Cognitive Walkthrough Report

for Project FreeDroid's Level Editor

University of Oulu
Information Processing Science
Cognitive Walkthrough Report
Markku Väisänen
Sami Mylly
Kaisa Anttila
05.05.08

Abstract

We evaluated FreeDroid's level editor with cognitive walkthrough. We went through several tasks and found that there were several possible problems that a new user might counter. Many problems have to do with lack of feedback and information. At several points a first-time user might feel that they don't know what available options do or if executing some tasks is even possible. We have also written some suggestions to solve problems and even ideas for new features.

Contents

1. Introduction

This document gathers the key findings regarding FreeDroid's Level Editor project that the team evaluated. The chosen usability evaluation method was cognitive walkthrough, and this document works as a final report of the evaluation work.

Cognitive walkthrough is a usability inspection method that focuses on system's ease of use for the first-time user. In this method the inspectors first define the typical users and their assumed technical background. Then a series of tasks is identified: what this typical user would like to do with this system and what is the correct action sequence to complete these tasks. After this defining work is done, inspectors start going through the tasks asking themselves questions about how the user would percieve the situation. This way it's possible to figure out how well the system helps the user to know what their supposed to do and how they can do it.

In cognitive walkthrough of FreeDroid's Level Editor the following description for typical user was used: *”Typical user for this level editor is a person who is interested in computer gaming, at least to some extent. This person does not necessary own an extensive background knowledge on computing, hardware, or programming. Therefore, (s)he is somewhat unable to comprehend modern software development. However, the person is interested in FreeDroid and is familiar with the game, having basic understanding of the game mechanics and user interface. This understanding limits only to the modern version of the game, not the old one. Therefore, the person expects the level editor also to follow FreeDroid's controls and analogies, hoping it would not be too ”professional” for a ”newbie” to use.*

This might be the first time for the person to actually try a level editor, even though the person has played RPGs before. This time the user wants to try a new approach by creating something personal as content. (S)he might have absolutely no idea what kind of level to create, or what kind of editor comes with FreeDroid. The user is pleased, and partly surprised, that the game even has this feature, and therefore (s)he impulsively gives the editor a shot; after all it might be fun, and who knows what the person comes up with his/her crazy ideas.”

Several tasks were identified, although only some of them could be walked through. Whole list might still be useful in identifying what kind of tasks a new user is likely to attempt. Following tasks were identified:

- Create a new map.
- Save/load chosen map.
- Able to test the map in action, ie. jump back/forth.
- Modify the terrain.
- Create different elements.

- Implement NPCs and enemies.
- Implement equipments.
- Implement machinery, vehicles, etc.
- Create dialogs for NPCs.
- Create triggers for different event types.
- Possibility to edit walking paths.
- Create chain events.
- Use own graphics to create NPCs or dialogs.

Walkthroughs that we had time to do are summarized and explained in this document. We have attempted to explain our findings in clear words and only short summary of each task's walkthrough is given.

2. Load / Save

Saving and loading are important functions in almost all software programs. Level editors are not an exception. The ability to access the work done earlier is a major determinant of how one will favour the software.

This chapter evaluates loading and saving functions of FreeDroid's level editor.

2.1 Load Level

- **Task description:** User wants to load a level to the editor.
- **Righ execution:** Start the editor – everything else happens automatically
- **Possible problems:** The user might be a bit confused when they don't get to choose which level they want to load.

Loading a level should be painless process since these days almost every software supports saving and loading some kind of content. Users are also expecting to get their hands on the work later on by loading it again. Since graphical user interfaces have made it possible to load existing files into editors fast and efficiently, one expects similar quality all the time.

The solution for loading a level in FreeDroid is interesting. The default level (which also appears to be the only one?) is loaded automatically when the editor is started. If we assume there is only one level to be edited, the solution of automatic loading is convenient to user. But if we assume there could be a possibility for user to create 100% own content, the solution is a little awkward.

