DOI: http://dx.doi.org/10.14236/ewic/EVA2017.41

Upside Down

Bhakti Duran
Texas A&M University
College Station, Texas, USA
Bnd005@tamu.edu

Jinisil Hwaryoung Seo
Texas A&M University
College Station, Texas, USA
Hwaryoung@tamu.edu

1. INTRODUCTION

Upside Down is an interactive sound installation that uses live gyroscope rotation data to rotate an audio environment. There is a binaural spatial sound environment developed from the sensation of being above water that once rotated on its axis is replaced with the sensation of being submersed underneath water. Through audio cues and the support of a visual translation, the audience connects their manipulation of the installation with the virtual sound environment in real-time. The participant is free to rotate and playfully interact with the installation to any degree they choose. In doing so, the sound environment played back through a pair of headphones feels as if it is being rotated around them. The minimal aesthetic design, along with the immersive environment challenge the participant's perception of space.

Figure 1: *Installation shown in gallery with audience*

2. ABSTRACT

The participatory installation *Upside Down* is an aim at challenging the audience's subconsciously passive sense of hearing that builds the perceived world around them. They are embodied by sensorimotor inputs that mirror a physically real environment. How can people enabled visually sense birth overpower their natural perception of space using a trait essential to immersion that has been overshadowed for so long? The spatial sound environment is meant to trigger our subconsciously alienated trait and bring it to the forefront. The project represents a dynamic participatory installation that essentially rewires the brain to focus on their perception of space by not only reproducing the illusion of realism within this virtual environment, but also the motor capability of influencing said environment through the manipulation of spherical installation in the audience's current reality.

3. TECHNICAL IMPLEMENTATION

To create this participatory installation, we made a patch within Max/MSP, a visual programing language for music and multimedia, that would be able to receive and manipulate real-time gyroscope data from an Arduino device. After receiving the live gyroscope data, the Max patch would connect the incoming coordinates to two stereo channels placed in a three-dimensional sphere using the spatial audio Max objects. These two channels are offset 180 degrees from one another to mimic the position of the ears. The channels then rotate on the X and Y axis corresponding with the movements of the Arduino gyroscope. The stereo channels output a looped ambient water environment audio file to a pair of connected headphones.

The audio file is created within a separate program called VibeStationLE. Within this software we develop our own representation of an ambient water environment by placing audio cues from

sound libraries in three-dimensional space. Just like Maya or 3ds Max for animation, VibeStation gives us control of a virtual three-dimensional space in which we can construct and then render a binaural or multi-channel audio track.

We updated the Arduino FLORA code for Max/MSP to send coordinate numbers we could understand and manipulate easily. From there we used conductive thread to sow on to a piece of felt the FLORA and gyroscope.

We placed the felt within a Styrofoam orb that we fitted to hold inside the centre. The Arduino cable would connect from there to our laptop, which is running the Max/MSP program.

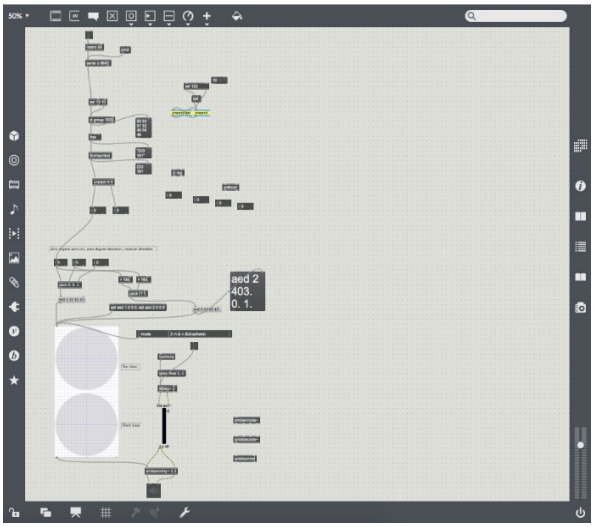

Figure 2: *Max/MSP patch development*

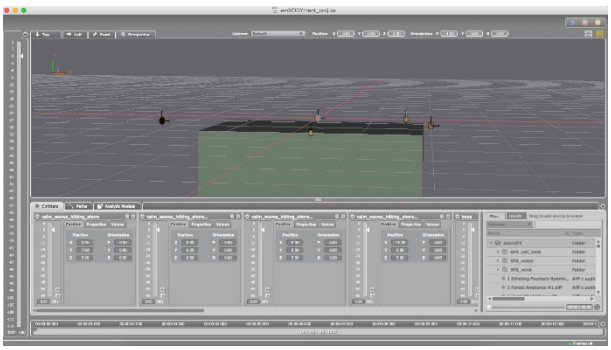

Figure 3: *VibeStationLE spatial audio environment*

```
ahrs_lsm9ds0_hwaryoung_

// Initialise the LSM9DS0 board.
if(!lsm.begin())
{
  // There was a problem detecting the LSM9DS0 ... check your connections
  Serial.print(F("Ooops, no LSM9DS0 detected ... Check your wiring or I2C ADDR
  while(1);
}

// Setup the sensor gain and integration time.
configureLSM9DS0();
}

void loop()
{
  sensors_vec_t   orientation;

  // Use the simple AHRS function to get the current orientation.
  if (ahrs.getOrientation(&orientation))
  {
    /* 'orientation' should have valid .roll and .pitch fields */
    /*Serial.print(F("Orientation: "));
    Serial.print(orientation.roll);
    Serial.print(F(" "));
    Serial.print(orientation.pitch);
    Serial.print(F(" "));
    Serial.print(orientation.heading);
    Serial.println(F(""));*/
    int mapValueX = map(orientation.roll,-180, 180, 0, 360);
    int mapValueY = map(orientation.pitch,-180, 180, 0, 360);

    Serial.print(mapValueX); Serial.print(" ");Serial.println(mapValueY);
  }

  delay(100);
}
```

Figure 4: *Arduino code development*

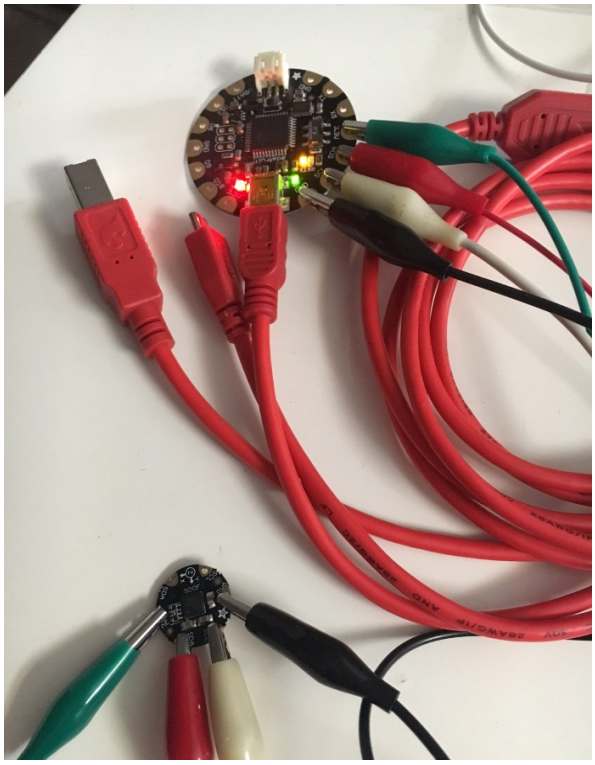

Figure 5: *Arduino FLORA testing*