



**KOÇ UNIVERSITY**

*College of Engineering*

**DEPARTMENT OF COMPUTER ENGINEERING**

**COMP 291 SUMMER PRACTICES I**

**Batuhan Arat**

**Internship Company and Department:**

**<Crs Soft / Software Engineering >**

**03.07.2023 / 04.08.2023**

**Internship Supervisor (Name, Department, Phone, Fax, E-mail, etc.):**

Beste Özalp Yeşilyurt

Talent & Employee Experience  
Manager

beste.ozalp@crssoft.com

0212 489 33 13

**Signature:**

  
YAZILIM HİZMETLERİ A.Ş.<sup>1</sup>  
Yıldırım Teknikleri Üni. Davutpaşa Kamp.  
Teknoloji Geliştirme Bölge B2-401  
Esenler / İSTANBUL  
Esenler V.D. 215 024 0232

## **Table Of Contents**

<b>1. Introduction</b>	3
<b>2. Company Description</b>	4
<b>3. Making Crazy Forest</b>	6
3.1. Problem Statement	6
3.2. Team	8
3.3. Tools and Techniques Used	8
3.4. Detailed Explanation	10
3.4.1. Server Structure and Database Settings	10
3.4.2 Game Interface and Gameplay Design	12
3.4.3. Animations	15
3.5. Results	16
<b>4. Conclusions</b>	21
<b>Appendix</b>	23
<b>References</b>	23

# 1. Introduction

During my internship at Crs Soft, a software and consultancy firm under the umbrella of Şık Makas Group, I was provided with a unique opportunity unlike any other. Unlike being assigned to a pre-defined project, I was asked to propose a project idea. With an eagerness to enhance my backend skills and delve into game development, I proposed a project that involved creating a game interfaced with a server. They welcomed my idea, and thus I have begun to the project.

I wanted to write a game that would interact with a server, for which I chose to use ASP.NET for the server-side, Azure Cloud SQL for database management, and Unity as the game engine. The reason behind choosing Unity was its compatibility with ASP.NET, enabling me to write consistent code in C#. Although I was unfamiliar with C# initially, I embraced the learning curve that came along with it.

During the internship, we met with supervisors in weekly meetings both individually and with other interns. We are introduced to the agile cycle at those meetings that CRS Soft applies in all of their projects. Our progress is tracked at meetings as well.

CRS Soft is also preparing to launch a company named Corvus Jay for the mobile game industry. They have not started the development cycle of the project yet. For now, they are mainly focused on the artistic approach of the game and game design in general. So, being in an environment where games are discussed constantly attracts me to CRS Soft. My interest in game development, combined with lots of different people to communicate with, gives me great insights into the game sector.

My transition from a student to an engineer was smooth enough thanks to the warm environment that CRS Soft ensured. The freedom that I was given in the project helped me to work at my own pace and learn about the project topics that I chose. It improved my programming skills with C#, and I got hands-on experience with industry-standard tools.

## 2. Company Description



*Figure 1: Crs Soft Logo*

CRS Soft is a software company established as an umbrella company of the Şık Makas Group in 2010. Şık Makas Group has a long-standing history dating back to 1939 with interests in the textile, retail, technology, energy, and real estate sectors. CRS Soft was established as their initiative in digital transformation. They have done business with international companies such as Switzerland, Germany, and so on, as well as national companies.

One of the significant brands of Crs Soft is its e-transformation brand, e-dünya. Edünya includes lots of digital solutions offered on a singular platform. These solutions are aimed at modernizing business processes, ensuring compliance with regulatory requirements, and enhancing operational efficiency. It has been giving service for ten years in the sector. They have exceeded 100,000 customers, showing their broad reach and acceptance among users.

The products under the e-dünya brand include:

**E-Invoice:** A solution for generating, sending, receiving, and storing invoices electronically

**E-SMM:** An electronic service for managing and monitoring sales and purchases, tracking accurate record-keeping, and compliance with tax regulations.

**E-Archive Invoice:** A digital solution for archiving invoices securely and efficiently

**E-MM:** A platform for managing and monitoring electronic messages, promoting effective communication and collaboration among teams.

**E-Currency:** A digital currency management solution that simplifies transactions and ensures compliance with financial regulations.

**E-Ledger:** An electronic ledger for recording and tracking financial transactions accurately and transparently, aiding financial reporting and analysis.

**E-Reconciliation:** A digital reconciliation platform that automates the matching and reconciliation of financial transactions

**E-Waybill:** An electronic solution for generating and managing waybills, streamlining the documentation process for goods transportation.

**E-Addition:** A platform for managing additional documents and information required for business operations, ensuring completeness and accuracy.

