A2 Computing Project

Exam centre: XXXX

Candidate's name: Beck

Centre number: XXXXX

Candidate number: XXXX

Contents Page

Problem Definition

Description of client and users

My organisation is a small tuition centre, in West London, which has about 400 students. They specialise in teaching the core subjects: English, Maths and Science and on a one-to-one basis and in small groups. My client is a teacher who works from 9 till 5 weekly. He currently teaches about 10 students in a class and teaches a total of 4 different classes a week. My client would like to vary his teaching methods and so would like for me to a game which he could use to help his students to remember “Key words”.

My users are the students that my client teaches. The user will range from 8 year olds to 18 year olds, and will be taught in the appropriate class, i.e. The 16 – 18 year old student will be taught in the A level class.

Description of Word Search

Word search is a popular puzzle game that was developed by Norman E. Gibat, on March 1st, 1968. The first word search made its debut appearance in small newspaper company named “Selenby Digest”. However, a Spanish puzzle create named Pedro Ocón de Oro, was known for creating and publishing "Sopas de letras", before the year 1968. The puzzle is universally known as “word find, word seek, word sleuth, mystery word, wordfind, wordseek and often has words that relate to a key theme, for example types of dog. Nowadays, word search puzzles appear in many newspapers, magazines, puzzle books and even on the internet.

The objective of the game is for the user to finding all the missing words; once they have located and marked out all the missing words. The grids come in different sizes, which are aimed to increase the difficult of gameplay. The grid is usually square or rectangular in shape; however, the shape of the grid may vary according to the theme or target audience. For example, a word search puzzle targeted towards children may be designed in the shape of a Christmas tree, if Christmas is there theme.

The words may be placed in all direction, horizontally, vertically or diagonally, and can be read either up, down or backwards and forwards. Words may overlap, by using a single character. Most word search games include a mixture of all these features. The player is also given a list of words to find, however, in more difficult version of the game the user may be left on his/her own to find the missing words.

In the modern era, online versions of word search have a timer feature this may make it more difficult for the player to win the game. Once the timer reaches zero the game ends, since the player has run out of time. A message box will then informing them of their defeat, it will then provide the user with an option to start a new game.

I will be using visual basic to build a simplified version of the word search game. The game will be called “The Word Search Game” and will be used by my client to help educated his pupils, in a fun and enjoyable way.

The aim of my word search game is to help school students to learn to spell and memorise new words, which they can use in everyday life.

This is a version of the original *Sopa de letras,* created by Pedro Ocón de Oro. The game has a theme related to countries. The game is made easier for the user, since there are many words to find in a small

grid of characters. The game is printed in black and white ink; the font used is simple and makes it easier for the user to read

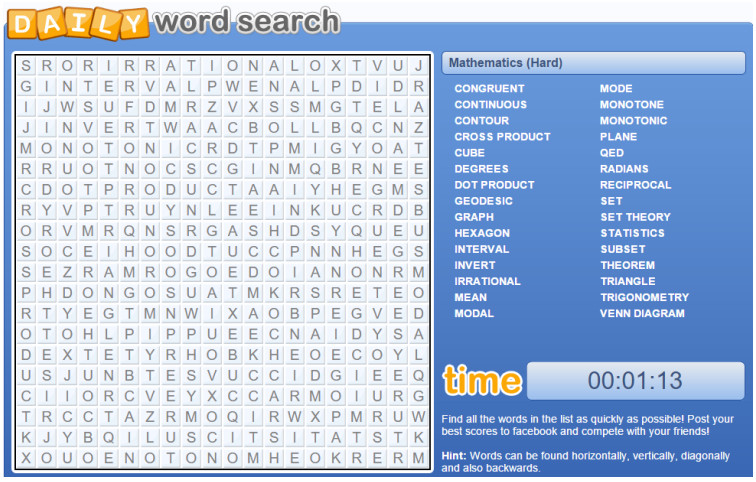
Sopa de Letras de Paises

C	O	R	A	A	I	D	N	I	D	V	L	Q	F	U
D	J	N	N	L	P	N	S	B	F	G	M	Z	Y	J
E	G	H	G	A	U	S	T	R	A	L	I	A	U	D
O	B	O	O	G	Q	U	E	N	I	A	Y	V	Y	X
H	D	L	L	U	F	A	A	N	D	O	R	R	A	W
C	F	A	A	T	L	R	H	C	I	T	A	L	I	A
C	S	N	T	R	T	U	P	N	G	N	M	R	I	D
Z	T	D	S	O	T	H	X	G	A	A	E	S	T	T
M	W	A	L	P	J	U	E	E	R	M	E	N	K	D
I	A	Q	I	T	C	S	R	R	M	N	E	J	T	E
C	T	L	M	S	P	H	O	Q	O	B	H	L	T	R
H	D	E	A	A	S	C	I	D	U	V	U	W	A	A
I	S	C	N	S	O	U	N	N	R	I	H	R	E	D
L	M	H	S	S	I	I	R	G	A	B	A	B	G	Y
E	A	N	D	L	Z	A	B	R	A	S	I	L	Y	O

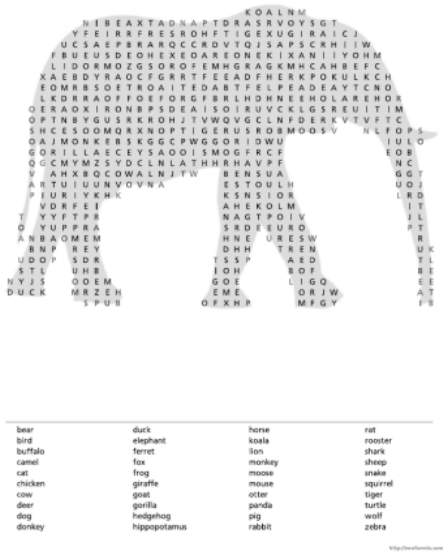
PORTUGAL
ESPANHA
HOLANDA
ALEMANHA
ITALIA
BRASIL
ANGOLA
INDIA
CHINA
QUENIA
AUSTRALIA
TURQUIA
MARROCOS
INDONESIA
RUSSIA
MALASIA
LUXEMBURGO
ANDORRA
CHILE

This is a modern version of the word search puzzle, which has high quality graphics. This specific example provides the user with a choice of 3 game modes: easy, medium and hard. All game modes feature a timer, which shows their remaining time, before the game ends. The style and theme are the same. What changes is the grid size, on easy mode; the user is given a reduced grid size to make it easier to find words. Furthermore, the numbers of words are reduced to help, making it easier to finish the game. I feel the game is missing a sound option and an exit button.

Below is the hardest game setting. The designers have made the game more difficult by increasing the number of words and grid size.



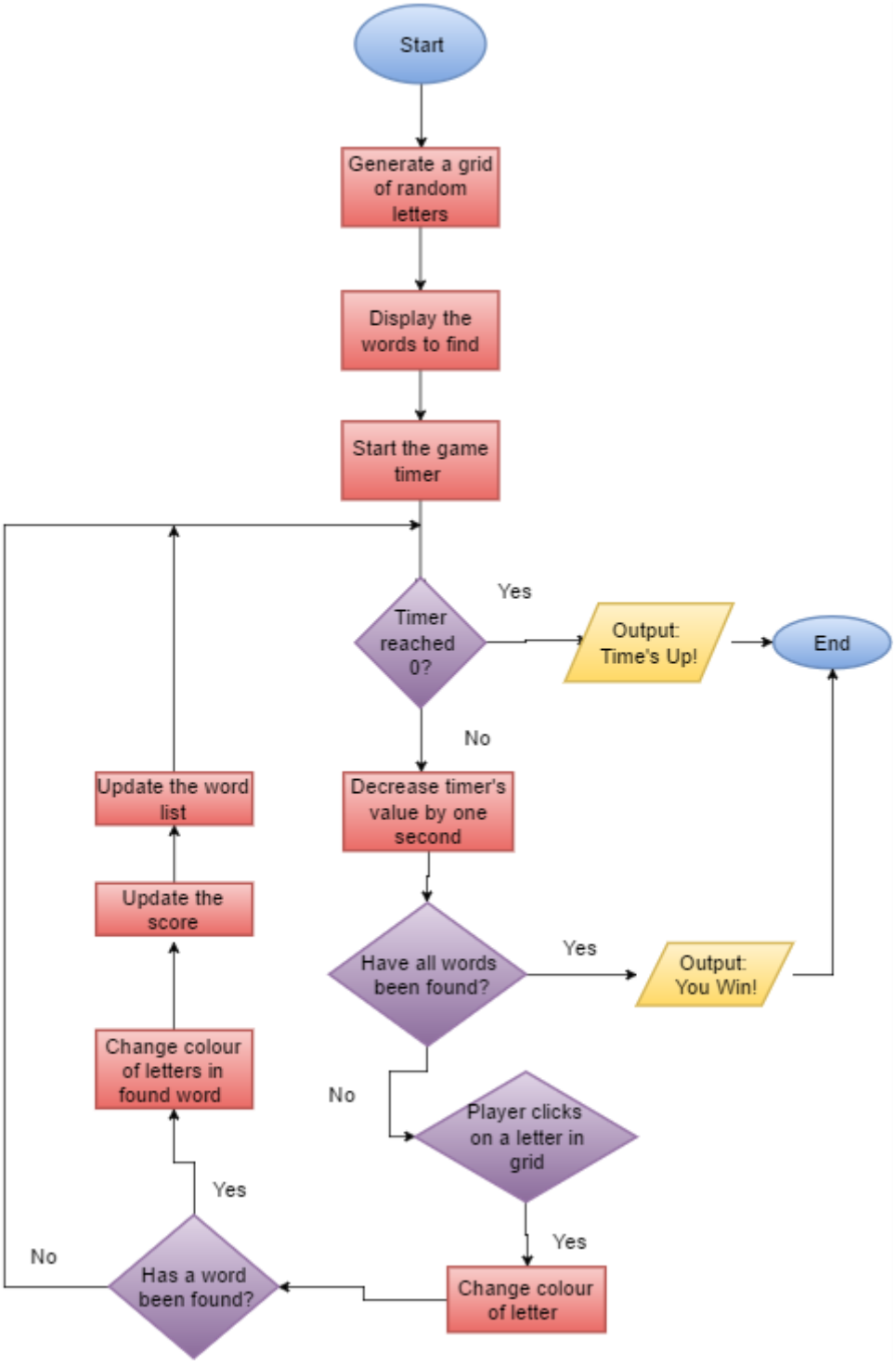
The final example shows a different design to the original ‘box’ shape design. This makes the word search puzzle more interesting and exciting for the younger age group.



bear
bird
buffalo
camel
cat
chicken
cow
deer
dog
donkey
duck
elephant
ferret
fox
frog
giraffe
goat
gorilla
hedgehog
hippopotamus
horse
koala
lion
monkey
moose
mouse
otter
panda
pig
rabbit
rat
rooster
shark
sheep
snake
squirrel
tiger
turtle
wolf
zebra

Current Process

I will now demonstrate the current process and basic functioning involved in a standard word search game.



Investigation and analysis

I will need to research, in detail, my users and client's requirements. This is because the users, client and I will have different views and ideas of the problem area. As a result, I have decided to prepare an interview with my client and two questionnaires for my users. I have chosen to research the student's needs in the form of a survey, as it is simple and easy for a student to complete, also I can target a wider diversity and volume of students, to improve data accuracy. An interview with a few teachers is decided, as it will provide me with more in-depth answers.

My questionnaire will mostly include closed questions, but there will also be some open questions too. The questionnaire will include questions about the different aspect of the game, such as, colours scheme and what the theme will be.

The interview will be different to the format used of the questionnaire, since the questions will be more flexible and specific to the end user. I will take a third person into the interview with me, whose role will be to note the end users comment. This will allow me to focus on what my end user is saying; so that I can uses follow up questions, to gain more information and to fully engage within the conversation. Finally, before I start the interview I will check whether the interviewee has any subjects that should not be brought up in the interview.

After completing my investigation I will summarise and analyse my results and then produce a user specification.

Arranging the interview

I will email my client, to arrange an interview, at a time that would be best suited to conduct the interview. I will send the following email:

Hi,
I am emailing you, so that I can arrange an interview with you to talk about the new system and the current method you use to teach your students. This opportunity will allow me to fully understand your needs and to get a clear image of the problem. I feel that your expertise will be beneficial in clearing up any misunderstanding.

Thank you,

Face-To-Face interview: Client

I have conducted a face to face interview with the computing teacher.

SIGNATURE: _____

DATE: _____

Questionnaire: Users

I am going to hand out my questionnaire to different students, aged 15 to 19. The students will be attending a school, college or sixth form. Below is the form which my potential users will complete and return once complete.

SIGNATURES: _____

DATE: _____

I am currently completing an A level computing project, in which I will create a word search puzzle game. As you may already know, word search puzzles involve finding and locating all the hidden words in a jumbled grid of letters. A list of the missing words is usually given. You win the game when all the words are found and highlighted

Your views are very important, as it will help me construct a requirement specification.

1. **Have you ever played word search before? If yes, how would you rate the gameplay?**

   Yes **No**

   1) Excellent **2) Quite good** **3) 50/50** **4) Quite bad** **5) Terrible**

   Extra comments:
   ……
   ……

2. **What is your preference of grid size, for a list of 5 words?**

   Small (e.g. 7x7) **Medium (e.g. 14x14)**

   Large (e.g. 20x20)

3. **How would you like the words to be hidden in the grid?**
   (You may choose more than one option)

   a. **Diagonally**
   b. **Horizontally**
   c. **Vertically**
   d. **Overlapping**
   e. **Backwards**
   f. **Forwards**
   g. **All of the above**

4. **Would you prefer to be given a choice of different game difficulties (e.g. easy, medium and hard)?**

   Yes **No**

5. **To increase the difficulty of the game, I have listed some extra features. Do you think any of these additional features are necessary?**

   Please circle the options that would apply to you. Please also feel free to add any other ideas you think would help to increase the difficulty of the game.

a. **Increased grid size**
b. **Reduced time limit**
c. **Hidden word list**

……
……

6. **Would you prefer a printout option, this is an icon which creates a printout of the game?**

   Yes **No**

……
……

7. **How would you prefer to be scored, i.e. a fixed score is given for every word found?**

   ……
   ……

8. **What colour should the highlighter (used to by the cursor to highlight the missing word) be?**
   a. **Blue**
   b. **Pink**
   c. **Yellow**
   d. **Green**
   e. **Brown**
   f. **Multicolour**
   g. **Other:**
      ……
      ……

9. **Is a timed game a good option?**
   (If no is chosen, then please move onto question 12)

   Yes **No**

10. **How long would you like the game to last?**

a. **Less than 5 minutes**
b. **6 to 10 minutes**
c. **11 to 20 minutes**
d. **More than 20 minutes**

11. **Do you think there should be hint button?**

    Yes **No**

Summary of results from questionnaire 1

All questionnaires have been returned and I shall now summarise the results below. The options are given below the questions. The number in the parenthesis shows how many users choose that option.

1. **Have you ever played word search before? If yes how would you rate the gameplay?**

Yes (9) No (1)

Excellent (1), Quite good (8), 50/50 (1), Quite bad (0), Terrible (0)

2. **What is your preference of grid size, for a list of 5 words?**

Small (0) Medium (6) Large (4)

3. **How would you like the words to be hidden in the grid?**

Diagonally (2), Horizontally (3), Vertically (2), Overlapping (1) Backwards (0), Forwards (1), All of the above (7)

4. **Would you prefer to be given a choice of different game difficulties (e.g. easy, medium and hard)?**

Yes (10) No (0)

5. **Do you think any of these additional features are necessary?**

Increase grid size (4) reduced time limit (8) Hidden word list (3)

6. **Would you prefer a printout option?**

Yes (8) No (2)

7. **How would you prefer to be scored, i.e. a fixed score is given for every word found?**

This was an open question and the general feedback was that most users would prefer to be given a fixed score for each word found. However, some of the other users also believed that bonus points should be awarded for more difficult words.

8. **What colour should the highlighter be?**

Blue (4), Pink (0), Yellow (3), Green (0), Brown (0), Multicolour (2), Red (1)

9. **Is a timed game a good option?** (If no is chosen, then please move onto question 12)

Yes (9), No (1)

10. **How long would you like the game to last?**

Less than 5 minutes (1), 6 to 10 minutes (3), 11 to 20 minutes (2), more than 20 minutes (3)

11. **Do you think there should be a hint button?**

Yes (6) No (4)

Analysis of questionnaire 1 results

1. All but one of my users have played word search before and they rated it as “Quite good”. This shows that my users enjoyed playing a version of a word search game and suggest that they are happy with the current features involved.
2. The majority of users preferred a medium sized grid, which will be approximately 14(L) x14 (W).
3. 70% of users wanted the words to be hidden in all directions.
4. All 10 users would prefer to be given a choice of different game difficulties. It would probably be best to stick with the categories suggested as easy, medium and hard.
5. The majority of user wanted to see a reduced time limit feature, if the hard mode is selected. The minority said a hidden word list would be a good idea to the idea of an increased grid size. I face the problem of developing a very difficult game if all these listed features are included in the game.
6. A print out option is a hand feature that 80% of user agreed to.
7. This was an open question and the general feedback was that most users would prefer to be given a fixed score for each word found. However, some of the other users also believed that bonus points should be awarded for more difficult words.
8. My user would like to see a blue colour used to highlight the word
9. Nearly all user agreed to the idea of a timed game
10. The results were so close that I feel I need to refine the results by asking another questions in the second questionnaire
11. There should be a hint button, as 60% of user would like to see one.

Second Questionnaire

The purpose of conducting a second interview is to refine and collect new data, which will help me to fully understand my users’ needs. I have included an image of a radio button and a checkbox below question 7 to clarify any misunderstanding. The questions I will ask my users are below:

1. **How large should the screen size for the game be (the standard screen size of a monitor is 1280x1024)?**
   - 600 x 500 (Small approximately 1/3 of the monitor size)
   - 700 x 600 (Medium over a half the size of the monitor size)
   - 800 x 800 (Large approximately 3/4 of the monitor size)
   - Other: ……

2. **What colour would you like to see used for the background?**
   - White
   - Brown
   - Purple
   - Grey

3. **When a word is found what colour would you like the words to be highlighted?**
   - Red
   - Yellow
   - Green
   - Brown
   - Pink
   - Purple
   - Do not mind

4. **Would it be a good idea to be awarded bonus points, if the player wins the game before the timer reaches zero?** (If you choose no, then please move onto question 6)

   Yes No

5. **How many bonus points should the player be awarded for the remaining game time?**

   Each Minute: ……

   Each Second: ……

6. **How many points should the player score for finding a word?**
   - 10 points
   - 20 points
   - 30 points
   - Other ……

7. **How long would you like the game to last for the following game difficulties: Easy, Medium and Hard?**

   Easy: ……

   Medium: ……

   Hard: ……

8. **Sound effects can often make gameplay more enjoyable, would a beep sound to confirm a found word be of your interest?**

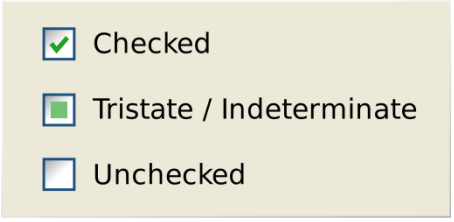
   Yes No

9. **Should the player be allowed to clear their highlighted choices by clicking on a "Clear" button?**

   Yes No

10. **How should the age, game difficulty and theme be selected?**
    (Please select only one from each category, if you are unsure of what a checkbox or radio button is then see the image below)

Radio Buttons or Checkboxes



Results from second questionnaire

Once again the frequency of the results are shown in parenthesis

1. **How large should the screen size for the game be (the standard screen size of a monitor is 1280x1024)?**
   - 600 x 500 (1)
   - 700 x 600 (4)
   - 800 x 800 (5)
   - Other (0)

2. **What colour would you like to see used for the background?**
   - White (3)
   - Brown (1)
   - Purple (1)
   - Grey (5)

3. **When a word is found what colour would you like the words to be highlighted?**
   - Red (7)
   - Yellow (0)
   - Green (1)
   - Brown (0)
   - Pink (0)
   - Purple (1)
   - Do not mind (1)

4. **Would it be a good idea to be awarded bonus points, if the player wins the game before the timer reaches zero?** (If you choose no, then please move onto question 6)
   Yes (10) No (0)

5. **How many bonus points should the player be awarded for the remaining game time?**
   This was an open question and so the average of the data will be stated below
   Each Minute: 21
   Each Second: 6 (to the nearest integer)

6. **How many points should the player score for finding a word?**
   - 10 points (3)
   - 20 points (7)
   - 30 points (0)
   - Other (0)

7. **How long would you like the game to last for the following game difficulties**
   Again this was an open question and so the modal results are shown below
   Easy: 15 minutes
   Medium: 10 minutes
   Hard: 5 minutes

8. **Would a beep sound to confirm a found word be of your interest?**
   Yes (8) No (2)

9. **Should the player be allowed to clear their highlighted choices by clicking on a "Clear" button?**
   Yes (9) No (1)

10. **How should the age, game difficulty and theme be selected?**
    Radio Buttons (2) Checkboxes (8)

Analysis of questionnaire 2 results

1. The most popular screen size for the game is 800 x 800. As the results were so close between 800 x 800 and 700 x 600, it is best to stick with the majority and decided upon the screen size when the interfaces have been designed.
2. Most users felt that the background of the screens should be grey.
3. It is obvious that all users wanted to see the found word highlighted in red.
4. Since all 10 users preferred to be awarded bonus points, I will need to work out the appropriate amount of bonus points to award, which follows in the question below.
5. The player should be awarded 21 bonus points for each minute and 6 bonus points for each second that is remaining on the timer once the game is completed.
6. The player should be given 20 points for finding a word
7. Again this was an open question and so the modal results are shown below
   Easy: 15 minutes
   Medium: 10 minutes
   Hard: 5 minutes
8. The beep feature is one to include in game, since 8 out of 10 users preferred to have that feature
9. Like with the beep feature, most users wish to see a clear button to clear their mistakes when selecting a word
10. The age, game difficulty and theme should be selected using checkboxes.

Interview Transcript

Interviewee: A school teacher – client
Interview Setting: In a school classroom. The interview was conducted at 4:00 PM on a Thursday afternoon

(Start of interview)

Interviewer: Do you play word search puzzles on a regularly basis?

Interviewee: No.

Interviewer: What about in lessons, do you use word search puzzles then?

Interviewee: Yes.

Interviewer: I would assume that you print off the word search puzzle and give it to your student to complete. Is this correct?

Interviewee: Yes that is the current system in place; we currently do not have an online or computer-based version of the game.

Interviewer: What would you say your students most enjoy about solving word search puzzles?

Interviewee: Most of my students often try to complete against each other in order to achieve the quickest time possible.

Interviewer: I remember trying to compete against the teacher at one stage, Ha-ha. Could you tell me who keeps track of the time?

Interviewee: That would have to be me; I must also ensure that nobody else is caught cheating

Interviewer: Which classes do you use word search puzzles?

Interviewee: GCSE and in a few a level classes.

Interviewer: It is very interesting to see that you uses word search puzzles in only a few a level classes, is there a reason for this?

Interviewee: The problem is that many a level students find these puzzles fairly simple and quite boring, which ultimately wastes lesson time. I do believe that word search puzzles can be made more challenging for A level students.

Interviewer: Very interesting, could you explain how a standard word search puzzle could be made more challenging.

Interviewee: I think by having sentences to replace the words in given list would help older (a level) student to memorise the key words better. It would allow a level students to think for themselves and to recall the missing words, which can be used as a useful revision tool.

Interviewer: That sounds like a brilliant idea!

Interviewer: How many words would you say is reasonable, for a 14 by 14 sized grid?

Interviewee: 8-10 words

Interviewer: I was kind of think along the same line as you – any justifications to why there should be only 8-10 words?

Interviewee: I have seen a few examples of some word search games and the general rule seems to be that the number of words required is in correlation with the grid size. However, having 14 words might take eat into the lesson time, so I think it is reasonable to stick with a lower amount.

Interviewer: Before the game begins, would you prefer to enter your name, so that it can be used in the game and so you can identify exactly which student won.

Interviewee: Yes, that sounds like a good idea, because it would make the game more personal.

Interviewer: As this word search game will be used to with young school children, how do you think I could provide enough help, such that any student should be to understand without any further assistance?

Interviewee: Well it may be a good idea to firstly add an instructions page and if you have time, it would help to have a tutorial mode or practice mode. All in all, it is necessary to have an instruction page though.

Interviewer: Is there anything else that you would like to see included in this word search game

Interviewee: It would be nice if you have some sort of way to reveal where all the words have been hidden at the end of the game.

Interviewer: Yeah, that sounds like a brilliant idea! I will try my best to include it. I'm sure your student would like the idea too.

Interviewer: Would you be happy if I developed a word search game, which you could you in your lessons. This word search game would be useful teaching aid.

Interviewee: Yes, I would appreciate having a helping hand.

Interviewer: Thank you so much, it has been a pleasure to interview you today.

Interviewee: No problem.

Analysis of interview results

The interview was very important to me as I could carefully understand what my client needed. The interview started steadily and I found out that my client uses word search puzzles in his lessons, but the word search puzzle he uses are paper based. This allows me to develop a computer based solution of The Word Search Game. The teacher states that the enjoyment the students get when playing the game is based on a timer or score feature.

In the middle of the interview, my client thinks it would be a reasonable to have a 14 x 14 sized grid, with 8 – 10 words. This is because the game shouldn't take too long to complete.

Towards the end of the interview, I indirectly suggest to my client if they would prefer to have their name displayed on screen and he agreed that it would be a good idea. My client then suggested a tutorial mode, which could be added to the game if there is sufficient time. My client said they would like to see where the words were hidden when the game ends.

The interview ended on a positive note and I reassured my client of the usefulness of the solution to his problem.

REQUIREMENT SPECIFICATION

User Requirements

Design Requirements:

a) The screen size will be large (800 x 800).
b) All screens will have a grey background.
c) The time interval for the timer will be 1000 milliseconds.
d) There should be an print label, on the game screen, to print out the form
e) The grid size will be 14 letters wide by 14 letters length.
f) There will be a checkbox for the different game difficulties
g) There will be a checkbox for the different age groups
h) There will be a checkbox for the different game themes
i) The user should be able to uncheck any checkboxes and recheck it if necessary.
j) There should be a clear button on the game screen, to clear any mistakes made.
k) There should be an instructions screen.

Input Requirements:

a) The player can go back to the menu page from the instructions screen, by clicking on the menu button.
b) The player can read the instructions by clicking on the instructions button.
c) The user can enter their name in the textbox – only letters, spaces, full stops and hyphens should be allowed.
d) The player can click the quit button, which must then prompt the player with a verification question. For example, “Are you sure you want to leave?”
e) The player can then click “yes” to exit and “no” to cancel.
f) The player can click on a letters in the grid of words to find.
g) The player can click on the clear button.

Processing Requirements:

a) The player is given 15 minutes and zero seconds if the easy game difficulty is selected. The player is given 10 minutes and zero seconds if the medium game difficulty is selected. The player is given 5 minutes and zero seconds if the hard difficulty is selected.
b) Every time the value of the timer changes, the system should check if the value of the timer equals zero, or if all the words have been found.
c) When the button “Start” is clicked, the timer should begin its count down and the grid should be filled with random letters and words.
d) When the printout icon is clicked the game should display the print dialogue.
e) When the instructions button is clicked the current form will close and the instructions screen will open.
f) When the “Play” button is clicked the current form will close and the game screen will open.
g) Every time a new letter is clicked in the grid, the system will check whether a word has been found.
h) When a letter is clicked in the grid, its colour will turn blue.
i) When a word is found, the colour of each letter in the found word will turn red.
j) Award 20 points for each word found.
k) When all the words have been found, the user should be awarded 21 bonus points for each minute and 6 points for each second that remains on the clock.

l) The system must record the number of words that have been found.

Output Requirements:

a) a message box should be displayed when the player wins the game or when time is up
b) A beep sound when a word is found
c) The user's score
d) The words to find list
e) The remaining game time
f) All the hidden when the game ends – Requested by my client in the interview

Hardware and Software requirements:

Hardware requirements:

1. A 1.6GHz or faster processor
   - This is required because it allows the end user to be able to use the system without any delay
2. 1024 MB RAM (1.5 GB if running in a virtual machine)
   - Minimum amount required for visual basic 10.0
3. 200 MB of available hard-disk space
   - This should be enough for The Word Search Game
4. Speakers or headphones
   - This is required so users can hear the sound effects when a word is found
5. Printer
   - Required to print out the game screen
6. Keyboard and mouse or other pointing device
   - Need for the user to click on different parts of the screen
7. 5400 RPM hard drive
   - Minimum amount required for visual basic 10.0

Software requirements:

1. Windows xp, windows 7, windows 8 or windows 10
   - Operating system needed to run visual basic 10.0
2. Visual basic 10.0
   - Software required to run the game

User review of requirements:

All my users have agreed to the specification above and are happy with the requirements stated.

Signatures:

Design – Nature of the solution

Design objectives

This stage continues from the user requirements, but with extra detail. I noticed if my users could click on any letter in the grid of words to find, then there is a problem that my users can cheat by clicking on letters that are in different locations to find the missing word. (See the example below, the word to find is CAT)

C X X X X X X X T
X X X U X X A X X
X A X X Z X X X X

In order to overcome this problem I have added additional processing requirement, which restrict my users to only a small space of clickable letters once the first letter has been clicked. I will discuss and agree my new consideration with my user.

Design Requirements:

a) The screen size will be large (800 x 800).
b) All screens will have a grey background.
c) The time interval for the timer will be 1000 milliseconds.
d) There should be an print label, on the game screen, to print out the form
e) The grid size will be 14 letters wide by 14 letters length.
f) There will be a checkbox for the different game difficulties
g) There will be a checkbox for the different age groups
h) There will be a checkbox for the different game themes
i) The user should be able to uncheck any checkboxes and recheck it if necessary.
j) There should be a clear button on the game screen, to clear any mistakes made.
k) There should be an instructions screen.

Input Requirements:

a) The player can go back to the menu page from the instructions screen, by clicking on the menu button.
b) The player can read the instructions by clicking on the instructions button.
c) The user can enter their name in the textbox – only letters, spaces, full stops and hyphens should be allowed.
d) The player can click the quit button, which must then prompt the player with a verification question. For example, “Are you sure you want to leave?”
e) The player can then click “yes” to exit and “no” to cancel.
f) The player can click on a letter in the grid and multiple times.
g) The player can click on the clear button.

Processing Requirements:

a) Determine if all the words have been found or the timer reaches zero – This signals the end of the game, see the output requirement below
b) When the button “Start” is clicked, the timer should begin its count down and the grid should be filled with random letters and words.

c) When the “Start” button is clicked, the word should be tested to see if the fit within the grid boundary and then be placed into that tested location. The words should be placed in all direction: North, East, South, West, Northeast, Northwest, Southeast, and Southwest.
d) When the printout icon is clicked the system should display the print dialogue.
e) When the instructions button is clicked the current form will close and the instructions page will open.
f) When the “Play” button is clicked the current form will close and the game screen will open.
g) Every time a new letter is clicked in the grid, the system will check whether a word has been found.
h) When a word is found, change the colour of each letter in the found word to red.
i) When the first letter in the grid is clicked, all other labels in the grid should be disabled, except the 8 surrounding letters.
j) When the second letter is clicked, all other labels in the grid become disabled, except for the next letter in the direction the player wishes to click.
k) When a letter is clicked in the grid, its colour will turn blue.
l) Award 20 points for each word found.
m) When all the words have been found, the user should be awarded 21 bonus points for each minute and 6 points for each second that remains on the timer.

Output Requirements:

a) a message box should be displayed when the player wins the game or when time is up
b) A beep sound when a word is found
c) The user’s score
d) The words to find list
e) The remaining time
f) All the hidden when the game ends – Requested by my client in the interview

User agreement and feedback

My users have reviewed my design specification and have agreed to it by signing below:

……

Module design

I will modularise The Word Search Game in order to break down the problem into smaller sections which can then be easily completed by the programmer. I will now describe my modules below and what follows is a diagram which illustrates how the different modules fit and work together to develop The Word Search Game.

Module 1: Loading the game

The purpose of this module is to prepare the game for the player. The player will input data by clicking on 3 different checkboxes from the 3 different categories. Some validation will have to be coded, so that the user has correctly input the right data to be used in the game. This module will contain the following tasks:

- Check to see if the word fits in the grid, if the word fits then an algorithm should run to place each letter of the word into the location tested
- Fill the blank spaces in the grid with random letters
- Check if each word filled in the grid is different

Module 2: Playing the game

This module contains the algorithms that are used to update the different controls, whilst the user is playing the game. There will be many different algorithms, so it is important that the algorithms in this module are efficient; as the user will notice lags when playing the game. The features required in this module are:

- Updating the score when a word is found
- Detecting when the user performs a mouse clicks.
- Displaying the words to find
- Pop the content from the stack whilst changing the colour of the letters to red – This should result in the whole word being highlighted in red.
- Clearing the mistakes when the player clicks on the clear button.

Module 3: Ending the game

This purpose of this module is to carry out the tasks for ending the game. The algorithms in this module will main output data, as this is what usually occurs when a game ends. This module will contain the following task:

- Show summary algorithm – this will display the player's score.
- Detecting and stopping the game's timer.
- Detecting when the user has found all the words.
- Verify if the player would like to end the game and closes the form if the player selects "yes".
- Show where all the words have been hidden in the grid.

Module 4: Instructions

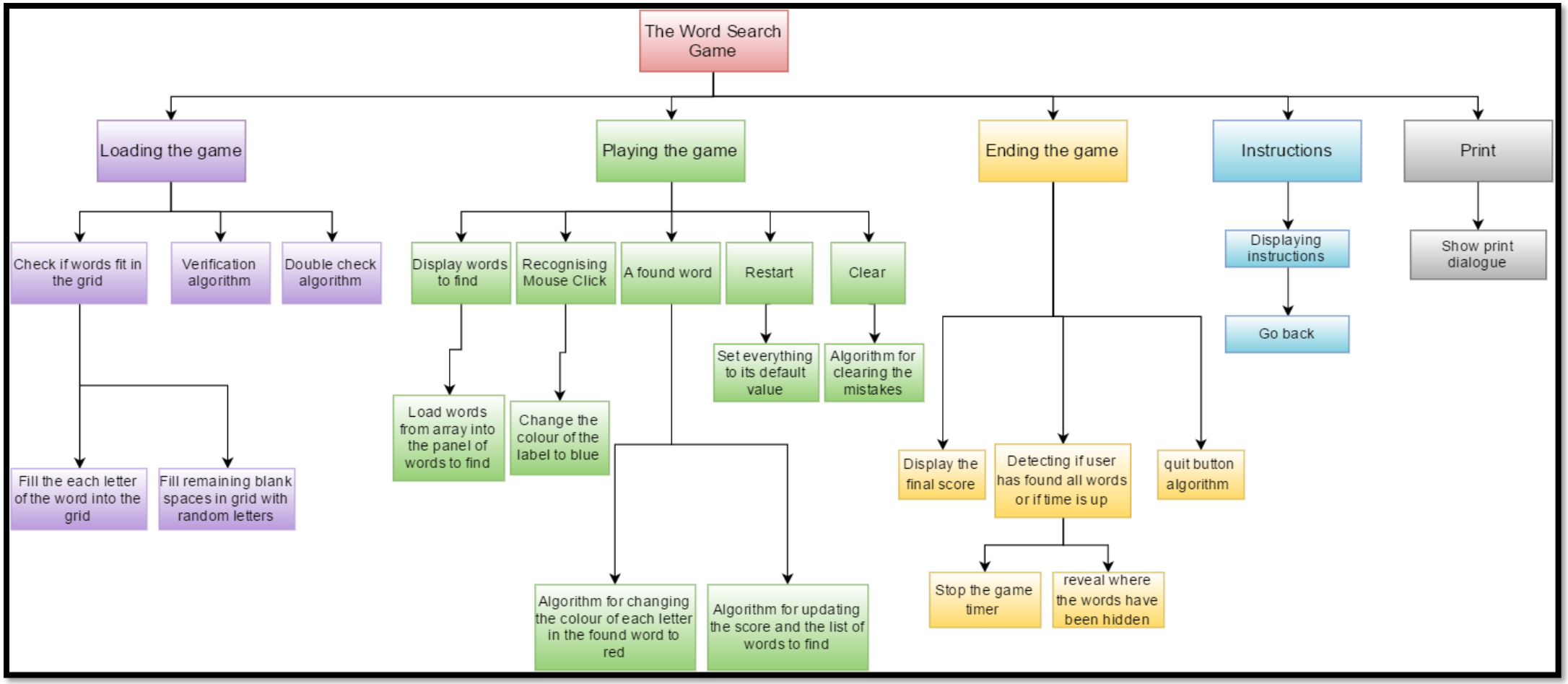
This is a very simple and short module, which will be concerned with displaying the instructions on screen. This module will include the following:

- Code for going back to the menu screen

Module 5: Print

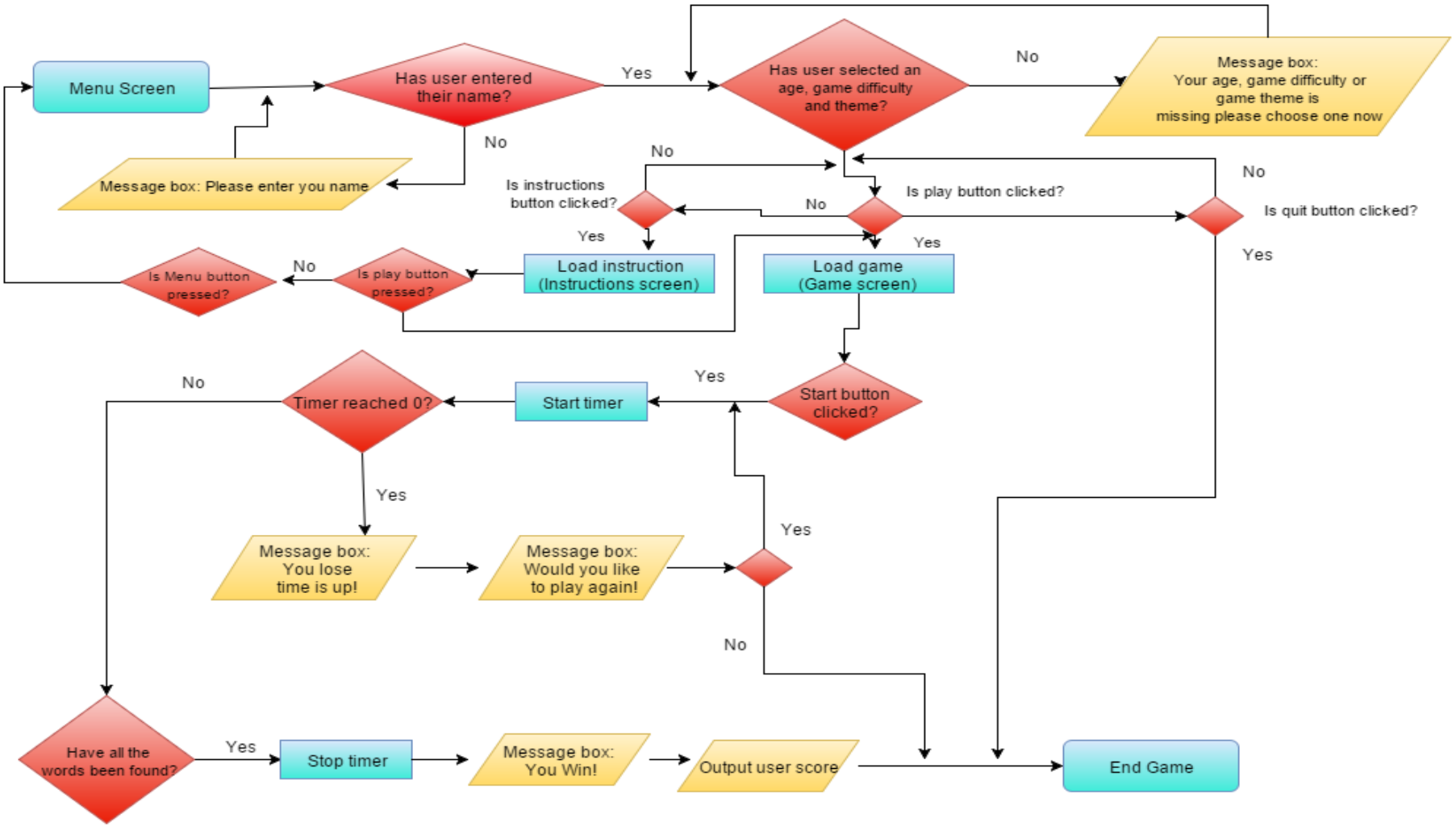
This is the final module in The Word Search Game and involves displaying the print dialogue when the player clicks on the print label.

