Documentation: Other-1 (smash)

ELEC-A7151 - Object oriented programming with C++ F2020

Anssi Lehtonen, Antti Huttunen, Eevi Rimmi, Juho Tuomaala

INDEX

1. Overview

The goal of this project was to make a Super Smash-like fighting game for two players. The game is a local multiplayer, in which the keyboard is shared by two players. The goal of the game is to survive longer than the other player. Players lose a life if they hit the bottom of the scene or touch spikes. The players can try to push each other off the platforms of the level, which stop the players from falling to the ground.

Players can modify settings of the game from “options”. Players can redefine their keyboards, change the volume of the background music and the volume of SFX. The expected keyboards are WASD and arrow keys, where S and down arrow are attack buttons. Volumes of the music and SFX are expected to be at ~ 50 %.

Both players can choose their game character from two different characters. Characters have small differences. The first one falls a bit slower and jumps a tad higher, than the other one, which moves a little faster and can jump a bit further.

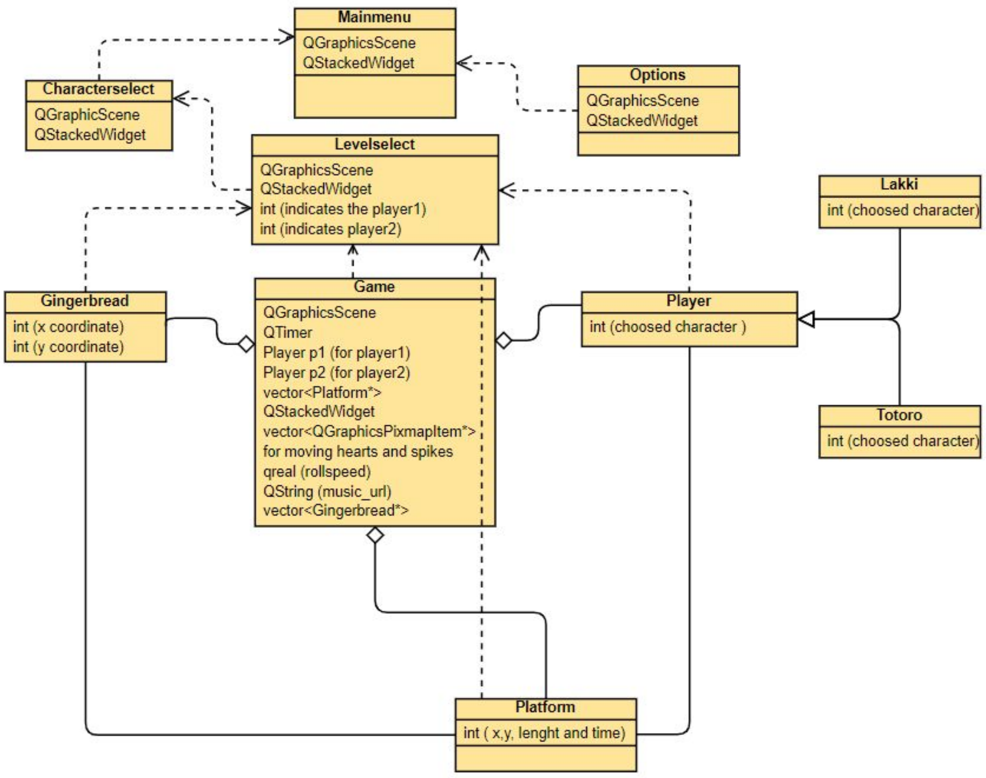
Players can also choose how many lives they have at the start of the game. The expected number is three. If the players give a text or number that is not in range of 1 to 10, the game expects the number of lives to be three.

There are two different levels which the players can choose. The levels are called “Candyland” and “Amfi”. Levels have different backgrounds and different background musics. Another difference between levels is that the level in Candyland is constantly moving while the level in Amfi is stable. Candyland also has spikes attached to the left wall of the scene that kills the player, so the players have to keep moving with the level.

2. Software structure

The program relies heavily on the Qt 5 library, and many of the classes implemented in the project inherit from various Qt classes. The fundamental underlying structure in the program is a stack of QGraphicsview-inheriting class / QGraphicsScene pairs. The View/Scene pairs each form a different user interface and depict one distinct layer of the program (main menu, options etc.), and are pushed & popped to and from the stack in the order they are needed.

