Evaluation of the User Experience

in Unwanted

David Rogers, in Group 14
Department of Computer and Systems Science, Stockholm University, Kista, Sweden
Daro6354@su.se

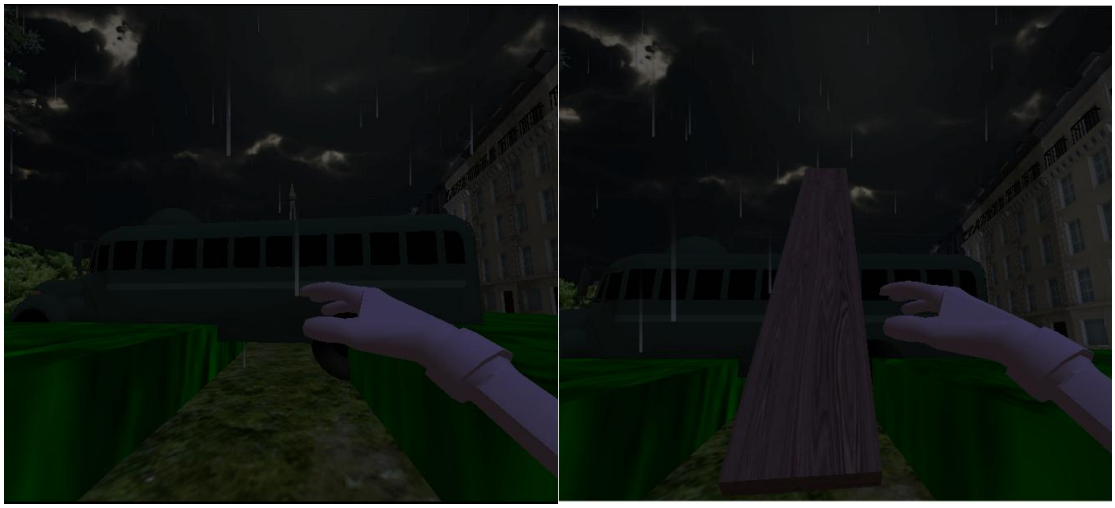
This game prototype is called Unwanted and it is a puzzle game with a lot of feel in it. It is set in a future world where the world is overgrown with vegetation and mutants are common. The player character is just such a mutant and the enemy is the "normal" person that just does not like you. The environment of this prototype is set in the overgrown and dark Champs de Mars in Paris. The game is made with the Unity game engine.
The method used to evaluate the user experience is a CEGE questionnaire. This is done electronically as 38 questions in the form of 7-point Likert scales. These relates to different core elements in the game. The responses are then categorized into different elements of the game experience, such as Enjoyment or Frustration. These categories are then used to evaluate positive and negative response for the responding elements in the game.
The result of the evaluation shows that the enjoyment of the game is lessened by the fact that the replay value is very low. Frustration also scored high and especially high when it comes to frustration while playing the game. These two together would suggest that there are some situations in the game that are more annoying than fun and the respondent would not like to go through these situations again. The best way to solve many of the problems, which were shown in the study, would be to add some sort of contextual help. This would be best done by having a story which can lead the player in the right direction.

About the game prototype

Unwanted is a slow-paced immersive puzzle game. It is slow-paced as the player character does not move very quickly and although there is some incentive to move quickly at the end it is in general better to take some time to advance through the game. It is immersive in the way that we, the developers, wanted to project a certain "feel" of the environment. The setting of the game world is taken from the Champ de Mars garden area in Paris, complete with the Eiffel Tower, with a distressed look. This makes the game world at least somewhat familiar to the player. The game is moody with darkness and rain. The puzzles are quite simple but figuring out the pieces can be difficult. The pieces to solve a puzzle are seemingly ordinary and naturally occurring objects. The puzzle in itself is something as simple as a hole in the ground.

