Grundlagen der Programmierung - Winter 2020/2021

Assignment 12 - Programming in the Tileset Universe

This assignment is the final one of this course. Use all your great new programming skills to create a game based on the provided tileset framework! The game should be based on the concept that you submitted for Assignment 9. It is probably helpful to read the details of assignment 9 again.

Deadline

March 7th 2021, 20:00.

Submission Details (Important)

Please combine all files into one ZIP file named *gdp_12_surnames.zip* and upload it in Stud.IP to the folder *Upload/Assignment 12*. The ZIP file should contain a folder that has the name *gdp_12_surnames*. Inside the folder, there should be one PDF file with the description of your game. We will correct this PDF file. In addition, we want to have your Processing program in the ZIP file so we can easily start your game. For this assignment, *your code does not need to be included in the PDF*!

Please do not forget to mention your names in the PDF file. Please also list the time (rough estimate in hours) you needed for the assignment.

Exercise

Your Game! (15 Points)

Program the game that you designed in Assignment 9 in Processing. Remember to add all graphics, sounds, etc. needed by the program into the appropriate folder inside the .zip file so we can directly start the game. Please pay attention to the following aspects:

- It has to be an Actual Game that can be Played

Please pay attention to required aspects of the game that have been described in Assignment 9. Your game must be startable and should not crash. Overall, it should make some sense as a game.

- It has to be a Tilset Game

Your game has to be a tilest game and has to make use of the provided `Map` class. However, you are free to choose the size and the content of the tiles. You are also free to extend the `Map` class. Please document any changes that you made.

- Classes for Game Character(s) and Other Objects

Program self-defined classes for the different types of game characters and other objects (at least two). Use an `ArrayList` to handle a multitude of game characters / objects of the same type. Of one of these types there must be several objects or characters with a number that varies over time, e.g. opponents that appear, move around for a while, and finally disappear.

Try to adhere to the "information hiding" principle, i.e. that the classes "know" as little as possible about the environment they are used in and that the main program "knows" as little as possible about the implementation of the classes.

- Window and File Size Constraints

The window size of your game should not exceed `1024 x 768`.

Please make sure that the included images and sound files are as small as possible (but as large as necessary) in terms of their file sizes as there is a maximum file size for a Stud.IP upload. Do not upload your game to any external hosing service, if it is too large. Instead, decrease the resolution of the included images or use a better compression for your sound files (MP3 instead of WAV, for instance).

Please note that it is not a requirement to have a game that makes use of images and sounds.

- Code Structure

Use global variables only for information that should be available globally in your program. If it is possible and reasonable, declare variables as local variables.

- Comments and Formatting

Please take care of a proper formatting of your program. Use the style of the example programs as well as Processing's *Auto Format* option.

The program must contain comments that explain what your code does. All functions and classes defined by you should have comments that explain what the function / class does. If anything has parameters or returns a value, there have to be additional comments that explain the parameters and/or return values. Furthermore, each global variable or class member also requires a comment.

- Game Structure

The game itself should display information about its controls. This could be right at the start or after the user has pressed something like a "help" key.

It should be possible to restart the game, i.e. to play again without ending and restarting the whole program. Please be careful to reset all relevant variables of your game, such as the score or any energy levels. Otherwise, things might go wrong.

It is not required (but also not forbidden) to have multiple levels.

- Documentation

Please submit - together with your game - a short documentation that describes your game. This can be similar to your draft at the beginning of Assignment 9. Please also add screenshots of all relevant screens of your game.

It would be great, if we could show your game other students! Thus, your documentation has to contain answers to the following questions: - Can we make your game (including the source code) available to the other students in *this* course? - Can we show the output of your game to the students of *this* course? - Can we show the output of your game in future courses? There are no points for "correct" answers, but you should answer these questions in your documentation.

- Respect Copyright

Do not use any images or sounds that have a license that does not allow their usage for your game or of which you do not know the license. For every external image or sound, please list the source and the license in the documentation.

Changelog of this Assignment

Version 1.0 - February 07, 2021

Initial document for winter 2020/2021.