Practical Software Engineering I.

Exercises for the 3rd assignments

Common requirements:

- Your program should be user friendly and easy to use. You have to make an object oriented implementation, but it is not necessary to use multilayer architectures (MV, MVC etc.).
- You have to use simple graphics for the game display. The "sprite" of the player's character should be able to moved with the well known WASD keyboard buttons. You can also implement mouse event handlers to other functions of the game.
- You can generate the game levels with an algorithm, or simply load them from files. If you load the levels from file, then each one should be put into different files, and you have to create at least 10 predefined levels. If you generate the levels, then take care that the levels should be playable (player should be able to solve it).
- Each game needs to have a timer, which counts the elapsed time since the start of the game level.
- The documentation should contain the description of the task, its analysis, the structure of the program (UML class diagram), a chapter for the implementation, which describes the most interesting algorithms (e.g. that generates the level) etc. of the program. Also, don't forget to show the connections between the events and their handlers.
- The task description should contain the minimal requirements. It is free to add new functionalities to the games.

Exercies:

1. **Yogi Bear**

   Yogi Bear wants to collect all the picnic baskets in the forest of the Yellowstone National Park. This park contains mountains and trees, that are obstacles for Yogi. Besided the obstacles, there are rangers, who make it harder for Yogi to collect the baskets. Rangers can move only horizontally or vertically in the park. If a ranger gets too close (one unit distance) to Yogi, then Yogi loses one life. (It is up to you to define the unit, but it should be at least that wide, as the sprite of Yogi.) If Yogi still has at least one life from the original three, then he spawns at the entrance of the park.

   During the adventures of Yogi, the game counts the number of picnic baskets, that Yogi collected. If all the baskets are collected, then load a new game level, or generate one. If Yogi loses all his lives, then show a popup messagebox, where the player can type his name and save it to the database. Create a menu item, which displays a highscore table of the players for the 10 best scores. Also, create a menu item which restarts the game.

2. Snake

We have a rattlesnake in a desert, and our snake is initially two units long (head and rattler). We have to collect with our snake the foods on the level, that appears randomly. Only one food piece is placed randomly at a time on the level (on a field, where there is no snake).
The snake starts off from the center of the level in a random direction. The player can control the movement of the snake's head with keyboard buttons. If the snake eats a food piece, then its length grow by one unit.

It makes the game harder that there are rocks in the desert. If the snake collides with a rock, then the game ends. We also lose the game, if the snake goes into itself, or into the boundary of the game level.
In these situations show a popup messagebox, where the player can type his name and save it together with the amount of food eaten to the database. Create a menu item, which displays a highscore table of the players for the 10 best scores. Also, create a menu item which restarts the game.

3. Labyrinth

Create the Labyrinth game, where objective of the player is to escape from this labyrinth. The player starts at the bottom left corner of the labyrinth. He has to get to the top right corner of the labyrinth as fast he can, avoiding a meeting with the evil dragon. The player can move only in four directions: left, right, up or down.

There are several escape paths in all labyrinths. The dragon starts off from a randomly chosen position, and moves randomly in the labyrinth so that it choose a direction and goes in that direction until it reaches a wall. Then it chooses randomly a different direction. If the dragon gets to a neighboring field of the player, then the player dies.
Because it is dark in the labyrinth, the player can see only the neighboring fields at a distance of 3 units. Record the number of how many labyrinths did the player solve, and if he loses his life, then save this number together with his name into the database. Create a menu item, which displays a highscore table of the players for the 10 best scores. Also, create a menu item which restarts the game.
Take care that the player and the dragon cannot start off on walls.

4. **Tron**

Create a game, with we can play the light-motorcycle battle (known from the Tron movie) in a top view. Two players play against each other with two motors, where each motor leaves a light trace behind of itself on the display. The motor goes in each seconds toward the direction, that the player has set recently. The first player can use the WASD keyboard buttons, while the second one can use the cursor buttons for steering.

A player loses if its motor goes to the boundary of the game level, or it goes to the light trace of the other player. Ask the name of the players before the game starts, and let them choose the color their light traces. Increase the counter of the winner by one in the database at the end of the game. If the player does not exit in the database yet, then insert a record for him. Create a menu item, which displays a highscore table of the players for the 10 best scores. Also, create a menu item which restarts the game.

5. **Custom game**

You can develop your own game idea.
The minimum criterie is that your game should satisfy the "Common requirement" and it has to have at least one database table, where the highscore of the players are saved. Moreover, it has to be able to read back the saved results from the database, and show them on the screen (rows can be filtered like e.g. top 10 scores).

The description and plan of your custom game should be accepted by your teacher at least one week before the assignment submission date!