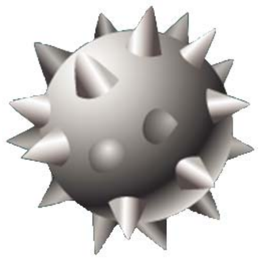
Minesweeper Game Interaction

Clement Fingerle
April 27, 2012

Project Advisor:
Hongli Luo

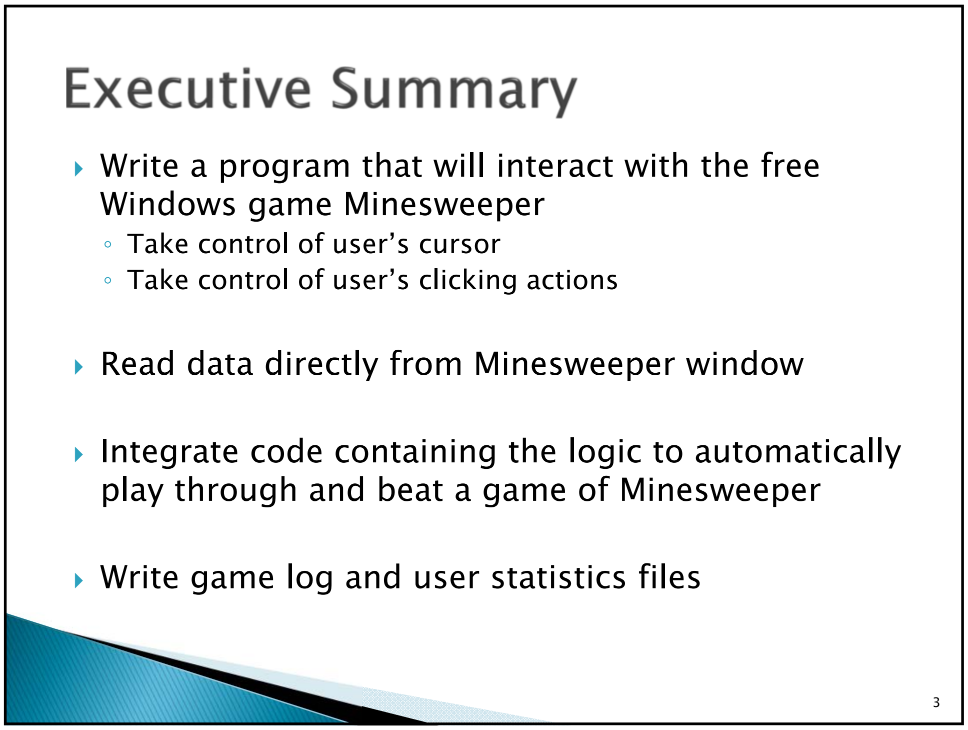
CPET 491 Instructor:
Paul I-Hai Lin



1

What Is Going To Be Covered

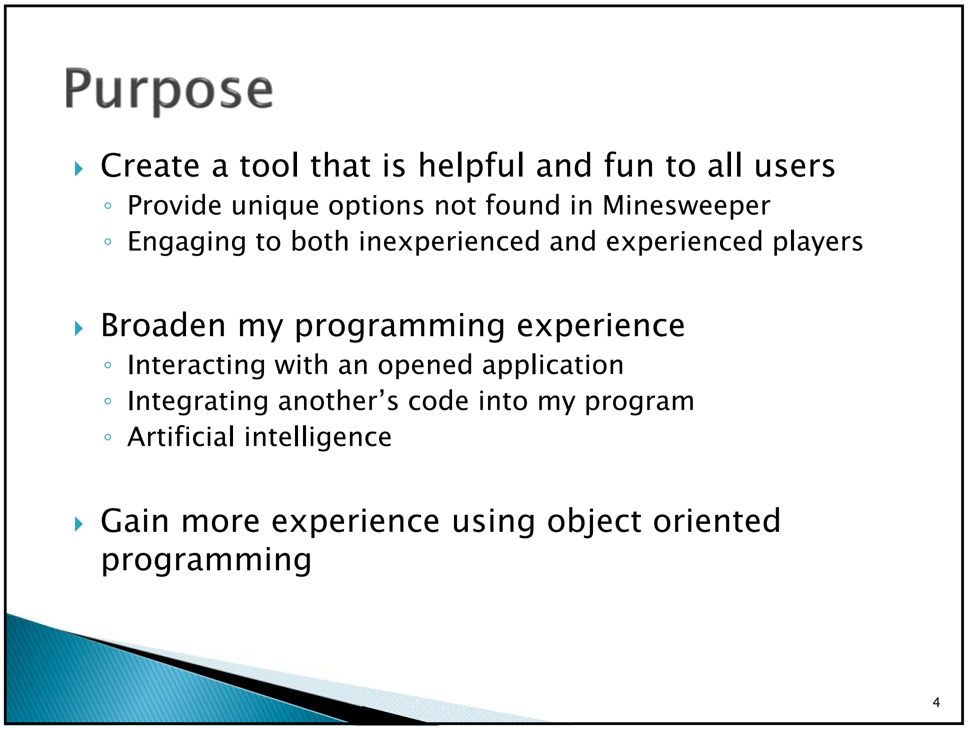
2



Executive Summary

- Write a program that will interact with the free Windows game Minesweeper
  - Take control of user’s cursor
  - Take control of user’s clicking actions
- Read data directly from Minesweeper window
- Integrate code containing the logic to automatically play through and beat a game of Minesweeper
- Write game log and user statistics files

