COMPUTER SCIENCE

PROJECT

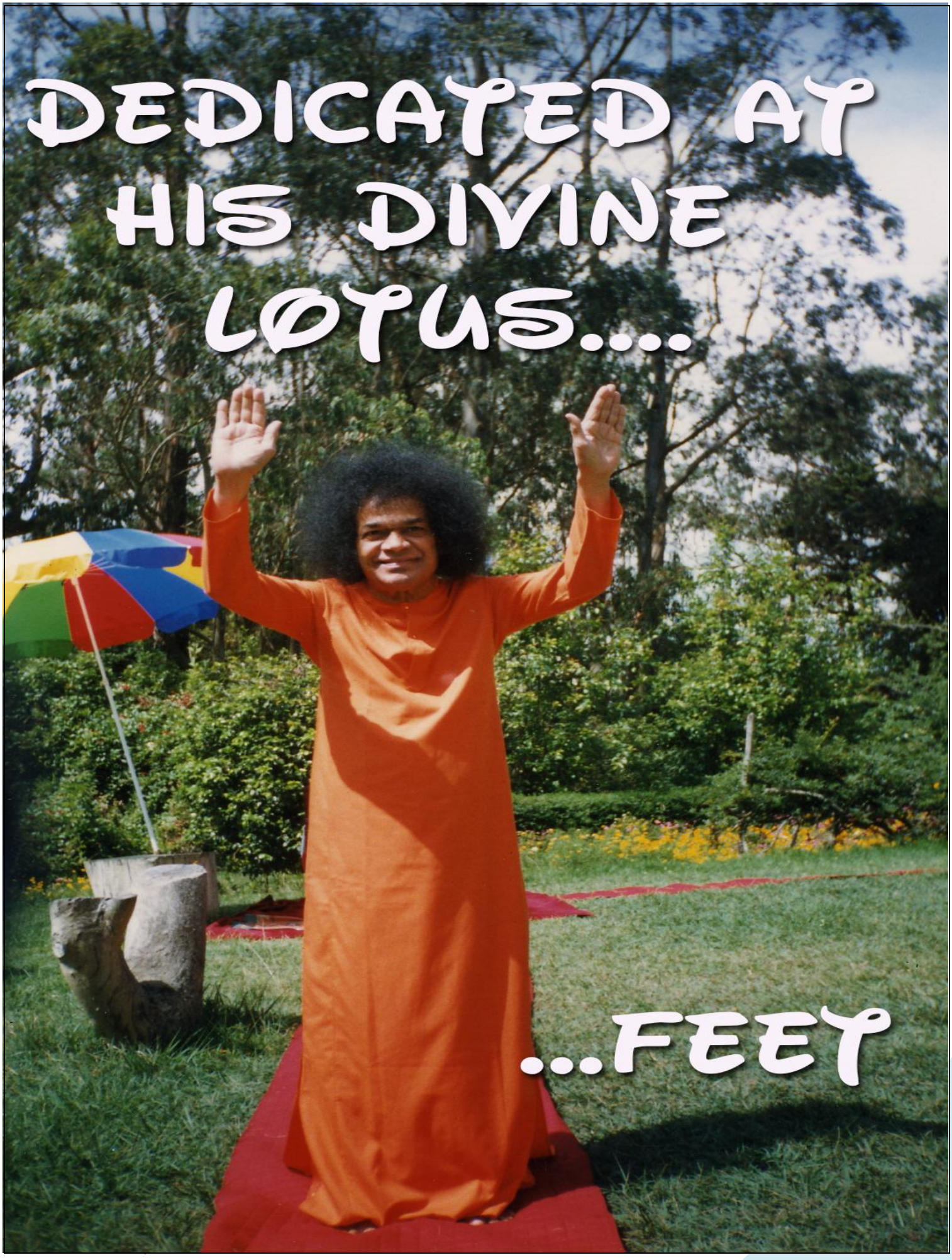
2017-2018

PRINCE OF PERSIA

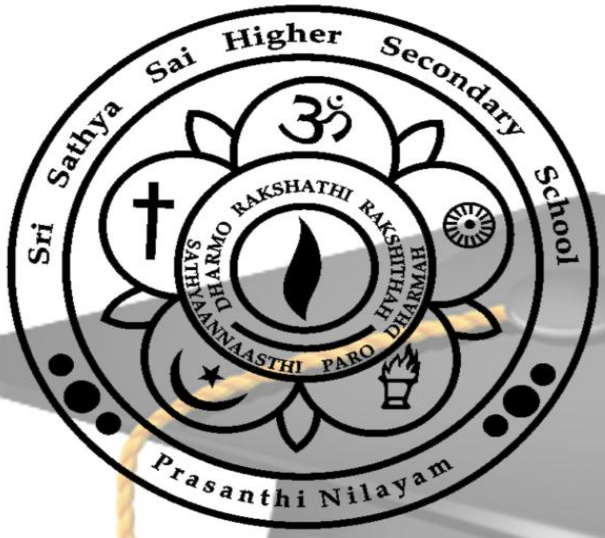
Struggle for Supremecy

SAI RAJESH

XII MPC


DEDICATED AT
HIS DIVINE
LOTUS....
...FEET

Sri Sathya Sai Higher Secondary School
Vidya Giri,Prashanthi Nilayam - 515134



CERTIFICATE

Regd. No:_____

Certified that this is the Bonafide project work done by SAI RAJESH in the AISSCE Course in COMPUTER SCIENCE during the year 2017-2018.

This Project work is submitted for the Practical Examination conducted by the Central Board Of Secondary Education, New Delhi in March 2018.

Date:

Principal ***Internal Examiner*** ***External Examiner***

Contents:

Acknowledgement

I most sincerely express my gratitude to Bhagawan Sri Sathya Sai Baba whose hands propelled me throughout my journey and at every moment of my association with this project. His Blessings and grace were the main source of ideas which has led to the success of my project.

I thank my school, SRI SATHYA SAI HIGHER SECONDARY SCHOOL, for providing me with the best of the infrastructure and resources which has made my task comfortable.

I honestly thank my best teacher, Shri Venkateswar Prusty, for all the guidance and encouragement provided by him. His inspiration and motivation helped me to realize and blossom the innate talent present in me. His teaching enriched my thinking and he carved me in a beautiful sculpture. I humbly thank my principal, Shri Shiva Ramakrishnaiah, for his support.

I express my deepest love to my parents for their ever loving words which helped me in analyzing my errors and their blessings and prayers that have made me what I am today.

I am very thankful to all my dear classmates. We learnt together and had wonderful and memorable time.

AIM

To develop a video game

"POP-AFA"

-PRINCE OF PERSIA,

STRUGGLE FOR SUPREMACY,

using Bloodshed Dev C++

Introduction

Entertainment, being the pre-requisite of today's generation, has become an inevitable part of life. Movies, computer games, mobile games have become an indelible part of people of this era. Fascination grips every kid who comes across a game to play. Some games are addictive.

Gaming as a profession, as a hobby, and as a fascination has stolen the hearts of many of the people so much so that they make movies from games. There are even games made from movies for example "The Star Wars",etc. Some games have many sequels with the same name for example "Temple run1", Temple run2"...etc.

My project is a sequel of Prince of Persia. A movie by this name has also released and this game has many sequels. The plot of the game makes it very exciting. My game too has a very addicting plot which is revealed as one gets past the levels. Prince of Persia, Struggle for supremacy, has been selected as the title of this game. The most interesting part of this 2D game is that you get lost in playing it.

This project is a humble attempt to recreate a game which involves the player completely.

Theory

The game has a story line of its own. The prince is the protagonist of the game as suggested by the name of the game. It has only 8 levels and is rather simple. This game requires quite a bit of patience and memory. This is because, while traversing the different routes, the player can't really see the entire track.

As the player moves around, the route has to be shifted manually so that some other part of the maze is visible. This makes the game more realistic and the prince dies if he goes out of the screen.

Prince of Persia is a fun filled, adventurous and a strategic game which reveals one's gaming skill, kindles your imagination and involves you completely.

The prince can do various actions. He can move right (->), left (<-), jump (Spacebar), climb up (up arrow), climb down (down arrow). There are power ups which can be collected. Coins, health potion (health of the prince becomes maximum), life potion (increases your life) are provided to make the game more interesting. There are soldiers and guards who attack you when you go near them. The prince has a sword. The fight mode (F) provided to the prince helps him to get ready to fight.

Few fighting techniques that the prince can use are hit side (X), hit down (Z) and hit combo (Enter) to create the vigour and energy in the player. An added feature is that he can defend (Ctrl). While defending if the soldier hits the prince, the prince smashes the soldier with greater vigour. The health potions collected earlier can be used any time (H) depending on the requirement.

In the beginning, the prince is provided with 3 health potions and 3 life potions. This will enable the player to get started. The game can be paused (P) and resumed.

Program Design

The game was created using the DEV-C++ IDE since it allows the easy display of images in the game. The 'ALLEGRO' library was used since it helps to deal with the various images very efficiently with its in-built functions. Many variables had to be used. So the best way to do it was by implementing Object Oriented Programming.

The classes used in this program code are:

- Prince
- Vilan
- Level
- Powerup

Class Prince contains many variables and functions. This class has a constructor which loads two images having different frames of the prince and initializes the prince's life, health, x and y coordinates and many more. There are different functions defined in the class which draws him, makes him run around, jump, climb up a wall, climb down a wall, take out his sword, fight in 3 different ways, defend himself, close the sword, die and various other actions.

Class Vilan has its variables and functions designed to represent a soldier or demon gauding the kingdom at various locations. This class has a constructor and a destructor which initializes various variables. Whenever the prince comes near any of them, they attack the prince by calling the functions (hit_down, hit_side).

Class Level is created to maintain the features and advancement of the player in each level. It loads the tile map in the constructor.

Class Powerup is an extra attachment to the game which has objects like coins, health potion and life potion. Its functions help to modify the position of the power ups.

