Keith Walsh
CS333
Final Project Top-Down Design Assignment
12/9/09

Note: I removed all implementation code that I'd used in my original top-down design since it's now (greatly modified) in the program. I hope this is okay.

Play Gravilex;

This is the gravilex.cpp file – it also handles the test set up, which wasn't in the initial top-down plan. Otherwise it's as described except there is no end game function since the spec doesn't call for any specific output when the grid is full.

//* Play Gravilex;

Set up game;
Play game until grid is full;

// Play Gravilex;

//*Set up game;
Game game;
This is in the game.cpp file, class Game, which really has just these two functions
game.setUpGame
{
Set up grid
Set up humanPlayer
Set up computerPlayer
~~Set up dictionary~~//I removed this function and instead made the class Dictionary a variable inside the Grid class, since that was the only thing that needed access. The vector is populated after the Grid class is created.
}
Play game until grid is full;

// Play Gravilex;

```
//Set up game;
Game game;
game.setUpGame();
{

//* Set up grid;
Grid gameGrid;
```
This is in the grid.cpp file, class Grid, which controls all activity inside the grid

```
gameGrid.setUpGrid()
{
Create a 6X7 two-dimensional vector of chars
Initialize each element of the vector with a '.' to indicate empty space
}
Set up humanPlayer
Set up computerPlayer
}
}
Play game until grid is full;
----------

// Play Gravilex;

//Set up game;
Game game;
game.setUpGame;
{

// Set up grid;
Grid gameGrid;
gameGrid.setUpGrid();

//* Set up humanPlayer;
//* Set up computerPlayer;
```
Setting up the players is accomplished by creating two instances of the class Player, in file player.cpp
My initial top-down had two different class for computer and human, but since they have the same attributes I decided to put them together.

```
Player humanPlayer;
humanPlayer.setUpPlayer();

Player computerPlayer;
computerPlayer.setUpPlayer();
```

setUpPlayer(); //For both players
{
Assign letters to player in vector<char> //This is all they need to get started
}

}
Play game until grid is full;

// Play Gravilex;

//Set up game;
Game game;
game.setUpGame();
{

// Set up grid;
gameGrid.setUpGrid();

// Set up players;
Player humanPlayer;
humanPlayer.setUpPlayer();

Player computerPlayer;
computerPlayer.setUpPlayer();

}
//* Play game until grid is full;

This is in the Game class. The main driver calls the function until the grid is full. I didn't need to make any changes to how this works except it doesn't need to check for end, since that's done in the main driver.
while(gameGrid is not full)
{
playRound(){
Display grid
Show remaining player letters
Prompt for letter
Process letter
Prompt for column
Process column
Display grid

Prompt for claimed word
Check for word in available played words //I added this
Check for word in entire grid //I added this
Check for word in dictionary
Calculate points
Display points for turn
Display grid
Show remaining computer letters
Make computer selection
~~Check for word in dictionary~~//Part of the make computer selection function
Claim all computer words
Calculate points
Display points for turn
Display score summary
~~Check to see if game is complete, if so end game~~//Happens in gravilex.cpp
}
}

// Play Gravilex;

//Set up game;
Game game;
game.setUpGame();
{

// Set up grid;
gameGrid.setUpGrid();

// Set up players;
Player humanPlayer;
humanPlayer.setUpPlayer();

Player computerPlayer;
computerPlayer.setUpPlayer();

}
// Play game until grid is full;

while(gameGrid is not full)
{
playRound(){

```
//* Display grid
gameGrid.displayGrid()
```

This is just how I initially described it initially

```
{
for each row of the board, display every column;
}
Show remaining player letters
Prompt for letter
Process letter
Prompt for column
Process column
Display grid
Prompt for claimed word
Check for word in available played words
Check for word in entire grid
Check for word in dictionary
Calculate points
Display points for turn
Display grid
Show remaining computer letters
Make computer selection
Claim all computer words
Calculate points
Display points for turn
Display score summary
}
}
```

```
// Play Gravilex;

//Set up game;
Game game;
game.setUpGame();
{

// Set up grid;
gameGrid.setUpGrid();

//  Set up players;
Player humanPlayer;
humanPlayer.setUpPlayer();
```

```
Player computerPlayer;
computerPlayer.setUpPlayer();

}
// Play game until grid is full;

while(gameGrid is not full)
{
playRound(){
// Display grid
gameGrid.displayGrid();
//* Show remaining player letters
humanPlayer.displayLettersRemaining()
{
print “Letters remaining to you:”
print each letter in the letters array for human player
}
Prompt for letter
Process letter
Prompt for column
Process column
Display grid
Prompt for claimed word
Check for word in available played words
Check for word in entire grid
Check for word in dictionary
Calculate points
Display points for turn
Display grid
Show remaining computer letters
Make computer selection
Claim all computer words
Calculate points
Display points for turn
Display score summary
}
}

----------

// Play Gravilex;
```

```
//Set up game;
Game game;
game.setUpGame();
{

// Set up grid;
gameGrid.setUpGrid();

//  Set up players;
Player humanPlayer;
humanPlayer.setUpPlayer();

Player computerPlayer;
computerPlayer.setUpPlayer();

}
// Play game until grid is full;

while(gameGrid is not full)
{
playRound(){
// Display grid
gameGrid.displayGrid();
// Show remaining player letters
humanPlayer.displayLettersRemaining();
//* Prompt for letter
humanPlayer.promptForLetter()
{
print “What letter would you like to drop?”
cin a letter and validate it.  Reprompt as necessary.
}
~~Process letter~~ // I ended up doing all validation in the promptForLetter() function
Prompt for column
Process column
Display grid
Prompt for claimed word
Check for word in available played words
Check for word in entire grid
Check for word in dictionary
Calculate points
Display points for turn
Display grid
Show remaining computer letters
```

