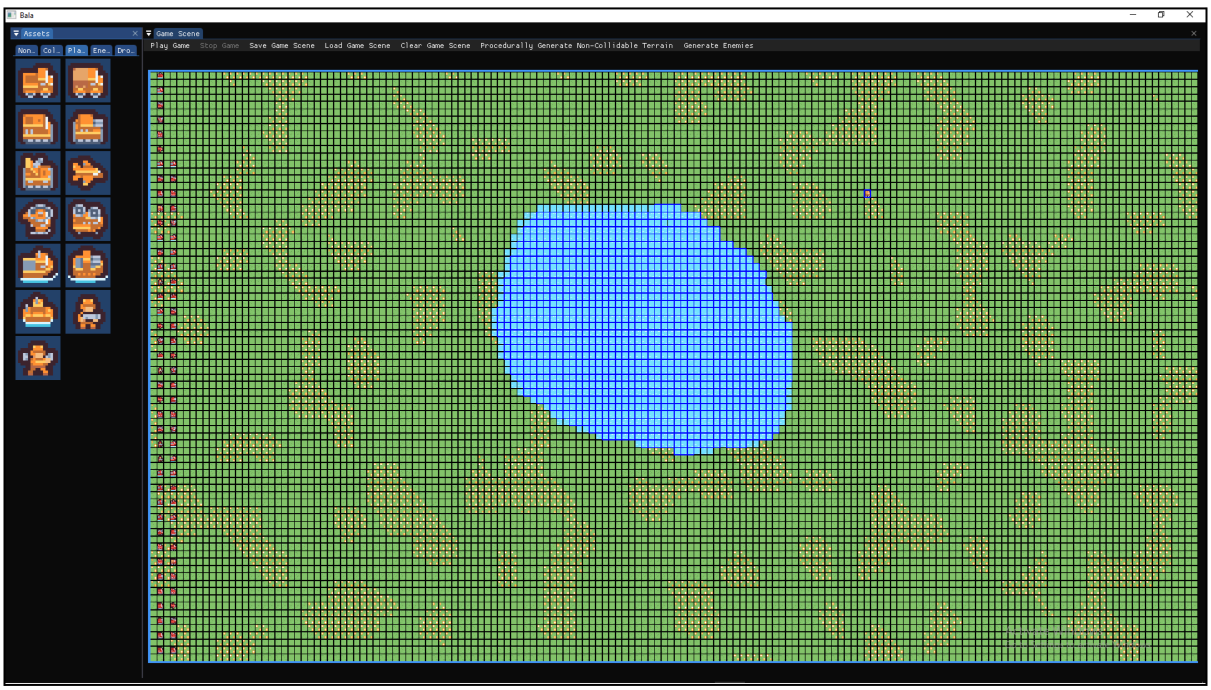


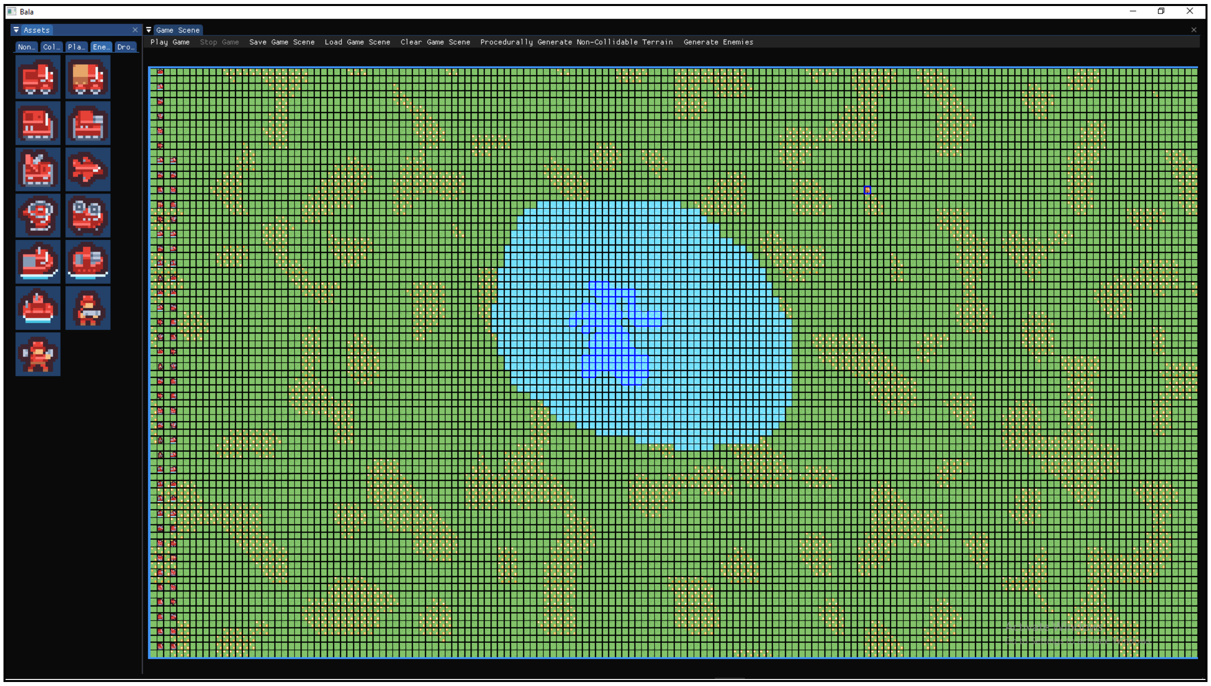


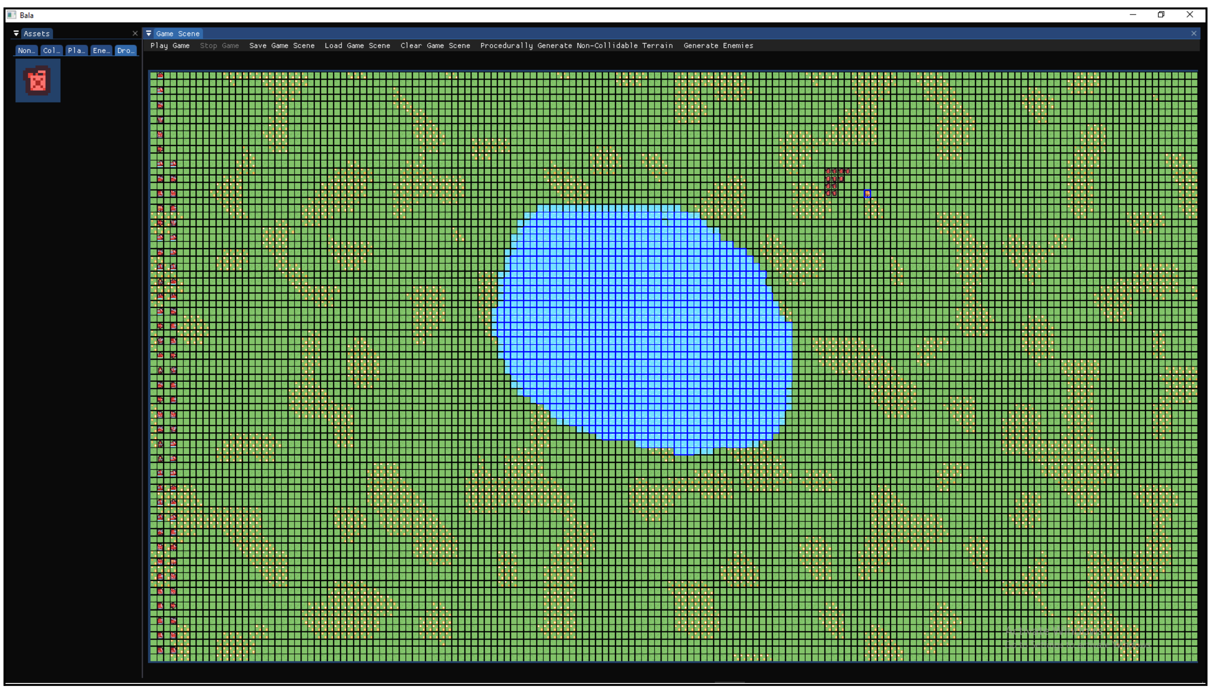

Bala
Assets
Game Scene
Play Game
Stop Game
Save Game Scene
Load Game Scene
Clear Game Scene
Procedurally Generate Non-Collidable Terrain
Generate Enemies


Bala
Assets
Game Scene
Play Game
Stop Game
Save Game Scene
Load Game Scene
Clear Game Scene
Procedurally Generate Non-Collidable Terrain
Generate Enemies

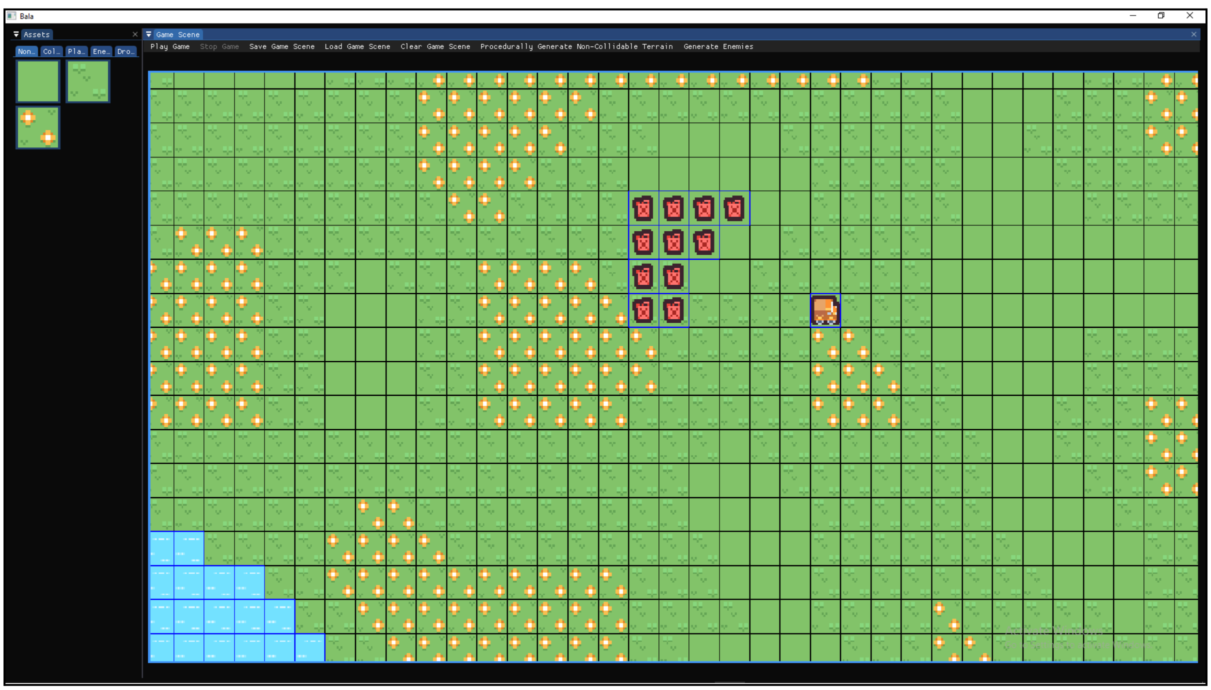

Bala
Assets
Game Scene
Players
Play Game
Stop Game
Save Game Scene
Load Game Scene
Clear Game Scene
Procedurally Generate Non-Collidable Terrain
Generate Enemies


Bala
Assets
Game Scene
Play Game
Stop Game
Save Game Scene
Load Game Scene
Clear Game Scene
Procedurally Generate Non-Collidable Terrain
Generate Enemies


Bala
Assets
Game Scene
Play Game
Stop Game
Save Game Scene
Load Game Scene
Clear Game Scene
Procedurally Generate Non-Collidable Terrain
Generate Enemies


Bala
Assets
Game Scene
Play Game
Stop Game
Save Game Scene
Load Game Scene
Clear Game Scene
Procedurally Generate Non-Collidable Terrain
Generate Enemies


Bala
Game Scene
Play Game
Stop Game
Save Game Scene
Load Game Scene
Clear Game Scene
Procedurally Generate Non-Collidable Terrain
Generate Enemies


Bala
Assets
Game Scene
Play Game
Stop Game
Save Game Scene
Load Game Scene
Clear Game Scene
Procedurally Generate Non-Collidable Terrain
Generate Enemies


Bala
Properties
Transform
SpriteRenderer
Game Scene
Play Game
Stop Game
Save Game Scene
Load Game Scene
Clear Game Scene
Procedurally Generate Non-Collidable Terrain
Generate Enemies

Appendix F
Hardware and Software Specification

Hardware Component	
CPU	AMD Ryzen 3 3100, 3.6 GHz
GPU	AMD Radeon RX 550 (Polaris 12/Lexa Pro), 4GB GDDR5
Memory	Kingston KHX2666C, 1x8GB DDR4, 2.6 GHz
Disk	HKVSN HS-SSD-C100, 240GB, 6.0GB/s
Monitor Resolution	1920 x 1080
Monitor Refresh Rate	75Hz
AMD Driver Version	31.0.12029.10015

Software Dependency	
Programming Language	Java Development Kit (JDK) (version 11 to 16)
Build Tool	Gradle (version 5.0 to 8.14)
Operating System	Microsoft Windows 10 Professional (x64) Build 19045.6216

Appendix G
University Plagiarism Certificate

Appendix H
Experts' Curriculum Vitae

Joaquin Paolo S. Andal

3646 Gen. Luna St., Bangkal, Makati City, 1233 | 09224074964 | paoloandal@gmail.com

TECHNICAL SKILLS

GAME DEVELOPMENT

- Unity Game Engine
- Visual Studio
- C#
- Vuforia
- GitHub
- PlayFab

WEB & MOBILE APP DEVELOPMENT

- Flutter SDK
- Dart
- Firebase
- Visual Studio Code
- Android Studio
- Git
- Riverpod
- BLoC
- Clean Architecture

BACK-END DEVELOPMENT

- MongoDB
- ExpressJS
- NodeJS
- EmailJS
- Postman
- Heroku

2D AND 3D ART

- Adobe Photoshop
- Blender
- Canva

WORK EXPERIENCE

SOFTWARE DEVELOPER | FREELANCE | JAN 2023 - PRESENT

· I create augmented reality apps, and mobile and desktop games using Unity Game Engine; and websites and mobile apps using Flutter.

LEAD UNITY DEVELOPER | CREATIVE BRAIN STUDIOS | MAY 2022 – DEC 2022

· I am in charge of creating design patterns to be implemented in various projects and concepting ideas for inhouse game projects.

SENIOR GAME DEVELOPER | BOOM BRAIN STUDIOS | AUGUST 2021 – APRIL 2022

· I implemented core game design using various programming techniques and maintained all game databases. I was also the recipient of the company's Employee of the Month award from January – February.

SOFTWARE ENGINEER I | REALTAIR INC. | FEBRUARY 2021 – JULY 2021

· I created online forms for the purchasing and selling of real estate as stated in client-discussed implementation documents. I also occasionally host company game nights and celebration programs.

SOLUTIONS DEPARTMENT INTERN | IT GROUP INC. | JUNE 2019 – JULY 2019

· My main responsibility revolved around creating proper documentation for all IT solutions provided, and creating proper package plans for clients.