It might also be a good idea to provide better feedback to the user about what level is now being loaded. User is confused because (s)he is not aware of what to actually edit. Is the level loaded the one that (s)he just played? If it is, is it possible for the user to edit something else instead? How do the changes affect the game? As you can see, lots of question rises from the beginning. Eventhough some features don't exist (and are not even meant to) user spends time thinking whether it is possible or not. Lack of this information gives user a feeling of not being able to control things at the necessary level.

In a case of being able to produce a whole new level to the game, do consider a possibilty to follow industry standard saving/loading functions. You can use any Windows application as a reference of how to load/save in a neet way that the user expects things to work. Consider having a menu which has the options like ”load a level” and ”save a level”.

2.2 Save Level

- **Task description:** User wants to save a level after editing it.
- **Righ execution:** Several options: menu (that can be accessed in several different ways) and save-button. Depending on whether user choosed menu or save-button, also confirming and returning to editor is different.
- **Possible problems:** User might not be sure if they've done something irreversible or not. User might also be confused about the different options.

Saving level (and data generally) is one of the corner stones of modern software processing. An application is almost useless unless something can be saved as an output. So, saving is eqally as important as loading, and most of the statements above also apply to saving.

In the level editor, saving is done in sophisticated manner. The option is available directly from the editing view, or through the main menu. These options serve their purpose well, and it is good to see development that makes things easy to access. However, depending on the way the level is saved, the output differs. User expects the both ways to mean the same saving procedure. But when the output is different, is the actual saving different. Unified solution to output should be obvious in this phase, at least if the saving really is the same. If it is not, what is happening?

In addition, confirmation dialog is strongly encouraged to use. It should prevent the user from overwriting the level.

3. Editing Functions

One of the basic concepts of editors of all kind is the possibilty to edit objects, elements, etc. These features have been around for ages; starting from command line operating systems, followed by graphical user interfaces. Nevertheless, the meaning has remained almost the same. Copy/Cut/Paste provides a standard interface for editing the essential features in the program, such as ability to move objects, produce multiple instances, and make repeatable tasks faster.

This chapter evaluates object editing functions of FreeDroid's level editor.

- **Task description:** User wants to copy a selected object to clipboard and then paste the copied object to some other place.
- **Righ execution:** This feature does not excist.
- **Possible problems:** User will probably try different standard solutions and possibly get annoyed when they don't work.

3.1 Copy Selection

The meaning of copying selection is very straightforward, you just select a target and copy it by choosing the command in a way that is convenient. That is really it. As the feature is simple, it also crucial. Imagine a text editing software without the possibilty to copy selected part of the text.

Level editor of FreeDroid is certainly not a text editor but, nevertheless, an editor. Therefore, the user expects to find a similar interface that for example Windows provides. This is why also copy -feature is expected. However, the feature does not exist (yet) in the editor, and this is a one step back from the familiar editing world of our user's.

Implementing copy feature is suggested since it would be step to more familiar user interface, and there is use for this feature without a doubt. Users expect this feature to work by using key command (Ctrl+C naturally), selecting either from menu (right mouse button), or by using a specific button from toolbar.

3.2 Cut Selection

Cut and paste selection can be thought of being a same package with copy, either the features are implemented or not.

Just like copy (and paste also) cut is convenient way of moving objects from one place to another. It would be nice to be able to select multiple elements (like a whole building

with all its walls etc.) and move it by using cut and paste. Using an alternative way is almost un-imaginable. And this kind of feature is highly relevant.

Implementing cut feature is suggested since it would be step to more familiar user interface, and there is use for this feature without a doubt. Users expect this feature to work by using key command (Ctrl+X naturally), selecting either from menu (right mouse button), or by using a specific button from toolbar.

3.3 Paste Selection

As stated earlier, paste selection also belongs to the same 'editing family' familiar from editors of different kind.

Apart from copy/cut, pastes purpose is to implement something to a location user chooses. Once again, it is about providing a well standardized interface to handle a specific task that all editor-like programs have. Therefore, pasting a selection should be also made possible just like copy/cut.

