Advanced Programming

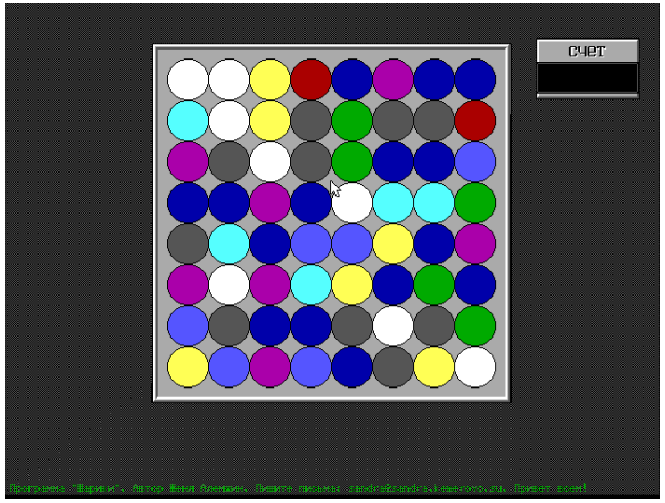
Programming Assignment #2

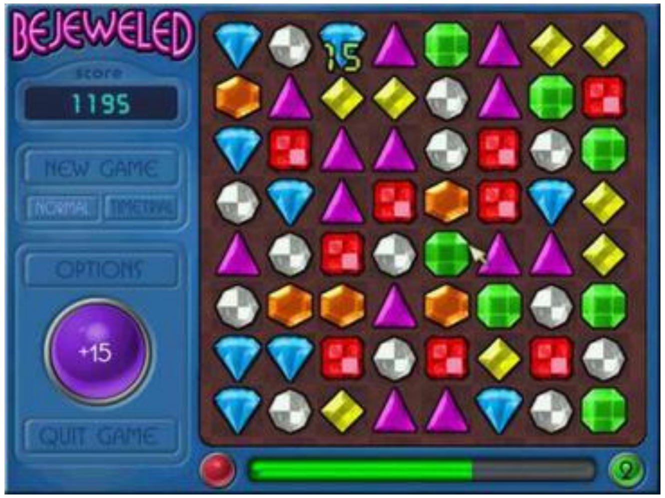
Kang Hoon Lee

Kwangwoon University

Bejeweled

- **A tile-matching puzzle video game by PopCap Games (2001)**
  - Goal
    - Clear gems of the same color, potentially causing a chain reaction
  - Legacy
    - Inspired by the 1995 MS-DOS game "Shariki"
    - Spawned several clones, collectively known as *match three games*





https://en.wikipedia.org/wiki/Bejeweled

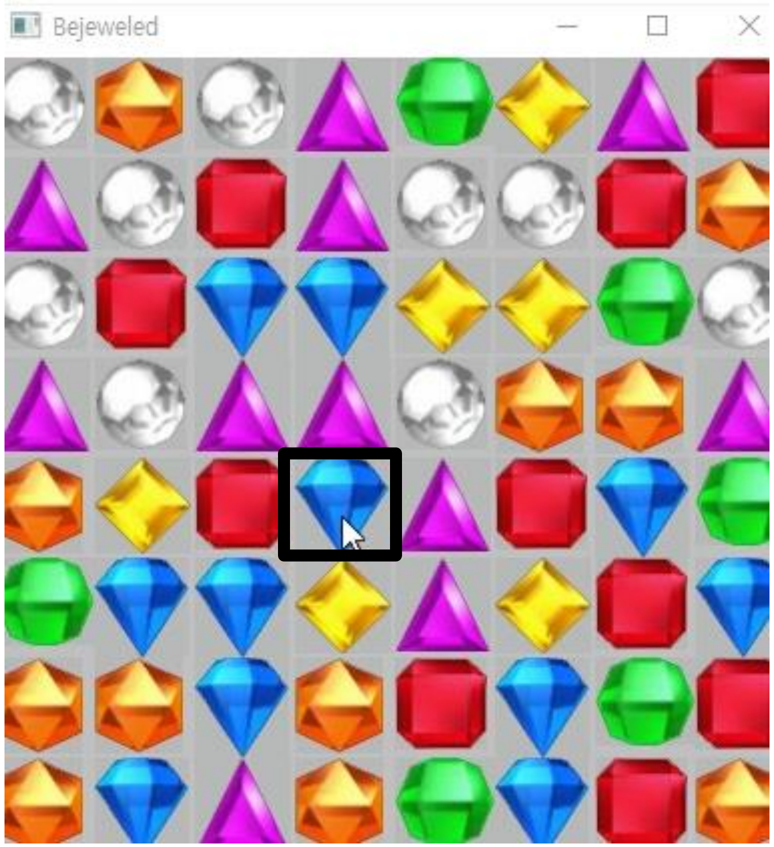
https://archive.org/details/msdos_Shariki_1994

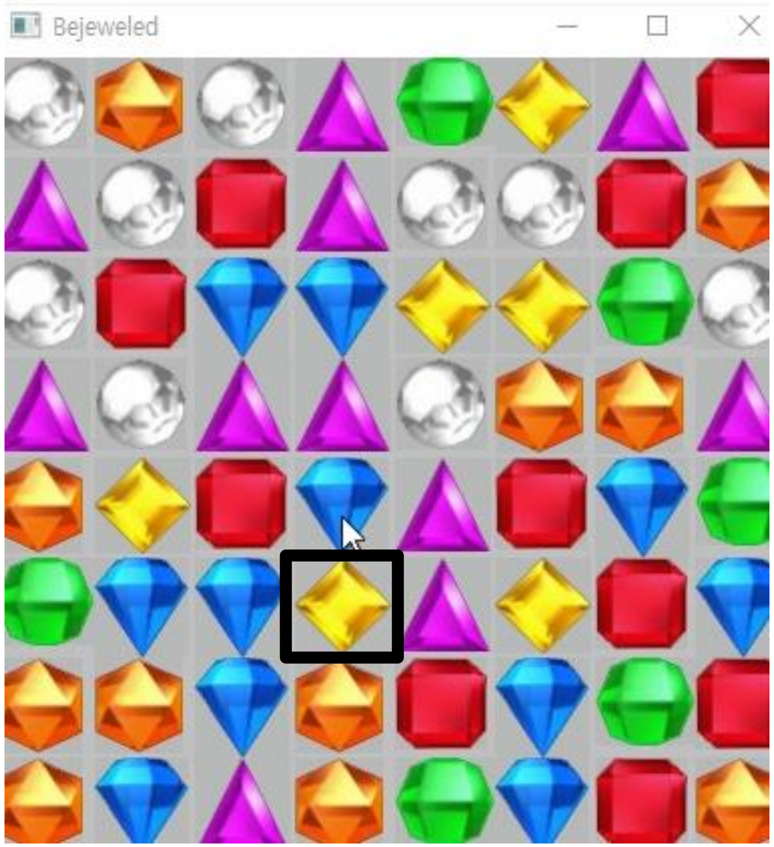
Bejeweled

- **How to play**
  - Swap one gem with an adjacent gem
    - To form a horizontal or vertical **chain**
    - **Chain**: three or more gems of the same color
  - Bonus points are given
    - When chains of more than three gems are formed
    - When two or more chains are formed in one swap
  - When chains are formed
    - Existing gems belonging to chains **disappear**
    - Existing gems above chains fall to **fill in gaps**
    - New gems are **spawned** and fallen to fill in the remaining gaps
  - Sometimes, **chain reactions** (cascades) are triggered
    - Where chains are formed by the falling gems
    - Chain reactions are usually awarded with bonus points

Bejeweled

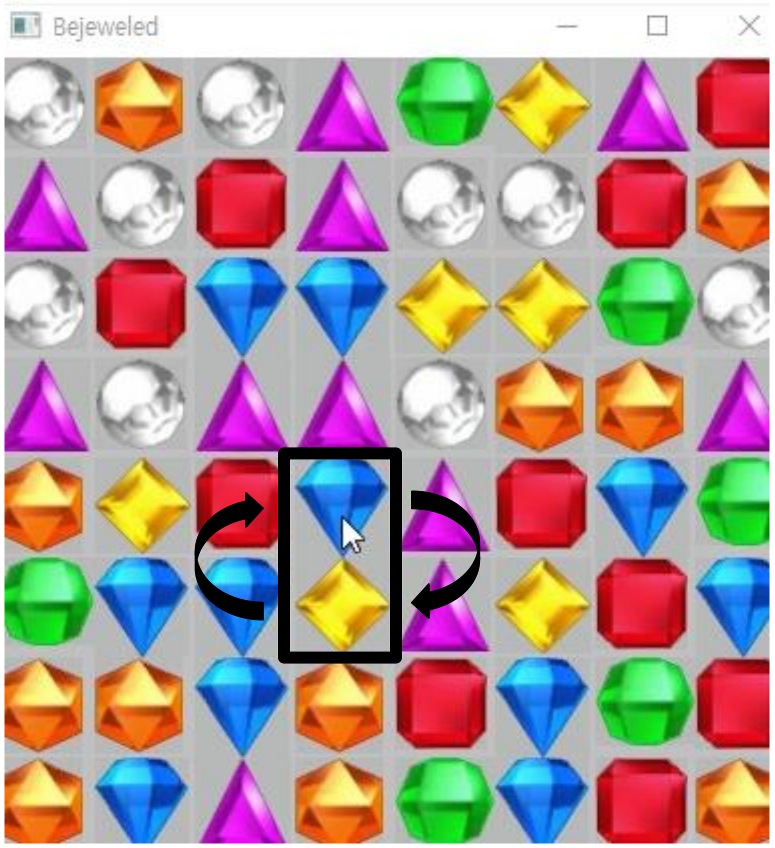
- How to play: Swap one gem with an adjacent gem

