

Pd-L2Ork

Final Report

CS4624 Multimedia, Hypertext, and Information Access

Dr. Edward A. Fox

Virginia Tech, Blacksburg, VA 24061

12/13/2022

Members: Nam Nguyen, Tyler Johnson, Haylin Kwok, Daniel Moreno

Client: Dr. Ico Bukvic

Table of Contents

1. Table of Figures	4
2. Executive Summary	7
3. Introduction	8
3.1. Objectives	8
3.2. Deliverables	8
3.3. Client.....	8
3.4. Team Members	9
4. Requirements	10
5. Plan	11
6. Implementation	12
6.1. Tooltips.....	12
6.2. K12-mode	12
7. Testing/Evaluation/Assessment	17
7.1. Tooltips.....	17
7.2. K12-mode	17
8. User's Manual	18
8.1. Use environment.....	18
8.2 Use cases/Tasks supported	18
8.3 Tutorial on use	18
8.3.1. Overview of Tooltips	18
8.3.2. Overview of K12-mode	19
8.3.3. Opening a pre-existing patch window or creating a new one.....	21
Switching to/exiting K12-mode	24
8.3.4. Toggling the K12 menu visibility	27
8.3.5. Using the K12 menu	30
9. Developer's Manual	36
9.1. Inventory.....	36
9.1.1. Tooltips files.....	36
9.1.2. K12-mode files	36
9.2. Installing Software	37
10. Lessons Learned	38

11. Acknowledgements40
12. References41

1. Table of Figures

Figures 1 & 2: What Pd-L2Ork normally looks like (left figure) vs. K12-mode (right figure)13

Figures 3 & 4: Button in File menu to switch to and from K12-mode13

Figure 5: Bottom part of K12 menu (Top part displayed in Figure 2)14

Figure 6: A section of the K12 menu. Here, the Wii tab is open and all objects associated with Wii are displayed.14

Figures 7 & 8: The newly designed K12 menu. The top is shown on the left and the bottom is shown on the right.15

Figure 9: Edit menu in Normal mode15

Figure 10: Edit menu in K12-mode15

Figure 11: K12 menu when hidden16

Figure 12: Description of object when hovering16

Figure 13 (left): K12 menu option. Currently, it is set to be not visible.16

Figure 14 (right): K12 menu option (checked). Now, the K12 menu appears.16

Figure 15 (left): Menus visible when not in K12-mode19

Figure 16 (right): Menus visible when in K12-mode19

Figure 17 (left): Edit menu options in normal mode19

Figure 18 (right): Edit menu options in K12-mode19

Figure 19 (left): Normal objects menu20

Figure 20 (right): K12 objects menu20

Figure 21 (left): Objects from the normal menu	20
Figure 22 (right): Object from the K12 menu	20
Figure 23: The start window of Pd-L2Ork	21
Figure 24: Clicking the “File” button	22
Figure 25: Choosing a file after clicking “Open”	22
Figure 26: A list of recently opened files	23
Figure 27: The “New” option	23
Figure 28: A new, unsaved .pd file	24
Figure 29: The “Switch to K12 Mode” option	24
Figure 30: After switching to K12 mode, new file	25
Figure 31: After switching to K12 mode, pre-existing file	25
Figure 32: The “Exit K12 Mode” option	26
Figure 33: K12-menu is still visible	26
Figure 34: “K12 menu” option in “Put” menu	27
Figure 35: No “Put” menu in K12-mode!	28
Figure 36: Checkmark indicating visibility of K12 menu	29
Figure 37 (left): The button when in the “playing” state	30
Figure 38 (right): The button when in the “building” state	30
Figure 39 (left): The menu before being minimized	31
Figure 40 (right): The menu after being minimized. Notice the menu tab. ...	31

Figure 41: Scrolling to the Logic submenu, which would otherwise be offscreen	31
Figure 42 (left): The “Arduino” submenu collapsed	32
Figure 43 (right): The “Arduino” menu expanded	32
Figure 44: The “Arduino,” “Raspberry Pi,” and “Math” submenus expanded	32
Figure 45: Tooltip for the “Arduino Connect” object. Notice that the button has a lighter shade.	33
Figure 46: We wish to place the “Wiimote Buttons” object	34
Figure 47: Placing the “Wiimote Buttons” object at a desired location	34
Figure 48: Moving the “Wiimote Buttons” object to a new location	35

2. Executive Summary

Our group's work dealt primarily with building upon pre-existing features in order to improve functionality of a real-time visual programming environment for A/V and graphics processing.

Our first task was to implement a tooltips functionality for the objects in Pd-L2Ork. This would enable users to see helpful information displayed whenever hovering their mouse over objects, or over certain parts of an object. For this task, we parsed through an index of information within the Pd-L2Ork file directory and connected each object to their corresponding tooltip found from that index, allowing for the functionality of displaying tooltips. The deliverable for this task was a merge-able patch containing our code allowing for this functionality, of which we sent to our client.

Next, we were tasked with implementing a K12 editing mode, different from the default mode and settings for creating and editing files in Pd-L2Ork. This mode deliberately restricts certain features that are normally available when a user wishes to create and edit files. It also provides users with a menu interface that is designed to make creating and editing easier. As the name suggests, this functionality is aimed at K-12th grade students. For this task, we created a K12 menu that contains K12 objects that the user can place. This menu is visible when the user is in K12-mode, but can also be shown in the normal mode via the "Put" menu. We also removed certain menus from the top menu in order to reduce the amount of features the user can use. The deliverable for this task was a patch that we sent to the client containing all our additions involving the K12-mode.

3. Introduction

Pd-L2Ork is an interactive visual programming environment that is designed for live manipulation of multimedia and digital signals. Pd-L2Ork is built off of the PureData programming language, which was first developed in the 1990s. Pd-L2Ork features a variety of pre-built components, all as source files, that allow users to experiment with the creation and manipulation of sounds, among other things. The programming environment also allows users to put together their own components for unique manipulations and compositions. The developers of Pd-L2Ork are aiming to create an application that is easy to learn and easy to use. To that end, they have begun to transition the application for web browser support. The work done by our group this semester therefore focuses primarily on improving the usability of the program by adding in certain features, as well as completing the transition of the application to be fully supported in a web browser.

3.1. Objectives

The primary objective of this project is to build upon a pre-existing application, adding certain user-friendly features to enhance the application, and transitioning the application to make it more portable.

3.2. Deliverables

Our first deliverable is tooltips functionality, which will cause useful information to be displayed when a user hovers over an object. Our second deliverable is functionality for a K12 objects menu, which can be easily toggled to appear or disappear. Our final deliverable is a version of the application that offers web browser support.

3.3. Client

Our client overlooking this project is Ico Bukvic, a professor in Creative Technologies in Music at Virginia Tech.

3.4. Team Members

Our team consists of four members, Nam Nguyen, Tyler Johnson, Haylin Kwok, and Daniel Moreno, with all of us majoring in Computer Science.

4. Requirements

There are three (3) things that need to be done for this project:

1. Tooltips
2. K12-mode
3. Web-browser support

Tooltips: Currently, Pd-L2Ork has help pages that can be displayed by right-clicking on an object and clicking on the help option. To make this help information more convenient to access, help information should also be displayed when the user hovers over specific parts of an object.

K12-mode: Currently, Pd-L2Ork has an implementation of K12-mode, but it is separate from the main program. The K12-mode is catered towards beginners to Pd-L2Ork, especially towards kids. The K12-mode should simplify the program so that kids are not overwhelmed as well as provide an easy-to-use interface to start learning how to use the program. In addition, the program should be able to switch to and from K12-mode which is something that the current version of Pd-L2ork doesn't have.

Web-browser support: The client would like a way for users to have easy access to Pd-L2Ork. To accomplish this, use of the program through a web browser should be supported (using Emscripten). Currently, functionality for this has already been made, however these changes must be carefully merged with the existing code.

5. Plan

The plan to complete the above requirements was to split into two groups. One group would work on tooltips while the other group would work on K12-mode. Once both groups have finished their work, the whole team would come together to work on the web-browser support. In addition, the team decided on the following timeline:

September:

- Get familiar with the code base
- Start work on Tooltips/K12-mode

October:

- Finish Tooltips/K12-mode

November:

- Begin work on web-browser support
- Testing and debugging

December:

- Finish work on web-browser support
- Walkthrough of project with client to ensure all deliverables have been sufficiently fulfilled

The team also planned to have weekly Friday meetings with the client to ensure that the work being done was satisfactory to the client.

6. Implementation

6.1. Tooltips

Before we could begin implementing the tooltip functionality for objects in Pd-L2Ork, we had to first gain an understanding of where the tooltip information for each object comes from, and then figure out how to extract and use that data in a way that would allow us to display tooltips to the user when they hover their cursor over objects.

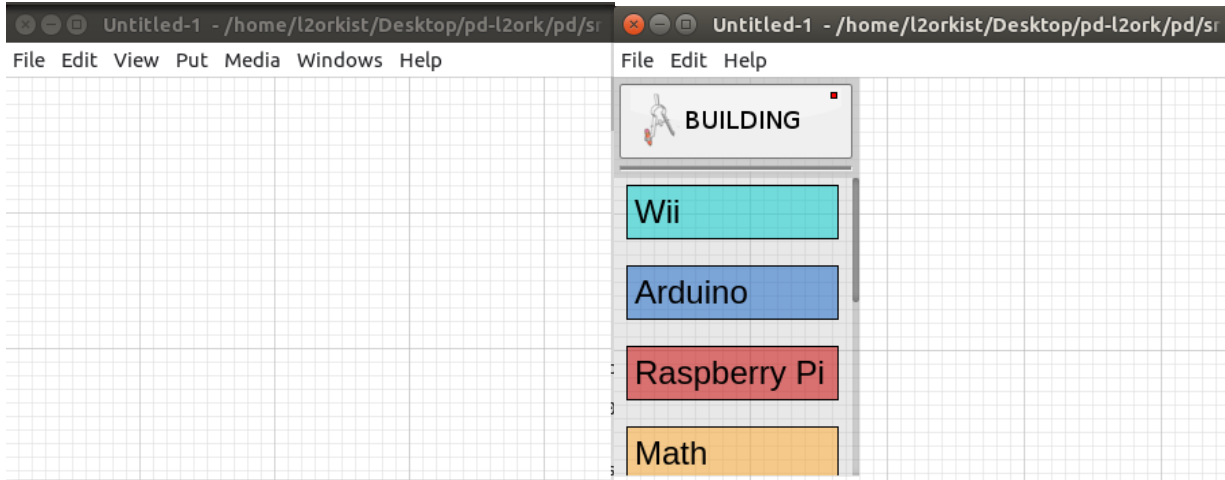
Within Pd-L2Ork's vast file system, there are several locations where the data used for object tooltips can come from, depending on the type of object. This made it challenging at first when trying to figure out how to consolidate all of this information in order to efficiently extract and display the data as tooltips. With help from our client, we learned that the majority of the objects already had the main part of their tooltip data, the object's description, contained in a JSON file (search.index) generated by a JavaScript file (pdgui.js). After altering some code within both pdgui.js and a C file (g_text.c), we were able to parse through the JSON file using an already existing search function, extract each object's description, and display the description for each object when hovered over.