Make computer selection
Claim all computer words
Calculate points
Display points for turn
Display score summary
}
}

// Play Gravilex;

//Set up game;
Game game;
game.setUpGame();
{

// Set up grid;
gameGrid.setUpGrid();

// Set up players;
Player humanPlayer;
humanPlayer.setUpPlayer();

Player computerPlayer;
computerPlayer.setUpPlayer();

}
// Play game until grid is full;

while(gameGrid is not full)
{
playRound(){
// Display grid
gameGrid.displayGrid();
// Show remaining player letters
humanPlayer.displayLettersRemaining();
// Prompt for letter
humanPlayer.promptForLetter();
//* Prompt for column
humanPlayer.promptForColumn()
{
print “What column would you like to drop that in? (1-7)”
cin a number and validate it. Reprompt as necessary

}
~~Process column~~// I put this in the promptForColumn() function
Display grid
Prompt for claimed word
Check for word in available played words
Check for word in entire grid
Check for word in dictionary
Calculate points
Display points for turn
Display grid
Show remaining computer letters
Make computer selection
Claim all computer words
Calculate points
Display points for turn
Display score summary
}
}

// Play Gravilex;

//Set up game;
Game game;
game.setUpGame();
{

// Set up grid;
gameGrid.setUpGrid();

// Set up players;
Player humanPlayer;
humanPlayer.setUpPlayer();

Player computerPlayer;
computerPlayer.setUpPlayer();

}
// Play game until grid is full;

