University of Leicester
Computer Science Department
CO2001: Mini Project

Samy Narrainen, SN209

Introduction

HCI essentially establishes the ‘guidelines’ for producing a usable piece of software with the user experience in mind. It would have been extremely beneficial if I could have applied many of the concepts in HCI to my project, however due to the restriction of time, this was not possible, somewhat respective of the real world as a common quality tradeoff is based on time.

Nevertheless, I would like to talk about the aspects of HCI I would have used in a more ‘reasonable’ scenario.

Who are our Users and what do they want?

First of all, we would need know our users. This would involve specifying a target audience and considering their general information. For simplicity, I would like to target the game at 16 - 28 year olds who already have some gaming experience. Although I’m personally within this group, I clearly don’t represent it entirely. Therefore a focus group of the target audience will give an understanding of what they generally like and what they’re used to. For instance, in order to make the game more relatable, we could ask about their favourite movies and map what they see to our art style. Additionally, we would ask about their hardware and general computer competence. This will help to dictate how simplistic our game should be and how efficiently it should perform. For instance, if we see a significant portion of the target audience uses shortcuts, it would be imperative to include them in the game.

Secondly, to understand what the user defines as ‘usable’ I would have to determine their needs. A contextual inquiry may be useful for this. However, in what context would we examine the candidates? Perhaps examining them playing games they’re familiar with and discussing their experience. From this we could conclude the common concepts they enjoy/dislike about the games. Thus, we can apply the beneficial concepts to our game whilst avoiding the negative ones.

Of course, to refine the data we collect the candidate pool size would need to be considered. However, convincing people to participate in such an inquiry is some what difficult. Common techniques involve using incentives to persuade candidates to join. Even something simple like chocolate is enough to convince people to participate. So long as we obtain a pool size of at least 5 we will have reached the ‘magic 5’ rule and we could consider our data reliable.

Therefore, we can conclude from our focus group and contextual inquiry the real needs of the user from which we can establish requirements for the game which will be the foundations for our designs and prototypes. These documents will be based on actual data, not just our opinions so they hold a strong sense of validity.

Prototype Creation and Analysis

So far we've collected information about our users and have a substantial understanding of them. Because of this we can begin to develop prototypes. Of course, depending on the significance of the project it may be better suited to develop low fidelity prototypes such as simple drawings first as they're a lot easier to make. Once shown to potential users it would give a good glimpse into whether or not our ideas match their needs.

Whilst low fidelity prototypes are good at conveying simple ideas, we won't be able to actually evaluate the software we develop without actual working executable prototypes. Therefore during development certain criteria or goals can be reached then tested with the user. For instance adding core functionality like monsters to the game. We could evaluate our prototypes efficiently and reliably through the use of questionnaires, specifically we could opt to use the system usability scale questionnaire, one which has already been tried and tested, saving us time and ensuring valid qualitative data. Using an expensive technique to perform usability evaluation such as Heuristic Evaluation is better suited for a more final product rather than prototypes. Although think aloud could be performed by the developers to ensure the prototypes we show are respectable.

In order to have focus in our development, we should set up realistic usability goals based on the documentation we produced from our data collection. They could be simple such as "Users should be able to get used to the combat system within 5 minutes". This way whilst testing and showing our prototyped we would have some kind of guideline to reference to in order to determine whether or not our prototypes are successful.