Modular design for The Word Search Game



System flow diagram

The Diagram shows how information flows around the system. The input and output are show in the yellow boxes, the red diamonds demonstrate a question being asked by the system and finally the blue boxes show an action taking place.



Interface Design

I will be using this section to produce some design ideas which I will present to my client. I will use the same font throughout, which is, Microsoft Sans Serif. Data is to be entered into the system using mouse clicks and keypad to key in the user's name. For example, the user may click on button.

Interface Design 1: Menu screen

The initial screen the user is presented with is the menu screen, named Frm_MenuScreen, which will have 3 buttons: Quit Instructions and Play. This screen collects information from the user, which will then be processed and used to provide the right game settings for the user. The rectangular textbox will allow the user to enter their name. This text box will need to be validated, so that when the user clicks the "Play" button. This form screen is where the most of the input will be taking place.

The background colour will be dark blue and the labels white. The title is yellow so that is clear and easy to see.

Validation: A presence check will be performed on the text box field once the play button is clicked; this is to ensure that the player has entered a name. Furthermore, the presence check will also check that only one radio button is clicked and only one check box is clicked on in each row. A character check will ensure that the players name is only in letters.

This table lists the properties of the objects for this screen.

Object	Property	Setting
Frm_MenuScreen	Back colour	Blue
	Text	Menu Screen
	Size	800,800
cmdQuit	Text	Quit
cmdInstructions	Text	Instructions
cmdPlay	Text	Play
LblTitle	Forecolour	Yellow
	Text	The Word Search Game
	Font	Times New Roman
	Font size	16 pt.
LblName	Forecolour	White
	Text	Name
LblTheme	Forecolour	White
	Text	Theme
LblAge	Forecolour	White
	Text	Age
LblGame Diffictulty	Forecolour	White
	Text	Game Difficulty
Radmaths	Checked	False
RadScience	Checked	False
RadEnglish	Checked	False
CkbxYoungAge	Checked	False
	Text	<10
CkbxMediumAge	Checked	False
	Text	11 – 15
CkbxEldestAge	Checked	False
	Text	16+
CheckboxEasy	Text	Easy
CheckboxMedium	Text	Medium
CheckboxHard	Text	Hard

Menu Screen Interview

I plan on interviewing my client in order to receive feedback on my initial design creation. The interview will help the client and me to agree on the same idea, so that the correct design is implemented. I plan on asking the following questions:

- What do you like about the design looks and how could it be improved (if necessary)?
- How could the game functions be improved (if necessary), e.g. buttons and instructions?
- Are you happy with the size of the form?

If any additional questions are raised by me, then I would include then directly below the question asked. The final design idea will be added directly below the interview and any changes to the lists the properties will be updated too.

What do you like about the design looks and how could it be improved (if necessary)?

It is simple and not overly crabbed. You could add an image as there is too much text on the screen which is boring.

What image could would you like me to include and where should it be placed on the screen?

Try to add an image of a magnifying glass, which should be placed below the title with a little bit of room

How could the game functions be improved (if necessary), e.g. buttons and instructions?

The background colour of the buttons could do with a lighter shade of grey

Are you happy with the size of the form?

There is a lot of empty space, so I feel the form is too large.

Could you tell me what size you would like the menu screen to be? (I will display the changes to the screen size to my user)

In the end, my users and client agreed that a screen size of 651 by 536 pixels is best.

Finally, are you satisfied with the different categories the user can choose from?

Yes!

Updated version of Frm_Menuscreen.

Updated row for frm_MenuScreen

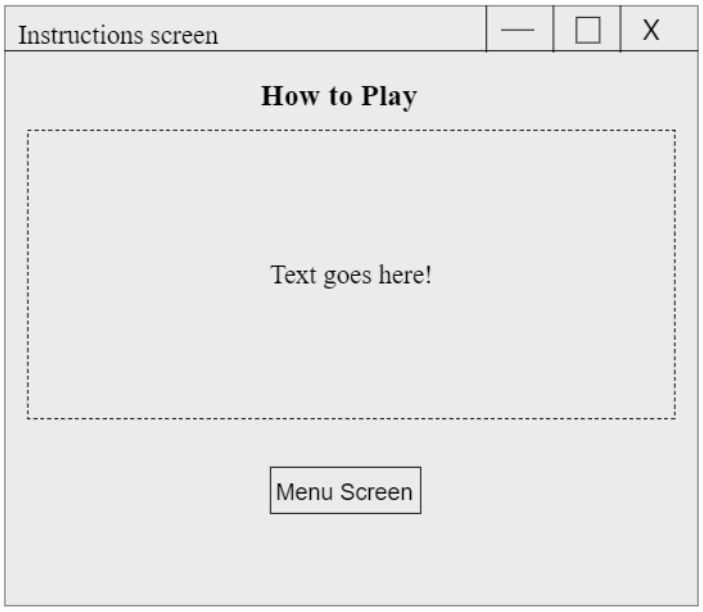
Object	Property	Setting
Frm_MenuScreen	Back colour	Control
	Size	651, 536

Interface design 2: Instruction screen

This screen is presented to the user when the instruction button on the previous screen is selected. The instructions screen will use a form named “frm_instructions”.

Below is the table of properties

Object	Property	Setting
Frm_instructions	Back colour	Control (Light grey)
	Text	Instructions Screen
	size	800, 800
Lbl_instructionsTitle	Text	How To Play
	Forecolour	Black
	Font	Times New Roman
	Location	241, 41
	Font size	12 pt.
	Bold	True
Lbl_InstructionsText	Font size	10 pt.
	Text	Rules: Locate the given words in the grid, running in one of eight possible directions horizontally, vertically, or diagonally. Good Luck!
CmdMenuScreen	Text	Menu Screen
	Location	49, 422



Instructions Screen: Interview

What do you like about the design looks and how could it be improved (if necessary)?

Yeah it's good, nice and simple; I don't think any design improvements are necessary

How could the game functions be improved (if necessary), e.g. buttons and instructions?

- The user should be able to progress to the game by clicking on a button and not have to go back to the menu screen.

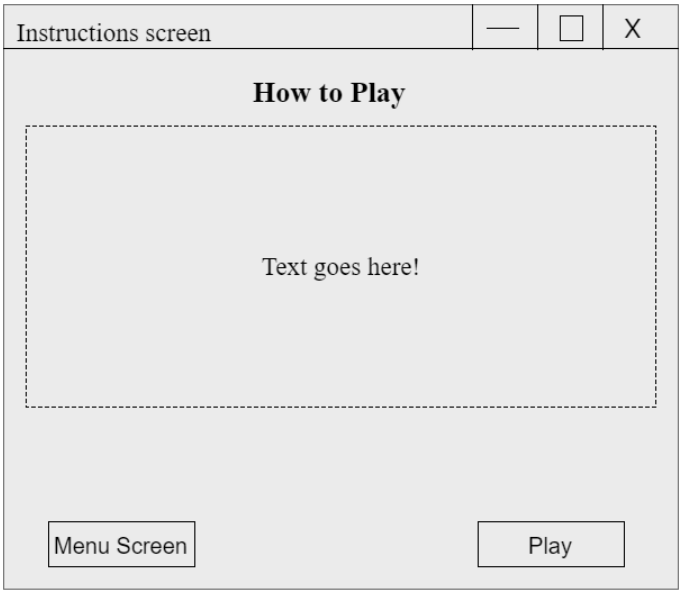
Are you happy with the screen size used?

No, the screens should be a lot smaller - like half the size.

I have now reduced the size of the screen to 497 pixels by 466 pixels. Are you happy with the size now?

Yes

Updated screen



New additions to the table of properties for frm_instructions

Object	Property	Setting
CmdPlayGame	Text	Play Game
	Location	426, 422
Frm_Instructions	Size	497, 466

Module 4: will now contain code for progressing onto the game screen.

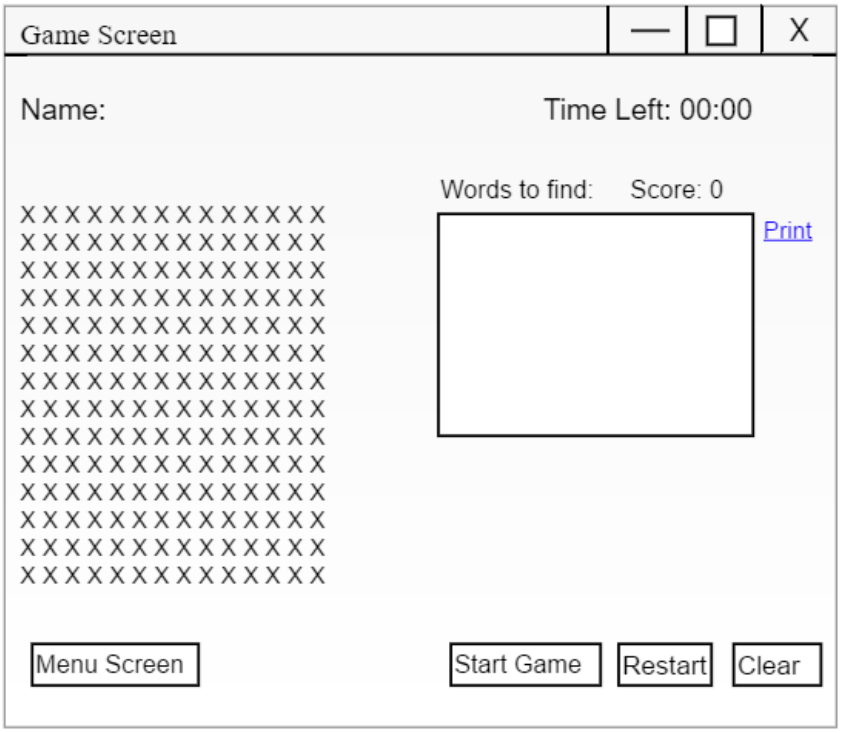
Interface 3: Game Screen

Game screen will use the form "frm_GameScreen", so that when the user clicks the Play button, the game screen is loaded. The purpose of the game screen is to provide the screen for the user to play the game and according their preferences which are saved from frm_MenuScreen.

This form will have a 14 by 14 grid displayed on the left hand side and there will be four buttons which sit below the grid and are all in line (horizontally). The background colour will be grey. The table of Properties is listed below.

Object	Property	Setting
Frm_GameScreen	Back colour	Control (Light grey)
	Text	Game Screen
	Size	651, 536
Lbl_Name	Text	Name:
	Forecolour	Black
Lbl_WordstoFind	Font size	8.25 pt.
	Text	Words To Find
LblScoreText	Text	Score:
	Location	491, 80
	Font size	8.25 pt.
LblScore	Text	0
	Location	535, 80
LblscoreSummary	Visible	False
TxtlblWordsToFind	Text	It's your job to find the missing words!

		There are only 8 words to find!
LinklblPrint	Text	Print
	Visible	False
CmdMenuScreen	Text	Menu Screen
	Location	42, 445
CmdStartGame	Text	Play Game
	Location	372, 445
CmdRestart	Text	Restart
	Location	457, 445
CmdClear	Text	Clear!
	Location	542, 445



Game Screen interview

What do you like about the appearance of the design and how could it be improved (if necessary)?

Everything, this is well designed! None to be specific.

How could the game functions be improved (if necessary), e.g. buttons and instructions?

None

Are you happy with the size of the form?

The form is too big. It should be adjusted to the same size, as used in the menu screen.

Are you happy with the changes in size?

Yes

Amendments to table of properties for Frm_Game screen

Object	Property	Setting
Frm_GameScreen	Size	651, 536

Error Message Design

Data input needs to be validated and the output data needs to be presented to the user. An example of this would be when the user forgets to enter their name in the textbox. The error message will look something like below:

If the user has not selected a theme, an age or a game difficulty theme then following error message box will be show:

Theme: “Error, you have not selected a theme yet. Please select a theme”

Age: “Error, you have not selected your age. Please select your age”

Game difficulty: “Error, you have not selected a game difficulty. Please select a game difficulty”

Design Requirements:

a) The screen size will be 651 by 531 pixels for the game screen and menu screen.
b) The screen size will be 497, 466 pixels for the instructions screen.
c) All screens will have a light grey background.
d) The time interval for the timer will be 1000 milliseconds.
e) There should be an print label, on the game screen, to print out the form
f) The grid size will be 14 letters wide by 14 letters length.
g) There will be a checkbox for: easy, medium and hard game difficulties
h) There will be a checkbox for: <10, 11-15 and 16+ age groups
i) There will be a checkbox for: maths, science and English themes
j) The user should be able to uncheck any checkboxes and recheck it if necessary.
k) There should be a clear button on the game screen, to clear any mistakes made.
l) There should be an instructions screen.

Input Requirements:

a) The player can go back to the menu page from the instructions screen, by clicking on the menu button.
b) The player can go to the game screen after reading the instructions, by clicking on the play button.
c) The player can read the instructions by clicking on the instructions button.
d) The user can enter their name in the textbox – only letters, spaces, full stops and hyphens should be allowed.
e) The player can click the quit button, which must then prompt the player with a verification question. For example, “Are you sure you want to leave?”
f) The player can then click “yes” to exit and “no” to cancel.
g) The player can click on a letter in the grid and multiple times.
h) The player can click on the clear button.

Processing Requirements:

a) Determine if all the words have been found or the timer reaches zero – This signals the end of the game, see the output requirement below
b) When the button “Start” is clicked, the timer should begin its count down and the grid should be filled with random letters and words.
c) When the “Start” button is clicked, the word should be tested to see if the fits within the grid boundary and then be placed into that tested location. The words should be placed in all directions: North, East, South, West, Northeast, Northwest, Southeast, and Southwest.
d) When the printout icon is clicked the system should display the print dialogue.
e) When the instructions button is clicked the current form will close and the instructions page will open.
f) When the “Play” button is clicked the current form will close and the game screen will open.
g) Every time a new letter is clicked in the grid, the system will check whether a word has been found.
h) When a word is found, change the colour of each letter in the found word to red.
i) When the first letter in the grid is clicked, all other labels in the grid should be disabled, except the 8 surrounding letters.

j) When the second letter is clicked, all other labels are disabled in the grid, except the next letter in the direction the player wishes to click.
k) When a letter is clicked in the grid, its colour will turn blue.
l) Award 20 points for each word found.
m) When all the words have been found, the user should be awarded 21 bonus points for each minute and 6 points for each second that remains on the clock.

Output Requirements:

a) A beep sound when a word is found
b) The user's score
c) The words to find
d) The remaining time
e) All the hidden when the game ends – Requested by my client in the interview

Signatures: Date: ……

Data Structure Design and Variable Table

Data structures and variables need to have sensible identifiers to avoid confusion. The Word Search Game will use the following text files stored in "The Word Search Game" folder on the USB. E:\Word Search Game\The Word Search Game\bin\Debug. The text files are called: *EnglishwordsYoungAge, EnglishwordsMiddleAge, EnglishwordsEldestAge, MathswordsYoungAge, MathswordsMiddleAge, MathswordsEldestAge, SciencewordsYoungAge, SciencewordsMiddleAge and SciencewordsEldestAge.* The text files are stored in the debug folder, because the computer will need to locate these files when debugging the game.

The game will also use a stack to hold the number of the label that has been clicked on. I choose to use a stack, because of the LIFO ordering.

The will be no variables used in Frm_Instructions

Below are the variables that will be used in Frm_MenuScreen:

Variable Name	Data Type	Size	Description
CheckFlagEasy	Boolean	0(False) - 1(True)	Flags up when the checkbox easy is checked.
CheckFlagMedium	Boolean	0(False) - 1(True)	Flags up when the checkbox Medium is checked.
CheckFlagHard	Boolean	0(False) - 1(True)	Flags up when the checkbox Hard is checked.
Ckbxflagmaths	Boolean	0(False) - 1(True)	Flags up when the checkbox maths is checked.
CkbxlflagScience	Boolean	0(False) - 1(True)	Flags up when the checkbox science is checked.
CkbxflagEnglish	Boolean	0(False) - 1(True)	Flags up when the checkbox English is checked.
CkbxflagYoungAge	Boolean	0(False) - 1(True)	Flags up when the checkbox <10 is checked.
CkbxflagMiddleAge	Boolean	0(False) - 1(True)	Flags up when the checkbox 11-15 is checked.
CkbxflagEldestAge	Boolean	0(False) - 1(True)	Flags up when the checkbox 16+ is checked.
Player_name	String	0 – 65536	Stores the player's name, so that it can be used in the game screen.
TempFlag	Boolean	0(False) - 1(True)	A temporary flag, which is used as a local variable in subroutines, to detect if a series of conditions are true or false.
intmsgbx	Integer	4 bytes	Stores the number returned from the message box function

Below are the variables that will be used in Frm_GameScreen:

Variable Name	Data Type	Size	Description
Gridarray	String	2 dimensions (13,13), 196 elements	An array that stores the content of each labels in the grid.
arrEnglishWordsYoungAge	String	1 dimensional 10 elements	Stores the words in the EnglishwordsYoungAge text file.
arrMathsWordsYoungAge	String	1 dimensional 10 elements	Stores the words in the MathswordsYoungAge text file
arrScienceWordsYoungAge	String	1 dimensional 10 elements	Stores the words in the SciencewordsYoungAge text file
arrEnglishWordsMiddleAge	String	1 dimensional 10 elements	Stores the words in the EnglishwordsMiddleAge text file
arrMathsWordsMiddleAge	String	1 dimensional 10 elements	Stores the words in the MathswordsMiddleAge text file.
arrScienceWordsMIddleAge	String	1 dimensional 10 elements	Stores the words in the SciencewordsMiddleAge text file.
arrEnglishWordsEldestAge	String	1 dimensional 10 elements	Stores the words in the EnglishwordsEldestAge text file.
arrMathsWordsEldestAge	String	1 dimensional 10 elements	Stores the words in the MathswordsEldestAge text file.
arrScienceWordsEldestAge	String	1 dimensional 10 elements	Stores the words in the SciencewordsEldestAge text file.
arrunscramble	String	2 dimensions (13,13), 196 elements	Stores the position of the randomly generated characters in the grid.
intCount	Integer	4 bytes	A local variable used in subroutines as a loop counter
Count	Integer	4 bytes	A local variable used in subroutines as a second counter
i	Integer	4 bytes	A local variable used as a counter in a for… next… loop
intscore	Integer	4 bytes	A global variable used to store the value of the player's score
intTimeBonus	Integer	4 bytes	Stores the player's time bonus points as an integer, which gets added to the players score at the end of the game
TimeEnd	Date (DateTime)	8 bytes	Stores the calculated

			end time
Timediff	Timespan	8 bytes	The difference Between Start and End Times
intItemsFound	Integer	4 bytes	Stores the number of words found.
arrItems	None	1 dimension, 8 elements	Array of labels, which display the individual words to find.
Lbltemp	None	None	A label, which copies the properties of the label (letter) that has been clicked on in the grid.
tempstr	String	0 – 65536	Temporarily holds a string
lblsel	String	0 – 65536	Concatenate a sequence of letters
Loopflag	Boolean	0(False) – 1(True)	A flag used to control the main loop.
Checkflag	Boolean	0(False) – 1(True)	A local variable that flags up when the word fits in the grid.
Random X	Integer	4 bytes	A randomly generated number used to represent a random row in the grid
Random Y	Integer	4 bytes	A randomly generated number used to represent a random column in the grid
Strscramble	String	0 – 65536	Stores each letter in the alphabet as a string
Randgen	Integer	4 bytes	A random number, ranging from 0 to 25
northpos	Integer	4 bytes	The label (letter) that is north of “lbltemp”
southpos	Integer	4 bytes	The label (letter) that is south of “lbltemp”
eastpos	Integer	4 bytes	The label (letter) that is east of “lbltemp”
westpos	Integer	4 bytes	The label (letter) that is west of “lbltemp”
northwestpos	Integer	4 bytes	The label (letter) that is northwest of “lbltemp”
northeastpos	Integer	4 bytes	The label (letter) that is northeast of “lbltemp”
southwestpos	Integer	4 bytes	The label (letter) that is southwest of “lbltemp”
southeastpos	Integer	4 bytes	The label (letter) that is southeast of “lbltemp”

Test Strategy

I intend to test The Word Search Game and will take screenshot of the testing. There will be different stage of the test strategy, to ensure my end users that the solution works will no bugs.

Alpha, Beta and Acceptance testing

Firstly, I will conduct alpha testing to test the fundamental functions of the system. Alpha testing will contain white box testing, so that each line of code is checked for errors. As part of alpha testing I will need to test the different algorithms of the solution and whether or not they produce the correct results. The part of the alpha testing will be carried out during software development, were I test if the algorithm produce the correct results.

I will use beta testing to test the inputs and output of The Word Search Game. I will be inputting different types of data to see if I get the outputs I expect.

After alpha testing has been completed, I will then carry out beta testing so I can get some feedback from my end-users. Beta testing will involve asking potential end users the questions below to see if the game meets the requirements stated. The potential end users will answer the questions below; any feedback will be written by the end user and then reported back to me, were then I can refine the game.

Question number	Question	Answer (Yes/No)	Comments
1	Where you able to start the game?		
2	Do all the buttons and checkboxes work?		
3	Are you able to print out the game?		
4	Are you happy with the colour used?		
5	Is there sufficient help available?		
6	Are the messages helpful and clear?		
7	Is it easy to find a word?		
8	Is the game too difficult?		
9	Is the game too easy?		
10	Is there enough time to play the game?		
11	Were you able to check and then uncheck a checkbox on the menu screen?		
12	Is it easy to navigate around the game?		
13	Could you exit the game?		
14	When you click the “Start” button, does the game randomly generate a new list of words?		
15	Are you happy with the sound effects used, for example when a word is found?		
16	If you have any other questions that have not been covered, please add your views to the comments section on the right.		

The final testing in this sub-section is acceptance testing, which can only commence after the completion of alpha and beta testing. Acceptance testing will involve me demonstrating to my users and client that the system works as agreed in the design specification. The client and users will be able to see the testing and so can state whether or not he agrees or not. I will now list the features that need testing to meet the client and users' requirements.

Module 1:

a) A click on the play button should take you to the game form, once all the required fields are completed.
b) The user can only enter letters and use the hyphen symbol in the textbox, symbols and digits are not permitted.
c) All 8 words appear once in the grid.
d) Each word is placed in the following directions once: North, East, South, West, Northeast, Northwest, Southeast and Southwest.

Module 2:

a) Click on 'Start game' to begin game
b) Use a mouse click to click on a letter in the grid
c) When the first letter is clicked (in the grid), all other letters should be disabled except for the 8 letters that surround the clicked letter.
d) After clicking on the next letter, you find you can only travel in one direction.
e) When a letter is clicked in the grid, its colour should change to blue.
f) If a word is found, every letter in the found word should change to red and remain red.
g) No word can be found more than once (this can be checked by looking at the player's score).
h) Clicking the clear button should enable all the letters in the grid and change the colour of the selected letters to black.
i) Double-clicking on a letter twice should not allow you to gain points for finding a word. E.g. CCAT instead of CAT.
j) User should be awarded 20 points for every word found
k) The timer decrements for every second in real time.

Module 3:

a) When the timer reaches zero, the game ends immediately and a message box is displayed showing "Time's up".
b) Click 'quit' to exit game
    i. Yes to confirm the exit and the window will close
    ii. No will close the message box.
c) All the words, including the found words, should be revealed at the end without the other random letters
d) The users final score summary will be shown immediately when the game ends, this summary will also show any bonus points that have been awarded to the user.

Module 4:

a) Display the instructions, upon clicking the instructions button
b) Progress onto the Game Screen, upon clicking the Play button
c) Return to the menu screen upon clicking the Menu Screen button

Module 5:

a) Display print dialogue
b) Print the content in the print preview

Below is the table the user will fill out to state whether the requirement was either met or not.

Module Number	Test (letter)	Success?	Comments
1	a		
	b		
	c		
	d		
2	a		
	b		
	c		
	d		
	e		
	f		
	g		
	h		
	i		
	j		
	k		
3	a		
	b		
	c		
	d		
4	a		
	b		
	c		
5	a		
	b		

Valid, Invalid, Extreme Data

I will need to test the three validation types, so that the data entered into the system is accepted if it is valid and not if it is invalid. Extreme data is input data that is on the boundary of acceptance and therefore it may or may not be accepted. I will fill the pass/fail column once the system is completed.

Function being tested	Test Number:	Input	Expected Results	Pass/Fail
Player's entering name	1	Letters	Valid – Data accepted	
	2	A symbol – except for hyphen	Invalid – Error message	
	3	numbers	Invalid – Error message	
	4	Nothing	Invalid - Error message	
	5	Letters and Numbers	Invalid – Error message	
	6	Letters and Symbols	invalid	
	7	Number and Symbols	Invalid – Error message	
	8	Letters, numbers and Symbols	Invalid – Error message	
Checkboxes on	1	Click one checkbox	Valid - data excepted	

Menu Screen	2	No checkboxes	Invalid – Error message
	3	Click on a checkbox then uncheck it and select another checkbox	Valid – Data accepted
Mouse click on letters in grid	1	User clicks on any letter in grid	Valid – Colour of the letter turn blue
	2	User clicks on letter not in grid	Invalid – nothing happens
	3	User clicks on a letter in the top row of the grid	Valid – data accepted
	4	User clicks on the top right letter in the grid	Extreme – data accepted
	5	User clicks on the bottom left letter in the grid	Extreme – data accepted
	6	User clicks on the bottom right letter	Extreme – data accepted
	7	User clicks on a letter on the edge of the grid	Extreme - data accepted
	8	User clicks on a letter in the bottom row of the grid	Valid – data accepted
	9	User clicks on a letter in the top row of the grid	Valid – data accepted
	10	User clicks on each letter of the missing word, in turn.	Valid – each letter of the word should change red and the system should beep.
	11	User double clicks on a letter	Extreme – data accepted
Displaying instructions	1	Clicking the instructions button	Instructions should be displayed
Returning from the instruction screen to menu screen	1	Clicking on menu button	Menu screen should be displayed
Progressing to the game screen from the instruction screen	1	Clicking on the play button	Game screen should be displayed
Leaving the game	1	Click on the quit button	Validation message “Are you sure you want to leave?”
	2	Click on yes	Game ends
	3	Click on no	Message box closes

Algorithms

The Word Search Game will have many algorithms which are in the 5 modules described on page 24. The algorithms will demonstrate how the different elements of the solution function to provide the desired outcome. I will describe my algorithms, in turn, using pseudo code. I will also use flow charts to show how the various algorithms in a module fit together to complete the module.

Module 1: Loading the game

- Verification algorithm – Checks if user has entered their name and checks if the user wishes to exit.
- Check to see if the word fits into the grid and then fill the characters of the word into the blank spaces (fill_word algorithm)
- Fill the remaining blank spaces with random characters (Scramble Algorithm)
- Check for Double Word algorithm (Validation check to determine whether the same word appears twice in the array)
- Read the text in the appropriate text file and store the content in an array.
- Load the words from the array into the panel box

Module 2: Playing the game

- Timer Ticks algorithm (Process and display the remaining time)
- Recognising mouse click
    a) Change the colour of the label to blue
- Found word
    a) Pop the contents from the stack whilst changing the colour of the label to red
    b) Increment the lblscoretext to the new score and update the word list, so that the found word says the word and found.
- Restart
    a) Set everything to its default value
- Clearing words
    a) Pop content from the stack, whilst changing the label colour to black.

Module 3: Ending the game

- Show summary algorithm
- Stop the timer
- Show hidden words algorithm

Module 4: Instructions

This module will contain the following; however, this module is not large enough to be classed as an algorithm.

- Display instructions screen
- Go back (menu screen)
- Go to game

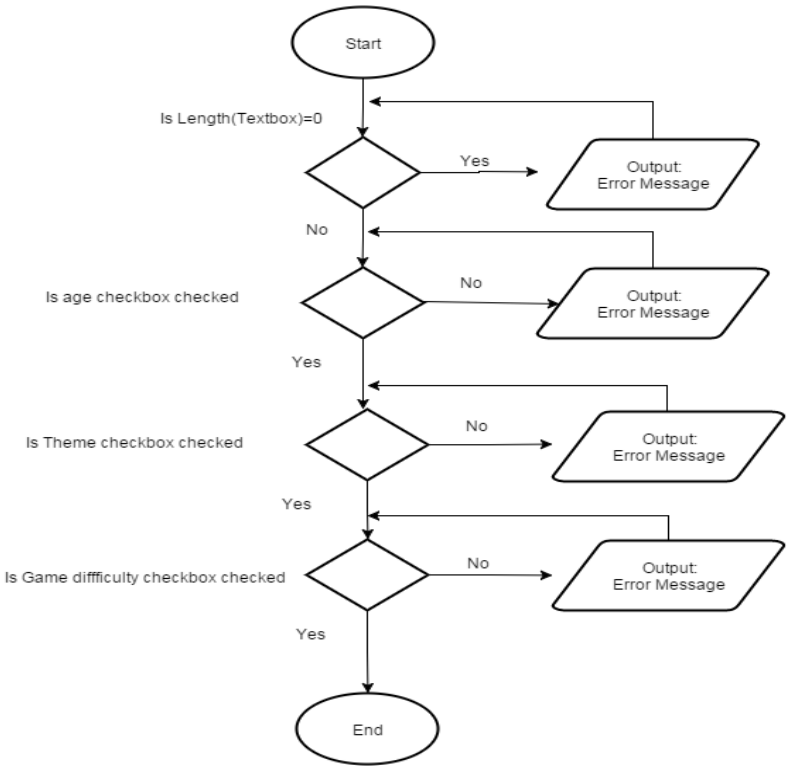
Module 5: Print

Again this module will contain; however, this module is not large enough to be classed as an algorithm.

- Show print dialogue

Module 1: Verification – Algorithms

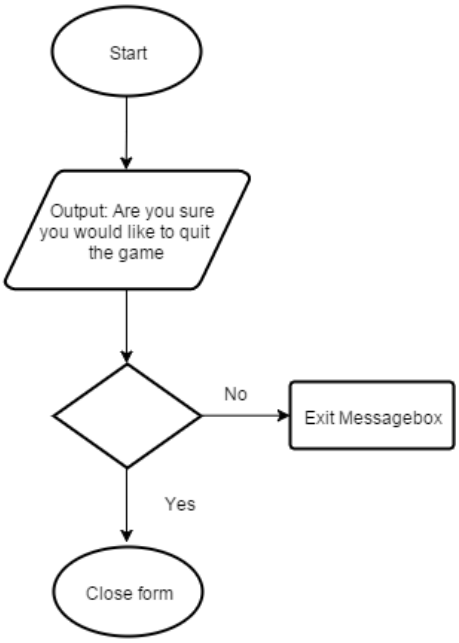
CmdPlay



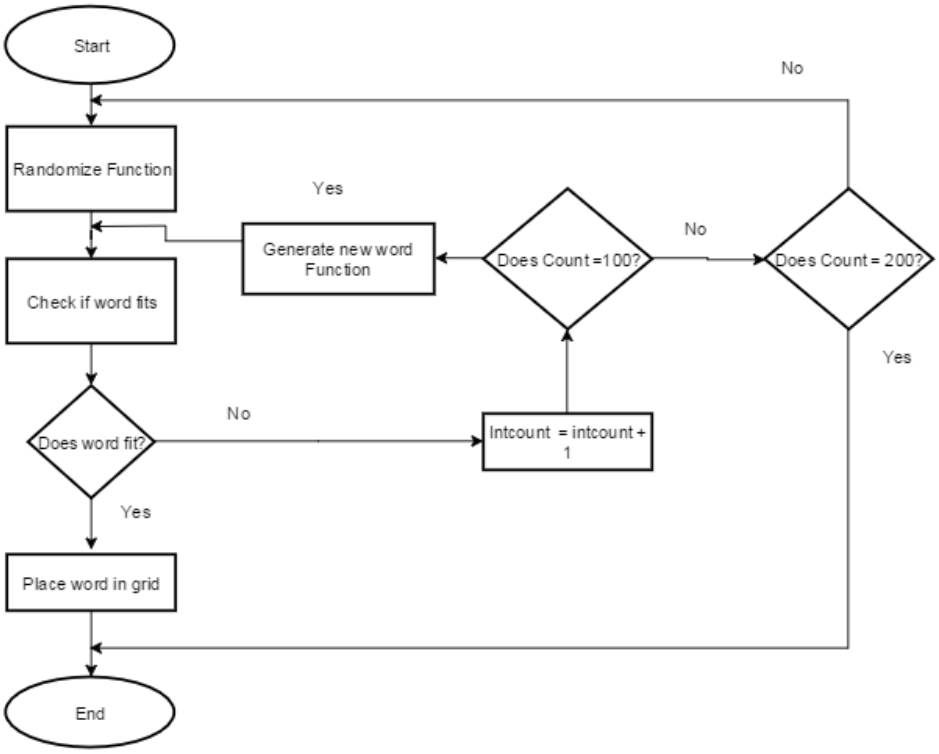
This validation runs when the play button is clicked on in frm_MenuScreen. It is a simple way of double checking that the user has selected the required fields, otherwise the game will not be correctly set up for the user.

CmdQuit

Again this is a simple verification method which just checks whether the user really wants to quit or if it was just a mistake.



Module 1:



(Figure 1): How the algorithms in this module relate to complete the module loading the game.

<u>Check to see if the word fits into the grid</u>

Grid array is an array which represents each location (letter) in the grid. RandomX and RandomY are randomised numbers which are declared in the subroutine. Random X and Random Y are random numbers between zero and (14 – length of word – 1). CharArr is an array which stores each character of the string to be filled in.

1. Count = 0
2. While count <= length of chararr - 1
3. If (GridArray(RandomX, randomY) <> " " And GridArray(RandomX, randomY) <> charArr(Count)) Then
4. checkflag = False
5. End If
6. Count = Count + 1
7. randomY = randomY + 1 (To fill a word in the East direction)
8. End While

I will now perform a dry run for the algorithm using the test word “IDLE”. chararr will hold the four characters and so will have a length of 4. Random X and Random Y will have already been generated and so for the test, so I will randomly choose a number for both Random X and Random Y.

Line number	Counter	Checkflag	Length of chararr	Random X	Random Y	Comments
00		True		8	5	
01	0					
02			6			True, so go to line 03
03						False, so move to line 06

06	1		
07		6	
08			Return to line 02
02			True, so go to line 03
03			False, go to line 06
06	2		
07		7	
08			Return to line 02
02			True, so go to line 03
03			False, go to line 06
06	3		
07		8	
08			Return to line 02
02			True, so go to line 03
03			False, go to line 06
06	4		
07		9	
08			Return to line 02
02			False, so move on next statement after loop.

Fill the characters of the word into the blank spaces.

Now if the word fits then the code will stop iterating and then this next loop is executed.

```
1.  Count = 0
2.  While Count  <= length of chararr  - 1
3.        GridArray(RandomX, randomY) = uppercase of character in chararr(count)
4.        Count += 1
5.        randomY += 1
6.    End While
```

Line number	Counter	Length of chararr	Random X	Random Y	GridArray (RandomX, Random Y)	Comments
00		4	8	5		
01	0					
02						True, so execute loop

03					I		
04	1						
05			6				
06							Return to line 2
02							True
03					D		
04	2						
05			7				
06							Return to line 2
02							True
03					L		
04	3						
05			8				
06							Return to line 2
02							True
03					E		
04	4						
05			9				
06							Return to line 2
02							False, so move onto next statement after loop

<u>Fill the remaining blank spaces with random characters</u>

Chararr is a local array which is currently holding each character of the alphabet, in-order.

```
1.  For row = 0 to 13
2.      For column = 0 to 13
3.        Dim RandIndex As Integer = RandGen.Next(0, 25)       (Generate New Random Letter)
4.         If GridArray(Row, Column) = " "   then
5.             GridArray(Row, Column) = Chararr(RandIndex)
6.              arrunscramble(row, column) = "Char"   (Stores the location of the randomly generated characters)
7.         End if
8.      Next Column
9.  Next Row
```

Again I shall perform a couple of dry runs to test the functioning of this algorithm, 4 passes should be enough.

Line number	Row	Column	RandIndex	Chararr (RandIndex)	Gridarray (Row,Column)	Arrunscramble (row,column)	Comments
01	0						
02		0					

03		2				
04						True
05			B	B		
06					Char	
07						End if
08						Return to line 02
02	1					
03		12				
04						True
05			L	L		
06					Char	
07						End if
08						Return to line 02
02	2					
03		15				
04						False
07						End if
08						Return to line 02
02	3					
03		20				
04						True
05			U	U		
06					Char	
07						End if
08						Return to line 02
02	4					
03		5				
04						False
07						End if
08						Return to line 02

Check for Double Word Algorithm

The purpose of this algorithm is to make sure that the same word does not appear twice in the grid. The algorithm will generate a new random number, so that each location in the array stores a different number. The random numbers stored in array intrandom used in this subroutine (see figure 1) will be used to access the word stored in the array arrsciencewordsEldestAge (). Since the numbers in array intrandom are different the words accessed in arrsciencewordseldestage must be different, which will see the grid only contain the word once.

