How we defined our User Interface

Marvin Kampen
University of Bremen
Bremen, Germany
mkampen@uni-bremen.de

***Abstract*—In this paper is described how our group designed the user interface for the student project 'Travis'. We will explain our main design goals and discuss difficulties that we have encountered while we created the graphical user interface for the software.**

***Index Terms*—unreal engine, unreal, trajectory, travis, visualisation, gui, user interface**

I. Introduction

Travis is a masters project operated by *University of Bremen (UoB)* and *Marum*, which tries to re-simulate a deep dive mission in the Unreal Engine. Re-simulating in this case means that log data is provided by an Autonomous Underwater Vehicle (AUV) that needs to be read by a computer and displayed in a simulated environment. The output layer of this simulation might be a computer display or any technology involving Virtual and/or Augmented Reality.

II. Design Goal

We wanted to archieve a software that suits the seven human-computer interaction usability criterias as mentioned in ISO 9241.
The first criteria is **task suitability**. The software should fit the main tasks in a way that the software is not full of unused features. Furthermore the software should contain all the abilities that a user needs to resolve his tasks.
The second criteria is **self-descriptiveness**. All icons have tool-tips. The software displays the help or error messages or results to guide the user.
The third criteria is **controllability**. The user controlls the program.
The fourth criteria is **conforming to expectations**. That means that we should design our interface with the user expactations and knowledge in mind.
The fifth criteria is **fault tolerance**. Errors can happen and will happen. The software should anticipate common errors and should handle them well. Every action can be reversed.
Customizability is the sixth criteria. The user interface should be customizable for the user.
The last criteria is an easy learnability for the user. The user interface should explain itself and easy to remind.

III. Components

The interface was planned with a main menue, a map selection, a tour view and things like options and credits in mind. The reality changed our vision in certain aspects. It was not necessary to divide the main menue and map selection it two components. Furthermore was orignally planned to create a software that could plan AUV tours and get in-situ data, but this is a longterm vision. Our current goal was just to provide a tool that contains the possibily to display an AUV tour. So we keept our interface simple.
The only components that we now created are the software launcher and the tour view plus a page with the credits.

IV. Layout

This section will explain how we designed the main component of the interface: the tour view. First we created a mock-up (see Fig. 1) as a design guideline.

Our intention with the design is a small window hierarchy. The user should see all important information and interactions in one view. In addition we would like to display the game view (a window that displays the 3d environment and the auv) the whole time as the main visualization.
The interface layout is inspired by modern games where a mini-map is displayed in a corner of the field of view. Furthermore are the interactions buttons divided in different groups. One group is for icons that interact with the game view and trajectory. Another group is for the interaction with the warning section. The date and battery are also defined as a group which can not be interacted with. The timeslider activities are grouped. We used groups for a better learnability and for the encapsulating features into certain spots.
Then we recreated the mock-up in Unreal Engine 4 blueprints. The blueprints are a visual scripting language that contains a wide set of predefined user interface elements. The elements were connected via functions in the last step.

V. Rendering the Game View

One huge issue that we had was the creation of an individual game view. The Unreal Engine 4 is not capable to resize the game view by hand, so we created a work around with the usage of a render camera.
The render camera is coupled with a normal camera and then it renders the picture of the camera on a texture. The texture is transformed into a material which is mapped on an image within the auv interface blueprint. In case the user switched the camera perspective it is necessary to turn off the current render target. Another approach was the useage of a split screen functionality ,but we realized that we could not control the size of the game view properly when enabling the built in

split screen. Furthermore the useage of the split screen ended up that the performance of the game were quite bad, because of the additional screen that needs to be rendered. This is why we used our work around.

VI. Handling the Cameras

There are two cameras placed in the map. One camera is placed beyond the auv. The camera is a 360 degree rotatable camera with fixation on the auv.
The second camera is a free flying camera that could free roam in the map. The camera starts always behind the auv.
The camera switch is accomplished by a click on the camera button.