Implementing paste feature is suggested since it would be step to more familiar user interface, and there is use for this feature without a doubt. Users expect this feature to work by using key command (Ctrl+V naturally), selecting either from menu (right mouse button), or by using a specific button from toolbar.

4. Terrain Editing

Terrain editing means here the formation of the landscape; how the different natural elements are edited. Terrain consists of various elements like water, soil, trees, hills, mountains, etc. These elements are to be edited easily and in a way that is efficient for a user to create functional and nice looking scenery.

4.1 Clear an Area

- **Task description:** Scroll to an area that user wants to clear, select the objects in that area and delete the objects.
- **Righ execution:** Use the arrow keys or mouse to move, select the objects individually by moving to them and finally press “X” or use the button to delete the object.
- **Possible problems:** Moving around is tiring and it's not clear when the user is moving between maps and when withing the map. Also user might be expecting to be able to select several objects at the same time and might get confused and frustrated when it's not possible.

Since the map is loaded by default, we found user wanting to start off by erasing instead of creating something. We reckon that this need to demolish comes from a need to actually create something own, and we also discovered that it is relatively difficult for the user to find a dedicated, already clear area to edit.

Clearing an area requires user to scroll around the map first in order to spot the potential area. However, scrolling turned out to be a confusing task; scrolling speed by using mouse varied, user confused moving between maps to scrolling one, and especially confusing are the buttons in the right down corner that resemble arrow-keys used to move around the map. Lack of mini-map (which GPS-coordinates can't cover) also makes it possible for a user to get lost when scrolling.

Clearing an area itself also proved to be a bit problematic. When deleting items, it is required to select them one at a time. Actual selection process itself turned out to be difficult; user spent time thinking whether using mouse is as accurate as keyboard commands when trying to hit the elements to be selected. After this problematic session, it is still required to delete elements one at a time. User found intuitive solution to be simply dragging ”a box” with a mouse, while pressing down thr left mouse button.

Despite the difficulties with selection and scrolling, the actual deleting mechanic worked fine. However, industry standardized controls didn't seem to apply in this case. We recommend using keys like 'delete' to delete.

5. Create different elements

- **Task description:** User wants to build an apartment consisting of four walls, floor and furniture.
- **Righ execution:** Build walls by adding element-blocks, selecting the wanted floor tile and filling the area with it, selecting the wanted door-element and adding it to an opening in the wall, and finally selecting the wanted furniture-element and adding it inside the apartment.
- **Possible problems:** Placing the objects at exactly the right place can be hard, especially if the user is trying to use mouse to do it. One particularly hard task is placing the objects near the walls. The paint bucket icon can be confusing at the floor making phase.

Creating different elements is an important part of the level editor as it's one of the key ingredients in making fresh material to the game. Thus adding elements should be really easy and motivating. User needs to feel that this is fun and that they can express their imagination with this level editor.

The very basics of adding elements in level editor is quite clear and simple. There's good pictures of the objects and they can be chosen by simply clicking on them. However, there was problems at several points of the task excecution. First of them was building a consistent wall. We assumed that the user would first try to use a mouse to control where the wall blocks are put and that proved to be very problematic. Also it got quite boring to add block after block. Maybe you could make it possible for the user to “draw” the wall like a line – just draw the area where they want the wall blocks to be. Another thing that could be good thing to check is how well the wall blocks fit together so that it's possible to make a good, consistent wall. Adding a door is very similar to adding a wall. One specific question rose to our minds – how do you open a locked door?

Editing floor tiles is also very basic function. This phase had less problems that building a wall, although we found something a bit confusing. There's a paint bucket tool in level editor which is familiar for users from other programs such as Paint. In those programs the paint bucket tool usually only fills the area inside borders. Level editor's tool sort of does this, but it stops at different kind of floor tile instead of walls and other bounding objects. This can be a bit confusing for the user – maybe the paint bucket tool could be edited so that it identifies walls as limits and not just different tiles.

