ICS2 – Programming Assignment 2

Topics: Everything up to Flowcharts

Marking:

We will be looking for a number of things for this assignment. **Remember that just because your program works does not mean you will get full marks; this is actually only a portion of the final mark for the question.**

- **Style**
  - Documentation (Header, variables, internal comments)
  - Structure (Indentation, white space, code location)
  - Decisions (Variable Naming, documentation descriptions, data types)
- **Design**
  - Although, not directly evaluated you may not begin work until your design is approved
- **Correctness** (Your program produces the expected results)

Academic Integrity

Plagiarism and other acts of academic dishonesty will be dealt with harshly in accordance with the school's rules. Talking about the questions is allowed, this is called collaboration. **Copying/Using/Giving code from/to friends or other sources is considered cheating.** To ensure you do not cross this line; when talking about a question with someone do so away from a computer with a paper and pen in hand and write out IDEAS, not code. Then when you are finished talking return to your computer and implement the ***ideas*** you have come up with. This is similar to doing research for a report, then applying your findings in your own way.

Follow the instructions below to get started…

- Download the most recent version of the Game Project and unzip it.
- Rename the folder PASS2 (You may have to delete any currently existing folders named PASS2 before renaming)
- When you are done Phase 2, Select the **PASS2** folder, **tap Alt** then **click File→Send To→Compressed(zipped) folder**, and submit the resulting .zip file on the Moodle.

Time Management

This time your assignment may potentially be much larger, it is up to you to decide how far you will go. That being said the level of detail and quality of your work will be reflected in your final mark. Do not waste time on this project; there is enough time to complete it in the class time given. However if you waste class time or you plan on adding enhancements you will most likely require outside work time.

This is the first program you will need to apply a substantial amount of logic. As such you may spend a lot of your time brain storming strategies and problem solving.

Phase 1 Design Option 2:

Your task is to complete the flowchart **ONLY** for a subprogram that would check what the health change will be after a successful move. The only input to the subprogram would be the current space the player is on and the String array of space types. The return value should be an integer valued at the change in health according to the hazard chart below. Remember, the flowchart is **ONLY** for this subprogram, which means you do not have to do any user input, that data will be contained in the parameters passed in. The signature line for this coded subprogram would look like:

```
private boolean CheckSpace(int curSpace, String[] spaceTypes)
```

Scenario: Back to the roots of gaming…

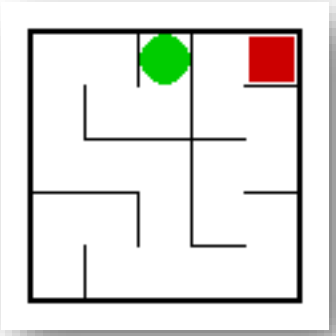
Before we had the amazing visuals you see in video games today things were much different for gamers. We were forced to visualize and use our imaginations much like you do when you read a book. One of the most popular genres of games 25 to 30 years ago was Text-Adventure; a classic example is the game Zork. Check out a remake here: http://www.web-adventures.org/cgi-bin/webfrotz?s=ZorkDungeon

These games were usually role playing games or adventure games that involved the user typing in what they wanted their persona in the game to do. Something like knock on door or look in mailbox or pick up shovel. As a user, you had to be precise, as the game could only recognize so many commands.

As a programmer they had to make their game flexible enough that the user could type in nearly anything. Yes, even for the usual profanities and slurs that all people would inevitably type in when they got frustrated with the game.

You will be creating your very own Text-Adventure game, the user's objective of your game will be to get from the start of a maze to the end of the maze, but they will encounter various dangers on the way. Your basic requirements do not require any visuals other than text, if you choose to enhance use the graphics lessons made available to you prior to this assignment.

General Requirements



1. Go to http://www.glassgiant.com/maze/ and generate a maze with **5 rows**, **5 columns**, **size 20** and Difficulty **Very Difficult.** (Repeat this step until you get a maze you like, the green circle is the start point, the red square is the finish point.)
    - Click on the generated maze to take you to a page of just the image
    - If you can right-click and Save-As **maze.png** in the **project folder**, do so.
    - If you cannot right click, take a screen shot of it by clicking on the **print-screen** button on your keyboard. Then open **MS Paint** and paste in by clicking **Ctrl-V**.
    - Use the selection tool to select only the maze and press **Ctrl-C** to copy.
    - Create a new picture by hitting **Ctrl-N (or File→New)**, no need to save your old picture.
    - Hit **Ctrl-V** to paste your maze in the picture and save your picture as **maze.bmp** in your **project folder** for the assignment (You will submit this with your project)
2. Create a program where the user only has 4 input options; Up, Down, Left and Right using the arrow keys.
3. Their goal is to get from the start point to the end point without dying and in as little time as possible.
4. The user will start with 100 health points, 100 seconds and on the start space in the generated maze.
5. The user will lose health by moving on top of a hazard space.
6. The use will gain health if they are lucky enough to land on a health pickup space.
7. If the user runs out time OR their health drops to 0 or less they die and the game is over.
8. When the game is over, the user is informed of the results and asked to play again
9. The start and finish are manually assigned to certain spaces which is decided by the maze you created
10. You will randomly assign a hazard, health pickup or empty space to the remaining spaces.

