



Ignacy Szczekot

programming & game design

Hidden Realm

A 2d adaptation of a popular oldschool MMORPG, Metin2.

Preview includes game code(Unity, C#, MySQL, SQLite), registration website(Wordpress), and dedicated server infrastructure(Linux Debian 10).

Type of game:	MMORPG
Author's name and surname:	Ignacy Szczekot
Idea(clone or original):	Clone
Date of writing the preview:	09-07-2020
Date of production:	June, July of 2020

Introduction

This is a short description of a preview/demo version of a 2d game clone I have been making for the last few weeks.

Original title, **Metin2**, originally developed by Ymir Entertainment, played a huge part in my childhood. It was hugely popular title in Europe back in late 00s and early 10s. It sparked in me the love for MMO games, and I wanted to create one since then. I wanted to test my skillset against network code, databases and server infrastructure, which I've never done myself from start to finish before this project (obviously I tried designing and programming multiplayer games before). In the preview build I tried to focus on core game mechanisms – original game is quite vast.

I decided on making a clone rather than an original title based purely on time I could dedicate for the development. I know the original very well, so I could focus mostly on programming familiar mechanics (from scratch).

To sum up, the goal of this project was to learn basic MMO design principles and test implementing whole game from start to finish.

Languages and Frameworks

1. Game Engine

I chose **Unity Engine** because I know its editor quite well. Also, in my opinion it's really good for designing UI. In terms of physics, the game actually doesn't demand much – Unity has all the bases covered (and even more). Implementing some basic animations and pretty much every other needed game element is also quite simple. Besides that, few of mine programmer friends are also familiar with Unity, so it's very quick to show and discuss some elements with the build-in cloud project sharing feature.

2. Main language

Choosing Unity pretty much imposed selecting **C#** as the language of choice – which I am more than fine with. I agree that it's not absolutely cutting edge in terms of performance, but I think at my experience level the language choice is not the bottleneck of optimization issues. Also, absolute optimization was never the goal of this non-commercial project to begin with.

3. Multiplayer API

For networking Client-Server solution, I decided on **Mirror** multiplayer High Level API (HLAPI). I was actually debating between doing that and creating a networking solution from scratch in C#, but ultimately set on Mirror, because it's integrated with Unity out of the box and therefore saves so much time. It's worth adding that it is actually open source, so I could always tweak few things here and there.

4. Database

In terms of implementing the database, I settled on **SQLite** – a C-language library that implements small, self-contained and full-featured SQL database. From my experience its very good and reliable for implementing small databases. I've already had some practice with using it alongside python and C#. It's worth noting that my database needs to be accessed from two main sources – account registration website, and game server client itself. Both of these instances I cover by including SQLite library to C# programs, and accessing the database through **mySQL** queries.

5. Server(VPS/dedicated)

When it comes to server machine, I chose **Linux** operating system because it's simply more reliable for such appliances. For tests on my own virtual machine(guest machine on **VirtualBox**) I run **Debian 10**, but any stable distribution would work, really. I took a very minimalistic approach with having database(**SQLite**), game server(**Unity – Linux build**) and also www server(**Apache2, MariaDB**) on the same machine running at the same time. It made it easier and faster to implement communication between the three. I realize, that on a bigger scale production the servers are distributed on many virtual machines hosted on many physical ones. I didn't see such division necessary. If I were to make the game open for public, I would just replicate my server build on a properly hosted VPS through **ssh** protocol.

6. Website

For a simple website having registration form and a client download link I used **Wordpress** website builder. Reasoning behind it was that I knew the builder quite well already(I have designed one commercial site with it in the past, <http://www.logotorpeda.pl/>). It let me skip many parts of backend web development, which I have yet to learn. Nonetheless, I decided to make the registration form myself through **html**, and then communicate the user input with the database through a small **php** script and a **C#** program(I run C# on Linux through open source .NET framework, **Mono**). To hash passwords I used **BCrypt** algorithm with static and also dynamic salt, to increase security.

Programs

1. Unity Editor

One of the core reasons for choosing Unity. Powerful, reliable and simple to use.

2. Visual Studio

My go-to IDE for .NET programming in Windows. One of the absolute best tools in that domain. Automatically integrates projects with Unity. All of my C# programming was done in this program.

3. Notepad++

Basic program to edit any text file or small amount of code.

4. VirtualBox

Virtualization software I used for creating the server embedded in Windows 10 as a guest Debian 10 machine.

5. DB Browser for SQLite

Basic graphical desktop software for managing the database. I used it to create tables, establish field rules, assign dependencies, and browse records added or edited via programs(registration c# program and game server program).

6. Other

Among other programs I used for essentially server administration were **Apache2** HTTP Server for establishing a web server, **Samba** for file sharing with the server virtual machine. I also planned on using **cron** accompanied by **bash** scripts for database backup system(I have some experience with it, just didn't manage to find the time to set those up yet).