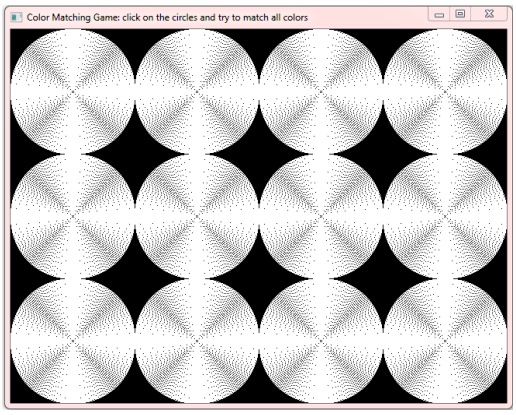
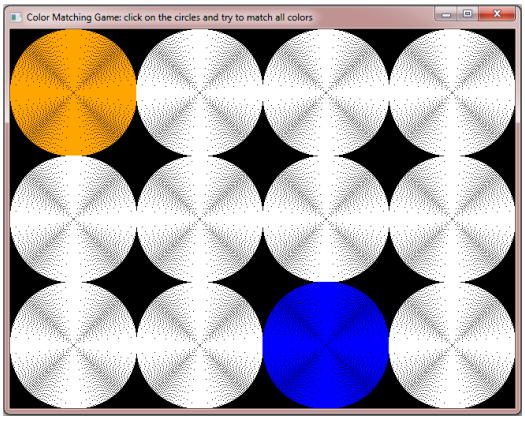
CS250 Assignment 5
Color Matching Game

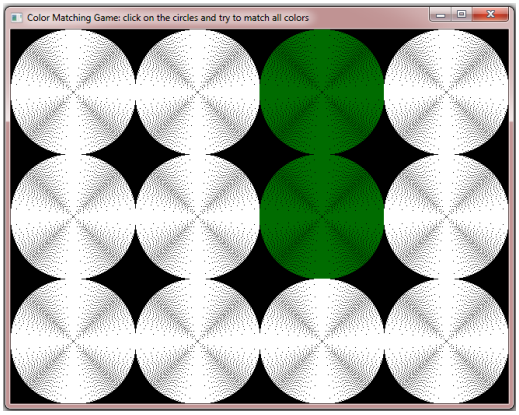
Date assigned: Monday, March 17, 2014
Date due: Monday, April 9, 2014
Points: 40

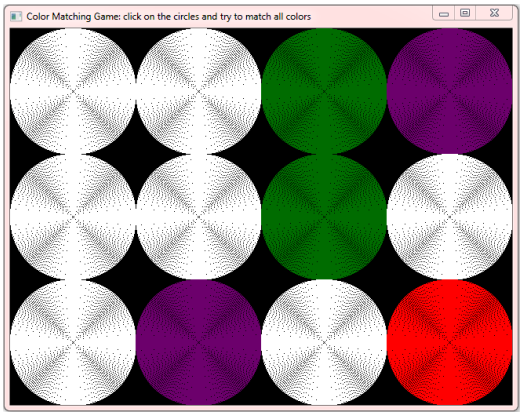
For this assignment you will be using DarkGDK, the Circle Animation project, and the Game2DUtilities that you created for your last assignment to implement a color matching game.

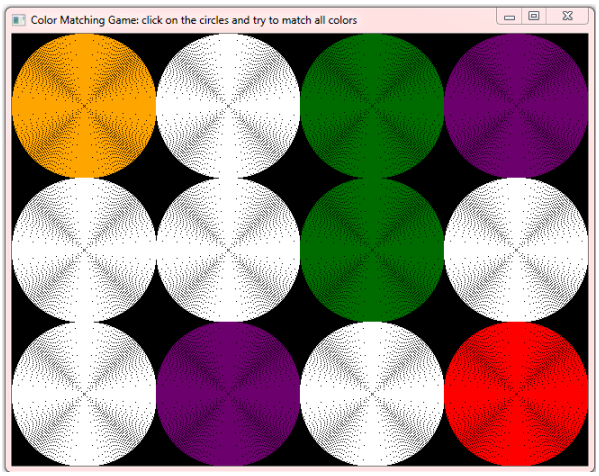
When the game first starts, the user is presented with a number of white circles filling the screen. The user then uses the mouse to uncover the circles two at a time. If the two circles match in color, then they remain uncovered. However, if the circles do not match, then they both are changed back to white. You can see screen shots of the game in progress below.

