# SONIC TILT MANUAL: CREATE YOUR OWN VERSION OF TILTIFICATION

*Steffen Kleinert*

University of Bremen
Computer Science Department
Bremen, Germany
`stkl@uni-bremen.de`

*Tim Ziemer*

University of Bremen
Bremen Spatial Cognition Center
Bremen, Germany
`ziemer@uni-bremen.de`

## ABSTRACT

This is a guide to walk you through the necessary steps to implement your own sonification ideas in `Sonic Tilt`, the open source version of our mobile app `Tiltification`. It contains all steps necessary to build the Android app on a Windows 10 computer using Flutter in combination with Android Studio, as well as the steps necessary to replace our psychoacoustic sonification with your own sound design using Pure Data. Besides the Pure Data sound design, no programming skills are necessary. Please hand in your own version of Tiltification for the Sonification Design Competition for the International Conference on Auditory Display (ICAD) 2023, together with a manuscript that explains and/or evaluates your sound design.

## 1. INTRODUCTION

In 2020 we developed Tiltification [1], a mobile spirit level app that utilizes our psychoacoustic sonification [2] as a user interface to guide a user when leveling furniture, etc. Besides being a means for teaching and research, our goal for Tiltification was science communication. We wanted to get as many users as possible in contact with sonification. Within a year, the app was downloaded over $20,000$ times. In [3] we analyzed app store statistics, review articles and user feedback to assess the appropriateness of our measures to develop, distribute, and market this and our previous app, the CURAT sonification game [4].

A lot of positive and critical feedback about Tiltification has reached us. Some users found our sonification not intuitive, not informative, and/or not appealing. This is a justified criticism, as we simply implemented our sonification for navigation [5] as is, without adapting it for the usage as a spirit level. In the sonification scene there is a wide consensus that sounds should be developed, or at least adapted for the specific use case [6, p. 219][7].

The good thing is that sound designers, composers, artists, audio engineers and researchers exist worldwide, having all it takes to conceptualize and implement just the right sonification for a spirit level. An adequate sound design may be musically, psychoacoustically or comically motivated and deliver the optimal compromise between information density, pleasantness and intuitiveness of sound.

With the open source project `Sonic Tilt` [8] we provide sonification designers with all the necessary infrastructure to create their own version of Tiltification, with their very own sound design. Except from Pure Data, no further programming skills are required if you follow this Guide.

The theme of the International Conference on Auditory Display (ICAD) 2023 will be “Sonification for the Masses”, and as a satellite event we will organize a sonification design competition. To participate, all you need to do is hand in your own build of `Sonic Tilt` together with a paper that describes your sound design, including motivation, explanation of the mapping, explanation of the implementation and, if you find the time, an evaluation. Participants in the competition can present their Sonic Tilt version at the ICAD conference to a larger audience. The guide at hand allows you to set up your system and get ready to implement and test your very own sonification design to create your own spirit level sonification app and participate in the competition.

To get an idea of how a spirit level sonification might sound, please have a look at the psychoacoustical sonification from the original Tiltification app under `https://youtu.be/CkzQPD7VYHo` and test it yourself using the Tiltification app that can be downloaded under `https://sonification.uni-bremen.de/downloads.html`. You can also watch a demo video of our musical sonification idea for sonic tilt on `https://youtu.be/ILvFaLfs78g` and download the respective apk from the University of Bremen server to play around with it.

## 2. MATERIALS AND EQUIPMENT

To build your own version of `Sonic Tilt` you need some hardware:

- A Windows 10 computer with an internet connection and 5 GB of free space on your hard drive (Other operating systems may work, too)
- An Android smartphone (A tablet may work, too)
- A USB data cable to connect the computer and the smartphone (a “charge-only” cable will not work)

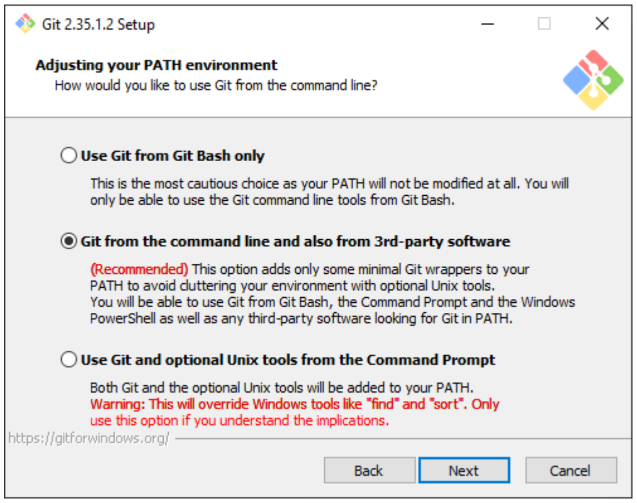
You will also need some software. We will explain in this guide where to download and how to install it:

- Git
- `Sonic Tilt`
- Flutter
- Android Studio
- Pure Data

## 3. GIT INSTALLATION

According to the official Flutter installation guide, Git is a system requirement to develop apps with Flutter. Besides that it is our choice for version control and collaborative programming and designing.

Download git from `https://git-scm.com/download/win`.



Git 2.35.1.2 Setup

**Adjusting your PATH environment**

How would you like to use Git from the command line?

○ **Use Git from Git Bash only**

This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

[✓ **Git from the command line and also from 3rd-party software**]

(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.

○ **Use Git and optional Unix tools from the Command Prompt**

Both Git and the optional Unix tools will be added to your PATH. Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

https://gitforwindows.org/

Figure 1: *Make sure to activate "Git from the command line and also from 3rd party software" during the installation*

## 4. PROJECT DOWNLOAD

Now that Git is installed, create a folder with a short path, such as `C:\tiltification` .

