23.10.2024

Turtle Graphics

Project plan

Mathias Castrén mathias.castren@aalto.fi
Hugo Tamm hugo.tamm@aalto.fi
Xiwei Zhao xiwei.zhao@aalto.fi
Giang Le giang.1.le@aalto.fi

I. Scope of the work

-

Features of the program:

- Turtle visualization with proper location and angle on a 2D field
- Basic commands: forward, turn, (pen) down, (pen) up
- Simple command line interface for entering the commands
- Different pen attributes such as color, line width
- Ability to save an image and turtle state, and later load it
- Ability to read command scripts for execution from file

Additional Features:

- (Using QtMultimedia for sounds)
- (QtQuick3D for 3D effects in the UI.)
- (Enhancing the drawing field with obstacles that block turtle movement)

How does the program work?

The program is built using the Qt library, which simplifies the creation of GUI windows and populating them with widgets such as input fields and labels. The program begins in the main.cpp file, where the main window is instantiated. This main window contains an input field and a history box, which displays all commands issued to the turtle, as well as any errors or tooltips. Additionally, the main window hosts the “map” where the turtle graphics are displayed (see figure 1 below).

Figure 1: The proposed design of the program

The user interface consists of two key files: mainWindow.cpp and inputController.cpp. These handle input validation, command history, and layout management. The actual turtle graphics are rendered in map.cpp, which manages location handling, grid formation, and line drawing. Turtle movements—such as “Forward”, “Turn”, “Pen up”, and “Pen down” are processed in turtle.cpp.

Saving and loading the turtle’s state is handled in storage.cpp, which uses the command history to track actions. Commands are saved in an ordered array of strings, allowing easy display of past commands and enabling quick restoration of previous states by replaying the command array. This array can be stored in a .txt file, which also allows external editing.

How is the program used?

1. Summarization:

The program allows users to control the turtle by entering executable commands. Users provide specific instructions to adjust turtle’s movements and actions accordingly.

2. User Interface of the application:

The user interface of the application consists of two primary windows:

- Main window is the large display are where the movements of the turtle are shown. It visually represents how user commands are executed, displaying how the turtle moves according to given instructions and illustrating the path created.
- Command window is the smaller window that receives the command lines from users. It also shows a history of previous commands, allows users to monitor their inputs and modify or re-execute commands as necessary. User also can use “help” command to get support and read the instructions how to use the application efficiently.

3. Interaction between user and application

Users interact with the application by entering commands into the command window. The movements and actions of the turtle within the main window are controlled by these commands. The step by step process is as follows:

- Input command: Users enter the line of instruction into the command window.
- Command Processing: The application reads the input line, queues them for execution. The input controller makes ensuring that commands follow the right logic and syntax.
- Execution: The turtle executes the queued commands in the correct order and the main windows dynamically displays the movements of the turtle.
- History and Support: Users can view the command history to track their previous inputs and make adjustments to the turtle's path, access support if needed.