There is a global 3D array of int type ***tilemap_level*** which is used to create the level background. Another global 3D array of int type named ***vilpos*** is used to set the initial positions of prince, soldiers, etc. There are a few character pointers to constant variable which contain information about the pictures and text to be shown as output. Some more global variables storing the no of coins, health potions, life potions, etc, have been created.

In the main function the allegro graphic mode gets initialized. An object prince of prince class, a 1 D array level[8] of level class, a 1D array soldier[10] of vilan class are initialized. A function –***introduction*** gives a brief entry to the game. The function ***start level*** initializes the level selected by the player using the ***levelintr*** function. A function called ***playlevel*** is called which does all the things for the player to interact and play the game. These functions are put in loops. An interesting feature is that at the end of each level the data related to the player's progress is recorded in a backup file and can be retrieved whenever he wants to continue.

Program Code

```
//////////////////////Header files//////////////////////////
#include <allegro.h>
#include <alfont.h>
#include <fstream>
#include <time.h>
#include <stdlib.h>
using namespace std;
//////////////////////Variables//////////////////////////
bool live =true;
long score=1000, prince_health=100, soldier_health=100, hpno=3, hpnol=0,
lpnol=0, lpno=3, coinol=0, coino=0 ;
bool ch_rest =true, level_cleared=false;
int soldierno=0;
int leveltilemaphandler=0;long points;
int coun=0;
const char *pow []= {
   "hpotion.bmp", "lpotion.bmp", "coin.bmp",
   "intrr1.bmp", "intrr2.bmp", "intrr3.bmp", "intrr4.bmp", "intrr5.bmp",
"st1.bmp", "st2.bmp", "st3.bmp","st4.bmp" ,"st5.bmp", "st6.bmp",    "st7.bmp"
};
const char *intrwriteup[8][4]={
   {"Hi! I managed to escape from prison","..I need to get to my lucky sword..","I
robbed a sword which will help me","        Lets get started.        "},
   {"Finally I got my lucky sword.....","I need to find my father,the king..","And
rescue my nation ...alone???...","I need to find the map to the dungons"},
   {"   The alarms have been rung!!   ","Gotcha move fast and be ready
.....","....for any attack.....silence.....","I will rescue my father at anyy cost."},
   {"        This Maapp!!!!        "," My father had shown it to me !!!  ","This will
lead me to the dungons ..","I will rescue my father at anyy cost."},
```

```
    {"Hi! I managed to escape from prison","..I need to get to my lucky sword..","I
robbed a sword which will help me","        Lets get started.        "},
    {"Finally I got my lucky sword.....","I need to find my father,the king..","And
rescue my nation ...????.....","I need to find the map to the dungons"},
    {"Hi! I managed to escape from prison","..I need to get to my lucky sword..","I
robbed a sword which will help me","        Lets get started.        "},
    {"Finally I got my lucky sword.....","I need to find my father,the king..","And
rescue my nation ...????.....","I need to find the map to the dungons"}
};
int level_rgb[9][3]={
   {220,150,65},
   {220, 150, 65},
   {220,150,65},
   {220,150,65},
   {220, 150, 65},
   {220,150,65},
   {220,150,65},
   {220, 150, 65},
   {220,150,65},
};
int vilpos [8][16][2]={
    {//level number 1
        {1100, 250},
        {1300, 250},
        {400, 250},
        {700, 750},
        {1500, 770},
        {1850, 900},
        {2300, 400},
        {2100, 100},
        {2700, 750},
        {2600, 250},
        {2800, 90},//end pos
```

```
        {-20, 30},//start pos
        {0, 0},//level bg start pos
        {1900, 350},
        {2400, 200},
        {2200, 950}
    },
    {//level number 2
        {500, 30},
        {400, 450},
        {300, 750},
        {300, 900},
        {1100, 1200},
        {2000, 1200},
        {1500, 1050},
        {2200, 900},
        {1700, 750},
        {1500, 750},
        {1300, 820},
        {500, 100},
        {0, 0},
        {100, 350},
        {2700, 950},
        {2700, 1250}
    },
    {//level number 3
        {300, 300},
        {1500, 1200},
        {1200, 1050},
        {400, 700},
        {800, 500},
        {1300, 600},
        {2800, 300},
        {1600, 30},
```

```
        {1400, 150},
        {1300, 30},
        {1400, 50},
        {1300, 750},
        {1100, 600},
        {400, 950},
        {2700, 500},
        {1100, 50}
    },
    {//level number 4
        {800, 150},
        {600, 450},
        {400, 300},
        {800, 600},
        {200, 600},
        {1900, 600},
        {2100, 300},
        {2000, 150},
        {2000, 30},
        {2100, 30},
        {1900, 30},
        {50, 30},
        {0, 0},
        {2100, 100},
        {2700, 750},
        {2600, 250}
    },
    {//level number 5
        {600, 30},
        {200, 45000},
        {300, 750},
        {700, 1050},
        {1200, 1050},
```

```
        {1400, 1050},
        {1700, 750},
        {2000, 450},
        {1500, 150},
        {1700, 150},
        {1800, 150},
        {300, 30},
        {0, 0},
        {2100, 100},
        {2700, 750},
        {2600, 250}
    },
    {//level number 6
        {700, 30},
        {1000, 150},
        {-20, 600},
        {100, 900},
        {300, 900},
        {500, 900},
        {900, 1050},
        {2300, 150},
        {2300, 300},
        {2300, 450},
        {400, 30},
        {300, 200},
        {0, 0},
        {2100, 100},
        {2700, 750},
        {2600, 250}
    },
    {//level number 7
        {300, 300},
        {1500, 1200},
```

```
        {1200, 1050},
        {400, 750},
        {800, 500},
        {1300, 600},
        {2800, 300},
        {1600, 30},
        {1400, 150},
        {1300, 30},
        {400, 300},
        {0, 950},
        {0, 900},
        {2100, 100},
        {2700, 750},
        {2600, 250}
    },
    {//level number 8
        {300, 300},
        {1500, 1200},
        {1200, 1050},
        {400, 750},
        {800, 500},
        {1300, 600},
        {2800, 300},
        {1600, 30},
        {1400, 150},
        {1300, 30},
        {400, 300},
        {1300, 750},
        {0, 0},
        {2100, 100},
        {2700, 750},
        {2600, 250}
    }
```

```
};
const char *tileframes_level [9][4]={
{"bgl4.bmp","tl4.bmp","tl4.bmp","thorn4.bmp"},
{"bgl3.bmp","tl3.bmp","tl3.bmp","thorn2.bmp"},
{"bgl3.bmp","tl3.bmp","tl3.bmp","thorn2.bmp"},
{"bgl4.bmp","tl4.bmp","tl4.bmp","thorn4.bmp"},
{"bgl4.bmp","tl4.bmp","tl4.bmp","thorn4.bmp"},
{"bgl3.bmp","tl3.bmp","tl3.bmp","thorn2.bmp"},
{"bgl4.bmp","tl4.bmp","tl4.bmp","thorn4.bmp"},
{"bga1.bmp","tla1.bmp","tla1.bmp","thorn1.bmp"},
{"bgl3.bmp","tl3.bmp","tl3.bmp","thorn2.bmp"}
};
const char *villain [][2]={
    {"gruntl.bmp", "gruntr.bmp"},
    {"soldierl.bmp", "soldierr.bmp"},
    {"sergeantl.bmp", "sergeantr.bmp"},
    {"skeletonl.bmp", "skeletonr.bmp"},
    {"lieutanentl.bmp", "lieutanentr.bmp"},
    {"captainl.bmp", "captainr.bmp"},
    {"snaker.bmp", "snaker.bmp"},
    {"dragonr.bmp", "dragonr.bmp"},
    {"skeletonl.bmp", "skeletonr.bmp"},
    {"jaffarl.bmp", "jaffarr.bmp"}
};
int tilemap_level[9][30][30]={
  //level number 1
   {
      {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
      {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0,},
      {0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0,},
      {2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 1, 1, 1, 1, 1, 0, 2, 2,},
      {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,},
      {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 3, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,},
```

```
        {1, 0, 0, 2, 0, 0, 0, 1, 0, 2, 2, 2, 1, 1, 1, 1, 0, 0, 2, 2, 1, 2, 2, 0, 2, 1, 0, 2, 2, 1,},
        {1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,},
        {1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,},
        {1, 0, 2, 1, 2, 2, 0, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 0, 2, 2, 1, 2, 0, 2, 2, 1, 2, 2, 0, 1,},
        {1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,},
        {1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,},
        {1, 2, 0, 2, 1, 2, 2, 2, 0, 0, 0, 0, 1, 2, 2, 0, 2, 2, 2, 2, 1, 0, 2, 2, 2, 1, 0, 0, 2, 1,},
        {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,},
        {1, 0, 2, 2, 2, 0, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 0, 2, 2, 0, 0, 0, 1, 2, 2, 0, 1,},
        {1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 2, 0, 1, 1, 0, 1, 2, 2, 2, 0, 1, 0, 0, 2, 2, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2, 2, 2, 2, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,}
    },
    //level number 2
    {
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {0, 0, 0, 0, 3, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {2, 2, 2, 0, 2, 2, 0, 1, 1, 0, 2, 1, 1, 2, 2, 0, 1, 2, 2, 2, 2, 2, 2, 0, 2, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,},
```

```
{1,0,2,2,2,2,2,1,1,2,0,1,1,2,2,2,2,2,2,2,0,0,2,2,1,2,2,0,0,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,0,0,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,0,0,1,},
{1,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,2,1,1,0,0,2,2,2,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,},
{1,2,2,2,2,2,0,0,0,0,2,2,2,0,0,0,2,2,2,2,0,1,1,1,0,2,2,2,2,1,},
{1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,1,1,1,1,1,},
{1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,1,1,1,1,1,},
{1,0,0,2,2,2,2,2,2,2,1,0,2,2,2,2,1,1,1,1,2,2,2,2,2,2,1,1,1,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,2,2,2,2,0,0,0,0,0,0,0,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,0,0,1,},
{1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,0,0,2,1,2,2,2,2,0,0,0,1,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,2,2,2,2,2,2,0,2,2,2,2,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,0,0,1,},
{1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,0,0,2,2,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,}
},
//level number 3
{
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,},
{0,0,0,0,0,0,0,1,1,0,0,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{0,0,0,0,0,0,0,1,1,0,0,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{2,2,2,0,2,2,0,1,1,0,2,1,1,2,2,0,1,2,2,2,2,2,2,0,2,0,0,0,0,1,},
{1,0,0,0,0,0,0,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,},
{1,0,0,0,0,0,0,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,},
```

```
{1,0,2,2,2,2,2,1,1,2,0,1,1,2,2,2,2,2,2,2,0,0,2,2,1,2,2,0,0,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,0,0,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,0,0,1,},
{1,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,2,1,1,0,0,2,2,2,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1,},
{1,2,2,2,2,2,0,0,0,0,2,2,2,0,0,0,2,2,2,2,0,1,1,1,0,2,2,2,2,1,},
{1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,1,1,1,1,1,},
{1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,1,1,1,1,1,},
{1,0,0,2,2,2,2,2,2,2,1,0,2,2,2,2,1,1,1,1,2,2,2,2,2,2,1,1,1,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,2,2,2,2,0,0,0,0,0,0,0,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,0,0,1,},
{1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,0,0,2,1,2,2,2,2,0,0,0,1,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,2,2,2,2,2,2,0,2,2,2,2,1,2,2,2,2,2,2,2,2,2,2,2,2,2,0,0,1,},
{1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,0,0,2,2,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,},
{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,}
},
//level number 4
{
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,},
{1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,},
{1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,},
{1,2,2,2,2,0,0,1,1,1,0,0,2,2,0,2,2,2,0,0,1,1,2,2,2,0,0,0,0,1,},
{1,1,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,},
{1,1,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,},
```

```
        {1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 0, 1, 2, 2, 2, 2, 0, 2, 1, 2, 2, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 2, 2, 2, 1, 2, 2, 2, 0, 0, 0, 2, 2, 2, 1, 0, 2, 2, 2, 0, 1, 2, 2, 2, 2, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 2, 2, 2, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 2, 2, 2, 0, 0, 1, 2, 2, 2, 0, 2, 2, 2, 1,},
        {1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1,},
        {1, 0, 0, 2, 2, 0, 0, 1, 1, 0, 2, 2, 0, 0, 0, 2, 2, 2, 0, 0, 0, 1, 0, 2, 1, 1, 1, 2, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 0, 0, 0, 0, 0, 1, 2, 2, 2, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,}
    },
    //level number 5
     {
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 2, 2, 2, 0, 0, 2, 2, 1, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,},
        {1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,},
```

```
    {1,0,0,0,1,1,2,2,0,0,2,2,1,0,0,2,2,2,2,2,2,2,0,0,2,0,0,0,0,1,},
    {1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,0,1,2,2,2,2,2,0,1,2,2,2,2,2,2,2,2,2,0,0,2,0,0,0,1,},
    {1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,2,2,2,2,2,2,0,2,1,2,2,0,2,2,2,0,0,0,0,0,2,0,0,0,0,1,},
    {1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,1,2,0,0,0,2,2,0,0,0,2,2,1,2,2,2,0,0,0,0,0,2,0,0,0,1,},
    {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,2,2,2,2,0,0,0,2,2,2,0,0,1,0,0,2,2,2,0,0,2,0,0,0,0,1,},
    {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,0,0,0,2,2,0,0,0,0,0,0,0,1,2,0,0,2,2,2,2,0,0,0,0,0,1,},
    {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,0,0,0,0,0,0,2,2,0,2,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,}
  },
  //level number 6
   {
    {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,},
    {1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,},
    {1,1,1,1,1,2,2,0,2,2,1,1,1,1,1,0,2,2,2,0,0,0,0,0,0,0,0,0,0,1,},
    {1,1,1,0,0,0,0,0,0,0,1,1,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,1,},
    {1,1,1,0,0,0,0,0,0,0,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,1,},
```

```
    {1, 1, 1, 2, 2, 2, 2, 2, 2, 0, 1, 1, 2, 2, 0, 2, 1, 1, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {1, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 2, 2, 1, 0, 2, 2, 2, 0, 0, 0, 0, 0, 2, 2, 2, 1,},