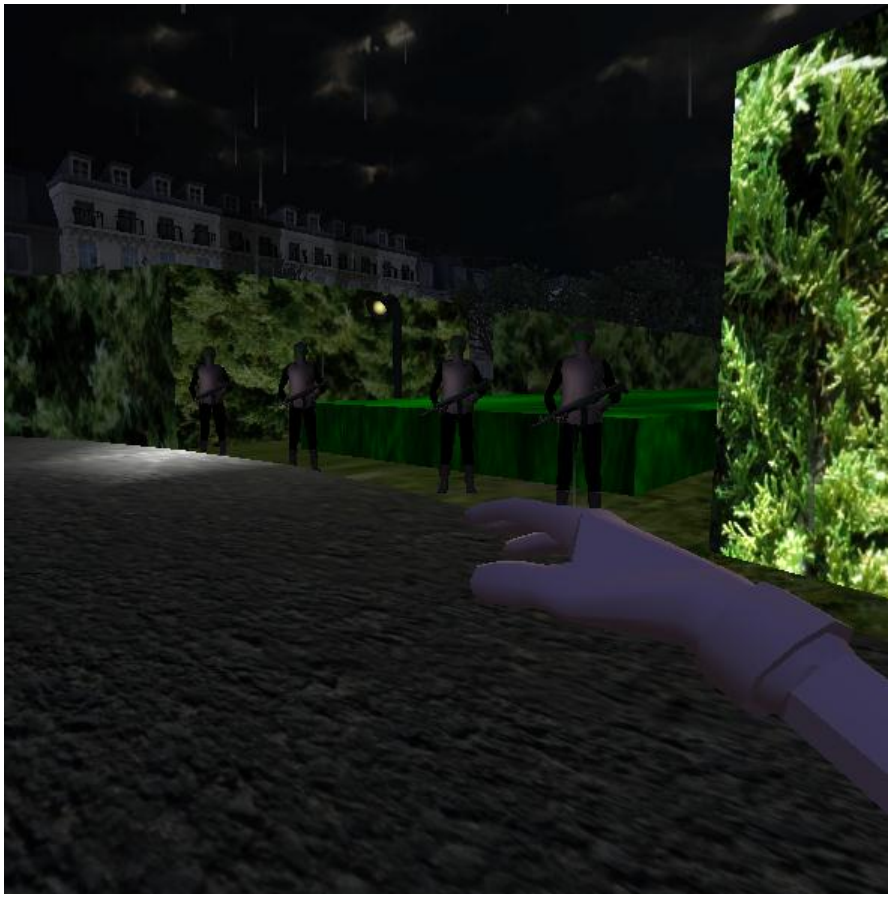
After testing Unity, UDK and Flash we decided to develop the game in Unity. This is because the developers as a whole had more experience with Unity but also because it is quite simple to make any type of game in Unity. In Flash it is quite difficult to make a 3D-game and in UDK it can be time-consuming to make a game that is not the game engines primary focus, a First Person Shooter game. In Flash it would have been quite easy to make a puzzle game but it would have been more difficult to incorporate the immersive feel into it. In UDK it would have been easier to get a very immersive feel, through good effects, lightning, etc., but creating the puzzles would have been more time consuming with its scripting

engine. We came to the conclusion that Unity could handle both these aspects fairly well and was then the best environment to develop the game in.



1. Screenshot of obstacle.

This first screenshot shows one of the puzzle elements of the game. You find the wooden plank in another part of the park where it has no seemingly use. When you then find this obstacle you have to backtrack to find the plank again.



2. Screenshot of bad guys. Slightly resized to fit.

Screenshot 2 is slightly resized to fit better into this report. The characters seen in the screenshot are the “bad guys”. You get told to avoid these and when you get

too close you get a warning sound in the form of a gun readying (cocking or reloading). If you still go closer you will be shot and re-spawned at the start of the map.

The CEGE elements in the game are not all very well developed. All elements that have to do with the *goal* of the game, for example, are not well defined as the goal of the game is simply to advance further in the game world. The *scenario* is not really described at all. You get a quick textual help to explain the *rules* of the game and the *rules* are also quite simple so they are easy to understand. You get the *controllers* explained in a simple fashion and they follow the usual WASD mapping so they should be quite easy to understand. Here the *small actions*, using objects to get past obstacles, are also hinted at. The reason they are not more precisely described is to give the player a stronger sense of accomplishment when they get past an obstacle, in part to facilitate *ownership* and give the player some sort of *reward*. The *memory* element does not play a significant role as the controls and rules of the game are so simple that you should not have to memorize anything important. Our main way to facilitate *ownership* is through the aesthetic values as they are there to deepen the immersion. As the game is built much like a maze with puzzles in it the *time* the player wishes to spend solving these puzzles, assuming that a maze is a type of puzzle. There is no "checkpoint" type of mechanic so a player may get frustrated if they fail at the later part of the game and have to replay the "easy" parts of the game several times.

