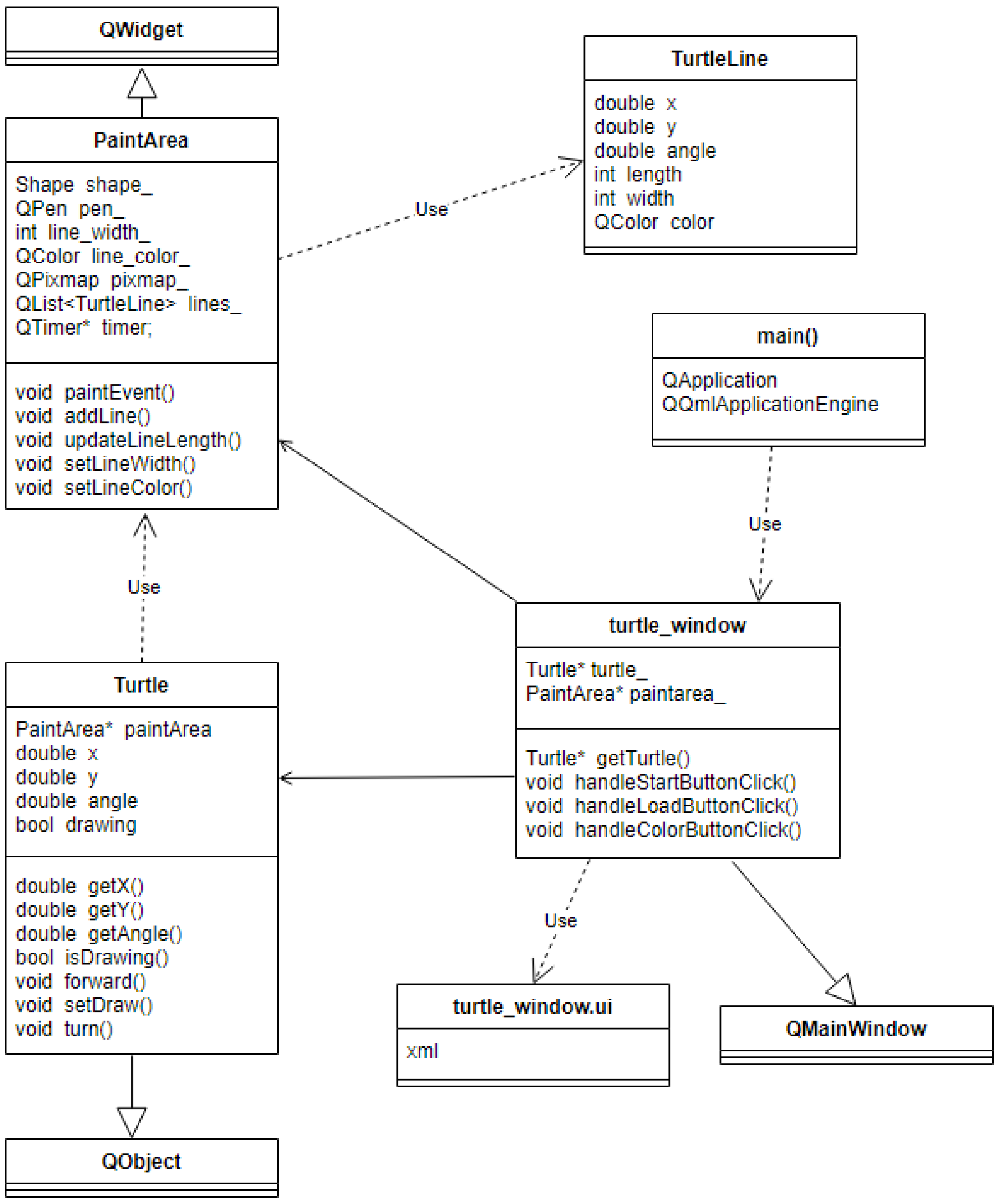
Overview

A turtle graphics program that is controlled through Qt widgets like sliders and buttons, and through the command line interface (CLI). The turtle operates on a 2D field where collision with the boundary plays a sound. To control the turtle there are three basic commands for the CLI: forward(distance), turn(degrees), and setDraw (true/false). These commands can be loaded from a JSON file into the CLI. Additional implemented features include the ability to change the width and color of the lines drawn by the turtle, as well as clearing the drawing area. Features that were not implemented include: saving and loading image, obstacles aside from boundary, and 3D effects.

The final program does not contain all the features that were initially planned. Some features were changed, some were implemented differently, and others were not implemented at all. However, there was an attempt to implement the program in such a way that further feature additions would be simple and unlikely to break the program.

