Projet configuration de claviers

Requirements Gathering

Nicolas, Antonin, Eva et Leïla

1. Vocabulary

Since our project involves designing a keyboard, we must specify some vocabulary related to the subject.

You can go directly to Section 2 and go back later.

1.1. Physical Keyboard

A key is the thing that is pressed in order to produce some behavior on the device.

It is made up of a switch and a keycap.

Keys are placed on a PCB.

1.1.1. Switch

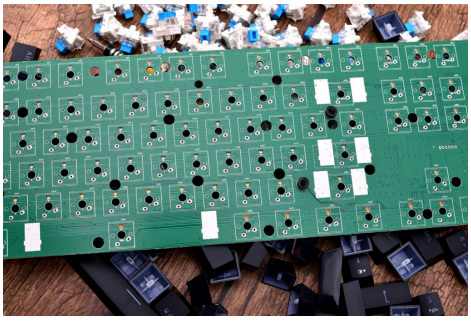
The switch is the physical button under the key, translating the movement into an electric signal.

1.1.2. Keycap

Keycaps are plastic pieces touched by the fingers.

1.1.3. PCB

A Printed Circuit Board (PCB) is the electronic material where all keys are placed.



1.2. HID/USB

The HID/USB protocol is the main protocol being used for USB and Bluetooth keyboards.

It sends keycodes and modifiers to the OS.

1.2.1. KeyCode

A keycode corresponds to the geometric location of a key. For example, “Right Arrow”, “KEY_A” and “F12” are keycodes.

Importantly, the “KEY_Q” keycode can output an A on the computer if the layout is set to AZERTY. A keycode does not correspond to a character.

1.2.2. Modifiers

A modifier is a special key. There are only 4 modifiers:

- SHIFT
- ALT
- CTRL
- SUPER (or WINDOWS)

The modifiers are sent to the OS, and they change the behavior of the touches being pressed at the same time.

1.3. Layout

A layout is a way to map the physical keycodes to characters and actions.

The OS often comes with a default layout for each locale (nationality/country).

For example, AZERTY in France, “QWERTY multilingual standard” in Canada, and QWERTY in the US.

Arbitrary layouts can be created, especially for custom keyboards.

2. Our goal

Our goal is to design a keyboard prototyping software for custom keyboard enthousiasts.

Designing a custom keyboard involves mainly 2 different phases:

- designing the physical keyboard, namely the size and placement of keys
- designing the layout

In general, people who end up designing a custom keyboard have a broad technical knowledge. We can nonetheless differentiate 2 kinds of users:

- **makers**, who have electrical engineering knowledge and build custom PCBs. They may not have a perfect understanding of how keyboard protocols work, but they know the basics
- **hobbyists**, who buy keyboards in kits and build them. They want to create their own layout and may not know how keyboard protocols work.

We focus on the first category of users, but our software should still be usable for the second category.

The 2 phases are not completely independent: the choice of how many characters are needed impacts the number of keys. The way keys are placed can in turn impact which characters and actions are easier to perform.

There is currently no software that allow to do the 2 parts in the same integrated tool, and that is what we want to do.

3. The problem with keyboards

The process of designing a custom keyboard or a custom layout requires to have some **conceptual understanding** of how keyboards work.

It is possible to mitigate part of this complexity if the software does things automatically for the user, but as soon as the user wants a specific feature, we cannot hide the complexity anymore. Imagine building a custom car: you don't need to understand everything to change the colour of the car, but if you want to add a pedal then you must be more informed.

As a consequence, the software has an educational role: the way the user understands the keyboard conceptually (its states, what is possible and what is not, how the OS will respond to unusual combos) may be wrong, so the software must help having a better understanding of keyboards. In particular, the way the software represents keys and character has to be somewhat coherent with the way it works in reality.

4. Usability criteria and scenarios

4.1. Handling incremental changes

The use of the program should be the most intuitive when doing simple changes: adding or removing a single key should be seamless independently of when it was originally placed or when it was modified last.

4.2. Handling backstepping

The user should be able to undo a certain amount of actions to reduce the impact of the mistakes made by the user. Thus, a go back and a go forward button should be implemented, with possible CTRL+Z key binds.

4.3. Easing the setup for the most common keys

Registering the most common keys (i.e., letters, numbers, and symbols directly available on keyboards) should be as seamless as possible, e.g., double-clicking into a key recording. For more complicated cases (untypable keys, characters not present on the user's keyboard, special keys…), there should be an interface to let the user search for certain keywords (like CTRL for the control key) to set up.