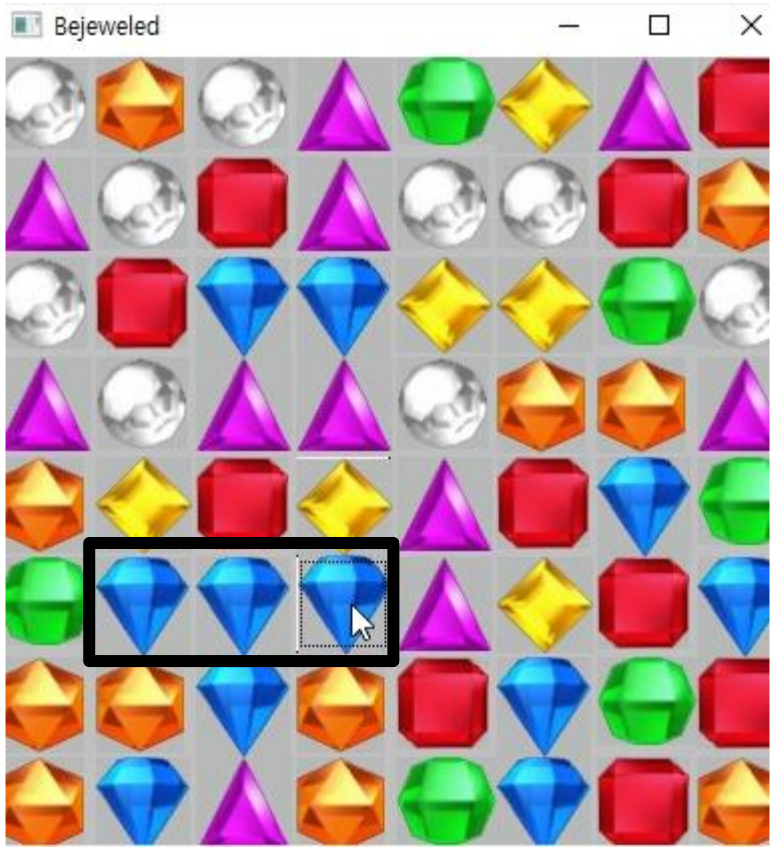




Bejeweled

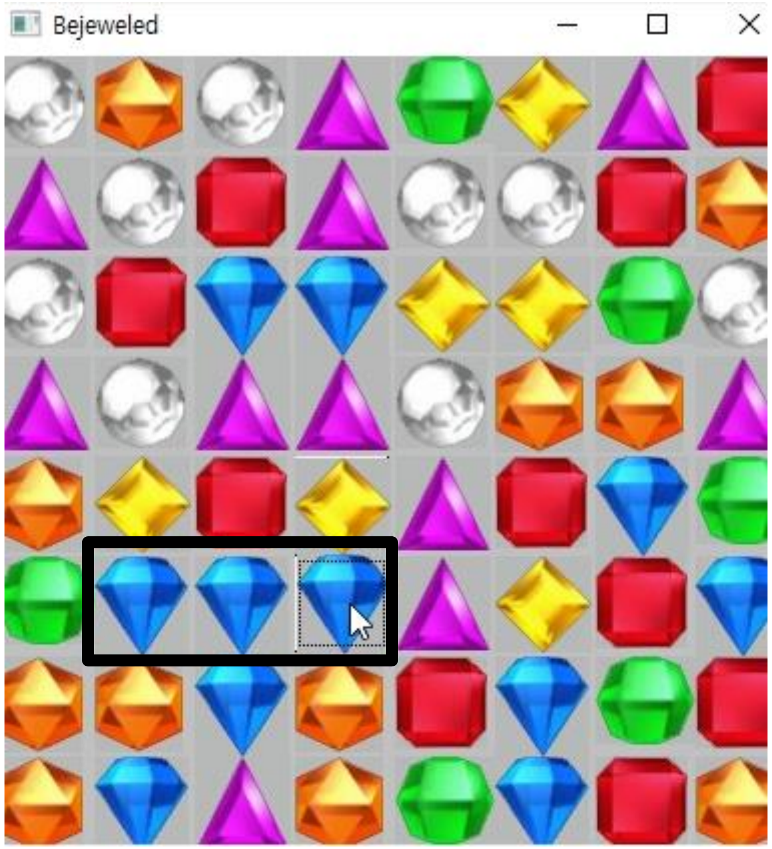
- How to play: To form a horizontal or a vertical chain

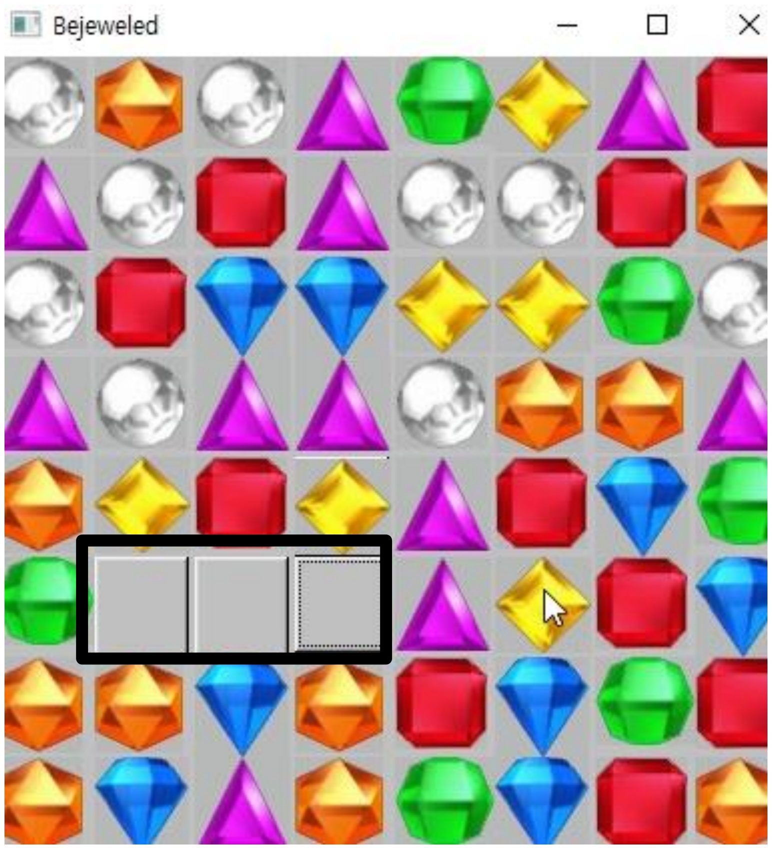




Bejeweled

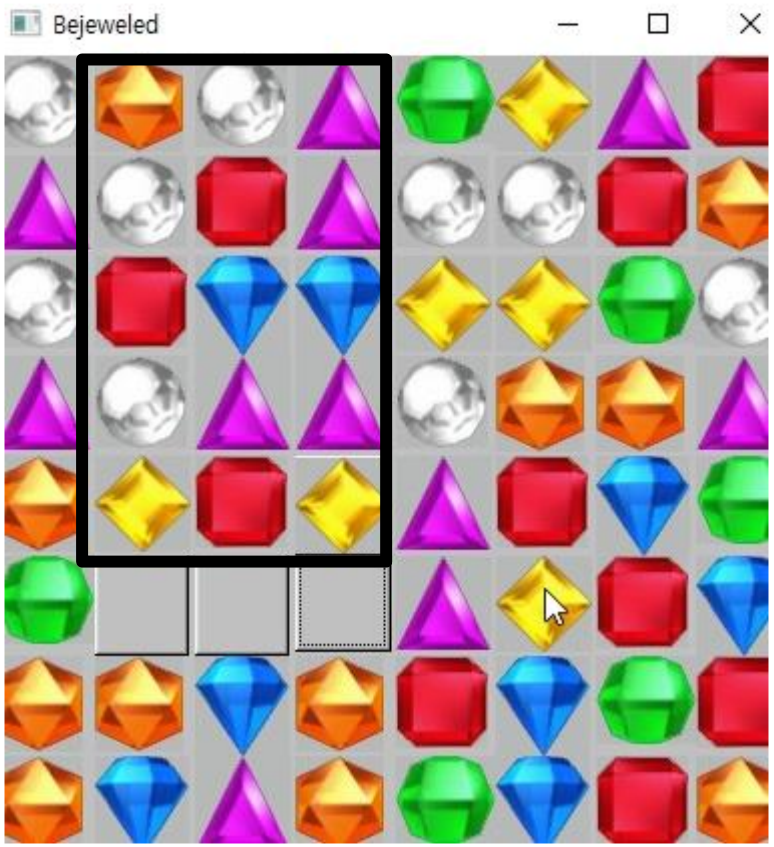
- How to play: Existing gems belonging to chains disappear

