Generative tabla on web w/ p5.js

Keshav Joshi

Email: kjoshfree@gmail.com
GitHub: kmjoshi
LinkedIn: keshavmjoshi
Twitter: keshavahsek

Who am I?

Georgia Tech

Physicist/Ocean Modeler

Los Alamos NATIONAL LABORATORY EST. 1943

Physicist/Ocean Modeler

UNIVERSITY of GUYANA

Physics Lecturer

gtt+

Data Scientist

UNESCO

United Nations
Educational, Scientific and
Cultural Organization

MGIEP

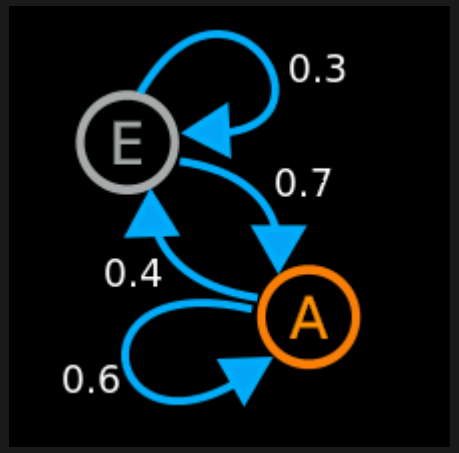
Mahatma Gandhi Institute
of Education for Peace
and Sustainable Development

Data Scientist

What is it?

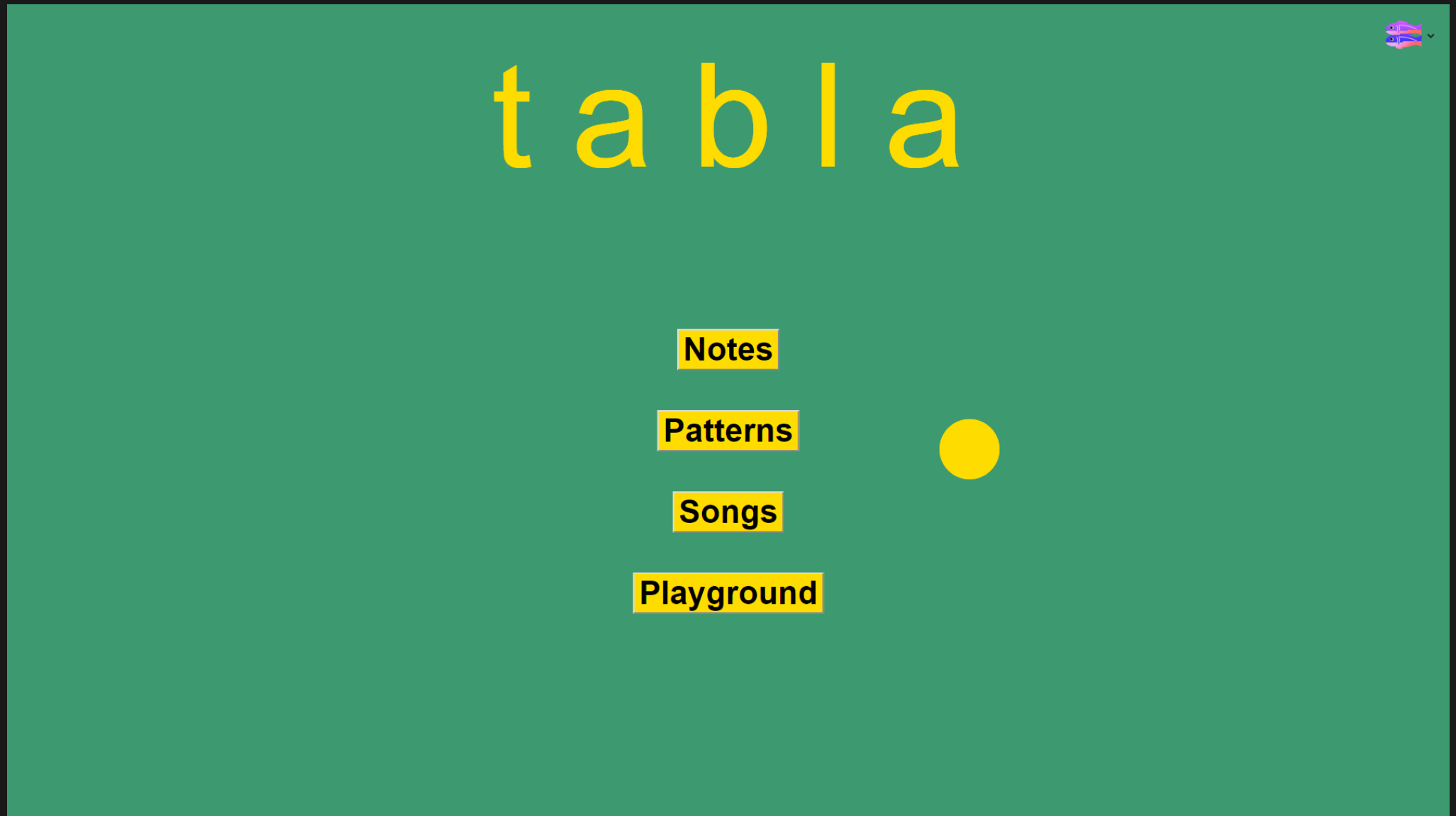
```
},
"compoundNotes": {
    "Tin": { "left": "Ga", "right": "Tun_Di" },
    "Dhin": { "left": "Ghe", "right": "Tun_Di" },
    "Dha": { "left": "Ghe", "right": "Na" },
    "Thun": { "left": "Ka_Ke", "right": "Tun_Di" }
},
"noteFiles": {
    "Ga": "assets/Ga_16.wav",
    "Ghe": "assets/Ghe_16.wav",
    "Na": "assets/Na_16.wav",
    "Tun_Di": "assets/Tun_Di_16.wav",
    "Ka_Ke": "assets/Ka_Ke_16.wav",
    "Ta_Te": "assets/Ta_Te_16.wav",
    "Te_Re": "assets/Te_Re_16.wav",
    "Ne": "assets/Ne_16.wav"
}
```