1. While intrandom(2) = intrandom(1)
2. intrandom(2) = A random number between 0 and the length of the array
3. End While

4. While intrandom(3) = intrandom(1) Or intrandom(3) = intrandom(2)

5. intrandom(3) = A random number between 0 and the length of the array
6. End While

7. While intrandom(4) = intrandom(1) Or intrandom(4) = intrandom(2) Or intrandom(4) = intrandom(3)
8. intrandom(4) = A random number between 0 and the length of the array
9. End While

10. While intrandom(5) = intrandom(1) Or intrandom(5) = intrandom(2) Or intrandom(5) = intrandom(3) Or intrandom(5) = intrandom(4)
11. intrandom(5) = A random number between 0 and the length of the array
12. End While

13. While intrandom(6) = intrandom(1) Or intrandom(6) = intrandom(2) Or intrandom(6) = intrandom(3) Or intrandom(6) = intrandom(4) Or intrandom(6) = intrandom(5)
14. intrandom(6) = A random number between 0 and the length of the array
15. End While

16. While intrandom(7) = intrandom(1) Or intrandom(7) = intrandom(2) Or intrandom(7) = intrandom(3) Or intrandom(7) = intrandom(4) Or intrandom(7) = intrandom(5) Or intrandom(7) = intrandom(6)
17. intrandom(7) = A random number between 0 and the length of the array
18. End While

19. While intrandom(8) = intrandom(1) Or intrandom(8) = intrandom(2) Or intrandom(8) = intrandom(3) Or intrandom(8) = intrandom(4) Or intrandom(8) = intrandom(5) Or intrandom(8) = intrandom(6) Or intrandom(8) = intrandom(7)
20. intrandom(8) = A random number between 0 and the length of the array
21. End While

Read the text in the appropriate text file and store the content in an array

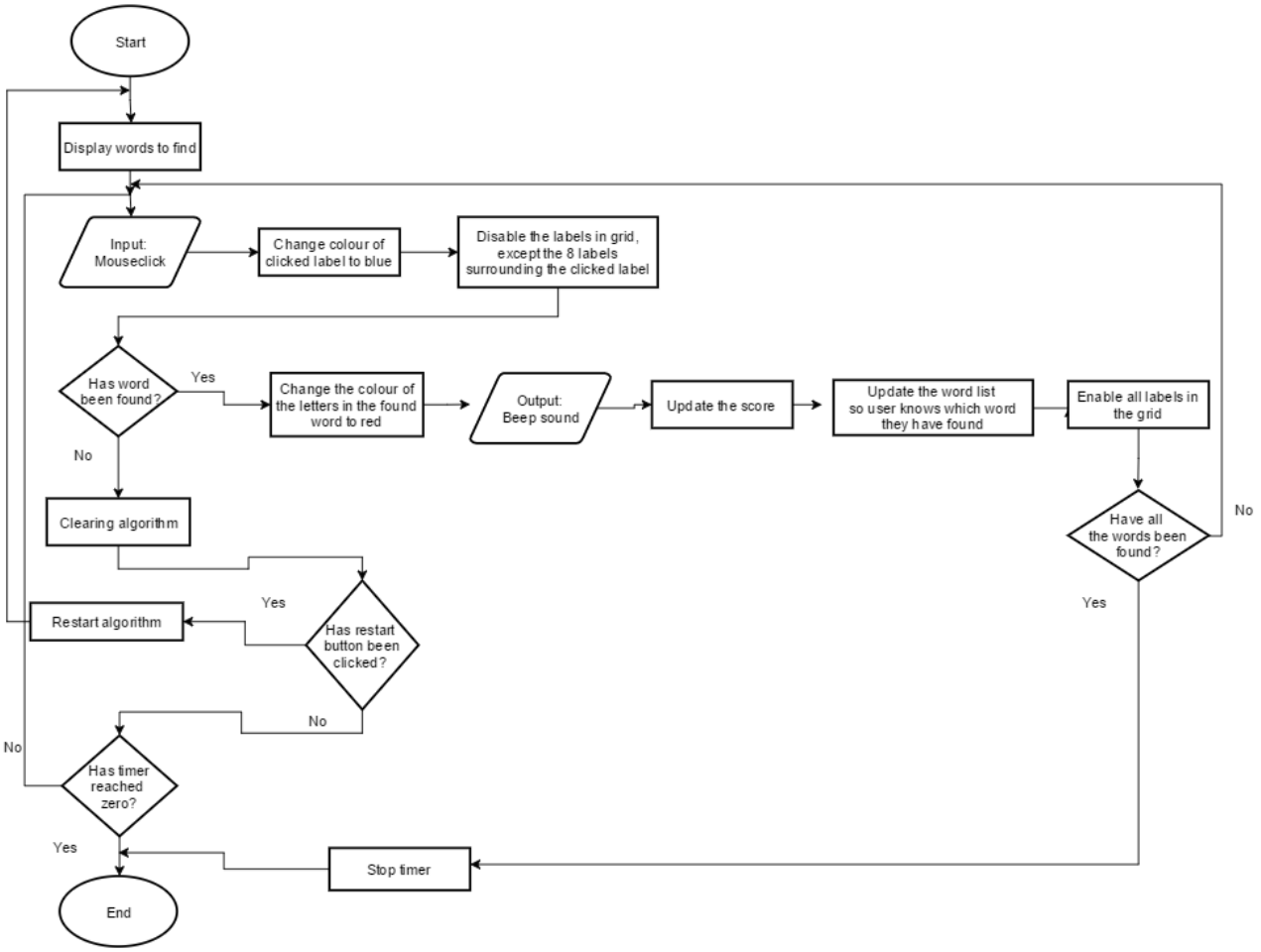
The following algorithm read from the EnglishwordsYoungAge text file and stores the word into the EnglishwordsYoungAge array. This array will execute if the user selects the English theme and selects the young age group. For example, if the user chooses a science theme and the eldest age group, then the SciencewordsEldestAge text file will open and the SciencewordsEldestAge array will be used. See page 37 for the data structure design.

1. Open file ("EnglishwordsYoungAge")
2. Namefile() reads a word from file and stores it in its array, until the end of the file
3. Close file
4. For i = 0 to length of arrenglishwordsYoungAge - 1
5. arrenglishwordsYoungAge(i) = namefile(i)
6. Next i

The following dry run will test if the algorithm correctly reads in a few words. arrenglishwordsYoungAge has 10 locations.

Line Number	i	Namefile(i)	arrenglishwordsYoungAge(i)	Comment
01				Open file
02				Reads from file
03				Close file
04	0			
05		QUICK	NOTHING	

06			QUICK	Return to line 04
04	1			
05		BOOM	NOTHING	
06			BOOM	Return to line 04
04	2			
05		DISAPPEAR	NOTHING	
06			DISAPPEAR	Return to line 04
04	3			
05		LEVERAGE	NOTHING	
06			LEVERAGE	Return to line 04
04	4			
05		SHOCKING	NOTHING	
06			SHOCKING	Return to line 04
04	5			
05		WICKED	NOTHING	
06			WICKED	Return to line 04
04	6			
05		VIBRANT	NOTHING	
06			VIBRANT	Return to line 04
04	7			
05		ABSTRACT	NOTHING	
06			ABSTRACT	Return to line 04
04	8			
05		TINY	NOTHING	
06			TINY	Return to line 04
04	9			
05		HOT	NOTHING	
06			HOT	Return to line o4
04	10			
05		ANALYSE	NOTHING	
06			ANALYSE	End of loop



Loading the words from the array into the panel box

This algorithm will display the words to find in a panel box.

1. For i = 1 to 8
2. Make left value of arritems(i) = 10
3. Make top value of arritems(i) = i * 20
4. Make back colour of arritems(i) transparent
5. Make arritems(i) visible
6. Add to panel (arritems(i))
7. Next i

Change the colour of the label to blue

This algorithm is performed every time the user clicks on a label. If a word is found the colour of the label is set equal to red (see algorithm below), so if the user clicks on a label that is red it must remain red to avoid confusion.

1. *Input* mouseclick
2. Lbltemp = clicked label
3. If ForeColour of Lbltemp <> red Then
4. ForeColour of LblTemp = blue
5. End If

Pop the contents from the stack whilst changing the colour of the label to red. Increment the lblscoretext to the new score and update the word list so that the found word says the word and found.

When a word is found the game need to be updated. This algorithm will ensure all the requirement for when a word is found is included.

```
For i = 0 to length of mathswordsMiddleAge
    If lblsel = mathswordsMiddleAge(i)
      Count = 0
       Do
          Count = Count + 1
           Forecolour of ("label" & top reference in mystack) =  red colour (item is removed from stack)
       Loop Until the content of mystack = 0
    Forecolour of lbltemp = red
    intscore = numeric value of text stored in lblscore
    intscore = intscore + 1
    Make text of lblscore = intscore
    arrmathswordsMiddleAge(i) = arrmathswordsMiddleAge(i) & a space & "-" & a space &   "FOUND!"
    tempstr = arrmathswordsMiddleAge(i)

For intcount = 1 To 8
     lengthofarray = length of text arrItems(intcount) - 3
    Make text of arrItems(intcount) = right of text (arrItems(intcount), lengthofarray)
 Next intcount

For number = 1 to 8
  If  text of arritems(number) = lblsel then
     Make text of arritems(number) = number &  ")"  & a space & tempstr
   ElseIf   text of arrItems(number) <> lblSel Then
     Make text of arrItems(number) = number & ")" & a space & text of arrItems(number)
   End If
 Next number
 Lblsel = ""
 Lbltemp = ""
End if
Next i
```

Set everything to its default value

This algorithm will make up the restart sections (subroutine). The purpose of this algorithm is to set up the next game the initialisation of variable values will ensure the user that they are not using the same values from previous runs.

```
Enable cmdStartGame
Make text of LblGameTime = "Time Left:  00:00"
Stop TmrCountDown
lblSel = ""
Make text of LblScore = 0
intItemsFound = 0
intTimeBonus = 0
Clear the content of the labels found within pnlitems
number = 0
 For row = 0 to 13
    For column = 0 to 13
```

```
12.         number = number + 1
13.         Remove event handler mouse click
14.         Change forecolour of  the label 'number' to black
15.         Make text of controls label 'number' = "X"
16.         Gridarray(Row,Column) =  "X"
17.         arrunscramble(row,column) = "X"
18.         Enable controls on label 'number'
19.      Next column
20.  Next row
```

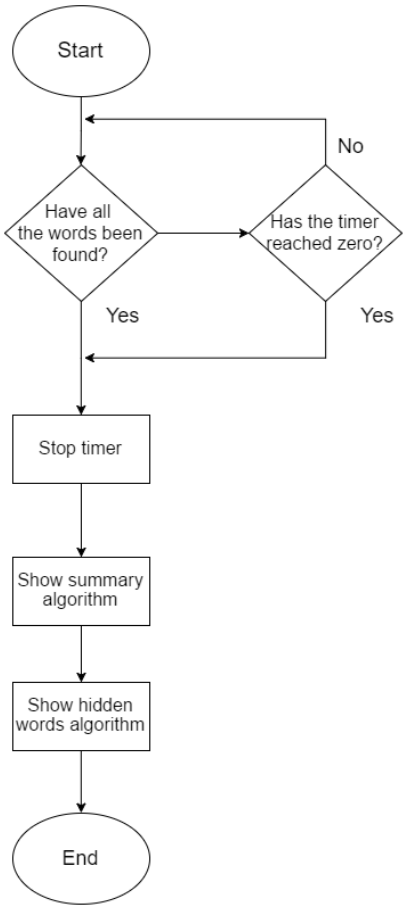
Clearing words

Aim is to pop content from the stack, whilst changing the label colour to black if a word is not found and leaving the word red if the word is found. This will allow the found labels to remain highlighted in red.

```
1.   Lblsel = " "
2.   Enable grid labels
3.    While count of mystack <> 0
4.         If forecolour of controls("label" referenced by top of mystack) = blue then
5.           forecolour of controls("label" referenced at top of mystack) = black  (item is removed from stack)
6.         Else if forecolour of controls("label" referenced at top of mystack) = red then
7.           forecolour of controls("label" referenced at top of mystack) = red   (item is removed from stack)
8.      End while
```

Line number	Count	"label" referenced by top of mystack	Comment
01			Lblsel becomes a blank space
02			Enable labels in the grid
03	3		True
04		Label 40	False, so go to line 06
06			True, so go to line 07
07			Forecolour of Label 40 remains red and the number 40 is removed from the stack
08			Return to line 03
03	2		True
04		Label 41	True, so go to line 05
05			Forecolour of Label 41 changes to black and the number 41 is removed from the stack
08			Return to line 03
03	1		True
04		Label 42	True, so go to line 05
05			Forecolour of Label 42 changes to black and the number 42 is removed from the stack
08			Return to line 08
03	0		False, move to line 08
08			End While Loop

Module 3:



Show summary algorithm

This algorithm will output the user score, when the game ends. The algorithm will add the user's bonus points to their fixed score. Timer diff is variable that stores the difference between the end time and the current time, basically the time that remains on the timer.

```
1.  If minutes of timer diff > 0
2.      intTimeBonus = minutes of timer diff * 21
3.  End if

4.  If seconds of timer diff > 0
5.      intTimeBonus = intTimeBonus + seconds of timer diff * 6
6.  End if

7.  intScore = (intScore + intTimeBonus)

8.  If itemsfound = 8 then                          'Checks if all the words have been found
9.    Stop timerCountDown
10.   Show messagebox with message "You Win"
11.   Call showhiddenwords subroutine
12.     For row = 0 to 13
13.       For column = 0 to 13
14.         intnum = intnum + 1
15.         Remove event handler mouseclick
16.       Next column
```

17. Next row
18. End if

19. Make text of strsummary = “Summary” & new line & new line & “Timebonus:” & intTimeBonus & new line & new line & “Score:” intScore
20. Make lblscoresummary visible
21. set text of lblscoresummary to strsummary

Stop the timer

1. If (ticks of timer diff < 0) then
2. Stop timerCountDown
3. Show messagebox with message “Time’s up”
4. Call ShowSummary subroutine
5. Call ShowHiddenWords summary
6. For row = 0 to 13
7. For column = 0 to 13
8. intnum = intnum + 1
9. Remove event handler mouseclick (i.e. this label cannot be clicked on)
10. Next column
11. Next row
12. End if

Show hidden words algorithm

This algorithm will display the missing words, by replacing all the randomised characters with a blank space.

1. For row = 0 To 13
2. For column = 0 To 13
3. If arrunscramble(row, column) = "Char" Then
4. arrunscramble(row, column) = " "
5. End If
6. Next column
7. Next row

8. For row = 0 To 13
9. For column = 0 To 13
10. counter = counter + 1
11. If arrunscramble(row, column) = " " Then
12. Text of controls("label" & counter) = arrunscramble(row, column)
13. End If
14. Next column
15. Next row

When testing the above algorithm, I will use the content in arrunscramble record in the test on page 48.

Line number	Row	Column	arrunscramble(row,coloumn)	Comments
01	0			
02		0		
03				true
04			Blank space	
05				End if

06					Return to line 02
02		1			
03					True
04				Blank space	
05					End if
06					Return to line 02
02		2			
03					True
04				Blank space	
05					End if
06					Return to line 02
02		3			
03					False, move to line 05
05					End if
06					Return to line 02

The above is repeated until all the random characters generated are replaced with a blank space. When that is done, I will test the second part of the algorithm, which involves displaying the content of arrunscramble on-screen.

Line number	Row	Column	Counter	Arrunscramble(row,column)	Comments
08	0				
09		0			
10			1		
11					True
12					Text of label 1 is set to a blank space
13					End if
14					Return to line 09
09		1			
10			2		
11					True
12					Text of label 2 is set to a blank space
13					End if
14					Return to line 09
09		2			
10			3		
11					True
12					Text of label 3 is set to a blank space
13					End if
14					Return to line 09
09		3			
10			4		
11					False, go to line 13
13					End if
14					Return to line 09

Software Development

The time has arrived for me to develop my software. The planning of the intended system has now been completed and documented. I shall now describe how I have progressed with the creation of the intended solution and explain any problems that I encountered and how they had been solved.

The first stage is to set up the global variable, so that the rest of the program can access the data.

```
Dim GridArray(13, 13) As String  'An array to store the content of the labels in the grid
Dim arrenglishwordsYoungAge(10) As String 'Array of English words for young age group
Dim arrmathswordsYoungAge(10) As String ' Array of Maths words for young age group
Dim arrsciencewordsYoungAge(10) As String ' Array of Science words for young age group
Dim arrenglishwordsMiddleAge(10) As String 'Array of English words for middle age group
Dim arrmathswordsMiddleAge(10) As String 'Array of Maths words for middle age group
Dim arrsciencewordsMiddleAge(10) As String 'Array of Science words for middle age group
Dim arrenglishwordsEldestAge(10) As String 'Array of English words for eldest age group
Dim arrmathswordsEldestAge(10) As String 'Array of Maths words for eldest age group
Dim arrsciencewordsEldestAge(10) As String 'Array of Science words for eldest age group
Dim mystack As New Stack ' Holds the Labels  that have been clicked on
Dim lblSel As String = Nothing 'Selected Label's Text
Private timeEnd As DateTime 'Calculates Ending Time
Private timeDiff As TimeSpan 'Difference Between Start and End Times
Private intTimeBonus As Integer = 0 'Stores the time bonus
Private intScore As Integer = 0 'Stores the value of the player's score
Private intItemsFound As Integer = 0 'Total Items Found
Private arrItems(8) As Label 'Array of Labels
Dim intrandom(8) As Integer 'Array of random numbers
Dim arrunscramble(13, 13) As String 'Array of the word "Char" used to located the random filled words.
Dim namefile() As String 'Array of words found in the text file
Dim newwordgencount As Integer = 0 'Count to store the number of times the subroutine(newwordgen) is accessed
```

After declaring the variables for all the forms, I began coding the first module, in which I performed alpha testing according to ensure that all functions and algorithms work are expected.

Module 1:

I will begin by creating the subroutines for storing the content in the text files (the words to find) into an array. This will use the algorithm for reading the words in the text file and store the content in an array. The code listed below will use the visual studio utility called *stream reader,* which will be used to read the file in the location as stated in parenthesis.

```
Private Sub EnglishwordsYoungAge() 'Stores the English words into its array
    Dim reader As System.IO.StreamReader 'Visual Studio tool used to read the content in the text file
    reader = My.Computer.FileSystem.OpenTextFileReader("EnglishWordsYoungAge.txt") 'Opens and reads the content in the file stated
    namefile = reader.ReadToEnd.Split(";") 'Array Name file stores the substring found in the file
    reader.Close() 'Closes the file and reader

    'For loop used to store the words in array namefile into arrenglishwordsyoungage
    For i As Integer = 0 To (arrenglishwordsYoungAge.Length - 1)
        arrenglishwordsYoungAge(i) = namefile(i)
    Next
End Sub
```

A variant of this subroutine will be produced to allow for the different data structures to be stored in their respective arrays. i.e. The SciencewordsMiddleAge text file will be stored in the array called arrsciencewordsmiddleage.

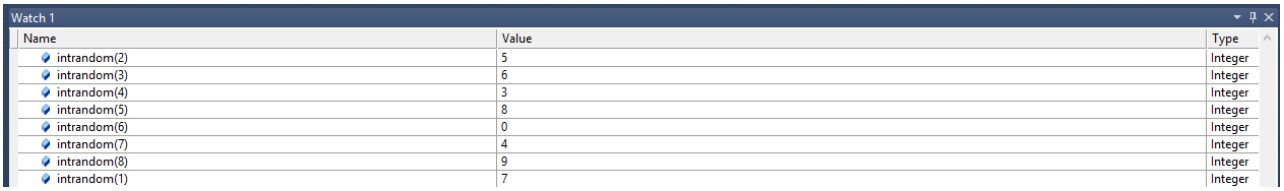
I will now code the algorithm for checking for a double word. To test if the numbers in the array intrandom are different, I will add breakpoints and use a watch to test if the values in the array are different. Note: intrandom is an array of random numbers, which will be used to access the words in the array.

```
Private Sub CheckforDoubleWord() 'Subroutine for ensuring that every element in array intrandom is different

    While intrandom(2) = intrandom(1) 'If the number in location 2 = number location 1 then a new number will be generated
        intrandom(2) = ((arrsciencewordsEldestAge.Length - 1) * Rnd()) 'Generates a new random number between 0 and the length of array
    End While

    While intrandom(3) = intrandom(1) Or intrandom(3) = intrandom(2) 'If number in location 3 = number in location 2 or location 1 then a new number is generated
        intrandom(3) = ((arrsciencewordsEldestAge.Length - 1) * Rnd()) 'Generates a new random number between 0 and the length of array
    End While

    While intrandom(4) = intrandom(1) Or intrandom(4) = intrandom(2) Or intrandom(4) = intrandom(3)
        intrandom(4) = ((arrsciencewordsEldestAge.Length - 1) * Rnd())
    End While

    While intrandom(5) = intrandom(1) Or intrandom(5) = intrandom(2) Or intrandom(5) = intrandom(3) Or intrandom(5) = intrandom(4)
        intrandom(5) = ((arrsciencewordsEldestAge.Length - 1) * Rnd())
    End While

    While intrandom(6) = intrandom(1) Or intrandom(6) = intrandom(2) Or intrandom(6) = intrandom(3) Or intrandom(6) = intrandom(4) Or intrandom(6) = intrandom(5)
        intrandom(6) = ((arrsciencewordsEldestAge.Length - 1) * Rnd())
    End While

    While intrandom(7) = intrandom(1) Or intrandom(7) = intrandom(2) Or intrandom(7) = intrandom(3) Or intrandom(7) = intrandom(4) Or intrandom(7) = intrandom(5) Or intrandom(7) = intrandom(6)
        intrandom(7) = ((arrsciencewordsEldestAge.Length - 1) * Rnd())
    End While

    While intrandom(8) = intrandom(1) Or intrandom(8) = intrandom(2) Or intrandom(8) = intrandom(3) Or intrandom(8) = intrandom(4) Or intrandom(8) = intrandom(5) Or intrandom(8) = intrandom(6) Or
        intrandom(8) = ((arrsciencewordsEldestAge.Length - 1) * Rnd())
    End While
```

Watch 1

Name	Value	Type
intrandom(2)	5	Integer
intrandom(3)	6	Integer
intrandom(4)	3	Integer
intrandom(5)	3	Integer
intrandom(6)	8	Integer
intrandom(7)	0	Integer
intrandom(8)	8	Integer
intrandom(1)	7	Integer

The watch above shows that intrandom 4 and 5 contain the same number. According to the above code, the game should generate a new random number for each element in the array. I will now run the code using the stepping, to see the change in the element in intrandom



Watch 1

Name	Value	Type
intrandom(2)	5	Integer
intrandom(3)	6	Integer
intrandom(4)	3	Integer
intrandom(5)	8	Integer
intrandom(6)	0	Integer
intrandom(7)	4	Integer
intrandom(8)	9	Integer
intrandom(1)	7	Integer

We can see that each element in intrandom is different, so this means that the grid will have different words and that the same word will not occur twice in the grid.

The next stage in the development of module one is the coding of the subroutines to fill the words for the different directions: North, Northeast, East, Southeast, South, Southwest, West and Northwest.

I will now begin writing the subroutine for filling a word in the east direction. The word will be broken into individual characters and these character will be stored in an array using the identifier chararr(). The algorithm for checking if a word fits will be used to see if the words fits in an empty location. The subroutine will also use the algorithm for filling in those charaters. The latter part of the subroutine will create a label of the word that has just been filled into the grid, so it can be used in the word list.

```
Private Sub FillEast(ByVal strletters As String, ByVal subject As String) 'Fill Blocks From Left To Right
    Dim intcount As Integer = 0 'Amount of times the subroutine is executed
    Dim loopflag As Boolean = True 'Flag for the main loop
    While loopflag = True 'The main loop - loops until word can be placed in grid or until loop counter reaches 100
        Randomize() 'Visual Studio tool that randomizes the subroutine
        Dim String_length As Integer = Len(strletters) 'Contains the number of character in the parameter strletters
        Dim charArr() As Char = strletters.ToCharArray 'Breaks Letters Appart
        Dim RandomX As Integer = (Int((14 * Rnd()))) 'Random number which represent the coordinate on the x-axis
        Dim randomY As Integer = (Int((14 - (String_length - 1)) * Rnd())) 'Random number which represent the coordinate on the y-axis
        Dim count1 As Integer = 0 'Counts the number of times the loop has been executed
        Dim count2 As Integer = 0 'Counts the number of times the loop has been executed
        Dim checkflag As Boolean = True 'Flag which determines if a word fits in an empty location in grid

        'Check if target word fits in the grid
        While count1 <= charArr.Length - 1 'Loops through each character in array
            If (GridArray(RandomX, randomY) <> " " And GridArray(RandomX, randomY) <> charArr(count1)) Then ' Checks if word fits in a location
                checkflag = False 'Flags up if the word doesn't fits in a location
            End If
            count1 += 1 ' increment counter
            randomY += 1 ' increment random Y for the next location to be filled
        End While

        If checkflag = True Then 'If the flag remains true then there is and empty location
            loopflag = False 'Stops the main loop from checking for new random location to fill the word
        End If

        randomY = randomY - (charArr.Length) 'returns the original value of random y

        While count2 <= (charArr.Length - 1) And checkflag = True 'Fill blocks in the empty location found
            GridArray(RandomX, randomY) = UCase(charArr(count2)) 'Set The Text of Grid Blocks
            count2 += 1 'increment count
            randomY += 1 'increment random y to next location to be filled
        End While

        intcount += 1 'Increment the loop counter

        If intcount = 100 Then
            strletters = NewwordGen(subject) 'Call the subroutine for a new word to replace the current word that couldn't not be filled.
        ElseIf intcount = 200 Then
            Exit Sub 'Exits the subroutine
        End If
    End While

    arrItems(1) = New Label 'Create New Label And Set Appropriate Properties
    arrItems(1).AutoSize = True
    arrItems(1).Text = (1).ToString & ") " & strletters 'Add Items To Be Found Label Text
End Sub
```

Subroutine for filling a word in the west direction

```
Private Sub FillWest(ByVal strletters As String, ByVal subject As String) 'Fill Blocks From Right To Left
    Dim intcount As Integer = 0 'Amount of times the subroutine is executed
    Dim loopflag As Boolean = True 'Flag for the main loop

    While loopflag = True 'A main loop - loops until word can be placed in grid or until loop counter reaches 100
        Randomize() 'Visual Studio tool that randomizes the subroutine
        Dim String_length As Integer = Len(strletters) 'Contains the number of character in the parameter strletters
        Dim charArr() As Char = strletters.ToCharArray 'Break Letters Appart'
        Dim RandomX As Integer = (Int((14 * Rnd()))) 'Random number which represent the coordinate on the x-axis
        Dim randomY As Integer = (Int((14 - (String_length - 1)) * Rnd())) ''Random number which represent the coordinate on the y-axis
        Dim count1 As Integer = 0 'Counts the number of times the loop has been executed
        Dim count2 As Integer = 0 'Counts the number of times the loop has been executed
        Dim checkflag As Boolean = True 'Flag which determines if a word fits in an empty location in grid

        Array.Reverse(charArr) 'Reverses the elements in the array

        'Check if target word fits in the grid
        While count1 <= charArr.Length - 1 'Loops through each character in array
            If (GridArray(RandomX, randomY) <> " " And GridArray(RandomX, randomY) <> charArr(count1)) Then 'Checks if word fits in a location
                checkflag = False 'Flags up if the word doesn't fits in a location
            End If
            count1 += 1 'increment count
            randomY += 1 'increment random y for next location to be filled
        End While

        If checkflag = True Then 'If the flag remains true then there is and empty location
            loopflag = False 'Stops the main loop from checking for new random location to fill the word
        End If

        randomY = randomY - (charArr.Length) 'returns the original value of random y

        While count2 <= charArr.Length - 1 And checkflag = True 'Fill blocks in the empty location found
            GridArray(RandomX, randomY) = UCase(charArr(count2)) 'Set The Text of Grid Blocks'
            count2 += 1 'Increment count
            randomY += 1 'increment random y to next location to be filled
        End While

        intcount += 1 'Increment intcount

        If intcount = 100 Then
            strletters = NewwordGen(subject) 'Call the subroutine for a new word to replace the current word that couldn't not be filled.
        ElseIf intcount = 200 Then
            Exit Sub 'Exits the subroutine
        End If
    End While


    arrItems(2) = New Label 'Create New Label And Set Appropriate Properties
    arrItems(2).AutoSize = True
    arrItems(2).Text = (2).ToString & ") " & strletters 'Add Items To Be Found Label Text
End Sub
```

Subroutine for filling a word in the south direction

```
Private Sub FillSouth(ByVal strletters As String, ByVal subject As String) 'Fill blocks Downwards (North to South)
    Dim intcount As Integer = 0 'Amount of times the subroutine is executed
    Dim loopflag As Boolean = True 'Flag for the main loop

    While loopflag = True 'The main loop - loops until word can be placed in grid or until loop counter reaches 100
        Randomize() 'Visual Studio tool that randomizes the subroutine
        Dim String_length As Integer = Len(strletters) 'Contains the number of character in the parameter strletters
        Dim charArr() As Char = strletters.ToCharArray 'Break Letters Appart'
        Dim RandomX As Integer = (Int((14 - (String_length - 1)) * Rnd())) 'Random number which represent the coordinate on the x-axis
        Dim randomY As Integer = (Int((14 * Rnd()))) 'Random number which represent the coordinate on the y-axis
        Dim checkflag As Boolean = True 'Flag which determines if a word fits in an empty location in grid
        Dim Count1 As Integer = 0 'Loop Counter 1
        Dim count2 As Integer = 0 'Loop counter 2

        'Check if target word fits in the grid
        While Count1 <= charArr.Length - 1  'Loops through each character in array
            If (GridArray(RandomX, randomY) <> " " And GridArray(RandomX, randomY) <> charArr(Count1)) Then 'Checks if word fits in a location
                checkflag = False 'Flags up if the word doesn't fits in a location
            End If
            Count1 += 1 'Increment count
            RandomX += 1 'Increment Random X for the next location to be filled
        End While

        If checkflag = True Then 'If the flag remains true then there is and empty location
            loopflag = False 'Stops the main loop from checking for new random location to fill the word
        End If

        RandomX = RandomX - (charArr.Length) 'Returns the original value of Random X

        While count2 <= charArr.Length - 1 And checkflag = True 'Fill blocks in the empty location found
            GridArray(RandomX, randomY) = UCase(charArr(count2)) 'Set The Text of Grid Blocks'
            count2 += 1 'Increment count2
            RandomX += 1 'Increment RandomX
        End While

        intcount += 1 'increment intcount for each loop execution

        If intcount = 100 Then
            strletters = NewwordGen(subject) 'Call the subroutine for a new word to replace the current word that couldn't not be filled.
        ElseIf intcount = 200 Then
            Exit Sub 'Exits the subroutine
        End If

    End While

    arrItems(3) = New Label 'Create New Label And Set Appropriate Properties
    arrItems(3).AutoSize = True
    arrItems(3).Text = (3).ToString & ") " & strletters 'Add Items To Be Found Label Text
End Sub
```

Subroutine for filling in a word in the north direction

```
Private Sub FillNorth(ByVal strletters As String, ByVal subject As String) 'Fill blocks Upwards (South to North)
    Dim intcount As Integer = 0 'Amount of times the subroutine is executed
    Dim loopflag As Boolean = True 'Flag for the main loop

    While loopflag = True 'The main loop - loops until word can be placed in grid or until loop counter reaches 100
        Randomize() 'Visual Studio tool that randomizes the subroutine
        Dim String_length As Integer = Len(strletters) 'Contains the number of character in the parameter strletters
        Dim charArr() As Char = strletters.ToCharArray 'Break Letters Appart
        Dim RandomX As Integer = (Int((14 - (String_length - 1)) * Rnd())) 'Random number which represent the coordinate on the x-axis
        Dim randomY As Integer = (Int((14 * Rnd()))) 'Random number which represent the coordinate on the y-axis
        Dim checkflag As Boolean = True 'Flag which determines if a word fits in an empty location in grid
        Dim count1 As Integer = 0 'Counts the number of times the loop has been executed
        Dim count2 As Integer = 0 'Counts the number of times the loop has been executed

        Array.Reverse(charArr) 'Reverses the elements in the array

        'Check if target word fits in the grid
        While Count1 <= charArr.Length - 1  'Loops through each character in array
            If (GridArray(RandomX, randomY) <> " " And GridArray(RandomX, randomY) <> charArr(Count1)) Then 'Checks if word fits in a location
                checkflag = False 'Flags up if the word doesn't fits in a location
            End If
            Count1 += 1 'Increment count
            RandomX += 1 'Increment Random X for the next location to be filled
        End While

        If checkflag = True Then 'If the flag remains true then there is and empty location
            loopflag = False 'Stops the main loop from checking for new random location to fill the word
        End If

        RandomX = RandomX - (charArr.Length) 'Ensures RandomX is set back to it original value'

        While Count2 <= charArr.Length - 1 And checkflag = True 'Fill blocks in the empty location found
            GridArray(RandomX, randomY) = UCase(charArr(Count2)) 'Set The Text of Grid Blocks'
            Count2 += 1 'Increment count2
            RandomX += 1 'Increment RandomX
        End While

        intcount += 1 'Increment intcount for each loop execution

        If intcount = 100 Then
            strletters = NewwordGen(subject) 'Call the subroutine for a new word to replace the current word that couldn't not be filled.
        ElseIf intcount = 200 Then
            Exit Sub 'Exits the subroutine
        End If

    End While

    arrItems(4) = New Label 'Create New Label And Set Appropriate Properties
    arrItems(4).AutoSize = True
    arrItems(4).Text = (4).ToString & ") " & strletters 'Add Items To Be Found Label Text
End Sub
```

Subroutine for filling in the southeast direction

```
Private Sub FillSouthEast(ByVal strletters As String, ByVal subject As String) 'Fill blocks diagonally (NorthWest to SouthEast)
    Dim intcount As Integer = 0 'Amount of times the subroutine is executed
    Dim loopflag As Boolean = True 'Flag for the main loop

    While loopflag = True 'The main loop - loops until word can be placed in grid or until loop counter reaches 100
        Randomize() 'Visual Studio tool that randomizes the subroutine
        Dim String_length As Integer = Len(strletters) 'Contains the number of character in the parameter strletters
        Dim charArr() As Char = strletters.ToCharArray 'Break Letters Appart'
        Dim RandomX As Integer = (Int((14 - (String_length - 1)) * Rnd())) 'Random number which represent the coordinate on the x-axis
        Dim randomY As Integer = (Int((14 - (String_length - 1)) * Rnd())) 'Random number which represent the coordinate on the y-axis
        Dim checkflag As Boolean = True 'Flag which determines if a word fits in an empty location in grid
        Dim count1 As Integer = 0 'Counts the number of times the loop has been executed
        Dim count2 As Integer = 0 'Counts the number of times the loop has been executed

        'Check if target word fits in the grid
        While Count1 <= charArr.Length - 1  'Loops through each character in array
            If (GridArray(RandomX, randomY) <> " " And GridArray(RandomX, randomY) <> charArr(Count1)) Then 'Checks if word fits in a location
                checkflag = False 'Flags up if the word doesn't fits in a location
            End If
            Count1 += 1 'Increment count
            RandomX += 1 'Increment Random X for the next location to be filled
            randomY += 1 'Increment Random y for the next location to be filled
        End While

        If checkflag = True Then 'If the flag remains true then there is and empty location
            loopflag = False 'Stops the main loop from checking for new random location to fill the word
        End If

        RandomX = RandomX - (charArr.Length) 'Ensures Random X is set back to it original value'
        randomY = randomY - (charArr.Length) 'Ensures Random y is set back to it original value'

        While Count2 <= charArr.Length - 1 And checkflag = True 'Fill blocks in the empty location found
            GridArray(RandomX, randomY) = UCase(charArr(Count2)) 'Set The Text of Grid Blocks'
            Count2 += 1 'Increment count2
            RandomX += 1 'Increment Random X
            randomY += 1 'Increment Random Y
        End While

        intcount += 1 'Increment intcount for each loop execution

        If intcount = 100 Then
            strletters = NewwordGen(subject) 'Call the subroutine for a new word to replace the current word that couldn't not be filled.
        ElseIf intcount = 200 Then
            Exit Sub 'Exits the subroutine
        End If

    End While

    arrItems(5) = New Label 'Create New Label And Set Appropriate Properties
    arrItems(5).AutoSize = True
    arrItems(5).Text = (5).ToString & ") " & strletters 'Add Items To Be Found Label Text
End Sub
```

Subroutine for filling a word in the northeast direction

```
Private Sub FillNortheast(ByVal strletters As String, ByVal subject As String) 'Fill blocks Diagonally (Southwest to Northeast)
    Dim intcount As Integer = 0 'Amount of times the subroutine is executed
    Dim loopflag As Boolean = True 'Flag for the main loop

    While loopflag = True 'The main loop - loops until word can be placed in grid or until loop counter reaches 100
        Randomize() 'Visual Studio tool that randomizes the subroutine
        Dim String_length As Integer = Len(strletters) 'Contains the number of character in the parameter strletters
        Dim charArr() As Char = strletters.ToCharArray 'Break Letters Appart'
        Dim RandomX As Integer = (Int((14 - (String_length - 1)) * Rnd())) 'Random number which represent the coordinate on the x-axis
        Dim randomY As Integer = (Int((14 - (String_length - 1)) * Rnd())) 'Random number which represent the coordinate on the y-axis
        Dim checkflag As Boolean = True 'Flag which determines if a word fits in an empty location in grid
        Dim count1 As Integer = 0 'Counts the number of times the loop has been executed
        Dim count2 As Integer = 0 'Counts the number of times the loop has been executed

        Array.Reverse(charArr) 'Reverses the elements in the array

        'Check if target word fits in the grid
        While Count1 <= charArr.Length - 1  'Loops through each character in array
            If (GridArray(RandomX, randomY) <> " " And GridArray(RandomX, randomY) <> charArr(Count1)) Then 'Checks if word fits in a location
                checkflag = False 'Flags up if the word doesn't fits in a location
            End If
            Count1 += 1 'Increment count
            RandomX += 1 'Increment Random X for the next location to be checked
            randomY += 1 'Increment Random y for the next location to be checked
        End While

        If checkflag = True Then 'If the flag remains true then there is and empty location
            loopflag = False 'Stops the main loop from checking for new random location to fill the word
        End If

        RandomX = RandomX - (charArr.Length) 'This is to make sure Random X is set back to it original value'
        randomY = randomY - (charArr.Length) 'This is to make sure Random y is set back to it original value'

        While Count2 <= charArr.Length - 1 And checkflag = True 'Fill blocks in the empty location found
            GridArray(RandomX, randomY) = UCase(charArr(Count2)) 'Set The Text of Grid Blocks'
            Count2 += 1 'Increment count2
            RandomX += 1 'Increment Random X for the next location to be filled
            randomY += 1 'Increment Random Y for the next location to be filled
        End While

        intcount += 1 'Increment intcount for each loop execution

        If intcount = 100 Then
            strletters = NewwordGen(subject) 'Call the subroutine for a new word to replace the current word that couldn't not be filled.
        ElseIf intcount = 200 Then
            Exit Sub 'Exits the subroutine
        End If

    End While

    arrItems(6) = New Label 'Create New Label And Set Appropriate Properties
    arrItems(6).AutoSize = True
    arrItems(6).Text = (6).ToString & ") " & strletters 'Add Items To Be Found Label Text
End Sub
```