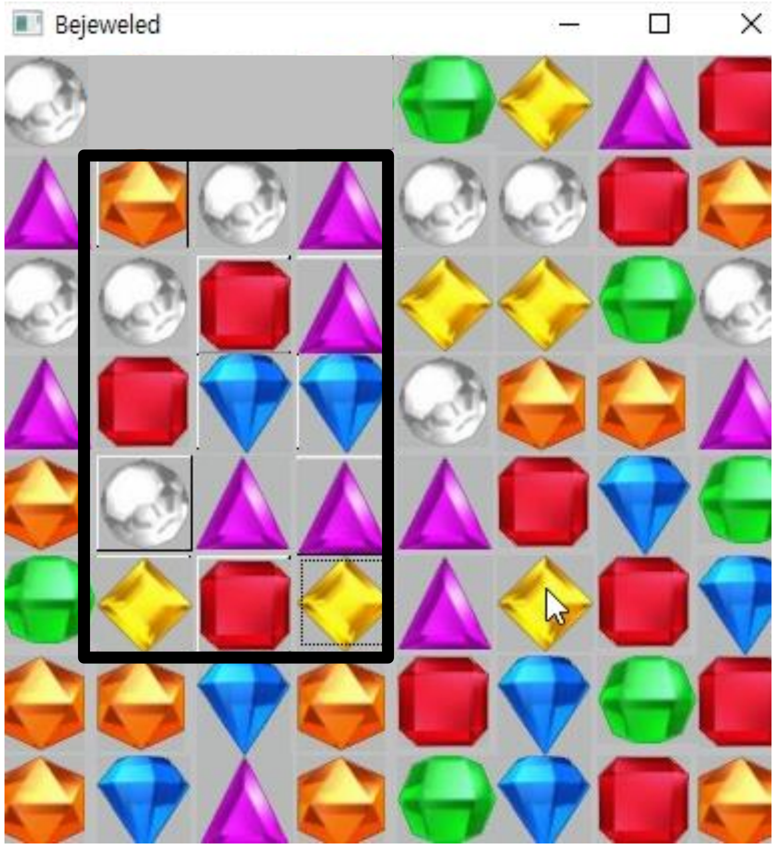




Bejeweled

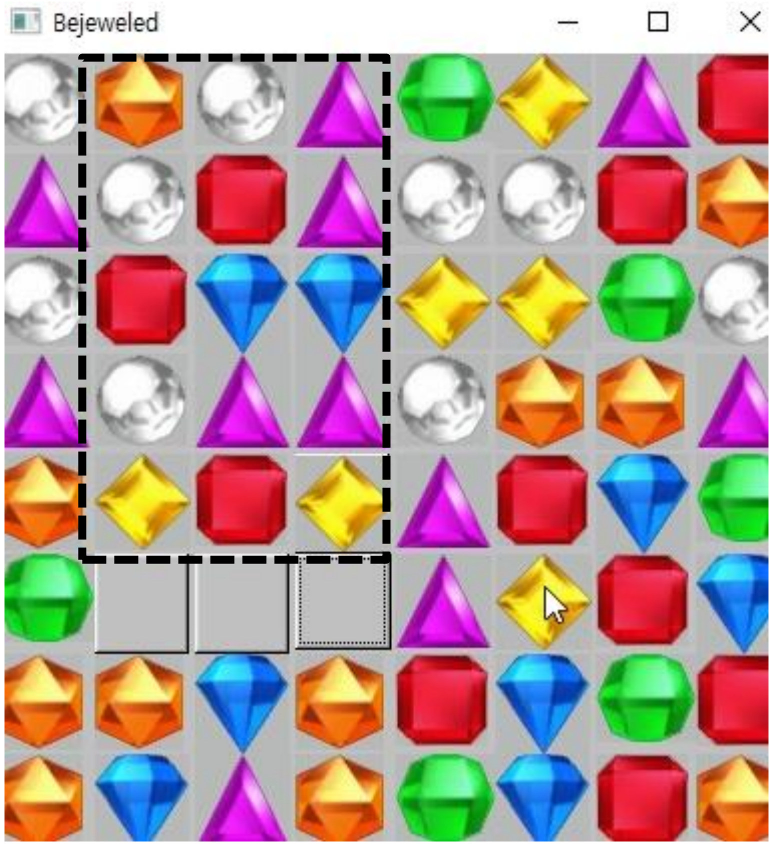
- How to play: Existing gems above chains fall to fill in gaps

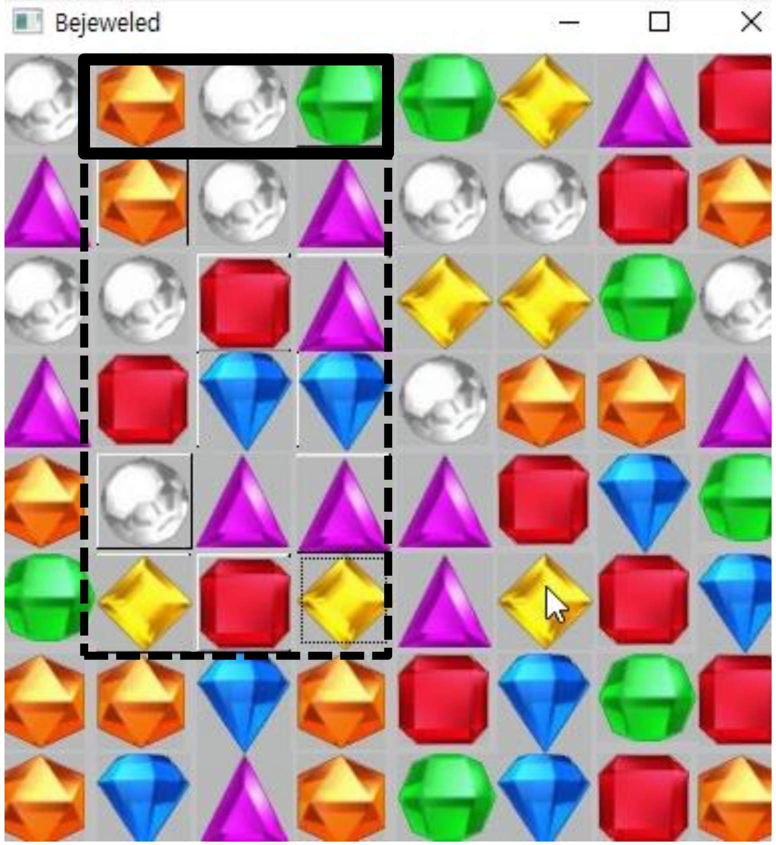




Bejeweled

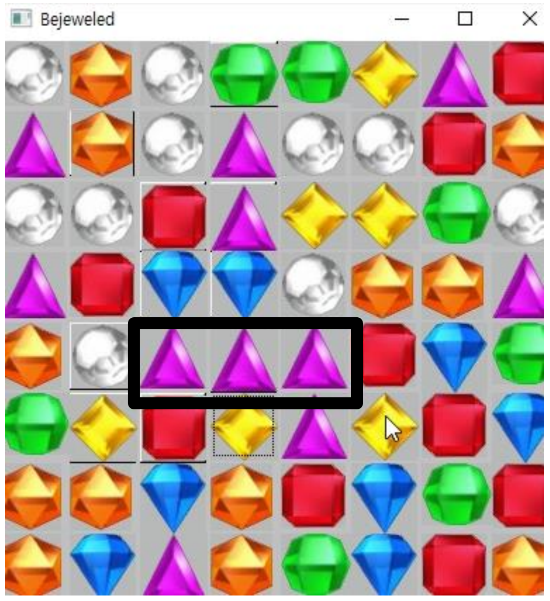
- How to play: New gems fall to fill in the remaining gaps

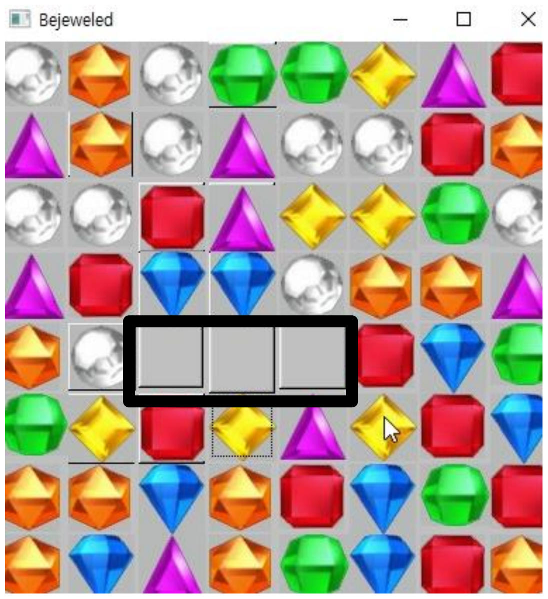


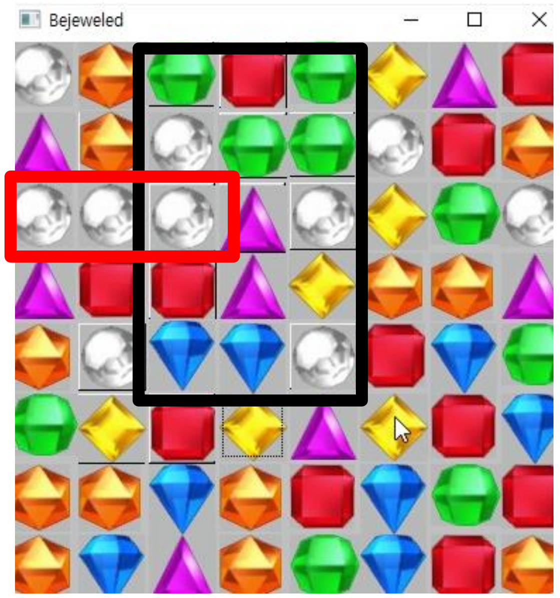


Bejeweled

- How to play
  - Sometimes, chain reactions (cascades) are triggered, where new chains are formed by the falling gems

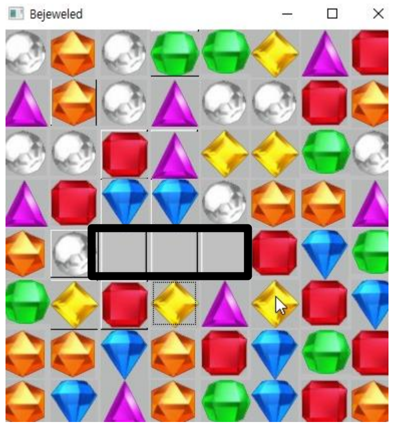


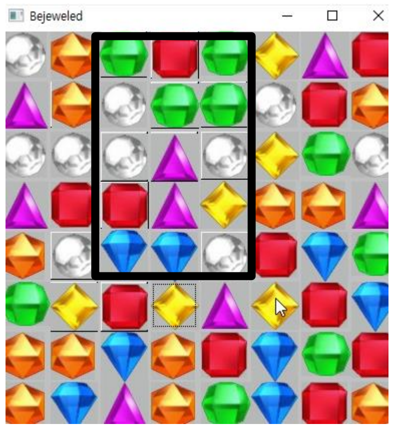


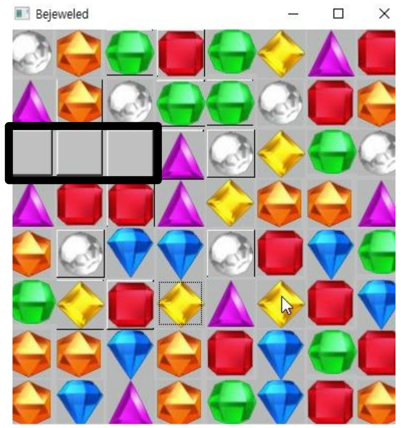


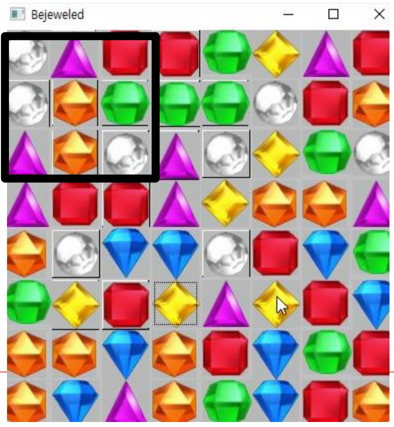
Bejeweled

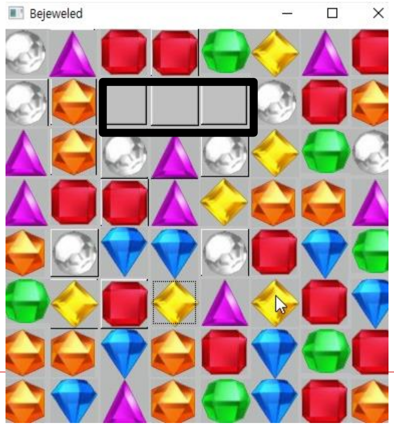
- **How to play**
  - And sometimes more and more chain reactions are triggered

