UPDATED VERSION

Peter Campbell


```
"""
*****************************************************************************
 FILE:      boggle.py

 AUTHOR:    Peter Campbell

 ASSIGNMENT: Laboratory 5

 DATE:      11/15/12

 DESCRIPTION: A program that makes the game Boggle! The code contains classes
 for pieces, the board, and players. It combines these in a final function
 'playGame' that contains the main loop of the game to switch between players
 and keep score.

*****************************************************************************
"""
from boggleUtils import *
from cs110graphics import *
from random import *

#initializes and creates the necessary methods for the class named Piece
class Piece:
    def __init__(self, loc = Point(0, 0)):
        """create a piece centered at point loc"""
        self._value = 'a'
        self._loc = loc
        self._border = Square(80, self._loc)
        self._border.setFillColor('blue')
        self._text = Text(str(self._value), self._loc, 18)
        self._text.setDepth(45)
        self._isUsed = False

    def setValue(self, newValue = 'b'):
        """sets the value of this piece to newValue"""
        self._value = newValue
        self._text.setTextString(str(self._value))

    def getValue(self):
        """return the letter this piece represents"""
        return self._value

    def addTo(self, container):
        """place the graphical components of this piece in the container"""
        container.add(self._border)
        container.add(self._text)

    def isUsed(self):
        """Return whether this piece has been used already"""
        return self._isUsed

    def markUsed(self):
        """Note that this piece has been used, changing piece color"""
        self._isUsed = True
        self._border.setFillColor('green')

    def markUnused(self):
        """Note that this piece has not been used, changing piece color"""
        self._isUsed = False
        self._border.setFillColor('blue')

#initializes and creates the necessary methods for the class named Board
class Board:
```

```
64      def __init__(self):
65          self._listPieces = []
66          self._listLetters = []
67          self._title = Text('Boggle', Point(300, 60), 48)
68
69          self._title.setDepth(40)
70
71      def make(self, win):
72          """This adds all of the pieces to the board with shuffled values. It
73      adds all of the pieces to a list and all of the values to a list. The
74      list of pieces is a list of lists to access its place on the board."""
75          row1 = []
76          row2 = []
77          row3 = []
78          row4 = []
79          win.add(self._title)
80          for i in range (16):
81              if i < 4:
82                  piece = Piece(Point(170 + 80 * (i), 180))
83                  row1.append(piece)
84              elif i < 8 and i > 3:
85                  piece = Piece(Point(170 + 80 * (i-4), 260))
86                  row2.append(piece)
87              elif i < 12 and i > 7:
88                  piece = Piece(Point(170 + 80 * (i-8), 340))
89                  row3.append(piece)
90              elif i > 11:
91                  piece = Piece(Point(170 + 80 * (i-12), 420))
92                  row4.append(piece)
93          self._listPieces = [row1, row2, row3, row4]
94          self.shuffle()
95          for row in self._listPieces:
96              for piece in row:
97                  piece.addTo(win)
98          for row in self._listPieces:
99              for piece in row:
100                 self._listLetters.append(piece.getValue)
101
102
103     def shuffle(self):
104         """This shiffles the values of the pieces on the board and makes sure
105     that a letter cannot be used twice."""
106         self._listLetters = []
107         listValues = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',
108                       'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
109                       'W', 'X', 'Y', 'Z']
110         for row in self._listPieces:
111             for piece in row:
112                 newValue = listValues.pop(randrange(len(listValues)))
113                 piece.setValue(newValue)
114                 self._listLetters.append(newValue)
115
116     def findWordStart(self, win, letter):
117         """This takes the letter that the player fist inputs for a word and
118     checks to see if it's on the board, if it is the piece is marked as
119     used."""
120         letter = letter.upper()
121         found = False
122         while not found:
123             if letter in self._listLetters:
124                 piece = self.getPiece(letter)
125                 piece.markUsed()
126                 found = True
```

```
127                 else:
128                     print 'That letter is not on the board.'
129                     sleep(1)
130                     letter = getLetter(win).upper()
131             return piece.getValue()
132
133         def isAdjascent(self, newLetter, oldPos):
134             """This takes the input of a letter that is not the first letter of a
135         word and checks to see if it is adjascent to the last valid letter
136         entered."""
137             ox, oy = oldPos
138             nx, ny = self.getPosition(newLetter)
139             if nx >= ox - 1 and nx <= ox + 1 and ny >= oy - 1 and ny <= oy + 1:
140                 return True
141
142
143         def getPiece(self, val):
144             """This takes a letter and checks to see if it is in the list of lists
145         for all of the pieces on the board."""
146             for i in range(len(self._listPieces)):
147                 for j in range(len(self._listPieces[i])):
148                     if self._listPieces[i][j].getValue() == val.upper():
149                         return self._listPieces[i][j]
150
151         def getPosition(self, val):
152             """This program returns a tuple of the row and column of the piece that
153         has the same value as the input value, essentially giving that pieces
154         position on the board."""
