Cosmic Whispers
Programación creativa aplicada a la creación de música generativa

Hugo Villanueva Duque

ÍNDICE

Motivación

Conocer la Programación creativa

Combinar conocimientos musicales con informáticos

Explorar la creación de música

Implementarlo de forma estética

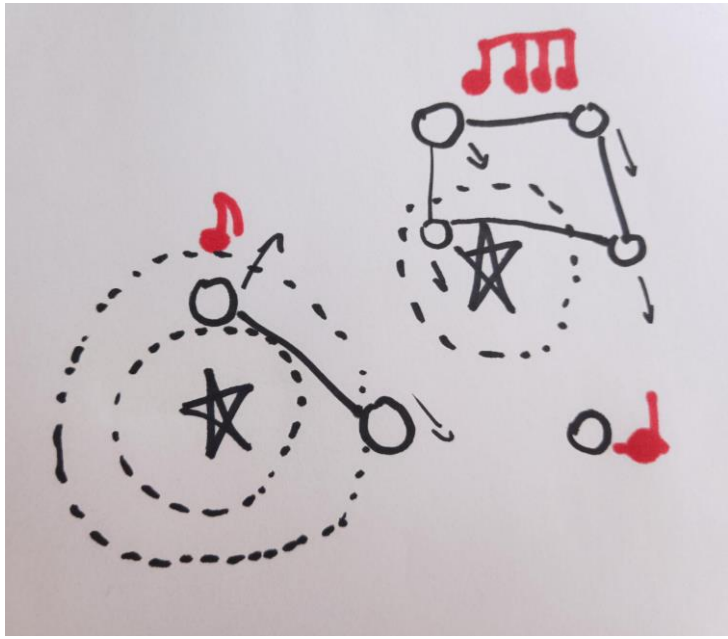
Planteamiento del proyecto

- Disciplinas:
  - Programación Creativa
  - Programación de Audio
  - Programación de Aplicaciones Web
- Investigación:
  - ¿Diseño?
  - ¿Librerías adecuadas?
  - ¿Integración?
- Progreso: Perseverancia con las librerías
  - Incorporación de nuevas funcionalidades

Desarrollo iterativo

Primer concepto:
Generador musical inspirado en el movimiento de los astros

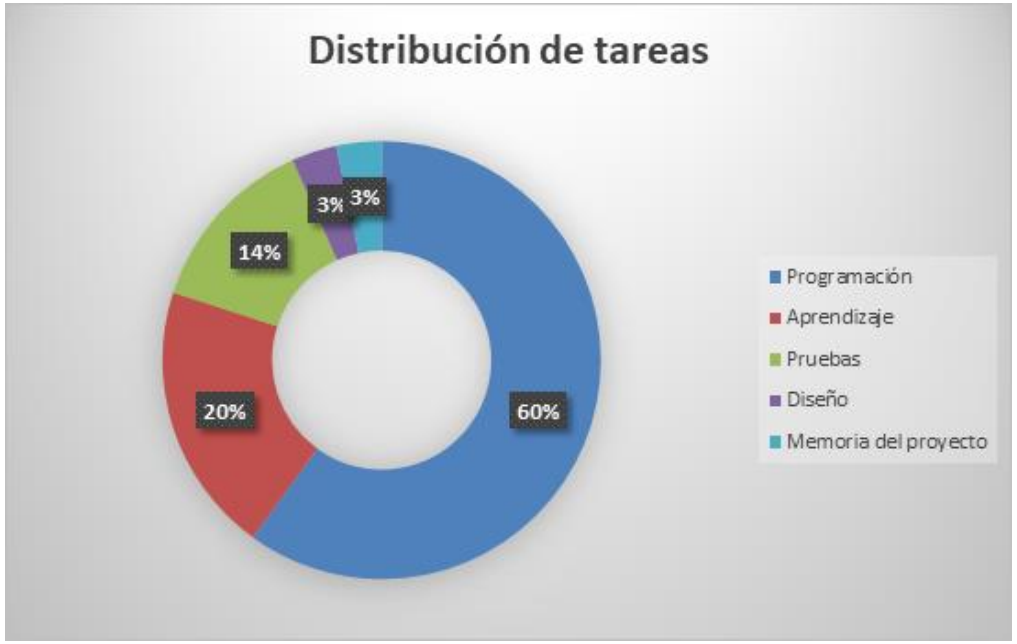
- Búsqueda de nuevos patrones musicales:
  - Cada astro tiene asociada una nota
  - Cada nota suena según la posición de otros astros
  - Se simulan físicas de movimiento
- Más astros ➔ Más entropía musical



Proceso de Desarrollo

Versión	Implementaciones añadidas
0.1	Puntos blancos que se mueven por el espacio, que crean conexiones al estar cerca
0.2	Los puntos tienen ahora color aleatorio al crearlos
0.3	Cada punto emite una nota al ser creado, dicha nota es asociada aleatoriamente en la creación, usando un sintetizador creado en Tone.js
0.4	Se implementa la lógica de generadores.
1.0	Se implementa la opción de selección de notas en la parte inferior de la pantalla, y la interfaz GUI para control de variables, junto con las escalas. La aplicación ya cumple su cometido principal original
1.1	Se implementa el Theremin. Mejoras estéticas
1.2	Se implementa el Sound Design Panel, y los Offset para los generatos
1.3	Se implementa el fondo animado
1.4	Arreglo de bugs. Mejoras estéticas y organización de los archivos