FREELANCE PROJECTS

THE SPACE SHUTTLE | 2025

- An augmented reality mobile application for teaching the Solar System topics for Grade 6 students of Malabon Elementary School.
- Features augmented reality display of celestial bodies, for knowledge assessment mini-games.
- Developed using Unity Game Engine, C#, and Vuforia SDK

IMEASURE | 2024

- A cross-platform web and mobile e-commerce application for the buying and selling of manufactured windows and raw materials of Heritage Aluminum Sales Corporation Los Banos.
- Features a fully functional e-commerce platform with an integrated 3D Window customization feature for the mobile application.
- Developed using Flutter SDK, Dart, Android Studio, and Firebase

LINOCATE | 2024

- A mobile application for the pick-up and disposal of pig and agricultural waste in Pila, Laguna.
- Features instant messaging, live geolocation, and analytics & report generation.
- Developed using Flutter SDK, Dart, Android Studio, and Firebase

COMPREHENZONE | 2024

- A cross-platform website and mobile application for facilitating asynchronous online class management & school portal for Grade 5 & 6 students and teachers of Sta. Ana elementary school.
- Includes oral exam feature which integrates speech to text for proper pronunciation detection.
- Developed using Flutter SDK, Dart, Android Studio, and Firebase

MATHEVATION | 2024

- A 2D Game-based learning web & mobile application for teaching basic Math topics for Grade 2 students
- Features turn-based combat gameplay, levelling system with star ratings, reward system, player inventory, and character cosmetic items
- Developed using Unity Game Engine and C#

UNOS | 2023

- A 3D education mobile game for teaching children proper disaster-preparedness and protocols for earthquake and typhoons.
- Features interaction and exploration of a 3D suburban village, dialogue system with non-playable characters, and multiple minigames.
- Developed using Unity Game Engine and C#

YOUTH CONNECT | 2023

- A mobile application for members of partner organizations under the Youth Welfare Development Agency of Laguna.
- Features mini-quizzes, mental health diary, events & announcements dissemination, and database registry of partner organizations
- Developed using Dart, Flutter SDK, and Google ML Kit

FITNESSCO | 2023

- A mobile application for gym management with integrated email notification, trainer selection, instant messaging and workout plan prescription.
- Features integration of a for live pose estimation for proper workout form detection
- Developed using Dart, Flutter SDK, and Google ML Kit

EDUCATION

PRIMARY EDUCATION | LA SALLE GREEN HILLS ·

· Silver Deportment Awardee

SECONDARY EDUCATION | 2012 - 2016 | LA SALLE GREEN HILLS

· 3rd Honors Graduate

TERTIARY EDUCATION | 2016 - 2021 | UNIVERSITY OF THE PHILIPPINES MANILA

· Bachelor of Science in Computer Science

ORGANIZATIONAL EXPERIENCE

UNIVERSITY STUDENT COUNCIL | UNIVERSITY OF THE PHILIPPINES MANILA | 2020 - 2021

· Gender, Students' Rights and Welfare Committee

ALPHA SIGMA FRATERNITY | UNIVERSITY OF THE PHILIPPINES MANILA | 2017 - 2021 ·

Master of Finance (2017-2019)

UPM DRAMATISTA | UNIVERSITY OF THE PHILIPPINES MANILA | 2016 - 2021

· Vice President (2018 – 2019)

· Logistics Officer (2020 – 2021)

DEBATE VARSITY | LA SALLE GREEN HILLS | 2014 – 2016

· Rotary Debate Open Octo-finalist

SEMINARS

VIRUS DETECTED: THE PHILOSOPHICAL IMPLICATION OF DATA AND TECHNOLOGY IN THE NEW NORMAL | SEPTEMBER 30, 2020

· Gave a talk to the students of the University of the Philippines Los Banos about the topic "Contemporary Issues Arising from Online Communities"

THE ART OF ANIMATION: *KLAUS* REPLAY | APRIL 30, 2020

· Attended an online international seminar held by CG Master Academy which discussed the animation industry and a behind the scenes of the animated movie "Klaus"

WELCOME TO GAME DEV: UNITY WORKSHOP AND CAREERS AT SECRET 6 | SEPTEMBER 23, 2017

· Attended a workshop held by Secret 6 Inc. showing a basic introduction to the Unity Game Engine, and a talk discussing the game development career path

CHARACTER REFERENCES

BERWIN JARRET T. YU| Undergraduate Thesis Adviser | 09453390636

JOSE MICO ALCANSE| Supervisor, Realtair Inc. | 09175721174

JAN BIEN GABRIELLE DANIEL| Supervisor, Creative Brain Studios | 09672852303

ALESSANDRA BETTINA GARCIA

GAME PROGRAMMER

alessandrabettina.garcia@gmail.com

https://www.linkedin.com/in/alessandra-bettina-garcia/

PROFILE SUMMARY

A passionate game programmer with experience in multiple roles in the game development process. Having worked on several game projects through years of study, is eager to collaborate with others and contribute my skills to the creation of enjoyable interactive experiences. Mostly familiar with C# used alongside Unity but is quick to learn other programming languages and software as needed.

WORK EXPERIENCE

Game Developer (Unity) — November 2023 - Present

Carmine Inc

- Programming UI and gameplay and fixing bugs. Also doing a little bit of VFX work.
- The project is MADE WITH Unity and git is being used for version control.
- Currently working on the demo release.

Game Developer Intern — August 2022 - March 2023

Taktyl Studios

- Programmed UI and gameplay, created temporary art assets and mock-ups, fixed bugs, and applied client feedback for the assigned project. The project was MADE WITH Unity and git was used for version control. Worked with an agile workflow.
- Developed a simple game within two weeks. Code was described as readable.
- Volunteered for game testing outside the project scope.

Development Intern — April 2019 - May 2019

Indigo Entertainment

- Singlehandedly created a game prototype MADE WITH Unity in the span of two weeks. Programmed all UI and gameplay, created art assets, designed, debugged, wrote the story and dialogue, and wrote a quick makeshift game design document.

PROJECTS

Support System MADE WITH Unity — February 2022 - July 2023

- A 2D adventure platformer that is intended to help people understand and learn more about dissociative identity disorder (DID).
- Created most of the game's systems. Was also the game designer and writer for this project.
- Used git for version control.
- Reached out to mental health professionals for interviews during the pre-production phase.
- Capstone project for my degree in BSEMC-GD. We were a team of 5.

I'm Ad Work MADE WITH **Unity** April 2022 - June 2022

- Published on itch.io. A simple game wherein you try to get as much work done as you can while ads are constantly popping up on your computer.
- Programmed, designed, and illustrated by myself. Art assets made with aseprite.
- A personal project completed within 2 months.

EDUCATION

Bachelor of Science in Entertainment and Multimedia Computing with Specialization in Game Development 2019 - 2023

iACADEMY

- Latin Honors - *magna cum laude*

TECHNICAL SKILLS

- Proficient in **C#**, especially when used with **Unity**
- Has prior experience with **C++** and **JavaScript (Phaser)**
- Quick to learn other programming languages as needed
- Knows how to use **Git** for version control
- Experienced in different aspects of game development, including programming, UI/UX, art, animation, game design, and debugging

PROFESSIONAL SKILLS

- Flexible working independently or with teams
- Can effectively lead a project team
- Good analytical and problem-solving skills
- Adaptable
- Meets deadlines

PROFESSIONAL REFERENCES

Ralph de Vera
Studio Production Manager at Dusk Wave Arts
ralph@duskwavearts.com

Carl Louie So
Game Development Chairperson at iACADEMY
carl.so@iacademy.edu.ph
09208919043

Arisa Alcantara
Part-time faculty (Programming) at iACADEMY
siararisa@gmail.com
09204206678

ARON JAKE RADAM

Caloocan City, Metro Manila | radam.aronjake@gmail.com | aronjakeradam.netlify.app

SUMMARY

Highly motivated Software Developer skilled in C# and Java, experienced in developing robust tools and enhancing system stability. Proven collaborator with a strong foundation in complex system design from game development.

TECHNICAL SKILLS

Logical Thinking
Game Quality Assurance
Computer Programming
Game Design
Game Development
Game Programming

PROFESSIONAL EXPERIENCE

Quality Assurance Analyst, GridLock-Games Studio **May 2024 - Apr 2025**

- Identified, reported, and verified over 100 bugs across test game builds, enhancing pre-launch stability and product quality, and collaborated with developers to streamline issue resolution and testing.
- Demonstrated a keen eye for detail and proactive approach in identifying issues early, consistently providing clear and constructive feedback that enhanced the quality of game features before release.

EDUCATION

BS Entertainment and Multimedia Computing **Aug 2021 - Apr 2025**
University of Caloocan City

- Major in Game Development.
- Capstone on "Trash Frenzy: A 2D Fast Paced Action Side Scroller Educational Android Game Promoting Water Pollution Awareness".

KEY PROJECTS

Save Encryption Toolkit **Apr 2025 - Apr 2025**

- Engineered a secure and flexible data encryption solution with advanced cryptographic techniques, developing comprehensive support for diverse serialization formats and intuitive editor tools.

Trash Frenzy | Capstone Project **Aug 2024 - Apr 2025**
University of Caloocan City

- Designed and launched a 2D Android mobile application promoting water pollution awareness, integrating gamified learning elements.
- Conducted user acceptance testing with 75 participants, earning 90%+ positive feedback from both players and IT experts.
- Led a team of 4 developers, managing the project from concept to prototype delivery and ensuring on-time completion.

John Anthony Silva

Game Developer

Central Luzon, Philippines • dev.john.silva@gmail.com • www.linkedin.com/in/johnsilva19

Game developer with experience across the full software lifecycle, skilled in project management, console porting, and team leadership. Driven by continuous learning and improving the value of each project and team I join.

PROFESSIONAL EXPERIENCE

Game Developer | RLUX Studios | Philippines
Jan 2022–Present

- Designed and built complete games from the concept phase through to release, ensuring milestones were achieved within project timelines.
- Adapted and ported original game builds to major consoles including Nintendo Switch, Playstation, and Xbox, managing distinct certification requirements for each platform.
- Guided university interns and on-the-job trainees, supporting both their technical growth and integration into professional workflows.
- Developed and maintained detailed, realistic project schedules to keep development efforts synchronized and on track.
- Collaborated cross-functionally with artists, designers, and QA to align technical and creative objectives.

Technical Support Representative | TopData Global IT Solutions | Philippines
Apr 2019–June 2021

- Resolved technical and user inquiries for multiple operating systems through chat and email, consistently maintaining high customer satisfaction.
- Documented recurring incidents and produced quick-reference guides to improve the efficiency of troubleshooting processes.
- Tested in-house software for usability, security, and performance prior to deployment, identifying issues early in development cycles.
- Provided remote support for software configuration and troubleshooting, adapting communication approaches depending on user technical familiarity.
- Served as the initial point of contact for technical questions within the team, streamlining internal support and communication.

Game Developer | Keybol Games | Philippines
Feb 2018–March 2019

- Contributed to the design and implementation of core gameplay features and mechanics as part of a collaborative team.
- Optimized game performance for targeted platforms while troubleshooting and resolving code defects post-release.
- Supported QA and playtesting processes, collecting feedback and iteratively refining game content.
- Worked alongside artists and audio engineers to integrate visual and sound assets according to design requirements.

Game Developer (OJT) | Playhaus | Philippines
Jun 2016–October 2016

- Assisted in feature implementation, bug testing, and game balancing for ongoing projects.
- Gained experience in collaborative version control environments and game engine fundamentals.
- Prepared technical reports documenting development progress and roadblocks for faculty advisors.

EDUCATION

Holy Angel University | Pampanga, Philippines
2017
Bachelor's of Science in Information Technology - Major in Animation

SKILLS

Game Development
Project Management
Game Porting
Unreal Engine 4
Console Certification Processes
Intern Mentorship
Technical Troubleshooting
Cross-Platform Development
Software Testing
Team Coordination
Windows, Linux, Android, iOS, Mac OS Support

CERTIFICATES

Best Game Pitch Idea | **Playhaus** | *2016*
Resource Speaker - Becoming Globally Competitive In Game Development | **Holy Angel University** | *2018*

Appendix I
Researchers' Curriculum Vitae

JULIAN MATHEW M. GUINTU

Address: Block 238 Lot 23, 16th Street, Metrogate Capaya, Angeles City
Phone: 09685800787
Email: jmm.guintu@gmail.com

SOFT SKILLS

- Communication
- Teamwork
- Problem-Solving
- Critical Thinking
- Adaptability
- Time Management
- Leadership
- Emotional Intelligence

TECHNICAL (IT) SKILLS & TOOLS

- Python
- Java
- HTML
- CSS
- Javascript
- React JS
- Groovy

CERTIFICATIONS

Data Analytics Essentials 2025
Cisco

AWS Academy Graduate - AWS Academy Cloud Foundations 2025
Amazon Web Services Training and Certification

Foundations of User Experience (UX) Design 2024
Google

Cyber Threat Management 2024
Cisco

CCNA: Introduction to Networks 2024
Cisco

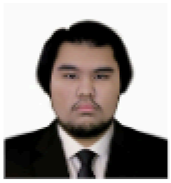
Cybersecurity Essentials 2024
Cisco

EDUCATIONAL BACKGROUND

Bachelor of Science in Computer Science 2022 - Present
Holy Angel University

ADDITIONAL INFORMATION

- **Languages:** English, Filipino, and Kapampangan.
- **Awards:** Dean's Lister during the 1st Semester of S.Y. 2024 - 2025, 2023 – 2024 and 2022 - 2023, and 2nd Semester of S.Y. 2024 - 2025.



GABRIEL V. PAGCU

Address: #133 Mercury St. San Fernando Subd. Brgy. Sto Nino City of San Fernando Pampanga 2000
Phone: 09054106200
Email: gabgab0518@gmail.com

SOFT SKILLS

- Communication
- Collaboration
- Attention to Detail
- Adaptability
- Work Ethic
- Flexibility
- Active Listening
- Perseverance

TECHNICAL (IT) SKILLS & TOOLS

- Python
- Java
- HTML
- CSS
- Javascript
- React JS
- C

CERTIFICATIONS

Introduction to Modern AI — **2025**
Cisco

AWS Academy Graduate - AWS Academy Cloud Foundations — **2025**
Amazon Web Services Training and Certification

Cyber Threat Management — **2024**
CIcso

Digital Awareness — **2024**
Cisco

Introduction to Data Science — **2024**
Cisco

Introduction to Cybersecurity — **2024**
Cisco

EDUCATIONAL BACKGROUND

Bachelor of Science in Computer Science — **2022 - Present**
Holy Angel University

ADDITIONAL INFORMATION

- **Languages:** English, Filipino, and Kapampangan.
- **Awards:** Dean's Lister during the 2nd Semester of S.Y. 2024 - 2025.

AIDAN ZABDIEL A. SERRANO

Address: 225 San Juan Nepomuceno Betis Guagua Pampanga
Phone: 09959245578
Email: aidanserrano122703@gmail.com

SOFT SKILLS

- Great Teamwork abilities
- Excellent Communication Skills
- Active Listener
- Problem-solving

TECHNICAL (IT) SKILLS & TOOLS

- Python
- Java
- HTML
- CSS
- MySQL
- Microsoft Office

CERTIFICATIONS

Data Analytics Essentials — **2025**
Cisco

AWS Academy Graduate - AWS Academy Cloud Foundations — **2025**
Amazon Web Services Training and Certification

Foundations of User Experience (UX) Design — **2024**
Google

Cyber Threat Management — **2024**
Cisco

CCNA: Introduction to Networks — **2024**
Cisco

Cybersecurity Essentials — **2024**
Cisco

EDUCATIONAL BACKGROUND

Bachelor of Science in Computer Science — **2022 - Present**
Holy Angel University

ADDITIONAL INFORMATION

- **Languages:** English, Filipino, and Kapampangan.
- **Awards:** Dean's Lister during the 1st Semester of S.Y. 2022 - 2023

CARLOS JOSE B. YSAIS

Address: 131 St. Catherine St. Villa Angelina Subd., San Jose, Angeles City
Phone: 09764762171
Email: carlosjose.ysais@gmail.com

SOFT SKILLS

- Teamwork
- Problem-Solving
- Adaptability

TECHNICAL (IT) SKILLS & TOOLS

- TypeScript
- HTML
- CSS
- MySQL
- Java
- Python

WORK EXPERIENCE

Full-Stack Developer **April 2024 - Present**
Cinelaya

EDUCATIONAL BACKGROUND

Bachelor of Science in Computer Science **2021 - Present**
Holy Angel University

ADDITIONAL INFORMATION

- **Languages:** English and Filipino.
- **Awards**: President's Lister during the 1st Semester of S.Y. 2021 - 2022, 2nd Semester of S.Y. 2022-2023, 1st and 2nd Semester of S.Y. 2023 - 2024, and 1st and 2nd Semester of S.Y. 2024 - 2025.