6.2. K12-mode

The majority of the work our group did for K12-mode was in JavaScript. Our work focused primarily on searching source files for pre-existing functionality, then combining the functionality that we found to create new features.

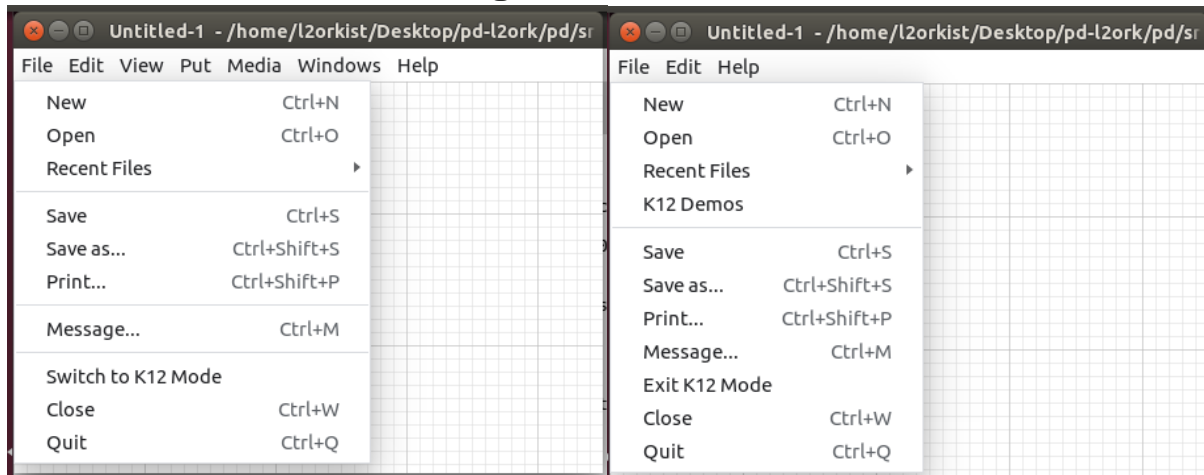
First, we figured out how to manually remove an existing menu without breaking the rest of the program. We did this by commenting out a section of code in the source file where the menu was being instantiated. Then, we tied the removal of certain menus to a global K12-mode flag, which caused those menus to be or not be displayed based on the value of the flag. However, our client wanted users to have the ability to add and remove certain menus at will, based on the current value of the K12-mode flag. Working with our client, we created a function that toggled the K12-mode

flag, then tied this function to a specific event which would cause the menus to appear or disappear whenever the user triggered this event.



Figures 1 & 2: What Pd-L2Ork normally looks like (left figure) vs. K12-mode (right figure)

We later tied the function to a more meaningful event. Now, the user could hide or show menus by clicking a specific button in one of the menus, which would always remain visible regardless of what the K12-mode flag's value was. However, this button did not have a meaningful label, which we would fix later (as shown in Figures 3 & 4).



Figures 3 & 4 (left, right): Button in File menu to switch to and from K12-mode

We then shifted our focus to designing the main K12 (side) menu displayed on the left (see Figure 2). We first made tabs that display different buttons when they are clicked on. We then added suitable pictures so that the menu looked more like what we would have in the final product. Finally, we added functionality to the buttons so that objects are created when the buttons are clicked. At this point, the K12 menu was at the top of the window and there was no scrollbar or “building” button. In addition, the tabs were labeled with pictures instead of words. After a brainstorming session with our client going over different menu designs, we ultimately decided on a side menu. We decided to design the K12 menu as shown in Figures 5 and 6.

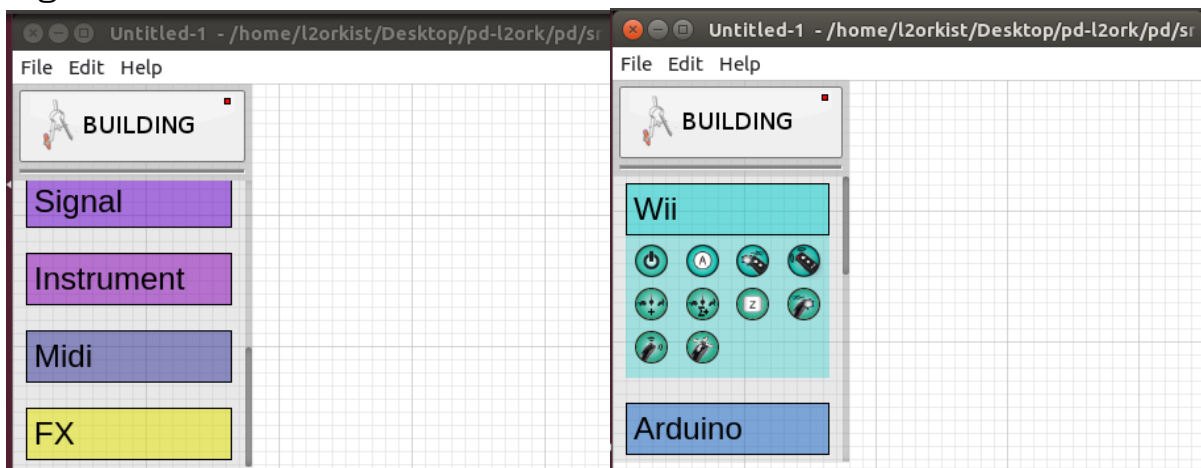
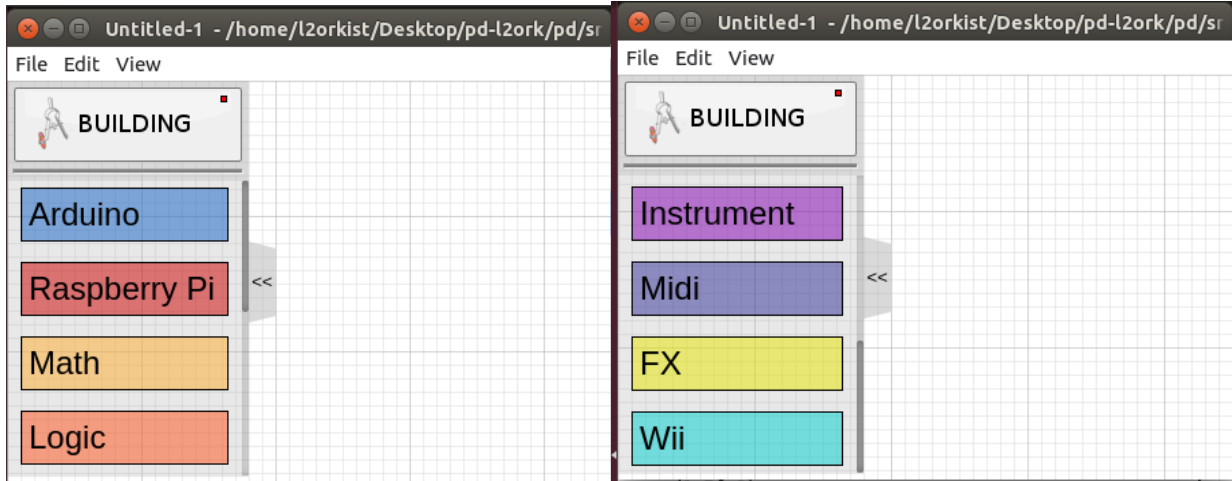


Figure 5 (left): Bottom part of K12 menu (Top part displayed in Figure 2)

Figure 6 (right): A section of the K12 menu. Here, the Wii tab is open and all objects associated with Wii are displayed.

We then went on to touch up the design of the K12 menu, including adjusting the order of the tabs and the spacing of the elements within the K12 menu. We also adjusted the menus and submenus in the top, removing the Help menu and adding the View menu in its place. Figures 7 and 8 show the newly designed menu while Figures 9 and 10 show the difference in menu options between modes.



Figures 7 & 8: The newly designed K12 menu. The top is shown on the left and the bottom is shown on the right.

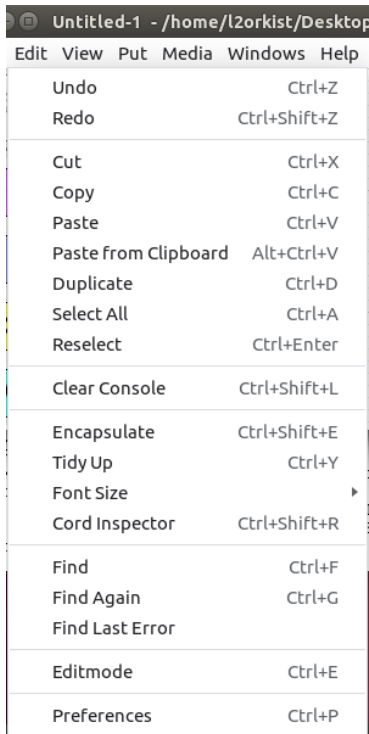


Figure 9: Edit menu in Normal mode

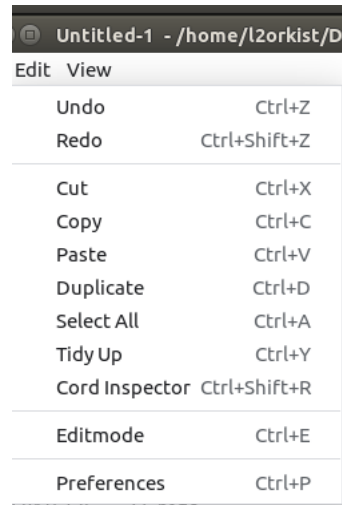


Figure 10: Edit menu in K12-mode

We also added a button that would hide the K12 menu when clicked. This is because the client was concerned about the amount of space that the menu would take up. Figure 11 shows how the program looks when the K12 menu is hidden. We also added descriptions of what the objects are in the menu when the user hovers over them. Figure 12 shows one such example.



Figure 11: K12 menu when hidden

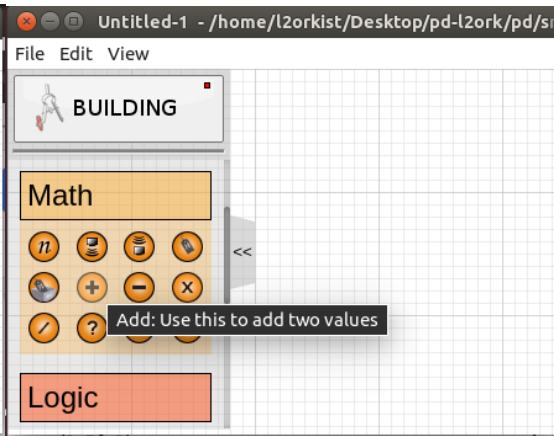


Figure 12: Description of object when hovering

The last thing we did was to add a button that would allow the user to see the K12 menu even if they were not in K12-mode. This is so that users still have access to the K12 menu even if they decide to move into normal mode where they have access to all of the features. Figures 13 and 14 show this feature.

