Hong Kong Diploma of Secondary Education Examination

School Based Assessment

Information and Communication Technology

Option D: Software Development

Topic: Puzzle & mini games

School: Cheung Sha Wan Catholic Secondary School

Name: Tang Hadrian Wai To

Class: 6B

Class number: 30

**Chapter 1: Preliminary Investigation and System Analysis**

**1.1: Background**

Nowadays, in this technologically-driven world, children are exposed to various kinds of entertainment such as video games and movies online. However, as the Internet is a free place where anyone can share anything, including some content deemed immoral or obscene by the general public, such as violent games and pornographic films, these children are in need of some games healthier to their minds.

In line with the worldwide trend of protecting children’s use of computers, such as the $170M fine over children’s privacy for YouTube by the U.S. Federal Trade Commission, I have developed a program that provides safer entertainment for children by providing them fun-to-play puzzle games.

**1.2: Aim**

To provide 3 different fun-to-play games that can automatically save progress, in order to train children’s logical and critical thinking alongside providing entertainment so that children can be willing to continue playing. The key aim is being fun to play, which is most suitable for children.

**1.3: Target Users**

This solution is aimed at lonely and bored people who wish for exciting games with moderate difficulty. Being easy to open and operate, this solution will be a good choice for them.

Children are another target. Being in need of exciting games that are non-violent, this can provide relief from homework and tests for them.

**1.4: Objectives**

1. Include 3 different games
2. Include clear and concise instructions for each game
3. Include user login
4. Include user registration
5. Include automatic progress saving
6. Include beautiful visuals to attract children’s attention
7. Include elements of fun

**Chapter 2: System Design**

**2.1: The User Interface**

This program runs on the Command Line Interface (CLI) instead of the Graphical User Interface (GUI), as it uses less system resources, as the program is text-based and does not require complex graphical processing, and is more cross-platform, as all computer operating systems have one. By detecting key presses instead of commands for user input and providing clear instructions on screen, the disadvantage of CLI where the user has to remember commands to input can be avoided. Moreover, using CLI, the overall aesthetics can be made more consistent as all text use the same font and the players can feel the beauty of consistency. The player will be more engaged in playing these games, thus increasing emotional responses and increasing the effectiveness of such system.

**2.2: Modularization**

In this program, there are 5 “rooms” where the controlled character can move through; of which 3 are games to play. Each room is a module where the player can navigate through different situations and obstacles, which increase the excitement and fun of the games. When the player successfully completes a room, the progress is automatically saved so that the player does not have to go through past progress again.

**2.3: Diagrams**

The following outline the design of the system, shown based on the functions and objectives stated in Chapter 1.

The following will be shown:

1. The Level 0 data flow of the system

2. The Level 1 data flow of the system

3. The Level 2 data flow of the system

4. The Structure Chart of the system

5. The System Flowchart of whole system

6. The System Flowchart of 3 Games

**2.4: The Level 0 data flow of the system**

The system consists of the puzzle game program and the player.

Player

Game input

Game state

Puzzle Game Program

Save user information

Loaded user information

D1 User information

**2.5: The Level 1 data flow of the system**

A more detailed flow can be found from this level 1 data flow diagram.

2.0

Registration Process

6.0

Connect Four Process

4.0

Number Guess Process

3.0

Title Screen Process

1.0

Login Process

D1 User information

ID & Password

Player

ID & Password Login information Registration Confirmation

Player

User IDs

New registration account

New progress

7.0

Winning Screen Process

New progress

New progress

New progress

New progress

5.0

Bulls and Cows Process

Avatar  
control Avatar control Game

Player

Avatar control and piece state

Avatar Game and number input placement  
movement state

Player

Player

Game  
 state

The details of sub-programs in the level 1 data flow diagram of the system are as follows.

|  |  |
| --- | --- |
| Sub-program | Description |
| 1.0  Login Process | This process allows user to input his/her ID and Password into the system. Then, the process will retrieve users' information from D1 data so as to do verification. |
| 2.0  Registration Process | This process allows user to input his/her ID and Password for registration. The process will then send new user's information to D1 data and send registration results to user. |
| 3.0  Title Screen Process | This process requires user to input his/her control of the avatar in-game. The process will then return the game state to the user. When this part is completed, the updated progress of the game will be saved to D1 data. |
| 4.0  Number Guess Process | This process requires user to input his/her guesses of the generated random number in-game. The process will then show the game state including whether the guess is smaller/greater than the actual number in game to user. If the guess is correct, then the gate blocking the way will open. When this part is completed, the updated progress of the game will be saved to D1 data. |
| 5.0  Bulls and Cows Process | This process requires user to input his/her guess of the code in the game. The process will then return the tips and the guessing results in-game to user. When this part is completed, the updated progress of the game will be saved to D1 data. |
| 6.0  Connect Four Process | This process requires user to input his/her choice of piece placement in the game. The process will then decide the opponent input and game state in-game to user. When this part is completed, the updated progress of the game will be saved to D1 data. |
| 7.0  Winning Screen Process | This process acts as the “prize” for the player and displays fireworks of different colors to congratulate the player. The player can choose to return to Title Screen Process and start again, where the updated progress of the game will be saved to D1 data. |

The details of the data file in the level 1 data flow diagram of the system are as follows.

|  |  |
| --- | --- |
| Database File | Description |
| D1  Data | This database file stores all the users' information, which contains users' ID, password and progress. |

**2.6 The level 2 data flow diagram of different processes**

A more detailed flow can be found from the following level 2 data flow diagrams.

Process 1.0:

1.1

Input

ID & Password

D1 Data

Player

ID & Password User Information

Verification Result

1.2

Verification

Process 2.0:

Player

2.3

Registration

2.2

Validation

2.1

Input ID

ID

ID

ID of all players

D1 Data

Password Validation flag

Registration result Player’s account

Process 3.0:

Move command

3.2

Move avatar

3.3

Update screen

3.1

Interpreting user input

Player

User input

New location

New progress

New screen

D1 Data

Process 4.0:

4.1

Interpreting user input

User input

D1 Data

Move command Add/Remove digit Submit command

New progress command

4.4

Compare input number and actual number

4.3

Update inputted number

4.2

Move avatar

Player

New screen Comparison result

New location Updated number

4.5

Remove obstacle

Updated game state

4.6

Update screen

Process 5.0:

5.1

Interpreting user input

User input

D1 Data

Move command Add/Remove digit Submit command

New progress command

5.4

Compare individual digits and their locations between input and actual

5.3

Update inputted number

5.2

Move avatar

Player

New screen Difference

New location Updated number

5.5

Remove obstacle

Updated game state

5.6

Update screen

Process 6.0:

6.1

Interpreting user input

User input

D1 Data

Move command Location to place a piece

New progress

6.5

Checking whether a win occurred

6.3

Player piece placement

6.2

Move avatar

Player

Win flag

New screen Location to place a piece Winning Player

New location

6.4

CPU piece placement

Location of pieces

Updated game state

6.6

Update screen

Process 7.0:

7.1

Interpreting user input

User input

7.4

Firework progression

D1 Data

Move command

New progress

Firework location

7.5

Firework removal

7.2

Move avatar

Player

7.3

Spawn firework

New screen

New location Firework Removed

New firework location firework

Random available Next  
 location sprite

7.6

Update screen

**2.7 The structure chart of the system**

A detailed structure of the system is shown in this chart.

Puzzle game system

User registration

Reading and writing to data file

Login system

IDs and passwords IDs and  
 passwords

User information

Authenticated flag

User login

Main game

Winning Screen

Connect Four

Bulls and Cows

Number Guessing

Title screen

Key press

Update game state

New game state

**2.8 The system flowchart of the whole system**

The flow of the whole system will be shown in the chart.

Start

Key press

Update game state

Data file

Save

Updated game state

Game

Data file

Check existence

Verification

ID & Password

ID & Password

Registration or Login

Registration ` Login

**2.9 The system flowchart of Number Guess**

This is the detailed system flowchart of the Number Guess game.

Next game

Number is too large

Data file

Save progress

Control avatar to move to the next room

Remove barrier

Number is correct

Number is too small

Compare input and actual number

Game

Enter guess

Input < Actual Input = Actual Input > Actual

**2.10 The system flowchart of Bulls and Cows**

This is the detailed system flowchart of the Bulls and Cows game.

Enter guess

Game

Check guess against answer

Trials and AnswerTrials and Answer Trials and Answer Trials and AnswerTrials and Answer Trials and AnswerTrials

Guess is incorrect Guess is correct

Display tips for this guess

Number is correct

Remove barrier

Control avatar to move to the next room

Save progress

Data file

Next game

**2.11 The system flowchart of Connect Four**

This is the detailed system flowchart of the Connect Four game.

Game

Enter piece placement position

Check horizontals

Exit and re-enter the room

CPU wins No one wins Player wins

Check verticals

CPU wins No one wins Player wins

CPU places piece at this position

CPU places piece at random position

You win!

Remove barrier

Control avatar to move to the next room

Save progress

Next game

You lose!

Check horizontals

Check verticals

Check diagonals

Simulate all 7 possible positions

Check horizontals

Check verticals

Check diagonals

You lose!

Check diagonals

CPU wins Player wins

No one wins

CPU/Player wins No one wins

CPU/Player wins No one wins

Data file

CPU/Player wins No one wins

CPU wins Player wins

No one wins Player wins

CPU wins

CPU wins No one wins Player wins

No one wins

**Chapter 3: System Implementation**

**3.1: Summary**

In this chapter, the implementation of different aspects of the system will be shown, including:

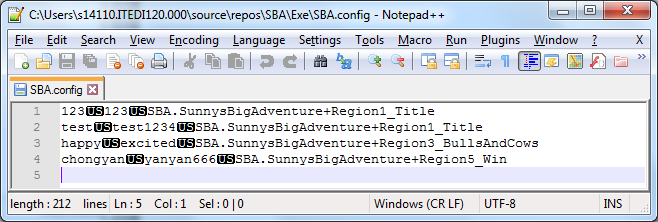
1. Database Implementation

2. User Interface Implementation

3. Process Implementation

**3.2: Database implementation**

The system consists of one data file only, named “SBA.config” with a structure as follows:



Each line is a record, where individual fields are separated by the Unit Separator (U+001F). The user’s ID is stored in the first field, the password is stored in the second field, and the progress is serialized to the third field.

The reason the Unit Separator is used instead of

1. Line breaks: The file can be viewed by an appropriate file viewer more easily like Notepad++, where each line break defines a record.
2. Commas or tabs: Users can input these characters easily and may corrupt the file.

Whenever the file is read, the framework-provided function System.IO.File.ReadAllLines is used to read the file to a string array, where each line break is interpreted as a record separator.

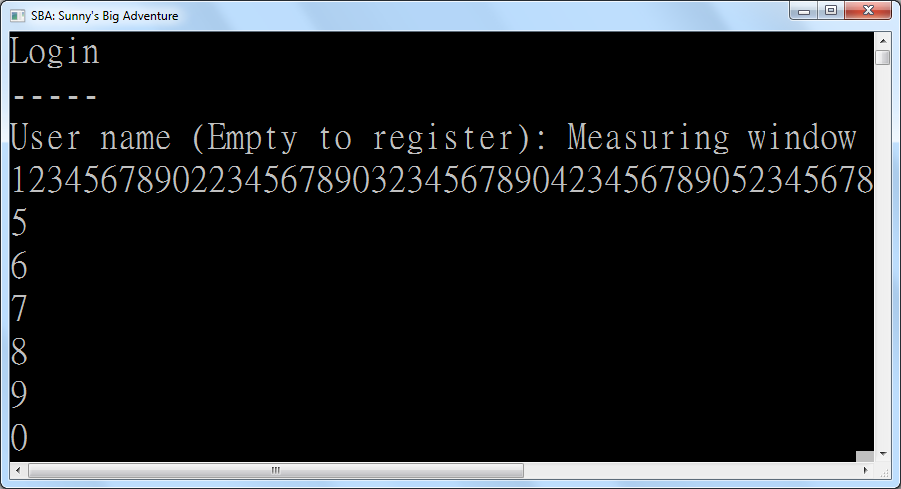
When the array is searched, each string will be split by the Unit Separator and the relevant field will be extracted and used.

If a field needs to be updated, the updated field will be combined with other unchanged fields through the Unit Separator and replace the old record in the array. The array will be saved back to the file through the framework-provided function System.IO.File.WriteAllLines.

The array will never be saved in the program memory to ensure that there is only one source of truth: the file, to avoid bugs arising from inconsistencies between the file and the program memory. Other than that, this approach can also ensure a more pleasant debugging experience where changes in the data file will be reflected in the program immediately, without having to restart the program. Moreover, program crashes will have a minimal chance of causing data loss due to not saving user data in time.

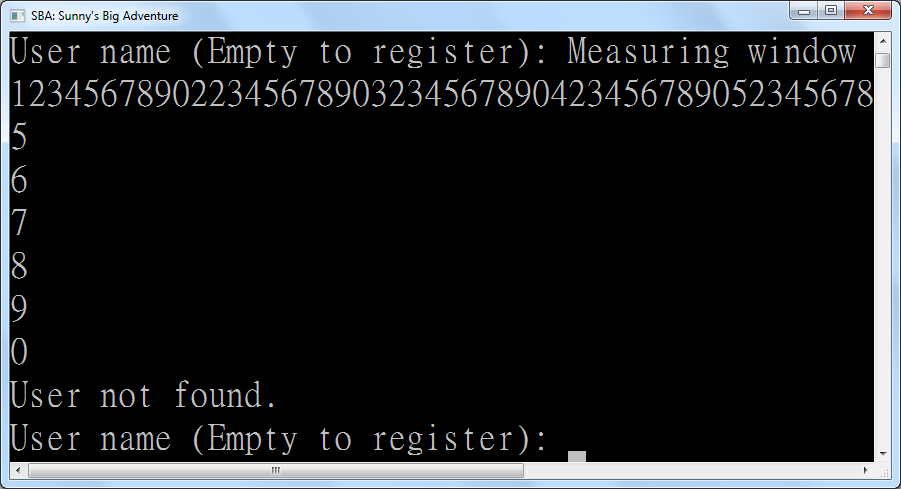
**3.3: User Interface implementation**

The program’s user interface consists of one single console window with a width of 48 blocks and a height of 10 blocks. It has a title of “SBA: Sunny’s Big Adventure” to indicate clearly that this program is a game.

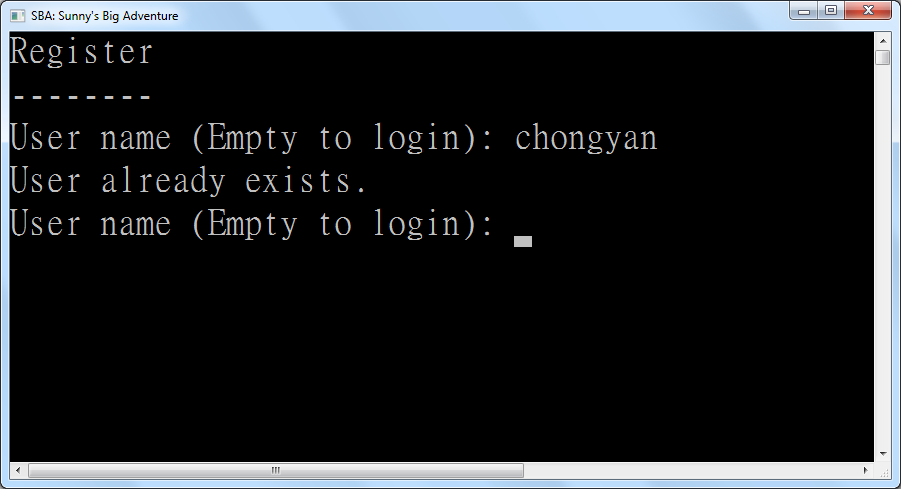


The console window is used to give instructions and guidelines to its users. Users can simply follow the instructions and guidelines given by the system and input movement keys or input keys in order to enjoy the game.

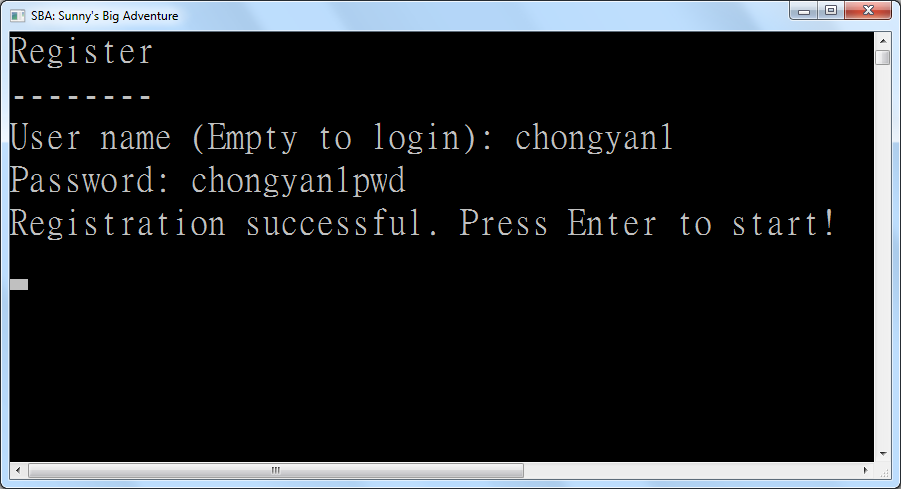
If errors related to invalid input occur, namely “User not found” and “Incorrect password” for login, as well as “User already exists” for registration, that error will be displayed to the user for inputting the data again. This will continue forever until the user finally provides valid data.



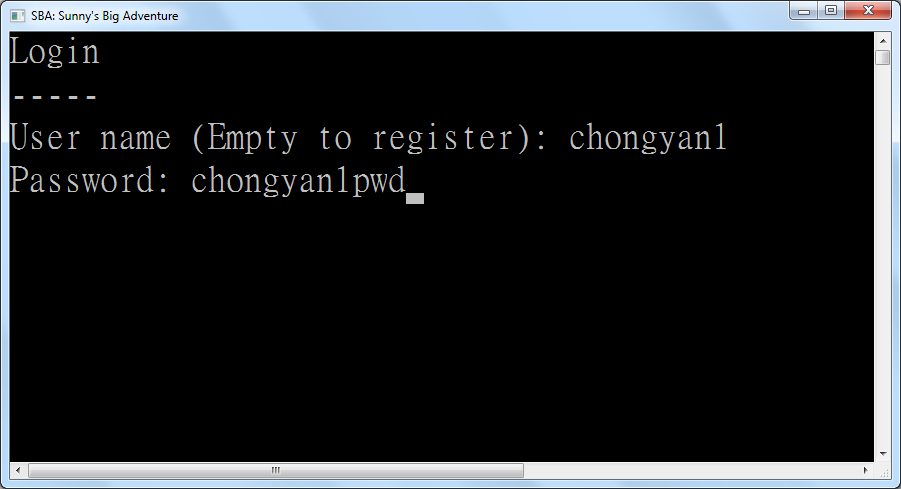
To eliminate the use of user-unfriendly “y” or “n” inputs, empty input is treated as the command for switching between login and registration screens. This way, less time is spent on inputting decisions that the user rarely inputs, like whether to register a new account, as each person usually only ever registers one account only. This way, user satisfaction is enhanced by making the usual path easy to take.



*The error for “User already exists”*



*Successful registration*



*Successful login*

**3.4: Data structure implementation**

Implementation of the processes can be described through the explanation of every structure and class. The function and characteristics of every structure and class will be mentioned one by one in this section.

**TO BE DONE**

**3.5: Process implementation (Login & Registration)**

The implementation of each non-game processes are best described through the explanation of the main function which is the entry point. The function and characteristics of the main function will be mentioned one by one in this section.



**3.6: Process implementation (Games)**

Implementation of each the game processes are contained within “regions” which act as a room for the avatar called Sunny to travel through. The implementation of each game processes are best described through the explanation of every region. The function and characteristics of every region will be mentioned one by one in this section.