Subroutine for filling a word in the Northwest direction

```
Private Sub FillNorthWest(ByVal strletters As String, ByVal subject As String) 'Fill blocks diagonally (SouthEast to NorthWest)
    Dim intcount As Integer = 0 'Amount of times the subroutine is executed
    Dim loopflag As Boolean = True 'Flag for the main loop

    While loopflag = True 'The main loop - loops until word can be placed in grid or until loop counter reaches 100
        Randomize() 'Visual Studio tool that randomizes the subroutine
        Dim String_length As Integer = Len(strletters) 'Contains the number of character in the parameter strletters
        Dim charArr() As Char = strletters.ToCharArray 'Break Letters Appart'
        Dim RandomX As Integer = (Int((14 - (String_length - 1)) * Rnd() + (String_length - 1))) 'Random number (starting from string_length - 1) which represent the coordinate on the x-axis
        Dim randomY As Integer = (Int((14 - (String_length - 1)) * Rnd())) 'Random number which represent the coordinate on the y-axis
        Dim checkflag As Boolean = True ''Flag which determines if a word fits in an empty location in grid
        Dim count1 As Integer = 0 'Counts the number of times the loop has been executed
        Dim count2 As Integer = 0 'Counts the number of times the loop has been executed

        'Check if target word fits in the grid
        While Count1 <= charArr.Length - 1 'Loops through each character in array
            If (GridArray(RandomX, randomY) <> " " And GridArray(RandomX, randomY) <> charArr(Count1)) Then 'Checks if word fits in a location
                checkflag = False 'Flags up if the word doesn't fits in a location
            End If
            Count1 += 1 'Increment count
            RandomX -= 1 'Decrement Random X for the next location to be checked
            randomY += 1 'Increment Random Y for the next location to be checked
        End While

        If checkflag = True Then 'If the flag remains true then there is and empty location
            loopflag = False 'Stops the main loop from checking for new random location to fill the word
        End If

        RandomX = RandomX + (charArr.Length) 'Ensures Random X is set back to it original value'
        randomY = randomY - (charArr.Length) 'Ensures Random y is set back to it original value'

        While Count2 <= charArr.Length - 1 And checkflag = True 'Fill blocks in the empty location found
            GridArray(RandomX, randomY) = UCase(charArr(Count2)) 'Set The Text of Grid Blocks'
            Count2 += 1 'Increment count2
            RandomX -= 1 'Decrement Random X for the next location to be filled
            randomY += 1 'Increment Random Y for the next location to be filled
        End While

        intcount += 1 'Increment intcount for each loop execution

        If intcount = 100 Then
            strletters = NewwordGen(subject) 'Call the subroutine for a new word to replace the current word that couldn't not be filled.
        ElseIf intcount = 200 Then
            Exit Sub 'Exits the subroutine
        End If

    End While
    arrItems(7) = New Label 'Create New Label And Set Appropriate Properties
    arrItems(7).AutoSize = True
    arrItems(7).Text = (7).ToString & ") " & strletters 'Add Items To Be Found Label Text
End Sub
```

Subroutine for filling a word in the southwest direction

```
Private Sub FillSouthWest(ByVal strletters As String, ByVal subject As String) 'Fill blocks diagonally (NorthEast to Southwest)
    Dim intcount As Integer = 0 'Amount of times the subroutine is executed
    Dim loopflag As Boolean = True 'Flag for the main loop

    While loopflag = True 'The main loop - loops until word can be placed in grid or until loop counter reaches 100
        Randomize() 'Visual Studio tool that randomizes the subroutine
        Dim String_length As Integer = Len(strletters) 'Contains the number of character in the parameter strletters
        Dim charArr() As Char = strletters.ToCharArray 'Break Letters Appart'
        Dim RandomX As Integer = (Int((14 - (String_length - 1)) * Rnd() + (String_length - 1))) 'Random number (starting from string_length - 1) which represent the coordinate on the x-axis
        Dim randomY As Integer = (Int((14 - (String_length - 1)) * Rnd())) 'Random number which represent the coordinate on the y-axis
        Dim checkflag As Boolean = True ''Flag which determines if a word fits in an empty location in grid
        Dim count1 As Integer = 0 'Counts the number of times the loop has been executed
        Dim count2 As Integer = 0 'Counts the number of times the loop has been executed

        Array.Reverse(charArr) 'Reverses the elements in the array

        'Check if target word fits in the grid
        While Count1 <= charArr.Length - 1 'Loops through each character in array
            If (GridArray(RandomX, randomY) <> " " And GridArray(RandomX, randomY) <> charArr(Count1)) Then 'Checks if word fits in a location
                checkflag = False 'Flags up if the word doesn't fits in a location
            End If
            Count1 += 1 'Increment count
            RandomX -= 1 'Decrement Random X for the next location to be checked
            randomY += 1 'Increment Random Y for the next location to be checked
        End While

        If checkflag = True Then 'If the flag remains true then there is and empty location
            loopflag = False 'Stops the main loop from checking for new random location to fill the word
        End If

        RandomX = RandomX + (charArr.Length) 'Ensures Random X is set back to it original value'
        randomY = randomY - (charArr.Length) 'Ensures Random y is set back to it original value'

        While Count2 <= charArr.Length - 1 And checkflag = True 'Fill blocks in the empty location found
            GridArray(RandomX, randomY) = UCase(charArr(Count2)) 'Set The Text of Grid Blocks'
            Count2 += 1 'Increment count2
            RandomX -= 1 'Decrement Random X for the next location to be filled
            randomY += 1 'Increment Random Y for the next location to be filled
        End While

        intcount += 1 'Increment intcount for each loop execution

        If intcount = 100 Then
            strletters = NewwordGen(subject) 'Call the subroutine for a new word to replace the current word that couldn't not be filled.
        ElseIf intcount = 200 Then
            Exit Sub 'Exits the subroutine
        End If

    End While

    arrItems(8) = New Label 'Create New Label And Set Appropriate Properties
    arrItems(8).AutoSize = True
    arrItems(8).Text = (8).ToString & ") " & strletters 'Add Items To Be Found Label Text
End Sub
```

The next task is write subroutines for filling in some words appropiate for the user's age and the game theme selected. The subroutine used to fill the words into the grid will depend on the user's age and the game theme selected. For example, if the user select the checkboxes for age <10 and theme english, then the subroutine fill_EnglishwordsYoungAge will be used. A young age user is less than 10, a middle age user is 11-15 and the eldest age user is 16+.

Fill_EnglishwordsYoungAge:

```
Private Sub Fill_EnglishwordsYoungAge() ''Fills the grid with English words appropriate for Young age user

    EnglishwordsYoungAge() 'Calls the subroutine EnglishwordsYoungAge

    For Row = 0 To 13 Step 1 'Loops thorugh every row in the grid
        For column = 0 To 13 Step 1 'Loops through every column in the grid
            GridArray(Row, column) = " " 'Initialises array to a blank space
        Next
    Next

    For i = 1 To 8
        intrandom(i) = ((arrenglishwordsYoungAge.Length - 1) * Rnd()) 'Generates a new random number for the locations in intrandom
    Next

    CheckforDoubleWord() 'Call subroutine CheckforDoubleWord

    'The following subroutine are called to fill the random word in the directions stated below
    FillEast(arrenglishwordsYoungAge(intrandom(1)), "english") 'East
    FillWest(arrenglishwordsYoungAge(intrandom(2)), "english") 'West
    FillSouth(arrenglishwordsYoungAge(intrandom(3)), "english") 'South
    FillNorth(arrenglishwordsYoungAge(intrandom(4)), "english") 'North
    FillSouthEast(arrenglishwordsYoungAge(intrandom(5)), "english") 'SouthEast
    FillNorthWest(arrenglishwordsYoungAge(intrandom(6)), "english") 'NorthWest
    FillNortheast(arrenglishwordsYoungAge(intrandom(7)), "english") 'NorthEast
    FillSouthWest(arrenglishwordsYoungAge(intrandom(8)), "english") 'SouthWest

    'Sets the property of the labels in "arritems"
    For i = 1 To 8
        arrItems(i).Left = 10
        arrItems(i).Top = i * 20
        arrItems(i).BringToFront()
        arrItems(i).Visible = True
        arrItems(i).BackColor = Color.Transparent
        pnlitems.Controls.Add(arrItems(i)) 'Add Items To Be Found
    Next

End Sub
```

Fill_MathswordsYoungAge:

```
Private Sub Fill_MathswordsYoungAge() ''Fills the grid with Maths words appropriate for a Young age user

    MathswordsYoungAge() 'Calls the subroutine MathswordsYoungAge

    'Intitalise GridArray to a blank space
    For Row = 0 To 13 Step 1
        For column = 0 To 13 Step 1
            GridArray(Row, column) = " "
        Next
    Next

    For i = 1 To 8
        intrandom(i) = ((arrmathswordsYoungAge.Length - 1) * Rnd()) 'Generates a new random number for the locations in intrandom
    Next

    CheckforDoubleWord() 'Calls subroutine CheckforDoubleWord

    'The following subroutine are called to fill the random word in the directions stated below
    FillEast(arrmathswordsYoungAge(intrandom(1)), "maths") 'East
    FillWest(arrmathswordsYoungAge(intrandom(2)), "maths") 'West
    FillSouth(arrmathswordsYoungAge(intrandom(3)), "maths") 'South
    FillNorth(arrmathswordsYoungAge(intrandom(4)), "maths") 'North
    FillSouthEast(arrmathswordsYoungAge(intrandom(5)), "maths") 'SouthEast
    FillNorthWest(arrmathswordsYoungAge(intrandom(6)), "maths") 'NorthWest
    FillNortheast(arrmathswordsYoungAge(intrandom(7)), "maths") 'NorthEast
    FillSouthWest(arrmathswordsYoungAge(intrandom(8)), "maths") 'SouthWest

    'Sets the property of the labels in "arritems"
    For i = 1 To 8
        arrItems(i).Left = 10
        arrItems(i).Top = i * 20
        arrItems(i).BringToFront()
        arrItems(i).Visible = True
        arrItems(i).BackColor = Color.Transparent
        pnlitems.Controls.Add(arrItems(i)) 'Add Items To Be Found
    Next

End Sub
```

Fill_SciencewordsYoungAge:

```
Private Sub Fill_sciencewordsYoungAge() 'Fills the grid with Science words appropriate for a Young age user

    SciencewordsYoungAge() 'Calls the subroutine SciencewordsYoungAge

    'Intitalise GridArray to a blank space
    For Row = 0 To 13 Step 1
        For column = 0 To 13 Step 1
            GridArray(Row, column) = " "
        Next
    Next

    For i As Integer = 1 To 8
        intrandom(i) = ((arrsciencewordsYoungAge.Length - 1) * Rnd())  'Generates a new random number for the locations in intrandom
    Next

    CheckforDoubleWord() 'Calls subroutine CheckforDoubleWord

    'The following subroutine are called to fill the random word in the directions stated below
    FillEast(arrsciencewordsYoungAge(intrandom(1)), "science") 'East
    FillWest(arrsciencewordsYoungAge(intrandom(2)), "science") 'West
    FillSouth(arrsciencewordsYoungAge(intrandom(3)), "science") 'South
    FillNorth(arrsciencewordsYoungAge(intrandom(4)), "science") 'North
    FillSouthEast(arrsciencewordsYoungAge(intrandom(5)), "science") 'SouthEast
    FillNorthWest(arrsciencewordsYoungAge(intrandom(6)), "science") 'NorthWest
    FillNortheast(arrsciencewordsYoungAge(intrandom(7)), "science") 'NorthEast
    FillSouthWest(arrsciencewordsYoungAge(intrandom(8)), "science") 'SouthWest

    'Sets the property of the labels in "arritems"
    For i = 1 To 8
        arrItems(i).Left = 10
        arrItems(i).Top = i * 20
        arrItems(i).BringToFront()
        arrItems(i).Visible = True
        arrItems(i).BackColor = Color.Transparent
        pnlitems.Controls.Add(arrItems(i)) 'Add Items To Be Found
    Next

End Sub
```

Fill_EnglishwordsMiddleAge:

```
Public Sub Fill_EnglishwordsMiddleAge() ''Fills the grid with English words appropriate for the Middle age user
    EnglishwordsMiddleAge() 'Calls the subroutine EnglishwordsMiddleAge

    'Intialises gridarray to a blank space
    For Row = 0 To 13 Step 1
        For column = 0 To 13 Step 1
            GridArray(Row, column) = " "
        Next
    Next


    For i As Integer = 1 To 8
        intrandom(i) = ((arrenglishwordsMiddleAge.Length - 1) * Rnd())  'Generates a new random number for the locations in intrandom
    Next

    CheckforDoubleWord() 'Calls subroutine CheckforDoubleWord

    'The following subroutine are called to fill the random word in the directions stated below
    FillEast(arrenglishwordsMiddleAge(intrandom(1)), "science") 'East
    FillWest(arrenglishwordsMiddleAge(intrandom(2)), "science") 'West
    FillSouth(arrenglishwordsMiddleAge(intrandom(3)), "science") 'South
    FillNorth(arrenglishwordsMiddleAge(intrandom(4)), "science") 'North
    FillSouthEast(arrenglishwordsMiddleAge(intrandom(5)), "science") 'SouthEast
    FillNorthWest(arrenglishwordsMiddleAge(intrandom(6)), "science") 'NorthWest
    FillNortheast(arrenglishwordsMiddleAge(intrandom(7)), "science") 'Northeast
    FillSouthWest(arrenglishwordsMiddleAge(intrandom(8)), "science") 'Southwest

    'Sets the property of the labels in "arritems"
    For i = 1 To 8
        arrItems(i).Left = 10
        arrItems(i).Top = i * 20
        arrItems(i).BringToFront()
        arrItems(i).Visible = True
        arrItems(i).BackColor = Color.Transparent
        pnlitems.Controls.Add(arrItems(i)) 'Add Items To Be Found
    Next

End Sub
```

Fill_MathswordsMiddleAge:

```
Private Sub Fill_MathswordsMiddleAge() 'Fills the grid with Maths words appropriate for the Middle age user
    MathswordsMiddleAge() 'Calls subroutine MathswordsMiddleAge

    'Intialises gridarray to a blank space
    For Row = 0 To 13 Step 1
        For column = 0 To 13 Step 1
            GridArray(Row, column) = " "
        Next
    Next

    For i = 1 To 8
        intrandom(i) = ((arrmathswordsMiddleAge.Length - 1) * Rnd())  'Generates a new random number for the locations in intrandom
    Next

    CheckforDoubleWord()  'Calls subroutine CheckforDoubleWord

    'The following subroutine are called to fill the random word in the directions stated below
    FillEast(arrmathswordsMiddleAge(intrandom(1)), "maths") 'East
    FillWest(arrmathswordsMiddleAge(intrandom(2)), "maths") 'North
    FillSouth(arrmathswordsMiddleAge(intrandom(3)), "maths") 'South
    FillNorth(arrmathswordsMiddleAge(intrandom(4)), "maths") 'North
    FillSouthEast(arrmathswordsMiddleAge(intrandom(5)), "maths") 'SouthEast
    FillNorthWest(arrmathswordsMiddleAge(intrandom(6)), "maths") 'NorthWest
    FillNortheast(arrmathswordsMiddleAge(intrandom(7)), "maths") 'NorthEast
    FillSouthWest(arrmathswordsMiddleAge(intrandom(8)), "maths") 'SouthWest

    'Sets the property of the labels in "arritems"
    For i = 1 To 8
        arrItems(i).Left = 10
        arrItems(i).Top = i * 20
        arrItems(i).BringToFront()
        arrItems(i).Visible = True
        arrItems(i).BackColor = Color.Transparent
        pnlitems.Controls.Add(arrItems(i)) 'Add Items To Be Found
    Next

End Sub
```

Fill_SciencewordsMiddleAge:

```
Private Sub Fill_sciencewordsMiddleAge() 'Fills the grid with Science words appropriate for the Middle age user
    SciencewordsMiddleAge() 'Calls subroutine SciencewordsMiddleAge

    'Intialises gridarray to a blank space
    For Row = 0 To 13 Step 1
        For column = 0 To 13 Step 1
            GridArray(Row, column) = " "
        Next
    Next

    For i = 1 To 8
        intrandom(i) = ((arrsciencewordsEldestAge.Length - 1) * Rnd()) 'Generates a new random number for the locations in intrandom
    Next

    CheckforDoubleWord() 'Calls subroutine CheckforDoubleWord

    'The following subroutine are called to fill the random word in the directions stated below
    FillEast(arrsciencewordsMiddleAge(intrandom(1)), "science") 'East
    FillWest(arrsciencewordsMiddleAge(intrandom(2)), "science") 'West
    FillSouth(arrsciencewordsMiddleAge(intrandom(3)), "science") 'South
    FillNorth(arrsciencewordsMiddleAge(intrandom(4)), "science") 'North
    FillSouthEast(arrsciencewordsMiddleAge(intrandom(5)), "science") 'SouthEast
    FillNorthWest(arrsciencewordsMiddleAge(intrandom(6)), "science") 'NorthWest
    FillNortheast(arrsciencewordsMiddleAge(intrandom(7)), "science") 'NorthEast
    FillSouthWest(arrsciencewordsMiddleAge(intrandom(8)), "science") 'SouthWest

    'Sets the property of the labels in "arritems"
    For i = 1 To 8
        arrItems(i).Left = 10
        arrItems(i).Top = i * 20
        arrItems(i).BringToFront()
        arrItems(i).Visible = True
        arrItems(i).BackColor = Color.Transparent
        pnlitems.Controls.Add(arrItems(i)) 'Add Items To Be Found
    Next
End Sub
```

Fill_EnglishwordsEldestAge:

```
Private Sub Fill_EnglishWordsEldestAge()  'Fills the grid with English words appropriate for the Eldest age user
    EnglishWordsEldestAge() 'Calls subroutine EnglishwordsEldestAge

    'Intialises gridarray to a blank space
    For Row = 0 To 13 Step 1
        For column = 0 To 13 Step 1
            GridArray(Row, column) = " "
        Next
    Next

    For i = 1 To 8
        intrandom(i) = ((arrenglishwordsEldestAge.Length - 1) * Rnd()) 'Generates a new random number for the locations in intrandom
    Next

    CheckforDoubleWord() 'Calls subroutine CheckforDoubleWord

    'The following subroutine are called to fill the random word in the directions stated below
    FillEast(arrenglishwordsEldestAge(intrandom(1)), "English") 'East
    FillWest(arrenglishwordsEldestAge(intrandom(2)), "English") 'West
    FillSouth(arrenglishwordsEldestAge(intrandom(3)), "English") 'South
    FillNorth(arrenglishwordsEldestAge(intrandom(4)), "English") 'North
    FillSouthEast(arrenglishwordsEldestAge(intrandom(5)), "English") 'SouthEast
    FillNorthWest(arrenglishwordsEldestAge(intrandom(6)), "English") 'NorhWest
    FillNortheast(arrenglishwordsEldestAge(intrandom(7)), "English") 'NorthEast
    FillSouthWest(arrenglishwordsEldestAge(intrandom(8)), "English") 'SouthWest

    'Sets the property of the labels in "arritems"
    For i = 1 To 8
        arrItems(i).Left = 10
        arrItems(i).Top = i * 20
        arrItems(i).BringToFront()
        arrItems(i).Visible = True
        arrItems(i).BackColor = Color.Transparent
        pnlitems.Controls.Add(arrItems(i)) 'Add Items To Be Found
    Next
End Sub
```

Fill_MathswordsEldestAge:

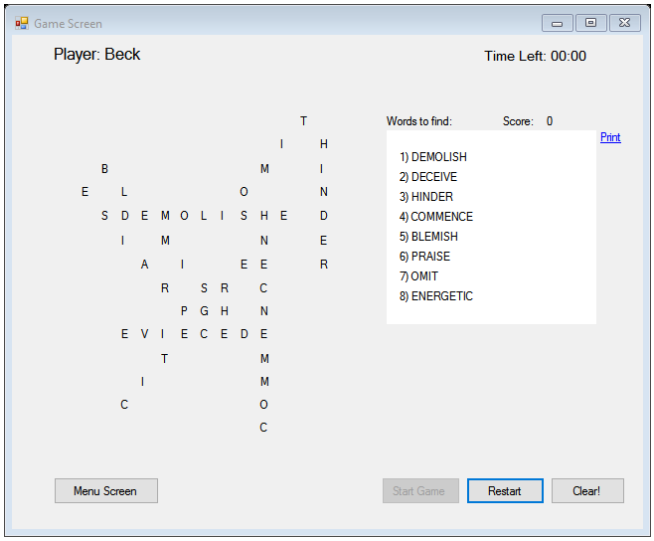
```
Private Sub Fill_MathswordsEldestAge() 'Fills the grid with Maths words appropriate for the Eldest age user
    MathsWordsEldestAge() 'Calls subroutine MathswordsEldestAge

    'Intialises gridarray to a blank space
    For Row = 0 To 13 Step 1
        For column = 0 To 13 Step 1
            GridArray(Row, column) = " "
        Next
    Next

    For i = 1 To 8
        intrandom(i) = ((arrmathswordsEldestAge.Length - 1) * Rnd()) 'Generates a new random number for the locations in intrandom
    Next

    CheckforDoubleWord()  'Generates a new random number for the locations in intrandom

    'The following subroutine are called to fill the random word in the directions stated below
    FillEast(arrmathswordsEldestAge(intrandom(1)), "Maths") 'East
    FillWest(arrmathswordsEldestAge(intrandom(2)), "Maths") 'West
    FillSouth(arrmathswordsEldestAge(intrandom(3)), "Maths") 'South
    FillNorth(arrmathswordsEldestAge(intrandom(4)), "Maths") 'North
    FillSouthEast(arrmathswordsEldestAge(intrandom(5)), "Maths") 'SouthEast
    FillNorthWest(arrmathswordsEldestAge(intrandom(6)), "Maths") 'NorthWest
    FillNortheast(arrmathswordsEldestAge(intrandom(7)), "Maths") 'NorthEast
    FillSouthWest(arrmathswordsEldestAge(intrandom(8)), "Maths") 'SouthWest

    'Sets the property of the labels in "arritems"
    For i = 1 To 8
        arrItems(i).Left = 10
        arrItems(i).Top = i * 20
        arrItems(i).BringToFront()
        arrItems(i).Visible = True
        arrItems(i).BackColor = Color.Transparent
        pnlitems.Controls.Add(arrItems(i)) 'Add Items To Be Found
    Next
End Sub
```

Fill_SciencewordsEldestAge:

```
Private Sub Fill_ScienceWordsEldestAge() 'Fills the grid with Science words appropriate for the Eldest age user
    ScienceWordsEldestAge() 'Calls subroutine SciencewordsEldestAge

    'Intialises gridarray to a blank space
    For Row = 0 To 13 Step 1
        For column = 0 To 13 Step 1
            GridArray(Row, column) = " "
        Next
    Next

    For i = 1 To 8
        intrandom(i) = ((arrsciencewordsEldestAge.Length - 1) * Rnd()) 'Generates a new random number for the locations in intrandom
    Next

    CheckforDoubleWord() 'Generates a new random number for the locations in intrandom

    'The following subroutine are called to fill the random word in the directions stated below
    FillEast(arrsciencewordsEldestAge(intrandom(1)), "Science") 'East
    FillWest(arrsciencewordsEldestAge(intrandom(2)), "Science") 'West
    FillSouth(arrsciencewordsEldestAge(intrandom(3)), "Science") 'South
    FillNorth(arrsciencewordsEldestAge(intrandom(4)), "Science") 'North
    FillSouthEast(arrsciencewordsEldestAge(intrandom(5)), "Science") 'SouthEast
    FillNorthWest(arrsciencewordsEldestAge(intrandom(6)), "Science") 'NorthWest
    FillNortheast(arrsciencewordsEldestAge(intrandom(7)), "Science") 'NorthEast
    FillSouthWest(arrsciencewordsEldestAge(intrandom(8)), "Science") 'SouthWest

    'Sets the property of the labels in "arritems"
    For i = 1 To 8
        arrItems(i).Left = 10
        arrItems(i).Top = i * 20
        arrItems(i).BringToFront()
        arrItems(i).Visible = True
        arrItems(i).BackColor = Color.Transparent
        pnlitems.Controls.Add(arrItems(i)) 'Add Items To Be Found
    Next
End Sub
```

I will now perform alpha testing to test the subroutine Fill_Englishwordsmiddleage.

(I select the english theme and the 11-15 age)



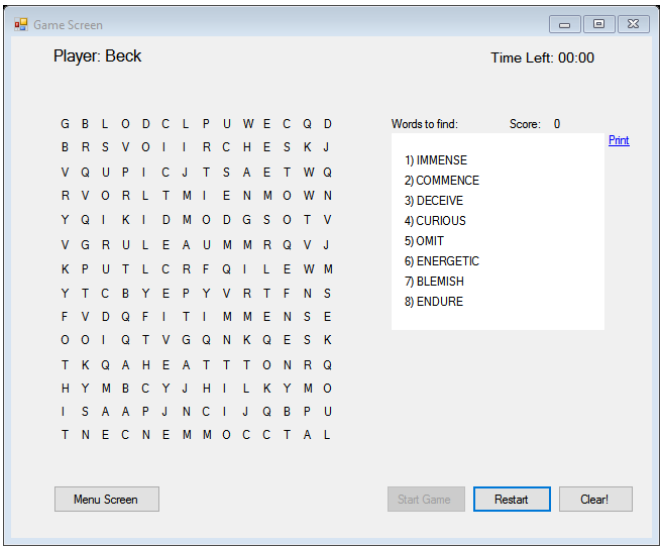
(The subroutine works perfectly)

Its good to see that the words are place in the grid, so what remains is to fill in the remaining blank spaces with randomly generated characters. I shall do this by writing the scramable subroutine.

Subroutine Scramble:

```
Private Sub Scramble() 'Fill In Blank Labels Randomly
    Dim strScramble As String = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" 'Each Character of the Alphabet - To Fill In Blanks
    Dim i As Integer = 0 'Start Index
    Dim RandGen As New Random(Now.Millisecond) 'Generate Random Number
    Dim charArr() As Char = strScramble.ToCharArray 'Break Alphabet String Apart Into Separate Chars

    For row = 0 To 13 ' Loop through each row in grid
        For column = 0 To 13 'Loop through each column in grid
            Dim RandIndex As Integer = RandGen.Next(0, 25) 'Generate New Random Letter
            If GridArray(row, column) = " " Then 'all blank spaces are checked
                GridArray(row, column) = charArr(RandIndex).ToString 'Fill in random letter
                arrunscramble(row, column) = "Char" 'Fill in string "char" to arrunscramble so grid can be unscrambled when required
            End If
        Next
    Next
End Sub
```

I will now test if the scramble subroutine replaces all the blanks spaces labels with a randomly generated letter.



(All blank spaces are replace with a randomly generated character)

The subroutine clearly works and all the blank labels are replaced with a randomly generated letter.

This concludes the end of the module, since all the algorithms are now functioning well.

Module 2:

Module 2 is all about the user playing the game. The size of the game screen will still be 800 x 800 pixels, as requested by my users.

Subroutine Lbl_MouseClick:

This subroutine is executed every time the user clicks on a label (letter) in the grid. Furthermore, it is the largest subroutine in the game and contains many algorithms, which are listed in italics below. I will now list the various algorithms that were written in this subroutine.

Changing the colour of the clicked label:

```
Private Sub lbl_MouseClick(ByVal sender As Object, ByVal e As MouseEventArgs) 'Common Event Handler For Labels
    Dim tempstr As String 'Temporarily stores a string when required
    Dim lblTemp As Label = DirectCast(sender, Label) 'System get the clicked Label

    If lblTemp.ForeColor <> Color.Red Then
        lblTemp.ForeColor = Color.Blue 'Selected label's text color is changed
    End If

    lblSel = lblSel & lblTemp.Text 'Concatenate Sequence of Letters

    Gridlbls(False) 'Disables all the labels in the grid
```

This algorithm changes the colour of the clicked label to blue, so that the user knows which letters (labels) they have clicked on. The colour blue was used since it was the colour that my users want to see, as evident from the results in the second survey on page 14.

Enabling the 8 labels surrounding label temp:

As a continuation from the above, the following algorithm will enable only the 8 labels that surround the clicked letter. This restricts the user to only a small space of clickable letters, this would help to minimise cheating from clicking on letters that lie in different locations in the grid.

```
If lblTemp.Enabled = False Then 'Checks if clicked label is disabled
    lblTemp.Enabled = True 'Enables the clicked label

    'The following code is to determine which label has been clicked on
    Dim lblposition As Integer 'Stores the label number of the clicked label
    If lblTemp.Name.Length = 6 Then 'Labels 1 to 9 each have 6 characters in the properties name, for example "label1"
        lblposition = Val(Microsoft.VisualBasic.Right(lblTemp.Name, 1)) 'Stores the digit from 1-9
    ElseIf lblTemp.Name.Length = 7 Then 'Labels 10 to 99 each have 7 characters  in the properties name , for example "label10"
        lblposition = Val(Microsoft.VisualBasic.Right(lblTemp.Name, 2)) 'Stores the number from 10- 99
    ElseIf lblTemp.Name.Length = 8 Then 'Labels 100 to 196 each have 8 characters in the properties name, for exmaple "label100"
        lblposition = Val(Microsoft.VisualBasic.Right(lblTemp.Name, 3)) 'Stores the number from 100 - 196
    End If

    mystack.Push(lblposition) 'Insert the value of labelposition at the top of the stack

    'The following code will enable the 8 labels surrounding the clicked label
    Dim northpos, northwestpos, northeastpos, southpos, southwestpos, southeastpos, westpos, eastpos As Integer

    northpos = lblposition - 14 'label number of the clicked label minus 14
    northwestpos = lblposition - 15 'label number of the clicked label minus 15
    northeastpos = lblposition - 13 'label number of the clicked label minus 13
    southpos = lblposition + 14 'label number of the clicked label plus 14
    southeastpos = lblposition + 13 'label number of the clicked label plus 13
    southwestpos = lblposition + 15 'label number of the clicked label plus 15
    westpos = lblposition - 1 'label number of the clicked label minus 1
    eastpos = lblposition + 1 'label number of the clicked label add 1

    If northwestpos < 1 And northpos < 1 And northeastpos < 1 Then 'Checks if North row does not fit on the grid
        northwestpos += 15
        northpos += 14
        northeastpos += 13 '
    End If

    If southeastpos > 196 And southwestpos > 196 And southpos > 196 Then 'Checks if the south row will not fit on the grid
        southeastpos -= 13
        southpos -= 14
        southwestpos -= 15
    End If

    If eastpos > 196 Then 'Checks if the east position is outside the grid limit
        eastpos -= 1
    End If

    If westpos < 1 Then 'Checks if the west position is outside the grid limit
        westpos += 1
    End If

    'Enables the 8 labels surrounding the clicked label
    Me.Controls("label" & northpos).Enabled = True
    Me.Controls("label" & eastpos).Enabled = True
    Me.Controls("label" & westpos).Enabled = True
    Me.Controls("label" & southpos).Enabled = True
    Me.Controls("label" & southwestpos).Enabled = True
    Me.Controls("label" & northwestpos).Enabled = True
    Me.Controls("label" & northeastpos).Enabled = True
    Me.Controls("label" & southeastpos).Enabled = True
```

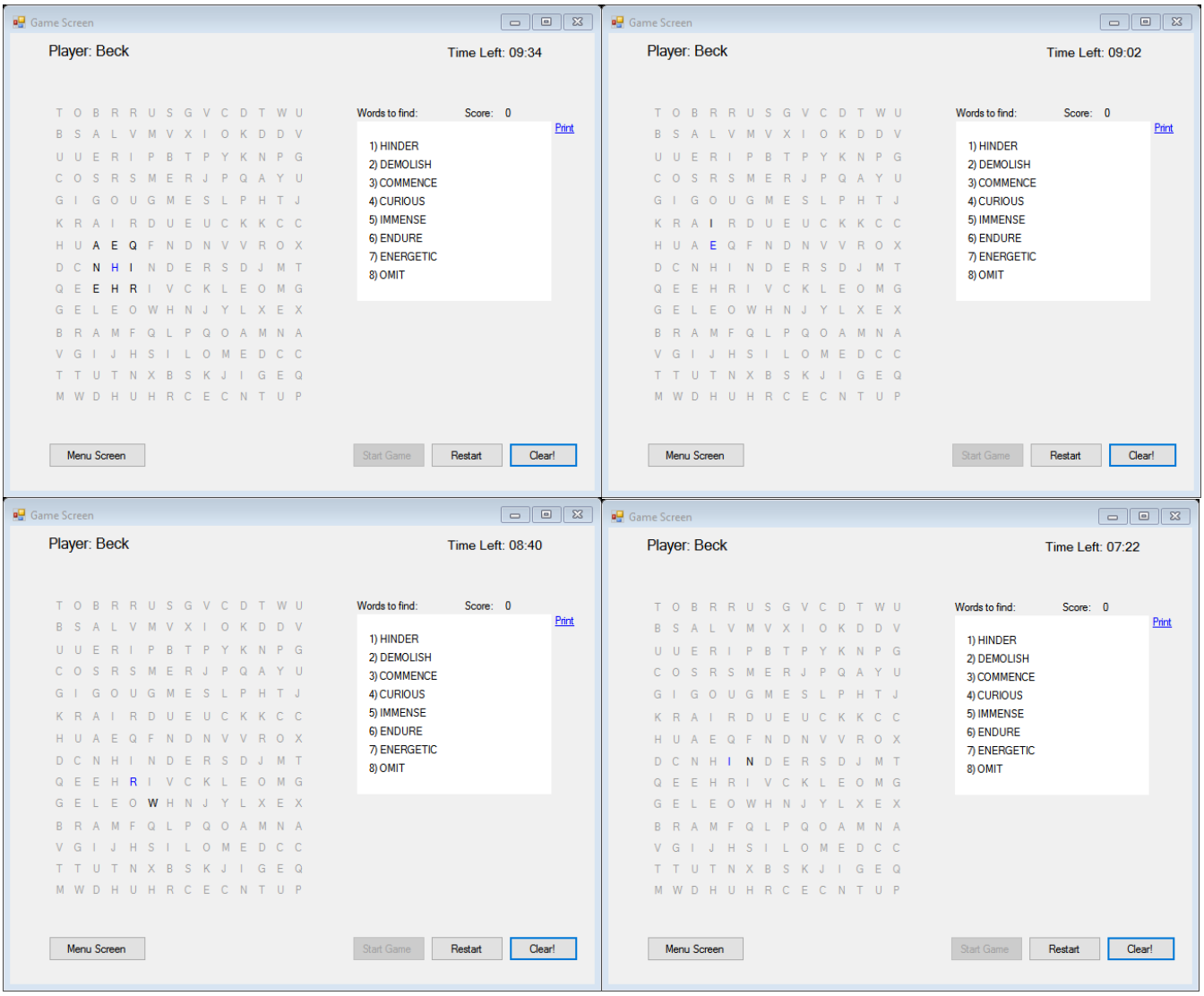
Further disabling the directional label:

The next stage was to further disable the directional labels that the user will not be using. By disabling the labels, I could ensure that the user could not cheat by selecting random letters to complete a word. The following screenshots show the code for only enabling the direction labels the user wishes to travel.

```
'This section will work out which direction the user wishes to go, and so will disable the other unessessary labels.
'It will calculate the direction by checking the colour of the pervious clicked label (lbltemp)
If Me.Controls("label" & eastpos).ForeColor = Color.Blue Then 'Checks if East label (from the clicked) is blue
    'All labels are disabled except for the west label and lbltemp(clicked label)
    Me.Controls("label" & northpos).Enabled = False
    Me.Controls("label" & eastpos).Enabled = False
    Me.Controls("label" & westpos).Enabled = True
    Me.Controls("label" & southpos).Enabled = False
    Me.Controls("label" & southwestpos).Enabled = False
    Me.Controls("label" & northwestpos).Enabled = False
    Me.Controls("label" & northeastpos).Enabled = False
    Me.Controls("label" & southeastpos).Enabled = False

ElseIf Me.Controls("label" & westpos).ForeColor = Color.Blue Then 'Checks if West label (from clicked label) is blue
    'All labels are disabled except for the east label and lbltemp(clicked label)
    Me.Controls("label" & northpos).Enabled = False
    Me.Controls("label" & eastpos).Enabled = True
    Me.Controls("label" & westpos).Enabled = False
    Me.Controls("label" & southpos).Enabled = False
    Me.Controls("label" & southwestpos).Enabled = False
    Me.Controls("label" & northwestpos).Enabled = False
    Me.Controls("label" & northeastpos).Enabled = False
    Me.Controls("label" & southeastpos).Enabled = False

ElseIf Me.Controls("label" & northwestpos).ForeColor = Color.Blue Then 'Checks if the Northwest label (from clicked label) is blue
    'All labels are disabled except for the Southwest label and lbltemp(clicked label)
    Me.Controls("label" & northpos).Enabled = False
    Me.Controls("label" & eastpos).Enabled = False
    Me.Controls("label" & westpos).Enabled = False
    Me.Controls("label" & southpos).Enabled = False
    Me.Controls("label" & southwestpos).Enabled = True
    Me.Controls("label" & northwestpos).Enabled = False
    Me.Controls("label" & northeastpos).Enabled = False
    Me.Controls("label" & southeastpos).Enabled = False

ElseIf Me.Controls("label" & northeastpos).ForeColor = Color.Blue Then 'Checks if the NorthEast label (from clicked label) is blue
    'All labels are disabled, except for the SouthEast label and lbltemp(clicked label)
    Me.Controls("label" & northpos).Enabled = False
    Me.Controls("label" & eastpos).Enabled = False
    Me.Controls("label" & westpos).Enabled = False
    Me.Controls("label" & southpos).Enabled = False
    Me.Controls("label" & southwestpos).Enabled = False
    Me.Controls("label" & northwestpos).Enabled = False
    Me.Controls("label" & northeastpos).Enabled = False
    Me.Controls("label" & southeastpos).Enabled = True

ElseIf Me.Controls("label" & southwestpos).ForeColor = Color.Blue Then 'Checks if the SouthWest label (from clicked label) is blue
    'All labels are disabled, except for the Northwest label and lbltemp(clicked label)
    Me.Controls("label" & northpos).Enabled = False
    Me.Controls("label" & eastpos).Enabled = False
    Me.Controls("label" & westpos).Enabled = False
    Me.Controls("label" & southpos).Enabled = False
    Me.Controls("label" & southwestpos).Enabled = False
    Me.Controls("label" & northwestpos).Enabled = True
    Me.Controls("label" & northeastpos).Enabled = False
    Me.Controls("label" & southeastpos).Enabled = False

ElseIf Me.Controls("label" & southeastpos).ForeColor = Color.Blue Then 'Checks if the SouthEast label (from clicked label) is blue
```

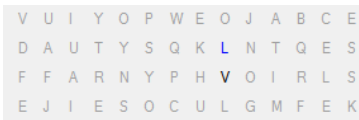
```
ElseIf Me.Controls("label" & southeastpos).ForeColor = Color.Blue Then 'Checks if the SouthEast label (from clicked label) is blue
    'All labels are disabled, except for the NorthEast label and lbltemp(clicked label)
    Me.Controls("label" & northpos).Enabled = False
    Me.Controls("label" & eastpos).Enabled = False
    Me.Controls("label" & westpos).Enabled = False
    Me.Controls("label" & southpos).Enabled = False
    Me.Controls("label" & southwestpos).Enabled = False
    Me.Controls("label" & northwestpos).Enabled = False
    Me.Controls("label" & northeastpos).Enabled = True
    Me.Controls("label" & southeastpos).Enabled = False
End If

If (southpos + 14) > 196 Then 'Adds 14 to southpos to check if the south position is outside the limits of the grid
    'All directional labels are enabled temporarily (until next lbl click)
    Me.Controls("label" & northpos).Enabled = True
    Me.Controls("label" & eastpos).Enabled = True
    Me.Controls("label" & westpos).Enabled = True
    Me.Controls("label" & southpos).Enabled = True
    Me.Controls("label" & southwestpos).Enabled = True
    Me.Controls("label" & northwestpos).Enabled = True
    Me.Controls("label" & northeastpos).Enabled = True
    Me.Controls("label" & southeastpos).Enabled = True

ElseIf Me.Controls("label" & southpos).ForeColor = Color.Blue Then 'If the south label (from the clicked label) is blue
    'All labels are disabled, except for the North label and lbltemp (clicked label)
    Me.Controls("label" & northpos).Enabled = True
    Me.Controls("label" & eastpos).Enabled = False
    Me.Controls("label" & westpos).Enabled = False
    Me.Controls("label" & southpos).Enabled = False
    Me.Controls("label" & southwestpos).Enabled = False
    Me.Controls("label" & northwestpos).Enabled = False
    Me.Controls("label" & northeastpos).Enabled = False
    Me.Controls("label" & southeastpos).Enabled = False
End If

If (northpos - 14) < 1 Then 'Subtracts 14 from northpos to check if the northposition is outside the limits of the grid
    'All directional labels are enabled temporarily (until next lbl click)
    Me.Controls("label" & northpos).Enabled = True
    Me.Controls("label" & eastpos).Enabled = True
    Me.Controls("label" & westpos).Enabled = True
    Me.Controls("label" & southpos).Enabled = True
    Me.Controls("label" & southwestpos).Enabled = True
    Me.Controls("label" & northwestpos).Enabled = True
    Me.Controls("label" & northeastpos).Enabled = True
    Me.Controls("label" & southeastpos).Enabled = True

ElseIf Me.Controls("label" & northpos).ForeColor = Color.Blue Then 'If the North label (from the clicked label) is blue
    'All labels are disabled, except for the South label and lbltemp (clicked label)
    Me.Controls("label" & northpos).Enabled = False
    Me.Controls("label" & eastpos).Enabled = False
    Me.Controls("label" & westpos).Enabled = False
    Me.Controls("label" & southpos).Enabled = True
    Me.Controls("label" & southwestpos).Enabled = False
    Me.Controls("label" & northwestpos).Enabled = False
    Me.Controls("label" & northeastpos).Enabled = False
    Me.Controls("label" & southeastpos).Enabled = False
End If
```

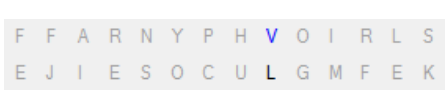
I will now take a screenshot of the game to test the algorithms above and to make sure that everything works:

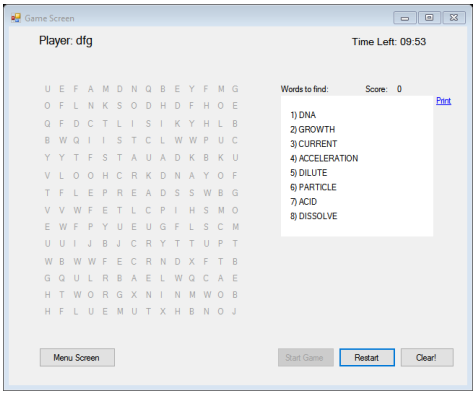


From the examples above, we can see that once the user selects a letter, the colour of lbltemp changes to blue. Furthermore, as a 'potential' end-user I am only allowed to click in the direction I wish to complete the word, this would prevent the user from cheating.