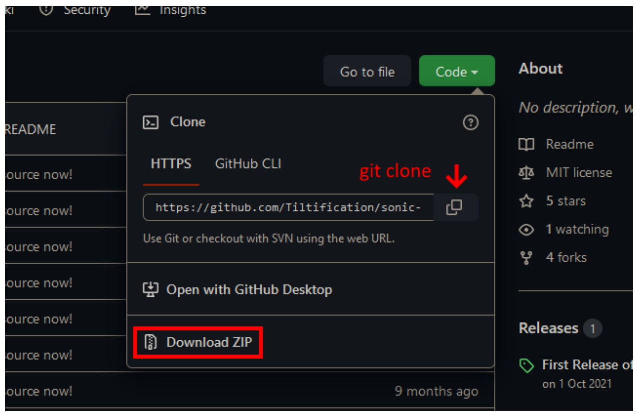
Here, you right-click and choose `Git Bash Here` to open a terminal window. In the terminal window you type `git clone https://github.com/Tiltification/sonic-tilt.git` or paste it by right-clicking on the terminal window and selecting `paste` (remember line break). Hit the return key and the repository should download and become visible as a directory called `sonic-tilt` .

Alternatively, you can download the repository as a ZIP file from `https://github.com/Tiltification/sonic-tilt/archive/refs/heads/master.zip` and extract it in `C:\tiltification` as illustrated in Fig. 2.

## 5. FLUTTER DOWNLOAD

We developed Tiltification using Flutter 1.22.0 from the 1st of October 2020. As newer versions may produce errors when compiling, we recommend installing the dated version from `https://docs.flutter.dev/development/tools/sdk/releases?tab=windows`.

Extract files to `C:\tiltification` so that the tiltification folder contains the two sub-folders `sonic-tilt` and `flutter_windows_1.22.0-stable`. This short file path will come in handy, as you will have to find and copy Flutter's file path later on.



`https://github.com/Tiltification/sonic-`

Use Git or checkout with SVN using the web URL.

Figure 2: *Clone our repository using the address from GitHub or download it's contents as a ZIP file*

## 6. ANDROID STUDIO INSTALLATION AND SETUP

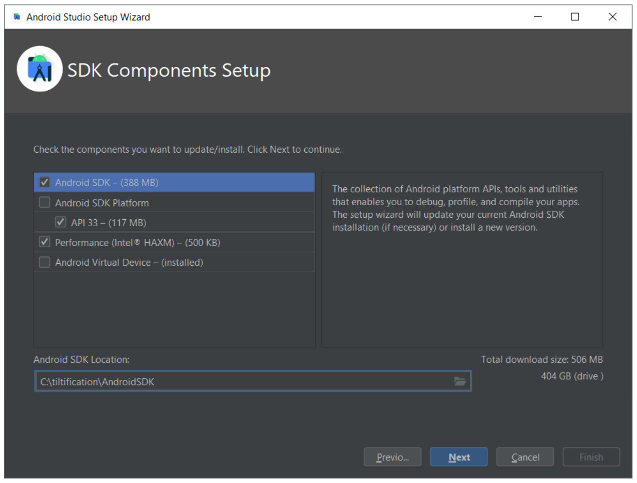
Android Studio comes along with all development kits that we need in order to build our app, and also provides us an IDE to work with. Our latest tested Android Studio version was 2021.1.1 Patch 3, but it is likely for the newest version to work just fine. It can be downloaded from `https://developer.android.com/studio/`.

After double clicking the Android Studio installer, it takes some seconds to respond. After clicking `Next` on the welcome message you will be asked to choose the components to install. Feel free to uncheck the `Android Virtual Device` to save some space, as you will be using your phone as a physical device to build on. Click `Next`.

As the installation location for Android Studio, choose a folder named `AndroidStudio` (without a space character) to `C:\tiltification` so that the `tiltification` folder contains the three sub-folders `AndroidStudio` , `sonic-tilt` and `flutter_windows_1.22.0-stable`. You can click through the next options. When clicking the final `Finish` button you can start Android Studio directly. Otherwise and in the future you can always start it from `C:\tiltification\AndroidStudio\bin\studio64.exe`.

You will need an internet connection upon the first start of Android Studio. In the first pop up window choose `do not import settings` and click `OK`. In the Android Studio Setup Wizard change the `Install Type` from `standard` to `custom` to be able to set your own path for it. You will be asked to choose the Java Development Kit (JDK) location to use, which should be the one installed with Android Studio under `C:\tiltification\AndroidStudio\jre`. After choosing the eye-friendly and energy-efficient `Darcula` theme you will get to the `SDK Components Setup`. Here, you can leave all boxes checked as they are. What we are interested in is to set the `Android SDK Location` at the bottom of the window to a new folder named `C:\tiltification\AndroidSDK` as indicated in Fig. 3 and click `next`. The last steps can be left to default. The installation wizard will finish by asking you to accept all licenses before you can click on `Finish`.

Now that the when Android Studio starts you will already find an option to install the plugins needed to work with Flutter,



Check the components you want to update/install. Click Next to continue.

- [✓] Android SDK – (388 MB)
- [ ] Android SDK Platform
  - [✓] API 33 – (117 MB)
- [✓] Performance (Intel® HAXM) – (500 KB)
- [ ] Android Virtual Device – (installed)

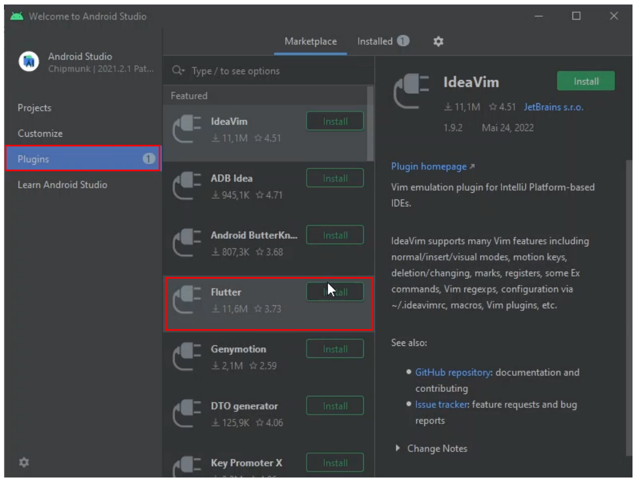
The collection of Android platform APIs, tools and utilities that enables you to debug, profile, and compile your apps. The setup wizard will update your current Android SDK installation (if necessary) or install a new version.