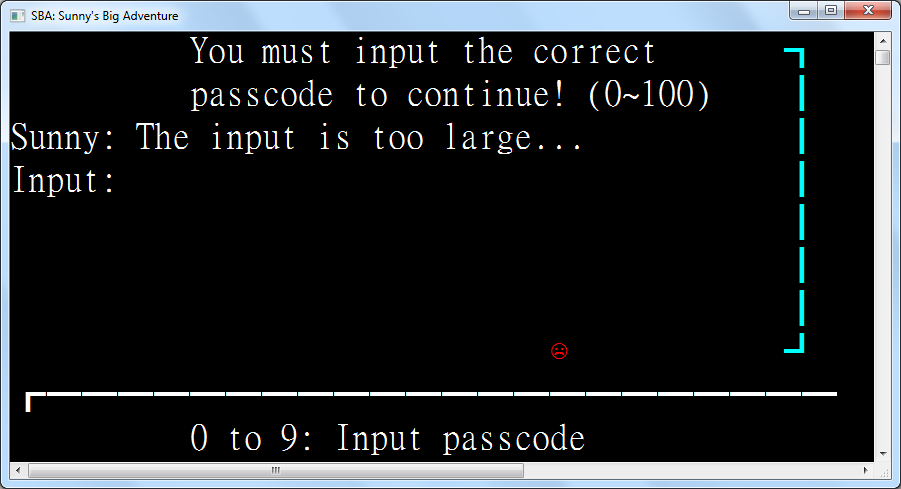
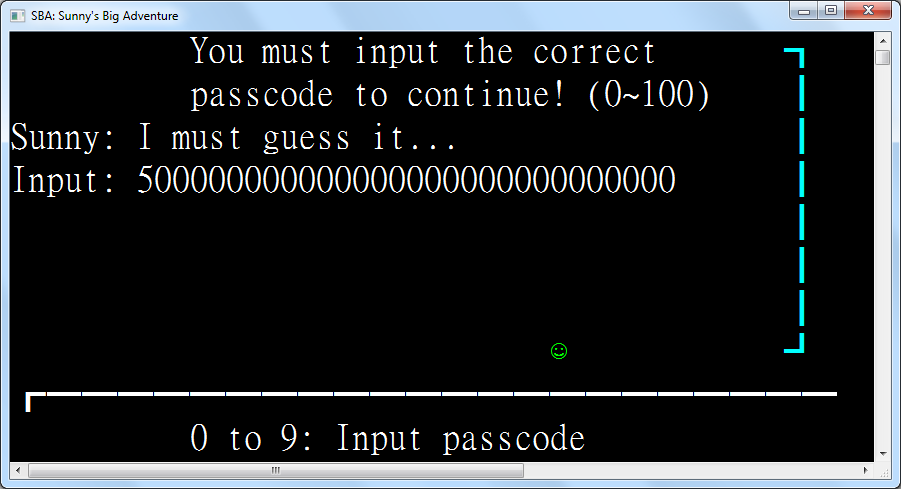
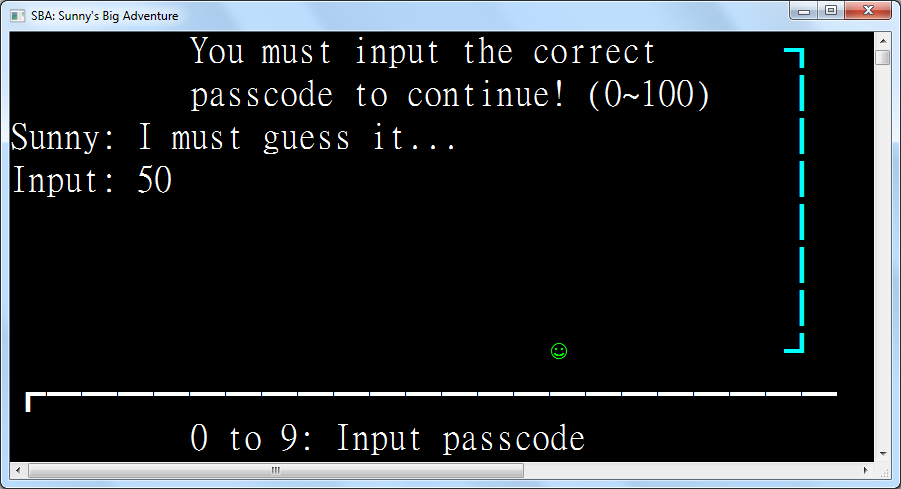
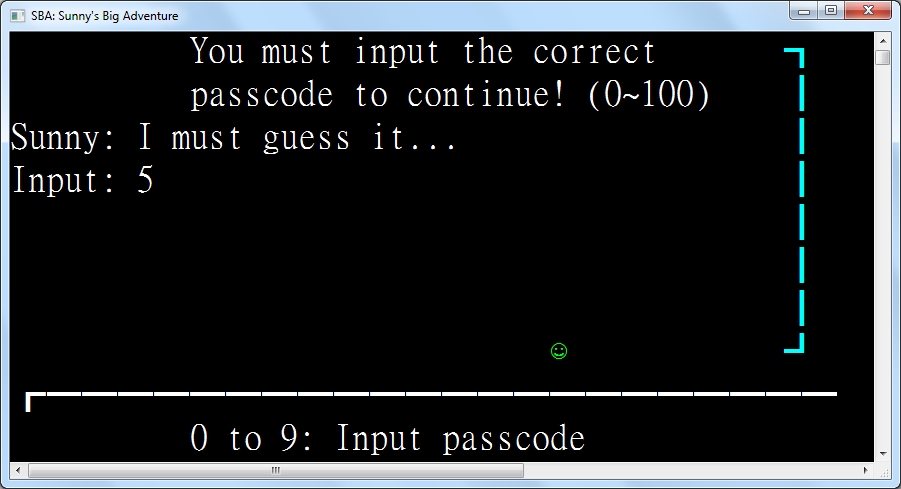
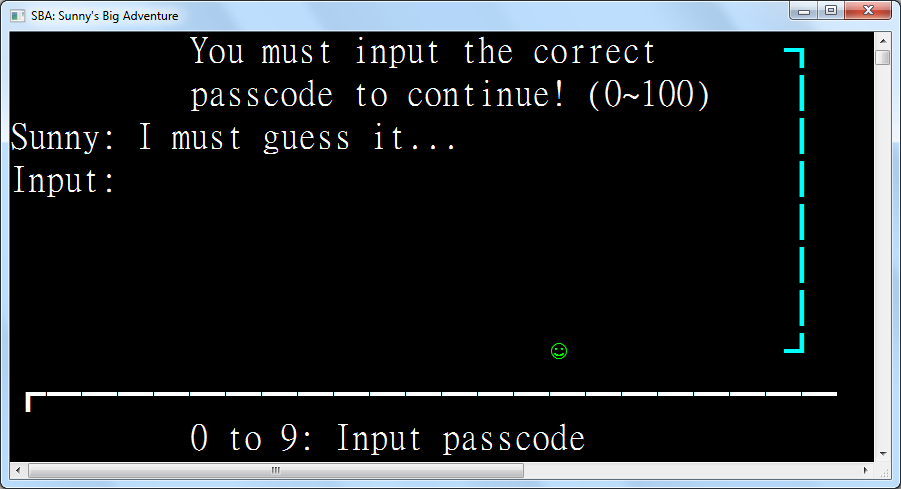
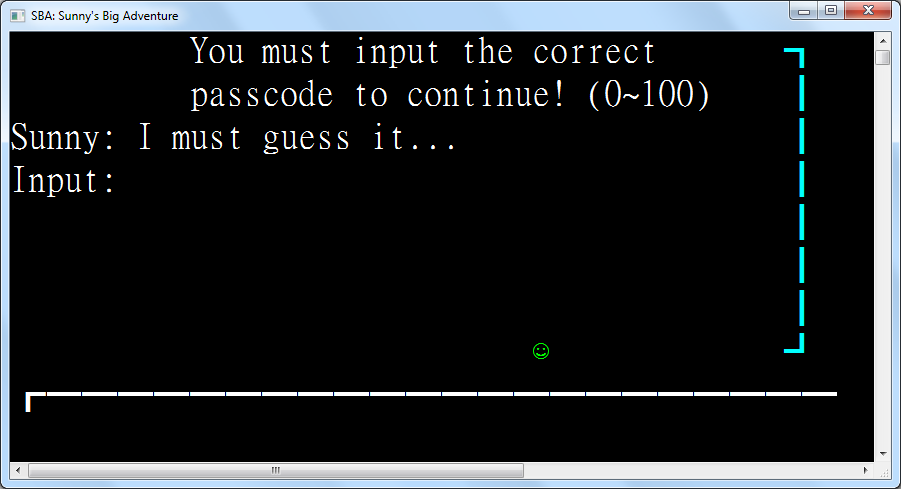
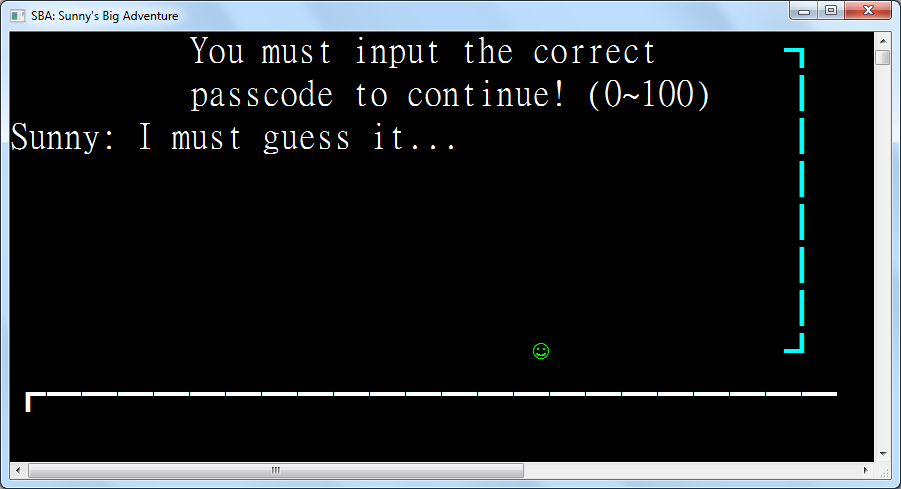
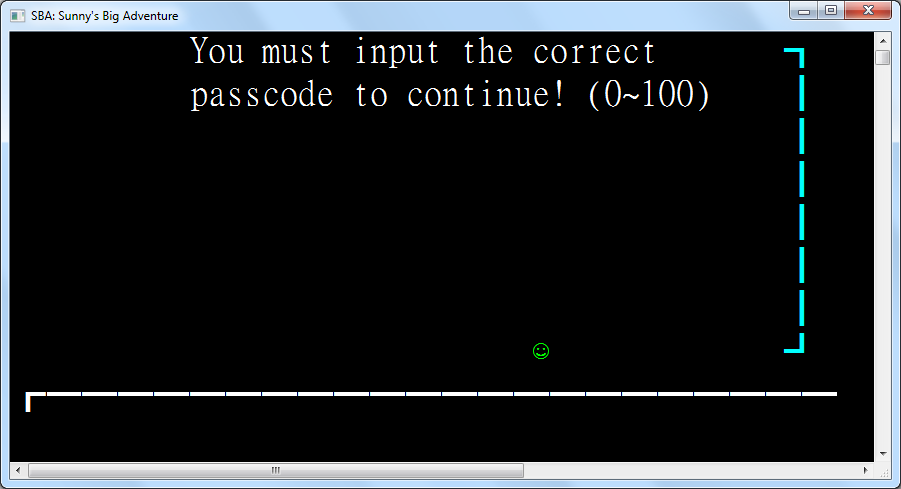
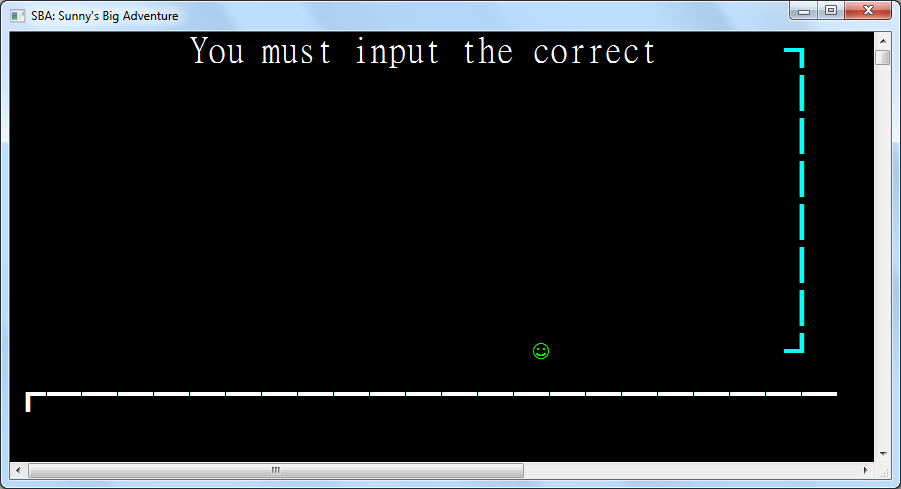
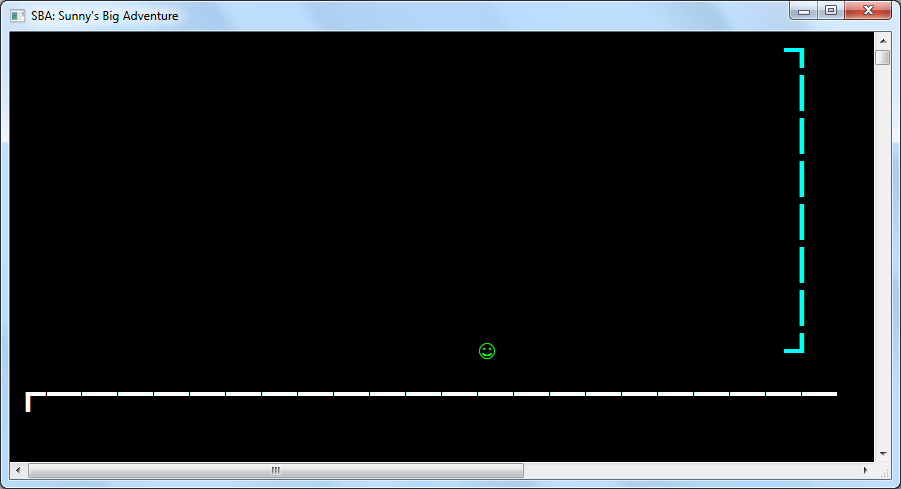
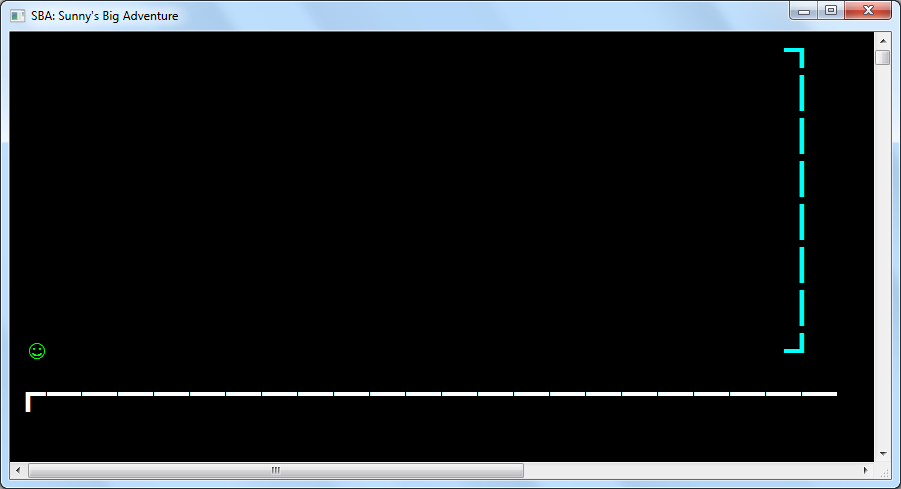
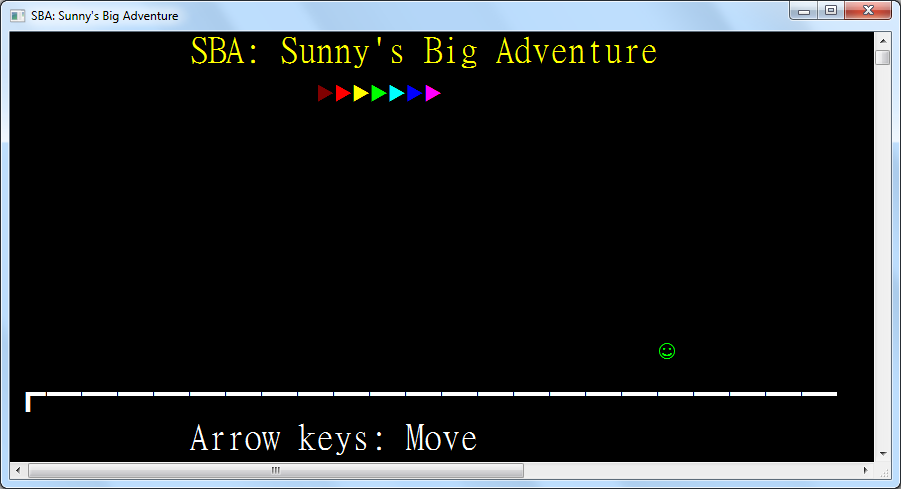
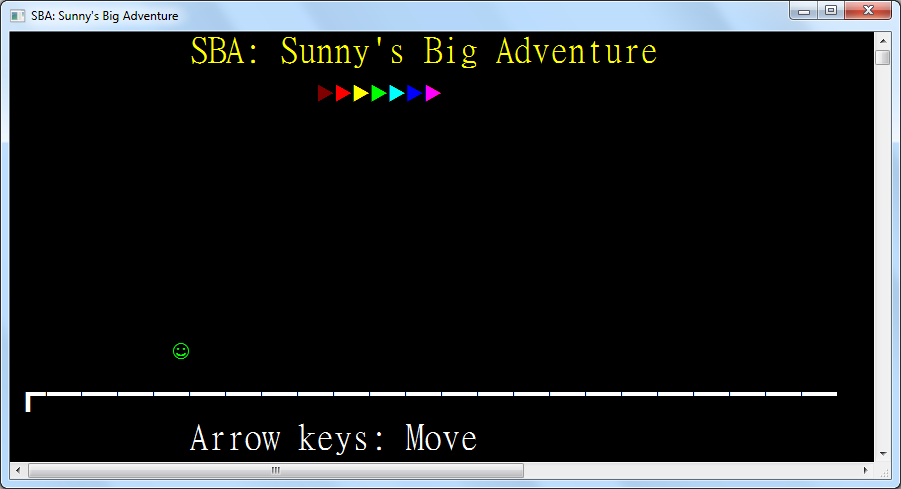
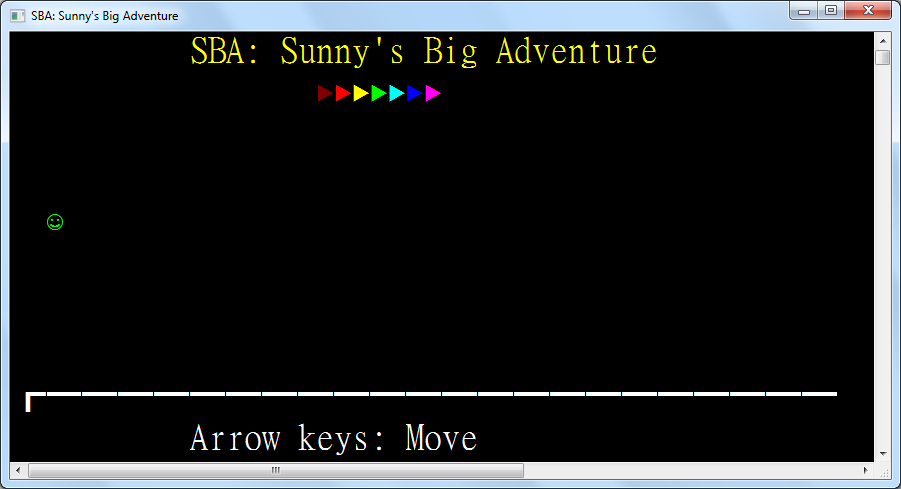
**3.6.1: Region 1 – Title Screen**

