Sections and Chapters

Anton Nørgaard

November 26, 2023

Contents

0.1 Introduction

Chapter 1

Prerequisites for compiling and running the game

This chapter details all the necessary requirements for setting up, preparing and running the game.

It wort noting that the game can **only** run on Linux operating systems

Getting the correct compiler

The game was and should be compiled using GCC. To my recollection, it uses GCC-specific, non c-standard features, so it is unlikely it will work otherwise.

All of this of course, assumes you have sufficient privileges to install software on the desired host.

Installing GCC via Ubuntu

Firstly, run

```
sudo apt update
```

To update the packages list. Next, run

```
sudo apt install build-essential
```

This actually installs a group of different tools, most of which you will need anyway to compile the project, like make as well.

To verify GCC has been installed, run

```
gcc --version
```

This should print out the version of GCC in use.

Additional required software

Installing ncurses

The code uses the ncurses library to draw all the characters on the screen. It is therefore necessary in order to compile the code.

Installing ncurses on systems with apt

You need to install two packages

1. libncurses5-dev (developer's libraries for ncurses)
2. libncursesw5-dev (developer's libraries for ncursesw)

To install both, run

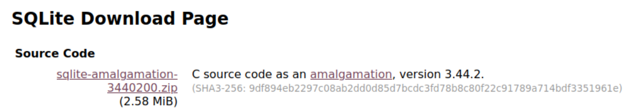
```
sudo apt-get install libncurses5-dev libncursesw5-dev
```

Installing SQLite

Revenant also uses SQLite to manage game state. Furthermore, it needs several other components for allowing compilation itself. There are actually several valid approaches for integrating, compiling and running SQLite, but this particular setup is based on what at the time seemed the best.

Getting the SQLite code

To get the SQLite source code, go to `https://www.sqlite.org/download.html`



SQLite Download Page

Source Code

sqlite-amalgamation-3440200.zip (2.58 MiB) C source code as an amalgamation, version 3.44.2.
(SHA3-256: 9df894eb2297c08ab2dd0d85d7bcdc3fd78b8c80f22c91789a714bdf3351961e)

Assuming you have wget

```
wget https://www.sqlite.org/2023/sqlite-amalgamation-3440200.zip
```

Chapter 2

Source code documentation

2.1 Management of game state and status

2.1.1 Keeping track of the creatures and the player

2.1.2 Checking for active effects of player

2.1.3 Creatures

In the following chapers, we will use several variables to compute various information regarding creatures in-game, their status and the way that they interact with the environment and the player. For that purpose, we will use the following defintions, some are explicitly computed/stored in-game, others are just definitions and are not explicitly declared in-game. The definitions we will use are

- min_local the smallest local coordinate value possible
- max_local the biggest local coordinate value possible
- $Player_{global\,coordinate}$ to represent a player's global coordinate, either X or Y coordinate
- $Creature_{global\,coordinate}$ to represent a creature's global coordinate, either X or Y coordinate
- The exact same variables are defined for the local coordinates
- max_moves is the maximum number of tiles a player/creature can move, before they move our of bounds of the current screen view. It is computed as (max_local - max_local) +1
- $Player_{set\,to\,min}$ is a subtraction of the players global coordinate s.t its local coordinate is the minimal. It is computed as $Player_{global\,coordinate}$ - ($Player_{local\,coordinate}$ - min_local)

- $Player_{set\,to\,max}$ is an addition of the players global coordinate s.t its local coordinate is the maximal. It is computed as $Player_{global\,coordinate}$ + (max_local - $Player_{local\,coordinate}$)

With that in mind, we can now go over the different mechanics that we need to consider for the creatures in game, such as spawning, movement, death, behavior, actions and so on

Spawning creatures

When it comes to spawning creatures, there are several things that we need to compute in relation to the player. One of them is the creatures local coordinates. The kicker about this is that while a player's local coordinates can be arbitrary without any hassle, the coordinates of the creature must be aligned relative to the players to that the view the creature has on screen matches that of the global coordinates. To do this, there are two cases that need to be considered, the trivial case and the nontrivial case.

In the trivial case, the distance between the player and the creature is s.t the distance for the particular axis, does not exceed the boundaries of the screen

In the non-trivial case, what we effectively do is reduce it to the nontrivial case. The way that we do this is by considering whether the axis-wise coordinate for the player is greater or smaller than the creature's just like in the non-trivial case. In the case that

$$Player_{global\,coordinate} > Creature_{global\,coordinate}$$

Then set the creature's local coordinate to

$$Creature_{local\,coordinate} = max\_local - (Player_{set\,to\,max} - Creature_{global\,coordinate}) \mod max\_moves$$

In the other case where

$$Player_{global\,coordinate} < Creature_{global\,coordinate}$$

Instead, set the creature's local coordinate to

$$Creature_{local\,coordinate} = min\_local + (Player_{set\,to\,min} - Creature_{global\,coordinate}) \mod max\_moves$$

The workings of this are a little hard to work around. Admittedly, this took rather a long while to figure out and I am unsure whether the computations can get simplified further. However, truth be told, I spent way too long to figure this out and so at this point that I cannot be bothered to come up with more clever solutions.

The way it works (informally) is as follows. Assume that

2.2 Controls

2.2.1 Keyboard Layout

a-m	n-z	A-M	N-Z+ESC
a - N/A	n - N/A	A - N/A	N - N/A
b - N/A	o - N/A	B - N/A	O - N/A
c - N/A	p - N/A	C - N/A	P - N/A
d - N/A	q - N/A	D - N/A	Q - N/A
e - equip item in inventory	r - N/A	E - Show equipped items	R - N/A
f - N/A	s - N/A	F - N/A	S - N/A
g - N/A	t - N/A	G - N/A	T - N/A
h - N/A	u - N/A	H - N/A	U - N/A
i - Display current inventory	v - N/A	I - N/A	W - N/A
j - N/A	w - N/A	J - N/A	X- N/A
k - N/A	x - N/A	K - N/A	Y - N/A
l - show past 10 events	y - N/A	L - N/A	Z - N/A
m - N/A	z - N/A	M - N/A	ESC - Quit command

Table 2.1: Keyboard bindings.

2.2.2 Character meanings on map

a-m	n-z	A-M	N-Z+ESC	Misc.
a - animal	n - N/A	A - N/A	N - N/A	@ - player character
b - N/A	o - N/A	B - N/A	O - N/A	! - interactable npc
c - N/A	p - N/A	C - N/A	P - N/A	
d - N/A	q - N/A	D - N/A	Q - N/A	
e - N/A	r - N/A	E - N/A	R - N/A	
f - N/A	s - N/A	F - N/A	S - N/A	
g - N/A	t - trader	G - N/A	T - N/A	
h - N/A	u - N/A	H - N/A	U - N/A	
i - N/A	v - N/A	I - N/A	W - N/A	
j - N/A	w - N/A	J - N/A	X- N/A	
k - N/A	x - N/A	K - N/A	Y - N/A	
l - N/A	y - N/A	L - N/A	Z - N/A	
m - N/A	z - N/A	M - N/A	ESC - Quit command	

Table 2.2: Meaning of characters on screen.