Countdown Adventure

Game Development Case Study

Project Overview

Countdown Adventure is a 2D platformer developed as a personal game development project. The game was created to explore game design, programming, user interface design and interactive media while developing a complete project from an initial concept through to a finished, submitted product.

The game takes inspiration from colourful platformers and arcade-style games from the early 2000s, combining simple platforming mechanics with a playful visual style.

The final version was developed in **Godot 4 using C#**, with the game designed for both desktop/web and Android.

Project type: 2D platformer
Engine: Godot 4.6.2 .NET
Programming language: C#
Platforms: Web, Android
Development: Solo project

1. Initial Concept

The original idea for Countdown Adventure was to create a small platforming game inspired by the games I enjoyed growing up.

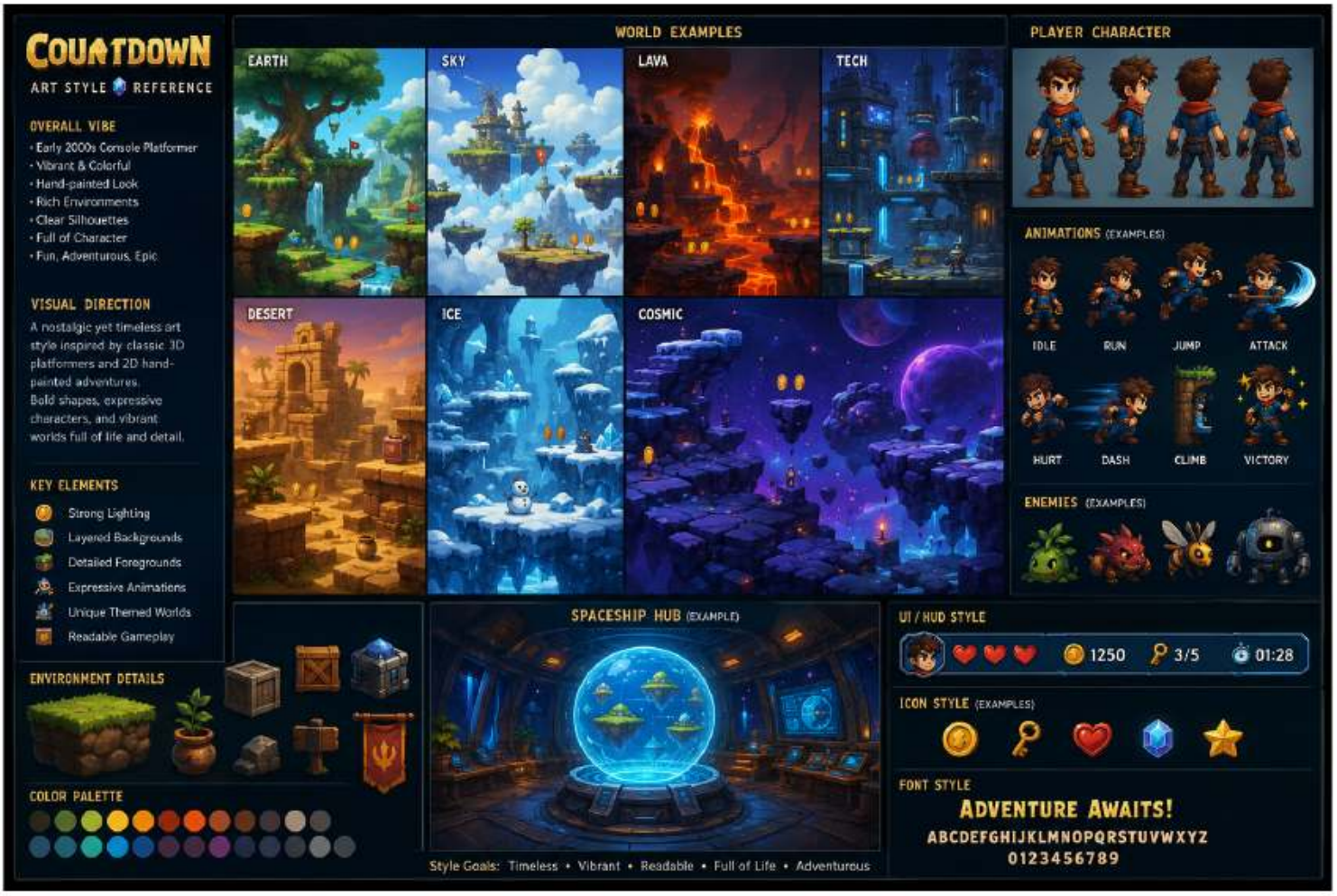
I wanted the game to have an accessible control system while still including enough mechanics to make each level feel different. The visual direction was influenced by the colourful, exaggerated style of early 2000s platform games.