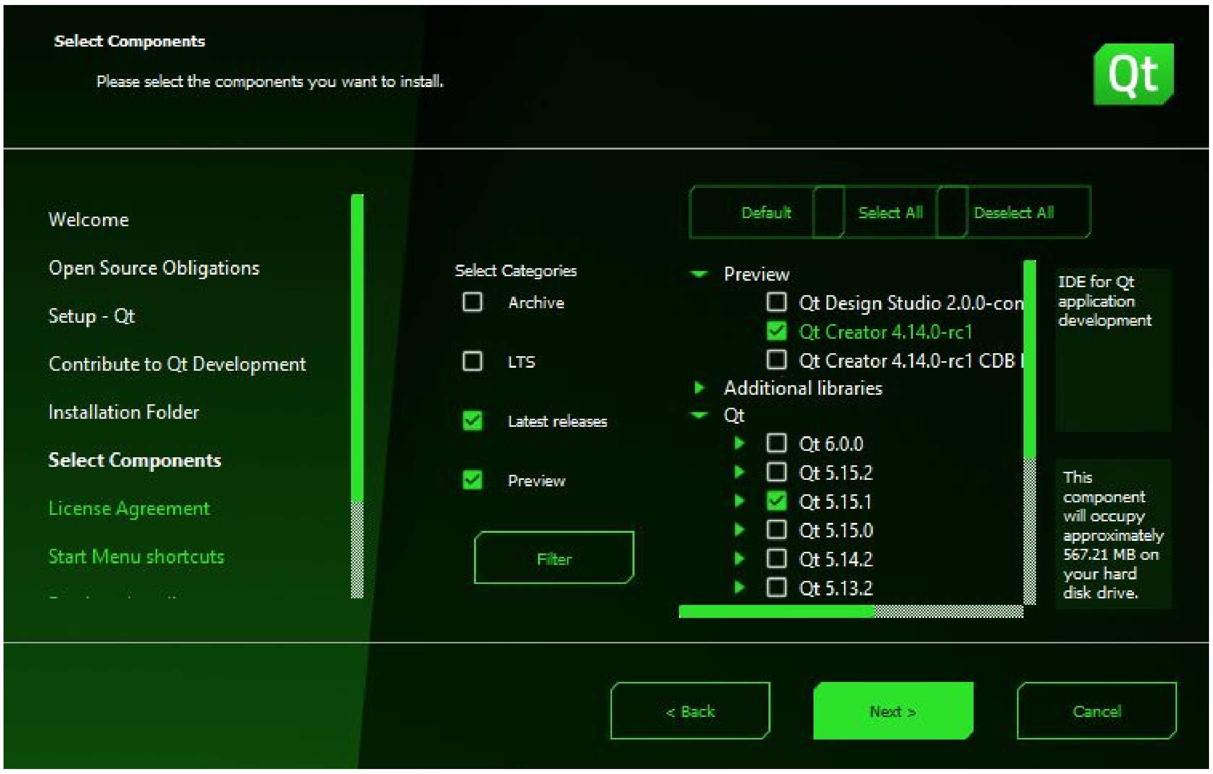
The backbone of the program is a QApplication object, which is necessary for running QWidget-based applications and displaying them on the screen. The QApplication object is also responsible for the main event handling and communication flow between the different classes in the application.



3. Instructions for building and use

Needed Libraries and installation

In Project the Qt5 library is heavily used. The version of Qt of Qt is 5.15.1. You can download Qt installer from Qt's official web page



Select Components

Please select the components you want to install.

Select Categories
- ☐ Archive
- ☐ LTS
- ☑ Latest releases
- ☑ Preview

- Preview
  - ☐ Qt Design Studio 2.0.0-con
  - ☑ Qt Creator 4.14.0-rc1
  - ☐ Qt Creator 4.14.0-rc1 CDB
- Additional libraries
- Qt
  - ☐ Qt 6.0.0
  - ☐ Qt 5.15.2
  - ☑ Qt 5.15.1
  - ☐ Qt 5.15.0
  - ☐ Qt 5.14.2
  - ☐ Qt 5.13.2

IDE for Qt application development

This component will occupy approximately 567.21 MB on your hard disk drive.