+



+ p5.js

What is it?

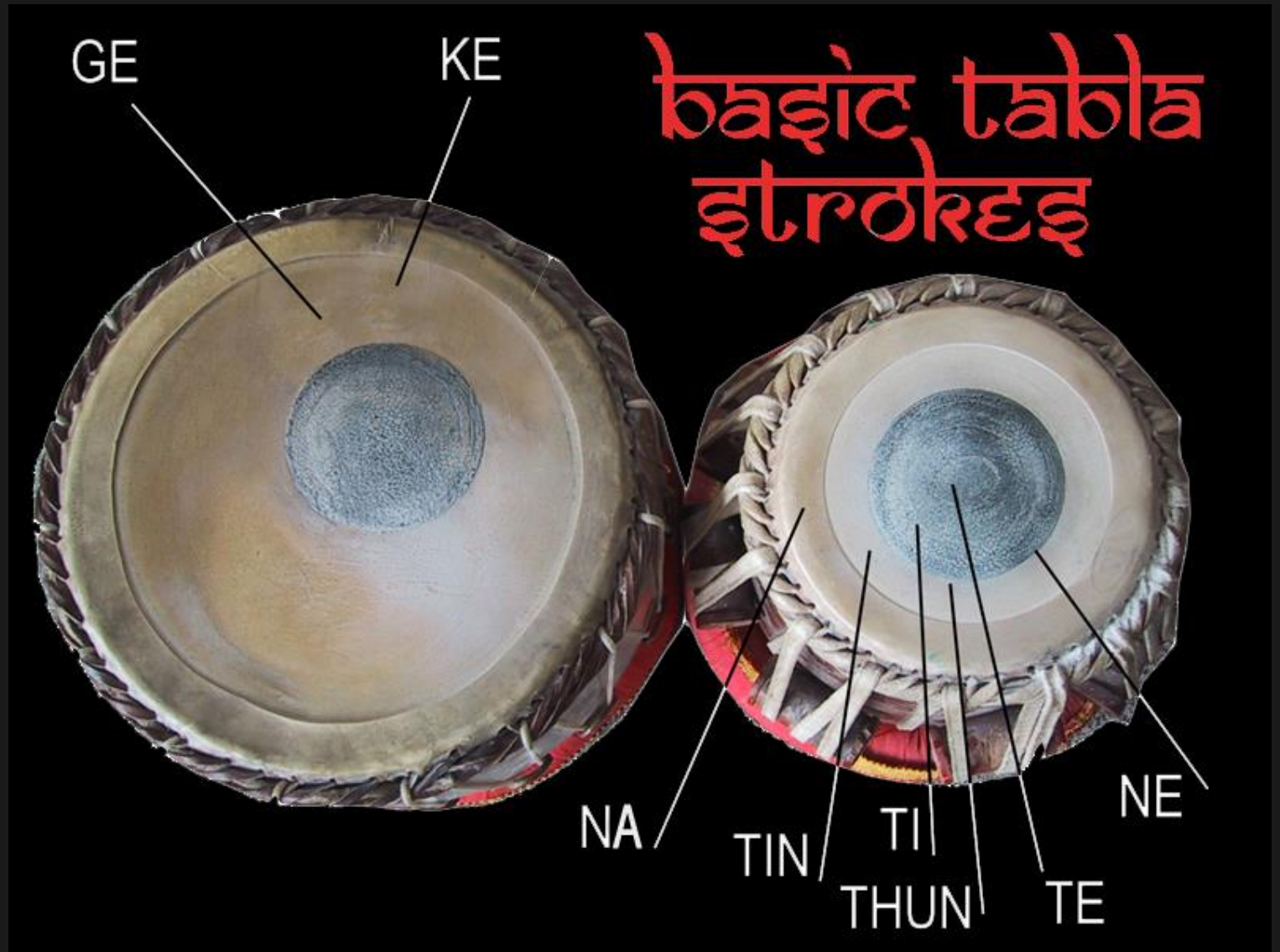


Demo

https://tabla-beats-me-again.glitch.me/

Notes

Notes



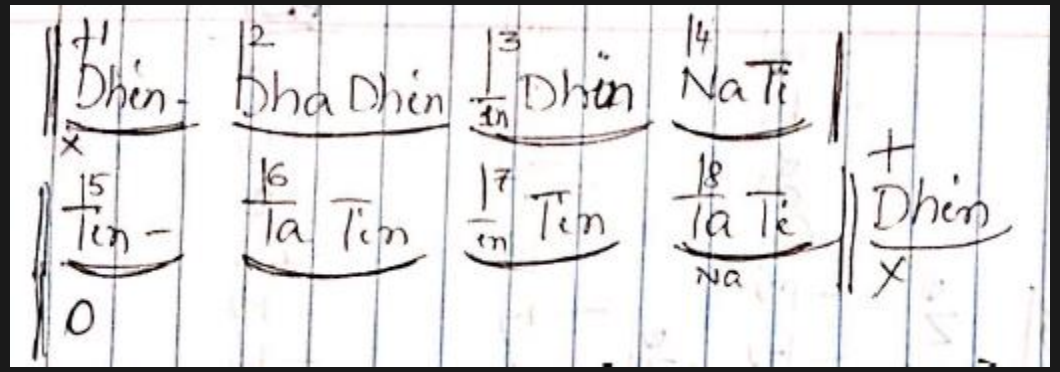
Notes

```
"pad": {
    "Ghe": "left",
    "Ga": "left",
    "Ka_Ke": "left",
    "Na": "right",
    "Tun_Di": "right",
    "Ta_Te": "right",
    "Te_Re": "right",
    "Ne": "right",
    "Tin": "both",
    "Dhin": "both",
    "Dha": "both",
    "Thun": "both"
},
"compoundNotes": {
    "Tin": { "left": "Ga", "right": "Tun_Di" },
    "Dhin": { "left": "Ghe", "right": "Tun_Di" },
    "Dha": { "left": "Ghe", "right": "Na" },
    "Thun": { "left": "Ka_Ke", "right": "Tun_Di" }
},
"noteFiles": {
    "Ga": "assets/Ga_16.wav",
    "Ghe": "assets/Ghe_16.wav",
    "Na": "assets/Na_16.wav",
    "Tun_Di": "assets/Tun_Di_16.wav",
    "Ka_Ke": "assets/Ka_Ke_16.wav",
    "Ta_Te": "assets/Ta_Te_16.wav",
    "Te_Re": "assets/Te_Re_16.wav",
    "Ne": "assets/Ne_16.wav"
},
"songs": [
    [ "GheNa", "KaTa", "KeNa", "KaTa" ],
    [ "GheNa", "GheNa", "GheNa", "KaTa", "KeNa", "GheNa", "GheNa", "KaTa" ],
    [ "GheNa", "_GheNa", "GheNa", "KaTa", "KeNa", "_KeNa", "GheNa", "KaTa" ],
    [ "GheNa", "_GheNa", "_GheNa", "KaTa", "KeNa", "_KeNa", "_GheNa", "KaTa" ],
    [ "DhinTa", "TinTa" ]
]
```