**E-Expenses:** A digital expense management solution that simplifies the tracking, approval, and reimbursement of business expenses.



*Figure 2: Corvus Jay Logo*

Additionally, CRS Soft is starting its initiative in the mobile gaming industry as a company named **Corvus Jay**. Their vision is focused on reflecting the artistic creativity of their team. Experiences.

Deloitte Technology Fast 50 Turkey awarded Crs Soft 6 times in a row starting from 2017. They are among the top 500 fastest-growing technology companies in the EMEA region, showing their progress in the tech industry.

### **3. Making Crazy Forest**

#### *3.1. Problem Statement*

##### **Game Description:**

I want to create a game that has basic gameplay mechanics and works with a server.

The main gameplay of the game is inspired by Icy Tower, which is a game in which the user progresses by jumping along the platforms. I have added enemies to that gameplay that we have to avoid contact with them. If a player contacts the enemy, the player dies unless the contact was made by the player who jumps to the enemy's

head. In that scenario, the enemy is killed, and the player continues to move on. If a player gets killed by the enemy, the player has to start from the beginning. The player also has a diving mechanic that boosts the fast gameplay experience.

The server is implemented to achieve two functionalities in the game: Authentication to handle the register and log-in process to the game and authorization to control which user has access to which character. Both logic was implemented in the server, and the hashing and salting algorithms achieved the server's security.

### **Key Features:**

**Jumping Mechanic:** Similar to "Icy Tower," players jump from platform to platform in a vertical environment.

**Enemy Interactions:** Enemies are encountered during jumps, and players can eliminate them by jumping on top of them.

**Starting Over:** Rogue-like nature, for in any mistake of yours, you start from the beginning.

**Engaging Assets and Music:** The game has carefully selected assets and music to give a unique experience to the player.

**Animations:** Animations are included to make the game more visually appealing. Tail wiggle animation when a character is crouching, jumping, and movement animations for both our character and the enemies are examples to make the game more responsive.

**Dive Mechanic:** A dive mechanic that increases the player's speed when descending to increase gameplay speed.

### 3.2. *Team*

I have worked on this project individually. I have a supervisor, Gözde Öcal, who controls my progression weekly. Also, we have another weekly meeting under the supervisor of Hilal Demirlenk. In this weekly meeting, all other software development interns participate. We report our progress and listen to the comments of other interns about our projects. It was a good opportunity to track the other's progression, see their work on their project, and receive comments about my work.

### 3.3. *Tools and Techniques Used*

Software development team supervisor Gözde Öcal helped me to form a project plan that follows Crs Soft's standard workflow. At Crs Soft, every project begins with an analysis phase. After the analysis phase, projects were divided into phases that included Software Development and Testing components. My project has three phases.

1. Server Structure and Database Settings
2. Game Interface and Gameplay Design
3. Animation

For each of these phases, we've scheduled Interface tests, Unit tests, and Regression tests. After completing these phases, we'll finish the project with Closing Tests. The project timeline is set to begin on July 3rd and conclude on August 4th

CRS Soft software development team highly utilizes C# and .NET for their development. I have complied with their tech stack. Additionally, some key tools and technologies I worked with include:

**C#:** Used as the primary programming language for development for game development and server structure



**Unity:** Used for game development engine. It is free, and it has lots of documentation which is available.

**ASP.NET Core 7:** Utilized in web development projects, 7 Is chosen for taking advantage of the latest features and improvements.

**Azure SQL Cloud Database:** Used as the database, providing secure and scalable storage for our project. This is chosen to access the database among different computers.

**Postman API:** Used to test and validate HTTP requests, ensuring the server's functionality.

**Entity Framework:** Used as an object-relational mapper to facilitate interactions with the SQL Cloud Database.

**Azure Data Studio:** Used for database management and inspection.

**Azure DevOps:** Used for version control and project management. It is the primary tool that is used at the Crs Soft.

**Microsoft Teams:** Used for team communication and collaboration.

### 3.4. *Detailed Explanation*

#### 3.4.1 *Server Structure and Database Settings*

I have created separate projects for the server and the game itself to make debugging easy. In that way, I can test my server without running the game and vice versa. I have created a shared model directory as a separate solution class in my server project. By creating a DLL at the game project and specifying the game project path in the server's project settings, it should transfer the shared directory files to the game at the post-build. In that way, I don't need to duplicate the same code.