Method

Design Science

Design Science is a way to bridge the gap between the rigidity of science with the cost-effectiveness with product development. It does this by actually incorporate the development of an artifact into a scientific process. This artifact can be any finished product. Another way of putting it is to use a theoretical evaluation to a practical development process. The artifact is developed in several iterations with a scientific evaluation at every iteration. The iterations follow the normal development iterations fairly closely, from defining the goal and demands of the artifact to testing the artifact. The evaluation is done more or less rigidly. Any normal data gathering methods can be used but, depending on the iteration, not all are optimal. It is this evaluation that can be repeated and reviewed by other people than the actual developers. It is also the results of this evaluation that is the most valuable part of this method. They can be used to help with future development to avoid making the same mistakes. After this method you are left with an artifact and more knowledge on how this artifact works and also how your development process works.

User experience evaluation methods

I will discuss the two methods most likely to be used, according to me, in this course. These two methods are the CEGE and GAP List. The GAP List can be used with two different methods: a heuristic evaluation or an empirical usability evaluation. The heuristic evaluation is done with an expert playing through the game and noting issues based on the GAP List. The empirical usability evaluation is done with a player playing the game and "thinking aloud", that's

when the player says things that he/she thinks of, while an expert is observing and taking notes. You later go through the notes and find issues based on the GAP List. These issues are also coded as negative or positive.
The CEGE method described in the literature is used to evaluate the game experience of a game. This is done by creating a number of questions that takes answers in the form of a 7-point Likert scale. These questions are then presented to the respondents in the form of a questionnaire. The questions are placed within categories. These categories are used when presenting the results. In this way you get a numerical result for each of the core elements of the game experience (CEGE).
It is quite easy to see that the GAP List methods are qualitative and the CEGE method is quantitative, although you could use either but it would make it more difficult. The GAP methods results in a number of issues, or problems, that are more or less precise. This means that you get a very good idea where the problems with the game are. The weaknesses with these methods differ a bit but one that is shared is that they take quite a lot of time. The heuristic evaluation also suffers from the fact that the "expert" in this case would also be one of the developers which would mean that there is a real danger that I would not be able to detach enough for this to become a meaningful evaluation. The empirical usability evaluation requires a safe setting for the respondent to be able to think aloud and would also require a few more respondents, since it is likely that some of them would not be able to give much meaningful data. The CEGE questionnaire is in some way the opposite of the GAP methods. It is quite fast to get several respondents and you can easily add more respondents without spending that more time on the evaluation. The results you get are quite simple and it is *very* easy to compare it to other studies with the same method, you can even add the results together and evaluate. It does give a more general result compared to the GAP methods though. With these results you can get a general result of where the problems in the game are. If these problems are all in the smaller details it is not certain you would actually get these results from the CEGE method.

Application of method

The heuristic evaluation with the GAP List I almost immediately discarded since I thought I was too close to the development of this game and therefor had too much insight into how the game worked. This would mean the results could be tainted by my pre-conceived ideas of the issues in the game. The empirical usability method I discarded since I would not have been able to provide a safe setting for the respondents to think aloud. The fact that this method would take quite a bit of time did matter as well.
Although the CEGE method described in the course literature is used to compare two different input types impact on the game experience I chose to use this method to evaluate a game without several independent variables. I did this because the results you get from this method would be very simple to compare to other studies made with the same game. In this way I could compare my results to other group member's studies or I could recreate this study after having made changes to the game and compare the results.

