

**2011-2012**

Joel M. Pinto

C++ Project

XII-Sci

Roll No.:

Sokoban

Index

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Name** | **Description** | **Pg. No.** |
| 1 | Acknowledgements | Credits to our benefactors | 3 |
| 2 | Sokoban.cpp | Contains main() | 4 |
| 3 | Map.cpp | Handles navigation in a level | 6 |
| 4 | Image.cpp | Handles sprites (images) | 11 |
| 5 | ImageLdr.cpp | Image Loader; Handles loading sprites for cells | 13 |
| 6 | KeyBrd.cpp | Encapsulates Keyboard input | 14 |
| 7 | LevelSys.cpp | Level System; Handles loading levels | 15 |
| 8 | GameEng.cpp | The game engine for Sokoban | 18 |
| 9 | MainMenu.cpp | Main Menu for the game | 20 |
| 10 | Screenshots | Screen captures of the game | 25 |

Acknowledgements

First and foremost, we would like to express our gratitude towards Principal Sir, **Mr. Joy Thaliyachira K.** for providing us this opportunity to learn programming in practice. We are also grateful to the school for the computer lab infrastructure.

We are extremely indebted towards our talented Computer Science Instructor, **Mr. Eroz T. Awari** for his guidance, effort, patience, and attention to every minute detail in teaching us effective programming skills.

We would like to give credit to all the testers of the code including Vishesh Sanghvi, Keyur Patel, Michael Pinto and Jennifer Pinto. They have also given suggestions for the improvement of the project. We also thank David W. Skinner for letting us use his Microban level pack (155 levels) in our project.

Sokoban.cpp

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\

Project name: Sokoban

Release version: 1.0

Programmers: Nishit Vashi, Jatin Patel, Joel M. Pinto

Created: 09-10-2011 06:42 pm

Completed: 26-12-2011 11:33 pm

About:

This C++ project aims to replicate the popular classic puzzle

game Sokoban.

[From Wikipedia - http://en.wikipedia.org/wiki/Sokoban]

Sokoban is a type of transport puzzle, in which the player

pushes boxes or crates around in a warehouse, trying to get

them to storage locations. Sokoban was created in 1981 by

Hiroyuki Imabayashi, and published in 1982 by Thinking Rabbit, a

Software house based in Takarazuka, Japan.

Rules:

1. Only one box can be pushed at a time.

2. A box cannot be pulled.

3. The player cannot walk through boxes or walls.

4. The puzzle is solved when all boxes are located at storage

locations.

Program files:

Sokoban.cpp - Contains main()

Map.cpp - Handles navigation in a level

Image.cpp - Handles sprites (images)

ImageLdr.cpp- Image Loader; Handles loading sprites for cells

Keybrd.cpp - Encapsulates Keyboard input

LevelSys.cpp- Level System; Handles loading levels

GameEng.cpp - The game engine for Sokoban

MainMenu.cpp- Main Menu for the game

\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Include libraries

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

//Include code files

#include "C:\SOKOBAN\mainmenu.cpp"

#include "C:\SOKOBAN\gameeng.cpp"

//Main body of program

int main()

{

//Initialize graphics mode

int gdriver=DETECT, gmode;

initgraph(&gdriver, &gmode, "..\\BGI");

int errorcode=graphresult();

if (errorcode!=grOk)

{

cout<<"Could not load graphics mode\n\n"

<<"Error Description: "<<grapherrormsg(errorcode)

<<"\nExiting...";

getch();

return 1;

}

//Show the Main Menu

while(1)

{

switch(Main\_Menu::show()) //Check the user's choice

{

case 1: //Play

Game\_Engine ge;

ge.start();

break;

case 2: //About

Main\_Menu::display\_about();

break;

case 3: //Help

Main\_Menu::display\_help();

break;

case 4: //Tutorials

Game\_Engine tut("C:\\sokoban\\levels\\tut",5);

tut.start();

break;

case 5: //Exit

//Show a confirmation dialog

if (!Main\_Menu::confirm\_exit()) break;

closegraph(); //Go back to text mode

return 0;

}

}

}

Map.cpp

/\*

Map.cpp - Handles navigation in a level

Part of the Sokoban project

(See Sokoban.cpp for details)

\*/

// The following if condition prevents accidental

// double inclusion in other files.

#ifndef MAP\_CPP

#define MAP\_CPP

#include<fstream.h>

#include<stdio.h>

#include "C:\SOKOBAN\imageldr.cpp"

//map data - possible cell values

const int c\_empty=0;

const int c\_wall=1;

const int c\_box=2;

const int c\_target=4;

const int c\_player=8;

//Map class and function declarations

class Map

{

private:

int r; //no. of rows in the cell grid

int c; //no. of columns in the cell grid

int grid[30][40]; //max 30 rows by 40 columns - stores the grid data

int plrx; //x-coordinate of player position

int plry; //y-coordinate of player position

int no\_of\_moves;

void move\_plr\_to(int, int, int); //Move player to position (x,y)

public:

int get\_cell(int, int); //Accessor for cell type at (x,y)

void set\_cell(int, int, int); //Sets the cell type at (x,y) with 'cell'

Map(); //Constructor

int load\_from\_file(char \*); //Loads the current instance from a file

int save\_to\_file(char \*); //Saves the current instance to a file

int num\_rows() //Accessor for number of rows

{

return r;

}

int num\_cols() //Accessor for number of columns

{

return c;

}

void num\_rows(int \_r) //Mutator for number of rows

{

if (\_r>=0&&\_r<=30)

r=\_r;

}

void num\_cols(int \_c) //Accessor for number of columns

{

if (\_c>=0&&\_c<=40)

c=\_c;

}

int player\_x() //Accessor for player position (x)

{

return plrx;

