**NEAR EAST UNİVERSITY**

**FACULTY OF ARTIFICAL INTELLIGENCE AND INFORMATICS**

**DEPARTMENT OF SOFTWARE ENGINEERING**

**COURSE TIMETABLE**

**SUMMER TRAINING I**

**SWE299**

**FADI AL-TURJMAN**

**ARDA TATAR 20230993**

**NICOSIA-2025**

**ABSTRACT**

Computer game developing is a process of creating software applications that process This field encompasses both native development, which targets specific platforms like Android or iOS using platform-specific languages , and cross-platform development, which allows a single code file to operate across multiple systems using frameworks like React Native. Key principles of the mobile application development includes user interfaces and user experiences, performance optimization, security, scalability and integration with backend services. The development process usually involves stages such as planning, desinging, coding, testing,deployment and maintenance.

Learning history is a hard process to complete because people might get bored while doing it. Sometimes people learn something wrong because they get bored, find incorrect information on the internet, or are in a hurry.  
  
 The goal of the project is to teach people while they are having fun, help them study efficiently, and give them correct information about their past. This process was hard to accomplish because combining fun and education in a balanced way is not easy.  
  
 My project is an endless running game that includes historical elements to educate people. While running in the game, players can collect items or reach certain points that give them historical facts or stories from the past of Cyprus. That way, people are not only playing a fun game, but also learning interesting things without getting bored.  
  
 I wanted to create something unique that people had fun meanwhile doing it learning something from the game and game educating people are unique and fun.

Table of Contents

1. Introduction

2. Team Members

3. Game Concept

4. Gameplay and Features

5. History Section

6. Development Tools and Workflow

7. Challenges and Solutions

8. Testing and Feedback

9. Future Improvements

10. Conclusion

11. Screenshots

12. References

2-1 Page Setup

2-2 Overview

2-3 Bibliography

INTRODUCTION

The word "engineer" comes from the Latin word "ingenium", which means being talented, smart, and skilled. A good engineer should have these qualities in order to do well in their job and life. But having talent alone is not enough to become an engineer. A person also needs to complete an engineering program at a good school or university.

At the end of their studies, and before graduation, engineering students must show what they have learned by doing a project. This project proves that they can use their knowledge and finish something important. The size and difficulty of the project are planned so that students can complete it in the time they are given.

What is the Aim of an Internship Project?

The purpose of an internship project becomes clear to most students after they complete it and present their work.

By the time you reach this stage, you will have learned many useful things that will help you in real life, especially when you begin working as an engineer.

Here are some of the things you can learn during an internship project:

Knowledge and Expertise: learning more about a topic and gaining specialized skills.

Time Punctuality: working on a project within a set time frame.

Time Management: planning your tasks to finish the project on time.Research: collecting information from different sources related to your project topic.

Analytical Thinking: understanding and analyzing the information you collect.

You can also improve in areas such as:

Taking Responsibility: making decisions and recording them during your project.

Engineering Design: creating ideas and making decisions based on your research and analysis.

Individual Creativity: thinking of new or better solutions connected to your topic.

English Reading & Writing Skills: your English skills may improve by reading, writing, and reporting during the project.

Communication Skills: talking about your work with your supervisor or teammates helps you become better at technical communication.

Self-Confidence: explaining your project at the end of your internship will increase your confidence in yourself and your work.CyprusRun is a 3D running game where the player runs through a scripted terrain, a stylized landscape of Cyprus in our imagination. The objective of the game is to survive until the end of the sections. If you fail, restart again and again until you reach the endpoint. Each time the player dies, the game restarts and the score increases by one point. The game has features like fixed terrain that regenerates in each try, and what makes it unique is the inclusion of a history section that educates players about the rich history of Cyprus with its historical

timeline from ancient Ionians to the modern era and even the imagination of future Cyprus and the people of Cyprus. The game has a high level of reflex-based gameplay, but with this gameplay, the game gives you a rich amount of information with the history section of the game. With every try, the player learns a segment and wonders about the segment and checks what it is about.

Using This Book The main aim of this book is to help computer engineering students with the rules and steps for doing their internship projects. It shows students how to choose a project, how to write the report, how to prepare for the presentation, and how to explain and defend their work. This book answers many common questions that students may have during their internship. It can be used along with help and advice from your supervisor. The book is divided into different chapters. In Chapter One (Introduction)... In Chapter Two (General Information)...

Final Words I wish all students success in their internship projects. We also hope that during this process, students gain useful knowledge and skills in the topic they choose skills that will help them as future engineers.

TEAM MEMBERS

Name:Arda Tatar

Role: Solo Developer, Designer, Programmer

General Information

As a student of Software Engineering at Near East University, I had the chance to work on an internship project. This internship was not just something I had to do to complete the program, but actually a really useful experience for improving my skills and understanding how software development works in the real world.

Instead of learning everything from books or slides, I learned things directly from the field. I solved problems, developed code, tested features, and sometimes even fixed bugs that didn’t seem like a problem at first. I also realized how important it is to work regularly and communicate with supervisors or teammates just like we do with friends from school when we’re stuck. They gave me a lot of useful feedback, especially about my game project.

The internship gave me a real taste of what it’s like to work as a software engineer. I didn’t just write random code; I had to really think about what I was building, plan it properly, and sometimes even start from scratch when something didn’t work as expected. It helped me grow both technically and personally.