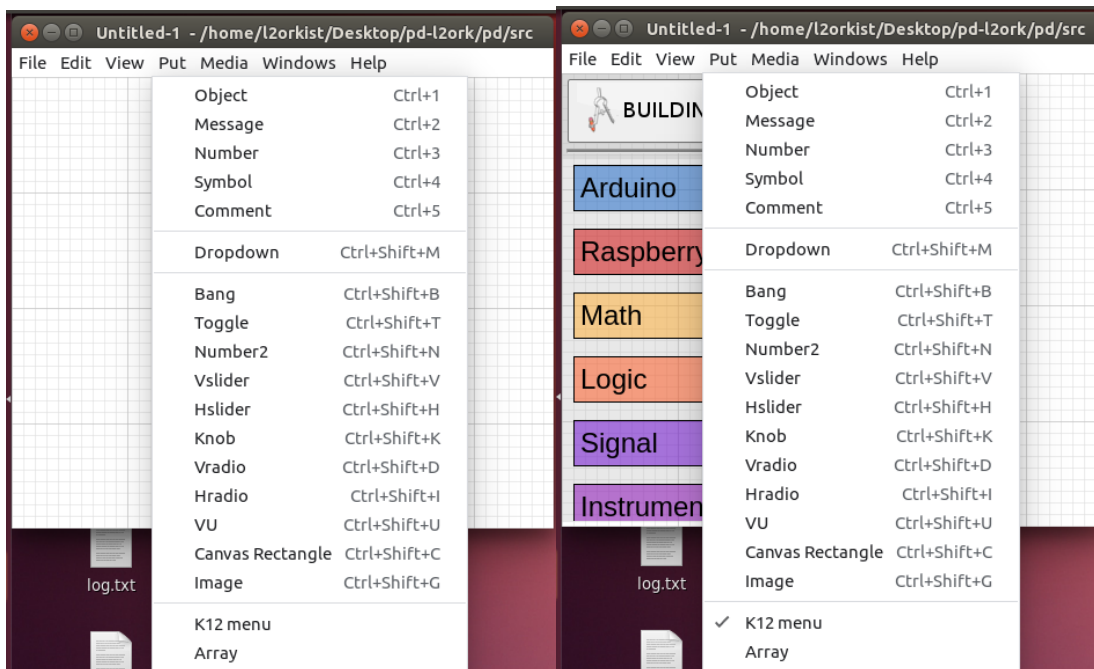


Figure 13 (left): K12 menu option. Currently, it is set to be not visible.

Figure 14 (right): K12 menu option (checked). Now, the K12 menu appears.

7. Testing/Evaluation/Assessment

7.1. Tooltips

The only way to really test whether or not our implementation of the tooltip functionality was successful or not was to first alter the code related to displaying tooltips, and then run Pd-L2Ork and hover over an object to see if a tooltip was being displayed or not. This made testing for complete functionality difficult considering how unclear some of our issues were at times.

7.2. K12-mode

Since most of the work done for K12-mode has been all frontend, testing was done by repeatedly re-running the program and seeing what was affected by the changes. Evaluation is done mainly by the client, who provides feedback on what could be improved or done differently. In addition, members of the Pd-L2Ork team working on the hardware side of things also provided feedback on the design and usability of the K12 menu.

8. User's Manual

8.1. Use environment

L2Ork stands for Linux Laptop Orchestra, so naturally the main operating system that Pd-L2Ork runs on is Linux. Although other operating systems are supported, we developed everything on Linux machines as our client strongly suggested we do so (and even allowed us to borrow a Linux laptop if we needed one). In addition, Pd-L2Ork is an application that can be installed onto your local machine, so use and development of the application is done there.

8.2 Use cases/Tasks supported

The tooltips are used in order to display help information when the user hovers over certain parts of an object. This will hopefully help the user understand how the object works in a fast and convenient way. The K12-mode is used to help beginners (mainly students in K-12th grade) learn how to use the application. Although there are not many features in this mode, there are special K12 objects that the user can play around with. In addition, descriptions of the K12 object are also displayed when the user hovers over its icon in the K12 menu.

8.3 Tutorial on use

Since we are building on a pre-existing system, we will not describe the functionality of the entire application here. Instead, we will explain how to use features directly related to what we implemented.

8.3.1. Overview of Tooltips

The functionality of the tooltips is fairly self-explanatory. The user simply hovers their cursor over an object in the programming environment, and a tooltip (box of text) explaining the usage of the object will be displayed at the same location of the cursor.

8.3.2. Overview of K12-mode

In K12-mode, simplicity and ease-of-use take precedence over user control. Various top menus are hidden (e.g., Put, Media, Windows, Help) from the user (Figure 16) that are normally visible (Figure 15).

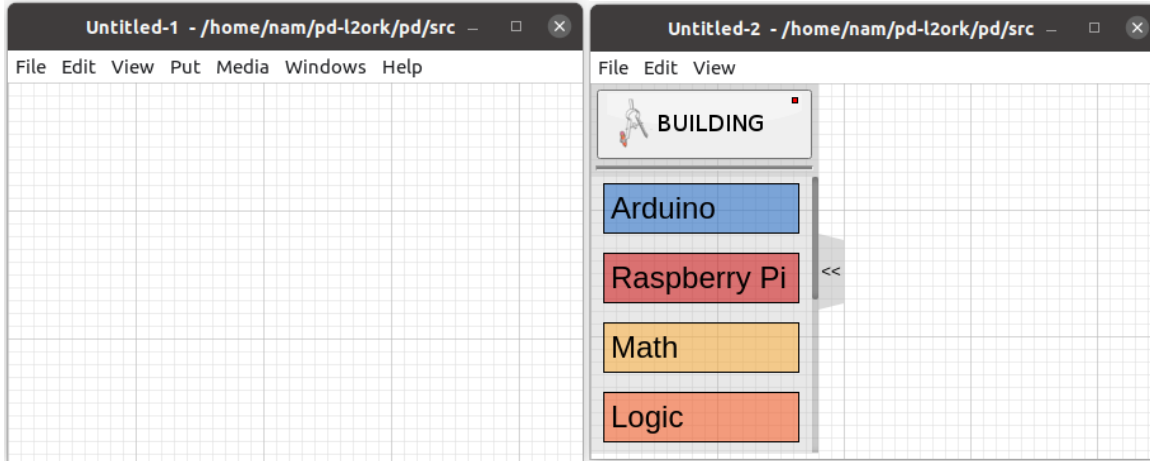


Figure 15 (left): Menus visible when not in K12-mode

Figure 16 (right): Menus visible when in K12-mode

Additionally, the menus that are not hidden may have less options. In the Edit menu, for instance, Figure 17 shows what options are normally available, many of which are hidden as seen in Figure 18.

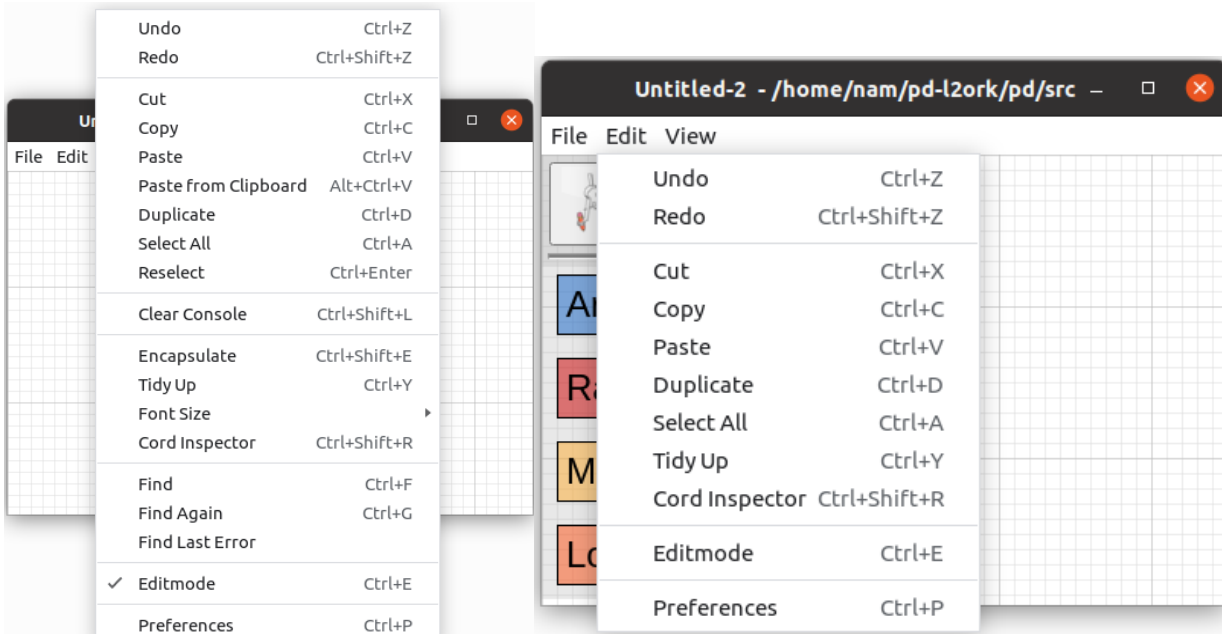


Figure 17 (left): Edit menu options in normal mode

Figure 18 (right): Edit menu options in K12-mode

The key feature of K12-mode is the K12 menu, which offers users an object menu (Figure 20) that is simpler and easier to understand and use than the standard object menu (Figure 19).

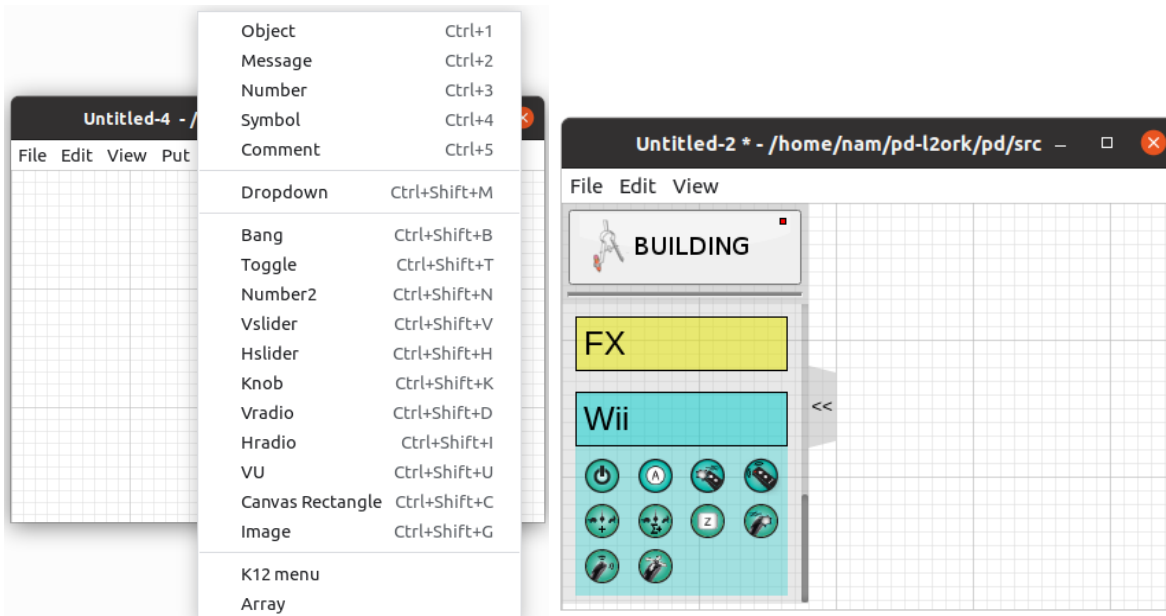


Figure 19 (left): Normal objects menu
 Figure 20 (right): K12 objects menu

Objects from the K12 menu (Figure 22) should be easier to understand than objects from the standard object menu (Figure 21).