Appendix J
Glossary

Term	Definition
B = Axis-Aligned Bounding Boxes (AABBs)	It is used for its accuracy that allows for a different length and width in collision detection. It is also can be seen on stress and performance testing to act as a point of comparison when Circle-Circle Collision is enabled.
Behaviour Simulation	It serves as the game artificial intelligence that was done through flocking.
Bullet Heaven	It refers to a sub-genre of shoot ’em up (shmup) games inspired by the bullet hell genre. In bullet heaven, players control a character capable of generating projectiles or attacks, often in the style of an auto-shooter.
C = Circle-Circle Collision Detection	It is used for its speed due to being able to calculate when two objects collide by comparing the distance between the two center points of the objects enclosed in a circle hitbox with the sum of their radii. It is also can be seen on stress and performance testing to act as a point of

	comparison when Axis-Aligned Bounding Boxes are enabled.
Collision Detection	Built-in functions which implement axis-aligned bounding boxes (AABBs) and circle-circle collision detection are available for the game developers to call so that objects such as the player and NPCs can interact with each other properly.
F = Flocking Algorithm	It is used to simulate the behaviour of flocks of animals such as birds and fishes, which is helpful in making sure that NPCs, especially enemies, won't hit each other so that no NPC gets covered. It can also be seen on stress and performance testing to act as a point of comparison when Flocking is disabled.
Frame Rate	It reflects the general performance of a game that is equal to Frames Per Second.
Frame Time	It reflects the stability of frame delivery for the smoothness of gameplay.

Game Artificial Intelligence	Built-in function implementing flocking is available for game developers to use for their NPCs so that they could interact properly.
Game Engine	It used to tailor to the unique demands of bullet heaven games by providing a user-friendly and visual interface by following good user interface design principles
nF = No Flocking	Flocking is disabled as can be seen on stress and performance testing to act as a point of comparison when flocking is enabled.
Non-Player Character	Enemies generated that are not controllable, going near to the player acting as a hostile.
P = Perlin Noise	It is a type of gradient noise, can generate complex patterns with minimal computational resources that involves generating a 2D float matrix with values between 0.0f and 1.0f. It can also be seen on stress and performance testing to act as a point of comparison when Simplex is enabled.
PBF = Perlin Noise with Axis-Aligned Bounding Boxes (AABBs) and Flocking	Combination of algorithms that can be seen on

	stress and performance testing to act as a point of comparison to all combinations.
PBnF = Perlin Noise with Axis-Aligned Bounding Boxes (AABBs) and no Flocking	Combination of algorithms that can be seen on stress and performance testing to act as a point of comparison to all combinations.
PCF = Perlin Noise with Circle-Circle Collision Detection and Flocking	Combination of algorithms that can be seen on stress and performance testing to act as a point of comparison to all combinations.
PCnF = Perlin Noise with Circle-Circle Collision Detection and no Flocking	Combination of algorithms that can be seen on stress and performance testing to act as a point of comparison to all combinations.
Performance Testing	It is used to assess the game engine performance in a single computer through using built-in logger in Java to measure the frame time of the game engine in a total of eight tests for each combinations and configuration of algorithms.
Procedural Content Generation	It is the automatic creation of game content using algorithms, reducing manual effort, development costs, and memory space while adding variety. It

	can be used to create levels, maps, quests, characters, vegetation, and textures.
S = Simplex Noise	It is faster than Perlin noise with lesser artifacts, but it is more complex to implement. It can be seen on procedural content generation to generate 2D backgrounds. It can also be seen on stress and performance testing to act as a point of comparison when Perlin is enabled.
SBF = Simplex Noise with Axis-Aligned Bounding Boxes (AABBs) and Flocking	Combination of algorithms that can be seen on stress and performance testing to act as a point of comparison to all combinations.
SBnF = Simplex Noise with Axis-Aligned Bounding Boxes (AABBs) and no Flocking	Combination of algorithms that can be seen on stress and performance testing to act as a point of comparison to all combinations.
SCF = Simplex Noise with Circle-Circle Collision Detection and Flocking	Combination of algorithms that can be seen on stress and performance testing to act as a point of comparison to all combinations.
SCnF = Simplex Noise with Circle-Circle Collision Detection and no Flocking	Combination of algorithms that can be seen on stress and performance testing to act as a point of comparison to all combinations.

Stress Testing	It is used to test the capability of game engine in a total of eight tests for all combinations and configuration of algorithms by creating 75 enemies per second, which is equivalent to 1 enemy per frame based on the testing monitor's refresh rate. This was done until the game engine crashed or was unresponsive.

Appendix K
Editor's Note

HOLY ANGEL UNIVERSITY SCHOOL OF COMPUTING

DR. MARLON I. TAYAG
Dean

This is to certify that the document of the capstone entitled **Bala: A 2D Bullet Heaven Game Engine** authored by the following proponents:

Julian Mathew M. Guintu
Gabriel V. Pagcu
Aidan Zabdiel A. Serrano
Carlos Jose B. Ysais

has been checked for grammatical and typographical errors.

Yours truly,

Maria Cristina C. Nogoy, LPT, MAEd
Faculty, Comm and Languages Department

October 16, 2025

#1 HOLY ANGEL AVENUE, STO. ROSARIO, ANGELES CITY, PHILIPPINES 2009
TEL. NOS.: (045) 888-8691; 888-2902; 887-5748; 887-2455; 624-5277; 625-9619 | FAX: (045) 888-1754; 888-2514
EMAIL: HAU@HAU.EDU.PH | WWW.HAU.EDU.PH

Appendix L
Library Referral Letters

HOLY ANGEL UNIVERSITY | LIBRARY DEPARTMENT

January 20-24, 2025

DR. JONAVIE S. QUIAMBAO
Director
Angeles University Foundation
Angeles City, Pampanga

Dear Ma'am:

With your kind permission, our student(s)/faculty would like to make use of your library resources in relation to his/her/their research:

1. **Angelor T. De Jesus**
2. **Julian Matthew M. Guintu**
3. **Kurt Marten I. Sardes**
4. **Aidan Zabdiel A. Serrrano**
5. **Gabriel V. Pagcu**
6. **Guiliani Louis H. Pobre**
7. **Dean Louis T. Santos**
8. **Carlos Jose B. Ysais**

CCS/CS
AquaScore
> Camera-based system
> Panthaulon - camera based system validating parts
> Implementation of Context aware advertising using camera
> Early Warning System for Rice Crop
> Police Composite Sketch Match
> Peaches: Voice to Melody Converter using OnePitch (OMP)
> Bala: A 2D Bullet Heaven Game Engine

Rest assured that all policies and rules in your library shall be followed. Your assistance to him/her/them would be greatly appreciated.

Thank you very much.

Very truly yours,

[signature]
Ms. DAISY M. HICBAN
Chief Librarian

Endorsed by:
[signature] 1/22/25
MICHELLE B. CUNANAN, RL

TR# 8465

#1 HOLY ANGEL AVENUE, STO. ROSARIO, ANGELES CITY, PHILIPPINES 2009
TEL. NOS.: (045) 888-8691; 888-2902; 887-5748; 887-2455; 624-5277; 625-9619 | FAX: (045) 888-1754; 888-2514
EMAIL: HAU@HAU.EDU.PH | WWW.HAU.EDU.PH

HOLY ANGEL UNIVERSITY | LIBRARY DEPARTMENT

January 20-24, 2025

MS. IRENE M. MUNGCAL
Director
Systems Plus College
Angeles City, Pampanga

Dear Ma'am:

With your kind permission, our student(s)/faculty would like to make use of your library resources in relation to his/her/their research:

1. **Angelor T. De Jesus**
2. **Julian Matthew M. Guintu**
3. **Kurt Marten I. Sardes**
4. **Aidan Zabdiel A. Serrrano**
5. **Gabriel V. Pagcu**
6. **Guiliani Louis H. Pobre**
7. **Dean Louis T. Santos**
8. **Carlos Jose B. Ysais**

Rest assured that all policies and rules in your library shall be followed. Your assistance to him/her/them would be greatly appreciated.

Thank you very much.

Very truly yours,

Ms. DAISY M. HICBAN
Chief Librarian

#1 Holy Angel Avenue, Sto. Rosario, Angeles City, Philippines 2009
Tel. Nos.: (045) 888-8691; 888-2902; 887-5748; 887-2455; 624-5277; 625-9619 | Fax: (045) 888-1754; 888-2514
Email: hau@hau.edu.ph | www.hau.edu.ph

Appendix M
Pre-Interview Documentation

Seeking Evaluators for our Game Engine for the "Bullet Heaven" genre.

Aidan <aidanserrano122703@gmail.co... Mon 17 Feb, 21:38 (10 days ago)
to ghara2art

Hello we apologize if we contacted you through this email we found the LinkedIn Profile for Alessandra Bettina and it was linked to a Twitter/X profile and we sent this email because we can't send this message through there.

We are a group of computer science students from Holy Angel University in Angeles Pampanga developing "Bala: A 2D Bullet Heaven Game Engine" as our proposal capstone project, in partial fulfilment of the requirements for the degree Bachelor of Science in Computer Science. Our project aims to provide a user-friendly way for students and novice developers to create games within the "Bullet Heaven" Genre. We are currently seeking experienced game developers and studios to be Evaluator/s for our engine and provide valuable feedback, if ever it is selected as our capstone to be made in our 4th Year, 1st Semester. Your expertise would help us assess the usability, performance, and overall effectiveness of "Bala" in simplifying game development.

If you were to agree we would need name/s of people/person to be the evaluator/s for us to send a formal letter signed by our Professor.

GROUP MEMBERS:
Aidan Zabdiel A. Serrano
Julian Mathew M. Guintu
Gabriel V. Pagcu
Carlos Jose B. Ysais

Aidan <aidanserrano122703@gm... 19 Feb 2025, 20:35 (8 days ago)
to Alessandra

Hello, currently we have three titles (including this) that we will be proposing in the future. If this title gets accepted by our panelists, you would be primarily giving us feedback as an evaluator by testing our game engine and answering the survey(s) that we will make. Though our main feedback tool are surveys, this may change in the future depending on the feedback of our panelists. If possible, we would also interview you after testing the engine to get more insights that the survey might have missed.

1. Around our final period at our 1st semester next school year. We currently don't have the academic calendar for the next school year (2025-2026), but based on our current academic calendar (2024-2025), the final period of the 1st semester starts at the 3rd week of September and ends at the last week of October.

2. We would like to get your feedback during specific milestones, especially the last one. Though we have not set these milestones yet.

3. Yes, everything can be done remotely.

4. There would most likely be an online video meeting for the last feedback. Currently, the persons involved would be me and my group mates which are mentioned in the attached letter.

We have attached a scan of the formal letter in this email along with the signatures of our adviser, chairperson, and dean. We kindly ask you to sign the paper on top of your printed name along (with the date you have signed the paper under your printed name) and send us a copy of the signed letter. It is okay if you sign it digitally or by hand.

...

One attachment • Scanned by Gmail

Seeking Evaluators for our Game Engine for the "Bullet Heaven" genre.

Aidan <aidanserrano122703@gmail.co... Mon 17 Feb, 22:17 (10 days ago)
to fishbump.studio

Hello, we discovered this contact from itch.io and it seems the link for Twitter/X isn't working so we are sending this email.

We are a group of computer science students from Holy Angel University in Angeles Pampanga developing "Bala: A 2D Bullet Heaven Game Engine" as our proposal capstone project, in partial fulfilment of the requirements for the degree Bachelor of Science in Computer Science. Our project aims to provide a user-friendly way for students and novice developers to create games within the "Bullet Heaven" Genre. We are currently seeking experienced game developers and studios to be Evaluator/s for our engine and provide valuable feedback, if ever it is selected as our capstone to be made in our 4th Year, 1st Semester. Your expertise would help us assess the usability, performance, and overall effectiveness of "Bala" in simplifying game development.

If you were to agree we would need name/s of people/person to be the evaluator/s for us to send a formal letter signed by our Professor.

GROUP MEMBERS:
Aidan Zabdiel A. Serrano
Julian Mathew M. Guintu
Gabriel V. Pagcu
Carlos Jose B. Ysais

Seeking Evaluators for our Game Engine for the "Bullet Heaven" genre.

Aidan <aidanserrano122703@gmail.co... Mon 17 Feb, 11:17 (10 days ago)
to contact

Hello, we are a group of computer science students from Holy Angel University in Angeles Pampanga developing "Bala: A 2D Bullet Heaven Game Engine" as our proposal capstone project, in partial fulfilment of the requirements for the degree Bachelor of Science in Computer Science. Our project aims to provide a user-friendly way for students and novice developers to create games within the "Bullet Heaven" Genre. We are currently seeking experienced game developers and studios to be Evaluator/s for our engine and provide valuable feedback if ever it is selected as our capstone to be made in our 4th Year, 1st Semester. Your expertise would help us assess the usability, performance, and overall effectiveness of "Bala" in simplifying game development.

Aidan Zabdiel A. Serrano
Julian Mathew M. Guintu
Gabriel V. Pagcu
Carlos Jose B. Ysais

Aidan <aidanserrano122703@gmail.co... Mon 17 Feb, 11:29 (10 days ago)
to contact

Another addition, if agreed we would need names of to be evaluator/s to send formal letter signed by our Professor.

Seeking Evaluators for our Game Engine for the "Bullet Heaven" genre.

Aidan <aidanserrano122703@gmail.com> Sat 22 Feb, 17:46 (5 days ago)
to josefyenko1

Hello, we are a group of Computer Science students from Holy Angel University in Angeles Pampanga developing "Bala: A 2D Bullet Heaven Game Engine" as our proposal capstone project, in partial fulfilment of the requirements for the degree Bachelor of Science in Computer Science. Our project aims to provide a user-friendly way for novice developers to create games within the "Bullet Heaven" Genre, game most commonly known in the genre is Vampire Survivor. We are currently seeking experienced game developers and studios to be Evaluator/s for our engine and provide valuable feedback, if ever it is selected as our capstone to be made in our 4th Year, 1st Semester. Your expertise would help us assess the usability, performance, and overall effectiveness of "Bala" in simplifying game development.

If you were to agree we would need name/s of people/person to be the evaluator/s for us to send a formal letter signed by our Thesis Advisor, Chairperson, and our Dean.

GROUP MEMBERS:
Aidan Zabdiel A. Serrano
Julian Mathew M. Guintu
Gabriel V. Pagcu
Carlos Jose B. Ysais

Seeking Evaluators for our Game Engine for the "Bullet Heaven" genre.

Aidan <aidanserrano122703@gmail.com> 20:38 (0 minutes ago)
to rlux@rluxcustoms.com

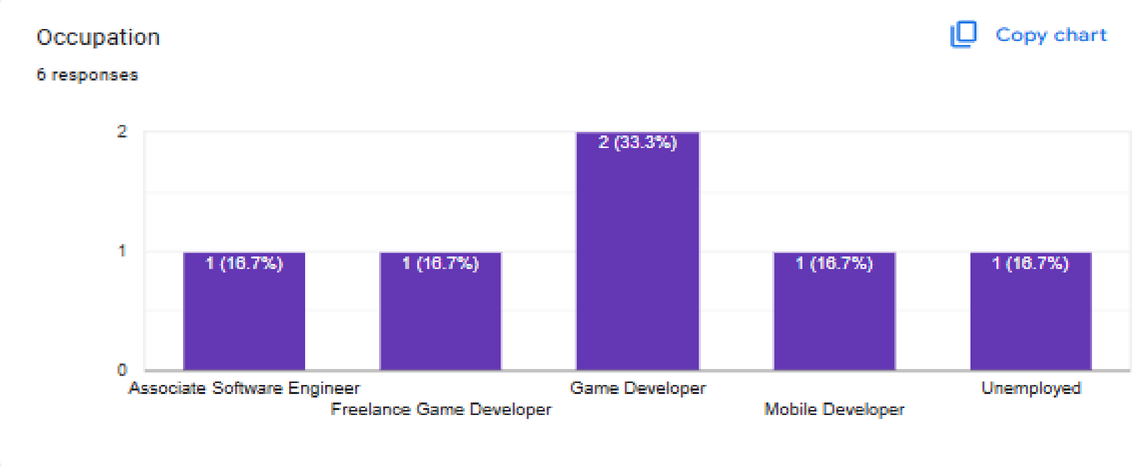
Hello, we are a group of Computer Science students from Holy Angel University in Angeles Pampanga developing "Bala: A 2D Bullet Heaven Game Engine" as our proposal capstone project, in partial fulfilment of the requirements for the degree Bachelor of Science in Computer Science. Our project aims to provide a user-friendly way for novice developers to create games within the "Bullet Heaven" Genre, game most commonly known in the genre is Vampire Survivor. We are currently seeking experienced game developers and studios to be Evaluator/s for our engine and provide valuable feedback, if ever it is selected as our capstone to be made in our 4th Year, 1st Semester. Your expertise would help us assess the usability, performance, and overall effectiveness of "Bala" in simplifying game development.

