**Hi.LtC – Listen to the City**

**What it is**

Hi.LtC (“Listen to the City”) is a website that allows people to experience cities through an abstract form of audio. This first version is Manhattan.

**What it does**

The site takes open building-related data from an API provided by the City of New York. It extracts the height of each building from the data, then converts that height value into a musical note. It then plays these notes in order, forming a unique melody for each city.

**Story behind Hi.LtC**

As an NYU student who’s never been NYC, I feel like I’m missing out on an important part of grad school life, so I decided to find another way to experience the Big Apple. Binging Sex in the City didn’t seem quite “ITP” enough for my taste.

Some of my classmates are in the music industry. Watching them use sound to present their ideas was really inspiring, so for our first project I decided to play around with sound production. When thinking about what kind of data I could use to create sounds, I wanted to go with an atypical connection – mixing different senses or unrelated concepts.

Cityscapes are one of the most iconic things about a city, and no cityscape is more iconic than Manhattan’s. I wanted to see what the city would sound like if I treated each of the buildings in its cityscape as a musical note. This would allow both myself and others to experience NYC in a different way.

**Data & Coding**

For Mathura’s classes I learned how to fetch JSON data using Javascript, and also how to use APIs.

For data, I found a website <https://opendata.cityofnewyork.us/>. It’s an open-source website that includes all sorts of data about New York City – everything from traffic data to crime data and building data. I found a file that had information about every building in Manhattan, and one of the data categories was height.

This is the code I used to extract the building data from the JSON file:

let buildingArray;

window.addEventListener('load', function () {

fetch("https://data.cityofnewyork.us/resource/hu6m-9cfi.json")

.then(response=>response.json())

.then(data => {

console.log('savingJson');

buildingArray = data;

});

After that, I created a button listener that would start looping through the building height data when a user clicked on the button on the website:

let button = document.getElementById('play-button');

button.addEventListener('click', function(){

In this block of code, I had to also do several other things: create an AudioContext object that would allow me to produce sound. Store some values like startDelay and duration that would allow me to manipulate the timing of each note. And then create a for loop that would go through each building object and extract its height, then convert it into a note.

I also had to create oscillator objects that play the notes. I had to choose between four different types of waves, and I decided to use square waves because they sounded the coolest, like old video game music.

var audioContext = new AudioContext();

let noteAdjustment = 100;

let startDelay = .27;

let duration = .25;

for(let i = 0; i < buildingArray.length/10; i++){

if(buildingArray[i].stories != null){

console.log(buildingArray[i].stories);

let buildingHeight = buildingArray[i].stories;

let buildingNote = buildingHeight \* noteAdjustment;

const oscillator = audioContext.createOscillator();

var gainNode = audioContext.createGain();

oscillator.connect(gainNode);

gainNode.connect(audioContext.destination);

gainNode.gain.value = 0.01;

oscillator.type = "square";

oscillator.start(audioContext.currentTime+(startDelay\*i));

oscillator.frequency.value = buildingNote;

oscillator.stop(audioContext.currentTime+duration+(startDelay\*i));

}

}

})

})

I had a very difficult time playing the notes in order. The program was going through the loop so fast that it was playing all of the notes at the same time. Fortunately, Craig showed me how to delay each note so Hi.LtC could play them in order.

After I solved timing issues, I was having trouble making the code play each individual note. It just played the same note over and over. When reviewing the documentation for these objects, I realized that I made a mistake on this line:

oscillator.frequency.value = buildingNote;

In the original version, I had written: oscillator.frequency = buildingNote;

I realized that you can’t set the frequency directly because the frequency is an object. I had to set the frequency value instead.

After I changed that line, it worked great!