About me

Hi, I'm Alexander Hennig, an indie game developer from Germany.
I'm currently working on my first major game project, Ruins of Blackspire, a 2D strategy dungeon crawler focused on tactical combat and resource management. This portfolio walks through my design thinking, the systems I built from scratch, and some of the challenges I faced along the way.

I really like RTS games, hence why Ruins of Blackspire has many mechanics in line with that. It's nice to be able to choose how you want to approach any given situation with the tools you have available, which is a feeling I really like to replicate in my game. But I also really like RPGs and shooters.

For Ruins of Blackspire I used Godot because it seemed like a very good starting point for me to learn more in-depth programming compared to something like Unreal or Unity which I used briefly, but both never really caught me for as long as Godot did.

I'm very much a generalist, no matter if we are talking about making games, playing games, or anything else. I really like variety in my tasks, and I thrive when working across disciplines and enjoy solving problems in multiple areas.

That said, I found myself especially drawn to designing complex systems like inventories, spell logic, and core gameplay loops. These systems challenge me the most and are very important to plan through so they don't break even under constant use of the players.

If you are interested you can reach out via E-Mail or Linkedin.

E-Mail: alexanderhennigbusiness@outlook.de
Linkedin: http://linkedin.com/in/alexander-hennig-309b00223

Ruins of Blackspire available on:
Itch.io: https://pixelbrew.itch.io/ruins-of-blackspire
Steam (Demo): *https://store.steampowered.com/app/3714780/Ruins_of_Blackspire_Demo/*

Ruins of Blackspire

“Ruins of Blackspire“ is a top-down, 2D, Dungeon Crawler with RTS inspired gameplay. You pick a class and go through a dungeon full of enemies, items and abilities.

Genre: Dungeon Crawler, Rogue-like
Engine: Godot
Team size: 2
Duration: December 2023 - July 2025
Platform: PC (Steam)

Roles and responsibilities:

Game Design:
- Designed the UI to support full mouse-only control, allowing players to play without a keyboard
- Developed slow-paced, RTS-inspired combat to emphasize tactical decision-making.
- Balanced unit design around high health and low damage to encourage deliberate “resource management” due to scarce healing opportunities.

Programming:
- Created a scalable inventory system that supports equippable items, consumables, and spells.
- Enabled easy creation and implementation of new items/spells.
- Focused on strong internal communication between inventory systems for features like cooldown tracking and spell slot behavior.

Art:
- Directed and coordinated with contributing artists, providing feedback and guidance to match the tone and needs of the game.
- Selected and implemented third-party asset packs where needed.

Marketing & Publishing:
- Managed the entire social media presence of the game.
- Handled all tasks related to Steam publishing: store page, trailer, tags, and everything else.
- Almost daily updates to the game published on GitHub

Design thoughts

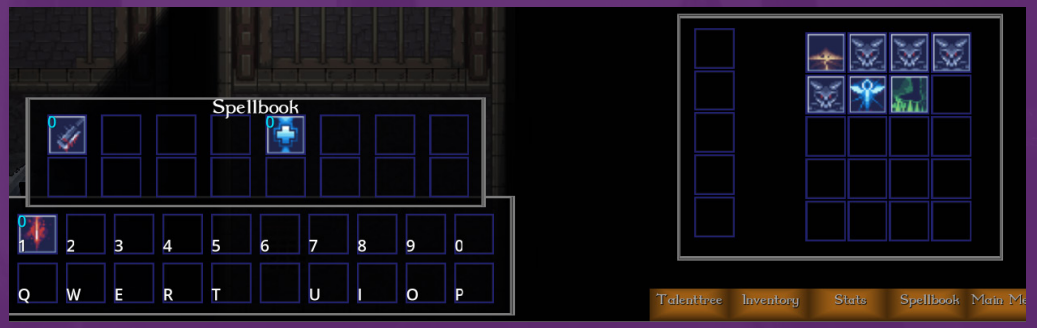
Item & Spell Inventory System

Designed a scalable inventory that allows me to have a lot of different items be inserted from different sources, and allows for flexible usage across different parts of the game

Design Process:
The original plan was for the player to have 20+ spell slots that can all have a spell attached to it. So I need a scalable inventory system that allows me to adjust it based on how many spells a player could access, which would vary with level progression.

I then used that spell inventory to basically create a “bag” for the player to put in consumables or spell tomes so they can use those items whenever they want and have a tooltip explaining what they just picked up instead of it being used immediately.

Since both inventories worked with the same code base, I could easily add things like drag and drop or interactivity to both and not worry about errors through missing types for example when moving something from one inventory to the other.



Retrospective:

This was the first major system I fully built while still learning Godot and GDScript, so the initial implementation had several rough edges.

One key challenge was handling cooldowns: originally, the cooldown logic was tied to the slot itself. That meant if a spell was moved mid-cooldown, the cooldown also had to be tracked and transferred, something I didn’t consider early on.

Another issue was scaling: as I added more features, I reused the system in ways I hadn’t planned for. This sometimes required copy-pasting new functionality into every version of some systems, which became time-consuming

When I do this again, I have a better understanding of what I need my systems to do before making a system that needs multiple rewrites and has a really messy code.