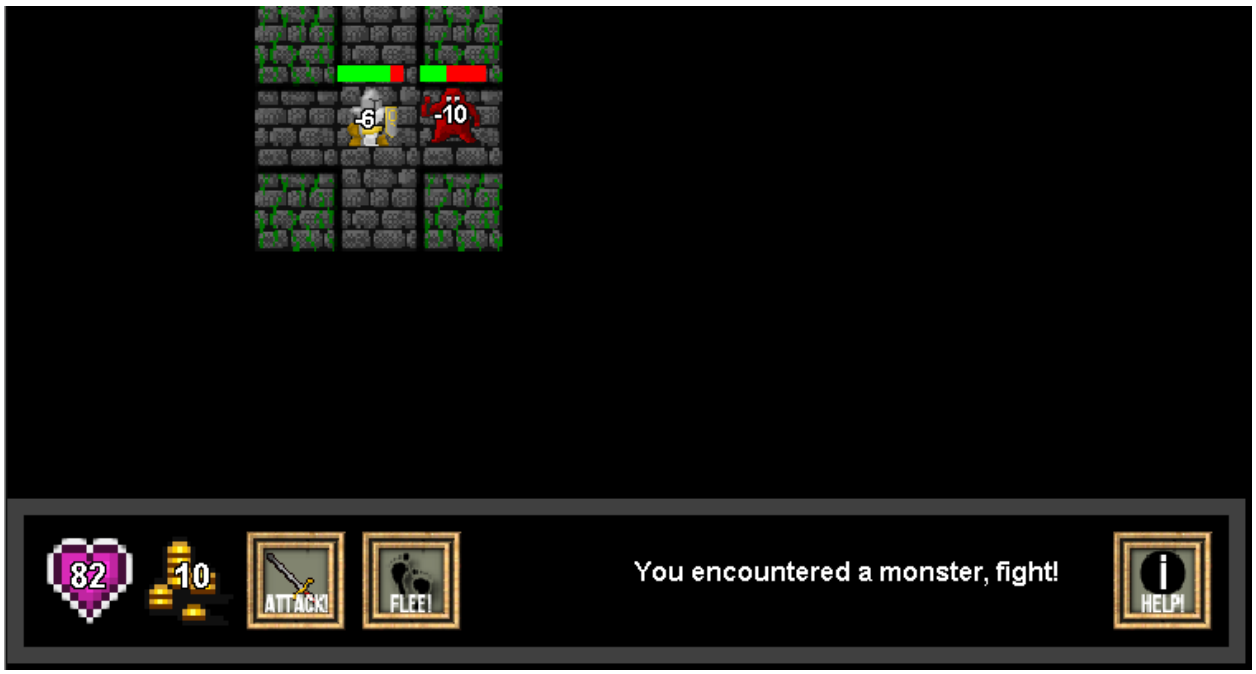
To conclude from our prototype evaluation we can begin to redesign the software to match the feedback we received. Doing this process regularly would ensure the software is always closely linked to the desires of the user. Thus through performing this iterative process we would eventually produce a working game that can be tested through more extensive means, specifically through a heuristic evaluation.

Heuristic Evaluation

This is where we can begin to transition to what I've actually done. Throughout the development of the game, I regularly referred back to peers/family/friends within the target audience to see what they think about what I've developed so far. I generally asked them questions based on the usability principles defined by Jakob Nielsen.

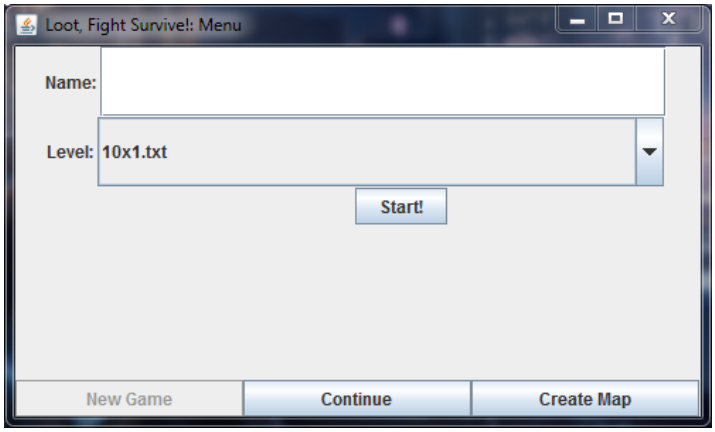
However, here I will begin to carry out a heuristic evaluation with the 'final' game in order to convey the features it contains to provide good usability and experience while reflecting on changes I have made throughout development or noting justification for changes.

