Video game development on Roblox platform using Lua programming language

Sumaia A. Eltomi [1], Rasheed S. Mottaleb [2]

[1,2] Dept. of Electronic, College of Industrial Tech, Misurata, Libya

Somaia_ali@cit.edu.ly [1]

الملخص

هذه الورقة تهدف إلى شرح عملية تطوير ألعاب الفيديو للمبتدئين، بما في ذلك تعلم البرمجة والتصميم، وإتمام عملية التطوير بشكل صحيح. تناقش الورقة الافتراضات الخاطئة في عملية التطوير وكيفية تجنبها، بالإضافة إلى المكونات الأساسية لأي لعبة. كما تغطي اختيار المحرك ولغة البرمجة المناسبين استنادًا إلى فئة اللعبة المستهدفة وعملية نشر اللعبة. تتناول الورقة أيضًا لغة لوا وعلاقتها بتطوير الألعاب، مع التركيز على استخدامها في منصة روبلوكس. الجانب العملي للورقة هو عملية نشر اللعبة على هذه المنصة. تستنتج الورقة أن الأفراد الذين يمتلكون خبرة في التصميم والبرمجة يمكنهم تطوير ألعاب فيديو عالية الجودة.

Abstract

This paper aims to explain the process of developing video games for beginners, including learning programming and design, and completing development correctly. It discusses misconceptions in the development process and how to avoid them, as well as the basic components of a game. It also covers selecting the appropriate engine and programming language based on the target game category and the game publishing process. The paper reviews the Lua language and its relationship to game development, with a focus on its use in the Roblox platform. The practical aspect of the paper is the process of publishing a game on the platform. The practical aspect of the paper is the process of publishing a game on the platform. The paper concludes that anyone can develop video games, but higher quality and greater flexibility in the game are achieved as developers gain experience in design and programming.

Keywords: Video game, video game development, Lua language, Roblox studio.

I. INTRODUCTION

Video games are interactive experiences enjoyed by players in virtual worlds filled with exciting stories and challenges. They include multiple genres such as action, adventure, sports, and shooting games. Advanced graphics, sound, and artificial intelligence technologies are used to create a realistic experience that captivates players and transports them into the game world. Players can interact with characters and environments and make decisions that affect the course of the game. Some games offer educational experiences that contribute to gaining knowledge and developing skills. Furthermore, companies use gaming as a marketing tool. Video game development is the process of creating and designing video games through software development. Video game development is undertaken by either individuals or large companies. Funding for well-known games is usually provided by publishers, and development times can vary. Indie games, on the other hand, can be produced by individuals or small developers and may require less time to develop. The indie game industry has experienced significant growth in recent years, thanks to the rise of new network distribution systems and the mobile game market. [1].

Video game development involves several stages. It begins with conceptualization and design, where developers brainstorm ideas, create game mechanics, and design characters and environments. This phase sets the foundation for the entire development process. Once the design is finalized, production begins. This involves creating game assets, such as 3D models, textures, animations, and audio elements [2]. Programmers write the code that brings the game to life, implementing mechanics, interactions, and artificial intelligence systems. This requires expertise in programming languages and game engines. [3].

Testing and quality assurance are crucial in ensuring the game is free from bugs and glitches. Testers play through the game meticulously, identifying any issues and providing feedback for improvement [4]. The visual and audio aspects of the game are vital for creating an immersive experience. Graphic designers create stunning visuals, while sound designers craft the audio elements. These elements contribute to the emotional impact of the game. [5].

After completing the development phase, the game undergoes a final round of testing and polishing. Then, marketing strategies are devised to promote the game. Video game distribution has evolved significantly in recent years. While traditional methods include physical copies sold in retail stores, digital distribution has become popular. This has opened up opportunities for independent developers to showcase their creations and reach a wider audience.

The paper is structured as follows: Section II provides an overview of related works in game development. Section III discusses the pre-development process, Section IV addresses game programming languages, Section V introduces Roblox Studio, Section VI summarizes video game development methodologies, and the final section summarizes the research results

II. RELATED WORK

Numerous academic studies have been conducted on game development in recent years. However, for our purposes, we will focus on the most relevant studies, which are discussed in this section.

This paper explores the use of video game engines, specifically the Roblox Studio engine, to create a virtual simulation of sculptural heritage in a classroom setting. A pilot experiment was conducted with 53 high school students who designed a virtual environment featuring 3D models of the sculptural heritage of a specific city. Students reported an increased awareness of the city's sculptural heritage and felt confident in their ability to create interactive worlds using Roblox [6].

The research paper explores moderating user-generated virtual worlds (UGVWs) on platforms like Roblox. It addresses issues with harmful user-generated content and the complexity of moderating game design. The study examines the impact of harmful design on Roblox users, identifying potential risks such as inappropriate content and problematic incentive mechanisms. It also discusses the opportunities and challenges in mitigating these issues [7].

In their study [8], the authors explore the potential benefits and opportunities of incorporating Roblox into university courses as an educational tool. They examine its impact on higher education and address the challenges faced by students and teachers in this context. The literature review highlights the limited exploration of Roblox's potential in higher education thus far. The aim of this research is to investigate students' views on the integration of Roblox as an educational tool using the TAM model.

A previous study [9] explored the use of digital games in a learning context, with a particular focus on game design strategy. The study highlighted the potential of Roblox technology to enhance the learning process. However, the article notes that the current evidence is mainly descriptive, and there is a lack of comparative studies and evidence-based frameworks. The article concludes with an analysis and review of current evidence, without presenting direct findings from original research. Based on the analysis, the article draws some observations and suggests future directions.