}

int player\_y() //Accessor for player position (y)

{

return plry;

}

void player\_x(int x) //Mutator for player position (x)

{

if(x>0&&x<c)

plrx=x;

}

void player\_y(int y) //Mutator for player position (y)

{

if(y>0&&y<r)

plry=y;

}

int num\_moves() //Accessor for number of moves

{

return no\_of\_moves;

}

void try\_move(char, int, int =1);

//Try to move player on 'axis' by 'offset' cells

//Also call show\_num\_moves() if show\_moves!=0

int check\_win(); //Check if the level has been completed

void update\_image(int, int); //Updates the image of the cell at (x,y)

void show\_num\_moves();

};

//function definitions

//Constructor

Map::Map()

{

//Initialize everything with valid data

r=30;

c=40;

for(int i=0;i<30;++i)

for(int j=0;j<40;++j)

grid[i][j]=0;

plrx=plry=0;

no\_of\_moves=0;

}

//Accessor for cell type at (x,y)

int Map::get\_cell(int x, int y)

{

return grid[y][x]; //Row=y, Column=x

}

//Sets the cell type at (x,y) with 'cell'

void Map::set\_cell(int x, int y, int cell)

{

if (x<0||x>=c || y<0||y>=r)

return; //Cell coordinates out of range, so reject

if (cell>=0&&cell<=12) //12 is max possible cell value

grid[y][x]=cell;

}

//Loads the current instance from a file. Returns 1 if error encountered.

int Map::load\_from\_file(char \*path)

{

ifstream level;

level.open(path,ios::in,ios::binary);

if (!level) return 1; //Error in file opening

if(!level.read((char \*)this,sizeof(Map))) //Store in current instance

{

level.close();

return 1; //Error while reading

}

level.close();

for(int x=0;x<c;++x)

for(int y=0;y<r;++y)

update\_image(x,y);

return 0;

}

//Saves the current instance to a file. Returns 1 if error encountered.

int Map::save\_to\_file(char \*path)

{

ofstream level;

level.open(path,ios::out,ios::binary);

if (!level) return 1; //Error in file creation

//level.seekp(0,ios::beg);

level.write((char \*)this,sizeof(Map)); //Write the current instance

level.close();

return 0;

}

//Try to move player on 'axis' by 'offset' cells

void Map::try\_move(char axis, int offset, int showmoves)

{

//adj1 = 1st adjacent cell in the direction the player wants to move

//adj2 = cell adjacent to 1st adjacent cell

int adj1\_x,adj1\_y,adj2\_x,adj2\_y; //cell position coordinates

int adj1\_cell,adj2\_cell; //cell types

if(axis=='x'||axis=='X')

{

adj1\_x=plrx+offset;

adj1\_y=plry;

adj2\_x=adj1\_x+offset;

adj2\_y=adj1\_y;

}

else if(axis=='y'||axis=='Y')

{

adj1\_x=plrx;

adj1\_y=plry+offset;

adj2\_x=adj1\_x;

adj2\_y=adj1\_y+offset;

}

adj1\_cell=get\_cell(adj1\_x,adj1\_y);

if (adj1\_cell==c\_empty||adj1\_cell==c\_target)

move\_plr\_to(adj1\_x,adj1\_y,showmoves);

else if (adj1\_cell==c\_wall)

return ; //cannot push wall

else if (adj1\_cell&c\_box) //cell has box (with or without target)

{

adj2\_cell=get\_cell(adj2\_x,adj2\_y);

if(adj2\_cell&c\_wall || adj2\_cell&c\_box)

return ;

//cannot push coz box is against a wall or another box

else //Box is pushable

{

//Add box to adj2

set\_cell(adj2\_x,adj2\_y,adj2\_cell|c\_box);

update\_image(adj2\_x,adj2\_y);

//Remove box from adj1

set\_cell(adj1\_x,adj1\_y,adj1\_cell&(~c\_box));

//Move player to adj1

move\_plr\_to(adj1\_x,adj1\_y,showmoves);

}

}

}

//Move player to position (x,y)

//if showmoves!=0, number of moves will be shown

void Map::move\_plr\_to(int x, int y, int showmoves)

{

//Remove player from old cell (plrx,plry)

int playercell=get\_cell(plrx,plry);

set\_cell(plrx,plry,(playercell&(~c\_player)));

update\_image(plrx,plry);

//Send player to new cell (x,y)

playercell=get\_cell(x,y);

set\_cell(x,y,(playercell|c\_player));

update\_image(x,y);

plrx=x;

plry=y;

no\_of\_moves++;

if(showmoves)

show\_num\_moves();

}

//Shows the number of moves on the screen

void Map::show\_num\_moves()

{

char strmoves[20]="";

sprintf(strmoves,"Moves - %d",no\_of\_moves);

//erase old label

setfillstyle(SOLID\_FILL,WHITE);

bar(640,textheight("Level")+20,640-textwidth("Moves - XXX"),

2\*textheight("Level")+20);

//display new label

settextjustify(RIGHT\_TEXT,TOP\_TEXT);

outtextxy(640,textheight("Level")+20,strmoves);

}

//Check if the level has been completed

int Map::check\_win()

{

for(int i=0;i<r;++i)

for(int j=0;j<c;++j)

if(get\_cell(j,i)&c\_target)

if((get\_cell(j,i)&c\_box)==0)

return 0;

return 1;

}

//Updates the image of the cell at (x,y)

void Map::update\_image(int x, int y)

{

int mapx=320-(c\*SPR\_WIDTH)/2;

int mapy=240-(r\*SPR\_HEIGHT)/2;

int cellx=mapx+x\*SPR\_WIDTH;

int celly=mapy+y\*SPR\_HEIGHT;

int c=get\_cell(x,y);

if(c==c\_empty)

{

il.Empty.display(cellx,celly,-1);

return ;

}

if(c==c\_wall)

{

il.Wall.display(cellx,celly,WHITE);

return ;

}

if(c&c\_target)

{

if (c==c\_target)

il.Target.display(cellx,celly,-1);

else

il.Target.display(cellx,celly,WHITE);

}

if(c&c\_box)

{

il.Box.display(cellx,celly,WHITE);

return ;

}

if(c&c\_player)

{

il.Player.display(cellx,celly,WHITE);

}

}

#endif //MAP\_CPP

Image.cpp

/\*

Image.cpp - Handles sprites (images)

Part of the Sokoban project

(See Sokoban.cpp for details)

\*/

// The following if condition prevents accidental

// double inclusion in other files.