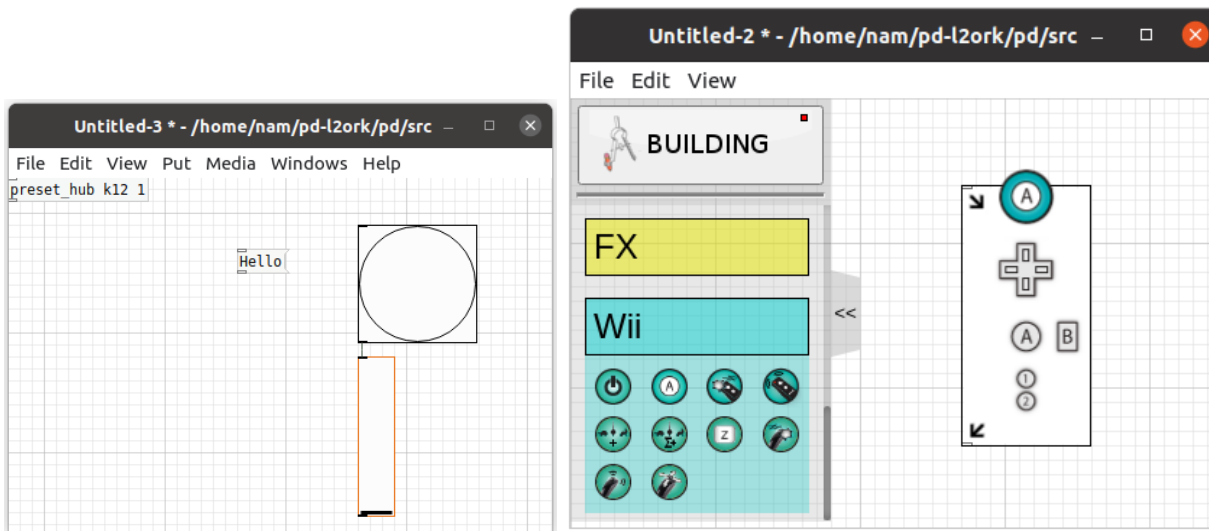


Figure 21 (left): Various objects from the normal menu
 Figure 22 (right): Object from the K12 menu

8.3.3. Opening a pre-existing patch window or creating a new one

In order to use the K12 menu and switch or exit K12-mode, you will need to open a pre-existing **.pd** file or create a new one. When you first start Pd-L2Ork, you will see a window like that shown in Figure 23.

```
Pd-L2Ork
File Edit View Media Windows Help
DSP
by HCS)
libdir_loader: added 'pan' to the global objectclass path
freeverb~ v1.2
libdir_loader: added 'hcs' to the global objectclass path
libdir_loader: added 'jmmmp' to the global objectclass path
libdir_loader: added 'ext13' to the global objectclass path
libdir_loader: added 'ggee' to the global objectclass path
libdir_loader: added 'ekext' to the global objectclass path
libdir_loader: added 'disis' to the global objectclass path
libdir_loader: added 'lyonpotpourri' to the global objectclass
path
pdlua 0.10.1 (GPL) 2014-2020 Martin Peach et al., based on
lua 0.6~svn (GPL) 2008 Claude Heiland-Allen <claude@mathr.co.uk>
pdlua: compiled for pd-0.48 on Sep 2 2022 23:46:42
Using lua version 5.3
pdlua: using JavaScript interface (Pd-l2ork nw.js version)
tried but couldn't sync A/D/A
error: audio I/O stuck... closing audio
[z~] part of zexy-2.3.1 (compiled Sep 2 2022)
    Copyright (c) 1999-2018 IOhannes m zmölnig,
forum::für::umläute & IEM
[limiter~] part of zexy-2.3.1 (compiled Sep 2 2022)
    Copyright (c) 1999-2018 IOhannes m zmölnig,
forum::für::umläute & IEM
```

Figure 23: The start window of Pd-L2Ork

To open a pre-existing file, click on “File” in the upper left corner of the start window, as shown in Figure 24.

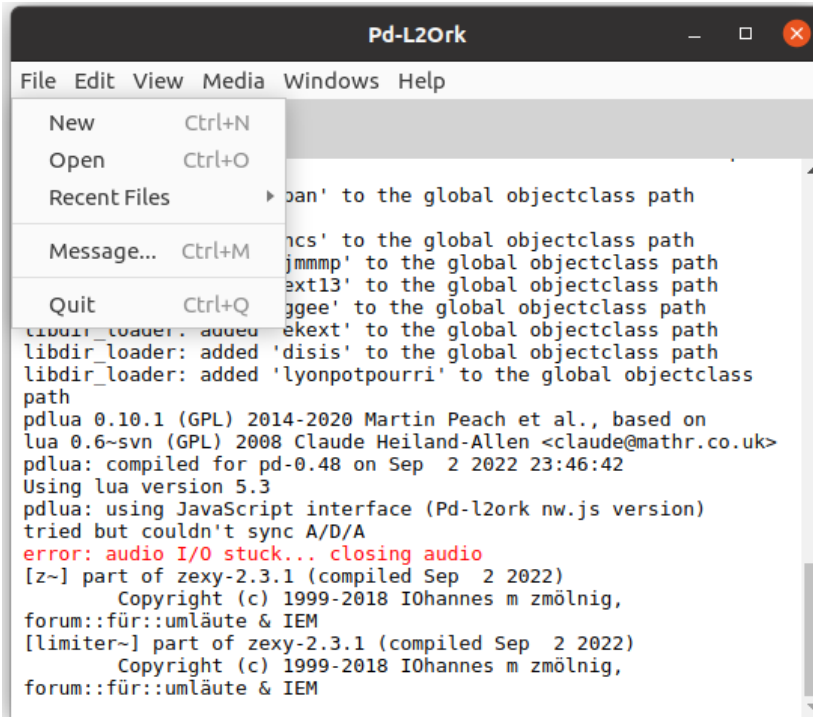


Figure 24: Clicking the “File” button

This will display a dropdown menu with several options. Click on the “Open” option. Then, select the **.pd** file that you wish to open (Figure 25).

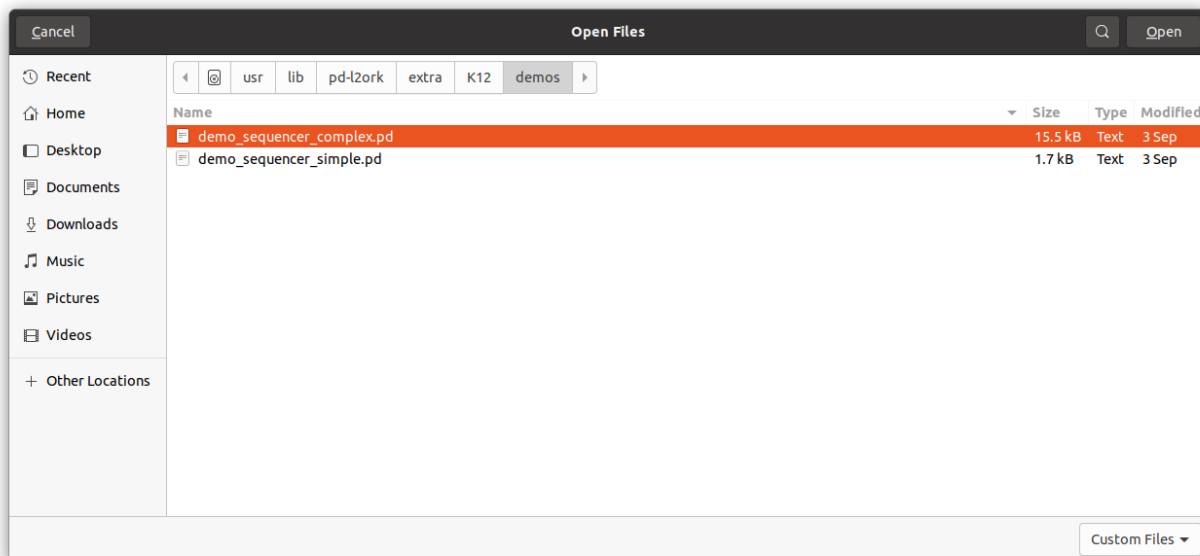


Figure 25: Choosing a file after clicking “Open”

If you have opened a file in the past, you can quickly reopen it by hovering over the “Recent Files” option in the “File” menu and selecting the desired file (Figure 26).

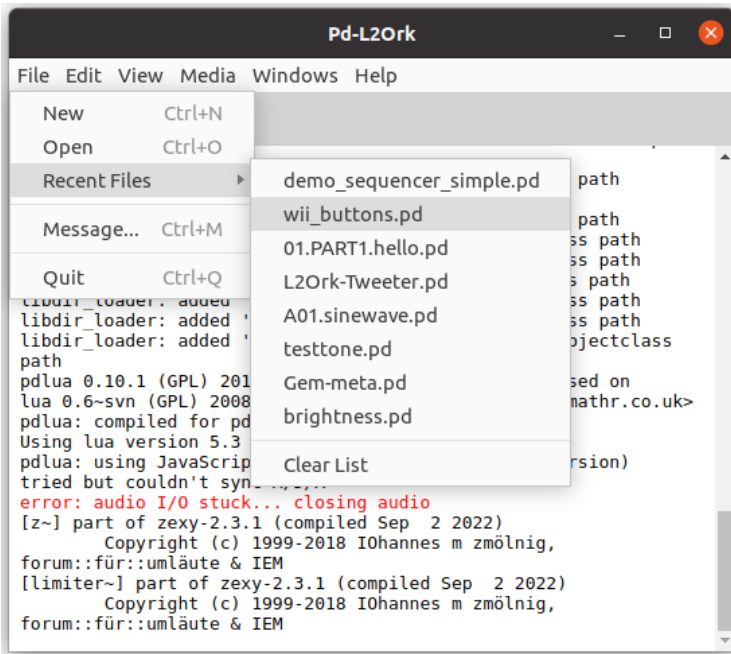


Figure 26: A list of recently opened files

To create a new **.pd** file, open the “File” menu and select “New,” as shown in Figure 27.

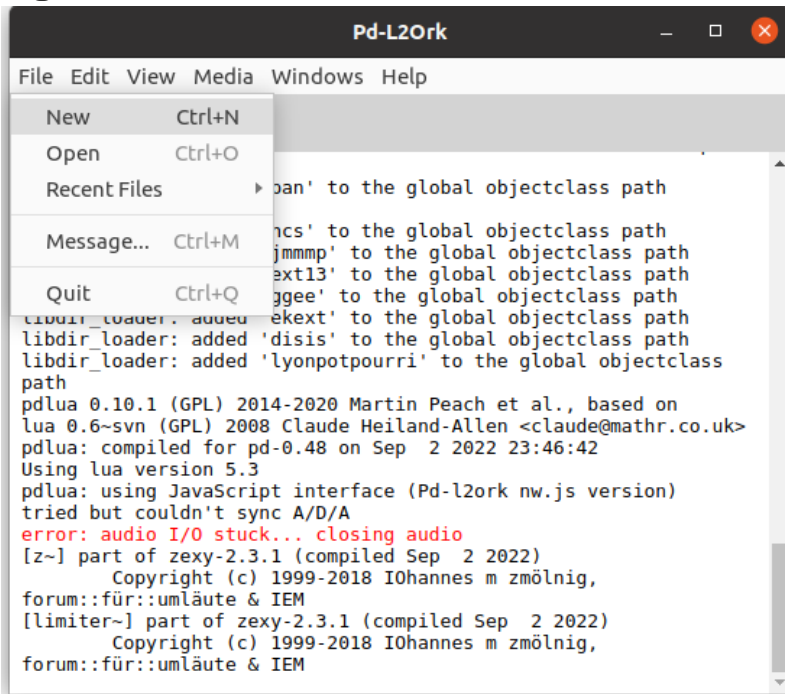


Figure 27: The “New” option

You should see a blank window appear as shown in Figure 28.

