

**UNIVERSITY TEKNOLOGI MARA**

**CODE[X]: NEXUS, A NARRATIVE-  
DRIVEN 2D TOP-DOWN STYLE  
ADVENTURE GAME ABOUT  
INTRODUCTORY-LEVEL C++  
PROGRAMMING USING GODOT 4**

**AMIR HAFIZI BIN MUSA**

**BACHELOR OF COMPUTER SCIENCE (HONS.)**

**JULY 2025**

**Universiti Teknologi MARA**

**Code[X]: Nexus, a Narrative-Driven 2D  
Top-Down Style Adventure Game about  
C++ Programming using Godot 4**

**Amir Hafizi Bin Musa**

**Thesis submitted in fulfillment for  
Bachelor of Computer Science (Hons.)  
Faculty of Computer and Mathematical Sciences**

**JULY 2025**

## **SUPERVISOR'S APPROVAL**


**CODE[X]: NEXUS, A NARRATIVE-DRIVEN 2D TOP-DOWN STYLE  
ADVENTURE GAME ABOUT C++ INTRODUCTORY-LEVEL  
PROGRAMMING USING GODOT 4**

By

**AMIR HAFIZI BIN MUSA  
2024745815**

This thesis was prepared under the direction of thesis supervisor, Ahmad Farid Bin Najmuddin. It was submitted to the Faculty of Computer and Mathematical Sciences and was accepted in partial fulfilment of the requirements for the degree of Bachelor of Computer Science (Hons.).

Approved by:

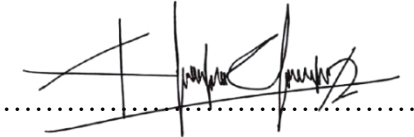


.....  
Ahmad Farid Bin Najmuddin  
Thesis Supervisor

JULY 9, 2025

## **DECLARATION**

I certify that this report and the research to which it refers are the product of my own work and that any ideas or quotation from the work of other people, published or otherwise are fully acknowledged in accordance with the standard referring practices of the discipline.

A handwritten signature in black ink, appearing to read 'Amir Hafizi Bin Musa', is written over a horizontal dotted line.

**AMIR HAFIZI BIN MUSA**

**2024745815**

**JULY 9, 2025**

# TABLE OF CONTENTS

<b>CONTENTS</b>	<b>PAGE</b>
<b>SUPERVISOR’S APPROVAL</b>	ii
<b>DECLARATION</b>	iii
<b>TABLE OF CONTENTS</b>	iv
<b>LIST OF FIGURES</b>	vi
<b>LIST OF TABLES</b>	vii
<b>LIST OF ABBREVIATIONS</b>	viii

## **CHAPTER ONE: INTRODUCTION**

1.1	Introduction	1
1.2	Background of Study	2
1.3	Problem Statement	3
1.4	Project Objectives	5
1.5	Scope of Study	5
1.6	Significance of Study	7
1.7	Summary	8

## **CHAPTER TWO: LITERATURE REVIEW**

2.1	Introduction	9
2.2	Overview of Computer Science Education	11
2.3	Overview of Game-Based Learning (GBL)	12
2.4	Specific Description of Game-Based Learning	13
2.5	Techniques in Game-Based Learning	14
2.5.1	Progression System	15
2.5.2	Challenge and Quest	16
2.5.3	Narratives and Storytelling	17

2.6	Common Features Related to Proposed Project	20
2.6.1	Colobot: Gold Edition	21
2.6.2	CodeCombat	23
2.6.3	Human Resource Machine	26
2.6.4	Features Comparison of Related Game-Based Learning Games	29
2.7	Highlight the Chosen Techniques & Features with Justification	30
2.8	Summary	35

### **CHAPTER THREE: METHODOLOGY**

3.1	Introduction	36
3.2	Project Methodology	36
3.2.1	Design	39
3.2.2	Develop / Re-Develop	42
3.2.3	Evaluate	42
3.2.4	Test	43
3.2.5	Review & Release Preparation	44
3.2.6	Release	44
3.3	Summary of Project Methodology	45
3.4	System Architecture	47
3.5	Hardware and Software Requirements	52
3.6	Conclusion	55

<b>REFERENCES</b>	<b>56</b>
-------------------	-----------

## LIST OF FIGURES

FIGURE	PAGE
2.1 Tree Structure of the Proposed Project	10
2.2 Examples of using leveling up and stage levels selection as the game's core progression system.	15
2.3 Examples of games using challenges to immerse players into the game world	16
2.4 Examples of games using quest system to give players goals to pursue	16
2.5 Examples of games using NPC dialogue system for story progression	17
2.6 Examples of games using cutscenes to lock players engagement	18
2.7 Main menu of Colobot: Gold Edition	22
2.8 World design concept of Colobot: Gold Edition	22
2.9 Mission menu of Colobot: Gold Edition	22
2.10 Program editor to control the robot mechanism	23
2.11 World selection map inside of CodeCombat	24
2.12 One of coding mission inside CodeCombat	24
2.13 Characters menu of CodeCombat	25
2.14 Map exploration inside chosen world	25
2.15 Human Resource Machine storyline and dialog system	27
2.16 Level design of Human Resource Machine stage	27
2.17 Mission challenges of Human Resource Machine	27
2.18 Levels progression of Human Resource Machine	28
2.19 Examples of games using own unique system and cutscene to deliver the storylines.	30
2.20 Examples of games using dialogue for NPC to progress the story.	31
2.21 Examples of direct narration to lead players gameplay direction	32
2.22 Examples of games using 2D top-down pixel art style genre	32
2.23 Examples of games using the freedom of art style creation	33
2.24 Godot 4 Game Engine Logo	33

2.25	Godot 4 Game Engine Scene Editor Interface	34
2.26	Godot 4 Game Engine Code Editor Interface	34
3.1	Game Development Life Cycle	37
3.2	Storyboard Draft for the "Conduit's Variable" Quest	40
3.3	Concept Art for the Player Character and a Game World Area	41
3.4	System Architecture Design of Proposed Project	49
3.5	Flowchart diagram for the proposed project	50
3.6	Use case diagram of the proposed project	51



## LIST OF TABLES

<b>TABLE</b>	<b>PAGE</b>
2.1 GBL Techniques Comparison	18
2.2 Related GBL games preview image	20
2.3 Comparison to other Related GBL Games	29
3.1 Project Methodology (GDLC)	37
3.2 Concept of the designed project quest's dialogues	41
3.3 Summary of Project Methodology	45
3.4 Software Requirements Details	52
3.5 Hardware Requirements Details	53

## **LIST OF ABBREVIATIONS**

CBOT	C++ inspired language
CS	Computer Science
GBL	Game-Based Learning
GDLC	Game Development Life Cycle
HRM	Human Resource Machine
HUD	Head-Up Display
MEEGA+	Multidimensional Evaluation of Educational Games Plus
NPC	Non-Player Character
RPGs	Role-Playing Games
SMEs	Subject Matter Experts
UI	User Interface
UX	User Experience

# CHAPTER 1

## INTRODUCTION

### 1.1 Introduction

In today's world, the processes of learning and teaching are transforming at a rapid rate. Conventional methods such as books and lectures are often insufficient to engage the interest of most students, particularly when it comes to complex subjects like Computer Science (Al-Maroofof et al., 2024). With technology becoming deeply integrated into our lives, there is an increasing demand for innovative and efficient ways to learn, one of which is Game-Based Learning (GBL).

GBL exploits the playfulness and interactive nature of computer games to present learning concepts in an easy and enjoyable manner. This approach can simplify learning, making it more enjoyable and facilitating an easier understanding of abstract ideas that otherwise cause problems (de Oliveira, de Oliveira, & de Souza, 2022). However, while GBL in Computer Science has the potential to reduce or eliminate study difficulties, it is still not as commonly utilized as it could be in learning games designed for Computer Science students (Kalogiannakis, Papadakis, & Zourmpakis, 2021).

Briefly, this study will examine how computer games are able to aid students in the Computer Science field by giving them a better, interactive way of learning while contributing to the growing area of educational technology.

## 1.2 Background of Study

Nowadays, the world is witnessing significant and rapid changes in technology, which are impacting all aspects of life, including education. With the prevalence of digital media, online environments, and interactive technologies, the effectiveness of traditional classroom learning in the 21st century is being called into question. While textbook and lecture-based learning have been central to formal education for centuries, contemporary students are increasingly drawn towards interactive, graphical, and game-like modes of engagement (Kalogiannakis et al., 2021).

In recent years, educational studies have highlighted a widening gap between traditional teaching methods and the engagement levels of students, particularly in demanding subjects like Computer Science, History, and Mathematics (Al-Marooof et al., 2024; Tan, Neill, & D'Souza, 2021). Such subjects often involve sequential concepts or abstract ideas that can be difficult to grasp and retain in passive learning environments. Traditional lectures, which are often non-interactive and lack modern storytelling elements, can leave students feeling bored, demotivated, or confused (Chan et al., 2023). This has led to the increased adoption of technology in education, particularly in the form of Game-Based Learning (GBL)

GBL is the application of video games and their mechanics to teach specific skills or knowledge domains in an interactive and engaging manner. Current research suggests that learning games have the potential to significantly improve memory recall, problem-solving abilities, and the motivation to learn (de Oliveira, de Oliveira, & de Souza, 2022). By combining a story with interactive challenges, learning games offer a dynamic alternative to standard educational methods. Among different game genres, 2D top-down style adventure games are ideally suited for educational experiences because they provide a linear progression that allows developers to seamlessly incorporate puzzles, narratives, and question-based challenges (van Gaalen et al., 2021). If designed with care, such games can become effective instruments for reinforcing learning goals while keeping the player interested and curious. Yet, despite the established advantages of gamification in education, many institutions continue to

rely on static content delivery. This disconnect contemporary learning preferences and conventional educational tools presents an opportunity to investigate new methods that can improve student learning outcomes.

Thus, this study proposes the development of the project, a 2D Learn & Gain top-down style adventure video game based on the Godot Engine (v4.4), for university students studying Computer Science. Interactive puzzle, story, and gamified obstacle perhaps will help the students to have a more engrossing, stimulating, and effective learning experience that meets the cognitive and motivational needs of the modern learner.

### **1.3 Problem Statement**

As the world of technology expands, the schooling system needs better ways of teaching because some of them still use passive methods of teaching like lectures. Traditional ways of teaching like lectures, textbooks, and slide presentations remain, but these methods do not necessarily engage students actively or support knowledge retention, especially in technical fields like Computer Science (Chan et al., 2023).

Most learners, especially the first-year diploma students of university, are struggling to be interested in Computer Science (Tan, et al., 2021; Holly, et al., 2024). This is based on the reality that the subject is very analytical and abstract, and is often delivered in non-interactive and static approaches that are criticized for being too theoretical (Al-Marroof et al., 2024; Holly et al., 2024). There is also a significant gap in technology that facilitates individuals to learn in a fun way, for example through games and stories (Cheong, Flippou & France, 2020). When there is no context, problem-solving activities, and stories, students may struggle to relate theory to real-life situations, which can lead to lower motivation and performance (de Oliveira, de Oliveira, & de Souza, 2022).

Game-Based Learning (GBL) has been one such remedy for these issues by the incorporation of learning content with engaging mechanics that encourage analytical thinking, collaboration, and exploration (Kalogiannakis, et al., 2021). However, though gamified learning solutions have gained growing popularity, most available solutions are either too simple or not exactly tailored to the requirements of the Computer Science curriculum (Tan, et al., 2021). In addition, students' varied learning needs demand experiences that are not only interactive, but narrative, image-centered and pedagogically designed.

The study by Kucher, T. (2021) conclude that games that have definite stories, decisions to proceed and associated challenges assisted students in recalling procedural and theoretical information more effectively than normal tests. No such tools are being created in tertiary education, particularly for Southeast Asian Computer Science students (Kucher, T., 2021).

To summarize, students struggle to remain engaged and retain information when learning Computer Science through traditional methods like lectures and textbooks. Existing game-based learning software often fails to address this, being either too broad or lacking the specific, structured content required for the subject. Consequently, there is a scarcity of interactive, story-driven educational games that are specifically mapped to the curriculum of first-year university diploma courses in computer science. This continuous lack of innovative tools negatively impacts students' motivation, engagement, and overall performance in their technical courses.

## 1.4 Project Objectives

To address the problem of student motivation and knowledge retention in Computer Science learning, this project aims to design a 2D Learn & Gain Top-down Style Adventure Game for first-year university diploma students to extend learning by using interactive storytelling and lesson gamification. In this respect, objectives are to:

1. To identify an effective game-based learning (GBL) method in programming education in enhance student motivation.
2. To develop a 2D top-down style learning adventure game, in learning programming concepts that includes GBL elements using Godot Engine v4.4.
3. To assess the usability of the developed GBL game towards the targeted students.