Final touch to building an apartment with the level editor is adding furniture. Biggest problems at this point was that some furniture, if put too close to a wall, goes partly inside that wall. Finally, when the whole building is ready, we thought the user might want to move it as a whole. Apparently this is not possible.

6. Implement NPCs and enemies + create dialogs for NPCs

- **Task description:** User wants to add an NPC or an Enemy to the map and edit it’s movement and properties. Then user wants to make a NPC to talk.
- **Righ execution:** Not possible to execute.
- **Possible problems:** Users can get annoyed for not being able to do this task. Also users might get confused as they can add walking paths but no NPCs to walk on them.

NPCs are pretty essential for a good adventure. Thus many users might expect to be able to add them. Right now in FreeDroid's level editor it's possible to add a walking path but not possible to create a NPC to use the paths. This might be confusing.

When adding NPCs is implemented, it should be a trivial task to add one. Also editing it's properties should be relatively easy to do and fit into the workflow. For example, here the level editor could give user hints. There's different ways how the program can tell the user “you just added an NPC but you need to edit it's properties to make it interesting”. For example after adding an NPC the properties-window could pop up or there could be some kind of written message to remind the user.

Some NPCs need to be able to speak. Most important NPCs tend to be able to have dialogs with the player. Being able to create dialogs is also an important part of creating a new adventure and should thus be easy to do. Again the progarm could assist the user and remind them to create the dialog. Perhaps, when they create a certain kind of NPC, the properties-window could include the dialog editor. Or maybe again some kind of written message could assist the user. Also the dialog editor itself needs to be clear and simple.

A nice addition to creating dialogs would be ability to make player's choices matter. For example it could be possible for user to define that if the player chooses to insult NPC the NPC will break the discussion, but if the player says the right things, they can get a quest. This seems to be done in ready-made NPCs to some extent so users would be expecting to do the same thing themselves.

7. Implement equipments

- **Task description:** User wants to add equipment to the map and adjust the quantity of items added.
- **Righ execution:** First choosing the “Add item” -menu (or pressing “G”-key). Then clicking the item to be added that has quantity (Diet supplement for example) and adjusting the quality to wanted value. Finally choosing the wanted properties by clicking the arrow icons below the item-selection and clicking the wanted item to drop it to the floor.
- **Possible problems:** Some users might miss how to edit the properties as they need to be set before the item is added or even chosen. Also equipment descriptions can be confusing if the user hasn't seen the equipment in the game and thus knows what they do.

Implementing equipments might be just a nice bonus that the user wants to give to a player, or they might even be essential for finishing the adventure the user has planned. Anyway implementing equipments is probably something that many users will attempt to do.

Right now adding items is a bit confusing for a new user. Connection between objects and their properties might not be very intuitive to a new user. When trial-and-error method is required to learn how a system works, there needs to be a way to undo changes and preferably also change some part of work (for example change equipment's properties insetad of undoing it's creation).

Currently it seems that FreeDroid's level editor allows undoing only one action and doesn't allow deleting equipments or changing their properties after they've been created. This makes learning adding equipment frustrating and even frightening. Some things that make learning this task difficult are unclear property and equipment descriptions. This makes learning more frightening as the user might not know what they're doing and knows that they can't easily undo their mistakes.

One intuitive task would be placing equipment inside a container object. We couldn't find a way to do that although we believe that it would be something a user would try to do.

8. Implement a moving object

- **Task description:** User wants to add a moving object (for example an armoured car).
- **Righ execution:** Not possible to execute.
- **Possible problems:** No probable problems.

Moving objects are not implemented in FreeDroid, so the user probably wouldn't expect to find them, either, or at least wouldn't get very confused when they don't find a way to implement one. We've written about this task here because this is something that would fit very well in this type of game.

There is already an option to add some basic machinery such as guns or turrets. However these machines don't move. It would be a nice thing to be able to create a mobile object. These objects should have editable properties and moving types. For example there could be an armored car that the player can use to move around or a jeep for NPCs to use. These would bring a little life and more options to game.