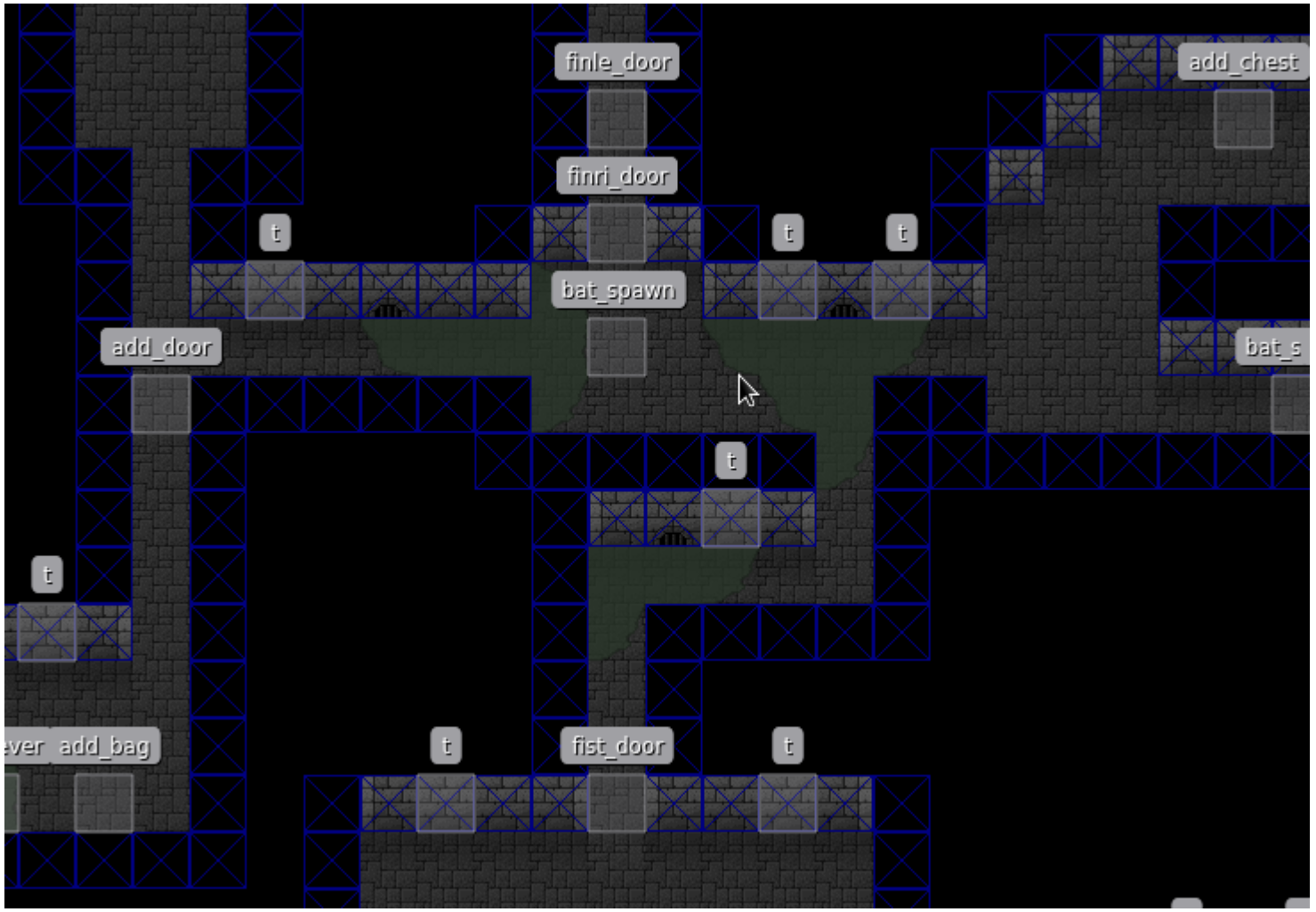
Dungeonz


finle_door
add_chest
finri_door
t
t
t
bat_spawn
add_door
bat_s
t
t
add_bag
t
fist_door
t