    {1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {1, 0, 2, 2, 2, 2, 0, 2, 1, 1, 2, 2, 0, 0, 1, 2, 2, 0, 2, 2, 1, 0, 0, 0, 0, 0, 0, 2, 2, 1,},
    {1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {1, 2, 1, 1, 2, 0, 2, 1, 1, 1, 0, 0, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1,},
    {1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
    {1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
    {1, 2, 2, 2, 0, 2, 1, 0, 2, 2, 2, 2, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
    {1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
    {1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
    {1, 0, 2, 1, 2, 2, 1, 2, 1, 0, 0, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
    {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
    {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
    {1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,}
 },
 //level number 7
 {
    {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
    {0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
    {2, 2, 2, 0, 2, 2, 0, 1, 1, 0, 2, 1, 1, 2, 2, 0, 1, 2, 2, 2, 2, 2, 2, 0, 2, 0, 0, 0, 0, 1,},
    {1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,},
    {1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,},
```

```
{1, 0, 2, 2, 2, 2, 2, 1, 1, 2, 0, 1, 1, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 1, 2, 2, 0, 0, 1,},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,},
{1, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 1, 1, 0, 0, 2, 2, 2, 1,},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,},
{1, 2, 2, 2, 2, 2, 0, 0, 0, 0, 2, 2, 2, 0, 0, 0, 2, 2, 2, 2, 0, 1, 1, 1, 0, 2, 2, 2, 2, 1,},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,},
{1, 0, 0, 2, 2, 2, 2, 2, 2, 1, 0, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1,},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
{1, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 1,},
{1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
{1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
{1, 0, 0, 2, 1, 2, 2, 2, 2, 0, 0, 0, 1, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1,},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
{1, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 1,},
{1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
{1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
{1, 0, 0, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1,},
{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,}

},
//level number 8
{
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
```

```
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 1, 1, 1, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 2, 1, 1, 1, 1, 1, 2, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 1, 1, 1, 0, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 1, 0, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 0, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,},
        {1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,},
        {1, 1, 1, 0, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 0, 1, 1, 1, 1,},
        {1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,},
        {1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,},
        {1, 0, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 0, 1, 1,},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,},
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,},
        {2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2,},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
    },
};
const char *end_pos[]={
    "sword.bmp",
    "scroll.bmp",
    "door.bmp",
    "kingg.bmp",
    "attk.bmp",
```

```
    "bro.bmp",
    "invin.bmp",
    "scroll.bmp"
};
const char *intr[8][2]={
    {"The Sword.","l0intro1.bmp"},
    {" Maze Map ","l1intro1.bmp"},
    {"The Escape","l0intro1.bmp"},
    {"My Father.","l1intro1.bmp"},
    {"A Disaster","l0intro1.bmp"},
    {" Dear Bro ","l1intro1.bmp"},
    {"The secret land","l0intro1.bmp"},
    {"Final Battle","l1intro1.bmp"}
};
//////////////////Function prototypes//////////////////
void drawpowerups();
//////////////////////Classes//////////////////////
class powerup{
    public:
        powerup (){
            x=0;
            y=0;
            use=1;
        }
        void init(int i){
           x=0;
           y=0;
           use=1;
           powe= load_bitmap(pow[i], NULL);
        }
        int used(){
           if (use) return 1;
           else return 0;
```

```
}
void draww(){
   if (use and x<1152 && x>0 && y<600 && y>0){
     masked_blit(powe, screen, 0, 0, x, y, powe->w, powe->h);
   }
}
void useit(){
   use=0;
}
void setlevel(int levelno){
   no=levelno;
}
void setxy(int m, int n){
   x=m;
   y=n;
}
void incr_x(){
   x+=20;
}
void incr_y(){
   y+=20;
}
void decr_x(){
   x-=20;
}
void decr_y(){
   y-=20;
}
int getx(){
   return x;
}
int gety(){
   return y;
```

```
        }
    private:
        BITMAP *powe;
        int x, y, no, use;//0 for used up , 1 for not used up
}hp[2],lp,coins[30];
class prince{
    public:
        int no;
        int frame;
        int res;
        bool fight;
        int healt;
        int dir;      ///right =0, left=1///
        prince(){
            lifeline=load_bitmap("lifelinep.bmp", NULL);
            fight=false;
            frame=1;
            x=0;
            y=0;
            no=0;
            dir = 0;
            life=1;
            healt=100;
            res=5;
            defe=false;
        }
        int check(int a, int b, int c){
          for (int i=-10; i<11; i++)
            for (int j=-10; j<11; j++)
              if(getpixel(screen, x+75+i, y+115+j)==makecol(a, b, c)){ return 1;
break;}
          return 0;
        }
```

```
void setx(int m){
    x=m;
}
void sety(int m){
    y=m;
}
void setxy(int m, int n){
    x=m;
    y=n;
}
void incr_x(){
    x+=20;
}
void incr_y(){
    y+=20;
}
void decr_x(){
    x-=20;
}
void decr_y(){
    y-=20;
}
int getx(){
    return x;
}
int gety(){
    return y;
}
void move_left(){
    if(check()){
            dir = 1;
            masked_blit(sprite[1], screen, frame*150, 0*150, x, y, 150, 150);
            blit(lifeline, screen, 0, 0, x+25, y+10, healt, 5);
```

```
            rest(res);
            frame++;
            x-=10;
            if (frame == 14)frame=5;
        }
        else
            freefall_start();
    }
    void move_right(){
        if(check()){
            dir = 0;
            masked_blit(sprite[0], screen, frame*150, 0*150, x, y, 150, 150);
            blit(lifeline, screen, 0, 0, x+25, y+10, healt, 5);
            rest(res);
            frame++;
            x+=10;
            if (frame == 14)frame=5;
        }
        else
            freefall_start();
    }
    void climb_up1(){
        masked_blit(sprite[dir], screen, frame*150, 2*150, x, y, 150, 150);
        rest(res);
        if(frame>7)y-=20;
    }
    void climb_up2(){
        masked_blit(sprite[dir], screen, frame*150, 3*150, x, y, 150, 150);
        rest(res);
        if(frame==4 or frame==5)y-=35;
    }
    void climb_down1(){
        masked_blit(sprite[dir], screen, (14-frame)*150, 3*150, x, y, 150, 150);
```

```
    rest(res);
    if(frame==11 or frame==10)y+=35;
}
void climb_down2(){
    masked_blit(sprite[dir], screen, (12-frame)*150, 2*150, x, y, 150, 150);
    rest(res);
    if(frame>7)y+=20;
}
void jump(){
    blit(lifeline, screen, 0, 0, x+25, y+10, healt, 5);
    masked_blit(sprite[dir], screen, frame*150, 1*150, x, y, 150, 150);
    rest(50);
    if(dir==0)x+=20;
    else x-=20;
}
int defend(){
    if(defe) return 1;
    else return 0;
}
void hit_down(){
    blit(lifeline, screen, 0, 0, x+25, y+10, healt, 5);
    masked_blit(sprite[dir],screen,frame*150,5*150,x,y,150, 150);
    rest(150);
}
void hit_side(){
    blit(lifeline, screen, 0, 0, x+25, y+10, healt, 5);
    masked_blit(sprite[dir], screen, frame*150, 5*150, x, y, 150, 150);
    rest(200);
}
void hit_combo(){
    blit(lifeline, screen, 0, 0, x+25, y+10, healt, 5);
    masked_blit(sprite[dir], screen, frame*150, 5*150, x, y, 150, 150);
    rest(150);
```

```
}
void opensword(){
   blit(lifeline, screen, 0, 0, x+25, y+10, healt, 5);
   masked_blit(sprite[dir], screen, frame*150, 4*150, x, y, 150, 150);
   rest(200);
}
void closesword(){
   blit(lifeline, screen, 0, 0, x+25, y+10, healt, 5);
   masked_blit(sprite[dir], screen, frame*150, 6*150, x, y, 150, 150);
   rest(200);
}
void die(){
   life--;
   for(int i=0; i<6; i++){
        masked_blit(sprite[dir], screen, i*150, 7*150, x, y, 150, 150);
        rest(res);
   }
   if(life==0) live=false;
}
void incr_life(){
   life++;
}
void decr_life(){
   life--;
   die();
}
void setsprite(const char *pathleft, const char *pathright){
   sprite[1]=load_bitmap(pathleft,NULL);
   sprite[0]=load_bitmap(pathright,NULL);
}
void draw(){
   if(check() and live){
    if(fight==false){
```

```
                masked_blit(sprite[dir], screen, 0, 0, x, y, 150, 150);
                rest(res);
            }
            else if(defe and fight){
               masked_blit(sprite[dir], screen, 6*150, 6*150, x, y, 150, 150);
               rest(res);
            }
            else{
               masked_blit(sprite[dir], screen, 8*150, 4*150, x, y, 150, 150);
               rest(res);
               }
            blit(lifeline, screen, 0, 0, x+25, y+10, healt, 5);
          }
          else if (live) {
            freefall_start();
          }
        }
       void chdefend(){
          if(defe) defe=false;
          else defe=true;
       }
       int check(){
          for (int i=-10; i<11; i++)
            for (int j=-10; j<11; j++)
              if(getpixel(screen, x+75+i, y+115+j)==makecol(level_rgb[no][0],
level_rgb[no][1], level_rgb[no][2])){return 1; break;}                ///true=1,
false=0///
          return 0;
       }
       int check_up(){
          for (int i=-10; i<11; i++)
            for (int j=-10; j<11; j++)
```

```
                if(getpixel(screen, x+75+i, y-50+j)==makecol(level_rgb[no][0],
level_rgb[no][1], level_rgb[no][2])) return 1;
          return 0;
        }
        void freefall(){

          masked_blit(sprite[dir], screen, 14*150, 0, x, y, 150, 150);
          rest(res);
          y+=10;
        }
        void freefall_start(){
          int i=0;
          if(dir)x-=20;
          else x+=20;
          while (!check()){
            freefall();
            if (i>30){
              die();
              break;
            }
            i++;
          }
          if(i<10) draw();
          else {
            decr_health();
            draw();
          }
        }
        void incr_health(){
          healt=100;
        }
        void decr_health(){
          healt-=10;
```

```
        }
        int check_lp(){
          for (int i=-10; i<11; i++)
            for (int j=-10; j<11; j++)
              if(getpixel(screen, x+75+i, y+115+j)==makecol(255, 0, 0)){ return
1; break;}
          return 0;
        }
        int check_hp(){
          for (int i=-20; i<11; i++)
            for (int j=-20; j<11; j++)
              if(getpixel(screen, x+75+i, y+115+j)==makecol(60, 50, 1)){ return
1; break;}
          return 0;
        }
    private:
        BITMAP *sprite[2],*lifeline;   ///right=0; left=1;///
        int x;
        int y;
        int life;
        bool defe;//defend
};
class level{
    public:
        int lock,maxscore;
        level(){
            layer=create_bitmap(150*30, 75*30);
            no=leveltilemaphandler;
            leveltilemaphandler++;
            draw_tilemaptolayer();
            font=alfont_load_font("RAGE.ttf");
            alfont_set_font_size(font, 50);
            alfont_text_mode(-1);
```

```
            strength1=load_bitmap("ss1.bmp", NULL);
            scc=load_bitmap("scc.bmp", NULL);
            strength2=load_bitmap("ss2.bmp", NULL);
            lx=vilpos[no][12][0];
            ly=vilpos[no][12][1];
            lock=1;
            maxscore=0;
        }
        void draw_tilemap(){
           blit(layer, screen, lx, ly, 0, 0, SCREEN_W, SCREEN_H);
           masked_stretch_blit(scc, screen, 0, 0, scc->w, scc->h, 1050, 30, 300,
200 );
           alfont_textprintf(screen, font, 1150,30, makecol(255,255,255),
"%d",coino);
           alfont_textprintf(screen, font, 1150,80, makecol(255,255,255),
"%d",hpno);
           alfont_textprintf(screen, font, 1150,135, makecol(255,255,255),
"%d",lpno);
           alfont_textprintf(screen, font, 1150,185, makecol(255,255,255),
"%d",score);
           drawpowerups();
        }
        void incr_lx(){
           lx+=20;
        }
        void incr_ly(){
           ly+=20;
        }
        void decr_lx(){
           lx-=20;
        }
        void decr_ly(){
           ly-=20;
```

```
        }
        int getlx(){
            return lx;
        }
        int getly(){
            return ly;
        }
    private:
        int no, lx, ly;
        ALFONT_FONT *font;
        BITMAP *layer, *strength1, *strength2, *scc;
        const char *path;
        void draw_tilemaptolayer(){
            BITMAP *tile;
            int e;
            for (int i=0; i < 30; ++i){
              for (int j=0; j<30; ++j){
                tile = load_bitmap(tileframes_level[no][tilemap_level[no][j][i]],
NULL);
                stretch_blit(tile, layer, 0, 0, tile->w, tile->h, 100*i, 50*j, 100, 50);
                clear_bitmap(tile);
              }
            }
            tile=load_bitmap(end_pos[no],NULL);
            masked_blit(tile, layer, 0, 0, vilpos[no][10][0], vilpos[no][10][1], tile-
>w, tile->h);
            clear_bitmap(tile);
        }
};
class vilan{
    public:
        int no;
        int frame;
```

```
bool live;
int vilno;
int dir;      ///right =0, left= 1///
vilan(){
      frame=1;
      x=0;
      y=0;
      no=0;
      dir = 0;
      life=1;
      healt=100;
      live=true;
      lifeline=load_bitmap("lifeline.bmp", NULL);
}
void recreate(){
      healt=100;
      live=true;
}
void setx(int m){
   x=m;
}
void sety(int m){
   y=m;
}
void setxy(int m, int n){
   x=m;
   y=n;
}
void incr_x(){
   x+=20;
}
void incr_y(){
   y+=20;
```

```
}
void decr_x(){
   x-=20;
}
void decr_y(){
   y-=20;
}
int getx(){
   return x;
}
int gety(){
   return y;
}
void hit_down(){
   masked_blit(sprite[dir], screen, frame*150, 150, x, y, 150, 150);