Topics for an Internship Project

There are many possible topics you can choose from during your internship in Software Engineering. Here are a few examples:

Software Development: Probably the most common topic building real applications for desktop, web, or mobile. You can use C#, Java, Python, JavaScript, etc. Personally, I think this is where we apply most of the things we’ve learned.

Artificial Intelligence: If you're into smart systems, this is a cool area. You can try out machine learning models, play with data, or create something like a classifier or recommendation system.

Neural Networks / Genetic Algorithms: These are more advanced than they seem, and I won’t lie they're not always easy. But they’re really interesting if you’re curious about how machines can “think” or evolve to solve problems.

Expert Systems: These are knowledge-based decision systems. You give them rules and logic, and they can help make decisions automatically.

Web & Internet Technologies: This includes full-stack development, frontend-backend communication, working with APIs, or cloud services. There’s a lot you can do here, and most companies need developers in this field.

Multimedia & Image Processing: This is a fun one if you enjoy working with images, sound, or video. For example: face recognition, speech-to-text, or detecting objects in a video. Let’s say your family member wants

someone removed from a photo if you can do that, it makes the photo even more heartwarming.

These are just examples. What really matters is choosing something you enjoy and can learn from. My advice: pick something that’s inside your comfort zone at first because learning is hard, and having fun while doing it makes everything easier.

The internship project is supposed to be your own work, so as a student, you’re responsible for everything inside it. We need to choose a topic, find a supervisor from the academic staff in the Software Engineering Department, and plan our time properly with their support.

At our faculty, we can’t just do the internship project on our own we need a supervisor to guide us. So yeah, they assign someone to help us out during the process. Their job is to support us, but that doesn’t mean they’ll do everything for us. it’s still up to us to stay on track, meet the deadlines, and get the work done.

One thing I really wanted to learn is respect and yeah, we got respect. You have to communicate properly with your supervisor and treat them well. They’re not just here to judge what you’ve done; they’re actually there to help you do it better. Sometimes they might not know your topic, and that’s okay.

They still trying to help you. It’s on you to explain them clearly what you want from them or project and ask the right questions whenever you get stuck.

At the end of the day, this is your project and you’re the one in charge. The supervisor is like your co-pilot they’re here to guide you, not to fly the whole thing for you.

Game Concept

CyprusRun is a 3D running game where the player runs through a scripted terrain, a stylized version of Cyprus in our imagination. The terrain isn’t just a simple background; it’s carefully designed to give a special and fun experience every time. “Scripted terrain” means that each run there is a scripted terrain, but with the difficulty of the game, these scripted terrains are feeling good for the player. Every run, you are exploring a new part of the map, so the game feels fresh and cool in a way. With the obstacles and structures in the way, it takes the player inside of it. The "stylized landscape of Cyprus" is inspired by the island’s nature and history, but it's not a realistic copy. Instead, it's an artistic and creative version, with things like reimagined seaports, colorful Mediterranean plants, and Unity’s cool vision that show Cyprus’s culture and past in a fun way.

The main goal of the game is to survive and reach the end of each section. This is the heart of the gameplay. Players must stay focused and react quickly to avoid obstacles and dangers. The game is split into 4 sections, each one with its own look and feel. This shows different

parts of the island or different time periods of the island. Players have to finish these sections without failing. If they do fail, they start again from the beginning. This adds a challenge and a reason to keep improving and inch to see other sections. Each section also reveals more about the game’s world and history as players progress.

If you fail, you restart again and again until you reach the goal. This is a big part of the game’s design. It’s okay to fail, because each try helps you get better. The game encourages you to keep playing, learn from

mistakes, and get used to the patterns. The idea is that every run makes you a little better and brings you closer to success. This retry system also supports the educational side of the game. Each attempt gives players a chance to learn something new.

Each time you die, the game restarts, and your score goes up by one. The score system is simple but meaningful. It doesn’t show how far you got; it shows how many times you tried. It rewards your encouragement to finish, not just success. Every time you fail, you earn a point. This turns failure into something valuable, showing your effort and how many times you’ve tried to improve and learn.

The game includes a fixed terrain that regenerates in every run, and its most unique feature is the history section. This section teaches players about Cyprus’s long and rich history, from ancient times to today and even into a future version of the island. The “fixed terrain” means that the world layout is the same, but obstacles and small elements change

each time. This keeps the game fun and makes sure players can’t just memorize a single path. This changing environment matches well with the history part of the game. The history section is not just extra; it’s a big part of the game. It gives players a full timeline of Cyprus, sharing knowledge about the island’s culture, people, and important moments in time. It also includes a creative look into what the future of Cyprus

might look like. The goal is to teach players meanwhile they are having fun.

The game is focused on reflex-based gameplay, meaning players have to act fast, jump, slide, or dodge obstacles with perfect timing. The fast speed and changing terrain make the game exciting and hard. But the real magic is how this fast gameplay connects to learning. The history section shares useful knowledge without slowing down the game. The way the game restarts after failure is used to show history in small parts. This keeps it light and easy to understand.

Gameplay And Features

Fixed 3D Terrain: From my past experience, this game type is best to play with 3D terrain, so we used 3D graphics for the game.

