RoboyXylophoneRecording Documentation

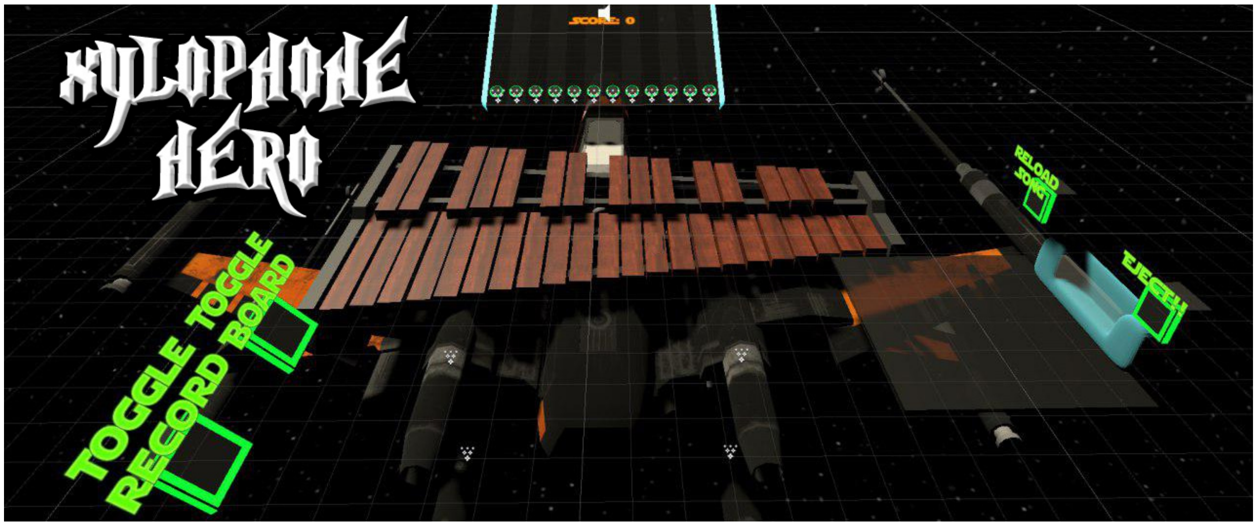
Arash, Ludwig, Yupei

Sep 16, 2018

Contents

CHAPTER 1

Introduction



1.1 What is Xylophone Recording?

This project is about capturing somebody playing xylophone in virtual reality. Processing the recording and then sending it to the the Roboy, so that the Roboy can play the same notes.

CHAPTER 2

Getting Started

This section explains how to install the project and later on which steps have to be done when you want to execute the program.

2.1 Installation

2.1.1 Requirements

We suggest to use Windows as a development system as the used HTC Vive is officially just supported on Windows. Nevertheless it's possible to run the Unity project without the HTC Vive for debugging purposes. This is also possible with other operating systems like Mac OSX.

2.1.2 Part 1: Install Unity

The Xylophone Recording Project is developped and tested in Unity. Take a look at the **ProjectSettings/ProjectVersion.txt** file to see what Version of Unity is currently used for the Project. Any Versions later than this should be compatible with this project. Using earlier versions should be avoided, because it may cause "missing prefab" and other problems.

Specific Unity Versions can be downloaded here: Unity Download Archive

Select the correct version and download the installer. After downloading, run the installer and follow the instructions to install Unity.

2.1.3 Part 2: Setup HTC Vive and SteamVR

Follow the official SteamVR HTC Vive PRE installation Guide to set up HTC Vive and SteamVR.

2.1.4 Part 3: Clone the project from Github

Use the following command in Git Bash or your preffered command line to clone the Xylophone Recording repository from Github:

```
git clone https://github.com/Roboy/ss18_xylophone_recording.git
```

2.1.5 Part 4: Setup ROS

This is just one way of installing ROS for Roboy. There are probably more suited approaches which are more elegant. But this one worked for us. ROS can be installed on another computer then were Unity is installed.

Install ROS like described in the Roboy repository.

Change to the Roboy directory and checkout our branch of the roboy_communication repository:

```
cd path-to-Roboy/src/roboy_communication
git checkout ss18_xylophone_recording
```

If you just want to run our program you can move all the other submodules/folders inside src (everything except CMakeLists.txt and roboy_communication) to another folder outside src so that they won't be built:

```
cd path-to-Roboy
mkdir donotbuild
mv src/common_utilities donotbuild
...
```

Build the project:

```
source devel/setup.bash
catkin_make
```

2.2 Starting the Program

2.2.1 Step 1: Launch ROSBridge Server

Before running the Unity Project launch the ROSBridge Server on a device which has ROS installed like this:

```
source path-to-Roboy/devel/setup.bash
roslaunch rosbridge_server rosbridge_websocket.launch
```

2.2.2 Step 2: Check for an available Midi Device

Check if you have any available Midi Devices. This can be done by using a Midi Management/Monitoring tool like Midi-OX (Windows). In Midi-OX use Options > Midi Devices to see the current Midi Devices available.

An Alternative for MacOSX would be MIDIMonitor.

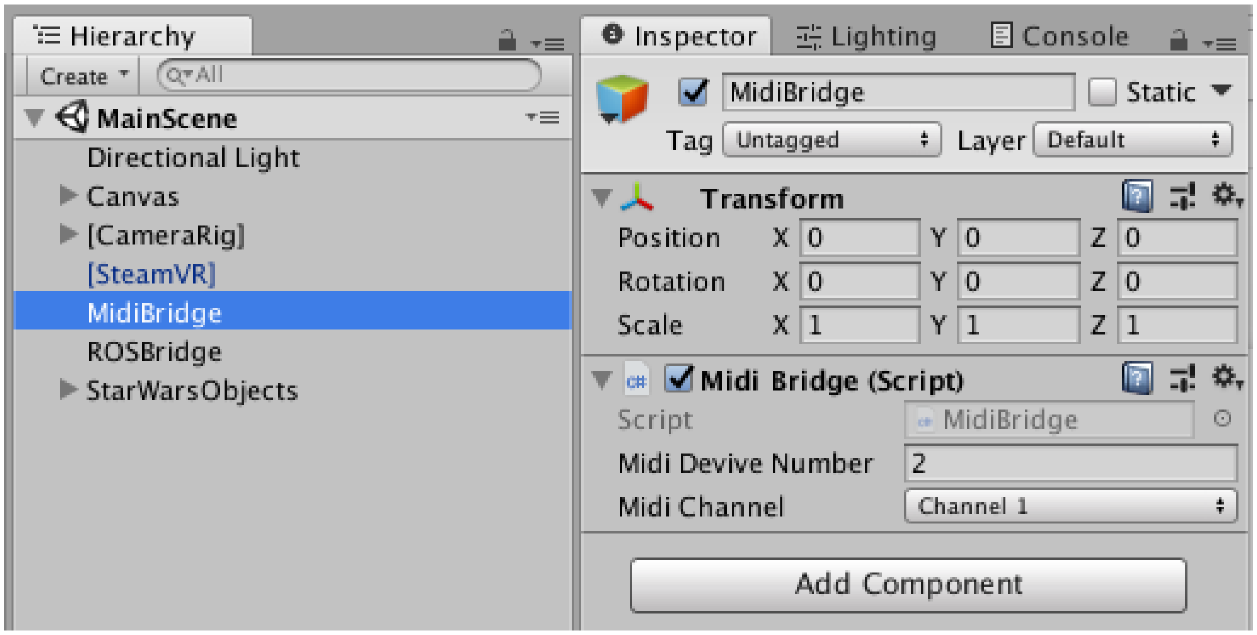
Note down the Midi Device Number of your prefered Midi Output as you need it later on.

If you don't have a local Midi Device you can use a loopback Driver like loopMIDI (Windows) for testing purposes. On MacOSX you can use the this guide to setup a loopback Driver.

If you want to send Midi over the network to another computer in an efficient manner check out the "Communication > Jack2 Network Setup" chapter as the setup for this is not trivial!

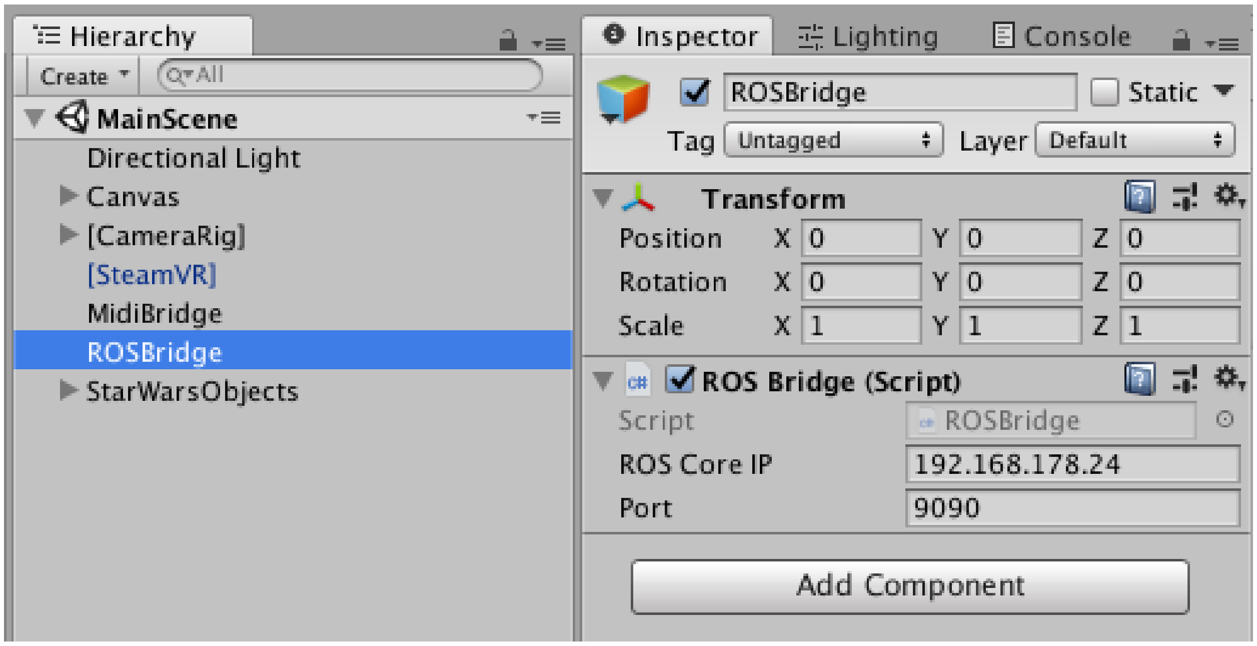
2.2.3 Step 3: Launch Unity

Launch Unity and check the GameObjects MidiBridge for the correct Midi Device Number which you wrote down earlier and your desired Midi Channel.