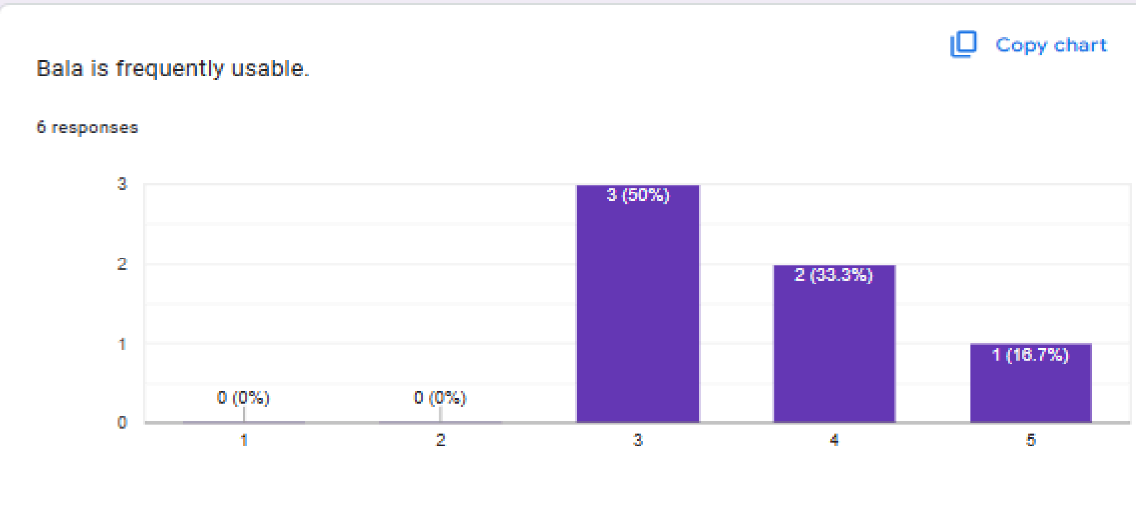
If you were to agree we would need name/s of people/person to be the evaluator/s for us to send a formal letter signed by our Thesis Advisor, Chairperson, and our Dean.

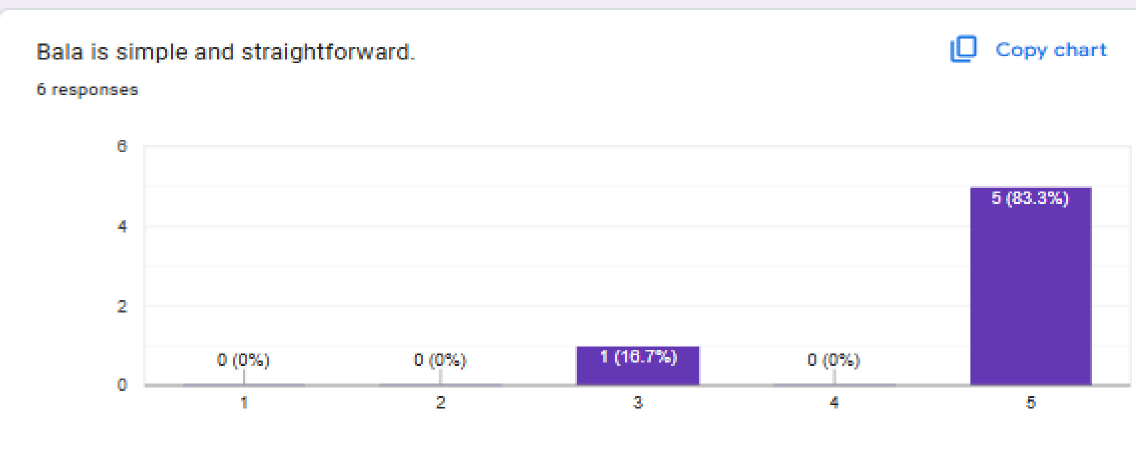
If there are any further questions we will happily answer them.

GROUP MEMBERS:
Aidan Zabdiel A. Serrano
Julian Mathew M. Guintu
Gabriel V. Pagcu
Carlos Jose B. Ysais

Appendix N
Sample Instrument Results

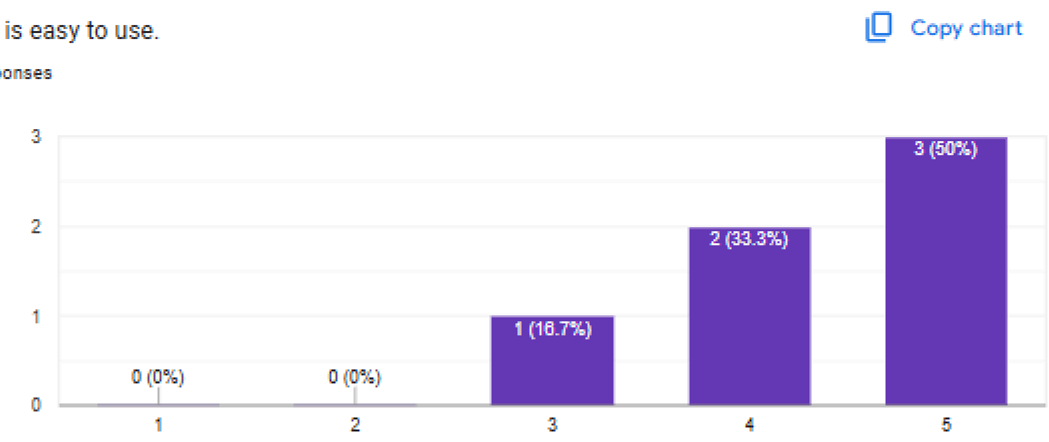






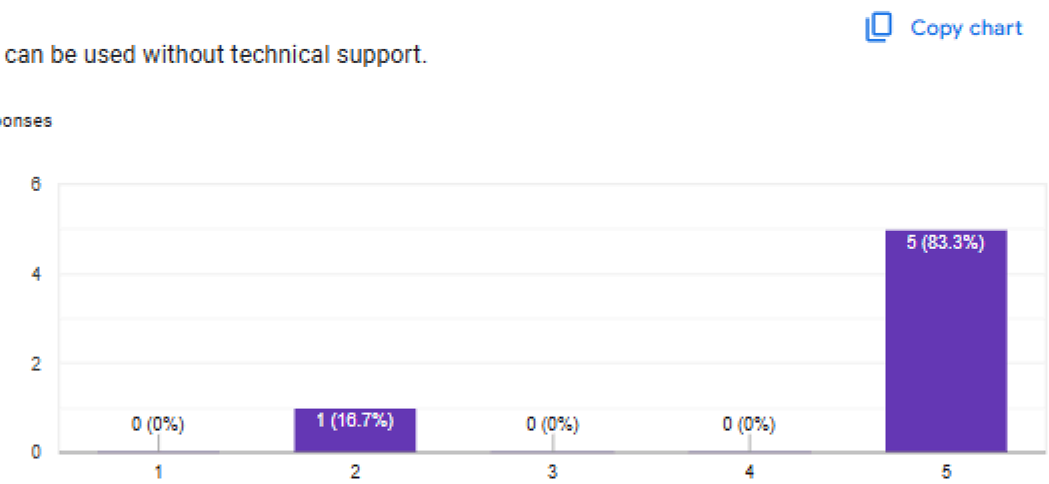
Bala is easy to use.

6 responses



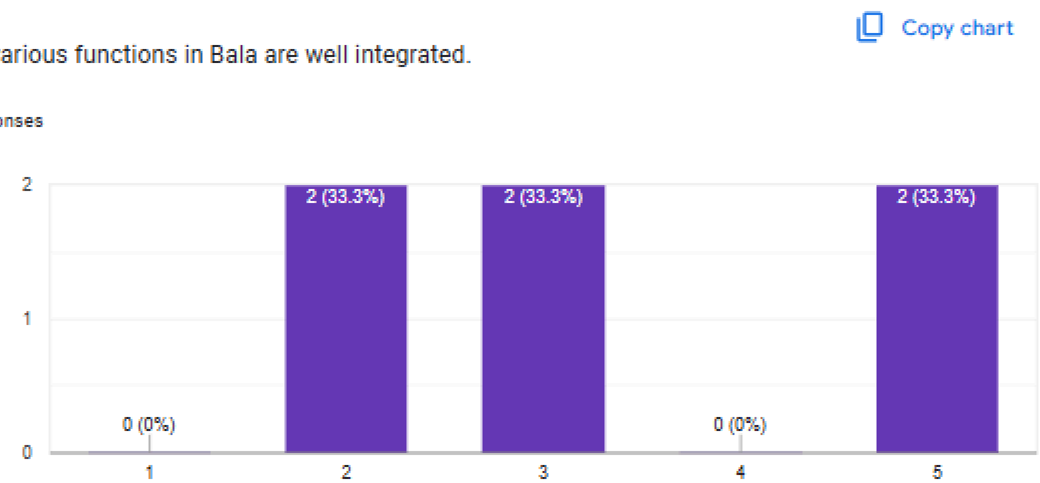
Bala can be used without technical support.

6 responses



The various functions in Bala are well integrated.

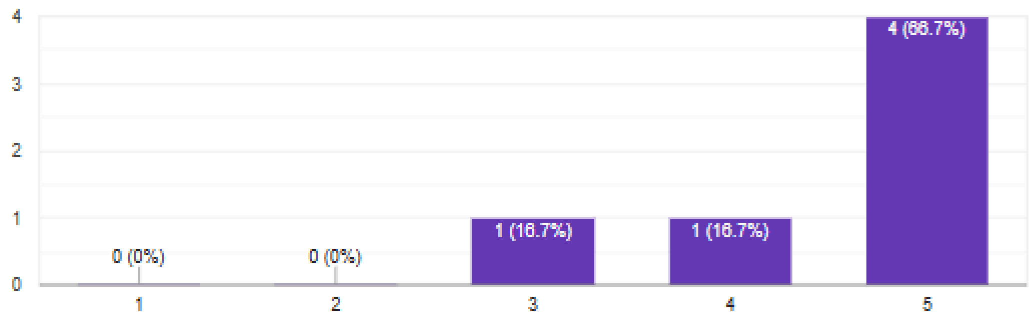
6 responses



Copy chart

Bala is consistent in its design and features.

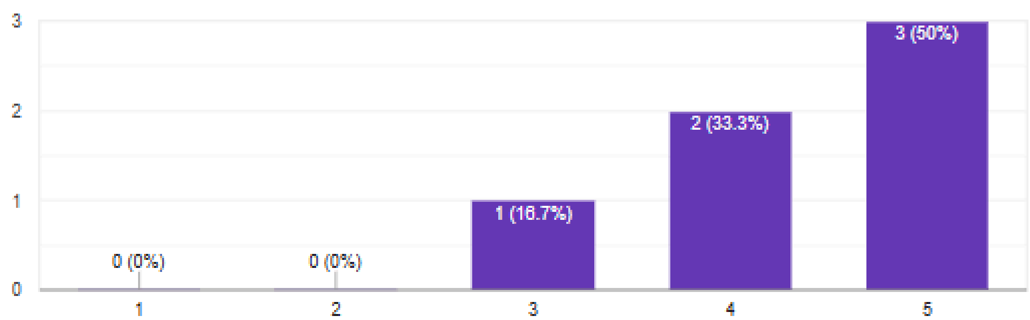
6 responses



Copy chart

Most people would learn to use Bala very quickly.

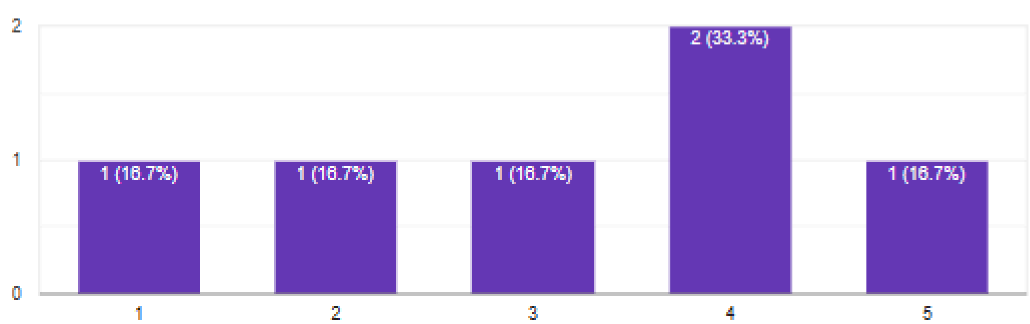
6 responses

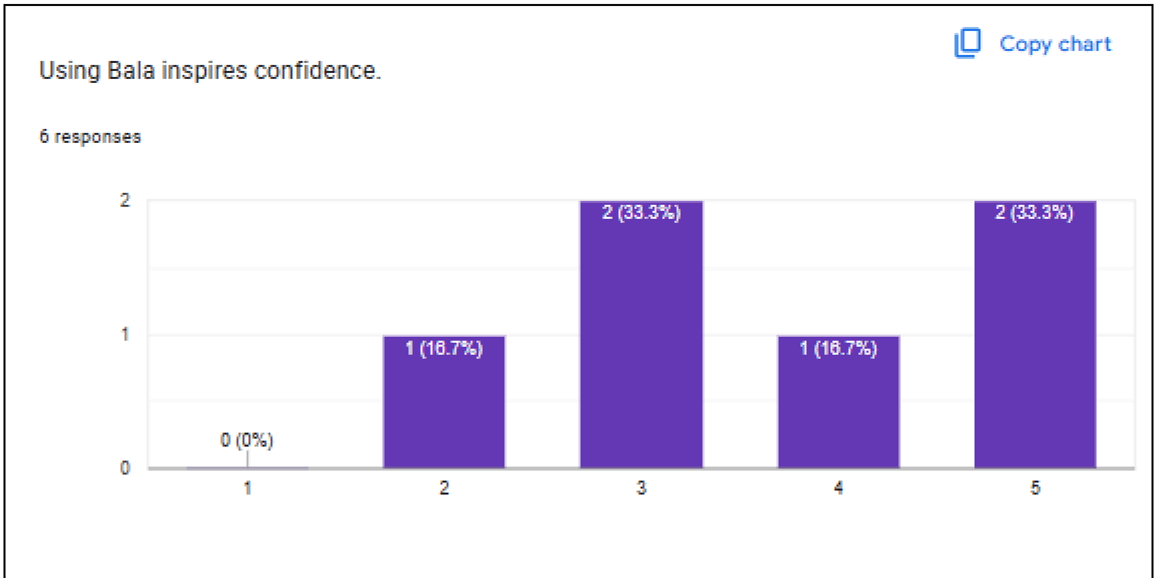


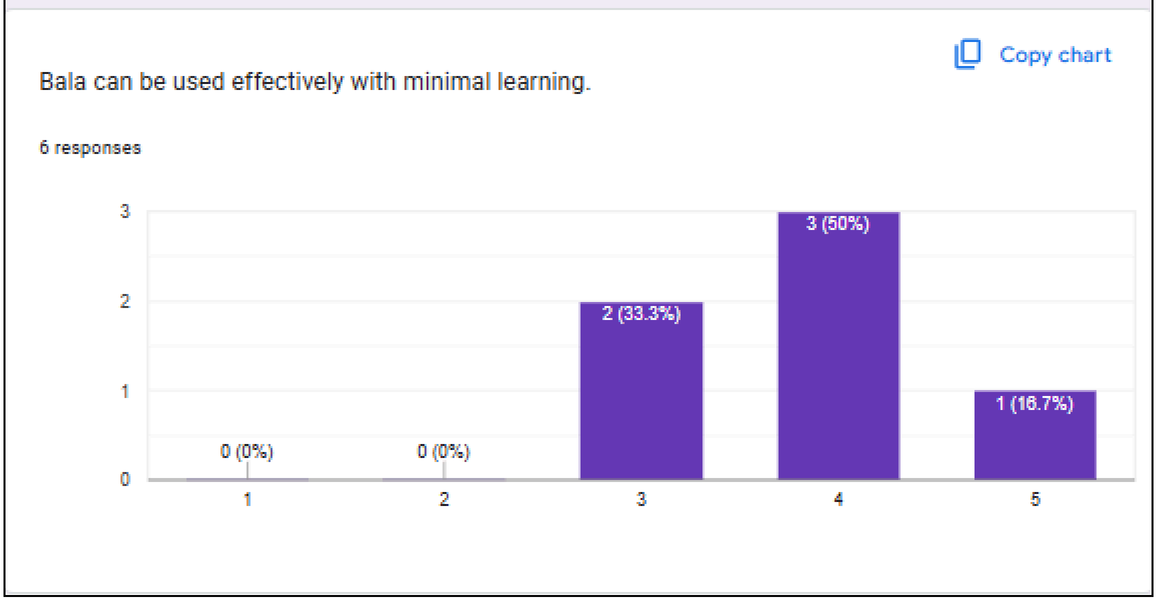
Copy chart

Bala is convenient and efficient to use.