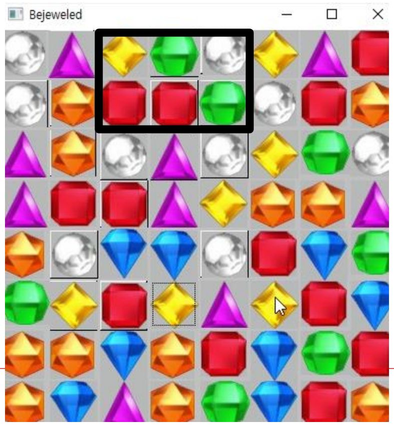












https://youtu.be/NR3_Tj7XwAE

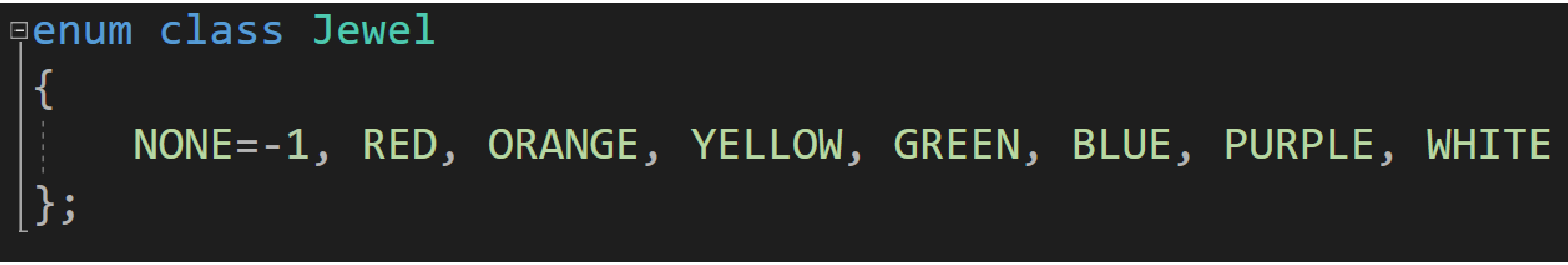
Implementing Bejeweled in C++

- **Step 1: Encapsulate the Bejeweled puzzle in a class**
  - Just a minimal template is provided (**`Puzzle.cpp/.h`**)
    - **`enum class Jewel { /* … */ };`**
    - **`class Puzzle { /* … */ };`**
  - Your class needs to provide a set of public functions as required
    - Explained in detail through the next few slides

- **Step 2: Provide a user interface for interacting with the puzzle**
  - Text-based UI (**`main_text.cpp`**)
    - You should implement this by yourself from the scratch
  - Graphical UI (**`main_gui.cpp, Puzzle_window.cpp/.h`**)
    - <u>Freely given to you</u> for testing your class and enjoying the game

Encapsulating Bejeweled in a Class

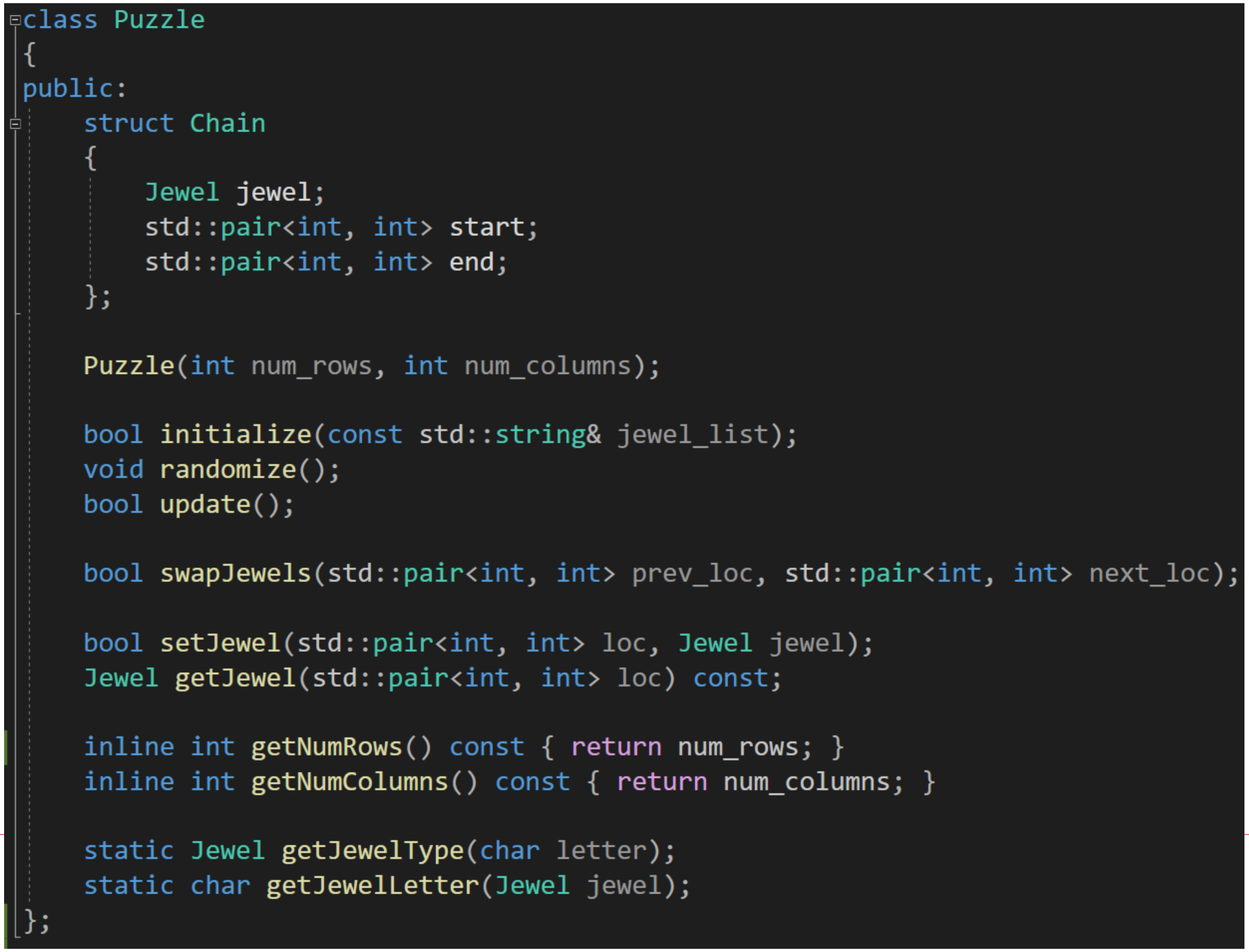
- `enum class Jewel`



```
enum class Jewel
{
    NONE=-1, RED, ORANGE, YELLOW, GREEN, BLUE, PURPLE, WHITE
};
```

Encapsulating Bejeweled in a Class

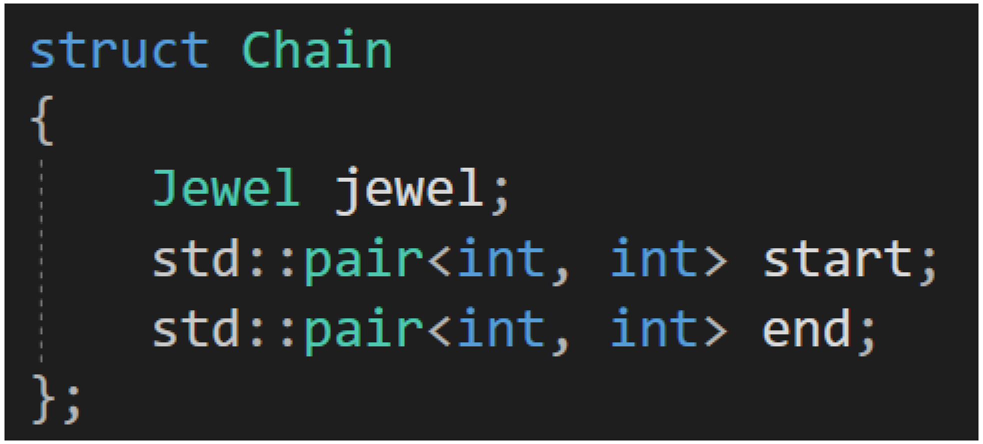
- class Puzzle



```
class Puzzle
{
public:
    struct Chain
    {
        Jewel jewel;
        std::pair<int, int> start;
        std::pair<int, int> end;
    };

    Puzzle(int num_rows, int num_columns);

    bool initialize(const std::string& jewel_list);
    void randomize();
    bool update();

    bool swapJewels(std::pair<int, int> prev_loc, std::pair<int, int> next_loc);

    bool setJewel(std::pair<int, int> loc, Jewel jewel);
    Jewel getJewel(std::pair<int, int> loc) const;

    inline int getNumRows() const { return num_rows; }
    inline int getNumColumns() const { return num_columns; }

    static Jewel getJewelType(char letter);
    static char getJewelLetter(Jewel jewel);
};
```

