
	<p>Uniwersytet Technologiczno-Przyrodniczy im. J.J. Śniadeckich w Bydgoszczy Wydział Telekomunikacji, Informatyki i Elektrotechniki Zakład Techniki Cyfrowej</p>			
Przedmiot	Projektowanie i projektowanie gier - projekt		Kierunek/Tryb	IS/NP
Numer lab.	05	Data wykonania	20.03.2021	Data oddania 22.03.2021
Imię i nazwisko	Antoni Malak			Grupa 1

22.03.2021

Sprawozdanie z prac nad projektem

Instalacja środowiska programistycznego oraz konfiguracja

Wstęp

Instalacja środowiska oraz konfiguracja w moim przypadku była dosyć problematyczna, ze względu na używany system operacyjny (linux), ale na szczęście przy odrobinie uporu udało mi się zainstalować odpowiednie składniki, potrzebne do tworzenia projektu.

Instalacja JDK, JRE, oraz JavaME SDK

By w ogóle zacząć prace nad projektem, wymagane jest zainstalowanie odpowiedniej wersji javy (JRE), oraz wersji developerskiej (JDK). Środowisko, które pobrałem jest netbeans w odpowiedniej wersji (8.2), ale jak się okazuje nawet najnowsza wersja Eclipse, jest w stanie wykryć zainstalowane składniki JavyME oraz w nim również praca jest możliwa. Dodatkowo potrzebne było zainstalowanie SDK dla JavyME, oraz emulatora prostych urządzeń javy.

Stworzenie próbnego projektu

W celu sprawdzenia poprawności działania środowiska, tworzę nowy projekt JavaMe. Przekopiowałem jeden z prostszych programów stworzonych na ćwiczeniach w poprzednim semestrze, oraz sprawdzam, czy program działa.



W tym przypadku jest to gra w zgadywanie liczb, jak widać emulator się uruchamia, a program działa.

Wnioski

Instalacja środowiska czasami może przysporzyć wiele problemów, ale najważniejsze jest to, że na ten moment wszystko działa prawidłowo. Dodatkowo, czasami trochę trudno jest znaleźć odpowiednie, działające wersje SDK. Następnym krokiem powinno być zapoznanie się z oryginałem, oraz wstępne rozpisanie

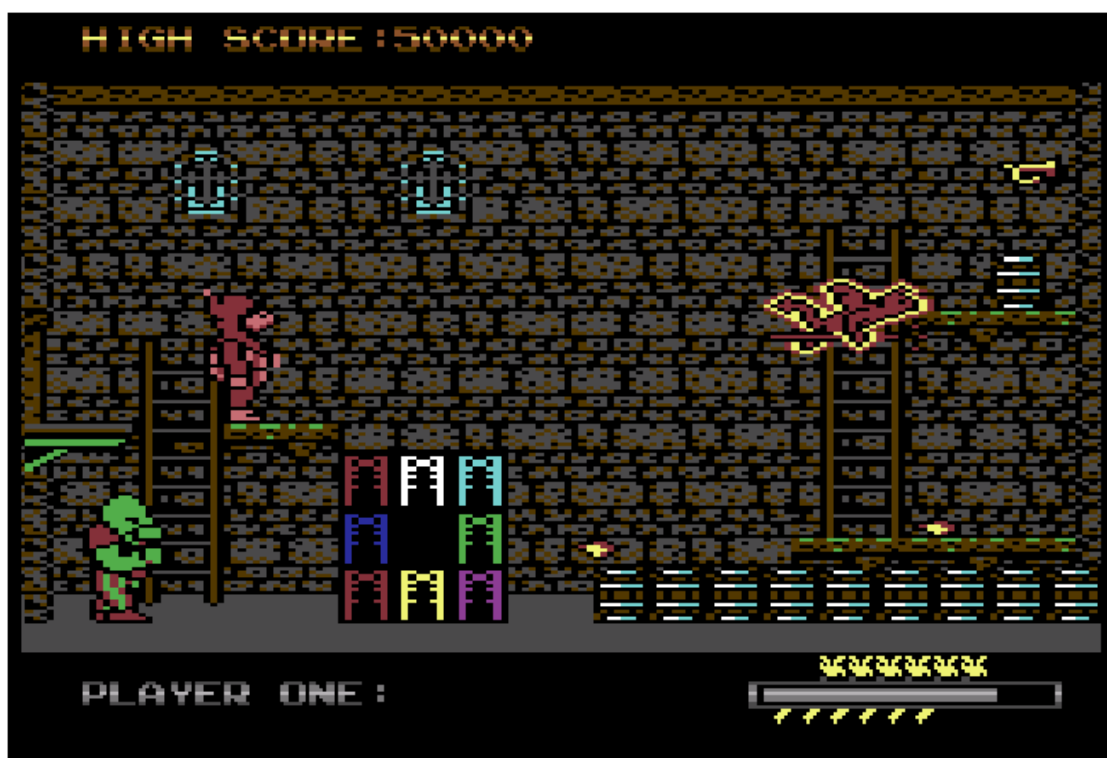
wymaganych klas do użycia w projekcie. Potem powinno stworzyć się podstawową pętlę, oraz szablon projektu.