#ifndef IMAGE\_CPP

#define IMAGE\_CPP

#define SPR\_HEIGHT 16 //Max height of sprites

#define SPR\_WIDTH 16 //Max width of sprites

#include<graphics.h>

#include<fstream.h>

class Image

{

private:

unsigned char \_pixdat[SPR\_HEIGHT][SPR\_WIDTH]; //pixel data

public:

void display(int, int, int);

int load\_from\_file(char \*); //file must be .raw file type

};

//Loads the current instance from a file. Returns 1 if error encountered.

int Image::load\_from\_file(char \*path)

{

ifstream imgfile;

imgfile.open(path,ios::in|ios::binary);

if (!imgfile) return 1; //Error in file opening

unsigned char dat;

for(int i=0;i<SPR\_HEIGHT;++i)

for(int j=0;j<SPR\_WIDTH;++j)

{

//Read each byte(char) from the file

imgfile.read((char \*)&dat,sizeof(char));

//Each byte in the RAW 4bpp format used by

//Photoshop is 0000KBGR (K=brightness).

//It should be converted to 0000KRGB, so

//swap bit 0 with bit 2 i.e, red and blue

dat=((dat&0x01)<<2)|((dat&0x04)>>2)|(dat&0x0A);

//0x0A = (bin)00001010

//0111 has to be interchanged with 1000 for correct color

if (dat==7) dat=8;

else if (dat==8) dat=7;

\_pixdat[i][j]=dat;

}

imgfile.close();

return 0;

}

void Image::display(int x, int y, int transparency\_key)

{

for(int i=0;i<SPR\_HEIGHT;++i)

for(int j=0;j<SPR\_WIDTH;++j)

{

if (transparency\_key!=-1&&\_pixdat[i][j]==transparency\_key)

continue;

putpixel(x+j,y+i,\_pixdat[i][j]);

//Use these lines instead of putpixel to scale the

//image by 2x:

//setcolor(\_pixdat[i][j]);

//rectangle(x+2\*j,y+2\*i,x+2\*j+1,y+2\*i+1);

}

}

#endif //IMAGE\_CPP

ImageLdr.cpp

/\*

ImageLdr.cpp - Image Loader; Handles loading sprites for cells

Part of the Sokoban project

(See Sokoban.cpp for details)

\*/

// The following if condition prevents accidental

// double inclusion in other files.

#ifndef IMAGELDR\_CPP

#define IMAGELDR\_CPP

#include "C:\SOKOBAN\IMAGE.CPP"

class Image\_Loader

{

public:

Image Empty;

Image Wall;

Image Box;

Image Target;

Image Player;

//Constructor

Image\_Loader()

{

Empty.load\_from\_file("C:\\sokoban\\sprites\\empty.raw");

Wall.load\_from\_file("C:\\sokoban\\sprites\\wall.raw");

Box.load\_from\_file("C:\\sokoban\\sprites\\box.raw");

Target.load\_from\_file("C:\\sokoban\\sprites\\target.raw");

Player.load\_from\_file("C:\\sokoban\\sprites\\player.raw");

}

}il;

#endif //IMAGELDR.CPP

KeyBrd.cpp

/\*

Keybrd.cpp - Encapsulates Keyboard input

Part of the Sokoban project

(See Sokoban.cpp for details)

Uses bioskey() function from bios.h

\*/

// The following if condition prevents accidental

// double inclusion in other files.

#ifndef KEYBRD\_CPP

#define KEYBRD\_CPP

#include<bios.h>

#define KEY\_ESC 1

#define KEY\_F1 59

#define KEY\_UP 72

#define KEY\_DOWN 80

#define KEY\_RIGHT 77

#define KEY\_LEFT 75

#define KEY\_ENTER 28

#define KEY\_P 25

#define KEY\_N 49

#define KEY\_R 19

#define KEY\_Q 16

#define KEY\_Y 21

class Keyboard

{

public:

//Functions are declared static so that they can be used directly

//using the class-name instead of creating an object.

static int check\_key(void)

{

return bioskey(1);

}

static int read\_key(void)

{

return bioskey(0)>>8;

}

};

#endif //KEYBRD\_CPP

LevelSys.cpp

/\*

LevelSys.cpp - Level System; Handles loading levels

Part of the Sokoban project

(See Sokoban.cpp for details)

\*/

// The following if condition prevents accidental

// double inclusion in other files.