MidiBridge
Tag Untagged Layer Default
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Script MidiBridge
Midi Devive Number 2
Midi Channel Channel 1

Check the GameObject ROSBridge for the correct ROS Core IP (computer where you started the ROSBridge Server) and Port.



ROSBridge
Tag Untagged Layer Default
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Script ROSBridge
ROS Core IP 192.168.178.24
Port 9090

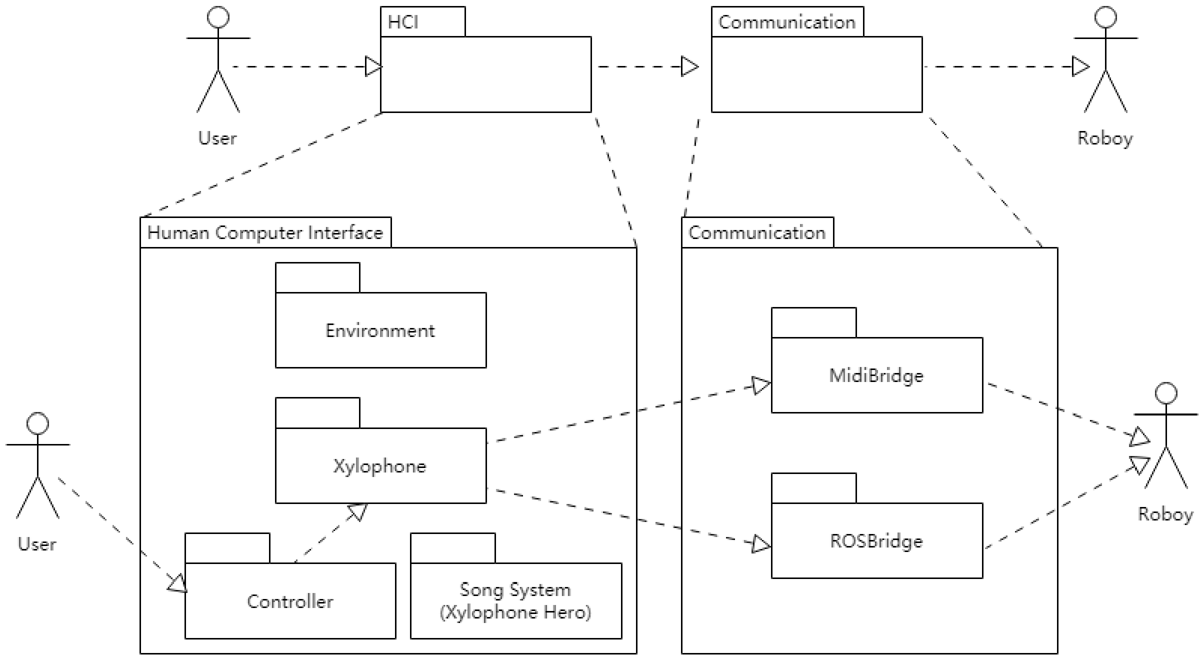
2.3 Extra Softwares for Development

2.3.1 Install Blender

We use Blender to build some of the 3D models in our project. If you want to modify the models or build some new ones, you can download Blender here, run the downloaded installation program to install blender and then you are ready to go. You can find very good tutorials online. If you are new to blender, This could be a good start.

CHAPTER 3

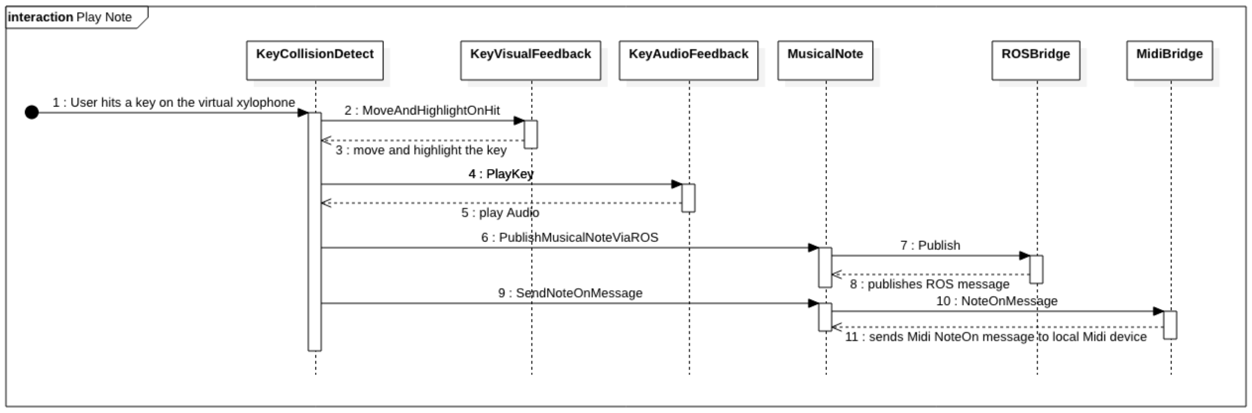
Project Structure



CHAPTER 4

Use Case

4.1 Play Note

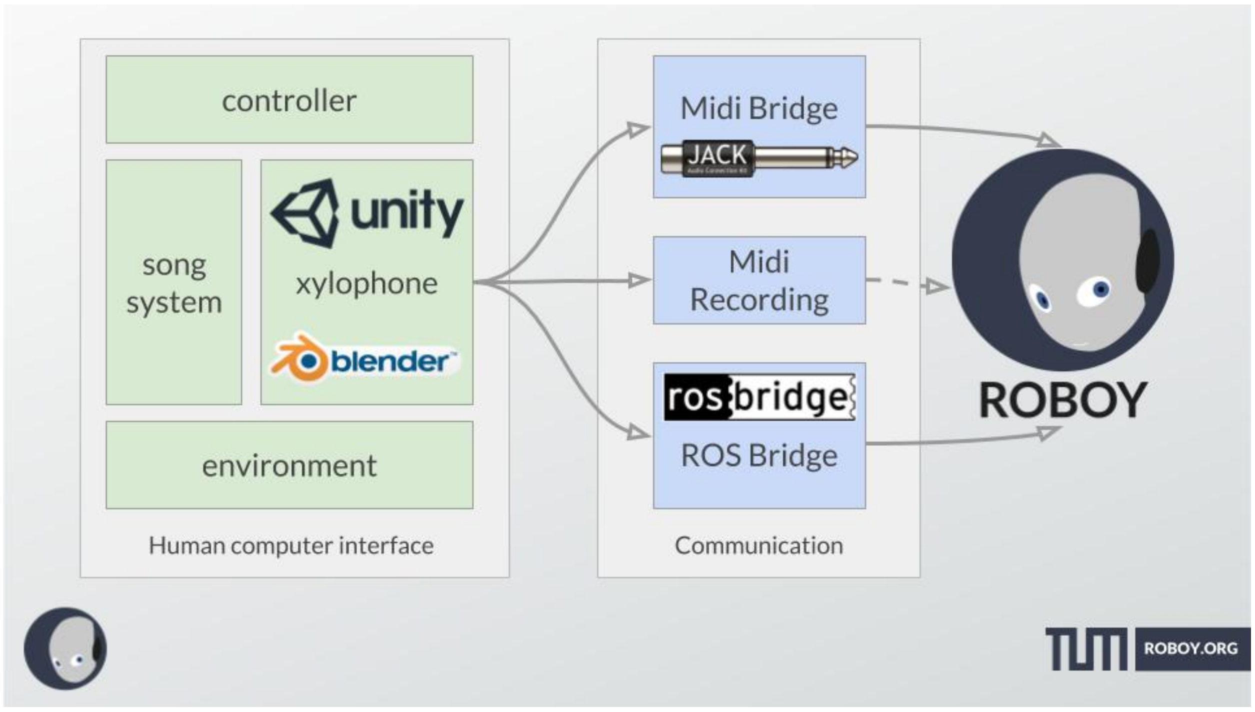

interaction Play Note
KeyCollisionDetect
KeyVisualFeedback
KeyAudioFeedback
MusicalNote
ROSBridge
MidiBridge
1 : User hits a key on the virtual xylophone
2 : MoveAndHighlightOnHit
3 : move and highlight the key
4 : PlayKey
5 : play Audio
6 : PublishMusicalNoteViaROS
7 : Publish
8 : publishes ROS message
9 : SendNoteOnMessage
10 : NoteOnMessage
11 : sends Midi NoteOn message to local Midi device

(right-click > show graphic for a bigger version of the sequence diagramm)