1. Visibility of System Status



From this screenshot, we see that all the relevant information about the current state of the game is displayed to the user. Specifically we see:

- The player's health and gold (in the menu)
- The current situation that the player is in (in the menu)
- A visual representation of the player's and enemies' health (the health bars)
- Damage splashes which detail damage done (the numbers on the sprites)



Here we see after we've selected new game, the option becomes unavailable, indicating that it has already been selected. I included this change after testing with peers proved that there was confusion with how to start a game as in whether or not they selected 'Start!' or 'New Game'.

2. Match Between System and the Real World

You will see that the imagery and language used throughout the game is extremely relevant to what they're trying to represent in respect to what the user would be used to and for games in general. For instance, one may associate attacking with a sword, hence the choice of a sword for the image of the button. To ensure these buttons are understood, they are usually accompanied by text.

An example of language is 'continue' on the main menu. Previously this was labelled with 'load' however because of the functionality and how I purposely limited the user to one save the concept of continuing was more relevant.

Although the bomb is not explicitly stated to be a trap, it's completely intuitive that it is. For instance in games a bomb is commonly used as a trap and it's quite easy to associate bombs with being hurt.

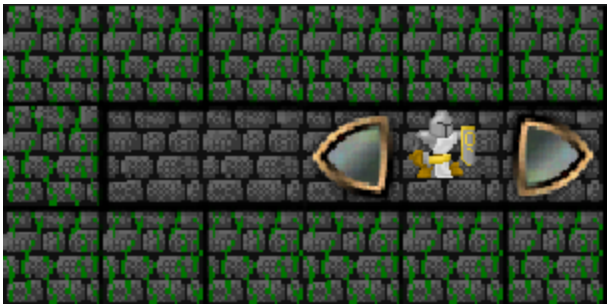
3. Navigation

The user has some lack of freedom in what they're able to do, however those limitations provide error prevention so I believe the user would prefer this. This is of course related to the buttons, for instance you only see the combat buttons whilst in combat.

4. Consistency

In terms of consistency, I've tried to relate my game as closely to other games of the same genre. I did some research to see how they were laid out and the type of user interfaces they used. This helped to ensure that my game could easily be picked up by people who like these kinds of games.

5. Error Prevention



Here's an example of error prevention. We see that the player is only shown the options to move left and right as these are the only valid moves for its position.

This is similar for attacking or fleeing. We observe that these action buttons only appear when the option to attack and flee is actually valid. Thus removing the possibility of an invalid attack/flee.

Additionally, because of the implementation of keybinds, these keys need to function similarly to the buttons. Therefore you will observe that if you try to attack outside of combat, nothing happens as it prevents you from attacking.

Of course, if the user's stuck, they always have the help button which details how to play the game.

6. Recognition

This applies to the help menu, here's a specific example. Instead of simply describing the action of moving, I accompany it with an image and the image displays the action in use so it's very easy to understand the functionality of the buttons and how they perform.

This also relates to principle 2 as the imagery and language used is clear so there's little for the user to actually remember based on its intuitivity.

7. Flexibility and Efficiency of Use

The player is able to interact with the game solely with the GUI. However, expert users such as game enthusiasts would potentially prefer to use shortcuts to interact with this game, so I made this option available and both are usable throughout the game and the shortcuts are detailed in the help menu, however they are all intuitive as they follow standards of the game industry. For instance, moving with directional keys is normal.