#ifndef LEVELSYS\_CPP

#define LEVELSYS\_CPP

#include<string.h>

#include<stdlib.h>

#include"C:\SOKOBAN\Map.cpp"

class Level\_System

{

private:

int curr\_level;

char \*level\_path;

int max\_levels;

Map \*m;

public:

Level\_System() : level\_path("C:\\sokoban\\levels\\level")

{

curr\_level=1;

max\_levels=155;

m=NULL;

}

Level\_System(char \*path, int maxlvls)

{

strcpy(level\_path,path);

curr\_level=1;

max\_levels=maxlvls;

m=NULL;

}

void load\_level(Map \*, int); //Loads (level\_no)th level into map

void change\_level(int); //Changes the level by delta(-1,0 or +1)

};

//Loads (level\_no)th level into map

void Level\_System::load\_level(Map \*map,int level\_no)

{

m=map;

char path[80]="";

strcpy(path,level\_path);

char strlno[5]=""; //string which stores level\_no

itoa(level\_no,strlno,10); //store level\_no in strlno

strcat(path,strlno);

strcat(path,".dat");

cleardevice();

setfillstyle(SOLID\_FILL,WHITE);

floodfill(1,1,WHITE);

if(m->load\_from\_file(path)) //Returns 1 on error else 0

{

//error occurred

settextjustify(CENTER\_TEXT,CENTER\_TEXT);

outtextxy(320,240,"File Error: Could not load level");

}

curr\_level=level\_no;

char strout[20]="Level ";

strcat(strout,strlno);

settextjustify(RIGHT\_TEXT,TOP\_TEXT);

outtextxy(639,0,strout);

m->show\_num\_moves();

//Write help strings if tutorials are being played

char \*tut[] = {

"Press the right arrow key to push the box into the target.",

"Good, now try to push these 2 boxes into those targets.",

"Great! Try this now. TIP: Press [R] to reset a level.",

"Another TIP: Press [P]/[N] for navigating to a previous/next level

respectively.",

"This is the last one. TIP: Press [Q]/[Esc] to go back to the main

menu."};

if (strstr(level\_path,"tut"))

//will return non-zero if "tut" found in path

{

settextjustify(LEFT\_TEXT,TOP\_TEXT);

setcolor(BLACK);

outtextxy(20,400,tut[level\_no-1]);

}

}

//Changes the level by delta(-1,0 or +1)

void Level\_System::change\_level(int delta)

{

int new\_level;

if(delta==0)

{

new\_level=curr\_level;

}

else if(delta==1)

{

if(curr\_level==max\_levels)

new\_level=1;

else

new\_level=curr\_level+1;

}

else if(delta==-1)

{

if(curr\_level==1)

new\_level=max\_levels;

else

new\_level=curr\_level-1;

}

else //Invalid argument

{

return ; //to prevent undefined behaviour

//Assume delta to be either -1, 0 or 1.

}

load\_level(m,new\_level);

}

#endif //LEVELSYS\_CPP

GameEng.cpp

/\*

GameEng.cpp - The game engine for Sokoban

Part of the Sokoban project

(See Sokoban.cpp for details)

\*/

// The following if condition prevents accidental

// double inclusion in other files.

#ifndef GAMEENG\_CPP

#define GAMEENG\_CPP

#include<stdio.h>

#include<dos.h>

#include "C:\SOKOBAN\keybrd.cpp"

#include "C:\SOKOBAN\levelsys.cpp"

class Game\_Engine

{

private:

Map m;

Level\_System ls;

int input(int keycode); //Processes input from keyboard

void win();

public:

//Constructors

Game\_Engine()

{ }

Game\_Engine(char \*level\_path,int max\_levels):ls(level\_path,max\_levels)

{ }

void start(); //Starts the Game Engine

};

//Starts the Game Engine

void Game\_Engine::start()

{

ls.load\_level(&m,1);

while(1)

{

if (Keyboard::check\_key())

if (input(Keyboard::read\_key()) == -1)

break; //-1 is request to exit to main menu

}

}

//Processes input from keyboard

int Game\_Engine::input(int keycode)

{

switch(keycode)

{

case (KEY\_UP):

m.try\_move('y',-1); //Move on y-axis one cell up

if (m.check\_win()) win();

break;

case (KEY\_DOWN):

m.try\_move('y',+1); //Move on y-axis one cell down

if (m.check\_win()) win();

break;

case (KEY\_LEFT):

m.try\_move('x',-1); //Move on x-axis one cell left

if (m.check\_win()) win();

break;

case (KEY\_RIGHT):

m.try\_move('x',+1); //Move on x-axis one cell right

if (m.check\_win()) win();

break;

case (KEY\_P): //Previous level

ls.change\_level(-1);

break;

case (KEY\_N): //Next level

ls.change\_level(+1);

break;

case (KEY\_R): //Reload level

ls.change\_level(0);

break;

case (KEY\_ESC): case (KEY\_Q): //Quit to main menu

return -1;

default: break;

}

return 0;

}

void Game\_Engine::win()

{

delay(200); //To prevent closing of the dialog before being seen

char \*str;

if (m.num\_moves()!=1)

sprintf(str,"You completed this level in %d moves",

m.num\_moves());

else

sprintf(str,"You completed this level in %d move",m.num\_moves());

const int dialogw=textwidth(str)+40; //Dialog box width

const int dialogh=2\*textheight(str)+60; //Dialog box height

setfillstyle(SOLID\_FILL,LIGHTCYAN);

bar(320-dialogw/2,240-dialogh/2,320+dialogw/2,240+dialogh/2);

settextjustify(CENTER\_TEXT,TOP\_TEXT);

outtextxy(320,240-dialogh/2+20,str);

str="Press any key for the next level";

outtextxy(320,240-dialogh/2+40+textheight(str),str);

while(!Keyboard::check\_key());

Keyboard::read\_key(); //dummy call to prevent unpredictable behaviour

ls.change\_level(+1);

}

#endif //GAMEENG\_CPP

MainMenu.cpp

/\*

MainMenu.cpp - Main Menu for the game

Part of the Sokoban project

(See Sokoban.cpp for details)

\*/

// The following if condition prevents accidental

// double inclusion in other files.