Movement-based character control: This is the main function of the game and innovative in a way that gives a totally different way of playing than Subway Surfers and Temple Run.

Run amount changes with each run attempt: Every run, the game counts your run amount so you know your run amount, and it adds some competition to the game.

In-game UI with access to a history section: The game has an in-game UI that helps you to reach the history section.

Every time the player restarts, they see a new part of Cyprus’s history. This is the smart part of the game’s learning system. After a difficult try, players get a short piece of historical information. This makes them curious, and they might want to learn more about it. They can then visit the history section to read more details of the history of Cyprus. This system turns failure into a chance to learn something new. The more players play, they see the history section after a certain point. Because of failing, they will have connection to the section and consider looking at the section’s history.

History Section

When you click the History button, six parts open up. Each one shows a different time in Cyprus. It’s not too detailed, just enough for people to understand how things changed over time.

1. Ancient Cyprus

In this phase, nationalities like the Ionians, Achaeans and others arrived. Small trade cities started growing and became rich. But after a while, the Greek kingdom invaded the island. The Ionians tried to rebel and take control of their country, but the Greeks won. Even though they won, they couldn’t reduce Cyprus’s autonomy.

Later, Alexander the Great came and took over Cyprus. This time it was

different. Cypriots became generals in his army and even had a big role in the government. But when Alexander died, Persians took control. Their rule was also different. They vassalised Cyprus, but Cyprus still had its own government and made its own decisions. Persians saw it as a vassal, but it was more like semi-independent.

After that, the Roman kingdom formed and took over. Cyprus became the Roman Cyprus Kingdom. They ruled for some time until the Roman Empire split. After that, Cyprus became part of the Eastern Roman Empire, also known as Byzantium.

2. Medieval and Byzantine Period

After the ancient era, Cyprus was controlled by the Byzantines for years. They ruled the west and continued as the Roman Empire in a new form. Eventually, the empire couldn’t stay together. It split again, and Cyprus stayed with the Byzantines. After the fall of the empire, Cyprus became independent for a while, but later came under the influence of the Mamluks.

3. Ottoman Era

Ottomans invaded Cyprus and brought their own system. Daily life started to change. You could see a mix in culture, food, architecture and how people lived. The Ottoman era had a big impact on Cyprus because they ruled it for a long time. This era ended during the time of Abdulhamid.

4. British Period

After the Ottomans, the British took control of the island. They built roads, schools and introduced modern systems. But this also made people want independence more than before. This period ended with protests and chaos as people fought for their future.

5. Modern Republic of Cyprus

Cyprus became independent from British rule. But things didn’t settle. Even Greek groups started fighting among themselves. In the end, the island got divided. That separation is still going on today. Both sides have moved forward in their own ways and built their own systems

6. Future of Cyprus

The future is about imagination. Maybe the people of Cyprus will build flying cars, maybe they’ll go to space, or maybe robots will guide tourists around the cities. No one knows what will happen. What’s possible is endless.

**Development Tools and Workflow**

Programming Language: C#

Design Tools: Unity UI system

Assets: Mix of custom and Unity Asset Store items

These are the tools I used for developing the game. Unity is a game engine that is free for students in many countries around the world. It is one of the most basic and important

C# is a high-level and general-purpose programming language. It supports different ways of programming, such as object-oriented, functional, and component-based programming. C# uses static and strong typing, and it is also lexically scoped.

The main creators of C# were Anders Hejlsberg, Scott Wiltamuth, and Peter Golde from Microsoft. The language was first released to the public in July 2000. Later, it became an international standard approved by ECMA. At the same time, Microsoft also released the .NET Framework and Microsoft Visual Studio. However, these tools were not open-source back then.

In 2004, a project called Mono started. It was free and open-source, and it allowed people to use C# on different platforms. About ten years later, Microsoft released new tools like Visual Studio Code (a code editor), Roslyn (a compiler), and the .NET platform. These tools support C#, and they are all free, open-source, and work on different operating systems. Microsoft also took over the Mono project, but it was not combined with .NET

The Ecma standard explains the main goals for designing the C# language:

C# is made to be a simple, modern, and general-purpose programming language. It follows the object-oriented style.

It should support good software engineering practices. These include strong type checking, checking array limits, warning when a variable is not initialized, and automatic memory cleaning (called garbage collection). The language focuses on making software stable, long-lasting, and easy to write.

C# is meant for creating software that can work across different systems, especially in networked (distributed) environments.

It's important that the source code can be used on different systems (portability), especially for programmers who already know C or C++.

The language also supports globalization, which helps create programs for users all around the world.

C# is suitable for making both large applications with full operating systems and small programs with limited functions (embedded systems).

C# is designed to work closely with the Common Language Infrastructure (CLI). Most of its basic types match the value types used in the CLI system. However, the C# language rules do not say that a compiler must create code for the Common Language Runtime (CLR), or produce Common Intermediate Language (CIL), or follow any specific format. Some C# compilers can even create machine code, just like traditional compilers for Objective-C, C, C++, Assembly, or Fortran.

In C#, a method is part of a class and acts like a function. It performs a set of actions, unlike a field, which just stores a value. Like in C++ or ANSI C, the structure of a method includes several parts in a specific order. First comes an optional access modifier like private. Then you write the return type such as int or void. After that, you write the method name, followed by a list of parameters in parentheses. Each parameter includes the type, a name, and sometimes a default value.