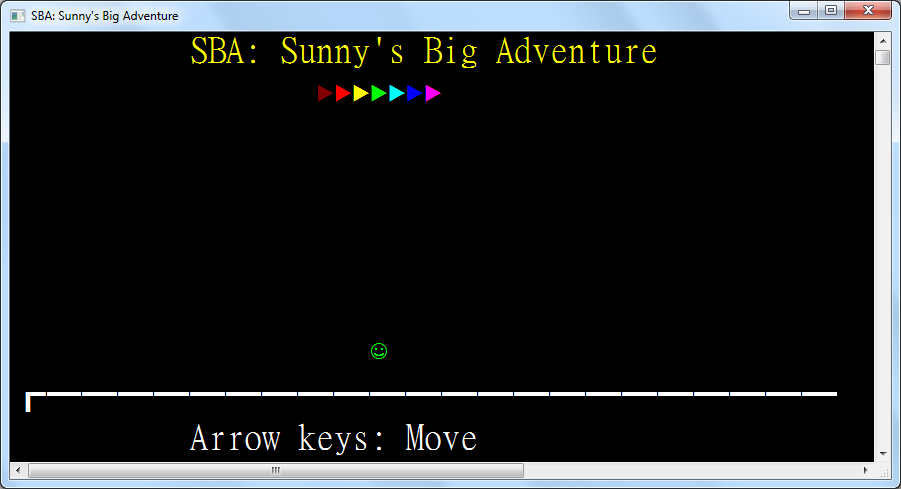
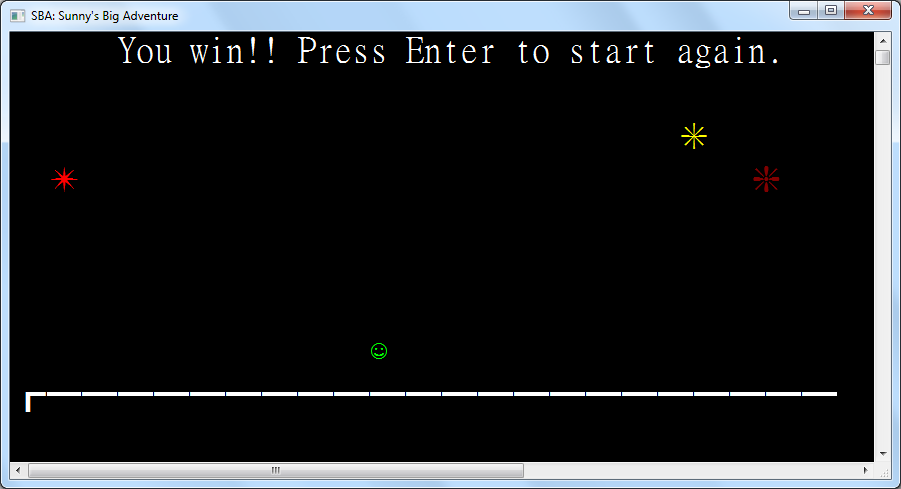
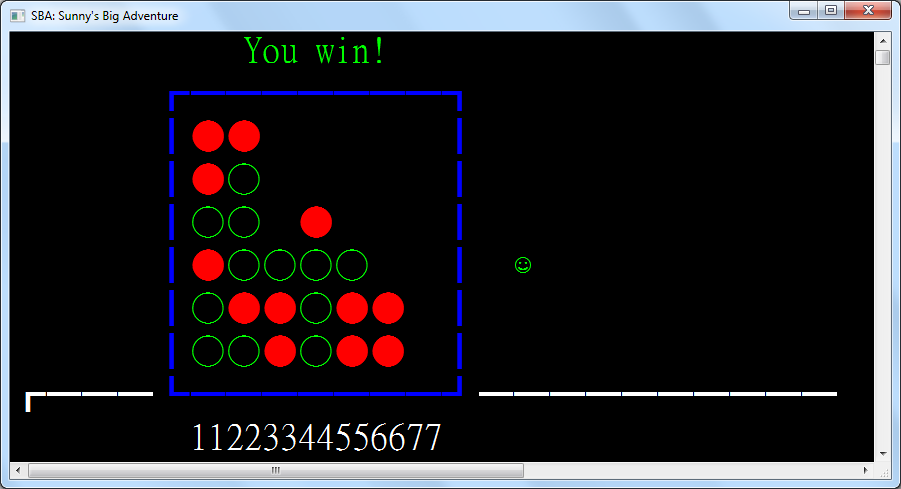
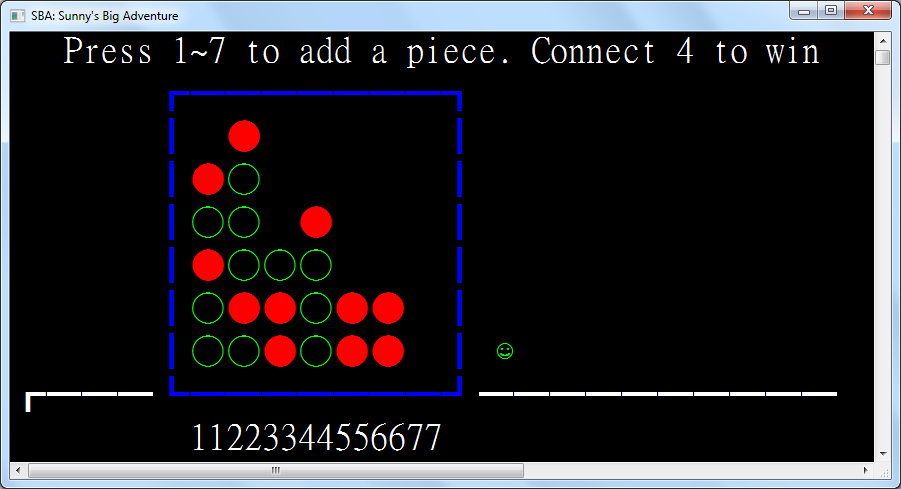
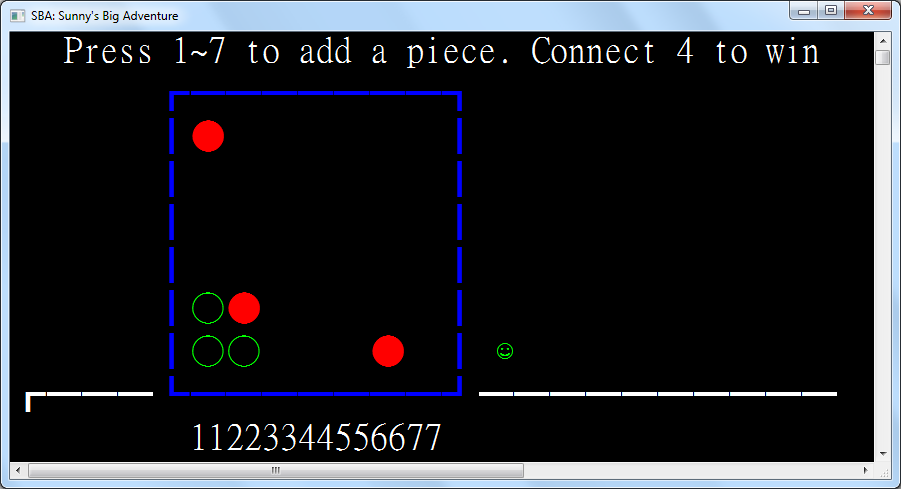
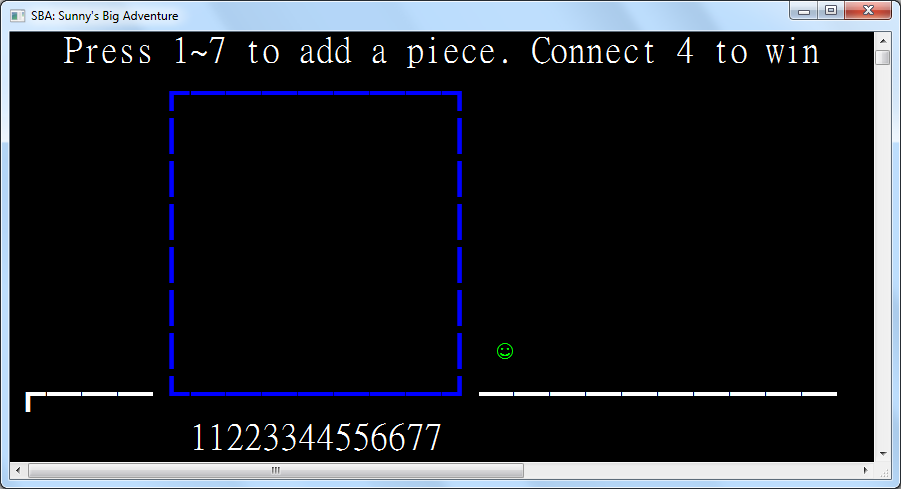
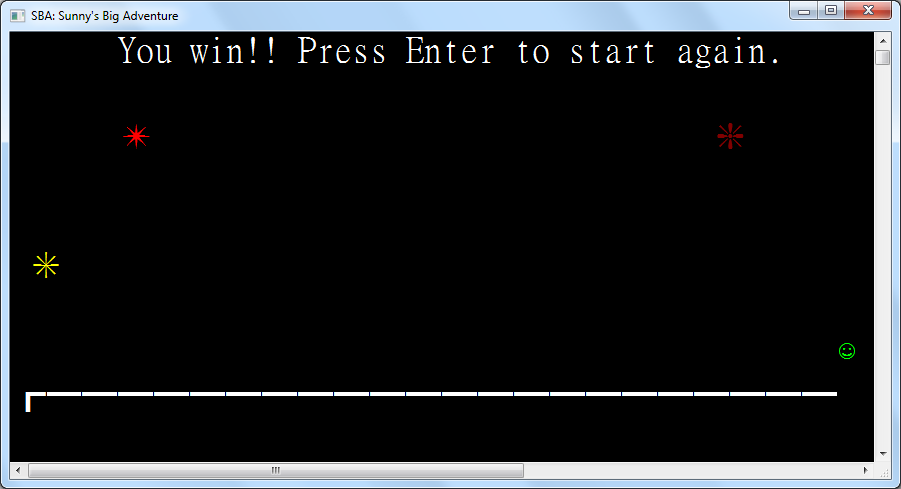
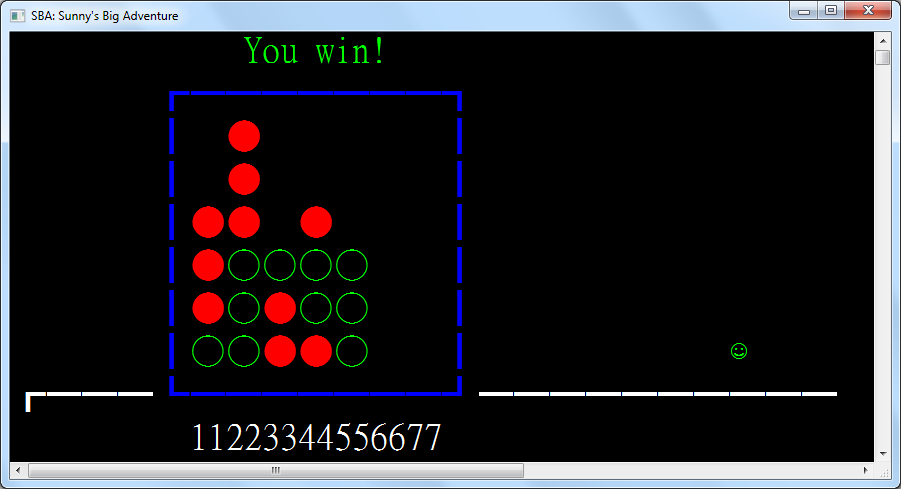
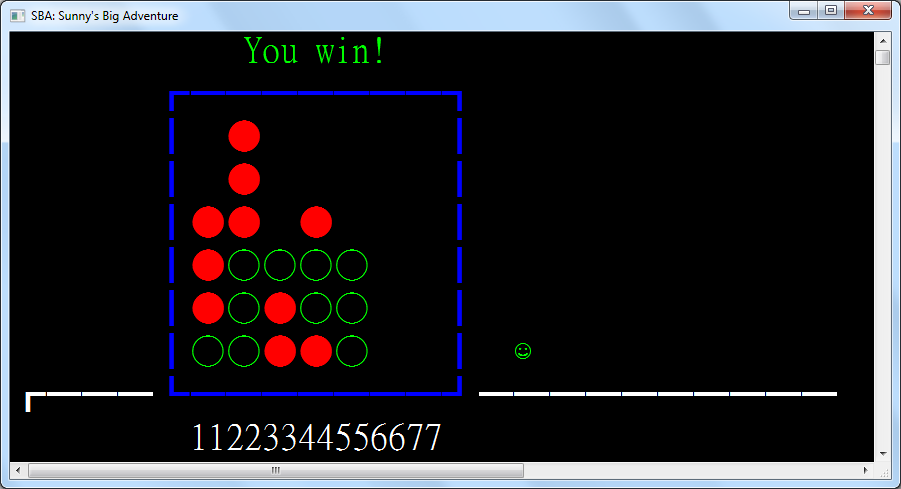
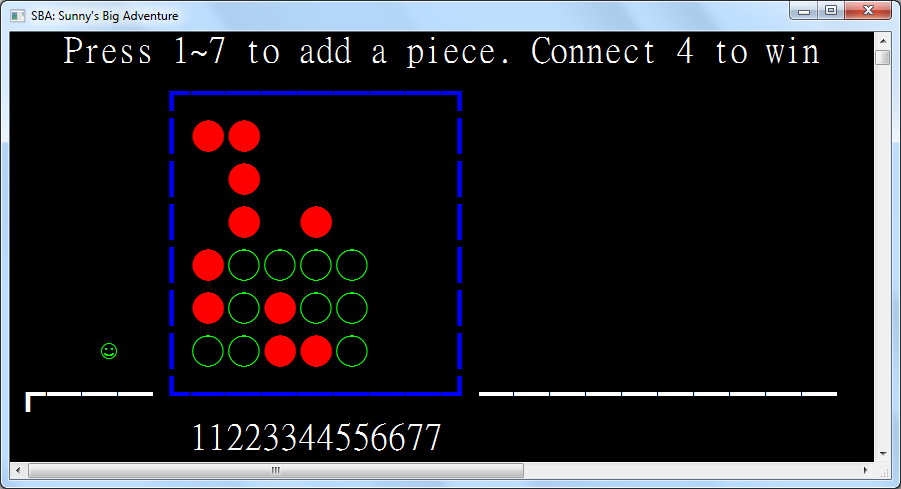
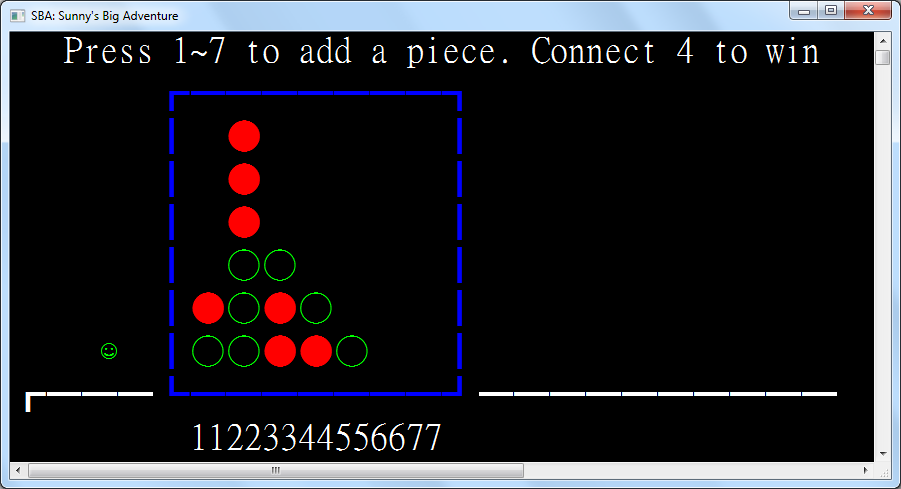
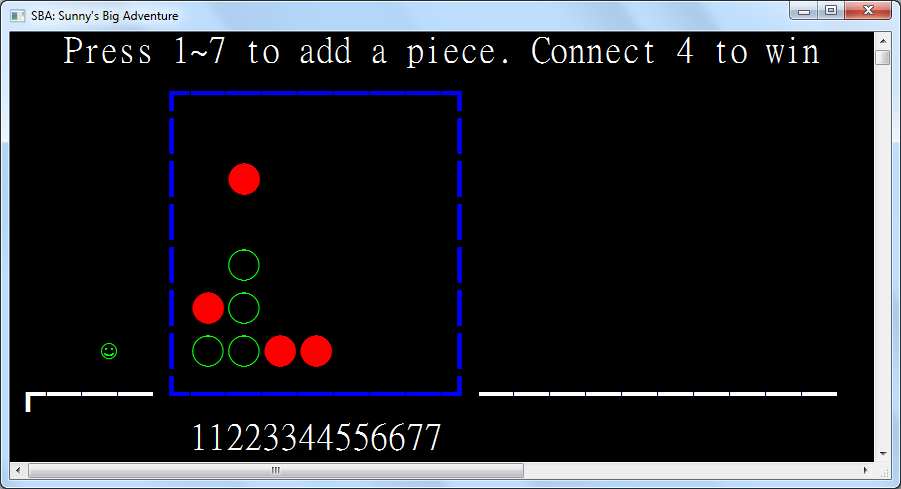
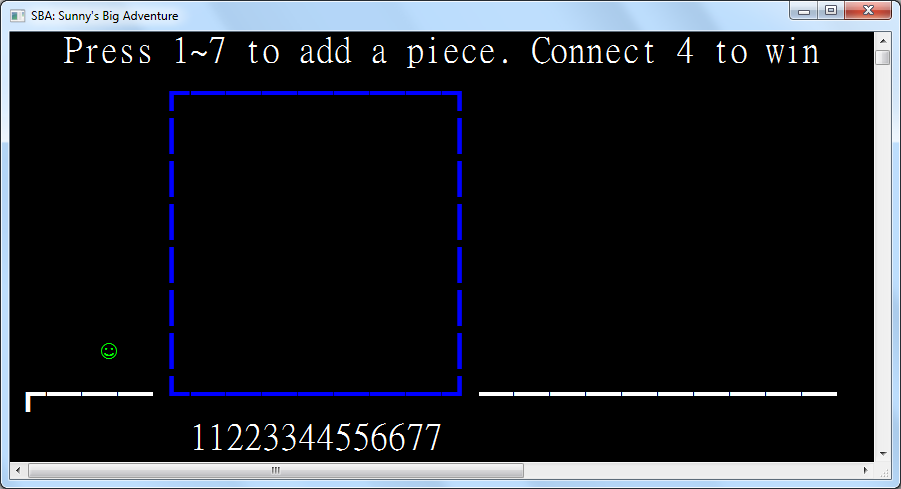
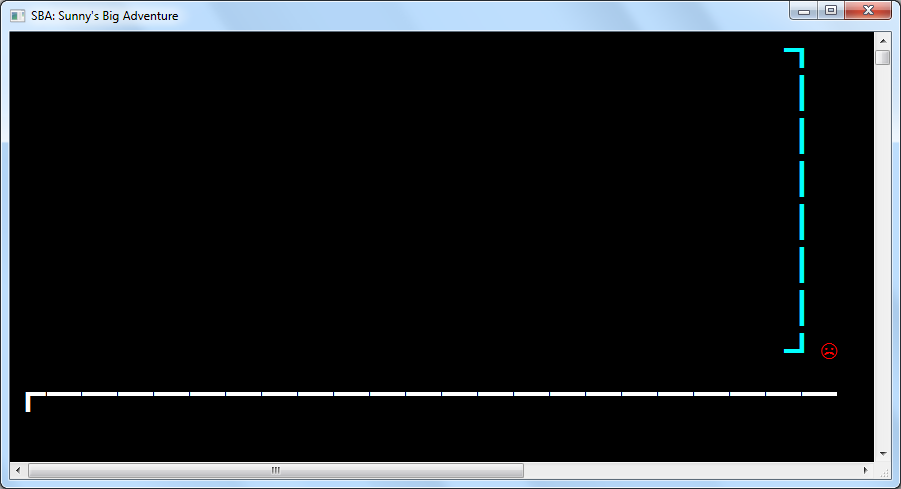
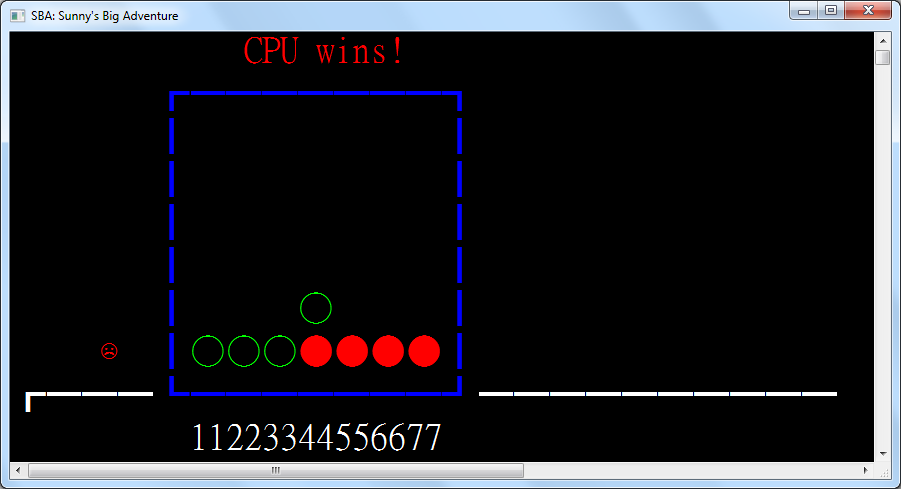
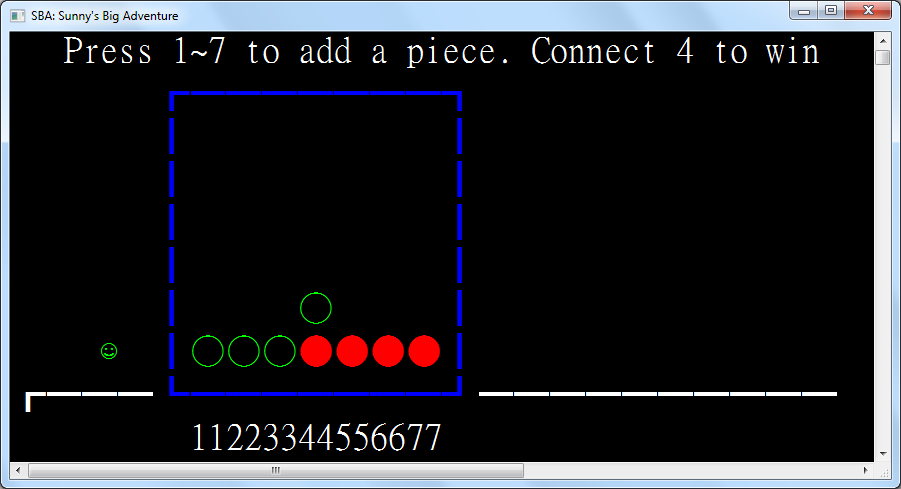
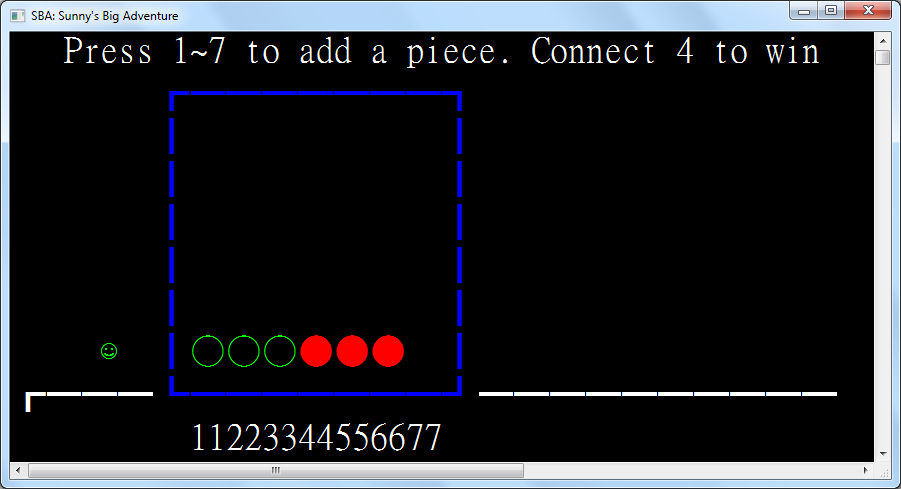