Patterns


back
NaNaTeTe
DhaTi
NaGhe
DhaTe
DhaDhaTeTe
KaTa
TeReKeTe
DhaDhinNa
DhaDhinNaNa
_KeNa
GaGhe
KeNa
DhinTa
TinTa
NaTe

Patterns


1 Dhin - x
2 Dha Dhin
3 Dhin
4 Na Ti
5 Tin - 0
6 Ta Tin
7 Tin
8 Ta Ti Na
+ Dhin x

Patterns

```
{
    "patterns": {
        "GheNa": { "beats": "8/8", "notes": [ "Ghe", "Na", "Ga", "Ghe", "Na", "Ga", "Ghe", "Na" ] },
        "_GheNa": { "beats": "8/8", "notes": [ 0, 0, 0, "Ghe", "Na", "Ga", "Ghe", "Na" ] },
        "KaTa": { "beats": "8/8", "notes": [ "Ka_Ke", "Ta_Te", "Ga", "Ghe", "Na", "Ga", "Ghe", "Na" ] },
        "TeReKeTe": { "beats": "4/16", "notes": [ "Ta_Te", "Te_Re", "Ka_Ke", "Ta_Te" ] },
        "KeNa": { "beats": "8/8", "notes": [ "Ka_Ke", "Na", "Ka_Ke", "Ka_Ke", "Na", "Ka_Ke", "Ka_Ke", "Na" ] },
        "_KeNa": { "beats": "8/8", "notes": [ 0, 0, 0, "Ka_Ke", "Na", "Ka_Ke", "Ka_Ke", "Na" ] },
        "DhinTa": { "beats": "8/8", "notes": [ "Dhin", 0, "Na", "Dhin", 0, "Dhin", "Na", "Te_Re" ] },
        "TinTa": { "beats": "8/8", "notes": [ "Tin", 0, "Na", "Tin", 0, "Tin", "Na", "Te_Re" ] },
        "GaGhe": {
            "beats": "4/4",
            "notes": [
                "Ga",
```