The original concept developed around multiple themed worlds, each introducing different environments and hazards.

The final game contains four main environments:

- Earth
- Sky
- Tech
- Lava

Each environment was designed to introduce a different visual atmosphere and gameplay challenge.



2. Planning and Game Design

Before developing the game, I considered the core gameplay loop and the features that would make the project feel like a complete game rather than simply a technical demonstration.

The main gameplay loop became:

Explore → Avoid hazards → Collect items → Reach the end of the level

The game was designed around short, focused levels so that the player could quickly understand the mechanics and progress through the game.

Some of the planned features included:

- Player movement
- Jumping
- Hazards
- Collectibles
- Health
- Checkpoints
- Enemies
- Boss encounters
- Multiple environments
- Level progression
- Pause and settings menus
- Mobile controls

During development, I also experimented with additional systems such as a central hub area, character customisation, a coin shop and hidden rewards. These systems were ultimately removed from the first release so that I could focus on completing and polishing the core gameplay experience.

3. Choosing the Technology

I initially experimented with developing the game using MonoGame and C#, but later moved the project to Godot.

Godot provided a more suitable environment for the project because it allowed me to work with scenes, nodes, animation, physics and UI systems while still allowing me to use C#.

Using C# also allowed me to build my programming skills through a practical project rather than working exclusively on small exercises.

The main development tools were:

Godot
Used to create the game, levels, scenes, UI and physics systems.

C#
Used to program gameplay mechanics and systems.

Web export
Used to create a playable browser version of the game.

Android export
Used to create the mobile version for submission.

4. Programming the Player

One of the first major systems I developed was the player controller.

The player needed to be able to:

- Move left and right
- Jump
- Interact with the environment
- Take damage
- Lose health
- Become temporarily invincible after taking damage
- Die when health reaches zero

I created the player system in C# and used Godot's CharacterBody2D system to handle movement and physics.

A simplified version of the gameplay logic was based around:

Input → Movement → Physics → Collision → Gameplay response

This became the foundation for the rest of the game because almost every level mechanic interacts with the player.

[SCREENSHOT: Player controller / player in-game]

5. Creating Hazards and Enemies

Once the player controller was working, I developed environmental hazards.