Android SDK Location: C:\tiltification\AndroidSDK

Total download size: 506 MB
404 GB (drive )

Figure 3: *Setting the SDK location, leaving check boxes on default*

shown in Fig. 4. If this window does not pop up, go to `File` → `Settings...` and select `Plugins` from the list in the left column. Under Plugins search the list for `flutter`, click the `Install` button next to it and accept the privacy note. A pop up window informs you about `Dart`, the next plugin to install. Click on `Install` inside that pop up window. The `Dart` plugin could otherwise also be installed over the same list like the `flutter` one.



IdeaVim — 11,1M ☆ 4.51 — JetBrains s.r.o. — 1.9.2 — Mai 24, 2022

Vim emulation plugin for IntelliJ Platform-based IDEs.

IdeaVim supports many Vim features including normal/insert/visual modes, motion keys, deletion/changing, marks, registers, some Ex commands, Vim regexps, configuration via ~/.ideavimrc, macros, Vim plugins, etc.

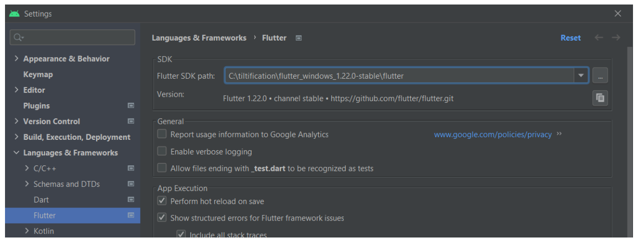
- GitHub repository: documentation and contributing
- Issue tracker: feature requests and bug reports

Figure 4: *Installing the Flutter plugin before opening the project*

After some installation progress the button next to Flutter should change to `Restart IDE`. Click it to finish the plugin installations.

Now that the plugins are installed you can click the folder icon `Open` to search your machine for our `sonic-tilt` project. It should exhibit an Android face icon (), as Android Studio recognizes it as an Android Studio project. Open it, click `Trust Project` if asked, and give the project some time to load. If your firewall pops up, allow `adb.exe` to access the network. Even though the interface is visible early, any inputs will be very laggy until the last progress bars at the bottom stops moving.

To connect Android Studio to the Flutter SDK, go to `File` → `Settings...` found in the menu bar at the top of Android Studio's main window. A menu opens as illustrated in Fig. 5. Inside the Settings menu double click on `Languages & Frameworks` where you can find `Dart` and `Flutter`. Click on `Flutter` and you will see the option to enter the path to your Flutter installation. Enter the path to where you unpacked Flutter into (the last folder of the path should be named `flutter` in all lowercase, unless you renamed things) and click `OK` to get back to the `Settings` menu.



Languages & Frameworks › Flutter

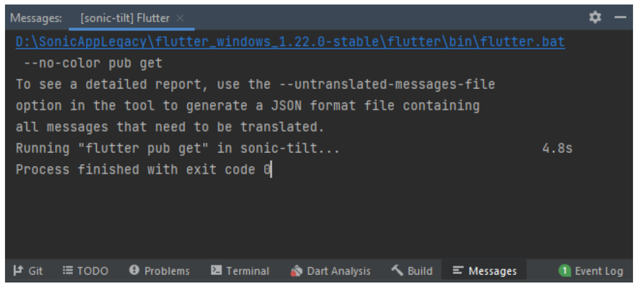
SDK
Flutter SDK path: C:\tiltification\flutter_windows_1.22.0-stable\flutter
Version: Flutter 1.22.0 • channel stable • https://github.com/flutter/flutter.git

General
- [ ] Report usage information to Google Analytics www.google.com/policies/privacy
- [ ] Enable verbose logging
- [ ] Allow files ending with **_test.dart** to be recognized as tests

App Execution
- [✓] Perform hot reload on save
- [✓] Show structured errors for Flutter framework issues
  - [✓] Include all stack traces

Figure 5: *Selecting the path to Flutter inside Android Studio's Settings window*

After clicking `OK`, you might already see a console at the bottom of Android Studio's main window become active. This can take up to 1 minute. Incompatible Flutter version will now display red complains. But if the message ends on `Process finished with exit code 0`, as illustrated in Fig. 6, everything is set. Don't worry if you see no messages. As soon as you build the project, the routines will be called automatically.

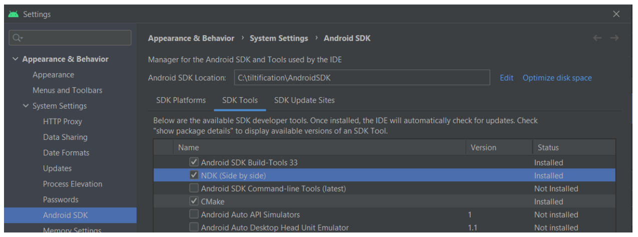


```
D:\SonicAppLegacy\flutter_windows_1.22.0-stable\flutter\bin\flutter.bat
 --no-color pub get
To see a detailed report, use the --untranslated-messages-file
option in the tool to generate a JSON format file containing
all messages that need to be translated.
Running "flutter pub get" in sonic-tilt...                           4.8s
Process finished with exit code 0
```

Figure 6: *"flutter pub get" signaling that the used Flutter version works with the project*

Another crucial part to make your own Pure Data sonifications usable is to install the Android NDK, as we are using code that isn't supported by just the normal Android SDK. At the top under `Tools` you need to click on the `SDK Manager`. A window as illustrated in Fig. 7 pops up. Here, you need to select the tab `SDK Tools`. This should bring up a big list of tools available to install. What we need are `NDK (Side by side)` and `CMake`, so check the boxes next to them and click `OK` at the bottom right.