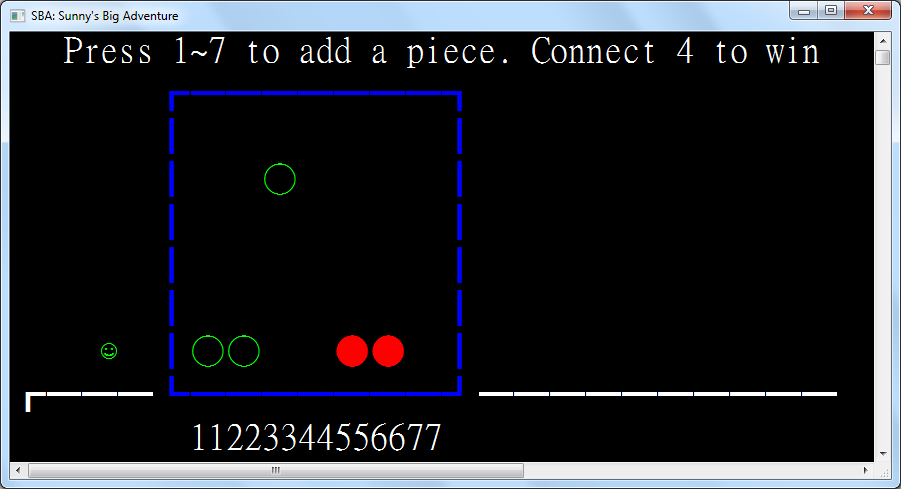
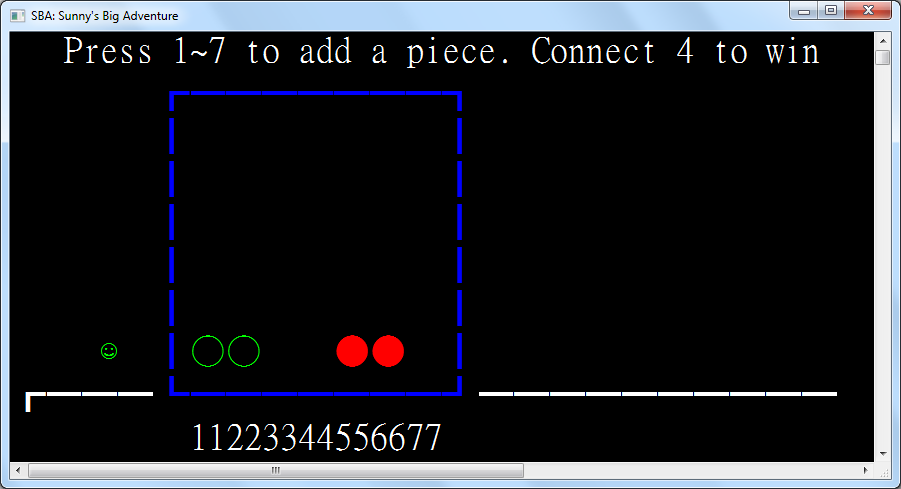
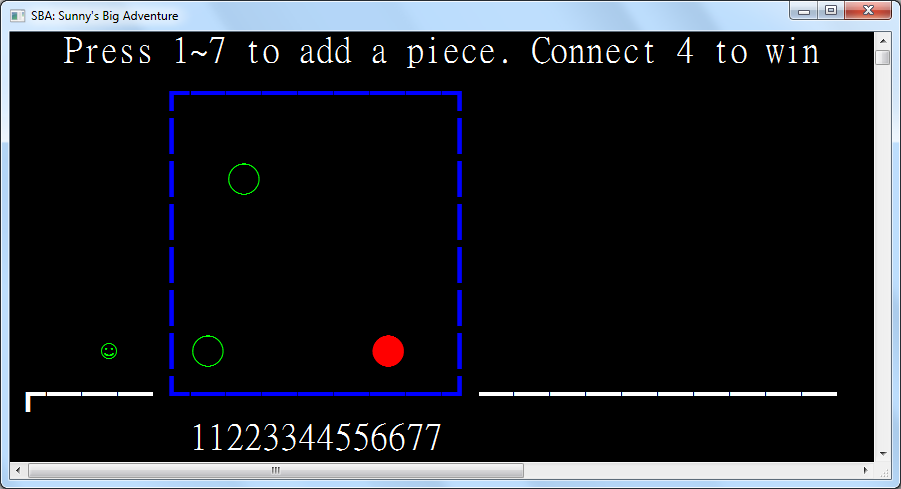
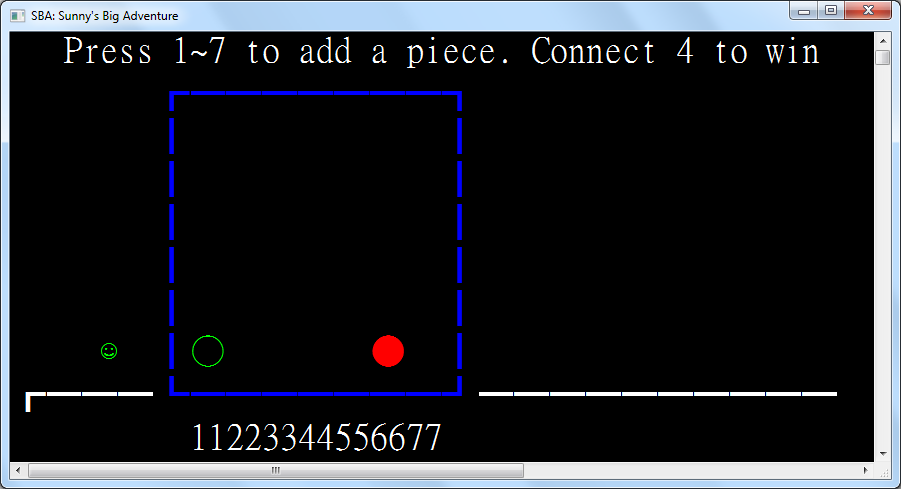
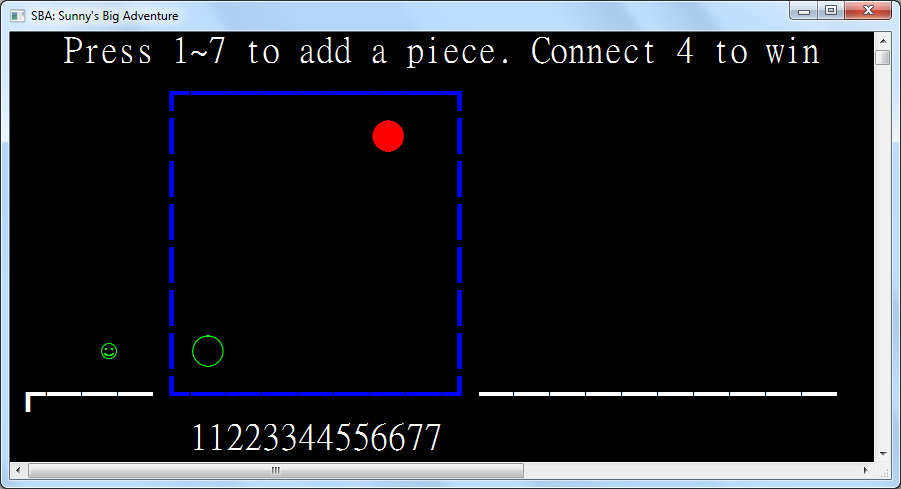
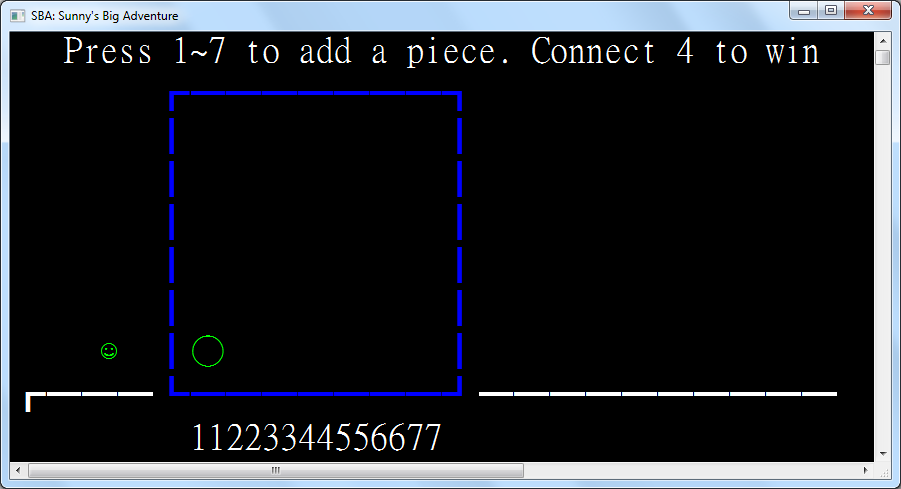
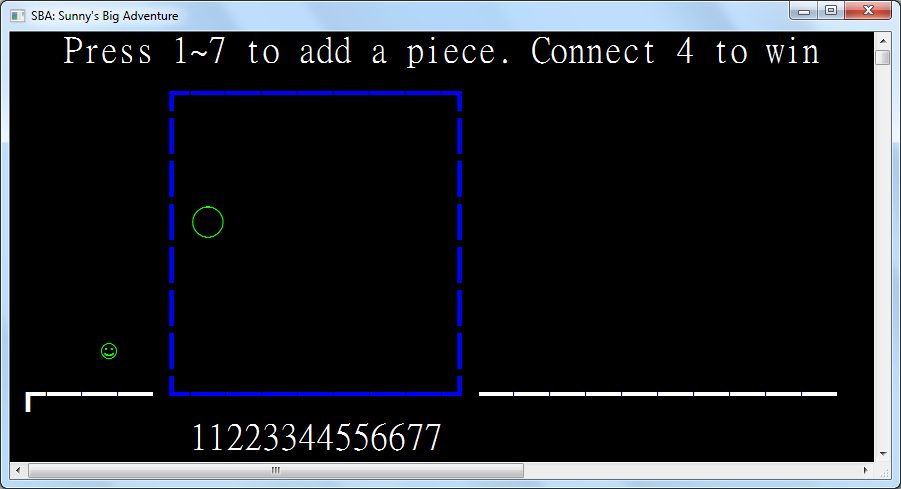
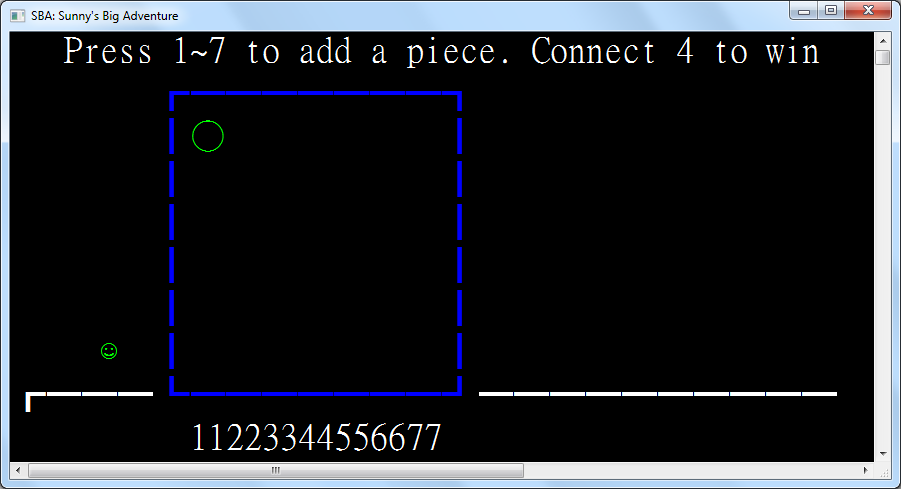
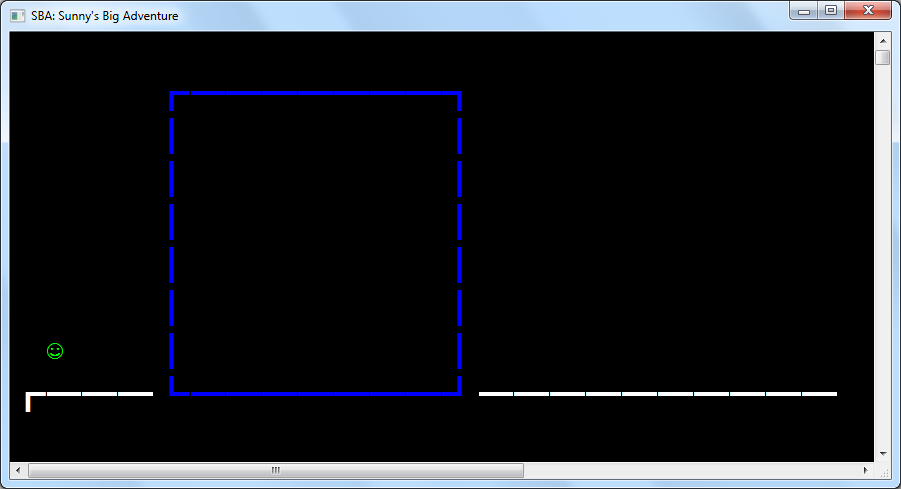
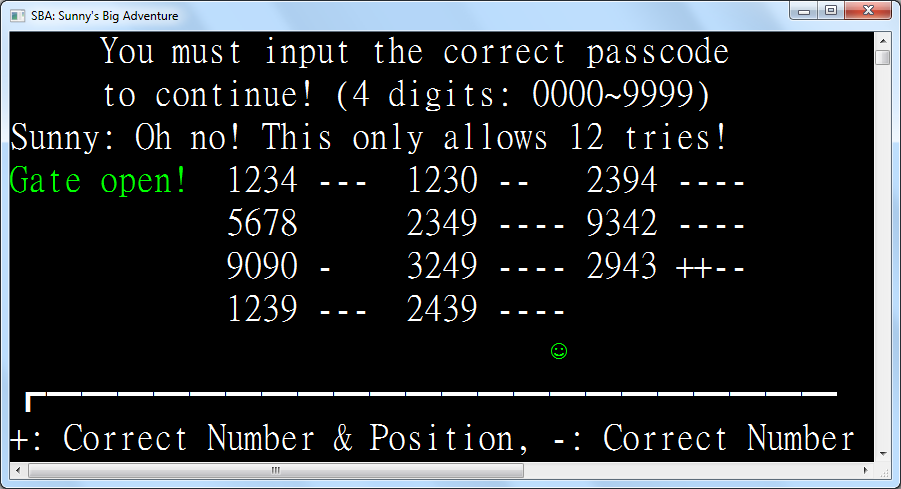
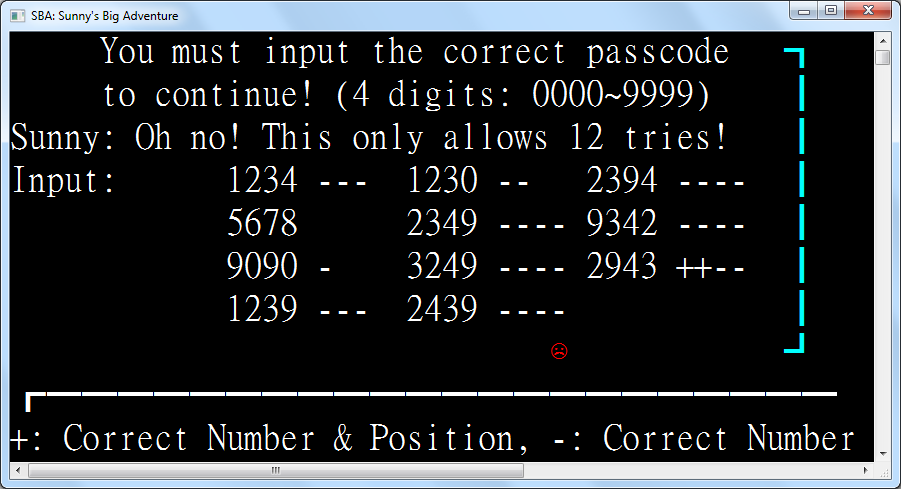
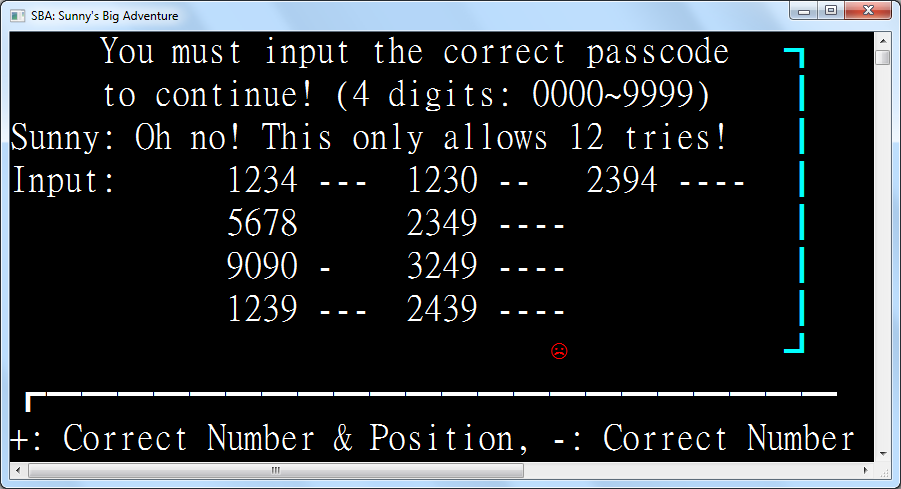
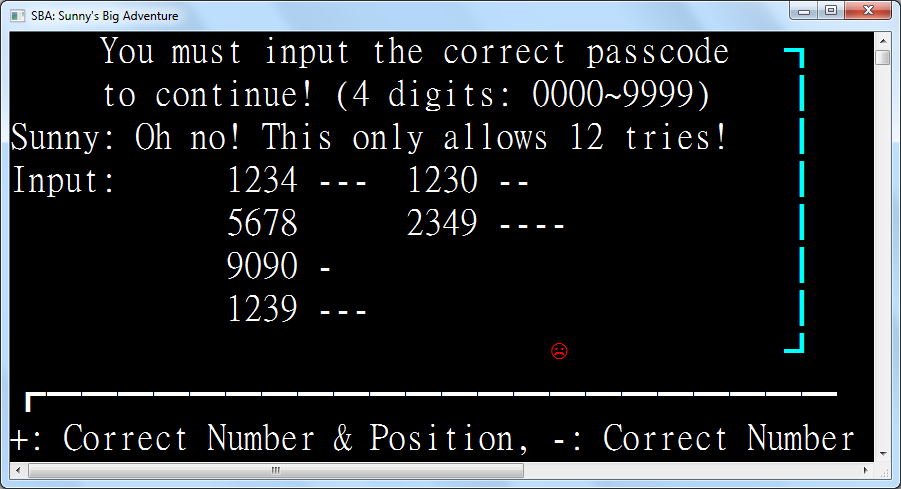
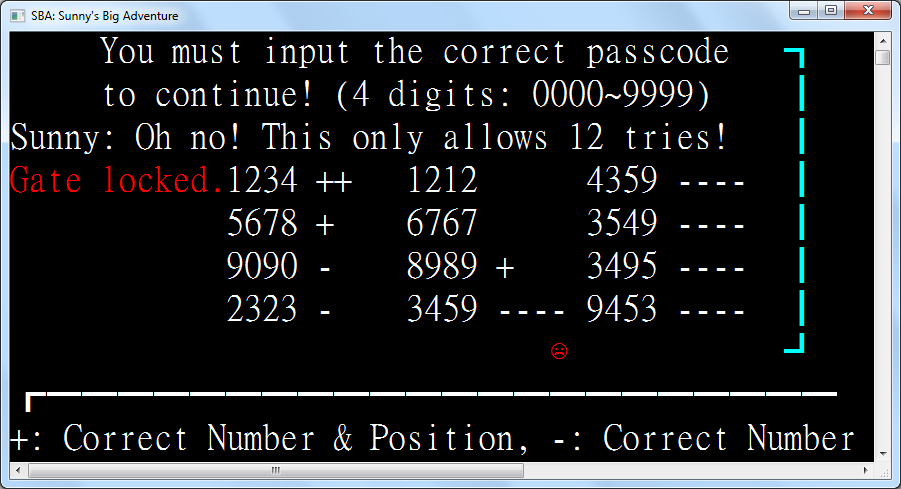
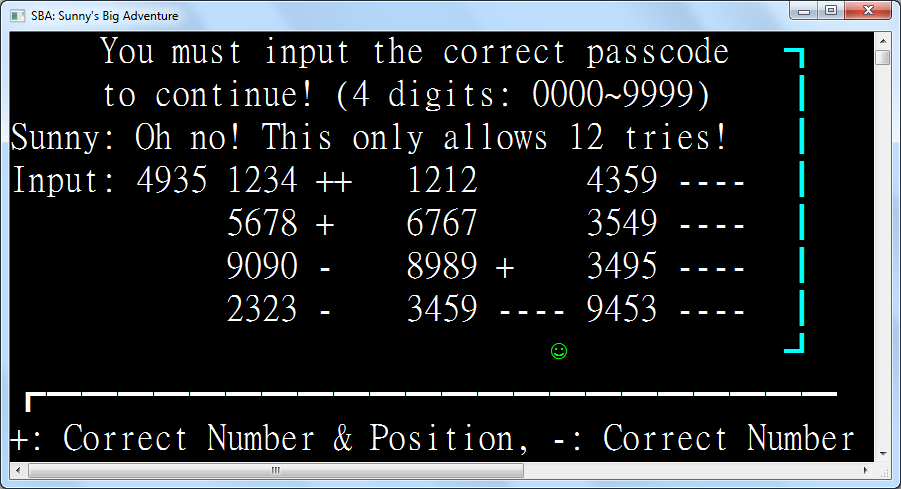
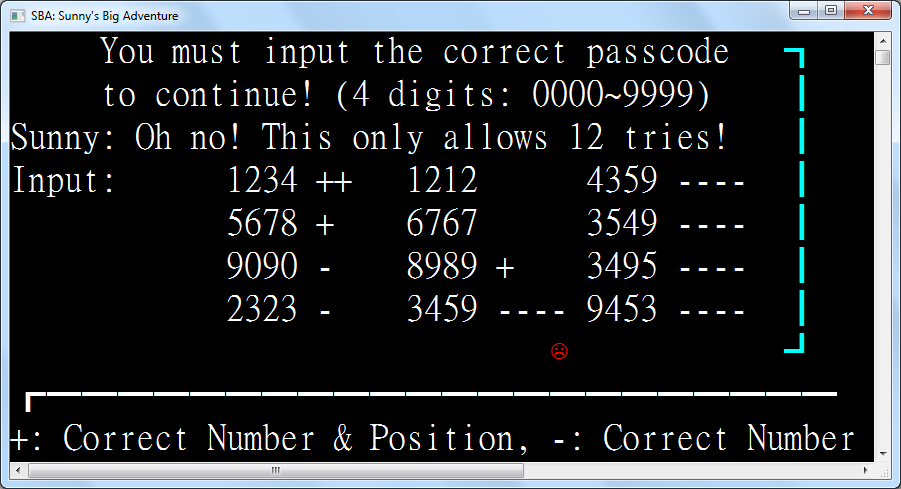