13.04.2021

Zapoznanie się z oryginałem

Pierwszym krokiem będzie zapoznanie się z oryginałem, czyli grą 'Black Lamp'. W tym przypadku jest to wersja na komputer Comodore 64.

Wyróżnić można parę elementów gry takie jak: tło, przedmioty, postacie oraz pociski. Elementy te będą określone za pomocą klas użytych w projekcie.



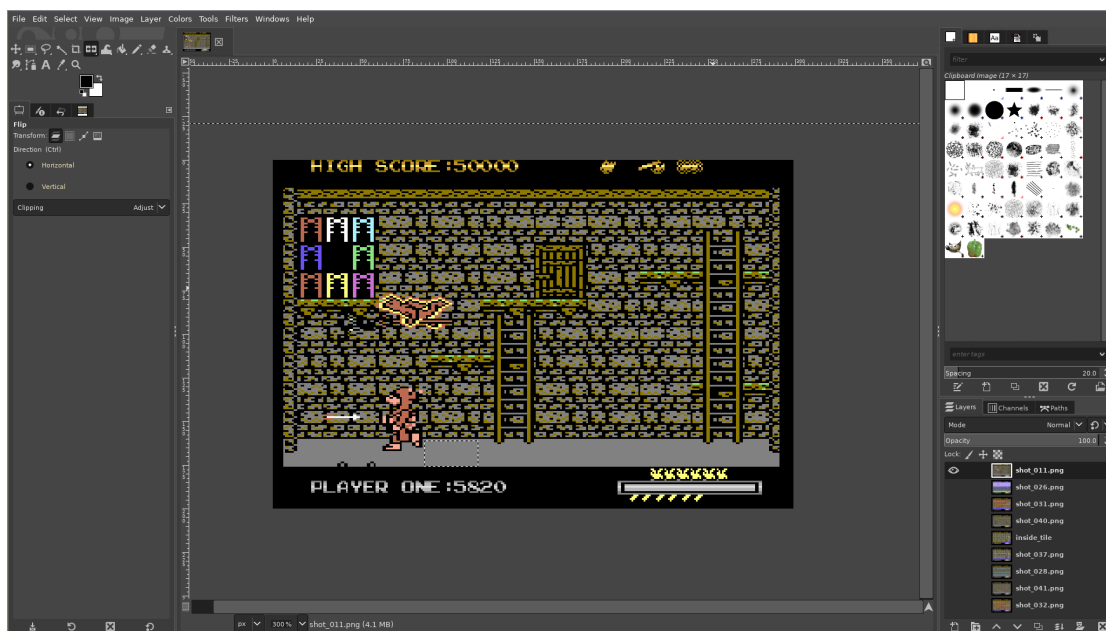
Emulator oraz pozyskanie grafik do gry

Pozyskanie grafik zaczynam od konfiguracji emulatora, by wyświetlał się w skali 1:1, dzięki czemu możliwe będzie pozyskanie grafik w oryginalnym rozmiarze. Takowe grafiki zawsze można zeskalować w górę, ale w naszym przypadku nie będzie to potrzebne, gdyż rozdzielczość ekranu komórki jest całkiem podobna do natywnej rozdzielczości gry.

