Week 16

Remind

- 抄襲一律 0 分 （包含被抄襲者）
- 繳交期限: **2021/1/3(Sun.) 11:59 p.m.**
- 2人一組 (建議至少一人使用Windows作業系統)
- 一組繳交一份作業即可，請自行決定由哪位組員繳交
- 可以一人一組
- 請適當分工，並附上分工說明檔案(.pdf) 一同於zip檔附上
- 繳交的檔案格式、名稱請符合以下規定
    - 請繳交 zip檔至 Ceiba作業區，名稱為 <student_id>.zip
    - 解壓縮後須符合格式、名稱
    - e.g. b12345678.zip
- 必須完成 Demo 才可以提早離開
- 若沒有完成 Demo 就中途早退，視同缺席
- 若當天沒有完成Demo，請以螢幕錄影解釋程式碼(請各自講解各自分工的部分)，並於繳交期限前將影片**連同程式碼**寄至助教信箱ee10042020@gmail.com，信的「主旨」格式: Tetris_<學號1>_<學號2>

Problem - Tetris (13%)

Introduction

Tetris is a very popular video game. In this assignment, you are going to implement Tetris.

Description

You will get these following files:

Brick_abs.cpp

Brick.cpp

Tetris.cpp

main.cpp

These files are available in a "Week16_HW.zip" on Ceiba. You need to implement functions by completing Brick.cpp and Tetris.cpp. Do not change anything in Brick_abs.cpp and main.cpp. However, you shall read them carefully and understand what they mean. Submit these files to Ceiba after you finish the assignment while keeping the file names.

Class `Brick_abs` (Brick_abs.cpp)

This is an abstract class for the different types of bricks in a typical Tetris game. Note that `int get_brick()` and `void random_set_brick()` are declared as pure virtual functions. (You must NOT modify anything written here)

```
class Brick_abs{
  public:
    char kind;
    char direction;
    int x;
    int y;
    Brick_abs(){};
    virtual ~Brick_abs(){};
    virtual int get_shape() const = 0;
    virtual void random_set_brick() = 0;
};
```

Class `Brick` (Brick.cpp) (1%)

This class inherits `Brick_abs`. Use public inheritance and implement this class by yourself in the `/* TO DO */` part.

More about bricks:

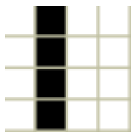
The shape of a brick is represented by a bit-stream of 16 bits, while the 16 bits can be directly mapped into a 4*4 grid.

For example, the I-brick can be represented by `0100 0100 0100 0100`.

Which correspond to the following shape in a 4*4 grid

```
0100
0100
0100
0100
```

or equivalently



As a result, the shape of an I-brick pointing vertically (the one shown above) can be stored by an integer which is `0100 0100 0100 0100` in binary, which is also 0×4444 in hexadecimal representation.

```
#include "Brick_abs.cpp"

const char backgroud_x_length = 7;
const char backgroud_y_length = 16;
const char start_x_position    = backgroud_x_length / 2 - 1;
const char start_y_position    = 0;

const char brick_kind_count = 7;

const int shapeList[brick_kind_count][4] = {
{0xc400,0x4c00, 0x8c00,0xc800},//Shape B
{0x4840,0x4a00, 0x8480,0xa400},//Shape T
{0xcc00,0xcc00, 0xcc00,0xcc00},//Shape O
{0xf00, 0x4444, 0xf00 ,0x4444},//Shape I
{0xe800,0xc440, 0x2e00,0x88c0},//Shape L
{0x8e00,0xc880, 0xe200,0x44c0},//Shape J
{0xc600,0x4c80, 0xc600,0x4c80} //Shape Z
};

/* TO DO */
```

The following bricks will be generated for you (the dot `.` denotes the white-space)

Shape B

`{0xc400, 0x4c00, 0x8c00, 0xc800}`

```
# # . .
. # . .
. . . .
. . . .
```

```
. # . .
# # . .
. . . .
. . . .
```

```
# . . .
# # . .
. . . .
. . . .
```

```
# # . .
# . . .
. . . .
. . . .
```