After testing the game a number of times, I had noticed that the user was not unable to select a letter, in any directions, if the letter reached the bottom row or top row. By clicking on the letter in the top or bottom row resulted in a run-time error, see the example below.







I then decided to add a breakpoint to locate the error:

```
If (southpos + 14) > 196 Then
    Me.Controls("label" & northpos).Enabled = True
    Me.Controls("label" & eastpos).Enabled = True
    Me.Controls("label" & westpos).Enabled = True
    Me.Controls("label" & southpos).Enabled = True
    Me.Controls("label" & southwestpos).Enabled = True
    Me.Controls("label" & northwestpos).Enabled = True
    Me.Controls("label" & northeastpos).Enabled = True
    Me.Controls("label" & southeastpos).Enabled = True

ElseIf Me.Controls("label" & southpos).ForeColor = Color.Blue Then
    Me.Controls("label" & northpos).Enabled = True
    Me.Controls("label" & eastpos).Enabled = False
    Me.Controls("label" & westpos).Enabled = False
    Me.Controls("label" & southpos).Enabled = False
    Me.Controls("label" & southwestpos).Enabled = False
    Me.Controls("label" & northwestpos).Enabled = False
    Me.Controls("label" & northeastpos).Enabled = False
    Me.Controls("label" & southeastpos).Enabled = False
End If
```

(A breakpoint and stepping being used)

southpos + 14	176	Integer
southpos	162	Integer

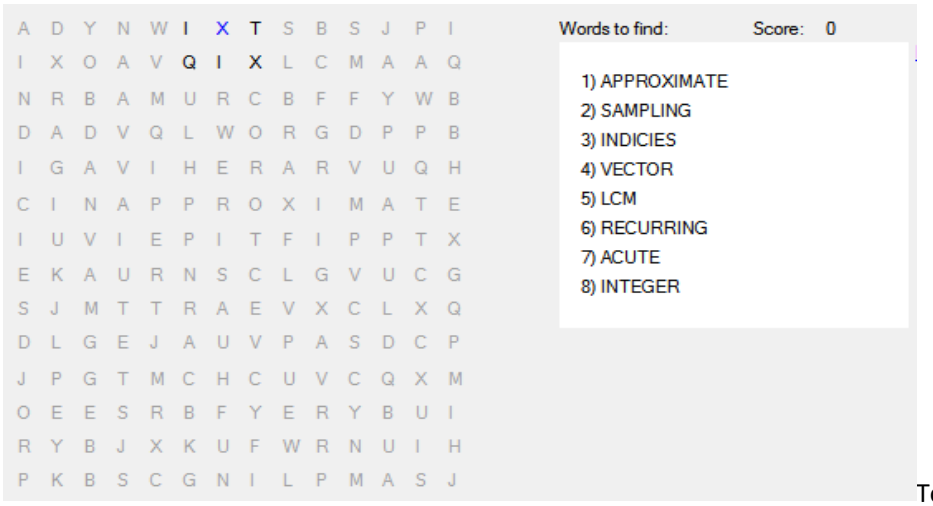
(A watch to displays any changes to the value of variable (Southpos) throughout the execution of the program.

After following the sequence of the breakpoint, I had realised that the code for enabling the directional labels was not correctly nested inside the for loop. Therefore, I immediately decided to update the code:

```
If (southpos + 14) > 196 Then 'Checks if south label is outside the grid limits
    Me.Controls("label" & northpos).Enabled = True
    Me.Controls("label" & eastpos).Enabled = True
    Me.Controls("label" & westpos).Enabled = True
    Me.Controls("label" & southpos).Enabled = True
    Me.Controls("label" & southwestpos).Enabled = True
    Me.Controls("label" & northwestpos).Enabled = True
    Me.Controls("label" & northeastpos).Enabled = True
    Me.Controls("label" & southeastpos).Enabled = True

ElseIf (northpos - 14) < 1 Then 'Checks if the north label is outside the grid limits
    Me.Controls("label" & northpos).Enabled = True
    Me.Controls("label" & eastpos).Enabled = True
    Me.Controls("label" & westpos).Enabled = True
    Me.Controls("label" & southpos).Enabled = True
    Me.Controls("label" & southwestpos).Enabled = True
    Me.Controls("label" & northwestpos).Enabled = True
    Me.Controls("label" & northeastpos).Enabled = True
    Me.Controls("label" & southeastpos).Enabled = True

ElseIf Me.Controls("label" & northpos).ForeColor = Color.Blue Then 'Checks if the north label is blue
    'User can only select the label in the south direction
    Me.Controls("label" & northpos).Enabled = False
    Me.Controls("label" & eastpos).Enabled = False
    Me.Controls("label" & westpos).Enabled = False
    Me.Controls("label" & southpos).Enabled = True
    Me.Controls("label" & southwestpos).Enabled = False
    Me.Controls("label" & northwestpos).Enabled = False
    Me.Controls("label" & northeastpos).Enabled = False
    Me.Controls("label" & southeastpos).Enabled = False

ElseIf Me.Controls("label" & southpos).ForeColor = Color.Blue Then 'Checks if the South label is blue
    'User can only select the label in the north direction
    Me.Controls("label" & northpos).Enabled = True
    Me.Controls("label" & eastpos).Enabled = False
    Me.Controls("label" & westpos).Enabled = False
    Me.Controls("label" & southpos).Enabled = False
    Me.Controls("label" & southwestpos).Enabled = False
    Me.Controls("label" & northwestpos).Enabled = False
    Me.Controls("label" & northeastpos).Enabled = False
    Me.Controls("label" & southeastpos).Enabled = False
End If
```

I can now select different letters even if it reaches the bottom or top row:

Bottom Row letter click



Top Row letter click

Algorithm for found word:

Now that my users can select a letter in any direction, I then decided to code the algorithm for when a word has been found. The game must detect what the user's age is and the theme selected. For example, a young age user is one who is <10.

EnglishwordsYoungAge:

```
ElseIf Menu_Screen.CkbxflagEnglish = True Then 'Checks if the user has selected an English theme
    If Menu_Screen.CkbxflagYoungAge = True Then 'System will check if the user is of a young age
        For i As Integer = 0 To (arrenglishwordsYoungAge.Length - 1) 'Loops through every element in the array
            If lblSel = arrenglishwordsYoungAge(i) Then 'Determine Matches
                Dim count As Integer = 0 'Temporary counter
                Do 'Loops until the stack is empty
                    count += 1 'Increment count
                    Me.Controls("label" & mystack.Pop()).forecolor = Color.Red 'Change the colour of the label in the stack to red and removes the top item
                Loop Until mystack.Count = 0

                lblTemp.ForeColor = Color.Red 'Change the colour of the clicked label to red
                intScore = Val(LblScore.Text) 'Takes the numeric value of the string in lblscore.text
                intScore = intScore + 20 'Add 20 to stack pointer
                LblScore.Text = intScore 'The text field of
                Beep() 'Outputs a beeps sound
                intItemsFound += 1 'Increment intitemsfound
                Gridlbls(True) 'Enables all the labels in grid

                'Output String - used for updating the word list in the pannel box
                arrenglishwordsYoungAge(i) = arrenglishwordsYoungAge(i) & Space(1) & "-" & Space(1) & "FOUND!"
                tempstr = arrenglishwordsYoungAge(i)

                'Loop removes unessessary symbols in the string so only the word remains in the array.
                For intcount As Integer = 1 To 8 'Loop through arritems
                    Dim intwordlength As Integer 'Number of characters in array items with index intcount
                    intwordlength = Len(arrItems(intcount).Text) - 3 'Calcualtes length of the word
                    arrItems(intcount).Text = Microsoft.VisualBasic.Right(arrItems(intcount).Text, intwordlength) 'Removes the number and ")" from the string. i.e 1) Boom
                Next

                For number As Integer = 1 To 8 'Loops through all the labels in array
                    If arrItems(number).Text = lblSel Then 'Determines if the word found matches the word in the array (words to find)
                        arrItems(number).Text = number.ToString & ")" & Space(1) & tempstr 'Updates the word list so the word has text found next to it.
                    ElseIf arrItems(number).Text <> lblSel Then 'If word does not match then word list remains the same
                        arrItems(number).Text = number.ToString & ")" & Space(1) & arrItems(number).Text
                    End If
                Next

                lblSel = "" 'Reset Everything For New Matches
                tempstr = "" 'Reset Everything For New Matches
            End If
        Next
```

EnglishwordsMiddleAge:

```
ElseIf Menu_Screen.CkbxflagMiddleAge = True Then 'System will check if the user is of a middle age group
    For i As Integer = 0 To (arrenglishwordsMiddleAge.Length - 1) 'Loops through every element in the array
        If lblSel = arrenglishwordsMiddleAge(i) Then 'Determine Matches
            Dim count As Integer = 0 'Temporary counter
            Do 'Loops until the stack is empty
                count += 1 'Increment count
                Me.Controls("label" & mystack.Pop()).forecolor = Color.Red 'Change the colour of the label in the stack to red and removes the top item
            Loop Until mystack.Count = 0

            lblTemp.ForeColor = Color.Red 'Change the colour of the clicked label to red
            intScore = Val(LblScore.Text) 'Takes the numeric value of the string in lblscore.text
            intScore = intScore + 20 'Add 20 to stack pointer
            LblScore.Text = intScore 'The text field of
            Beep() 'Outputs a beeps sound
            intItemsFound += 1 'Increment intitemsfound
            Gridlbls(True) 'Enables all the labels in grid

            'Output String - used for updating the word list in the pannel box
            arrenglishwordsMiddleAge(i) = arrenglishwordsMiddleAge(i) & Space(1) & "-" & Space(1) & "FOUND!"
            tempstr = arrenglishwordsMiddleAge(i)

            'Loop removes unessessary symbols in the string so only the word remains in the array.
            For intcount As Integer = 1 To 8 'Loop through arritems
                Dim intwordlength As Integer 'Number of characters in array items with index intcount
                intwordlength = Len(arrItems(intcount).Text) - 3 'Calcualates length of the word
                arrItems(intcount).Text = Microsoft.VisualBasic.Right(arrItems(intcount).Text, intwordlength) 'Removes the number and ")" from the string. i.e 1) Boom
            Next

            For number As Integer = 1 To 8 'Loops through all the labels in array
                If arrItems(number).Text = lblSel Then 'Determines if the word found matches the word in the array (words to find)
                    arrItems(number).Text = number.ToString & ")" & Space(1) & tempstr 'Updates the word list so the word has text found next to it.
                ElseIf arrItems(number).Text <> lblSel Then 'If word does not match then word list remains the same
                    arrItems(number).Text = number.ToString & ")" & Space(1) & arrItems(number).Text
                End If
            Next

            lblSel = "" 'Reset Everything For New Matches
            tempstr = "" 'Reset Everything For New Matches
        End If
    Next
```

EnglishwordsEldestAge:

```
ElseIf Menu_Screen.CkbxflagEldestAge = True Then 'System will check if the user is of the Eldest age group
    For i = 0 To (arrenglishwordsEldestAge.Length - 1) 'Loops through every element in the array
        If lblSel = arrenglishwordsEldestAge(i) Then 'Determine Matches
            Dim count As Integer = 0 'Temporary counter
            Do 'Loops until the stack is empty
                count += 1 'Increment count
                Me.Controls("label" & mystack.Pop()).forecolor = Color.Red 'Change the colour of the label in the stack to red and removes the top item
            Loop Until mystack.Count = 0

            lblTemp.ForeColor = Color.Red 'Change the colour of the clicked label to red
            intScore = Val(LblScore.Text) 'Takes the numeric value of the string in lblscore.text
            intScore = intScore + 20 'Add 20 to stack pointer
            LblScore.Text = intScore 'The text field of
            Beep() 'Outputs a beeps sound
            intItemsFound += 1 'Increment intitemsfound
            Gridlbls(True) 'Enables all the labels in grid

            'Output String - used for updating the word list in the pannel box
            arrenglishwordsEldestAge(i) = arrenglishwordsEldestAge(i) & Space(1) & "-" & Space(1) & "FOUND!"
            tempstr = arrenglishwordsEldestAge(i)

            'Loop removes unessessary symbols in the string so only the word remains in the array.
            For intcount As Integer = 1 To 8 'Loop through arritems
                Dim intwordlength As Integer 'Number of characters in array items with index intcount
                intwordlength = Len(arrItems(intcount).Text) - 3 'Calcualates length of the word
                arrItems(intcount).Text = Microsoft.VisualBasic.Right(arrItems(intcount).Text, intwordlength) 'Removes the number and ")" from the string. i.e 1) Boom
            Next

            For number As Integer = 1 To 8 'Loops through all the labels in array
                If arrItems(number).Text = lblSel Then 'Determines if the word found matches the word in the array (words to find)
                    arrItems(number).Text = number.ToString & ")" & Space(1) & tempstr 'Updates the word list so the word has text found next to it.
                ElseIf arrItems(number).Text <> lblSel Then 'If word does not match then word list remains the same
                    arrItems(number).Text = number.ToString & ")" & Space(1) & arrItems(number).Text
                End If
            Next

            lblSel = "" 'Reset Everything For New Matches
            tempstr = "" 'Reset Everything For New Matches
        End If
    Next
End If
```

MathswordsYoungAge:

```
ElseIf Menu_Screen.CkbxflagMaths = True Then 'Checks if the user has selected an Maths theme
    If Menu_Screen.CkbxflagYoungAge = True Then ''System will check if the user is of a Young age group
        For i As Integer = 0 To (arrmathswordsYoungAge.Length - 1) 'Loops through every element in the array
            If lblSel = arrmathswordsYoungAge(i) Then 'Determine Matches
                Dim count As Integer = 0 'Temporary counter
                Do 'Loops until the stack is empty
                    count += 1 'Increment count
                    Me.Controls("label" & mystack.Pop()).forecolor = Color.Red 'Change the colour of the labels in the stack to red and removes the top item
                Loop Until mystack.Count = 0

                lblTemp.ForeColor = Color.Red 'Change the colour of the clicked label to red
                intScore = Val(LblScore.Text) 'Takes the numeric value of the string in lblscore.text
                intScore = intScore + 20 'Add 20 to stack pointer
                LblScore.Text = intScore 'The text field of
                Beep() 'Outputs a beeps sound
                intItemsFound += 1 'Increment intitemsfound
                Gridlbls(True) 'Enables all the labels in grid

                'Output String - used for updating the word list in the pannel box
                arrmathswordsYoungAge(i) = arrmathswordsYoungAge(i) & Space(1) & "-" & Space(1) & "FOUND!"
                tempstr = arrmathswordsYoungAge(i)

                'Loop removes unessessary symbols in the string so only the word remains in the array.
                For intcount As Integer = 1 To 8 'Loop through arritems
                    Dim intwordlength As Integer 'Number of characters in array items with index intcount
                    intwordlength = Len(arrItems(intcount).Text) - 3 'Calcualates length of the word
                    arrItems(intcount).Text = Microsoft.VisualBasic.Right(arrItems(intcount).Text, intwordlength) 'Removes the number and ")" from the string. i.e 1) Boom
                Next

                For number As Integer = 1 To 8 'Loops through all the labels in array
                    If arrItems(number).Text = lblSel Then 'Determines if the word found matches the word in the array (words to find)
                        arrItems(number).Text = number.ToString & ")" & Space(1) & tempstr 'Updates the word list so the word has text found next to it.
                    ElseIf arrItems(number).Text <> lblSel Then 'If word does not match then word list remains the same
                        arrItems(number).Text = number.ToString & ")" & Space(1) & arrItems(number).Text
                    End If
                Next

                lblSel = "" 'Reset Everything For New Matches
                tempstr = "" 'Reset Everything For New Matches
            End If
        Next
```

MathswordsMiddleAge:

```
ElseIf Menu_Screen.CkbxflagMiddleAge = True Then 'System will check if the user is of the Middle age group
    For i As Integer = 0 To (arrmathswordsMiddleAge.Length - 1) 'Loops through every element in the array
        If lblSel = arrmathswordsMiddleAge(i) Then 'Determine Matches
            Dim count As Integer = 0 'Temporary counter
            Do 'Loops until the stack is empty
                count += 1 'Increment count
                Me.Controls("label" & mystack.Pop()).forecolor = Color.Red 'Change the colour of the labels in the stack to red and removes the top item
            Loop Until mystack.Count = 0

            lblTemp.ForeColor = Color.Red 'Change the colour of the clicked label to red
            intScore = Val(LblScore.Text) 'Takes the numeric value of the string in lblscore.text
            intScore = intScore + 20 'Add 20 to stack pointer
            LblScore.Text = intScore 'Updates the onscreen score
            Beep() 'Outputs a beeps sound
            intItemsFound += 1 'Increment intitemsfound
            Gridlbls(True) 'Enables all the labels in grid

            'Output String - used for updating the word list in the pannel box
            arrmathswordsMiddleAge(i) = arrmathswordsMiddleAge(i) & Space(1) & "-" & Space(1) & "FOUND!"
            tempstr = arrmathswordsMiddleAge(i)

            'Loop removes unessessary symbols in the string so only the word remains in the array.
            For intcount As Integer = 1 To 8 'Loop through arritems
                Dim intwordlength As Integer 'Number of characters in array items with index intcount
                intwordlength = Len(arrItems(intcount).Text) - 3 'Calcualates length of the word
                arrItems(intcount).Text = Microsoft.VisualBasic.Right(arrItems(intcount).Text, intwordlength) 'Removes the number and ")" from the string. i.e 1) Boom
            Next

            For number As Integer = 1 To 8 'Loops through all the labels in array
                If arrItems(number).Text = lblSel Then 'Determines if the word found matches the word in the array (words to find)
                    arrItems(number).Text = number.ToString & ")" & Space(1) & tempstr 'Updates the word list so the word has text found next to it.
                ElseIf arrItems(number).Text <> lblSel Then 'If word does not match then word list remains the same
                    arrItems(number).Text = number.ToString & ")" & Space(1) & arrItems(number).Text
                End If
            Next

            lblSel = "" 'Reset Everything For New Matches
            tempstr = "" 'Reset Everything For New Matches
        End If
    Next
```

MathswordsEldestAge:

```
ElseIf Menu_Screen.CkbxflagEldestAge = True Then 'System will check if the user is of the Eldest age group
    For i As Integer = 0 To (arrmathswordsEldestAge.Length - 1) 'Loops through every element in the array
        If lblSel = arrmathswordsEldestAge(i) Then 'Determine Matches
            Dim count As Integer = 0 'Temporary counter
            Do 'Loops until the stack is empty
                count += 1 'Increment count
                Me.Controls("label" & mystack.Pop()).forecolor = Color.Red 'Change the colour of the labels in the stack to red and removes the top item
            Loop Until mystack.Count = 0

            lblTemp.ForeColor = Color.Red 'Change the colour of the clicked label to red
            intScore = Val(LblScore.Text) 'Takes the numeric value of the string in lblscore.text
            intScore = intScore + 20 'Add 20 to stack pointer
            LblScore.Text = intScore 'Updates the onscreen score
            Beep() 'Outputs a beeps sound
            intItemsFound += 1 'Increment intitemsfound
            Gridlbls(True) 'Enables all the labels in grid

            'Output String - used for updating the word list in the pannel box
            arrmathswordsEldestAge(i) = arrmathswordsEldestAge(i) & Space(1) & "-" & Space(1) & "FOUND!"
            tempstr = arrmathswordsEldestAge(i)

            'Loop removes unessessary symbols in the string so only the word remains in the array.
            For intcount As Integer = 1 To 8 'Loop through arritems
                Dim intwordlength As Integer 'Number of characters in array items with index intcount
                intwordlength = Len(arrItems(intcount).Text) - 3 'Calcualates length of the word
                arrItems(intcount).Text = Microsoft.VisualBasic.Right(arrItems(intcount).Text, intwordlength) 'Removes the number and ")" from the string. i.e 1) Boom
            Next

            For number As Integer = 1 To 8 'Loops through all the labels in array
                If arrItems(number).Text = lblSel Then 'Determines if the word found matches the word in the array (words to find)
                    arrItems(number).Text = number.ToString & ")" & Space(1) & tempstr 'Updates the word list so the word has text found next to it.
                ElseIf arrItems(number).Text <> lblSel Then 'If word does not match then word list remains the same
                    arrItems(number).Text = number.ToString & ")" & Space(1) & arrItems(number).Text
                End If
            Next

            lblSel = "" 'Reset Everything For New Matches
            tempstr = "" 'Reset Everything For New Matches
        End If
    Next
End If
```

SciencewordsYoungAge:

```
ElseIf Menu_Screen.Ckbxflagscience = True Then 'Checks if the user has selected Science theme
    If Menu_Screen.CkbxflagYoungAge = True Then 'System will check if the user is of the Young age group
        For i As Integer = 0 To arrsciencewordsYoungAge.Length - 1 'Loops through every element in the array
            If lblSel = arrsciencewordsYoungAge(i) Then 'Determine Matches
                Dim count As Integer = 0 'Temporary counter
                Do 'Loops until the stack is empty
                    count += 1 'Increment count
                    Me.Controls("label" & mystack.Pop()).forecolor = Color.Red 'Change the colour of the labels in the stack to red and removes the top item
                Loop Until mystack.Count = 0

                lblTemp.ForeColor = Color.Red 'Change the colour of the clicked label to red
                intScore = Val(LblScore.Text) 'Takes the numeric value of the string in lblscore.text
                intScore = intScore + 20 'Add 20 to stack pointer
                LblScore.Text = intScore 'Updates the onscreen score
                Beep() 'Outputs a beeps sound
                intItemsFound += 1 'Increment intitemsfound
                Gridlbls(True) 'Enables all the labels in grid

                'Output String - used for updating the word list in the pannel box
                arrsciencewordsYoungAge(i) = arrsciencewordsYoungAge(i) & Space(1) & "-" & Space(1) & "FOUND!"
                tempstr = arrsciencewordsYoungAge(i)

                'Loop removes unessessary symbols in the string so only the word remains in the array.
                For intcount As Integer = 1 To 8 'Loop through arritems
                    Dim intwordlength As Integer 'Number of characters in array items with index intcount
                    intwordlength = Len(arrItems(intcount).Text) - 3 'Calcualates length of the word
                    arrItems(intcount).Text = Microsoft.VisualBasic.Right(arrItems(intcount).Text, intwordlength) 'Removes the number and ")" from the string. i.e 1) Boom
                Next

                For number As Integer = 1 To 8 'Loops through all the labels in array
                    If arrItems(number).Text = lblSel Then 'Determines if the word found matches the word in the array (words to find)
                        arrItems(number).Text = number.ToString & ")" & Space(1) & tempstr 'Updates the word list so the word has text found next to it.
                    ElseIf arrItems(number).Text <> lblSel Then 'If word does not match then word list remains the same
                        arrItems(number).Text = number.ToString & ")" & Space(1) & arrItems(number).Text
                    End If
                Next
                lblSel = "" 'Reset Everything For New Matches
                tempstr = "" 'Reset Everything For New Matches
            End If
        Next
```

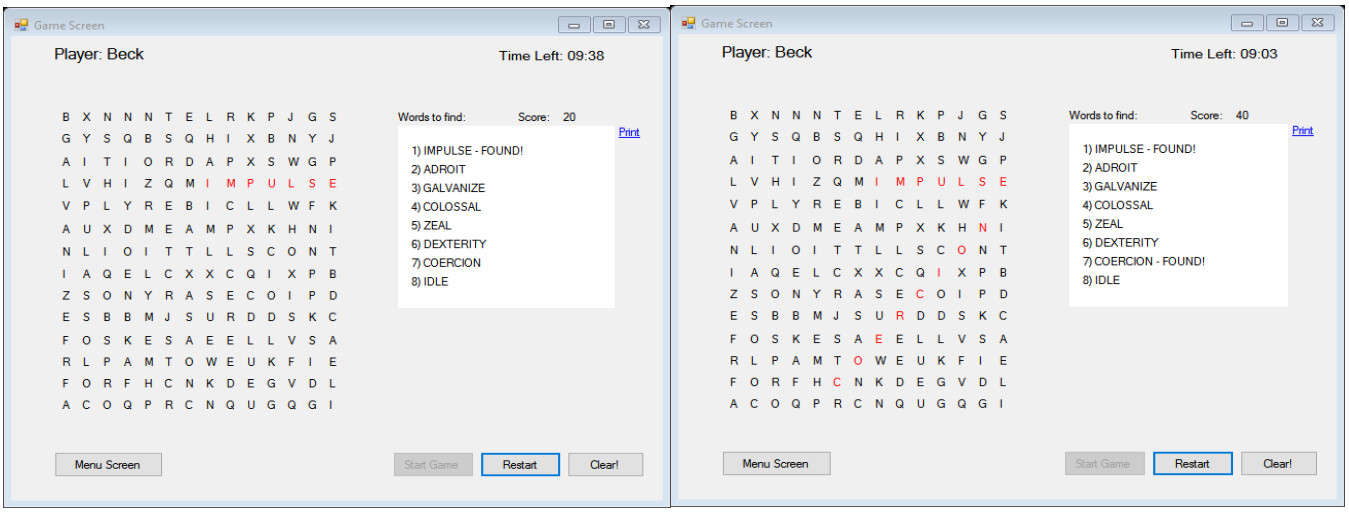
SciencewordsMiddleAge:

```
ElseIf Menu_Screen.CkbxflagMiddleAge = True Then 'System will check if the user is of the Middle age group
    For i As Integer = 0 To arrsciencewordsMiddleAge.Length - 1 'Loops through every element in array
        If lblSel = arrsciencewordsMiddleAge(i) Then 'Determine Matches
            Dim count As Integer = 0 'Temporary counter
            Do 'Loops until the stack is empty
                count += 1 'Increment count
                Me.Controls("label" & mystack.Pop()).forecolor = Color.Red 'Change the colour of the labels in the stack to red and removes the top item
            Loop Until mystack.Count = 0

            lblTemp.ForeColor = Color.Red 'Change the colour of the clicked label to red
            intScore = Val(LblScore.Text) 'Takes the numeric value of the string in lblscore.text
            intScore = intScore + 20 'Add 20 to stack pointer
            LblScore.Text = intScore 'Updates the onscreen score
            Beep() 'Outputs a beeps sound
            intItemsFound += 1 'Increment intitemsfound
            Gridlbls(True) 'Enables all the labels in grid

            'Output String - used for updating the word list in the pannel box
            arrsciencewordsMiddleAge(i) = arrsciencewordsMiddleAge(i) & Space(1) & "-" & Space(1) & "FOUND!"
            tempstr = arrsciencewordsMiddleAge(i)

            'Loop removes unessessary symbols in the string so only the word remains in the array.
            For intcount As Integer = 1 To 8 'Loop through arritems
                Dim intwordlength As Integer 'Number of characters in array items with index intcount
                intwordlength = Len(arrItems(intcount).Text) - 3 'Calcualates length of the word
                arrItems(intcount).Text = Microsoft.VisualBasic.Right(arrItems(intcount).Text, intwordlength) 'Removes the number and ")" from the string. i.e 1) Boom
            Next

            For number As Integer = 1 To 8 'Loops through all the labels in array
                If arrItems(number).Text = lblSel Then 'Determines if the word found matches the word in the array (words to find)
                    arrItems(number).Text = number.ToString & ")" & Space(1) & tempstr 'Updates the word list so the word has text found next to it.
                ElseIf arrItems(number).Text <> lblSel Then 'If word does not match then word list remains the same
                    arrItems(number).Text = number.ToString & ")" & Space(1) & arrItems(number).Text
                End If
            Next

            lblSel = "" 'Reset Everything For New Matches
            tempstr = "" 'Reset Everything For New Matches
        End If
    Next
```

SciencewordsEldestAge:

```
ElseIf Menu_Screen.CkbxflagEldestAge = True Then 'System will check if the user is of the Eldest age group
    For i As Integer = 0 To arrsciencewordsEldestAge.Length - 1 'Loops through every element in array
        If lblSel = arrsciencewordsEldestAge(i) Then 'Determine Matches
            Dim count As Integer = 0 'Temporary counter
            Do 'Loops until the stack is empty
                count += 1 'Increment count
                Me.Controls("label" & mystack.Pop()).forecolor = Color.Red 'Change the colour of the labels in the stack to red and removes the top item
            Loop Until mystack.Count = 0

            lblTemp.ForeColor = Color.Red 'Change the colour of the clicked label to red
            intScore = Val(LblScore.Text) 'Takes the numeric value of the string in lblscore.text
            intScore = intScore + 20 'Add 20 to stack pointer
            LblScore.Text = intScore 'Updates the onscreen score
            Beep() 'Outputs a beeps sound
            intItemsFound += 1 'Increment intitemsfound
            Gridlbls(True) 'Enables all the labels in grid

            'Output String - used for updating the word list in the pannel box
            arrsciencewordsEldestAge(i) = arrsciencewordsEldestAge(i) & Space(1) & "-" & Space(1) & "FOUND!"
            tempstr = arrsciencewordsEldestAge(i)

            'Loop removes unessessary symbols in the string so only the word remains in the array.
            For intcount As Integer = 1 To 8 'Loop through arritems
                Dim intwordlength As Integer 'Number of characters in array items with index intcount
                intwordlength = Len(arrItems(intcount).Text) - 3 'Calcualates length of the word
                arrItems(intcount).Text = Microsoft.VisualBasic.Right(arrItems(intcount).Text, intwordlength) 'Removes the number and ")" from the string. i.e 1) Boom
            Next

            For number As Integer = 1 To 8 'Loops through all the labels in array
                If arrItems(number).Text = lblSel Then 'Determines if the word found matches the word in the array (words to find)
                    arrItems(number).Text = number.ToString & ")" & Space(1) & tempstr 'Updates the word list so the word has text found next to it.
                ElseIf arrItems(number).Text <> lblSel Then 'If word does not match then word list remains the same
                    arrItems(number).Text = number.ToString & ")" & Space(1) & arrItems(number).Text
                End If
            Next

            lblSel = "" 'Reset Everything For New Matches
            tempstr = "" 'Reset Everything For New Matches
        End If
    Next
End If
```

Testing found word algorithms:

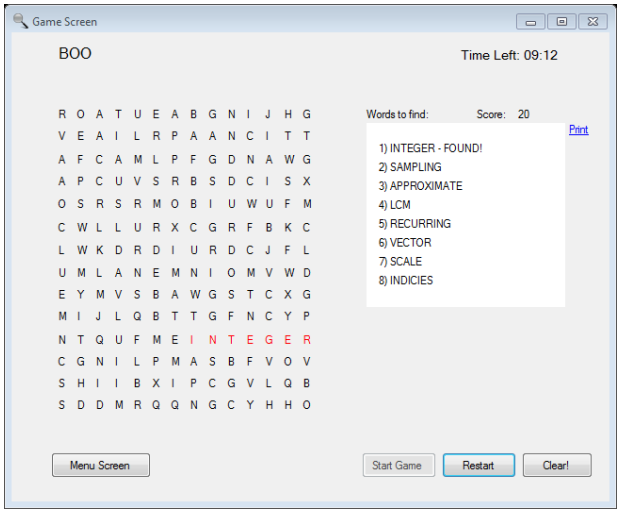


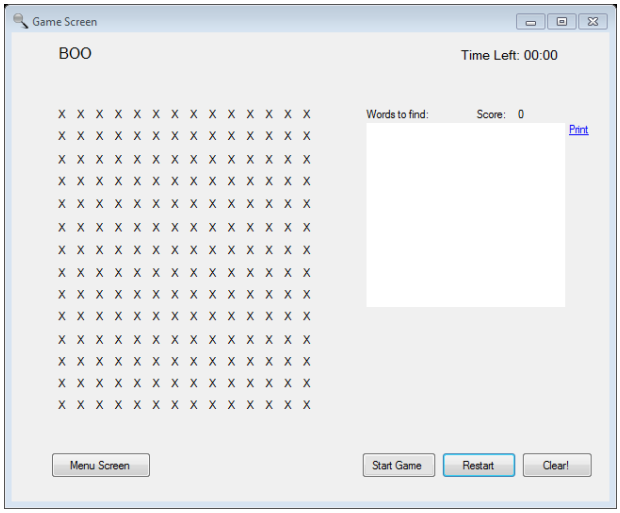
In the image above I try finding a word. 20 points is added to the score when a word is found each character in the found word changes to red.

Subroutine for the Restart button:

```
Private Sub CmdRestart_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles CmdRestart.Click
    cmdStartGame.Enabled = True 'Enables the Start Game button
    tmrCountDown.Stop() 'Stops the game timer
    LblGameTime.Text = "Time Left: 00:00" 'Changes the text of the label
    lblSel = "" 'initialises lbl select - contians no characters(Empty)
    LblScore.Text = 0 'LblScore is set to 0
    intItemsFound = 0 'Items found is initialised to 0
    intTimeBonus = 0 'Time Bounus is initialised to 0
    pnlitems.Controls.Clear() 'Words in the panel box are clear

    Dim number As Integer 'Counts the label number
    For row = 0 To 13 Step 1 'Loops through each row
        For column = 0 To 13 Step 1 'Loops through each column
            number += 1 'Increment number
            RemoveHandler (Me.Controls("label" & number).MouseClick), (AddressOf lbl_MouseClick) 'Removes event handler MouseClick
            Me.Controls("label" & number).ForeColor = Color.Black 'Changes the colour of each label to black
            Me.Controls("label" & number).Text = "X" 'Changes the text of each label to "X"
            GridArray(row, column) = "X"  'Initialises array to "X"
            arrunscramble(row, column) = "X" 'initialises array "X"
            Me.Controls("label" & number).Enabled = True 'All labels in grid are enabled
        Next
    Next
End Sub
```

Immediately after writing up the subroutine, I decided to test it by debugging the program. The results of testing are shown below.