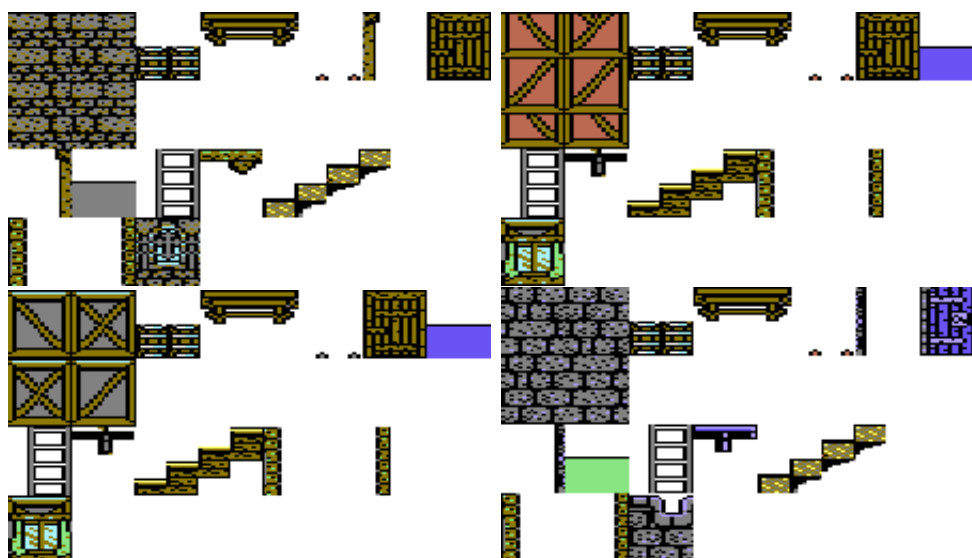


Elementy gry

Po pobieżnym zapoznaniu się z grą można wydzielić z jakich elementów (graficznych) składa się grafika gry. Każdy poziom zdaje się używać tej samej liczby grafik, ale one różnią się stylem pomiędzy typami poziomów. Postanowiłem stworzyć 3 foldery które będą posiadały odpowiedniki tych samych grafik ale w różnych stylach reprezentowanych przez poziomy. Poziom będzie deilić się na elementy tła nie kolidujące, itemy(elementy otoczenia) oraz postacie. Po zrobieniu odpowiedniej ilości screenshotów można w programie graficznym powycinać i pozyskać grafiki potrzebne do stworzenia gry.