Shape T

`{0x4840, 0x4a00, 0x8480, 0xa400}`

```
. # . .
# . . .
. # . .
. . . .
```

```
. # . .
# . # .
. . . .
. . . .
```

```
# . . .
. # . .
# . . .
. . . .
```

```
# . # .
. # . .
. . . .
. . . .
```

Shape O

`{0xcc00, 0xcc00, 0xcc00, 0xcc00}`

```
# # . .
# # . .
. . . .
. . . .
```

```
# # . .
# # . .
. . . .
. . . .
```

```
# # . .
# # . .
. . . .
. . . .
```

```
# # . .
# # . .
. . . .
. . . .
```

Shape I

`{0xf00, 0x4444, 0xf00 , 0x4444}`

```
. # . .
. # . .
. # . .
. # . .
```

```
. . . .
# # # #
. . . .
. . . .
```

```
. # . .
. # . .
. # . .
. # . .
```

```
. . . .
# # # #
. . . .
. . . .
```

Shape L

`{0xe800, 0xc440, 0x2e00, 0x88c0}`

```
# # # .
# . . .
. . . .
. . . .
```

```
# # . .
. # . .
. # . .
. . . .
```

```
. . # .
# # # .
. . . .
. . . .
```

```
# . . .
# . . .
# # . .
. . . .
```

Shape J

`{0x8e00, 0xc880, 0xe200, 0x44c0}`

```
# . . .
# # # .
. . . .
. . . .
```

```
# # . .
# . . .
# . . .
. . . .
```

```
# # # .
. . # .
. . . .
. . . .
```

```
. # . .
. # . .
# # . .
. . . .
```

Shape Z

`{0xc600, 0x4c80, 0xc600, 0x4c80}`

```
# # . .
. # # .
. . . .
. . . .
```

```
. # . .
# # . .
# . . .
. . . .
```

```
# # . .
. # # .
. . . .
. . . .
```

```
. # . .
# # . .
# . . .
. . . .
```

main.cpp

We have already done this part for you.

```
#include "Tetris.cpp"

int main()
{
  hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
  srand(std::time(NULL));
  start();
  return 0;
}
```

Tetris.cpp

We will show you how to play the game in this file. First, let's see how the result should look like:

```
00              00
01              01
02              02
03              03
04              04
05              05
06              06
07              07
08              08
09              09
10              10
11              11
12              12
13              13
14              14
15              15
0 1 2 3 4 5 6 7 8
```

```
00              00            NEXT:
01              01
02              02
03              03
04              04
05              05            Score:0
06              06            O: Rotate
07              07            J: Down
08              08            I: Left P: Right
09              09
10              10
11              11            SPACE: Hard Drop
12              12
13              13            R: Restart
14              14            Q: Exit
15              15
0 1 2 3 4 5 6 7 8
```

This is how we play the game:

```
00      #       00            NEXT:
01      #       01
02      #       02
03      #       03
04              04
05              05            Score:0
06              06            O: Rotate
07              07            J: Down
08              08            I: Left P: Right
09              09
10              10
11              11            SPACE: Hard Drop
12              12
13              13            R: Restart
14              14            Q: Exit
15              15
0 1 2 3 4 5 6 7 8
```

```
00    #         00            NEXT:
01    #         01
02    #         02
03    #         03
04              04
05              05            Score:0
06              06            O: Rotate
07              07            J: Down
08              08            I: Left P
09              09
10              10
11              11            SPACE: Ha
12              12
13              13            R: Restar
14              14            Q: Exit
15              15
0 1 2 3 4 5 6 7 8
```

```
00      #       00            NEXT:
01      #       01
02      #       02
03      #       03
04              04
05              05            Score:0
06              06            O: Rotate
07              07            J: Down
08              08            I: Left P: Right
09              09
10              10
11              11            SPACE: Hard Drop
12              12
13              13            R: Restart
14              14            Q: Exit
15              15
0 1 2 3 4 5 6 7 8
```

```
00        #     00            NEXT:
01        #     01
02        #     02
03        #     03
04              04
05              05            Score:0
06              06            O: Rotate
07              07            J: Down
08              08            I: Left P
09              09
10              10
11              11            SPACE: Ha
12              12
13              13            R: Restar
14              14            Q: Exit
15              15
0 1 2 3 4 5 6 7 8
```

```
00      #       00            NEXT:
01      #       01
02      #       02
03      #       03
04              04
05              05            Score:0
06              06            O: Rotate
07              07            J: Down
08              08            I: Left P: Right
09              09
10              10
11              11            SPACE: Hard Drop
12              12
13              13            R: Restart
14              14            Q: Exit
15              15
0 1 2 3 4 5 6 7 8
```

```
00              00            NEXT:
01      #       01
02      #       02
03      #       03
04      #       04
05              05            Score:0
06              06            O: Rotate
07              07            J: Down
08              08            I: Left P
09              09
10              10
11              11            SPACE: Ha
12              12
13              13            R: Restar
14              14            Q: Exit
15              15
0 1 2 3 4 5 6 7 8
```