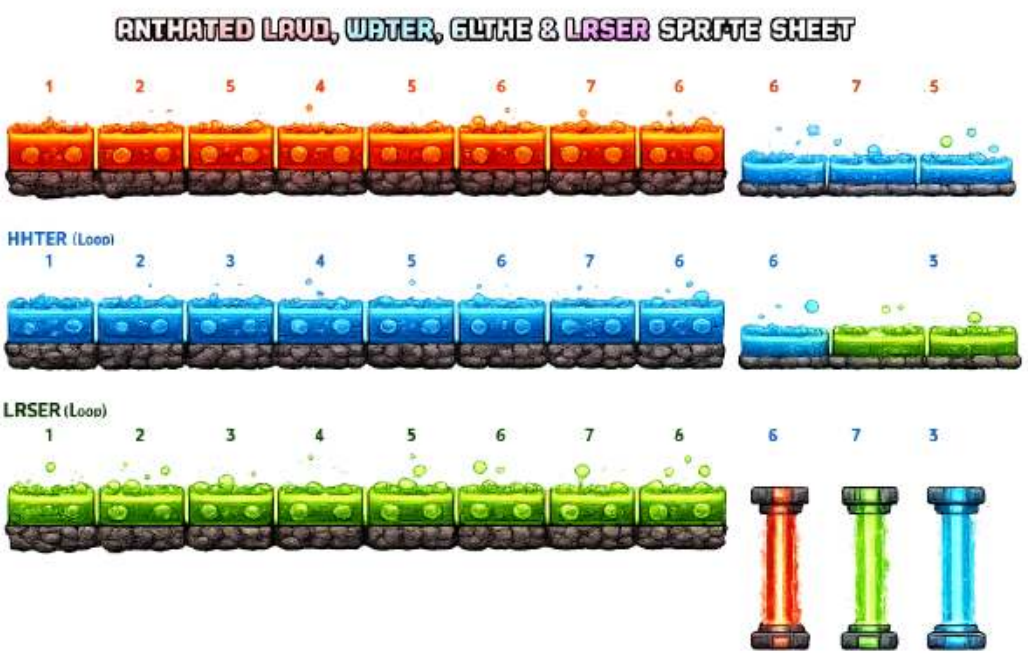
Different hazards were used to make the environments feel distinct.

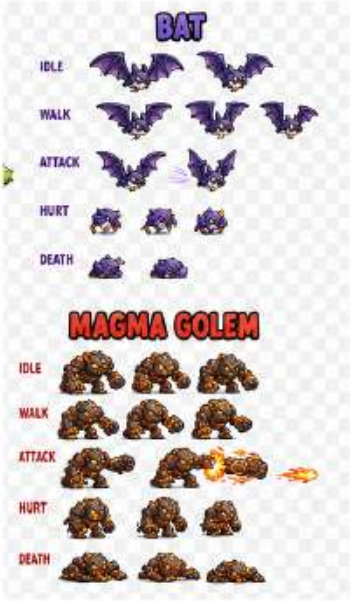
Examples included:

- Lava
- Water
- Slime
- Lasers
- Environmental obstacles

Enemies were introduced into selected environments, particularly the Tech and Lava areas.

The purpose of these systems was not simply to make the game harder, but to encourage the player to pay attention to their surroundings and change how they approach each section of a level.





6. Collectibles and Health

I added collectibles and health mechanics to give the player additional reasons to explore the levels.

Coins were used as collectible items, while hearts represented health.

The player begins with a limited amount of health, meaning that collisions with hazards and enemies have consequences.

This created a simple risk-and-reward system:

Explore more → Find collectibles → Take greater risks

The health system also required the player to respond to damage rather than simply restarting immediately after every mistake.

7. Checkpoints and Level Progression

Checkpoints were introduced to make the levels more forgiving. Without checkpoints, reaching a difficult section and then losing all progress could become frustrating.The checkpoint system allows the player to continue from a more recent position after losing a life.I also developed level progression so that the game could move between different scenes as the player completed levels.

8. User Interface

The user interface was designed to communicate important information without distracting from the gameplay.

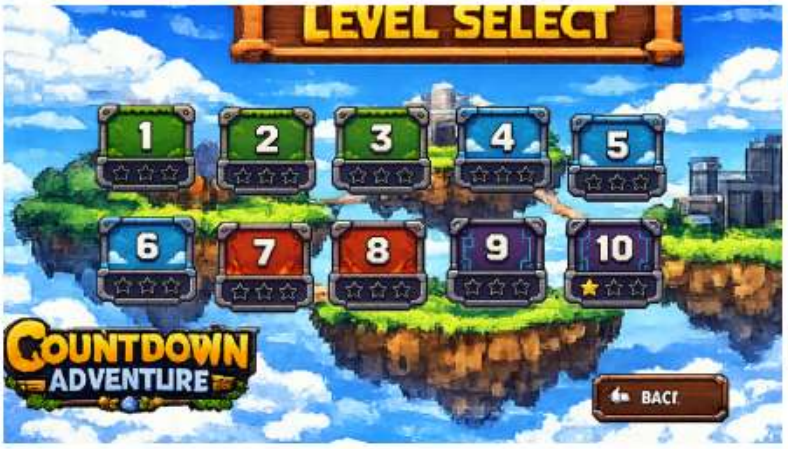
The HUD displays information such as:

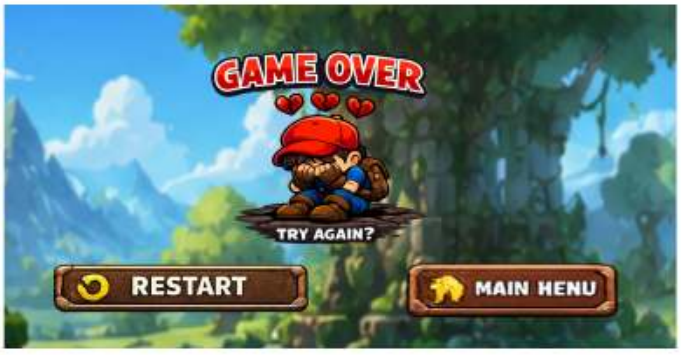
- Health
- Coins
- Gameplay status

I also created menus for functions such as pausing the game and changing settings.

Developing the UI was particularly useful because it required me to consider both programming and visual design.

The interface had to be functional, readable and consistent with the game's visual identity.





9. Mobile Development

One of the biggest challenges was adapting the game for Android.

Desktop games can rely on keyboard controls, but a mobile version needs an interface that works with touch input.

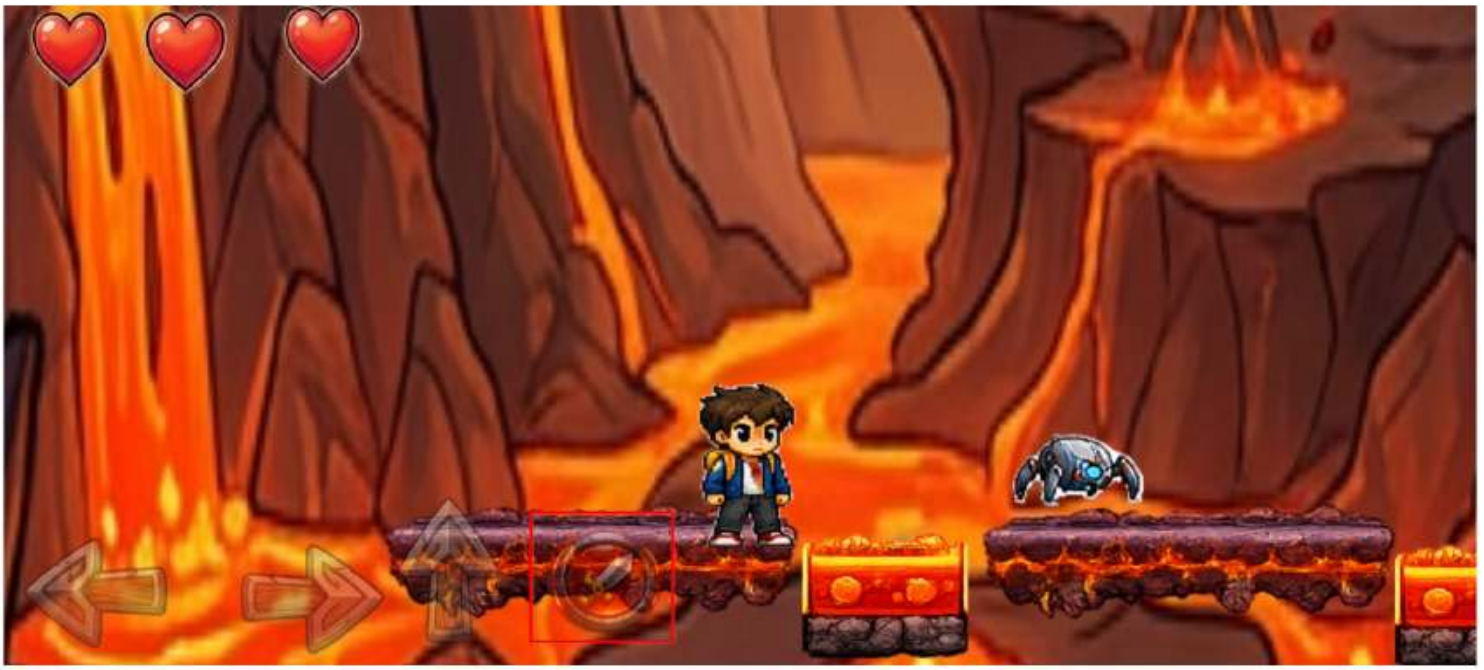
I therefore created mobile controls for:

- Left
- Right
- Jump
- Attack

The controls were connected to the gameplay systems so that the same core player mechanics could be used with a different input method.

Testing the Android version revealed several problems that were not obvious when playing the desktop version.

This was an important part of the project because it showed me that supporting another platform is not simply a matter of exporting the game. The interface and interaction model also need to be considered.



10. Problems and Debugging

A significant part of the development process involved debugging.

Some of the problems I encountered included:

- C# compiler errors
- UI elements not appearing correctly
- Player state not being maintained between scenes
- Save system and autoload issues
- Mobile controls not responding correctly
- Android export problems
- Gradle errors
- Signing and keystore issues
- Web export problems
- Scene-loading issues

Rather than treating these problems as separate failures, I used them as opportunities to understand how the different systems in Godot interacted.

For example, when the HUD failed to load correctly, I had to investigate the scene structure and how UI nodes were being instantiated.

Similarly, when the save system was not functioning correctly, I discovered the importance of configuring persistent systems correctly within Godot.

Debugging became one of the largest parts of the project and significantly improved my understanding of software development.

11. Reducing the Scope

One of the most important decisions I made was reducing the scope of the game before release.

At one stage, I had planned additional features including:

- A headquarters area
- Character skins
- A wardrobe
- A coin shop
- Hidden secrets
- Additional rewards

Although these features were interesting, they were not necessary for the core game to function.

For the first release, I decided to remove these systems and concentrate on the main platforming experience.

This allowed me to prioritise:

Playable levels → Reliable mechanics → Working UI → Platform testing → Release

This was an important lesson in project management.

A smaller finished project is more valuable than a larger project that remains unfinished.

12. Testing

Testing took place throughout development rather than only at the end.

I repeatedly tested:

- Player movement
- Jumping
- Collision
- Hazards
- Enemy interactions
- Health
- Checkpoints
- Level transitions
- UI
- Menus
- Mobile controls
- Web export
- Android export

Testing across different platforms also revealed issues that would not have appeared during development on my main computer.

This helped me understand the difference between **development testing** and **real-world use**.

A feature can work correctly in the editor but behave differently once the game has been exported.