CHAPTER 5

Communication

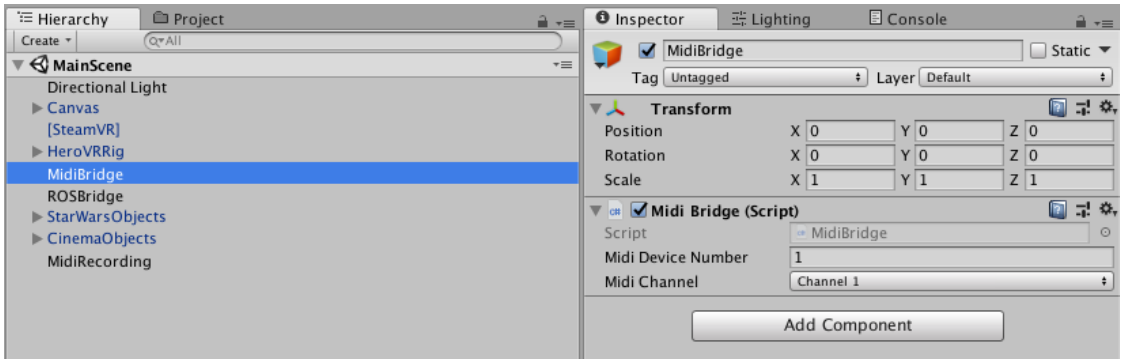
5.1 Structure



There are multiple communication/output channels in order to save or communicate the played xylophone notes:

5.1.1 MidiBridge

Live local Midi Output of the Xylophone notes. The Midi note Velocity is static. The **MidiBridge** GameObject is used to configure the **Midi Device Number** and the **Midi Channel**. Jack2 can be used to transport the Midi Data via Ethernet to another device (see Jack2 Network Setup).

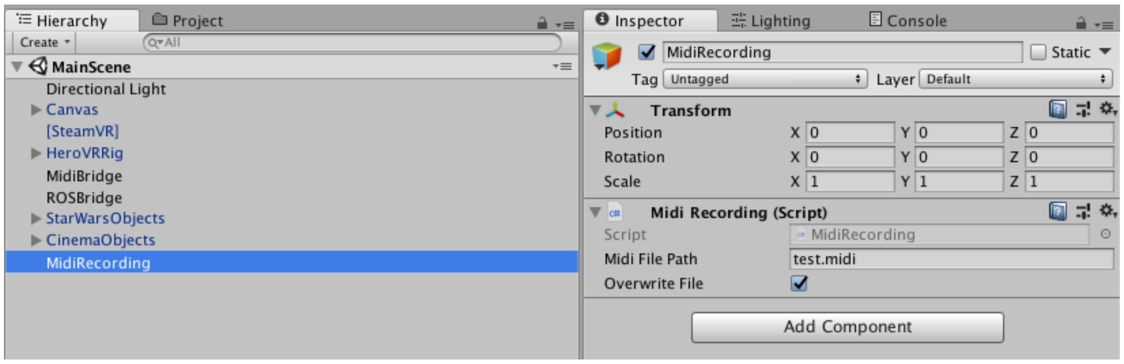


MainScene
Directional Light
Canvas
[SteamVR]
HeroVRRig
MidiBridge
ROSBridge
StarWarsObjects
CinemaObjects
MidiRecording

MidiBridge Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Midi Bridge (Script)
Script MidiBridge
Midi Device Number 1
Midi Channel Channel 1

5.1.2 MidiRecording

Writes the Midi NoteOn and NoteOff events of the xylophone to a Midi file. The recording can be started and stoped inside the VR environment. Via the **MidiRecording** GameObject the **Midi File Path** and the option to overwrite this file can be set.

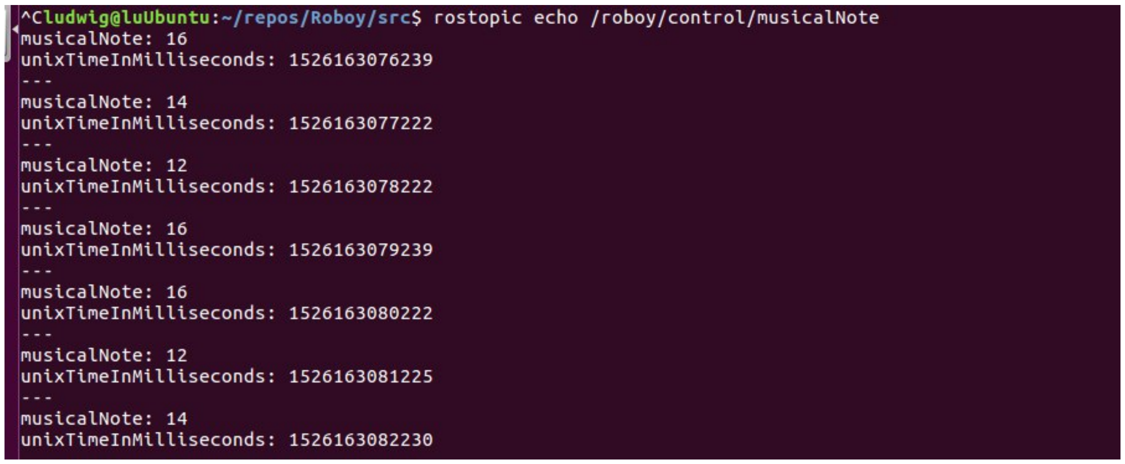


MainScene
Directional Light
Canvas
[SteamVR]
HeroVRRig
MidiBridge
ROSBridge
StarWarsObjects
CinemaObjects
MidiRecording

MidiRecording Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Midi Recording (Script)
Script MidiRecording
Midi File Path test.midi
Overwrite File ✓

5.1.3 ROSBridge

ROS Message (/roboy/control/musicalNote) sent in a Subscriper/Publisher fashion. The message is sent when the note is triggered and just contains the note as Midi Integer representation and the UNIX time in Milliseconds when it was played.



```
^Cludwig@luUbuntu:~/repos/Roboy/src$ rostopic echo /roboy/control/musicalNote
musicalNote: 16
unixTimeInMilliseconds: 1526163076239
---
musicalNote: 14
unixTimeInMilliseconds: 1526163077222
---
musicalNote: 12
unixTimeInMilliseconds: 1526163078222
---
musicalNote: 16
unixTimeInMilliseconds: 1526163079239
---
musicalNote: 16
unixTimeInMilliseconds: 1526163080222
---
musicalNote: 12
unixTimeInMilliseconds: 1526163081225
---
musicalNote: 14
unixTimeInMilliseconds: 1526163082230
```

5.2 ROS Debugging/Demo Cheatsheet

This was used for debugging/demo purposes to see the Midi messages:

```
cd path/to/Roboy
source devel/setup.bash
rostopic list
rostopic echo /roboy/control/musicalNote
```

5.3 Current State

So far, there are three ways of communication/output via ROSBridge, MidiRecording and via MidiBridge.

The ROS Messages are pretty basic and can be extended if needed. We didn't extend the ROS approach as the Jack2 approach seems to have a better performance as Jack2 is based on UDP packets and not on TCP packets like ROS with the Unity ROSBridge.

Jack2 could probably be integrated more tightly on a library level in Unity and not just on a programm level which uses the Midi Data coming from the MidiBridge as Input.

CHAPTER 6

Communication > Jack2 Network Setup

In the following subsections it's explained how you can use the local Midi Output of the MidiBridge and send it to other computers with Jack2. In this guide Jack is being used as a shortcut for Jack2. To be more specific Netjack2 which is a module of Jack2 is used for our Midi Network Setup.

6.1 Requirements

- LAN with Multicast Support (IGMP Snooping and IGMP Querier)
- two computers capable of running a Jack2 server

6.2 Setup Steps

In this example we're using a Linux machine as the Jack Server and a Windows machine with Unity and the HTC Vive as a Jack Client.

6.2.1 Linux Jack Server

Install Jack2 and the needed tools with the following commands:

```
sudo apt-get update
sudo apt-get install jackd2 qjackctl
```

For better performance add your user to the audio group which has elevated rights for realtime audio enhancing features. How to do this in Ubuntu is explained here.

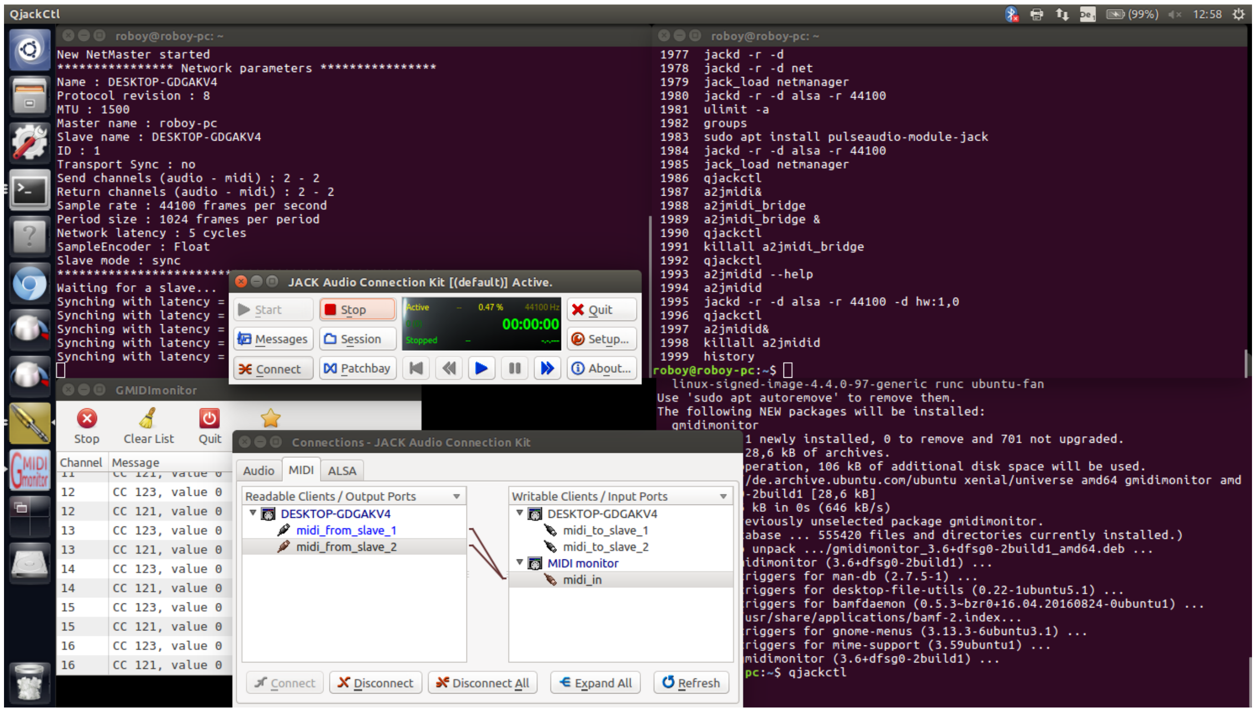
Use the following commands to start the Jack2 Server on the receiving side of your network setup with Linux:

```
jackd -r -d alsa -r 44100
#start the following programm in another terminal or in the programm launcher
qjackctl
#in another terminal start the following to load/unload the netjack2 module
jack_load netmanager
jack_unload netmanager
```