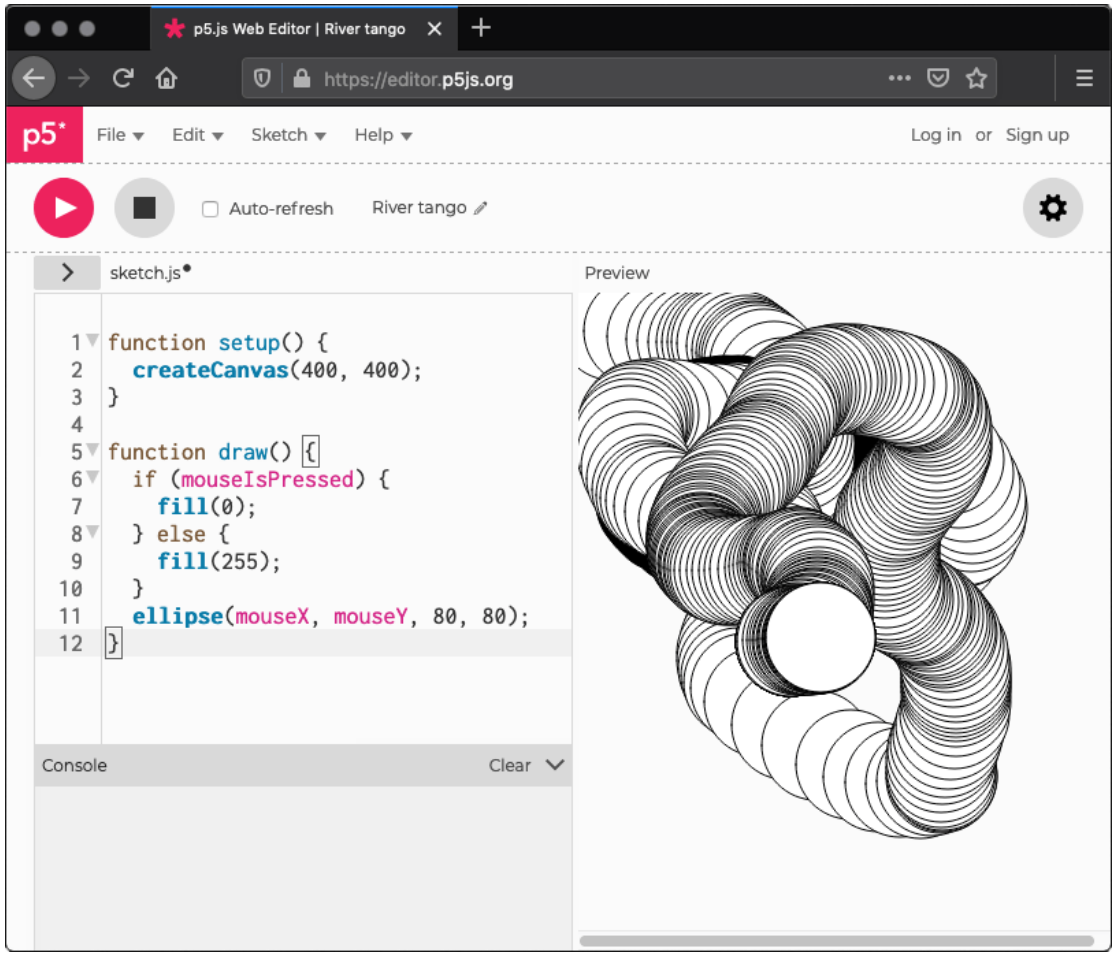
Desarrollo iterativo



Tecnologías utilizadas

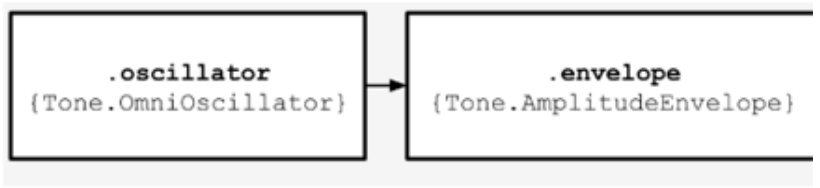
p5.js

- Estándar de programación creativa
- Dibujo de geometría
- Control de eventos
- Input en navegador
- Simulación de físicas



Tone.js

- Sintetizador:
  - Generación de onda
  - Diseño de sonido
  - Efectos
- Control de tiempo y ritmo



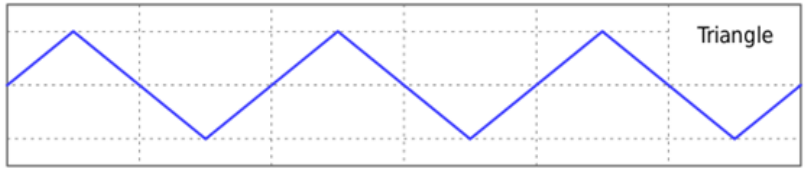
CONSTRUCTOR

```
new Tone.Synth ( [ options ] )
```

options — the options available for the synth see defaults
TYPE: Object
OPTIONAL

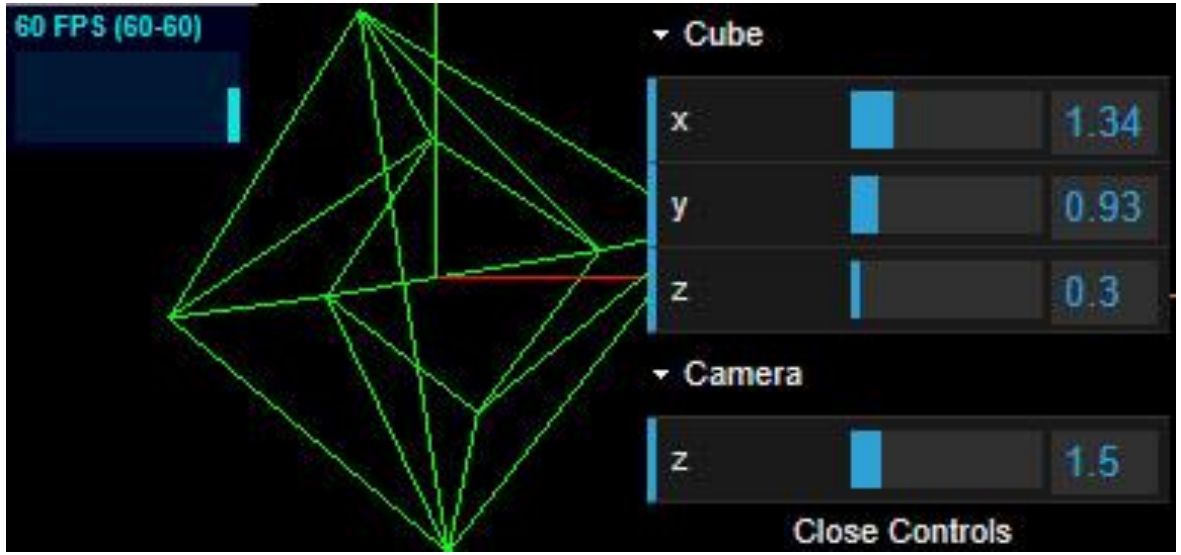
DEFAULTS

```
{
   oscillator  : {
      type  : triangle
   },
   envelope  : {
      attack  : 0.005,
      decay  : 0.1,
      sustain  : 0.3,
      release  : 1
   }
}
```