#ifndef MAINMENU\_CPP

#define MAINMENU\_CPP

#include<graphics.h>

#include<dos.h>

#include"C:\SOKOBAN\map.cpp"

#include"C:\SOKOBAN\keybrd.cpp"

#include"C:\SOKOBAN\imageldr.cpp"

class Main\_Menu

{

public:

static int show(); //Shows the Main Menu and returns the user's choice

static void display\_help(); //Shows the Help screen

static void display\_about(); //Shows the About screen

static int confirm\_exit(); //Shows an exit confirmation dialog

};

//Shows the Main Menu and returns the user's choice

int Main\_Menu::show()

{

cleardevice();

setfillstyle(SOLID\_FILL,WHITE);

floodfill(1,1,WHITE);

//---------------------SOKOBAN TITLE--------

setcolor(BLACK);

char \*strheading="Sokoban";

settextstyle(9, HORIZ\_DIR, 0);

setusercharsize(4, 3, 4, 3);

settextjustify(LEFT\_TEXT,TOP\_TEXT);

outtextxy(15,0,strheading);

int h=textheight(strheading)+10;

int w=textwidth(strheading);

line(0,h,20+w,h);

//Serifs

line(15+w,h,20+w,h+5);

line(20+w,h,25+w,h+5);

//---------------------MAIN MENU TITLE------

char \*strmenu="Main Menu";

settextstyle(9, HORIZ\_DIR, 0);

setusercharsize(1, 2, 1, 2);

settextjustify(RIGHT\_TEXT,TOP\_TEXT);

outtextxy(630,60,strmenu);

h=textheight(strmenu)+70;

w=textwidth(strmenu);

line(640,h,640-w-20,h);

//Serifs

line(640-w-15,h,640-w-20,h+5);

line(640-w-20,h,640-w-25,h+5);

//---------------------OPTION LABELS--------

//Calculate offset of label position from center

int off=(7\*SPR\_WIDTH)/2 + 5;

//7=no of cells horizontally&vertically, 5=margin

settextjustify(CENTER\_TEXT,CENTER\_TEXT);

settextstyle(9, HORIZ\_DIR, 0);

setusercharsize(1, 3, 1, 3);

outtextxy(320,240-off-textheight("Play")/2,"Play");

outtextxy(320-off-textwidth("Tutorial")/2,240,"Tutorial");

outtextxy(320+off+textwidth("About")/2,240,"About");

outtextxy(320,240+off+textheight("Exit")/2,"Exit");

//---------------------INSTRUCTIONS---------

settextjustify(CENTER\_TEXT,TOP\_TEXT);

settextstyle(9, HORIZ\_DIR, 0);

setusercharsize(1, 3, 1, 3);

h=textheight("T")+5;//5=line spacing

outtextxy(320,480-2\*h,

"To select a menu option, push in that direction");

outtextxy(320,480-h,"Press F1 for Help");

//---------------------MENU OPTIONS---------

Map menu;

menu.load\_from\_file("C:\\sokoban\\levels\\mainmenu.dat");

while(1)

{

if(Keyboard::check\_key())

switch(Keyboard::read\_key())

{

case (KEY\_F1):

return 3;

case (KEY\_UP): //Play

menu.try\_move('y',-1,0);

delay(500);

return 1;

case (KEY\_LEFT): //Tutorials

menu.try\_move('x',-1,0);

delay(500);

return 4;

case (KEY\_RIGHT): //About

menu.try\_move('x',+1,0);

delay(500);

return 2;

case (KEY\_DOWN): //Exit

menu.try\_move('y',+1,0);

delay(500);

//deliberate fall-through

case (KEY\_Q):

case (KEY\_ESC): return 5; //Exit

}

}

}

//Shows the Help screen

void Main\_Menu::display\_help()

{

cleardevice();

char \*strhelp="Help";

setfillstyle(SOLID\_FILL,WHITE);

floodfill(1,1,WHITE);

setcolor(BLACK);

settextstyle(9,HORIZ\_DIR,0);

setusercharsize(1,2,1,2);

settextjustify(RIGHT\_TEXT,TOP\_TEXT);

outtextxy(630,10,strhelp);

int h=textheight(strhelp)+20; //help string underline height

int w=textwidth(strhelp); //help string width

line(640,h,640-w-20,h);

//Serifs

line(640-w-15,h,640-w-20,h+5);

line(640-w-20,h,640-w-25,h+5);

h=50+textheight(strhelp);

line(320,h,320,480-30);

setusercharsize(1,3,1,3);

outtextxy(640/4,h,"How to Play");

outtextxy(640-640/4,h,"Controls");

//How to play

settextjustify(LEFT\_TEXT,TOP\_TEXT);

setusercharsize(1,3,1,4);

h+=textheight("H")+20;

il.Player.display(16,h,-1);

outtextxy(48,h,"®-This is the player");

il.Box.display(16,h+32,-1);

outtextxy(48,h+32,"®-This is a box");

il.Wall.display(16,h+64,-1);

outtextxy(48,h+64,"®-This is a wall");

il.Target.display(16,h+96,-1);

outtextxy(48,h+96,"®-This is a target space");

char line1[]="Aim: You need to move all boxes to";

char line2[]=" the target spaces.";

char line3[]="Rules:";

char line4[]="1 Only one box can be pushed at a";

char line5[]=" time.";

char line6[]="2 A box cannot be pulled.";

char line7[]="3 The player cannot walk through";

char line8[]=" boxes or walls.";

char line9[]="4 The puzzle is solved when all";

char line10[]=" boxes are located at targets";

outtextxy(16,h+128,line1);

outtextxy(16,h+144,line2);

outtextxy(16,h+176,line3);

outtextxy(16,h+192,line4);

outtextxy(16,h+208,line5);

outtextxy(16,h+224,line6);

outtextxy(16,h+240,line7);

outtextxy(16,h+256,line8);

outtextxy(16,h+272,line9);

outtextxy(16,h+288,line10);