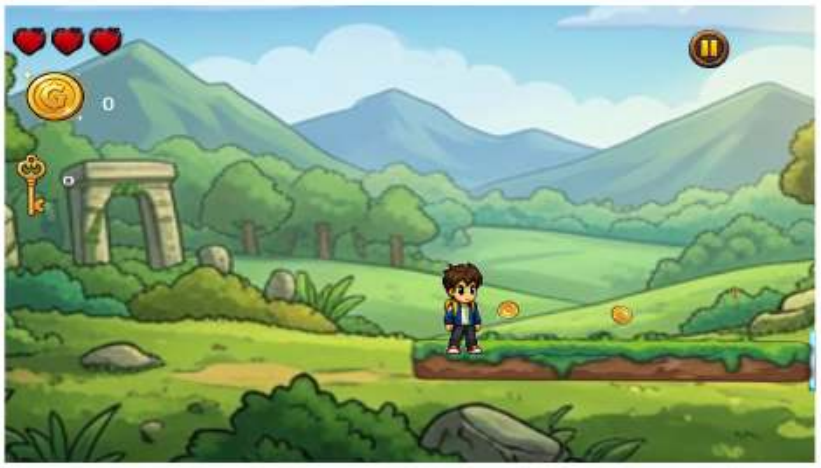
13. Final Product

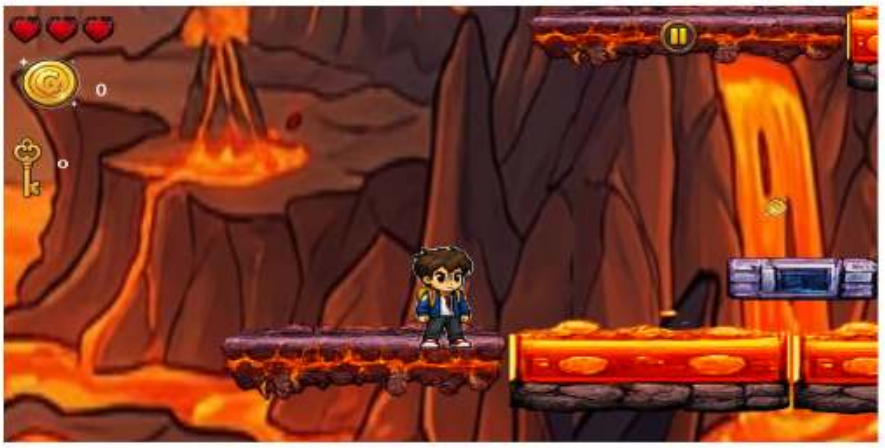
The final version of Countdown Adventure is a playable 2D platformer featuring multiple themed environments, hazards, enemies, collectibles, health, checkpoints, menus and mobile controls.

The game was exported for web and Android and submitted for release.

The final version represents a significant change from the original concept because the project became more focused during development.

Rather than trying to include every feature I had imagined, I prioritised creating a complete and playable experience.





14. What I Learned

Developing Countdown Adventure taught me skills across several areas of digital development.

Programming

I gained practical experience with:

- C#
- Object-oriented programming
- Game physics
- Collision detection
- Input systems
- UI programming
- Scene management
- Game state
- Debugging

Game Development

I learned about:

- Level design
- Gameplay loops
- Difficulty
- Player feedback
- Platform-specific controls
- Testing

Project Management

The project taught me how to:

- Break a large idea into smaller systems
- Prioritise features
- Debug problems independently
- Reduce scope when necessary
- Test a product across platforms
- Prepare a project for release

Design

I also developed my understanding of:

- Visual consistency
- User interface design
- Colour and environment design
- Player experience
- Creating a recognisable visual identity

15. Reflection

Looking back at the development process, the most valuable part of the project was not simply creating a playable game. It was learning how many different disciplines are involved in producing an interactive product.

A game combines programming, design, art, user experience, testing and project management.

There were several points where the project became technically difficult, particularly when dealing with exports, Android development and communication between different scenes and systems. However, solving these problems gave me a much stronger understanding of how software works beyond simply writing code.

The decision to reduce the scope was also an important learning experience. I realised that good development involves making decisions about what *not* to build.

Countdown Adventure therefore became more than a game project. It became a practical demonstration of my ability to take an idea, develop it into a working product, identify problems, adapt the project and ultimately prepare it for release.

16. Final Outcome

Countdown Adventure was completed and submitted as a finished game project.

The project demonstrates my interest in combining **programming, technology and creative design**, and has given me practical experience of taking a digital product through the development process from concept to submission.

It is also a foundation for future projects, where I can build on the technical and creative skills developed during this project.

Final project status: Completed and submitted. 🎮