## 1.5 Scope of Study

The project will focus on the implementation of Game-Based Learning (GBL) in Computer Science focusing on programming topics education through a 2D top-down style adventure game. The project aims to explore on how interactive narrative and puzzle goes within a top-down game can helps students' interest, improve concept comprehension, and improve memory recall, especially among university students. The research investigates on the way of how study content can be more gamified and placed within a context of a narrative. This integrates learning with advancement in a game in overcoming common problems related to the traditional means of learning.

The proposed solution of the project is able to provide:

- A narrative-driven gameplay experience with progression themed levels representing variety of fundamental topics in C++ programming field.
- A challenge-based system that enables students lean and progress by answering questions or solving puzzles in relation to the subject field.

- A visual cues and storytelling elements designed to enhance and improve students understanding regarding the concept topics such as history and overview of C++, basic syntax and data types, basic data structures and basic algorithms.

This project will be developed using the Godot v4.4 game engine, it will be targeted towards the Windows and Web (HTML5) platforms for deployment. The game is designed to be lightweight, has low system specification requirements and easily accessible via browser or Windows executable file (.exe), while also suitable for both classroom use or even independent learning. The target users are first-year of diploma in Computer Science of UiTM who are new to Computer Science or are struggling with the foundation concept.

The scope of the game's functionality revolves around a single-player, 2D top-down RPG-style adventure with level-based progression. It is designed to offer a narrative-driven gameplay experience where levels are themed to represent fundamental topics in the C++ programming language. The core learning mechanic is a challenge-based system that allows students to progress by answering questions or solving puzzles related to their studies. This is supported by visual cues, storytelling elements, assets, and animations designed to enhance the learning themes. The game will also feature basic performance tracking, monitoring metrics such as completed questions and total playtime.

The project is intentionally restricted to introductory-level programming topics. Specifically, the educational content will cover the history and overview of C++, basic syntax and data types, basic data structures, and basic algorithms. The target users are specifically first-year Diploma in Computer Science students at UiTM who are either new to the field or struggling with foundational concepts. This project will be developed using the Godot Engine v4.4 and will be targeted for deployment on Windows and Web (HTML5) platforms.



## **1.6 Significant of Study**

With the presented solutions, the significant towards the body of knowledge lies within the utilization of Game-Based Learning (GBL) principles in Computer Science education, more precisely the learning of abstract and procedural material by students. This will help in emerging the area of educational technology and gamification, showing how narrative-based game environments can enhance learning experiences and retention rates among students in the related field.

The significance of this study is twofold, first it offers a practical model for the development of learning games for topic-specific content towards other game developers and it presents the argument for the value of interactivity, narrative, and challenge-based progression while fostering student motivation and understanding. In doing so, it helps bridge the gap between abstract theory and actual practice in an interesting and easily accessible way.

This study is primarily useful for first-year university diploma computer science students who struggle with traditional learning, offering them an entertaining and enjoyable alternative that transforms difficult programming concepts into a more easily digestible format. It also benefits lecturers, who can utilize the resulting game as an alternative teaching tool to enhance their lessons, encourage critical thinking, and introduce variation in classwork. Ultimately, the study aims to provide a convenient learning environment where students can learn at any time and from anywhere, thereby promoting self-learning.

Through the integration of narrative and subject-based challenges, the project demonstrates the ways in which purposeful game design can positively enhance the practice of education by promoting more effective creation and accessible learning software in the digital era where we must adapt, not reject.

## 1.7 Summary

This chapter describes that Computer Science education must be enhanced by innovative and interactive approaches like Game-Based Learning (GBL). Conventional teaching cannot hold students' attention span and interest, particularly if the subject matter is complicated or abstract in nature, such as in the case of algorithms and binary logics.

Because of this, in order to address this issue, the proposed project will provide interactive and narrative learning experiences. The purpose of this proposal is to enhance student engagement and retention by incorporating educational materials within a gamified system, developed using the Godot Engine v4.4.

The solution scope involves targeting first-year university diploma students with an emphasis on programming courses at the introductory level through single-player mode with puzzles and quizzes that are dialogue-based. The study significance involves the contribution to educational technology through the demonstration that storytelling, game design, and visual learning can come together to work effectively in augmenting and supplementing the current model of formal education.

## **CHAPTER 2**

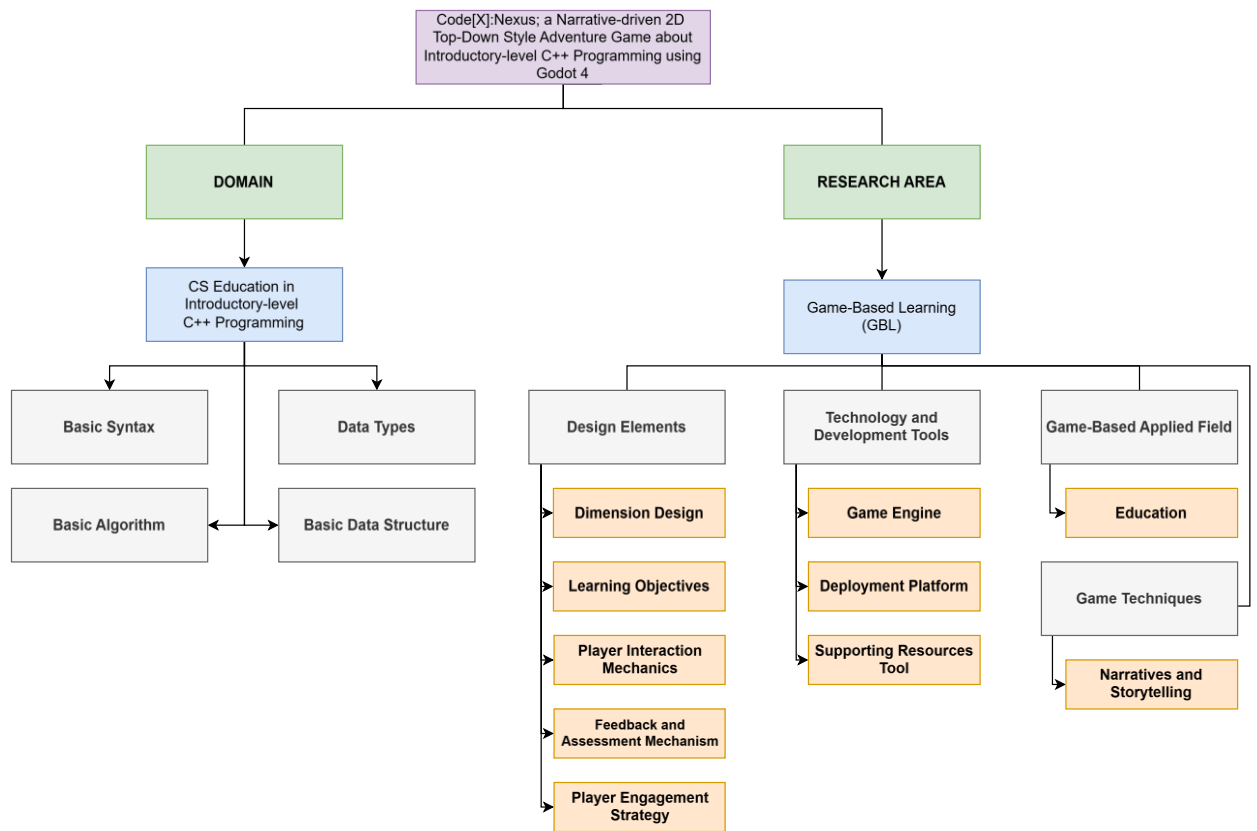
### **LITERATURE REVIEW**

#### **2.1 Introduction**

This chapter provides an overview of current research regarding Game-Based Learning (GBL) and its implementation in Computer Science education, specifically in the instruction of introductory programming. This is to lay down a theoretical background for this project, an interactive 2D top-down adventure game created with the Godot Engine as an instruction medium of C++ programming for first-year diploma Computer Science students.

Classical teaching methods lead to disengagement among students since programming concepts are intangible and decontextualized (Tan et al., 2021). Educators have subsequently looked for substitute interactive and engaging learning tools, like games, that promote active learning, problem-solving and engagement through gameplay mechanics including progression mechanisms, challenges and story (Qian & Clark, 2019; Cheong et al., 2020).

This literature review aims to explore relevant GBL approaches in education program games such as Colobot: Gold Edition, CodeCombat and Human Resource Machine that identifies elements which enabling effective learning process. These will be utilized to inform design within this project in a way that considers the integration of actual programming tasks in an interactive story environment that is appropriate for new students (Kucher, 2021). Figure 2.1 shows the overall tree structure of the proposed project.



**Figure 2.1** Tree Structure of the Proposed Project

## 2.2 Overview of Computer Science Education

Computer Science (CS) education has become a widely viewed technical education in the modern era, particularly at diploma and undergraduate levels. Nevertheless, it is documented that many freshman CS students struggle with fundamental programming concepts, most notably due to the fact that subjects such as variables, loops, conditionals and object-oriented design are often abstract in nature (Papastergiou, 2009; Holly et al., 2024).

Traditional education heavily relies on methods like lectures, notes, slides, textbook and coding exercises, which has shown often fail to fully engage students or provide meaningful context for learning process (Tan et al., 2021). This causes the drive in having low motivation, poor retention rates, low attention span and high dropout rates especially among first-year students who are just completely new to programming (Arnedo & García, 2023).

Studies have determined that the majority of students consider programming as difficult and out of reach, primarily because ideas are taught in isolation without clear real-world applications (Cheong et al., 2020). In addition, the lack of immediate feedback and interactivity in conventional settings limits channels for active learning, trial and error, which required to gain mastery in programming field (Qian & Clark, 2019).

To counter these challenges, educators and research professionals have resorted to alternative learning strategies, including interactive tools, visual programming environments and gamified learning systems (Kucher, 2021). All these are designed to increase access to programming by positioning learning in immersive and contextualized experiences.

One of the most promising areas is the integration of Game-Based Learning (GBL) into the teaching of CS. Learning games are structured but enjoyable environments in which students can learn by trying out, problem-solving and gradually developing mastery over concepts and representations of intellectual processes employed in actual programming (van Gaalen et al., 2021). Examples of games such as Colobot: Gold

Edition, CodeCombat and Human Resource Machine showcase how learning goals can be transferred to game mechanics to improve participation and comprehension (Heithausen, 2020).

With the growing demand for skilled programmers and the importance of early success in shaping students' confidence in CS, there is a strong need to explore new methods, which aligned with today's digitally native students' demands and learning patterns (Boyle et al., 2016).

## **2.3 Overview of Game-Based Learning (GBL)**

Game-Based Learning (GBL) is the practice of using digital games in learning contexts to facilitate improved learning outcomes through an interactive and dynamic experience (Kalogiannakis, et al., 2021). While traditional methods often rely on passive learning approaches such as reading books and attending lectures, GBL promotes active engagement by allowing students to tackle problems, receive immediate feedback, and progress at their own pace (de Oliveira, et al., 2022).

The theoretical foundation of GBL is closely linked to experiential learning, which posits that learning occurs most effectively when students engage in a concrete experience, followed by reflection and application (Lozano-Lozano et al., 2023; Cheong et al., 2020). Computer games inherently support this cycle by immersing players in challenges that require decision-making, experimentation, and iterative refinement. Research suggests that games are effective because they align with the learning patterns of contemporary students, who are accustomed to fast-paced, visually rich, and interactive digital environments (Chan et al., 2023).

Because of this, it increases the motivation for them, which is a key factor in student engagement and long-term knowledge retention (Holly et al., 2024). Furthermore, games can also reduce cognitive load by presenting information in manageable chunks, allowing the students to build confidence gradually (Choi & Choi, 2024).

GBL has shown promises in helping students grasp abstract concepts like loops, conditionals, functions and object-oriented programming (Tan et al., 2021). Existing GBL games like Colobot: Gold Edition, CodeCombat and Human Resource Machine have proved that engaging world building and well-crafted gameplay mechanics, including progression systems, meaningful challenges and immediate feedback significantly enhance learning while improving understanding of programming basics (Arnedo & García, 2023; Heithausen, 2020).

Even though GBL will not replace traditional teaching practices, it can be used as a tool that enhances motivation, promotes self-learning and supports experimentation, which are key processes in acquiring programming skills (van Gaalen et al., 2021).

These insights lay the foundation for the proposed project, which integrates GBL elements into a narrative-driven 2D top-down adventure game designed in teaching introductory-level C++ programming to first-year diploma students.

## **2.4 Specific Description of Game-Based Learning**

Game-Based Learning (GBL) goes beyond entertainment by integrating structured learning objectives into gameplay, which promotes active engagement rather than passive reception (Kalogiannakis et al., 2021). Games act as interactive systems that can simulate real-world challenges, promote decision-making, and offer immediate feedback for key elements that enhance deep learning and retention (Cheong et al., 2020).

According to recent studies, well-designed educational games boost student motivation and understanding, particularly for complex or abstract topics like programming logic and syntax (Al-Marroof et al., 2024). This aligns with experiential learning theory, which posits that meaningful learning happens through hands-on experiences, reflection, and application. Digital games naturally facilitate this process by immersing players in structured challenges that involve problem-solving, trial and error, and iterative improvement, effectively mirroring real-world programming tasks (Lozano-Lozano et al., 2023).

