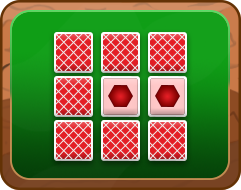
CST8219 – C++ Programming

Final Project due April 16, 11:59 pm

For this project, you will create a memory game using various shapes. The user interface should show 20 cards, in 5 rows and 4 columns. The user can click on a card and see what is on the other side. The user then clicks on a second card to see what shape is under that card. If the two cards match, then they are removed from the interface and some happy sound file should be played. Otherwise, they should be flipped over and the user can continue clicking on cards, and a sad sound file should be played. It should look something like this:



You must use the C++ Nana library for drawing pictures, playing sounds, and listening for mouse input from the user. Using any other software will be assigned a mark of 0 automatically.

Requirements:

1. The memory game has 20 cards, arranged in 5 rows and 4 columns. At the start of the game, the cards all show the same back of the card. The other side of the card can be any picture you want.
2. Clicking on the mouse on a card turns it over to show the shape underneath. The user can then click on a second card to reveal its hidden shape. If the shapes are the same, then the two cards are removed and a happy sound clip is played, otherwise they are turned over again to show the back of the card and a sad sound clip is played. There should be a 3 second pause to show the user what they clicked on before removing them or turning them back over.
3. When all of the cards are removed from the game, there should be a final message showing how many turns it took to remove all of the cards, and there should be some kind of celebration sound clip. There should be a button to start a new game, which picks all new cards and picks a new random placement. Nana uses the label class for showing HTML code of an <img src=”..” tag> which shows a PNG or JPG file that says “Do you want to play again”.
4. All of the classes you write should be enclosed in a namespace called FinalProject. You must have a Game class which keeps track of the score, a Card class which keeps track of which picture to show, along with the row and column where it should be drawn. A card can have 3 states: down, up, or removed. You should then also have an Image class which represents which Image to show. In order to save memory, you should only load images once in memory. All cards that are face down should all be pointing to the same image to show on the top of the card. When two cards are showing the same image when turned over (two cats, or triangles, ..) should both be pointing to the same image object. As a rule, 20 cards should only have 10 images (2 cards share 1 image), and 1 top of card image for when all cards are face down.

Hints: You should use a std::vector object to store the Cards. The Cards should know their row, column, and hidden shape. You should implement the comparison operator ( ***operator==( Card &other)*** ) so that it only compares shapes and not position. When y

Since there are 20 cards, but two of them repeat, there are really only 10 unique cards. To shuffle the cards, use the std::random\_shuffle function to randomize the vector. Then you can position the cards in order from row, column (0, 0) to (5, 4).

1. When the game is over, the game should show you how many mouse clicks it took to end the game, and then as the user to enter their name. Then, you should save the name and number of clicks to a file. You should also then show a high score table of the 5 best scores (lowest number of clicks to finish).

Marks: Total of 12 marks

GUI Window shows 20 cards organized in 5 rows of 4 cards +1

Cards all have the same image showing at the start of the game +1

Clicking on a card changes the image shown on the card +1

The game can compare two cards. If they match, then they are removed. If they don’t match, then they show the back of the card again +2

The game plays a sound indicating if the two cards are a match or not +1

The game can detect when the game is over and all cards are gone +1

The game plays a sound when the game is over +1

The game displays the number of turns taken to end the game +1

At the end of the game, the user is presented the option to quit or play again +1

The game asks for the person’s name and saves the score in a file. +1

The game shows a list of the 5 best scores and player’s name. +1

Deductions:

-5% for memory leaks (using new without delete)