For some reason Android Studio doesn't always stick to it's own naming conventions. The NDK you just installed should now lie in a folder structure `ndk/some.version.number/` inside the directory you installed the Android SDK into earlier. For Android Studio to find it during a build process however, you sometimes need to rename that `ndk` folder to `ndk-bundle` and move all the content inside the folder named by the NDK's version number up into the renamed `ndk-bundle` folder directly. So try this



Manager for the Android SDK and Tools used by the IDE

Android SDK Location: C:\tiltification\AndroidSDK

Below are the available SDK developer tools. Once installed, the IDE will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

| Name | Version | Status |
|---|---|---|
| ✓ Android SDK Build-Tools 33 | | Installed |
| ✓ NDK (Side by side) | | Installed |
| Android SDK Command-line Tools (latest) | | Not Installed |
| ✓ CMake | | Installed |
| Android Auto API Simulators | 1 | Not installed |
| Android Auto Desktop Head Unit Emulator | 1.1 | Not installed |

Figure 7: *Make sure to set the Android SDK location, install NDK (Side by side) and CMake*

renaming if errors around the NDK occur when walking through the next section about building the app or go to the troubleshooting section 10.4 directly for more help.

Flutter needs to be informed explicitly where it can find the Android SDK. Go to the `Terminal` tab in Android Studio. It can be found on the bottom of the window in the same line as `Git`, `TODO`, `Problems`, (sometimes `Messages`) and `Dart Analysis`.
This opens a terminal window right above the tab. Here, write `C:\tiltification\flutter_windows_1.22.0-stable\flutter\bin\flutter.bat config --android-sdk C:\tiltification\AndroidSDK\ .`
This tells Flutter where it can find the Android SDK. After pressing `[Return]`, it should only take a second or so and then the text `You may need to restart any open editors for them to read new settings.` should be displayed in the terminal window. Do so, exit Android Studio and open it again.

## 7. FIRST TEST BUILD

To be able to run a test build on your Android device, you first have to go into it's settings and enable the developer mode to make debugging on it possible. This is done differently in every Android version, and you may look for a detailed description on the official Android Website.

In principle, you need to go to `Settings` and scroll all the way down to `About phone`. Here, you scroll down to `Build number` and tap it 7 times. Then you need to type in your password. This unlocks the developer options. Now go to `System`, tap on `Advanced`, look for `USB debugging` and activate it. Sometimes, this option is hidden under `Developer options`.

After connecting the phone via USB cable to your PC with Android Studio open, your phone now asks you to allow debugging. Allow it (you might have to allow it multiple times) and your phone should load as a device at the top of Android Studio near a green `play` button 8.

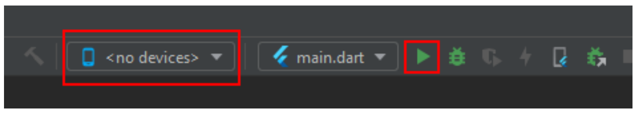


Figure 8: *Left rectangle: where the phone shows up when connected; Right rectangle: the button to create a test build on the selected phone*

If everything went right you are now ready to click said green button to create a test build of the app on your phone. Especially on the first build you can expect it to load for several minutes. But unless the process stops you do not have to worry about any warnings on the console. And in case it does unfortunately stop on an error, you should check our troubleshooting section 10.4, as it is likely a development kit not being found.

Once built, the app will start directly on your phone. From that point on it is an actual new app called `Sonic Tilt` on your phone, having the Sonic Tilt icon:

Note that you can unplug your phone, close the app and reopen it from your phone's app menu at any time. There are no conflicts in case you have our original app `Tiltification` installed already. This is an independent app now, that will update every time you make changes to the Android Studio project and rebuild it. You can use and familiarize with the app and its functions, and explore how the psychoacoustic sonification responds to the two tilt angles. We created a YouTube playlist introducing the app, showing some use cases, and explaining the buttons.

Then, it is about time to replace the sound by your own sonification idea. The following section will explain how.

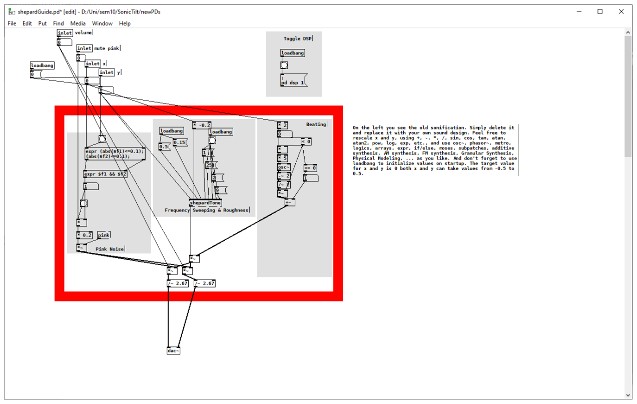
## 8. PURE DATA

The sonification of our app is designed using Pure Data. Download Pure Data from `https://puredata.info/downloads`, install it, and associate `.pd` files with it. Your Pure Data version (like Vanilla or Extended) does not matter at this point as it is just an editor for you to work on pd patches and not part of the build process.

What matters is to know the correct paths to the files you want to edit with Pure Data. For Android builds this is `C:\tiltification\sonic-tilt\android\app\src\main\res\raw`, where all Pure Data files can be found in a ZIP file called `streamingassets.zip`. Extract all files to `C:\tiltification\sonic-tilt\android\app\src\main\res\raw`, which should produce a folder named `streamingassets`. If so, cut the ZIP file and paste it to `C:\tiltification\streamingassets.zip` as a backup.