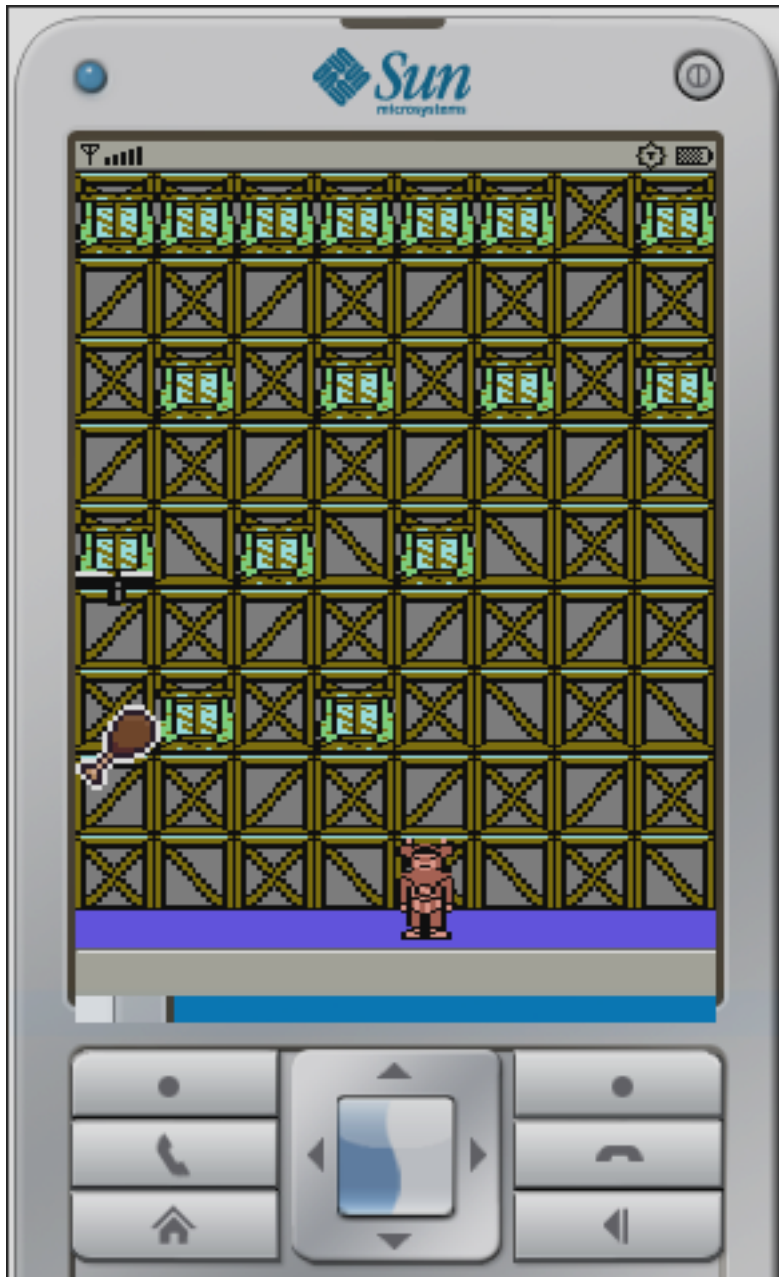


Wydzielić można cztery różne style poziomów, które będą z sobą kompatybilne.



Programowanie podstaw gry

Pierwszymi krokami będzie zaimplementowanie i przetestowanie działania najbardziej podstawowych funkcji gry, takich jak sterowanie, wyświetlanie zdjęć, kolizje oraz pętla główna gry. Swój kod oparłem na kod wcześniej napisany na laboratoriach.



Kod pętli głównej gry

```
class HeartBeat extends Thread{
public void run(){
    while(isAlive==true){
        try{
            sleep(10000);
            pong();
            System.out.println("Heartbeat!!!");
        }catch(Exception e){
            isAlive=false;
        }
    }
}
}
```

Po przetestowaniu tych, podstawowych funkcji, możliwe było rozrysowanie oraz rozplanowanie budowy gry. Jako że język Java obiera się na tworzenie klas, najlepszym rozwiązaniem byłoby jak najczęstsze użycie możliwości, które dają. Tło poziomiu będzie jedynie zwykłym obrazkiem, wyświetlanym z pętli, na początku rysowania poziomu.

```
public class Item {
    public int x,y;
    public int width,height;
    public Image sprite;
    boolean colision;

    public String name;
    public String type;
}
```

Potem wyświetlane będą obiekty klasy Item, które reprezentują elementy które znajdują się w danym poziomie, takie jak platformy, schody, drabinki itd. Są to elementy, które będą wymagały detekcji kolizji, którą zapewni ta klasa. Dodatkowo przechowywać będzie ona położenie, rozmiary oraz obrazek danego przedmiotu. Klasa będzie zapewniać podstawowe funkcję dotyczące detekcji kolizji, przemieszczania, oraz rysowania przedmiotu.

```
public class Creature extends Item{
    public double xspd=0;
    public double yspd=0;
```

```

    public boolean flipped=false;

    public int bulletIndex=0;
    public Bullet bullets[];

    public int cooldown=0;
    public int maxCooldown=20;

    public void shotDown(){
    public void shot(){
    public void move(){
    public void drawBullets(Graphics g, int scroll){
    public void draw(Graphics g,int scroll){
}

```

Następna klasa Creature miała opisywać stworzenia, albo przedmioty poruszające się w świecie gry. Rozbudowują one klasę item o możliwość strzelania oraz funkcję przemieszczania się na podstawie ich energii kinetycznej (xspd,yspd). Zaimplementowana również została funkcja, która tworzy odbicie lustrzane obrazków, które są wyświetlane gdy postać jest zwrócona w lewą stronę.

```

public class Enemy extends Creature{
    Animation animationFly;
    int cooldown=300;
    int counter=0;

    public void ai(int scroll,int playerX,int playerY){
    public void draw(Graphics g,int scroll,int playerX,int
        playerY){
}

```

Klasa Enemy, będzie odpowiadać za stworzenie przeciwników, dodaje ona głównie schemat poruszania się i strzelania.

```

public class Player extends Creature{
    Controls controls;

    public double maxSpeed=2.5;
    public double speed=0.4;
    public double jumpForce=4;

    public boolean onLadder;
    public boolean onStairs;
    public boolean jumping=false;

```



```

public boolean colisionDown=false;
public boolean colisionUp=false;
public boolean colisionLeft=false;
public boolean colisionRight=false;

public String nextLevel="";

public boolean lamps[]=new boolean[8];

int marginx=5;
int marginy=5;

Animation walkAnimation;
Animation clmbingAnimation;