The paper analyses the requirements of metaverse platforms and evaluates existing platforms to determine their ability to meet those requirements. It discusses the growing interest in virtual worlds, also known as metaverse, in an objective manner. The text highlights the limitations of these platforms and identifies the need for further research and development in areas such as pervasive thinking, security and privacy, and integration of emerging technologies like blockchain and artificial intelligence. To build a robust and secure metaverse in the future, it is recommended to pay more attention to these aspects. [10].

This paper examines the use of ChatGPT in creating educational and interactive environments within the Roblox metaverse. The study shows how ChatGPT streamlines the development process by optimizing scripts, creating well-structured dictionaries, and simplifying the implementation of effects and interactions. College-level student testing demonstrated that integrating ChatGPT improved student engagement and understanding, indicating its potential for developing metaverse-based educational content [11].

In a study by researchers [12], the use of the game Roblox as an educational tool in Indonesia was discussed as a means of improving second-grade students' digital multimedia literacy abilities. The results showed the importance of using Roblox as a positive educational method to improve students' digital multimedia literacy abilities. The research concludes that lesson design for middle school English teaching should incorporate the use of games as a teaching tool.

III. PRE-DEVELOPMENT EXPLORATION

New game developers often make the mistake of relying solely on online videos and attempting to create games sequentially without a sufficient background in game development basics. This approach often results in the development of undistinguished games that fail to meet player needs and expectations. New developers may become distracted and frustrated when they do not achieve satisfactory results quickly. To avoid this problem, they should research common mistakes in game development and how to avoid them, as well as follow the essential steps to learn the basics. [13].

A. Hidden Challenges

Aspiring game developers encounter challenges beyond technical and creative skills. It is crucial to address common mistakes when designing their first game.

- Cloning a game without any innovation significantly diminishes its value.
- Randomly incorporating artistic elements affects game's attractiveness Figure1.
- Game size and complexity: Novice developers may struggle to create large and complex games, which can negatively impact game quality.
- Imbalanced speed negatively impacts the player's experience and ability to challenge.
- Matching the difficulty of a game with the target group of players is crucial for a positive game experience.
- Early criticism before completion can negatively impact the game's quality and cause frustration.
- It is important to focus on learning the basics before adding details and stages.

- Bias towards the game can lead to false sentiment and affect its quality and acceptance.
- To meet the needs of players, it is important to focus on a specific category rather than trying to satisfy everyone.
- Instant success is not guaranteed, so it is important to come up with a new idea that aligns with the interests of players, as demonstrated in the games in Figure2.

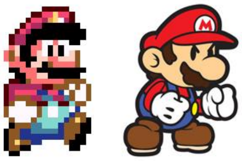


Figure 1: Random merging of graphics

Figure 2: Famous games

B. The resources

Game resources are essential for creating a unique and engaging user experience. By customising resources independently, developers can create more distinctive games.

- Game development ideas can come from various sources, and the success of a game depends on its design concept. To generate ideas for game development, one can analyse existing games, draw inspiration from different media such as books and movies, or explore a specific concept through game design. For instance, an educational game about the environment can be created. Players' interests can be met while providing a fun experience. Experienced developers can be consulted for their knowledge and guidance in this field. [14].
- Developers often have limited resources when it comes to models and graphics, especially in the early stages. However, there are many websites that offer free game assets, as shown in Table1. These resources can provide a variety of visual elements that can be used in game design.

Table 1: Website and store free artwork

CraftPix	Kenney	Poly Haven	DOTOWN	Super Game Assets
Game Art 2D	Quaternius	Textures	itch.io	Unreal Engine Marketplace
Gamefresco	OpenGameArt	GameDev Market	Unity Assets Store	Reddit Free Game Asset Forum

- Beginner programmers can find reclaimed code online through code-sharing websites and online programming communities. These resources offer ready-made code for specific functions and programming instructions that can be modified to meet specific needs. Table 2 displays ready-made coding sites.
- Code sharing enables programmers to share their code with others for debugging, education, or collaboration. Numerous code-sharing websites exist on the web, where programmers can interact and share experiences. Table 3 lists some of the most important of these websites.
- Sound sources: Independent developers have access to a variety of free sound effects and music for their games. They can find these resources on comprehensive audio databases or genre-specific blogs, such as those listed in Table 4. Utilizing these resources can help developers add creative and suitable sound and music to their games.

Table 2: Code snippets

GenerateWP	CodePen
StackOverflow	chatGPT

Table 3: Code Review

Github Gist	JS Bin
JSFiddle	Codeshare

Table 4: Free Sound Effects

Freesound	Soundjay
Game Burp	99 Sounds

C. The game engine

Game development software includes a store app, a publishing platform, and tools essential for development. Choosing the right engine is crucial for novice developers, considering game type, device requirements, budget, and other factors. The selected engine significantly impacts team capabilities and project progression, varying with game diversity. These six factors are crucial for engine selection. [15].

- The choice of game engine should be based on team's skills and the ready time.
- Team size: Some engines can be used alone, while others require collaboration.
- The cost of game engines varies by type and features. Free engines are accessible, but advanced features may necessitate a subscription or purchase, while paid engines usually require a subscription fee or outright purchase.
- Programming skills: The ability to write basic and object-oriented coding, graphics, sound, AI development, visual programming and databases.
- Game Type: Select an engine that supports your desired game type and provides tools and features to facilitate the creation of various game types.
- Final Product: Game development assesses project scope and target platform compatibility. Engine choice varies; smaller games like mobile or web games can use different engines, while larger 3D games require more powerful options.

There are many popular game engines widely used in the industry. Table 5 compares the characteristics of Roblox, Unity, Unreal, and Godot and evaluates whether they are advantageous or disadvantageous.