The following tools might also be of interest for different purposes:

- Qsynth can be used as a Synthesizer to play sounds of the received Midi notes
- Gmidimonitor can be used as a Midi Monitor to visualize the received Midi notes
- a2jmidid can be used as a alsa midi to jack midi bridge for software which doesn't natively support jack

The following picture shows all the launched programms:



```
New NetMaster started
*************** Network parameters ***************
Name : DESKTOP-GDGAKV4
Protocol revision : 8
MTU : 1500
Master name : roboy-pc
Slave name : DESKTOP-GDGAKV4
ID : 1
Transport Sync : no
Send channels (audio - midi) : 2 - 2
Return channels (audio - midi) : 2 - 2
Sample rate : 44100 frames per second
Period size : 1024 frames per period
Network latency : 5 cycles
SampleEncoder : Float
Slave mode : sync
*************************
Waiting for a slave...
Synching with latency =
Synching with latency =
Synching with latency =
Synching with latency =
Synching with latency =
```

Channel	Message
11	CC 121, value 0
12	CC 123, value 0
12	CC 121, value 0
13	CC 123, value 0
13	CC 121, value 0
14	CC 123, value 0
14	CC 121, value 0
15	CC 123, value 0
15	CC 121, value 0
16	CC 123, value 0
16	CC 121, value 0

```
1977  jackd -r -d
1978  jackd -r -d net
1979  jack_load netmanager
1980  jackd -r -d alsa -r 44100
1981  ulimit -a
1982  groups
1983  sudo apt install pulseaudio-module-jack
1984  jackd -r -d alsa -r 44100
1985  jack_load netmanager
1986  qjackctl
1987  a2jmidi&
1988  a2jmidi_bridge
1989  a2jmidi_bridge &
1990  qjackctl
1991  killall a2jmidi_bridge
1992  qjackctl
1993  a2jmidid --help
1994  a2jmidid
1995  jackd -r -d alsa -r 44100 -d hw:1,0
1996  qjackctl
1997  a2jmidid&
1998  killall a2jmidid
1999  history
roboy@roboy-pc:~$
```

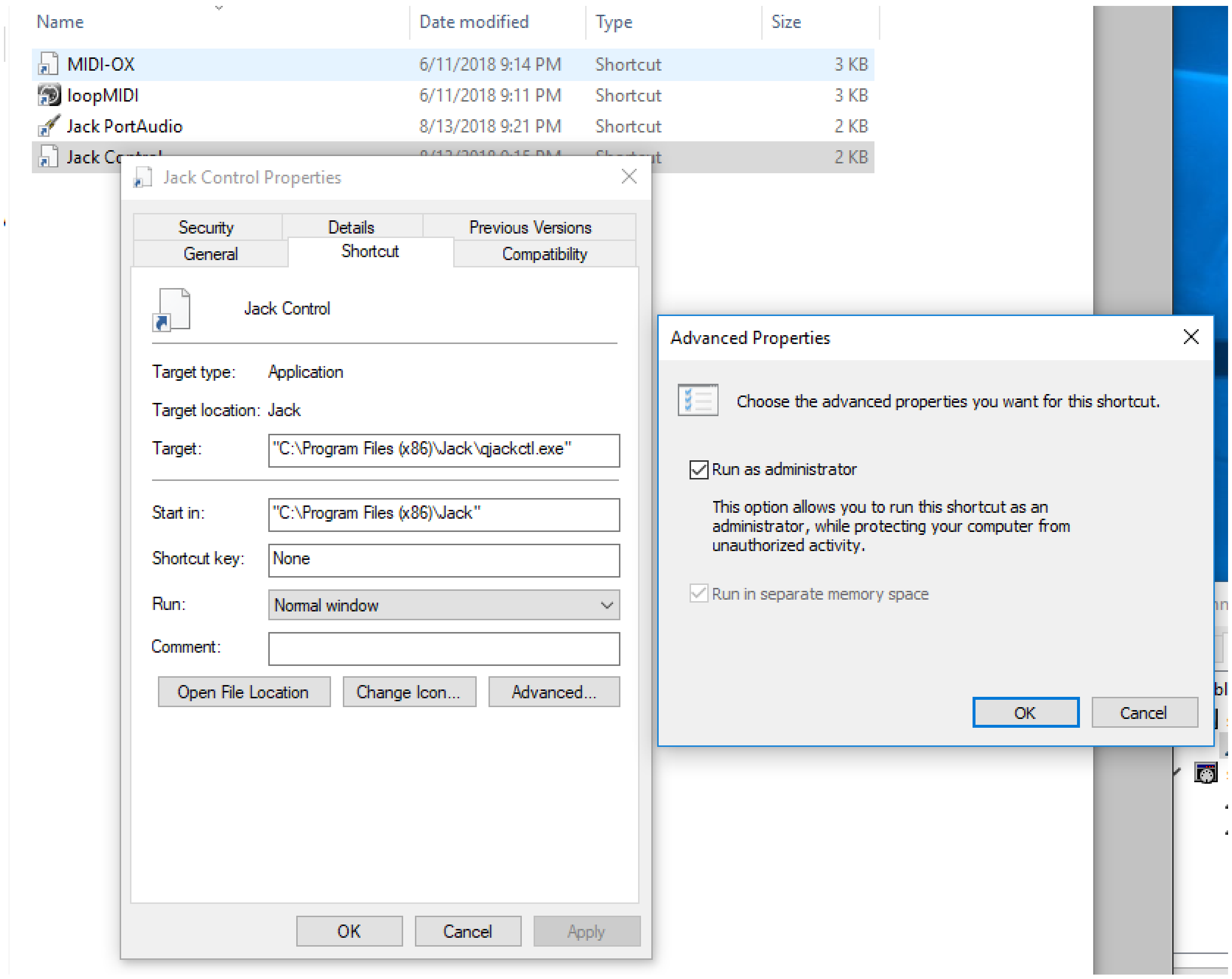
```
  linux-signed-image-4.4.0-97-generic runc ubuntu-fan
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  gmidimonitor
1 newly installed, 0 to remove and 701 not upgraded.
28,6 kB of archives.
peration, 106 kB of additional disk space will be used.
/de.archive.ubuntu.com/ubuntu xenial/universe amd64 gmidimonitor amd
-2build1 [28,6 kB]
 kB in 0s (646 kB/s)
eviously unselected package gmidimonitor.
abase ... 555420 files and directories currently installed.)
 unpack .../gmidimonitor_3.6+dfsg0-2build1_amd64.deb ...
idimonitor (3.6+dfsg0-2build1) ...
riggers for man-db (2.7.5-1) ...
riggers for desktop-file-utils (0.22-1ubuntu5.1) ...
riggers for bamfdaemon (0.5.3~bzr0+16.04.20160824-0ubuntu1) ...
usr/share/applications/bamf-2.index...
riggers for gnome-menus (3.13.3-6ubuntu3.1) ...
riggers for mime-support (3.59ubuntu1) ...
midimonitor (3.6+dfsg0-2build1) ...
pc:~$ qjackctl
```

In the qjackctl window (JACK Audio Connection Kit) press the Connect button when the Client is connected and connect the Clients Midi Output (DESKTOP-GDGAKV4 in the Screenshot) with your desired Midi Application like Qsynth or Gmidimonitor.

6.2.2 Windows Jack Client

To install the client side of Jack and its tools on Windows use this guide.

To be a 100 percent sure we launched qjackctl.exe as Administrator, but maybe this step is not even needed.



