Author Tool Design

Serious Game Project at JMU University of Würzburg, by

Lena Lang

Jessica Topel

Lydia Bartels

Marlon Franz

Benedict Bihlmaier

Introduction

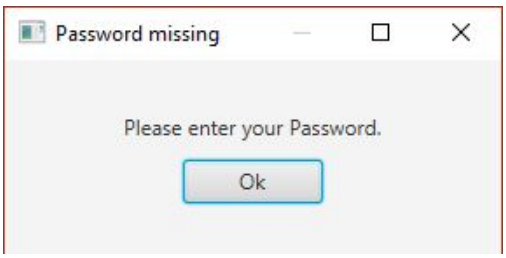
This document details the design decisions that were made based on best practices and heuristics, especially Nielsen's Heuristics, when creating the Author Tool of the TypeFighter game.

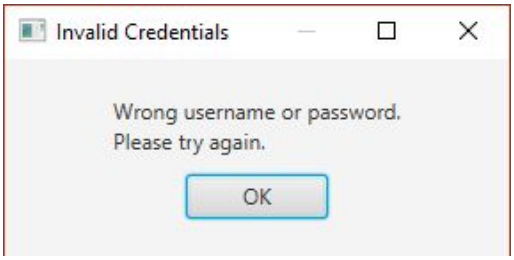
Implementation

Login Page

All information is displayed at the center of the screen to catch the user's attention, letting them know immediately what to do. A minimum of text is displayed, giving the user only the information they need to understand what to do, following the heuristic of minimalist design. The order in which this information is displayed follows a logical flow. First, a welcome message that also lets the user know what application they are using. Then the fields for their username and password followed by a button to confirm their login. To slightly accelerate the login process, it is also possible to hit the enter button instead of using the login button, which fulfills the flexibility and efficiency of use heuristic.

Checks were implemented to let the user know if they made a mistake and to help them fix the issue. For example, if the user forgot to input their password before hitting login, a message will be displayed that asks them to enter their password. Likewise, a message is being displayed when they have entered a wrong username or password, asking them to review their credentials and try again.





As the program needs a database connection to work, it will only open if such a connection is present. In case there is no connection, it will ask the user to make sure they are connected to a server of the University of Würzburg before the program gets closed.

Level Editor

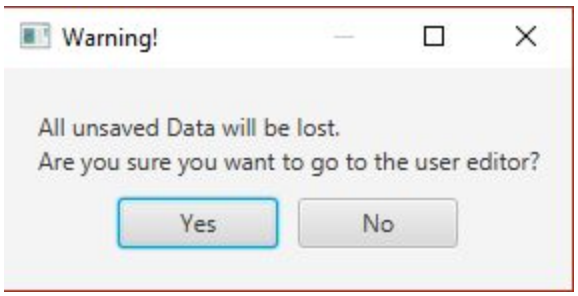
The Level Editor interface was designed to be minimalist as well. Most of the screen is used to display the text area where users can type in their text for the level, as it is the primary function of this page. Apart from that, there are only two other elements, the text field for the name of the

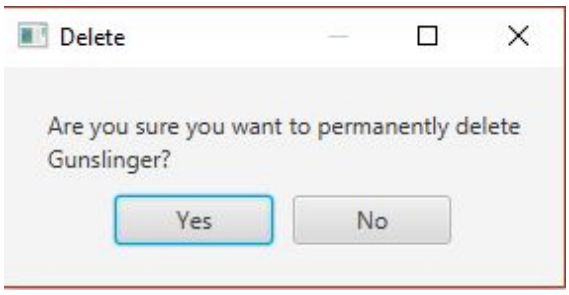
Level and the menu bar on top of the page.

The items in the menu bar are divided in two sections. One for the tools that can be used for editing the levels and one for navigation of the program, either allowing the user to log out or to go to the User Editor. The items follow a logical order, as in one must first create a new level before they can save it, which then can be loaded from the database and deleted afterwards. The logout button was placed in the top right corner as it is a common spot for this feature seen in many websites.