Table 5: Characteristics of game engines

	Roblox [16]	Unity [17]	Unreal [18]	Godot [19]	
Platforms	Multi	Multi	Multi	Limited	
	Huge	Big	Big	Limited	Audience
Correction	Easy	Easy	Learn C++	Difficult	
	One	Team	Team	One	Developers
Graphics	Good	Medium	High	Good	
	Free	Almost free	Not free	Free	license
RAM usage	Acceptable	High	Very high	Acceptable	
	Lua	C#	C++	GDScript	language
Lead time	Short	Moderate	long	Moderate	
	Enable	Enable	Not enable	Enable	Live-editing
Popular games	Most	Moderately	Moderately	Less	
	More users	Independent	companies	Less used	Users
Exp. Required	Less	High	High	Moderate	
	Easy	Moderate	Moderate	Moderate	Usability
Community	Strong	Strong	Average	Weak	
	Dependent	independent	independent	independent	Independence
Profit sharing	Weak	NA	NA	NA	

This table provides an overview of the advantages and disadvantages of different engines. Ratings may vary depending on the needs and requirements of the developer and their project. It is recommended that developers try and compare different engines to choose the most appropriate one for their project.

D. Publishing

Publishers play a crucial role in the gaming industry by publishing, marketing, and investing in games. They offer a range of services, including advising and managing game development, financing, distribution, and marketing across various channels [16].

In the video game industry, publishers provide assistance in game production financing, marketing strategies, and product plan development and support. Publishers use various techniques to enhance the effectiveness of games, such as distribution and advice. They also handle tasks like obtaining licenses, writing guides, and creating visual elements. Without the help of a publishing company, these tasks can be difficult or even impossible for small publishers.

IV. GAME PROGRAMMING LANGUAGES

A game programming language or script is the system that game programmers use to shape game settings and environments. It is the mechanism that makes a video game work in a specific way. Game programming languages vary in complexity and power. Some are simple and easy to learn, designed to produce quick-to-build games, while others are more complex and allow programmers to create highly sophisticated and intricate games.

Programming languages have specific purposes and structures that make them more efficient at certain tasks. Although they may overlap and be compatible with each other, programmers developing games often use a variety of languages. [17].

The choice of programming language for game development depends on various factors, such as the game type, target platform, and complexity [18]. Table 6 displays some of the frequently used programming languages in video game development.

Table 6: Popular game programming languages

	Key Programming Languages in Game Development	
	Compatible with games, small projects, web applications, and image processing.	Lua
C++	Powerful for developing high-quality video games, and efficient memory management.	
	Powerful for video games, relatively easy to learn, popular among novice programmers.	C#
java	The product is versatile and can be upgraded, making it competitive in the modern technology industry.	
	Popular in browser-based game development and web-based games.	JavaScript
Python	It is based on object-oriented programming and facilitates the creation of prototypes.	
	Exclusive to mobile operating systems, such as Android and iOS.	Objective-C

A. Lua programming language

It was inspired by SOL (Simple Object Language), and named after the Portuguese word for 'moon'. The team of professors chose the name 'Lua' for the new language, as it means 'moon'. Figure 3 displays the official logo for the Lua programming language. Lua is a dynamically typed scripting language that is composable. The code in Lua is read line by line, from top to bottom, making it easy to read and write. [19].

Figure 3: Lua Official logo

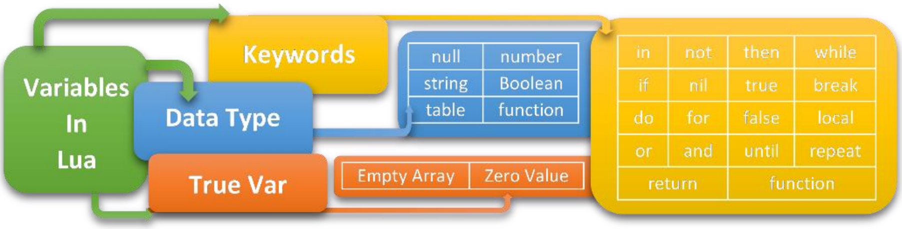
Lua is an exceptional language due to its design as an extension language that can be integrated into any platform or application. This allows developers to fully customize products. Additionally, Lua can handle any C and C++ code, making it easy to create new game systems. Lua programs are small, flexible, and portable, making them an ideal starting point for extending various types of programs. [20].

Lua is a versatile programming language that is commonly used in gaming and web development. It has been used in a variety of applications, including the mobile payment app Venmo and the popular game Angry Birds. Overall, Lua's flexibility and compatibility make it a valuable tool for developers and designers alike. Additionally, Lua is a popular choice for game developers on platforms such as Roblox, where it is used to write code for creating new games. Designers can also create plug-ins for image editing software Adobe Lightroom using Lua. [21].

B. Concepts and Fundamentals

To become proficient in Lua, the programmer must first explore its syntax and become familiar with its basic concepts and terminology. Starting with simple commands, they can gradually progress to more complex instructions, gaining greater control over the language and using it more professionally.

- **Variables in Lua:** A variable is a value that can be altered throughout the program. The value of the variable can be stored and referenced in later sections of the program [22]. Lua requires variables to be declared before use, and variable names are case sensitive. For instance, 'Rody', 'rody', and 'RODY' are distinct variables in Lua. Refer to Figure 4 for additional information on Lua variables.



null	number
string	Boolean
table	function

Empty Array	Zero Value

in	not	then	while
if	nil	true	break
do	for	false	local
or	and	until	repeat
return		function	

Figure 4: Variables in Lua

- **Functions in Lua:** A function is a group of statements that carry out a particular task. Functions help to break down code into clear and understandable units, with each function performing a specific task within the program. [23]. The Lua programming language defines functions in the general form shown in Figure 5.