3

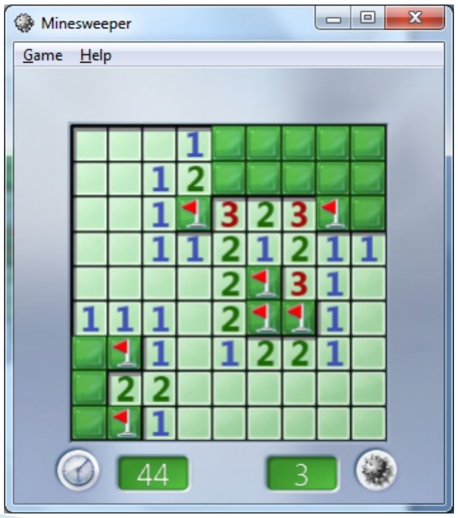


Purpose

- Create a tool that is helpful and fun to all users
  - Provide unique options not found in Minesweeper
  - Engaging to both inexperienced and experienced players
- Broaden my programming experience
  - Interacting with an opened application
  - Integrating another’s code into my program
  - Artificial intelligence
- Gain more experience using object oriented programming

4

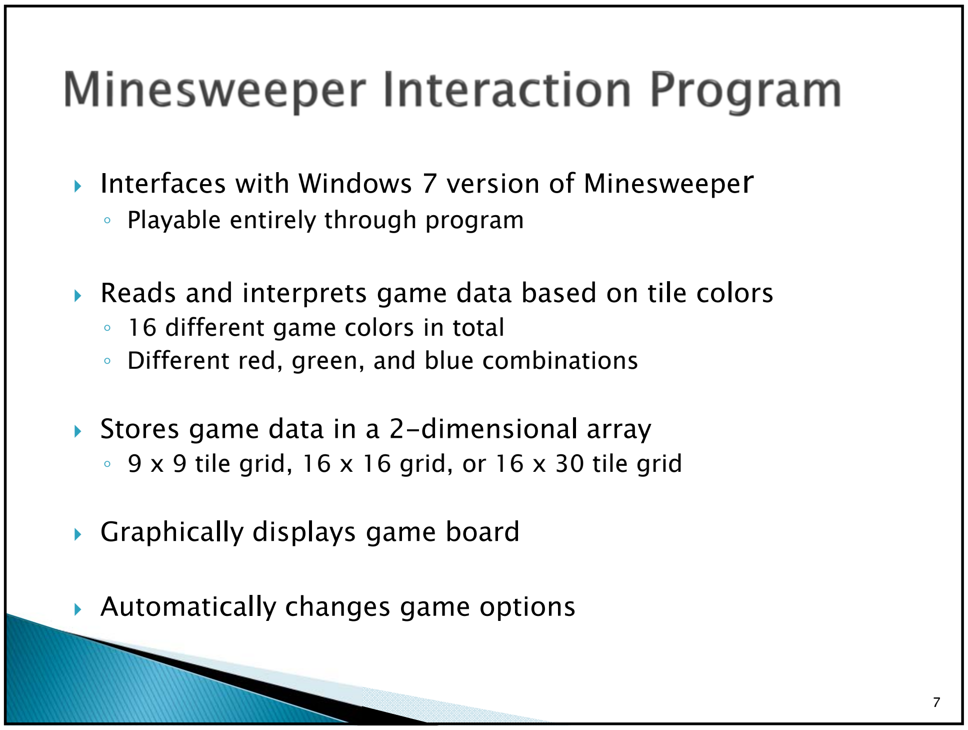
What is Minesweeper?



5

System Analysis

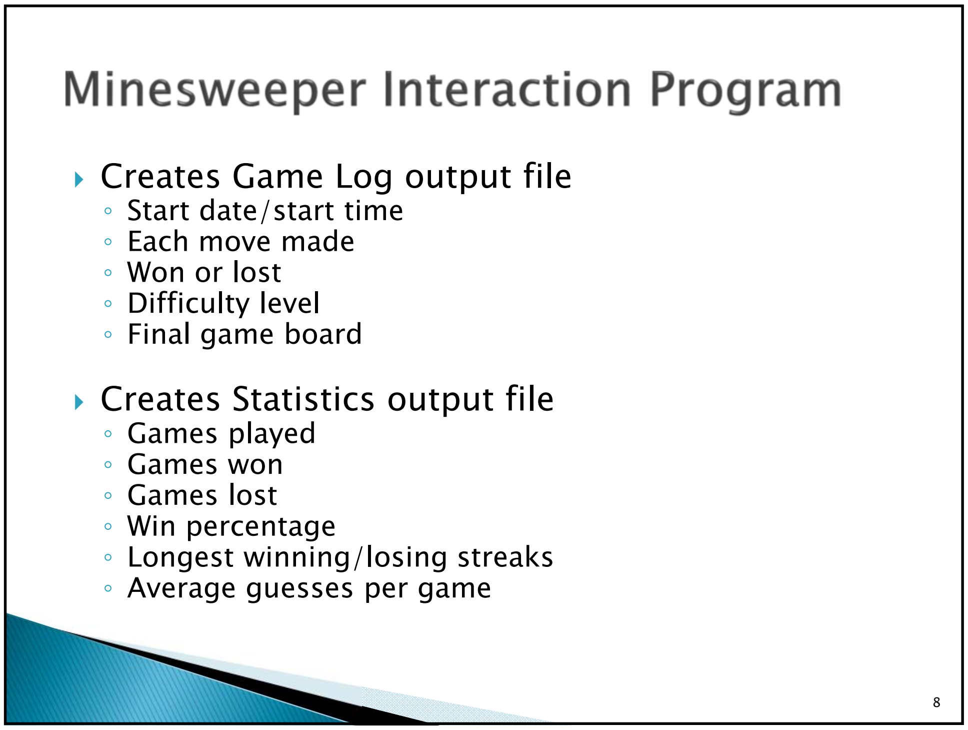
6



Minesweeper Interaction Program

- Interfaces with Windows 7 version of Minesweeper
  - Playable entirely through program
- Reads and interprets game data based on tile colors
  - 16 different game colors in total
  - Different red, green, and blue combinations
- Stores game data in a 2-dimensional array
  - 9 x 9 tile grid, 16 x 16 grid, or 16 x 30 tile grid
- Graphically displays game board
- Automatically changes game options