//Controls

w=320+16;

outtextxy(w,h,"Arrow keys to move/push");

outtextxy(w,h+32,"P = Previous level");

outtextxy(w,h+64,"N = Next level");

outtextxy(w,h+96,"R = Reset level");

outtextxy(w,h+128,"Q / Esc = Quit to Main Menu");

while (!Keyboard::check\_key()) ;

Keyboard::read\_key();

}

//Shows the About screen

void Main\_Menu::display\_about()

{

cleardevice();

setfillstyle(SOLID\_FILL,LIGHTCYAN);

floodfill(1,1,LIGHTCYAN);

settextjustify(CENTER\_TEXT,CENTER\_TEXT);

settextstyle(9,HORIZ\_DIR,0);

setusercharsize(6,10,6,10);

outtextxy(320,40,"Sokoban");

setusercharsize(3,8,3,8);

settextjustify(CENTER\_TEXT,TOP\_TEXT);

int h=textheight("A")+5;

outtextxy(320,90,"Version 1.0. Coded by:");

outtextxy(320,90+h,"Jatin Patel,Joel Pinto and Nishit Vashi");

char \*lines[]={

" Sokoban is a type of transport puzzle which was created in 1981",

" by Hiroyuki Imabayashi. This project aims to replicate this puzzle.",

" Microban Levels by David W. Skinner (155 Microban levels)",

" http://users.bentonrea.com/~sasquatch/sokoban/",

" Tutorial levels by Joel"

};

for(int i=0;i<5;++i)

outtextxy(320,90+h\*(i+2),lines[i]);

while (!Keyboard::check\_key()) ;

Keyboard::read\_key();

}

//Shows a confirmation dialog at the time of exit

int Main\_Menu::confirm\_exit()

{

char \*str=" Are you sure you want to exit?";

const int dialogw=textwidth(str)+40; //Dialog box width

const int dialogh=2\*textheight(str)+60; //Dialog box height

setfillstyle(SOLID\_FILL,LIGHTCYAN);

bar(320-dialogw/2,240-dialogh/2,320+dialogw/2,240+dialogh/2);

settextjustify(CENTER\_TEXT,TOP\_TEXT);

outtextxy(320,240-dialogh/2+20,str);

str="Y=Yes, N=No";

outtextxy(320,240-dialogh/2+40+textheight(str),str);

while(1)

{

if(Keyboard::check\_key())

switch(Keyboard::read\_key())

{

case (KEY\_N): return 0;

case (KEY\_Y): return 1;