6 responses




Copy chart
Using Bala inspires confidence.
6 responses
2
1
0
0 (0%)
1 (16.7%)
2 (33.3%)
1 (16.7%)
2 (33.3%)
1
2
3
4
5


Copy chart
Bala can be used effectively with minimal learning.
6 responses
3
2
1
0
0 (0%)
0 (0%)
2 (33.3%)
3 (50%)
1 (16.7%)
1
2
3
4
5

Appendix O

Stress Tests Raw Data

	A	B	C	D
1	Elapsed Time in Seconds	FPS	Average Frame Time in Seconds per Second	Enemy Count
2	1.4259295	38	0.0375245	0
3	1.0044508	66	0.0152190	75
4	1.0101224	73	0.0138373	150
5	1.0047936	71	0.0141520	225
6	1.0008545	72	0.0139008	300
7	1.0088227	75	0.0134510	375
8	1.0005873	74	0.0135215	450
9	1.0087719	73	0.0138188	525
10	1.0039362	70	0.0143419	600
11	1.0146369	70	0.0144948	675
12	1.0051365	68	0.0147814	750
13	1.0014597	63	0.0158962	825
14	1.0116132	65	0.0155633	900
15	1.0006501	56	0.0178688	975
16	1.0176556	52	0.0195703	1050
17	1.0188612	48	0.0212263	1125
18	1.0190348	45	0.0226452	1200
19	1.0031852	42	0.0238854	1275
20	1.0222314	41	0.0249325	1350
21	1.0041047	39	0.0257463	1425
22	1.0171098	37	0.0274895	1500
23	1.0152406	35	0.0290069	1575
24	1.0288945	34	0.0302616	1650
25	1.0259438	32	0.0320607	1725
26	1.0112472	30	0.0337082	1800
27	1.0213613	30	0.0340454	1875
28	1.0013719	28	0.0357633	1950

PBnF PBF PCnF PCF SBnF

	A	B	C	D
1	Elapsed Time in Seconds	FPS	Average Frame Time in Seconds per Second	Enemy Count
2	1.3006008	40	0.0325150	0
3	1.0017652	67	0.0149517	75
4	1.0005780	75	0.0133410	150
5	1.0150632	74	0.0137171	225
6	1.0074536	75	0.0134327	300
7	1.0090306	75	0.0134537	375
8	1.0128074	73	0.0138741	450
9	1.0052731	75	0.0134036	525
10	1.0090381	73	0.0138224	600
11	1.0101797	71	0.0142279	675
12	1.0137036	68	0.0149074	750
13	1.0019771	65	0.0154150	825
14	1.0099887	63	0.0160316	900
15	1.0032112	60	0.0167202	975
16	1.0059711	52	0.0193456	1050
17	1.0166333	49	0.0207476	1125
18	1.0089012	46	0.0219326	1200
19	1.0049549	44	0.0228399	1275
20	1.0147750	44	0.0230631	1350
21	1.0060223	42	0.0239529	1425
22	1.0026820	41	0.0244557	1500
23	1.0128922	39	0.0259716	1575
24	1.0054409	37	0.0271741	1650
25	1.0202373	36	0.0283399	1725
26	1.0081179	34	0.0296505	1800
27	1.0271737	33	0.0311265	1875
28	1.0253493	32	0.0320422	1950
29	1.0216206	30	0.0340540	2025
30	1.0118534	28	0.0361376	2100

PBnF PBF PCnF PCF SBnF

	A	B	C	D
1	Elapsed Time in Seconds	FPS	Average Frame Time in Seconds per Second	Enemy Count
2	1.8240855	56	0.0325730	0
3	1.0184323	68	0.0149769	75
4	1.0011393	74	0.0135289	150
5	1.0000287	75	0.0133337	225
6	1.0004404	73	0.0137047	300
7	1.0003798	74	0.0135186	375
8	1.0001386	74	0.0135154	450
9	1.0004661	73	0.0137050	525
10	1.0066763	70	0.0143811	600
11	1.0121636	70	0.0144595	675
12	1.0000751	67	0.0149265	750
13	1.0123379	63	0.0160689	825
14	1.0106937	60	0.0168449	900
15	1.0114854	55	0.0183906	975
16	1.0119270	50	0.0202385	1050
17	1.0053647	47	0.0213907	1125
18	1.0053014	44	0.0228478	1200
19	1.0119432	42	0.0240939	1275
20	1.0107496	40	0.0252687	1350
21	1.0047105	38	0.0264398	1425
22	1.0187912	37	0.0275349	1500
23	1.0114608	35	0.0288989	1575
24	1.0226652	33	0.0309899	1650
25	1.0023067	31	0.0323325	1725
26	1.0073421	30	0.0335781	1800
27	1.0146887	29	0.0349893	1875

PBnF PBF PCnF PCF SBnF

	A	B	C	D
1	Elapsed Time in Seconds	FPS	Average Frame Time in Seconds per Second	Enemy Count
2	1.5007547	47	0.0319310	0
3	1.0097901	68	0.0148499	75
4	1.0102086	73	0.0138385	150
5	1.0003541	73	0.0137035	225
6	1.0015672	75	0.0133542	300
7	1.0074074	75	0.0134321	375
8	1.0053689	76	0.0132285	450
9	1.0033094	75	0.0133775	525
10	1.0071028	73	0.0137959	600
11	1.0150547	71	0.0142965	675
12	1.0065944	69	0.0145883	750
13	1.0095070	67	0.0150673	825
14	1.0138725	65	0.0155980	900
15	1.0192038	59	0.0172746	975
16	1.0030120	52	0.0192887	1050
17	1.0020285	48	0.0208756	1125
18	1.0113612	45	0.0224747	1200
19	1.0087719	42	0.0240184	1275
20	1.0235939	41	0.0249657	1350
21	1.0225064	39	0.0262181	1425
22	1.0103210	38	0.0265874	1500
23	1.0171230	37	0.0274898	1575
24	1.0150772	36	0.0281966	1650
25	1.0282652	35	0.0293790	1725
26	1.0161526	34	0.0298868	1800
27	1.0116070	32	0.0316127	1875
28	1.0262756	31	0.0331057	1950
29	1.0327797	29	0.0356131	2025

SCnF SCF

	A	B	C	D
1	Elapsed Time in Seconds	FPS	Average Frame Time in Seconds per Second	Enemy Count
2	1.6810886	63	0.0266839	0
3	1.0102207	66	0.0153064	75
4	1.0215022	41	0.0249147	150
5	1.0110889	43	0.0235137	225
6	1.0267236	29	0.0354043	300

PBnF PBF

	A	B	C	D
1	Elapsed Time in Seconds	FPS	Average Frame Time in Seconds per Second	Enemy Count
2	1.6326260	49	0.0333189	0
3	1.0077808	67	0.0150415	75
4	1.0064861	52	0.0193555	150
5	1.0014733	44	0.0227608	225
6	1.0097983	34	0.0297000	300
7	1.0239400	17	0.0602318	375

SBnF SBF

	A	B	C	D
1	Elapsed Time in Seconds	FPS	Average Frame Time in Seconds per Second	Enemy Count
2	1.4073953	47	0.0299446	0
3	1.0087976	71	0.0142084	75
4	1.0170130	33	0.0308186	150
5	1.0917296	14	0.0779807	225

PCnF PCF

Elapsed Time in Seconds	FPS	Average Frame Time in Seconds per Second	Enemy Count
1.3871126	42	0.0330265	0
1.0003489	66	0.0151568	75
1.0308966	31	0.0332547	150
1.0417995	14	0.0744142	225

SCnF SCF

Appendix P

Performance Tests Raw Data

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
2	8.8083900	0.9821099	2025
3	8.8725701	0.0641801	2025
4	8.9164461	0.0438760	2025
5	8.9532825	0.0368364	2025
6	8.9870443	0.0337618	2025
7	9.0223229	0.0352786	2025
8	9.0555349	0.0332120	2025
9	9.0860558	0.0305209	2025
10	9.1203200	0.0342642	2025
11	9.1536688	0.0333488	2025
12	9.1890019	0.0353331	2025
13	9.2212039	0.0322020	2025
14	9.2621389	0.0409350	2025
15	9.2987233	0.0365844	2025
16	9.3342647	0.0355414	2025
17	9.3702047	0.0359400	2025
18	9.4032936	0.0330889	2025
19	9.4336955	0.0304019	2025
20	9.4636954	0.0299999	2025
21	9.4912001	0.0275047	2025
22	9.5182517	0.0270516	2025
23	9.5433417	0.0250900	2025
24	9.5685018	0.0251601	2025
25	9.5938674	0.0253656	2025
26	9.6389293	0.0450619	2025
27	9.6637256	0.0247963	2025
28	9.6904809	0.0267553	2025
29	9.7174188	0.0269379	2025
30	9.7624191	0.0450003	2025
31	9.7858755	0.0234564	2025
32	9.8105737	0.0246982	2025
33	9.8387010	0.0281273	2025
34	9.8643774	0.0256764	2025
35	9.8874375	0.0230601	2025

\+ ≡ PBnF PBF PCnF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
1969	92.3917162	0.0505279	2025
1970	92.4428053	0.0510891	2025
1971	92.4934743	0.0506690	2025
1972	92.5444313	0.0509570	2025
1973	92.5944144	0.0499831	2025
1974	92.6461474	0.0517330	2025
1975	92.6973464	0.0511990	2025
1976	92.7475915	0.0502451	2025
1977	92.8095814	0.0619899	2025
1978	92.8602540	0.0506726	2025
1979	92.9189844	0.0587304	2025
1980	92.9682753	0.0492909	2025
1981	93.0183715	0.0500962	2025
1982	93.0677093	0.0493378	2025
1983	93.1168945	0.0491852	2025
1984	93.1666428	0.0497483	2025
1985	93.2149983	0.0483555	2025
1986	93.2639052	0.0489069	2025
1987	93.3127842	0.0488790	2025
1988	93.3635585	0.0507743	2025
1989	93.4155394	0.0519809	2025
1990	93.4674780	0.0519386	2025
1991	93.5192700	0.0517920	2025
1992	93.5779864	0.0587164	2025
1993	93.6453160	0.0673296	2025
1994	93.7131387	0.0678227	2025
1995	93.7820735	0.0689348	2025
1996	93.8368047	0.0547312	2025
1997	93.8890378	0.0522331	2025
1998	93.9421952	0.0531574	2025
1999	93.9949341	0.0527389	2025
2000	94.0474913	0.0525572	2025
2001	94.1004981	0.0530068	2025

+ ≡ PBnF PBF PCnF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
2	8.3288306	0.7984609	2025
3	8.5370455	0.2082149	2025
4	8.6841185	0.1470730	2025
5	8.8276237	0.1435052	2025
6	8.9723373	0.1447136	2025
7	9.1102801	0.1379428	2025
8	9.2476024	0.1373223	2025
9	9.3920105	0.1444081	2025
10	9.5372570	0.1452465	2025
11	9.6822495	0.1449925	2025
12	9.8513113	0.1690618	2025
13	10.0003296	0.1490183	2025
14	10.1444417	0.1441121	2025
15	10.2948335	0.1503918	2025
16	10.4446214	0.1497879	2025
17	10.5841571	0.1395357	2025
18	10.7240526	0.1398955	2025
19	10.8690865	0.1450339	2025
20	11.0052562	0.1361697	2025
21	11.1378265	0.1325703	2025
22	11.2742313	0.1364048	2025
23	11.4124909	0.1382596	2025
24	11.5455011	0.1330102	2025
25	11.6808318	0.1353307	2025
26	11.8161585	0.1353267	2025
27	11.9509206	0.1347621	2025
28	12.0848039	0.1338833	2025
29	12.2212299	0.1364260	2025
30	12.3583935	0.1371636	2025
31	12.4936939	0.1353004	2025
32	12.6270779	0.1333840	2025
33	12.7665360	0.1394581	2025
34	12.9010516	0.1345156	2025
35	13.0392739	0.1382223	2025

PBnF PBF PCnF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
69	17.8611863	0.1635871	2025
70	18.0174170	0.1562307	2025
71	18.1730282	0.1556112	2025
72	18.3400875	0.1670593	2025
73	18.5029186	0.1628311	2025
74	18.6610484	0.1581298	2025
75	18.8167462	0.1556978	2025
76	18.9922737	0.1755275	2025
77	19.1479403	0.1556666	2025
78	19.3036501	0.1557098	2025
79	19.4595322	0.1558821	2025
80	19.6244582	0.1649260	2025
81	19.7906205	0.1661623	2025
82	19.9529618	0.1623413	2025
83	20.1087170	0.1557552	2025
84	20.2667509	0.1580339	2025
85	20.4232773	0.1565264	2025
86	20.5786284	0.1553511	2025
87	20.7412260	0.1625976	2025
88	20.9053241	0.1640981	2025
89	21.0655046	0.1601805	2025
90	21.2255606	0.1600560	2025
91	21.3808783	0.1553177	2025
92	21.5380431	0.1571648	2025
93	21.6980690	0.1600259	2025
94	21.8628897	0.1648207	2025
95	22.0331661	0.1702764	2025
96	22.2251088	0.1919427	2025
97	22.4209961	0.1958873	2025
98	22.7347764	0.3137803	2025
99	23.3672932	0.6325168	2025
100	24.0705201	0.7032269	2025
101	27.0811669	3.0106468	2025