7

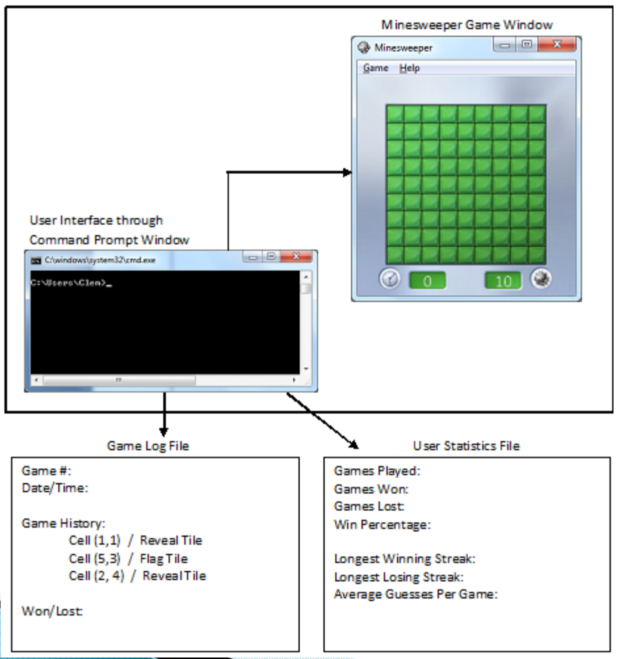


Minesweeper Interaction Program

- Creates Game Log output file
  - Start date/start time
  - Each move made
  - Won or lost
  - Difficulty level
  - Final game board
- Creates Statistics output file
  - Games played
  - Games won
  - Games lost
  - Win percentage
  - Longest winning/losing streaks
  - Average guesses per game

8

System Architecture



```
C:\Users\Clem>_
```

Game #:
Date/Time:

Game History:
Cell (1,1) / Reveal Tile
Cell (5,3) / Flag Tile
Cell (2, 4) / Reveal Tile

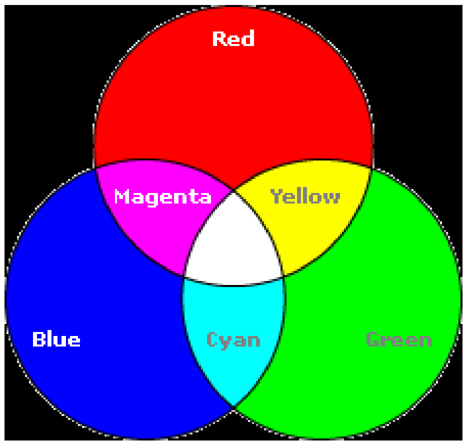
Won/Lost:

Games Played:
Games Won:
Games Lost:
Win Percentage:

Longest Winning Streak:
Longest Losing Streak:
Average Guesses Per Game:

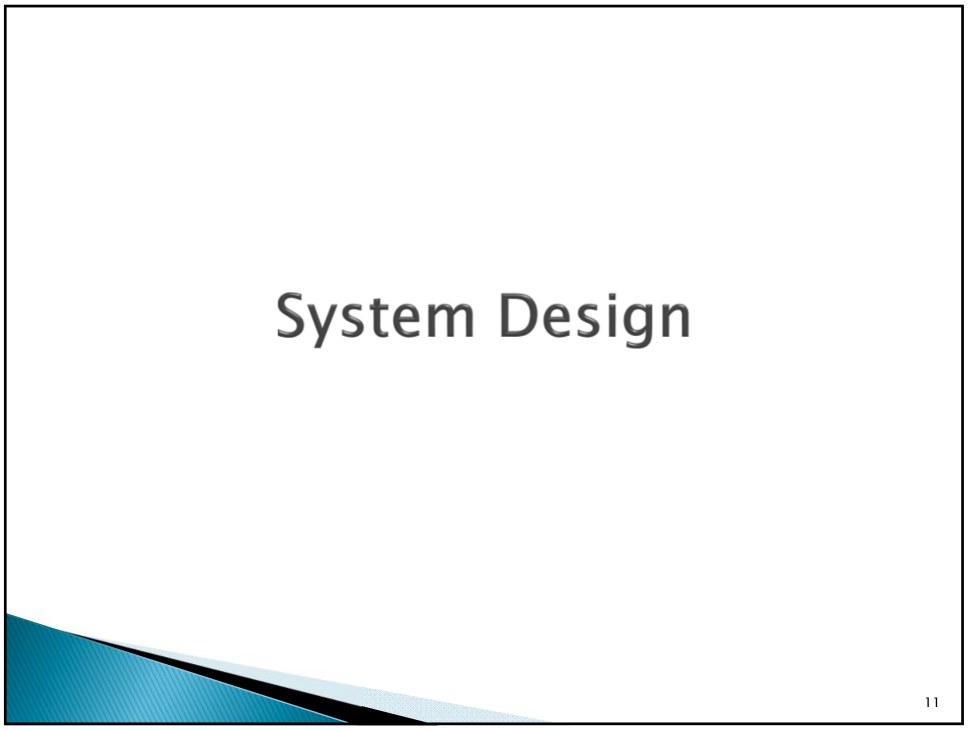
9

The RGB Color Model



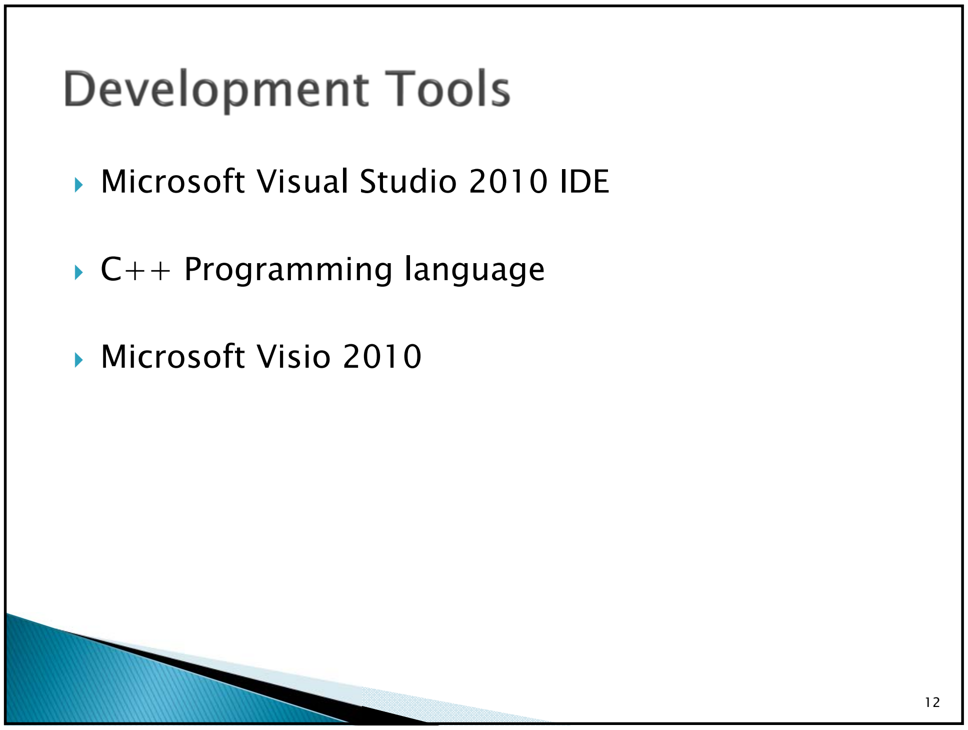
In computers, each color is assigned an intensity level from 0 to 255

10



System Design

11



Development Tools

- Microsoft Visual Studio 2010 IDE
- C++ Programming language
- Microsoft Visio 2010

12

Key Components

- Window Handler (HWND) Object allows:
  - Data retrieval from window
  - Manipulation of window's size, position, or appearance
- Window Size Detection
  - Size of window changes as difficulty level changes
  - Boundaries of game must be known
  - Retrieve dimensions with GetWindowRect() function

Difficulty Level	**Window Width**
Beginner	223 pixels
Intermediate	349 pixels
Advanced	601 pixels

Key Components

- Resizing Game Window
  - Program only works when window size is smallest
  - ShowWindow() function restores window to its original size and position
  - SetWindowPos() function shrinks window
- Changing Game Options
  - When problems are encountered, options are changed
  - Uses function keys (F5, F7) and hot keys to navigate menus

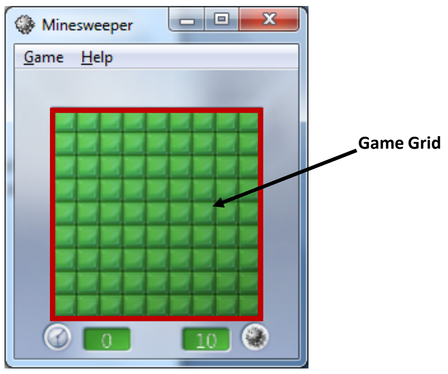
Key Components

- Simulating Key Presses
  - keybd_event() function
- Simulating Mouse Movement
  - SetCursorPos() function
  - Given x and y pixel coordinates
- Simulating Mouse Clicks
  - SendInput() function