II. The high-level structure of the software

-

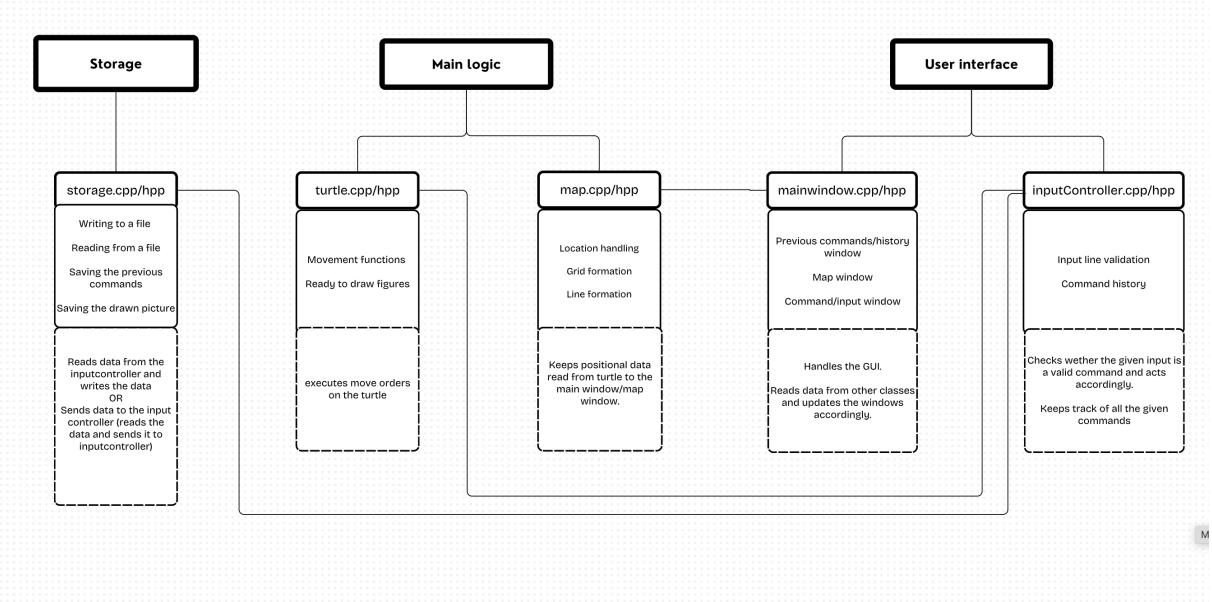

Figure 2: The basic structure and interaction between the classes.
The storage, main logic and user interface blocks represent modules.

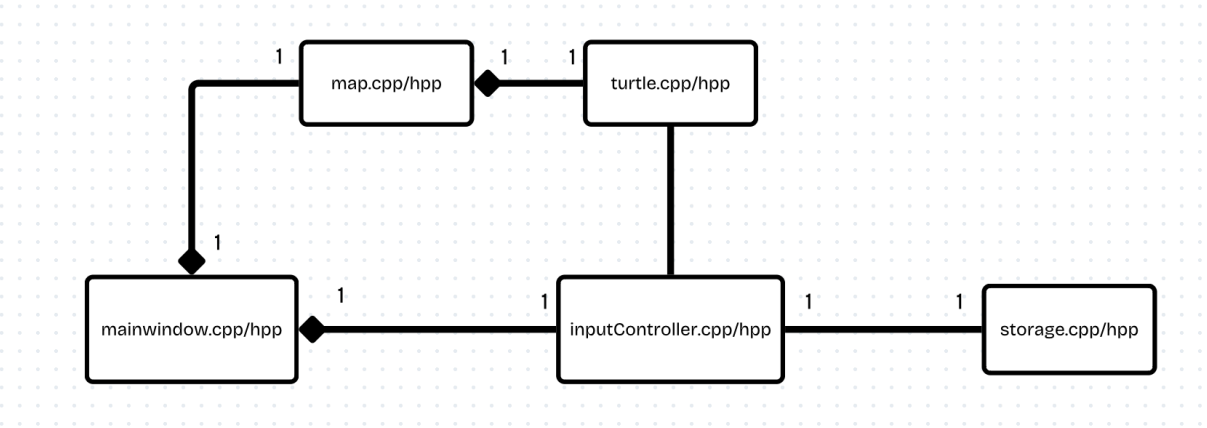

Figure 3: UML class chart (according to current understanding.)

III. The planned use of external libraries

-

1. Basic Qt tutorials

- Introduction to Qt Quick
- Introduction to QML
- Building with Cmake: Getting Started with CMake and Qt
- QML Integration Basics
- QML Modules
- How to Expose C++ to QML
- Introduction to Qt Quick 3D

2. GitHub example templates and tutorials

- ProgrammingPrinciplesAndPracticeUsingQt
- C++ Qt Game Tutorial 7 - Adding Sound Effects/Music

IV. More detailed plans for the sprints

-

Our regular long meeting happens every Thursday afternoon. Here is the table denoting our phase goal and schedule:

Week	**Goal**	**Tasks Details**
Week 01 (Oct.23 ~ Oct.31)	Complete Project Plan	1. Installed the Qt; 2. Software architecture completed; 3. Project plan completed.

Week	Goal	Tasks Details
Week 02 (Nov.01 ~ Nov.07)	Complete basic functions	1. User interface implemented; 2. Make the turtle move using the keyboard;
Week 03 (Nov.08 ~ Nov.14)	Optimize basic functions	1. Present the turtle trajectory; 2. Start and end control of the turtle; 3. Polish user interface.
Week 04 (Nov.15 ~ Nov.21)	Complete advanced functions	1. Change the colour and line features; 2. Detect if the turtle has reached the boundary and send warnings; 3. Add sounds and a 3D turtle.
Week 05 (Nov.22 ~ Nov.28)	Optimize advanced functions	1. Add some fixed drawing features; 2. Optimize storage functions.
Week 06 (Nov.29 ~ Dec.05)	Submit demo to advisor	1. Meeting with advisor; 2. Polish project.
Week 07 (Dec.06 ~ Dec.12)	Project final commit to git	1. Make the final submission.