In the `streamingassets` folder (not the zip file) the `shepardGuide.pd` is your entry point. Open this file with Pure Data. You will see several inlets, some wild routing and subpatches, and the `[dac~]` output. Delete the wild routing and subpatches and replace it with your own sonification. As indicated in Fig. 9, the parts to delete are marked by a red rectangle and a comment on the right will give you some hints on how to start implementing some sound designs yourself. The inlet labeled `sound` is the volume control that can take values from 0 to 1. The inlet `pink` is either 0 or 1. It tells you whether the smartphone is tilted by more than $3^\circ$ or not. Tiltification uses this information to (un-)mute pink noise as a confirmation to the user that the mobile phone is almost leveled. The inlet `tarX` is the tilt angle along the left-right dimension. Here, angles from $-45^\circ$ to $45^\circ$ are linearly scaled to values from $-0.5$ to $0.5$, telling you by how much the smartphone is tilted to the left or right. Angles that have a larger absolute value than $45^\circ$ will still be transferred to `tarX` as $-0.5$ or $0.5$, respectively. The inlet `tarY` is the tilt angle along the front-back dimension, again, with angles between $-45^\circ$ and $-45^\circ$ scaled to values between $-0.5$ and $0.5$. It tells you by how much the smartphone is tilted towards or away from you. Again, larger tilt angles will still send a $-0.5$ or $0.5$ to `tarY`.

These are all inputs that your sonification needs. The output of your sonification has to be routed to both inlets of `[dac~]`. Even though you can use stereo, we recommend to stick to mono, i.e., route the same signal to the left and the right inlet of `[dac~]`. Furthermore, all important auditory information should be contained in the frequency region between 200 Hz and 4 kHz. This is because many smartphones only have a mono tweeter, and most users will certainly not use headphones while leveling furniture or alike. Consequently, stereo information and frequencies outside the bandwidth of the tweeter will be lost.


shepardGuide.pd* [edit] - D:/Uni/sem10/SonicTilt/newPDs
File Edit Put Find Media Window Help
Toggle DSP
loadbang
Beating
shepardTone
Frequency Sweeping & Roughness
Pink Noise
dac~

On the left you see the old sonification. Simply delete it and replace it with your own sound design. Feel free to rescale x and y, using +, -, *, /, sin, cos, tan, atan, atan2, pow, log, exp, etc., and use osc~, phasor~, metro, logics, arrays, expr, if/else, moses, subpatches, additive synthesis, AM synthesis, FM synthesis, Granular Synthesis, Physical Modeling, ... as you like. And don't forget to use loadbang to initialize values on startup. The target value for x and y is 0 both x and y can take values from -0.5 to 0.5.

Figure 9: *The shepardGuide.pd contains our psychoacoustic sonification that you can replace by your own sonification approach*

The objective of the user will be to reach a value of 0 for `tarX` and `tarY`, and your responsibility as a sonification designer is to guide the user there. If you use any externals, please copy the respective PD files into the streamingassets folder.

To test your sonification, open `receiverLibPD.pd`. This patch opens the `shepardGuide.pd` as a subpatch. In `receiverLibPD.pd` you can manually manipulate `tarX` using the green, horizontal slider, `tarY` using the blue, vertical slider, `pink` (un-)checking the toggle box, and `volume` using the white, vertical slider as illustrated in Fig. 10.

To test your sonification inside the `Sonic Tilt` app on your mobile phone, use the Windows explorer, navigate to `C:\tiltification\sonic-tilt\android\app\src\main\res\raw\streamingassets`, right-click and choose `New` → `Compressed (zipped) folder` that you name `streamingassets.zip`. Then, copy all files from the folder to the ZIP file. When finished, move the file to `C:\tiltification\sonic-tilt\android\app\src\main\res\raw`. This way of zipping files seems unnecessarily complicated. But if you simply zip the complete folder directly, the ZIP file will not contain all files directly, but within a subfolder, in which case the file structure is wrong and the built app will stay silent.

With your phone connected via USB (and USB debugging enabled) it is now just another click on the green button in Android Studio (remember Fig. 8) to build the app with your new sonification. App builds may take a minute. After the successful build, the `Sonic tilt` app opens automatically on your android phone.

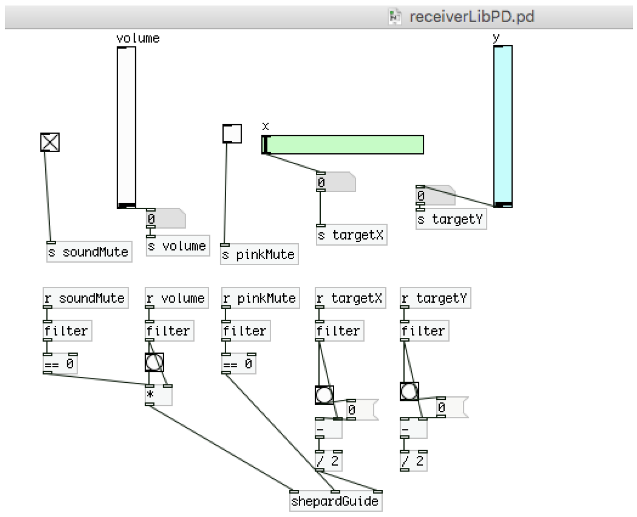


Figure 10: *The receiverLibPD.pd file that allows you to manipulate all variables that Tiltification will pass to your sonification*

## 9. APP BUILD

While you develop your own bullseye spirit level sonification in `Sonic Tilt`, you will make changes in the Pure Data files over and over, create the `streamingassets.zip` file, run the app on your smartphone and explore how the sonification responds to your actions. Note that in the `Sonic Tilt` menu there is an option to use the spirit level only in one dimension. This option will set `tarY` to 0 and the only variables controlled by the smartphone sensors are `tarX`, `pink` and `mute`.

When you are happy with the sound inside the app, and its response to the smartphone orientation and motion, you should build an `APK`. To do so, navigate to `Build` → `Flutter` → `Build APK`. The compilation may take a minute. You can find the `APK` file under `C:\tiltification\sonic-tilt\build\app\outputs\flutter-apk\app.apk`. Note that the app you create is yours. You can use it and even publish it, as long as you do this under an appropriate license, like the MIT license, and mention that you created this app using Flutter and Sonic Tilt. Please refer to our open source Git project [8] and our Tiltification paper [1] wherever you present your Sonic-Tilt app. Note that you are also free to modify other parts of `Sonic Tilt`, such as the Graphical User Interface (GUI). You are free to include menu sounds or a screen reader, apply additional signal processing on the sensors, include the compass as a third dimension, develop a sensor calibration routine or whatever you like. We even recommend you to replace our app icon by your own design. This app icon is a `PNG` file with a resolution of 2000 times 2000 pixels that you can find under `C:\tiltification\sonic-tilt\assets\logo.png`.