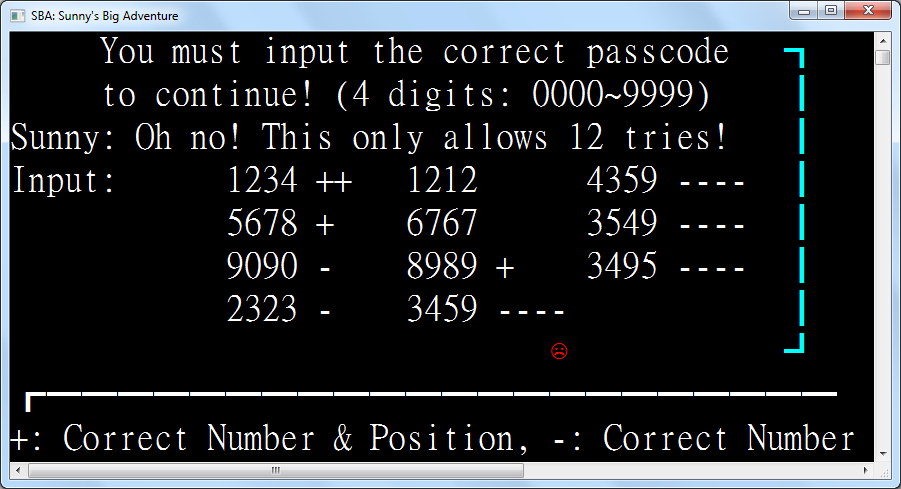
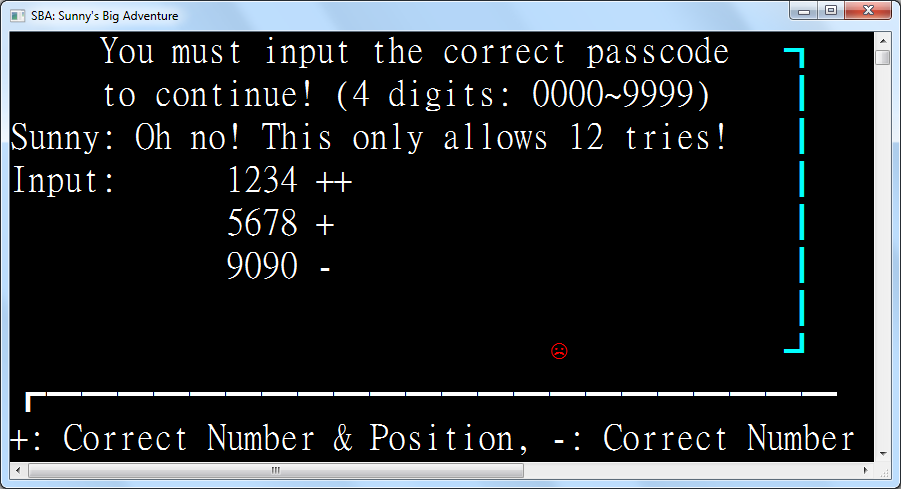
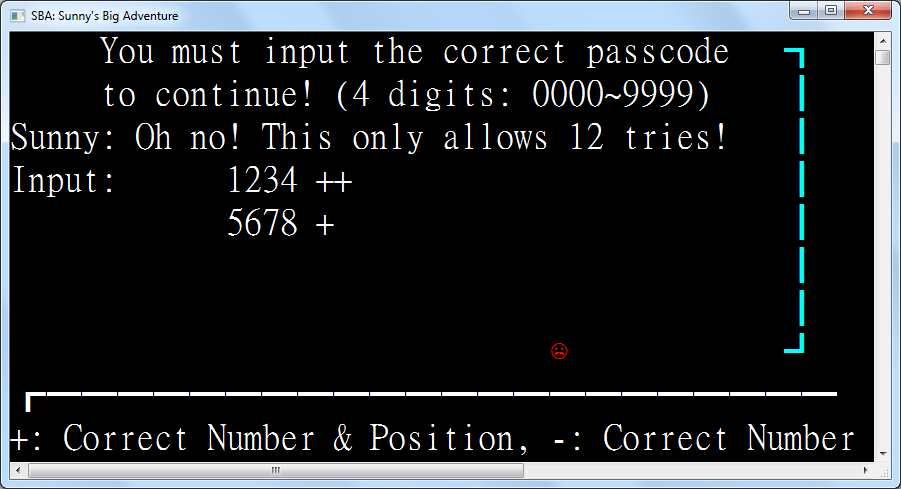
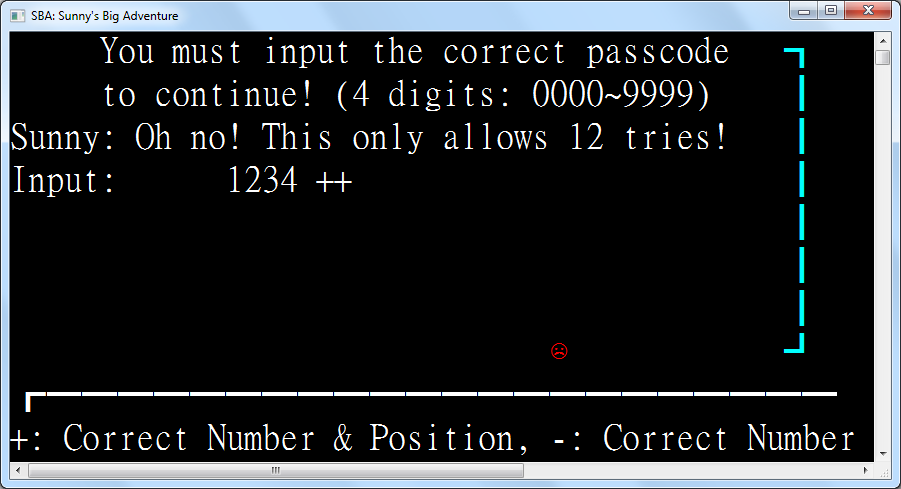
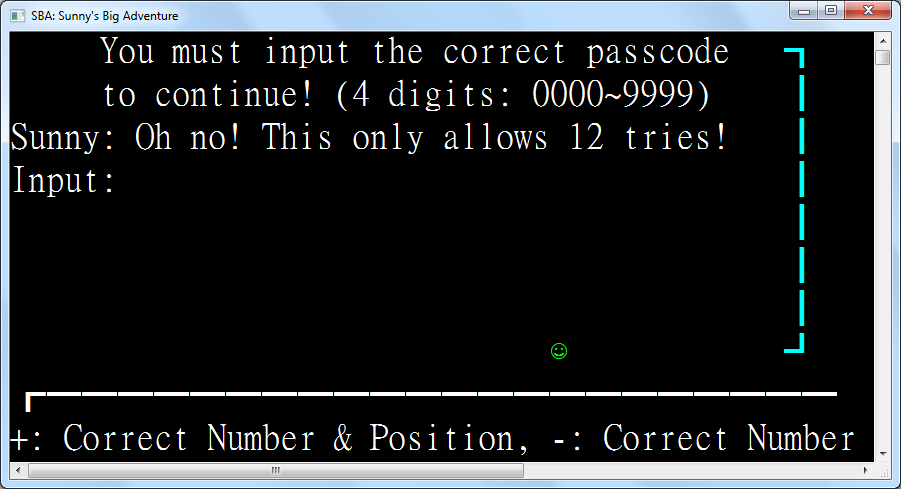
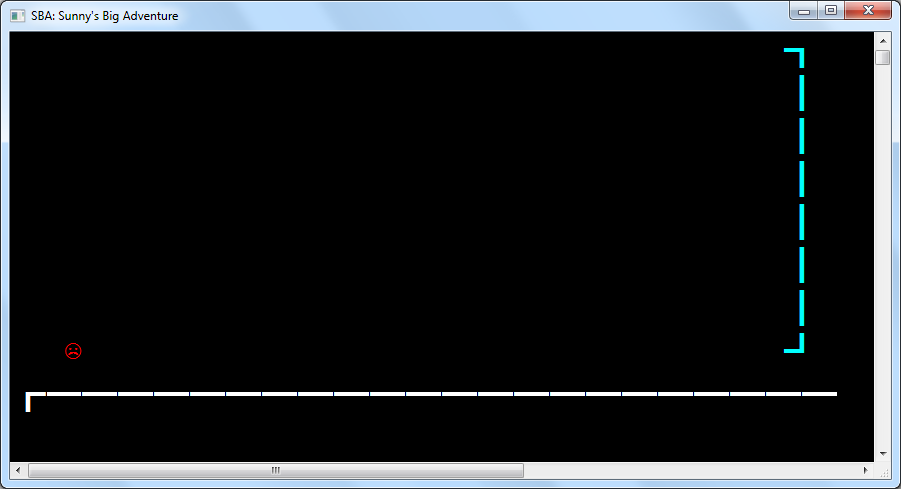
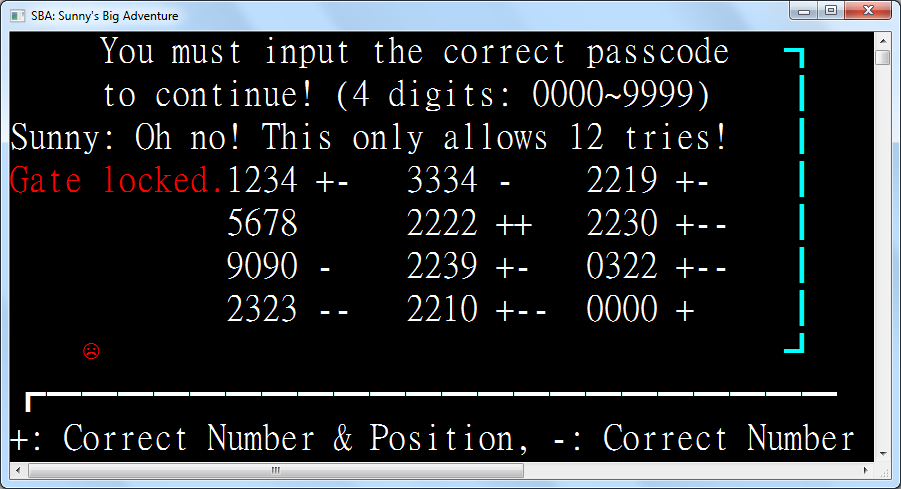
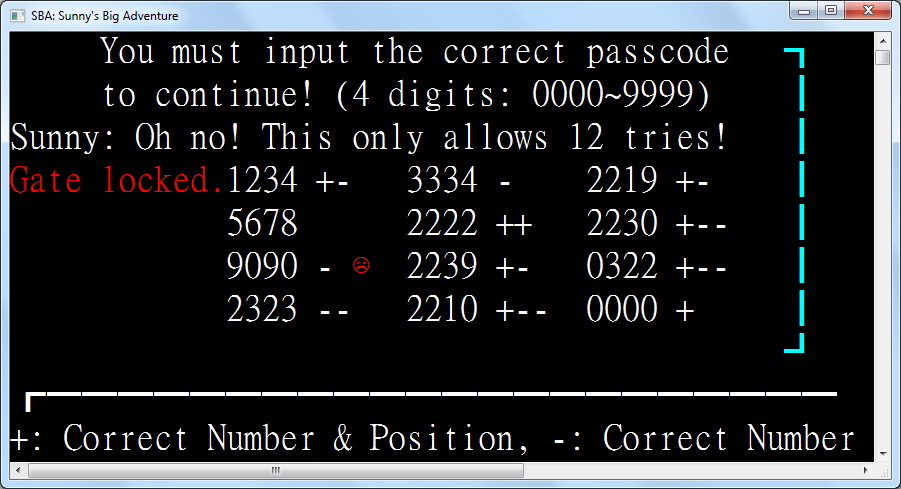
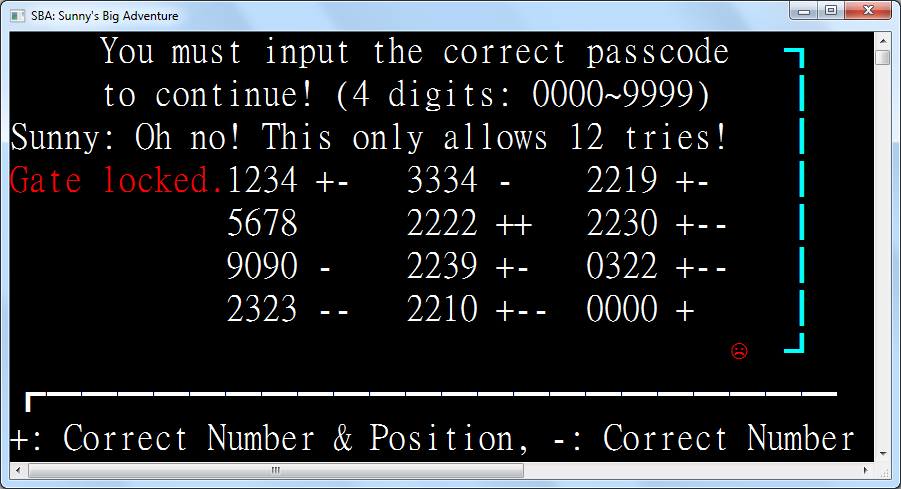
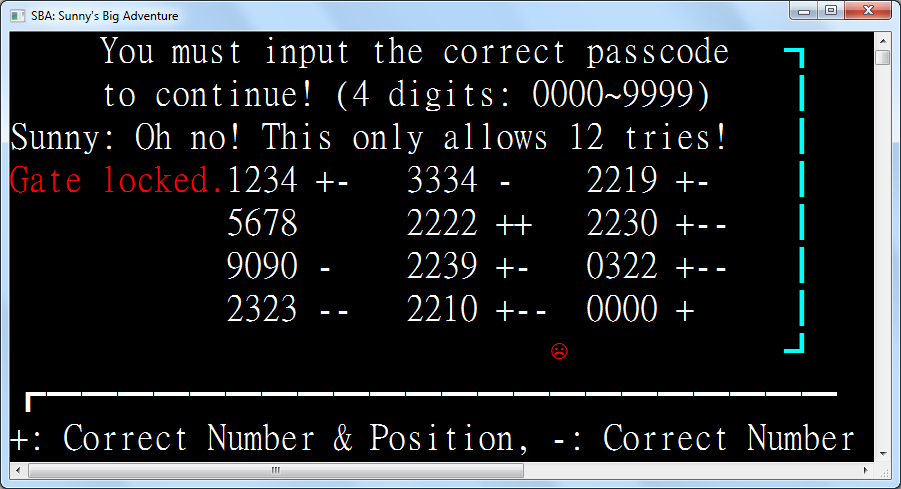
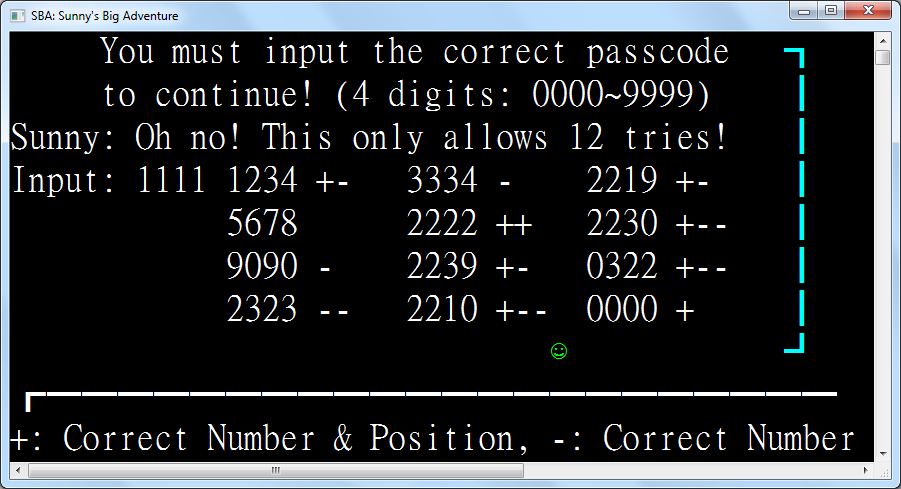
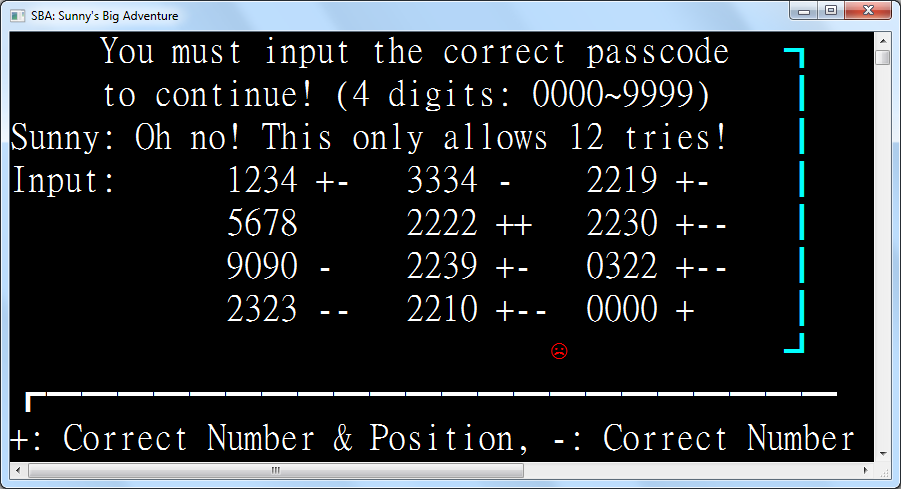
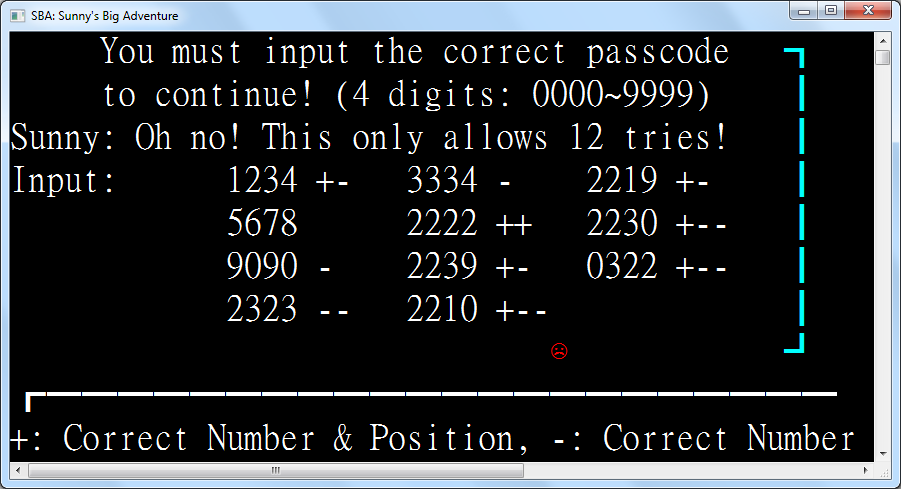
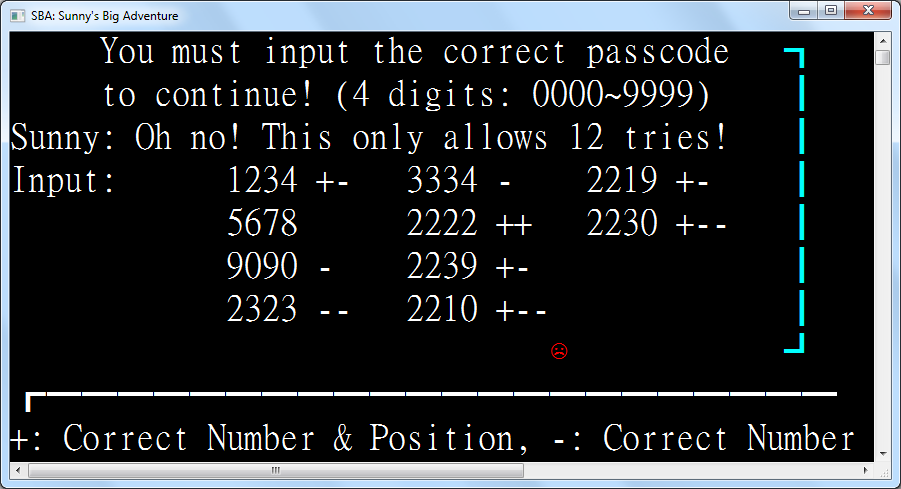
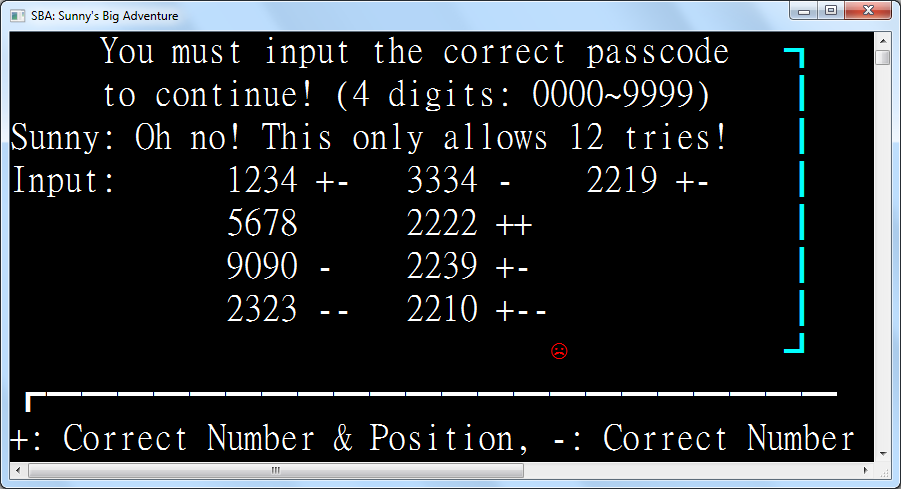
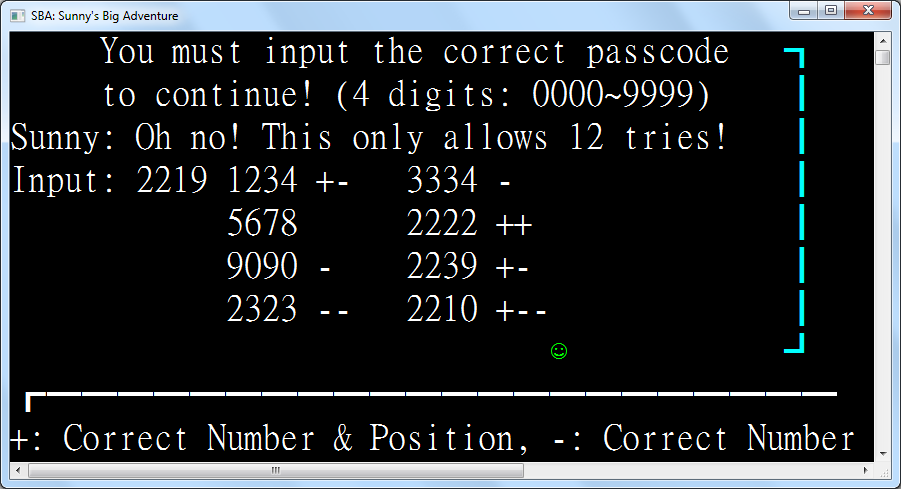
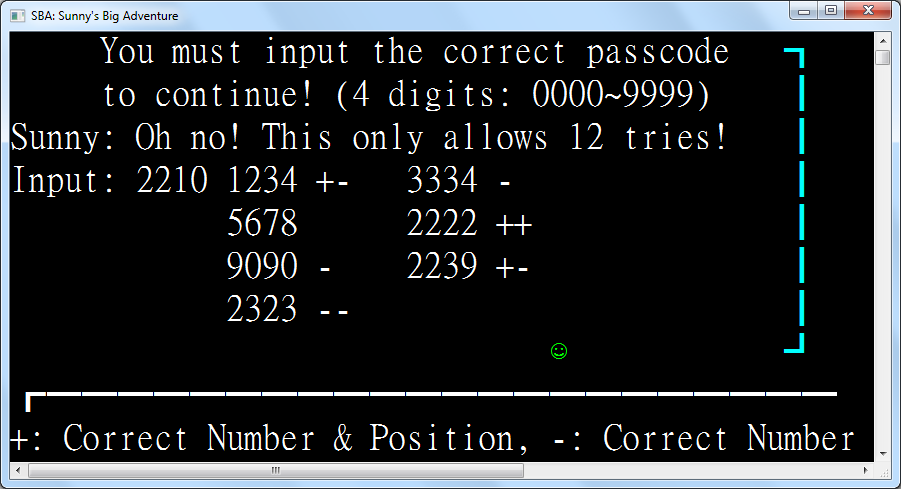