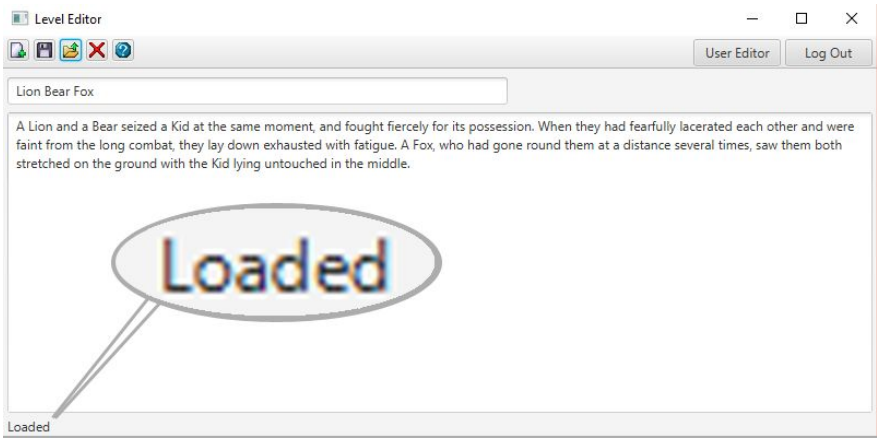
The icons used in the menu bar were based on commonly known ones, such as using a picture of a floppy disk to symbolize the save feature. Although these icons were designed to be recognized without further information, tooltips for all icons were added providing a short piece of information on their use. This also fulfills the principle of recognition rather than recall. Furthermore, a help button was added that describes the icons in more detail as well as the overall use of the Level Editor.

The heuristic of error prevention was considered for various scenarios. Users are being warned that all unsaved data will be lost if they switch to the User Editor. They must first confirm this before being redirected, preventing them from accidentally choosing this feature. Users are also being warned if they try to save a level with a name that is already present in the database and are being asked if they wish to overwrite it, preventing them from accidental data loss. The same is true for when they try to delete a level. To also support the heuristic of recognition rather than recall, the name of the level is being displayed in the dialogue box as well. Users are also prevented from saving a level without giving it a name first, as doing so would make it irretrievable from the database from within the Level Editor.



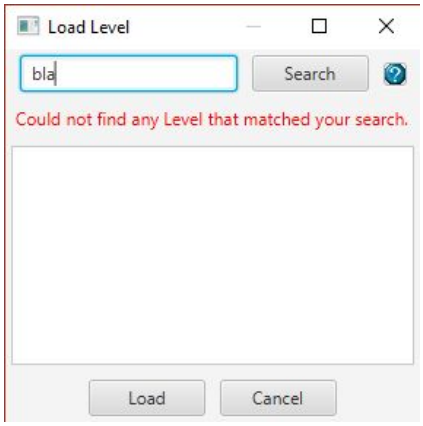


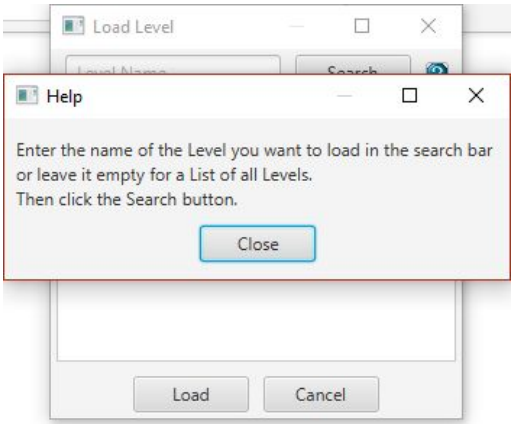
As visibility of the system's status is an important feature to consider as well, we implemented a method that displays a short status message in the bottom left of the screen for a few seconds.



Such a message will be displayed after the user created a new level, saved or overwrote one, and after a level was loaded or deleted. That way the user can be certain that the action they chose was actually performed.

Various principles were also considered when designing the level load box. Its design is minimalist as it only contains the currently needed information. Further information is provided when it is due, with the help of a help button that displays information on how to use this feature, as well as an error text that informs the user that no level was found if they input faulty data. In case a user accidentally chose the load function, it is possible to easily exit this window with the help of a cancel button.





User Editor

The design of the User Editor was based on that of the Level Editor, meaning that most principles and heuristics that apply to the Level Editor apply here as well.

The menu bar is almost identical and follows the same logical flow. The differences lie in there being no save button, as any changes to user data are being saved in the same window where were they were changed. The load button was not needed either, as the user search function takes care of retrieving user data from the database. Instead, an edit button was added.

The three tools present in the menu bar are also available in form of buttons to right of the user list. This was done to decrease the distance between the place the user clicked on to select a user from the list and choosing one of the tools to either edit or delete this user, allowing for a faster and more efficient workflow.

Checks were implemented to prevent the user from creating a user with an empty username or password.

Error Messages

All error messages are expressed in plain language and explain what caused the error sufficiently. Additionally, a simple solution is being suggested. This fulfills the heuristic of helping users recognize, diagnose, and recover from errors.