In GBL environments, progressive difficulty, embedded tutorials and storytelling help students build confidence and competence gradually (van Gaalen et al., 2021). Games like Colobot: Gold Edition, CodeCombat and Human Resource Machine align gameplay with learning goals to boost engagement and understanding (Heithausen, 2020; Arnedo & García, 2023). They introduce programming concepts step-by-step, starting with basic commands, then advancing to loops, conditionals, and functions (Heithausen, 2020).

GBL supports self-directed learning, letting students explore content at their own pace and revisit difficult areas without pressure (Kucher, 2021). It also enables mastery-based progression, requiring a full understanding of one concept before advancing, which reinforces fundamentals before introducing greater complexity (Zainuddin et al., 2020).

In introductory-level programming, GBL offers a low-stakes environment where mistakes are part of learning, not a failure (Choi & Choi, 2024). This is crucial for first-year diploma students lacking confidence in technical subjects. By presenting programming as gameplay, students become more open to experimenting, debugging, and refining code, concluding the key skills in software development.

## **2.5 Techniques in Game-Based Learning**

Game-Based Learning (GBL) relies on a variety of engagement-enhancing techniques that promote motivation, knowledge retention and problem-solving skills. Among the most impactful are progression systems, challenge-based puzzles and narrative-driven storytelling (van Gaalen et al., 2021). These techniques are particularly effective in educational contexts such as introductory programming instruction, where abstract concepts can be difficult for students to grasp without meaningful context or structured support (Al-Marroof et al., 2024).



## 2.5.1 Progression System

A progression system is a game design mechanism that allows players to advance through increasing levels of difficulty, often unlocking new abilities, tools, or content along the way (Cheong et al., 2020). In educational settings, progression systems help enhance learning by gradually introducing new challenges while reinforcing previously acquired knowledge (Zainuddin et al., 2020). This approach aligns with mastery-based learning, where students must demonstrate understanding before moving to more complex topics.

In programming education, this structure mirrors how coding concepts are typically introduced, starting with basic syntax and logic before progressing to loops, conditionals, functions, and object-oriented principles (Arnedo & García, 2023). Games like Colobot: Gold Edition and CodeCombat use level-based progression to guide players through increasingly complex programming tasks, making them ideal models for this project (Heithausen, 2020). Figure 2.2 shows the example usage of leveling system as the main progression system.

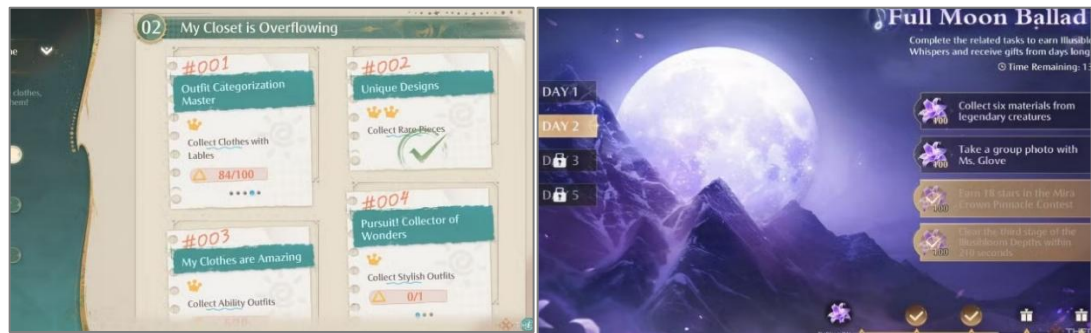


**Figure 2.2** Examples of using leveling up and stage levels selection as the game's core progression system.

## 2.5.2 Challenge and Quest

Challenge-based gameplay involves presenting students with problems or quests that require logical reasoning, decision-making, and iterative trial-and-error (Kucher, 2021). Puzzles are one of the most commonly used challenge formats in educational games, especially those designed for teaching programming fundamentals (Heithausen, 2020).

Games like Human Resource Machine present players with input-output puzzles that simulate real-world programming problems (e.g., sorting values, comparing numbers, managing memory cells), encouraging algorithmic thinking and debugging skills (van Gaalen et al., 2021). These mechanics not only reinforce technical knowledge but also foster self-directed exploration, which is essential for mastering complex subjects like programming (Choi & Choi, 2024). Figure 2.3 shows the usage of challenges system, meanwhile figure 2.4 shows the example of quest system.



**Figure 2.3** Examples of games using challenges to immerse players into the game world



**Figure 2.4** Examples of games using quest system to give players goals to pursue

### 2.5.3 Narratives and Storytelling

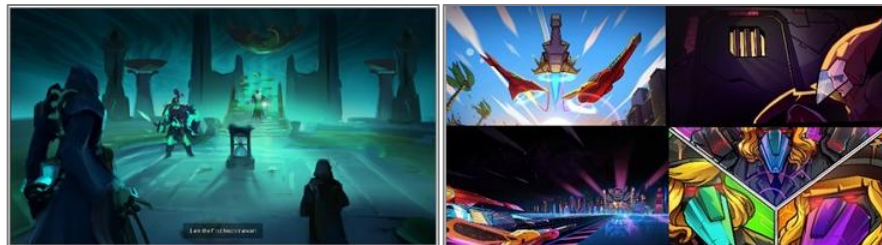
Narrative elements provide emotional engagement and contextual meaning to gameplay, making them powerful tools for enhancing motivation and memory recall (Kucher, 2021). In educational games, stories help frame abstract problems within relatable scenarios, allowing players to better understand the relevance and application of what they're learning (Cheong et al., 2020).

For example, Colobot: Gold Edition immerses players in a science fiction storyline where they must program robots to colonize alien planets using a C++ inspired language called CBOT (Alves & Letouze, 2018). This narrative-driven approach increases student investment and encourages deeper cognitive processing of programming concepts (Giannakoulas & Xinogalos, 2024).

While some educational games like CodeCombat focus more on mechanics than story, research suggests that adding strong narrative components significantly enhances both emotional engagement and long-term interest in learning (van Gaalen et al., 2021). This shapes the proposed project's design, uniting all three techniques within a cohesive, narrative-driven experience. Figure 2.5 shows the usage of narrative dialogue system, meanwhile figure 2.6 shows the usage of narrative cutscenes.



**Figure 2.5** Examples of games using NPC dialogue system for story progression



**Figure 2.6** Examples of games using cutscenes to lock players engagement

## 2.5.4 Techniques Comparison

**Table 2.1** GBL Techniques Comparison

	Progression System	Challenge And Quest	Narratives And Storytelling
<b>Definition</b>	Players advance through increasing levels of difficulty, unlocking new content or abilities.	Players solve problems or quests requiring logical reasoning and problem-solving.	Players engage with a storyline that provides emotional context and immersion.
<b>Educational Impact</b>	Promotes mastery-based learning; reinforces prior knowledge before introducing complexity.	Encourages algorithmic thinking, debugging, and iterative learning.	Enhances memory recall, contextual understanding, and emotional investment.
<b>Engagement Level</b>	- High Players feel a sense of achievement as they progress.	- High Challenges stimulate curiosity and persistence.	- Very High Narrative increases emotional connection and long-term interest.
<b>Complexity</b>	Moderate	Moderate to High	High
<b>Relevance to Programming Education</b>	Supports gradual introduction of programming concepts (e.g., syntax, logic, functions).	Reinforces problem-solving and logical reasoning essential for coding.	Helps contextualize abstract concepts within relatable scenarios.
<b>Examples in Educational Games</b>	CodeCombat, Colobot	HRM, CodeCombat	Colobot, HRM, CodeCombat
<b>Limitations</b>	May become repetitive if not paired with variety.	Can frustrate learners if too difficult or unclear.	Time-consuming and may distract from core learning.

Table 2.1 compares the Game-Based Learning (GBL) techniques in Colobot: Gold Edition, CodeCombat, and Human Resource Machine. It highlights that while progression systems and challenges are common, strong narrative integration is not always present, informing the design choices for this project.

## 2.6 Common Features Related to Proposed Project

Several educational games have been developed over the years to support introductory programming instruction, particularly through Game-Based Learning (GBL) techniques such as progression systems, challenge-based puzzles, and narrative elements (Cheong et al., 2020; Qian & Clark, 2019). Three notable examples that align closely with the proposed project are Colobot: Gold Edition, CodeCombat, Human Resource Machine. Each of these games provides a unique approach in teaching programming concepts and has been used in both formal and informal learning environments. Table 2.2 provides the preview of the game's look.

**Table 2.2** Related GBL games preview image

	
Colobot: Gold Edition	CodeCombat
	
Human Resource Machine	



### 2.6.1 Colobot: Gold Edition

Colobot: Gold Edition is a science fiction-themed game in which players control programmable robots to colonize alien planets using own C++ like language called CBOT (Alves & Letouze, 2018). The gameplay involves solving missions that teach core programming concepts such as loops, conditionals, functions, arrays and object detection.

The game possesses several key strengths, notably its use of a C++ inspired language that makes it highly relevant for students transitioning to real-world programming. Its integration of a strong narrative enhances emotional engagement and motivation, while the gameplay encourages self-directed exploration, problem-solving, and a mastery-based progression. Furthermore, it supports community-driven content creation, which allows educators to extend or customize the game for their specific classroom needs.

However, the game is not without its limitations. The interface, with its 3D environment and complex controls, can be overwhelming for beginners. This issue is compounded by the lack of a structured tutorial system, potentially hindering accessibility for novice students. Additionally, due to its strong technical focus, the game may not appeal equally to all learning styles.

Studies show that Colobot: Gold Edition promotes active learning and enhances understanding of programming logic when integrated into curricula (Alves & Letouze, 2018). Students who engaged with the game reported increased interest in programming and improved confidence in tackling complex tasks. This is because of the comprehensive game design as shown in figure 2.7, 2.8, 2.9 and 2.10.



**Figure 2.7** Main menu of Colobot: Gold Edition

This figure shows a main menu screenshot from *Colobot: Gold Edition*. It illustrates how players interact with the UI system of the game.



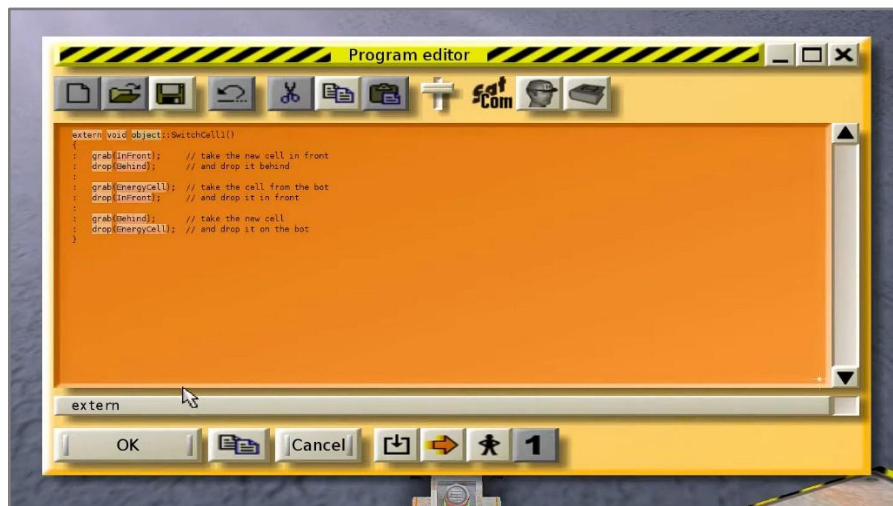
**Figure 2.8** World design concept of Colobot: Gold Edition

This figure displays the world design of the game, which the users are able to fully explore and learn on their own.



**Figure 2.9** Mission menu of Colobot: Gold Edition

This figure presents a mission selection system of the game, players will be able to easily select which mission to start on first.



**Figure 2.10** Program editor to control the robot mechanism

This figure shows the usage of in-game program editor for players to use in completing the required mission of the game.



### 2.6.2 CodeCombat

CodeCombat is a browser-based game that teaches real Python and JavaScript syntax by having players write code to solve puzzles and progress through levels (Kroustalli & Xinogalos, 2021). Each level presents a challenge that must be solved using actual programming commands, often involving loops, conditionals, and function calls.

The platform's strengths lie in its practical approach to teaching real coding syntax, making it ideal for students preparing for professional development environments. It offers immediate feedback, which helps students identify and correct mistakes quickly, and includes a gamified progression system that increases motivation and persistence. Furthermore, its effectiveness in reducing cognitive load and improving syntax understanding has been proven.