#### **Authentication - Security**

Storing passwords as plain text in the database is a very bad practice in terms of security reasons. People should not, but they use the same password for different accounts, too. So, if they hacked a single application like our game, we are violating their security in a bigger sense. Therefore, our server should implement a security layer, and hashing is one of them. Hashing is a one-way function that takes plain text and converts it to a unique piece of data. Hashed data is acted as a password and is stored in the database. I use the SHA-256 HMAC hashing algorithm for this purpose.

There is another rare case of security vulnerability that I should take care of, which is differentiating the same passwords. People that have the same password, which has exactly the same hash in the database. Thus, hacking one of the hashed data exposes the other password, which uses the same password. In order to prevent that, there is a technique called salting. Salt gives different hashes to people who have the same password; therefore, another security level is sustained. In the end, in our code, hashed and salted passwords are stored in our database.

When a user logs in with a wrong username or wrong password, the return message of this activity should not involve the specific part that is not correct. People who use brute force attacks to guess a username or password take advantage of this type of situation. If they correctly guess the username but their password guess is wrong, replying to them with a message of the wrong password makes them brute force attack to password only. Thus, replying to messages of incorrect data should not be specific about the parts. "Username or password is not correct" message can be returned.

### **Authentication - Authorization JWT Token**

A JSON Web Token (JWT) is a secure digital package for sharing user and authorization info between the app and a server. When a user logs in successfully, the server gives them a JWT. They use this JWT to do something on the website, like accessing their account; they forward the JWT in the background. The server checks the JWT to make sure it's really from that user and hasn't been messed with and then lets the user do what they want.

The most helpful feature of the JWT token is to decrease the coupling between the server and the database. Jwt has information about the user's identity and permissions. A signature on the server side encapsulates this information. There is no need to make time-consuming database calls about who the user is and what they can do; it is hidden on the Jwt itself.

So, JWTs are like a secret handshake that helps websites and apps know who you are and what you can do, all without slowing things down with constant database checks. We are using this technique in authentication and authorization in our server by Bearer Keys.

### 3.4.2 Game Interface and Gameplay Design

In the game's domain model, I have a player, enemy, platform, and managers. The player is jumping to the platform. Jumping from below should be transient to the player, and oppositely, landing on the platform should be impermeable. This behavior can be obtained by adding Platform Effector to the platform with Edge Collider's set used by the effector. I wrote a complex movement controller for the player to handle dive mechanics, crouch mechanics, jumping mechanics, and movement mechanics. My code will use a solid approach, which I implement in a reusable way.

#### **Usage of Scriptable Objects**

*While I was learning Unity, Scriptable Objects caught my attention. It has very few explanations about what is in the documents, but the things that can be used with them motivate me. They are derived from the same root from the MonoBehaviour class, but they cannot be assigned to the Game Object directly as MonoBehaviour does. The most important feature that I used in my game is that their lifecycle does not depend on the scenes; it is a scene-independent asset whose lifecycle can be fully controlled by the user. So, if you change some data at the runtime, it persists after the runtime, unlike other data types that are stored in MonoBehaviors. That makes them perfect data containers, but they can be used for architecture methods.*

*To create this, you should create a script inherited from Scriptable Objects and [CreateAssetMenu(fileName = "", menuName = "")] tag before the class declaration. By doing this, you can create any scriptable object instance by right-clicking the asset folder in Unity and selecting from the menu.*

*In my game, I will use Scriptable Objects as data containers, modular, pluggable AI for enemies, and event architecture. Using them as a data container is selfexplanatory. All game stats that need to be saved per play can be used in the scriptable object data container.*

## **Pluggable AI**

*I have created 2 type of enemy, which has two kinds of behavior. The beetle only moves horizontally, and when it reaches the end, it turns around and moves in the other direction. Vulture is following our character when it is spawned. Those 2 types of different behavior can be represented by two types of brains. Encapsulating those brains using scriptable objects offers me a modular, reusable, and data-driven approach. These templates can be easily adjusted in the Unity Editor without code changes, making it accessible for designers to create and manage different enemy actions.*

## **Scriptable Object Event Architecture**

*In every event architecture, we should take three things to take into account. First, we should have something that triggers the event. Second, we should use something that receives the triggered event. Third, we should know what should be done when this event is triggered.*

*The event itself can be a scriptable object. This allows us to create a sceneindependent event structure. Remember that scriptable objects do not depend on the lifecycle of the scenes, so we can have modular events that can be referenced from any scene. We don't have to create persistent manager and their unrelated references. We should group the event types as their parameter. VoidEventSO should be created as a base structure for events that don't need any parameter to receive. (i.e. Die Event). IntEventSO should be created in the same fashion for the events that need some integer value to receive. (i.e., Health Change Event). If we need to create an event from this base, we can create special events by rightclicking the asset menu.*