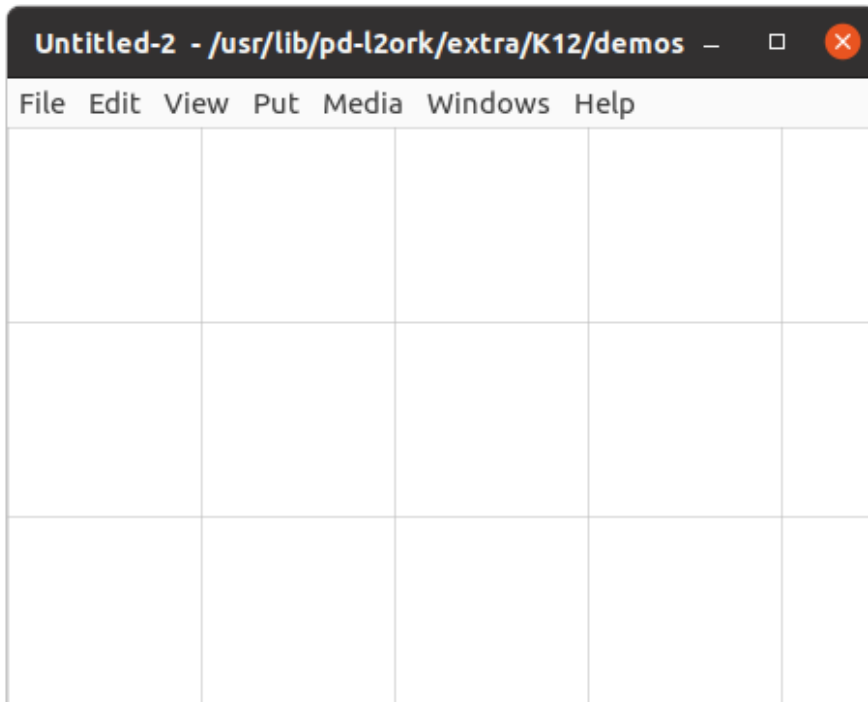


Figure 28: A new, unsaved .pd file

Switching to/exiting K12-mode

After opening a pre-existing **.pd** file or creating a new one, you will be able to enter or exit K12-mode. To enter K12-mode, open the “File” menu in the upper left corner and click the “Switch to K12 Mode” option (Figure 29).

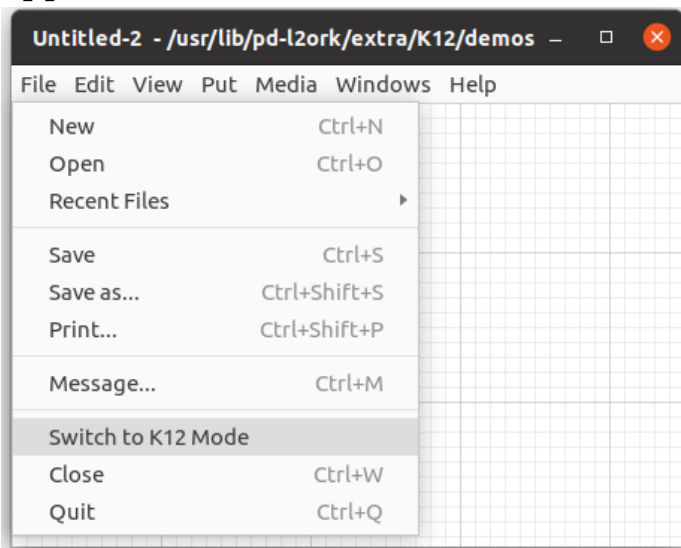


Figure 29: The “Switch to K12 Mode” option

For new files, the window should then look like Figure 30.

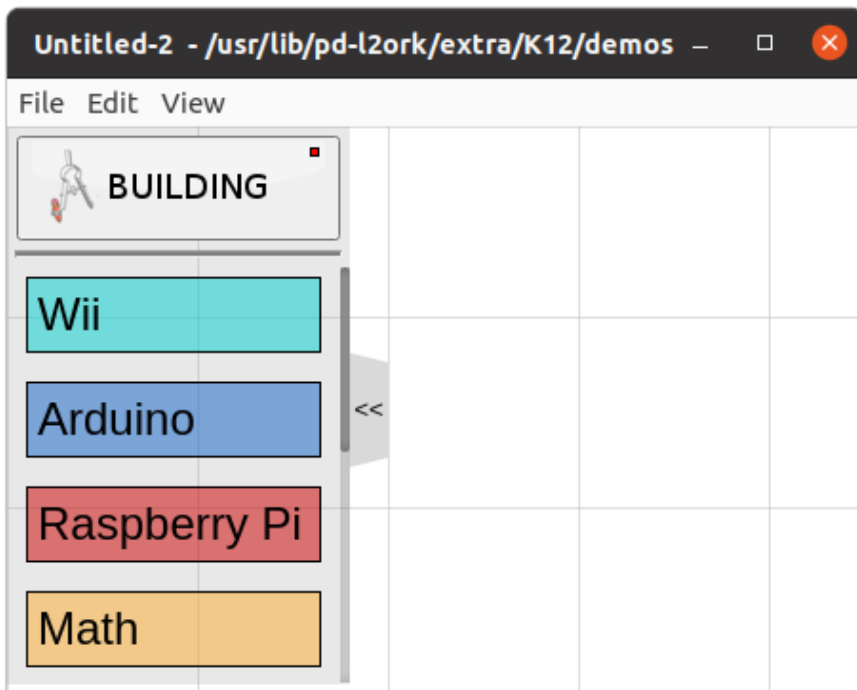


Figure 30: After switching to K12 mode, new file

Figure 31 shows what the window may look like for a pre-existing file.

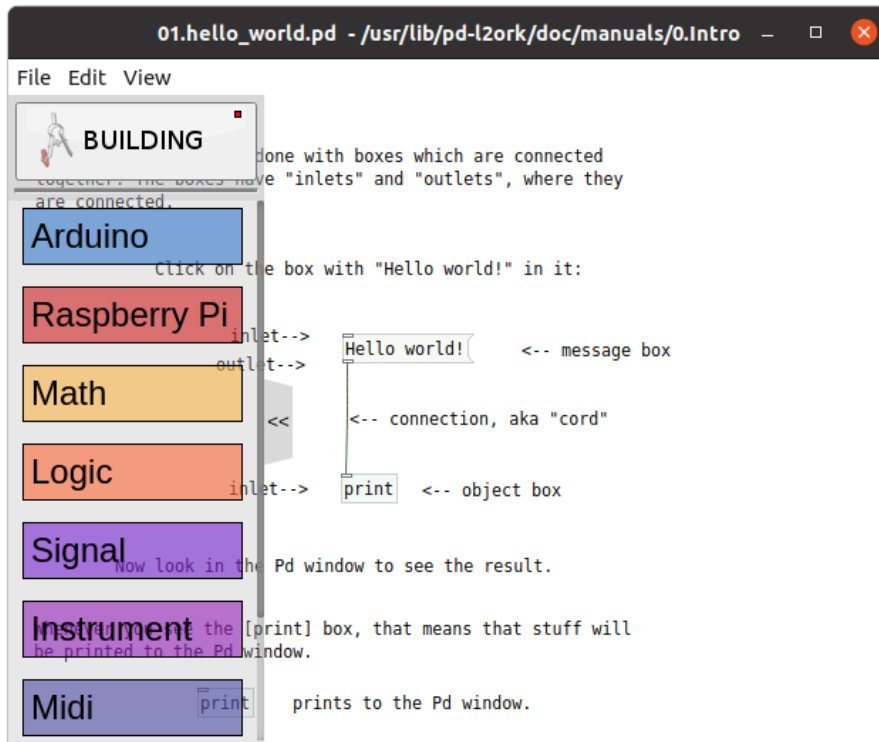


Figure 31: After switching to K12 mode, pre-existing file

To exit K12-mode, open the “File” menu and select the “Exit K12 Mode” option (Figure 32).

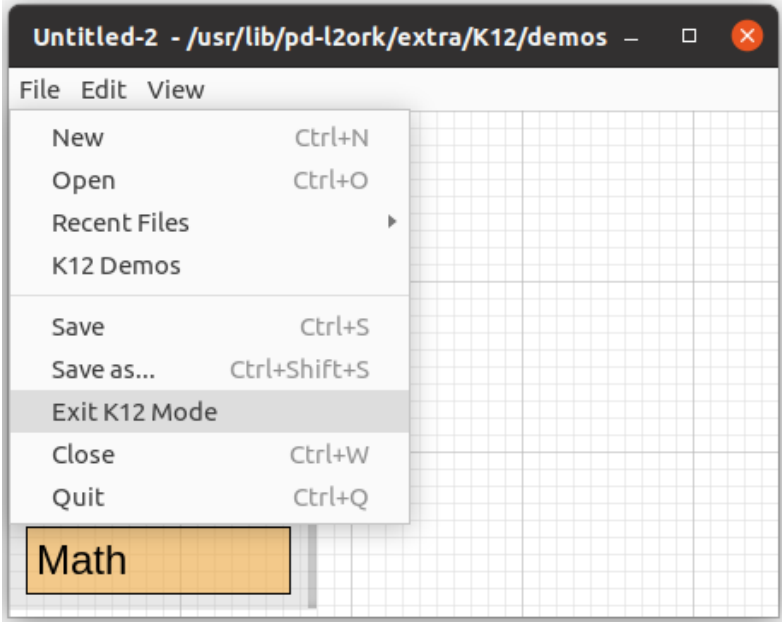


Figure 32: The “Exit K12 Mode” option

Notice in Figure 33 that the K12-menu is still visible, even after exiting K12-mode.