You should use option custom installation in installer. You need to create a Qt-account if you do not already have one. Choose from select components menu preview Qt→ Creator 4.14.0-rc1 and Qt→ Qt 5.15.1. Finish installation.

Building and running the program

The program is built by using qmake. You need to use Qt Creator to build and run program by using qmake. If you have problems opening Qt Creator or building and running the project on Linux system. You can find solutions to solve troubles from part *Qt Troubles in Linux*.

To open a project open Qt Creator on your system. From welcome menu choose Projects and click the open icon. Browse to source(src/smash) files and there you should find file smash.pro. Open that file.

Now you can build the project from the build icon and run the project from the run icon on the down left.

🔨 Build icon ▶ Run icon

You can also run the project on Windows by running Smash.exe which can be found from exe/Windows_executer. You need to have all the files in the Windows_executer directory in your computer to run this. You can also use Installer (smash_installer.exe) for windows which installs the project into your computer. Installer can be found from directory exe/installer. Allways when running the program The program needs to be run as administrator.

Qt Troubles in Linux

Solutions for the troubles you might face using linux(Ubuntu 18.04 or 20.04). These troubles were found and fixed on two different Ubuntu virtual machines. Project is developed on the Windows system and Qt5 will work better on Windows. So Windows would be recommended to use for building and running this project.

if QtCreator won't open you might need to install libxcb packages by running command[1]:

```
sudo apt-get install libxcb-randr0-dev libxcb-xtest0-dev libxcb-xinerama0-dev
libxcb-shape0-dev libxcb-xkb-dev
```

If QtCreator won't find toolkits you haven't installed the compiler for c/c++. You can do that by running command:

```
sudo apt-get install gcc g++
```

If you get error `cannot find -lGL` while trying to build or run the project, you need to instal package "libgl1-mesa-dev" by running command[2]:

```
sudo apt install libgl1-mesa-dev
```

As a student you are not able to build the project in Aalto's Linux computers because you don't have access to use `sudo`. If you try to use Aalto's linux computer, you probably get an error `Unknown module(s) in QT: multimedia.` You could solve the trouble by installing package "qtmultimedia5-dev"[3] if you had access to use sudo.

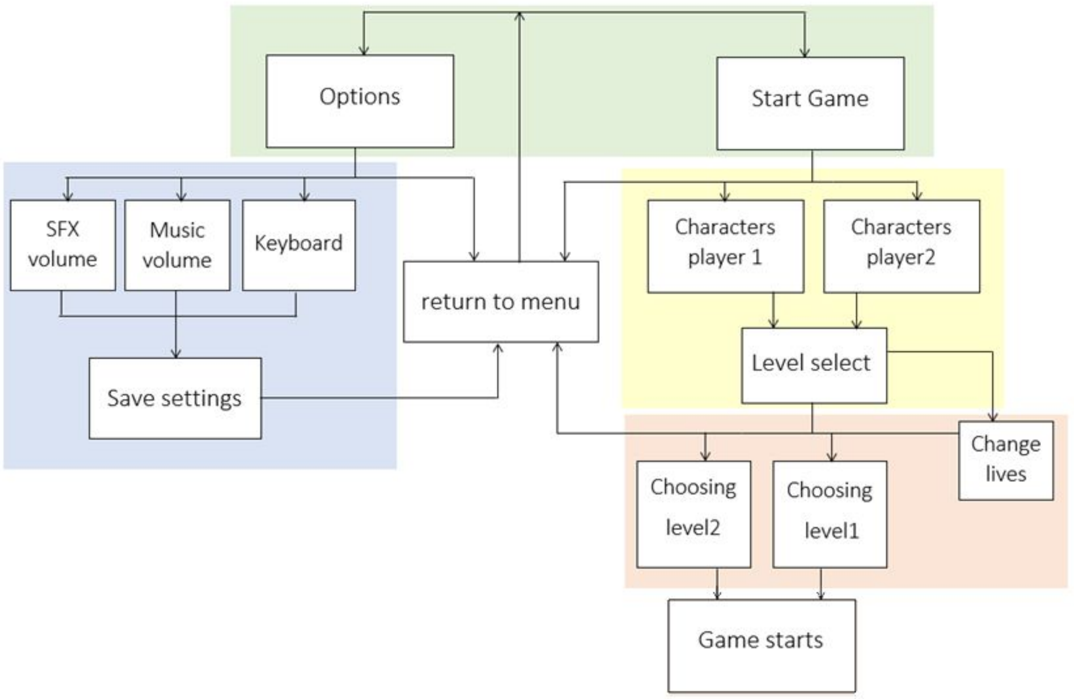
```
sudo apt-get install qtmultimedia5-dev
```