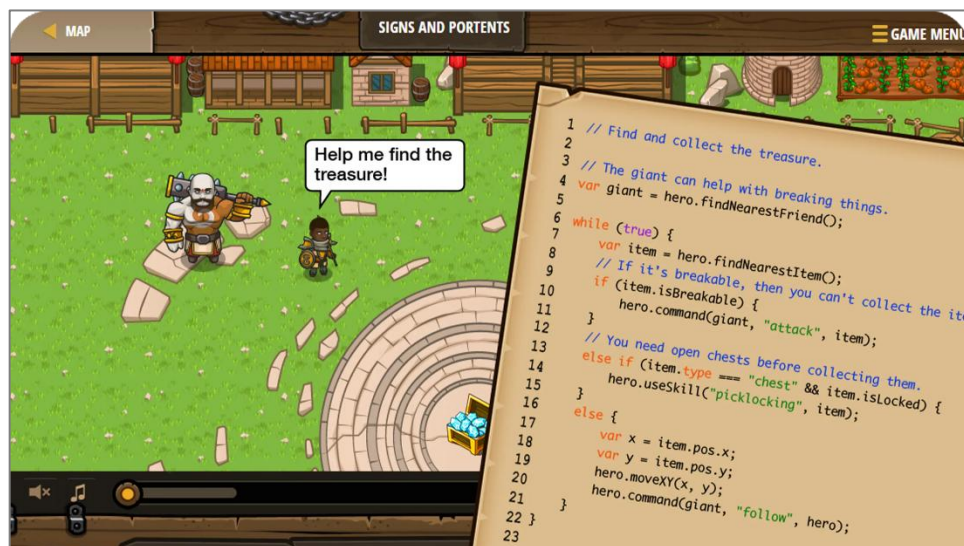
Nevertheless, the platform has some limitations, including a minimal narrative depth that can limit emotional investment compared to more story-driven games. Some students also found the experience repetitive after completing the core levels, and its limited gameplay variety could potentially affect long-term engagement.

Research indicates that CodeCombat significantly improves students' interest in programming, especially among those who previously found traditional methods unengaging (Kroustalli & Xinogalos, 2021). However, many students expressed willingness to continue only if new challenges or narrative elements were introduced. Figure 2.11, 2.12 , 2.13 and 2.14 shows the overall game structure of this CodeCombat.



**Figure 2.11** World selection map inside of CodeCombat

This figure displays the world selection map in CodeCombat, showing themed regions (e.g., "Web Development," "Game Development") with levels marked by progress indicators. Players can select levels to engage in coding challenges.



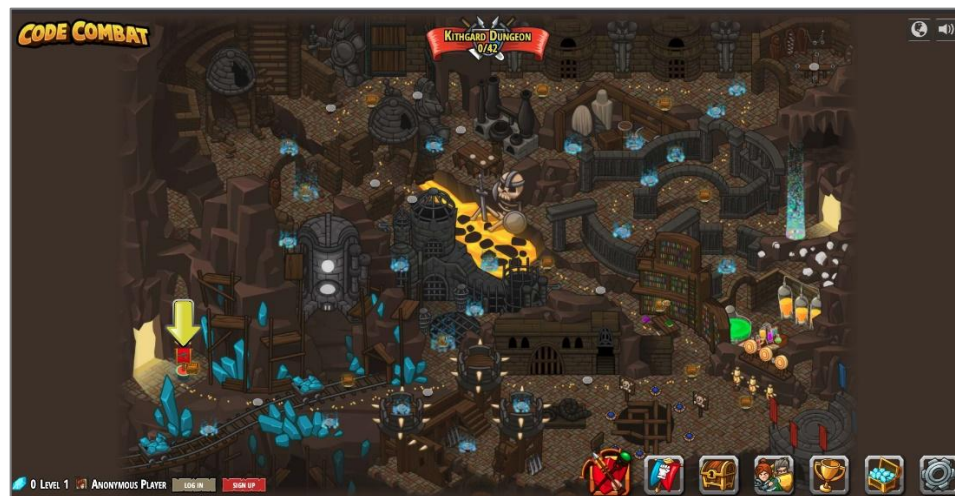
**Figure 2.12** One of coding mission inside CodeCombat

This figure illustrates a coding mission in CodeCombat, where players write Python code to solve a puzzle. The interface shows an in-game scene with a character seeking help to find treasure, alongside a code editor displaying syntax for tasks like finding items, checking conditions, and moving characters.



**Figure 2.13** Characters menu of CodeCombat

This figure shows the character selection menu in CodeCombat, where players choose their avatar for gameplay. The interface displays a character named "Captain Ana Weston," along with options to select a programming language (e.g., JavaScript) and save their choice.



**Figure 2.14** Map exploration inside chosen world

This figure depicts the in-game map exploration screen in CodeCombat, showing a detailed dungeon-like environment with obstacles, pathways, and interactive elements. The interface includes character controls and mission objectives, allowing players to navigate and solve coding challenges within the chosen world.

### 2.6.3 Human Resource Machine

Human Resource Machine (HRM) is a puzzle-based game that introduces players to algorithmic thinking and basic assembly-level logic using a simplified command set (Heithausen, 2020). Players write programs to move items between input, output conveyors and memory cells, simulating real-world computational tasks.

The platform's strengths are centered on reinforcing essential skills for beginner programmers, such as logical reasoning, debugging, and understanding step-by-step execution. It effectively encourages trial-and-error learning, which supports a deep conceptual understanding, and its simple interface allows for quick immersion without the distraction of visual complexity.

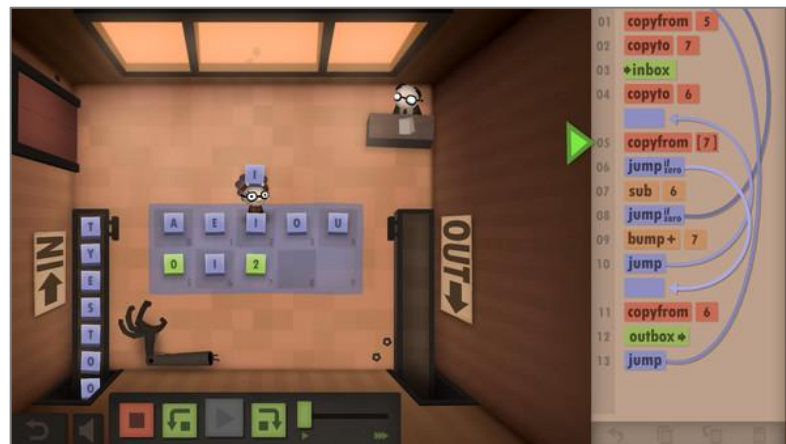
However, it has notable limitations, including a lack of real-code integration that makes it less suitable for students who aim to learn standard programming languages like C++. Additionally, its minimal narrative and visual immersion may reduce long-term engagement, and the platform does not scale well to cover advanced topics or more complex programming structures.

Despite its simplicity, HRM has been praised for its ability to introduce core programming logic in an accessible way (Heithausen, 2020). It supports self-paced learning and encourages optimization through score tracking (e.g., minimizing steps or commands), which motivates players to refine their solutions. Figure 2.15, 2.16, 2.17, and 2.18 shows the great game and world design of for a coding-style game implementation.



**Figure 2.15** Human Resource Machine storyline and dialog system

This figure illustrates the storyline and dialog system in *Human Resource Machine*. It shows a character speaking with a speech bubble that reads, "I have your photo here. Which one is yours?" Below the dialogue, four framed portraits are displayed, suggesting a choice-based interaction or puzzle element central to the game's narrative and gameplay mechanics.



**Figure 2.16** Level design of Human Resource Machine stage

This figure depicts the grid-based level design in *Human Resource Machine*, showing a workspace with numbered tiles and commands like `copy_from` and `jump`, illustrating the game's procedural programming mechanics.





**Figure 2.17** Mission challenges of Human Resource Machine

This figure illustrates a mission challenge in *Human Resource Machine*, displaying a task titled "Sta Challenge" with objectives like using 10 or fewer commands and completing the task quickly. The interface includes options to return or continue, highlighting the game's focus on efficiency and problem-solving.



**Figure 2.18** Levels progression of Human Resource Machine

This figure shows the levels progression map in *Human Resource Machine*, featuring a network of interconnected stages such as "Inventory Report," "String Reverse," and "Cumulative Countdown." The layout illustrates the structured, branching progression system that guides players through increasingly complex programming challenges.

## 2.6.4 Features Comparison of Related Game-Based Learning Games

**Table 2.3** Comparison to other Related GBL Games

Games / Features	Colobot: Gold Edition	CodeCombat	Human Resource Machine	Code[X]:Nexus (Own Designed Project)
<b>Overview</b>	A 3D sci-fi strategy game where players program robots to solve puzzles and complete missions. Available for download on multiple platforms.	An online browser-based coding game that teaches programming through interactive challenges and quests. Focuses on JavaScript, Python, and other languages.	A visual programming puzzle game set in a corporate office environment. Players use assembly-like commands to automate tasks involving moving data between inbox/outbox and memory spaces.	Interactive, narrative-driven 2D top-down adventure game teaching introductory C++ concepts using Godot Engine v4.4. Deployed on Windows and Web (HTML5).
<b>Characteristics</b>	<ul style="list-style-type: none"> <li>- Strategy/Adventure genre</li> <li>- Medium play duration</li> <li>- Continuity across missions</li> <li>- Programming-based gameplay</li> <li>- Story-driven narrative</li> </ul>	<ul style="list-style-type: none"> <li>- Puzzle/Adventure genre</li> <li>- Short to medium play duration</li> <li>- Continuity across levels</li> <li>- Code editor integration</li> <li>- Quest-driven narrative</li> </ul>	<ul style="list-style-type: none"> <li>- Puzzle genre</li> <li>- Short play duration</li> <li>- Independent puzzles</li> <li>- Assembly language metaphor</li> <li>- Minimal narrative</li> </ul>	<ul style="list-style-type: none"> <li>- Adventure genre</li> <li>- Long play duration</li> <li>- Continuity across levels</li> <li>- Puzzle-based</li> <li>- Dialogue-driven narrative</li> </ul>
<b>Goal</b>	Players program robot units using CBOT, a C++ like language to solve complex problems and progress through missions in a futuristic world.	Players learn programming by writing code to control characters and solve challenges in a fantasy setting.	Automate repetitive office tasks using a simple instruction set to solve puzzles based on assembly language principles.	Players learn and reinforce C++ fundamentals by solving puzzles, completing quests, and interacting with NPCs.
<b>Game Dimension</b>	3D	2D	2D	2D
<b>Game Engine</b>	Colobot Engine (Custom)	Custom In-House Browser-Based Engine	Tomorrow Corporation Custom Engine	Godot Engine v4.4
<b>Features</b>	<ul style="list-style-type: none"> <li>- Single-player/Multiplayer</li> <li>- Robotic programming interface</li> <li>- Mission-based progression</li> <li>- Storyline-driven objectives</li> <li>- Teamwork mechanics (in multiplayer)</li> </ul>	<ul style="list-style-type: none"> <li>- Single-player/Multiplayer</li> <li>- Real-time code editor</li> <li>- Quest-driven progression</li> <li>- Level-based challenges</li> <li>- Collaborative features (in multiplayer)</li> </ul>	<ul style="list-style-type: none"> <li>- Single-player</li> <li>- Drag-and-drop command interface</li> <li>- Optimization challenges</li> <li>- Step-by-step debugging</li> <li>- Assembly language metaphor</li> </ul>	<ul style="list-style-type: none"> <li>- Single-player</li> <li>- Narrative-driven storyline</li> <li>- Mini-code editors for simple syntax practice</li> <li>- Progression system based on C++ topics</li> <li>- Visual storytelling elements</li> </ul>
<b>Accessibility</b>	Available for download on Windows, macOS, Linux, and Android.	Browser-based, accessible via any device with internet access.	Available on PC, Mac, Linux, iOS, Android, and Nintendo Switch.	Deployable on Windows and Web (HTML5), ensuring accessibility across devices.
<b>Suitability</b>	Students and hobbyists interested in robotics and programming.	Learners of all ages who want to learn programming in a fun way.	Beginner to intermediate programmers interested in low-level computing concepts.	First-year diploma Computer Science students struggling with traditional teaching methods.
<b>Methodology Used</b>	<ul style="list-style-type: none"> <li>- Programming Challenges</li> <li>- Mission-Based Objectives</li> <li>- Strategic Planning</li> <li>- Robotic Simulation</li> </ul>	<ul style="list-style-type: none"> <li>- Interactive Coding Challenges</li> <li>- Immediate Feedback</li> <li>- Quest-Driven Learning</li> <li>- Collaborative Play</li> </ul>	<ul style="list-style-type: none"> <li>- Visual Programming</li> <li>- Assembly Language Metaphor</li> <li>- Puzzle Solving</li> <li>- Optimization Challenges</li> </ul>	<ul style="list-style-type: none"> <li>- Puzzle-Based Learning</li> <li>- Level-Based Progression</li> <li>- Question &amp; Feedback System</li> <li>- Narrative Integration</li> <li>- Visual Learning Aids</li> </ul>
<b>Evaluation Model</b>	Kearney and Pivec's Educational Game Framework	Serious Game Constructivist Framework	Constructivist Learning Framework	MEEGA+ (Multidimensional Evaluation of Educational Games Plus)

Table 2.3 briefly compares key features of *Colobot*, *CodeCombat*, *Human Resource Machine*, and *Code[X]:Nexus*, focusing on gameplay, programming language, accessibility, target audience, and evaluation approach.