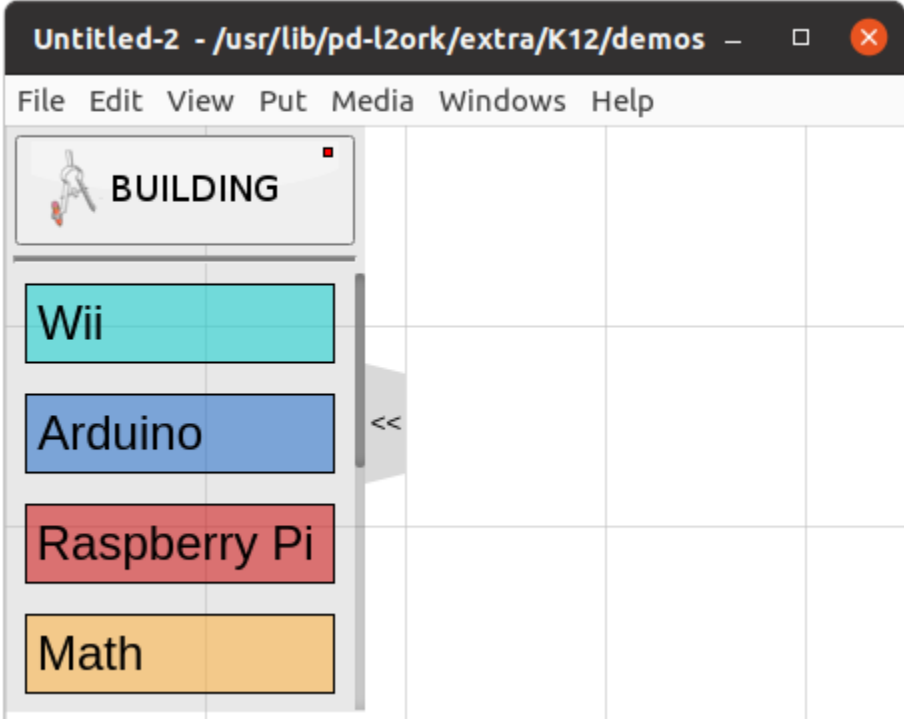


Figure 33: K12-menu is still visible

8.3.4. Toggling the K12 menu visibility

There are two ways to toggle the visibility of the K12 menu. The first is to select the “Switch to K12 Mode” option in the “File” menu. If the menu is not visible, then it will be displayed. However, if the menu is already visible, then switching to K12-mode will not hide the menu. To directly toggle the visibility of the K12 menu, open the “Put” menu and select the “K12 menu” option, as shown in Figure 34.

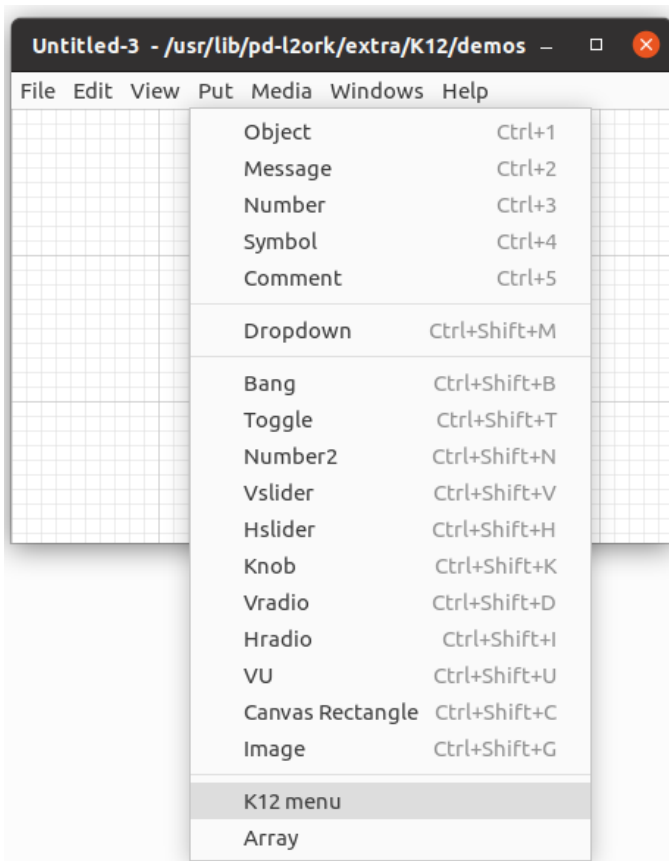


Figure 34: “K12 menu” option in “Put” menu

Notice in Figure 35 that the “Put” menu is hidden when in K12-mode, which means that you cannot toggle K12 menu visibility while in K12-mode.

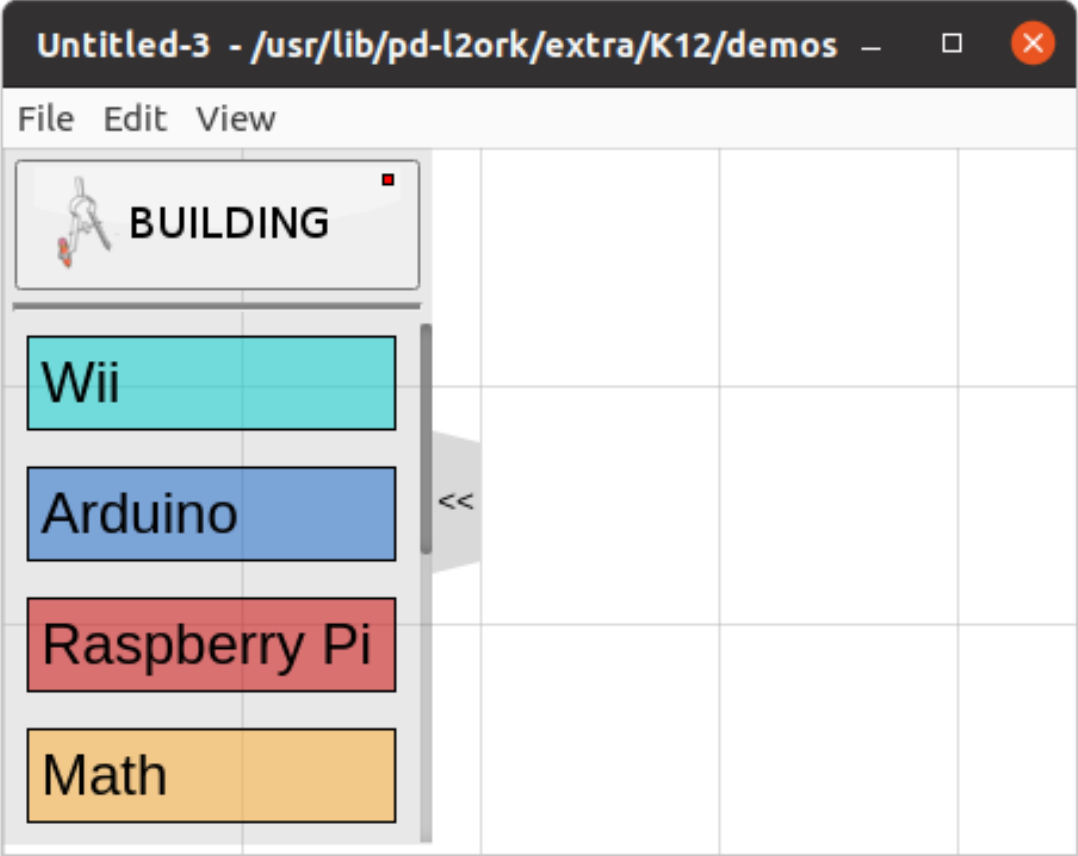


Figure 35: No “Put” menu in K12-mode!

A checkmark next to the “K12 menu” option (Figure 36) indicates that the K12 menu is currently visible. The checkmark is not present when the K12 menu is hidden.

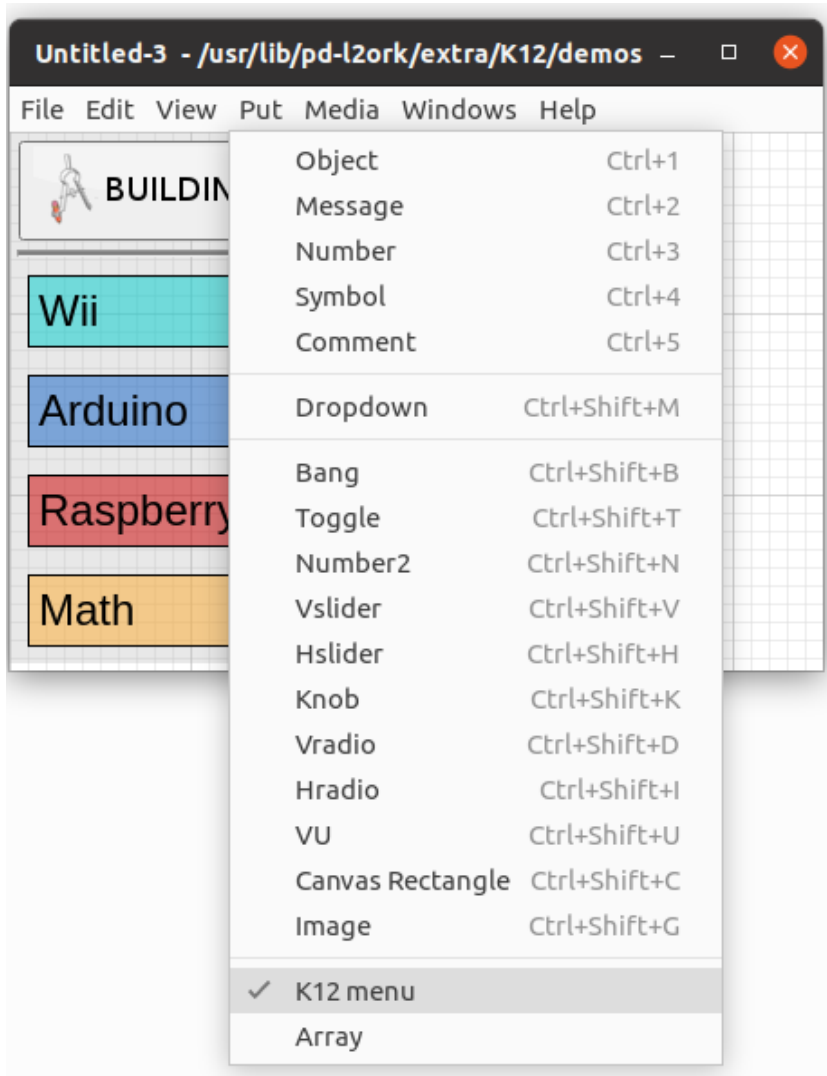


Figure 36: Checkmark indicating visibility of K12 menu

8.3.5. Using the K12 menu

There are two major states when you have a **.pd** file open: the “building” state and the “playing” state. When in the “building” state, you may edit the canvas and add or remove objects, but you cannot interact with the objects. When in the “playing” state, you may interact with objects on the canvas, if there are any, but you may not modify the canvas. To that end, there is a button at the top of the K12 menu that allows for easy switching between the states. Simply click the button to switch states. At any given time, the button will be labeled “BUILDING” (Figure 38) or “PLAYING” (Figure 37). Its current label reflects the state that you are in. For instance, the button will be labeled “BUILDING” when you are in the “building” state.

