052483 - COMPUTER MUSIC: LANGUAGES AND SYSTEMS

POLITECNICO DI MILANO

MUSIC AND ACOUSTIC ENGINEERING

Homework 3: Interaction Design

Group:
Radical Geeks (ID: 7)
Authors:
Gerardo Cicalese (ID: 10776504)
Alberto Bollino (ID: 10865248)
Umberto Derme (ID: 10662564)
Giorgio Granello (ID: 10869436)

Date: May 30, 2022

1 Purposes

The aim of this project is to implement a complete computer music system with interaction design principles. Learning objectives:

- Design the whole CMS: from the purpose of your final system (i.e., music performance system, human-interactive effects, augmented reality synthesis, and different cool things!);
- Decide and state the tools used and the communication protocols for sharing information;
- Implement your computer music system.

The project needs to comply with the interaction scheme presented in Fig. 1.

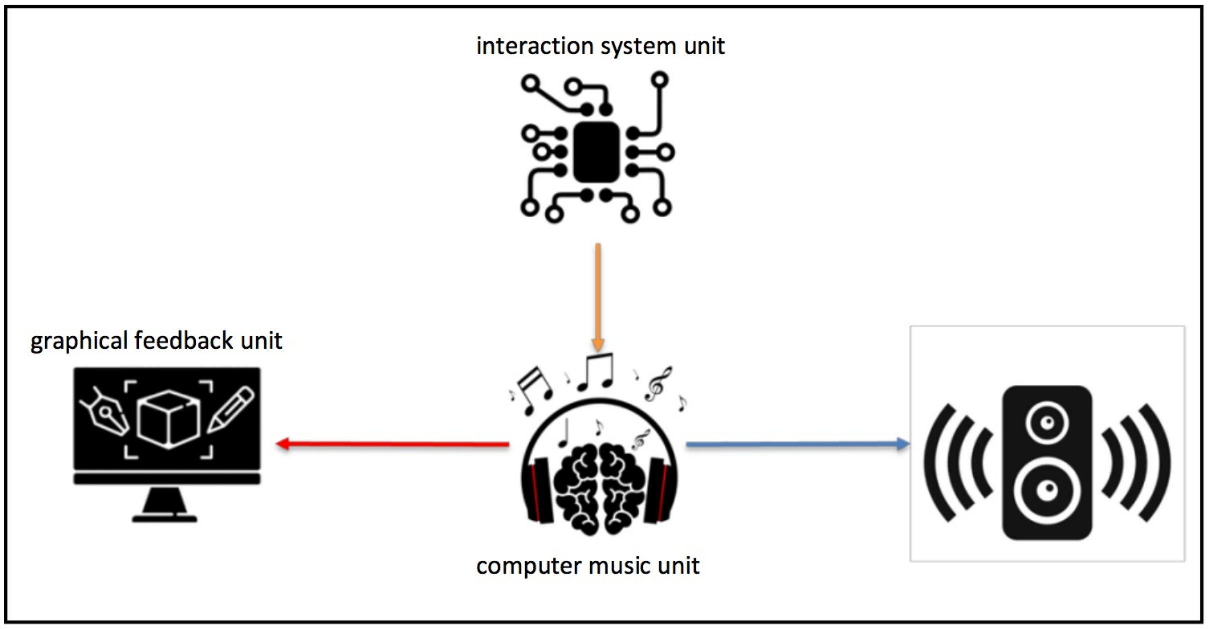


Figure 1: Interaction scheme.

2 Introduction to Chordophone Champion

Chordophone Champion (codenamed Ch-2) takes inspiration on the famous game **Guitar Hero**. The user can play Ch-2 using his mobile phone as input method.

3 Block diagram

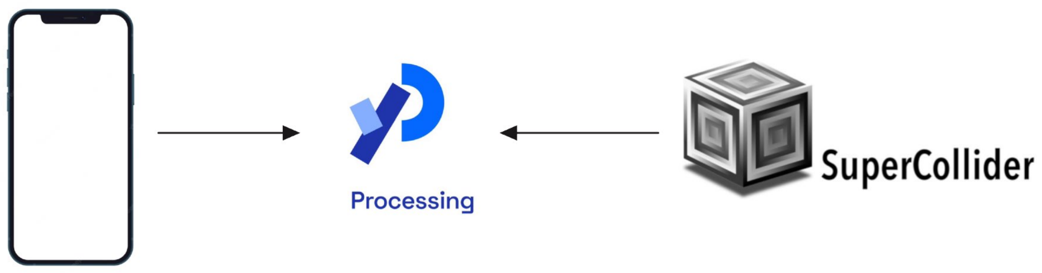


Figure 2: Blocks diagram of our Interaction Design project.