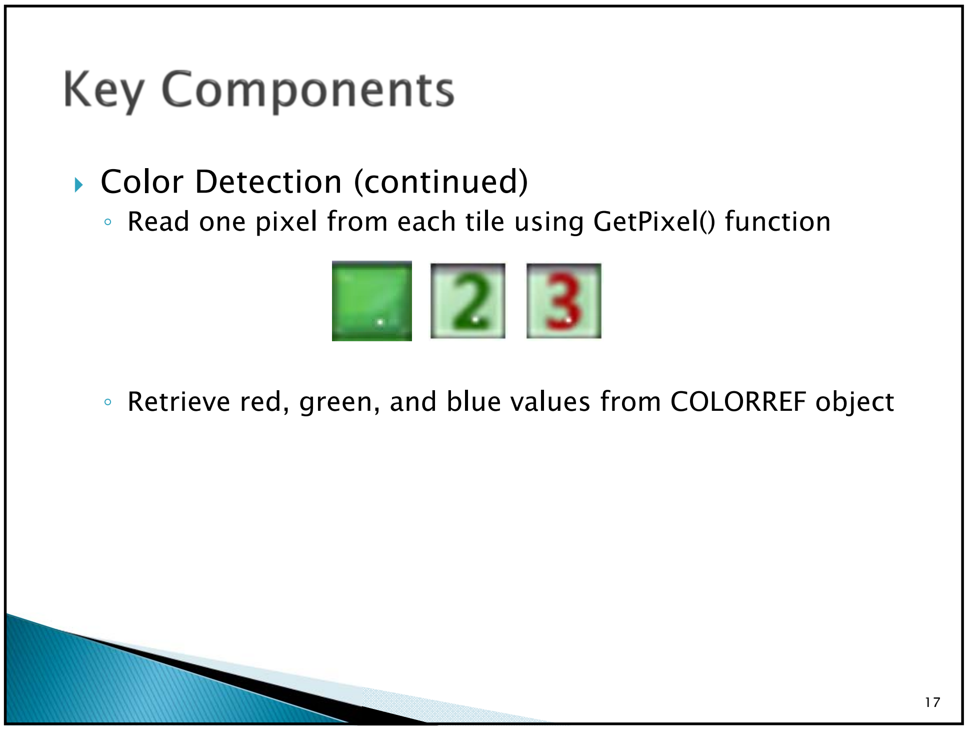
15

Key Components

- Color Detection
  - First, find location of game grid

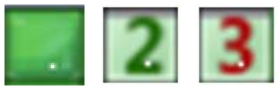


16



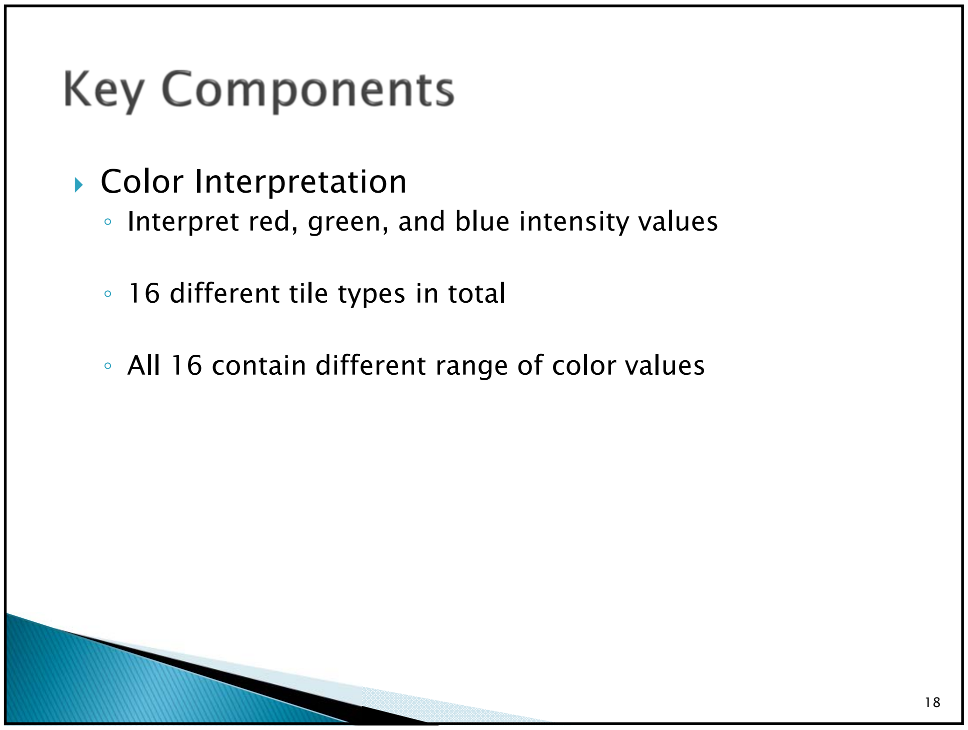
Key Components

- Color Detection (continued)
  - Read one pixel from each tile using GetPixel() function



  - Retrieve red, green, and blue values from COLORREF object

17

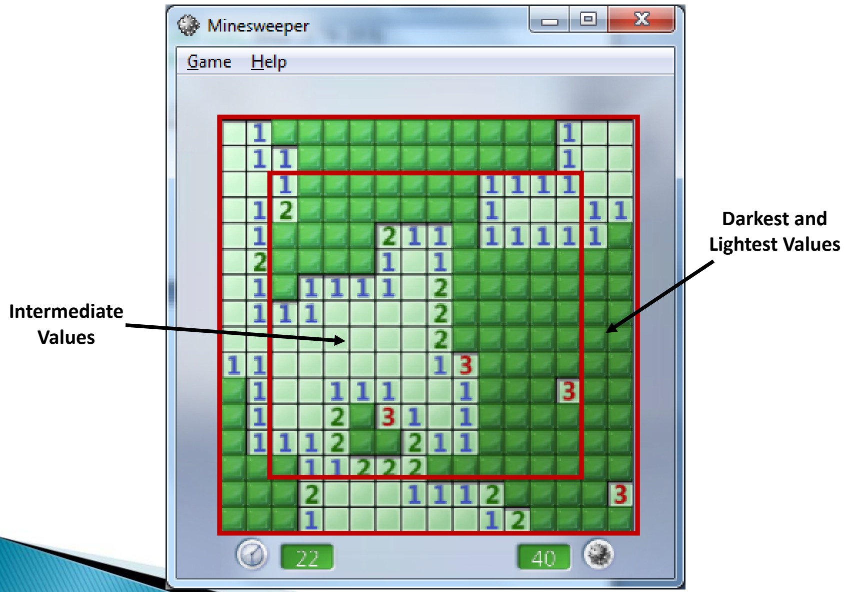


Key Components

- Color Interpretation
  - Interpret red, green, and blue intensity values
  - 16 different tile types in total
  - All 16 contain different range of color values

18

Lightness/Darkness of Game Window



19

Tile Types and their Color Values (for version 1)

Tile Type	Tile Appearance	Red Values	Blue Values	Green Values	Additional Constraints
Empty	🟩	147-219	148-222	200-254	green - red > 20
Number 1	1	63-116	155-189	80-144	none
Number 2	2	40-92	15-84	106-136	red > blue
Number 3	3	143-173	8-64	8-84	none
Number 4	4	11-125	125-154	14-162	none
Number 5	5	115-127	4-49	5-64	none
Number 6	6	23-125	129-148	133-188	none
Number 7	7	Unknown	Unknown	Unknown	none
Number 8	8	Unknown	Unknown	Unknown	none