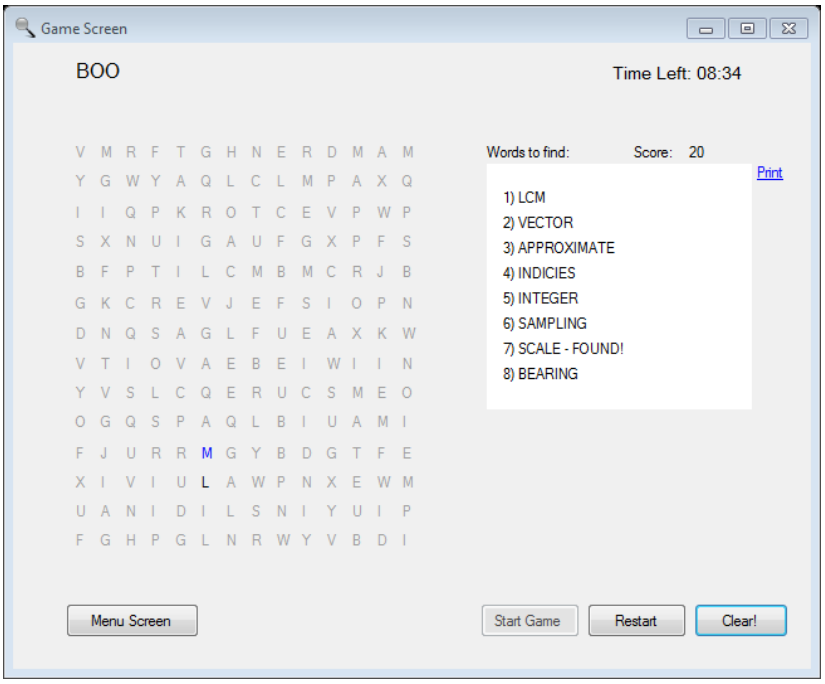




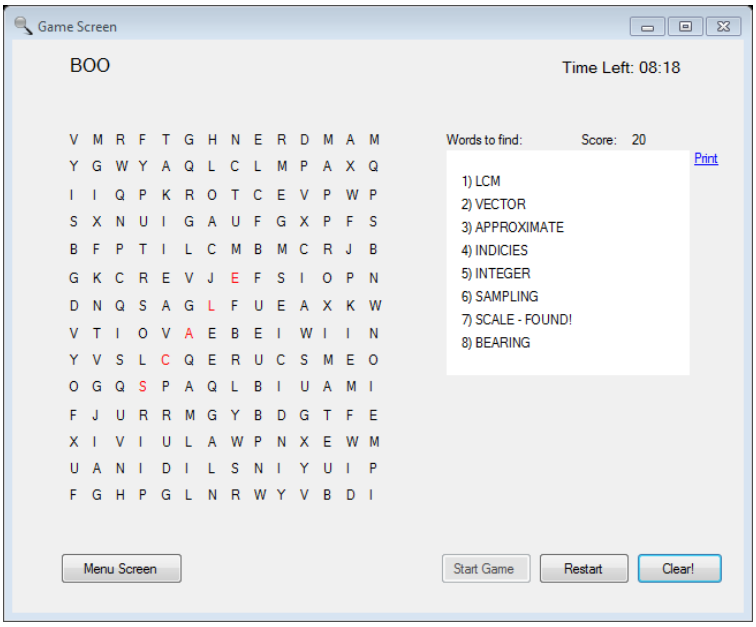
The screenshots above demonstrate the success of the restart button. All the controls are reset to their original state/value.

Subroutine for the clear button:

```
Private Sub CmdClear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles CmdClear.Click
    lblSel = "" 'Initialises lblsel
    Gridlbls(True) 'Enables all labels in grid

    While mystack.Count <> 0 'Loop is executed if there is content in the stack
        If Me.Controls("label" & mystack.Peek()).forecolor = Color.Blue Then 'Check if colour of top item in stack is blue
            Me.Controls("label" & mystack.Pop()).forecolor = Color.Black 'Remove item from stack and changing the colour of the label to black
        ElseIf Me.Controls("label" & mystack.Peek()).forecolor = Color.Red Then 'Check if colour of the top item in stack is red
            Me.Controls("label" & mystack.Pop()).forecolor = Color.Red 'Remove item from stack and keep the colour of the label to red
        ElseIf Me.Controls("label" & mystack.Peek()).forecolor = Color.Black Then 'Check if the colour of the top item in stack is black
            Me.Controls("label" & mystack.Pop()).forecolor = Color.Black 'Remove the label from stack and ensure colour of label is black
        End If
    End While
End Sub
```



(User testing to find a word)



(Clear button clearing the words, except the found word)

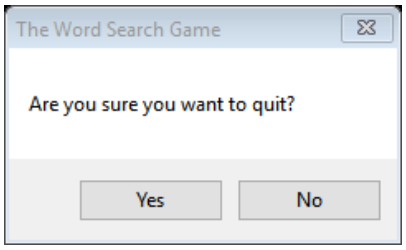
Module 3:

This module is contains the subroutines for ending the game.

Firstly, I began to code the quit algorithm, which executes when the user clicks on the quit button (cmdquit) on the menu screen. I will use the variable intmsgbx to store the integer returned from the message box function, which will help to determine whether the user clicked “yes” or “no”.

```
Private Sub CmdQuit_Click(sender As System.Object, e As System.EventArgs) Handles CmdQuit.Click 'Validates if the user actually wishes to quit
    Dim intmsgbx As Integer 'Stores the integer indicating whether the yes button or no button has been clicked
    intmsgbx = MsgBox("Are you sure you want to quit?", MsgBoxStyle.YesNo) 'Message displayed to the user when button is clicked
    If intmsgbx = 6 Then 'The number 6 represents if the user clicked on the yes button in the message box
        Close() 'Closes the form
    End If
End Sub
```

Upon clicking the quit button I was prompted with the following message box, to confirm that I wish to exit the game:



When I click on “Yes” the form exits and when I click on “No” the message box closes.

The next stage was to write the subroutine for calculating and displaying the user final score. This subroutine is shown below.

Subroutine Show Summary:

My users’ required 21 bonus points for each minute that remains on the timer and 6 points for every second remaining. The if statement will detect if the user has found all 8 words, in which case the timer is stopped immediately.

```
Private Sub ShowSummary() 'Shows Summary After Game Completion

    Dim strSummary As String 'Summary String (displayed to the user)

    If timeDiff.Minutes > 0 Then 'Calculate Bonus For Minutes Left
        intTimeBonus = timeDiff.Minutes * 21
    End If

    If timeDiff.Seconds > 0 Then 'Calculates Bonus For Seconds Left
        intTimeBonus += timeDiff.Seconds * 6
    End If

    intScore = (intScore + intTimeBonus) 'Add Calculated Bonus To Current Score

    If intItemsFound = 8 Then 'If All Words Are Found
        tmrCountDown.Stop() 'Stop the timer
        MessageBox.Show("You Win!!") 'DIsplay messagebox
        Showhiddenwords() 'Reveal the hidden words
        Dim intnum As Integer 'Counts the current label to be accessed
        For row = 0 To 13 Step 1 'Loops through each Row
            For column = 0 To 13 Step 1 'Loops through each column
                intnum += 1 'Increment intnum
                RemoveHandler (Me.Controls("label" & intnum).MouseClick), (AddressOf lbl_MouseClick) 'Remove Event Handler Click
            Next
        Next
    End If

    'Compose String For Output
    strSummary = "Summary" & Environment.NewLine & Environment.NewLine & _
        "TIME BONUS: " & intTimeBonus.ToString & Environment.NewLine & Environment.NewLine & _
        "SCORE: " & intScore
    LblScoreSummary.Visible = True 'Make the label visible to the user
    LblScoreSummary.Text = strSummary 'Display The Summary On Screen
End Sub
```

Subroutine Show Hidden Words:

As part of the design specification, client and users would like to see where the words have been hidden in the grid. This subroutine uses the algorithm show hidden words to display where the words were hidden in the grid. After writing the subroutine for displaying the user's final score summary, I then wrote the code for displaying where all the words had been hidden in the grid, as seen below:

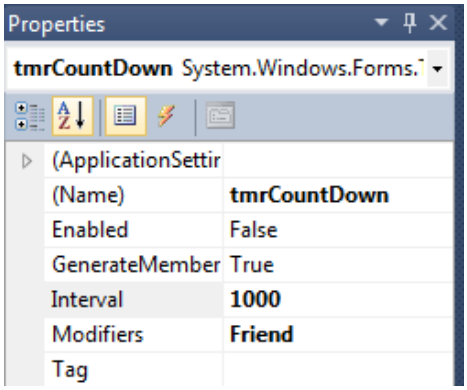
```
Private Sub Showhiddenwords() 'Reveals the hidden words when game ends
    For row = 0 To 13
        For column = 0 To 13
            'The String Char Represents A Random Character When Game First Loads
            If arrunscramble(row, column) = "Char" Then
                arrunscramble(row, column) = " "
            End If
        Next
    Next

    Dim counter As Integer 'Counts the current label in grid

    For row = 0 To 13
        For column = 0 To 13
            counter = counter + 1
            If arrunscramble(row, column) = " " Then 'Check if grid contains a random character or a character of the hidden word
                Me.Controls("label" & counter).Text = arrunscramble(row, column) 'Replace random characters with a space
            End If
        Next
    Next
End Sub
```

The final part to code the subroutine for detecting and stopping the timer when it reaches zero.

Subroutine TmrCountdown_tick

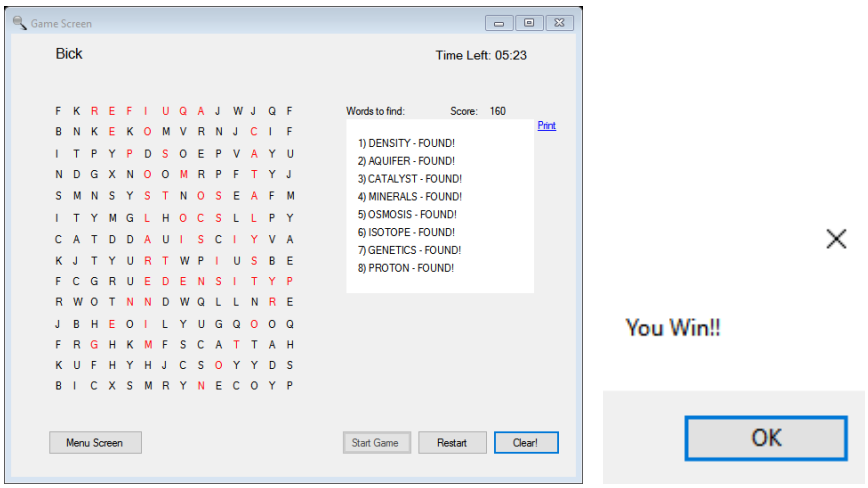
This subroutine contains the statements that are to be executed everytime time the value of tmrCountDown changes. This subroutine will format the time to be displayed on-screen into a single string and check if the time has reached zero.

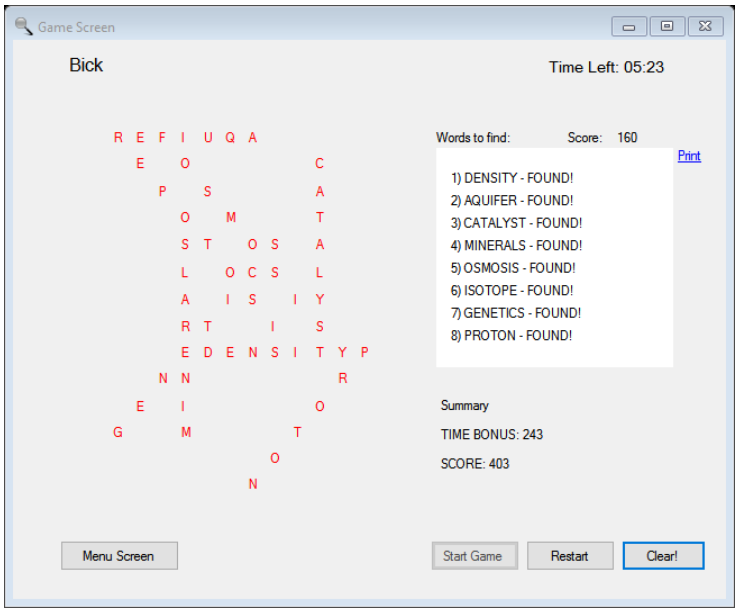
```
Private Sub tmrCountDown_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tmrCountDown.Tick 'Subroutine executes everytime the timer ticks (Changes value)
    Dim strFinalOutput As String 'Output String of the time remaining

    timeDiff = timeEnd - DateTime.Now 'Caclulate Difference Between Start and End times
    Dim output As TimeSpan = New TimeSpan(timeDiff.Hours, timeDiff.Minutes, timeDiff.Seconds) 'Visual Studio tool which represents a time interval

    strFinalOutput = "Time Left: " & output.ToString().Substring(3) 'Concatenates Time

    LblGameTime.Text = strFinalOutput 'Lbl is set to the remaining time left

    If (timeDiff.Ticks < 0) Then 'If value of time is Up
        tmrCountDown.Stop() 'Stops the timer
        MessageBox.Show("Time's up!!") 'Output a messagebox
        ShowSummary() 'Call show summary subroutine
        Showhiddenwords() 'Call Showhiddenwords summary
        Dim intnum As Integer 'Act as a counter, so every label in the grid can be accesed
        For row = 0 To 13 Step 1
            For column = 0 To 13 Step 1
                intnum += 1 'Increment intnum
                RemoveHandler (Me.Controls("label" & intnum).MouseClick), (AddressOf lbl_MouseClick) 'Removes Event Handler Click on labels in grid
            Next
        Next
    End If
End Sub
```



The properties of game timer, the interval is set to 1000 milliseconds.

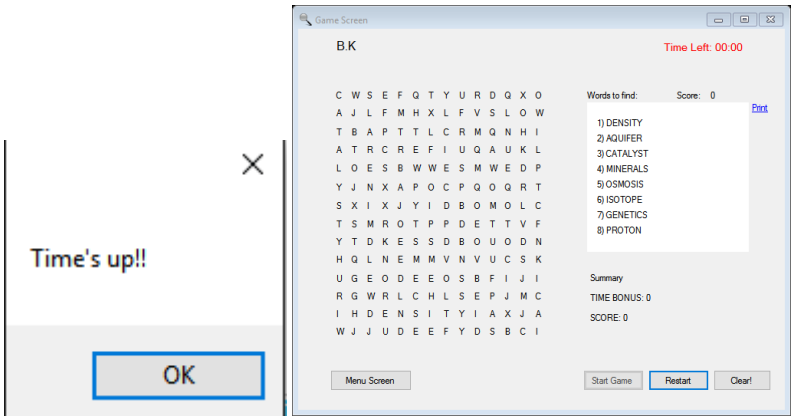
Now that all the tasks in this module have been completed, I shall test if they work by performing alpha testing on the game. I will test if the game can detect whether I have won or lost the game (by running out of time).





(Player wins the game)

When I found all the words the timer stopped ticking and a message box appeared, “You Win!!” Upon clicking on the “OK” button, the score summary section was presented and the hidden words were reveals by replacing all the scrambled letters with a blank space. I also needed to test if the game could detect if I lost the game, in which case I should be presented with another message box.



(Player runs out of time)

Module 4:

Module 4 contains the two simple statements, which close the form and open another form.

I shall now add the code for returning to the menu screen and progressing onto the game screen.

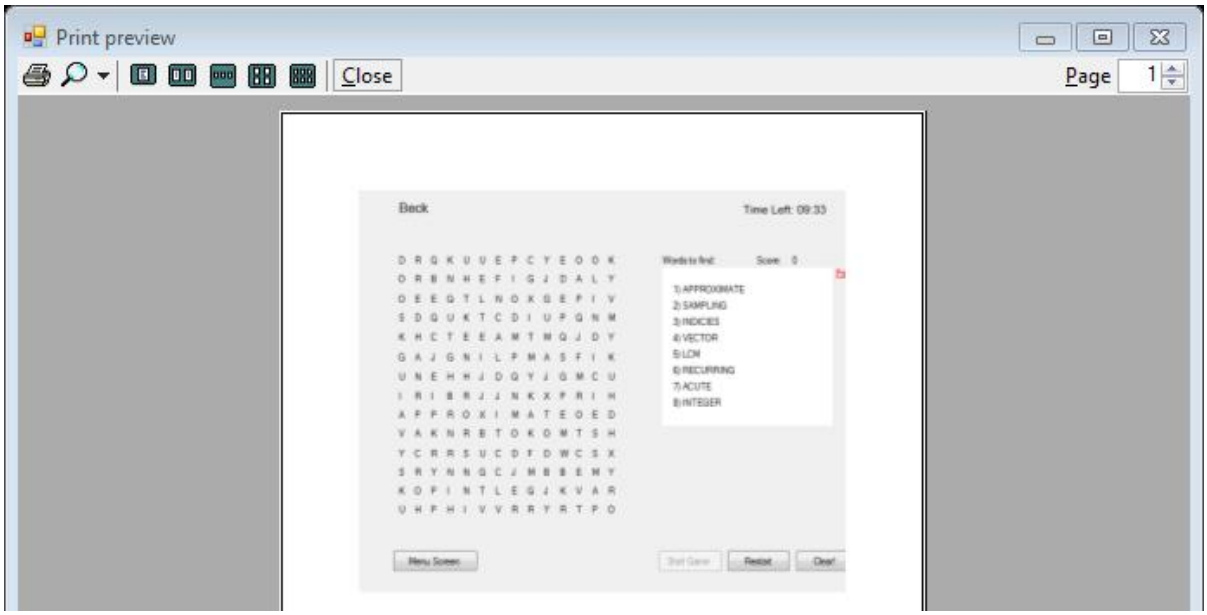
```
Public Class Frm_Instructions

    Private Sub cmdMenuScreen_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdMenuScreen.Click
        'Executes when the Menu Screen button has been clicked
        Frm_Menu_Screen.Show() 'Display the Menu Screen
        Me.Close() 'Closes the current screen
    End Sub

    Private Sub CmdPlayGame_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles CmdPlayGame.Click
        'Executes when the Play button has been clicked
        Frm_GameScreen.Show() 'Display the Game Screen
        Me.Close() 'Closes the current screeen
    End Sub
```

Module 5:

This is the final section of the software development phase. I will make use of the visual studio tool "PrintToPreview", which should display open a new window, in which the user is able to print the game screen.

```
Private Sub LinkLabelPrint_LinkClicked(ByVal sender As System.Object, ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles LinklblPrint.LinkClicked
    'Opens a new window, which displays the print dialogue
    PrintForm1.PrintAction = Printing.PrintAction.PrintToPreview
    PrintForm1.Print()
End Sub
```

After writing the code for the print button, I decided to test it out by clicking on the print button.



Software Testing

It is essential The Word Search Game is thoroughly tested, so that I am confident that the solution works as intended and that there will be no bugs or errors when presenting my software to my user.

Testing Valid, Invalid and Extreme data input

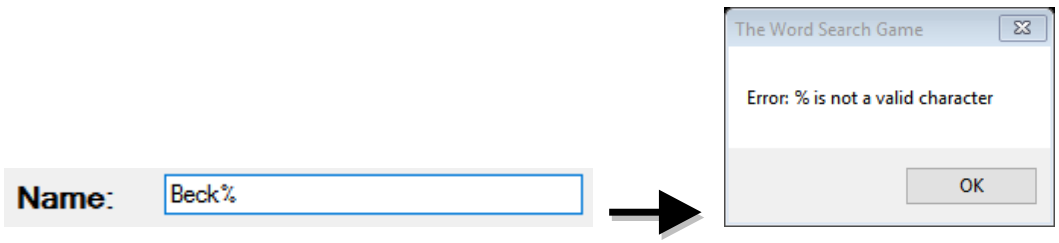
To test the following inputs I added a message box, so that I can test if the condition has pass or failed. Name text box should only allow letters

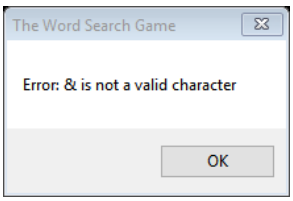
Test Number:	Input	Expected Results	Pass/Fail
1	Letters	Valid – Data accepted	Pass
2	A symbol – except for hyphen	Invalid – Error message	Pass
3	numbers	Invalid – Error message	Fail
4	Nothing	Invalid - Error message	Pass
5	Letters and Numbers	Invalid – Error message	Fail
6	Letters and Symbols	Valid if redundant characters are replaced with a blank	Pass
7	Number and Symbols	Invalid – Error message	Fail
8	Letters, numbers and Symbols	Invalid – Error message	Fail

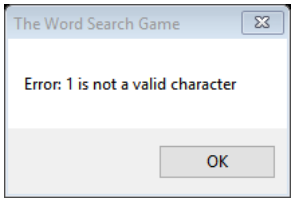
I realised that the user could enter alphanumeric characters in the text box and symbols, so I have now updated the code so that the change have been made and tested. Below are is the updated code and images of testing the algorithm:

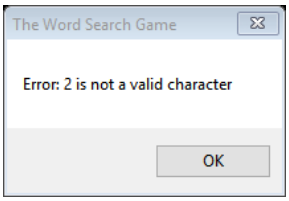
```
If Len(TextBoxName.Text) = 0 Then
    MsgBox("Please enter your name", MsgBoxStyle.Critical + MsgBoxStyle.OkCancel, "Error")
    tempFlag = True
ElseIf Len(TextBoxName.Text) > 0 Then 'Checks if the user has enetered something in the textbox
    Dim regex As Regex = New Regex("[^a-zA-Z-áéíóúÁÉÍÓÚ]") 'valid characters
    Dim match As Match = regex.Match(TextBoxName.Text) 'textboxname will be searched for the invalid characters
    Dim symbol As Char 'Stores the invalid symbol

    If match.Success Then 'checks if string contains invalid characters
        symbol = match.Value 'Return the invalid character
        Dim tempstr As String = "Error:" & Space(1) & symbol & Space(1) & "is not a valid character"
        MsgBox(tempstr) 'Outputs a messagebox
        Exit Sub 'exits the subroutine
    End If
    users_name = TextBoxName.Text 'users name will be the lbl containing the palayers name on the game screen
End If
```

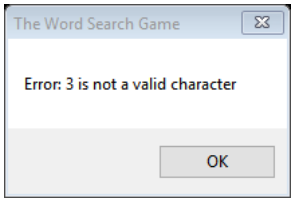
The above code makes use of the Visual Studio tool regex and match. This regex is an abbreviation for regular expression, by which a sequence of symbols and characters expressing a string or pattern to be searched for within a longer piece of text. "^" this symbol means the following characters are valid, everything else is invalid.





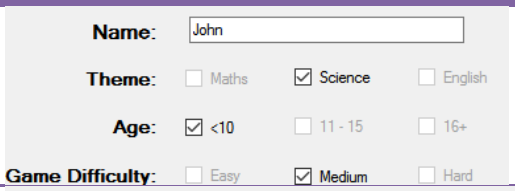
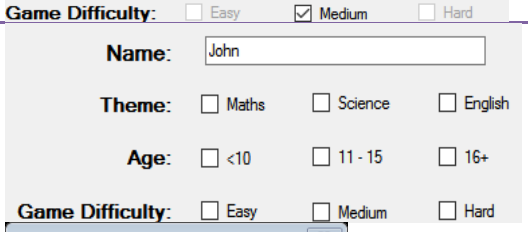
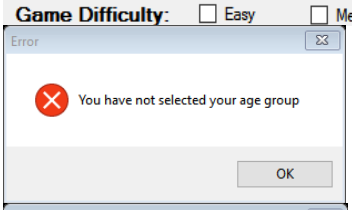
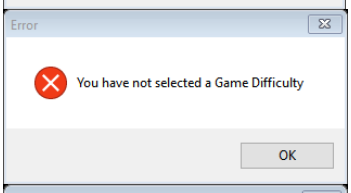
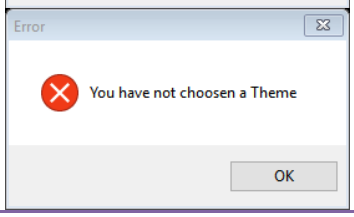


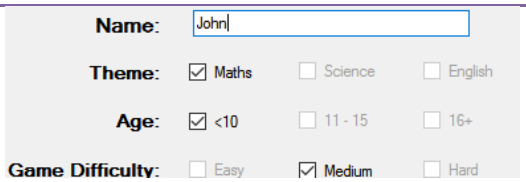
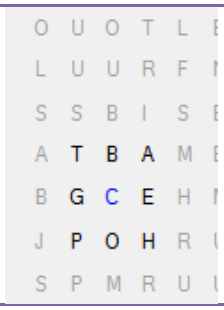
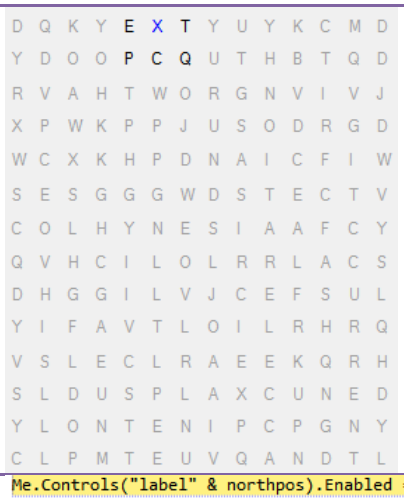




Below is the updated table for the testing of Valid, Invalid and Extreme data input

Test Number:	Input	Expected Results	Pass/Fail
1	Letters	Valid – Data accepted	Pass
2	A symbol – except for hyphen	Invalid – Error message	Pass
3	numbers	Invalid – Error message	Pass
4	Nothing	Invalid - Error message	Pass
5	Letters and Numbers	Invalid – Error message	Pass
6	Letters and Symbols	Valid if redundant characters are replaced with a blank	Pass
7	Number and Symbols	Invalid – Error message	Pass
8	Letters, numbers and Symbols	Invalid – Error message	Pass

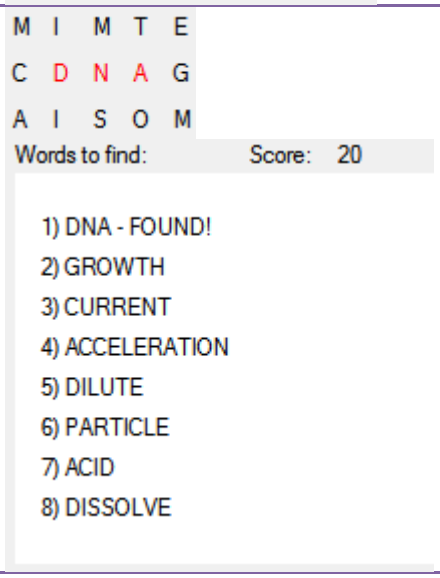
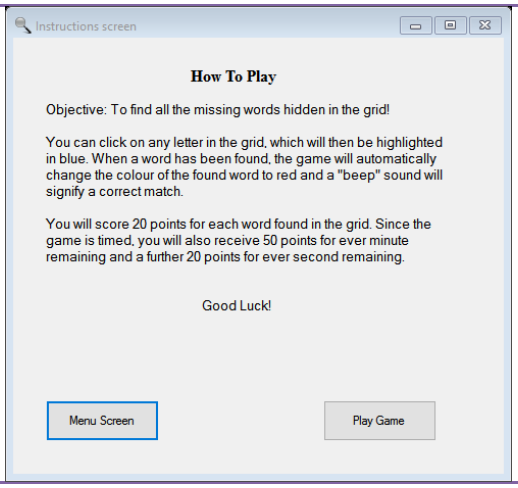
Function being tested	Test Number	Input	Expected Results	Evidence	Pass/Fail
Checkboxes on Menu Screen	1	Click one checkbox	Valid – only one is accepted per row		Pass
	2	No checkboxes	Invalid – Error message	   	Pass

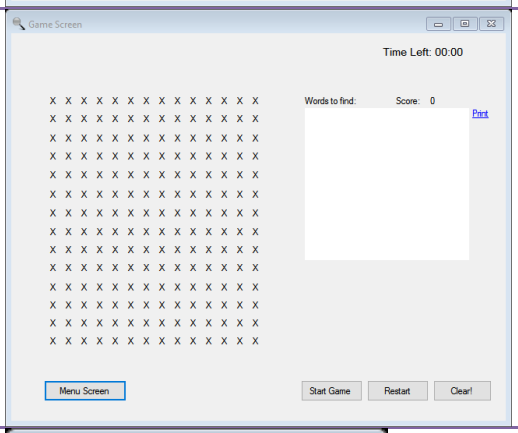
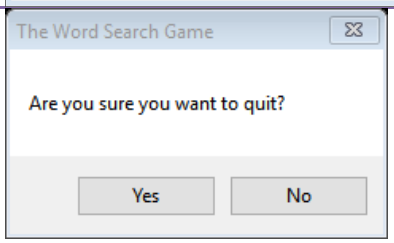
	3	Clicks on a checkbox then uncheck it and select another checkbox	Valid – Data accepted		Pass – John has change his theme to Maths (see test number 1)
Mouse click on letters in grid	1	User clicks on any letter in grid	Valid – Colour of the letter turn blue		Pass
	2	User clicks on letter not in grid	Invalid – nothing happens	See acceptance testing results	Pass
	3	User clicks on a letter in the top row of the grid	Valid – data accepted		Pass

	4	User clicks on the top right letter in the grid	Extreme – data accepted	`Me.Controls("label" & northpos).Enabled = True`
				`Me.Controls("label" & eastpos).Enabled = True`
				`Me.Controls("label" & westpos).Enabled = True`
				`Me.Controls("label" & southpos).Enabled = True`
				`Me.Controls("label" & southwestpos).Enabled = True`
				`Me.Controls("label" & northwestpos).Enabled = True`
				`Me.Controls("label" & northeastpos).Enabled = True`
				`Me.Controls("label" & southeastpos).Enabled = True`
					Fail

	5	User clicks on the bottom left letter in the grid	Extreme – data accepted	`Me.Controls("label" & northpos).Enabled = True`
				`Me.Controls("label" & eastpos).Enabled = True`
				`Me.Controls("label" & westpos).Enabled = True`
				`Me.Controls("label" & southpos).Enabled = True`
				`Me.Controls("label" & southwestpos).Enabled = True`
				`Me.Controls("label" & northwestpos).Enabled = True`
				`Me.Controls("label" & northeastpos).Enabled = True`
				`Me.Controls("label" & southeastpos).Enabled = True`
					Fail

	6	User clicks on the bottom right letter	Extreme – data accepted		Pass
	7	User clicks on a letter on the edge of the grid	Extreme - data accepted		Pass
	8	User clicks on a letter in the bottom row of the grid	Valid – data accepted		Pass

	9	User clicks on the top left letter	Valid – data accepted		Pass
	10	User clicks on each letter of the missing word, in turn.	Valid – each letter of the word should change red.		Pass
	11	User double clicks on a letter	Extreme – data accepted	See beta testing results on page 99.	Pass
Displaying instructions	1	Clicking the instructions button	Instructions should be displayed		Pass – The evidence show the instruction screen when the instructions button is pressed.

Returning from the instruction screen to menu screen	1	Clicking on menu button	Menu screen should be displayed		Pass
Progressing to the game screen from the instruction screen	1	Clicking on the play button	Game screen should be displayed		Pass
Leaving the game	1	Click on the quit button	Validation message “Are you sure you want to leave?”		Pass
	2	Click on yes	Game ends	See beta testing results on page 99	Pass
	3	Click on no	Message box closes	See beta testing results on page 99	Pass

In the above test, the function mouse click test numbers 4 and 5 failed, which resulted in a run-time error. After recognising this error I had realised that the code needed to be modified so that it checks if the letter, which is to enable actually exists in the grid. The new code is stated below:

```
If northeastpos >= 1 And northwestpos >= 1 And northpos >= 1 And southeastpos <= 196
And southwestpos <= 196 And southpos <= 196 Then
```

Note: The number of the smallest label is one and the number of the largest label is 196.

Beta Testing

I will now present the results from my beta testing. The results for yes and no are shown in parenthesis and any comments the user added.

Question number	Question	Answer (Yes/No)
1	Where you able to start the game?	Yes (5) No(2) - They didn't see the "Start" button
2	Do all the buttons and checkboxes work?	Yes(7) No(0)
3	Are you able to print out the game?	Yes (5) No (2) – Did not try to print the game
4	Are you happy with the colour used?	Yes (7) – One user said some stranger colours would be nice No (0)
5	Is there sufficient help available?	Yes (6) No (1) – This user said to have a tutorial mode
6	Are the messages helpful and clear?	Yes (7) No (0)
7	Is it easy to find a word?	Yes (4) No (3) – One user said they are bad at word searches.
8	Is the game too difficult?	Yes (2) – Don't know what words to find. Another user said this occurred for the hard mode. No (5) – Not too easy or too difficult.
9	Is the game too easy?	Yes (7) No (0)
10	Is there enough time to play the game?	Yes (3) – Timer makes the game more exciting Maybe (1) – Depends on the level of difficulty No (3) – One user said there isn't enough time for the hard mode.
11	Were you able to check and then uncheck a checkbox on the menu screen?	Yes (7) No (0)
12	Is it easy to navigate around the game?	Yes (7) No (0)
13	Could you exit the game?	Yes (7) No (0)
14	When you click the "Start" button, does the game randomly generate a new list of words?	Yes (7) No (0)
15	Are you happy with the sound effects used, for example when a word is found?	Yes (7) – Good No (0)
16	If you have any other questions that have not been covered, please add your views to the comments section on the right.	None.

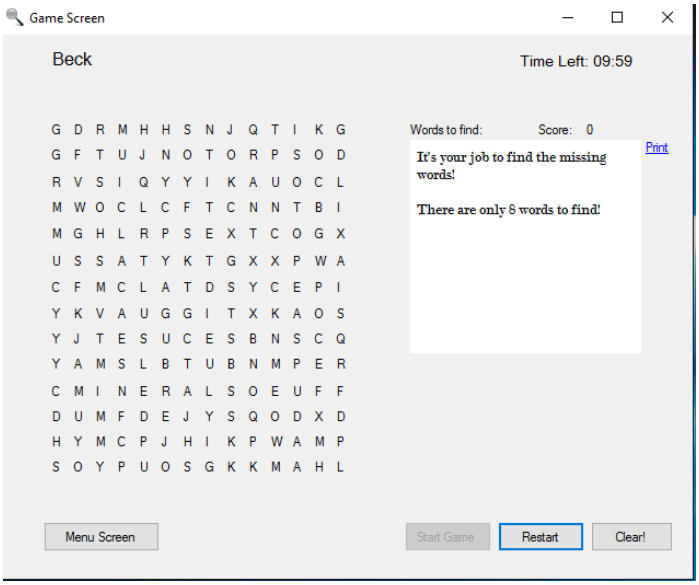
Analysis of beta testing results

Most of my users tested the game and felt positive that everything works. However, some users gave negative comments about some features. As a result of the negative comments I decided to make some changes to The Word Search Game:

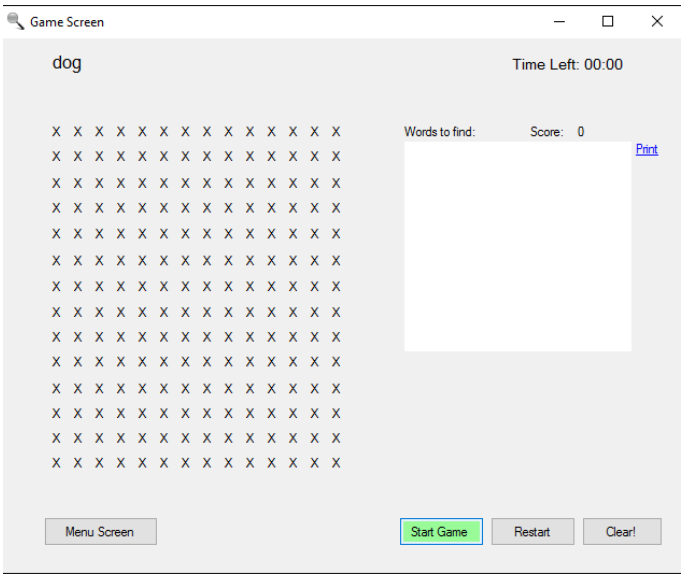
- For the hard mode the user/player will be given 10 minutes to find all the hidden words. This replaces the current 8 minutes the given
- I will change the colour of the “Start” button to light green, so the users can clear locate the start button. This will be done by change the colour property for the start button

Below is the code for the updated time for the hard mode, what follows is the tested code.

```
ElseIf Frm_Menu_Screen.checkflaghard = True Then
    Dim minute As Double = System.Convert.ToDouble(10) '10 Minutes
    Dim second As Double = System.Convert.ToDouble(0) '0 Seconds

    timeEnd = timeEnd.AddMinutes(minute) 'Add Mintues
    timeEnd = timeEnd.AddSeconds(second) 'Add Seconds
```



I shall now change the colour of the start button to light green.



Acceptance testing (client)

The tests that I gave to my client to test the game are split into modules. I will get my client to test the functions listed and report back if they are satisfied with the test. The feedback will be listed in the table below

Module 1:

a) A click on the play button should take you to the game form, once all the required fields are completed.
b) The user can only enter letters and use the hyphen symbol in the textbox, symbols and digits are not permitted.
c) All 8 words appear once in the grid.
d) Each word is placed in the following directions once: North, East, South, West, Northeast, Northwest, Southeast and Southwest.

Module 2:

a) Click on 'Start game' to begin game
b) Use a mouse click to click on a letter in the grid
c) When the first letter is clicked (in the grid), all other letters should be disabled except for the 8 letters that surround the clicked letter.
d) After clicking on the next letter, you find you can only travel in one direction.
e) When a letter is clicked in the grid, its colour should change to blue.
f) If a word is found, every letter in the found word should change to red and remain red.
g) No word can be found more than once (this can be checked by looking at the player's score).
h) Clicking the clear button should enable all the letters in the grid and change the colour of the selected letters to black.
i) Double-clicking on a letter twice should not allow you to gain points for finding a word. E.g. CCAT instead of CAT.
j) User should be awarded 20 points for every word found
k) The timer decrements for every second in real time.

Module 3:

a) When the timer reaches zero, the game ends immediately and a message box is displayed showing "Time's up".
b) Click 'quit' to exit game
    i. Yes to confirm the exit and the window will close
    ii. No will close the message box.
c) All the words, including the found words, should be revealed at the end of the game
d) The users final score summary will be shown immediately when the game ends, this summary will also show any bonus points that have been awarded to the user.