*Action/Trigger should be MonoBehaviour to raise an event. We also need an Event Listener as MonoBehaviour to listen to event. Raised and listened scriptable object events can be dragged from the asset menu.*

## **Performance Optimization**

*Crazy Forest is a game that, in theory, can be played forever. It's end condition is dependent on the user. If the user never makes any mistake, our character keeps jumping forever. Therefore, we need to take care of the object spawn methods and possible object destruction methods. Destruction is important because if we are going to spawn more platforms and enemies, as long as the character continues, it accumulates in the memory and, in the end, freezes the game. So, we need to destroy it after some distance is passed. But creation and destruction take time, and doing those for the same objects creates performance issues, too.*

## **Object Pool Design Pattern**

*Object Pool design pattern is a perfect solution for this type of situation. It creates a certain amount of objects at the beginning of the game and stores them in queues. When we need the object, we are retrieving them from the queue, so there is no need for creating it again. When we don't need this object, we can place it in the queue without destroying the whole object completely. So, instead of creation and destruction, the logic becomes activation and deactivation of the object from the pool.*

*I have created a list of pools that includes platforms and enemy types. Each pool is a dictionary that contains string and Queue<GameObject>. Spawning from the pool and returning to the pool are just enqueueing and dequeuing algorithms due to the queue logic.*

## **Singleton Design Pattern**

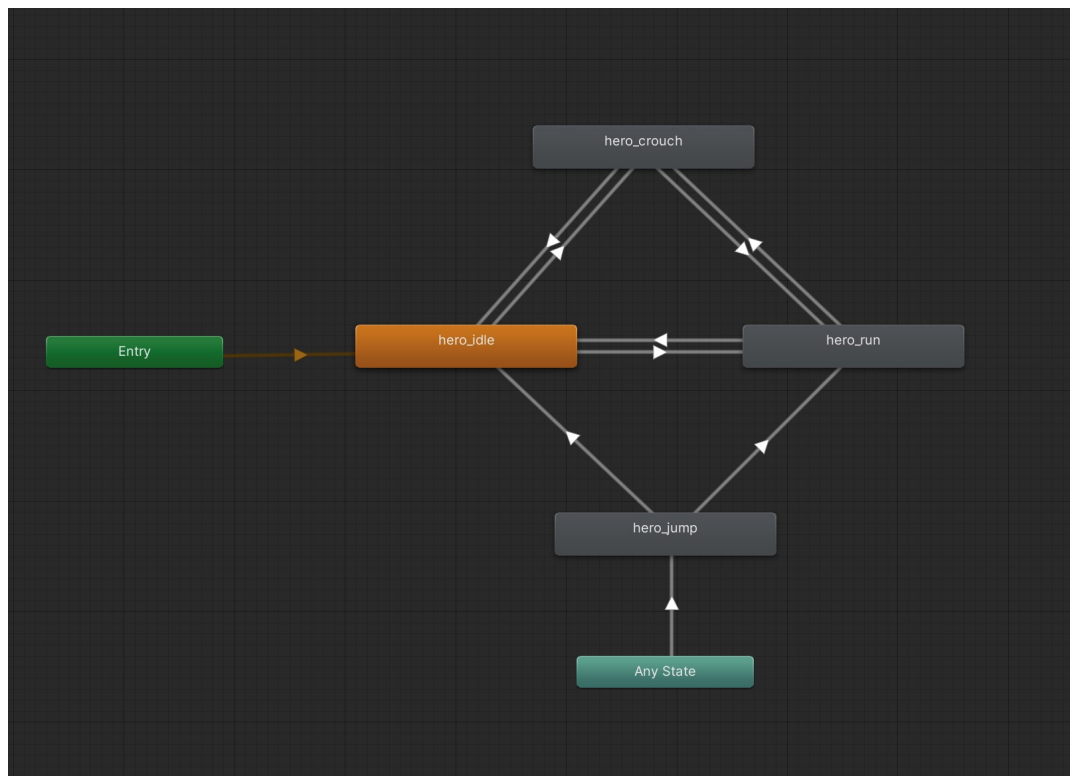
*In terms of optimizing performance, the singleton pattern is also essential. Singleton pattern ensures the single existence of the object. Since I have many generic managers to handle states like audio, screen management, and so on, ensuring no duplication of those managers could lead to increased memory usage and potential conflict when managing scenes. So, using singleton gives us globally accessible and single instance.*

*Typically, Singleton behavior is achieved by creating a static instance we access using its getter method. In this getter method, we check the existence of it, create it if it does not exist, and return the static instance if it does exist already. We should also make the constructor private to prevent creating new instances.*