```

Klasa Player, jest jedną z najbardziej rozbudowanych, posiada ona logikę głównej postaci, oraz odpowiada za inicjację reakcji z otoczeniem.

```

public class Level {

    String levelType;
    char tiles[][];
    int rows,cols;

    //parts of bacground, colision detection not needed
    Image background;
    Image floor;
    Image window;

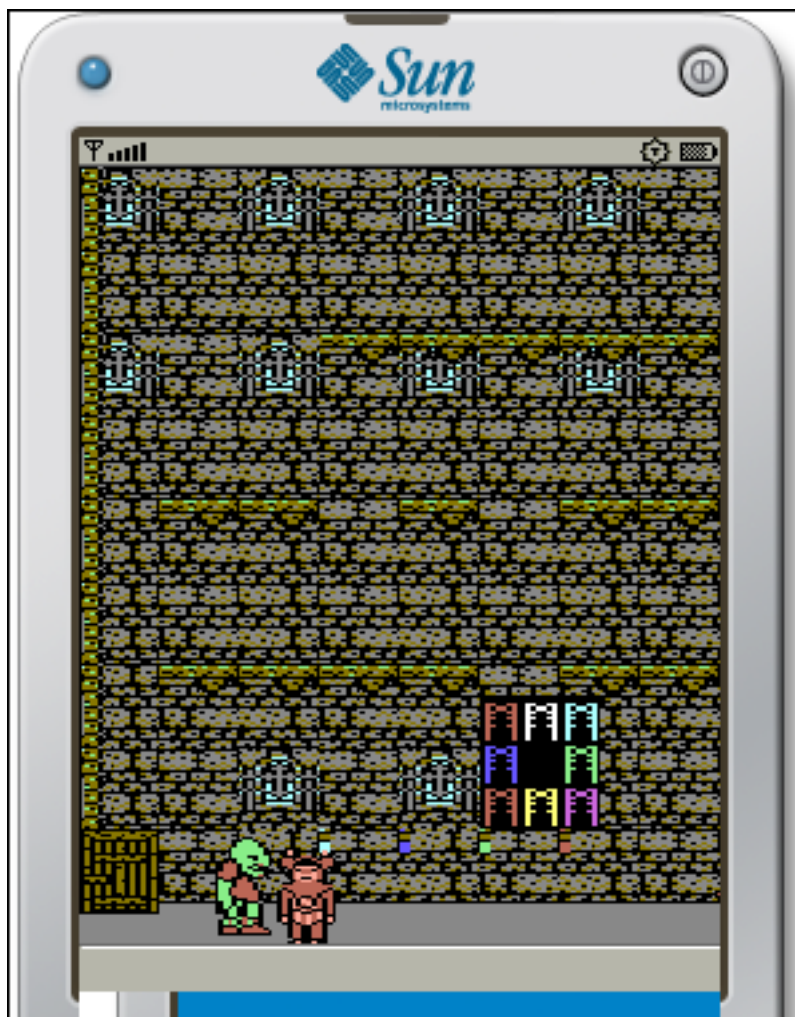
    public Item items[];
    public int width=0;

    public int lampSlotX=-60;
    public int lampSlotY=-60;

```

Klasa level jest odpowiedzialna za wczytywanie oraz wyświetlanie poziomów gry, które mogą być zmieniane z locie. Plik txt jest wczytywany oraz tłumaczony na schemat poziomu. Zbudowany jest on z jednolitych kafelków, które są Item-ami z którymi gracz będzie mógł wchodzić w interakcję.