[1]original solution: https://forum.qt.io/topic/116299/qt-creator-ubuntu-20-04/20

[2]original solution: https://stackoverflow.com/questions/18406369/qt-cant-find-lgl-error

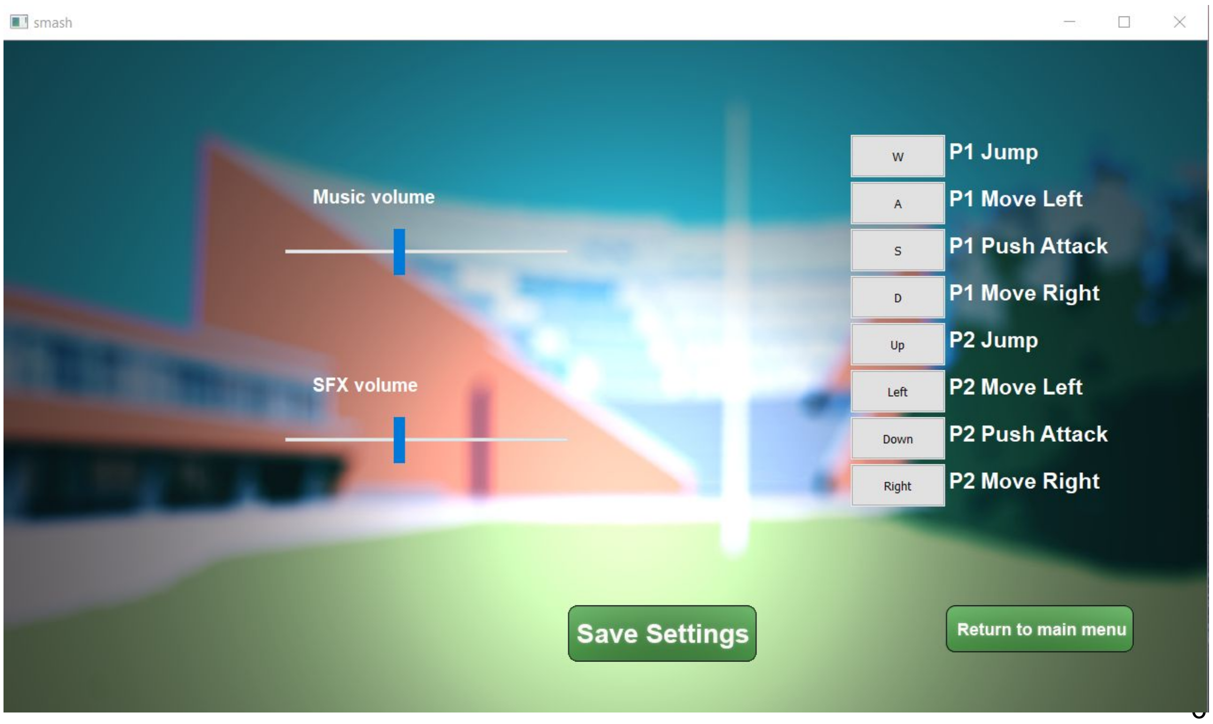
[3] original solution:https://forum.qt.io/topic/27608/unknown-module-multimedia/9

Use of the program



From the diagram above the usage of program can be seen. Every box is a button or other corresponding widget in the actual program. The colors behind boxes indicate different scenes. “Return to menu” possibility is on all three scenes that have arrows pointed at it.

When you run the program you start from the main menu. There are two buttons to choose; Start Game and Options. From Options you get the Options menu.



W	P1 Jump
A	P1 Move Left
S	P1 Push Attack
D	P1 Move Right
Up	P2 Jump
Left	P2 Move Left
Down	P2 Push Attack
Right	P2 Move Right

You can change the volume of background music or SFX from scroll bars. You can change the keybinds by clicking the grey box and clicking the wanted key from your keyboard. Settings can be saved by clicking the Save settings button otherwise the game uses default settings. You get back to the main menu from Return the main menu button.