*In Unity, MonoBehaviour classes don't have constructors in the traditional sense. This creates a necessity to work around to achieve Singleton behavior. Creating a static Instance variable within the class and assigning this in the Awake method to mimic the Singleton behavior. However, we need additional code to check whether an instance already exists. This requires expensive calls like `FindObjectOfType<DesiredManager>`. If this instance exists already, call `Destroy(gameobject)` to ensure single existence.*

### 3.4.3 Animations

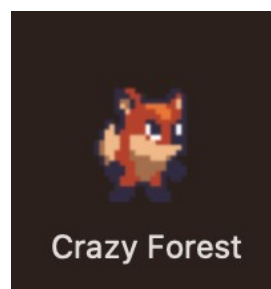
All of the animations are done with sprite sheets. I have 2D assets of characters for different scenarios such as jumping, running and crawling. Those sprites contain images that when it is seen in consecutive it feels like it is moving. All of the sprites were free assets from Sunnyland, Pixel Adventure 1 and 2. I have opened the animator controller for each object that I want to animate. In the animation controller, I place the sprites accordingly. In the controller it is created as states. By creating transitions between states and creating conditions in order to achieve this transition, it gives us a control scheme in the code level to execute the wanted animation.



*Figure 3: Animation State Machine*

### 3.5. Results

I have completed my game with a working server. I have built my game in macOS platform settings. I have created a shortcut icon on the desktop from that build, and I can launch my game without opening Unity.



*Figure 4: Crazy Forest Desktop Icon*



I haven't completed the authorization implementation of my game that we can unlock characters after certain levels. UI of the character selection screen is implemented, and in the server, authorization requests are handled. But in the game engine, I did not animate other characters, so I did not add them into the game as playable characters.

Also, I did not implement level mechanics to unlock them in the gameplay, but it is stored on the server.

The reason that I did not implement them is completely related to the period of my internship. I should create a more time-managed plan to include unlockable player characters in my game.

My project has qualitative metrics in terms of its gameplay mechanics, graphics, and animation level. It also has qualitative metrics in terms of the usefulness of CRUD operations in server-database relations.

## *Comments*

Comments from my coworker

Ibrahim Kılıç (Software Developer):

At the codewise level, I can say that Batuhan wrote a modular code that fits SOLID principles. Design patterns are well implemented, and they are used adequately. At the gameplay level, it can include more mechanics, such as camera shakes or moving platforms. The server implementation is simple enough, and CRUD operations are working. Overall, the game and the server are well done in a short period of time.

Comments from my project supervisor

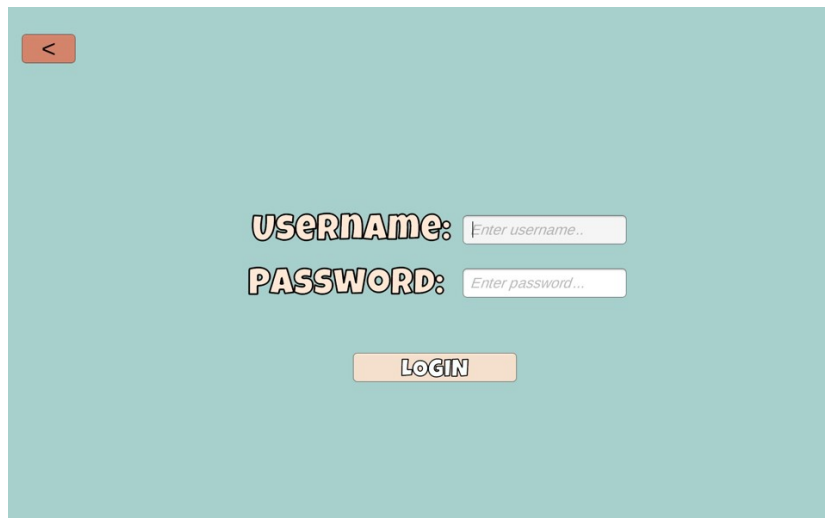
Gözde Ocal (Project Manager):

The gameplay level is very fun to play. The animations are looking smooth.

