Matching Game User Story

What I want it to do:

1. From the first form, I want it to be able to display the title of the game and a button to display the game.
2. After the Begin Game button is clicked, it hides the Title screen and brings up an instance of the first level.
3. From here the player will select two pictures and if the two pictures are a match, the game tells the player it was a match and keeps the two pictures face-up. If not, the pictures go face down and the player guesses again. When the player has all of the pictures face-up, the system tells the player they have beaten the level closes and control is brought back to the title screen.
4. The title screen will then determine if the player has beaten the first level and if so, it will bring up an instance of the second level. (If the player exits a level, the title screen still checks and resets the game if this occurs.) This process occurs until the end of the third level.
5. After the third level it returns to the title screen.

Title Screen Breakdown:



Branching if statements that Control levels and resetting:

These variables keep track of whether a player has beaten each level. These are local to make sure they are not always false on pressing the “Begin Game” Button

//LevelCheck Variables

bool level\_1\_Complete = false;

bool level\_2\_Complete = false;

bool level\_3\_Complete = false;

This branching if statement controls what levels can come up and calls a reset function when it notices that the player did not beat the level. Each time a Level closes, Functions that set the local variables to true are called.

//Check to see if level 1 has been beaten, if so it moves on.

//If not, it opens level 1.

if (level\_1\_Complete == false)

{

//Bring up level one

level\_1.ShowDialog();

if (level\_1.LevelFinished() == true)

Level1Finished();

else

LevelReset();

}

//Check to see if level 1 has been beaten first and then checks level 2.

//If so, it moves on. If not, it opens level 2.

if (level\_1\_Complete == true && level\_2\_Complete == false)

{

level\_2.ShowDialog();

if (level\_2.LevelFinished() == true)

Level2Finished();

else

LevelReset();

}

//Check to see if level 2 has been beaten first and then checks level 3.

//If so, it moves on. If not, it opens level 3.

if (level\_2\_Complete == true && level\_3\_Complete == false)

{

level\_3.ShowDialog();

if (level\_3.LevelFinished() == true)

Level3Finished();

else

LevelReset();

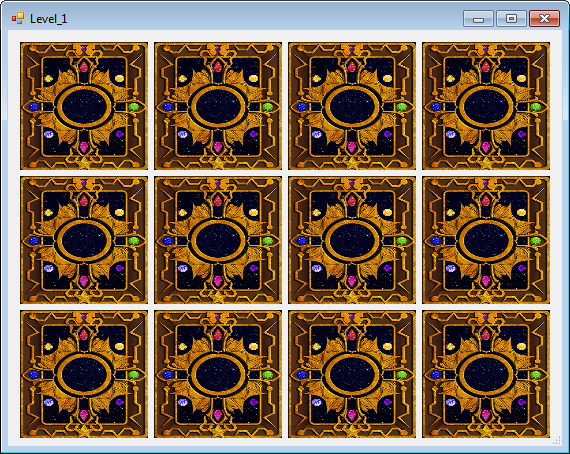
}

//Resets the game if the user has gone through every level and completed it.

if (level\_1\_Complete == true && level\_2\_Complete == true && level\_3\_Complete == true)

LevelReset();

Level 1-3 breakdown







How these levels work:

Upon clicking 2 pictures, the system takes in their numbers and compares them in a series of if statements that are matches. If the any of the if statements prove true, the system tells the user that it’s a match. If not, a function is called that sets any non-matched pictures visibility to false.

Each level has a number of global variables that are Boolean. They are necessary for keeping track of each match.

PictureChecker() takes in the number of the picture clicked on. If that number was the first number it sets it to the first picture variable. If second, then it sets it to the second one. If both numbers have been set, then the CheckMatch() function is called using those two numbers as arguments.

CheckMatch() checks for matches using the two arguments given and if any of the series of if statements are true, a match has been made. If not, the Boolean variable “MTR” is declared as false and an if statement is called.

private void PictureChecker(int pictureNumber)

private void CheckMatch(int picture1, int picture2)

The MTR variable stands for “Match This Round?”. If yes, it ignores the calling of the NoMatches() Function. If so, it calls it.

if (MTR == false)

{

NoMatches();

}

NoMatches() Contains the series of if statements that compares whether any cards should stay up if they have been matched. If any pictures that have been matched have been turned face-up, there will be turned face-down.

private void NoMatches()

While WinGame is called, it only takes effect if all matches have been made. If all matches have been made, the player is told they have won and will then close the level. Control will be shifted back to the title screen to check if the level was beaten.

private void WinGame()

The LevelFinished() Function is only utilized by the Title screen when it checks for a Level Complete. The function contains an if statement that returns true if met. The statement checks to see if all of the matches have been made for that level.

public bool LevelFinished()