8. Aesthetic and Minimalist Design

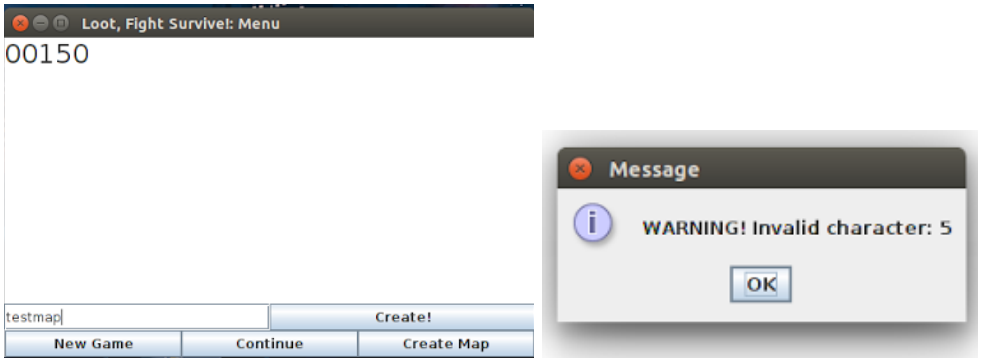
In order to ensure all text is visible in game, I have drawn them against a shadow. Additionally only information/actions that are relevant are visible at anytime. This helped to prevent errors and in turn created a simplistic design.

In terms of actual aesthetics, the response I got from peers was that the look of the game was generally good, so based on this I can conclude it is fairly aesthetically pleasing.

9. Recovery

Recovery doesn't directly apply to the game because of the level of error prevention. However it does apply to the creation of maps. For example, here's a situation where the user inputs 5

into the map creator. Because 5 isn't a legal room type it shouldn't work. Instead of just telling the user that the map isn't valid, I indicate why it isn't valid so they can easily correct it.



10. Help and Documentation

Although I feel that the game is fairly intuitive, through testing with peers I found that some people weren't able to understand the game and required external help from me. This lead to the development of the help window which loads of the help documentation image that describes how to control the game.

I kept the text clear and concise such that it's easily understood without ambiguities. This is important as help documentation shouldn't confuse the user further! Additionally, like mentioned in recognition, the text is accompanied by images so it's very clear what the text is related to.

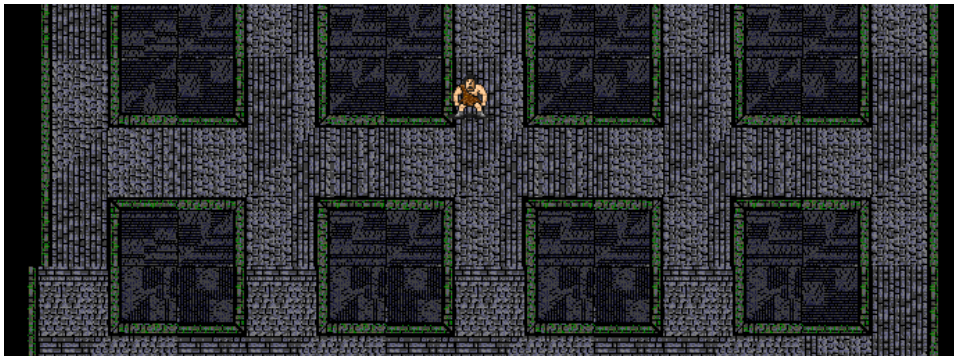
Concluding our Heuristic Evaluation

I've mostly highlighted the advantageous aspects of the system in this evaluation so perhaps it's not completely respective of a trained evaluator. However it does provide evidence that the system is usable and does not violate heuristics.

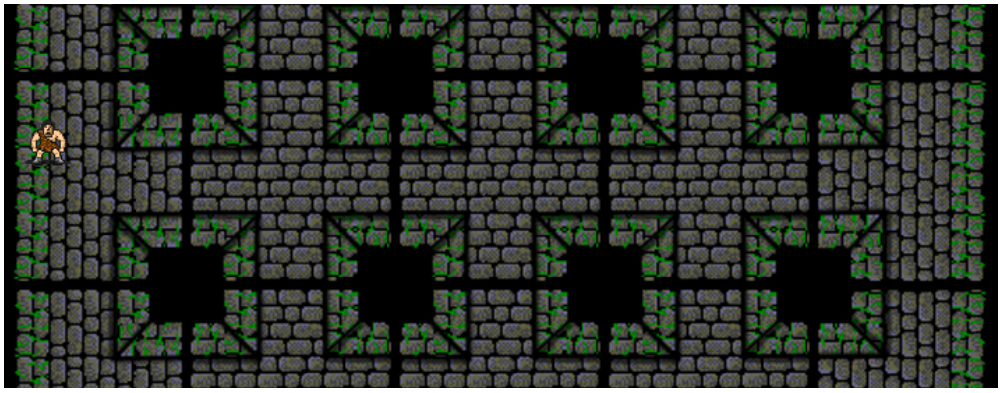
Although it would be interesting to have the opportunity to have a trained evaluator run a heuristic evaluation over the software so that they could propose fixes such that I can further improve my game.