   blit(lifeline, screen, 0, 0, x+25, y+10, healt, 5);
   rest(50);
}
void hit_side(){
   masked_blit(sprite[dir], screen, frame*150, 1*150, x, y, 150, 150);
   blit(lifeline, screen, frame*150, 1*150, x+25, y+10, 150, 150);
   rest(50);
}
void die(){
      masked_blit(sprite[dir], screen, frame*150, 3*150, x, y, 150, 150);
      rest(100);
}
void kill(){
   live=false;
   soldierno++;
}
void setsprite(const char *pathleft, const char *pathright){
   sprite[1]=load_bitmap(pathleft,NULL);
```

```
        sprite[0]=load_bitmap(pathright,NULL);
    }
    void draw(){
        if(live and check() and x<1102 && x>0 && y<550 && y>0){
         masked_blit(sprite[dir], screen, 0, 0, x, y, 150, 150);
         blit(lifeline, screen, 0, 0, x+25, y+10, healt, 5);
         ch_rest=true;
        }
        else if (live and x<1102 && x>0 && y<550 && y>0) {
           freefall_start();
           ch_rest=false;
        }
        else ch_rest=true;
    }
    int check(){
       for (int i=-10; i<11; i++)
         for (int j=-10; j<11; j++)
            if(getpixel(screen, x+75+i, y+115+j)==makecol(level_rgb[no][0],
level_rgb[no][1], level_rgb[no][2])){return 1; break;}                ///true=1,
false=0///
       return 0;
    }
    void freefall(){
        masked_blit(sprite[dir], screen, 14*150, 0, x, y, 150, 150);
        rest(50);
        y+=10;
    }
    void freefall_start(){
        int i=0;
        if(dir)x+=20;
        else x-=20;
        while (!check()){
            freefall();
```

```
            i++;
            if (i>30){
              die();
              break;
            }
        }
        if(i<10) draw();
        else {
           decr_health(0);
           draw();
        }
        if(i>30){
           decr_life();
        }
      }
      void incr_health(){
        healt=100;
      }
      void decr_health(int m){
        healt-=m;
        if(healt<=0)die();
      }
      int gethealt(){
        return healt;
      }
  private:
      BITMAP *sprite[2], *lifeline;   ///right=0; left=1;///
      int x;
      int y;
      int life;
      int healt;
};
```

```
void name_intro(int f,const char *name, float m, int x=0, int p=0,int r=255, int
g=255,int b=255){
   BITMAP *c2 = create_bitmap(SCREEN_W, SCREEN_H);
   blit(screen, c2, 0, 0, 0, 0, SCREEN_W, SCREEN_H);
   ALFONT_FONT *font;
   if(f==0)
   font=alfont_load_font("alien.ttf");
   else if(f==1)
   font=alfont_load_font("Fiddum.ttf");
   else if (f==2)
   font=alfont_load_font("zombie.ttf");
   alfont_text_mode(-1);
   alfont_set_font_size(font, 0);
   int y,z;
   for (int i=0; i<200; i++){
      y=i*2;
      if(x==0) clear_to_color(screen, makecol(i, i, i));
      else blit(c2, screen, 0, 0, 0, 0, SCREEN_W, SCREEN_H);
      if (p==0) alfont_textout(screen, font, name, int(600-i*m), 300-i, makecol(r-i,
g-i, b-i));
      else if (p==2) alfont_textout(screen, font, name, int(600-i*m), 300,
makecol(r, g, b));
      else{
         alfont_set_font_size(font, 100);
         alfont_textout(screen, font, name, int(600-50*m), 300-50, makecol(r-i,
g-i, b-i));


      }
      alfont_set_font_size(font, i);
      if(key[KEY_ENTER])break;
      vsync();
   }
}
```

```
void try_intro1(BITMAP *c1){
   int j=700,i=0;
   BITMAP *c2 = create_bitmap(SCREEN_W, SCREEN_H);
   blit(screen, c2, 0, 0, 0, 0, SCREEN_W, SCREEN_H);
   for (; i<400; i+=5,j-=5){
      blit(c2, screen, 0, 0, 0, 0, SCREEN_W, SCREEN_H);
      stretch_blit(c1, screen, 0, 0, c1->w, c1->h, i, j, 1352-i*2, 700-j);
      vsync();
   }
   rest(10);
   for (; i>0; i-=5,j-=5)
      stretch_blit(c1, screen, 0, 0, c1->w, c1->h, i, j, 1352-i*2, 700-j);
   stretch_blit(c1, screen, 0, 0, c1->w, c1->h, 0, 0, SCREEN_W,
SCREEN_H);
   while(!key[KEY_ENTER]){}
}
int random(int x, int y){
  return((rand()%y)+x);
}
void resume_write(level level[], prince &prince, vilan soldier[]){
   ofstream fout("backup.txt",ios::out);
   fout.write((char*) &prince, sizeof(prince));
   for(int i=0;i<10;i++)
   fout.write((char*) &soldier[i], sizeof(vilan));
   for(int i=0;i<8;i++)
   fout.write((char*) &level[i], sizeof(level));
   fout<<lpno<<" "<<hpno<<" "<<coino;
   fout.close();
}
void resume_read(level level[], prince &prince, vilan soldier[]){
   ifstream fin("backup.txt",ios::in);
   fin.read((char*) &prince, sizeof(prince));
   for(int i=0;i<10;i++)
```

```
    fin.read((char*) &soldier[i], sizeof(vilan));
    for(int i=0;i<8;i++)
    fin.read((char*) &level[i], sizeof(level));
    fin>>lpno>>hpno>>coino;
    fin.close();
}
void init();
void deinit();
void controls(){
    BITMAP *c1, *c2;
    c1=load_bitmap("inst1.bmp", NULL);
    c2=load_bitmap("inst2.bmp", NULL);
    name_intro(0,"Controls",2.5,0);
    try_intro1(c1);
    try_intro1(c2);
    rest(200);
}
void levelend() {
    ofstream finout("hscore.txt",ios::app);
    BITMAP *h1=load_bitmap("heart1.bmp", NULL);
    BITMAP *h2=load_bitmap("heart2.bmp", NULL);
    BITMAP *bg=load_bitmap("scorer.bmp", NULL);
    stretch_blit(bg, screen, 0, 0, bg->w, bg->h, 0, 0, SCREEN_W,
SCREEN_H);
    coino+=coinol;
    ALFONT_FONT *font;
    font=alfont_load_font("PAPYRUS.ttf");
    alfont_text_mode(-1);
    alfont_set_font_size(font, 70);
    alfont_textprintf(screen, font, 1200, 15, makecol(255,255,255), "%d",coino);
    alfont_textprintf(screen, font, 1200, 75, makecol(255,255,255), "%d",hpno);
    alfont_textprintf(screen, font, 1200, 135, makecol(255,255,255), "%d",lpno);
    alfont_textprintf(screen, font, 450, 190, makecol(255,0,0), "        %d",score);
```

```
   alfont_textprintf(screen, font, 450, 250, makecol(255,0,0), "
%d",coinol*10);
   alfont_textprintf(screen, font, 450, 310, makecol(255,0,0), "
%d",soldierno*100);
   int lexp=int(coinol*1.5+soldierno*40+lpnol*5+hpnol*5);
   alfont_textprintf(screen, font, 450, 400, makecol(255,0,0), "       %d
percent",lexp);
   points =soldierno*100+score+coinol*10+lexp*10;
   alfont_textprintf(screen, font, 450, 470, makecol(255,0,0), "       %d",points);
   rest(4000);
   masked_stretch_blit(h2, screen, 0, 0, h2->w, h2->h, 400, 600, 300, 60 );
   for(int i=0; i<points/10; i++){
   masked_stretch_blit(h1, screen, 0, 0, i, h1->h, 400, 600, (300*i/1000), 60 );
   rest(1);
   }
    finout<<points<<" ";
    finout.close();
   rest(1000);
}
int exit_game(){
  BITMAP *ex=load_bitmap("exit_portal.bmp", NULL);
   stretch_blit(ex, screen, 0, 0, ex->w, ex->h, 0, 0, SCREEN_W,
SCREEN_H);
   ALFONT_FONT *font;
   font=alfont_load_font("Altgotisch.ttf");
   alfont_text_mode(-1);
   alfont_set_font_size(font, 75);
   int select=0;
   while(1) {
        if(key[KEY_LEFT] and select<2)select++;
        if(key[KEY_RIGHT] and select>0)select--;
        switch (select){
            case 0:
```

```
                stretch_blit(ex, screen, 0, 0, ex->w, ex->h, 0, 0, SCREEN_W,
SCREEN_H);
                alfont_set_font_size(font, 75);
                alfont_textout(screen, font, "YES", 200, 400, makecol(255, 255,
255));
                alfont_set_font_size(font, 100);
                alfont_textout(screen, font, "NO", 500, 400, makecol(255, 0, 0));
                vsync();
                break;
            case 1:
                stretch_blit(ex, screen, 0, 0, ex->w, ex->h, 0, 0, SCREEN_W,
SCREEN_H);
                alfont_set_font_size(font, 75);
                alfont_textout(screen, font, "NO", 500, 400, makecol(255, 255,
255));
                alfont_set_font_size(font, 100);
                alfont_textout(screen, font, "YES", 200, 400, makecol(255, 0,
0));
        }
        if (key[KEY_ENTER]){
          if(select==0) return 0;
          else{
              name_intro(0,"Game Over",3.3);
              return 1;
          }
        }
    }
}
void store_highscore(long score){//dont make it &score////
    fstream finout("hscore.txt", ios::in | ios::app );
    finout<<score<<" ";
    finout.close();
}
```

```
void powerdisp(){
    ALFONT_FONT *font;
    font=alfont_load_font("RAGE.ttf");
    alfont_set_font_size(font, 100);
    alfont_text_mode(-1);
   BITMAP *powerdisp1=load_bitmap("pp.bmp",NULL);
   blit(powerdisp1, screen, 0, 0, 400, 100, powerdisp1->w, powerdisp1->h);
   alfont_textprintf(screen, font, 700,250, makecol(255,255,255), "%d",lpno);
   alfont_textprintf(screen, font, 700,350, makecol(255,255,255), "%d",hpno);
   alfont_textprintf(screen, font, 700,450, makecol(255,255,255), "%d",coino);
   rest(200);
   while(1){if(key[KEY_ENTER])break;}
}
void hscr(){
   ALFONT_FONT *font;
   font=alfont_load_font("PAPYRUS.ttf");
   alfont_text_mode(-1);
   alfont_set_font_size(font, 75);
   BITMAP *h=load_bitmap("hh.bmp",NULL);
   stretch_blit(h, screen, 0, 0, h->w,h-
>h,0,0,SCREEN_W,SCREEN_H);
   ifstream fin("hscore.txt",ios::in);
   int a[5]={0,0,0,0,0},g=0,u=0;
   for(int c=0;c<5;c++){
     for(int i=0;i<1000;i++){
        fin>>u;
        if(u>g and u!=a[0] and u!=a[1] and u!=a[2] and u!=a[3] and u!=a[4]){g=u;}
     }
     a[c]=g;
     alfont_textprintf(screen, font, 300, (c+5)*50, makecol(255, 255, 255),
"%d",a[c]);
   }
   rest(200);
```

```
    while(key[KEY_ENTER]){}
}
void store(){
    BITMAP *st[7];
    int select=1;
    ALFONT_FONT *font;
     font=alfont_load_font("RAGE.ttf");
     alfont_set_font_size(font, 50);
     alfont_text_mode(-1);
    for(int i=0; i<7;i++)st[i]=load_bitmap(pow[8+i],NULL);
    for(;!key[KEY_ESC];){
        if(key[KEY_RIGHT] and select<6){select++;rest(100);}
        if(key[KEY_LEFT] and select>0){select--;rest(100);}
        alfont_textprintf(screen, font, 1200,30, makecol(255,255,255),
"%d",coino);
        alfont_textprintf(screen, font, 1200,85, makecol(255,255,255),
"%d",lpno);
        alfont_textprintf(screen, font, 1200,140, makecol(255,255,255),
"%d",hpno);
        alfont_textout(screen, font, "Press ESC to go back", 900, 650,
makecol(255, 255, 255));
        stretch_blit(st[select], screen, 0, 0, st[select]->w, st[select]->h, 0, 0,
SCREEN_W, SCREEN_H);
        if (key[KEY_ENTER]){
            if (select==1) if(hpno-5>=0){hpno-=5;lpno++;}
            if (select==2) if(coino-60>=0){coino-=60;lpno++;}
            if (select==3) if(lpno-1>=0){hpno+=5;lpno--;}
            if (select==4) if(coino-15>=0){hpno++;coino-=15;}
            if (select==5) if(lpno-1>=0){coino+=50;lpno--;}
            if (select==6) if(hpno-1>=0){coino+=10;hpno--;}
            if (select==0) break;
            powerdisp();
            rest(200);
```

```
        }
        vsync();
    }
}
void options(){
    int select=0;
    BITMAP *intrr[5];
    for(int i=0;i<5;i++)intrr[i]=load_bitmap(pow[i+3],NULL);
    for(;;){
        if(key[KEY_DOWN] and select<4){select++;rest(100);}
        if(key[KEY_UP] and select>0){select--;rest(100);}
        stretch_blit(intrr[select], screen, 0, 0, intrr[select]->w, intrr[select]->h, 0,
0, SCREEN_W, SCREEN_H);
        if (key[KEY_ENTER]){
          if (select==0){
                  break;
          }
          if(select==1){
                  controls();
                  }
          if(select==2){store();}
          if (select==4){
                  if(exit_game())exit(1);
                  rest(100);
          }
        }
    }
}
int levelintr(level l[]){
    l[0].lock=1;
    ALFONT_FONT *font,*font2,*font3;
     font=alfont_load_font("zombie.ttf");
     font2=alfont_load_font("Elektra.ttf");
```

```
    font3=alfont_load_font("RAGE.ttf");
    BITMAP *h1=load_bitmap("heart1.bmp", NULL);
    BITMAP *h2=load_bitmap("heart2.bmp", NULL);
    alfont_text_mode(-1);
    alfont_set_font_size(font, 100);
    alfont_set_font_size(font2, 50);
    alfont_set_font_size(font3, 50);
   BITMAP *popbg=load_bitmap("pop.bmp", NULL);
   BITMAP *lvlsel=load_bitmap("lvl.bmp", NULL);
   stretch_blit(popbg, screen, 0, 0, popbg->w, popbg->h, 0, 0,
SCREEN_W, SCREEN_H);
   int select=0,cont=0;
    for(;;){
        if(key[KEY_RIGHT] and select<5){select++;rest(100);}
        if(key[KEY_LEFT] and select>0){select--;rest(100);}
        stretch_blit(popbg, screen, 0, 0, popbg->w, popbg->h, 0, 0,
SCREEN_W, SCREEN_H);
        alfont_textprintf(screen, font3, 1200, 60, makecol(255,255,255),
"%d",coino);
        alfont_textprintf(screen, font3, 1200, 130, makecol(255,255,255),
"%d",hpno);
        alfont_textprintf(screen, font3, 1200, 200, makecol(255,255,255),
"%d",lpno);
        if (select){
          stretch_blit(lvlsel, screen, (select-1)*500, l[select-1].lock*300, 500,
300, 0, 300, 350, 200 );
        }
        blit(lvlsel, screen, select*500, l[select].lock*300, 400, 250, 500, 300);
        alfont_textout(screen, font, intr[select][0], 450, 550, makecol(0, 0, 0));
        if(l[select].lock) alfont_textout(screen, font2, "Play", 600, 450,
makecol(255, 255, 255));
        else alfont_textout(screen, font2, "Locked", 550, 450, makecol(255,
255, 255));
```

```
        alfont_textout(screen, font, intr[select][0], 450, 550, makecol(0, 0, 0));
        stretch_blit(lvlsel, screen, (select+1)*500, l[select+1].lock*300, 500, 300,
1000, 300, 350, 200 );
        masked_stretch_blit(h2, screen, 0, 0, h2->w, h2->h, 500, 500, 300, 60 );
        for(int i=0; i<l[select].maxscore/10; i++){
        masked_stretch_blit(h1, screen, 0, 0, i, h1->h, 500, 500, (300*i/1000),
60 );
        }
        vsync();
        if (key[KEY_ENTER]){
           if(select==5)options();
           if(l[select].lock)
           return select;
        }
    }
}
void introduction(){
    BITMAP *popbg2;
    popbg2 = load_bitmap ("firstbg.bmp", NULL);
    ALFONT_FONT *font,*font2;
    font=alfont_load_font("AlienR.ttf");
    alfont_text_mode(-1);
    alfont_set_font_size(font, 100);
    for (int i=0,c=0; i<1000; i+=5){
       clear_to_color(screen, makecol(1000-i,1000-i,1000-i));
       alfont_textout(screen, font, "SR", i, 200, makecol(255, 255, 255));
       alfont_textout(screen, font, "PRESENTS", 1000-i, 400,
makecol(255, 255, 255));
       rest(1);
       if(i>450 and i<750){rest(40);}
    }
    BITMAP *layer1 =create_bitmap(SCREEN_W, SCREEN_H);
    font=alfont_load_font("Metal.ttf");
```

```
    font2=alfont_load_font("RAGE.ttf");
    popbg2 = load_bitmap ("popbg2.bmp", NULL);
    stretch_blit(popbg2, screen, 0, 0, popbg2->w, popbg2->h, 0, 0,
SCREEN_W, SCREEN_H);
    for( int i=0; i<450; i+=5){
       alfont_set_font_size(font, i);
       alfont_set_font_size(font2, 400-i);
       stretch_blit(popbg2, screen, 450-i, 450-i, popbg2->w-(450-i)*2, popbg2-
>h-(450-i)*2, 0, 0, SCREEN_W, SCREEN_H);