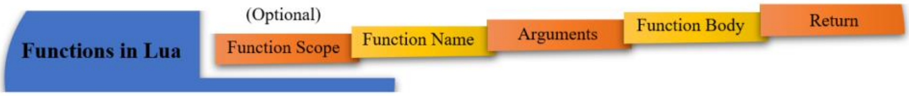


Figure 5: Official logo of the Lua language

- **Lua in game engines:** Lua programming language is used by many game engines. This does not imply that the engine was created in Lua, but rather that it supports Lua as a programming language [24]. Table 7 and Table 8 include 2D and 3D game engines respectively.

Table 7: Exploring 2D Games with Lua-Powered Engines

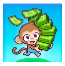
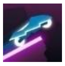
Icon	Ex; game	2D Lua Engines:	Engines
	Monkey Mart	This engine is built using a component-based system for optimal performance.	Defold
	Rider	This cross-platform framework is perfect for developing and publishing games across mobile, TV, and PC.	Corona
	Move or Die	This is an open-source and free Lua framework for 2D games that is compatible with various operating systems.	LÖVE
	Sweet Combo Match	The engine is a free and open-source technology that enables the creation of amazing games across multiple platforms.	Gideros
	cat vs Roomba	This software library is designed for programming video games with a simple interface and no additional tools beyond programming.	Raylib
	Aeroplane Adventure	This tool is designed to create text-based games that combine interactive fiction, narrative, and visual elements.	Instead

Table 8: Exploring 3D Games with Lua-Powered Engines

Icon	Ex; game	3D Lua Engines:	Engines
	Zero-K	Versatile and fully customizable, including the GUI.	Spring RTS
	Racing Fever	This software provides powerful features for creating, playing, and sharing games on a PC without requiring any prior programming knowledge.	GameGuru
	Kumoon	This software offers all the features of contemporary 3D games, such as resource management, scenery, and user interface.	Shiva
	Leadwerks	Provides a smooth learning curve for beginners to become professional game developers with ease.	Leadwerks
	RSM game	The practical part of this paper will be reviewed later.	Roblox Studio
	Tux!	This is a lightweight and free game engine that supports both 2D and 3D graphics.	Urho3D

V. ROBLOX PLATFORM

Roblox is an online gaming and game creation platform with over 40 million user-generated experiences. Users can create and share their own experiences.

Roblox is a social gaming network within the metaverse, allowing users to communicate and interact with others. Developers using Roblox Studio must use the Lua programming language to create 3D games. The software provides templates and components to facilitate game creation. [25].

A. User interface

Roblox Studio has a beginner-friendly interface that simplifies game development and provides customizable templates. The platform enables easy importation of shapes, models, and sounds, as demonstrated in Figure 6.

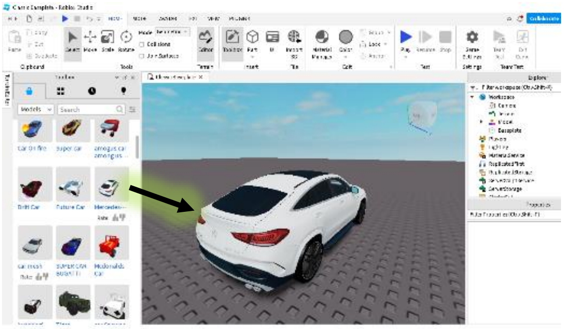


Figure 6: Importing in Roblox Studio

B. Design pattern in Roblox

The game model floor demonstrates Roblox's design approach, which was created by implementing a sequence of operations on shapes. Figure 7 shows the overlapping cylinders prior to the execution of these operations. The island model in Figure 8 was produced after further processing of the components.

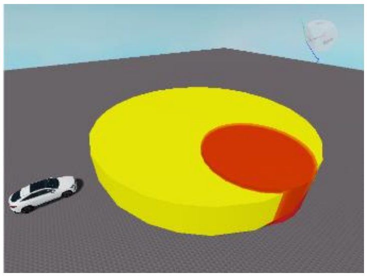


Figure 7: Two overlapping cylinders

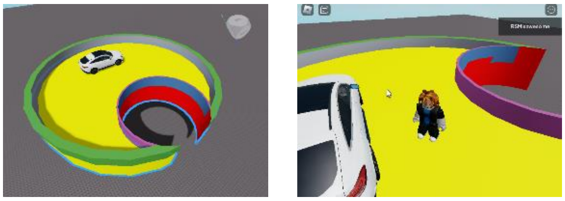


Figure 8: Island prototype

C. Integration with Lua

Game design requires time and effort. However, the game remains incomplete until programming statements define its behavior. Roblox Studio uses Lua.

To link an object, the user clicks "+" and selects "Script," opening a new programming window Figure 9.

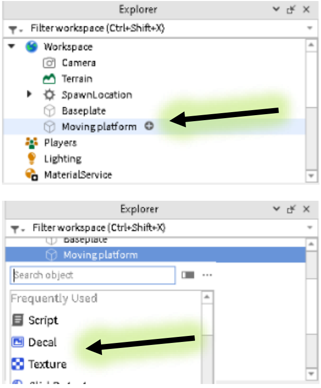


Figure 9: Adding a Lua file

```
    while true do
        wait(1)
        for i= 1, 100 do
            script.Parent.CFrame = script.Parent.CFrame * CFrame.new(0, 0, 0.3)
            wait()
        end
        wait(1)
        for i= 1, 100 do
            script.Parent.CFrame = script.Parent.CFrame * CFrame.new(0, 0, -0.3)
            wait()
        end
end
```

D. Game testing

Game testing is a crucial development stage. It allows evaluating the design and programming, identifying issues, and demonstrating the product.

Roblox developers have two testing methods available, shown in Figure 10.