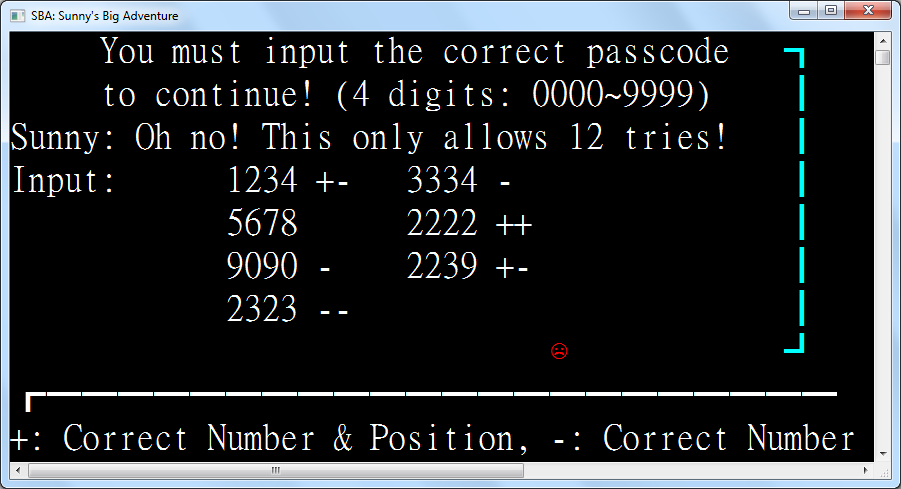
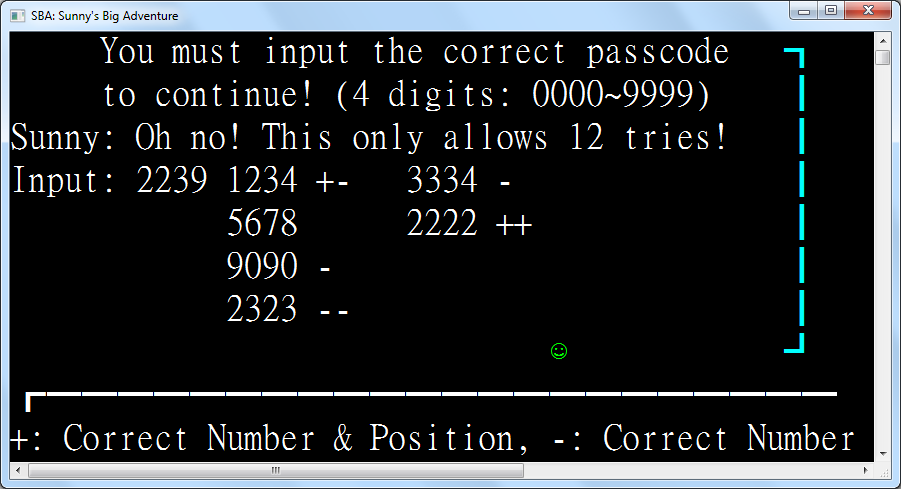
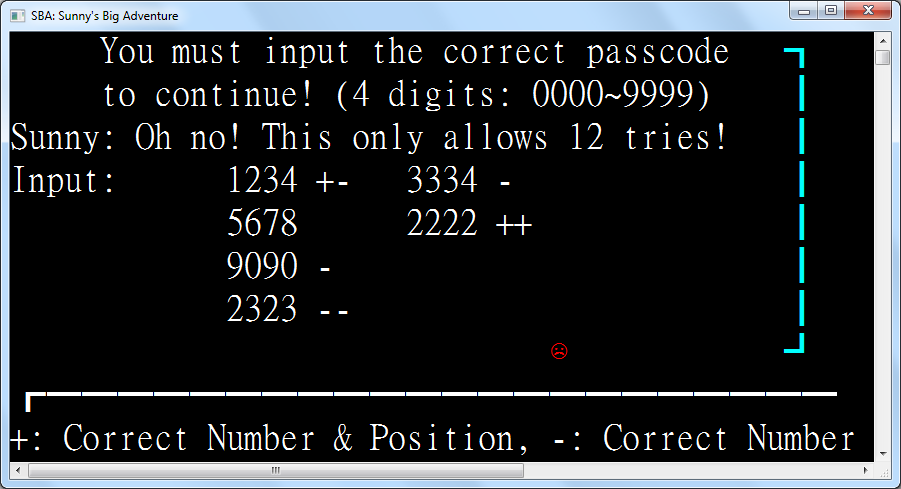
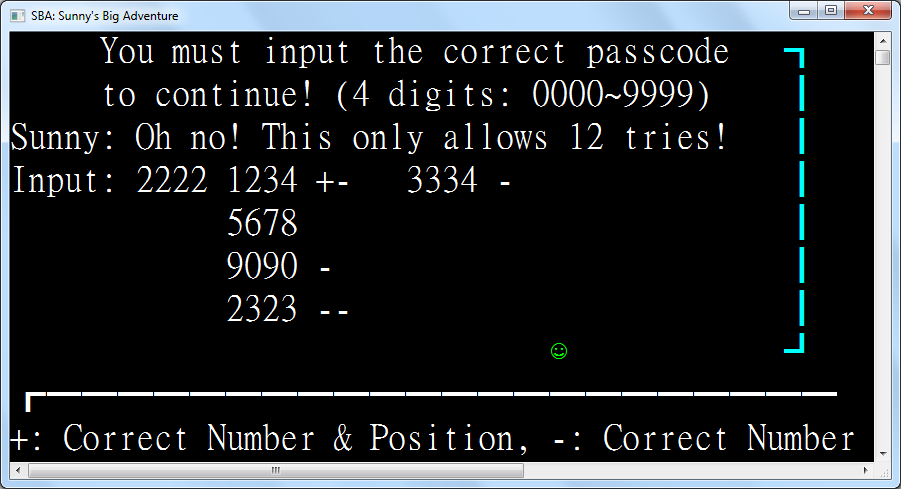
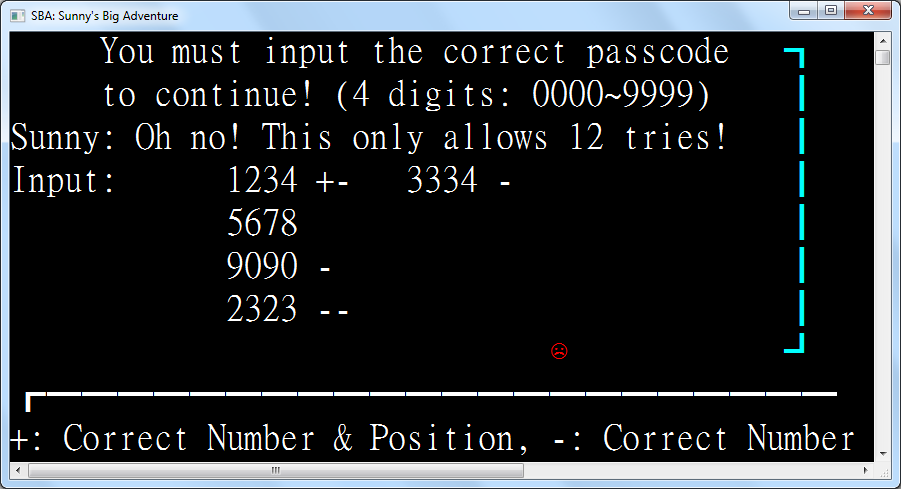
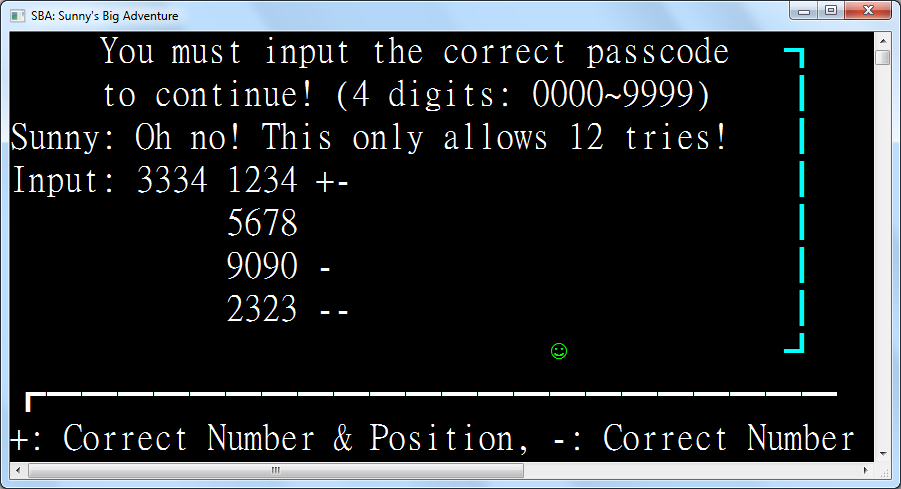
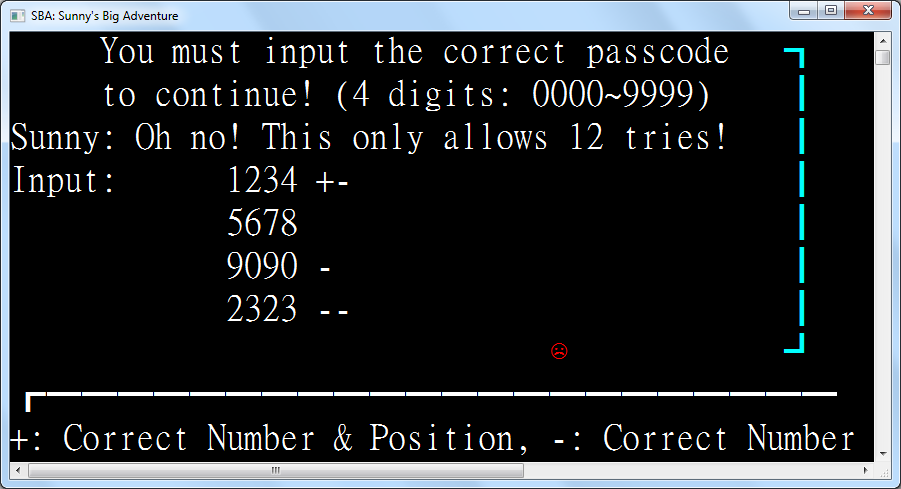
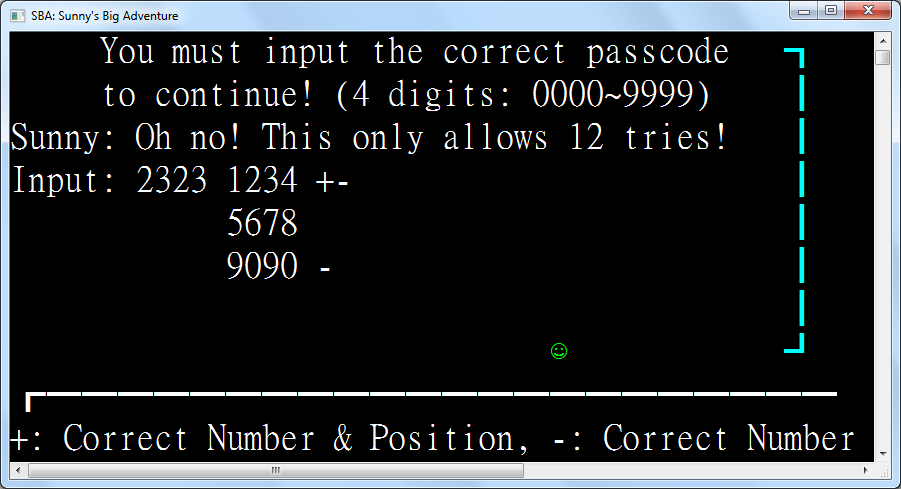
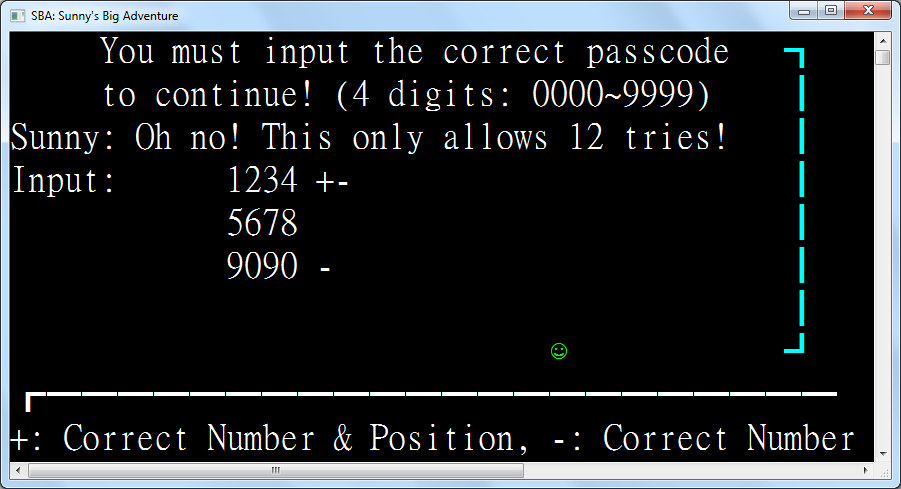
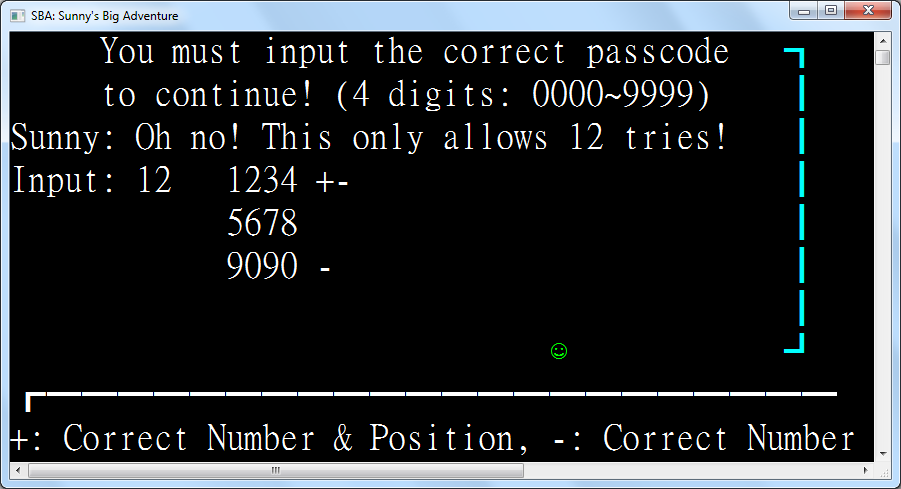
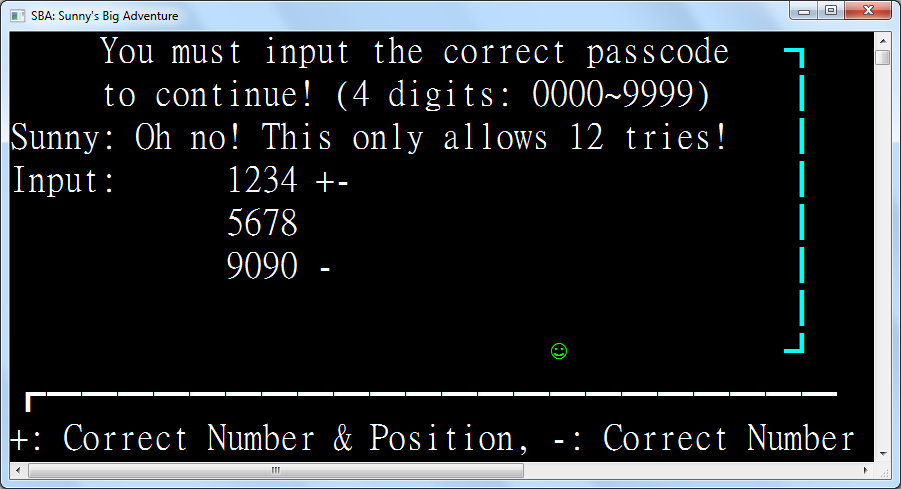
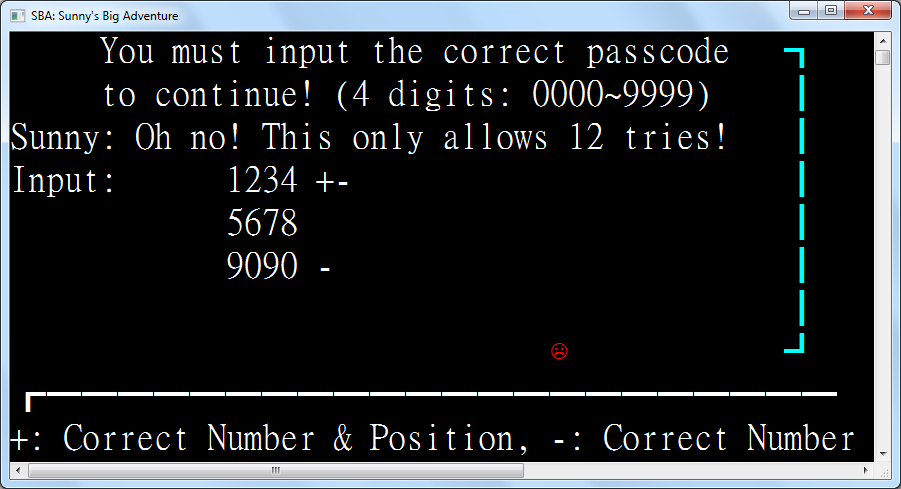
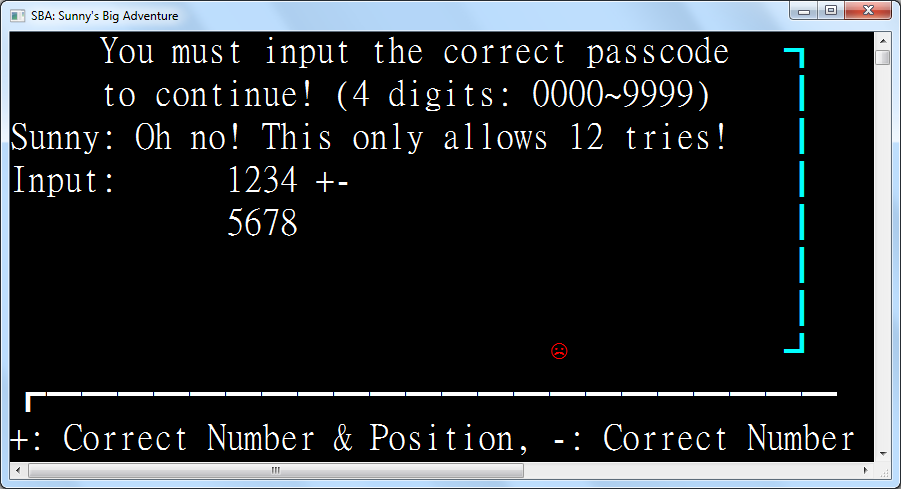
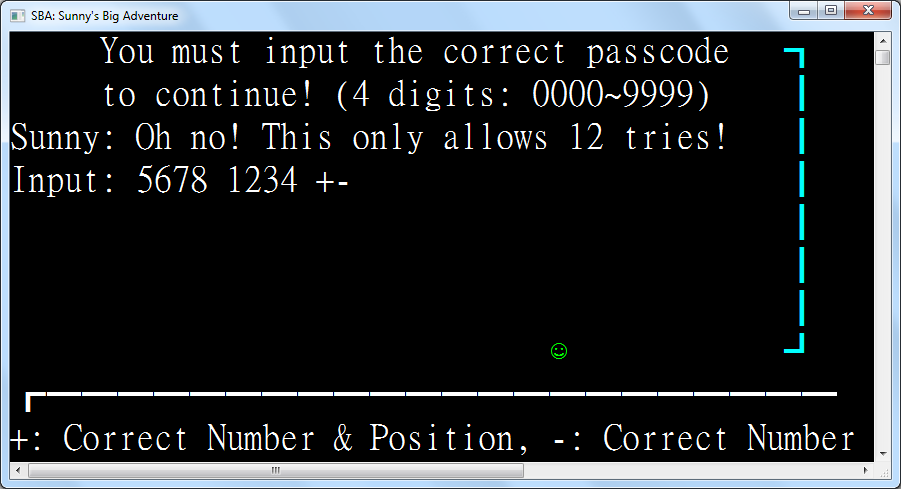
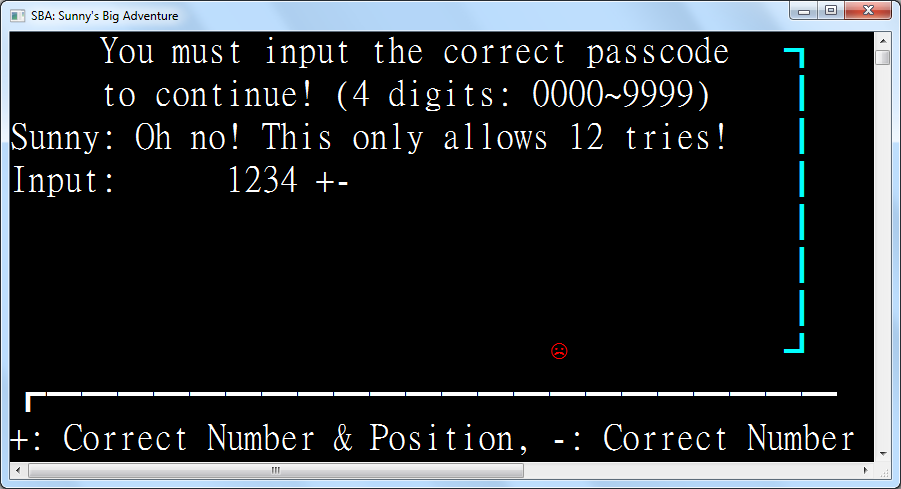
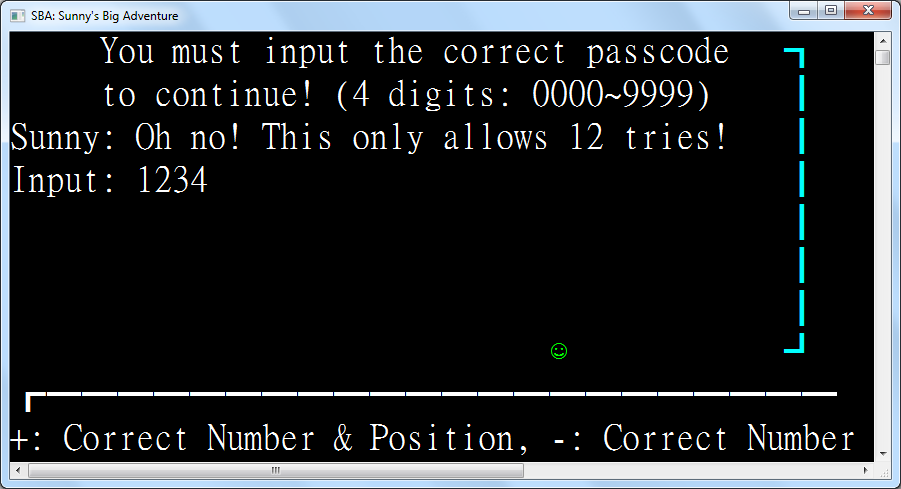
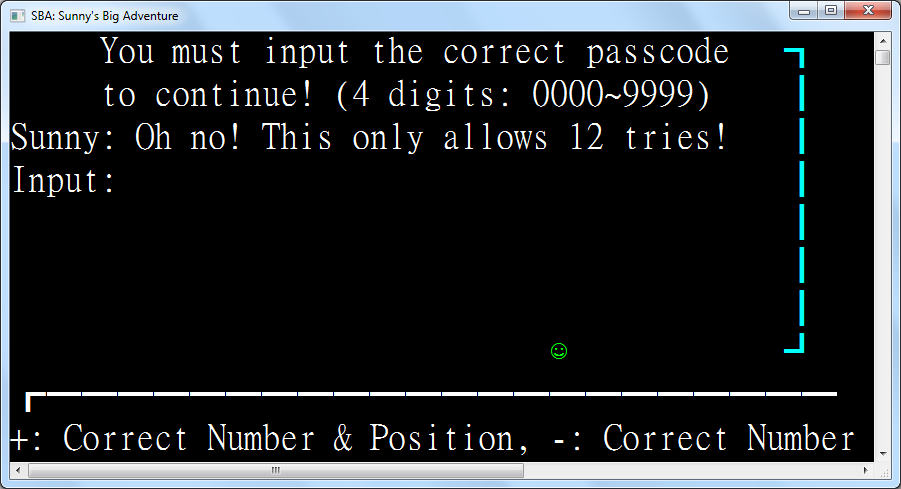