       alfont_textout(screen, font, "Prince Of Persia", 650-i*2, 400-i,
makecol(255, 255, 255));
       alfont_textout(screen, font2, "Struggle for Supremecy", 0+i, 400,
makecol(255, 255, 255));
       rest(10);
       vsync();
       if(i>250 and i<400)rest(100);
    }
    clear_bitmap(layer1);
    options();
}
void drawpowerups(){
          lp.draww();
          hp[0].draww();
          hp[1].draww();
          for(int a=0;a<=30;a++)coins[a].draww();
}
void level_intr(int &levelno){
   BITMAP *intro;
   ALFONT_FONT *font;
   font=alfont_load_font("alien.ttf");
   alfont_text_mode(-1);
   alfont_set_font_size(font, 100);
   clear_to_color(screen, makecol(0, 0, 0));
```

```
    alfont_textprintf(screen, font, 550, 200, makecol(255, 255, 255), "PART
%d",levelno+1);
    rest(700);
    alfont_set_font_size(font, 50);
    name_intro(2,intr[levelno][0],3,0,1,0,255,255);
    intro = load_bitmap (intr[levelno][1], NULL);
    stretch_blit(intro, screen, 0, 0, intro->w, intro->h, 0, 0, SCREEN_W,
SCREEN_H);
    for(int i=0;i<4;i++){
        stretch_blit(intro, screen, 0, 0, intro->w, intro->h, 0, 0, SCREEN_W,
SCREEN_H);
        alfont_textprintf(screen, font, 800, 600, makecol(255, 255, 255), "Press
enter to proceed",levelno+1);
        name_intro(1,intrwriteup[levelno][i],3.1,1,2); vsync();
        if(key[KEY_ENTER])break;
    }
}
int resume(){
  BITMAP *b[4];
  b[0]=load_bitmap("R2.bmp",NULL);
  b[1]=load_bitmap("R3.bmp",NULL);
  b[2]=load_bitmap("R4.bmp",NULL);
  b[3]=load_bitmap("R5.bmp",NULL);
  int select=0;
  for(;;){
        if(key[KEY_DOWN] and select<3){select++;rest(100);}
        if(key[KEY_UP] and select>0){select--;rest(100);}
        stretch_blit(b[select], screen, 0, 0, b[select]->w, b[select]->h, 0, 0,
SCREEN_W, SCREEN_H);
        if (key[KEY_ENTER]){
                if(select==0)return 0;
                if(select==1)return 1;
                if(select==2)hscr();
```

```
                    if(select==3){options();return 0;}
            }
        }
    }
    void start_level (int &levelno, level &level, prince &prince, vilan soldier[]){
        prince.no=levelno;
        soldierno=0;
        coinol=0;
        hpnol=0;
        lpnol=0;
        prince.setxy(vilpos[levelno][11][0]-level.getlx(), vilpos[levelno][11][1]-
    level.getly());
        int lx=level.getlx(),ly=level.getly();
        for (int soldiern=0, i=0; soldiern<10; soldiern++,i++){
            soldier[soldiern].vilno=10-i%(levelno+2);
            soldier[soldiern].setsprite(villain[i%(levelno+2)][0],villain[i%(levelno+2)][1]);
            soldier[soldiern].setxy(vilpos[levelno][soldiern][0],
    vilpos[levelno][soldiern][1]);
            soldier[soldiern].no=levelno;
            soldier[soldiern].recreate();
        }
        for (int a=0; a<2; a++){
            hp[a].init(0);
            hp[a].setxy(vilpos[levelno][a+13][0], vilpos[levelno][a+13][1]);
        }
        lp.init(1);
        lp.setxy(vilpos[levelno][15][0], vilpos[levelno][15][1]);
        for(int a=0; a<30; a++){
                coins[a].init(2);
                int n1=random(0,29);
                int n2=random(0,29);
                if (tilemap_level[levelno][n2][n1]==0){
                  coins[a].setxy(n1*100, n2*50);
```

```
        }
        else a--;
    }
}
int pause_game(){
  BITMAP *ex=load_bitmap("pause_portal.bmp", NULL);
   stretch_blit(ex, screen, 0, 0, ex->w, ex->h, 0, 0, SCREEN_W,
SCREEN_H);
   ALFONT_FONT *font;
   font=alfont_load_font("Altgotisch.ttf");
   alfont_text_mode(-1);
   alfont_set_font_size(font, 75);
   int select=0;
   while(1) {
        if(key[KEY_RIGHT] and select<2)select++;
        if(key[KEY_LEFT] and select>0)select--;
        switch (select){
            case 1:
                stretch_blit(ex, screen, 0, 0, ex->w, ex->h, 0, 0, SCREEN_W,
SCREEN_H);
                alfont_set_font_size(font, 75);
                alfont_textout(screen, font, "YES", 200, 400, makecol(255, 255,
255));
                alfont_set_font_size(font, 100);
                alfont_textout(screen, font, "NO", 1000, 400, makecol(255, 0,
0));
                break;
            case 0:
                stretch_blit(ex, screen, 0, 0, ex->w, ex->h, 0, 0, SCREEN_W,
SCREEN_H);
                alfont_set_font_size(font, 75);
                alfont_textout(screen, font, "NO", 1000, 400, makecol(255, 255,
255));
```

```
                alfont_set_font_size(font, 100);
                alfont_textout(screen, font, "YES", 200, 400, makecol(255, 0,
0));
                vsync();
        }
        if (key[KEY_ENTER]){
          if(select==0) return 0;
          else return 1;
        }
    }
}
int checkxy (prince &prince, vilan &soldier){
  if (prince.getx()>soldier.getx()) soldier.dir=0;
  else soldier.dir=1;
  if (prince.getx()>soldier.getx()-150 and prince.getx()<soldier.getx()+50 and
prince.gety()==soldier.gety() and prince.dir!=soldier.dir) return 1;
  else return 0;
}
void play_level(int &levelno, level level[], prince &prince, vilan soldier[]){
        if (key[KEY_H]){
                if(hpno){
                        prince.incr_health();
                        hpno--;
                        }
                powerdisp();
        }
        if (key[KEY_LCONTROL] or key[KEY_RCONTROL]){
          prince.chdefend();
        }
        if (key[KEY_F]){
          if (prince.fight==true){
            prince.fight=false;
            prince.closesword();
```

```
        }
        else{
            prince.fight=true;
            prince.closesword();
        }
      }
      if (key[KEY_LEFT]){
        prince.move_left();
        for(int a=0; a<10; a++)soldier[a].draw();
      }
      else if (key[KEY_RIGHT]){
        prince.move_right();
        for(int a=0; a<10; a++)soldier[a].draw();
      }
      else if (key[KEY_SPACE]){
        if
(tilemap_level[levelno][int((prince.gety()+50+level[levelno].getly())/50)][int((pri
nce.getx()+!prince.dir*200+level[levelno].getlx())/100)]==0){
          for (prince.frame=0; prince.frame<6; prince.frame++){
            level[levelno].draw_tilemap();
            prince.jump();
            for(int a=0; a<10; a++)soldier[a].draw();
          }
          }
            prince.frame=0;
      }
      else if(key[KEY_UP]){

if(tilemap_level[levelno][int((prince.gety()+level[levelno].getly())/50)][int((princ
e.getx()+75+level[levelno].getlx())/100)]==0){
            for (prince.frame=0; prince.frame<12; prince.frame++){
              level[levelno].draw_tilemap();
              prince.climb_up1();
```

```
            for(int a=0; a<10; a++)soldier[a].draw();
          }
          rest(100);
          for (prince.frame=0; prince.frame<14; prince.frame++){
            level[levelno].draw_tilemap();
            prince.climb_up2();
            for(int a=0; a<10; a++)soldier[a].draw();
          }
          prince.frame=0;
        }
      }
      else if(key[KEY_DOWN]){

if(tilemap_level[levelno][int((prince.gety()+150+level[levelno].getly())/50)][int((
prince.getx()+prince.dir*100+level[levelno].getlx())/100)]==0){
          for (prince.frame=0; prince.frame<14; prince.frame++){
            level[levelno].draw_tilemap();
            prince.climb_down1();
            for(int a=0; a<10; a++)soldier[a].draw();
          }
          for (prince.frame=0; prince.frame<12; prince.frame++){
            level[levelno].draw_tilemap();
            prince.climb_down2();
            for(int a=0; a<10; a++)soldier[a].draw();
          }
          prince.frame=0;
        }
      }
      else{
        prince.draw();
        for(int a=0; a<10; a++)soldier[a].draw();
        score-=2;
      }
```

```
if (key[KEY_ENTER]){
        prince.incr_x();
        for (prince.frame=0; prince.frame<6; prince.frame++){
          prince.hit_side();
          for(int a=0; a<10; a++)soldier[a].draw();
          rest(5);
          level[levelno].draw_tilemap();
        }
        prince.decr_x();
        for (int a=0; a<10; a++){
          if (soldier[a].live and checkxy(prince, soldier[a])){
            soldier[a].decr_health(soldier[a].vilno*6);
            score++;
          }
        }
}
else
   for (int a=0; a<10; a++){
     if (soldier[a].live and checkxy(prince, soldier[a])){
       if (coun==20){
        for (soldier[a].frame=0; soldier[a].frame<6; soldier[a].frame++){
          level[levelno].draw_tilemap();
          prince.draw();
          soldier[a].hit_side();
        }
        if(!prince.defend())prince.decr_health();
        else{
          prince.incr_x();
          for (prince.frame=0; prince.frame<6; prince.frame++){
            prince.hit_side();
            for(int a=0; a<10; a++)soldier[a].draw();
            rest(5);
            level[levelno].draw_tilemap();
```

```
                }
                prince.decr_x();
                for (int a=0; a<10; a++){
                  if (soldier[a].live and checkxy(prince, soldier[a])){
                    soldier[a].decr_health(soldier[a].vilno*6);
                    score++;
                  }
                }
              }
              coun=0;
            }
            coun++;
          }
        }
        if (key[KEY_Z]){
                prince.incr_x();
                for (prince.frame=0; prince.frame<4; prince.frame++){
                  prince.hit_down();
                  for(int a=0; a<10; a++)soldier[a].draw();
                  rest(5);
                  level[levelno].draw_tilemap();
                }
                prince.decr_x();
                for (int a=0; a<10; a++){
                  if (soldier[a].live and checkxy(prince, soldier[a])){
                    soldier[a].decr_health(soldier[a].vilno*3);
                    score++;
                  }
                }
      }
      else
        for (int a=0; a<10; a++){
          if (soldier[a].live and checkxy(prince, soldier[a])){
```

```
            if (coun==20){
              for (soldier[a].frame=0; soldier[a].frame<6; soldier[a].frame++){
                 level[levelno].draw_tilemap();
                 prince.draw();
                 soldier[a].hit_down();
              }
               if(!prince.defend())prince.decr_health();
              else{
                 prince.incr_x();
                 for (prince.frame=0; prince.frame<4; prince.frame++){
                   prince.hit_down();
                   for(int a=0; a<10; a++)soldier[a].draw();
                   rest(5);
                   level[levelno].draw_tilemap();
                 }
                 prince.decr_x();
                 for (int a=0; a<10; a++){
                   if (soldier[a].live and checkxy(prince, soldier[a])){
                     soldier[a].decr_health(soldier[a].vilno*3);
                     score++;
                   }
                 }
              }
              coun=0;
            }
            coun++;
          }
        }
        if (key[KEY_X]){
                 prince.incr_x();
                 for (prince.frame=4; prince.frame<7; prince.frame++){
                   prince.hit_side();
                   for(int a=0; a<10; a++)soldier[a].draw();
```

```
                rest(5);
                level[levelno].draw_tilemap();
              }
              prince.decr_x();
              for (int a=0; a<10; a++){
                if (soldier[a].live and checkxy(prince, soldier[a])){
                  soldier[a].decr_health(soldier[a].vilno*3);
                  score++;
                }
              }
        }
        else
          for (int a=0; a<10; a++){
            if (soldier[a].live and checkxy(prince, soldier[a])){
              if (coun==20){
                for (soldier[a].frame=0; soldier[a].frame<6; soldier[a].frame++){
                  level[levelno].draw_tilemap();
                  prince.draw();
                  soldier[a].hit_down();
                }
                 if(!prince.defend())prince.decr_health();
                else{
                  prince.incr_x();
                  for (prince.frame=4; prince.frame<7; prince.frame++){
                    prince.hit_down();
                    for(int a=0; a<10; a++)soldier[a].draw();
                    rest(5);
                    level[levelno].draw_tilemap();
                  }
                  prince.decr_x();
                  for (int a=0; a<10; a++){
                    if (soldier[a].live and checkxy(prince, soldier[a])){
                      soldier[a].decr_health(soldier[a].vilno*3);
```

```
                    score++;
                }
            }
        }
        coun=0;
    }
    coun++;
}
}
if (key[KEY_A]){
        level[levelno].decr_lx();
        prince.incr_x();
        for(int a=0; a<10; a++)soldier[a].incr_x();
        hp[0].incr_x();
        hp[1].incr_x();
        lp.incr_x();
        for(int a=0; a<30; a++)coins[a].incr_x();
}
if (key[KEY_D]){
        level[levelno].incr_lx();
        prince.decr_x();
        for(int a=0; a<10; a++)soldier[a].decr_x();
        hp[0].decr_x();
        hp[1].decr_x();
        lp.decr_x();
        for(int a=0; a<30; a++)coins[a].decr_x();
}
if (key[KEY_W]){
        level[levelno].decr_ly();
        prince.incr_y();
        for(int a=0; a<10; a++)soldier[a].incr_y();
        hp[0].incr_y();
        hp[1].incr_y();
```

```
                lp.incr_y();
                for(int a=0; a<30; a++)coins[a].incr_y();
        }
        if (key[KEY_S]){
                level[levelno].incr_ly();
                prince.decr_y();
                for(int a=0; a<10; a++)soldier[a].decr_y();
                hp[0].decr_y();
                hp[1].decr_y();
                lp.decr_y();
                for(int a=0; a<30; a++)coins[a].decr_y();
        }
        if (key[KEY_P]){
                if(pause_game()){
                        options();
                        }
        }
        if
(tilemap_level[levelno][int((prince.gety()+50+level[levelno].getly())/50)][int((pri
nce.getx()+75+level[levelno].getlx())/100)]!=0)
          {
            if(key[KEY_RIGHT] and !prince.dir)prince.decr_x();
            if(key[KEY_LEFT] and prince.dir)prince.incr_x();
if(key[KEY_SPACE]) if(prince.dir)prince.decr_x(); else prince.incr_x();
          }
      for (int a=0; a<2; a++){
        if (hp[a].used() and prince.getx()>hp[a].getx()-50 and
prince.getx()<hp[a].getx()+50 and prince.gety()>hp[a].gety()-50 and
prince.gety()<hp[a].gety()+50){
          hp[a].useit();
          hpnol++;
        }
      }
```