Encapsulating Bejeweled in a Class

- **`struct Chain`**

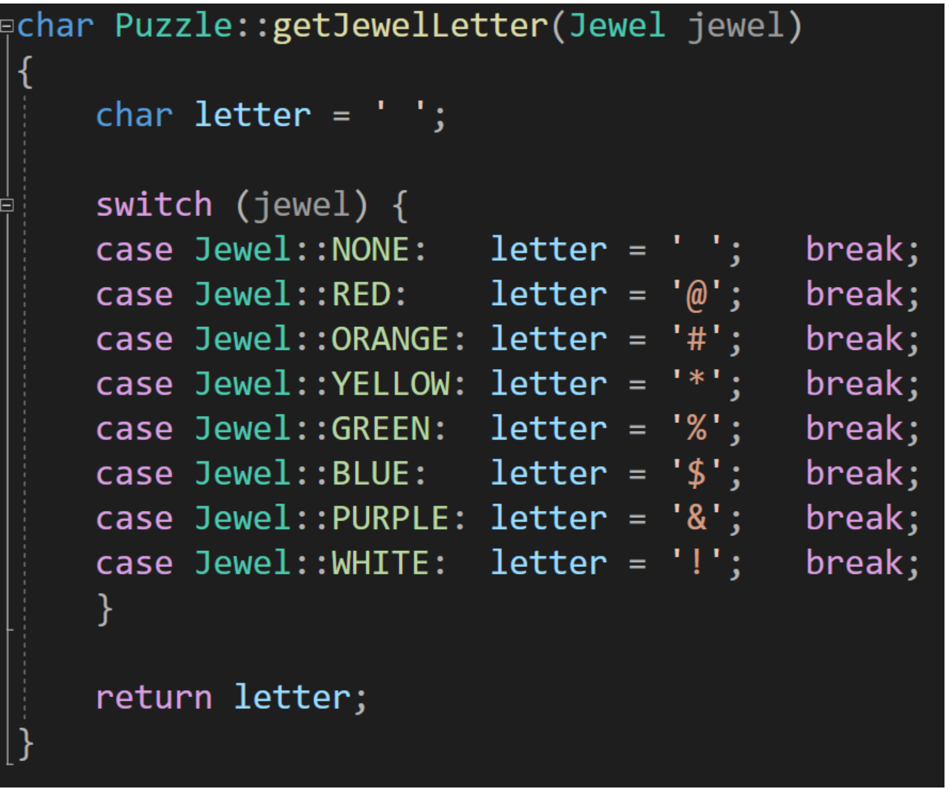


```
struct Chain
{
    Jewel jewel;
    std::pair<int, int> start;
    std::pair<int, int> end;
};
```

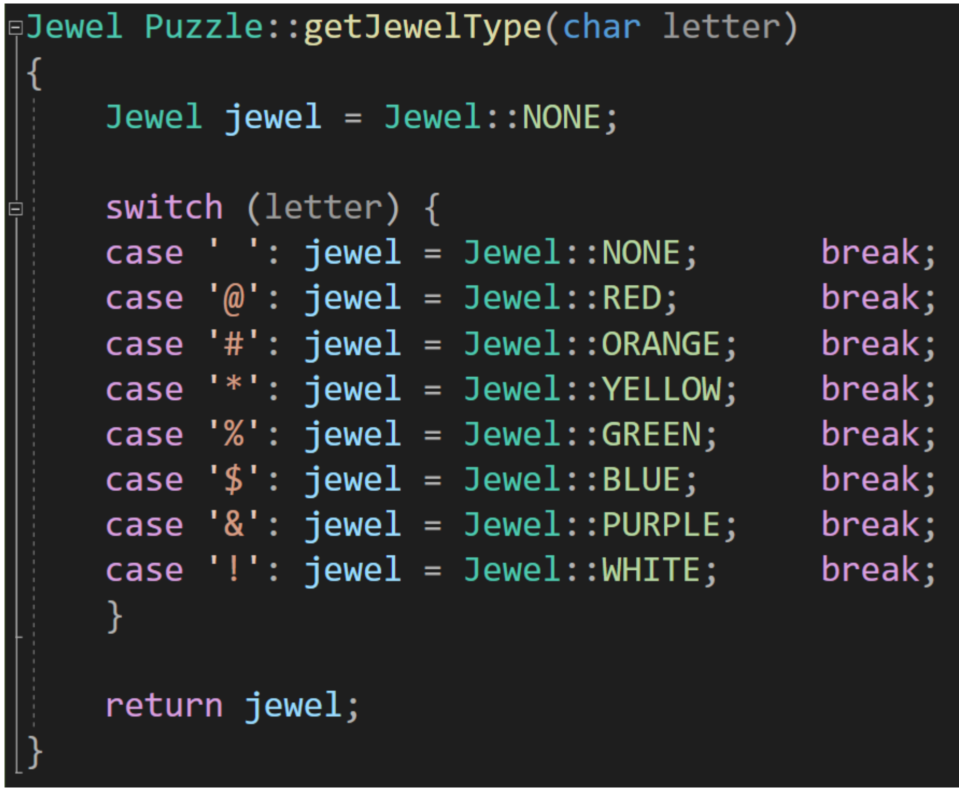
  - Represents a **`Chain`** of three or more **`Jewel`**s of the same color, connected either horizontally or vertically
  - **`start`** and **`end`** corresponds to the locations of both ends of **`Chain`**
    - You can make a **`std::pair`** object by calling **`std::make_pair()`**
    - You can access each element of a **`std::pair`** object **`obj`** via **`obj.first`** and **`obj.second`** members

Encapsulating Bejeweled in a Class

- Two *static functions* are pre-defined to convert between each Jewel type and its associated letter http://tcpschool.com/cpp/cpp_encapsulation_staticConst



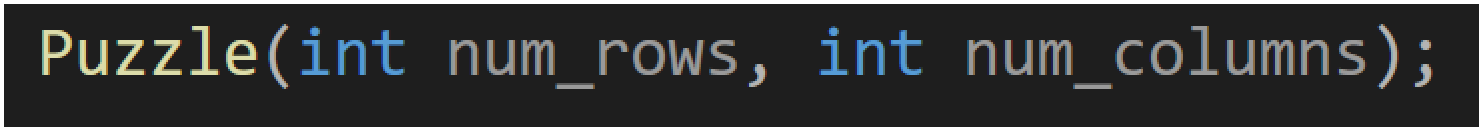
```
char Puzzle::getJewelLetter(Jewel jewel)
{
    char letter = ' ';

    switch (jewel) {
    case Jewel::NONE:   letter = ' ';    break;
    case Jewel::RED:    letter = '@';    break;
    case Jewel::ORANGE: letter = '#';    break;
    case Jewel::YELLOW: letter = '*';    break;
    case Jewel::GREEN:  letter = '%';    break;
    case Jewel::BLUE:   letter = '$';    break;
    case Jewel::PURPLE: letter = '&';    break;
    case Jewel::WHITE:  letter = '!';    break;
    }

    return letter;
}
```



```
Jewel Puzzle::getJewelType(char letter)
{
    Jewel jewel = Jewel::NONE;

    switch (letter) {
    case ' ': jewel = Jewel::NONE;       break;
    case '@': jewel = Jewel::RED;        break;
    case '#': jewel = Jewel::ORANGE;     break;
    case '*': jewel = Jewel::YELLOW;     break;
    case '%': jewel = Jewel::GREEN;      break;
    case '$': jewel = Jewel::BLUE;       break;
    case '&': jewel = Jewel::PURPLE;     break;
    case '!': jewel = Jewel::WHITE;      break;
    }

    return jewel;
}
```

Encapsulating Bejeweled in a Class

- **Constructor**



```
Puzzle(int num_rows, int num_columns);
```

  - Store the number of rows and the number of columns
  - Allocate memory space for storing **`Jewel`**s for $(num\_rows \times num\_columns)$ cells
    - It is recommended to use **`std::vector`** for storage
  - Initialize every **`Jewel`** to **`Jewel::NONE`**

Encapsulating Bejeweled in a Class

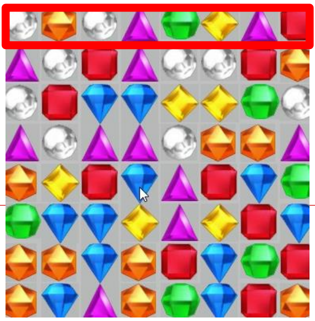
- **initialize()**

```
bool initialize(const std::string& jewel_list);
```

- Initialize every **Jewel** as described in the **jewel_list**
- If the length of **jewel_list** doesn't equal to the number of **Jewel**s, the function returns **false**
- You need to use the **getJewelType()** function for decrypting each letter in **jewel_list**

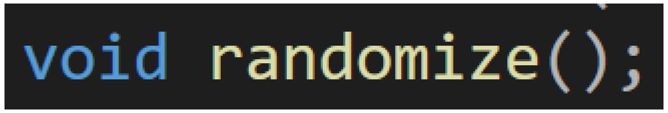
```
Puzzle puzzle(8, 8);
puzzle.initialize("!#!&%*&@&!@&!!@#!@$$**%!&!&&!##&#*@$&@$%%$$*&*@$##$#@$%@#$&#%$@#");
```

```
! # ! & % * & @
& ! @ & ! ! @ #
! @ $ $ * * % !
& ! & & ! # # &
# * @ $ & @ $ %
% $ $ * & * @ $
# # $ # @ $ % @
# $ & # % $ @ #
```



Encapsulating Bejeweled in a Class

- **`randomize()`**

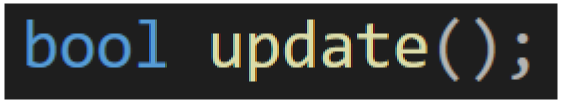


```
void randomize();
```

  - Initialize every **`Jewel`** randomly by using **`rand()`** function
    - You may devise a simple expression for yielding a random **`Jewel`** based on **`rand()`**, modulo operator (%), and explicit type conversion between **`Jewel`** and **`int`**

Encapsulating Bejeweled in a Class

- **`update()`**



```
bool update();
```

  - Update the **`Jewel`**s by applying one of the following rules in an alternate way for each call (A→B→A→B→…):
    A. Identify all **`Chain`**s (both horizontal and vertical) and remove **`Jewel`**s in every **`Chain`** (set to **`Jewel::NONE`**)
    B. Fill in the gaps (set to **`Jewel::NONE`**) by first falling down the existing **`Jewel`**s above the **`Chain`**s and then creating new **`Jewel`**s randomly in the remaining gaps
  - Returns **`true`** if either of the rules has been applied, and **`false`** otherwise (i.e. no more updates are allowed)

Encapsulating Bejeweled in a Class

- **`setJewel()`**

```
bool setJewel(std::pair<int, int> loc, Jewel jewel);
```

  - Set the **`Jewel`** at the location **`loc`** to the given **`jewel`**
  - Returns **`true`** if **`loc`** is a valid location, and **`false`** otherwise