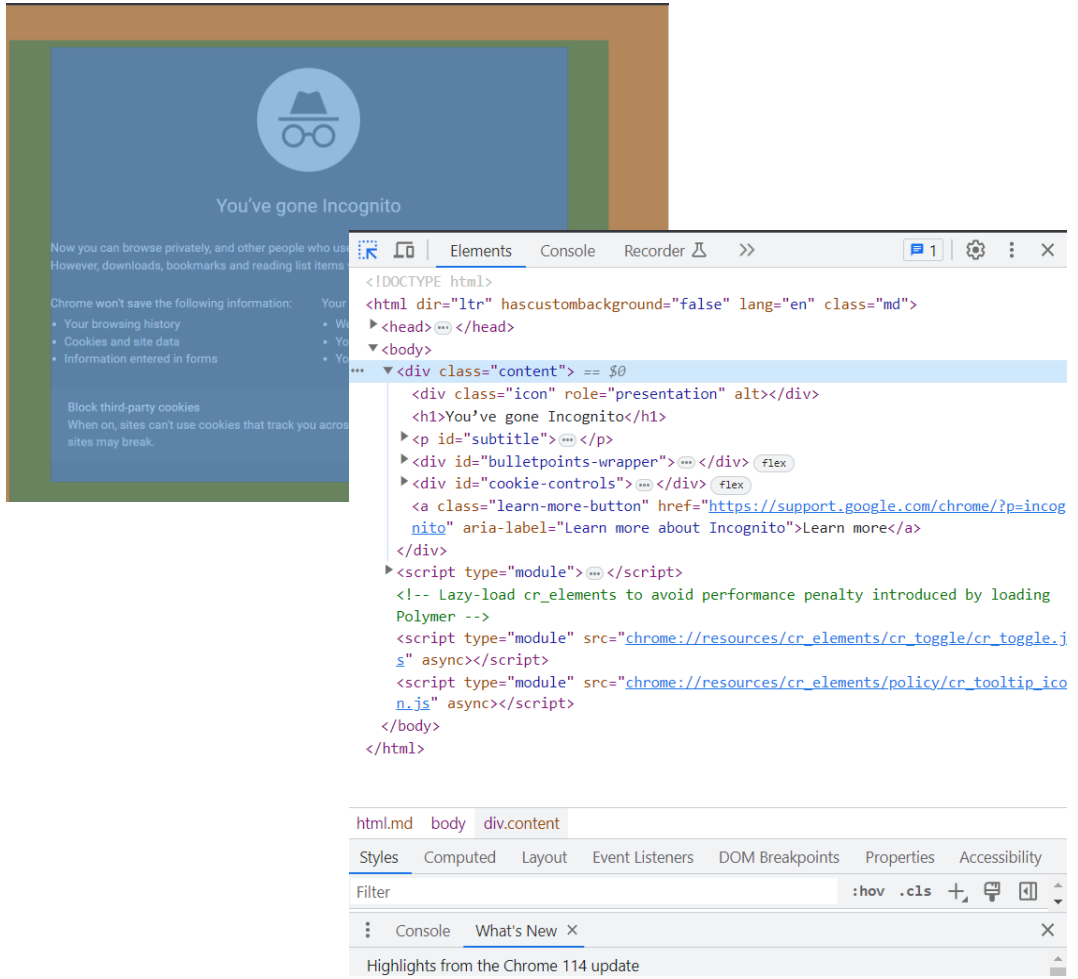
dat.GUI

- Control de variables en tiempo real
- Diseño propio de UI
- Implementación simple
- Ligero y eficiente



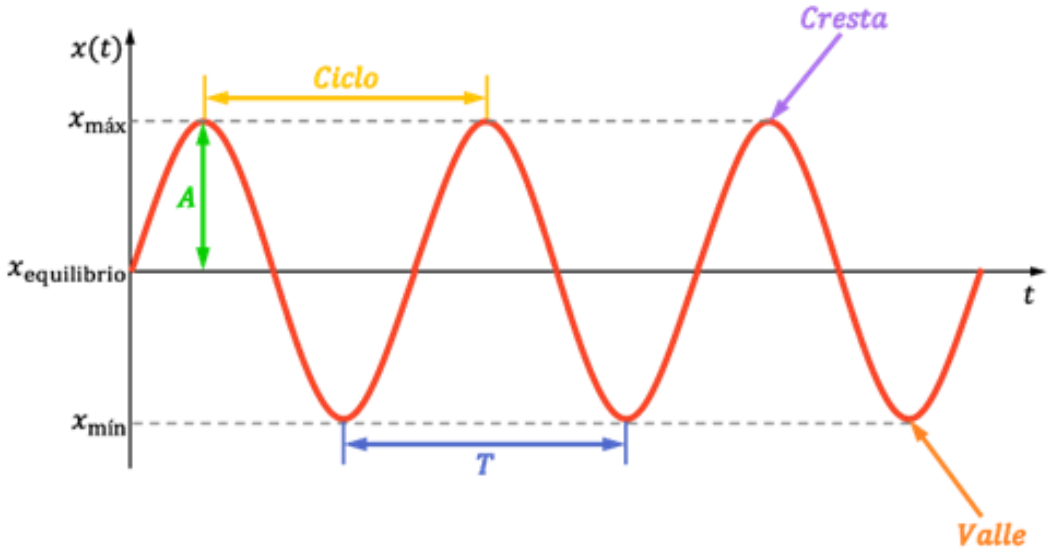
HTML + CSS

- Lienzo para JS
- Menús estáticos
- Iframes para paneles
- Usado durante los estudios