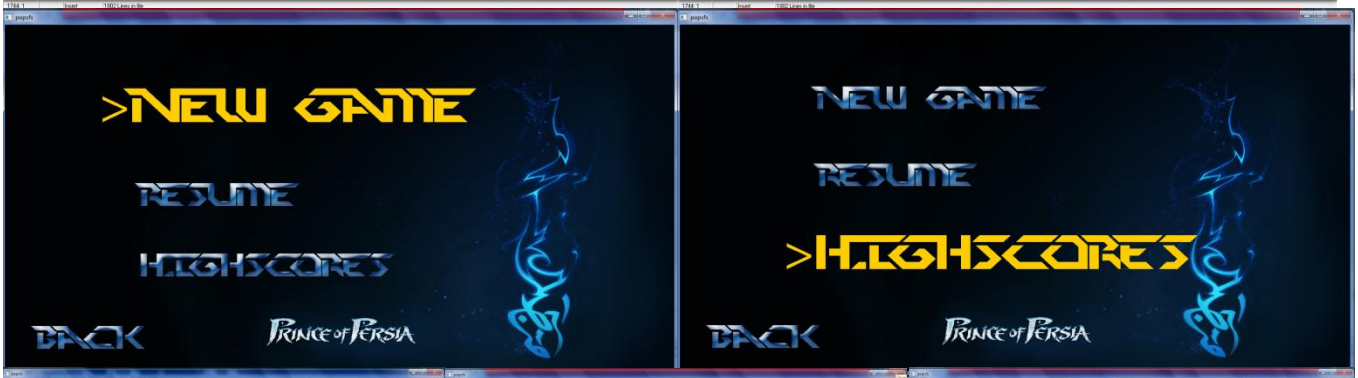
```
        for (int a=0; a<30; a++){
          if (coins[a].used() and prince.getx()>coins[a].getx()-50 and
prince.getx()<coins[a].getx()+50 and prince.gety()>coins[a].gety()-100 and
prince.gety()<coins[a].gety()+50){
            coins[a].useit();
            coinol++;
          }
        }
        if (lp.used() and prince.getx()>lp.getx()-50 and prince.getx()<lp.getx()+50
and prince.gety()>lp.gety()-50 and prince.gety()<lp.gety()+50){
          lp.useit();
          lpnol++;
        }
        if (prince.check(255, 255, 204)){
                --prince.healt;
        }
        if(prince.healt<=0){
          for (prince.frame=0; prince.frame<5; prince.frame++){
              prince.die();
              level[levelno].draw_tilemap();
              rest(100);
           }
        }
        for (int a=0; a<10; a++)
        if(soldier[a].live and soldier[a].gethealt()<=0){
          for (soldier[a].frame=0; soldier[a].frame<5; soldier[a].frame++){
              soldier[a].die();
              level[levelno].draw_tilemap();
              rest(100);
           }
           soldier[a].kill();
        }
        score++;
```

```
        prince_health=prince.healt;
        for (int i=-10; i<11; i++)
                for (int j=-10; j<11; j++)
                  if(getpixel(screen, prince.getx()+75+i,
prince.gety()+115+j)==makecol(0, 50, 255))
{
  level_cleared=true;
  level[levelno+1].lock=1; break;
}                    ///true=1, false=0///
        if(prince.defend()) prince.chdefend();
        vsync();

}
int main() {
        init();
        srand(time(0));
        level level[8];
        prince prince;
        vilan soldier[10];
        prince.setsprite("spleft.bmp", "spright.bmp");
        introduction();
        while(1){
   if(resume())resume_read(level, prince, soldier);
          for (int levelno=0; live and levelno<8; levelno++){
        levelno=levelintr(level);
        start_level (levelno, level[levelno], prince, soldier);
        level_intr (levelno);
        level_cleared=false;
              while (live and !key[KEY_ESC]) {
              level[levelno].draw_tilemap();
              play_level (levelno, level, prince, soldier);
              if (level_cleared){resume_write(level, prince, soldier); break;}
              rest(10);
```