PBnF PBF PCnF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
2	3.6734982	0.9599911	2025
3	3.7347762	0.0612780	2025
4	3.7652231	0.0304469	2025
5	3.7931093	0.0278862	2025
6	3.8180383	0.0249290	2025
7	3.8409143	0.0228760	2025
8	3.8622998	0.0213855	2025
9	3.8845337	0.0222339	2025
10	3.9065990	0.0220653	2025
11	3.9278656	0.0212666	2025
12	3.9489534	0.0210878	2025
13	3.9714018	0.0224484	2025
14	3.9929251	0.0215233	2025
15	4.0244127	0.0314876	2025
16	4.0511536	0.0267409	2025
17	4.0856397	0.0344861	2025
18	4.1142163	0.0285766	2025
19	4.1405597	0.0263434	2025
20	4.1674640	0.0269043	2025
21	4.1923167	0.0248527	2025
22	4.2190282	0.0267115	2025
23	4.2490923	0.0300641	2025
24	4.2777251	0.0286328	2025
25	4.3037774	0.0260523	2025
26	4.3302834	0.0265060	2025
27	4.3586853	0.0284019	2025
28	4.3865708	0.0278855	2025
29	4.4150771	0.0285063	2025
30	4.4392372	0.0241601	2025
31	4.4628407	0.0236035	2025
32	4.4872662	0.0244255	2025
33	4.5143937	0.0271275	2025
34	4.5402395	0.0258458	2025
35	4.5693891	0.0291496	2025

PBnF PBF PCnF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
1137	47.0573065	0.0390969	2025
1138	47.0979698	0.0406633	2025
1139	47.1376473	0.0396775	2025
1140	47.1769634	0.0393161	2025
1141	47.2172543	0.0402909	2025
1142	47.2569304	0.0396761	2025
1143	47.2961894	0.0392590	2025
1144	47.3357467	0.0395573	2025
1145	47.3747377	0.0389910	2025
1146	47.4140767	0.0393390	2025
1147	47.4539459	0.0398692	2025
1148	47.4931347	0.0391888	2025
1149	47.5322691	0.0391344	2025
1150	47.5714337	0.0391646	2025
1151	47.6124085	0.0409748	2025
1152	47.6521367	0.0397282	2025
1153	47.6915049	0.0393682	2025
1154	47.7316441	0.0401392	2025
1155	47.7705175	0.0388734	2025
1156	47.8106119	0.0400944	2025
1157	47.8498483	0.0392364	2025
1158	47.8892958	0.0394475	2025
1159	47.9294174	0.0401216	2025
1160	47.9684341	0.0390167	2025
1161	48.0079467	0.0395126	2025
1162	48.0477045	0.0397578	2025
1163	48.0868344	0.0391299	2025
1164	48.1260883	0.0392539	2025
1165	48.1655610	0.0394727	2025
1166	48.2052518	0.0396908	2025
1167	48.2443657	0.0391139	2025
1168	48.2833342	0.0389685	2025
1169	48.3227110	0.0393768	2025

+ ≡ PBnF PBF PCnF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
2	3.9632398	1.2944969	2025
3	4.2568232	0.2935834	2025
4	4.4477796	0.1909564	2025
5	4.6236808	0.1759012	2025
6	4.8131022	0.1894214	2025
7	5.0223851	0.2092829	2025
8	5.1917292	0.1693441	2025
9	5.3487036	0.1569744	2025
10	5.5023555	0.1536519	2025
11	5.6730413	0.1706858	2025
12	5.8427783	0.1697370	2025
13	6.0042249	0.1614466	2025
14	6.1574188	0.1531939	2025
15	6.3107620	0.1533432	2025
16	6.4630566	0.1522946	2025
17	6.6168776	0.1538210	2025
18	6.7723104	0.1554328	2025
19	6.9240134	0.1517030	2025
20	7.0698868	0.1458734	2025
21	7.2168374	0.1469506	2025
22	7.3653370	0.1484996	2025
23	7.5146115	0.1492745	2025
24	7.6642181	0.1496066	2025
25	7.8166507	0.1524326	2025
26	7.9652514	0.1486007	2025
27	8.1171422	0.1518908	2025
28	8.2613188	0.1441766	2025
29	8.4065256	0.1452068	2025
30	8.5509517	0.1444261	2025
31	8.6964703	0.1455186	2025
32	8.8447610	0.1482907	2025
33	8.9919798	0.1472188	2025
34	9.1389875	0.1470077	2025
35	9.2848665	0.1458790	2025

PCF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
53	11.9784430	0.1489823	2025
54	12.1366539	0.1582109	2025
55	12.2856842	0.1490303	2025
56	12.4374059	0.1517217	2025
57	12.5873093	0.1499034	2025
58	12.7438833	0.1565740	2025
59	12.8963123	0.1524290	2025
60	13.0569728	0.1606605	2025
61	13.2130863	0.1561135	2025
62	13.3634521	0.1503658	2025
63	13.5329387	0.1694866	2025
64	13.6987587	0.1658200	2025
65	13.8603039	0.1615452	2025
66	14.0204162	0.1601123	2025
67	14.1784593	0.1580431	2025
68	14.3398787	0.1614194	2025
69	14.4988044	0.1589257	2025
70	14.6550704	0.1562660	2025
71	14.8161047	0.1610343	2025
72	14.9738756	0.1577709	2025
73	15.1307177	0.1568421	2025
74	15.2975884	0.1668707	2025
75	15.4599612	0.1623728	2025
76	15.6203084	0.1603472	2025
77	15.7849589	0.1646505	2025
78	15.9497555	0.1647966	2025
79	16.1066806	0.1569251	2025
80	16.2760368	0.1693562	2025
81	16.4367201	0.1606833	2025
82	16.5941805	0.1574604	2025
83	16.7529707	0.1587902	2025
84	16.9141111	0.1611404	2025
85	17.0756307	0.1615196	2025
86	17.2770146	0.2013839	2025

\+ ≡ PCF ▾ S

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
2	5.2683518	1.0664388	2025
3	5.3258311	0.0574793	2025
4	5.3697496	0.0439185	2025
5	5.4086925	0.0389429	2025
6	5.4437190	0.0350265	2025
7	5.4725908	0.0288718	2025
8	5.5073499	0.0347591	2025
9	5.5385484	0.0311985	2025
10	5.5729387	0.0343903	2025
11	5.6046800	0.0317413	2025
12	5.6369272	0.0322472	2025
13	5.6837394	0.0468122	2025
14	5.7163433	0.0326039	2025
15	5.7482500	0.0319067	2025
16	5.7771689	0.0289189	2025
17	5.8280466	0.0508777	2025
18	5.8582443	0.0301977	2025
19	5.8884937	0.0302494	2025
20	5.9183378	0.0298441	2025
21	5.9461894	0.0278516	2025
22	5.9723525	0.0261631	2025
23	5.9990990	0.0267465	2025
24	6.0243467	0.0252477	2025
25	6.0482912	0.0239445	2025
26	6.0724427	0.0241515	2025
27	6.0976879	0.0252452	2025
28	6.1246897	0.0270018	2025
29	6.1499939	0.0253042	2025
30	6.1751406	0.0251467	2025
31	6.2002768	0.0251362	2025
32	6.2249806	0.0247038	2025
33	6.2492440	0.0242634	2025
34	6.2730028	0.0237588	2025
35	6.2998080	0.0268052	2025

SBnF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
1968	85.1252497	0.0435193	2025
1969	85.1694679	0.0442182	2025
1970	85.2125414	0.0430735	2025
1971	85.2564359	0.0438945	2025
1972	85.3016866	0.0452507	2025
1973	85.3460257	0.0443391	2025
1974	85.3898682	0.0438425	2025
1975	85.4344289	0.0445607	2025
1976	85.4868380	0.0524091	2025
1977	85.5307740	0.0439360	2025
1978	85.5746736	0.0438996	2025
1979	85.6194006	0.0447270	2025
1980	85.6632314	0.0438308	2025
1981	85.7083439	0.0451125	2025
1982	85.7525295	0.0441856	2025
1983	85.7967367	0.0442072	2025
1984	85.8415813	0.0448446	2025
1985	85.8858624	0.0442811	2025
1986	85.9300618	0.0441994	2025
1987	85.9739751	0.0439133	2025
1988	86.0177431	0.0437680	2025
1989	86.0618775	0.0441344	2025
1990	86.1062702	0.0443927	2025
1991	86.1497865	0.0435163	2025
1992	86.2016352	0.0518487	2025
1993	86.2454823	0.0438471	2025
1994	86.2897471	0.0442648	2025
1995	86.3332800	0.0435329	2025
1996	86.3776680	0.0443880	2025
1997	86.4228360	0.0451680	2025
1998	86.4666662	0.0438302	2025
1999	86.5116443	0.0449781	2025
2000	86.5563346	0.0446903	2025
2001	86.6014878	0.0451532	2025

SBnF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
2	3.8971606	1.4713292	2025
3	4.1586103	0.2614497	2025
4	4.4514267	0.2928164	2025
5	4.6517810	0.2003543	2025
6	4.8010136	0.1492326	2025
7	4.9401521	0.1391385	2025
8	5.0845448	0.1443927	2025
9	5.2617366	0.1771918	2025
10	5.4763082	0.2145716	2025
11	5.6477892	0.1714810	2025
12	5.8120574	0.1642682	2025
13	5.9633123	0.1512549	2025
14	6.1157190	0.1524067	2025
15	6.2619138	0.1461948	2025
16	6.4069695	0.1450557	2025
17	6.5516229	0.1446534	2025
18	6.6964310	0.1448081	2025
19	6.8410318	0.1446008	2025
20	6.9853388	0.1443070	2025
21	7.1128920	0.1275532	2025
22	7.2472143	0.1343223	2025
23	7.3779883	0.1307740	2025
24	7.5043647	0.1263764	2025
25	7.6338917	0.1295270	2025
26	7.7635378	0.1296461	2025
27	7.8911640	0.1276262	2025
28	8.0178615	0.1266975	2025
29	8.1465117	0.1286502	2025
30	8.2730581	0.1265464	2025
31	8.4026200	0.1295619	2025
32	8.5296422	0.1270222	2025
33	8.6593885	0.1297463	2025
34	8.7890624	0.1296739	2025
35	8.9175263	0.1284639	2025

SBF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
69	13.3569546	0.1287163	2025
70	13.4853314	0.1283768	2025
71	13.6129507	0.1276193	2025
72	13.7406462	0.1276955	2025
73	13.8708402	0.1301940	2025
74	14.0007164	0.1298762	2025
75	14.1329136	0.1321972	2025
76	14.2630298	0.1301162	2025
77	14.3936629	0.1306331	2025
78	14.5240522	0.1303893	2025
79	14.6526395	0.1285873	2025
80	14.7813483	0.1287088	2025
81	14.9122200	0.1308717	2025
82	15.0405566	0.1283366	2025
83	15.1738365	0.1332799	2025
84	15.3060863	0.1322498	2025
85	15.4342767	0.1281904	2025
86	15.5647738	0.1304971	2025
87	15.6946051	0.1298313	2025
88	15.8231820	0.1285769	2025
89	15.9711445	0.1479625	2025
90	16.1051196	0.1339751	2025
91	16.2338692	0.1287496	2025
92	16.3647198	0.1308506	2025
93	16.4944029	0.1296831	2025
94	16.6393433	0.1449404	2025
95	16.7868671	0.1475238	2025
96	17.0353495	0.2484824	2025
97	17.2489559	0.2136064	2025
98	17.5113973	0.2624414	2025
99	17.8043178	0.2929205	2025
100	18.3522803	0.5479625	2025
101	19.3325431	0.9802628	2025
102	22.3056114	2.9730683	2025

\+ ≡ SBF S < >

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
2	23.6610915	0.9157049	2025
3	23.7186107	0.0575192	2025
4	23.7492701	0.0306594	2025
5	23.7757998	0.0265297	2025
6	23.8004467	0.0246469	2025
7	23.8222972	0.0218505	2025
8	23.8445963	0.0222991	2025
9	23.8666893	0.0220930	2025
10	23.8885258	0.0218365	2025
11	23.9038119	0.0152861	2025
12	23.9245116	0.0206997	2025
13	23.9449246	0.0204130	2025
14	23.9662158	0.0212912	2025
15	23.9877019	0.0214861	2025
16	24.0144017	0.0266998	2025
17	24.0375947	0.0231930	2025
18	24.0618316	0.0242369	2025
19	24.0848486	0.0230170	2025
20	24.1075572	0.0227086	2025
21	24.1308951	0.0233379	2025
22	24.1554788	0.0245837	2025
23	24.1801525	0.0246737	2025
24	24.2050471	0.0248946	2025
25	24.2307259	0.0256788	2025
26	24.2578138	0.0270879	2025
27	24.2829564	0.0251426	2025
28	24.3088584	0.0259020	2025
29	24.3349190	0.0260606	2025
30	24.3601340	0.0252150	2025
31	24.3858273	0.0256933	2025
32	24.4117442	0.0259169	2025
33	24.4366211	0.0248769	2025
34	24.4620565	0.0254354	2025
35	24.4868930	0.0248365	2025

SCnF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
1406	77.0867652	0.0461777	2025
1407	77.1274961	0.0407309	2025
1408	77.1681451	0.0406490	2025
1409	77.2100290	0.0418839	2025
1410	77.2508626	0.0408336	2025
1411	77.2916358	0.0407732	2025
1412	77.3327777	0.0411419	2025
1413	77.3735142	0.0407365	2025
1414	77.4148605	0.0413463	2025
1415	77.4548082	0.0399477	2025
1416	77.4963321	0.0415239	2025
1417	77.5363946	0.0400625	2025
1418	77.5787402	0.0423456	2025
1419	77.6199871	0.0412469	2025
1420	77.6621108	0.0421237	2025
1421	77.7037164	0.0416056	2025
1422	77.7439363	0.0402199	2025
1423	77.7855043	0.0415680	2025
1424	77.8260149	0.0405106	2025
1425	77.8679972	0.0419823	2025
1426	77.9080604	0.0400632	2025
1427	77.9485064	0.0404460	2025
1428	77.9900997	0.0415933	2025
1429	78.0315760	0.0414763	2025
1430	78.0728736	0.0412976	2025
1431	78.1148555	0.0419819	2025
1432	78.1570902	0.0422347	2025
1433	78.1972294	0.0401392	2025
1434	78.2388651	0.0416357	2025
1435	78.2789976	0.0401325	2025
1436	78.3205842	0.0415866	2025
1437	78.3617146	0.0411304	2025
1438	78.4031901	0.0414755	2025
1439	78.4452679	0.0420778	2025

SCnF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
2	6.1496463	0.9584984	2025
3	6.3590404	0.2093941	2025
4	6.5133639	0.1543235	2025
5	6.6590032	0.1456393	2025
6	6.8035261	0.1445229	2025
7	6.9477332	0.1442071	2025
8	7.0898618	0.1421286	2025
9	7.2340483	0.1441865	2025
10	7.3818732	0.1478249	2025
11	7.5300049	0.1481317	2025
12	7.7091614	0.1791565	2025
13	7.8627202	0.1535588	2025
14	8.0177798	0.1550596	2025
15	8.1760617	0.1582819	2025
16	8.3289449	0.1528832	2025
17	8.4794972	0.1505523	2025
18	8.6244454	0.1449482	2025
19	8.7732155	0.1487701	2025
20	8.9197887	0.1465732	2025
21	9.0678504	0.1480617	2025
22	9.2174164	0.1495660	2025
23	9.3612657	0.1438493	2025
24	9.5103268	0.1490611	2025
25	9.6572425	0.1469157	2025
26	9.8025065	0.1452640	2025
27	9.9479692	0.1454627	2025
28	10.0900044	0.1420352	2025
29	10.2355357	0.1455313	2025
30	10.3814475	0.1459118	2025
31	10.5267514	0.1453039	2025
32	10.6689300	0.1421786	2025
33	10.8120937	0.1431637	2025
34	10.9601879	0.1480942	2025
35	11.1024222	0.1422343	2025