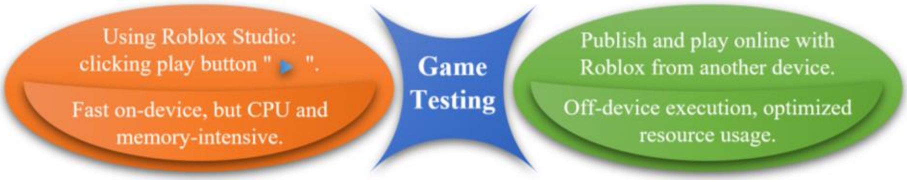

Using Roblox Studio: clicking play button " ▶ ".
Fast on-device, but CPU and memory-intensive.
Game Testing
Publish and play online with Roblox from another device.
Off-device execution, optimized resource usage.

Figure 10: Game testing in Roblox

VI. GAME DEVELOPMENT METHODOLOGY

This section outlines the proposed method for the developed game, featuring two worlds: one for entertainment and the other for adventure and combat. The method includes conceptualizing the game idea, importing necessary models using Roblox, programming game behavior in Lua, designing a unique icon and cover for server login, and concluding with testing and documentation. Figure11 displays the development method.

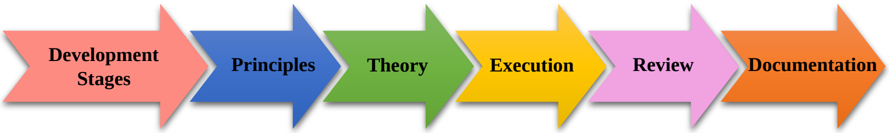


Figure 11: Game Development Stages

A. Game Principles

It involves clarifying the main ideas and features of the game, determining the sequence of events and basic elements. Game planning is a crucial process in game development. The following factors are considered in game planning:

- Define the game's objective and desired player experience.
- Create a map of the game world and label distinct areas and levels.
- Arrange challenges and stages in a consistent sequence.
- Identify characters and design their unique abilities and skills.
- Analyse preliminary layouts to identify potential issues.

The initial layout of the proposed game is shown in Figure 12.

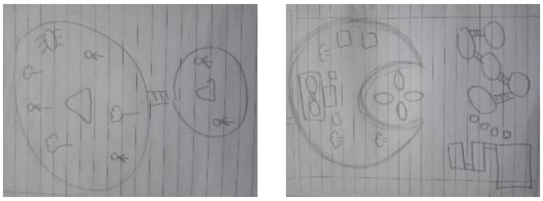


Figure 12: The proposed game layout

B. Game Theory

The design phase begins after planning. Import required items through Roblox, selecting desired items and checking for programming files as needed. Place forms in locations outlined in the planning document.

The spawn point, where players first appear and can return to in case of loss, is set by clicking the publishing board and placing it in the game world, as shown in Figure 13. This supports development, particularly testing, and the spawn location can be changed.

Figure 14 shows creating castle rooms using merging and removing, adding effects to enhance the fantasy appearance. One room was converted to a cloud room, with the clouds serving to provide lighting and a scattered object effect, as in Figure 15.

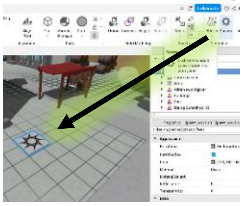


Figure 13: Create a spawn

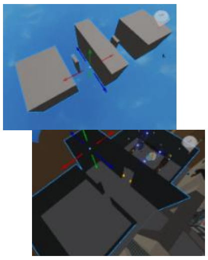


Figure 14: Create rooms and corridors

Figure 15: Effects in clouds room

The room serves as the entrance to the second world. Navigation boards are used for teleportation to the second world. There is a painting in the room and another in the second world. However, a problem has arisen as the character keeps moving back and forth between the two worlds. The touch and sensor feature were removed from the second panel. Both panels were made 100% transparent. The first panel was placed on the door in the clouds room to create the illusion that the door is responsible for the transfer. Figure 16 displays the panels before and after transparency.

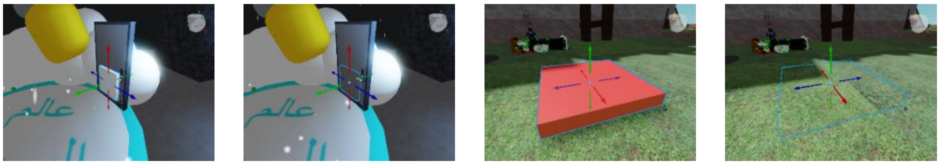


Figure 16: Transition between worlds

- Characters

The second world comprises two islands in the sky, as shown in Figure 17. One can move between them using a group of small clouds:

Weaponry Isle: This is a location for practicing exploration, shooting, and aiming. It is recommended to limit the number of weapons in the game and avoid making them easily accessible to create a challenge. To reach the next monster island, players must jump on moving clouds that travel from point A to point B. Figure 18 displays the programming file for one of these moving clouds, as shown in Figure 19.

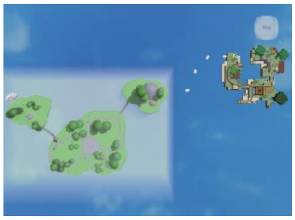
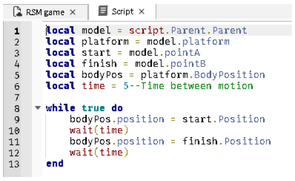


Figure 17: Second world in-game



```
local model = script.Parent.Parent
local platform = model.platform
local start = model.pointA
local finish = model.pointB
local bodyPos = platform.BodyPosition
local time = 5--Time between motion

while true do
    bodyPos.position = start.Position
    wait(time)
    bodyPos.position = finish.Position
    wait(time)
end
```

Figure 18: Animated clouds programming file

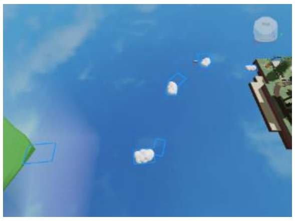


Figure 19: Moving clouds between two islands

Monsters Isle: It contains three types of monsters:

- Cloud Fairy: was imported and modified, and a machine gun was integrated into it.
- Spider: Three copies are included, each with the ability to move and multiple attack options. Adjustments have been made to their size, sound, and life points.
- Samurai: He is the leader in this world and possesses multiple software files. Despite recent changes that have reduced its speed, it remains very fast and difficult to evade. Defeating him requires a team effort. Figures 20, 21, and 22 depict the monsters.