C# has special rules for passing values by reference. You must mark these parameters both in the method definition and when calling the method. You can choose between ref and out. Ref is used when the variable already has a value. Out is used when the variable will get its value inside the method and come out with a result.

You can also let a method accept a flexible number of arguments by using the params keyword on the last parameter.

Some short methods that only get or set a field value do not need a full method signature. But most of the time, you must fully define each method in the class.

In C#, like in C++, you must use the virtual keyword to let a method be overridden in a child class. But unlike C++, you must also use the override keyword in the child class. This helps avoid mistakes between overriding and overloading. If you want to hide a method instead of overriding it, you must use the new keyword. If you want to stop other classes from overriding a method or class, you can use the sealed keyword.

C# also has a feature called extension methods. These let you use static methods as if they were part of a class. This way, you can add new methods to existing types without changing their original code.

In C#, you can use memory-address pointers only inside code blocks marked as unsafe. Programs that include unsafe code need special permissions to run. Most of the time, objects are accessed through safe references. A safe reference always points either to a live object or to null; it can never point to an object that has already been garbage-collected or to a random piece of memory.

An unsafe pointer may point to an unmanaged value type. That means the type must not contain any references to objects that could be garbage-collected, such as other class instances, arrays, or strings. Code that is not marked unsafe can still store pointer values inside the System.IntPtr type, but it cannot dereference those pointers.

Managed memory is not freed manually. Instead, the garbage collector reclaims it automatically, which helps prevent memory leaks. If your code holds on to references longer than necessary, it may still consume extra memory, but once the last reference disappears, the memory becomes available for collection.

C# supplies a set of standard exceptions. Library methods throw these exceptions in documented situations, and you can create your own exception classes to handle special cases.

game development programs that you must learn if you want to create games with passion. Unity’s design tools are built into the engine, and with its own UI system, you can design anything you imagine in a short period of time.

Unity also provides assets from its website. Whenever you want to try something or are curious if an idea would look good in your project, you can easily find free assets, test them, and see how they fit your game. If you like them, you can even buy the full versions from the Unity Asset Store and improve your project even more.

The best part is that you don’t need to be an expert to use Unity. It is beginner-friendly and there are lots of tutorials online to help you learn step by step. With Unity, you can focus on both visuals and gameplay, and it gives you the freedom to make your dream game real.

Challanges and Solutions

UI scaling issues on multiple devices:

Solved with Canvas Scaler. Unity provides auto scaling for canvas. I used it and fixed my problems in another devices. Before that, everything was looking weird on different screens. On some phones buttons were too big, on some they were out of the screen. After I used Canvas Scaler and set it to scale with screen size, all UI elements started looking normal. It really saved my time and now the game looks fine on small and big screens without any extra work.

Score increment not working:

Fixed using collectable script that works like coin. So I put coin in the start of the game, and you are collecting it every single time you start game so game keeps track of your runs with coin script. Before this, I couldn’t really track how many times the player played the game. I wanted something simple and fun, so using coin logic

was easy and made sense. I also added a counter system that updates with the coin, so it gives motivation to keep running and getting a better high score.

Button navigation to history panel:

Fixed with SceneManager and put variable for every scene. At the start, buttons weren’t working or going to the wrong places. Then I made a small script for buttons to load the scenes or enable the panels. History panel opens with one click now, and other panels

too. It helped the menu feel smoother and more like a finished game.

Character getting out of map:

I put limitation to character's limit of going X, Y, Z so character is not going out of bounce. Sometimes player would fall off the map or just run into

empty space. That was looking bad. So I made a system to keep the player inside a certain area. I also made invisible walls to make sure even if you try, you can’t leave the main map. This helped the gameplay feel more controlled and clean.

Animation Problem:

I had problem with character not doing its animation. I took animations from Mixamo and made a script that uses animator in Unity without player seeing it. Script GetComponent<Animator> was savior in this action. Without this, my character looked frozen or buggy. After connecting animations to the states (like running, jumping), it started looking alive. Now when the player runs, jumps or dies, you can see the animation playing smoothly.

Game being easy:

When segments auto generated I had to be careful of game not forcing you to die. So I made one huge map that is hard to finish, so gameplay time and difficulty of the game increased by it. Before, the game was finishing too early or was boring. Now with the longer map and more obstacles, the game feels more like a challenge. I also made sure it’s not too hard from the beginning, but it gets harder while you play more. That keeps players interested and trying to go further each time.

Testing And Feedback

I tested the game with my friends and game feedback was: my friends appreciated the educational aspect and suggested smoother transitions and sound effects for history section of the game, while first tests game was too easy so we had to change segment generation.

Future Improvements

Unlockables Based on History Milestones

We got a second character, Kamil. If you read every part of the history section and pass the secret quiz, you might unlock a hidden character to play instead of Necmi. Just a fun little bonus for the lore nerds.

Randomized Map / Endless Mode

Once you finish the main game, we might unlock an Endless Mode where you can play on randomized terrain. Not in yet, but 100% something we can add later replayability boost.