Biblioteki

Bez nich tak naprawdę nic by nie było

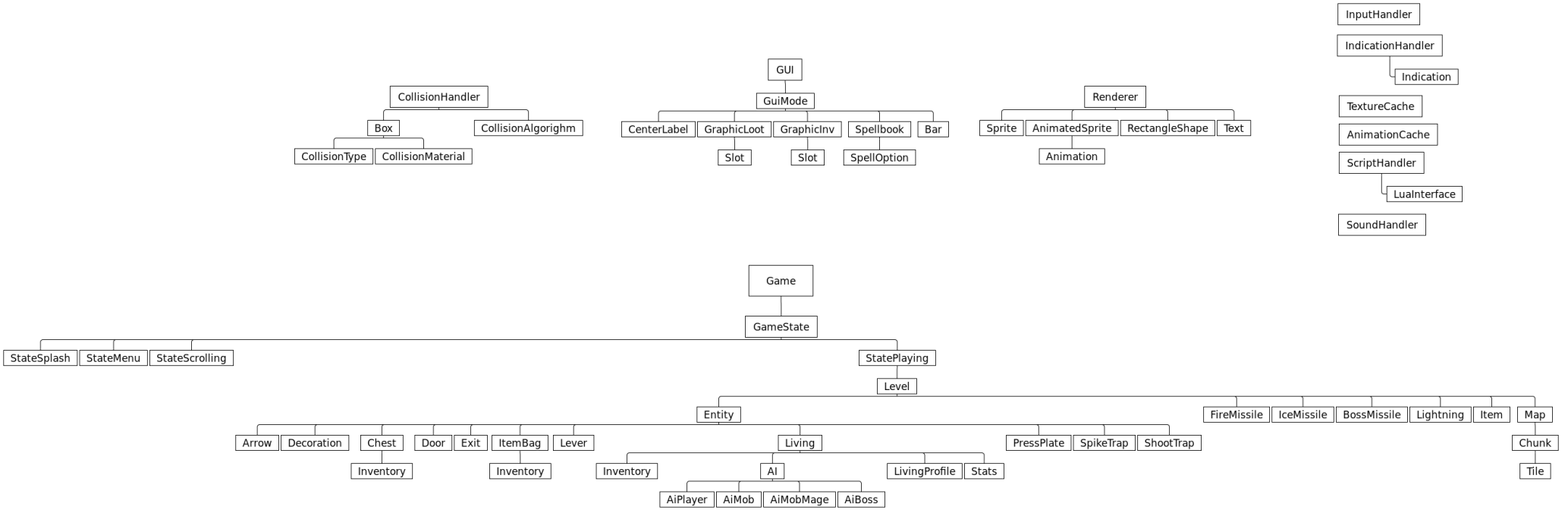
- SFML – Renderowanie, Input, Okno
- Lua 5.2 – Skrypty Itemów (tylko to)
- LuaBridge – Wrapper dla Lua 5.2
- RapidXml – Wczytywanie map z plików *.tmx* (*.xml*)
- Zlib – Dekompresja wczytanych plików