Evaluate Design Artefacts

Throughout development, I regularly asked peers to test what was currently developed so they could provide feedback such that I can develop a more refined product. Because of this, the software went through various noticeable stages before reaching a final state.



For instance, here is an early prototype of the game. I received feedback the textures were jagged and movement was inconsistent which made it nauseating to play. I obviously couldn't release a piece of software with that kind of feedback so I used this feedback to perform iterative design.



We see another prototype which is closer to the final game, however still different. Again from feedback I received it was necessary to improve the designs so the game is clear and less frustrating.

Another example are the character sprites I transitioned through.

The initial sprite was a quick attempt by myself. It was clear from the feedback that the quality of sprite I could produce would not cut it and I looked online for more professional sprites. The feedback that I received on the second sprite was that a barbarian character may not be relevant for the maze. Thus we transition into the knight, a set of sprites that I could animate to provide a clear and enjoyable representation of the player that is both relatable to an adventurer and fun!

Interestingly, Nielsen suggests that 3 iterations are required before an optimum design and the majority of my transitions considered 3 iterations or more so I have some evidence to suggest that my designs are appropriate

Testing

Testing is imperative in the process as we can't provide a game that doesn't work or is faulty. There were many bugs throughout development and I tried my best to resolve them, especially the crucial ones.

For instance, if no save file existed and the player attempted to continue the game would completely crash. This was not user friendly so I implemented a feature to tell the player that no save file existed and the game menu would continue to run.

There are so many examples of bugs I've fixed so I won't list them all. However, there are still known bugs in the system but the restriction of time limits my ability to completely make the software perfect. Although the game is completely functional in its current state and the bugs that are present are minor or are the result of 'bad' users who intentionally want to break it.

Perhaps if I wanted to 'professionally' test the software I could perform checks against the usability problems. If any of the 6 criteria is flagged then the software clearly isn't perfect. Due to the simplicity of the game and error prevention however there is little that could go wrong so from an 'armchair' evaluation one might conclude that the severity rating of usability is at the very most minor.

Efficiency and Performance

Low frame rate and bad performance negatively impacts user experience. Depending on the severity it can completely render a game unplayable. I've tried to develop my game so that it performs efficiently and consistently throughout platforms. This is necessary because I included animation. For instance, my threads end themselves after they've performed their purpose and are only started when necessary.

Another satisfying example is that the game is capable of representing a crazy sized map without affecting performance of the actual gameplay. For instance, I tested a 200x1000 map which equates to 200,000 rooms. Because of how I've developed the game, only a 5x5 portion of the array is drawn at any one time so the game easily handled this insane number of rooms without affecting performance at all!

One unfortunate thing about the performance of the game is that on Linux the positioning of text is slightly different to windows. Although not directly related to performance it does concern portability and hardware/operating system considerations. Therefore in a perfect scenario I would have two versions of the game, one to run on Windows and one for Linux. However, the differences are very slight and do not affect the gameplay directly.

Final Thoughts

Thus that we conclude that because of the lack in time the options available to us were limited. However in a realistic scenario, say one in industry the application of HCI concepts and methods to ensure usability and user experience is imperative, hence why I would've like to apply them to my game.

Although from what I have been able to apply in the time available I believe I've developed a robust piece of software that takes the user into consideration throughout.