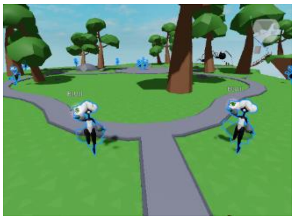


Figure 20: Monster: fairy

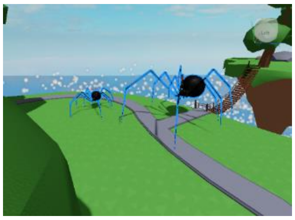


Figure 21: Monster: spiders

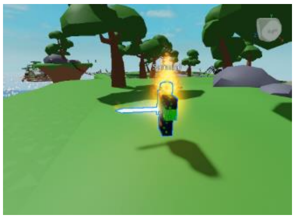


Figure 22: Monster: Boss

A cube has been created beneath the sky islands, as depicted in Figure 23. Its purpose is to end the player's life points and return them to spawn upon falling. The program file below should be added to the cube.

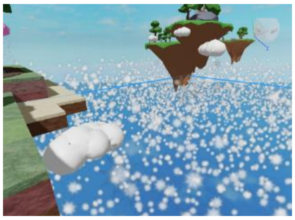


Figure 23: Transparent cube under the sky islands

```
    function onTouched(hit)

            local human = hit.Parent:FindFirstChild("Humanoid")

          --Basically this just checks to see if it is a real player
touching this brick.

            if (human ~= nil) then --If it is a real player, then
DESTROY THEM!

                        human.Health = 0 --Your Health Is Now 0.

            end

    end

    if (script.Parent ~= nil) and (script.Parent.className ==
"Part") then --Work if in a block

            connection =
script.Parent.Touched:connect(onTouched)

end
```

- Challenges and Excitement

Figure 24 shows the most important challenges that have been undertaken.

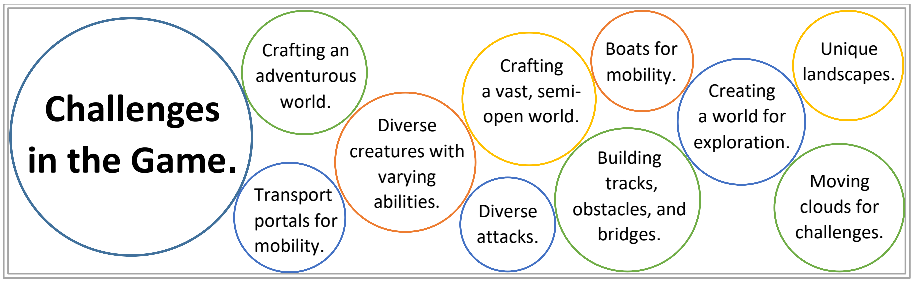

Challenges in the Game.
Crafting an adventurous world.
Transport portals for mobility.
Diverse creatures with varying abilities.
Crafting a vast, semi-open world.
Diverse attacks.
Boats for mobility.
Building tracks, obstacles, and bridges.
Creating a world for exploration.
Unique landscapes.
Moving clouds for challenges.

Figure 24: Challenges in the game

- Effects

Effects are additions to objects, whether visual, audio or software. At the entrance to the clouds room, shown in Figure 25, there are two fire pillars. The fire effect has been added to them and their properties have been modified to suit the entrance. Additionally, the fire ignition sound effect has been added and its sound characteristics have been modified so that it only plays when approaching the entrance.

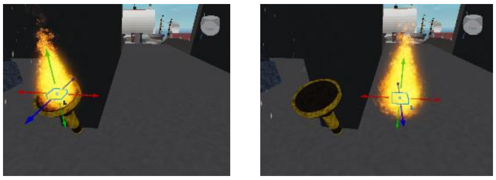


Figure 25: Add a fire effect

The game features various software effects, including the ability to run quickly by pressing the 'Shift' button, a programming file to allow the character to jump twice consecutively in mid-air, and a software file to enable swimming in the game's sea.

- Non-player character (NPC)

The game includes various non-player characters (NPCs) that the player does not control, such as portal characters, arms dealers, a shark, and characters in the clouds room and monsters in the second world.

C. Game Execution

To access the game, use the Roblox Player application from the official website. Select the desired game or search for it using the search bar. For the 'RSM game', either enclose the name in quotation marks or use the QR code shown in Figure 26. Finding the game by chance is difficult due to the large number of games on the platform.

The game features a unique icon, personally designed and shown in Figure27, as well as a custom cover, also displayed in Figure 28.

Figure 29 shows the game icon and cover. This is the login process.

Figure26: game's QR

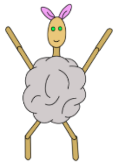


Figure27: Game icon

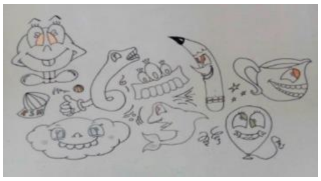


Figure28: Game cover

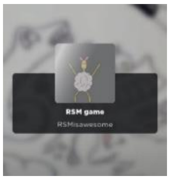


Figure29: Login to server

At the start of the game, the player emerges from the spawn portal into the first world. They can explore the world, drive cars, take boat rides, and swim in the sea, but must be cautious of sharks. Figure 30 displays various weapons stores, some of which are free while others require purchase. The boat and car are operated by touch.