Now it is time for you to prepare a documentation of your sonification and submit it together with the `APK` file of your app for the sonification design competition at ICAD 2023. Before you hand them in, please make sure that the `APK` file contains your latest changes: Uninstall `Sonic Tilt` from your smartphone and use the `Windows Explorer` to copy the `APK` file from your computer to your smartphone. On your smartphone, open the `APK` file to install the app. If prompted, toggle `Allow from this` `source`. You may need to allow `APK` installs in your Android settings first. Depending on your Android version this can be done under `Settings` → `Apps` → ⋮ → `Special access` → `Install unknown apps` or under `Settings` → `Apps & notifications` → `Advanced` → `Special app access` → `Install unknown apps`.

## 10. TROUBLESHOOTING

Even when you strictly follow our instructions, issues can occur. Here are some common problems and potential solutions:

### 10.1. No Device in Android Studio

To see whether Android Studio recognizes your smartphone, click on the Device Manager icon that is highlighted as number 3 in Fig. 11. Here, choose the tab `Physical`. This should show a list of all devices connected so far. The first column shows the device name, the second shows the API, which can be translated into the respective Android version via a table in Wikipedia, the third shows the connection type. Your currently connected device should have the USB icon ( ) in the third column. If not, there are four potential solutions:

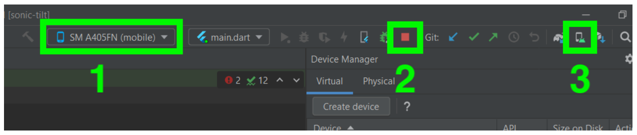


Figure 11: *Screenshot of Android Studio showing that Flutter recognizes the connected device (1), the* `Stop` *button (2) that turns red while Flutter is trying to run or debug an app, and Android Studio's* `Device Manager` *that should show the connected device under the* `Physical` *tab*

1. Make sure you have enabled the developer mode and activated USB debugging on your smartphone.

2. On some smartphones you have to allow USB debugging over and over. So unlock your smartphone and check, whether it asks you to allow USB debugging and confirm. If your are not getting asked, try disabling and then enabling USB debugging.

3. Some USB cables are "charge only" cables that will not transfer data. Make sure to connect your phone with a "data cable". As soon as you connect your phone via data cable, Android should ask you to allow USB debugging from the connected device, which you should confirm.

4. Sometimes, drivers are corrupted or dated. Download and install the latest drivers for your smartphone.

### 10.2. Device Recognized by Android Studio but not by Flutter

In Fig. 11 you see how it looks like when Flutter recognizes the connected smartphone. If the dropdown menu (1 in Fig. 11) states `<no devices>`, even though the `Device manager` in Android Studio lists your smartphone as being connected via USB, some approaches may solve this issue:

1. Sometimes, Flutter takes quite a while to recognize connected devices. You should be patient and wait for two minutes for the connected device to be recognized. You should also try and click on the dropdown list 1 in Fig. 11, click `Refresh` and wait for another two minutes.

2. Unexpected error may occur, but the Flutter Doctor is there to help. In Android Studio, navigate to `Tools` → `Flutter` → `Flutter Doctor`. This starts Flutter's internal self-test. In the `Messages` window categories with issues are marked with a `[!]`, the concrete problem is marked with an `X`. For some issues, potential solutions are described and links for further reading are provided.

### 10.3. Problems with Pure Data

Sometimes it looks as if the `ShepardGuide.pd` file was empty. In this case, make sure to scroll to the upper left corner of the Pure Data window using the two scroll bars.

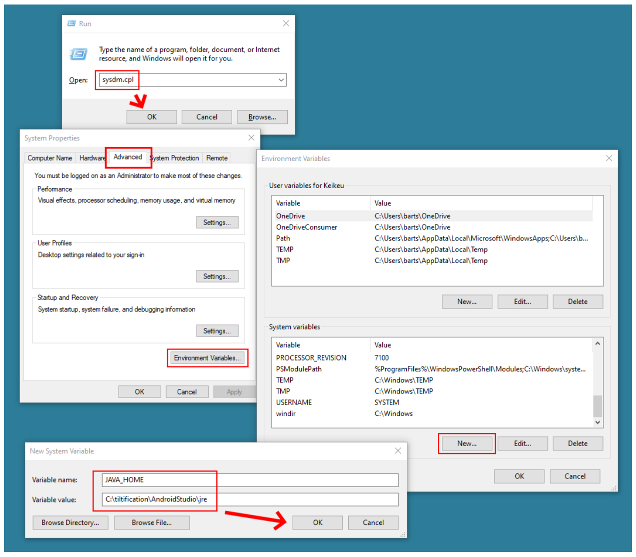
### 10.4. Failing to Build

If your app does not build, this can have many reasons:

#### *10.4.1. Environment Variables*

Most errors in the build process occur, because Android Studio doesn't find the correct paths to it's development kits. For the NDK we already suggested two different paths to put it in, but the most robust solution for all the development kits is to set an *environment variable* for them in the operating system.

The fastest way to do this is illustrated in Fig. 12. Use the Windows shortcut `[Win] + [R]`, type in `sysdm.cpl` and click `OK` to get to the System Properties window. Here, select the tab `Advanced`. In that Tab click on `Environment Variables...` in the bottom right corner. Here you can click on `New...` either just for your account or for all accounts on your machine. Environment variables need a name like `JAVA_HOME` and a value to point to like `C:\tiltification\AndroidStudio\jre`. Click `OK` to save the environment variable.