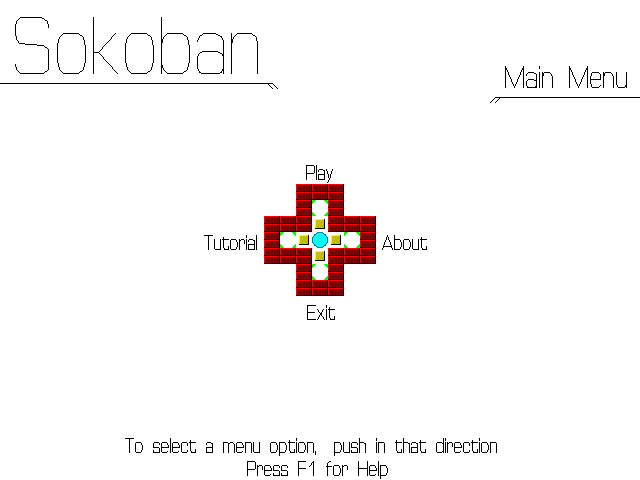
}

}

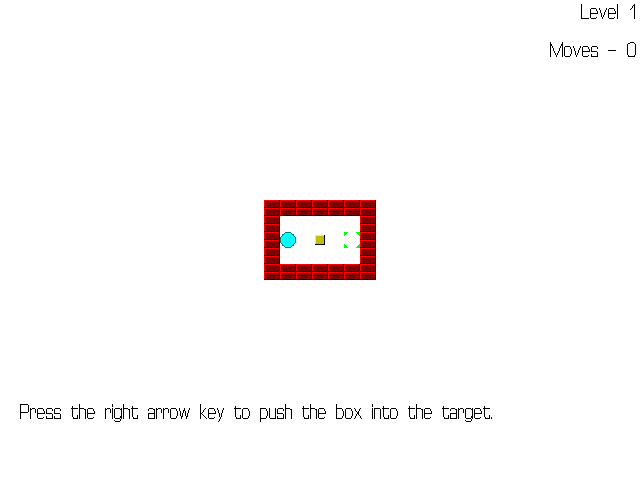
}

#endif //MAINMENU\_CPP

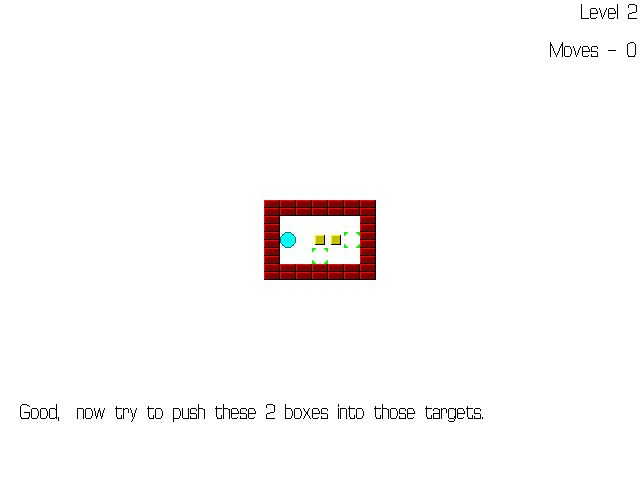
Screenshots



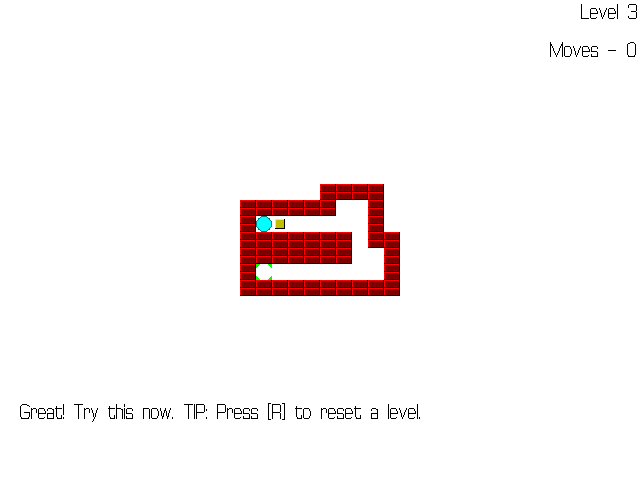
Screenshot 1: Main Menu



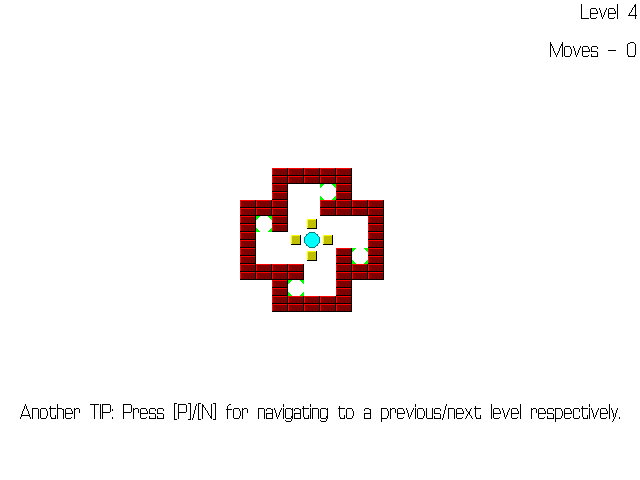
Screenshot 2: Tutorial Level 1



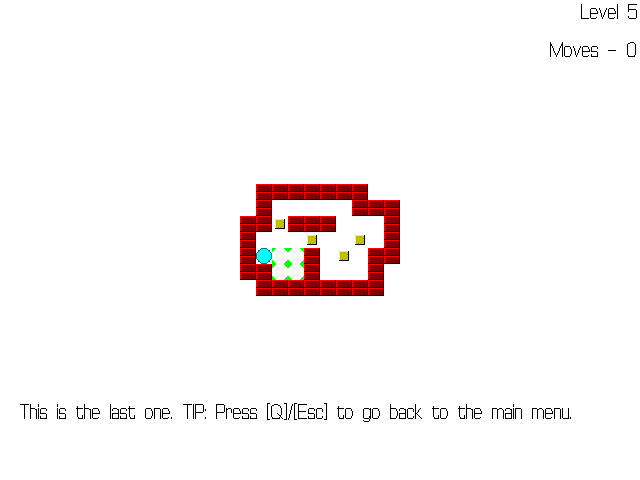
Screenshot 3: Tutorial Level 2



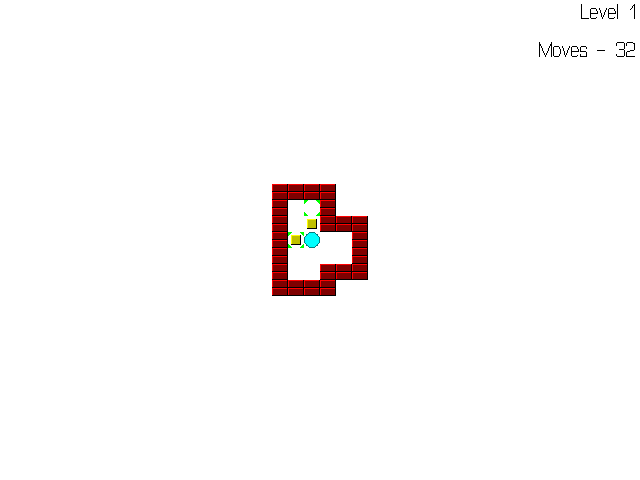
Screenshot 4: Tutorial Level 3



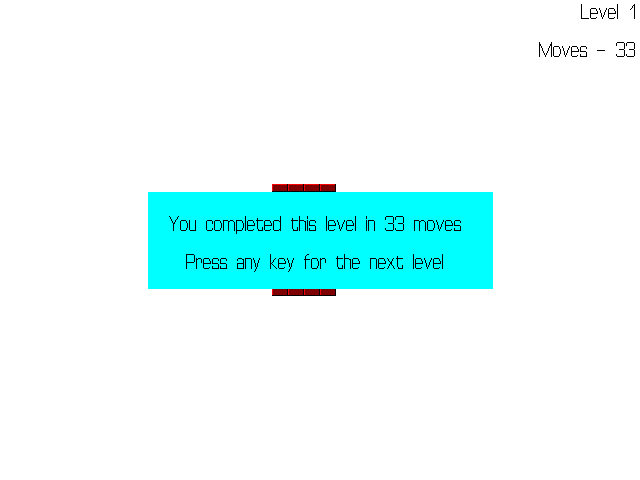
Screenshot 5: Tutorial Level 4



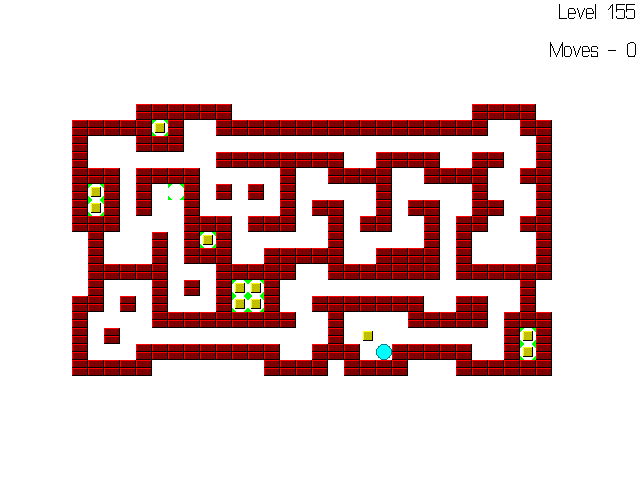
Screenshot 6: Tutorial Level 5



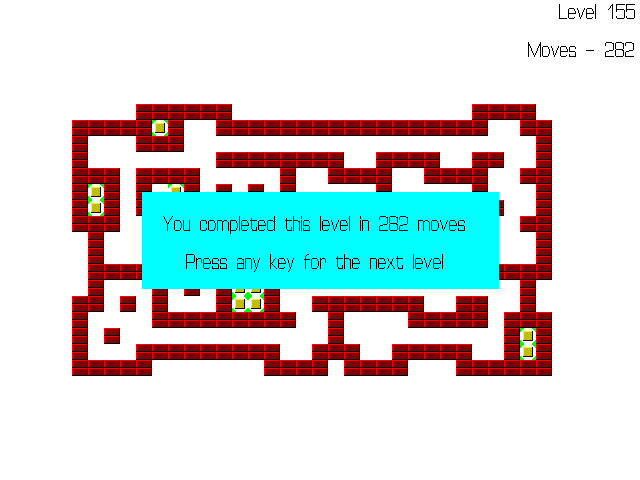
Screenshot 7: Start of Level 1