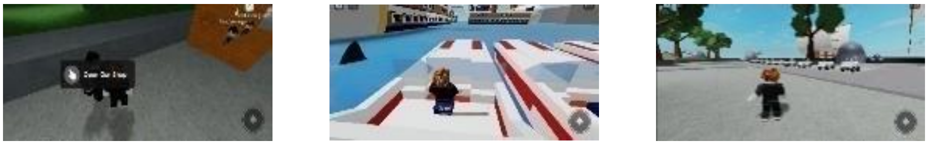


Figure 30: Vehicles and weapons shop

To communicate with friends, form a team, and plan the battle, the player can access the chat function by pressing the button illustrated in Figure 31.

There is a castle located in the centre of the first world. The castle includes a clouds room, as shown in Figure 32.

In this room, there is a dish labelled 'The World of Clouds in Arabic', conceals a door. When touched, the door leads to the second world depicted in figure 33.

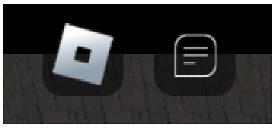


Figure 31: Chat button

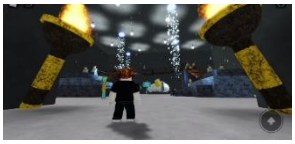


Figure 32: Clouds room gate

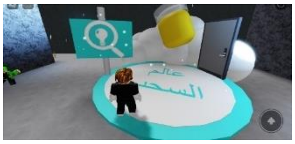


Fig 33: second world Gate

In the second world, the player is taken to Weapons Island where they can collect equipment such as rifles, shells, and more as depicted in figure 34.

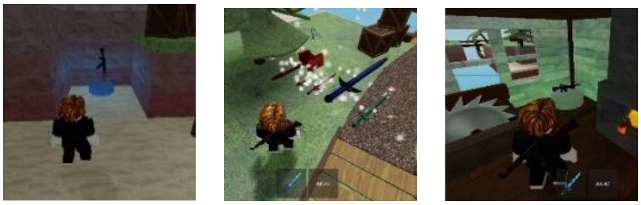


Figure 34: Weapons on Weapons Island

After preparing for the confrontation, the player travels to Monster Island. The only way to reach it is by travelling on small moving clouds. Figure 35 illustrates this. It is important to avoid falling down as there are small cloud balls that can cause death upon contact.

Upon arriving at Monsters Island, the player is immediately attacked by the cloud fairies using rain sprinklers. The player must avoid the attack and escape. As the player progresses, they encounter three spiders, each with a weak point. Although slow, the spiders can be defeated. In the final stage, the player confronts the boss, a swift samurai with a deadly strike and rapid regeneration upon death. Refer to figure 36 for a visual of all three monsters.

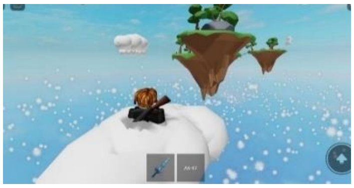


Figure 35: Moving clouds

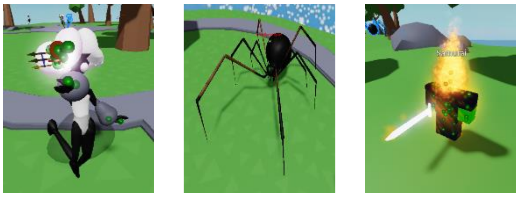


Figure 36: Three main monsters

The developers intentionally designed the game with unresolved challenges and unfinished elements as a strategy to conceal production flaws and prolong the development process.

VII. CONCLUSION

Developing games on the Roblox platform using Lua is an interesting topic. This paper establishes that game development is not limited to high-specification devices. Novice developers can also create their own games without joining a specific development company. However, the success of development depends on the developer's programming and design experience. The greater the experience, the better the development. The developer must understand, assimilate and develop their skills and information as in figure 37.

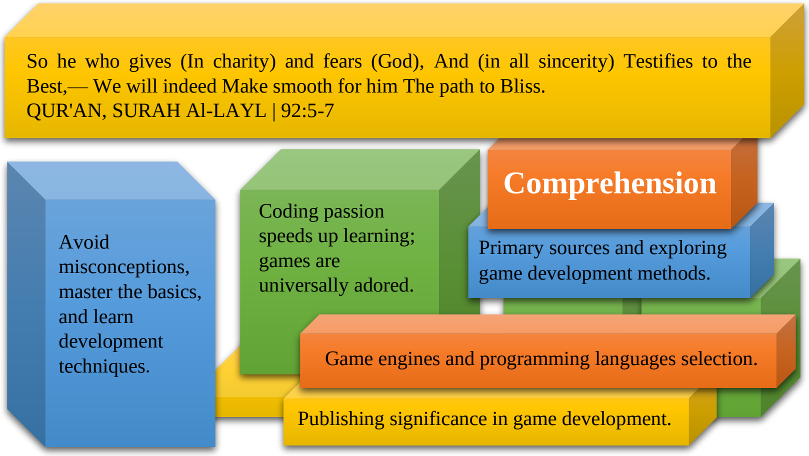

So he who gives (In charity) and fears (God), And (in all sincerity) Testifies to the Best,— We will indeed Make smooth for him The path to Bliss.
QUR'AN, SURAH Al-LAYL | 92:5-7
Avoid misconceptions, master the basics, and learn development techniques.
Coding passion speeds up learning; games are universally adored.
Comprehension
Primary sources and exploring game development methods.
Game engines and programming languages selection.
Publishing significance in game development.

Figure 37: Development understanding requirements

Developers must remember that video game development is a creative and technical process that requires commitment and perseverance. The success and quality of the game increase as the developer's skills and knowledge in this exciting field improve. Therefore, developers should strive to enhance their skills and knowledge in this field.