## 2.7 Highlight the Chosen Techniques & Features with Justification

The design of this proposed project is based on research-backed Game-Based Learning (GBL) techniques and features that have been shown to enhance student engagement, motivation and learning outcomes, particularly in introductory programming education (Qian & Clark, 2019; van Gaalen et al., 2021). Based on findings from related educational games such as Cobot: Gold Edition, CodeCombat and Human Resource Machine, one core technique and three key features were selected for implementation:

### 1) Chosen Techniques

- Narrative and Storytelling

**Justification:** Research shows that narrative elements increase emotional investment, contextual understanding, and memory recall in students (Kucher, 2021). In programming education, stories help frame abstract concepts within relatable scenarios, improving motivation and long-term interest (Cheong et al., 2020).

**Relevance:** A strong storyline will immerse players in a meaningful world where solving programming challenges is essential to progress. This approach builds on the success of *Colobot: Gold Edition*, which uses a science fiction narrative to teach C++ inspired logic (Alves & Letouze, 2018).



**Figure 2.19** Examples of games using own unique system and cutscene to deliver the storylines.



By this method as shown in figure 2.19, highlighting visual and dialogue can enhance the storytelling while providing engaging narratives experiences to students.

## 2) Chosen Features

- Narrative Driven Gameplay

**Justification:** Emotional engagement is a powerful driver of learning. Games that integrate storylines see higher retention and motivation compared to mechanics-only designs (Kucher, 2021). Colobot: Gold Edition demonstrates that combining storytelling with programming tasks increases student interest and self-directed exploration (Heithausen, 2020).

**Relevance:** The narrative will provide context and purpose to programming tasks, helping students understand the importance and practical application of what they're learning.



**Figure 2.20** Examples of games using dialogue for NPC to progress the story.

Figure 2.20 shows characters interactions help a lot in guiding and engaging the players especially if being directly integrated to the gameplay.



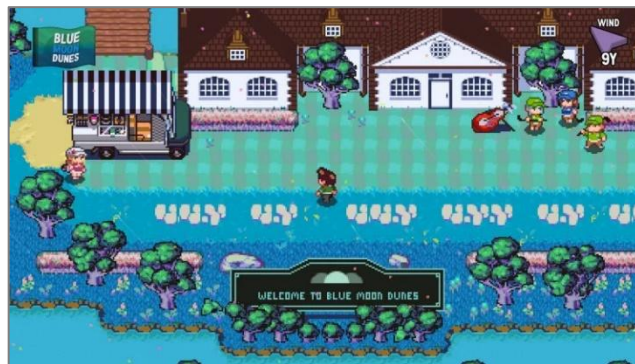
**Figure 2.21** Examples of direct narration to lead players gameplay direction

When effectively being combined with storytelling, direct narration delivers seamless gameplay mechanics in guiding players while maintaining immersion as shown in figure 2.21.

- 2D Top-Down Adventure Game Style

**Justification:** The 2D top-down adventure genre supports structured exploration and puzzle-solving while maintaining visual simplicity, making it ideal for educational purposes (Prensky, 2001). It allows for navigation, clear feedback and manageable complexity is important factors when teaching beginners.

**Relevance:** This genre aligns well with the goal of integrating narrative, challenge, and progression into a cohesive experience tailored for first-year diploma Computer Science students.



**Figure 2.22** Examples of games using 2D top-down pixel art style genre

Figure 2.22 demonstrate visual simplicity from this game genre style is effective for structured gameplay, narrative depth and educational game designed for beginner learners.



**Figure 2.23** Examples of games using the freedom of art style creation.

Figure 2.23 shows by having the right art style direction easily transform the game concept into captivating and memorable visual experience, this will also support game's mechanics, narratives and theme goals.

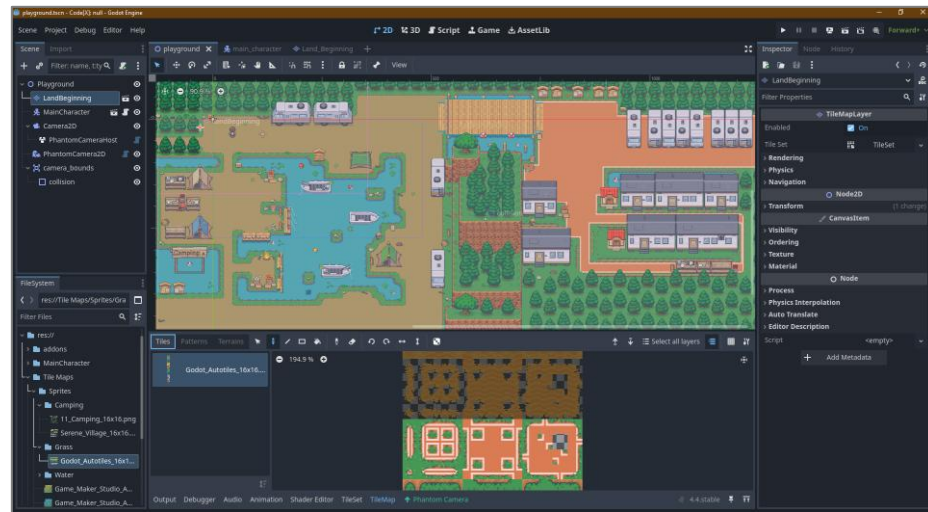
- Godot 4 Game Engine

**Justification:** Godot Engine offers a lightweight, open-source platform suitable for creating accessible educational games. Its ease of use, cross-platform support, and flexibility make it ideal for projects targeting educational institutions with limited resources (Dobroskok et al., 2022).

**Relevance:** Using Godot enables rapid development and modifiability, supporting future enhancements or adaptations by educators. Figure 2.24, 2.25 and 2.26 shows what information and capabilities of Godot Engine.

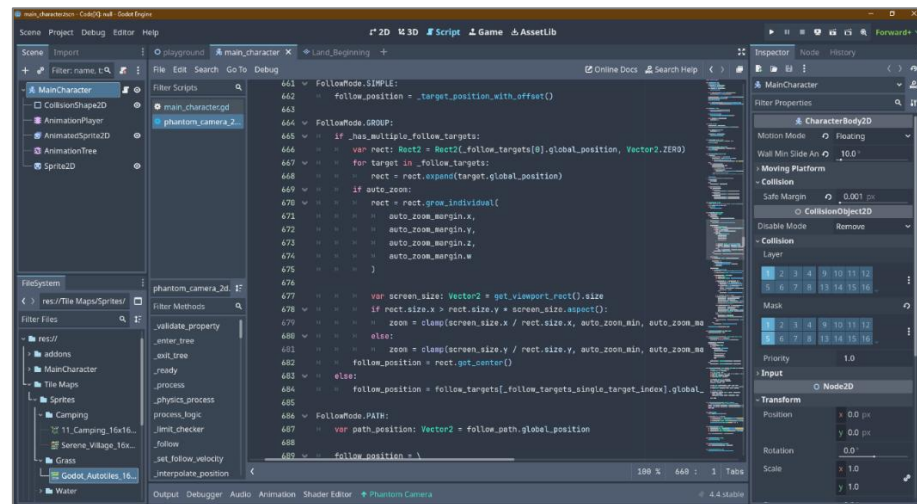


**Figure 2.24** Godot 4 Game Engine Logo



**Figure 2.25** Godot 4 Game Engine Scene Editor Interface

In figure 2.25, the Godot 4 Engine Scene editor is a visual tool used for designing and assembling game scenes, allowing placing nodes in a hierarchical structure, configure their properties, and interactively preview the scene layout and behavior based on developer's creativity.



**Figure 2.26** Godot 4 Game Engine Code Editor Interface.

In figure 2.26, this is where all mechanics like player movement, collision, animation and the core of game elements implementation will be done inside of here. Godot uses its own programming language which is the GDScript, a high-level, object-oriented, imperative and gradually typed programming language built specially for Godot Engine.

## 2.8 Summary

This literature review focuses the need for more engaging methods in Computer Science education, particularly for first-year diploma students who often struggle with abstract programming concepts. Traditional teaching methods are typically passive and struggle to sustain interest in topics like algorithms and data structures.

Game-Based Learning (GBL) offers a compelling alternative by incorporating elements like progression systems, challenges, and storytelling to promote active learning and motivation. Educational games such as Colobot: Gold Edition, CodeCombat and Human Resource Machine demonstrate the potential of GBL, though few effectively combine all key elements.

To fill this gap, the proposed project introduces a narrative-driven 2D top-down adventure game built with Godot 4 Game Engine. It uses question-based puzzles, mastery-driven progression, and narrative storytelling to support self-paced learning of introductory C++ programming.

By blending storytelling, gameplay and interactivity, the proposed project aims to improve the accessibility and effectiveness of programming education for the first-year diploma students in Computer Science education.

## **CHAPTER 3**

### **METHODOLOGY**

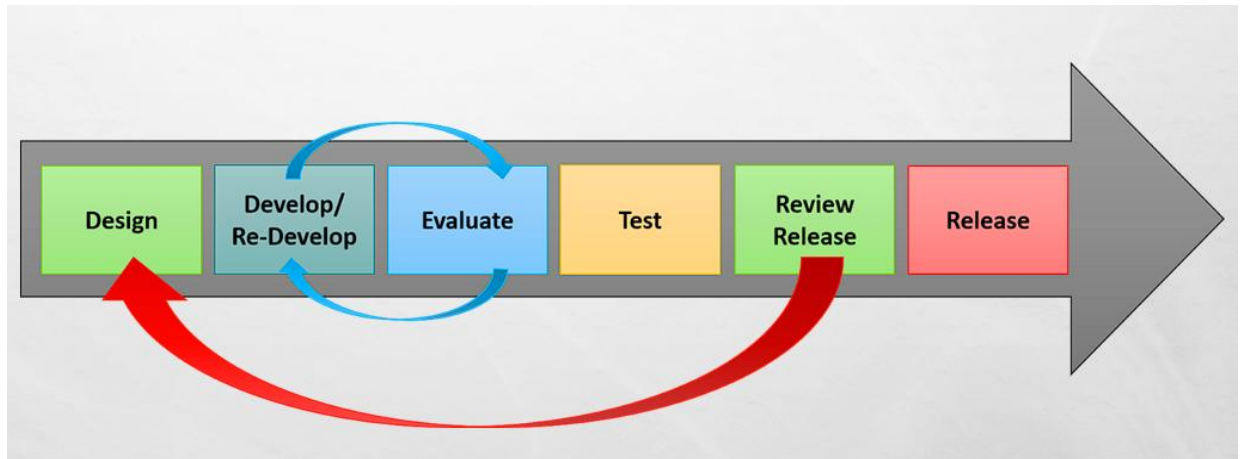
#### **3.1 Introduction**

This chapter presents the methodology adopted in the design and development of proposed project. The methodology follows a structured Game Development Life Cycle (GDLC) approach, ensuring that each phase contributes directly to achieving the project's core objectives.

Each phase of the GDLC is aligned with educational goals, ensuring that gameplay mechanics such as puzzles, progression systems and storytelling are not only entertaining but also pedagogically sound and technically feasible. This section provides a concise overview of how the game was conceptualized, designed, implemented, tested and prepared for release, while emphasizing the use of the Godot Engine v4.4 as the development platform. The subsequent sections will elaborate on the specific phases of the GDLC, system architecture and technical requirements.

#### **3.2 Project Methodology**

The development of followed a Game Development Life Cycle (GDLC) framework to ensure alignment with pedagogical goals and technical feasibility. This structured approach enabled systematic progression from conceptualization to deployment, ensuring the game meets the needs of first-year Computer Science students. Figure 3.1 shows the project methodology of the game development life cycle.



