SDI Document/View App help from:

http://www.scribd.com/doc/29336159/06-Introduction-to-Mfc

http://www.scribd.com/doc/7124911/Visual-Cpp-and-MFC-Programming

http://aclacl.brinkster.net/MFC/ch09c.htm

http://msdn.microsoft.com/en-us/library/ms821625.aspx

http://msdn.microsoft.com/en-us/library/6w6cd538(v=vs.80).aspx

http://www.functionx.com/visualc/howto/calldlgfromsdi.htm

Changes made to the 1999 "Programming Windows with MFC" SDI Square Color application:

Part 1: Added a random number generator to pick a number 1-6 and call the respective color function to get that color back to the OnLButtonDown function which changes the color of the specified box by getting it's ***x*** and ***y*** positions. Also the original tutorial never calls the different functions for colors, that is why I was getting only one color at first. (part 1 is commented out, as I changed the way the program works)

Part 2: When a box is clicked, the user is asked to choose a color of the box, which changes the color of the box to the specified color.

Game Rules inspired by:

http://www.colormatchgame.com/default.asp

http://powayusd.sdcoe.k12.ca.us/teachers/smiddleton/Geo/Proofs/Colored%20Square%20Game.htm

**Note:** Rules of the game can be found by clicking Help ~> How To Play ☺

Additional Functions that I created:

void SetSquare(int i, int j, COLORREF color); - Sets the chosen color on the corresponding square
void ComputerMoves(); - is called after the user chooses a box to pick a random color to play at the square chosen from SetSquare
void UserMoves(); - is called when the user plays
void SelectSquare(int& i, int& j); helps the computer find a random Square that is not already used to play with random colors.
void Results(); - after all the squares have been colored, result checks to see if the user won and how many times.
afx_msg void OnLButtonDown(UINT nFlags, CPoint point); - once the user clicks a square, this function is activated

afx_msg void OnPickacolor(); - activates the color picker window for the user to pick a color for the game
void doSelectColor(hwnd); helps the previous function
bool IsDone(); - a Boolean variable that checks whether or not all the squares have been colored.
bool ComputerWon() - Checks whether the computer was able to use 4 consecutive squares.

The i and j values for the 4x4 Square

| | | | |
|---|---|---|---|
| **00** | **01** | **02** | **03** |
| **10** | **11** | **12** | **13** |
| **20** | **21** | **22** | **23** |
| **30** | **31** | **32** | **33** |

# Rules

The user picks a color from the menu bar to play. If the user selects a square without choosing a color, an error message will pop up advising the user to pick a color. As soon as the user picks a color, the computer also plays. The computer is programmed to prevent the user from winning by either making black lines or preventing the user's moves. It selects a random square that has not been used yet around the area of the last user position and fills it with BLACK. Therefore, the user can choose any other color besides BLACK. It is definitely very possible to win the computer, as I didn't check all the possible steps☺. The user cannot click a square that has already been filled, neither can the computer. Options to win are any horizontal or vertical 4 consecutive squares:

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |