Projet configuration de claviers

Requirements Gathering

Nicolas, Antonin, Eva et Leïla

1. Vocabulary

Since our project involves desining keyboard, we must precise 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 behaviour on the device.

It is made up of a **switch**, and a **keycap**

Keys are placed on a **PCB**

1.1.1. Switch

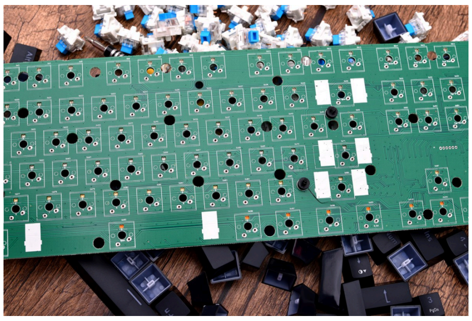
The switch is the physical button under the key, translating the movement into an electric signal.

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 a 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 behaviour of the touches being pressed at the same time.

1.3. Layout

A layout is a way to map the **Phyical 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 nontheless differentiate 2 kinds of users:

- **makers**, who have eletrical 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 keyboard 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 allows 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 do things automatically for the user, but as soon as the user want a specific feature, we cannot hide the complexiy anymore. Imagine builing a custom car: you don't need to understand everything to change the colour of the car, but if you want add a pedal then you must be more informed.

As a consequence, the software has an educational role: the way the user understand the keyboard conceptually (its states, what is possible and what is not, how the OS will resond 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

4.1. Handling incrmental 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 to 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 (ie, letters, numbers and symbols directly available on keyboards) should be as seamless as possible, ex, double click 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 setup.

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 shoud be provided to the user to obtain a working base (AZERTY, QWERTY, 75%, 60%…) to modify freely. This serves both as a basis for an end-goal as well 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 regularily give 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 keyboard 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 out testing phase, we will provide the testers with a 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