Narzędzia

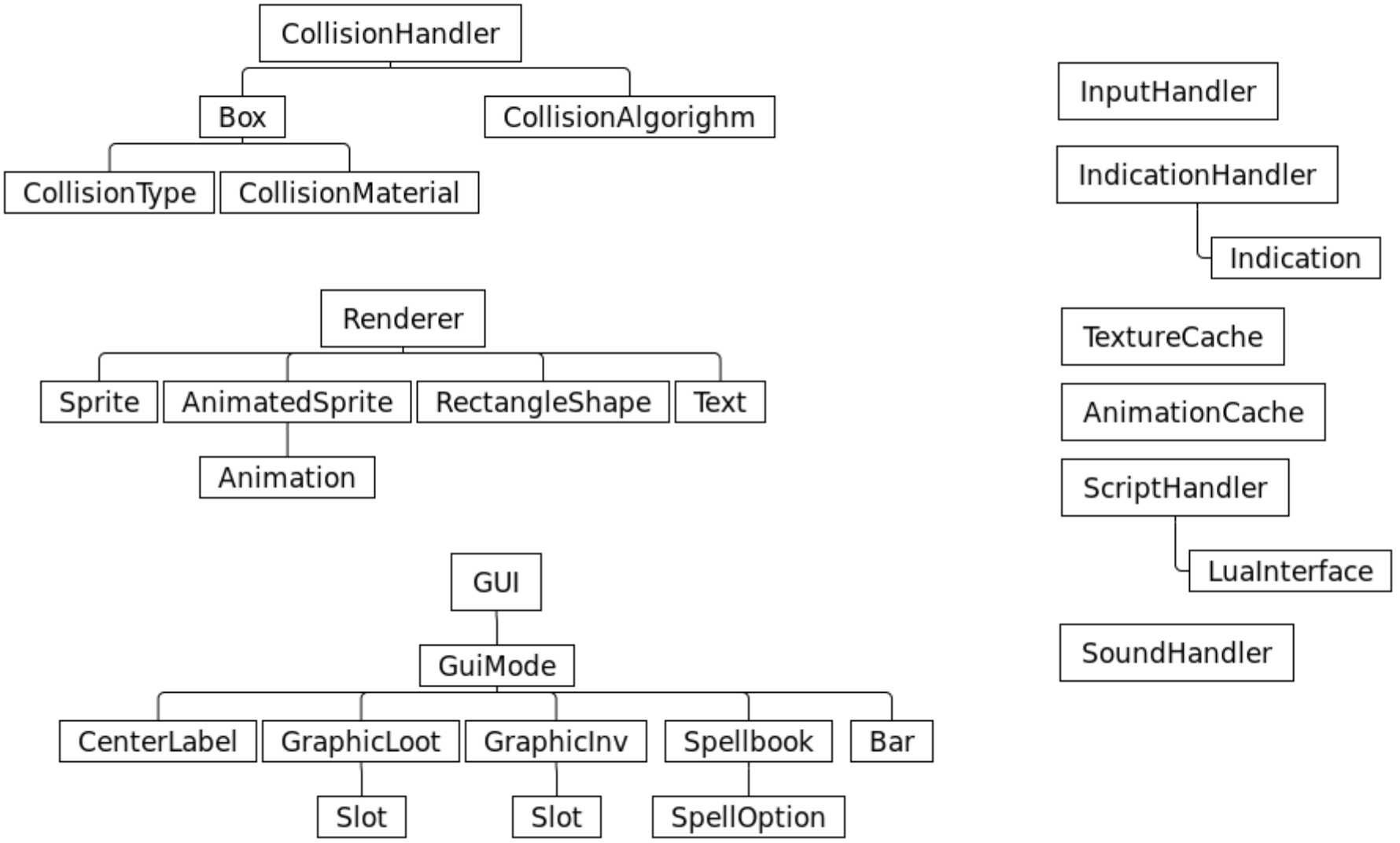
Niezwykle przydatne

- GIMP – Całość grafiki występującej w grze (Mojego autorstwa)
- Sublime Text 3 – Cały kod źródłowy, skrypty, animacje itd..
- Tiled Map Editor – Pliki map (Rozmieszczenie kafelków i przedmiotów na mapie)