❖ Checking the validity of **`loc`**
- **`loc.first >=0 && loc.first < num_rows`**
- **`loc.second >=0 && loc.second < num_columns`**

- **`getJewel()`**

```
Jewel getJewel(std::pair<int, int> loc) const;
```

  - Returns the **`Jewel`** at the location **`loc`** if **`loc`** is a valid location, and **`Jewel::NONE`** otherwise

Encapsulating Bejeweled in a Class

- **`swapJewels()`**

```
bool swapJewels(std::pair<int, int> prev_loc, std::pair<int, int> next_loc);
```

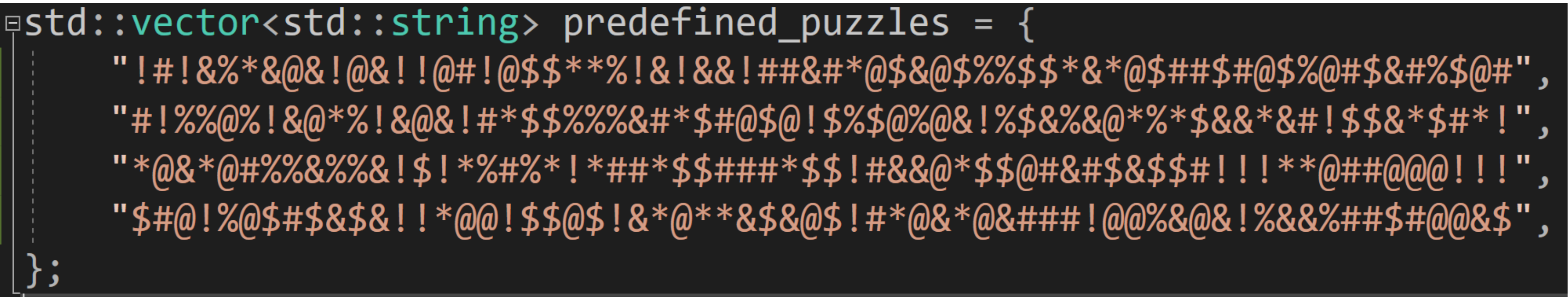
- If all of the following conditions are met,
  - both **`prev_loc`** and **`next_loc`** are valid locations
  - **`prev_loc`** and **`next_loc`** are adjacent (horizontally or vertically)
- Then,
  - Get the **`Jewel`**s at **`prev_loc`** and **`next_loc`**, called **`jA`** and **`jB`**
  - Set the **`Jewel`**s at **`prev_loc`** and **`next_loc`** to **`jB`** and **`jA`**
  - Return **`true`**
- Otherwise,
  - Return **`false`**

Providing UI for Playing Bejeweled

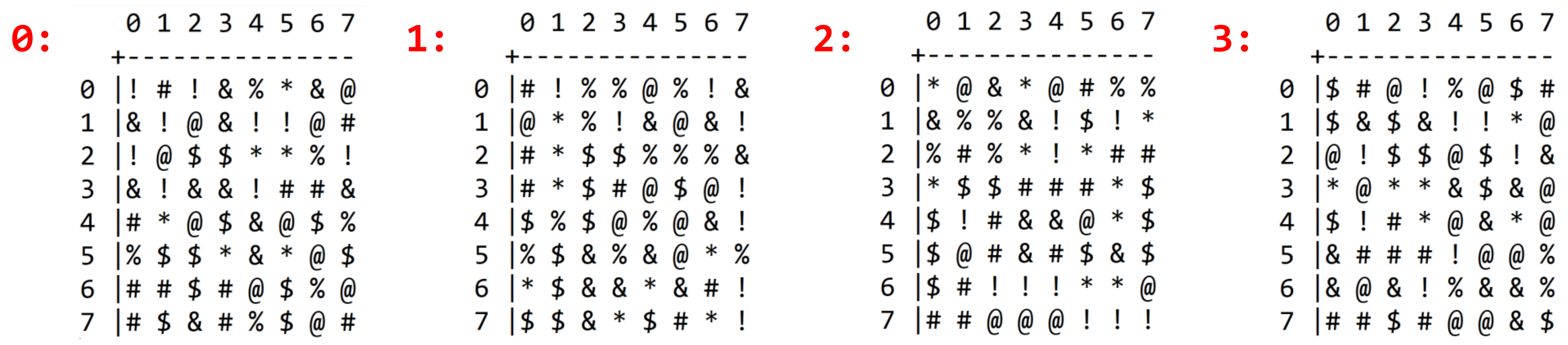
- **Text-based interface**
  - Create a 8-by-8 **`puzzle`** object
    - **`Puzzle puzzle(8, 8);`**
  - Allow the user to choose one of the followings:
    1. Start a new random puzzle
    2. Start a pre-defined puzzle
    3. Exit
  - If the user chooses 1, **`randomize()`** the **`Jewel`**s
  - If the user chooses 2, ask the puzzle number additionally (0~3), and **`initialize()`** the **`Jewel`**s according to one of the pre-defined configurations as depicted on the next slide
  - If the user chooses 3, terminate the program
  - Otherwise, ask the user to choose again

Providing UI for Playing Bejeweled

- **Text-based interface**
  - Pre-defined configurations



```cpp
std::vector<std::string> predefined_puzzles = {
        "!#!&%*&@&!@&!!@#!@$$**%!&!&&!##&#*@$&@$%%$$*&*@$##$#@$%@#$&#%$@#",
        "#!%%@%!&@*%!&@&!#*$$%%%&#*$#@$@!$%$@%@&!%$&%&@*%*$&&*&#!$$&*$#*!",
        "*@&*@#%%&%%&!$!*%#%*!*##*$$###*$$!#&&@*$$@#&#$&$$#!!!**@##@@@!!!",
        "$#@!%@$#$&$&!!*@@!$$@$!&*@**&$&@$!#*@&*@&###!@@%&@&!%&&%##$#@@&$",
};
```



0:

```
   0 1 2 3 4 5 6 7
  +---------------
0 |! # ! & % * & @
1 |& ! @ & ! ! @ #
2 |! @ $ $ * * % !
3 |& ! & & ! # # &
4 |# * @ $ & @ $ %
5 |% $ $ * & * @ $
6 |# # $ # @ $ % @
7 |# $ & # % $ @ #
```

1:

```
   0 1 2 3 4 5 6 7
  +---------------
0 |# ! % % @ % ! &
1 |@ * % ! & @ & !
2 |# * $ $ % % % &
3 |# * $ # @ $ @ !
4 |$ % $ @ % @ & !
5 |% $ & % & @ * %
6 |* $ & & * & # !
7 |$ $ & * $ # * !
```

2:

```
   0 1 2 3 4 5 6 7
  +---------------
0 |* @ & * @ # % %
1 |& % % & ! $ ! *
2 |% # % * ! * # #
3 |* $ $ # # # * $
4 |$ ! # & & @ * $
5 |$ @ # & # $ & $
6 |$ # ! ! ! * * @
7 |# # @ @ @ ! ! !
```

3:

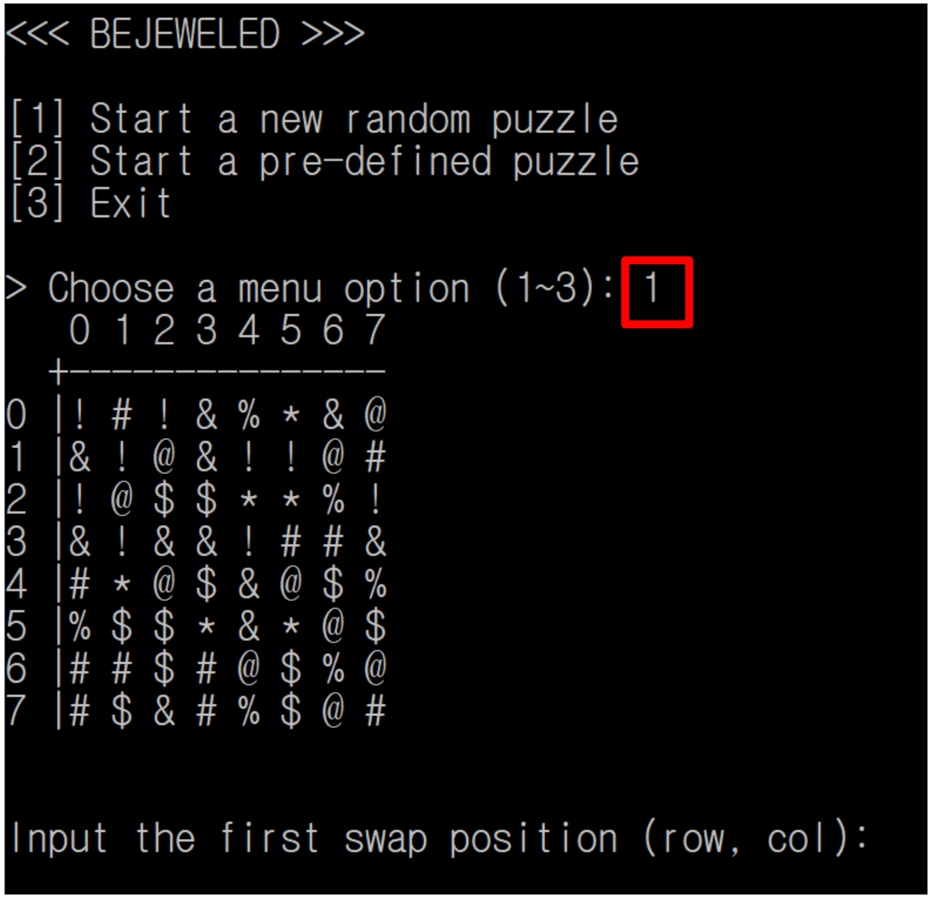
```
   0 1 2 3 4 5 6 7
  +---------------
0 |$ # @ ! % @ $ #
1 |$ & $ & ! ! * @
2 |@ ! $ $ @ $ ! &
3 |* @ * * & $ & @
4 |$ ! # * @ & * @
5 |& # # # ! @ @ %
6 |& @ & ! % & & %
7 |# # $ # @ @ & $
```