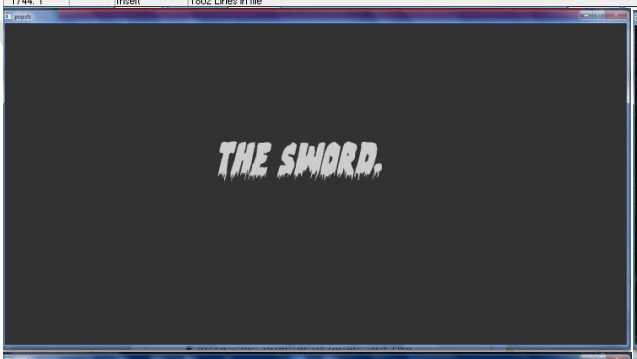
```
            vsync();
        }
           levelend();
           level[levelno].maxscore=points;
           if(lpno){
            live=1;
            lpno--;
            prince.incr_health();
            prince_health=prince.healt;
       }
      if (!level_cleared)start_level (levelno, level[levelno], prince, soldier);
    }
   options();
   }
        deinit();
        return 0;
}
END_OF_MAIN()
void init() {
        int depth, res;
        allegro_init();
        depth = desktop_color_depth();
        if (depth == 0) depth = 32;
        set_color_depth(depth);
        res = set_gfx_mode(GFX_AUTODETECT_WINDOWED,
1352, 700, 0, 0);
        if (res != 0) {
                allegro_message(allegro_error);
                exit(-1);
        }
        install_timer();
        install_keyboard();
        install_mouse();
```