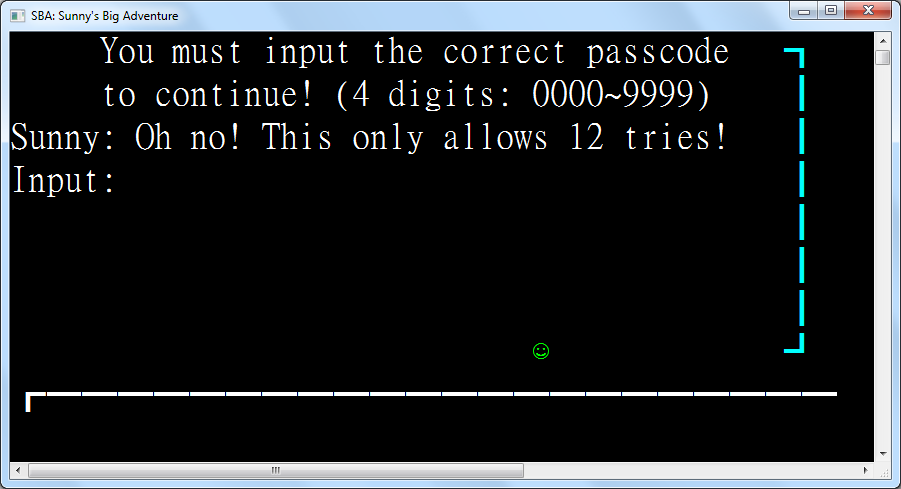
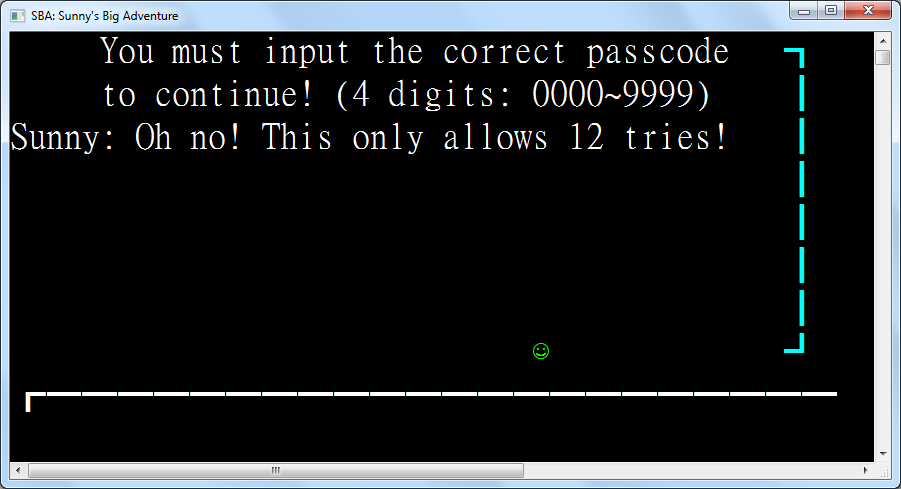
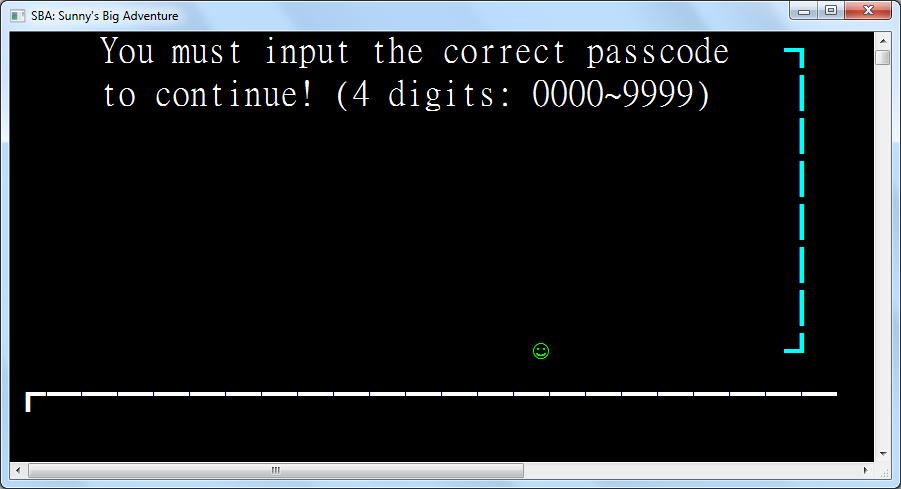
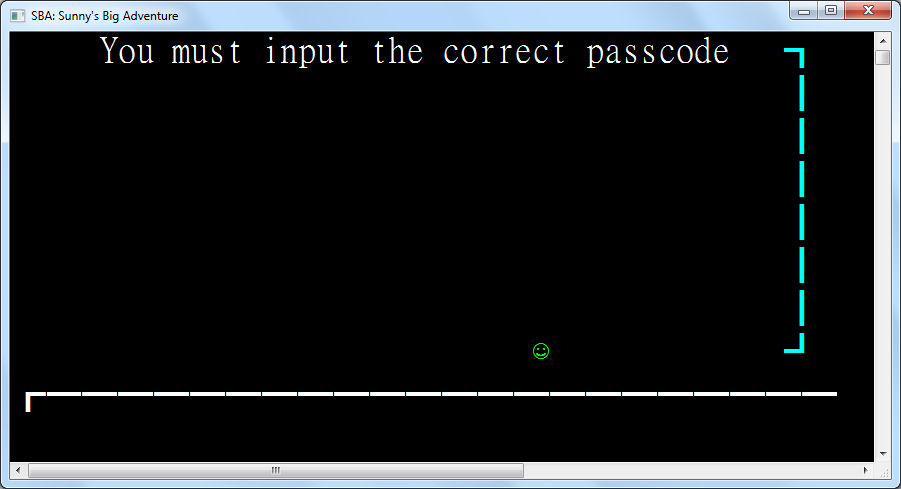
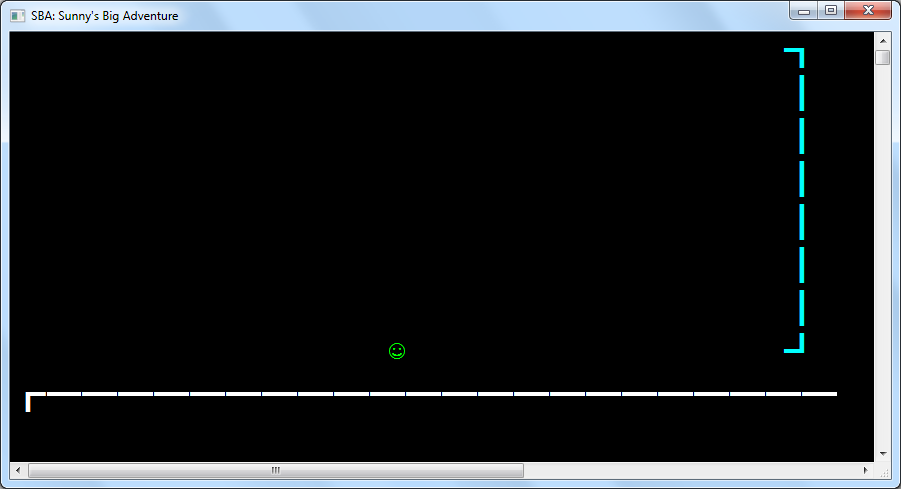
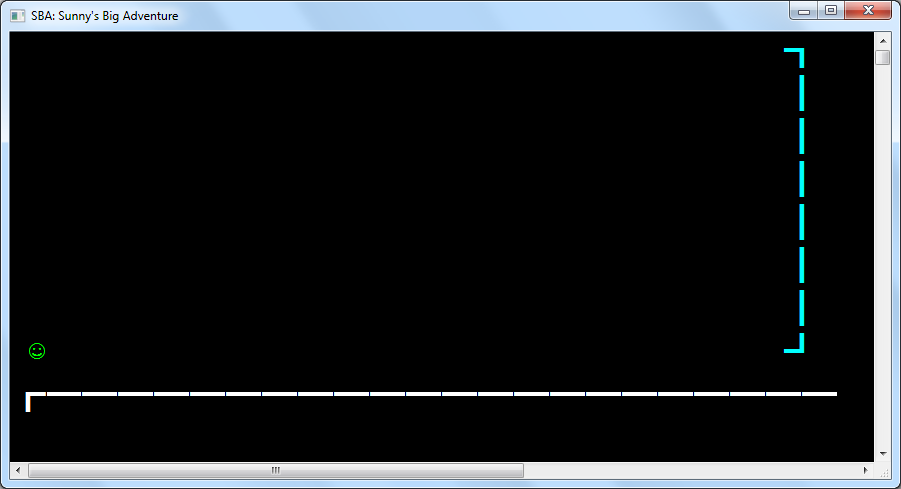
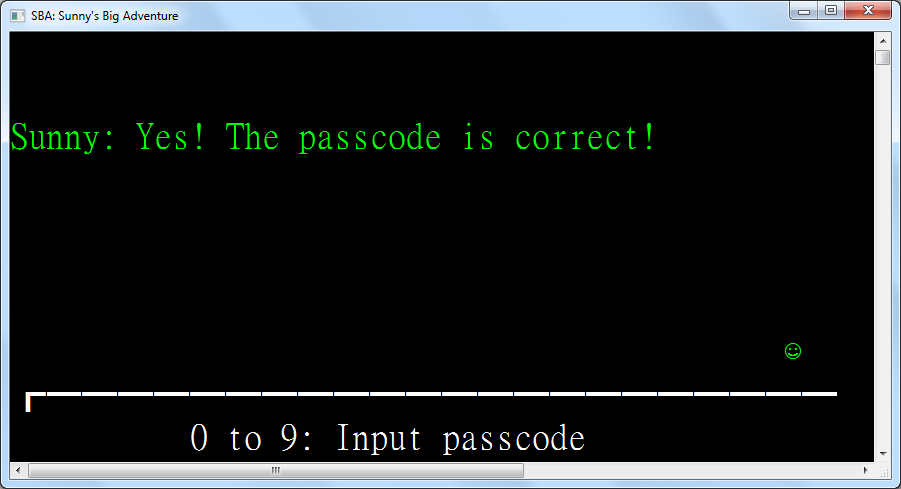
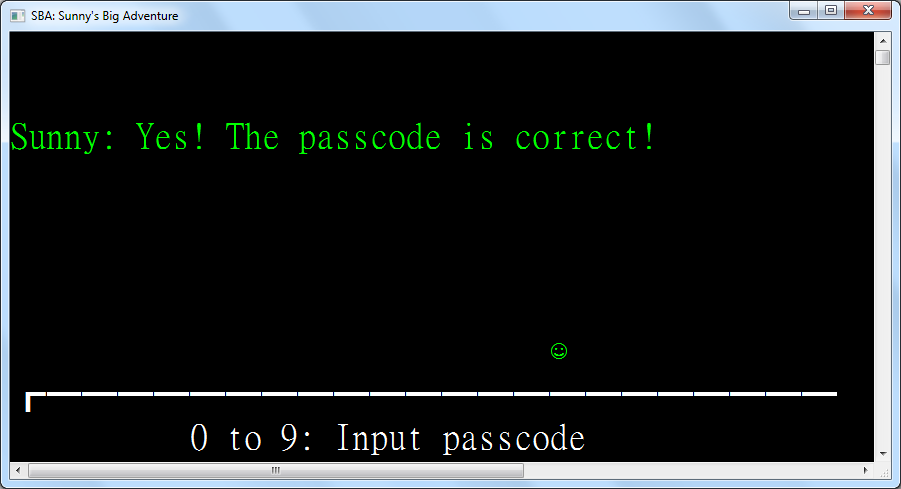
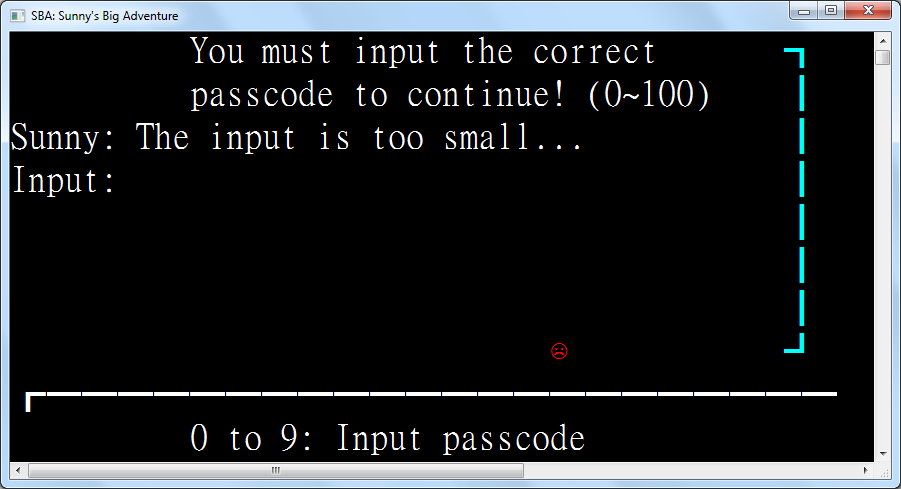
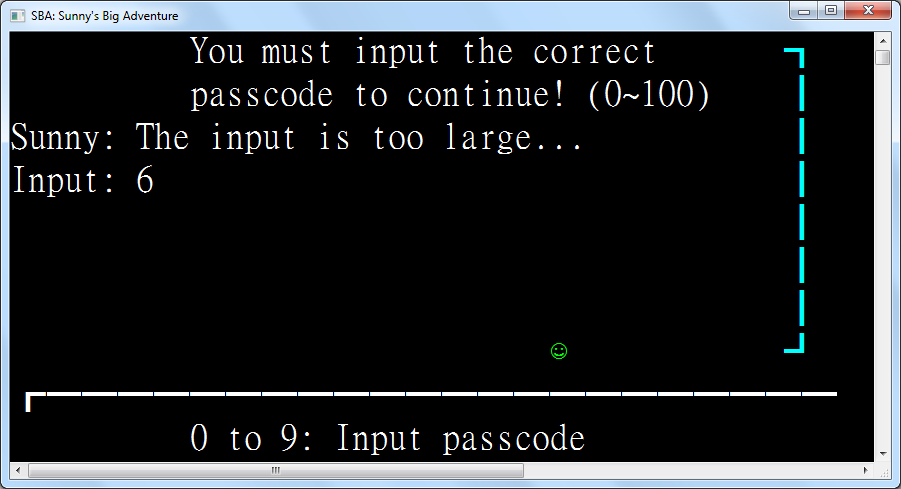
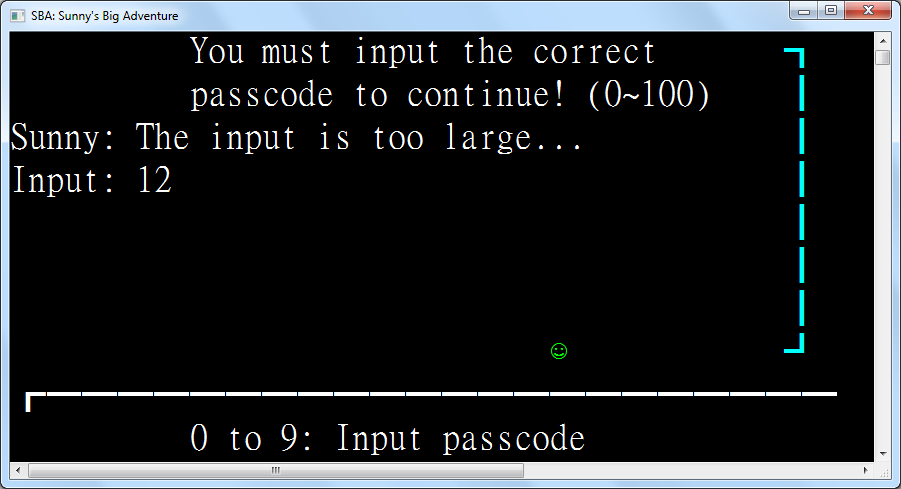
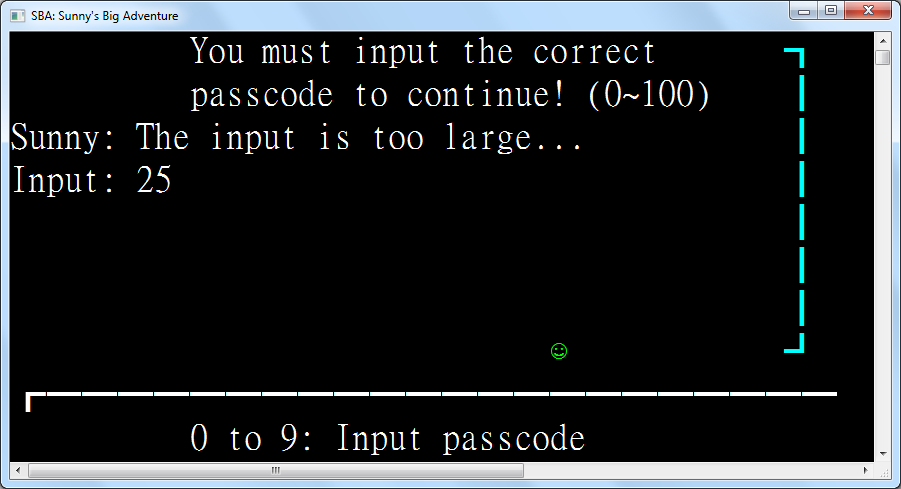
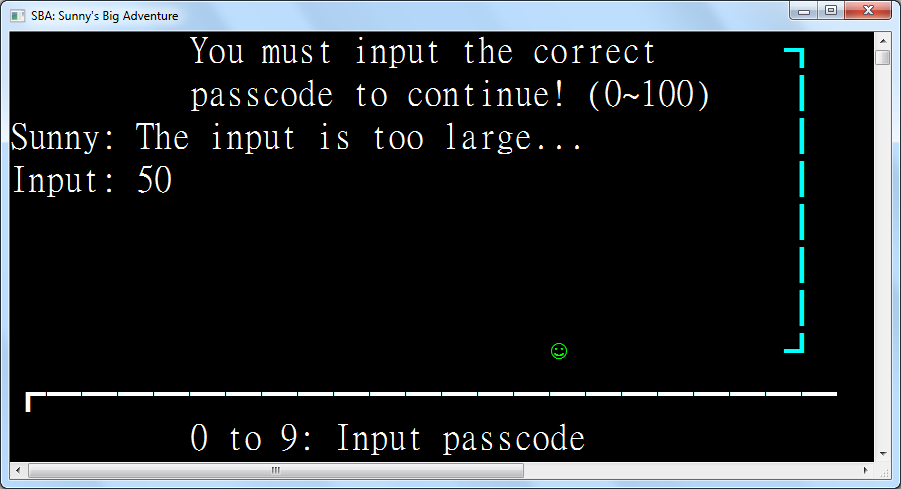
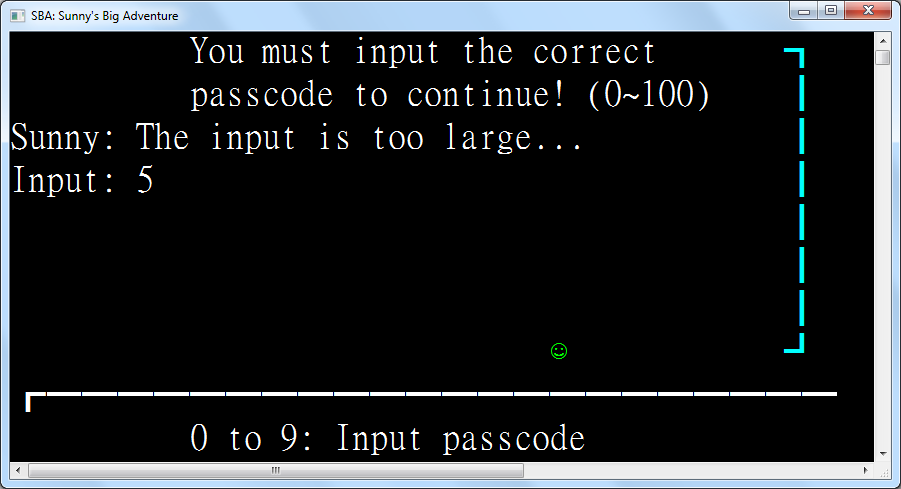
This is the first region in the system. This part will explain the details inside this region.

Process of the Procedure:



1) Input '1' in the menu so as to select the game Peaceful War. Once the





**Chapter 4: System Testing and Evaluation**

**4.1: The User Interface**

In this chapter, testing and evaluation of different aspects in the system will be shown.

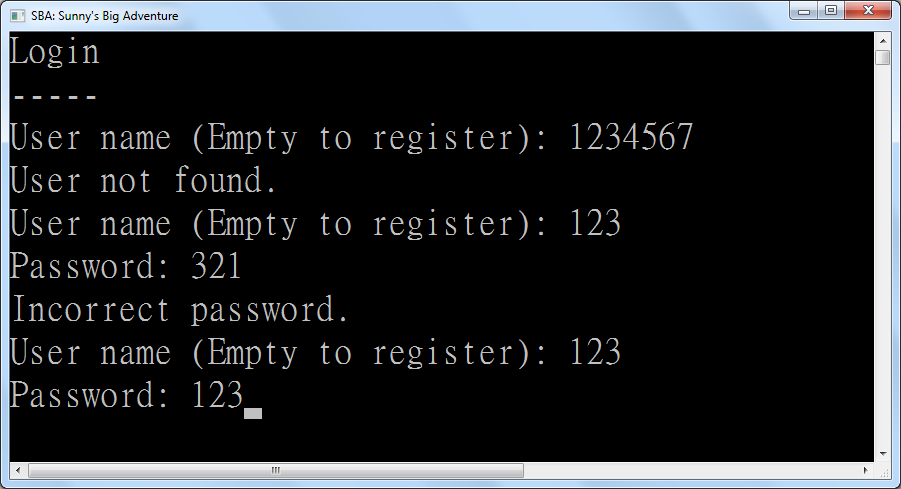
The following will be shown:

1. Test Cases Design and Test Results
2. Further Improvements to the system

**4.2: Test Cases Design and Results**

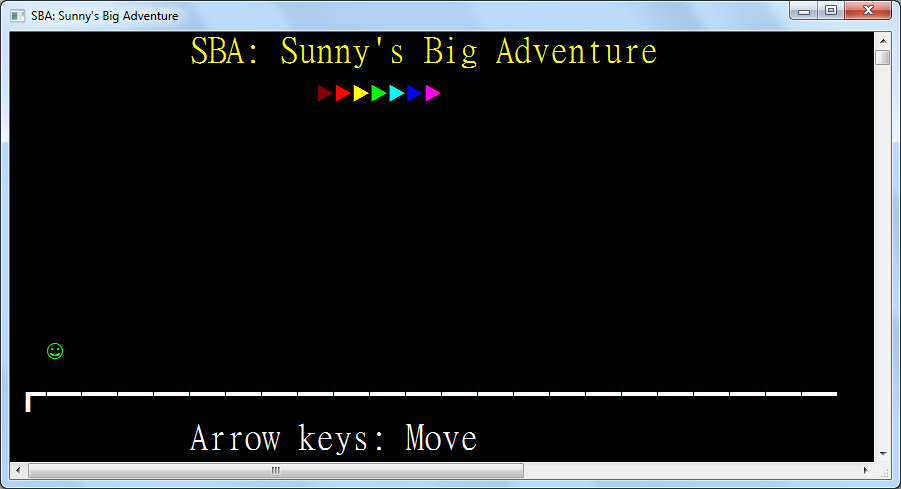
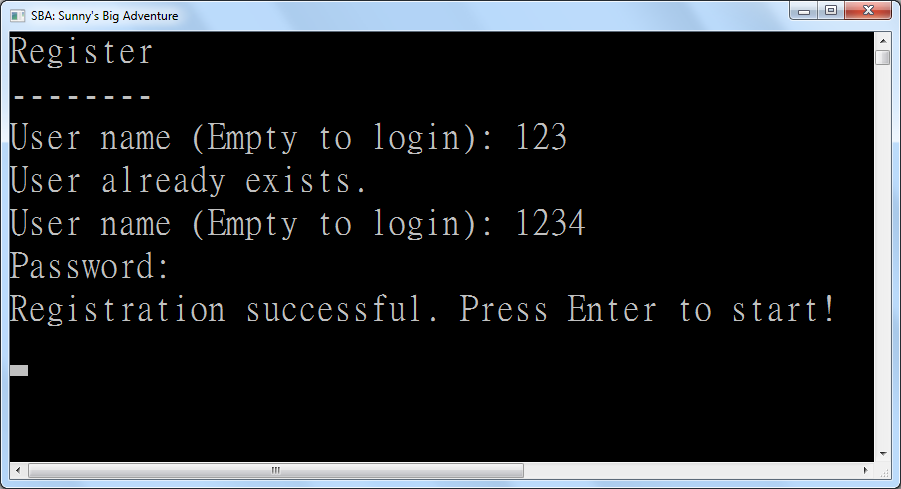
In this sub-chapter, testing is divided for each subprogram where test cases will first be described, then the expected outcome and the test results, and finally whether it matches the expected outcome and problems encountered if any. The details of the testing are as follows.

**4.2.1: Tests for Login**



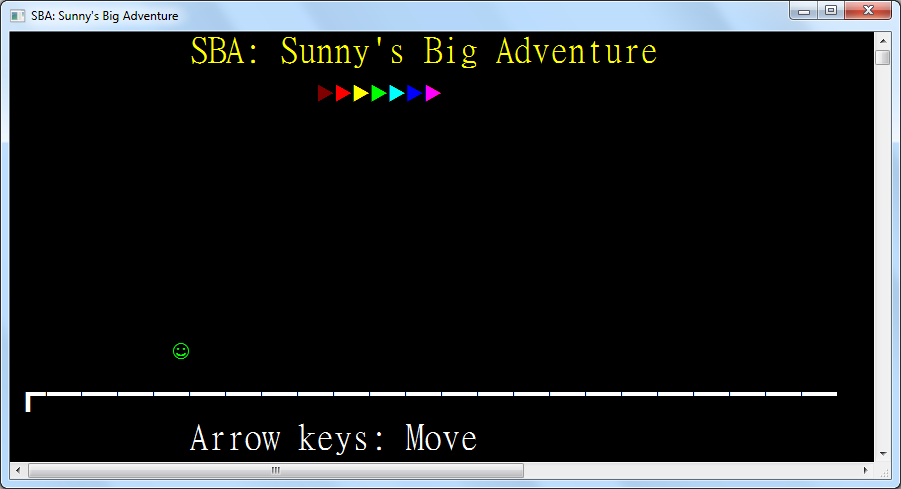
|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Test Type | Expected Outcome | Results |
| Name: 1234567 | Invalid Input | Display “User not found” | As expected |
| Name: 123  Pass: 321 | Invalid Input | Display “Incorrect password” | As expected |
| Name: 123  Pass: 123 | Valid Input | Successfully login | As expected |
| Name: 123  Empty Pass | Invalid Input | Display “Incorrect password” | As expected |
| Empty Name | Valid Input | Transfer to Registration Page | As expected |

**4.2.2: Tests for Registration**



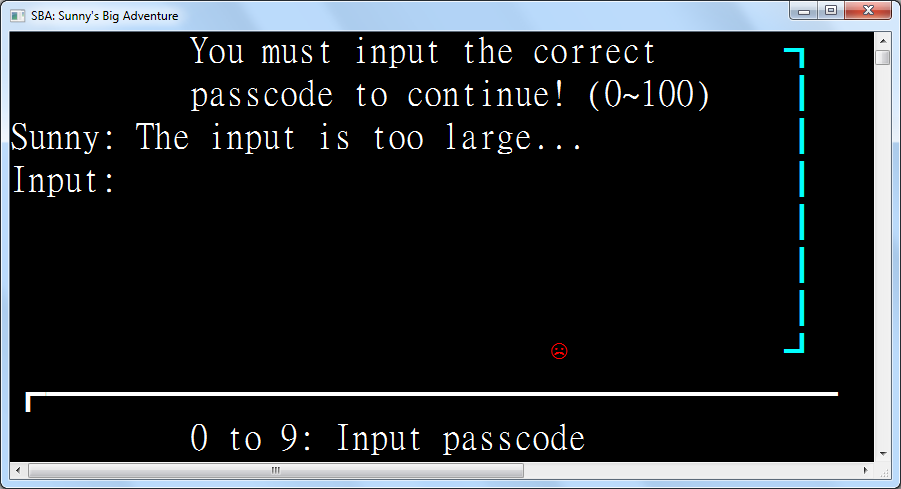
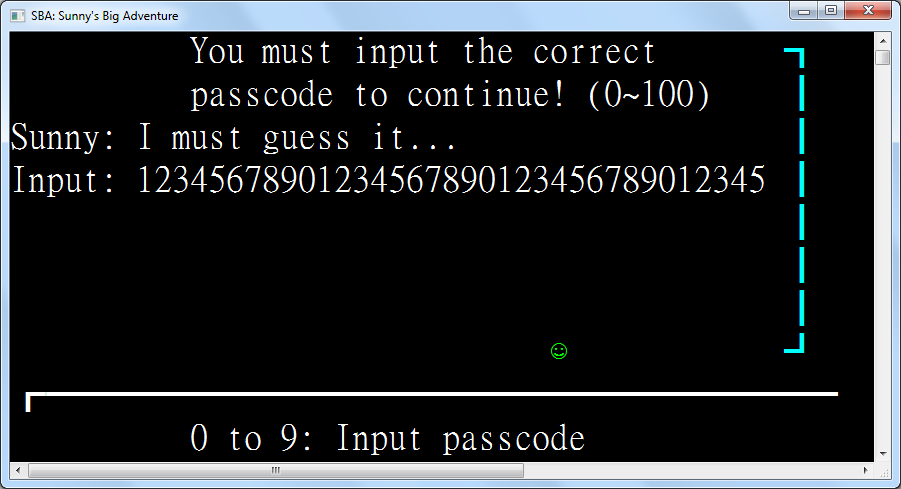
|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Test Type | Expected Outcome | Results |
| Name: 123 | Invalid Input | Display “User already exists” | As expected |
| Name: 1234  Empty PW | Valid Input | Successfully register | As expected |
| Name: 12345  Pass: 12345 | Valid Input | Successfully register | As expected |
| Name: ~!@#$%^&\*()\_+  Pass: +\_)(\*&^%$#@!~ | Valid Input | Successfully register | As expected |
| Empty Name | Valid Input | Transfer to Login Page | As expected |

**4.2.3: Tests for Title Screen**



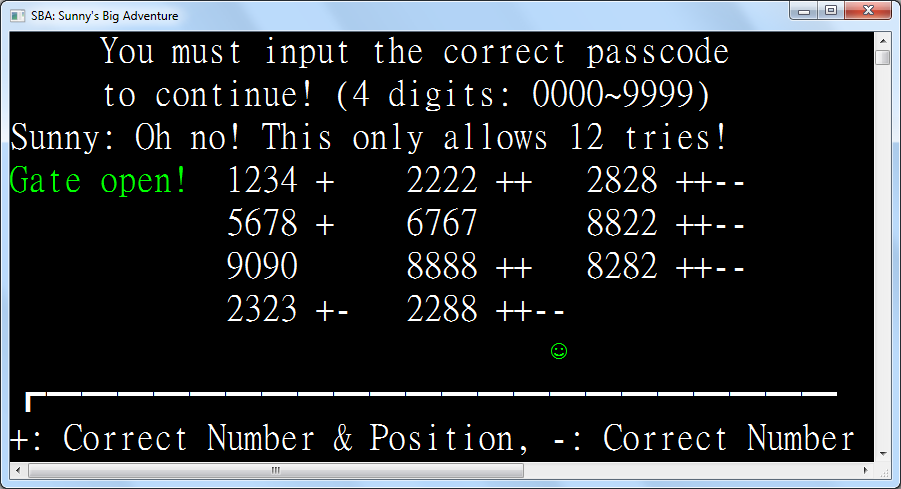
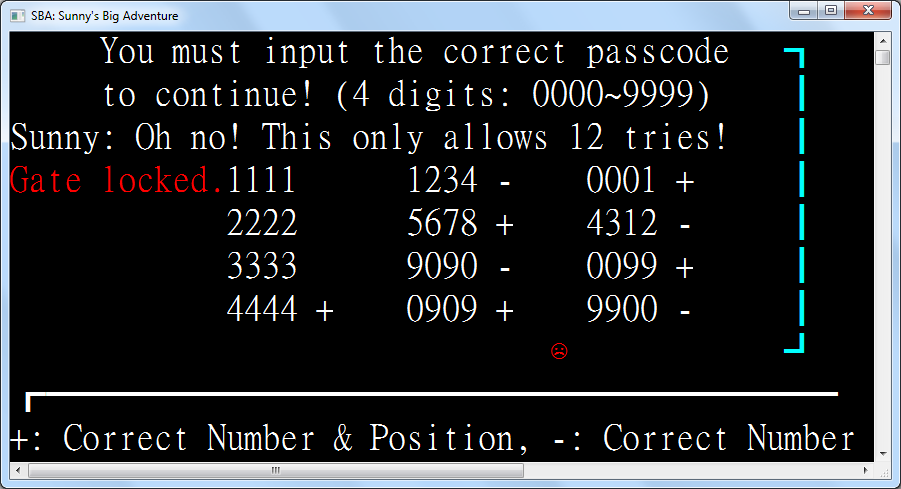
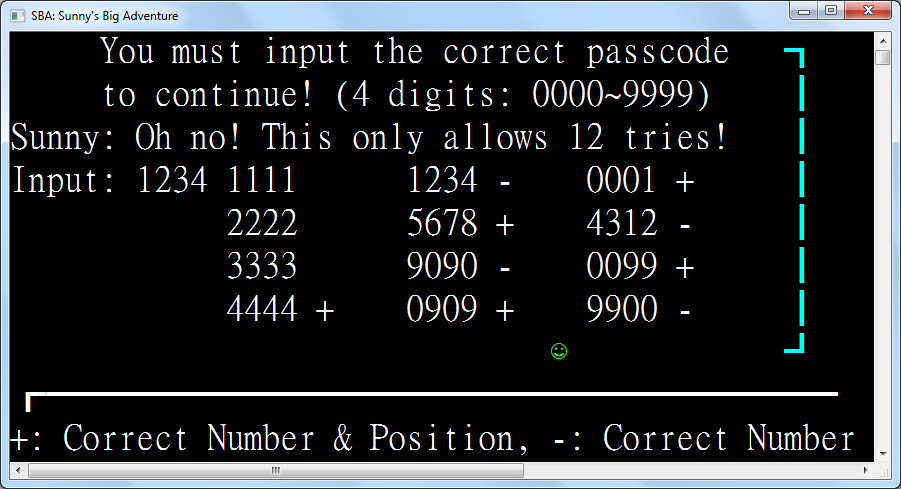
|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Test Type | Expected Outcome | Results |
| Left arrow | Valid Input | Avatar moves left | As expected |
| Right arrow | Valid Input | Avatar moves right | As expected |
| Up arrow | Valid Input | Avatar jumps and is pulled down by gravity | As expected |
| Down arrow when avatar is in air | Valid Input | Avatar drops faster | As expected |
| The 1 key | Invalid Input | Nothing happens | As expected |
| Backspace | Invalid Input | Nothing happens | As expected |
| Escape | Invalid Input | Nothing happens | As expected |

**4.2.4: Tests for Number Guess**



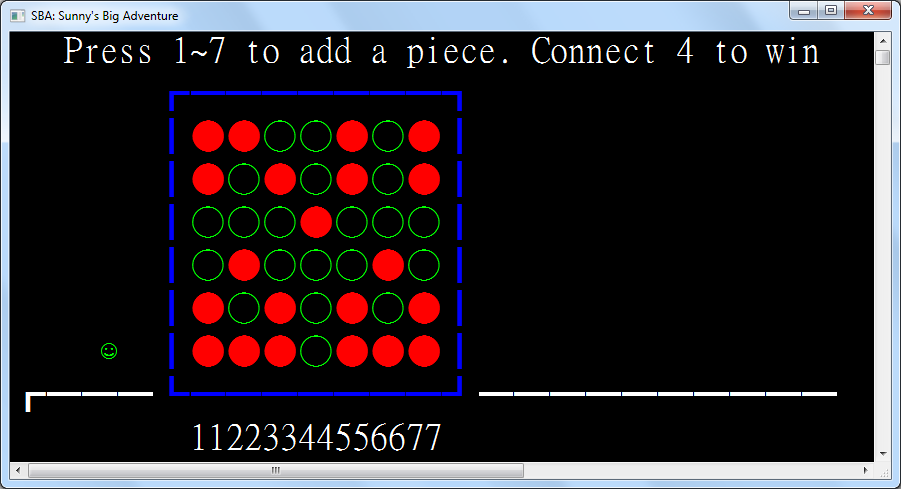
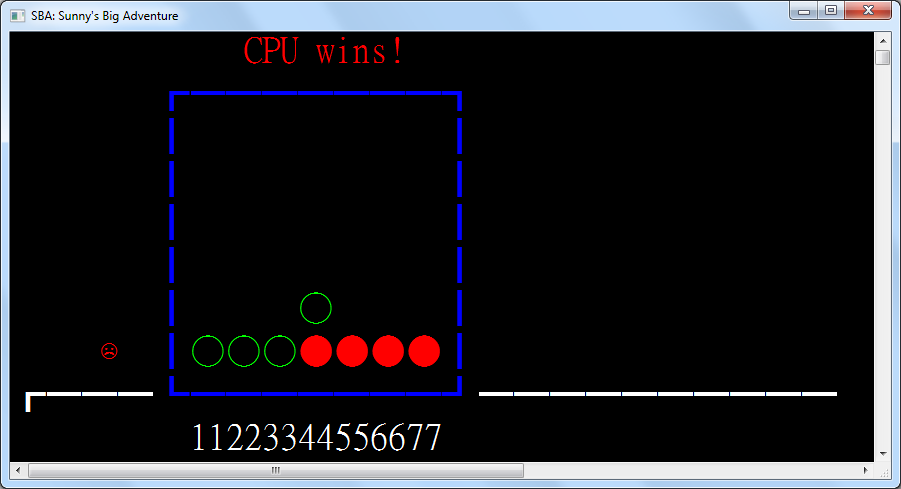
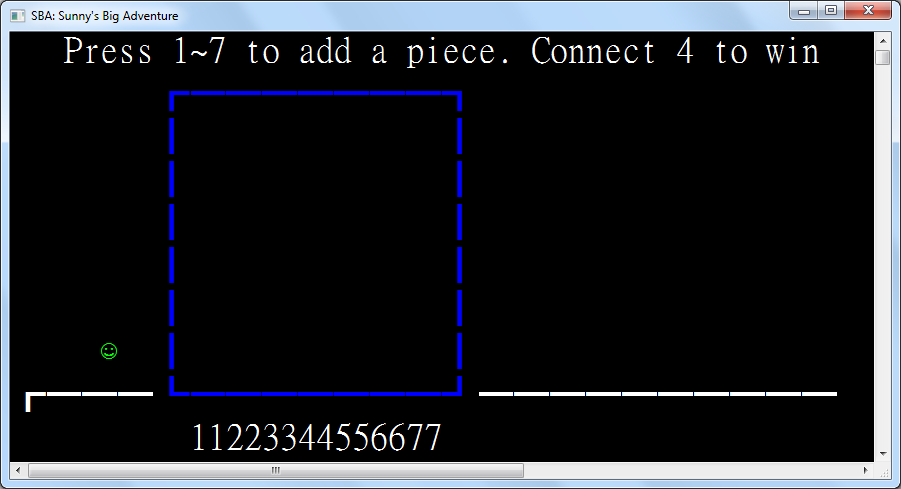
|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Test Type | Expected Outcome | Results |
| 12345678901234567890123456789012345 | Valid Input | Display “Input too large” | As expected |
| 0 | Valid Input | Display “Input too small” | As expected |
| -1 | Invalid Input | The minus sign is ignored | As expected |
| abc | Invalid Input | Keys are ignored | As expected |
| 100 | Valid Input | Display “Input too large” | As expected |
| Empty | Extreme input | Nothing happens | “Input too large” is displayed |
| Backspace when input is present | Valid Input | Deletes the last inputted character if any | As expected |
| Backspace when input is empty | Extreme input | Nothing happens | As expected |
| Escape key | Invalid Input | Nothing happens | As expected |

**4.2.5: Tests for Bulls and Cows**



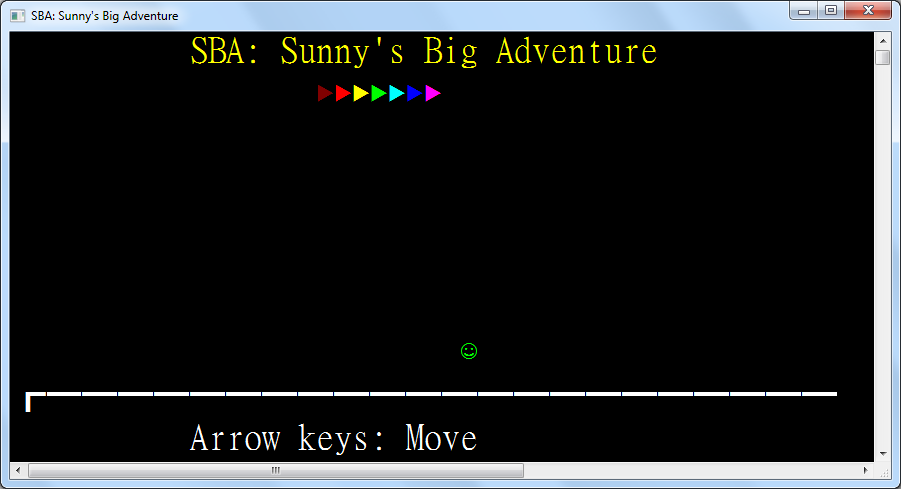
|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Test Type | Expected Outcome | Results |
| 1111 | Extreme Input | Display +s and –s depending on the digits | As expected |
| 1234 | Valid Input | Display +s and –s depending on the digits | As expected |
| 5678 | Valid Input | Display +s and –s depending on the digits | As expected |
| 9090 | Valid Input | Display +s and –s depending on the digits | As expected |
| 0000 | Extreme Input | Display +s and –s depending on the digits | As expected |
| Empty | Invalid Input | Input is ignored and reset | As expected |
| 1 | Invalid Input | Input is ignored and reset | As expected |
| 123 | Invalid Input | Input is ignored and reset | As expected |
| -1234 | Invalid Input | Minus sign is ignored | As expected |
| 12345678 | Invalid Input | Digits after the 4th one are ignored | As expected |
| Backspace when input is present | Valid Input | Deletes the last inputted character if any | As expected |
| Backspace when input is empty | Extreme input | Nothing happens | As expected |
| Escape key | Invalid Input | Nothing happens | As expected |

**4.2.6: Tests for Connect Four**



|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Test Type | Expected Outcome | Results |
| Key 1 | Extreme Input | Player puts a piece at position 1 | As expected |
| Key 4 | Valid Input | Player puts a piece at position 4 | As expected |
| Key 7 | Extreme Input | Player puts a piece at position 7 | As expected |
| Key 8 | Invalid Input | Nothing happens | As expected |
| Key 0 | Invalid Input | Nothing happens | As expected |
| Backspace key | Invalid Input | Nothing happens | As expected |
| Escape key | Invalid Input | Nothing happens | As expected |
| Player places 4 pieces horizontally | Valid Input | Player wins | As expected |
| CPU places 4 pieces horizontally | Valid Input | CPU wins | As expected |
| Player places 4 pieces vertically | Valid Input | Player wins | As expected |
| CPU places 4 pieces vertically | Valid Input | CPU wins | As expected |
| Player places 4 pieces diagonally | Valid Input | Player wins | As expected |
| CPU places 4 pieces diagonally | Valid Input | CPU wins | As expected |
| Player places 3 pieces horizontally | Valid Input | CPU places piece to stop the player from winning | As expected |
| Player places 3 pieces vertically | Valid Input | CPU places piece to stop the player from winning | As expected |
| Player places 3 pieces diagonally | Valid Input | CPU places piece to stop the player from winning | As expected |
| Player places piece with no immediate chance of winning | Valid Input | CPU places piece randomly | As expected |

**4.2.7: Tests for Winning Screen**



|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Test Type | Expected Outcome | Results |
| Left arrow | Valid Input | Avatar moves left | As expected |
| Right arrow | Valid Input | Avatar moves right | As expected |
| Up arrow | Valid Input | Avatar jumps and is pulled down by gravity | As expected |
| Down arrow when avatar is in air | Valid Input | Avatar drops faster | As expected |
| Enter | Valid Input | Player is taken back to the Title Screen | As expected |
| The 1 key | Invalid Input | Nothing happens | As expected |
| Backspace | Invalid Input | Nothing happens | As expected |
| Escape | Invalid Input | Nothing happens | As expected |
| Player jumps to a firework | Valid input | Player is blocked from overlapping with the firework | As expected |

**Chapter 5:**