4.4. Giving templates for common layouts and geometries

To make the job of non-power users easier, especially on non-fully custom jobs, a set of predefined layouts should be provided to the user to obtain a working base (AZERTY, QWERTY, 75%, 60%, etc.) to modify freely. This serves both as a basis for an end goal and as a possible way for the users to better understand certain aspects of the tool.

4.5. Tipping the user about the program's features

There should be a tips feature available that regularly gives the user small, bite-sized information pieces about some not-so-clear features of the tool. Even if it doesn't replace it, this might be more useful than a full tutorial for less patient users to give them more tools to learn how to use the tool incrementally.

5. Evaluation

As building custom keyboards is pretty niche and requires some technical knowledge, we will have to set the scene by giving missions and goals to test users.

5.1. Connection to a varied group of students

Our group has connections with a widely varied amount of students from all the studying paths at Télécom. All of these varied students with highly varied sets of expertise will be prime candidates for testing. Outside of students, we might also connect with non-engineers through our parents, family and friends to widen our testing field.

5.2. Listing the available tasks

During our testing phase, we will provide the testers with an itemized list of bite-sized tasks (placing a key, setting up the keybind, replacing the key, going back, going forward…) to evaluate the user's interaction over the smallest of parts of the interaction process.

5.2.1. Evaluating tips

During this process, random tips will be given to the user through the program's normal function. We also might want to directly and indirectly evaluate how the users interact with the tips depending on which tip is displayed and at what time. We might also query the user about if he saw the tips and how they influenced their interaction with the program.

5.3. Reddit: a connection to people from all around the world

We posted this message:

"Hi! We are a group of engineering students in a human-machine interaction course. We chose to prototype a tool to create ergonomic keyboards.

In particular, the tool we want to build has 2 modes: one to sketch the geometry of the keyboard and one to create the layout.

The first part of the project is to identify the needs of the users. We would love to have the following information:

- What tools you are currently using to prototype/create layouts, and their limitations.
- What tool would you like in an ideal world?
- What is the feature we should spend the most time on (e.g., predefined geometries and layouts, statistics about keys per language, custom dimensions for keys …)?

We are excited to read your answers, thank you for the time you give us !" on the subreddits r/olkb and r/MechanicalKeyboards which are focused on keyboard personnalisation and creation. We hope to get answers from people that would be interested in using this type of software to give us insights on what they would want in such a tool and what would be more practical for them.

6. Persona

As said before, our software is designed for different types of users, that we can roughly divide in two types : makers and hobbyists. The needs are different between all the users so the next section will allow us to explain more precisely the specificities and scenarios for each type.

6.1. Maker

The makers know the basics of keyboard protocols and have more professional needs. There are high chances that they have already used similar softwares so they also have a good idea about how this type of software works. As the makers are our principal users, this explains why there will not be a full tutorial for our software. However, we need to implement helping features. Indeed, like we mentioned in part 2, no software combines designing the physical aspect and de-

signing the layout. With our software, the makers will expect to have a smooth way to go from one part to the other. The connection between these two parts will be new to these users, so we decided to add a tips feature to help bit by bit.

6.2. General Hobbyist

These users may custom their keyboards for the first time. As they are new to this task, we propose templates for layout. With these and the tips feature, the hobbyists can understand better the general process of designing a keyboard. A documentation will also be available to help these users get used to the software by describing the possible actions they can do and the specificities of possible actions (for instance : how to add a specific layout for when a modifier is pressed).

6.3. Gamer Hobbyist

Gamers continually use their keyboard and, compared to the previous hobbyists, have a general idea of what type of keyboard they would like to work with. Comfort, speed and easy key combinations are often the properties gamers look for. They will want to try different layouts to evaluate for themselves how much the layouts are usable and efficient for video games. This is why we will implement an emulator of the layouts created by the users, to allow them to realize the disposition they created and make changes according to the results.

6.4. Hobbyist with disability (extension)

Some disabilities can limit hand movement and dexterity. Therefore, these could be users of our tool to design a keyboard that would be adapted to their specific needs. To adapt our design to these users who might not know much about keyboard layouts and customization, the most important part would be to make our website controller compatible and offer predesigned configurations to start from a certain basis. A lot of controllers have been created to be adapted to different handicaps, such as the x-box adaptive controller which allows users to create their own controller and therefore adapt it to their impairments. This compatibility would be an extension we add to our website if there is time, but in any case, our tool would allow the creation of more practical keyboards for people with a handicap, whether the keyboard is created by the handicapped person or someone else. The predesigned configurations would also be an extension if the time allows it. We are planning on predesigning basic configurations for a predefined number of keys (such as azerty, qwerty, left-handed adapted keyboard, etc).