Wygląd po implementacji

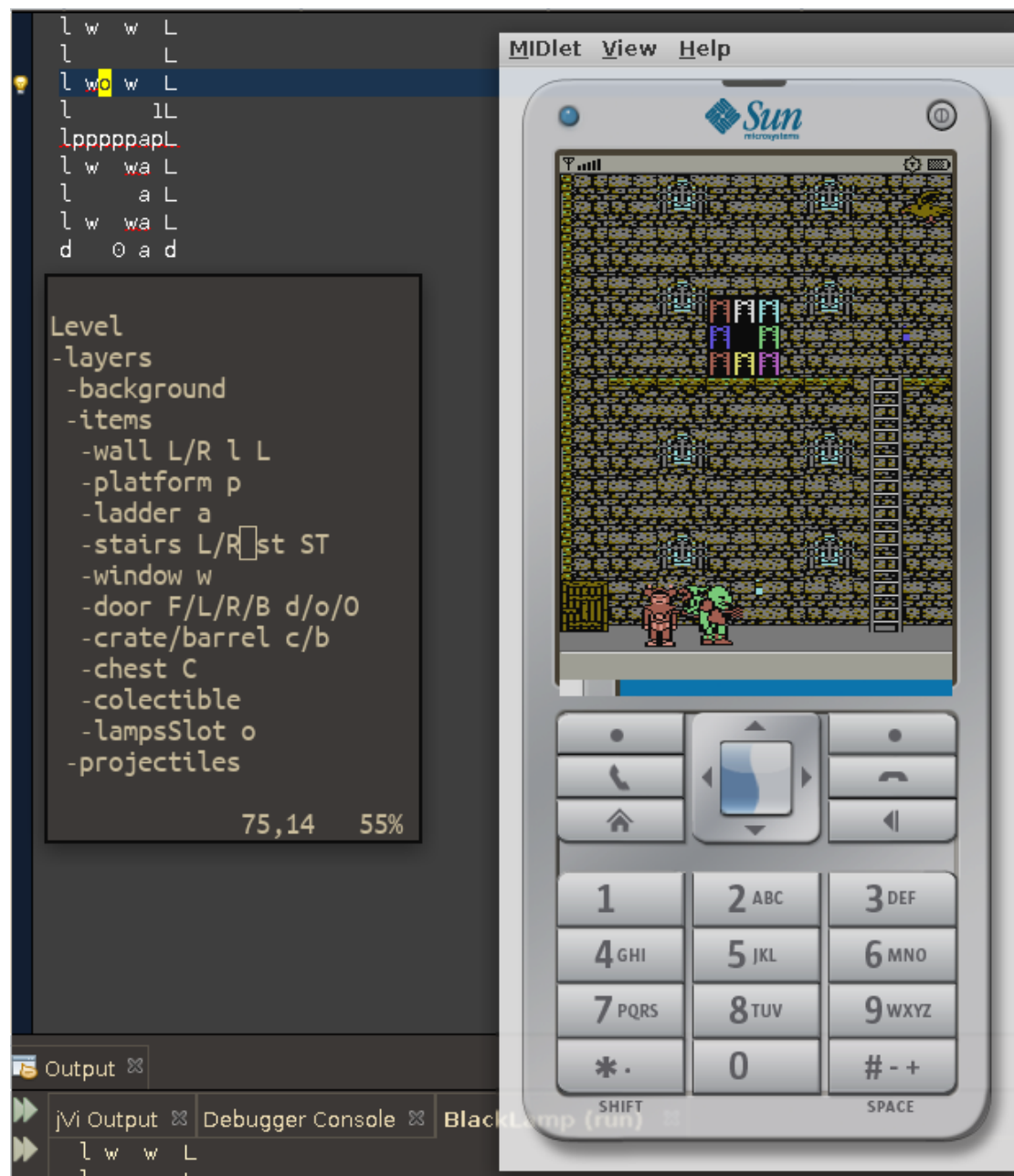


Dalsze funkcjonalności gry

W grze został zaimplementowany jedynie wygląd z pierwszych poziomów, oraz nie została zaimplementowana cała gra i jej przeciwnicy. Można by określić wersję tej gry jako wersję demonstracyjną.

Poziomy są zapisywane za pomocą plików w formacie txt, gdzie spacja jest pustym miejscem, a odpowiednie litery odpowiadają odpowiednim elementom otoczenia wyświetlonych w grze. W domyśle, drzwi przenoszą gracza do poziomu mieszczącego się w danym kierunku. Format pliku wygląda 'level+0+0+0.txt' gdzie pierwsza liczba to x (ze znakiem), a druga to y. Przechodząc przez lewe/prawe skrajne drzwi poruszamy się w osi OX, a przechodząc przez drzwi mieszczące się w środku pomieszczenia

przemieszczamy się w osi OY.



Na zdjęciu widać w jaki sposób, plik txt jest reprezentowany jako poziom gry.



Możliwe też jest zbieranie lamp oraz umieszczanie ich w slotach.

Wnioski

Użycie mechanizmu dziedziczenia oraz odpowiednie rozplanowanie klas w projekcie pozwoliło na znaczne skrócenie czasu potrzebnego na napisanie projektu. Raz napisana funkcja mogła zostać użyta ponownie, gdyż wiele klas dziedziczyło po sobie.