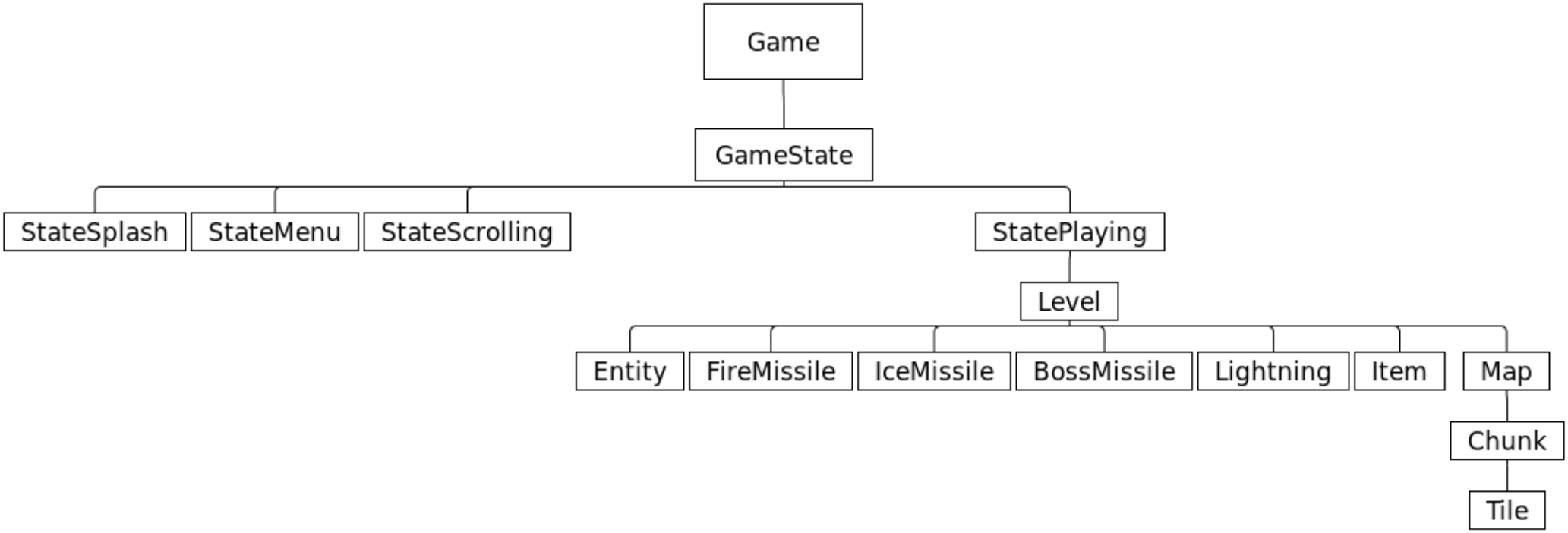
Ogólna Architektura



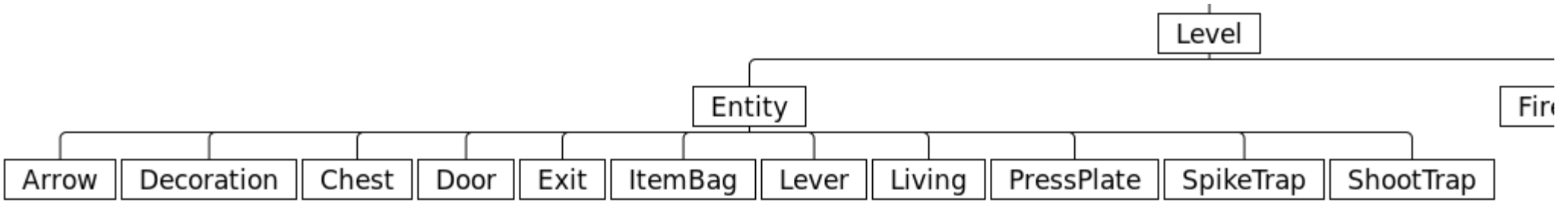
Singletony – Globalne systemy



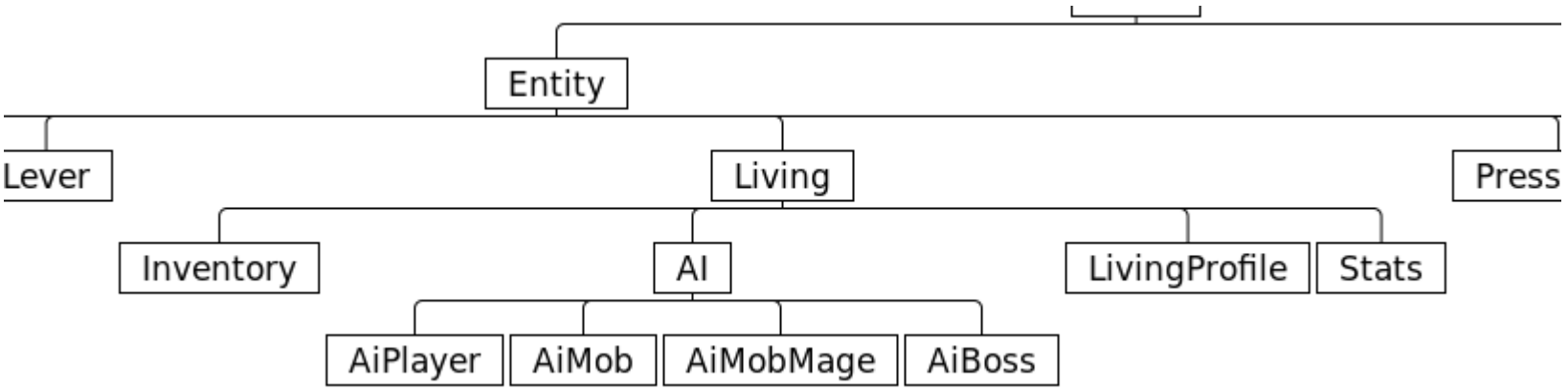
Najwyższy poziom



Objekty w grze



AI - Czyli kontrolery



Cechy programu

- Podejście Obiektowe
- Prawie cała funkcjonalność zawarta jest w klasach
- Nadużycie Singletonów
- Przyzwoita jakość kodu źródłowego

Wybrane Klasy

- Game
- Level
- Map
- Renderer
- GUI

Game

Główna klasa w programie zawiera wszystkie inne klasy. Jest całkowicie hermetyczna.
Czyli nie można jej modyfikować z zewnątrz,
z kilkoma wyjątkami takimi jak zmiana stanu gry.

```
1 #include "Core/Game.hpp"
2
3 int main(int argc, char* argv[])
4 {
5     Game game;
6     game.mainLoop();
7     return 0;
8 }
```

Level

Przechowuje zarówno obiekty, itemy jak i mapę.
Oraz umożliwia interakcję między nimi.
Przy wyjściu z poziomu wszelkie informacje
o graczu i jego ekwipunku są zapisywane do pliku
travel.sav
Przy wejściu dla wszystkich prócz pierwszego
poziomu są one wczytywane, dzięki czemu
w łatwy sposób można uniezależnić gracza od
tego na jakim poziomie się znajduje.

Map

Odpowiada głównie za renderowanie poziomów. Choć spełnia też jedną z najważnieszych funkcji w grze, a mianowicie wczytuje z pliku informacje o wszelkich obiektach na mapie i przekazuje je „w górę” hierarchii do Level’u, który potem przejmuje za nie odpowiedzialność w trakcie właściwego działania programu.