Module 4:

a) Display the instructions, upon clicking the instructions button and the current form closes.
b) Progress onto the Game Screen, upon clicking the "Play" button and the current form closes.
c) Return to the menu screen upon clicking the "Menu Screen" button and the current form closes.

Module 5:

a) Display print dialogue
b) Print the content in the print preview

Module Number	Test (letter)	Success?	Comments
1	a	Yes	None
	b	Yes	Guide the user
	c	Yes	None
	d	Yes	None
2	a	Yes	None
	b	Yes	None
	c	Yes	Helps find the words faster
	d	Yes	None
	e	Yes	None
	f	Yes	None
	g	Yes	None
	h	Yes	None
	i	Yes	Good control of the game – No cheating is allowed
	j	Yes	None
	k	Yes	None
3	a	Yes	None
	b	Yes	None
	i	Yes	None
	ii	Yes	None
	c	Yes	None
	d	Yes	None
4	a	Yes	None
	b	Yes	None
	c	Yes	None
5	a	Yes	None
	b	Yes	None

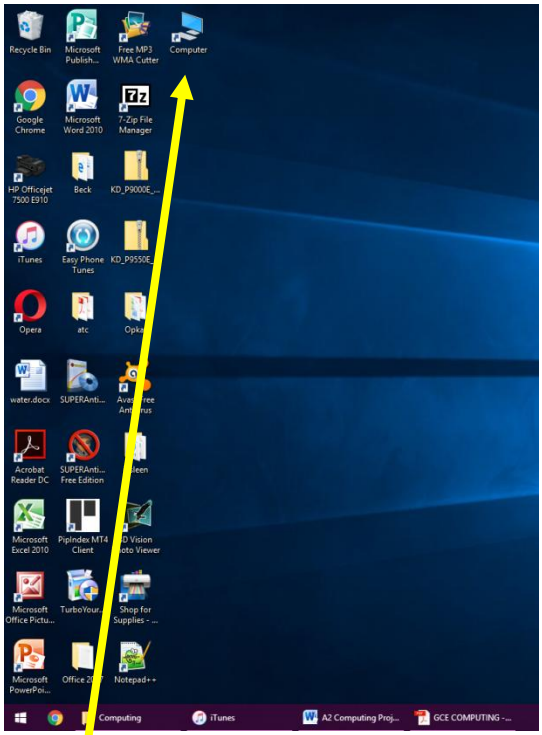
Signature: …… Date: ……

Overall, I fell that my client and users are overall happy with The Word Search Game. There is no need to further improve the game at this stage.

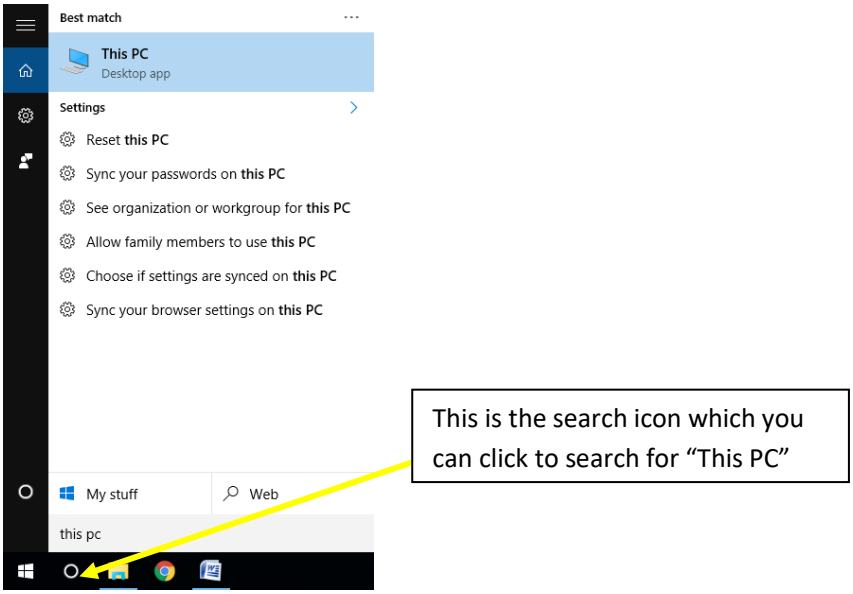
Contents page

Installation

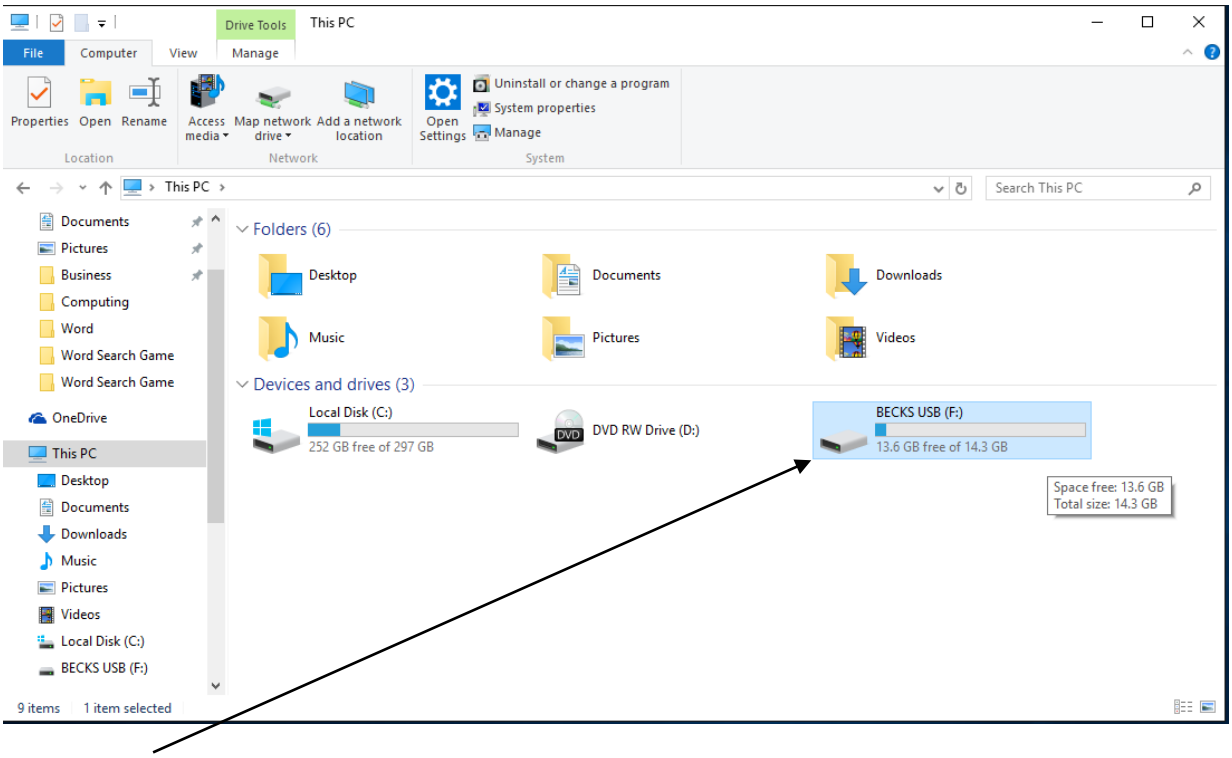
1. Insert the USB into the computer's USB port.
2. Either double click on "My Computer" on the computer's desktop (see below), or type in "My computer" in the search bar located in the start menu. Note: if you are using the window 10 operating system search for "This PC" instead of "My Computer".



This is the icon you want to search for, which should have the text "My computer".

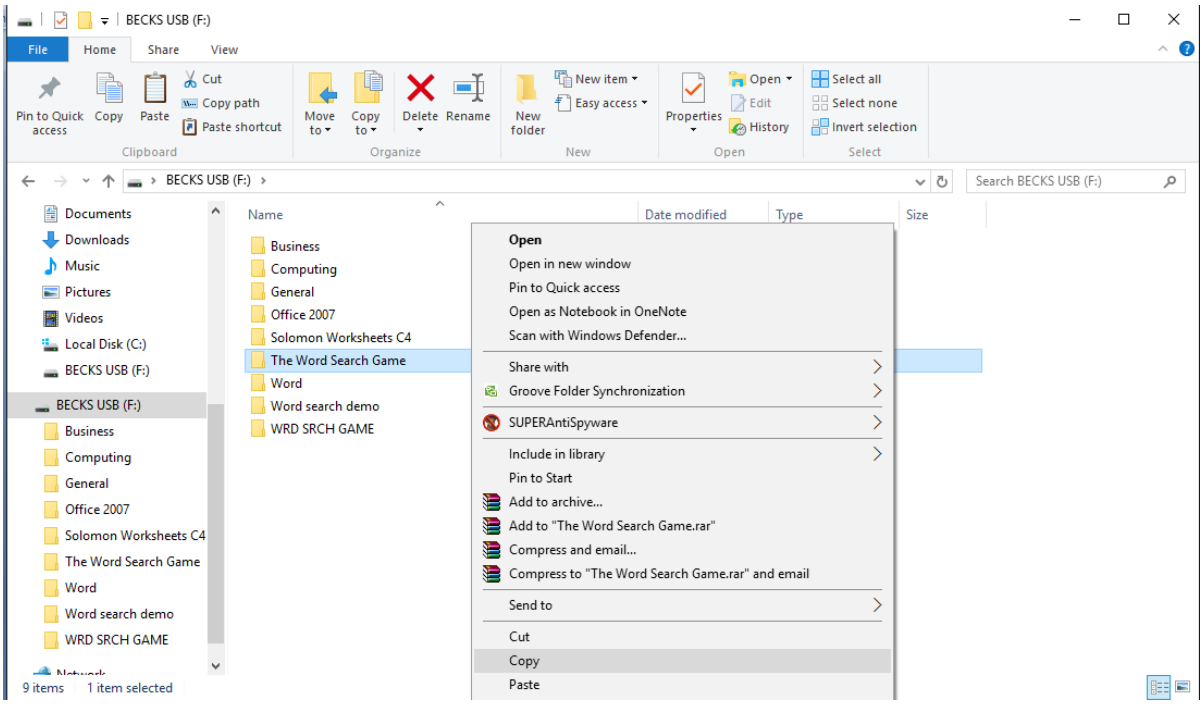


When you have double-clicked on “This PC” or “My computer”, a window will appear. You need to double-click on the USB icon where the game is stored.



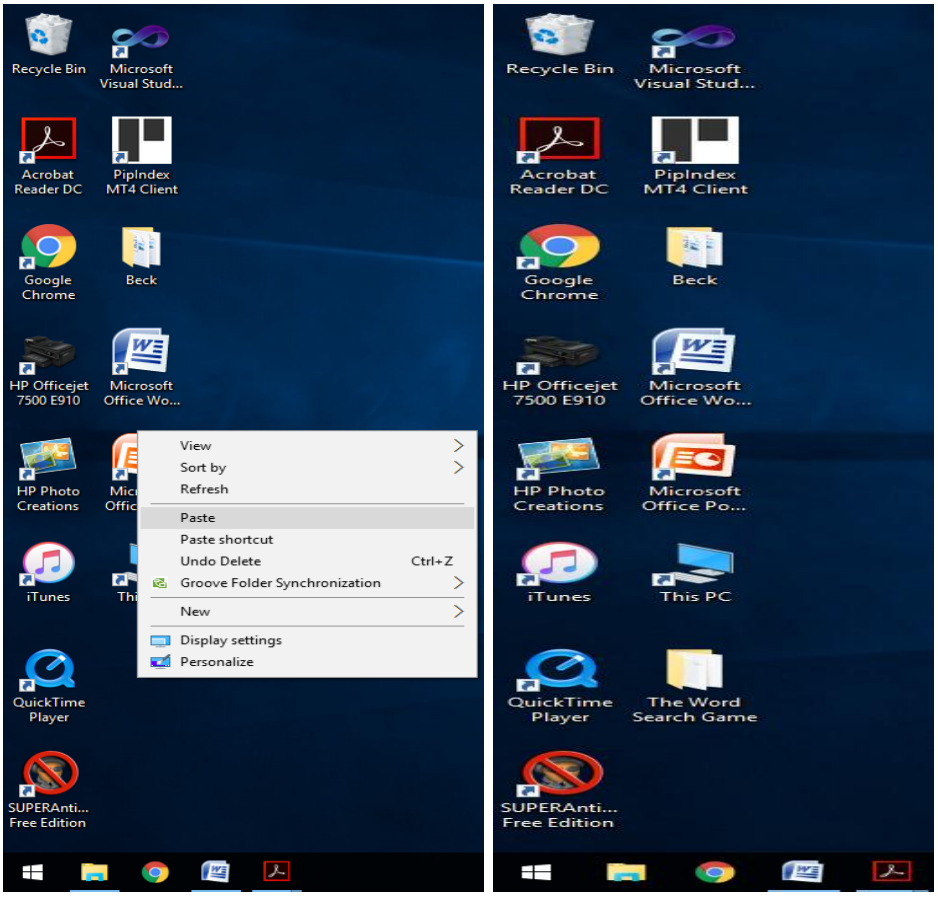
This is the USB icon you want to click on:

Once you have clicked on the USB icon, you will see everything stored in the USB. You will need to copy the folder “The Word Search Game”. You can copy the folder onto your computer’s desktop by using the key combination CTRL + C, or by a right mouse click on the folder “The Word Search Game” and then a left mouse click on the copy option.



Next, you want to find a place on the computer where you would like to copy the game, for example on your computer's desktop.

You will now want to paste your new file into the new location, you can do this by using the key combination CTRL + V, or alternatively by right clicking the mouse and selecting the "paste" option by using a single left click on the mouse.



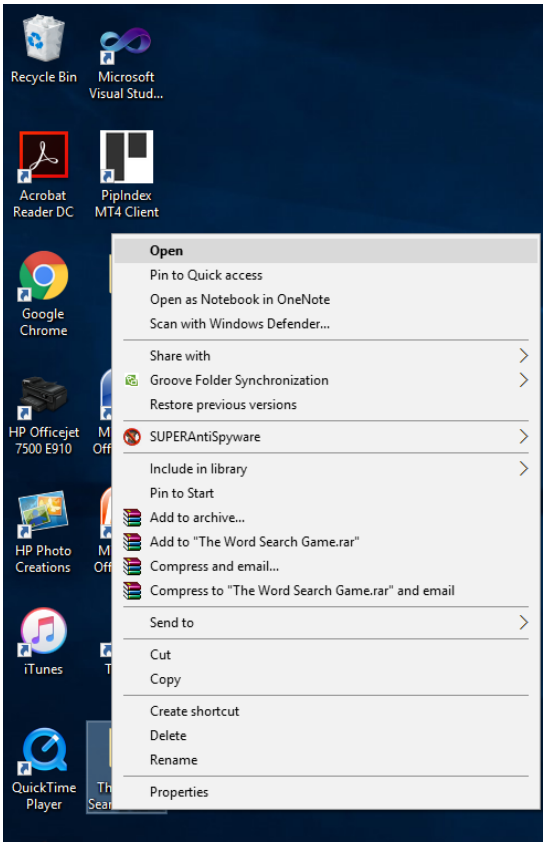
The game has now been copied onto the computer.

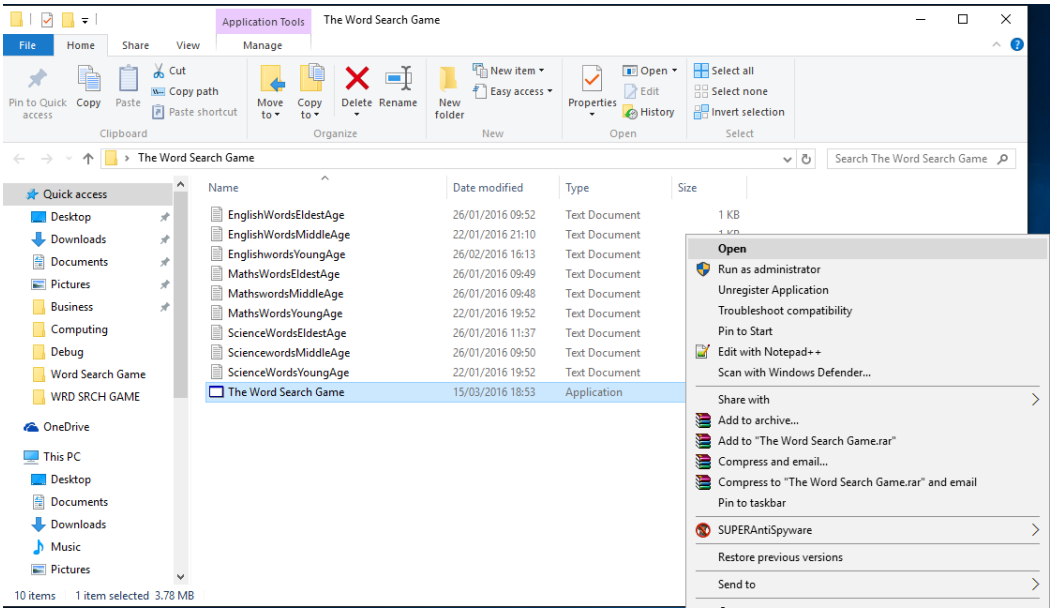
2) Getting Started

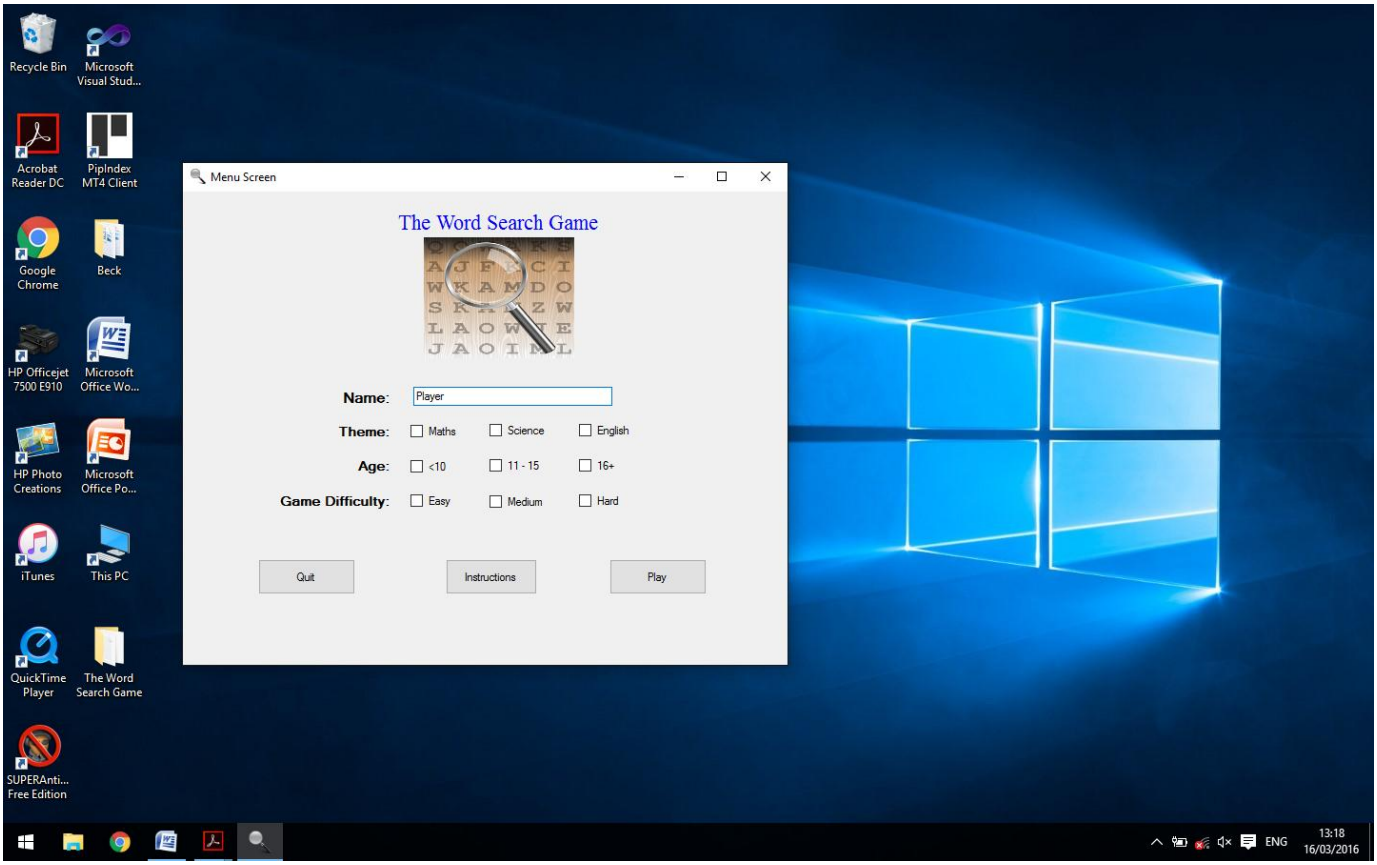
2 a) Starting the application

How you will load the game:

1. Go to the place where you stored the game
2. Double click on the folder to open it, or right click and then select open
3. Double click on the application file, or right click and then select open



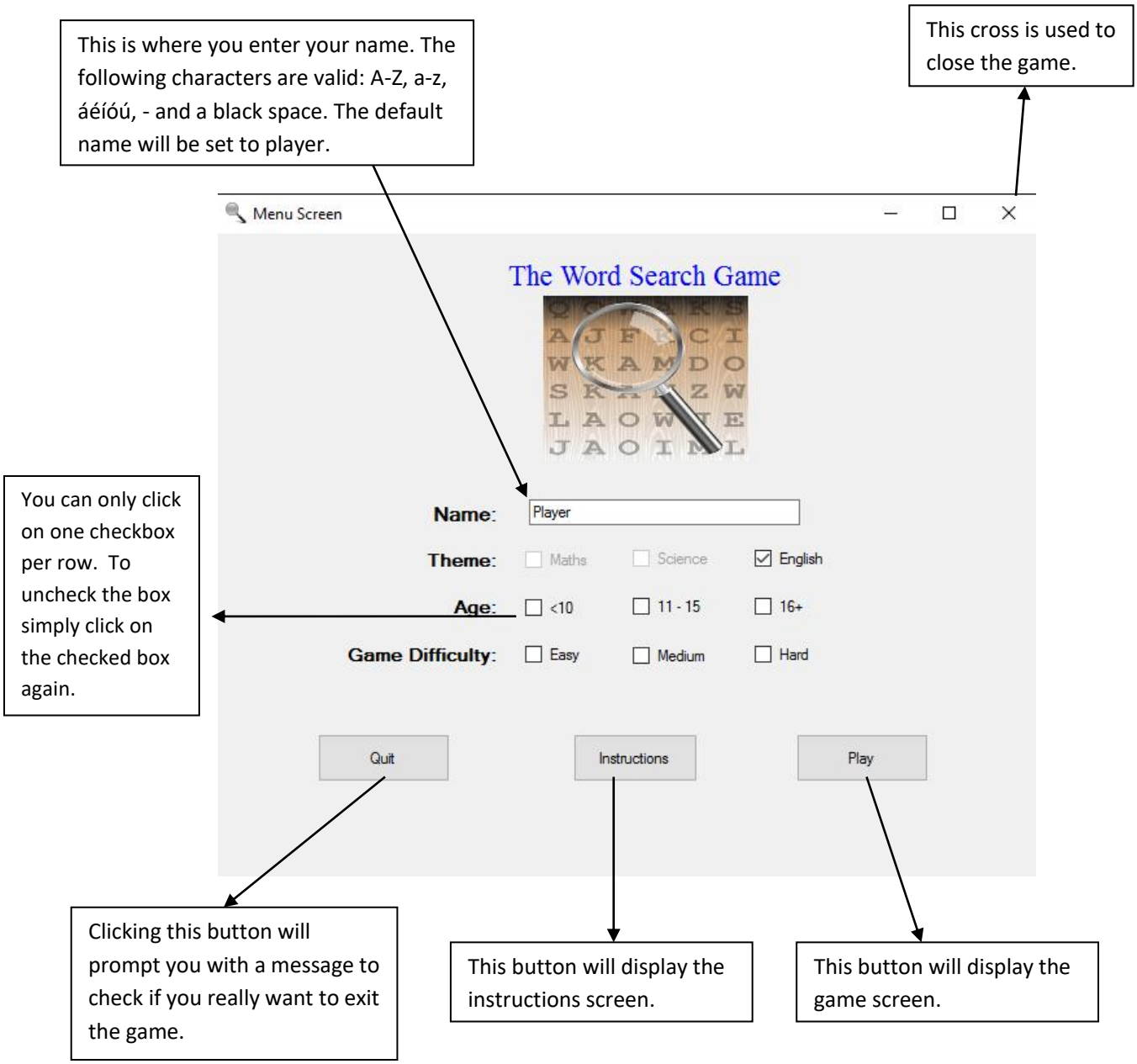




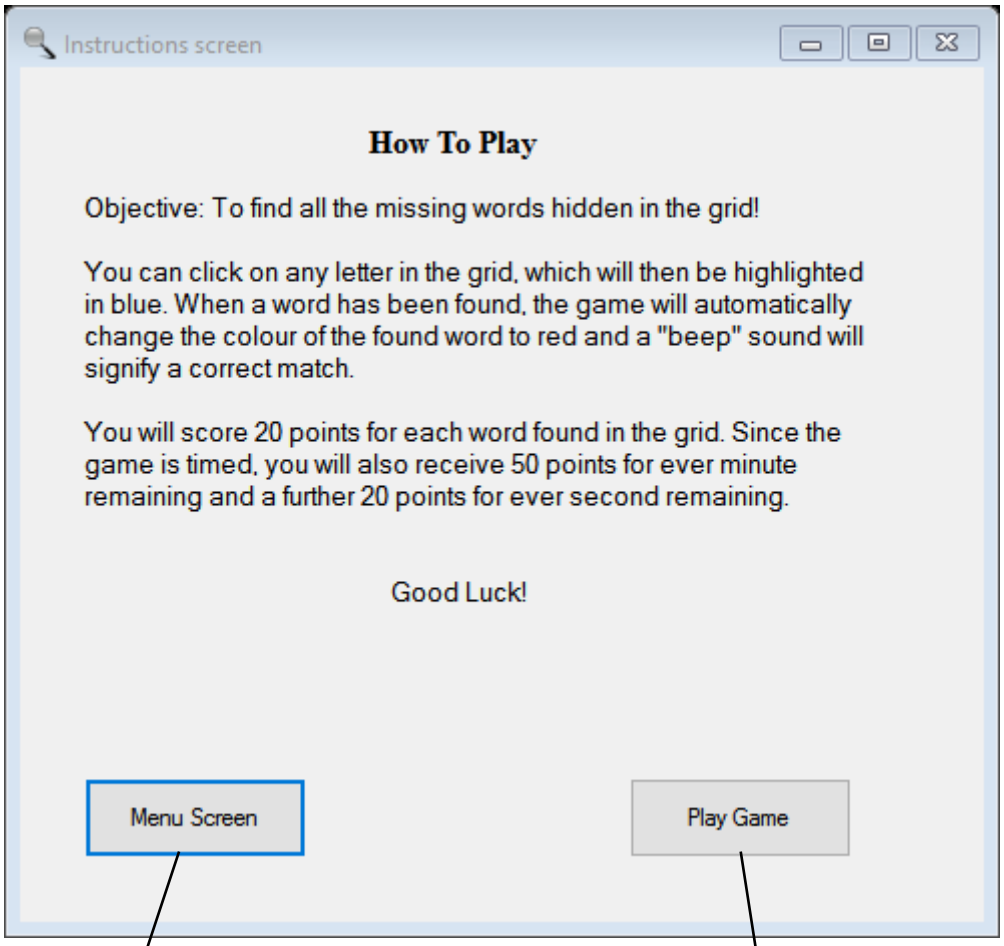
The game opens immediately, starting with the initial screen.

2 b) Navigation

Menu screen:



Instruction Screen:



Clicking this button allows you to return to the “Menu Screen”.

Clicking this button allows you to start a new game.

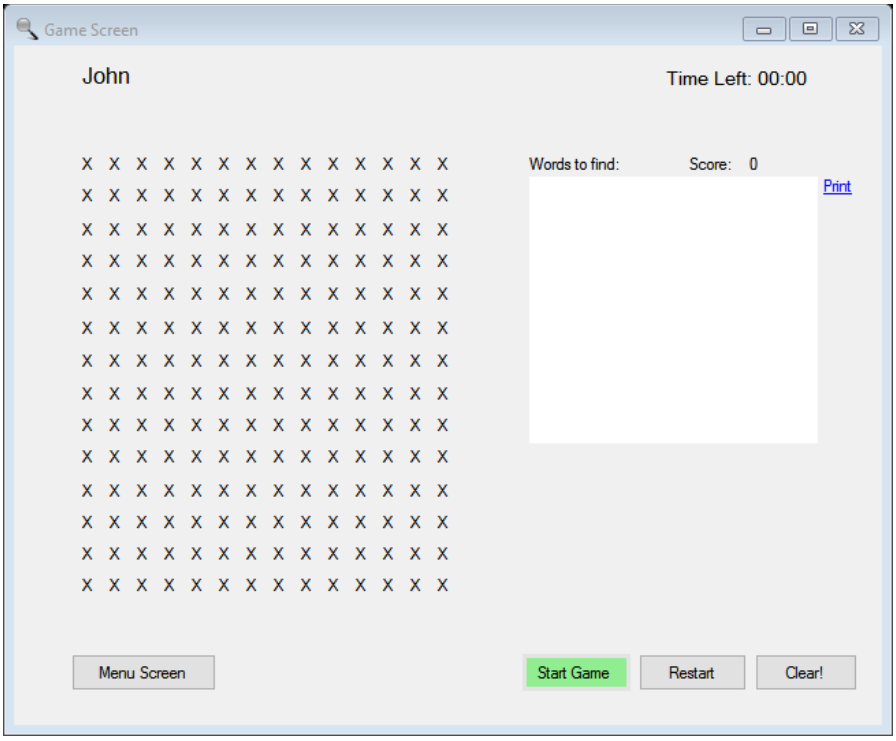
3) How to play the game

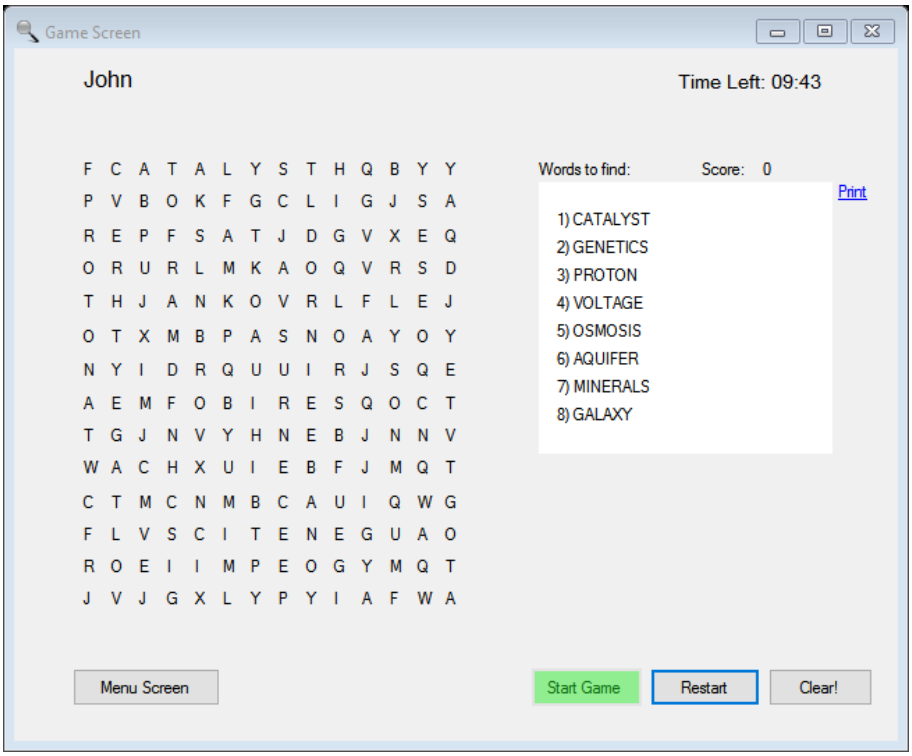
3 a) Starting the game

The Word Search Game does not automatically start. This is because the timer can only begin its count down when you are ready to play the game. Likewise, the game does not assume you want to restart immediate and so a restart button is there if required.

To start the game, you must click the **green** "Start Game" button, as outlined below.

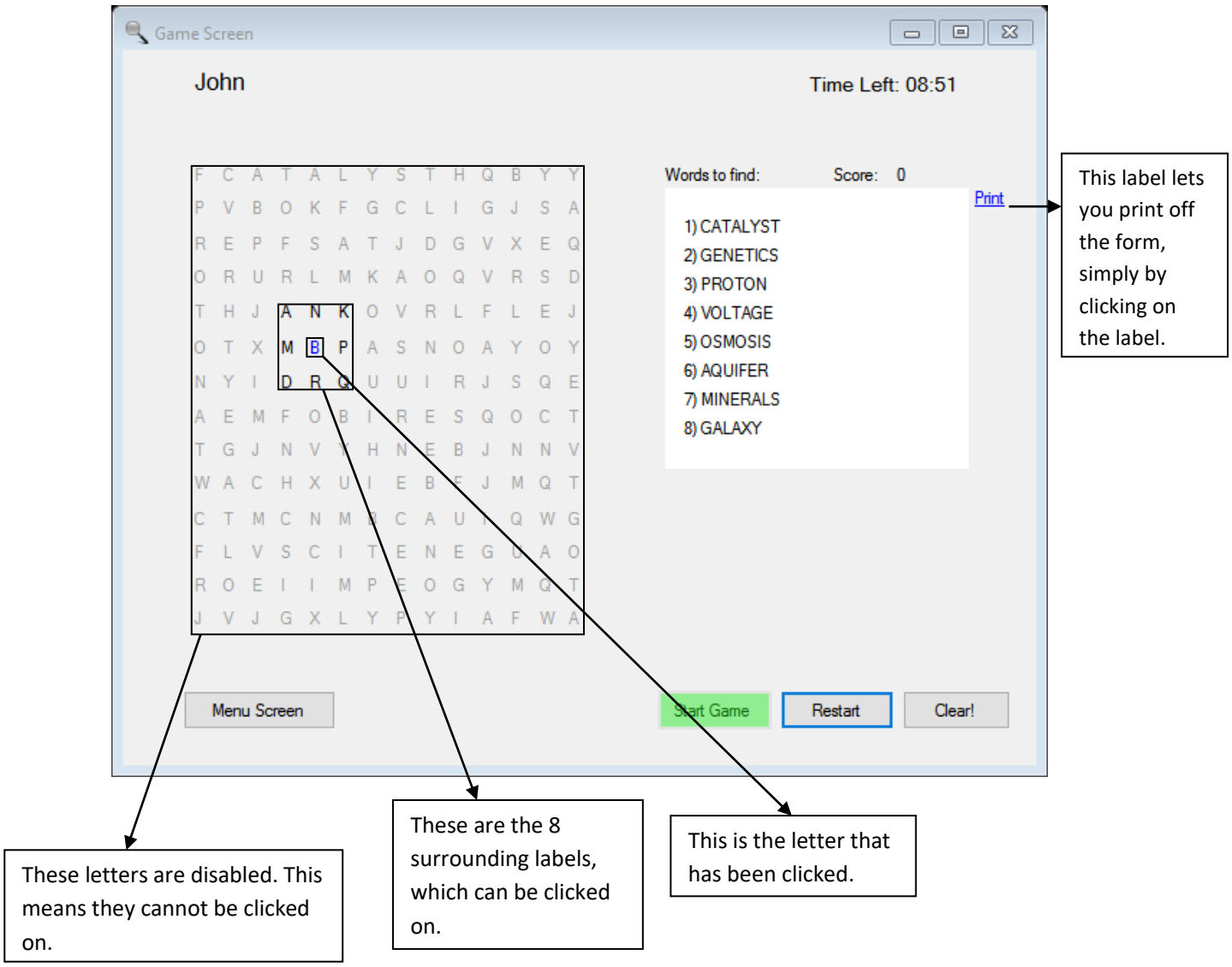
To restart the game (this will start a new game), you need to click on the "Restart" button, which is next to the "Start game" button.