Name	Date modified	Type	Size
MIDI-OX	6/11/2018 9:14 PM	Shortcut	3 KB
loopMIDI	6/11/2018 9:11 PM	Shortcut	3 KB
Jack PortAudio	8/13/2018 9:21 PM	Shortcut	2 KB
Jack Control			2 KB

Jack Control Properties

Jack Control

Target type: Application

Target location: Jack

Target: "C:\Program Files (x86)\Jack\qjackctl.exe"

Start in: "C:\Program Files (x86)\Jack"

Shortcut key: None

Run: Normal window

Comment:

Advanced Properties

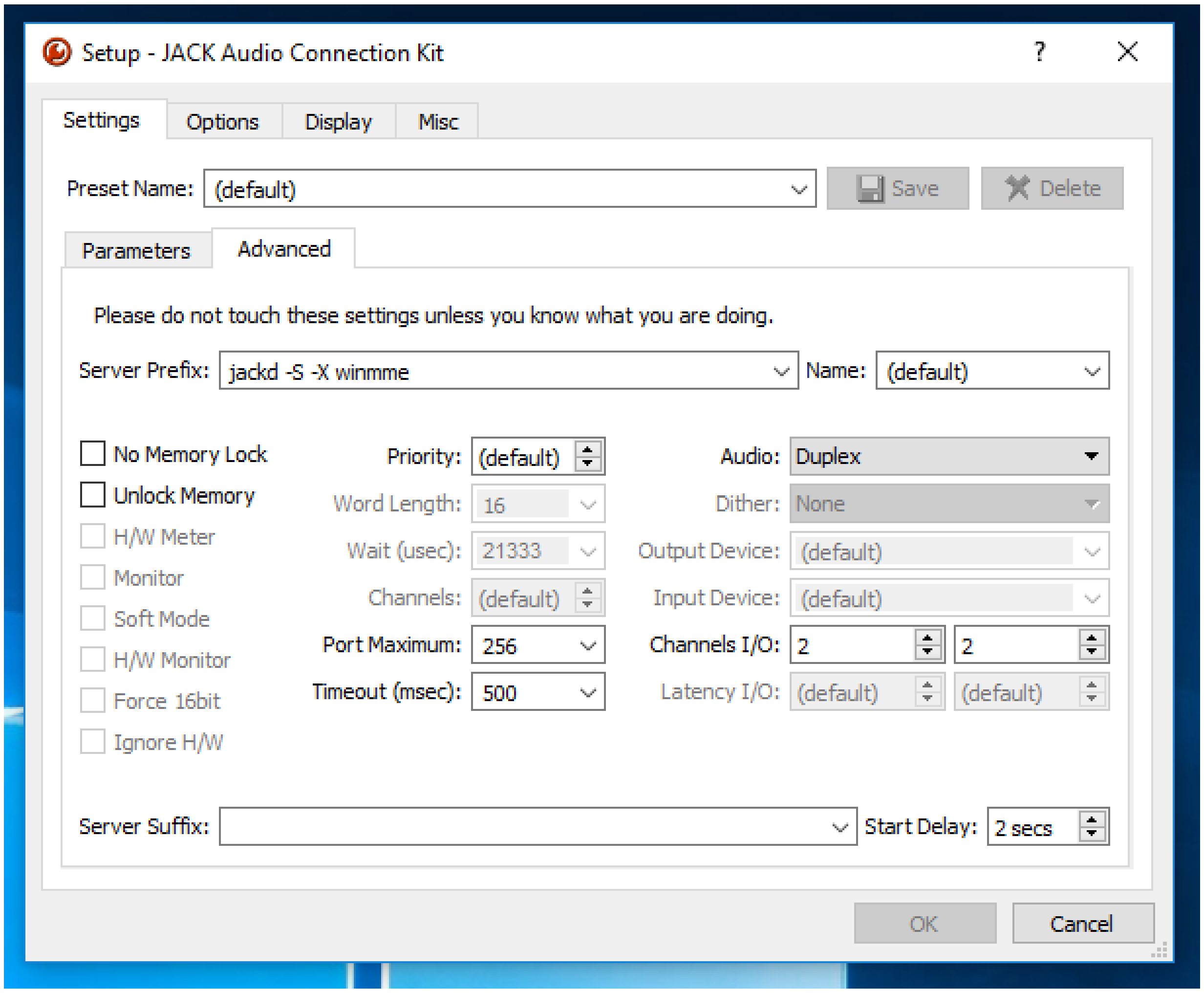
Choose the advanced properties you want for this shortcut.

[✓] Run as administrator

This option allows you to run this shortcut as an administrator, while protecting your computer from unauthorized activity.

[✓] Run in separate memory space

After that the Jack Server was configured with the following parameters:



Setup - JACK Audio Connection Kit

Settings | Options | Display | Misc

Preset Name: (default)

Save | Delete

Parameters | Advanced

Please do not touch these settings unless you know what you are doing.

Server Prefix: jackd -S -X winmme

Name: (default)

- [ ] No Memory Lock
- [ ] Unlock Memory
- [ ] H/W Meter
- [ ] Monitor
- [ ] Soft Mode
- [ ] H/W Monitor
- [ ] Force 16bit
- [ ] Ignore H/W

Priority: (default)

Word Length: 16

Wait (usec): 21333

Channels: (default)

Port Maximum: 256

Timeout (msec): 500

Audio: Duplex

Dither: None

Output Device: (default)

Input Device: (default)

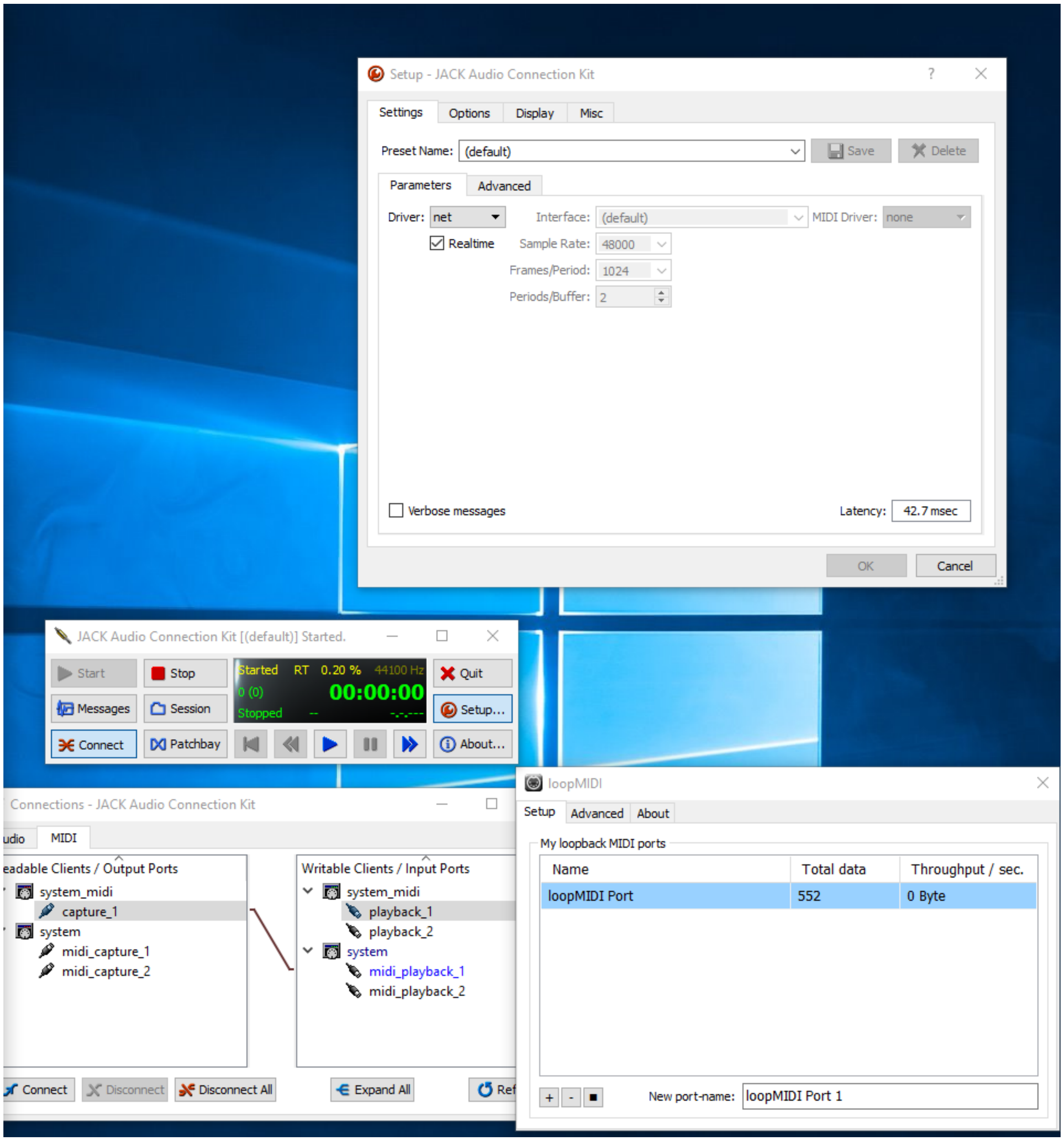
Channels I/O: 2 2

Latency I/O: (default) (default)

Server Suffix:

Start Delay: 2 secs

OK | Cancel



Preset Name: (default)

Driver: net
Interface: (default)
MIDI Driver: none
[✓] Realtime
Sample Rate: 48000
Frames/Period: 1024
Periods/Buffer: 2
[ ] Verbose messages
Latency: 42.7 msec

Started RT 0.20 % 44100 Hz
0 (0) 00:00:00
Stopped -- --:--.---

eadable Clients / Output Ports
- system_midi
  - capture_1
- system
  - midi_capture_1
  - midi_capture_2

Writable Clients / Input Ports
- system_midi
  - playback_1
  - playback_2
- system
  - midi_playback_1
  - midi_playback_2

Name	Total data	Throughput / sec.
loopMIDI Port	552	0 Byte

New port-name: loopMIDI Port 1

In the previous screenshot you could also see how we used the loopMIDI midi loopback driver as an readable client (capture_1) and send it to a writable client (system > midi_playback_1 - the Linux Jack Server).