Providing UI for Playing Bejeweled

- **Text-based interface**
  - If either 1 or 2 has been chosen, **`update()`** the **`puzzle`** iteratively till no more updates are allowed, to remove all of the **`Chain`**s
  - Do the followings while the **`swap()`** is successfully done
    - Ask the user to input the first swap location (**`r1, c1`**)
    - Ask the user to input the second swap location (**`r2, c2`**)
    - Call **`swap(std::make_pair(r1,c1), std::make_pair(r2, c2));`**
    - If **`swap()`** returns **`true`**, **`update()`** the **`puzzle`** iteratively till no more updates are allowed, to remove all of the **`Chain`**s
  - Allow the user to play the game repeatedly by going back to the initial menu again
  - Each modification of the **`puzzle`**, due to **`initialize()`**, **`randomize()`**, or **`update()`**, must be followed by printing all of the **`Jewel`**s in a two-dimensional grid

Providing UI for Playing Bejeweled

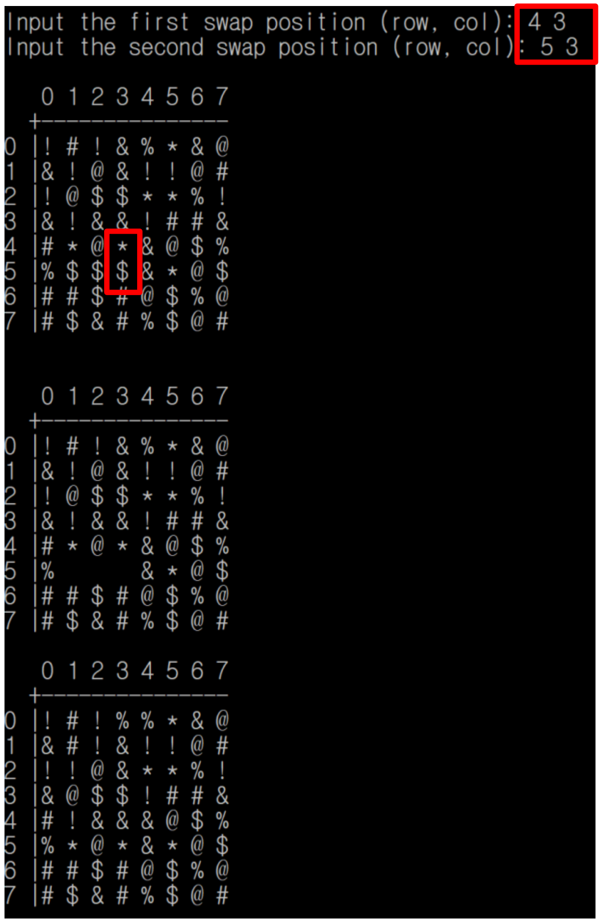
- Example



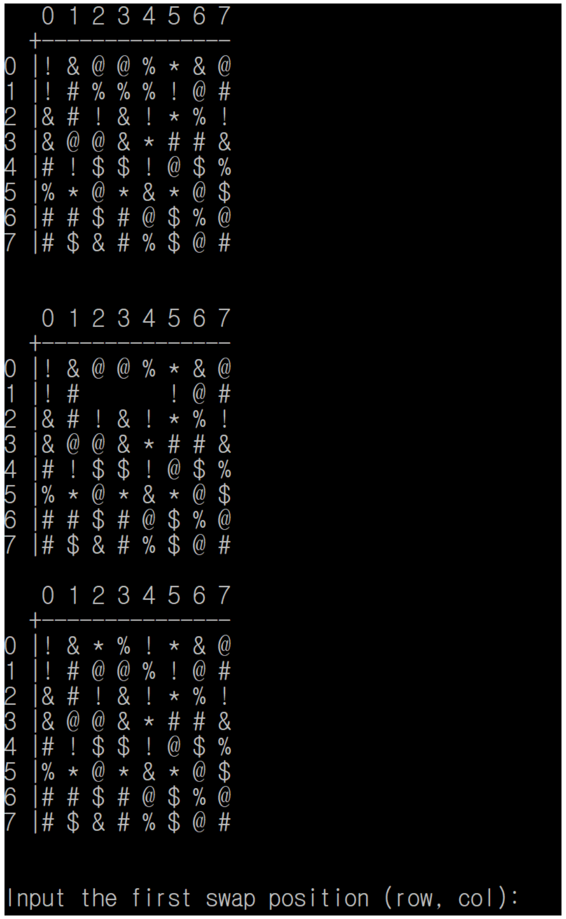
```
<<< BEJEWELED >>>

[1] Start a new random puzzle
[2] Start a pre-defined puzzle
[3] Exit

> Choose a menu option (1~3): 1
   0 1 2 3 4 5 6 7
  +---------------
0 |! # ! & % * & @
1 |& ! @ & ! ! @ #
2 |! @ $ $ * * % !
3 |& ! & & ! # # &
4 |# * @ $ & @ $ %
5 |% $ $ * & * @ $
6 |# # $ # @ $ % @
7 |# $ & # % $ @ #


Input the first swap position (row, col):
```

Providing UI for Playing Bejeweled

- Example



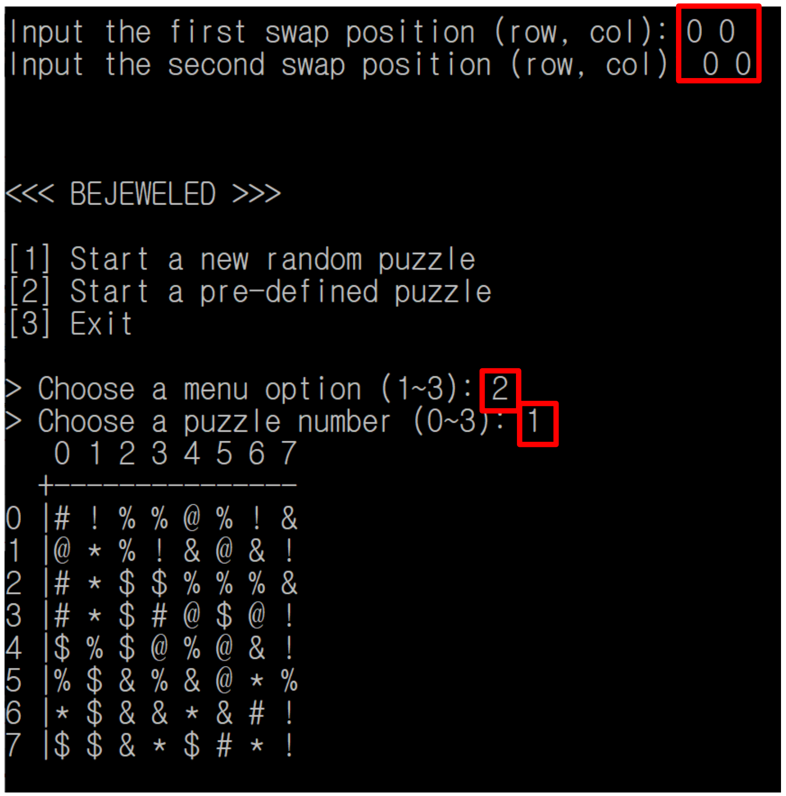
```
Input the first swap position (row, col): 4 3
Input the second swap position (row, col): 5 3

   0 1 2 3 4 5 6 7
  +----------------
0 |! # ! & % * & @
1 |& ! @ & ! ! @ #
2 |! @ $ $ * * % !
3 |& ! & & ! # # &
4 |# * @ * & @ $ %
5 |% $ $ $ & * @ $
6 |# # $ # @ $ % @
7 |# $ & # % $ @ #


   0 1 2 3 4 5 6 7
  +----------------
0 |! # ! & % * & @
1 |& ! @ & ! ! @ #
2 |! @ $ $ * * % !
3 |& ! & & ! # # &
4 |# * @ * & @ $ %
5 |%       & * @ $
6 |# # $ # @ $ % @
7 |# $ & # % $ @ #

   0 1 2 3 4 5 6 7
  +----------------
0 |! # ! % % * & @
1 |& # ! & ! ! @ #
2 |! ! @ & * * % !
3 |& @ $ $ ! # # &
4 |# ! & & & @ $ %
5 |% * @ * & * @ $
6 |# # $ # @ $ % @
7 |# $ & # % $ @ #
```



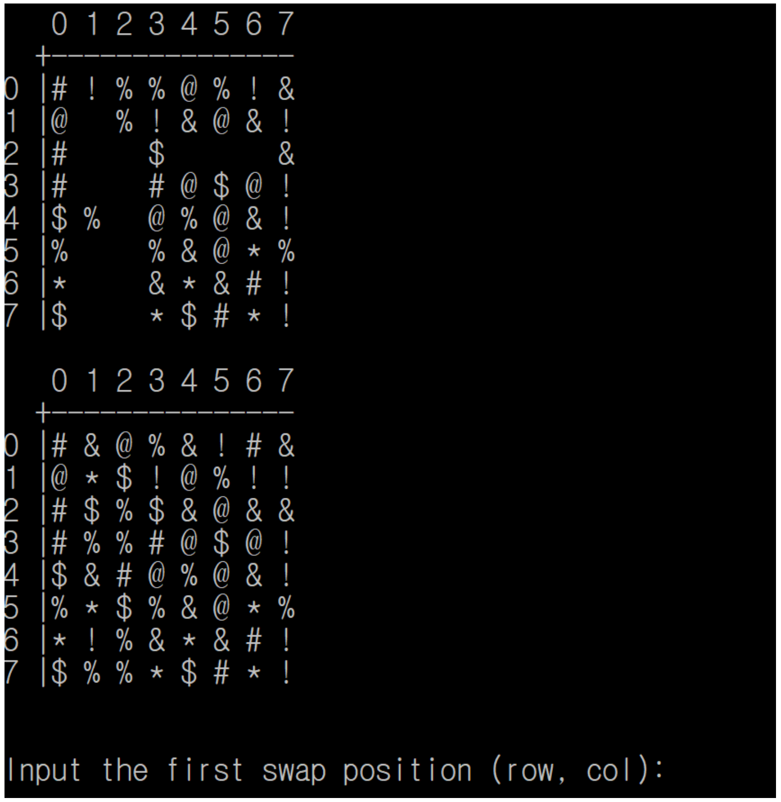
```
   0 1 2 3 4 5 6 7
  +----------------
0 |! & @ @ % * & @
1 |! # % % % ! @ #
2 |& # ! & ! * % !
3 |& @ @ & * # # &
4 |# ! $ $ ! @ $ %
5 |% * @ * & * @ $
6 |# # $ # @ $ % @
7 |# $ & # % $ @ #


   0 1 2 3 4 5 6 7
  +----------------
0 |! & @ @ % * & @
1 |! #       ! @ #
2 |& # ! & ! * % !
3 |& @ @ & * # # &
4 |# ! $ $ ! @ $ %
5 |% * @ * & * @ $
6 |# # $ # @ $ % @
7 |# $ & # % $ @ #

   0 1 2 3 4 5 6 7
  +----------------
0 |! & * % ! * & @
1 |! # @ @ % ! @ #
2 |& # ! & ! * % !
3 |& @ @ & * # # &
4 |# ! $ $ ! @ $ %
5 |% * @ * & * @ $
6 |# # $ # @ $ % @
7 |# $ & # % $ @ #


Input the first swap position (row, col):
```