4 GUI

In the title screen, the user can choose which song to play, the instrument and the operating system of his mobile phone (by the means of three drop-down menus). The operating system choice is needed because, depending if you're using Android or iOS, there is a different application that allows you to control the game: multisense OSC for Android and gyrOSC for iOS. Finally, after all these setups, you can click the ”Start” button and start playing the game! The graphical user interface of the game is very simple: the total space of the interface is divided in twelve equal parts, each one representing one of the twelve existing notes, written in the English notation (so C,C,D,D,...). On the right of the screen we can notice that there is a vertical column in which there is a plectrum: in this column the plectrum can move up and down vertically. The last part of the interface is composed by some colored dots that move horizontally following one of the twelve tracks that we have described before and of course they represent a note of the song that the user is playing in real time. The usage of the game is also pretty simple: the only thing that the user is supposed to do is to move up and down the smartphone in order to control the plectrum that is present on the right in the interface. The main goal of the player is to center as many times as possible the colored dots that run from left to right, and do it on time. We can also notice that every line in the GUI represents a specific note of the song that is played in real time by the application, so when you move up and down with your phone you are simply playing a different note of the song that you have chosen. On the top right of the interface you can also find your current score and your combo, that will update both in real time. When the user has finished playing the song, there is the possibility for him/her to save the data of the score's last game. For doing this we have provided a simple user interface that allows the user to insert a name in a input and then click a submit button in order to confirm the operation. In this part of the interface we can also find a “Restart” button that allows the user to play again the game, from the beginning, so it will redirect the application to the initial main menu.

5 Usage

In order to use the game properly the user has to follow some steps in order to connect the smartphone to the application. For doing this the procedure changes a bit from the two operating systems iOS and Android. Here are the main steps for both the operating systems:

Android:

1. Download and install the MultisenseOSC app on the smartphone and then run it.
2. Insert the IP address of your PC and the port number '12000' in the first page of the interface, then press the "Next" button.
3. Enable the orientation in the main menu of the application.
4. Click the orientation card in order to expand it.
5. Take the smartphone straight in front of the PC and click the "OFFSET" button in order to calibrate the device properly.
6. Start the OSC clicking the start button on top.
7. 7. If you have done everything right you will see the application like in the figure below.

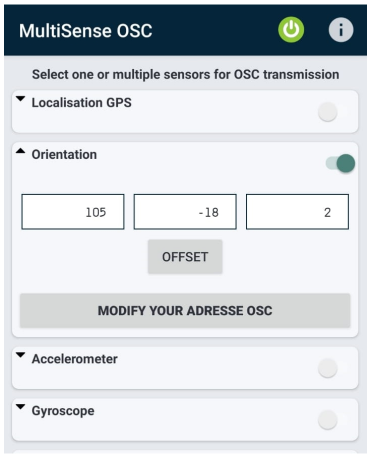


Figure 3: MultiSense OSC application for Android.

iOS:

1. Download and install the GyrOSC app on the smartphone and then run it.

2. Insert in the ‘Target IP Address’ field, the IP address of your PC, then the port number ‘12000’ and finally ‘Computer’ in the ‘Tag” textcell.

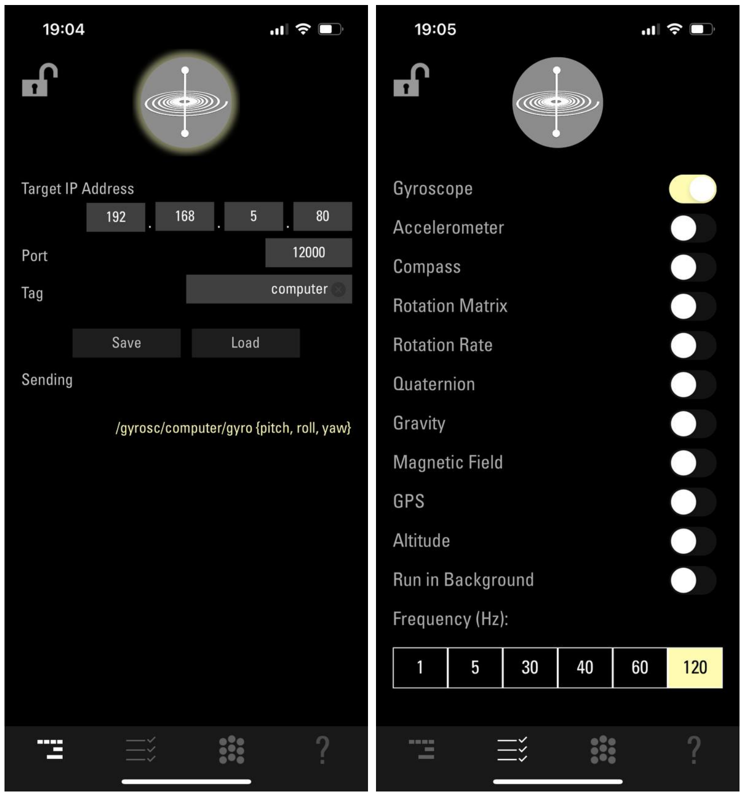
3. Move to the second window from the left, clicking the second image present in the bottom bar.

4. Switch to activate the ‘Gyroscope’ field and do not activate anyone of the others.

5. Set the frequency to ‘120 Hz’.

6. Take the smartphone straight in front of the PC, parallel to the floor and then click the top icon to calibrate the device.

7. Now back in the first window you will see in the ’Sending’ section the following line: /gyrosc/computer/gyropitch, roll, yaw



Target IP Address 192 . 168 . 5 . 80
Port 12000
Tag computer
Sending
/gyrosc/computer/gyro {pitch, roll, yaw}

Gyroscope	●
Accelerometer	○
Compass	○
Rotation Matrix	○
Rotation Rate	○
Quaternion	○
Gravity	○
Magnetic Field	○
GPS	○
Altitude	○
Run in Background	○

Frequency (Hz): 1 5 30 40 60 [✓ 120]

Figure 4: First and second windows of the GyrOSC app for iOS

6 Implementation

6.1 Instruments

The available instruments have been coded in **Supercollider**. The instruments are not original creation of the developers of this project, though they have been adapted to be controlled with the same set of parameters; credits go to the original developers:

1. Piano - FM Rhodes by snappizz
2. Bell - risset bells by Fredrik Olofsson
3. Flute - farfisa organ like sound by nicolaariutti (it sounds more like a flute rather than an organ)
4. Guitar - guitar feedback by snappizz
5. Tri - Zelda's select theme by Sacco_Belmonte

6.2 Song format

The available songs are loaded as MIDI files, stored in the **data** folder, that have been selected and adapted by the developers. The code looks for the first track and loads all the **Note - On** messages (up to a max hardcoded number), discarding chords: it only loads the first note of the chord (it discards the next notes with the same timestamp). It also forces the first note to have timestamp equal to zero (of course, it properly anticipates the next notes).
The timestamps are then converted to ball position by scaling them according to a configuration file, so that the game is playable. All balls (each one representing a note) are forced to stay in the same octave on the screen using modulo 12; keep in mind that the notes actually played are the original notes, not the constrained ones.

6.3 Save file format

The save file is used to store the high scores of the users; the following information is saved:

1. Username
2. Song name
3. Score

The results are sorted by descending score. The file is basically a **CSV**, in fact the fields are separated by a semicolon.

6.4 Config file format

The config file is used to store song settings: some songs can be played better if slower or faster than original. Also this file is a **CSV**, and we save:

1. Song name
2. Speed

The speed will be used as a scale factor to convert from timestamp to ball position.

6.5 Score management

When an user hits a ball, his combo counter increases by 1 and his score is incremented according to the following rules:

- $combo < 10 \implies \text{new score} = \text{old score} + 1$
- $combo < 20 \implies \text{new score} = \text{old score} + 2$
- $combo < 50 \implies \text{new score} = \text{old score} + 3$
- $combo >= 50 \implies \text{new score} = \text{old score} + 5$

When an user misses a ball, his combo counter is set to 0 and his score is decremented by 1.

6.6 Interaction between phone and processing

The interaction between the user mobile phone and processing has been possible thanks to the oscP5 (OSC implementation for programming environment processing)library, written by Andreas Schlegel, where OSC, acronym for Open Sound Control, is a protocol for communication among computers, sound synthesizers, and other multimedia devices. As a first thing the oscP5 needs to be started in the **setup()** function

```
/* start oscP5, listening for incoming messages at port 12000 */
oscP5 = new OscP5(this, 12000);
```

Figure 5: oscP5 listening for incoming messages at port 12000.

then the incoming messages are forwarded to the oscEvent method where we can see the two different approaches depending on the user's mobile phone operating system and consequently on the different applications used for the OSC comunications

```
/* incoming osc message are forwarded to the oscEvent method. */
void oscEvent(OscMessage theOscMessage) {
  if (!isAndroid) {
    if (theOscMessage.addrPattern().equals("/gyrosc/computer/gyro")) {
      y = -1 + 10 * theOscMessage.get(1).floatValue();
    }
  } else {
    if (theOscMessage.addrPattern().equals("/multisense/orientation/pitch")) {
      y = theOscMessage.get(0).floatValue();
    }
  }
```

Figure 6: oscP5 listening for incoming messages at port 12000.

6.7 Interaction between processing and supercollider

The SuperCollider platform has been used to create the instruments sounds that are being played during the game. In order to interface the Supercollider synthesis engine with Processing, we've used the **SuperCollider client for Processing** library that is available in the library repository of Processing. The client allows the user to initialize, change the arguments and create a synth (as shown in figure 5) that has been already defined inside a SuperCollider project that must be saved in the same folder as the processing one.
Another very important step in the developing of the game was to free the synth after each note execution (every time the playSound function is being called we free the last used synth and initialize a new one). If we didn't do it the nodes reserved for the synths would be occupied, so supercollider wouldn't be able to play any sound. All the thing that have been described above are implemented in the 'playSound' function as follows:

```
void playSound(int n) {
  // free the last synth
  if (last != null)
    last.free();
  // uses default sc server at 127.0.0.1:57110
  synth = new Synth(instrumentStr);

  // set initial arguments
  synth.set("amp", 0.5);
  synth.set("freq", 440 * pow(2, (float)(n - 69) / 12)); // convert midi to frequency

  // create synth
  synth.create();

  last = synth;
}
```

Figure 7: playSound function.