Diseño

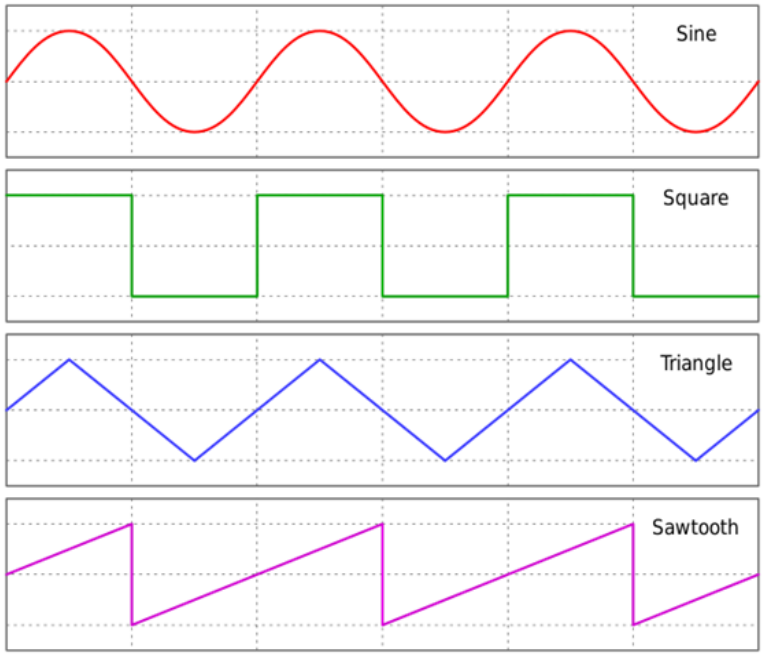
Generación de notas

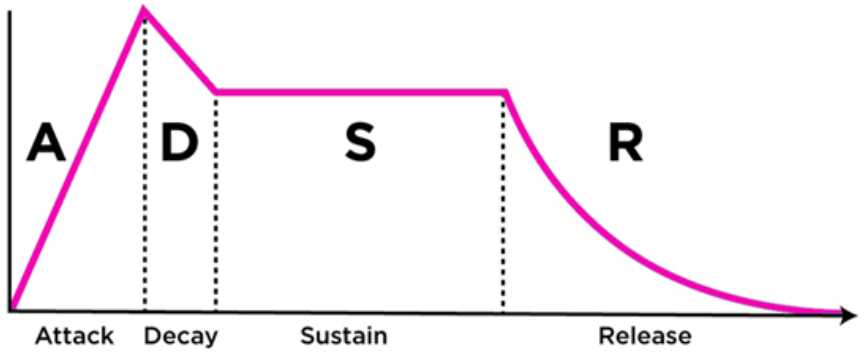


Una nota no es más que un oscilador configurado a una frecuencia determinada

Equal-tempered	
Note	*Hz*
C	523.25
B	493.88
$B^b/A^\#$	466.16
A	440.00
$A^b/G^\#$	415.30
G	392.00
$G^b/F^\#$	369.99
F	349.23
E	329.62
$E^b/D^\#$	311.13
D	293.66
$D^b/C^\#$	277.18
C	261.63

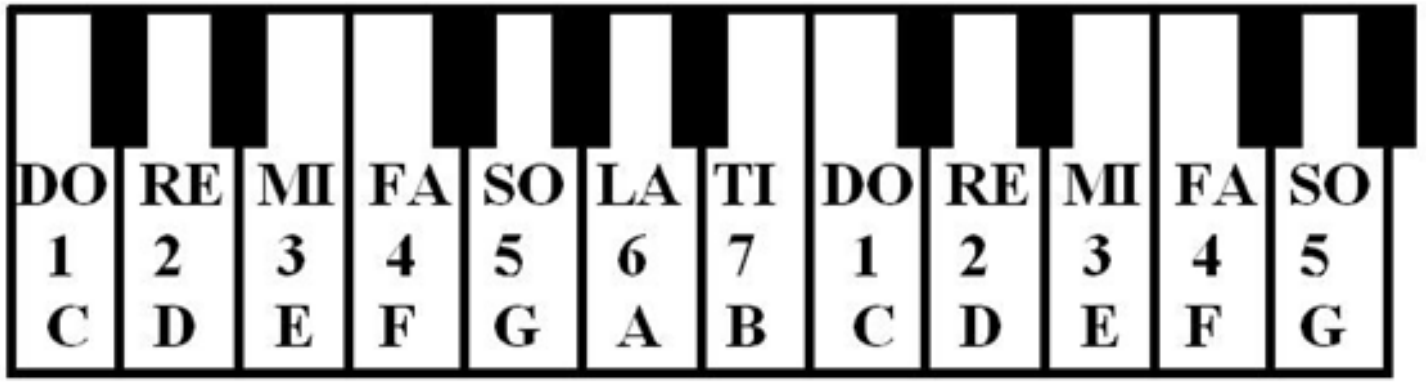
Generación de sonido

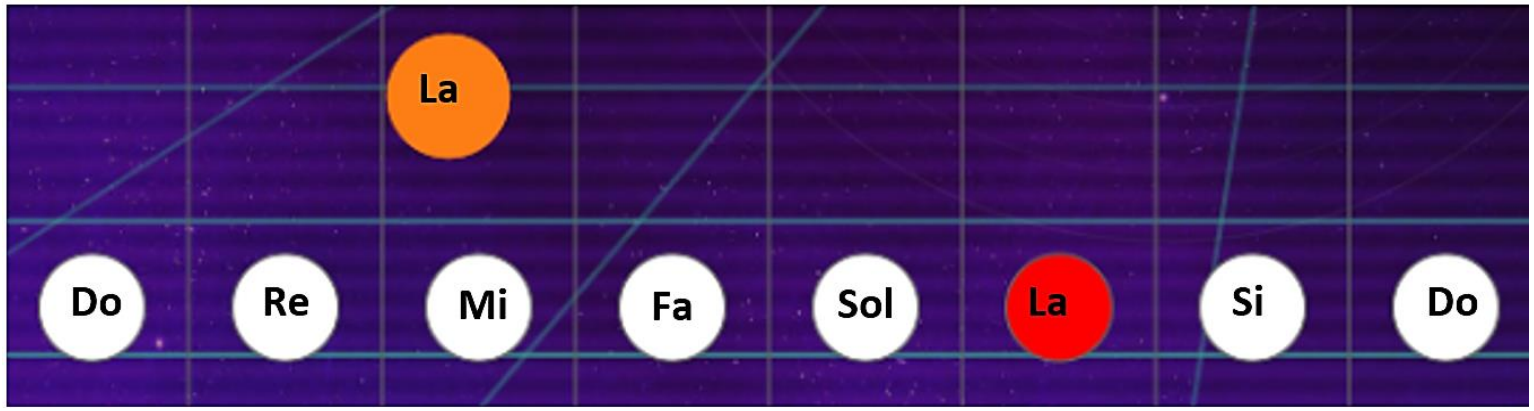





```
Synth = new Tone.Synth({
  oscillator: {
    type: "sine",
  envelope: {
    attack: 0.2,
    decay: 0.2,
    sustain: 0.5,
    release: 15,
}}).toDestination();
```

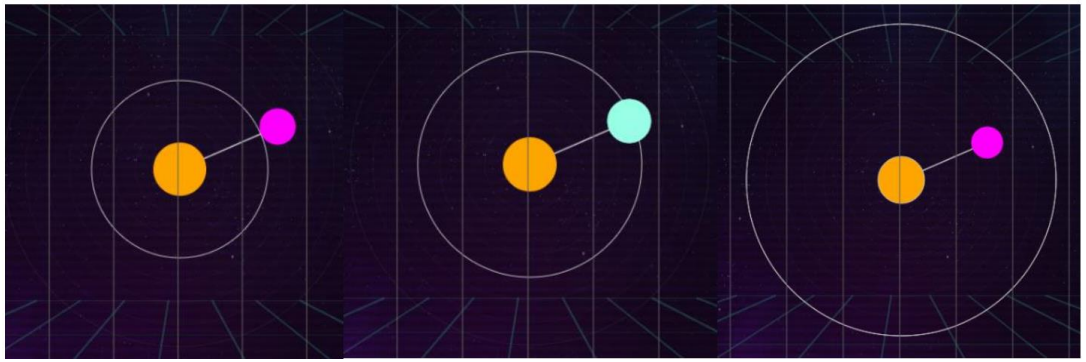
Notas musicales

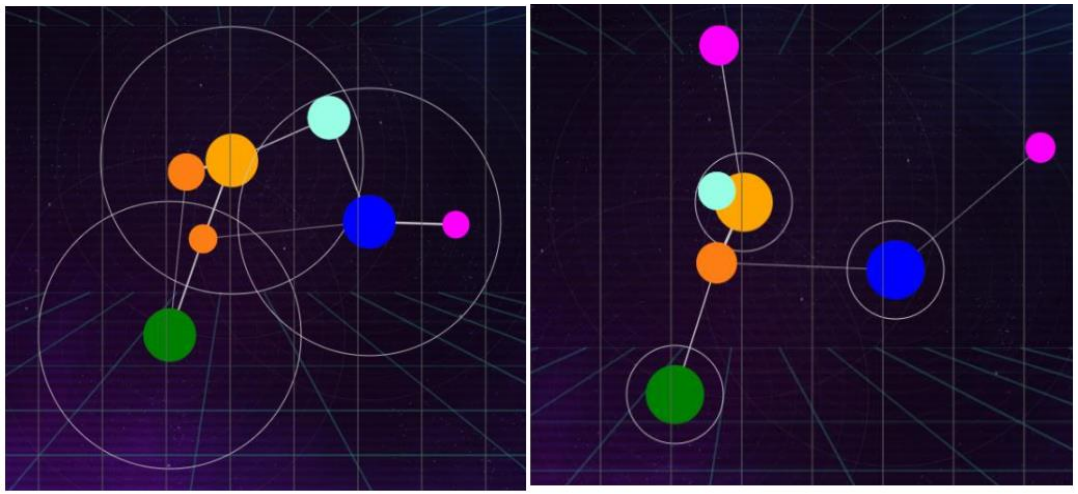




Generadores

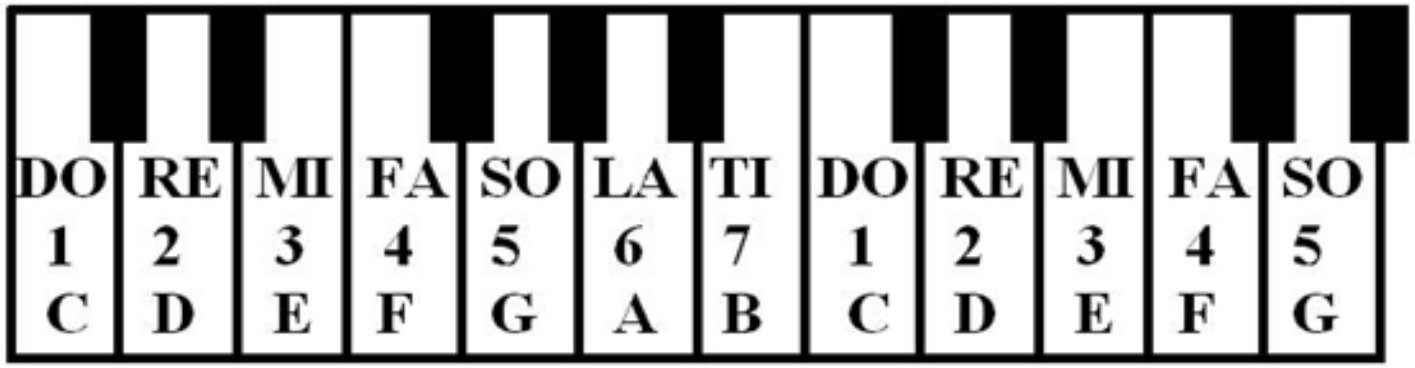
- La órbita reproduce los nodos
- 8 tiempos disponibles
- 3 tipos de generadores
  - Octava baja
  - Octava media
  - Octava alta





Escalas

- Conjunto de notas a utilizar
- Sensaciones diferentes



Escala Mayor (Do M)

Do	Re	Mi	Fa	Sol	La	Si	Do
0	2	4	5	7	9	11	12

Escala Menor (Do m)

Do	Re	Mi ♭	Fa	Sol	La ♭	Si ♭	Do
0	2	3	5	7	8	10	12

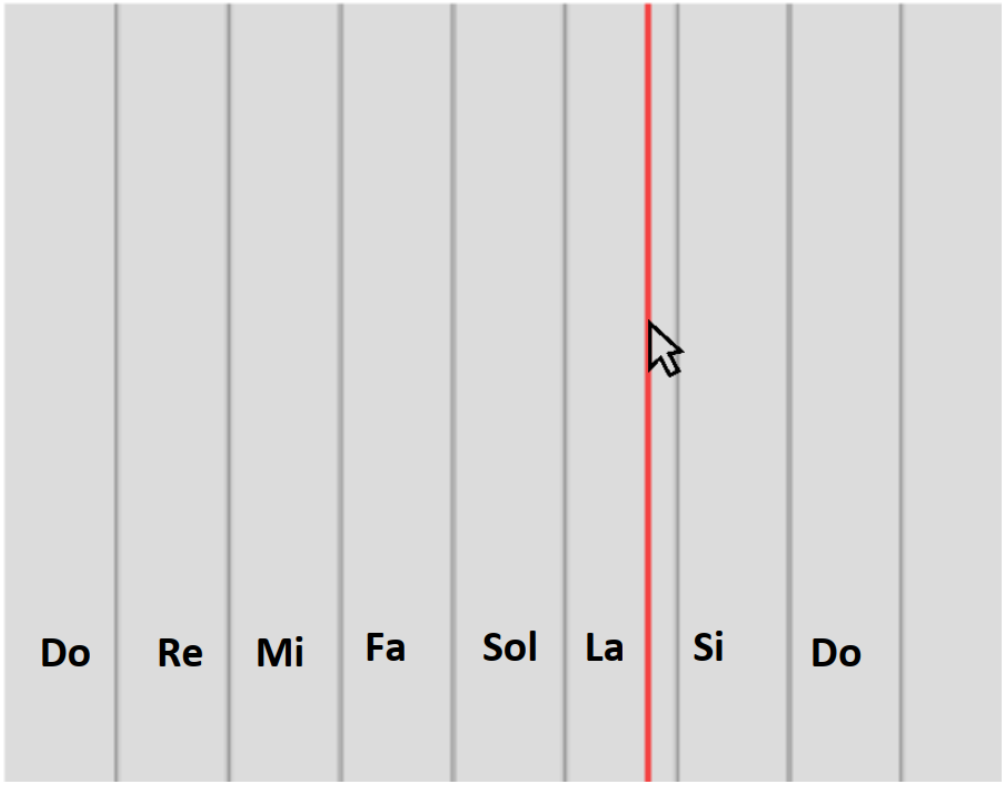
Escala Pentatónica

Do	Re	Mi	Sol	La	Do
0	2	4	7	9	12

```
const scales = {
  Major: [0, 2, 4, 5, 7, 9, 11, 12, 14, 16, 17, 19, 21, 23],
  Natural_Minor: [0, 2, 3, 5, 7, 8, 10, 12, 14, 15, 17, 19, 20, 22],
```