while(gameGrid is not full)
{
playRound(){

```
// Display grid
gameGrid.displayGrid();
// Show remaining player letters
humanPlayer.displayLettersRemaining();
// Prompt for letter
humanPlayer.promptForLetter();
// Prompt for column
humanPlayer.promptForColumn();
//* Add letter to column //I had to add this additional step from my intial top-down
gameGrid.addToGrid()
// function will add the letter to the grid, returning the row
{
in column, determine the highest row number with a '.'
replace that item in the vector with the specified letter
return row
}
Display grid
Check for word in available played words
Check for word in entire grid
Check for word in dictionary
Calculate points
Display points for turn
Display grid
Show remaining computer letters
Make computer selection
Claim all computer words
Calculate points
Display points for turn
Display score summary
}
}

----------

// Play Gravilex;

//Set up game;
Game game;
game.setUpGame();
{

// Set up grid;
gameGrid.setUpGrid();
```

```
//  Set up players;
Player humanPlayer;
humanPlayer.setUpPlayer();

Player computerPlayer;
computerPlayer.setUpPlayer();

}
// Play game until grid is full;

while(gameGrid is not full)
{
playRound(){
// Display grid
gameGrid.displayGrid();
// Show remaining player letters
humanPlayer.displayLettersRemaining();
// Prompt for letter
humanPlayer.promptForLetter();
// Prompt for column
humanPlayer.promptForColumn();
// Add letter to column
gameGrid.addToGrid();
gameGrid.displayGrid();
//* Prompt for claimed word
humanPlayer.promptForWord()
{
print "Enter a word that you would like to claim (empty line to end): "
if the player enters hits return with no input, leave loop
if text entered, take all letters before whitespace and treat them as a claimed word
check for word in available played words on the grid (needs to be function in the grid)
check for word in entire grid (needs to be function in the grid)
check for word in the dictionary
}
~~Check for word in available played words~~ //These tasks I put in the promptForWord() function
~~Check for word in entire grid~~
~~Check for word in dictionary~~
Calculate points
Display points for turn
Display grid
Show remaining computer letters
Make computer selection
```

```
Claim all computer words
Calculate points
Display points for turn
Display score summary
}
}

----------

// Play Gravilex;

//Set up game;
Game game;
game.setUpGame();
{

// Set up grid;
gameGrid.setUpGrid();

//  Set up players;
Player humanPlayer;
humanPlayer.setUpPlayer();

Player computerPlayer;
computerPlayer.setUpPlayer();

}
// Play game until grid is full;

while(gameGrid is not full)
{
playRound(){
// Display grid
gameGrid.displayGrid();
// Show remaining player letters
humanPlayer.displayLettersRemaining();
// Prompt for letter
humanPlayer.promptForLetter();
// Prompt for column
humanPlayer.promptForColumn();
// Add letter to column
gameGrid.addToGrid();
gameGrid.displayGrid();
// Prompt for claimed word
```

```
humanPlayer.promptForWord();
//* Calculate points
gameGrid.scoreWord()
//I put this in the Grid class for ease of use with the computer decision function to follow
{
take string, identify its size, return number of points to be awarded
}
Display points for turn
Display grid
Show remaining computer letters
Make computer selection
Claim all computer words
Calculate points
Display points for turn
Display score summary
}
}

----------

// Play Gravilex;

//Set up game;
Game game;
game.setUpGame();
{

// Set up grid;
gameGrid.setUpGrid();

//  Set up players;
Player humanPlayer;
humanPlayer.setUpPlayer();

Player computerPlayer;
computerPlayer.setUpPlayer();

}
// Play game until grid is full;

while(gameGrid is not full)
{
playRound(){
// Display grid
```

```
gameGrid.displayGrid();
// Show remaining player letters
humanPlayer.displayLettersRemaining();
// Prompt for letter
humanPlayer.promptForLetter();
// Prompt for column
humanPlayer.promptForColumn();
// Add letter to column
gameGrid.addToGrid();
gameGrid.displayGrid();
// Prompt for claimed word
humanPlayer.promptForWord();
// Calculate points
gameGrid.scoreWord();
//* Display points for turn
humanPlayer.updatePlayerPoints()
// I made this a function inside Player to keep track of points.  The only time points are displayed is when new points are (potentially) added
{
print "Points scored on this turn: " points
total points += points;
}
Display grid
Show remaining computer letters
Make computer selection
Claim all computer words
Calculate points
Display points for turn
Display score summary
}
}

----------

// Play Gravilex;

//Set up game;
Game game;
game.setUpGame();
{

// Set up grid;
gameGrid.setUpGrid();
```

```
//  Set up players;
Player humanPlayer;
humanPlayer.setUpPlayer();

Player computerPlayer;
computerPlayer.setUpPlayer();

}
// Play game until grid is full;

while(gameGrid is not full)
{
playRound(){
// Display grid
gameGrid.displayGrid();
// Show remaining player letters
humanPlayer.displayLettersRemaining();
// Prompt for letter
humanPlayer.promptForLetter();
// Prompt for column
humanPlayer.promptForColumn();
// Add letter to column
gameGrid.addToGrid();
gameGrid.displayGrid();
// Prompt for claimed word
humanPlayer.promptForWord();
// Calculate points
gameGrid.scoreWord();
// Display points for turn
humanPlayer.updatePlayerPoints();
//Display grid
gameGrid.displayGrid();
//* Show remaining computer letters
computerPlayer.displayLettersRemaining(); //This is the same function, different class instance, as human player
//* Make computer selection
gameGrid.decideBestMove()
// This function in the Grid class call all necessary functions to review all possible plays and pick the one specified in the decision making spec
{
go through loop of players letters, using addToColumn to find out which rows they'll land in
review each move using a findAllWords function, checking each against the dictionary
```

score each of the found words and store the information in a struct
Once all moves for all letters have been scored, evaluate the moves and scores against the decision tree provided in the spec
once a decision has been reached, add the letter to the grid
print "I will drop the letter X in column Y"
for each word claimed, print "I claim the word: " word
}
~~Claim all computer words~~//Built into the function above
Calculate points
Display points for turn
Display score summary
}
}

```
// Play Gravilex;

//Set up game;
Game game;
game.setUpGame();
{

// Set up grid;
gameGrid.setUpGrid();

//  Set up players;
Player humanPlayer;
humanPlayer.setUpPlayer();


Player computerPlayer;
computerPlayer.setUpPlayer();

}
// Play game until grid is full;

while(gameGrid is not full)
{
playRound(){
// Display grid
gameGrid.displayGrid();
// Show remaining player letters
humanPlayer.displayLettersRemaining();
// Prompt for letter
```

```
humanPlayer.promptForLetter();
// Prompt for column
humanPlayer.promptForColumn();
// Add letter to column
gameGrid.addToGrid();
gameGrid.displayGrid();
// Prompt for claimed word
humanPlayer.promptForWord();
// Calculate points
gameGrid.scoreWord();
// Display points for turn
humanPlayer.updatePlayerPoints();
//Display grid
gameGrid.displayGrid();
// Show remaining computer letters
computerPlayer.displayLettersRemaining();
// Make computer selection
gameGrid.decideBestMove();
// Calculate points
computerPlayer.updatePlayerPoints();//This is the same function, different class instance, as human player
// Display points for turn
computerPlayer.updatePlayerPoints();//This is the same function, different class instance, as human player
//* Display score summary
{
this is simple enough to contain in the playRound(); loop
print "Score so far: You="
print the points for the humanPlayer points variable (uses humanPlayer.showPoints();)
print " Me="
print the points for the computerPlayer points variable (uses computerPlayer.showPoints();)
}
}
```