SCF

	A	B	C
1	Time in Seconds	Exact Frame Time in Seconds	Enemy Count
57	14.3039630	0.1423484	2025
58	14.4597717	0.1558087	2025
59	14.6236751	0.1639034	2025
60	14.7870793	0.1634042	2025
61	14.9498119	0.1627326	2025
62	15.1183579	0.1685460	2025
63	15.2823368	0.1639789	2025
64	15.4435528	0.1612160	2025
65	15.6307847	0.1872319	2025
66	15.7937781	0.1629934	2025
67	15.9601299	0.1663518	2025
68	16.1240837	0.1639538	2025
69	16.2889450	0.1648613	2025
70	16.4480302	0.1590852	2025
71	16.6078164	0.1597862	2025
72	16.7740122	0.1661958	2025
73	16.9378331	0.1638209	2025
74	17.1009241	0.1630910	2025
75	17.2656872	0.1647631	2025
76	17.4289526	0.1632654	2025
77	17.5937326	0.1647800	2025
78	17.7564151	0.1626825	2025
79	17.9200332	0.1636181	2025
80	18.0843151	0.1642819	2025
81	18.2465536	0.1622385	2025
82	18.4129813	0.1664277	2025
83	18.5738514	0.1608701	2025
84	18.7374723	0.1636209	2025
85	18.8984280	0.1609557	2025
86	19.0590859	0.1606579	2025
87	19.2240411	0.1649552	2025
88	19.3869864	0.1629453	2025
89	19.5522927	0.1653063	2025
90	19.7471273	0.1948346	2025

SCF

Appendix Q

Paired T-Test Calculation

	A	B	C	D	E	F	G	H
1	PBnF	PBF	A-B	Mean of C	(C-D2)^2	Summation of E	Sample Size	Degrees of freedom
2	982.1099	798.4609	183.65	-158.09	116786.14	8757535.45	100	99
3	64.1801	208.2149	-144.03		197.57			
4	43.8760	147.0730	-103.20		3013.34		Variance	Standard Deviation
5	36.8364	143.5052	-106.67		2644.23		88459.95	297.42
6	33.7618	144.7136	-110.95		2222.09			
7	35.2786	137.9428	-102.66		3072.12		Paired Samples T-Test Statistic	
8	33.2120	137.3223	-104.11		2913.90		-5.315	
9	30.5209	144.4081	-113.89		1953.96			
10	34.2642	145.2465	-110.98		2219.22			
11	33.3488	144.9925	-111.64		2157.34			
12	35.3331	169.0618	-133.73		593.52			
13	32.2020	149.0183	-116.82		1703.59			
14	40.9350	144.1121	-103.18		3015.52			
15	36.5844	150.3918	-113.81		1961.03			
16	35.5414	149.7879	-114.25		1922.33			
17	35.9400	139.5357	-103.60		2969.72			
18	33.0889	139.8955	-106.81		2630.08			
19	30.4019	145.0339	-114.63		1888.67			
20	29.9999	136.1697	-106.17		2695.80			
21	27.5047	132.5703	-105.07		2811.68			
22	27.0516	136.4048	-109.35		2375.36			
23	25.0900	138.2596	-113.17		2017.92			
24	25.1601	133.0102	-107.85		2524.13			
25	25.3656	135.3307	-109.97		2316.09			
26	45.0619	135.3267	-90.26		4600.38			
27	24.7963	134.7621	-109.97		2316.02			
28	26.7553	133.8833	-107.13		2597.21			
29	26.9379	136.4260	-109.49		2362.23			
30	45.0003	137.1636	-92.16		4346.44			
31	23.4564	135.3004	-111.84		2138.77			
32	24.6982	133.3840	-108.69		2440.86			
33	28.1273	139.4581	-111.33		2186.50			
34	25.6764	134.5156	-108.84		2425.73			
35	23.0601	138.2223	-115.16		1842.87			

PBnF vs PBF | PBnF vs PCnF | PBnF vs PCF | PBnF vs SBnF

	A	B	C	D	E	F	G	H
1	PBnF	PCnF	A-B	Mean of C	(C-D2)^2	Summation of E	Sample Size	Degrees of freedom
2	982.1099	959.9911	22.12	-0.10	493.63	23025.95	1168	1167
3	64.1801	61.2780	2.90		9.01			
4	43.8760	30.4469	13.43		183.01		Variance	Standard Deviation
5	36.8364	27.8862	8.95		81.89		19.73	4.44
6	33.7618	24.9290	8.83		79.78			
7	35.2786	22.8760	12.40		156.29		Paired Samples T-Test Statistic	
8	33.2120	21.3855	11.83		142.22		-0.762	
9	30.5209	22.2339	8.29		70.33			
10	34.2642	22.0653	12.20		151.24			
11	33.3488	21.2666	12.08		148.38			
12	35.3331	21.0878	14.25		205.76			
13	32.2020	22.4484	9.75		97.07			
14	40.9350	21.5233	19.41		380.67			
15	36.5844	31.4876	5.10		27.00			
16	35.5414	26.7409	8.80		79.20			
17	35.9400	34.4861	1.45		2.41			
18	33.0889	28.5766	4.51		21.26			
19	30.4019	26.3434	4.06		17.29			
20	29.9999	26.9043	3.10		10.21			
21	27.5047	24.8527	2.65		7.57			
22	27.0516	26.7115	0.34		0.19			
23	25.0900	30.0641	-4.97		23.77			
24	25.1601	28.6328	-3.47		11.38			
25	25.3656	26.0523	-0.69		0.35			
26	45.0619	26.5060	18.56		348.01			
27	24.7963	28.4019	-3.61		12.30			
28	26.7553	27.8855	-1.13		1.06			
29	26.9379	28.5063	-1.57		2.16			
30	45.0003	24.1601	20.84		438.45			
31	23.4564	23.6035	-0.15		0.00			
32	24.6982	24.4255	0.27		0.14			
33	28.1273	27.1275	1.00		1.21			
34	25.6764	25.8458	-0.17		0.00			
35	23.0601	29.1496	-6.09		35.89			

PBnF vs PBF | PBnF vs PCnF | PBnF vs PCF | PBnF vs SBnF

	A	B	C	D	E	F	G	H
1	PBnF	PCF	A-B	Mean of C	(C-D2)^2	Summation of E	Sample Size	Degrees of freedom
2	982.1099	1294.4969	-312.39	-132.49	32364.73	53993.49	85	84
3	64.1801	293.5834	-229.40		9393.16			
4	43.8760	190.9564	-147.08		213.03		Variance	Standard Deviation
5	36.8364	175.9012	-139.06		43.29		642.78	25.35
6	33.7618	189.4214	-155.66		537.06			
7	35.2786	209.2829	-174.00		1723.85		Paired Samples T-Test Statistic	
8	33.2120	169.3441	-136.13		13.30		-48.178	
9	30.5209	156.9744	-126.45		36.38			
10	34.2642	153.6519	-119.39		171.54			
11	33.3488	170.6858	-137.34		23.54			
12	35.3331	169.7370	-134.40		3.68			
13	32.2020	161.4466	-129.24		10.50			
14	40.9350	153.1939	-112.26		409.10			
15	36.5844	153.3432	-116.76		247.31			
16	35.5414	152.2946	-116.75		247.49			
17	35.9400	153.8210	-117.88		213.28			
18	33.0889	155.4328	-122.34		102.84			
19	30.4019	151.7030	-121.30		125.08			
20	29.9999	145.8734	-115.87		275.94			
21	27.5047	146.9506	-119.45		170.02			
22	27.0516	148.4996	-121.45		121.82			
23	25.0900	149.2745	-124.18		68.90			
24	25.1601	149.6066	-124.45		64.62			
25	25.3656	152.4326	-127.07		29.35			
26	45.0619	148.6007	-103.54		837.88			
27	24.7963	151.8908	-127.09		29.06			
28	26.7553	144.1766	-117.42		226.92			
29	26.9379	145.2068	-118.27		202.10			
30	45.0003	144.4261	-99.43		1092.91			
31	23.4564	145.5186	-122.06		108.63			
32	24.6982	148.2907	-123.59		79.08			
33	28.1273	147.2188	-119.09		179.39			
34	25.6764	147.0077	-121.33		124.41			
35	23.0601	145.8790	-122.82		93.43			

PBnF vs PBF | PBnF vs PCnF | PBnF vs PCF | PBnF vs SBnF

	A	B	C	D	E	F	G	H
1	PBnF	SBnF	A-B	Mean of C	(C-D2)^2	Summation of E	Sample Size	Degrees of freedom
2	982.1099	1066.4388	-84.33	1.94	7441.86	61505.95	2000	1999
3	64.1801	57.4793	6.70		22.69			
4	43.8760	43.9185	-0.04		3.92		Variance	Standard Deviation
5	36.8364	38.9429	-2.11		16.35		30.77	5.55
6	33.7618	35.0265	-1.26		10.25			
7	35.2786	28.8718	6.41		19.98		Paired Samples T-Test Statistic	
8	33.2120	34.7591	-1.55		12.14		15.619	
9	30.5209	31.1985	-0.68		6.84			
10	34.2642	34.3903	-0.13		4.26			
11	33.3488	31.7413	1.61		0.11			
12	35.3331	32.2472	3.09		1.32			
13	32.2020	46.8122	-14.61		273.82			
14	40.9350	32.6039	8.33		40.88			
15	36.5844	31.9067	4.68		7.51			
16	35.5414	28.9189	6.62		21.95			
17	35.9400	50.8777	-14.94		284.77			
18	33.0889	30.1977	2.89		0.91			
19	30.4019	30.2494	0.15		3.19			
20	29.9999	29.8441	0.16		3.17			
21	27.5047	27.8516	-0.35		5.22			
22	27.0516	26.1631	0.89		1.10			
23	25.0900	26.7465	-1.66		12.92			
24	25.1601	25.2477	-0.09		4.10			
25	25.3656	23.9445	1.42		0.27			
26	45.0619	24.1515	20.91		359.98			
27	24.7963	25.2452	-0.45		5.69			
28	26.7553	27.0018	-0.25		4.77			
29	26.9379	25.3042	1.63		0.09			
30	45.0003	25.1467	19.85		320.99			
31	23.4564	25.1362	-1.68		13.08			
32	24.6982	24.7038	-0.01		3.77			
33	28.1273	24.2634	3.86		3.71			
34	25.6764	23.7588	1.92		0.00			
35	23.0601	26.8052	-3.75		32.29			

+ ≡ PBnF vs PCnF ▾ PBnF vs PCF ▾ PBnF vs SBnF ▾ PBnF vs SBF

	A	B	C	D	E	F	G	H
1	PBnF	SBF	A-B	Mean of C	(C-D2)^2	Summation of E	Sample Size	Degrees of freedom
2	982.1099	1471.3292	-489.22	-159.53	108693.50	8935170.68	101	100
3	64.1801	261.4497	-197.27		1424.10			
4	43.8760	292.8164	-248.94		7993.80		Variance	Standard Deviation
5	36.8364	200.3543	-163.52		15.88		89351.71	298.92
6	33.7618	149.2326	-115.47		1941.42			
7	35.2786	139.1385	-103.86		3099.42		Paired Samples T-Test Statistic	
8	33.2120	144.3927	-111.18		2337.88		-5.364	
9	30.5209	177.1918	-146.67		165.42			
10	34.2642	214.5716	-180.31		431.60			
11	33.3488	171.4810	-138.13		457.97			
12	35.3331	164.2682	-128.94		936.19			
13	32.2020	151.2549	-119.05		1638.58			
14	40.9350	152.4067	-111.47		2309.82			
15	36.5844	146.1948	-109.61		2492.20			
16	35.5414	145.0557	-109.51		2501.80			
17	35.9400	144.6534	-108.71		2582.56			
18	33.0889	144.8081	-111.72		2286.10			
19	30.4019	144.6008	-114.20		2055.12			
20	29.9999	144.3070	-114.31		2045.32			
21	27.5047	127.5532	-100.05		3538.33			
22	27.0516	134.3223	-107.27		2731.28			
23	25.0900	130.7740	-105.68		2899.64			
24	25.1601	126.3764	-101.22		3400.76			
25	25.3656	129.5270	-104.16		3065.94			
26	45.0619	129.6461	-84.58		5617.22			
27	24.7963	127.6262	-102.83		3215.17			
28	26.7553	126.6975	-99.94		3550.98			
29	26.9379	128.6502	-101.71		3343.16			
30	45.0003	126.5464	-81.55		6081.85			
31	23.4564	129.5619	-106.11		2854.43			
32	24.6982	127.0222	-102.32		3272.79			
33	28.1273	129.7463	-101.62		3353.95			
34	25.6764	129.6739	-104.00		3084.12			
35	23.0601	128.4639	-105.40		2929.90			

BnF vs SBnF | PBnF vs SBF | PBnF vs SCnF | PBnF vs SCF

	A	B	C	D	E	F	G	H
1	PBnF	SCnF	A-B	Mean of C	(C-D2)^2	Summation of E	Sample Size	Degrees of freedom
2	982.1099	915.7049	66.41	1.64	4194.84	30068.17	1438	1437
3	64.1801	57.5190	6.66		25.24			
4	43.8760	30.6590	13.22		134.09		Variance	Standard Deviation
5	36.8364	26.5300	10.31		75.15		20.92	4.57
6	33.7618	24.6470	9.11		55.91			
7	35.2786	21.8510	13.43		139.01		Paired Samples T-Test Statistic	
8	33.2120	22.2990	10.91		86.04		13.574	
9	30.5209	22.0930	8.43		46.11			
10	34.2642	21.8360	12.43		116.44			
11	33.3488	15.2860	18.06		269.79			
12	35.3331	20.7000	14.63		168.89			
13	32.2020	20.4130	11.79		103.05			
14	40.9350	21.2910	19.64		324.24			
15	36.5844	21.4860	15.10		181.20			
16	35.5414	26.7000	8.84		51.90			
17	35.9400	23.1930	12.75		123.42			
18	33.0889	24.2370	8.85		52.05			
19	30.4019	23.0170	7.38		33.03			
20	29.9999	22.7090	7.29		31.96			
21	27.5047	23.3380	4.17		6.40			
22	27.0516	24.5840	2.47		0.69			
23	25.0900	24.6740	0.42		1.49			
24	25.1601	24.8950	0.27		1.88			
25	25.3656	25.6790	-0.31		3.81			
26	45.0619	27.0880	17.97		266.88			
27	24.7963	25.1430	-0.35		3.94			
28	26.7553	25.9020	0.85		0.61			
29	26.9379	26.0610	0.88		0.58			
30	45.0003	25.2150	19.79		329.34			
31	23.4564	25.6930	-2.24		15.01			
32	24.6982	25.9170	-1.22		8.16			
33	28.1273	24.8770	3.25		2.60			
34	25.6764	25.4350	0.24		1.95			
35	23.0601	24.8360	-1.78		11.65			

BnF vs SBnF | PBnF vs SBF | PBnF vs SCnF | PBnF vs SCF

	A	B	C	D	E	F	G	H
1	PBnF	SCF	A-B	Mean of C	(C-D2)^2	Summation of E	Sample Size	Degrees of freedom
2	982.1099	958.4984	23.61	-124.72	22002.49	36340.53	89	88
3	64.1801	209.3941	-145.21		419.97			
4	43.8760	154.3235	-110.45		203.73		Variance	Standard Deviation
5	36.8364	145.6393	-108.80		253.38		412.96	20.32
6	33.7618	144.5229	-110.76		194.87			
7	35.2786	144.2071	-108.93		249.40		Paired Samples T-Test Statistic	
8	33.2120	142.1286	-108.92		249.77		-57.900	
9	30.5209	144.1865	-113.67		122.22			
10	34.2642	147.8249	-113.56		124.55			
11	33.3488	148.1317	-114.78		98.76			
12	35.3331	179.1565	-143.82		364.91			
13	32.2020	153.5588	-121.36		11.32			
14	40.9350	155.0596	-114.12		112.28			
15	36.5844	158.2819	-121.70		9.14			
16	35.5414	152.8832	-117.34		54.45			
17	35.9400	150.5523	-114.61		102.18			
18	33.0889	144.9482	-111.86		165.42			
19	30.4019	148.7701	-118.37		40.36			
20	29.9999	146.5732	-116.57		66.38			
21	27.5047	148.0617	-120.56		17.34			
22	27.0516	149.5660	-122.51		4.87			
23	25.0900	143.8493	-118.76		35.54			
24	25.1601	149.0611	-123.90		0.67			
25	25.3656	146.9157	-121.55		10.05			
26	45.0619	145.2640	-100.20		601.17			
27	24.7963	145.4627	-120.67		16.44			
28	26.7553	142.0352	-115.28		89.13			
29	26.9379	145.5313	-118.59		37.55			
30	45.0003	145.9118	-100.91		566.89			
31	23.4564	145.3039	-121.85		8.26			
32	24.6982	142.1786	-117.48		52.42			
33	28.1273	143.1637	-115.04		93.79			
34	25.6764	148.0942	-122.42		5.30			
35	23.0601	142.2343	-119.17		30.77			