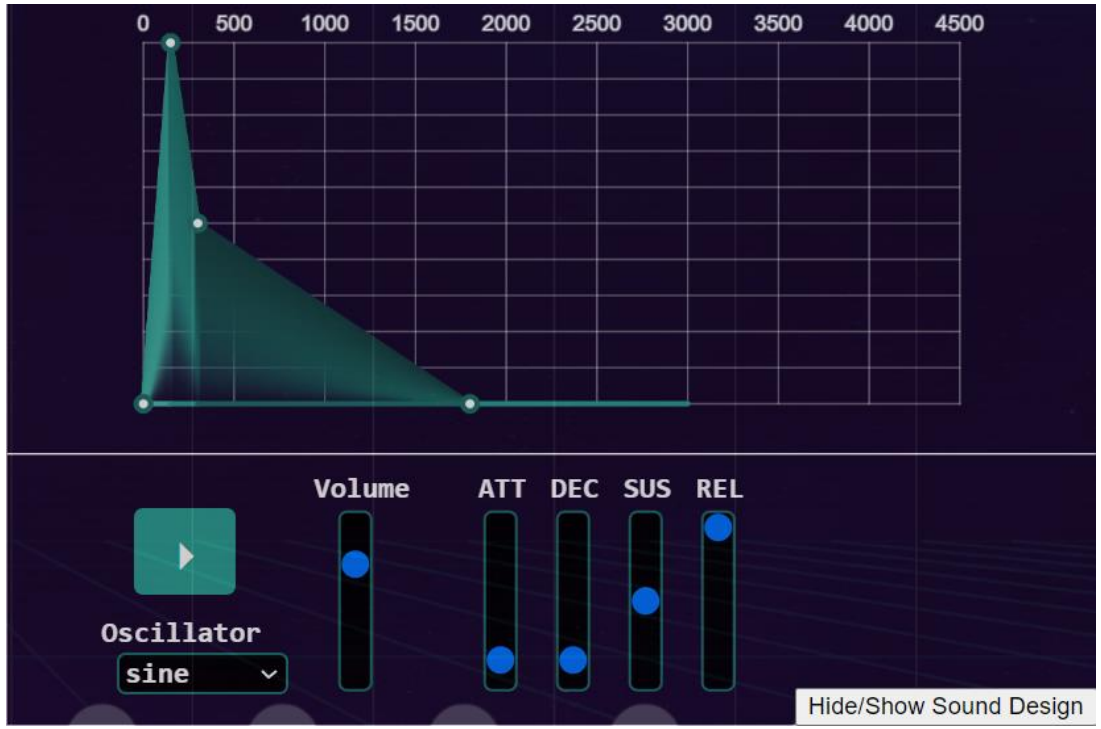
Theremin

- Control manual
- Sonido constante, con el ratón
  - Eje horizontal: Tono
  - Eje vertical: Volumen



Panel de Diseño de Sonido

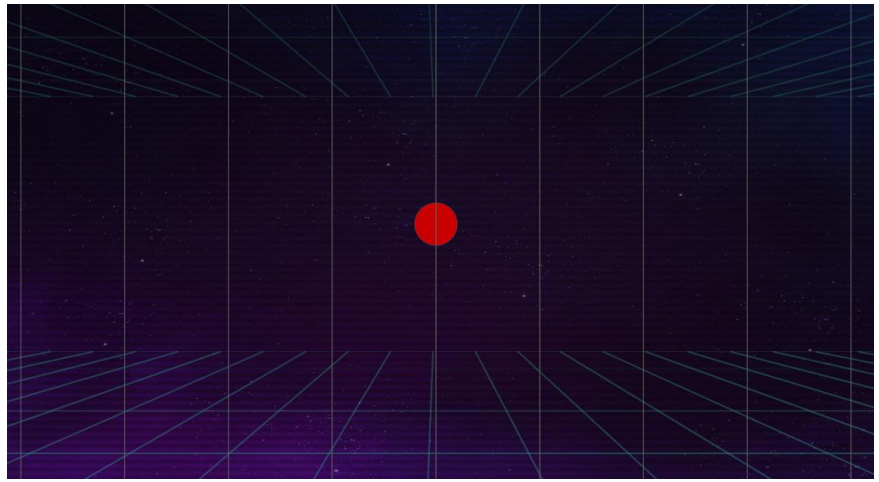
- Control en tiempo real del sintetizador
- Visualización de onda y ADSR
- Para Theremin y Notes