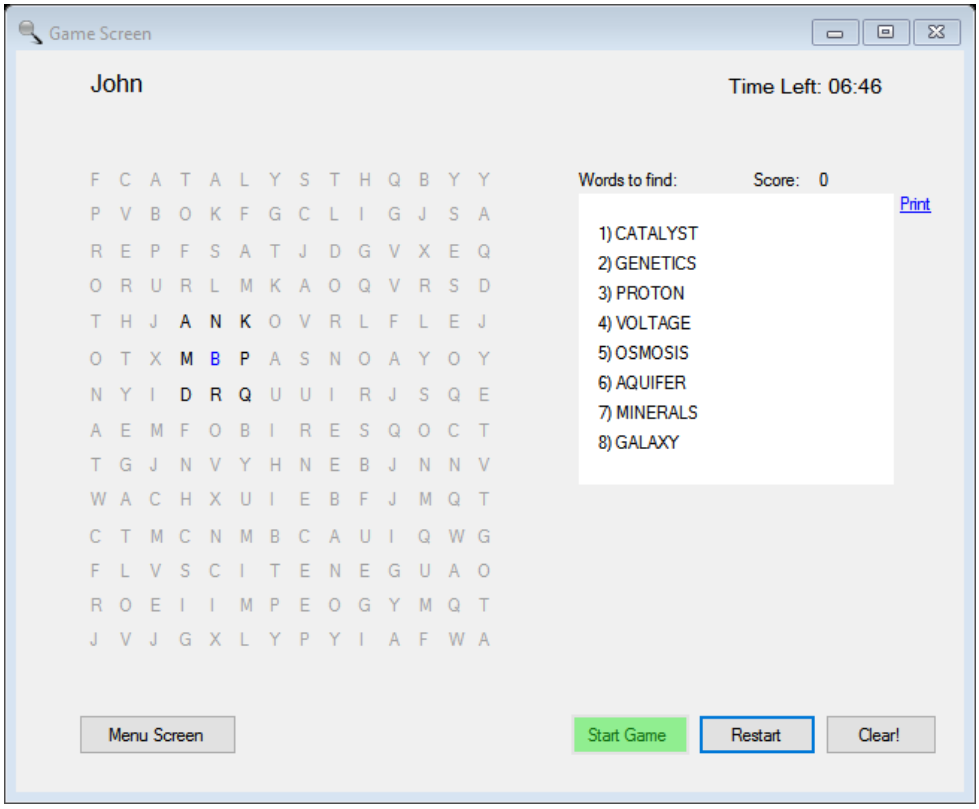


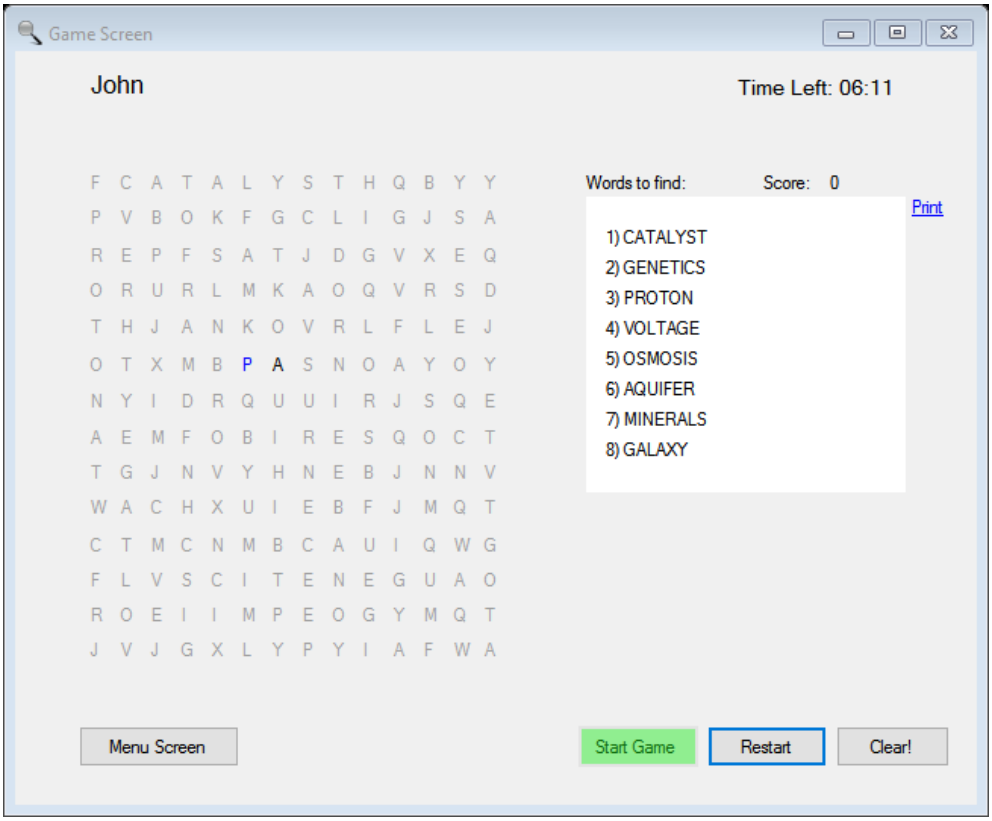
3 b) Clicking on a letter

To find a missing word is simple, all you have to do is click on any letter in the grid. However, once you have clicked on a letter, all other labels are disabled except for the 8 labels which surround the clicked label. These 8 surrounding labels are in the following direction: North, North-east, East, South-East, South, South-West, West and North-West.



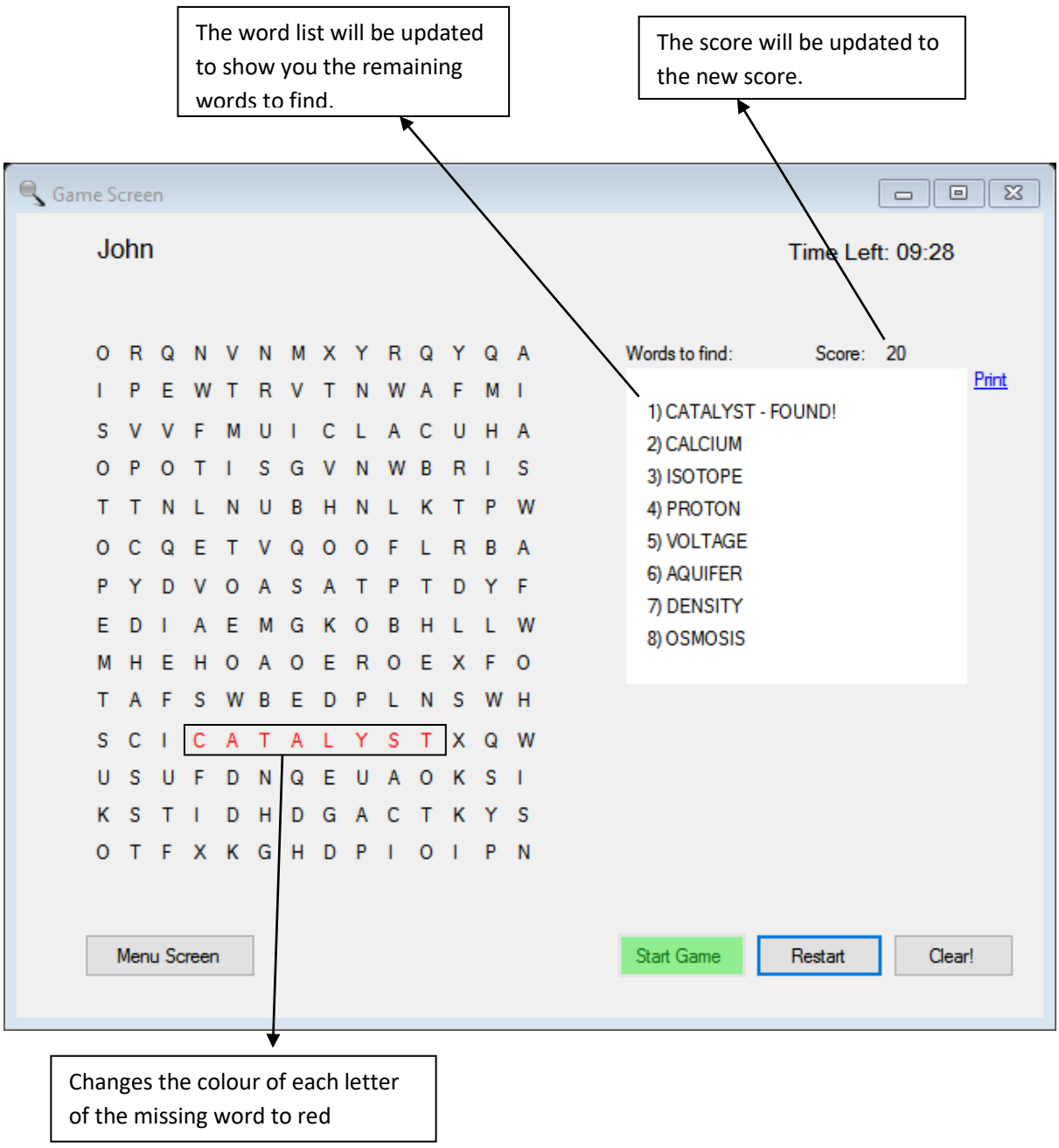
Note: The **first** click on a letter will only enable the 8 surrounding labels. **Every other letter click** will only enable the next label in that direction you wish to go. See the example below, where I can only click on letters going in the east direction.


Game Screen
John
Time Left: 06:46
F C A T A L Y S T H Q B Y Y
P V B O K F G C L I G J S A
R E P F S A T J D G V X E Q
O R U R L M K A O Q V R S D
T H J A N K O V R L F L E J
O T X M B P A S N O A Y O Y
N Y I D R Q U U I R J S Q E
A E M F O B I R E S Q O C T
T G J N V Y H N E B J N N V
W A C H X U I E B F J M Q T
C T M C N M B C A U I Q W G
F L V S C I T E N E G U A O
R O E I I M P E O G Y M Q T
J V J G X L Y P Y I A F W A
Words to find:
Score: 0
Print
1) CATALYST
2) GENETICS
3) PROTON
4) VOLTAGE
5) OSMOSIS
6) AQUIFER
7) MINERALS
8) GALAXY
Menu Screen
Start Game
Restart
Clear!


Game Screen
John
Time Left: 06:11
F C A T A L Y S T H Q B Y Y
P V B O K F G C L I G J S A
R E P F S A T J D G V X E Q
O R U R L M K A O Q V R S D
T H J A N K O V R L F L E J
O T X M B P A S N O A Y O Y
N Y I D R Q U U I R J S Q E
A E M F O B I R E S Q O C T
T G J N V Y H N E B J N N V
W A C H X U I E B F J M Q T
C T M C N M B C A U I Q W G
F L V S C I T E N E G U A O
R O E I I M P E O G Y M Q T
J V J G X L Y P Y I A F W A
Words to find:
Score: 0
Print
1) CATALYST
2) GENETICS
3) PROTON
4) VOLTAGE
5) OSMOSIS
6) AQUIFER
7) MINERALS
8) GALAXY
Menu Screen
Start Game
Restart
Clear!

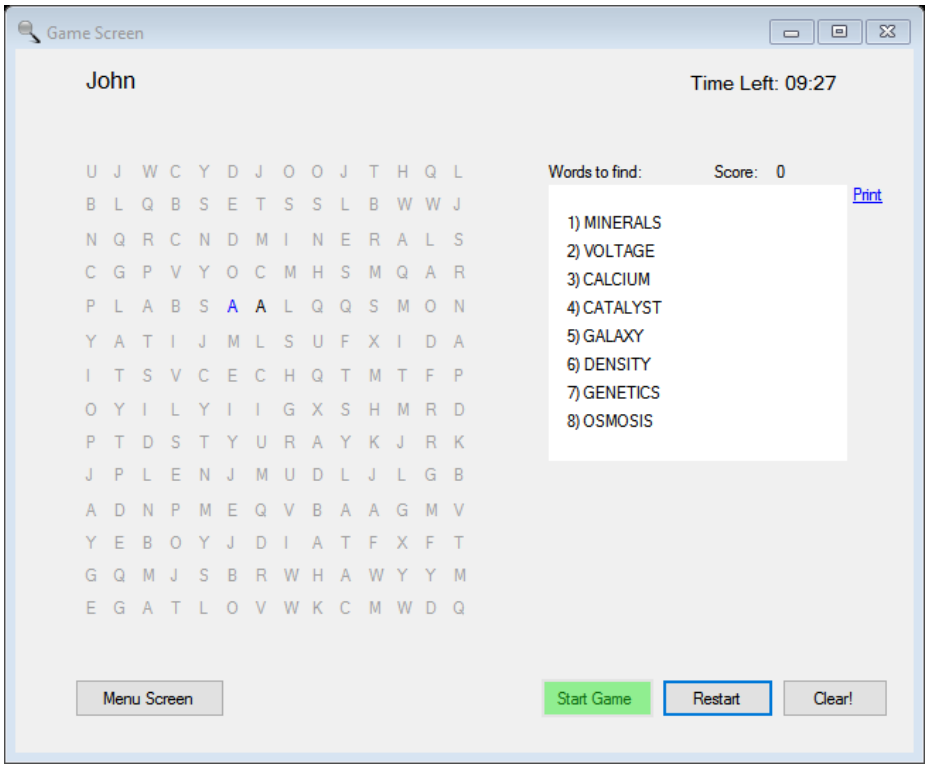
3 c) When a word is found

The objective is to find all the missing words. When you have clicked on each letter of the missing word, the game will output a beep sound and do the following:

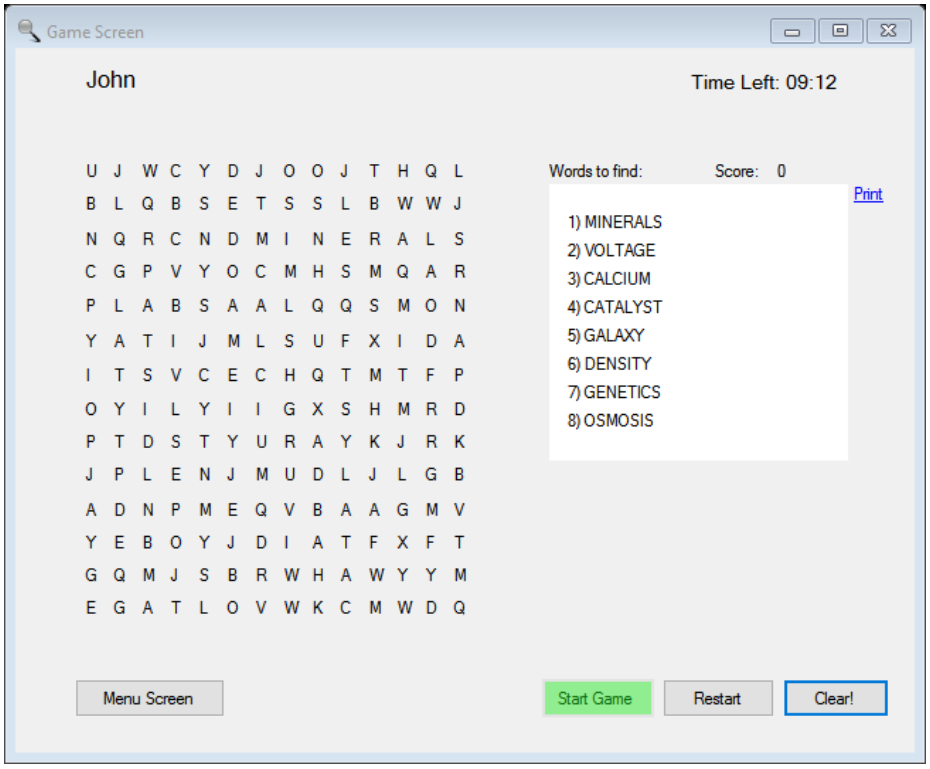


3 d) Clearing Mistakes

At times we feel like we have found a word, but actually it was a mistake! The Word Search Game has a "Clear" button, which we can use to clear our mistake(s).



Snap! You have made a mistake. The next stage is to clear this error, by clicking on the "Clear" button.

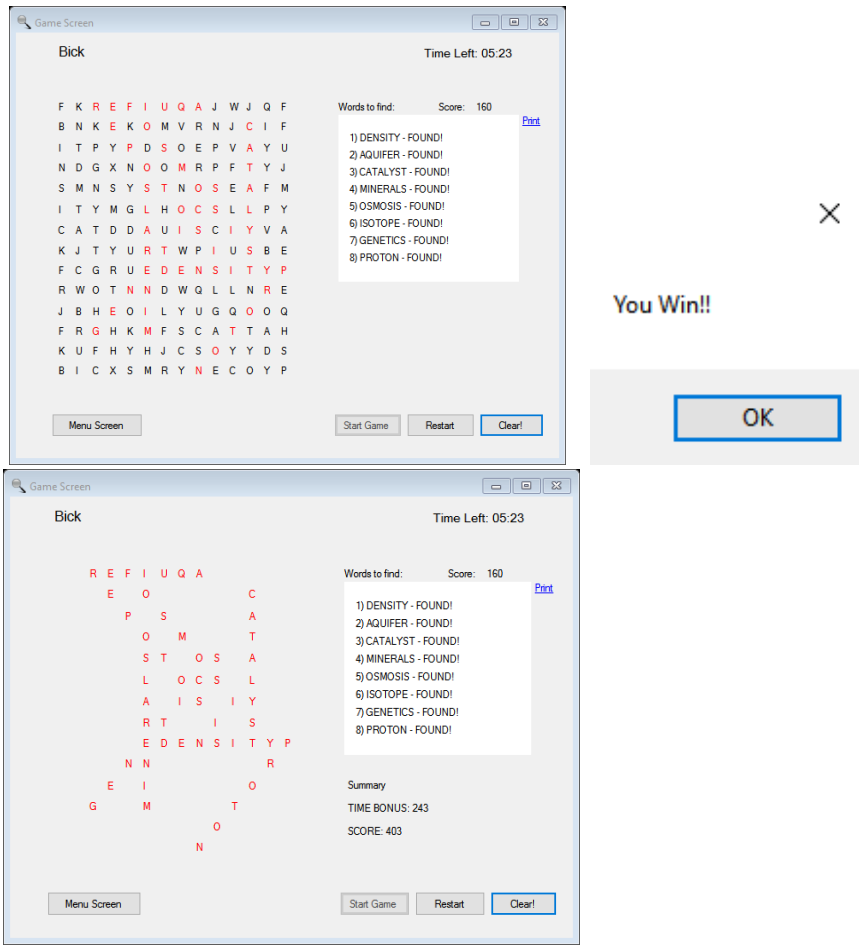


Well done! You have now cleared that silly mistake ☺.

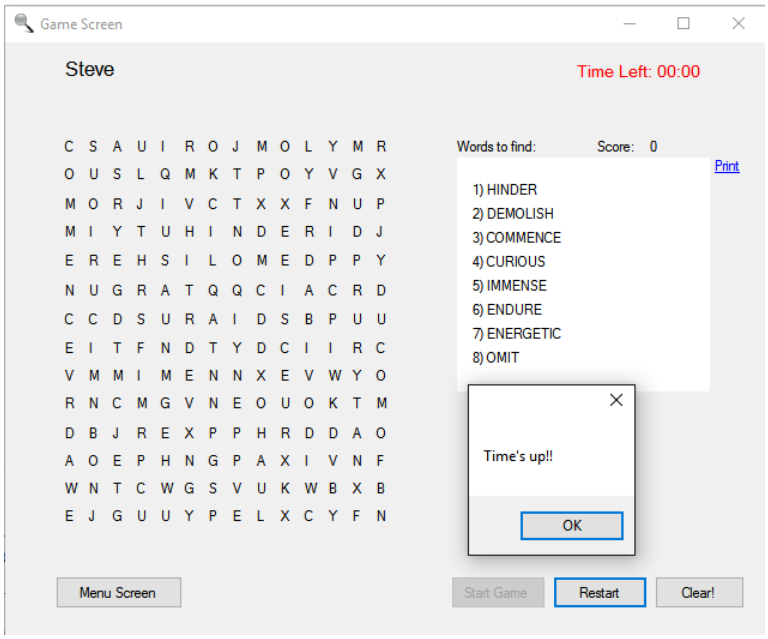
3 e) How the game ends

The Word Search Game can end in two ways:

1) When you have found all the missing words, you will be present with the screens below.



2) When the timer reaches zero, you will be presented with the screen below.



Backup

Backing up the text files

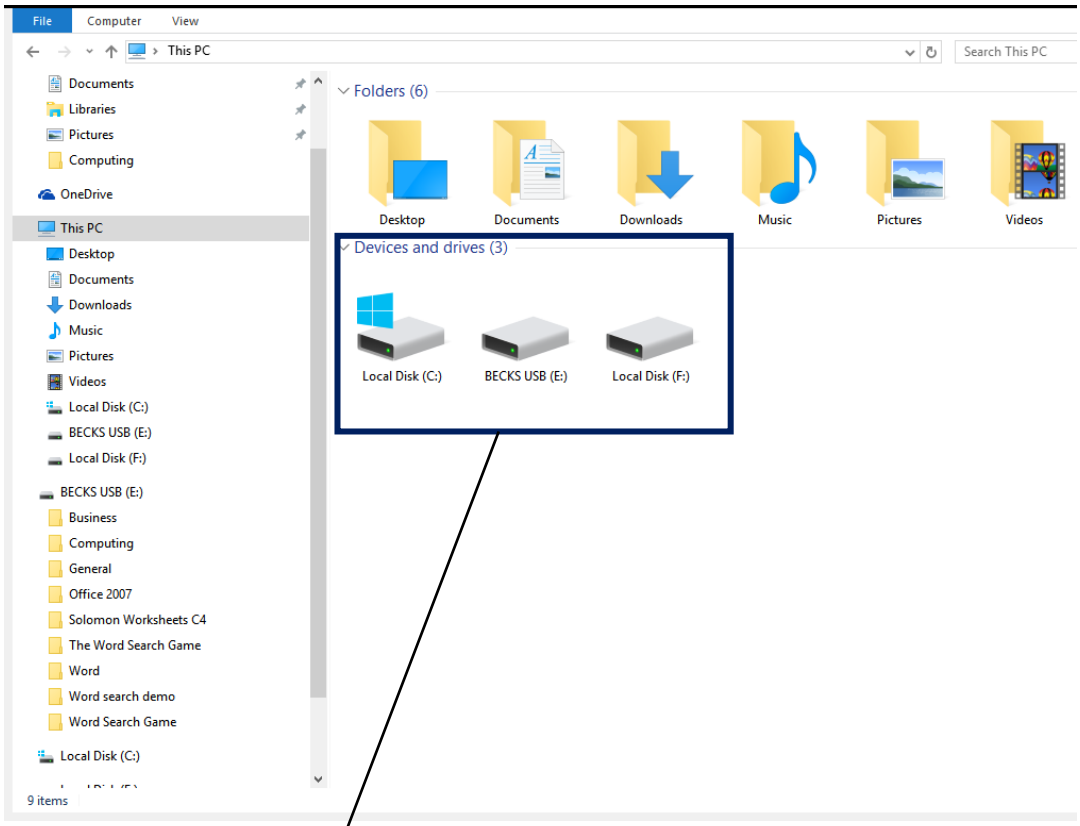
The Word Search Game uses 9 text files, which can be found in “The Word Search Game” folder on page 107. It is important to back up these text files, so that they can be recopied into “The Word Search Game” folder if necessary.

We can save the data onto storage devices, such as on a memory stick, on a CD, or on a hard drive.

Storage Device

I will use a memory stick to store the backup data.

The first stage is to insert the memory stick into the computer USB port. When that has been done, you want to go to “My Computer” and find the section named ‘Devices and drives’.

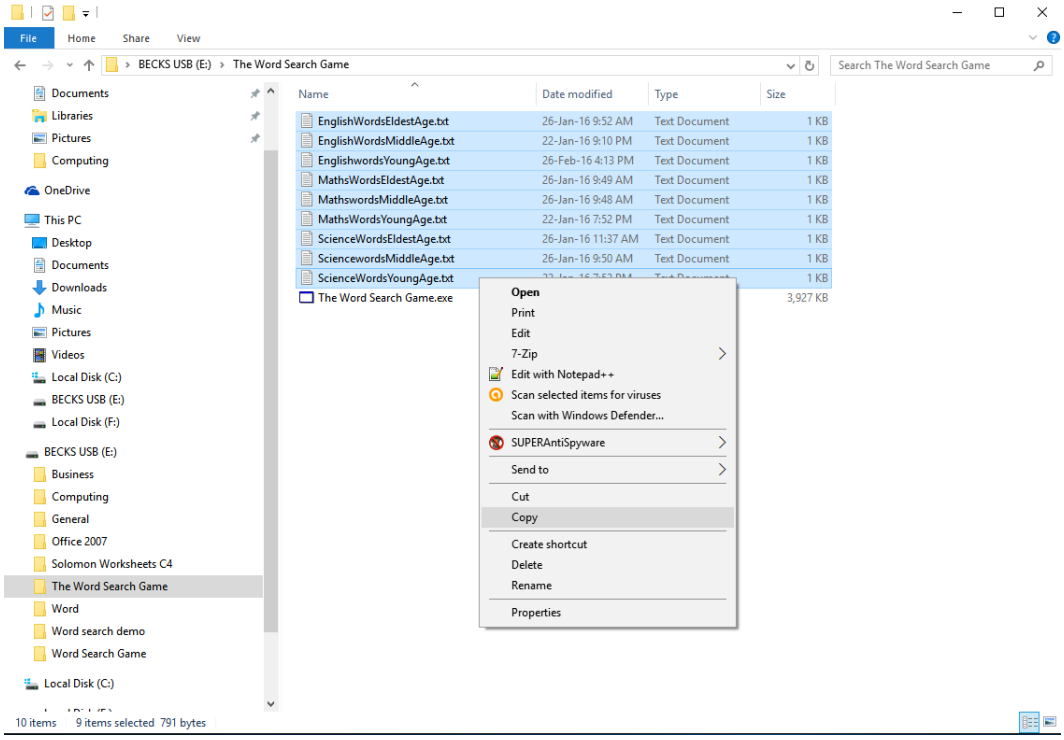


This is the Devices and drives section, which shows the available devices recognised by the computer.

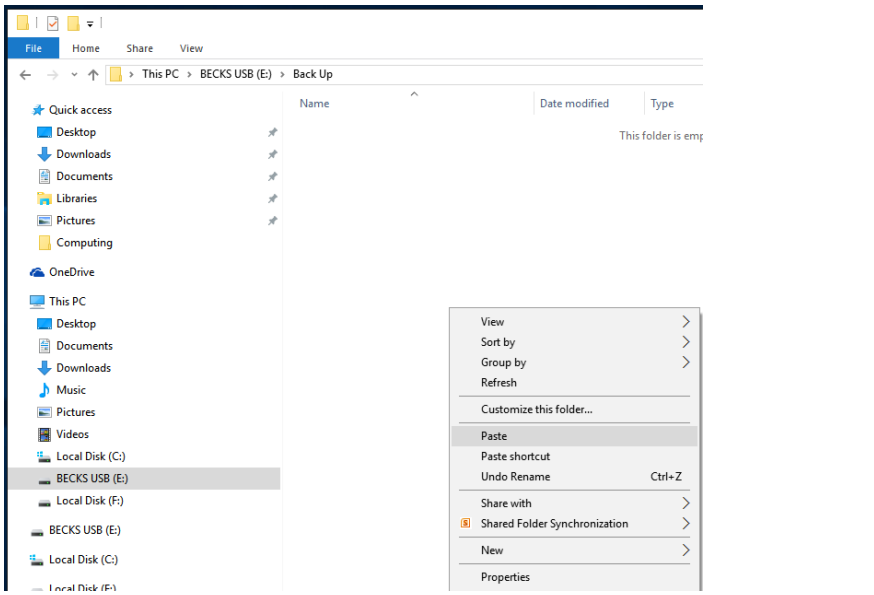
You need to double click on the device you would like to back up your files onto.

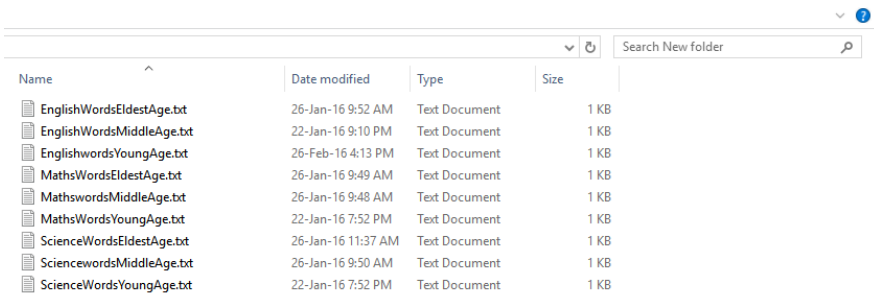
The next stage is to open the “The Word Search Game” folder and highlight all the text files. You can highlight the text files by clicking on each text files, whilst holding the Ctrl key.

When all the text files are highlighted in blue, right click and press copy.



Now that the files have been copied, go back to your memory stick and paste the content by using the key combination Ctrl + V, or alternatively right-click on the memory sick folder and click paste.





Good job, now all you files are back up onto your new device.

Note: You can also back up the complete “The Word Search Game” folder in the same way as backing up the text files. Remember, there is no limit to how many times you back your files.

Troubleshooting

4 a) Download and Installation issues

Download and installation problems

Problem	Possible cause	Solution
Program does not save or copy across.	There may not be enough space on your hard drive or storage device you wish to copy the game to.	Make sure there is enough space on the storage device. To check if there is sufficient storage space on the device: 3) Open "My PC" 4) Right-click on the local drive C 5) Click on "properties"
The game does open or fails to open after numerous attempts.	The current copy of the game is bad or has become corrupt.	Delete the current copy of the game and try to recopy the game from the original USB.
	You may not have the correct operating system to run The Word Search Game	Verify that you have the correct operating system to run the game, which is Microsoft Windows 95 or later.
The game cannot find the text files, for example, EnglishwordsYoungage.txt is missing.	You have not copied across the folder with the text files, or the text files may be missing in the folder	Check "The Word Search Game" folder to see if all 9 text files are there.

4 b) General issues

General issues

Problem	Possible Cause	Solution
The game appear to lag and freeze up	The computer doesn't have adequate RAM	Ensure that your computer/device has enough 512 megabytes (MB) of RAM
	The computer does not have adequate processing power	Make sure the device has a least 1.3 GHZ processing speed
The game stops working – the timer will not continue to count down	N/A	Close the program by clicking on the red cross at the top of the window. If the above fails, then try the key combination Ctrl + Alt + Delete. Open task manager and scroll down the processes list until you find "The Word Search Game", by this point you can end the task by clicking the button "end task".
Not enough paper error message	There is not enough paper in the printer	Double-check the printer is not empty and add extra paper if necessary.
Low ink or insufficient ink	There is not enough ink in the printer	Add ink to your printer, or alternatively, try printing in black and white if colour is not required.

Glossary

Term	Glossary
Key press combination: Ctrl + Alt + Delete	A group consisting of three buttons, which if pressed at the same time, will open the window's security options.
USB	This is a "Universal Serial Bus"; it is used to transfer data from one machine to another. A USB is inserted into the device's USB port. Data from a USB can be copied onto the device.
RAM	Stands for "Random Access Memory" and is used by the processor to temporarily store data for processing.
Megabytes (MB)	A unit to measure the number of bytes in data. 1 megabyte is equal to 1000000 bytes.
Form	A window used to input and display data
Window	A separate viewing area on a computer display screen.
GHz	This is an abbreviation of gigahertz. GHz is a unit of measurement for frequency.
Hard drive	A disk drive used to store data to your computer.
Operating System	Software that supports a computer's basic functions, which allows other programs to run.
Label	The text that you see on the forms. For example, the letter "B" or the name "John".
Processor	A part of the computer that carries out calculations and processes instructions.
Backup	Another copy of a file.
File	A collection of data stored under a single identifying name.

Evaluation

(i) Discussion of the degree of success in meeting the original objectives

The only requirements that were not met, was the size of the game screen and the amount of time the user is given when the select the easy, medium and hard categories. My users have accepted this fault, since The Word Search Game has met all the other objectives. Furthermore, the user agreed to accept the downfalls. This can be seen from the beta testing questionnaire on page 99.

My users and client accepted the failure of the objectives above, since The Word Search Game solved all other objectives stated. My users and client response to whether they were satisfied that The Word Search Game met all other objectives is shown following this section on page 125.

I will now assess how well The Word Search Game met the original requirements agreed on page 19.

Requirement	Why the requirement was met
Design a: The screen size will be large (800 * 800).	This requirement was not met, since the screen size for the different forms match those on the updated design objectives and not those on the original requirements.
Design b: All screens will have a grey background.	All the screens have a grey background, which can be seen on various screens from 58 onwards.
Design c: The time interval for the timer will be 1000 milliseconds.	The timer interval for the game is exactly 1000 milliseconds. This can be seen on the image of the timer's properties on page 90.
Design d: There should be a print label, on the game screen to print out the form.	There is a print label on all the game screens from page 58.
Design e: The grid size will be 14 letters wide by 14 letters length.	Each letter in each row and column add up to 14. This can be seen on every game screen from pages 58 onwards.
Design f: There will be a checkbox for the different game difficulties.	For each option in the different categories the user is provided with a checkbox to select their preferred game difficult and theme also to select their age. This can be seen on page 58 onwards.
Design g: There will be a checkbox for the different age groups.	
Design h: There will be a checkbox for the different game themes.	
Design i: The user should be able to uncheck any checkboxes and recheck it if necessary.	Fortunately, The Word Search Game does not allow the user to select more than one box in the same category. Therefore, if the user selects a checkbox in a category, they must uncheck it before selecting another checkbox. This can be seen on page 94 to 95.
Design j: There should be a clear button on the game screen, to clear any mistakes made.	There is a clear button on the game screen which can be screen on every screen from page 58 onwards.
Design k: There should be an instructions screen.	There is an instruction screen; this can be found in the user guide on page 110.
Input a: The player can go back to the menu page from the instructions screen, by clicking on the menu button.	From my beta testing results we can see that the users said that all the buttons work and that it is easy to navigate around the system. The beta test results can be found on page 99.

Input b: The player can read the instructions by clicking on the instructions button.	When showing the game to my client through acceptance testing (Page 102), my client was satisfied with module 4 a, which was that they could read the instructions by clicking on the instructions button.
Input c: The user can enter their name in the textbox – only letters, space, full stops and hyphens should be allowed	The requirement was met, because the text box did not allow any unnecessary character, other than those stated in input c. The text box field was test thoroughly on page 93 and 94.
Input d: The player can click on the quit button, which must then prompt the player with a verification question. For example, "Are you sure you want to quit?"	Upon clicking the quit button the player was prompted with a verification message. This can be seen on page 89, where I performed alpha testing to see whether I was prompted with a message. Furthermore, the results from the beta testing showed that all of my user could exit the game. The results can be found on page 99.
Input e: The player can then click "Yes" to exit and "No" to cancel.	Same as above.
Input f: The player can click on a letter in the grid of words to find	This was met, as the player can select any letter in the grid. This can be seen from the various testing of the mouse click on the letters in the grid on page 95 - 98.
Input g: The player can click on the "Clear" button	This was tested during the development of the clear button subroutine. The screenshots of the tests are found on page 88.
Processing a: The player is given 15 minutes for the easy game difficulty, 10 minutes for the medium game difficulty and 5 minutes for the hard game difficulty.	This objective was not fully meet, because the game time for the hard more needed to be increased to 10 minutes. Furthermore, my beta test said that there was enough time to play the game, which is why the game time had to be increased. This can be seen on page 99.
Processing b: Every time the value of the timer changes, the system should check if the value of the timer equal zero, or if all the words have been found	The game was able to detect when the player has won the game by finding all the words and when the time was up. The screenshots of the test can be found on page 91.
Processing c: When the "Start" button is clicked, the timer should begin its count down and the grid should be filled with random letters and words	Upon clicking the start button the player is present with a grid full of random letters and the missing words (words to find) page 74.
Processing d: When the printout icon is clicked the system should display the print dialogue	Page 92 shows how the print label print dialogue box, when the user click on the print label.
Processing e: When the instructions button is clicked the current form will close and the instructions screen will open.	The instructions screen opens when the instructions button is clicked and the current form closes. This can be seen from acceptance testing 4 a) results are on page 102, where they agree the can read the instructions and progress onto the game.
Processing f: When the "Play" button is clicked the current form will close and the game screen will open	The current form closes and the game screen is shown. This can be seen from the acceptance testing of the game, were the client was satisfied the objective was met. This can be seen on page 102, objective 4b)
Processing g: Every time a new letter is clicked in the grid, the system will check if a word has been	It is hard to really prove this objective, because the user cannot actually see the process happening in

found.	real-time. However, there is evidence on page 86 were I test the system by finding a word.
Processing h: When a letter is clicked in the grid, its colour will turn blue.	Any letter that is selected in the grid does change colour from black to blue. This can be seen from 79.
Processing i: When a word is found, change the colour of each letter in the found word to red.	When a word is found each letter of the found word changes colour to red. This can be seen on from page 86 onwards. There is clear evidence of this test on page 97, test number 10.
Processing j: Award 20 points for each word found.	The user is awarded 20 points for every word found, which can be seen on page 97, test number 10.
Processing k: When all the words have been found, the user should be awarded 21 bonus points for each minute and 6 points for each second that remains on the timer.	The user is awarded 21 points bonus points for each minute and 6 points each second that remains on the timer. There is a screenshot of this test on page 91.
Processing l: The system must record the number of words that have been found.	The system record the number of word found, because from evidence on page 116 we can see that the game stops when all the words have been found.
Output a: A message box should be displayed when the player wins the game or when time is up	A message box is displayed when the user wins the game or when time is up. This evidence can be found on page 116.
Output b: A beep sound when a word is found.	The system does output a beep sound when the user finds a word. This can be seen from the beta testing results on page 99.
Output c: The user's score.	The user score is shown on all the game screens from page 86 onwards.
Output d: The words to find list.	The word list is show on all the game screens from page 59 onwards.
Output e: The remaining game time.	This can also been seen on all the game screen from page 86 onwards.

(ii) An evaluation to the user's response to the system

In order to receive user feedback from the game, I have design a questionnaire, which my end-users will fill out. After collecting all the results, I have tabled the results from the questionnaire.

Question	Answer/Comment
Is the game enjoyable?	Yes (10) – I really liked the hard mode, it makes me more competitive. No (0)
Was there clear on-screen help?	Yes (10) No (0)
Is the user-guide clear and easy to use?	Yes (10) No (0)
Was it easy to enter your name and select the checkboxes?	Yes (10) No (0)
Did The Word Search Game meet its requirements?	Yes (10) No (0)
Do any system faults still exist?	Yes(0) No (10)
Are you happy with the colours used in the game?	Yes (8) No(2) – I would prefer to see some lighter colours, grey is a bit "dull"
Are there any problems with the game?	Yes (0) No (10) – The system works well
What new feature(s) do you think could be added in the future?	• It would be great if the system could save your highest score or the quickest time to complete the game • A negative beep for finding the wrong word and a positive beep for find the correct word. • You could add bonus time for find the correct word.

Signature: …… **Date:** ……

My response to the feedback

The feedback received is positive and my users have shown that they are happy with the new system and are satisfied that the new system meets all its requirements. Any problems revealed in acceptance testing have been noted and the new system modified. The only changes that need to be made are those that the users have stated in the comments above. I will explain these changes in the next section, since these comments form part of the desirable extension.

(iii) Desirable extension

Having completed the whole project, it is now time to reflect on the whole process. I shall comment on the overall success of the project, highlighting the good and the not so good points. Furthermore, I will comment the limitations and how I can improve the system further in the future.

Good points

- Enough testing ensured no bugs existing in the game, as shown on page 93.
- All the checkboxes, buttons and textbox function well.
- The design of the system matches closely to the original designs in the design specification. This is further praised by my users in the feedback given on page 124.
- The game is fun and enjoyable as revealed by the user on page 124.
- The was sufficient on-screen help as reveal from the user feedback on page 124

Bad points

- Due to very little time I was unable to fully add all the additions my users wanted.
- The system sometimes runs slowly due to inefficient algorithms.

Limitations and Extensions

The system was unable to save the highest score or the quickest time to complete the game. The game will show the user their score at the end. However, a new algorithm was required to hold the users' previous score and to compare it with their new score, to display their highest score.

The system was a bit slow at times; this was partly due to the inefficient algorithms used to create the game. A possible extension would be to reduce this inefficiency by finding more efficient ways of coding the algorithms. The current system uses a lot of iterations, which could cause the system to lag at times.

My users would like to hear a different beep tone, to signal whether a word is found or not. The current system already outputs one type of beep sound when a word is found, so as a future improvement I could add a different beep sound, i.e. when a word has not been found then a different sound would play through the speakers.

My final extension, which was requested by my user on page 124, is to add bonus time for finding a word. The current system add bonus points for when a word is found, by taking the current score and add a bonus score to it. I can add bonus time to the current time in the same way the current score is changed.