Phase 2 – Programming: Detailed Description

A Turn Sequence as Seen by the User (4 steps):

1. At all times the user can see the following information in a user-friendly format
    a. What space they are currently on
    b. How much health they have left
    c. How much time is remaining
    d. What happened as a result of the previous move (e.g. space type, health change, etc.)
    e. The program indicates to the user how to move (the arrow keys)
2. The user tells the program how they want to move (Up, Right, Down or Left) using the arrow keys
3. The program attempts to make the move and reports the results and consequences of that attempt on the screen. The program will then inform the user if they lose due to complete loss of health, out of time or if they won due to landing on the finish space.
4. All other statistics are updated (space number, health, moves made, etc.)

Move Consequences:

ACTION	CONSEQUENCES		
	Space	Health	Status Message (or your choice)
Move into a wall	Same	Same	Ran into a wall, possible loss
Move onto an empty space	Update	Same	Nothing of importance, possible loss
Move onto a hazard	Update	See Special Space Chart	Indicate hazard and damage, possible loss
Move onto a health pickup	Update	See Special Space Chart	Indicate the pickup and health change, possible loss
Move onto the end space	Update	Same	Winning message with time taken

NOTE: After a hazard/health is triggered, it is then converted to an "Empty" space so it cannot be triggered again.

How are the spaces on the maze numbered?

In order to be consistent across the class we will all use the same space numbering to know what space you are currently dealing with.

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

How is the space number updated after a move choice?

Look at the chart above and choose a space, how does that number change if you move Up, Right, Down or Left?

Hazards, Health pickup & Empty Spaces:

At the beginning of the game 2 spaces are set based on your maze, the start and end space. Both of these spaces are considered "Empty". However, the other 23 spaces are randomly decided **AT THE BEGINNING OF THE GAME (NOT EVERY MOVE)**.

There are 3 types of hazards and a health pickup, which all change the health and have a certain % of appearing in a space, the table below shows all the information you will need to assign your spaces in the maze.

Space Type	Health Change to User	Odds of being on a space	
Empty (Start/End are not random)	0	60% (Empty Space)	(1-60)
Trap	-20	15%	(61-75)
Enemy	-25	10%	(76-85)
Pit	-100	5%	(86-90)
Health Pickup	+15	10%	(91-100)

What does this mean?

Before the player starts playing a game, for EVERY space except the spaces for **START** and **END** you will decide what type of space it will be by generating a random number from 1 – 100 (see homework for help here). Depending on what the number is, a space type will be assigned to that space, e.g. if a 77 is generated, that space will be "ENEMY". This should happen in a subprogram that returns the space type and you store it into an array of 25 Strings. Meaning you will need to call the subprogram 23 times and the other two spaces (START and END) are just assigned to be "EMPTY".

Every time the player moves you will have to check what type of space they landed on by looking at the array using your current space as the index into the array and then apply the consequence, if any.

Enhancements (Want to earn that level 4+…this is the only way)

An enhancement is an improvement in the program that improves the overall experience. This can include many things; the creativity, quantity, quality and complexity of these enhancements is how you will be awarded marks. Below is a list of possible enhancements, do not feel limited by this list. If you have other ideas, USE THEM! Do you have to do all of these enhancements to get perfect? Not necessarily, the quality of the enhancement itself can sometimes be enough.

- Sound effects
- Background music
- Reset Game Option
- Images
- Animation
- Extra Statistics
- Visual Layout

NOTES:

Subprograms, Arrays & Constants

You program needs to make good appropriate use of subprograms you create as well as event subprograms. To do this remember subprograms should be used for both readability and reusability purposes. Any collections of related data should use an array. It is expected that there will be no magic numbers with the common and small +1 exceptions.

BE VERY CAREFUL OF PLAGIARISM IN THIS PORTION OF THE ASSIGNMENT, YES YOU MAY RESEARCH ON THE INTERNET AND LEARN HOW THINGS ARE DONE, BUT DO NOT SIMPLY CUT-AND-PASTE CODE FROM THE INTERNET. THIS IS CHEATING AND WILL BE DEALT WITH ACCORDINGLY.

Programming Assignment Rubric

The following rubric acts as a checklist, after grading each box will be in one of the three states shown above. These states will be used to determine your level for that achievement category.

NAME:

Achievement Category	Level	Achievement Category	Level
Knowledge & Understanding 1. Structure a. Indentation b. White Space c. Code Location d. Variable Blocking e. Logic Blocking 2. Subprograms and arrays effectively done a. Definitions b. Calling/Accessing c. Procedure vs. Function		**Application** 1. Software Correctness a. Game is fully playable as expected b. All required data is displayed as necessary c. Control works as expected (UP/RIGHT/DOWN/LEFT) d. Wall Collisions detected properly e. Hazards and Health pickups are in the game with consequences f. Hazards and Health pickups are generated properly g. UI is aesthetically pleasing h. End-of-game works as described in all scenarios	
Communication 1. Documentation a. Headers b. Variables c. Blocks d. Subprograms		**Thinking & Inquiry** 1. Decisions a. Naming b. Documentation Descriptions c. Data Types 2. Appropriate use of subprograms and arrays	

Level 1 (50 – 59%)	Level 2 (60 – 69%)	Level 3 (70 – 79%)	Level 4 (80 – 100%)
- shows little understanding of … - rarely adheres to …	- shows some understanding of … - mostly adheres to …	- shows understanding of … - adheres to …	- shows considerable understanding of … - completely adheres to … - goes above and beyond in showing …