BnF vs SBnF | PBnF vs SBF | PBnF vs SCnF | PBnF vs SCF

Appendix R

Coefficient of Variation Calculation

	A	B	C	D	E	F	G
1	Frame Time in Seconds	Frame Time in Milliseconds	Mean of B	(Frame Time in Milliseconds-Mean of B)^2	Summation of C	Sample Size	Sample Size-1
2	1.4713292	1471.3292	196.83	1624349.47	10459263.21	101	100
3	0.2614497	261.4497		4175.77			
4	0.2928164	292.8164	Mean of A	9213.48		Variance	Standard Deviation
5	0.2003543	200.3543	0.20	12.42		104592.63	323.41
6	0.1492326	149.2326		2265.47			
7	0.1391385	139.1385		3328.25		Coefficient of Variation	Estimated FPS
8	0.1443927	144.3927		2749.62		1.64	5.08
9	0.1771918	177.1918		385.64			
10	0.2145716	214.5716		314.78			
11	0.1714810	171.4810		642.55			
12	0.1642682	164.2682		1060.24			
13	0.1512549	151.2549		2077.04			
14	0.1524067	152.4067		1973.39			
15	0.1461948	146.1948		2563.87			
16	0.1450557	145.0557		2680.53			
17	0.1446534	144.6534		2722.35			
18	0.1448081	144.8081		2706.23			
19	0.1446008	144.6008		2727.84			
20	0.1443070	144.3070		2758.61			
21	0.1275532	127.5532		4799.21			
22	0.1343223	134.3223		3907.15			
23	0.1307740	130.7740		4363.33			
24	0.1263764	126.3764		4963.64			
25	0.1295270	129.5270		4529.63			
26	0.1296461	129.6461		4513.61			
27	0.1276262	127.6262		4789.10			
28	0.1266975	126.6975		4918.50			
29	0.1286502	128.6502		4648.42			
30	0.1265464	126.5464		4939.71			
31	0.1295619	129.5619		4524.93			
32	0.1270222	127.0222		4873.06			
33	0.1297463	129.7463		4500.16			
34	0.1296739	129.6739		4509.88			

SBF SBnF SCF SCnF PBF PBnF

	A	B	C	D	E	F	G
1	Frame Time in Seconds	Frame Time in Milliseconds	Mean of B	(Frame Time in Milliseconds-Mean of B)^2	Summation of C	Sample Size	Sample Size-1
2	1.0664388	1066.4388	41.20	1051115.03	1121330.18	2000	1999
3	0.0574793	57.4793		265.02			
4	0.0439185	43.9185	Mean of A	7.39		Variance	Standard Deviation
5	0.0389429	38.9429	0.04	5.09		560.95	23.68
6	0.0350265	35.0265		38.11			
7	0.0288718	28.8718		151.98		Coefficient of Variation	Estimated FPS
8	0.0347591	34.7591		41.48		0.57	24.27
9	0.0311985	31.1985		100.03			
10	0.0343903	34.3903		46.37			
11	0.0317413	31.7413		89.46			
12	0.0322472	32.2472		80.15			
13	0.0468122	46.8122		31.50			
14	0.0326039	32.6039		73.89			
15	0.0319067	31.9067		86.36			
16	0.0289189	28.9189		150.82			
17	0.0508777	50.8777		93.66			
18	0.0301977	30.1977		121.05			
19	0.0302494	30.2494		119.91			
20	0.0298441	29.8441		128.95			
21	0.0278516	27.8516		178.17			
22	0.0261631	26.1631		226.10			
23	0.0267465	26.7465		208.90			
24	0.0252477	25.2477		254.47			
25	0.0239445	23.9445		297.74			
26	0.0241515	24.1515		290.64			
27	0.0252452	25.2452		254.55			
28	0.0270018	27.0018		201.58			
29	0.0253042	25.3042		252.67			
30	0.0251467	25.1467		257.70			
31	0.0251362	25.1362		258.04			
32	0.0247038	24.7038		272.12			
33	0.0242634	24.2634		286.84			
34	0.0237588	23.7588		304.19			

\+ ≡ SBF SBnF SCF SCnF PBF PBnF

	A	B	C	D	E	F	G
1	Frame Time in Seconds	Frame Time in Milliseconds	Mean of B	(Frame Time in Milliseconds-Mean of B)^2	Summation of C	Sample Size	Sample Size-1
2	0.9584984	958.4984	163.55	631942.43	652232.55	89	88
3	0.2093941	209.3941		2101.65			
4	0.1543235	154.3235	Mean of A	85.13		Variance	Standard Deviation
5	0.1456393	145.6393	0.16	320.81		7411.73	86.09
6	0.1445229	144.5229		362.04			
7	0.1442071	144.2071		374.16		Coefficient of Variation	Estimated FPS
8	0.1421286	142.1286		458.89		0.53	6.11
9	0.1441865	144.1865		374.96			
10	0.1478249	147.8249		247.29			
11	0.1481317	148.1317		237.73			
12	0.1791565	179.1565		243.55			
13	0.1535588	153.5588		99.83			
14	0.1550596	155.0596		72.09			
15	0.1582819	158.2819		27.76			
16	0.1528832	152.8832		113.79			
17	0.1505523	150.5523		168.95			
18	0.1449482	144.9482		346.04			
19	0.1487701	148.7701		218.46			
20	0.1465732	146.5732		288.22			
21	0.1480617	148.0617		239.90			
22	0.1495660	149.5660		195.56			
23	0.1438493	143.8493		388.13			
24	0.1490611	149.0611		209.94			
25	0.1469157	146.9157		276.71			
26	0.1452640	145.2640		334.39			
27	0.1454627	145.4627		327.16			
28	0.1420352	142.0352		462.90			
29	0.1455313	145.5313		324.69			
30	0.1459118	145.9118		311.12			
31	0.1453039	145.3039		332.93			
32	0.1421786	142.1786		456.75			
33	0.1431637	143.1637		415.61			
34	0.1480942	148.0942		238.89			

SBF SBnF SCF SCnF PBF PBnF

	A	B	C	D	E	F	G
1	Frame Time in Seconds	Frame Time in Milliseconds	Mean of B	(Frame Time in Milliseconds-Mean of B)^2	Summation of C	Sample Size	Sample Size-1
2	0.9157049	915.7049	38.73	769077.46	824513.63	1438	1437
3	0.0575190	57.5190		352.87			
4	0.0306590	30.6590	Mean of A	65.21		Variance	Standard Deviation
5	0.0265300	26.5300	0.04	148.94		573.77	23.95
6	0.0246470	24.6470		198.45			
7	0.0218510	21.8510		285.05		Coefficient of Variation	Estimated FPS
8	0.0222990	22.2990		270.12		0.62	25.82
9	0.0220930	22.0930		276.93			
10	0.0218360	21.8360		285.55			
11	0.0152860	15.2860		549.82			
12	0.0207000	20.7000		325.24			
13	0.0204130	20.4130		335.67			
14	0.0212910	21.2910		304.27			
15	0.0214860	21.4860		297.50			
16	0.0267000	26.7000		144.82			
17	0.0231930	23.1930		241.53			
18	0.0242370	24.2370		210.17			
19	0.0230170	23.0170		247.03			
20	0.0227090	22.7090		256.81			
21	0.0233380	23.3380		237.05			
22	0.0245840	24.5840		200.23			
23	0.0246740	24.6740		197.69			
24	0.0248950	24.8950		191.53			
25	0.0256790	25.6790		170.44			
26	0.0270880	27.0880		135.64			
27	0.0251430	25.1430		184.72			
28	0.0259020	25.9020		164.67			
29	0.0260610	26.0610		160.61			
30	0.0252150	25.2150		182.77			
31	0.0256930	25.6930		170.08			
32	0.0259170	25.9170		164.28			
33	0.0248770	24.8770		192.02			
34	0.0254350	25.4350		176.87			

\+ ≡ SBF SBnF SCF SCnF PBF PBnF

	A	B	C	D	E	F	G
1	Frame Time in Seconds	Frame Time in Milliseconds	Mean of B	(Frame Time in Milliseconds-Mean of B)^2	Summation of C	Sample Size	Sample Size-1
2	0.7984609	798.4609	195.51	363552.23	8984227.84	100	99
3	0.2082149	208.2149		161.47			
4	0.1470730	147.0730	Mean of A	2345.95		Variance	Standard Deviation
5	0.1435052	143.5052	0.20	2704.29		90749.78	301.25
6	0.1447136	144.7136		2580.07			
7	0.1379428	137.9428		3313.75		Coefficient of Variation	Estimated FPS
8	0.1373223	137.3223		3385.57		1.54	5.11
9	0.1444081	144.4081		2611.20			
10	0.1452465	145.2465		2526.22			
11	0.1449925	144.9925		2551.81			
12	0.1690618	169.0618		699.40			
13	0.1490183	149.0183		2161.29			
14	0.1441121	144.1121		2641.54			
15	0.1503918	150.3918		2035.47			
16	0.1497879	149.7879		2090.32			
17	0.1395357	139.5357		3132.90			
18	0.1398955	139.8955		3092.75			
19	0.1450339	145.0339		2547.63			
20	0.1361697	136.1697		3521.03			
21	0.1325703	132.5703		3961.15			
22	0.1364048	136.4048		3493.18			
23	0.1382596	138.2596		3277.38			
24	0.1330102	133.0102		3905.97			
25	0.1353307	135.3307		3621.30			
26	0.1353267	135.3267		3621.79			
27	0.1347621	134.7621		3690.06			
28	0.1338833	133.8833		3797.60			
29	0.1364260	136.4260		3490.68			
30	0.1371636	137.1636		3404.07			
31	0.1353004	135.3004		3624.95			
32	0.1333840	133.3840		3859.39			
33	0.1394581	139.4581		3141.59			
34	0.1345156	134.5156		3720.07			

SBF SBnF SCF SCnF PBF PBnF

	A	B	C	D	E	F	G
1	Frame Time in Seconds	Frame Time in Milliseconds	Mean of B	(Frame Time in Milliseconds-Mean of B)^2	Summation of C	Sample Size	Sample Size-1
2	0.9821099	982.1099	43.14	881669.90	1012913.01	2000	1999
3	0.0641801	64.1801		442.81			
4	0.0438760	43.8760	Mean of A	0.55		Variance	Standard Deviation
5	0.0368364	36.8364	0.04	39.70		506.71	22.51
6	0.0337618	33.7618		87.90			
7	0.0352786	35.2786		61.76		Coefficient of Variation	Estimated FPS
8	0.0332120	33.2120		98.51		0.52	23.18
9	0.0305209	30.5209		159.17			
10	0.0342642	34.2642		78.73			
11	0.0333488	33.3488		95.81			
12	0.0353331	35.3331		60.90			
13	0.0322020	32.2020		119.58			
14	0.0409350	40.9350		4.85			
15	0.0365844	36.5844		42.94			
16	0.0355414	35.5414		57.69			
17	0.0359400	35.9400		51.80			
18	0.0330889	33.0889		100.97			
19	0.0304019	30.4019		162.19			
20	0.0299999	29.9999		172.59			
21	0.0275047	27.5047		244.37			
22	0.0270516	27.0516		258.74			
23	0.0250900	25.0900		325.70			
24	0.0251601	25.1601		323.17			
25	0.0253656	25.3656		315.83			
26	0.0450619	45.0619		3.70			
27	0.0247963	24.7963		336.39			
28	0.0267553	26.7553		268.36			
29	0.0269379	26.9379		262.41			
30	0.0450003	45.0003		3.47			
31	0.0234564	23.4564		387.33			
32	0.0246982	24.6982		339.99			
33	0.0281273	28.1273		225.29			
34	0.0256764	25.6764		304.88			

+ ≡ PBF ▾ PBnF ▾ PCF ▾ PCnF ▾ CV SUMMARY ▾

	A	B	C	D	E	F	G
1	Frame Time in Seconds	Frame Time in Milliseconds	Mean of B	(Frame Time in Milliseconds-Mean of B)^2	Summation of C	Sample Size	Sample Size-1
2	1.2944969	1294.4969	171.86	1260309.07	1305280.85	85	84
3	0.2935834	293.5834		14816.09			
4	0.1909564	190.9564	Mean of A	364.60		Variance	Standard Deviation
5	0.1759012	175.9012	0.17	16.31		15539.06	124.66
6	0.1894214	189.4214		308.33			
7	0.2092829	209.2829		1400.32		Coefficient of Variation	Estimated FPS
8	0.1693441	169.3441		6.34		0.73	5.82
9	0.1569744	156.9744		221.64			
10	0.1536519	153.6519		331.61			
11	0.1706858	170.6858		1.38			
12	0.1697370	169.7370		4.52			
13	0.1614466	161.4466		108.48			
14	0.1531939	153.1939		348.50			
15	0.1533432	153.3432		342.95			
16	0.1522946	152.2946		382.88			
17	0.1538210	153.8210		325.48			
18	0.1554328	155.4328		269.92			
19	0.1517030	151.7030		406.39			
20	0.1458734	145.8734		675.41			
21	0.1469506	146.9506		620.58			
22	0.1484996	148.4996		545.80			
23	0.1492745	149.2745		510.20			
24	0.1496066	149.6066		495.30			
25	0.1524326	152.4326		377.50			
26	0.1486007	148.6007		541.09			
27	0.1518908	151.8908		398.85			
28	0.1441766	144.1766		766.48			
29	0.1452068	145.2068		710.50			
30	0.1444261	144.4261		752.73			
31	0.1455186	145.5186		693.98			
32	0.1482907	148.2907		555.61			
33	0.1472188	147.2188		607.29			
34	0.1470077	147.0077		617.74			

+ ≡ PBF ▾ PBnF ▾ PCF ▾ PCnF ▾ CV SUMMARY ▾

	A	B	C	D	E	F	G
1	Frame Time in Seconds	Frame Time in Milliseconds	Mean of B	(Frame Time in Milliseconds-Mean of B)^2	Summation of C	Sample Size	Sample Size-1
2	0.9599911	959.9911	39.05	848134.40	900267.00	1168	1167
3	0.0612780	61.2780		494.13			
4	0.0304469	30.4469	Mean of A	74.00		Variance	Standard Deviation
5	0.0278862	27.8862	0.04	124.61		771.44	27.77
6	0.0249290	24.9290		199.37			
7	0.0228760	22.8760		261.57		Coefficient of Variation	Estimated FPS
8	0.0213855	21.3855		312.00		0.71	25.61
9	0.0222339	22.2339		282.75			
10	0.0220653	22.0653		288.45			
11	0.0212666	21.2666		316.21			
12	0.0210878	21.0878		322.60			
13	0.0224484	22.4484		275.58			
14	0.0215233	21.5233		307.15			
15	0.0314876	31.4876		57.17			
16	0.0267409	26.7409		151.49			
17	0.0344861	34.4861		20.82			
18	0.0285766	28.5766		109.67			
19	0.0263434	26.3434		161.43			
20	0.0269043	26.9043		147.49			
21	0.0248527	24.8527		201.53			
22	0.0267115	26.7115		152.21			
23	0.0300641	30.0641		80.73			
24	0.0286328	28.6328		108.50			
25	0.0260523	26.0523		168.91			
26	0.0265060	26.5060		157.33			
27	0.0284019	28.4019		113.36			
28	0.0278855	27.8855		124.62			
29	0.0285063	28.5063		111.15			
30	0.0241601	24.1601		221.68			
31	0.0236035	23.6035		238.56			
32	0.0244255	24.4255		213.85			
33	0.0271275	27.1275		142.12			
34	0.0258458	25.8458		174.32			

PBF PBnF PCF PCnF CV SUMMARY