Audio Narration for History

We don’t have rich sound design yet, especially in the history parts. Maybe in the future we throw in some narration to make it more alive. Depends on time/resources, but it’s on the radar.

Leaderboard for Run Counts

Would be cool to show a leaderboard based on how many runs it took you to finish the game or maybe how many times you’ve beaten it total. Gives speedrunners and perfectionists something to chase.

Interactive History w/ AI

Later down the line, we could make the history section interactive like, you import requests or ask Gemini AI stuff about Cyprus history, and it throws questions back at you. Would make learning way more dynamic.

Mobile app implementation

I think game mostly fitts in mobile gaming industry i did it for computers so this becomes future improvement for the game

Chat with other people

I think adding chat to game might be good people can share of their ideas inside of it or they can give advices to newer players on how to do some segments of the game

Conclusion

With this game i aimed towards a lot of stuff that can be accomplish game is all about teaching people meanwhile letting them have fun this was unique for me that i havent seen lots of time so doing that was fun with the rich data of c sharp i made a huge game that will let people have fun in the game

CyprusRun has a unique gameplay style that no other game really has. It's a solo-developed project that tries to mix fun and learning in a different way. In future versions, the plan is to bring more ways to learn through AI, add leaderboards so people can compete, and include a lot more stuff that goes beyond what you’d expect

I wanted to create a 3D endless runner game, so I used Unity and C# to build it. The game has four different parts that show how Cyprus looked throughout history - from way back in ancient times all the way to what it might look like in the future. I thought it would be cool to teach people about history while they're playing, so I added a section with information about six different time periods.

Making this game wasn't easy at all. I ran into tons of problems along the way. The biggest headache was getting the user interface to look right on different phone and tablet screens - it kept getting stretched or squished in weird ways. I also had to figure out how to make the characters move naturally, which meant learning how to use something called Mixamo for animations. Every time I solved one of these problems, I felt like I was getting better at programming.

The scoring system in my game works differently than most endless runners. Instead of counting how far you run, it counts how many times you attempt the level. I did this because I wanted players to feel okay about failing and trying again. When you're not worried about losing your high score, you're more willing to take risks and actually learn something from the history parts.

Working on this project taught me that game development is much harder than I expected, but also way more rewarding when you finally get everything working properly.

Screenshotsfig1

fig2

fig3



Fig4

In the first 4 figures, I showed the game, and in the last 3 figures, I showed some of the scripts that I used for the project that has been done.  
  
In the second figure, I showed the first segment of the game. This one has a rich history for me and for Cyprus.  
  
For me, the second picture showed me the way of how a game can be created how you add assets, physics, scripts, and more. In the further figures, I showed some scripts. Most of them or the ones that are not in the screenshots were created during the era of this figure.  
  
For Cyprus history, this picture represents the first Cypriot people that colonized the island. This is unique because it is the beginning of the island's history. This island always attracted world powers, so it affected the whole world.  
  
In the first figure, I showed the second segment of the game. This was the time I was trying to find assets for the game, and the map was hard to implement. For me, it’s the best one in the game.  
  
Figure 3 is the one about modern Cyprus. This segment was one of the hardest parts of the game to create. But because of this hard implementation, I made the next segment much easier to finish since players might have been frustrated by the difficulty of this part.  
  
Figure 4 is the end of the game. It’s much easier than the other segments and has a futuristic style. It might represent what can happen in the future big structures, futuristic cars, and at the end, a big ship that has turbos. For me, that ship can fly out of the world. So I thought about it like the end of the game is leaving the island with a big ship that flies away

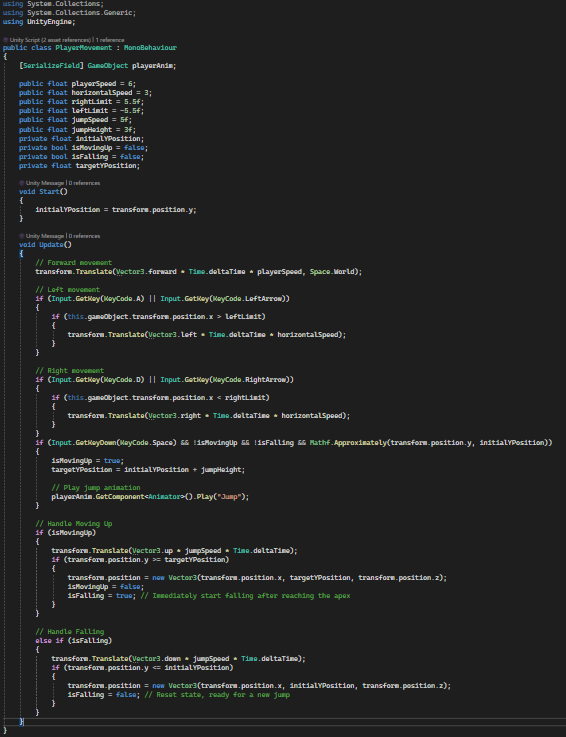
Fig5

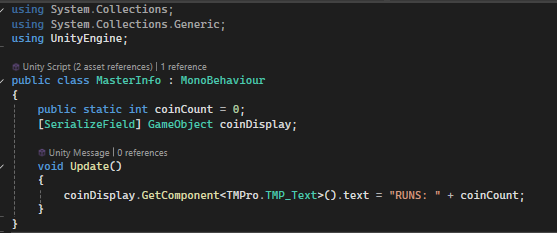
Fig 5, Fig 6, and Fig 7 are the most important scripts for the game, in my opinion. While working on them, I created the most important parts of the game like running animations and the main goal counting the runs.