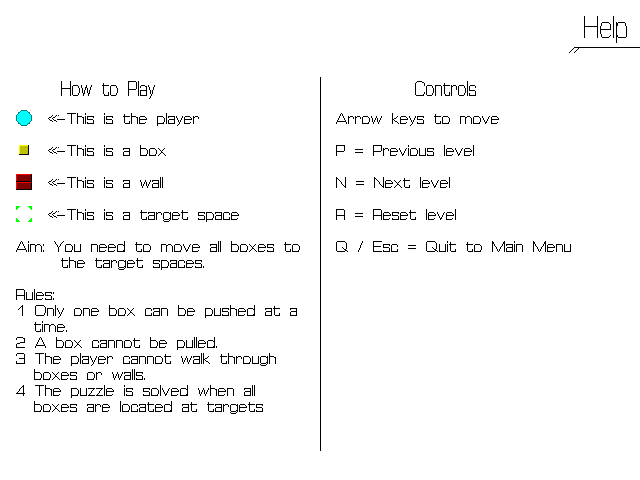
Screenshot 8: End of Level 1



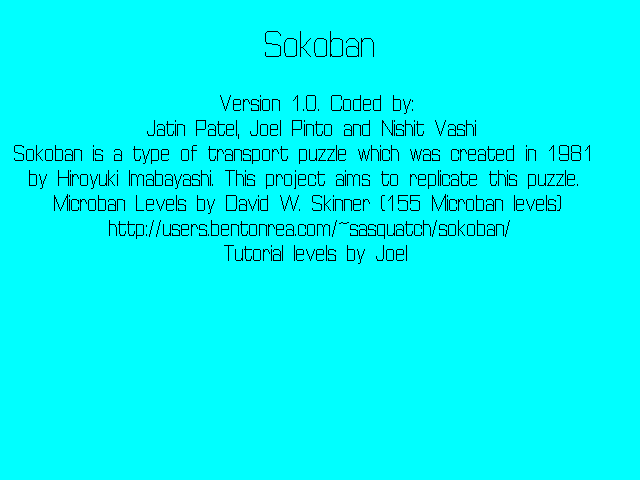
Screenshot 9: Start of Level 155



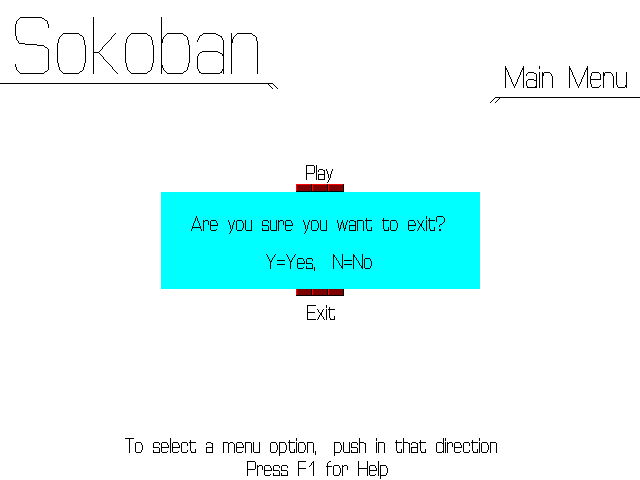
Screenshot 10: End of Level 155



Screenshot 11: F1 Help



Screenshot 12: About page



Screenshot 13: Exit confirmation dialog

References

1. David W. Skinner (155 Microban levels)

<http://users.bentonrea.com/~sasquatch/sokoban/>

1. Sokoban - Wikipedia, the free encyclopedia

<http://en.wikipedia.org/wiki/Sokoban>

1. Sokoban Clone

<http://fluxrider.com/projects/sokoban/index.html>

1. Koder’s Code Search: Sokoban.cpp

<http://www.koders.com/cpp/fid079D3C8B44170453FA2A30069DD6477C78D59DCD.aspx>