Fondo Interactivo y metrónomo

- Estética neón
- Superposición con imagen fija
- Interactividad: Aceleración

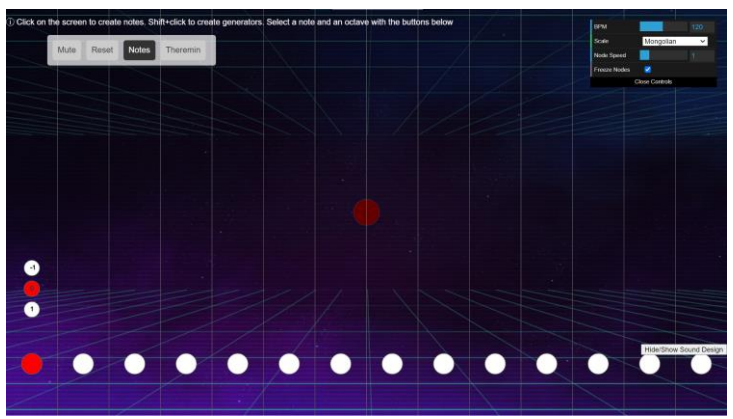
- Metrónomo: 4 parpadeos

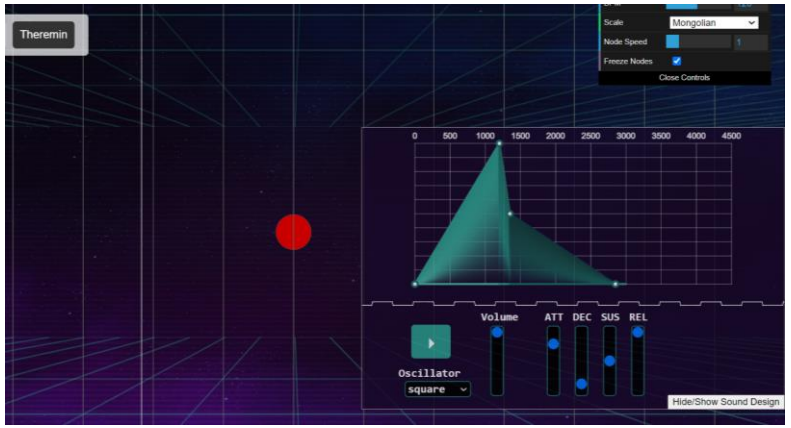


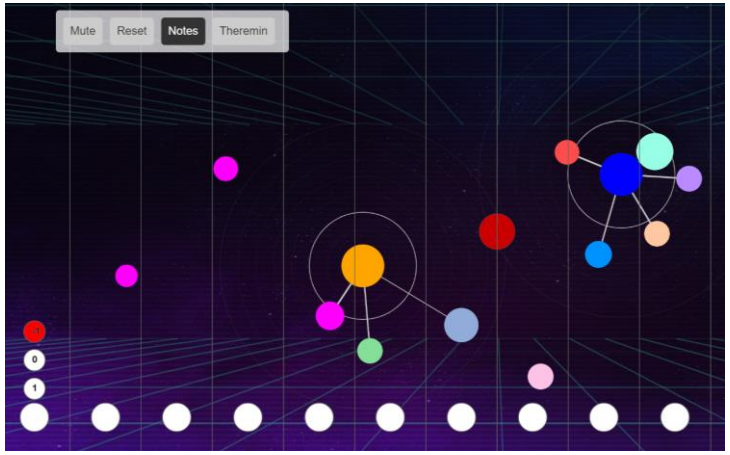
Demostración

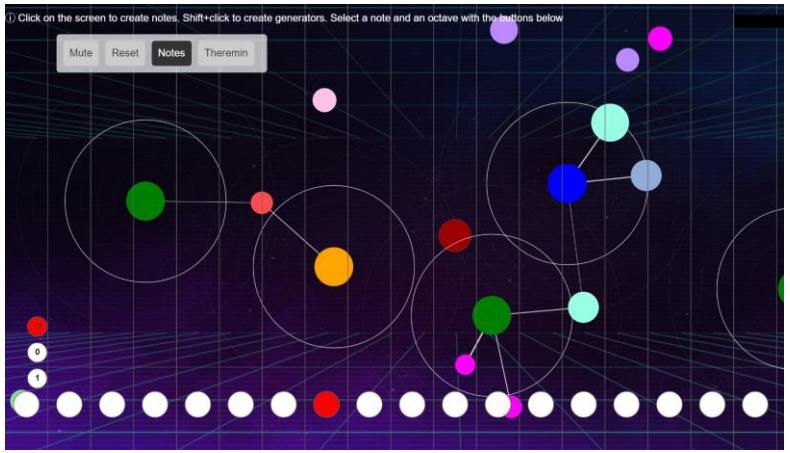
Material

Proyecto: https://cosmicwhispers.000webhostapp.com/
Vídeo: https://www.youtube.com/watch?v=mMwEvSCUG-k









Implementación

Configuración de sintetizador en Tone.js

Sintetizador (Kalimba)

```
let polySynth = new
Tone.PolySynth(Tone.FMSynth, {
  "harmonicity": 8,
  "modulationIndex": 2,
  "oscillator": {
    "type": "sine"
  },
  "envelope": {
    "attack": 0.01,
    "decay": 2,
    "sustain": 0.1,
    "release": 1
  },
  "modulation": {
    "type": "square"
  },
  "modulationEnvelope": {
    "attack": 0.002,
    "decay": 0.2,
    "sustain": 0,
    "release": 0.2
  }}).toDestination();
```

Efectos de sonido

```
// High Pass Filter
const highpassFilter = new
Tone.Filter({
  type: "highpass",
  frequency: 1000,
  Q: 100
});
```

Limpieza de agudos

```
// Reverb
const reverb = new Tone.Reverb({
  decay: 2,
  preDelay: 0.01,
  wet: 0.5
});
```

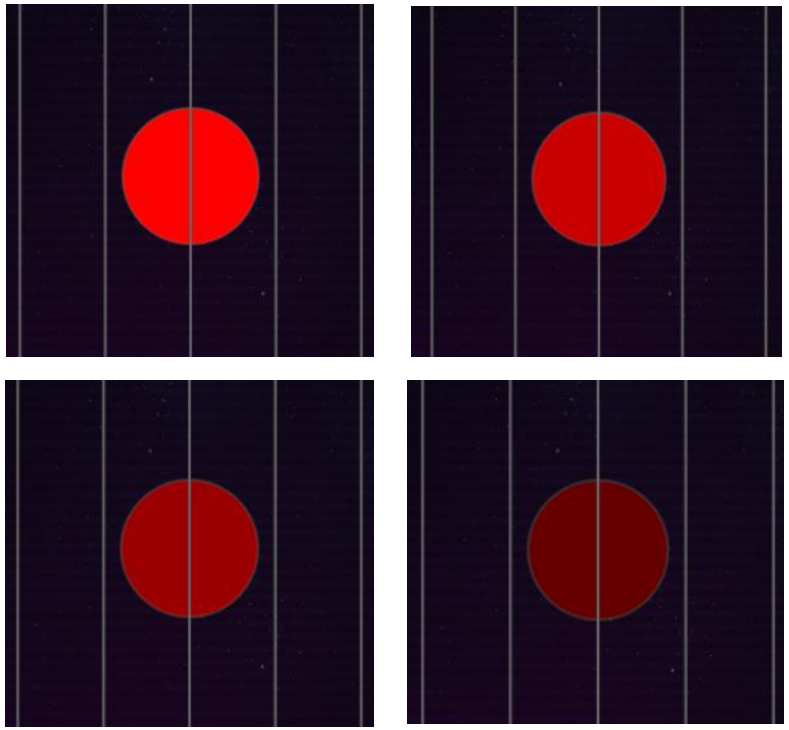
Reverb

Conexión del pipeline

```
polySynth
.connect(compressor)
  .connect(reverb)
  .connect(echo)
  .connect(lowpassFilter)
  .connect(highpassFilter)
```