REFERENCES

[1] Á. Torres, "Game Development and Design: Principles and Practice," Paper presented at the University of La Frontera, June 15, 2023.

[2] B. Keogh, "The Videogame Industry Does Not Exist: Why We Should Think Beyond Commercial Game Production," The MIT Press, 2023.

[3] R. B. Jiwo and A. F. Aini, "Rancang Bangun Media Pembelajaran Berbasis Game menggunakan RPG Maker MV," Universitas Trunojoyo Madura, Indonesia, 2023.

[4] G. B. Escorcia, "Game Online: Ancaman 'Candu Digital'," Syntax Idea, vol. 4, no. 12, pp. 1786-1795, 2022.

[5] K. Khosiin, "Game Online: Ancaman 'Candu Digital'," Syntax Idea, vol. 4, no. 12, pp. 1786, 2022.

[6] C. Meier, J. L. Saorín, A. Bonnet de León, and A. Guerrero Cobos, "Using the Roblox Video Game Engine for Creating Virtual tours and Learning about the Sculptural Heritage," International Journal of Emerging Technologies in Learning (iJET), vol. 15, no. 20, pp. 268, October 2020.

[7] Y. Kou and X. Gui, "Harmful Design in the Metaverse and How to Mitigate it: A Case Study of User-Generated Virtual Worlds on Roblox," Proceedings of the ACM SIGCHI Conference on Designing Interactive Systems 2023 (DIS 2023), Pennsylvania State University, May 2023.

[8] K. Alhasan, K. Alhasan, and S. Al Hashimi, "Roblox in Higher Education: Opportunities, Challenges, and Future Directions for Multimedia Learning," International Journal of Emerging Technologies in Learning (iJET), vol. 18, no. 19, pp. 32-46, 2023. University of Kent, University of Bahrain.

[9] A. Tinterri, M.A. Guerriero, S. Annoscia, and A. Dipace, "Constructionism and game-making for learning in the age of Roblox: An analysis of current evidence and future perspectives," Italian Journal of Health Education, Sports, and Inclusive Didactics, vol. 7, no. 2, 2023. Edizioni Universitarie Romane.

[10] A. Abilkaiyrkyzy, A. Elhagry, F. Laamarti, and A. El Saddik, "Metaverse Key Requirements and Platforms Survey," IEEE Access, vol. 11, pp. 117765-117787, 2023.

[11] W. Ho and D. Lee, "Enhancing Engineering Education in the Roblox Metaverse: Utilizing chatGPT for Game Development for Electrical Machine Course," International Journal on Advanced Science, Engineering & Information Technology, vol. 13, no. 3, pp. 1052, 2023.

[12] T. S. Sinar, M. A. Budiman, R. Ganie, and R. N. Rosa, "Students' Perceptions of Using Roblox in Multimodal Literacy Practices in Teaching and Learning English," World Journal of English Language, vol. 13, no. 7, pp. 146, 2023.

[13] B. Berg Marklund, H. Engström, M. Hellkvist, and P. Backlund, "What Empirically Based Research Tells Us About Game Development," [Journal Name], vol. 8, no. 3, pp. 179-198, 2019.

[14] Y. Ma, "Design of Serious Games for Teaching Industrial Engineering Methodologies: A Design Process Based on V-Model and an Application in Innovation Engineering," (PhD thesis in Industrial Engineering), Université Paris-Saclay, 2021.

[15] D. Vagavolu, V. Agrahari, S. Chimalakonda, and A. S. M. Venigalla, "GE526: A Dataset of Open-Source Game Engines," in Mining Software Repositories, 2021, pp. 605-609.

[16] W. Amaldoss and J. Du, "How Can Publishers Collaborate and Compete with News Aggregators?" Journal of Communication, vol. 60, no. 6, 2023.

[17] C. McBride and G. Nakov, "Measuring with confidence: leveraging expressive type systems for correct-by-construction software," Acta IMEKO, vol. 12, no. 1, pp. 1-5, 2023.

[18] É. Piette, D.J.N.J. Soemers, M. Stephenson, C.F. Sironi, M.H.M. Winands, and C. Browne, "Ludii - The Ludemic General Game System," Frontiers in Artificial Intelligence and Applications, vol. 325, pp. 411-418, 2020.

[19] C. Martínez, C. Nicaud, and P. Rotondo, "Mathematical Models to Analyze Lua Hybrid Tables and Why They Need a Fix," arXiv:2208.13602v2 [cs.DM], 2023.

[20] A. Vlasova, M. Tigina, I. Vlasov, A. Birillo, Y. Golubev, and T. Bryksin, "Programming language usage," in Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22), pp. 398-402.

[21] A. Apicella, L.D. Lorenzo, F. Isgrò, A. Pollastro, and R. Prevete, "Strategies to exploit XAI to improve classification systems," in Proceedings of the 1st World

Conference on eXplainable Artificial Intelligence (xAI 2023), July 26-28, 2023, Lisboa, Portugal, arXiv:2306.05801 [cs.AI].

[22] F. Luo, W. Yan, R. Liu, and M. Israel, "Elementary Students' Understanding of Variables in Computational Thinking-Integrated Instruction: A Mixed Methods Study," in SIGCSE 2022: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1, pp. 523-529.

[23] M. Chernoff, "Automatically removing widows and orphans with lua-widow-control," TUGboat, vol. 43, no. 1, pp. 28-39, 2022.

[24] L. Maliang, L. Ma, D. Liu, and J. Li, "Lua-based method and a Lua-based device for obtaining a mobile phone application memory," published on June 11, 2019.

[25] Korean Institute of Communications and Information Sciences, "Design and Implementation of a Volumetric Video Player based on Point Cloud," Journal of the Korean Institute of Communications and Information Sciences, vol. 47, no. 10, pp. 1,660-1,668, 2022.