Push Push(aka sokoban) game

**김병주, 김찬현, 김태훈, 박준형, 최종윤**

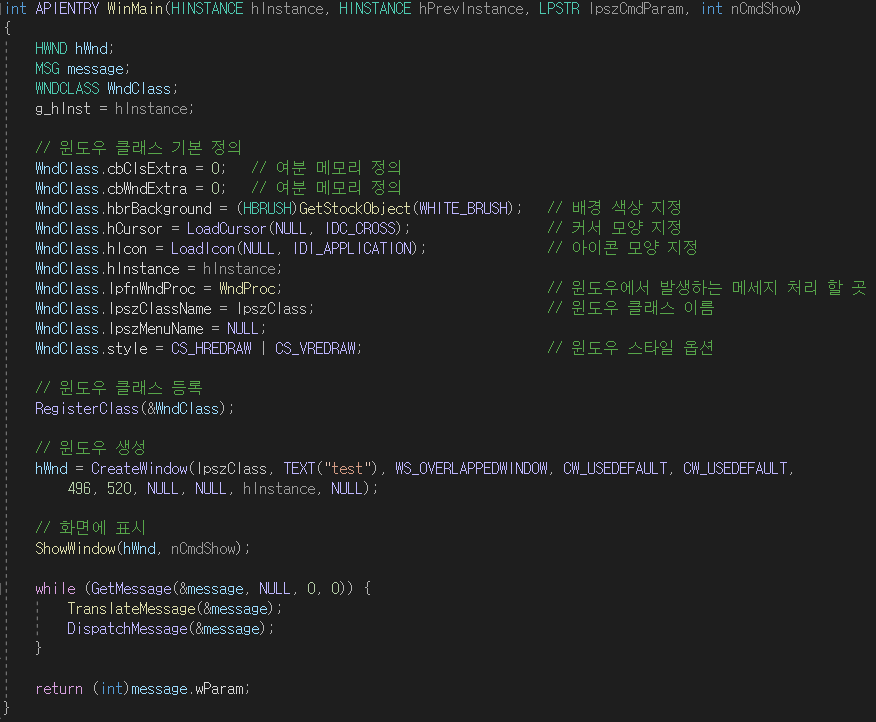
**(a) Brief project description**

- Using Windows API, we developed sokoban game. The game rule is simple. Player pushes every balls on the map into the home by statically moving balls.

**(b) system requirement for compilation and execution**

- We used Microsoft Visual Studio 2019. You may need a version that includes **Windows API.**

**(c) description on functionality**



To use Windows API, we need to know how to use WinMain() and WinProc() functions. The main principle is following. First, in WinMain() function, create a window class and set several values. Then register the window class to create a window by using RegisterClass(). Now you can show your window on the screen by using ShowWindow(). By calling GetMessage(), TranslateMessage() and DispatchMessage() in the loop, you can get the message which is the user’s input such as keyboard and mouse. Then use the message to change the status of your window class. Proper reaction of each changes will be described in WinProc() function.

**CASE WM\_CREATE** : create bitmap variable to load bitmap image for each block

**CASE WM\_KEYDOWN** : direction button (call move function for each direction)

space bar (show up the main page from the start page)

‘R’ (restore the starting state of the stage)

‘P’ (skip this stage and move on to the next stage)

‘ESC’ (finish the program)

**CASE WM\_PAINT** : draw each blocks on the screen

**CASE WM\_DESTROY** : destroy bitmap variables after used.

**Cell class** : It presents each blocks on the map such as the wall, home, bomb, player, and empty space. It contains its x coordinates and y coordinates on the map and attribute variable which tells you what kind of block it is. This class **inherit** to **wall, home, bomb and player classes**. Each child class has their own set() function that set their position on the map and draw function to draw the Bitmap Image on the screen. Player class especially has move() function that moves the player to up, down, left, right direction.