Fig 6 is the running code of the game. It controls how you move forward, go right, and left. You can press A and D to dodge the obstacles coming at you or running towards them. Jumping is done with the space button, and you move in the Y direction while jumping.

Fig 5 is about the game animations jumping, stumbling backwards, and running. Running happens constantly when you don’t do any other animation, so your character is always moving. Stumbling backwards happens when you die, and jumping happens when you press the space button. Without these animations, the game would lack its spirit, so I put a lot of care into this script.

Fig 7 this is the code that is about run count it keeps the runs you have tried this code this creates goal of the game and something that you grind for

fig6

fig7

References

[https://assetstore.unity.com/packages/3d/environments/landscapes/lowpoly-environment-pack-99479](https://assetstore.unity.com/packages/3d/environments/landscapes/lowpoly-environment-pack-99479https:/assetstore.unity.com/packages/3d/environments/old-sea-port-environment-36897https:/assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-futuristic-environment-pack-v2-0-246983https:/assetstore.unity.com/packages/3d/environments/urban/simple-city-pack-plain-100348https:/en.wikipedia.org/wiki/CyprusHill,%20Sir%20George.%20A%20History%20of%20Cyprus,%20Vol.%204.https:/en.wikipedia.org/wiki/History_of_Cyprus#Ottoman_Cyprus_(1571%E2%80%931878)https://en.wikipedia.org/wiki/Millet_(Ottoman_Empire)https://en.wikipedia.org/wiki/Ottoman_conquest_of_Cyprushttps://books.openedition.org/efa/3081https://cyprustravellerguide.com/the-city-kingdoms-of-ancient-cyprus/https://en.wikipedia.org/wiki/Ten_city-kingdoms_of_Cyprushttps://www.metmuseum.org/essays/ancient-greek-colonization-and-trade-and-their-influence-on-greek-art)

[https://assetstore.unity.com/packages/3d/environments/old-sea-port-environment-36897](https://assetstore.unity.com/packages/3d/environments/landscapes/lowpoly-environment-pack-99479https:/assetstore.unity.com/packages/3d/environments/old-sea-port-environment-36897https:/assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-futuristic-environment-pack-v2-0-246983https:/assetstore.unity.com/packages/3d/environments/urban/simple-city-pack-plain-100348https:/en.wikipedia.org/wiki/CyprusHill,%20Sir%20George.%20A%20History%20of%20Cyprus,%20Vol.%204.https:/en.wikipedia.org/wiki/History_of_Cyprus#Ottoman_Cyprus_(1571%E2%80%931878)https://en.wikipedia.org/wiki/Millet_(Ottoman_Empire)https://en.wikipedia.org/wiki/Ottoman_conquest_of_Cyprushttps://books.openedition.org/efa/3081https://cyprustravellerguide.com/the-city-kingdoms-of-ancient-cyprus/https://en.wikipedia.org/wiki/Ten_city-kingdoms_of_Cyprushttps://www.metmuseum.org/essays/ancient-greek-colonization-and-trade-and-their-influence-on-greek-art)

[https://assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-futuristic-environment-pack-v2-0-246983](https://assetstore.unity.com/packages/3d/environments/landscapes/lowpoly-environment-pack-99479https:/assetstore.unity.com/packages/3d/environments/old-sea-port-environment-36897https:/assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-futuristic-environment-pack-v2-0-246983https:/assetstore.unity.com/packages/3d/environments/urban/simple-city-pack-plain-100348https:/en.wikipedia.org/wiki/CyprusHill,%20Sir%20George.%20A%20History%20of%20Cyprus,%20Vol.%204.https:/en.wikipedia.org/wiki/History_of_Cyprus#Ottoman_Cyprus_(1571%E2%80%931878)https://en.wikipedia.org/wiki/Millet_(Ottoman_Empire)https://en.wikipedia.org/wiki/Ottoman_conquest_of_Cyprushttps://books.openedition.org/efa/3081https://cyprustravellerguide.com/the-city-kingdoms-of-ancient-cyprus/https://en.wikipedia.org/wiki/Ten_city-kingdoms_of_Cyprushttps://www.metmuseum.org/essays/ancient-greek-colonization-and-trade-and-their-influence-on-greek-art)

[https://assetstore.unity.com/packages/3d/environments/urban/simple-city-pack-plain-100348](https://assetstore.unity.com/packages/3d/environments/landscapes/lowpoly-environment-pack-99479https:/assetstore.unity.com/packages/3d/environments/old-sea-port-environment-36897https:/assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-futuristic-environment-pack-v2-0-246983https:/assetstore.unity.com/packages/3d/environments/urban/simple-city-pack-plain-100348https:/en.wikipedia.org/wiki/CyprusHill,%20Sir%20George.%20A%20History%20of%20Cyprus,%20Vol.%204.https:/en.wikipedia.org/wiki/History_of_Cyprus#Ottoman_Cyprus_(1571%E2%80%931878)https://en.wikipedia.org/wiki/Millet_(Ottoman_Empire)https://en.wikipedia.org/wiki/Ottoman_conquest_of_Cyprushttps://books.openedition.org/efa/3081https://cyprustravellerguide.com/the-city-kingdoms-of-ancient-cyprus/https://en.wikipedia.org/wiki/Ten_city-kingdoms_of_Cyprushttps://www.metmuseum.org/essays/ancient-greek-colonization-and-trade-and-their-influence-on-greek-art)