**Figure 3.1** Game Development Life Cycle

The project was divided into six iterative phases, as outlined in Table 3.1. Each phase was tailored to address specific project objectives:

**Table 3.1** Project Methodology (GDLC)

Phases	Activities	Technique/Software	Deliverable
Design	<ul style="list-style-type: none"> <li>• Create narrative-driven storyline</li> <li>• Map C++ concepts to gameplay puzzles</li> <li>• Plan UI/UX elements</li> </ul>	<ul style="list-style-type: none"> <li>• Canva and draw.io</li> </ul>	<ul style="list-style-type: none"> <li>• Narrative design document</li> <li>• Scene layout and UI/UX diagrams</li> </ul>
Develop / Re-Develop	<ul style="list-style-type: none"> <li>• Build game scenes in Godot Engine</li> <li>• Script puzzles using GDScript</li> <li>• Create assets (sprites, sounds)</li> </ul>	<ul style="list-style-type: none"> <li>• Godot Engine v4.4</li> <li>• GDScript</li> <li>• Aseprite / Piskel</li> <li>• Audacity / Bfxr</li> <li>• Git</li> </ul>	<ul style="list-style-type: none"> <li>• Functional game prototype</li> </ul>

	<ul style="list-style-type: none"> <li>• Test basic functionality</li> </ul>		
Evaluate	<ul style="list-style-type: none"> <li>• Review puzzle accuracy</li> <li>• Validate with instructors</li> <li>• Cross-check against curriculum</li> </ul>	<ul style="list-style-type: none"> <li>• Excel / Google Sheets</li> <li>• Feedback forms</li> </ul>	<ul style="list-style-type: none"> <li>• Revised learning content</li> </ul>
Test	<ul style="list-style-type: none"> <li>• Conduct alpha testing</li> <li>• Collect player feedback</li> <li>• Identify bugs</li> <li>• Measure cognitive load</li> </ul>	<ul style="list-style-type: none"> <li>• Google Forms</li> <li>• Trello / Notion</li> <li>• Playtesting sessions</li> </ul>	<ul style="list-style-type: none"> <li>• Player feedback summary</li> </ul>
Review Release	<ul style="list-style-type: none"> <li>• Finalize graphics and audio</li> <li>• Optimize performance</li> <li>• Prepare export templates</li> <li>• Write instructions manual.</li> </ul>	<ul style="list-style-type: none"> <li>• Godot Export</li> <li>• Performance profiling tools</li> <li>• Documentation editors</li> </ul>	<ul style="list-style-type: none"> <li>• Stable build for deployment</li> </ul>
Release	<ul style="list-style-type: none"> <li>• Publish on Itch.io</li> <li>• Distribute to target users</li> <li>• Gather usage data</li> <li>• Monitor feedback</li> </ul>	<ul style="list-style-type: none"> <li>• Web hosting (Itch.io Page)</li> <li>• Google Forms</li> </ul>	<ul style="list-style-type: none"> <li>• Deployed finished game version</li> </ul>

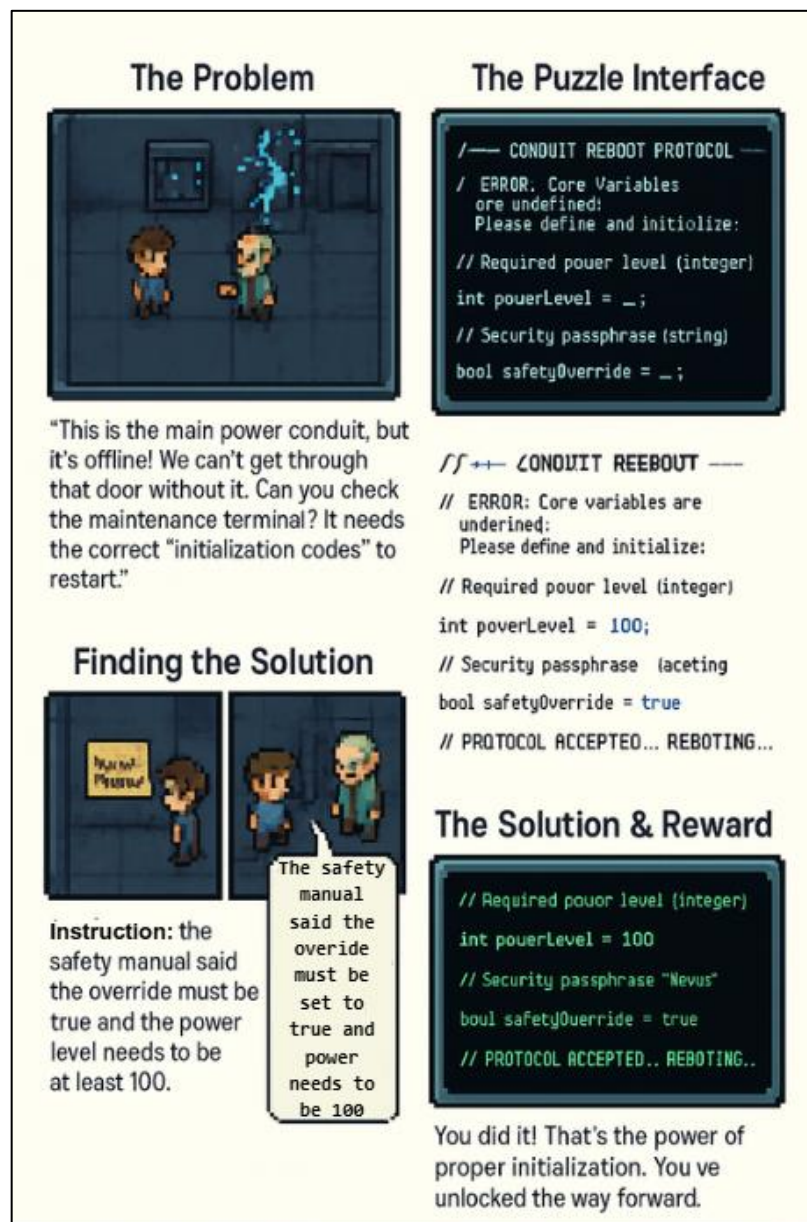


### 3.2.1 Design

This phase focused on identifying effective Game-Based Learning (GBL) methods. By integrating narrative storytelling and puzzle-based challenges, the project aimed to enhance student engagement and comprehension. Drawing from recent research on GBL's impact on motivation (Kroustalli & Xinogalos, 2021), the narrative was designed to contextualize C++ concepts (e.g., variables, loops) within a cohesive storyline. To achieve this, several key design artifacts were produced, including storyboard drafts, concept art, and dialogue scripts, ensuring that all gameplay mechanics align with pedagogical goals to foster a structured yet interactive learning environment.

#### A. Storyboard and Narrative Flow

Storyboarding was used to visualize the sequence of events, player interactions, and the integration of learning puzzles within the narrative. A key storyboard sequence, for example, outlines the player's quest, where they encounter a broken "transformer" and must use their knowledge of C++ variables to repair it. This process ensures the narrative pacing and puzzle difficulty are balanced.



**Figure 3.2** Storyboard Draft for the "Conduit's Variable" Quest

From the figure 3.2, this storyboard sequence visualizes the "Conduit's Variable" quest, which is designed to teach the fundamental C++ concepts of variable declaration and initialization. It depicts the narrative flow where the player is presented with a problem, gathers information from the environment and NPCs, and applies their knowledge to solve a puzzle. This process directly integrates learning (int, string, bool data types) with gameplay, providing immediate feedback and in-game progression as a reward.

## B. Concept Art and World Building

To establish the visual identity and atmosphere of “Code[X]:Nexus”, concept art was developed for the main character and the game world. The 2D top-down pixel art style was chosen for its clarity and low-performance overhead, ensuring accessibility. The character design aims to be relatable for students, while the world design combines elements of a futuristic tech environment with classical adventure game tropes.



**Figure 3.3** Concept Art for the Player Character and a Game World Area

Figure 3.3 shows the player character sprite with 23 animation frames (left) and a concept mockup of "Vector Valley," the starting game area (right).

## C. Dialogue and Narrative Integration

Dialogue is a critical component for delivering the narrative and framing the C++ challenges. Scripts were written to be engaging while subtly embedding educational content. The dialogue not only guides the player through quests but also serves as a hint system and provides contextual feedback for the puzzles. Below is a sample interaction where an NPC introduces the concept of for loops.

**Table 3.2** Concept of the designed project quest’s dialogues.

Character	Dialogue
Professor Ada	“Greetings, young lads. This data stream is fragmented. We need to process every packet, one by one.”
Player	"How can I do that?"
Professor Ada	"Think in cycles. A for loop would be perfect. It can initialize a counter, set a condition to stop, and increment through each packet. Can you build the loop structure to fix it?"

### **3.2.2 Develop / Re-Develop**

This phase addressed developing a 2D top-down adventure game using the Godot Engine v4.4. Leveraging Godot's node-based architecture and GDScript, the game's core mechanics includes character movement, quest triggers and puzzle logic were implemented. Assets like sprites and audio were created using tools like Aseprite and Audacity, ensuring the game met technical specifications (e.g., low system requirements) for accessibility in educational settings (Dobroskok et al., 2022). Iterative development allowed for refinement of gameplay elements, such as interactive dialogue systems and level transitions, to align with the project's narrative-driven vision.

### **3.2.3 Evaluate**

This phase aimed to assess the game's effectiveness. It involved cross-checking puzzle accuracy against C++ curriculum standards and validating content with instructors. For example, puzzles on variables and conditionals were reviewed to ensure they accurately reflected foundational programming concepts, ensuring pedagogical alignment (Holly et al., 2024). Feedback from educators was used to refine learning modules, directly supporting the objective of evaluating the game's instructional value.

#### **A. Evaluation Participants and Activities**

The evaluation will involve two key participant groups: Subject Matter Experts (SMEs) and the project supervisor. The SMEs, comprising a lecturer from the Computer Science faculty who specialize in teaching introductory C++ programming, will play a critical role in assessing and validating the pedagogical accuracy and effectiveness of the game's content. Their expertise ensures that the educational material aligns with curriculum standards and learning objectives. Additionally, the project supervisor will oversee the evaluation process, providing guidance and conducting a final review to confirm the validity and reliability of the findings before finalizing the assessment outcomes. During

the evaluation session, Subject Matter Experts (SMEs) will engage in a guided walkthrough of the game prototype, during which they will critically review elements such as in-game puzzles, dialogue, and instructional text to assess their pedagogical relevance, accuracy, and alignment with introductory C++ programming concepts.

#### **B. Evaluation Criteria**

The game's content will be evaluated using criteria grounded in pedagogical best practices and the project's objectives, adapted from models like MEEGA+ (Multidimensional Evaluation of Educational Games Plus). Subject Matter Experts (SMEs) will assess four key areas, Content Accuracy, Pedagogical Alignment, Clarity and Comprehensibility, and Engagement Potential. Feedback from this phase will inform refinements to learning modules and puzzle designs, ensuring the game is both educationally effective and engaging before student testing begins.

### **3.2.4 Test**

This phase involves alpha testing to evaluate the game's usability, engagement, and preliminary learning outcomes. A small group of 15 first-year Computer Science diploma students will participate in structured playtesting sessions. The evaluation uses a mixed-methods approach: direct observation of player behavior, a post-session survey (via Google Forms) with scales and open-ended questions to assess usability, engagement, and comprehension, and short informal interviews for qualitative insights. Feedback and identified bugs are logged and prioritized for iteration. This combination of observation, survey, and interview serves as the primary mechanism to evaluate engagement.

### **3.2.5 Review & Release Preparation**

This phase covers the final review and preparation activities required to create a polished and stable build of the game for PC deployment. Before creating the final build, a comprehensive internal review is conducted, which includes a code review for efficiency, a final asset review for all graphics and audio, a functional review to test all game mechanics, and a documentation review for the user guide. For release, the game is prepared for the PC (Windows) platform. This involves using Godot's export tools to generate a standalone executable file (.exe) and packaging all necessary assets into a single .zip file suitable for distribution on platforms like Itch.io. A key activity during this phase is performance optimization; performance profiling tools are used to reduce load times and ensure the game is accessible on low-end devices, which is critical for classroom use. As the release is focused on PC, requirements for other platforms like the Google Play Store are not applicable. The detailed minimum and recommended hardware specifications required to run the game are formally outlined later in this document in Section 3.5.

### **3.2.6 Release**

This phase enabled gathering real-world data on effectiveness. By publishing the game on platforms like Itch.io Page and distributing it to students, the project collected initial user feedback to inform future iterations. Metrics like playtime, completion rates and post-game surveys provided insights into the game's impact on student motivation and learning outcomes (Kroustalli & Xinogalos, 2021).

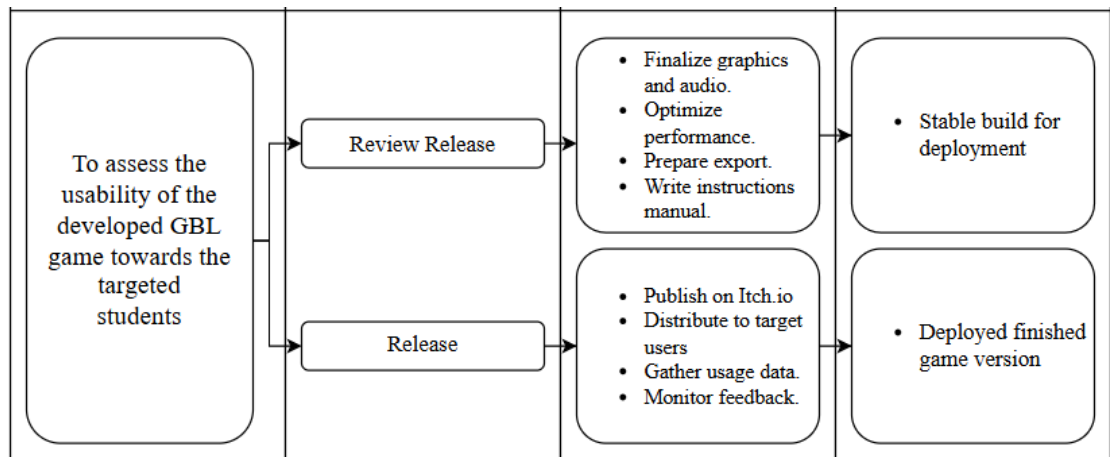
Together, the GDLC phases created a systematic development process that directly addressed each project objective, from conceptualizing GBL methods to deploying a functional, engaging educational tool.