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: sysdm.cpl

You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your sign-in

System startup, system failure, and debugging information

User variables for Keikeu

| Variable | Value |
|---|---|
| OneDrive | C:\Users\barts\OneDrive |
| OneDriveConsumer | C:\Users\barts\OneDrive |
| Path | C:\Users\barts\AppData\Local\Microsoft\WindowsApps;C:\Users\b... |
| TEMP | C:\Users\barts\AppData\Local\Temp |
| TMP | C:\Users\barts\AppData\Local\Temp |

System variables

| Variable | Value |
|---|---|
| PROCESSOR_REVISION | 7100 |
| PSModulePath | %ProgramFiles%\WindowsPowerShell\Modules;C:\Windows\syste... |
| TEMP | C:\Windows\TEMP |
| TMP | C:\Windows\TEMP |
| USERNAME | SYSTEM |
| windir | C:\Windows |

Variable name: JAVA_HOME

Variable value: C:\tiltification\AndroidStudio\jre

Figure 12: *Setting environment variables on Windows*

With environment variables like this, programs like Android Studio can easily find what they are looking for. Make sure to restart Andoid Studio after setting them, for it to register the newly set environment variables.

In case environment variable are a possible solution, the error messages down in the `Run`-console of Android Studio will mention their names. Below are examples of environment variables often needed for Android Studio to find everything important:

- `ANDROID_NDK_HOME` is the environment variable name to set the NDK location with. This value should become `C:\tiltification\AndroidSDK\ndk-bundle\` according to our naming conventions.
- `JAVA_HOME` is the environment variable name to set the JDK location with. This value should be set to the Java Development Kit that came with Android Studio. According to our naming conventions that value would be `C:\tiltification\AndroidStudio\jre`.

If that does not work, here is another way to set environment variables as a plan b: Press the `[Win]` key, type `cmd`, right-click on `command promt` and click `run as administrator`. Confirm that you want to run it as administrator and enter your admin password if asked. A terminal window pops up.

Here, type `setx -m JAVA_HOME C:\tiltification\AndroidStudio\jre\` or copy it from this manuscript, right-click on the terminal window to paste it, and delete any line breaks that may appear. The terminal should print `SUCCESS: Specified value was saved.` as an output.

#### 10.4.2. *Flutter and Dart*

Make sure you downloaded Flutter 1.22.0. We have also tested Flutter 1.22.4 successfully. But other versions may not work, especially Flutter 2 and newer versions. Make sure you enabled the plugins for `Flutter` and `Dart` in Android Studio under `File` → `Settings` → `Plugins`. Make sure you entered `C:\tiltification\AndroidSDK` as the Android SDK location under `Tools` → `SDK Manager`.

#### 10.4.3. *streamingassets*

Flutter does not like unexpected files inside the project. Make sure to create the `streamingassets.zip` file and delete the `streamingassets` folder and all its containing files before you run the app.

### 10.5. No Audio in App

Naturally, you should ensure that your phone is not muted. Keep the "volume up" button on the side of your smartphone pressed for several seconds to maximize the smartphone volume.

Apart from that we identified four common reasons for the `Sonic Tilt` app to stay silent:

#### 10.5.1. *The Mute Button*

On default the built `Sonic Tilt` app will be set to mute and you will have to tap the loudspeaker icon at the bottom left of the app to unmute it. If unmuted, the loudspeaker symbol is not crossed out anymore.

#### 10.5.2. *streamingassets.zip*

Firstly, it can easily happen that your latest changes of `shepardGuide.pd` are only contained in the `streamingassets` folder and not yet in the ZIP file. Make sure to create a new `streamingassets.zip` file every time you want to test a new build.

Secondly, it is easy to mess up the folder structure of the `streamingassets.zip` file. Make sure the Pure Data files are directly inside that zip and not in another folder beneath it or they won't be found.

#### 10.5.3. *Incompatible Device*

Download and run Tiltification from the Google Play Store or the University of Bremen Server. If the sound doesn't work either, `Sonic Tilt` may be incompatible with your device. In this case you need to find another smartphone or tablet to test your app.

#### 10.5.4. *Erroneous shepardGuide.pd*

Maybe your Pure Data file(s) contain an error. Try replacing your sonification by a simple pure tone that is constantly playing by putting an object `[osc~ 440]` in your file and connecting its outlet to the `[dac~]` inlets, save the file, zip all files and compile a new test build. It this works, try to debug your Pure Data files.

### 10.6. Audio Artefacts in the App

Audio artefacts can have several causes.

#### 10.6.1. *Clipping*

It is important to know that the audio output in Pure Data only allows values from $-1$ to $1$. This is true for the sum of all inputs to the `[dac~]`. All values outside this range will cause clipping, which may sound as distortion, crackling, clicks or noise. Try connecting every audio to an object `[*~ 0.01]`, route that to the `[dac~]` and listen carefully, if the artefacts are gone.

#### 10.6.2. *Unknown Pure Data Externals*

`Sonic Tilt` uses libpd to use Pure Data as a sound engine on Android. It is possible that libpd does not know all the externals that you installed on your computer. To be on the safe side, install Pure Data without any additional externals. Every object that does not work with the native install of Pure Data should be added to the `streamingassets.zip` to make sure that `iemlib` can call it.

For example, to use the `[pink~]` object from the `iemlib` external, you should either copy the `pink~.pd` file to `streamingassets.zip`, or you create your own `pink~.pd` file that carries out all the expected operations or signal processing.

#### 10.6.3. *Computational Demands*

Smartphones tend to have a quite high computational power. But if your Pure Data files contain a lot of real-time signal processing, like convolutions, Fourier transforms, and dozens of oscillators, this may be too much for some smartphones, and they fail processing it in real time, which leads to audible artefacts. In this case try to reduce the signal processing demands of your sonification.

#### *10.6.4. Errors in Pure Data Signal Chain*

Signal processing in Pure Data can produce many artefacts. Sometimes, no artefacts occur when you change values using sliders, especially because you slowly modify one value after another. But then, when you use the smartphone sensors, several values may alter at once, and the alterations may be much faster and contain more jitter as compared to using a slider. Use the app in one-dimensional mode to see whether the artefact still occurs. If not, `tarY` may cause the artefact. Try tilting the smartphone slowly to check whether fast motions may cause the artefact. Tilt the smartphone near 3° to hear whether the `[pink]` object causes the artefact. Use the app with headphones and listen carefully. If the artefacts are gone, low frequencies may cause them. These frequencies are practically inaudible when played via the internal smartphone speakers. Nonetheless, they deflect the speaker membrane and, together with the audible sounds, may produce artefacts. In the case of low frequency artefacts, either try to eliminate the cause of low frequencies in your signal processing, or add a high pass filter to the signal chain.