155             for i in range(len(self._listPieces)):
156                 for j in range(len(self._listPieces[i])):
157                     if self._listPieces[i][j].getValue() == val.upper():
158                         return (i, j)
159
160         def getWord(self, win, firstLetter):
161             """This program calls 'getLetter' and is adjascent repeadtedly to get a
162         word from the user. It then checks to see if the word is in the
163         dictionary."""
164             word = ''
165             word = word + firstLetter
166             oldPos = self.getPosition(firstLetter)
167             newLetter = getLetter(win)
168             while newLetter != 'Return':
169                 while newLetter.upper() not in self._listLetters:
170                     print "That letter is not on the board"
171                     sleep(1)
172                     newLetter = getLetter(win)
173                 piece = self.getPiece(newLetter)
174                 if self.isAdjascent(newLetter, oldPos) and not piece.isUsed():
175                     word = word + newLetter
176                     piece.markUsed()
177                     oldPos = self.getPosition(newLetter)
178                 else:
179                     print 'Letter is not adjascent or has already been used.'
180                     sleep(1)
181                     newLetter = getLetter(win)
182                 newLetter = getLetter(win)
183             if isWord(word):
184                 print word + ' is a word!'
185                 sleep(1)
186                 return word
187             else:
188                 print word + ' is not a word'
189                 sleep(1)
```

```
190             self.clearBoard()
191
192
193     def clearBoard(self):
194         """This marks all of the pieces on the board as unused."""
195         for row in self._listPieces:
196             for piece in row:
197                 piece.markUnused()
198
199 #initializes and creates the necessary methods for the class named Player
200 class Player:
201     def __init__(self, name, loc):
202         self._name = name
203         self._loc = loc
204         self._score = 0
205         self._text = Text(str(self._name + ':' + str(self._score)), self._loc)
206         self._text.setDepth(45)
207
208     def addTo(self, container):
209         """Place the graphical components of the player in the container."""
210         container.add(self._text)
211
212     def addScore(self, word):
213         """Add the score of a word to the players total score."""
214         self._score = self._score + len(word)
215         self._text.setTextString(str(self._name + ':' + str(self._score)))
216
217     def getScore(self):
218         """Accesses the players score."""
219         return self._score
220
221 def getLetter(win):
222     """Accepts a keystroke that is either a letter or return."""
223     allowed = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
224                'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
225                'y', 'z', 'Return']
226     ev = str(win.wait('Select a letter or press enter to end word').getKey())
227     if ev in allowed:
228         return ev
229
230 def playGame():
231     """The main game function that creates the window, board, and players. It
232   accepts a first letter from the first player and then starts the game loop.
233   While a players score is below the winning total and they don't press return
234   they can input words to be added to their score. Once they press enter the
235   turn switches to the other player and the board is shuffled. When a player
236   reaches or passes 100 points the game is over."""
237     win = Window(600, 600)
238     win.setBackgroundColor('darkred')
239     board = Board()
240     playerA = Player('PlayerA', Point(100, 520))
241     playerA.addTo(win)
242     playerB = Player('PlayerB', Point(100, 550))
243     playerB.addTo(win)
244     board.make(win)
245     player = playerA
246     firstLetter = win.wait('Type a word or press enter to pass.').getKey()
247     while playerA.getScore() <= 100 and playerB.getScore() <= 100:
248         listWords = []
249         while firstLetter != 'Return':
250             board.findWordStart(win, firstLetter)
251             word = board.getWord(win, firstLetter)
252
```

```
253                 #if the word is not in the dictionary it goes through the loop again
254                 #CITE: TA Leah Wolf
255                 #DETAILS: Helped me understand why the game wasn't looping properly
256                 #and that I needed to creat another while loop for if the word was
257                 #not in the dictionary.
258                 while word == None and firstLetter != 'Return':
259                     firstLetter = getLetter(win)
260                     board.findWordStart(win, firstLetter)
261                     word = board.getWord(win, firstLetter)
262                 if word != None and word not in listWords and firstLetter != \
263                         'Return':
264                     listWords.append(word)
265                     player.addScore(word)
266                     board.clearBoard()
267                     firstLetter = win.wait('Type a word or press enter to ' +
268                                            'pass.').getKey()
269                 elif word in listWords:
270                     print "That word has already been used."
271                     sleep(1)
272                     board.clearBoard()
273                     firstLetter = getLetter(win)
274                 print "Turn passed to opponent, shuffling..."
275             board.shuffle()
276             if player == playerA:
277                 player = playerB
278             elif player == playerB:
279                 player = playerA
280             firstLetter = win.wait('Type a word or press enter to ' +
281                                    'pass.').getKey()
282         if playerA.getScore() >= 100:
283             print 'PlayerA wins!'
284         else:
285             print 'PlayerB wins!'
286         win.wait()
287         win.close()
288
289
290 if __name__ == "__main__":
291     StartGraphicsSystem(playGame)
```