**Map class**: It gets map data from MapResource.h and create a new map.

This was our original development goal and we tried to keep this way but we had to change our direction for the efficient development…

**- Below of these are Final Design of Classes**

**Map class**

MAP (int arr[][10][10], int mapindex) : a creator that makes a new map class

void setinitialize(int arr[][10][10], int mapindex) : set a map using the data from resource.h file with mapindex

void move\_up(), void move\_down(), void move\_left(), void move\_right()

->implement the movement of the player

int getstatus(int x, int y) : get the attributes(what kind of block it is) of each map cells.

int checkhole(int x, int y) : check if the bomb can get into the home

**Gamemanager class**

HBITMAP createbitmap(int bitmapindex)

-> create a bitmap variable for the image that will be drawn on the screen.

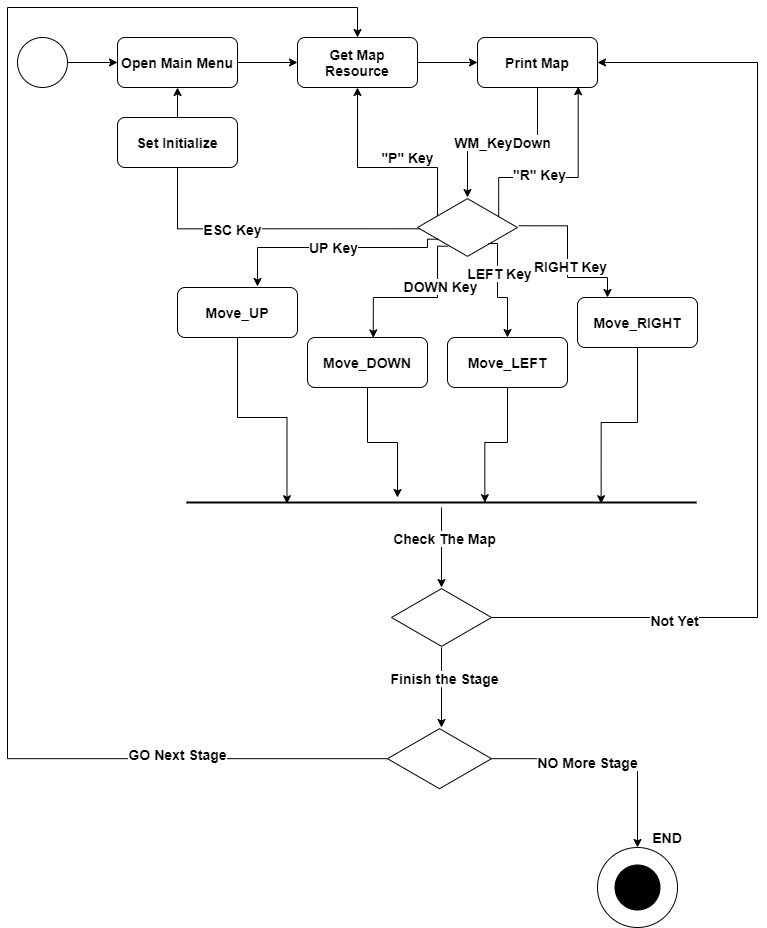
void draw(HDC hdc, HDC Memdc, HBITMAP draw, int x, int y, int cx, int cy)

-> draw a image on the screen

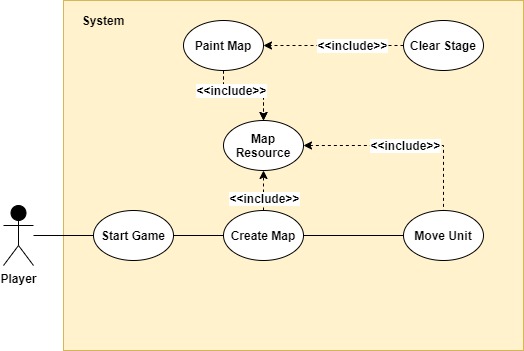
bool checkgameend(MAP map) : check if the stage is cleared