Note, for example, that the `[osc~]` object in Pure Data is a cosine function that can produce an audible click when switched on and/or off. This may also be true for other oscillators. Here, a ramp at the onset and offset may help.

## 11. CONCLUSION

This paper is an instruction manual to use and modify `Sonic Tilt`, our open source bulseye spirit level sonification app. As you can see there are a good amount of steps needed to successfully work with our open source repository. But once everything is installed there are just a few edits in Pure Data files needed to let the app display a whole new sonification of yours. No other programming skills are required. If you managed to implement your own sonification in `Sonic Tilt`, we recommend you to take part in the ICAD 2023 sonification design competition. You can also use the app for teaching purposes, as a demonstrator for science communication or as as interactive arts project. You may even release the app.

## 12. FURTHER READING

After following our instructions, you should be ready to go. However, if you have never user Pure Data before, you should take two weeks and learn it. [9] teaches you the use of Pure Data from scratch in a number of tutorial videos, all including comprehensive examples. For those who prefer reading books over watching videos, [10] and [11] contain all the necessary information to learn Pure Data. Note that Pure Data is very similar to Max (a.k.a. Max/MSP/Jitter), so Max users should readily implement their own Pure Data sonification.

## Acknowledgments

We thank the students from the Master's Project "Sonification Apps": Fida Ahmadi, Malte Asendorf, Ahmed Awadin, Debaditya Bhowmik, Jiumeng Chen, Kelly Huynh, Navid Mirzayousef Jadid, Hasanur Jaman, Moritz Kienzle, Jatawan Kruesilp, Ying Ying Lee, Wei Luo, Varun Raval, Rachel Ringe, Eve Emily Sophie Schade, Kashish Sharma, Xin Wang, Colin Weber and Helena Winkler.

We thank Stephen Barrass who had the idea of the `Sonic Tilt` sonification design competition, Myounghoon "Philart" Jeon who gave green light from the ICAD board as well as Niklas Rönnberg who offered to include the `Sonic Tilt` competition as a satellite event at ICAD 2023.

Last but not least we celebrate every contribution for the Sonic Tilt Competition and every app demo and release that is based on our open source project.

## 13. REFERENCES

[1] M. Asendorf, M. Kienzle, R. Ringe, F. Ahmadi, D. Bhowmik, J. Chen, K. Huynh, S. Kleinert, J. Kruesilp, Y. Lee, X. Wang, W. Luo, N. Jadid, A. Awadin, V. Raval, E. Schade, H. Jaman, K. Sharma, C. Weber, H. Winkler, and T. Ziemer, "Tiltification — an accessible app to popularize sonification," in *Proc. 26th International Conference on Auditory Display (ICAD2021)*, Virtual Conference, Jun. 2021. doi: 10.21785/icad2021.025 pp. 184–191.

[2] T. Ziemer and H. Schultheis, "Psychoacoustical signal processing for three-dimensional sonification," in *25th International Conference on Auditory Displays (ICAD2019)*, Newcastle, Jun. 2019. doi: 10.21785/icad2019.018 pp. 277–284.

[3] T. Ziemer and N. M. Jadid, "Recommendations to develop, distribute and market sonification apps," in *The 27th International Conference on Auditory Display (ICAD 2022)*, Virtual Conference, Jun. 2022. [Online]. Available: https://icad2022.icad.org/wp-content/uploads/2022/06/ICAD2022_3.pdf

[4] T. Ziemer and H. Schultheis, "The CURAT sonification game: Gamification for remote sonification evaluation," in *26th International Conference on Auditory Display (ICAD2021)*, Virtual conference, Jun. 2021. doi: 10.21785/icad2021.026 pp. 233–240.

[5] T. Ziemer and H. Schultheis, "A psychoacoustic auditory display for navigation," in *24th International Conference on Auditory Displays (ICAD2018)*, Houghton, MI, June 2018. doi: 10.21785/icad2018.007 pp. 136–144.

[6] G. Kramer, "Some organizing principles for representing data with sound," in *Auditory Display: Sonification, Audification, and Auditory Interfaces*, ser. Santa Fe Studies in the Science of Complexity, Proc. Vol. XVIII, G. Kramer, Ed. Reading, MA: Addison-Wesley, 1994, pp. 185–221.

[7] D. Verona and S. C. Peres, "A comparison between the efficacy of task-based vs. data-based semg sonification designs," in *The 23rd International Conference on Auditory Display (ICAD 2017)*, 2017, pp. 49–56. [Online]. Available: http://hdl.handle.net/1853/58380

[8] M. Asendorf, M. Kienzle, R. Ringe, F. Ahmadi, D. Bhowmik, J. Chen, K. Hyunh, S. Kleinert, J. Kruesilp, X. Wang, Y. Y. Lin, W. Luo, N. Mirzayousef Jadid, A. Awadin, V. Raval, E. E. S. Schade, H. Jaman, K. Sharma, C. Weber, H. Winkler, and T. Ziemer, "Tiltification/sonic-tilt: First release of sonic tilt," in *Git Repository*, 2021. doi: 10.5281/zenodo.5543983

[9] R. Hernandez, "cheetomoskeeto / puredata," in *https://github.com/cheetomoskeeto/PureData*, 2016. [Online].

Available: https://github.com/cheetomoskeeto/PureData (Accessed 2022-06-25).

[10] T. Hillerson, *Programmimg Sound With Pure Data*. Dallas, TX, USA: The Pragmatic Bookshelf, 2014. ISBN 978-1-93778-566-6

[11] P. Brinkmann, *Making Musical Apps*. Sebastopol, CA, USA: O'Reilly, 2012. ISBN 978-1-449-31490-3