Now, you should implement the following functions by filling in the `/* TO DO */` parts:

`void clear_line()` (2%)

Clear line if the line is full of bricks. Think about the conditions when the line is full of bricks.

```
void clear_line()
{
    /* TO DO */
}
```

`bool is_feasible(const Brick &target)` (2%)

Check whether it is feasible for the brick to move or rotate. If it is feasible, return true. Otherwise, return false.

```
bool is_feasible(const Brick &target)
{
    /* TO DO */
  return true;
}
```

`bool rotate(Brick &A)` (1%)

Rotate the brick. This function takes a brick as an argument, and return true if the rotation is executed successfully, false otherwise. There are four directions in total, which is shown in the `shapeList`. For example, `shapeList[0]` encodes the four direction of the 0th brick `{0xc400, 0x4c00, 0x8c00, 0xc800}`. First, randomly choose a direction to rotate in. Following this, use the `is_feasible(A)` function to check if the rotation is feasible or not. If not, don't rotate, i.e., undo the rotation if you have rotated the brick, and return false directly, indicating the rotation is not executed. Otherwise, set the direction fo the brick according to the specified direction. Make sure you use the `print_brick` function to wipe out the old brick and print out the newly rotated brick. Return true to indicate a successful rotation.

```
bool rotate(Brick &A)
{
    /* TO DO */
}
```

`bool move(Brick &A, char XOffset, char YOffset)` (2%)

Similar to `bool rotate(Brick &A)`, move the brick if feasible. This function takes three arguments, a brick, x offset and y offset. Let's examine the following codes line by line.

- `char oldX = A.x, oldY = A.y;` - Store the old x and y

- `print_brick(A, "  ",0,0);` - Wipe out the old brick
- `A.x += XOffset, A.y += YOffset;` - Set new x and y
- `if(!is_feasible(A)){...}` - Check feasibility
- `print_brick(A, "#",0,0);` - Print out the brick
- `return true;` - Return true for a successful move

```
bool move(Brick &A, char XOffset, char YOffset)
{
  char oldX = A.x, oldY = A.y;
  print_brick(A, "  ",0,0);
  A.x += XOffset, A.y += YOffset;
  if(!is_feasible(A))
  {
      /* TO DO */
  }
  print_brick(A, "#",0,0);
  return true;
}
```

Show your work in the `if(!is_feasible(A)){...}` section.

`void fixed_position(const Brick &target)` (2%)

Once the brick is no longer movable, fix the position of the brick. That is, make the brick become the background.

```
void fixed_position(const Brick &target)
{
    /* TO DO */
}
```

Result

```
Play it by yourself, and you will see the result.
```

File

Tetris.cpp

Brick_abs.cpp

Brick.cpp

main.cpp

Challenge Problem (3%)

Adding more functions to your Tetris game, e.g.

- Add bonus scores when clear double, triple, tetris lines once.
- Hold the current brick
- Clockwise rotate, counter-clockwise rotation, and rotation 180°
- Add garbage (attacking lines appear at the bottom)… etc

You are allowed to modify anything of your code and submit the files with following naming constraint.

File

Tetris_adv.cpp

Brick_adv.cpp

Brick_abs.cpp

main.cpp