From the Start Game button you get the character menu where you need to choose a character for each player. After that you get to the Level menu by clicking the Level select button. The Level Select button won't work until characters for both players are chosen.

From the level menu you can change the amount of lives from the upper left corner. Amount needs to be integer which is between 1 to 10 otherwise there will be three lives. The Game starts when you click the level you want to use.

In the game you can manage your character with keybinds you have set. default binds are for player 1 A, W, S, D and for Player 2 UP, DOWN, LEFT, RIGHT. Game ends after one of the players loses. Then you can return to the main menu by clicking the Return to main menu button.

4. Testing

We tested our program with QtCreator, using Memcheck in Debug mode to see if there were any errors in memory. Test was successful and did not show any memory errors in the program.

We started our project by making a view and a scene, where we added our QObjects, that we created. Immediately after this moving of the objects via keyboard was made. This way we were able to test every function that we created immediately on scene by mechanically playing the game.

5. Work log

Responsibilities

Anssi	Design and implementation of core game mechanics, animations, sound effect and music design and implementation
Antti	Setting graphics for the buttons(CSS), Character select menu and choosing different characters, Graphics and animation of totoro, Testing with Linux system, building the project on windows and Linux and troubleshoot relating to that
Eevi	Platforms, gravity that interact with platforms, dying, lives and changing number of them, choosing different levels, platforms read from file, temporal platform, background gingerbreads, template, subclasses to player, drawing the background pictures
Juho	Implementation of menus, options and other UI elements, general program design, graphical layout and design

Timeline

Group had a couple of meetings every week. During the meetings the group checked what we had done, what should be done next and What would be a good structure. Also at the end of the week the group merged branches to Git and solve the big merges together.

26.10. - 1.11 .

- Project plan : Everyone / 4h
- Feedback on the plan received: Everyone / 1h
- started research for the implementation: Everyone / 2h
- Setting up Git environment: Everyone/ 2h

2.11. - 8.11.

- First lines of code: Anssi / 1h
- First draft of Player class + basic scene to draw graphics in: Anssi / 2h
- basic WASD movement implemented: Anssi/ 2h
- First versions of jumping and gravity functionality: Anssi / 2h
- Started to work on the menus: Juho / 3h

9.11. - 15.11.

- Restructuring of the program: functionality divided into more separate classes, changes into the ownerships of critical objects: Everyone / 2h
- first platform implementation: Eevi / 4h

- gravity that interact with platforms: Eevi/ 3h
- rolling screen functionality added: Anssi / 3h
- “housekeeping” in the project repository: Antti / 3h

16.11. - 22.11.

- Started implementing push attack and hitboxes for players: Anssi / 5h
- Post game screen & returning to main menu: Juho / 3h
- proper implementation for dying and lives system: Eevi / 3h
- player falls to platform when dying Eevi/ 3h

23.11. - 29.11.

- Platforms are now read and saved into a file: Eevi / 4h
- additional map added with basic level select functionality: Eevi / 3h
- differences between maps: Eevi/ 2h
- The amount of lives can be changed: Eevi / 2h
- animations for push attack implemented: Anssi / 6h
- A bunch of new graphical additions and changes (UI buttons, backgrounds, player sprites): Antti/Juho/ 5h

30.11. - 6.12.

- Addition of sound effects and music: Anssi/4h
- bug fixes to player movement and closing & starting of the game: Anssi/Eevi / 3h
- CSS styles added to menu buttons Antti/ 10h
- added an installer for Windows Antti/ 2h
- Temporary spawn platforms added to the rolling map: Eevi / 5h
- Graphical improvements for menu buttons: Antti/Juho / 2h
- Cleaning up the code: Eevi / 2h
- Template so players and background objects jump Eevi / 2h
- dividing different kind of players into subclasses: Eevi / 4h
- Started drafting options functionality: Juho / 4h
- volume changing implemented in options Juho / 3h

7.12. - 11.12.

- Fixed some warnings/errors related to graphics: Anssi / 2h
- Troubleshoot with building and Linux system: Antti/ 15 h
- Implemented customizable keybindings. Juho / 6h
- more graphics improvements in menus: Antti/ 3 h
- cleaning up: Everyone / 5h
- Finishing of readme.md and project documentation: Everyone/ 5 h