### 3.3 Summary of Project Methodology

This section presents a summary of how each phase in the Game Development Life Cycle (GDLC) aligns with the specific objectives of the proposed project, ensuring that every stage of development directly supports the educational and technical goals outlined in Chapter 1 as shown in the table 3.3 for the project methodology summary.

**Table 3.3** Summary of Project Methodology

Objective	Phases	Task	Deliverables
To identify an effective game-based learning (GBL) method in programming education in enhance student motivation	Design	<ul style="list-style-type: none"> <li>• Create narrative-driven storyline.</li> <li>• Map C++ concepts to gameplay puzzles.</li> <li>• Plan UI/UX elements.</li> </ul>	<ul style="list-style-type: none"> <li>• Narrative design document</li> <li>• Scene layout and UI/UX diagrams</li> </ul>
To develop a 2D top-down style learning adventure game, in learning programming concepts that includes GBL elements using Godot Engine v4.4	Develop / Re-Develop	<ul style="list-style-type: none"> <li>• Create narrative-driven storyline.</li> <li>• Map C++ concepts to gameplay puzzles.</li> <li>• Plan UI/UX elements.</li> </ul>	<ul style="list-style-type: none"> <li>• Create narrative-driven storyline.</li> <li>• Map C++ concepts to gameplay puzzles.</li> <li>• Plan UI/UX elements.</li> </ul>
	Evaluate	<ul style="list-style-type: none"> <li>• Review puzzle accuracy.</li> <li>• Validate with instructors.</li> <li>• Cross-check against curriculum.</li> </ul>	<ul style="list-style-type: none"> <li>• Revised learning content</li> </ul>
	Test	<ul style="list-style-type: none"> <li>• Conduct testing.</li> <li>• Collect player feedback.</li> <li>• Identify bugs</li> <li>• Measure cognitive load.</li> </ul>	<ul style="list-style-type: none"> <li>• Player feedback summary</li> </ul>



As the summary illustrates, the project methodology provides a clear and structured workflow. Each project objective is systematically mapped to distinct development phases (from Design to Release) and tied to concrete tasks and deliverables. This methodical alignment ensures that every development effort is purposefully directed toward solving the core challenge of student disengagement. By following this approach, the project ensures the final product is not merely a functional game, but a cohesive, narrative-driven educational experience designed to improve learning outcomes for introductory C++ students.



### 3.4 System Architecture

The system architecture of proposed project was designed to ensure a modular, scalable and pedagogically effective structure that supports both gameplay mechanics and educational content delivery. The game will be developed using the Godot Engine v4.4, leveraging its node-based scene system and lightweight performance for accessibility on low-end systems targeting deployment on Windows PC and HTML5 Web Browser.

The game follows a Node-Based Scene Architecture, a core feature of the Godot Engine. In this style, each game component (e.g., player, NPC, puzzle) is represented as a node within a hierarchical scene graph. Nodes encapsulate specific behaviors (e.g., movement, dialogue) and can be reused across levels, promoting code reusability and scalability (Dobroskok et al., 2022). This is the native architecture style used in Godot, where each scene contains a hierarchy of nodes that handle specific roles such as player movement, dialogue display, puzzle logic, UI elements, and progress tracking. Each scene can be treated as a self-contained module, which aligns with best practices in software design and supports reusability and scalability.

This architecture style offers several key functionalities: modular scenes can be reused across different levels, making it easy to add new puzzles or narrative segments without disrupting existing code. Additionally, there is a clear separation between gameplay, UI, and logic layers, which enhances maintainability and scalability. The design is also lightweight and efficient, ensuring compatibility with low-end systems.

#### A. User Interface Layer

This layer encompasses all the visual elements the player interacts with, including menus, the Head-Up Display (HUD), dialogue boxes, and the puzzle interface. It is built using Godot's native Control nodes and CanvasItem system. The primary goal of this layer is to ensure clarity and accessibility, promoting an intuitive interaction model that is easy for first-year students to grasp and use effectively.

## **B. Game Logic Layer**

The Game Logic Layer is the core engine of the gameplay experience, responsible for managing player input, character movement, the triggering of quests, and transitions between levels. It utilizes GDScript, a Python-like language chosen for its simplicity and readability, to script all gameplay events. This layer also includes state machines to govern NPC behavior and validate puzzle solutions, ensuring the game world is dynamic and responsive to player actions.

## **C. Learning Content Layer**

This layer directly integrates the educational objectives into the game by containing embedded C++ programming puzzles that are mapped to specific curriculum topics. Each puzzle is designed to represent a fundamental concept such as variables, loops, conditionals, or functions, thereby reinforcing theoretical knowledge through direct, practical application. The design is inspired by games like "Human Resource Machine," where players solve algorithmic problems through visual interaction to foster computational thinking.

## **D. Narrative Integration Layer**

To enhance engagement and provide a contextual framework for learning, the Narrative Integration Layer weaves the educational content into a compelling story. It uses story-driven missions and branching dialogue trees to guide the player's progression through the game. This approach is inspired by successful educational games that leverage narrative to motivate learners and create a cohesive, meaningful experience.

## **E. Data Management Layer**

This layer is responsible for the persistence of player information, which includes tracking progress, saving game states, and logging performance data. This functionality enables personalized learning paths and allows for effective progress monitoring. It uses a simple file-based storage system, which eliminates the need

for an external database, thereby simplifying deployment and reducing system overhead. This also provides players with the convenience of resuming gameplay from their last completed level.

High-Level System Architecture Diagram (Descriptive)

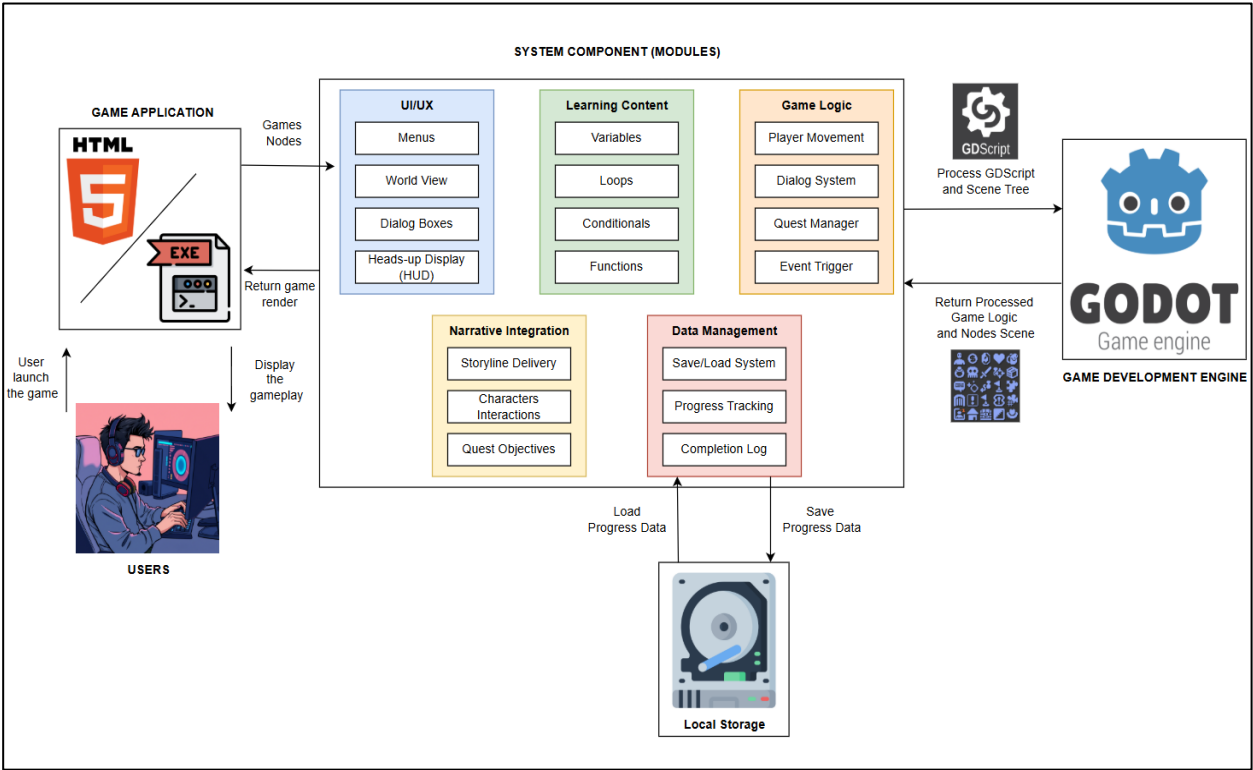
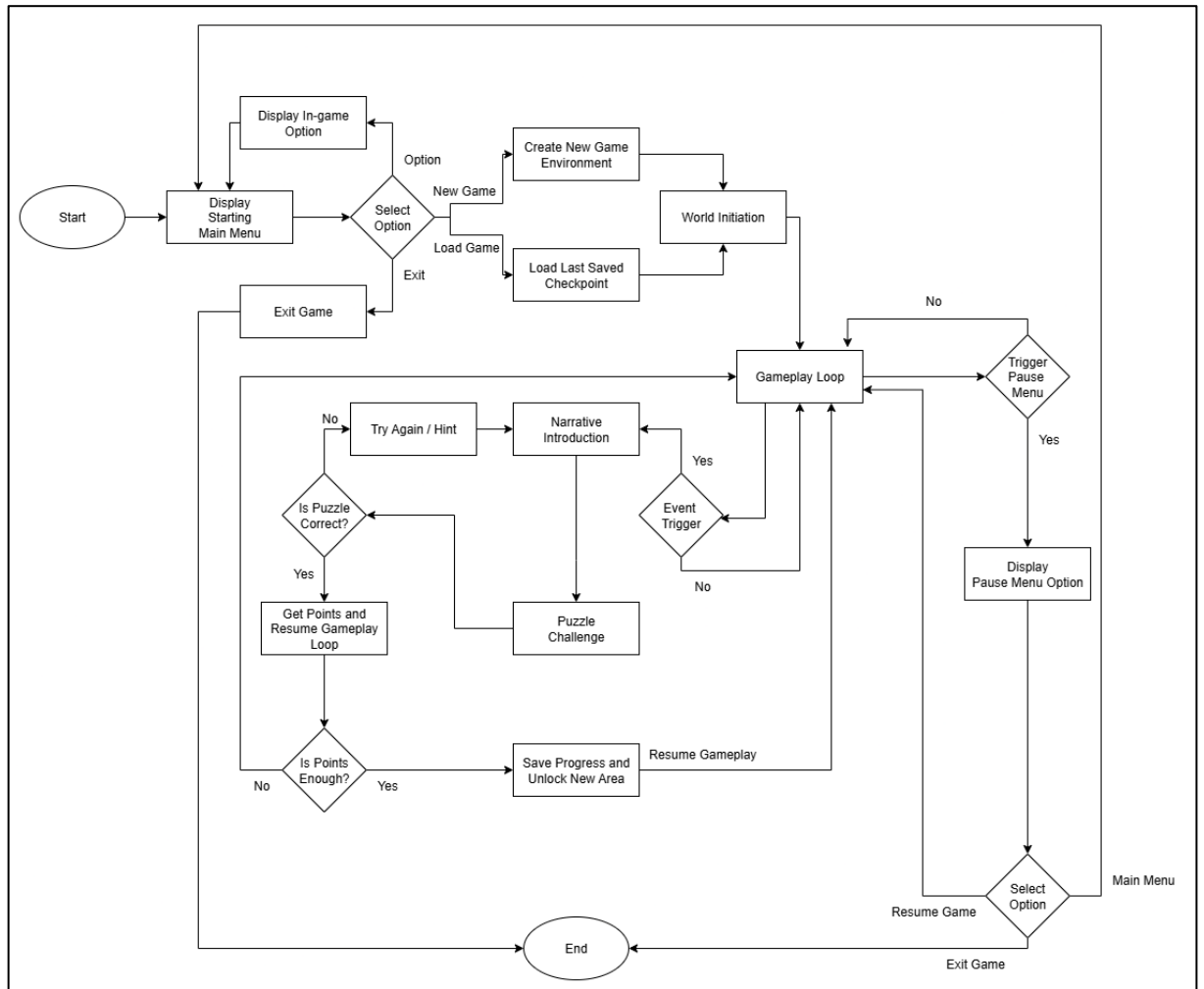


Figure 3.4 System Architecture Design of Proposed Project

Based on Figure 3.4, the System Architecture Design for “Code[X]: Nexus” illustrates a modular structure centered around the Godot Game Engine. Users launch the Game Application, which processes game nodes and returns rendered gameplay. The core System Components are divided into: UI/UX, Learning Content, Game Logic, Narrative Integration, and Data Management. All game data is persistently stored and retrieved from Local Storage, ensuring player progress is maintained. The Godot Engine processes GDScript and scene trees, returning processed game logic and scenes to the application for display, thus orchestrating the entire game experience.