[https://en.wikipedia.org/wiki/Cyprus Hill, Sir George. A History of Cyprus, Vol. 4.](https://assetstore.unity.com/packages/3d/environments/landscapes/lowpoly-environment-pack-99479https:/assetstore.unity.com/packages/3d/environments/old-sea-port-environment-36897https:/assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-futuristic-environment-pack-v2-0-246983https:/assetstore.unity.com/packages/3d/environments/urban/simple-city-pack-plain-100348https:/en.wikipedia.org/wiki/CyprusHill,%20Sir%20George.%20A%20History%20of%20Cyprus,%20Vol.%204.https:/en.wikipedia.org/wiki/History_of_Cyprus#Ottoman_Cyprus_(1571%E2%80%931878)https://en.wikipedia.org/wiki/Millet_(Ottoman_Empire)https://en.wikipedia.org/wiki/Ottoman_conquest_of_Cyprushttps://books.openedition.org/efa/3081https://cyprustravellerguide.com/the-city-kingdoms-of-ancient-cyprus/https://en.wikipedia.org/wiki/Ten_city-kingdoms_of_Cyprushttps://www.metmuseum.org/essays/ancient-greek-colonization-and-trade-and-their-influence-on-greek-art)

[https://en.wikipedia.org/wiki/History\_of\_Cyprus#Ottoman\_Cyprus\_(1571%E2%80%931878)](https://assetstore.unity.com/packages/3d/environments/landscapes/lowpoly-environment-pack-99479https:/assetstore.unity.com/packages/3d/environments/old-sea-port-environment-36897https:/assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-futuristic-environment-pack-v2-0-246983https:/assetstore.unity.com/packages/3d/environments/urban/simple-city-pack-plain-100348https:/en.wikipedia.org/wiki/CyprusHill,%20Sir%20George.%20A%20History%20of%20Cyprus,%20Vol.%204.https:/en.wikipedia.org/wiki/History_of_Cyprus#Ottoman_Cyprus_(1571%E2%80%931878)https://en.wikipedia.org/wiki/Millet_(Ottoman_Empire)https://en.wikipedia.org/wiki/Ottoman_conquest_of_Cyprushttps://books.openedition.org/efa/3081https://cyprustravellerguide.com/the-city-kingdoms-of-ancient-cyprus/https://en.wikipedia.org/wiki/Ten_city-kingdoms_of_Cyprushttps://www.metmuseum.org/essays/ancient-greek-colonization-and-trade-and-their-influence-on-greek-art)

[https://en.wikipedia.org/wiki/Millet\_(Ottoman\_Empire)](https://assetstore.unity.com/packages/3d/environments/landscapes/lowpoly-environment-pack-99479https:/assetstore.unity.com/packages/3d/environments/old-sea-port-environment-36897https:/assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-futuristic-environment-pack-v2-0-246983https:/assetstore.unity.com/packages/3d/environments/urban/simple-city-pack-plain-100348https:/en.wikipedia.org/wiki/CyprusHill,%20Sir%20George.%20A%20History%20of%20Cyprus,%20Vol.%204.https:/en.wikipedia.org/wiki/History_of_Cyprus#Ottoman_Cyprus_(1571%E2%80%931878)https://en.wikipedia.org/wiki/Millet_(Ottoman_Empire)https://en.wikipedia.org/wiki/Ottoman_conquest_of_Cyprushttps://books.openedition.org/efa/3081https://cyprustravellerguide.com/the-city-kingdoms-of-ancient-cyprus/https://en.wikipedia.org/wiki/Ten_city-kingdoms_of_Cyprushttps://www.metmuseum.org/essays/ancient-greek-colonization-and-trade-and-their-influence-on-greek-art)

[https://en.wikipedia.org/wiki/Ottoman\_conquest\_of\_Cyprus https://books.openedition.org/efa/3081](https://assetstore.unity.com/packages/3d/environments/landscapes/lowpoly-environment-pack-99479https:/assetstore.unity.com/packages/3d/environments/old-sea-port-environment-36897https:/assetstore.unity.com/packages/3d/environments/sci-fi/sci-fi-futuristic-environment-pack-v2-0-246983https:/assetstore.unity.com/packages/3d/environments/urban/simple-city-pack-plain-100348https:/en.wikipedia.org/wiki/CyprusHill,%20Sir%20George.%20A%20History%20of%20Cyprus,%20Vol.%204.https:/en.wikipedia.org/wiki/History_of_Cyprus#Ottoman_Cyprus_(1571%E2%80%931878)https://en.wikipedia.org/wiki/Millet_(Ottoman_Empire)https://en.wikipedia.org/wiki/Ottoman_conquest_of_Cyprushttps://books.openedition.org/efa/3081https://cyprustravellerguide.com/the-city-kingdoms-of-ancient-cyprus/https://en.wikipedia.org/wiki/Ten_city-kingdoms_of_Cyprushttps://www.metmuseum.org/essays/ancient-greek-colonization-and-trade-and-their-influence-on-greek-art)