I did an electronic questionnaire via Google Survey. In part I did this to try and get more respondents, but I still only ended up with 3, but also to have the

respondents get away from the laboratory setting and play the game in a more natural setting. This, of course, comes with the possibility that things like hardware, other people, etc. had a negative influence on the study. The respondents were not chosen from DSV as I wanted them to be more "normal" players, as many of the students from DSV are quite often avid gamers. I got the respondents from another university in Sweden which means they would be quite similar in age, education, etc. The university does not provide computer science education, however, so the respondents should not have a deeper understanding of game development. I did copy the CEGE questionnaire in course literature exactly. At first I thought I would change the questions that were negatively worded but I decided against it to try to duplicate that method as closely as possible. To analyze the results I split the questions into the same categories that the authors did and then analyzed the results category for category with one difference; I split the categories in two, one for negatively worded questions (where a high score is a negative result) and one for positively worded questions (where a high score is a positive result). It is through the results of each category I then evaluated what problems the game suffered from. If the Standard Deviation (SD) was low I assumed that the results were fairly representative for this category and did not do a more in-debt analysis into this category. If the SD was high I assumed that the questions in this category then described more than one problem and did a more in-debt analysis.

Results of mini-study

I used the categories described in the course literature with one modification. I split the scales that had one or more negatively worded questions into two scales, one positive and one negative. The scale for frustration I also counted as negative. I then calculated the mean (M), standard deviation (SD) and average (A) for each scale and also for all the negative questions and for the positive questions. With so few respondents the mean is somewhat skewed which is why I also calculated the average. The results for the overall positive questions are quite average (M=5, SD=1,47 and A=4,37) and the negative questions are a bit lower (M=3, SD=1,43, A=3,33). Here you can see where the mean is skewed with the mean for positive is 5 and the mean for negative is 3 but the average is 4,37 for the former and 3,33 for the latter. The SD for the two is quite close to each other. The two extreme scales for positive questions are *Environment* (M=5, SD=1,32 and A=4,8) and *Enjoyment* (M=4, SD=1,22 and A=3,66). The highest score is actually *Control/Ownership* (M=5, SD=1 and A=5) but that only includes one question. For the negative questions the highest scoring scale is *Frustration* (M=4, SD=1,41 and A=4) and the lowest scoring is *Environment* (M=1, SD=0,57 and A=1,33). From these results you can quickly come to the conclusion that the strongest aspect of the game is the environment and the weakest is the enjoyment (if you assume enjoyment and frustration as two ends of the same scale). A more in-debt look into the weak points of the game we see that *Frustration* is based around two questions with one asking for frustration after having played the game (A=3) and frustration during the game (A=5). This could mean that there is some quite frustrating elements in the game that flows over to the end of the game as well or it could mean there are some building frustration building over time in the game that gets alleviated at the end of the game.

Overall the positive scales that are below average (4) are *Enjoyment* that and *Ownership* (M=4, SD=1,46and A=3,83). An in-debt look into *Enjoyment* shows us that it is based on three questions where two are at average score, "I enjoyed the game" (A=4) and "I liked the game" (A=4,66), and one at far below average score, "I would play the game again" (A=2,33). This would suggest that the respondents liked the game somewhat but the replay value is very, very low.
An overall look at the results would suggest that the environment and game-play elements are the strong points of the game. The weak points of the game would be enjoyment and ownership. The fact that enjoyment scores low seems to have been heavily influenced by the low replay value. Another interesting fact is that the lowest scoring positive question is "I knew what I was supposed to do" (A=1,66) which may have been a cause for the high frustration.

Recommendations for improvements

First recommendations that come to mind would be to somehow increase the replay value of the game and reduce the frustration while playing the game. These two may very well be tied to each other so if you manage to alleviate the frustration from the game the replay value may increase. The first thing I would change is to add more story to the game as it is a good way to explain what the player is supposed to do and still keep the sense of immersion. It is also a good way to hint at what to do to get through the more frustrating points of the game. It is also a good way to get the player to relate to the player character and then deepen immersion and maybe feel more responsible for the in-game actions, which is a question that scored low. Another problem the respondents had with the game was they did not understand the goal of the game. This is also quite easily solved with some sort of contextual help (a story would be appropriate). The bigger weaknesses in this game prototype seem to be that it is difficult for the player to understand what to do. As I have already mentioned, the only way to actually solve that problem is to help the player along. We would have to be careful to not give away too much though, since the game is driven by puzzles and a large part is solving them.