Chapter 6

Final Assignment

In the final assignment you will make a Pacman game! The class diagram created in the previous exercise should be implemented.

This is a large and challenging exercise. We expect you to finish at least the first 4 challenges for a sufficient mark. Show your work after each day to one of the TA's.

6.1 GUI library

A simple GUI Pacman starterkit was written for this exercise. This starterkit provided on Canvas (`pacmanLib.zip`) uses the SDL2 starterkit https://www.libsdl.org/. Follow the `README.md` file to install SDL2 and to get the starterkit to compile and run.

The starterkit uses a 2D `std::vector` for the map (similar to the recursion exercise). The starterkit uses sprites from a sprite sheet. A sprite is a two-dimensional image that is integrated into a larger scene. The starterkit consists of three folders:

`resources`	contains the sprite sheet.
`src`	contains the C++ source + skeleton main file.
`include`	contains C++ header files.

The source folder contains a skeleton main file you can use to create the Pacman game. The main file contains a loop that runs constantly to render new frames to the screen. Input events are also handled in this loop in a case structure. Setting the score and the amount of lives left is also done in this loop by two the two functions `setLives` and `setScore`. These functions change the values shown on the screen.

The UI uses a `std::vector` containing the type `GameObjectStruct`. This vector contains all the objects that are rendered to the screen. The type `GameObjectStruct` contains the position and type and direction of the objects. The possible different types are listed in the `GameObjectStruct` header file. The different directions are also listed in the file as a `enum` type.

The given main file also starts a timer with a *callback* function. The callback function is `gameUpdate(Uint32, void *)`. This callback function is called every 100ms. It can be used to realise game mechanics like the movement speed of Pacman.

6.2 Challenge 1: Creating the board

In this challenge the board will be created for the Pacman game. In the previous exercise a class diagram of the game Pacman is made. Create a new project for the Pacman game and create the class structure you have determined before. You don't need to implement these

classes yet. The only classes that will be implemented in this challenge are the classes which are responsible for creating the game board and initialising the game. Use the provided GUI library to display the game board on the screen with the walls. Also the amount of points and lives must be visible. The layout of the board is also provided in the file 'board.h'. In the provided 'main.cpp' file there is a function where you can give the layout of the map. This function reads all the 1's for a wall and creates an image of the board.

Requirements

- Create a new C++ project for the Pacman game.
- Create the class structure in the C++ project. (Use the class wizard of eclipse)
- Implement the classes which are responsible for creating the Pacman game board and initialising the game.
- The walls must be visible on the board.
- The score must be visible on the board
- The amount of lives must be visible on the board

6.3 Challenge 2: Creating Pacman

In this challenge Pacman will be included in the game. Pacman should be able to move in response to the users inputs. When Pacman has a moving direction, it should keep moving in that direction over time.

Requirements

- Include Pacman on the game board.
- Implement the movements of Pacman.
- Implement collisions with the wall.
- The sprite used should point in the correct direction.
- The portal function of the map should also work: When Pacman exits the map on the left side it should re-appear on the right side of the map and vice versa.

6.4 Challenge 3: Creating dots

In this challenge the dots will be included in the game. The dots represent the 'food' of Pacman.

Requirements

- Place dots on the board.
- Implement collisions with Pacman.
- Dot should be removed after collision and points should be increased.

6.5 Challenge 4: Creating Ghosts

In this challenge the ghosts will be included in the game.

Requirements

- Initialise the game with 4 ghosts in the center of the map.
- There should be 4 different ghosts (Inky, Pinky, Blinky and Clyde).
- Implement some kind of random movement for the ghosts. Ghosts should also not be able to move through walls. They may behave in the same way.

6.6 Challenge 5: Ghosts

Implement the collision behaviour of the ghosts.

Requirements

- Implement collisions between the ghosts and Pacman. On collision the position of Pacman and the ghosts should be reset to the start position. Also, the amount of lives should be decremented.

6.7 Challenge 6: Creating energizers

In this challenge the energizers will be included in the game. When Pacman eats an energiser, the ghosts become scared. Pacman is now able to eat the ghosts. When a timeout expires the ghosts stop being scared.

Requirements

- Place 4 energisers in each corner of the board.
- Implement collisions between Pacman and the energiser. On collision the ghosts should become scared, edible and a timer should be started. When this timout expires the ghosts should be normal again.
- Implement collision between Pacman and a scared ghosts. On collision the ghost should reset to the start position and not be scared anymore. The amount of points should also be incremented when a ghost is eaten.

6.8 Challenge 6: Creating fruit

In this challenge the fruit will be included in the game. When Pacman eats the fruit, the points are increased by a certain amount

Requirements

- Place fruit randomly on the board after a certain amount of points gathered.
- Implement te collision between Pacman and the fruit. Increase the points on collision.

6.9 Submission

After you are finished or the time is up, show your work to one of the TAs.

Furthermore, hand in your workspace with the projects above in a zip called [Group number]-3ECAssignment6.zip (eg. 01-3ECAssignment6.zip).

Also hand in a separate report in pdf with explanation of the choices you made in this project. Also hand in the answer to the following question:

1. Does the structure of classes of your implementation of the PacMan game still represent the class diagram you handed in previously? Draw a class diagram of your final implementation. Indicate the changes and motivate why you have changed it.