**(e) the result of UML modeling for system design**

**1. Activity diagram**

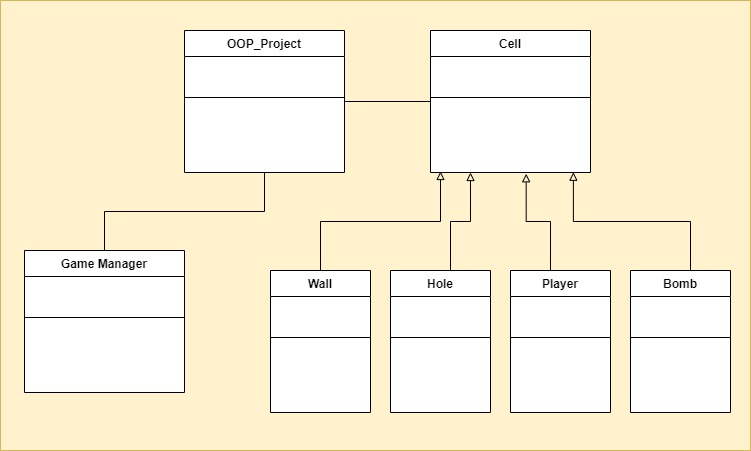


**2. Use-case diagram**

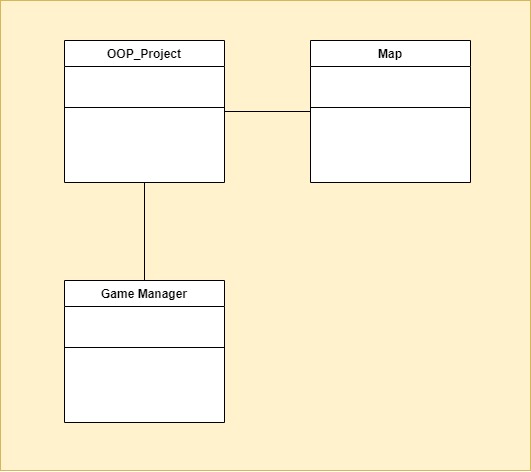


**3. Class diagram**

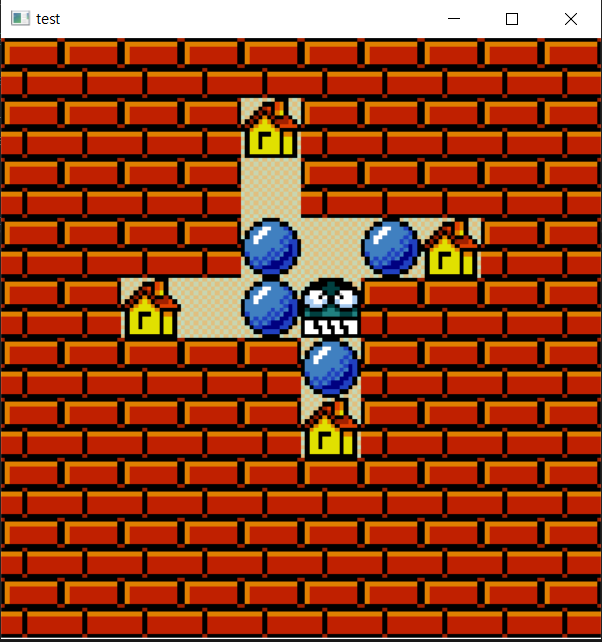
**3.1 First goal of our Class diagram**

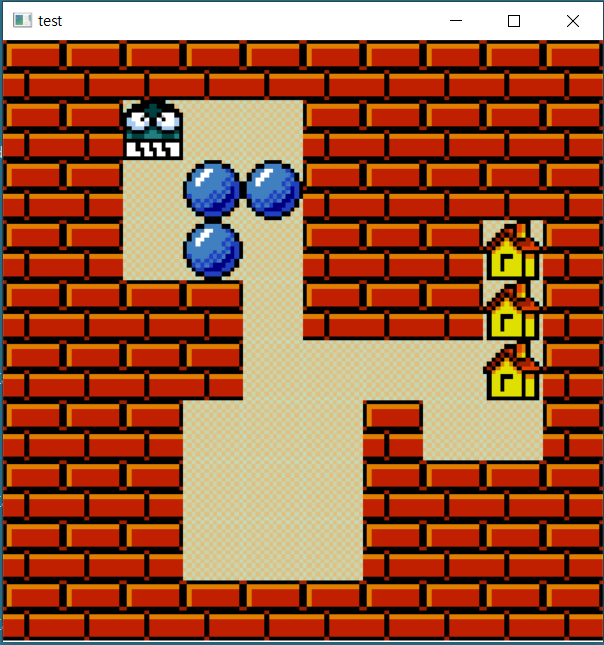


**3-2. Final design of our Class diagram**

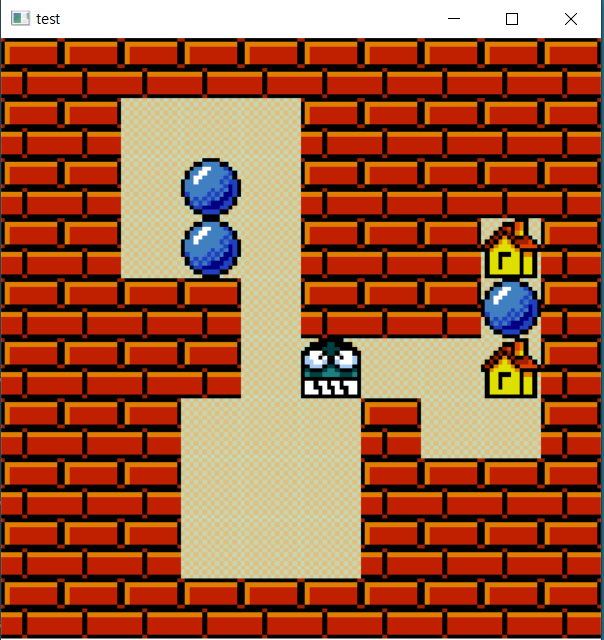


**(f) execution results**





Pictures above are the start page, stage1, stage2 and stage3.



<This is a screenshot after few movements on stage2.>

**(g) object oriented concepts to the development for your project**

**- Inheritance**

We created Cell class that as a superclass that represents each blocks on the map. Then wall, home, bomb and player class were inherited Cell class as a child class. Because they have similarity as a block on the map and they all have coordinates of the block on the map.

**- Polymorphism**

Cell class has draw function that draw Bitmap image on the screen but each child class needs different kind of image so we have to override draw function in each child class.

**3) Encapsulation**

By declaring each cell’s variable as private, reliability of those variables from each function could be increased.

**(h) Conclusion**

Through this team project, we could fully understand the features of object-oriented programming and its advantage. Before we learned object-oriented programming, we just wrote some codes, compiled the codes, and tested the codes. We just made complex codes as known as “the Spaghetti”.

First, we can reuse the code of the parent class. Through the inheritance, we could reuse the code and the function without rewriting the same code. It made us the working speed and the productivity.

Second, we can understand a feature of object-oriented programming, polymorphism. When we wrote our code, there were some cases to need overload, override, and polymorphism. In fact, we failed to succeed all of these features. But we could fully understand how to use it and why these features are essential. In this process, we could notice an importance of the initial total plan project. So, in other project, we have confidence that we produce the better projects and results.

However, unfortunately, we failed to use more various class object & skill in our project. But we learn by this trial and error, can find next goal of our OOP development Skill.

To sum up, we fully understood what we have learned about object-oriented programming in this class. It is expected that we will be able to apply it and develop the codes in the future.