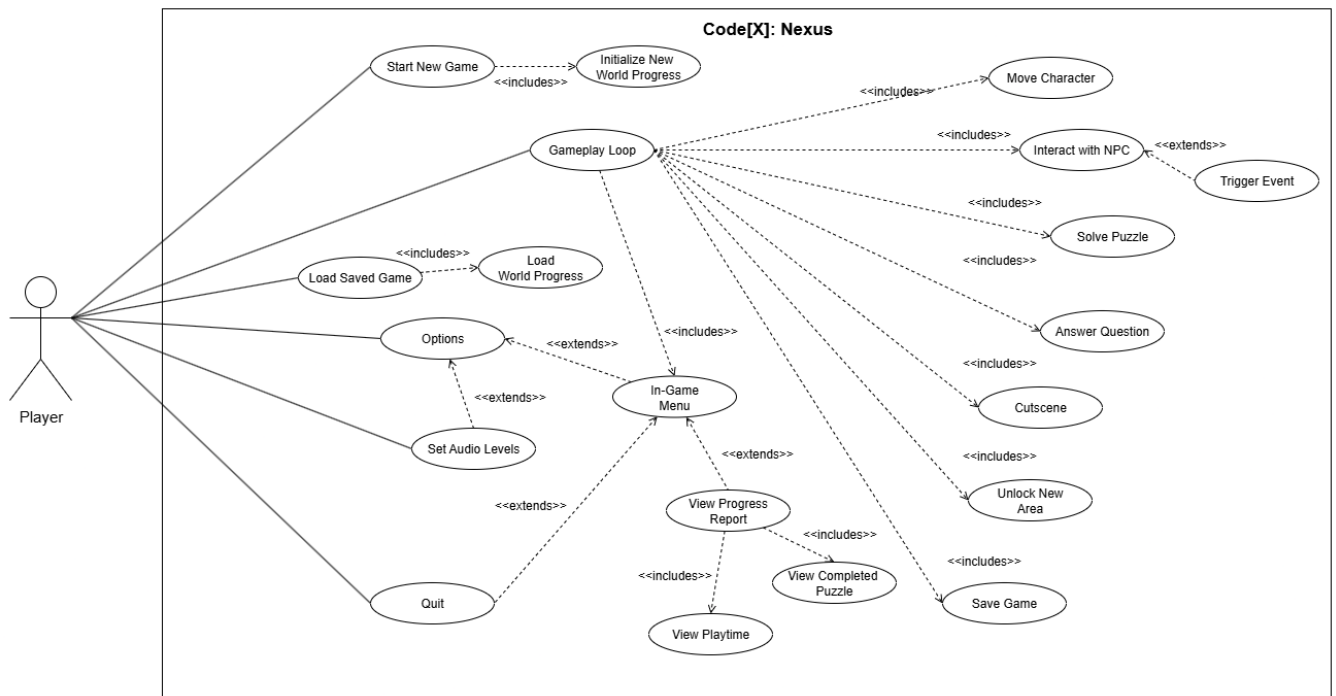
## Flowchart Design Diagram



**Figure 3.5** Flowchart diagram for the proposed project

Based on Figure 3.5, the operational flow of “Code[X]: Nexus” begins at the start state, presenting the player with a main menu to initiate a new game, load game, access in-game options or exit game. The core gameplay unfolds within the gameplay loop, which can be paused to access menu options for resuming, returning to the main menu or exiting. Within the loop, event triggers lead to narrative introductions and puzzle challenges. Successful puzzle completion awards player with points which will leads in unlocking new area if sufficient points are accumulated. Thus, the game process concludes at the End state if exiting the game.

## Use Case Diagram Design Diagram



**Figure 3.6** Use case diagram of the proposed project

Based on the Figure 3.6, the use case diagram illustrates how the Player interacts with the game system to achieve various goals. The Player can manage their game experience by entering the main menu to start a new game, load a saved one, adjust options like audio levels, or quit. Once in the Gameplay Loop, the Player actively engages with the game world by navigating environments, interacting with NPCs (which may involve dialogue or triggering story events), and experiencing the narrative. Central to the game's purpose, the Player also solves programming challenges, providing code solutions and receiving immediate feedback. As they progress, the Player can unlock new content and track their achievements by viewing their overall progress, including completed challenges and playtime. Essentially, the diagram outlines all the key functionalities the game offers to its sole user, the Player.

### 3.5 Hardware and Software Requirements

To ensure broad accessibility and ease of use, “Code[X]:Nexus” was developed with minimal system requirements in mind, making it suitable for deployment in educational environments where students may be using low-end devices or shared computing resources (Tan et al., 2021). The game is designed to run efficiently on standard classroom computers, personal laptops or tablets. This section outlines both the software tools used during development and the minimum hardware specifications required for end-users to play the game. Table 3.4 and 3.5 shows the required software and hardware requirements needed.

#### Software Requirements:

**Table 3.4** Software Requirements Details

Component	Description
Game Engine	Built with Godot Engine v4.4, an open-source engine for rapid prototyping and cross-platform deployment.
Programming Language	Gameplay logic was implemented in GDScript for its simplicity and readability.
User Interface Design	UI elements such as dialogue boxes, puzzle interfaces and HUD components were created using Godot’s built-in node system.
Asset Creation Tools	<ul style="list-style-type: none"><li>• Piskel and Aseprite for creating pixel art graphics.</li><li>• Bfxr and Audacity for sound effects and music.</li><li>• Trello and Notion for task tracking and documentation.</li></ul>
Version Control	Git and GitHub were used for version control and collaboration during development.
Export Targets	The final build was exported for two platforms: <ul style="list-style-type: none"><li>• Windows PC</li><li>• HTML5 Web Browser</li></ul>

## Hardware Requirements:

**Table 3.5** Hardware Requirements Details

<b>Requirements</b>	<b>Minimum Specification</b>	<b>Recommended Specification</b>
<b>Operating System</b>	Windows 7 SP1 or higher / Web browser (Chrome, Firefox, Edge)	Windows 10 (64-bit) or higher / Modern web browser (latest versions)
<b>Processor</b>	Intel Core i3-2100 or equivalent (AMD A6-Series or equivalent)	Intel Core i5-3570 or better (AMD A8-7410)
<b>RAM</b>	1 GB DDR3 SDRAM	2 GB DDR3 SDRAM
<b>Graphics</b>	Integrated graphics card (e.g., Intel HD Graphics 2000 or AMD Radeon HD 6310)	Dedicated GPU not required; mid-range integrated GPU (e.g., Intel UHD Graphics 630/ Radeon HD 7660G)
<b>Storage</b>	200 MB available space on the primary drive (HDD or SSD)	500 MB available space on the primary drive (SSD preferred)
<b>Input Device</b>	Keyboard and mouse (PS/2 or USB)	Keyboard, mouse or gamepad support (USB)

The minimal requirements reflect the project's focus on accessibility for first-year students, many of whom may use low-end devices in classroom settings (Tan et al., 2021). By supporting both Windows and HTML5, the game can be deployed in labs, libraries, or at home, aligning with the objective of creating an "anytime, anywhere" learning tool (Cheong et al., 2020).

### 3.6 Conclusion

This chapter outlined the methodology used in the development of “Code[X]:Nexus”, a narrative-driven 2D top-down adventure game designed to teach introductory-level C++ programming to first-year diploma students. The development process followed a structured Game Development Life Cycle (GDLC), ensuring that each stage starting from initial design to final release was aligned with both educational goals and technical feasibility.

Each phase of the GDLC contributed directly to the project's core objectives, starting with the Design phase, which established a strong foundation for integrating narrative and learning content. The subsequent Development phase implemented these ideas using the Godot Engine v4.4, leveraging its accessibility for educational purposes. Rigorous Evaluation and Testing phases then ensured the learning modules were accurate, engaging, and appropriately challenging. Finally, the Release phase made the game available to its target audience, enabling valuable feedback collection for future improvements.

By following this systematic approach, the proposed project was developed as an accessible, engaging and educationally meaningful tool that bridges the gap between traditional instruction and interactive learning. The modular system architecture and minimal hardware requirements ensure that the game can be used in a variety of educational environments, including classrooms with limited technological resources.



## References

- Al-Marooof, I. A. J., Al-Emran, M., Al-Shadafan, S., & Al-Ahbabi, S. (2024). A review of game-based learning in computer science education. *International Journal of Information Management Data Insights*, 4(1), 100220.
- Alves, J. R. M., & Letouze, P. (2018). The curriculum integration of a course of “Introduction to Programming Logic” with a serious game – Colobot. *International Journal of Social Science and Humanity*, 8(11), 275–280. <https://doi.org/10.18178/IJSSH.2018.V9.974>
- Arnedo, J., & García Solórzano, D. (2023). Coding is fun: Engaging adult online learners using programming games. In A. García Holgado & F. J. García Peñalvo (Eds.), *Proceedings TEEM 2022* (pp. 606–614). Springer. [http://doi.org/10.1007/978-981-99-0942-1\\_63](http://doi.org/10.1007/978-981-99-0942-1_63)
- Boyle, E. A., Hainey, T., Connolly, T. M., Gray, G., Earp, J., Ott, M., ... & Pereira, J. (2016). An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and serious games. *Computers & Education*, 94, 178–192. <https://doi.org/10.1016/j.compedu.2015.11.003>
- Chan, H. L., Lin, H. C., & Lee, T. H. (2023). Editorial: New educational technology and its impact on student-centered learning. *Frontiers in Psychology*, 14, 1189423.
- Cheong, C., Flippou, F., & France, D. (2020). Game-based learning in computing education: A mapping review. *ACM Computing Surveys*, 53(4), 1–37. <https://doi.org/10.4018/IJGBL.2020010101>

- Choi, W. C., & Choi, I. C. (2024). Investigating the effect of the serious game CodeCombat on cognitive load in Python programming education. In Proceedings of the 2024 IEEE World Engineering Education Conference (EDUNINE) (pp. 1–10). IEEE. <https://doi.org/10.1109/EDUNINE60625.2024.10500551>
- de Oliveira, P. F. F., de Oliveira, A. L. M., & de Souza, W. S. L. (2022). Gamification in programming education: A systematic mapping study. *Journal of the Brazilian Computer Society*, 28(1), 1–28.
- Dobroskok, I., Tan, A., Bekirogullari, Z., Kurucay, M., & Ivanova, T. (2022). Game development software tools in higher educational institutions: Experience of Ukraine, Turkey and Bulgaria. ResearchGate. <https://www.researchgate.net/publication/365574044>
- Giannakoulas, V., & Xinogalos, S. (2024). When Narratives Meet Algorithms: How Story-Based Educational Games Transform Learning into a Computational Adventure. GESS Education.
- Heithausen, C. (2020, February 15). A look at "Human Resource Machine" according to Papert's ideas. Cologne Game Lab, TH Köln – University of Applied Sciences. <https://www.researchgate.net/publication/361112910>
- Holly, S., Tan, K. L., & Lim, W. Y. (2024). Gamification in programming education: A systematic review. *IEEE Transactions on Education*, 67(2), 123–135. <https://doi.org/10.48550/arXiv.2406.03055>
- Kucher, T. (2021). Storytelling and decision-making in educational games: Enhancing memory recall and engagement. *Journal of Educational Technology Development and Exchange*, 14(1), 45–60.

- Kroustalli, C., & Xinogalos, S. (2021). Studying the effects of teaching programming to lower secondary school students with a serious game: A case study with Python and CodeCombat. *Education and Information Technologies*, 26(5), 6069–6095. <https://doi.org/10.1007/s10639-021-10596-y>
- Kalogiannakis, K., Papadakis, S., & Zourmpakis, A. I. (2021). Gamification in science education. A systematic review of the literature. *Education Sciences*, 11(1), 22. <https://doi.org/10.3390/educsci11010022>
- Lozano-Lozano, C., Cárdenas-Robledo, L. A., & Gonzalez-Ramirez, T. (2023). A serious game to learn the OSI model based on experiential learning. *Computer Applications in Engineering Education*, 31(4), 932-953.
- Qian, M., & Clark, K. R. (2019). Game-based learning and 21st-century skills: A review of recent research. *Educational Research Review*, 27, 1–16. <https://doi.org/10.1016/j.chb.2016.05.023>
- Tan, S. L., Neill, S., & D’Souza, D. (2021). Motivation and learning in gamified computer science courses: An exploratory study. *Computers & Education Open*, 2, 100050.
- van Gaalen, A. E. J., Brouwer, J., Schönrock-Adema, J., Bouwkamp-Timmer, T., Jaarsma, A. D. C., & Georgiadis, J. R. (2021). Gamification of health professions education: A systematic review. *Advances in Health Sciences Education*, 26, 683-711. <https://doi.org/10.1007/s10459-020-10000-3>

Zainuddin, Z., Chu, S. K. W., & Shujahat, M. (2020). Mastery learning in gamified classrooms: A systematic literature review. *Education and Information Technologies*, 25(5), 4525–4549.