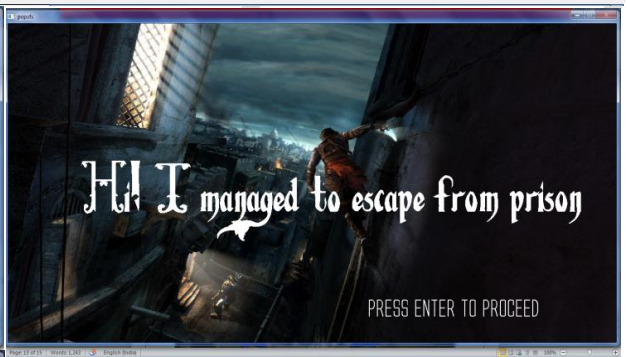
```
        alfont_init();
}

void deinit() {
        clear_keybuf();
}
```

Screenshots



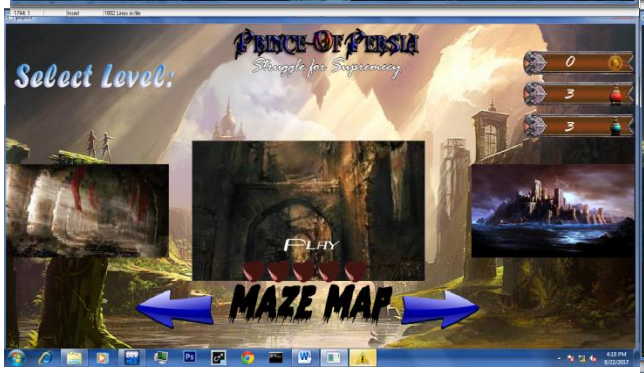

popsfs
PRINCE OF PERSIA
Struggle for Supremecy
>Buy Life potion
>Buy Health potion
>Buy Coins
X5
X60
X15
X30
X10
1744: 1
Insert
1802 Lines in file


THE SWORD.


Hi! I managed to escape from prison
PRESS ENTER TO PROCEED


BACKGROUND CONTROLS:
screen movement
UP
W
LEFT
A
D
RIGHT
S
DOWN
PRINCE OF PERSIA


PRINCE CONTROLS:
UP
MOVE LEFT
MOVE RIGHT
DOWN
ENTER
HIT COMBO
SPACE BAR
JUMP
FIGHT MODE
DEFEND
PAUSE
USE ONE HEALTH POTION


PRINCE OF PERSIA
Struggle for Supremecy
Select Level:
PLAY
MAZE MAP


PRINCE OF PERSIA
Struggle for Supremecy
Select Level:
PLAY
THE ESCAPE


popsfs
Prince Of Persia
Struggle for Supremecy
Select Level:
12
2
2
Play
THE SWORD.
Page: 18 of 25
Words: 1,248
English (U.S.)
100%
Prince Of Persia
Struggle for Supremecy
Level Points: 1256
Coin Bonus: 120
Kill Bonus: 400
Level explored: 183 percent
Total Points: 3606
Prince Of Persia
Struggle for Supremecy
Level Points: 1287
Coin Bonus: 20
Kill Bonus: 0
Level explored: 3 percent
Total Points: 1337
691 Score
Prince Of Persia
Struggle for supremecy
press enter to continue...
567 Score
156 Score

Future Enhancements

This game is a wonderful creation and can be enhanced by-

- adding background music
- increasing number of levels and the difficulty at every level
- better graphics
- making more simulations of the prince
- increasing the types and number of powers which the prince can collect and soldiers.

Bibliography

- Computer science C++ for class 11~ Sumita Arora.
- Computer science C++ for class 12~ Sumita Arora.
- Allegro manual.
- Alfont manual.