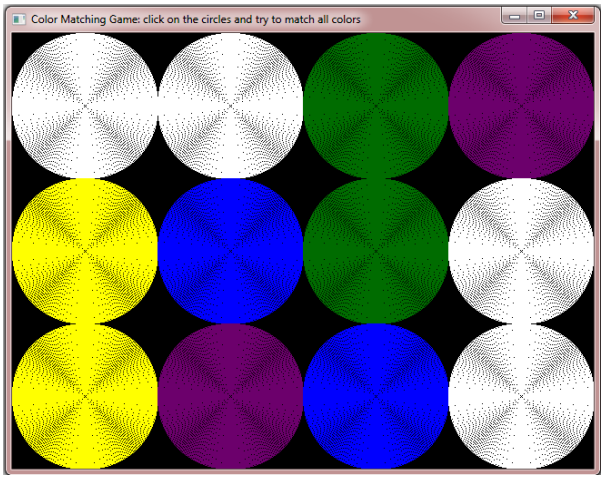


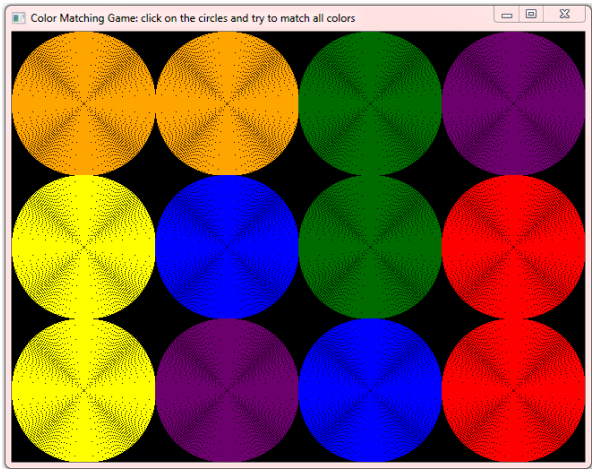




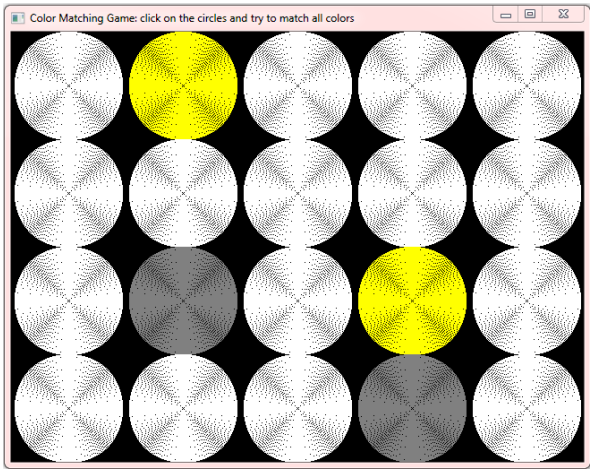








To allow for players of multiple abilities, you need to make it easy to change the number of colors that they are playing with. In your main game driver, create a constant called NUM_COLORS. For example, if this constant is set to 6, then you will display the screen below. If the constant is set to 4, then you will show the circles in a 2 rows and 4 columns formation. Further, the circles must completely cover the screen. So, the circles in games that have more colors (hence more circles) will have a smaller radius than the circles in games that have fewer colors. Below is an example of a game with 10 colors (20 circles)



In order to implement the game, you must do the following:

1) Add a Color class to Game2DUtilities: Create two new files in your Game2DUtilities called **Color.h** and **Color.cpp**. This class will contain information about the colors that will be used in the game, and functionality to return the RGB of the colors. In the CS 250 public folder you will find the content for those files. Copy the content of the files on Turing to the files in your project and make sure that your Game2DUtilities builds without errors.