20

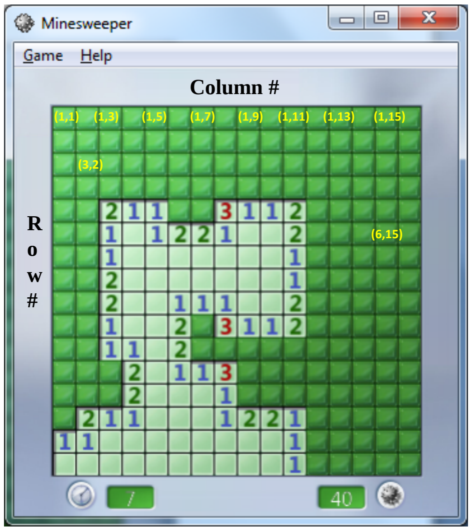
Tile Types and their Color Values (for version 1)

Green Unopened	🟩	**48-115**	**49-140**	**112-215**	**blue + 3 ≥ red AND red + blue - 40 ≤ green**
Green Unopened (highlighted)	🟩	**80-147**	**112-203**	**176-255**	**none**
Blue Unopened	🟦	**48-162**	**179-253**	**65-216**	**none**
Blue Unopened (highlighted)	🟦	**80-194**	**242-255**	**129-255**	**none**
Flag	🚩	**160-238**	**164-239**	**165-240**	**green - red ≤ 20**
Flag (highlighted)	🚩	**191-247**	**244-252**	**225-252**	**none**
Mine	●	**50-180**	**50-170**	**70-170**	**none**

Key Components

- Different versions of Minesweeper
  - Three known versions of Minesweeper
  - Each displays colors differently
- Different versions of my program
  - Two versions of the Minesweeper Interaction Program
  - Two sets of color interpretation rules
  - Often detects if wrong version is being used

Coordinate System



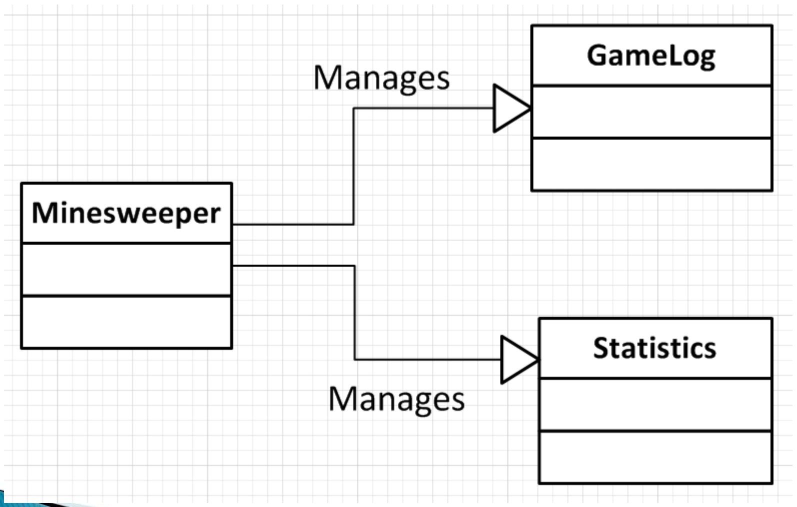
Each tile's coordinate can be referred to by a function of its row number and column number.

23

UML Class Diagrams

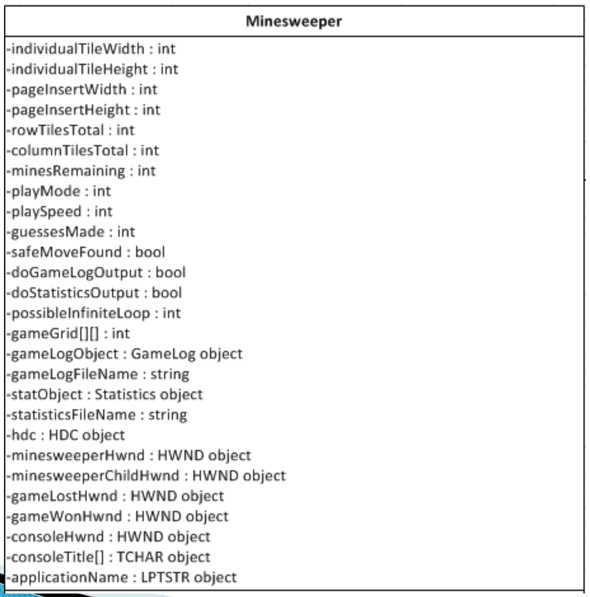
24

Class Relationships



25

Minesweeper Class (attributes)



Minesweeper
-individualTileWidth : int
-individualTileHeight : int
-pageInsertWidth : int
-pageInsertHeight : int
-rowTilesTotal : int
-columnTilesTotal : int
-minesRemaining : int
-playMode : int
-playSpeed : int
-guessesMade : int
-safeMoveFound : bool
-doGameLogOutput : bool
-doStatisticsOutput : bool
-possibleInfiniteLoop : int
-gameGrid[][] : int
-gameLogObject : GameLog object
-gameLogFileName : string
-statObject : Statistics object
-statisticsFileName : string
-hdc : HDC object
-minesweeperHwnd : HWND object
-minesweeperChildHwnd : HWND object
-gameLostHwnd : HWND object
-gameWonHwnd : HWND object
-consoleHwnd : HWND object
-consoleTitle[] : TCHAR object
-applicationName : LPTSTR object

26

Minesweeper Class (functions)