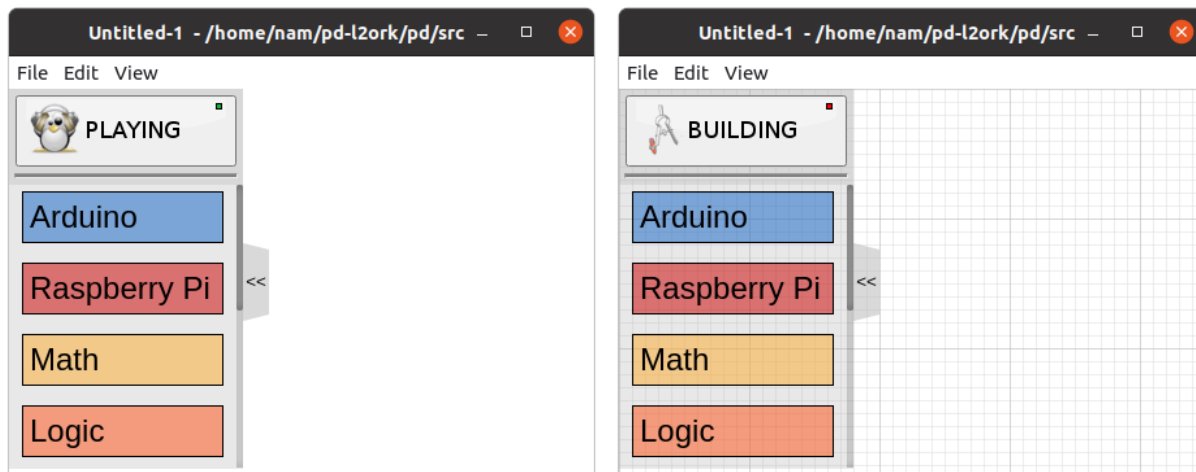


Figure 37 (left): The button when in the “playing” state

Figure 38 (right): The button when in the “building” state

The K12 menu can also be minimized or maximized. To minimize the menu, click the tab on the very right of the menu (Figure 39). When the menu is minimized, only the tab will be visible (Figure 40). To maximize the menu, click the menu tab.

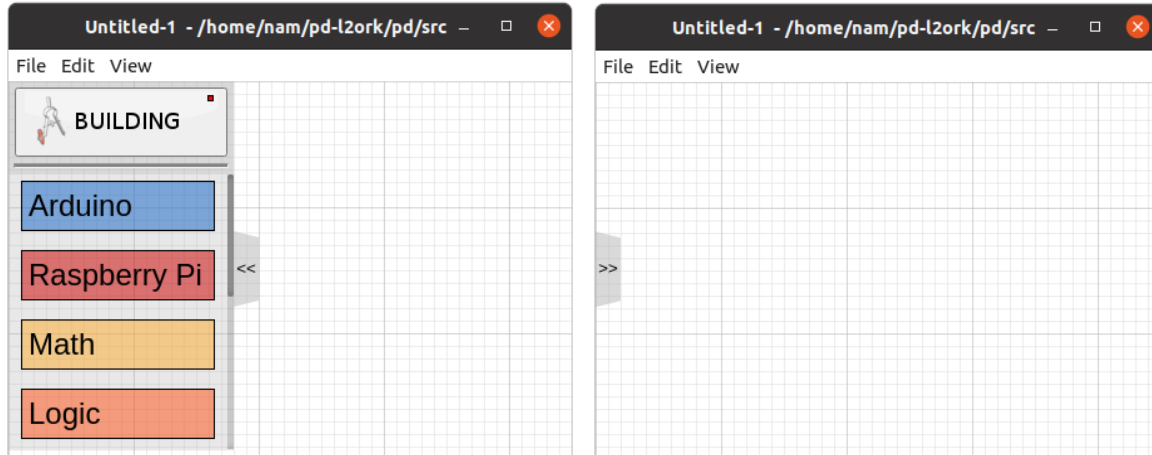


Figure 39 (left): The menu before being minimized

Figure 40 (right): The menu after being minimized. Notice the menu tab.

The K12 menu has a built-in scrollbar for navigating to different submenus (Figure 41). You can scroll in the menu using a mouse scrollwheel or by clicking and dragging the scrollbar.

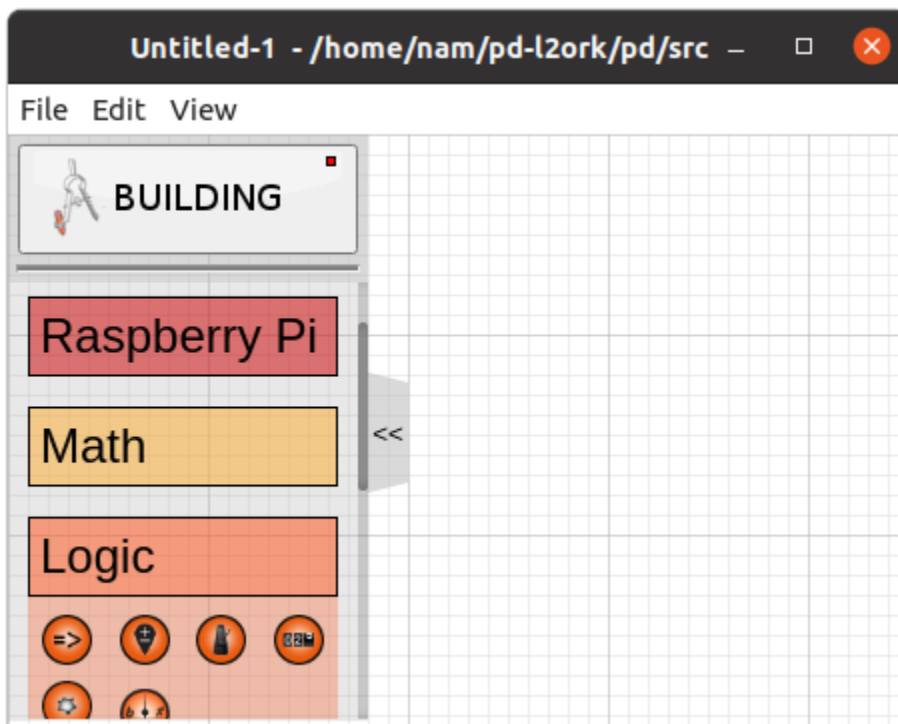


Figure 41 : Scrolling to the Logic submenu, which would otherwise be offscreen

The K12 menu allows for the expanding and collapsing of its various submenus. Click the header label of a collapsed submenu to expand it (Figure 43). Similarly, click the header label of an expanded submenu to collapse it (Figure 42).

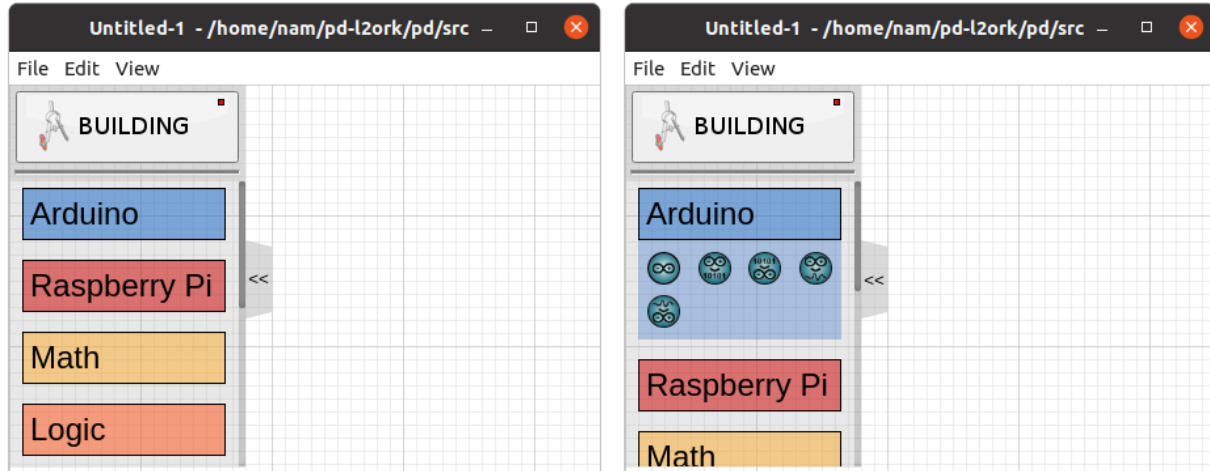


Figure 42 (left): The “Arduino” submenu collapsed
Figure 43 (right): The “Arduino” menu expanded

As shown in Figure 44, there is no limit to the number of submenus you may expand or collapse at any given time.

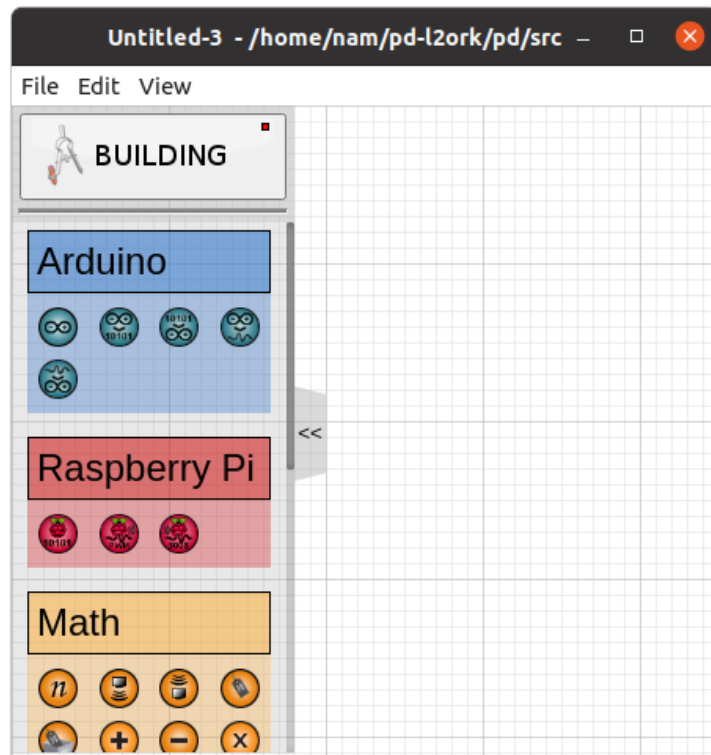


Figure 44: The “Arduino,” “Raspberry Pi,” and “Math” submenus expanded

The K12 menu has helpful tooltips that tell you more about the objects in its submenus. To display a tooltip for any given object, simply hover your mouse over the menu button for placing that object. The button will change to a lighter shade to indicate that you are hovering over it, as shown in Figure 45.

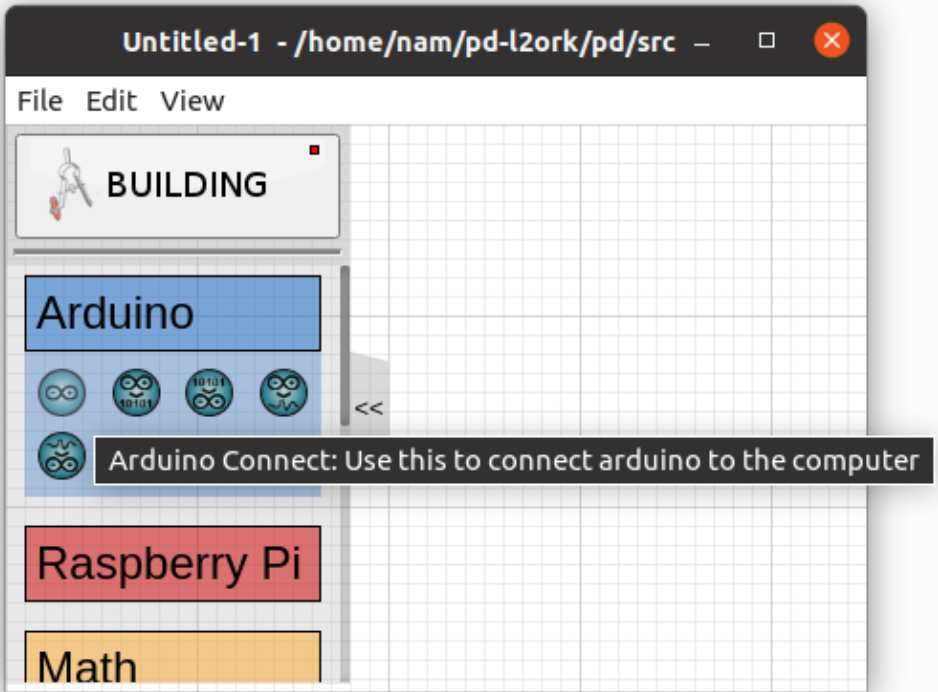


Figure 45: Tooltip for the “Arduino Connect” object. Notice that the button has a lighter shade.