### *Screenshots from the Crazy Forest*

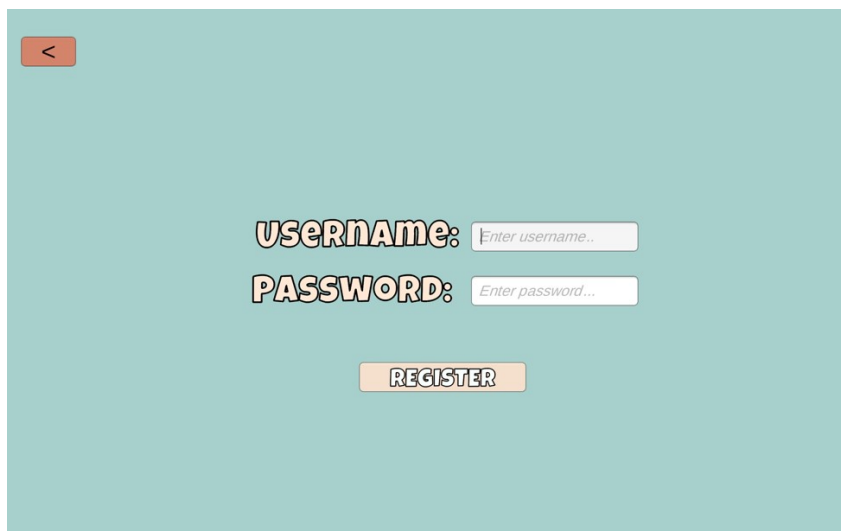


*Figure 5: Main Menu Screen*



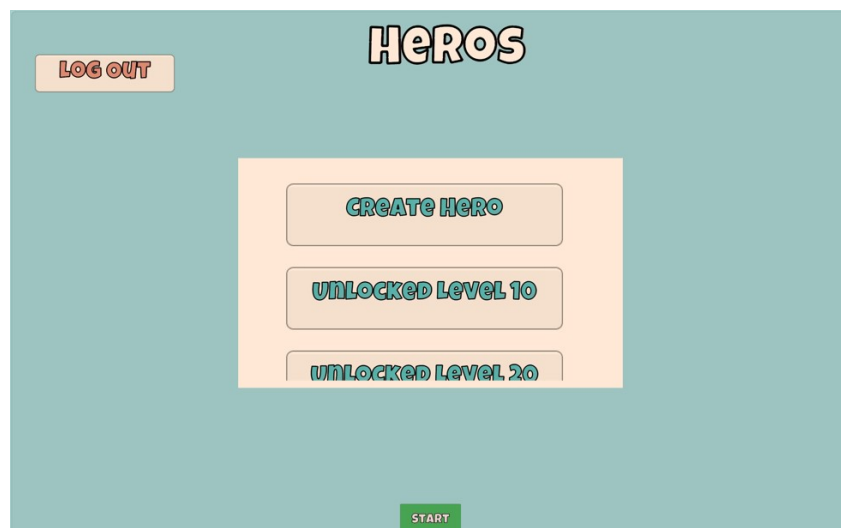
A login menu screen with a teal background. In the top-left corner, there is a red button with a white left-pointing arrow. Centered on the screen are the labels "USERNAME:" and "PASSWORD:" in a bold, black, outlined font. To the right of each label is a white text input field with a light gray border and placeholder text "Enter username.." and "Enter password.." respectively. Below these fields is a single orange button with the word "LOGIN" in black, bold, uppercase letters.

Figure 6: Login Menu Screen



A register menu screen with a teal background. In the top-left corner, there is a red button with a white left-pointing arrow. Centered on the screen are the labels "USERNAME:" and "PASSWORD:" in a bold, black, outlined font. To the right of each label is a white text input field with a light gray border and placeholder text "Enter username.." and "Enter password.." respectively. Below these fields is a single orange button with the word "REGISTER" in black, bold, uppercase letters.

Figure 7: Register Menu Screen



A hero selection screen with a teal background. In the top-left corner, there is an orange button with the text "LOG OUT" in black, bold, uppercase letters. At the top center, the word "HEROS" is displayed in a large, black, outlined font. In the center of the screen, there is a light orange rectangular container. Inside this container are three orange buttons stacked vertically, each with text in a teal, bold, uppercase font: "CREATE HERO", "UNLOCKED LEVEL 10", and "UNLOCKED LEVEL 20". At the bottom center of the screen, there is a small green button with the word "START" in white, bold, uppercase letters.

Figure 8: Hero Selection Screen

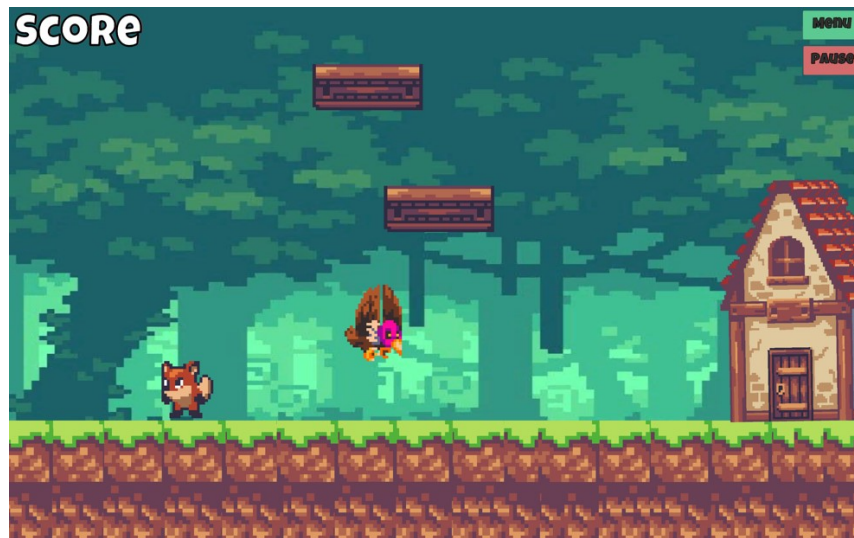


Figure 9: In Game Screen Starting Point

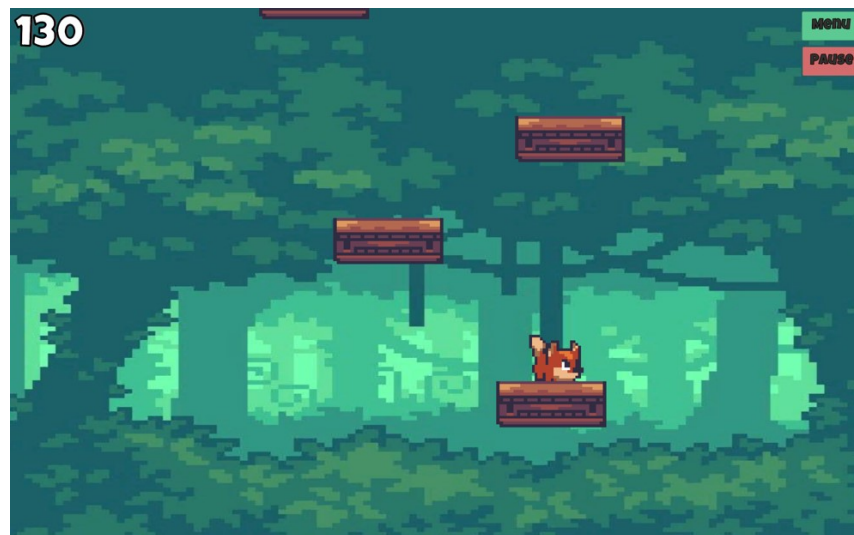


Figure 10: In Game Screen Crouching Hero

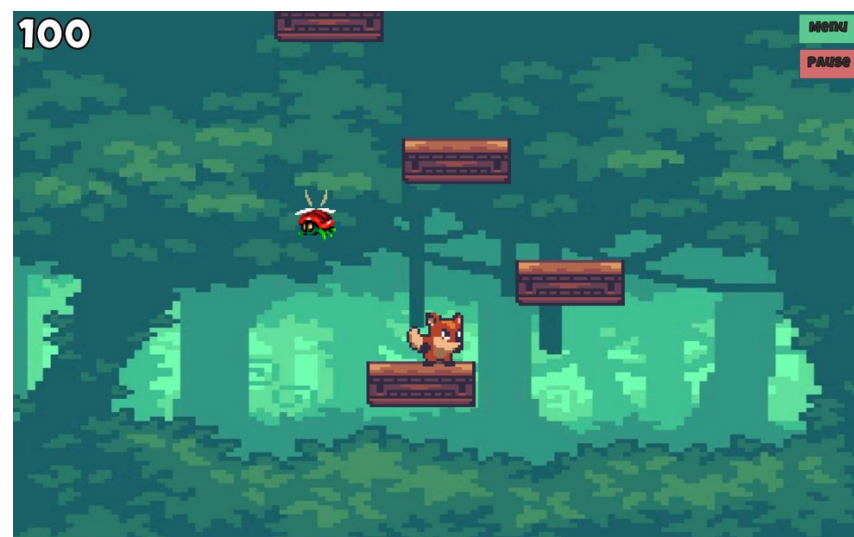


Figure 11: In Game Screen Beetle

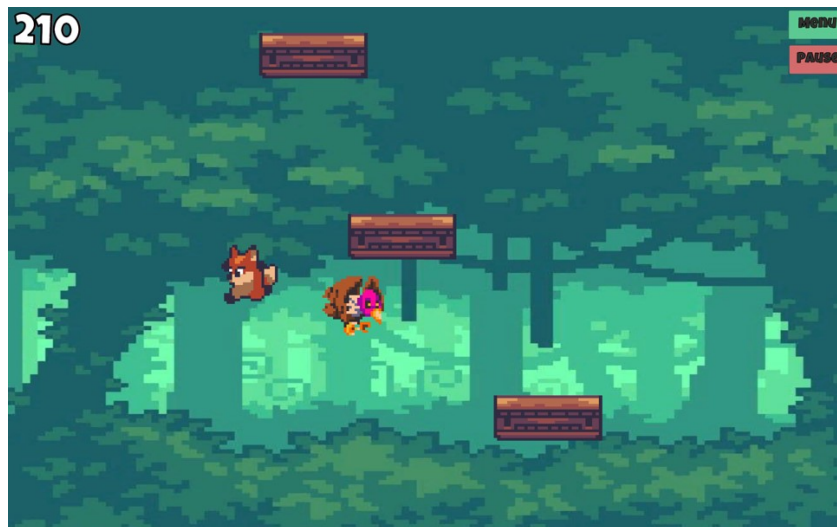


Figure 12: In Game Screen Vulture Attack

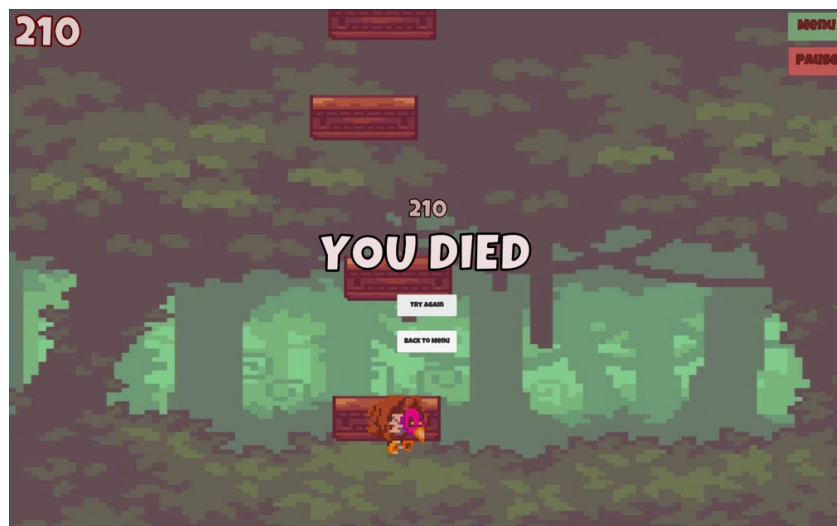


Figure 13: In Game Screen Die

## 4. Conclusions

My internship at Crs Soft was a great chance to apply what I learned in my courses at university, like COMP302, which is Software Engineering, and COMP306, which is Database Management. COMP302 taught me about design patterns, and during my internship, I got to use them in different scenarios. Patterns that I have learned in Comp302 won't suit Unity's logic, so I have tailored some of them to fit Unity's logic. I have also learned new design patterns that I didn't know from Comp302. This course also helped me in planning and implementing my project.

I use the database knowledge that I learned from Comp306 a lot. I usually write SQL queries, but in this project, I used an object-relational mapper, which was a new approach for me. I could tell if the tables it created were efficient, thanks to what I learned in COMP306.