+MinesweeperInteraction()
+~MinesweeperInteraction()
+AutomaticPlaythrough() : int
+UserPlaythrough()
+GameGridChecker()
+GameGridDisplay()
+ChangeDifficulty()
+ChangePlaySpeed(in givenPlaySpeed : int)
+StartNewGame()
+Help()
+CreateGameLogFile() : bool
+CreateStatisticsFile() : bool
+CloseMinesweeper()
+OpenMinesweeper()
+SelectProperOptions()
+UpdateCurrentGameDimensions()
+UpdateTotalMinesRemaining()
+SetPlayMode(in playingModeInput : int)
+SetSafeMoveFound(in safeMoveFoundInput : bool)
+GetSafeMoveFound() : bool
+AdvancedAnalysis()
+AdvancedAnalysis2(in rowOfRelatedTile : int, in columnOfRelatedTile : int, in rowOffLimits : int, in columnOffLimits : int, in rowOffLimits2 : int, in columnOffLimits2 : int) : bool
-MoveMouseClick(in rowNumber : int, in columnNumber : int, in whichClick : int)
-LeftClick()
-RightClick()
-LeftPlusRightClick()
-TileDetection(in rowNumber : int, in columnNumber : int) : int
-ColorInterpretation(in r : int, in b : int, in g : int) : int
-DetectGameOver() : bool
-SafeMove(in safeRowNumber : int, in safeColumnNumber : int, in whichClick : int)
-FindMinesweeperWindow()

27

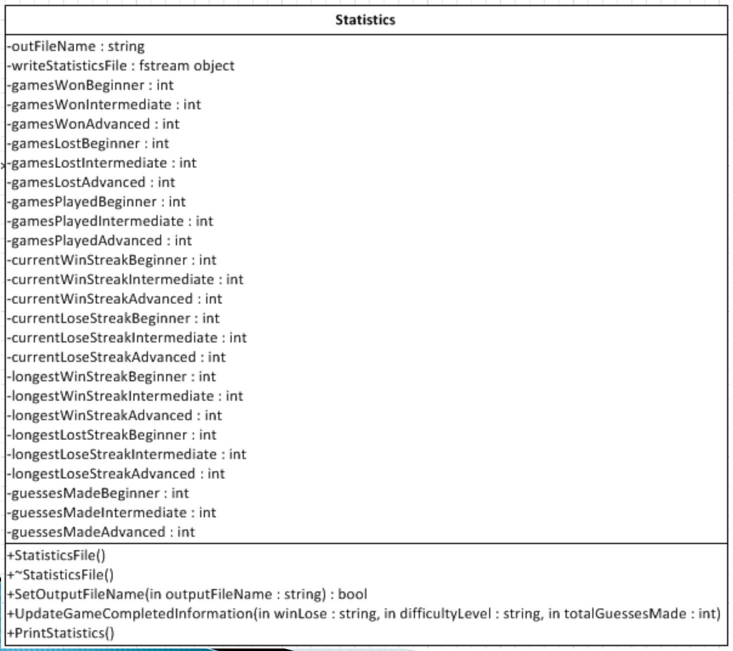
GameLog Class

GameLog
-currentGameNumber : int -outFileName : string -writeGameLogFile : fstream object

+GameLogFile()
+~GameLogFile()
+SetOutputFileName(in outputFileName : string) : bool
+PrintHeadingInformation()
+PrintGameHistory(in xPosition : int, in yPosition : int, in whichClick : int)
+PrintFinalGameBoard(in visualToPrint : string)
+PrintGameCompletedInformation(in winLose : string, in difficultyLevel : string)

28

Statistics Class



Statistics

```
-outFileName : string
-writeStatisticsFile : fstream object
-gamesWonBeginner : int
-gamesWonIntermediate : int
-gamesWonAdvanced : int
-gamesLostBeginner : int
-gamesLostIntermediate : int
-gamesLostAdvanced : int
-gamesPlayedBeginner : int
-gamesPlayedIntermediate : int
-gamesPlayedAdvanced : int
-currentWinStreakBeginner : int
-currentWinStreakIntermediate : int
-currentWinStreakAdvanced : int
-currentLoseStreakBeginner : int
-currentLoseStreakIntermediate : int
-currentLoseStreakAdvanced : int
-longestWinStreakBeginner : int
-longestWinStreakIntermediate : int
-longestWinStreakAdvanced : int
-longestLostStreakBeginner : int
-longestLoseStreakIntermediate : int
-longestLoseStreakAdvanced : int
-guessesMadeBeginner : int
-guessesMadeIntermediate : int
-guessesMadeAdvanced : int
```

```
+StatisticsFile()
+~StatisticsFile()
+SetOutputFileName(in outputFileName : string) : bool
+UpdateGameCompletedInformation(in winLose : string, in difficultyLevel : string, in totalGuessesMade : int)
+PrintStatistics()
```

29

System Testing

- All threshold requirements met
- Five of seven objective requirements met

ID#	Requirement	Inspect, Analyze, Demo or Test	Pass/ Fail
A-3	The system shall assign a grid system to the active playing window that stores the status of the game window.	Analyze	Pass
A-6	The system should emulate an intelligent human playing the game.	Inspect	Pass
A-7	The system should not trigger any anti-virus warnings.	Test	Fail
B-1	The system shall be able to accurately interpret the colors of at least 13 out of 16 different tile types.	Analyze	Pass
B-2	The system shall be able to accurately navigate a user's cursor to *every* tile on the game grid on the hardest difficulty.	Demo	Pass
B-3	The system shall be able to accurately left click on *every* tile on the game grid on the hardest difficulty.	Demo	Pass

30

System Testing

ID#	Requirement	Inspect, Analyze, Demo or Test	Pass/ Fail
C-2	The system shall maintain a file of the game log.	Inspect	Pass
C-3	The system shall maintain a file of user statistics.	Inspect	Pass
D-3	The system shall operate on the Windows 7 operating system.	Test	Pass
D-4	The system should operate on the Windows Vista operating system.	Test	Fail
E-5	The system shall read the colors of pixels to interpret the game information.	Analyze	Pass
E-6	The system should be able to detect when a game has ended.	Analyze	Pass
E-7	The system should be able to detect if a user either wins or loses a game.	Analyze	Pass
E-8	The system should be able to detect the dimensions of the active playing window.	Analyze	Pass