2) Add a ColorSet class to Game2DUtilities: Create two new files in your Game2DUtilities called **ColorSet.h** and **ColorSet.cpp**. This class will be used to maintain the colors that are used in the game. The interface for the ColorSet class is provided for you on Turing. You are to implement ColorSet.cpp. Make sure that Game2Utilities builds without errors.

3) Modify the Circle class in CircleAnimation: Add a Color data member named **mColor**, which is an object of class Color, to the Circle class. Add functions **setColor** and **getColor** to provide functionality for mColor. Add a function **drawFilled** that will draw a filled circle. The filled circle is accomplished by drawing a circle with radius 1, then again with radius 2, then 3, and so on until it is the size of mRadius (the data member in Circle). Make sure that CircleAnimation builds without errors. Be sure to test the new functions that you added. Try to display multiple circles to the screen that are different colors.

4) Create the ColorMatchingGame project: Add a new project to your solution and call it **ColorMatchingGame**. This project must be of type C++ Project (not a console project). Minimally, this project is to contain a class called **ColorMatchingGame** (**ColorMatchingGame.h** and **ColorMatchingGame.cpp**) and a driver called **ColorMatchingGameDriver.cpp**. Link this project to DarkGDK, Game2DUtilities, and CircleAnimation. Make sure that the ColorMatchingGame project builds without errors.

Steps 1-4 are setting up all of the basic functionality that you need to create the game. The next step is to start developing the game itself. All of the code that is particular to the game will be in the project ColorMatchingGame. You should not need to add any more code to Game2DUilities and CircleAnimation. Also, the ColorMatchingGameDriver should only need to initialize the ColorMatchingGame class, then call a draw and update function similar to what you did in the previous assignment.

Here are the steps that I would advise you to take next:

5) Display the circles to the screen. Don't worry about giving them any color at the moment, just set them to white. You will need to determine the x, y, and radius of each circle based on the number of colors being used. Here are some built-in DarkGDK functions that will help you: dbScreenWidth(), dbScreenHeight().

6) Randomly assign colors to the circles making sure that each color appears on two circles exactly. A function that you would find useful here is dbRND().

7) Now its time to handle the gameplay. Use dbMouseClick() to catch a mouse click, and dbMouseX() and dbMouseY() to determine the (x,y) location that was clicked. dbWaitMouse() can be used to wait until the mouse is clicked.

8) Optional: Indicate when the user has completed the game. You could play a sound, make the circles flash, or show the user a message!

To complete this assignment you must

1. Create a project called **ColorMatchingGame** with the proper interfaces and implementations for playing the game.

2. Type the solution (fully documented/commented) to the problem into your projects. The hard copy must be placed on the instructor's desk by the time class starts on the day that it is due. The hard copy must be printed in color, double-sided, and stapled in the upper left corner.

3. Remember to enter in your name as the author of the program. Also, each file is the have file header documentation.

4. Make sure that your program compiles and runs correctly. If you get any errors, double check that you typed everything correctly. Be aware that C++ is case-sensitive. Also, there must not be any warnings when compiling your program (other than those produced by Dark GDK) or you will lose points.

5. Once you are sure that the program works correctly, it is time to submit your solution. You do this by logging on to Turing and placing your complete solution folder with all projects working correctly in the proper CS250 Drop folder. Make sure that you copy your program folder and don't move the folder. If you move the folder, then you will not have your own copy!

6. Play the game until about half of the circles are correctly matched. Then use a snipping tool to copy the screen, then print it out in color and staple it to the end of your print out.

We have given you enough time to finish the assignment without having to work over spring break. We would advise that you complete steps 1-5 by Friday. If you wait until after spring break to start your assignment, then you will most likely not complete it in time. Good Luck and Have Fun!!!