Providing UI for Playing Bejeweled

- Example



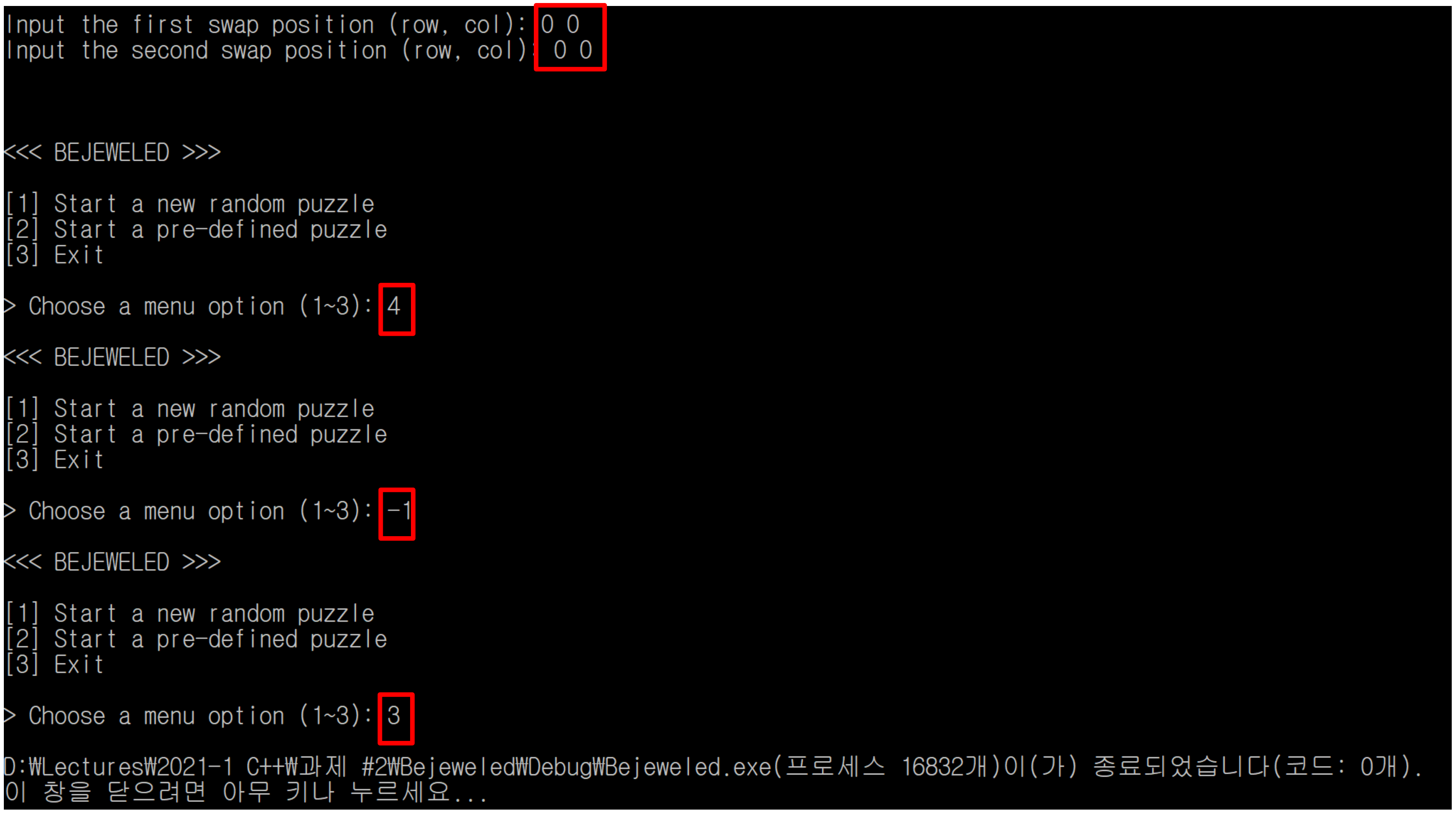
```
Input the first swap position (row, col): 0 0
Input the second swap position (row, col)  0 0



<<< BEJEWELED >>>

[1] Start a new random puzzle
[2] Start a pre-defined puzzle
[3] Exit

> Choose a menu option (1~3): 2
> Choose a puzzle number (0~3): 1
   0 1 2 3 4 5 6 7
  +---------------
0 |# ! % % @ % ! &
1 |@ * % ! & @ & !
2 |# * $ $ % % % &
3 |# * $ # @ $ @ !
4 |$ % $ @ % @ & !
5 |% $ & % & @ * %
6 |* $ & & * & # !
7 |$ $ & * $ # * !
```



```
   0 1 2 3 4 5 6 7
  +---------------
0 |# ! % % @ % ! &
1 |@   % ! & @ & !
2 |#     $       &
3 |#     # @ $ @ !
4 |$ %   @ % @ & !
5 |%     % & @ * %
6 |*     & * & # !
7 |$     * $ # * !

   0 1 2 3 4 5 6 7
  +---------------
0 |# & @ % & ! # &
1 |@ * $ ! @ % ! !
2 |# $ % $ & @ & &
3 |# % % # @ $ @ !
4 |$ & # @ % @ & !
5 |% * $ % & @ * %
6 |* ! % & * & # !
7 |$ % % * $ # * !


Input the first swap position (row, col):
```

Providing UI for Playing Bejeweled

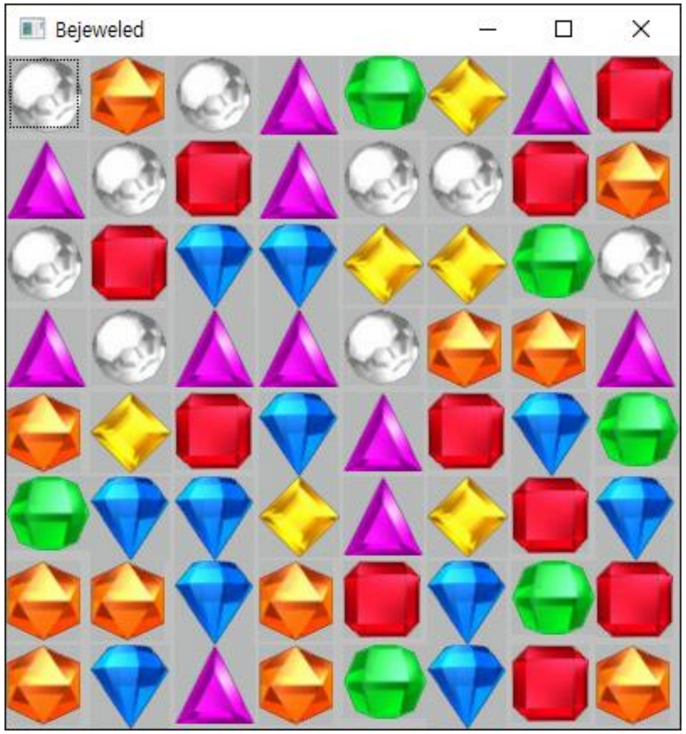
- Example



```
Input the first swap position (row, col): 0 0
Input the second swap position (row, col): 0 0



<<< BEJEWELED >>>

[1] Start a new random puzzle
[2] Start a pre-defined puzzle
[3] Exit

> Choose a menu option (1~3): 4

<<< BEJEWELED >>>

[1] Start a new random puzzle
[2] Start a pre-defined puzzle
[3] Exit

> Choose a menu option (1~3): -1

<<< BEJEWELED >>>

[1] Start a new random puzzle
[2] Start a pre-defined puzzle
[3] Exit

> Choose a menu option (1~3): 3

D:WLecturesW2021-1 C++W과제 #2WBejeweledWDebugWBejeweled.exe(프로세스 16832개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

Providing UI for Playing Bejeweled

- Graphical UI
  - You can test if your **`Puzzle`** class works correctly and enjoy the game by simply using the pre-implemented GUI code
    - Remove **`main_text.cpp`**
    - Add **`main_gui.cpp`**
    - Build and run!



Submission

- **Report**
  - Title page
    - Course title, submission date, affiliation, student ID, full name
  - Explain how you implemented in detail
    - **`Puzzle`** class (**`Puzzle.cpp/.h`**)
    - Text-based user interface (**`main_text.cpp`**)
  - Demonstrate the correctness of your class, focusing on the following functions:
    - **`initialize()`**
    - **`randomize()`**
    - **`update()`**
    - **`swap()`**
  - For each additional feature, if exist, explain what it is and how you implemented it
    - e.g. game-like features (scoring, ranking, etc.), additional rules, more graphical UIs, etc.
  - Conclude with some comments on your work
    - Key challenges you have successfully tackled
    - Limitations you hope to address in the future

Submission

- **Compress your code and report into a single `*.zip` file**
  - Code
    - The entire project folder including **`*.sln`**, **`*.cpp`**, **`*.h`**, **`*.jpg`**, etc.
    - Remove unnecessary folders such as **`.vs`** and **Debug**
    - The grader should be able to open the **`*.sln`** and build/run the **text-based UI** project immediately without any problems
  - Report
    - A single **`*.pdf`** file
    - You should convert your word format (**`*.hwp`**, **`*.doc`**, **`*.docx`**) to PDF format (**`*.pdf`**) before zipping
  - Name your zip file as your student ID
    - ex) **`2012726055.zip`**

- **Upload to homework assignment menu in KLAS**

- **Due at 5/27 (Mon), 11:59 PM**