31

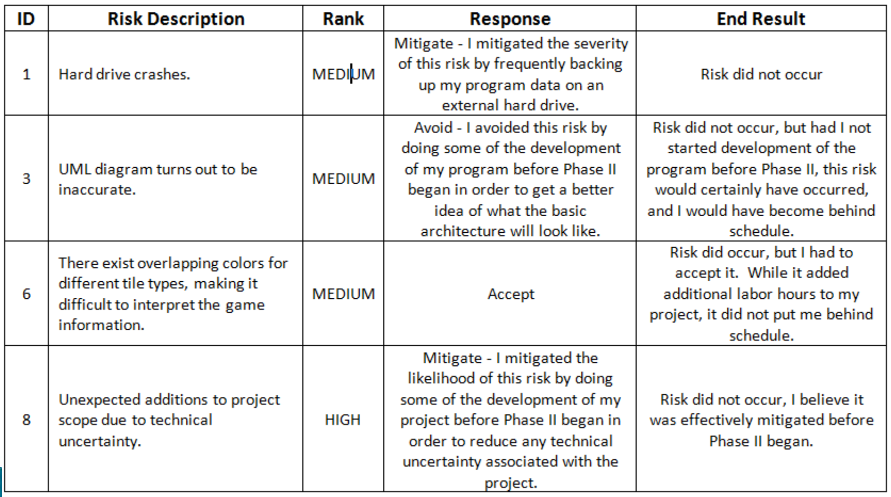
Cost Management

- Total cost for materials for this project was $0
- Estimated labor hours: 90 hours
- Actual labor hours: 140 hours

Task	Estimated Hours	Actual Labor Hours
System Design	7 hours	8 hours
Detailed Design	42 hours	66 hours
Data Detection & Interpretation	15 hours	31 hours
User Statistics Storage & Output	12 hours	10 hours
Game Interaction	15 hours	25 hours
System Integration	15 hours	22 hours
Requirements Verification	6 hours	5 hours
Final Report Development	10 hours	30 hours
Presentation Development	10 hours	9 hours

32

Risk Analysis



ID	Risk Description	Rank	Response	End Result
1	Hard drive crashes.	MEDIUM	Mitigate - I mitigated the severity of this risk by frequently backing up my program data on an external hard drive.	Risk did not occur
3	UML diagram turns out to be inaccurate.	MEDIUM	Avoid - I avoided this risk by doing some of the development of my program before Phase II began in order to get a better idea of what the basic architecture will look like.	Risk did not occur, but had I not started development of the program before Phase II, this risk would certainly have occurred, and I would have become behind schedule.
6	There exist overlapping colors for different tile types, making it difficult to interpret the game information.	MEDIUM	Accept	Risk did occur, but I had to accept it. While it added additional labor hours to my project, it did not put me behind schedule.
8	Unexpected additions to project scope due to technical uncertainty.	HIGH	Mitigate - I mitigated the likelihood of this risk by doing some of the development of my project before Phase II began in order to reduce any technical uncertainty associated with the project.	Risk did not occur, I believe it was effectively mitigated before Phase II began.

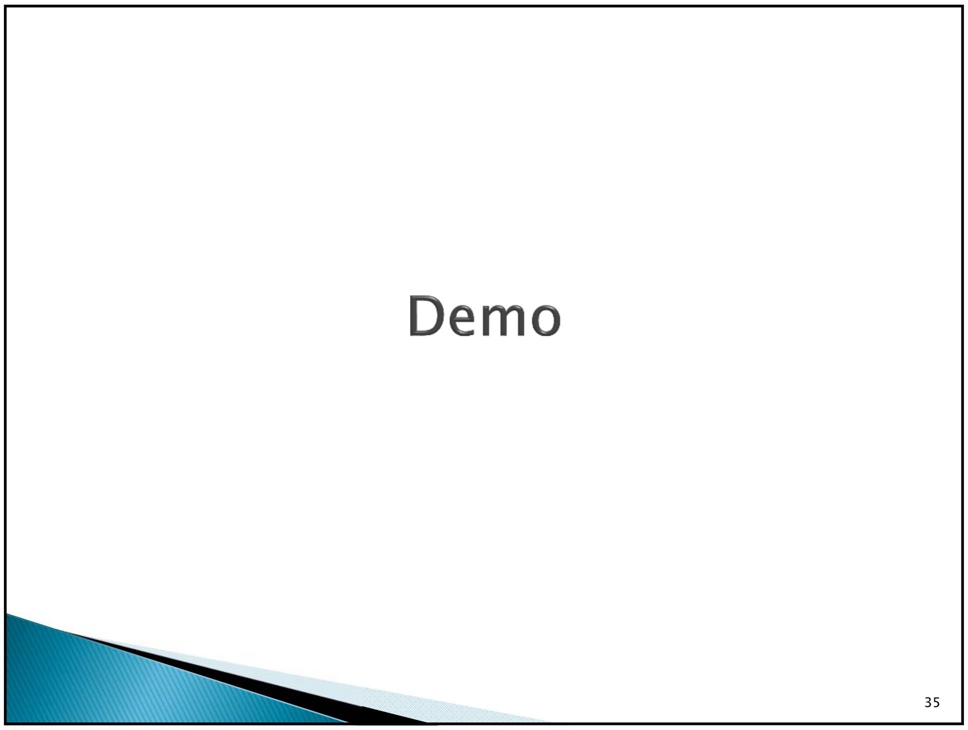
33

Conclusion

- Project was completed on schedule
- Project requirements all met

- Lessons Learned
  - Programming techniques I've learned in school can be applied to virtually any program
  - Gained experience modifying and interpreting another person's code
  - Programming computers to think like humans is remarkably difficult
  - Projects are unpredictable, a little bit of planning can go a long way

34



Demo

35