Software structure



turtle_window

- Owns and manages Turtle.
- Contains PaintArea
- Contains all widgets
- Handles command line interface

Turtle

- Handles movement
- Handles turning

- Handles penUp and penDown

PaintArea

- Handles storing and rendering lines (TurtleLine struct)
- Renders turtle graphics
- Stores current line width and color

External libraries

- Multiple Qt objects
- cmath, fstream, algorithm

External code

- nlohmann JSON parser code integrated into project

Instructions

How to compile:

The application depends on the following Qt libraries:

- Qt 6.7.1
    - libQt6Widgets.so.6
    - libQt6Multimedia.so.6
    - libQt6Qml.so.6
    - libQt6Gui.so.6
    - libQt6Core.so.6
    - libQt6Network.so.6
    - libQt6DBus.so.6

And also on the following system libraries:

- OpenGL:
    - libGL.so.1
    - libOpenGL.so.0
    - libGLX.so.0

- Audio:
    - libpulse.so.0
    - libsndfile.so.1
    - libFLAC.so.8
    - libvorbis.so.0
    - libopus.so.0
- Fonts and Images:
    - libfreetype.so.6
    - libfontconfig.so.1
    - libpng16.so.16
- Compression:
    - libz.so.1
    - liblzma.so.5
    - libbrotlidec.so.1
    - libzstd.so.1

1) Install the required dependencies using your package manager.
2) Clone the git repository.
3) Configure the build environment, using *qmake* in your terminal.
4) Compile the program: run *make* in your terminal to build the project.
5) To run it, after compilation, find the executable in the build directory.

User guide:

When the turtle graphics program is running and the user interface window is open, you can:

- Read the list of available commands → In the text box at the top right of the window.
- Write commands to move the turtle → write individual commands in the box at the bottom of the window, separated by a space character between each one.
- Change the width of the drawing line → Use the *Width Slider* at the bottom right of the window (left is thinner, right is thicker).
- Change the color of the drawing line → Click the *Color Picker* button. A menu appears to choose the color of your choice.
- Make the turtle move without drawing → Press *Toggle Drawing* button off.

- Make the turtle draw → Press the *Start* button to start drawing according to the commands you have written.
- Erase the drawing and reset the turtle position → Press the *Reset* button.
- Import files from your computer to command box→ Press the *Load* button. Choose the JSON file of your choosing.
- Exit program → Close the window.

Testing

- Visual testing from GUI
    - Movement
    - Color
    - Boundary detection
- Test files in project document
    - Draw square correctly
    - Commands
    - Boundary decetion
- Lint to test code style and best practices
    - No issues
- Program stability and memory with valgrind
    - No problem when run from qt creator
    - From command line some problem but doesn't seem to be related to turtle

Work log

The software development process was divided into one-week sprints. Each week began with a Monday meeting where participants reviewed the activities completed last week and the tasks for the upcoming week were distributed. Martin was in charge of sprint documentation and spent 1 hour a week on them. The hourly contributions are only rough estimates and don't account for all thinking, planning and bug fixing.

Sprint 1   28.10 - 3.11.2024

Planned and completed actions:

- Set up git
- Project plan
- Plan feature importance - everyone

Condibutions and time used:

- Nicke: plan - scope of work 5 h
- Martin: set up git 3 h, inititating project 2 h (project meeting iteration, location, memo etc.), initiating project in Qt creator 3h.
- Leevi: plan – sprint schedule 4h
- Andrei: plan - high level structure 5h

Sprint 2   4.11 – 10.11.2024

Planned and completed actions:

- Implement raw class structure
- Hard coded commands
- Get project to run on every system - everyone
- Plan first version of UI

Condibutions and time used:

- Nicke: UI plan 47h
- Martin: hard coded commands and class structure 10 h
- Leevi: holiday 0 h
- Andrei: class structure 8 h

Sprint 3   11.11 - 17.11.2024

Planned and completed actions:

- Add CLI
- Integrate mainwindow
- Fix turtle agent
- Combine UI, mainwindow and other components to one structure

Planned but not completed actions:

- Implement UI
- Animated turtle movement

Condibutions and time used:

- Nicke: implement UI 7 h
- Martin: CLI 5 h, setting boundaries for agent 2 h, project admin 3 h (meeting memos, issue board update)
- Leevi: plan turtle movement 6 h
- Andrei: combine code 9 h

Sprint 4 18.11 – 24.11.2024

Planned and completed actions:

- Add drawing options
- Add turtle picture
- Execute commands from file

Planned but not completed actions:

- Refactor application

Condibutions and time used:

- Nicke: turtle picture 9 h
- Martin: commands from file 10 h, project admin 3 h (meeting memos, issue board update)
- Leevi: refactor application 10 h
- Andrei: drawing options 8 h

Sprint 5 25.11 – 1.12.2024

Planned and completed actions:

- Fix command line logic
- Add animated turtle movement
- Finalize GUI

- Bug fizes
- Clean code

Condibutions and time used:

- Nicke: finalize GUI 6 h
- Martin: Bug fixes and clening 3 h, initiating configuring of tests 2 h, project admin 3 h (meeting memos, issue board update)
- Leevi: fix command line logic and animated turtle movement 9 h
- Andrei: bug fixes and cleaning 6 h

Sprint 6 2.12 – 8.12.2024

Planned and completed actions:

- Implement final features before demo
- Sound
- Boundary check
- UI cleanup
- Reset button
- tests

Planned but not completed actions:

- 3D graphics

Condibutions and time used:

- Nicke: UI cleanup 6 h
- Martin: tests 10 h, project admin 3 h (meeting memos, issue board update)
- Leevi: sound and boundary check 8 h
- Andrei: reset button and code cleaning 6 h

Sprint 7 9.12 – 13.12.2024

Planned and completed actions:

- Project document
- Lint
- Final valgrind checks

- Clean code – everyone
- Demo – everyone

Condibutions and time used:

- Nicke: Document – instructions 7 h
- Martin: Lint 4 h and valgrind 4 h
- Leevi: Document – work log 7 h
- Andrei: Document – software structure 8 h