The K12 menu has various objects that you can place on the canvas. To place an object from the K12 menu, navigate to the K12 submenu containing the object you would like to place (Figure 46). You may need to expand the submenu.

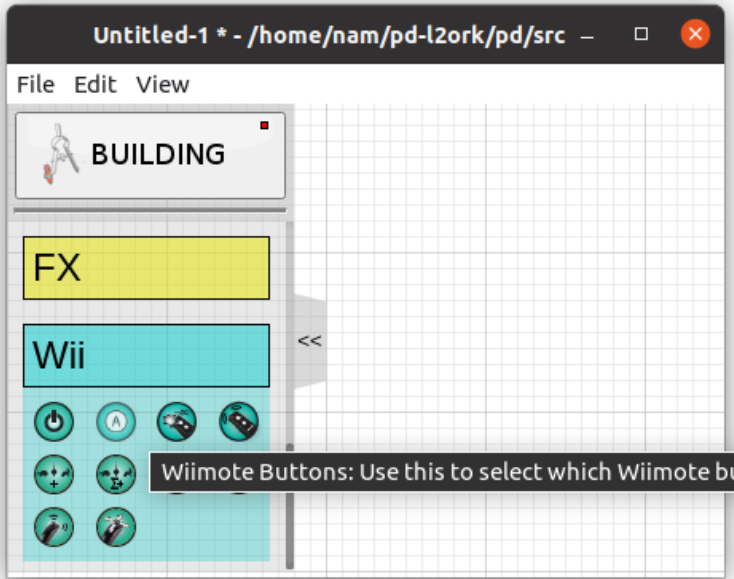


Figure 46: We wish to place the “Wiimote Buttons” object

Select the object you would like to place. An instance of the object will follow your mouse around. Move your mouse around until the object is where you would like it to be on the canvas, then click to place the object, as shown in Figure 47.

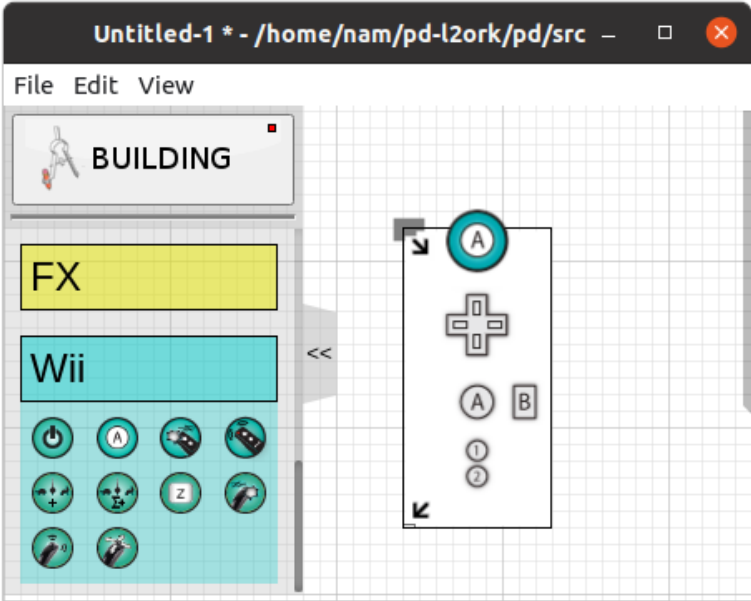


Figure 47: Placing the “Wiimote Buttons” object at a desired location

After placing the object, you may also move the object to another location on the canvas (Figure 48).

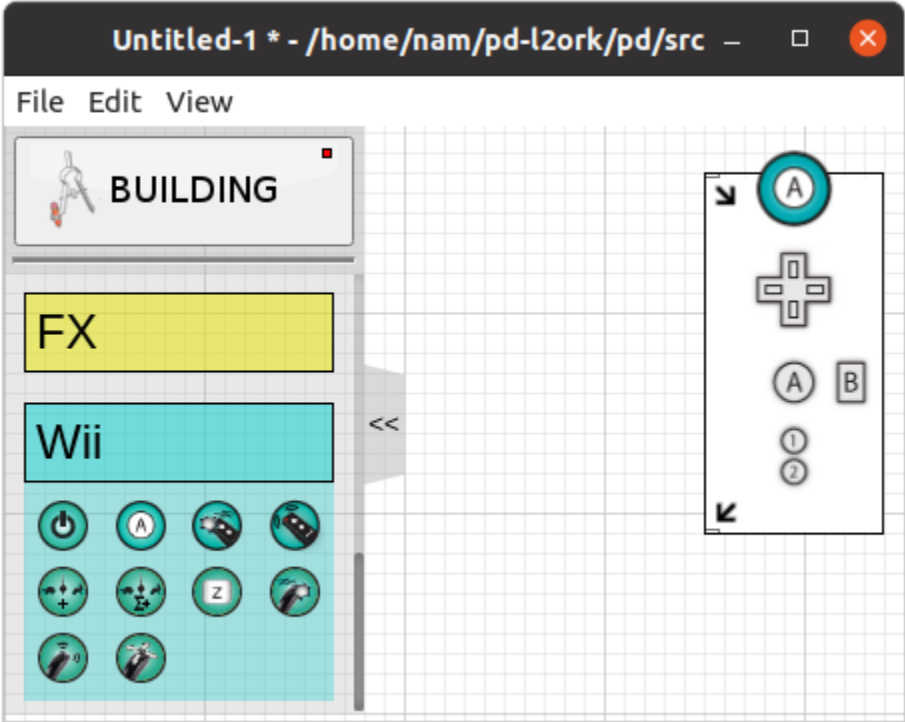


Figure 48: Moving the “Wiimote Buttons” object to a new location

9. Developer's Manual

9.1. Inventory

There are many files in the application, some of which are not relevant with our implementation, so we will only explain the files that we changed.

9.1.1. Tooltips files

Found in /pd-l2ork/pd/nw folder

- pdgui.js: JavaScript file where functionality for displaying tooltips is implemented, along with where search.index file is generated

Found in pd-l2ork/pd/src folder

- g_text.c: C file where minor changes were made to assist with displaying tooltips

Found in home/<user>/pd-l2ork folder

- search.index: JSON file where the majority of objects in Pd-L2Ork have their tooltip metadata stored.

9.1.2. K12-mode files

All of these files can be found from the nw folder.

- pdgui.js: This is the main JavaScript file of the program. It is used to provide functions that can communicate with the backend.
- pd_menus.js: This is where the menus at the top are created, as well as its submenu components.
- pd_canvas.html: This is the HTML file for a blank canvas. This is where the K12 menu is created.
- css/default.css: This is the CSS file for a blank canvas. This is where different elements of the K12 menu are styled so that it looks nice.
- pd_canvas.js: This is the JavaScript file for a blank canvas. This is where the functionality of the K12 menu is located, from making the menu appear/disappear to creating objects when the icons are clicked.
- locales/en/translation.json: This is where the descriptions of the objects are stored.

9.2. Installing Software

To install the software, simply visit the repository from Github and follow the instructions in the README. To start Pd-L2Ork, navigate to the pd/src folder and run pd-l2ork in the terminal. To make changes to the frontend of the application, copy all modified files to /usr/lib/pd-l2ork/bin and run pd-l2ork. You can chain these commands using && (e.g., cp ../nw/<file> /usr/lib/pd-l2ork/bin && pd-l2ork).

Github link: <https://github.com/pd-l2ork/pd-l2ork/>

10. Lessons Learned

Timeline:

- 9/6: Met client for the first time to set up a meeting time and learn about the project.
- 9/9: First meeting with client, set up weekly Friday meetings and formed teams. Worked towards getting Pd-L2Ork installed and familiarizing ourselves with the code.
- 9/23: All team members got Pd-L2Ork installed and started working on Tooltips and K12-mode.
- 10/14:
 - K12-mode team: Got menus to disappear and created a button to switch between modes. Started working on the K12 menu itself.
 - Tooltips team: Given assistance on where to find metadata used for a given object's tooltips
- 11/4:
 - K12-mode team: Got most of the K12 menu functionality working, started working on menu design.
 - Tooltips team: Advised by client on where to add code that would allow for tooltips to be displayed
- 11/19:
 - K12-mode team: Finished K12 menu and design, started debugging and making final touches before submitting a patch to our client.
 - Tooltips team: Implementing functionality for parsing through metadata in order to display tooltips
- 12/2 (Final meeting):
 - K12-mode team: Final version of K12-mode is complete; made some minor adjustments before giving our client a patch of our K12-mode implementation.
 - Tooltips team: Majority of functionality for displaying tooltips achieved.

Problems:

- Working with a massive project
 - Several source files, each with thousands of lines
- Figuring out how certain parts of program worked
 - Source files were too big to look at everything
 - Mediocre documentation
 - Certain parts of program had relevant functionality
- Modifying source files

- Modifying files to produce a noticeable change
- Modifying files without breaking the program

Solutions:

- Trial and error
 - Modify certain parts of program; revert if nothing happens/errors
- Get help
 - Go to client for help
 - Use Google to look up certain code snippets

Future Work:

On the tooltips side of things, the tooltips for the inlets and outlets of objects still need to be implemented, along with tooltips for objects that did not have their metadata stored within the search.index JSON file.

On the K12-mode side of things, there are a couple of things that still need to be done. First, Pd-L2Ork also supports French and German. When hovering over an object in the K12 menu, we only have a description of the object in English, so support for other languages would need to be added. Also, since we never touched the backend, the K12 flag in the backend is never updated. While this might not be an issue right now, this is definitely something to think about moving forward.

As for the project as a whole, we never got around to integrating the web browser support, although the client seemed more interested in focusing on the tooltips and K12-mode implementations. Nevertheless, it is still something that would need to be done in the future.

11. Acknowledgements

Professor Ivica Ico Bukvic - ico@vt.edu

12. References

1. Ivica Ico Bukvic, "L2Ork Linux Laptop Orchestra: No, Really, What is L2Ork?", 2022, Virginia Tech DISIS Linux Laptop Orchestra (L2Ork).

Available at: <http://l2ork.music.vt.edu/main/no-really-what-is-l2ork/>

(Accessed: November 10, 2022).

2. Wikipedia contributors, "Pure Data", 12 November 2022, Wikipedia, The Free Encyclopedia. Available at:

https://en.wikipedia.org/w/index.php?title=Pure_Data&oldid=1121392616/

(Accessed: November 15, 2022).