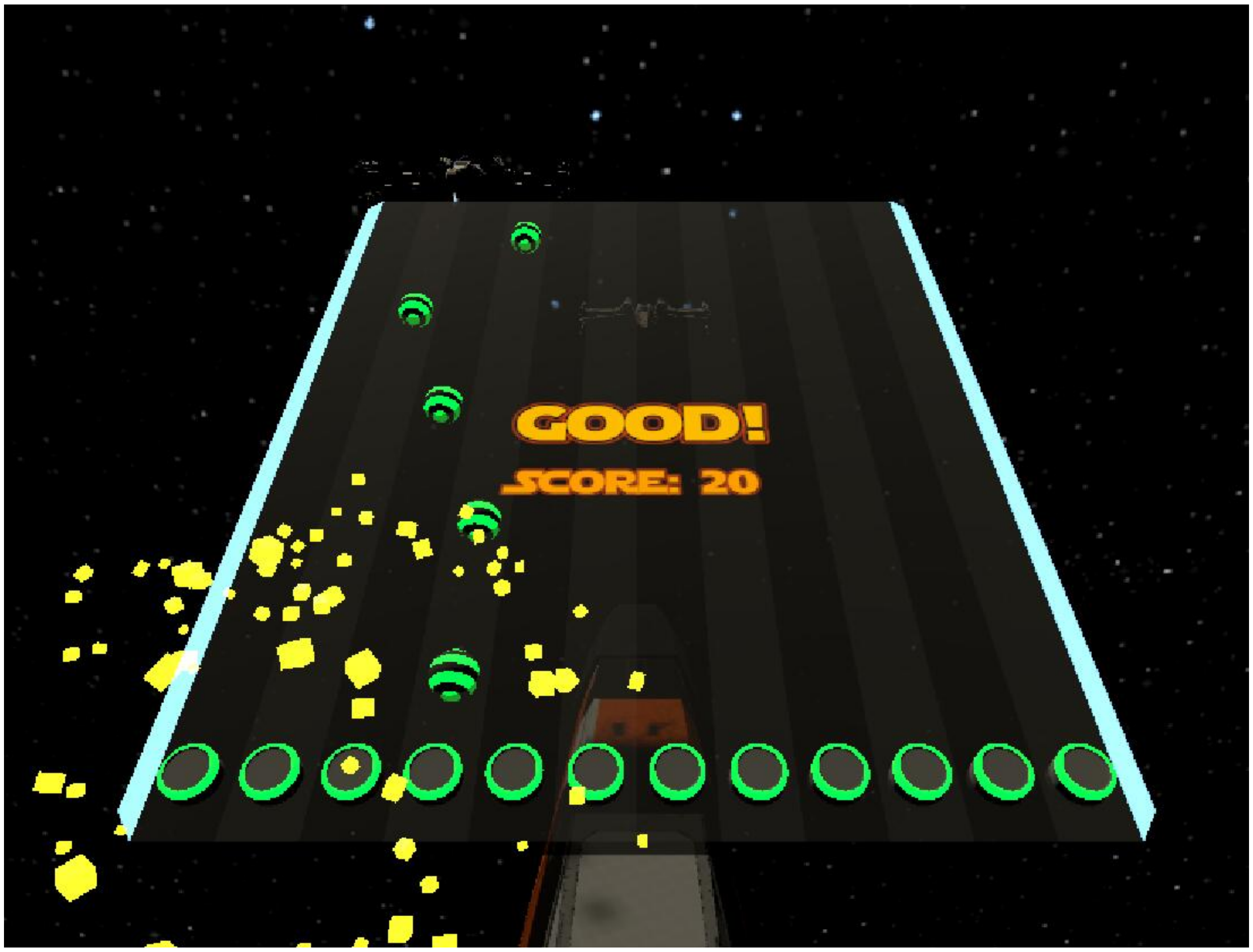
CHAPTER 7

Song System (Xylophone Hero)

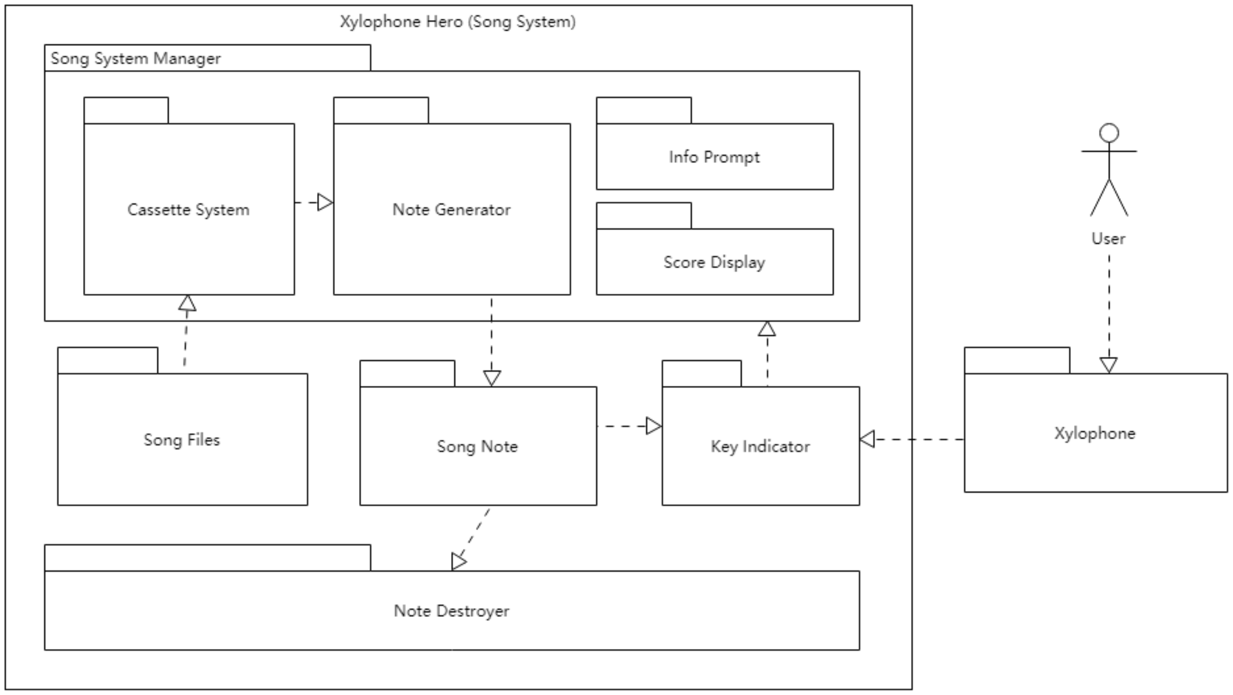
7.1 Motivation and Introduction

When the user is playing the xylophone in virtual reality, he may sometimes feel lost and don't know what to play. Therefore, we decided to add a song system, which can tell the user which note should he/she play and when to play. Furthermore, we considered that some gaming features can be added to the system, such as a score system and more fancy visual effects.

Inspired by the famous video game serious Guitar Hero, we name our song system Xylophone Hero.


GOOD!
SCORE: 20

7.2 Structure



The Song System, aka. Xylophone Hero, consists of 3 Major Parts: **Song System Manager**, **Key Indicator** and **Note Destroyer**. It reads song files from the disk and generate in game playable songs. Users interact with the song system through the xylophone. Details are described below.

7.2.1 Song File

The song files are not actual audio recordings of the song (`.mp3`, `.wav`, etc.). They contain only the sequences of the notes which will be used to generate the notes in the song system. The format is inspired by the `.sm` file of StepMania which looks as follows:

```
100000000000
000000000000
000000000000
010000000000
001000000000
000000000000
000000000000
100000000000
```

Each line represents one beat. There are 12 keys in the xylophone now so each row has 12 digits, each one corresponds to one key. 1 means this key should be played on this beat, 0 means it should not. So the note generator can generate the notes of the song according to this song file.

New songs can be easily added to the system by simply adding new song files in `txt` format to `Assets/Resources/Songs`.

7.2.2 Song System Manager

Song System Manager is the most important and core part of the song system. It consists of 3 parts: **Note Generator**, **Cassette System** and **Score and Feedback System**. And these parts are all organized and managed by the **Song Manager**. The Song System Manager manages the state of the song system, displays useful information, and generates the notes of songs.

Song Manager

The SongManager mainly manages the state of song playing. When the song system starts, it loads the song files from the designated path, prepares songs for the cassette system. It has functions to switch between state of playing and stop as well as switch songs. These functions also control the Note Generator to generate proper notes. In addition, the Song Manager monitors how good the player plays the xylophone then calculate scores and give proper feedback.

Attention: The `SongManager` class is a Singleton. Only one instance of this class is allowed. Therefore don't put two song boards in one scene! In the case of multiple environments (like the cinema and STARWARS), either only one environment having the song board or all environments sharing one song board is the correct solution.

Note Generator

Just as its name indicates, the note generator generates the notes according to the song file. It starts a coroutine, which respawns the notes at a certain rate according to the set BPM (Beats Per Minute). The notes then fall down. When they reach the corresponding key indicator, the User should play the key.