<https://www.mixamo.com/#/?page=2&type=Character>

<https://assetstore.unity.com/packages/p/environmental-asset-pack-170036>

<https://assetstore.unity.com/packages/3d/environments/hand-painted-nature-kit-lite-69220>

<https://assetstore.unity.com/packages/3d/environments/low-poly-environment-315184>

<https://assetstore.unity.com/packages/3d/environments/lowpoly-environment-nature-free-medieval-fantasy-series-187052>

<https://cyprustravellerguide.com/the-city-kingdoms-of-ancient-cyprus/>

<https://en.wikipedia.org/wiki/Ten_city-kingdoms_of_Cyprus>

<https://www.metmuseum.org/essays/ancient-greek-colonization-and-trade-and-their-influence-on-greek-art>

<https://en.wikipedia.org/wiki/C_Sharp_(programming_language>)

Page Setup

Type the manuscript on one side of the standard-sized heavy white bond paper, (A4 format 297x210, portrait orientation). The page setup margins are: (2.5 cm) at the top, bottom and right, whereas the left margin is (3.5 cm). The reason for increasing the left margin is that the project will be bound and enough space should be left so as not to cover the text.

In addition to setting up the margins, the following line spacing settings should be used uniformly throughout the project:

One and half (1.5) spacing throughout the project

One and half (1.5) spacing after every line of the title, headings, quotations, references, figures, tables, ...etc

Page numbering should be consecutive. Type the numbers in the bottom center using Arabic numerals for the main parts of the project (i.e. Introduction, the chapters and Conclusion).

Arrange the pages of the manuscript as follows:

Title page should not be numbered

Acknowledgment should be numbered with (i)

Abstract page should be numbered with (ii)

Contents page(s) should be numbered starting with (iii)

Text starts on a new page numbered (1)

Acknowledgment should be numbered with (i)

Abstract page should be numbered with (ii)

Contents page(s) should be numbered starting with (iii)

Text starts on a new page numbered (1)

Punctuation and Spacing

Punctuation marks provide clearer meaning to the sentences and paragraphs. When using punctuation marks, make sure your use of spacing is applied as follows:

One space to appear after all punctuation marks

Exceptions:

No space after internal periods in abbreviations (a.m., i.e., U.S.)

No space after the colon in ratios (6:1)

No space before or after hyphens (trail-by-trail analysis)

B5. Abbreviations

An abbreviation is a short form of a word that can be obtained by omitting some letters of the end of the word. There are also contractions, which are also short forms of words, but obtained by omitting some middle letters of the word

Examples of frequently used common abbreviations are:

ca. – (circa) meaning "at or near a given date"

cf. – (confer, compare, or consult)

i.e. – (id est, "that is")

et al. – (et alii) meaning "and others", to be used for multiple authors, etc.

e.g. – (exempli gratia) meaning "for example"

n. – ("note")

Passim – ("here and there")

Sic – ("thus") italicized in brackets [ ] to indicate error in original quote

n.d. – ("no date") if no date is given for publication, letter, etc.

Overview

All software engineering students are required to give oral presentations at the end of their internship project. The audience usually consists of the internship jury, which may include academic staff and fellow students.

The presenting student is allowed to schedule their presentation only after their internship supervisor has confirmed that the internship period has been successfully completed and the project outcomes are ready to be presented.

The student is expected to summarize the internship topic, the responsibilities undertaken, and the conclusions or deliverables achieved. The student may also provide insight into how the internship

experience developed over time, such as:

What was the main objective of the project?

What challenges occurred?

What unexpected issues occurred during the project?

From the student’s point of view, what are the strengths of doing the internship project?

What aspects of the project could be improved further?

The internship jury are invited here to ask questions about the project scope, the student’s technical contribution to the problem, problem-solving methods, and the outcomes produced.

There is no time limit for the presentation, but it is still recommended to keep it clear, structured, and well-delivered.

Bibliography

"Learning C# by Developing Games with Unity" by Harrison Ferrone

Unity in Action – Joseph Hocking

Developing 2D Games with Unity – Jared Halpern

Mastering Unity 2D Game Development – Simon Jackson

Unity 2023 Mobile Game Development – John P. Doran

C# Programming for Unity Game Development – Jesse Freeman

Beginning 3D Game Development with Unity – Sue Blackman

Unity Game Development Cookbook – Paris Buttfield-Addison et al.

Pro Unity Game Development with C# – Alan Thorn

Unity From Zero to Proficiency (Foundations) – Patrick Felicia

Unity 2D Game Development Cookbook – Claudio Scolastici

Game Programming Patterns – Robert Nystrom (free online)

Introduction to Game Design, Prototyping, and Development – Jeremy Gibson Bond

Game Development with Unity for .NET Developers – Jiadong Chen

The Art of Game Design: A Book of Lenses – Jesse Schell

Game Engine Architecture – Jason Gregory

Beginning C# Programming with Unity – Patrick Felicia

Unity Game Optimization – Chris Dickinson

Holistic Game Development with Unity – Penny de Byl

Unity AI Game Programming – Ray Barrera et al.