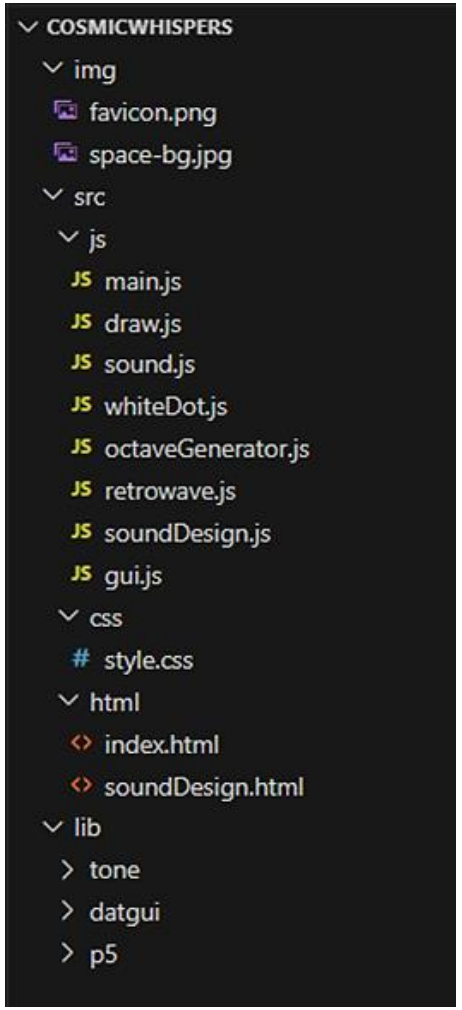
Ejemplo de función p5.js: drawMetronome

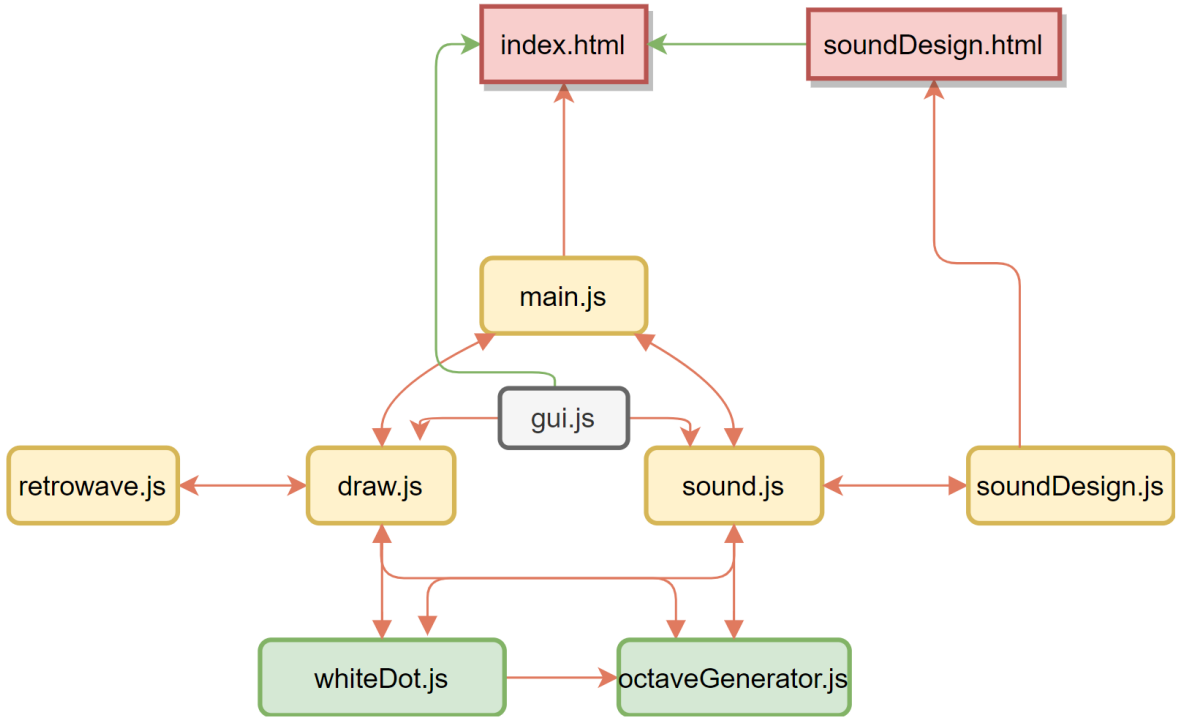
```
function drawMetronome() {
    let currentPosition = Tone.Transport.position.split(":");
    let currentBeat = parseInt(currentPosition[1]);
    beatNum = currentBeat;


    let metronomeSize = 50;
    switch (beatNum) {
        case 0:
            fill(255, 0, 0); //red color for the first beat
            break;
        case 1:
            fill(202, 0, 0);
            break;
        case 2:
            fill(155, 0, 0);
            break;
        default:
            fill(102, 0, 0);
    }


    ellipse(width / 2, height / 2, metronomeSize);
}
```



Función que utiliza funciones de p5, y que recibe variables calculadas por Tone.js

Sistema de archivos





11 archivos de código, 1519 Líneas (excl. librerías)

Conclusiones

Conclusiones

Fuentes y Bibliografía

- **Imágenes y conceptos teóricos**

  - Bypeople (n.d.). Tone.js. Recuperado de https://www.bypeople.com/tone-js-music-framework/
  - Sbcode.net. (2019). Dat GUI. Recuperado de https://sbcode.net/threejs/dat-gui/
  - p5. (2015). Your First Sketch. Recuperado de https://p5js.org/get-started/
  - Ingenierizando. (2021). Ciclo de una Onda. Ingenierizando. Recuperado de: https://www.ingenierizando.com/cinematica/ciclo-de-una-onda/
  - Nugent, J. (2020). What is a Synthesizer? Higher Hz. Recuperado de: https://higherhz.com/what-is-synthesizer/
  - Wikipedia. (s.f.). Sonido. Recuperado de: https://es.wikipedia.org/wiki/Sonido
  - Forster, J. (2015). How Many Ways Can a Piano Be Out of Tune? Sterling Piano Tuning. Recuperado de: https://www.sterlingpianotuning.com/how-many-ways-can-a-piano-be-out-of-tune/

Fuentes y Bibliografía

- **Documentación**
  - Mozilla Developer Network. (s.f.). HTML Documentation. Recuperado de: https://developer.mozilla.org/en-US/docs/Web/HTML
  - Mozilla Developer Network. (s.f.). CSS Documentation. Recuperado de: https://developer.mozilla.org/en-US/docs/Web/CSS
  - p5.js. (s.f.). p5.js Reference. Recuperado de: https://p5js.org/reference/
  - Tone.js. (s.f.). Tone.js Documentation. Recuperado de: https://tonejs.github.io/docs/14.7.77/index.html
- **Librerías**
  - Tone.js: https://github.com/Tonejs
  - p5: https://github.com/processing/p5.js
  - dat.gui: https://github.com/dataarts/dat.gui

Cosmic Whispers
Programación creativa aplicada a la creación de música generativa

Hugo Villanueva Duque