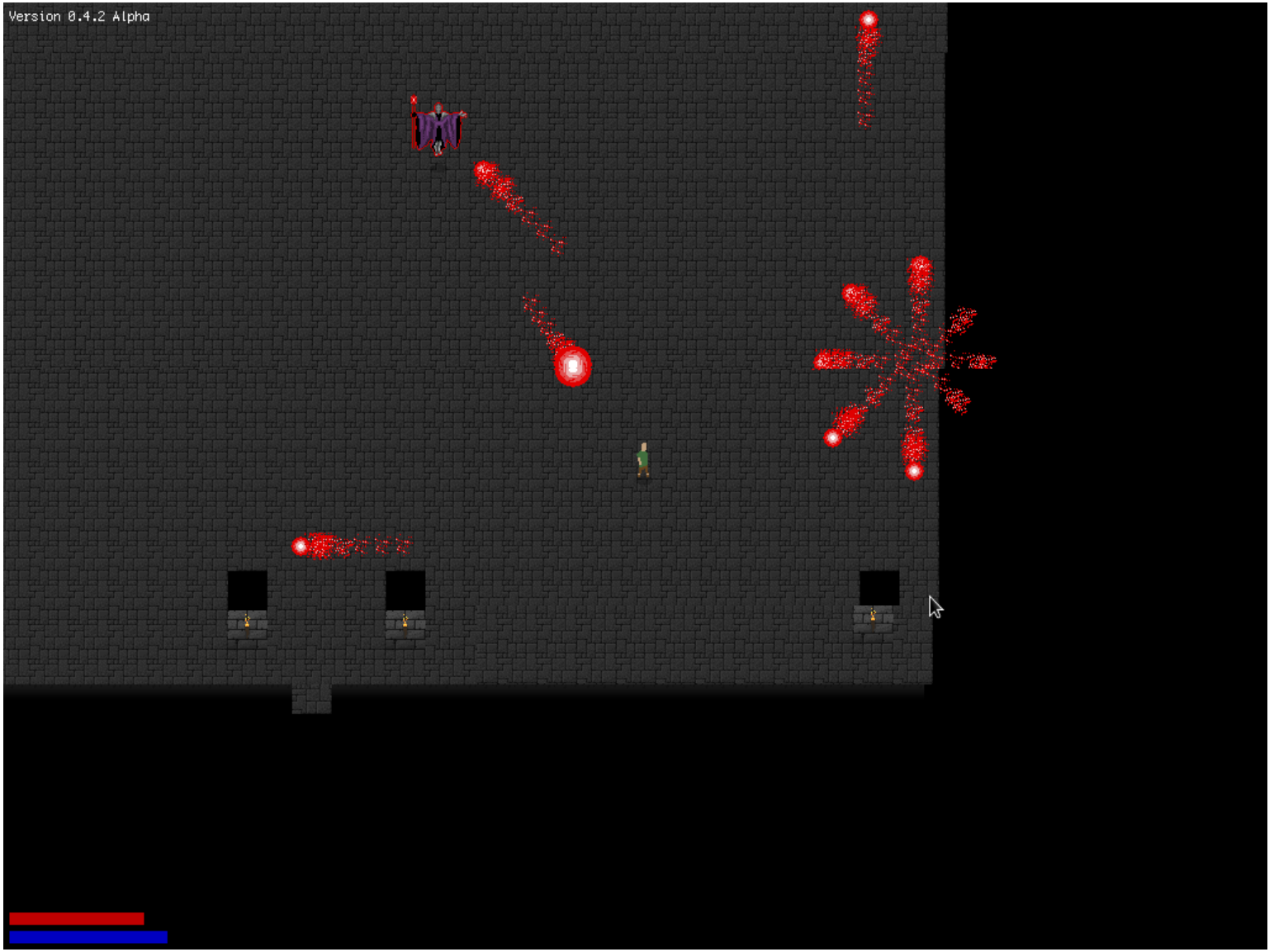
Renderer

Jego zadaniem jest zebranie, przygotowanie i wyświetlenie obiektów, które zostały do niego wysłane.
Jego działanie dzieli się na 4 fazy.
Zebranie obiektów.
Usunięcie tych niewidocznych z listy. (Culling)
Posortowanie. (Algorytm malarza)
Właściwe wyświetlenie na ekranie.

GUI

Pierwotnie odpowiadał tylko za Interfejs Graficzny (HUD), czyli wyświetlanie imion, pasków życia, Inventory itd..
Lecz w krótkim czasie rozrósł się i stał się niejako pośrednikiem pomiędzy różnymi niezwiązanymi ze sobą częściami programu, co było złe i teraz nikt go nie lubi.

Gameplay



W końcu to gra...

O co w niej właściwie chodzi?

- Eksploracja
- Mordowanie potworków
- Walka o przetrwanie
- Rozwój postaci
- Znajdowanie kluczy
- Uczenie się nowych zaklęć

Skąd ten pomysł?

Okrojony
RPG?

- Ograniczony czas
- Ograniczone zasoby ludzkie
- Musiało chociaż przypominać RPG
- Inspirowane The Legend of Zelda i Ultima Underworld

Czego się nauczyłem

- Planować projekt. (chociaż trochę)
- Nie używać więcej singletonów.
- Kończyć to co zacząłem.
- Trochę rysować.
- Github’a.
- Nie wstydzić się kodu. (jakoś trzeba zacząć)
- Jak nakłaniać ludzi do grania w twoją grę, choć tego nie chcą.

Jakieś pytania?

Dziękuję za uwagę.

Dominik Korotkiewicz

Gra, Prezentacja i kod
https://github.com/JLMPL/Dungeonz