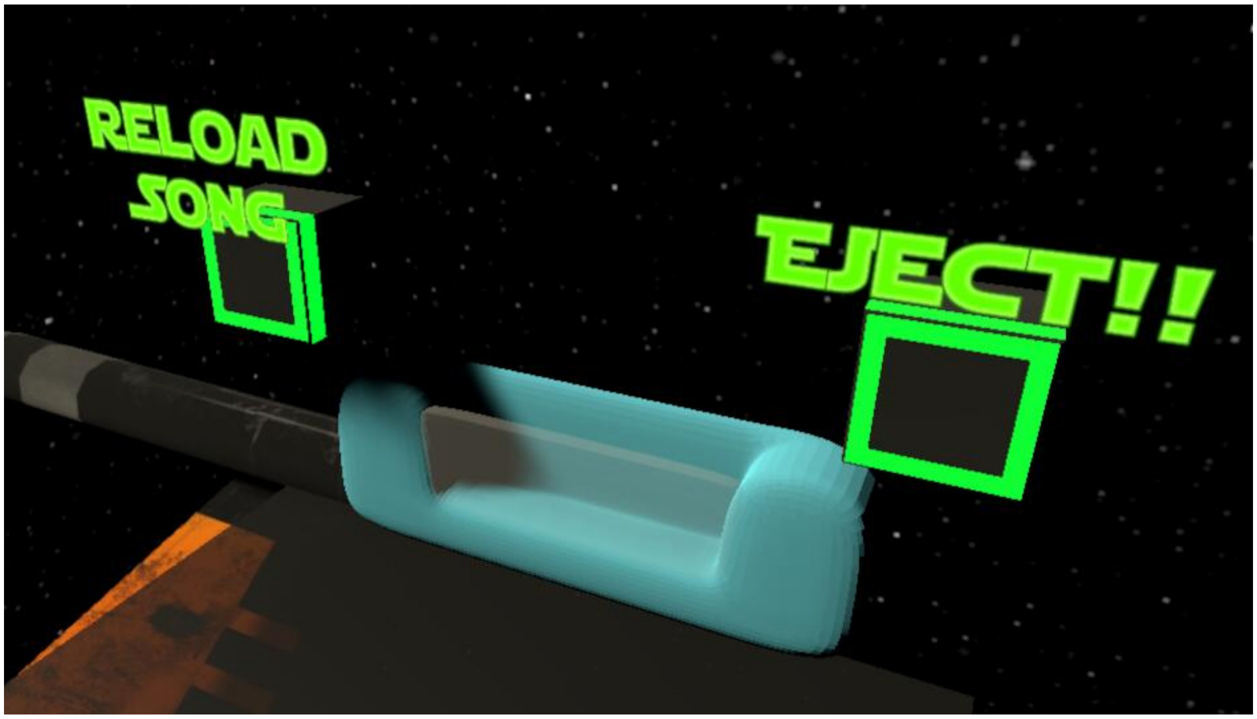
Cassette System

We seek to make the song system more interactive, more intuitive and more interesting to play. Therefore, instead of having a song list displaying all songs and player selecting songs and controlling the system by pressing buttons, we decided to use cassettes to represent songs and a cassette player to control the song playing.

After the Song Manager loads the songs, it will generate several cassette, each one carries the information of one song, on the desk. The player can use the stick with Roboy's head to grab a cassette and then put it in the cassette player, which will make the Song Manager start the song.

DOREMI
LITTLESTAR

There are two buttons above the cassette player. The left one is *Reload Song*. By pressing this button the Song Manager clears the current loaded songs, destroys all existing cassettes, and then loads the songs and generates the cassettes again. This is useful when the player drops some cassettes to the space or can not reach the cassettes for some reasons. The right button is *Eject* button. This makes the cassette player eject the current cassette and stop playing the song.



Score and Feedback System

Scores and Feedbacks help players become xylophone masters. When the player keep playing the right keys at the right time, he/she gets *combos*, otherwise he/she gets *misses*. When combos or misses reaches certain level, the player will hear cheers or boos and see texts of praise or disappointment. In addition, the player gets scores with every good hit, and combos make more score per good hit.

7.2.3 Key Indicator

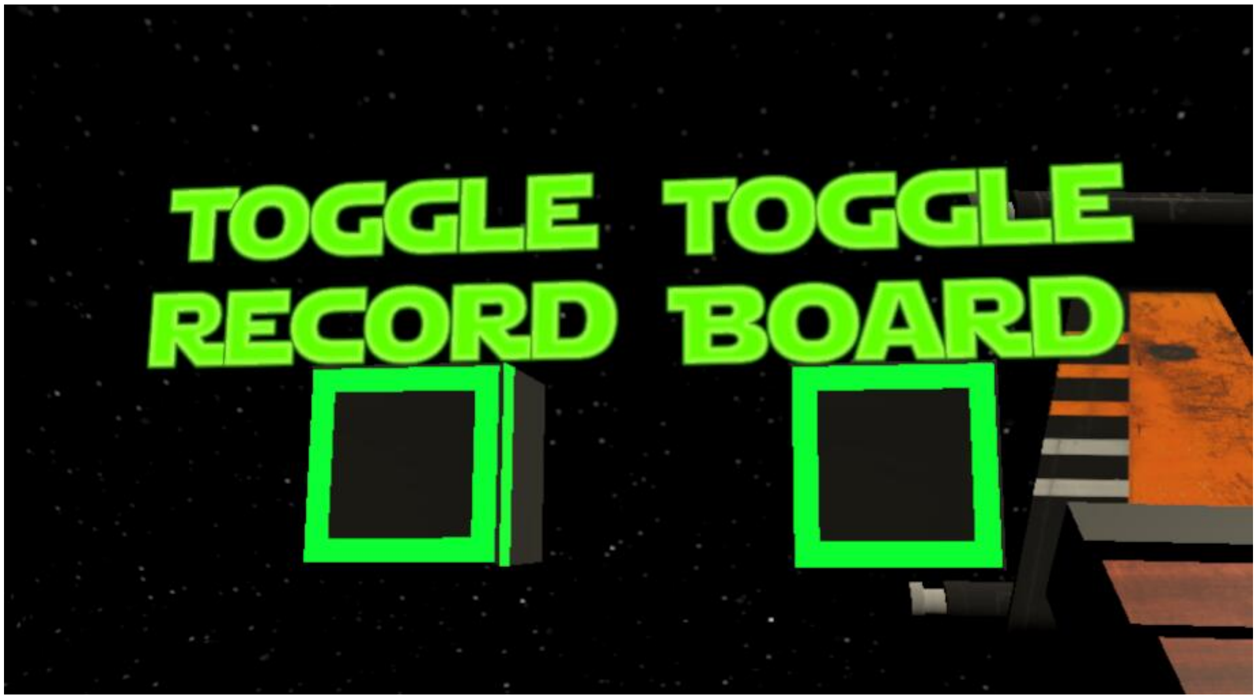
The key indicator indicates which key should be play now. When the song note falls and hits the key indicator, the corresponding key should be played. if the key is played, the song note would be destroyed and the key indicator would tell song system manager that the user has scored. If not, the song note would fall through the indicator and reach the note destroyer.

7.2.4 Note Destroyer

The Note Destroyer destroys all game objects with the tag `SongNote` which enter its trigger zone. This basically "cleans up" the notes that are not played by the user.

7.2.5 Some Other Things

There are two buttons on the left side of the player (when facing the xylophone). One is *Toggle Recording*. This one toggles converting the played notes to midi. The other one is *Toggle Board*. This one shows or hides the song board.



Under the `SongSystemManager` Game Object there is a set of **Control Buttons** which are disabled. They are not used in the release version of Xylophone Hero, but they are good helpers for debugging. When they are enabled, the player or developer can use the keyboard to start, stop, play the previous or play the next song. Of course the player can also interact with these buttons using the sticks.

7.3 Current State

The song system is playable and fun to play. It can load multiple songs from the designated path and generate cassettes for the songs. the cassette system is ready to use. Score and feedback system works fine, but still has room for improvements (better rules, ranking system, etc.). In addition, the models of the song system can be further polished or beautified.

CHAPTER 8

Environments

8.1 Scenes

This section would describe the xylophone playing environment in VR and how to interact with that both on users and developer view. You can play xylophone in two different environments the first one called Star Wars and the second one is Cinema. In the first scene the xylophone player is in the space and there are moving object around him/her. The player can activate the songboard and play with that in this environment. In the second scene, a concert environment simulated for the xylophone player. Behind the xylophone player there are simulated faces sitting on chairs and waiting for you to play the xylophone. In front of the xylophone player there is a big scene which shows the output camera from VR. So when you are playing xylophone you can also see what is actually happening in front of you.

Notes:

- When You start the program, it will automatically transform the center of all objects coordinate to the HTC Vive position.
- Both Controller should be activated.

8.1.1 Developer view

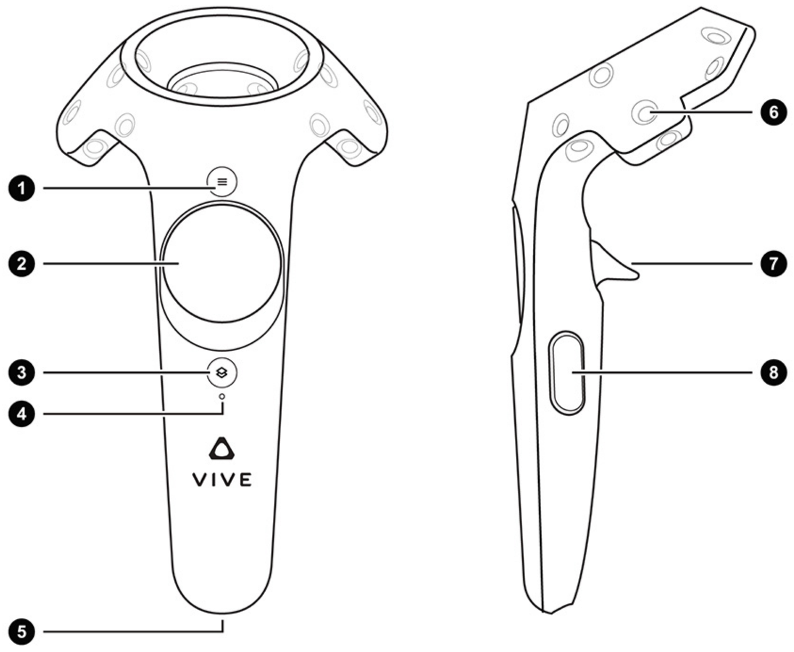
Every Object inside the Star Wars scene are under StarWarsObject unity object and the cinema objects are under CinemaObject. Objects movement inside Star Wars environment are defined in scripts/StarWars/MovingObjectsInStarWarsSkyBox.cs.