For the next semester, I am planning to take COMP416, Computer Networks. I want to learn the logic of multiplayer games. I'm also interested in the COMP430 Privacy and Security course because while designing the server, I realized how important security is. Designing a server gives me new perspectives that I have never thought about when I was just the user. Now, I want to learn more about creating better security layers.

This internship was a great way to learn Unity and get started with game development. I also learned C# got to use Azure DevOps Server for version control. I enjoyed working with a helpful and friendly team at Crs Soft. The supportive environment made this internship enjoyable and helped me see how I can move forward in game development.

In conclusion, my internship at Crs Soft introduced me to game development, which I was eager to step into. Now I have gained knowledge and experience in this sector, I am considering continuing in this field. The experience of working with supportive people and learning new technologies was valuable. I am looking forward to building up the knowledge that I gained from Crs Soft.

## Appendix

<https://www.crssoft.com/brands>

## References

Crs Soft History:

<https://sikmakas.com.tr> and <https://www.crssoft.com/newsroom>

Figure 1: Crs Soft Logo <https://www.crssoft.com/e-dunya>

Figure 2: Corvus Jay Logo <https://www.crssoft.com/corvus-jay>

Figure 3: Crazy Forest Desktop Icon

Figure 4: Unity Animation Controller

Figure 5,6,7,8,9,10,11,12,13 Crazy Forest Game Screenshots