Pathfinding

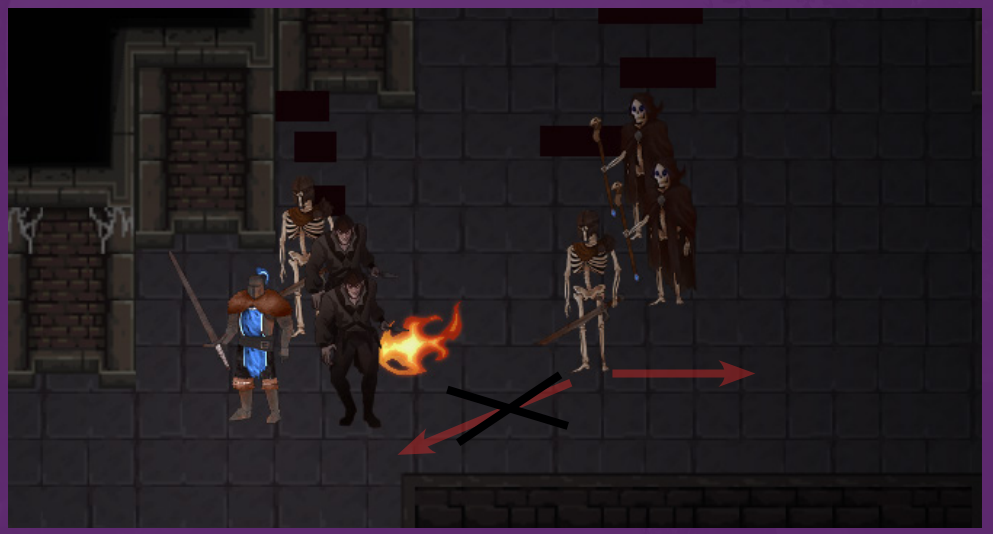
Implemented a pathfinding system that allows all units to detect other units and static obstacles and dynamically navigate to targets around those obstacles.

Design Process:

For a game inspired by RTS mechanics, responsive and reliable pathfinding is critical to the player experience. From the start, I knew this would be one of the most time-consuming systems to research and implement.

Over the course of 3–5 months, I iterated heavily on this system. My goal was to have it detect all types of path blockers that I might put in and calculate paths dynamically.

The system I have in the game right now allows units to path around others and navigate around obstacles, but not with the precision I was planning for.



Retrospective:

One of the main challenges was getting the system to recognize that a location was blocked by another unit and make the moving unit react accordingly. Since units were constantly moving, I had to find a way to mark a path as blocked and then update it when a unit is moving.

In the end, I created a solution where units generate temporary collision data that influences the nav mesh and triggers a rebake at regular intervals. This workaround allowed me to simulate “occupied” tiles without relying on default navigation obstacles, which didn’t behave as expected.

This along with a lot of number tweaks, allowed me to have all my units take paths while recognizing each other when it matters (f.e. when surrounding the player). There are some performance concerns I have with interval rebaking, but that is something I will be improving when I do more iterations on this system.

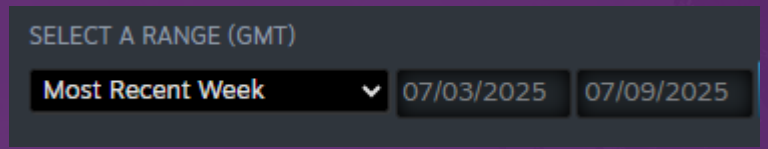
Demo Launch

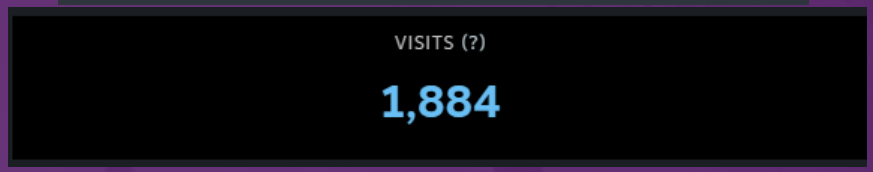
It took a lot of learning to get the game ready for its first release on Steam. Especially setting up everything, such as the trailer and art in the proper proportions and all the other things that must be on a Steam page.

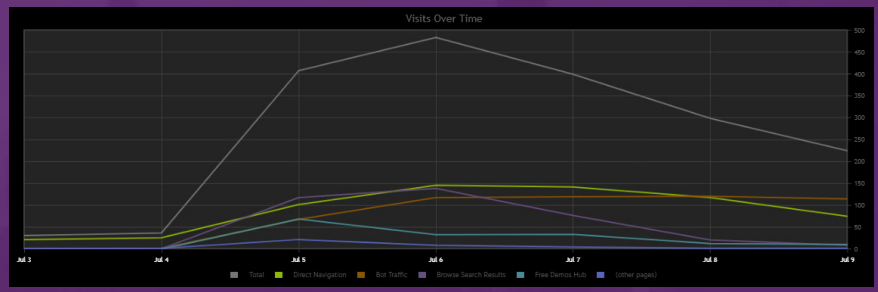
The site was already out in May and gained a decent amount of traffic with a couple hundred impressions a day. Which at the time was pretty big considering the only „marketing“ I did was some development streams.

After the demo was actually out over the first week, there was a big spike in traffic with almost 2000 Steam Page visits without any active marketing.

It was interesting to see that it took about 2 days before that big traffic spike happened.







Thoughts:

These numbers are not very meaningful. But for the scope of the Demo it was still interesting to see how the numbers grew over time especially since the growth is very organic, there was no outside marketing so 90% of traffic comes from Steams search and discovery functions.