And if you want to add an object which you want to move like tieFighters or xwing just put it inside StarWarsObject or if you want to add more laser-bolt create your objects inside StarWarsObject and assign laserBolt tag for them. In the scripts/StarWars folder there is another script named FixStartPosition.cs, which would translate every object in the scene based on the location of VR. In the cinema scene when you move the sticks some chips would be throwed out of the box. The script related to this action is at script/Cinema/ThrowedChipsHandler.cs. Another script under script/Cinema is WebCam.cs, in this script you can choose other source of camera output which connected to the PC, would be shown on the front scene. Just locate the following code and change the value of “HTC Vive” to the name of your camera source. “if (cameraDevices[viveCameraIndex].name.Equals(“HTC Vive”))”

8.2 Controllers

In the VR you can play xylophone with different sticks which each one of them produce different sound, for example the Lightsaber stick would create more electronic xylophone sound and the chips stick would create classical xylophone sound.

For changing sticks model in VR, you must press Grip button of Vive controller (button number 8 in the picture) then a menu which contains 3 controller model, would appear. You can choose each model with controller trackpad (button number 2 in picture). The controller would disappear as soon as you do not touch the trackpad.



When the xylophone player changes the sticks model the environments also change for him/her to base on the sticks. The lightsaber (the one that is like a sword like Star Wars movie) would change the environment to the Star Wars scene. The sticks with chips would change the environment to the cinema scene. The sticks with Roboy head on it would not change any scene, but you can grab cassette with this stick.

8.2.1 Developer view

If you want to add new controller model you should add the model at position (0,0,0) in both controller (Controller (left) and Controller (right)) object inside cameraRig in unity. Also, you have to add the model picture in controller menu, so users could be able to change the sticker to that model. For modifying the controller menu functionality, you have to change controller.cs.

If you want to change the song for each note when changing the stick, you have to modify StarWarsObjects/Self/Xylophone/KeyManager/XylophoneKeyWrapper##/XylophoneKeyCollider. First Audio source is for the normal stick, second one is for the stick which had Roboy head on that, and the third one is for Lightsaber.

CHAPTER 9

Current State

9.1 Xylophone

The Xylophone is modeled like in the real world. Only thing still missing due to no testing is the integration of a variable velocity detection which forwarded to the communication modules.

9.2 Environment and Controllers

We build a quite good-looking environment for the user and the xylophone. The environment is a STARWARS theme. The xylophone sits on the back of a X-Wing Fighter and surrounded by thousands of stars. Several TIE Fighters is flying by, and the death star is also in sight. It is a very fancy environment for the user to play xylophone. Also, there is another scene which simulate a concert environment for the player.

There are currently three controllers, or, sticks for the user to use, which are a chips stick, a stick with a Roboy head, and a light saber. The user can switch between them and use each one of them to play the xylophone. When playing xylophone, different sticks play different sounds in different scene.

9.3 Communication

For current state of communication part, please refer to Communication.

9.4 Song system (Xylophone Hero)

For current state of song system part, please refer to Song System.

CHAPTER 10

Troubleshooting

10.1 Unity Errors concerning MidiBridge or ROSBridge

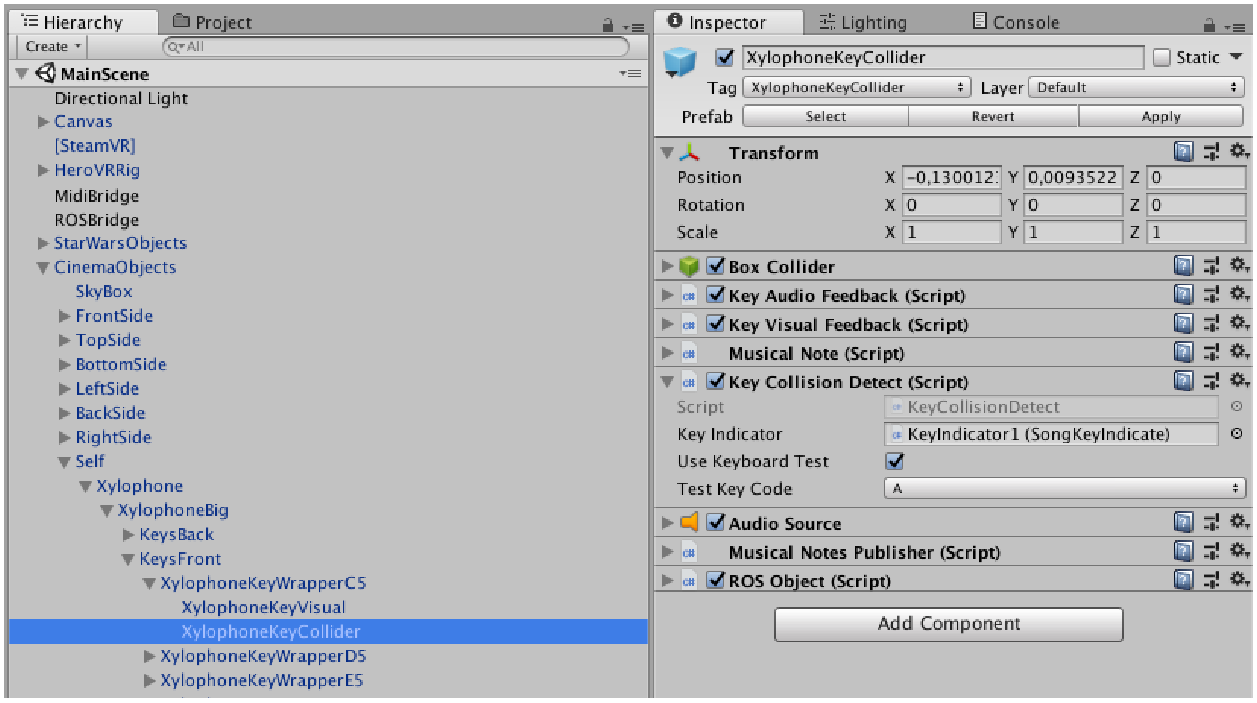
Make sure you have a Midi Device and selected an existing one. Also make sure the ROSBridge is reachable and running. If you disconnect the Midi Device or RosBridge Connection while running the program it will probably crash.

10.2 Problems with the Jack2 Installation and Config

If our given Setup Guide doesn't work for you and you're running Linux the Arch Wiki good a lot of hints on how to get it to work. Our Setup Guide was specified for Ubuntu so a few steps aren't the same but maybe if you're using another distro this will help you.

10.3 Debugging without a HTC Vive

In the Unity Xylophone model we added debugging support for playing the xylophone without having a HTC Vive at hand. You can set a computer keyboard key to a xylophone key in the **XylophoneKeyCollider** GameObject.



```
MainScene
  Directional Light
  ▶ Canvas
  [SteamVR]
  ▶ HeroVRRig
  MidiBridge
  ROSBridge
  ▶ StarWarsObjects
  ▼ CinemaObjects
      SkyBox
    ▶ FrontSide
    ▶ TopSide
    ▶ BottomSide
    ▶ LeftSide
    ▶ BackSide
    ▶ RightSide
    ▼ Self
      ▼ Xylophone
        ▼ XylophoneBig
          ▶ KeysBack
          ▼ KeysFront
            ▼ XylophoneKeyWrapperC5
                XylophoneKeyVisual
                XylophoneKeyCollider
            ▶ XylophoneKeyWrapperD5
            ▶ XylophoneKeyWrapperE5
```

```
✓ XylophoneKeyCollider                      Static
Tag  XylophoneKeyCollider   Layer  Default
Prefab  Select  Revert  Apply

▼ Transform
Position   X -0,130012   Y 0,0093522   Z 0
Rotation   X 0           Y 0           Z 0
Scale      X 1           Y 1           Z 1

▶ ✓ Box Collider
▶ ✓ Key Audio Feedback (Script)
▶ ✓ Key Visual Feedback (Script)
▶   Musical Note (Script)
▼ ✓ Key Collision Detect (Script)
Script              KeyCollisionDetect
Key Indicator       KeyIndicator1 (SongKeyIndicate)
Use Keyboard Test   ✓
Test Key Code       A
▶ ✓ Audio Source
▶   Musical Notes Publisher (Script)
▶ ✓ ROS Object (Script)
```

Some other GameObjects also have this keyboard debugging feature, including

- **ToggleBoardButton**
- **ToggleRecordingButton**
- **KeyIndicator**

Attention: Please avoid using keyboard dubugging for KeyIndicators and XylophoneKeyColliders at the same time because XylophoneKeyColliders can trigger KeyIndicators. If you have to apply keyboard debugging for them simultaneously, assign different keys for them.

CHAPTER 11

Libraries and External Software

11.1 Libraries

11.1.1 HCI

- TextMesh Pro (Substitute of the original unity text mesh)
- Post Processing Stack 1.0.4

11.1.2 Communciation

MidiBridge

- RtMidi.Core (Version: 1.0.38 netstandard2.0)
- Serilog (Dependecy for RtMidi.Core, Version: 2.6.0 netstandard1.3)

MidiRecording

- Melanchall.DryWetMidi (Version: 3.1.0 net45)

ROSBridge

- ROSBridge

11.2 External Software

These external programs have been used for developing and testing. Some of them could probably be substituted with different programs.

- Unity (VR Development)
- SteamVR (VR Development - HTC Vive)
- ROS (ROS Communication)
- MIDI-OX (Midi Testing - Midi Monitor Windows)
- MIDIMonitor (Midi Testing - Midi Monitor MacOSX)
- loopMIDI (Midi Testing - Midi Loopback Driver)
- Jack2 (Midi via Ethernet)
- Blender (3D modeling)

All libraries and models which were used from external sources are inside the ThirdParty folder:

- the lightsaber or laserSword model were downloaded from https://github.com/jjxtra/UnityLightsaber.git
- darth_vaders fighter, tieFighters, xwing were downloaded from clara.io and free3d.com

All textures we used for our own models are inside the Materials folder:

- the xylophone texture is from textures.com