VII. The Mini-map Problem

With the mini-map we have encountered the issue that we have dynamic maps. No map is the same, so our mini-map must be dynamic, too. THe current solution is that we let a render camera render their view on the minimap texture. This approach has some shortcomings like it is not an effective way to do a mini-map, because of the additional renderer. Also the mini-map cannot really adjust to the sturcture of the map. Plus the mini-map could be designed more beautiful, if we know how the map would look like.

VIII. Conclusion

We did not archieved all goals that we defined prior the implementation of the user interface, but the result is still accepted by our customer. The advice from the ISO 9241 criterias helped, but we could not fulfill all the criterias.

IX. Future Contributions

Future versions of the TRAVIS software should consider the following features or additions to the user interface. A mini-map that do not rely on a render camera would improve the software performance and it could look way better than the current solution. The Marum said that it would be cool, if we they could create custom profile for scientific or for engineering purpose. The profile should show only the relevant aspect for their work.

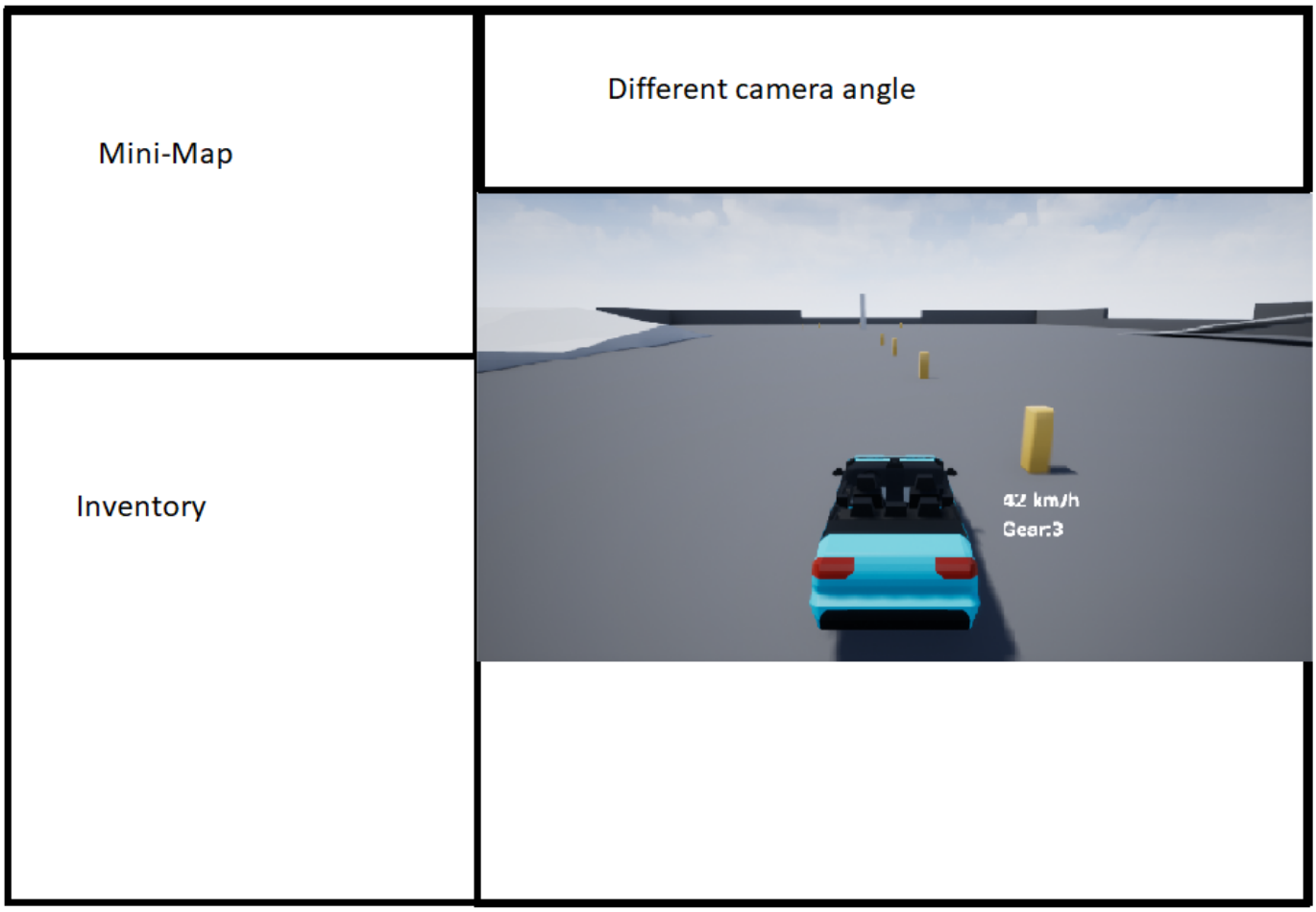

Fig. 1. Design Mock-up

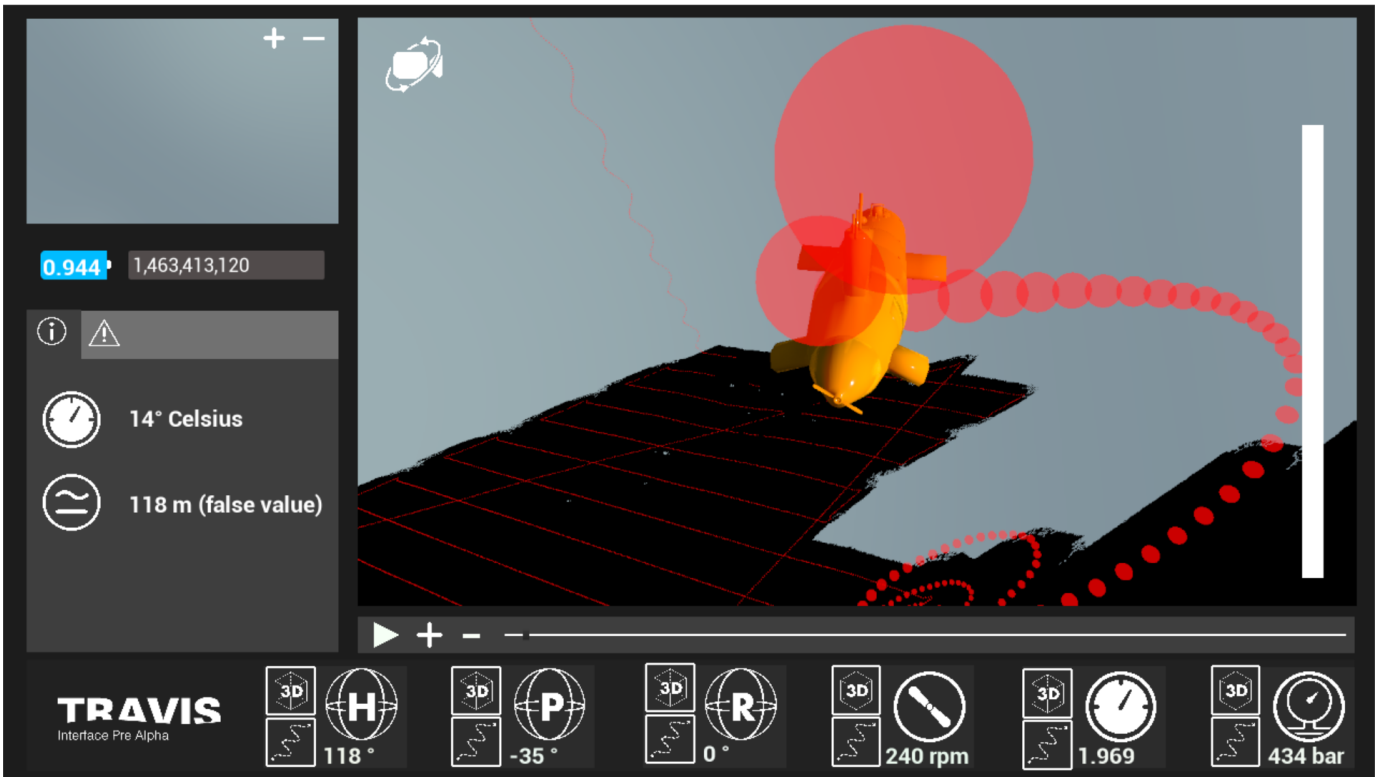

Fig. 2. Current screenshot from the final product