Songs

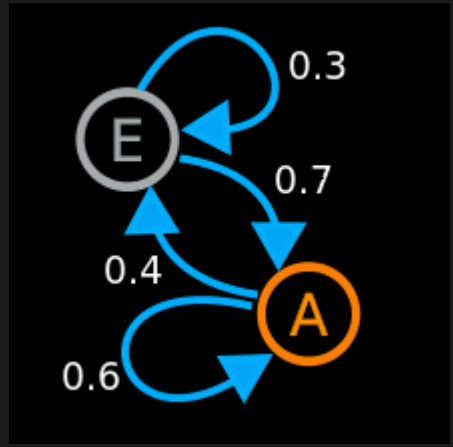
```
"pad": {
    "Ghe": "left",
    "Ga": "left",
    "Ka_Ke": "left",
    "Na": "right",
    "Tun_Di": "right",
    "Ta_Te": "right",
    "Te_Re": "right",
    "Ne": "right",
    "Tin": "both",
    "Dhin": "both",
    "Dha": "both",
    "Thun": "both"
},
"compoundNotes": {
    "Tin": { "left": "Ga", "right": "Tun_Di" },
    "Dhin": { "left": "Ghe", "right": "Tun_Di" },
    "Dha": { "left": "Ghe", "right": "Na" },
    "Thun": { "left": "Ka_Ke", "right": "Tun_Di" }
},
"noteFiles": {
    "Ga": "assets/Ga_16.wav",
    "Ghe": "assets/Ghe_16.wav",
    "Na": "assets/Na_16.wav",
    "Tun_Di": "assets/Tun_Di_16.wav",
    "Ka_Ke": "assets/Ka_Ke_16.wav",
    "Ta_Te": "assets/Ta_Te_16.wav",
    "Te_Re": "assets/Te_Re_16.wav",
    "Ne": "assets/Ne_16.wav"
},
"songs": [
    [ "GheNa", "KaTa", "KeNa", "KaTa" ],
    [ "GheNa", "GheNa", "GheNa", "KaTa", "KeNa", "GheNa", "GheNa", "KaTa" ],
    [ "GheNa", "_GheNa", "GheNa", "KaTa", "KeNa", "_KeNa", "GheNa", "KaTa" ],
    [ "GheNa", "_GheNa", "_GheNa", "KaTa", "KeNa", "_KeNa", "_GheNa", "KaTa" ],
    [ "DhinTa", "TinTa" ]
]
```

Samples and music

- First iteration: self-recorded single-notes
- Second iteration: single-note samples from https://archive.org/details/MihirSarkar CC3.0

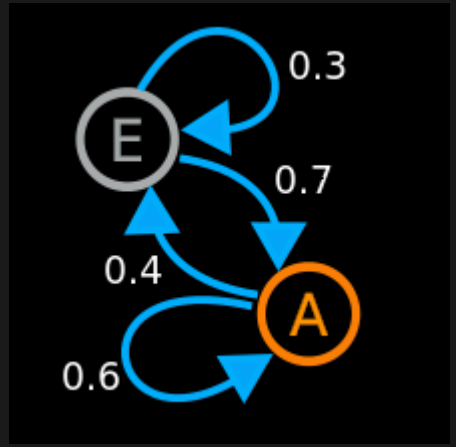
Markov Chains

- What is a markov chain?



Markov Chains

- What is a markov chain?
- Sequence of events / state transitions
- Where each state transition/event is independent of history, only depends on the last state
- Side effect: short-term memory



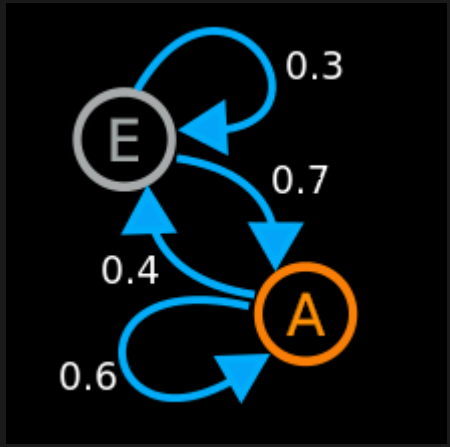
Markov States

Left Pad	Right Pad	Both
Ghe	Na	Dha
Ga	Tun_Di	Dhin
Ka_Ke	Ta_Te	Thun
	Te_Re	
	Ne	
	Tin	

Markov States

Markov Chains

- Transition probability distribution, calculated from JSON of patterns (actually done with python! #PyData)



Markov Chains


```
{
    "patterns": {
        "GheNa": { "beats": "8/8", "notes": [ "Ghe", "Na", "Ga", "Ghe", "Na", "Ga", "Ghe", "Na" ] },
        "_GheNa": { "beats": "8/8", "notes": [ 0, 0, 0, "Ghe", "Na", "Ga", "Ghe", "Na" ] },
        "KaTa": { "beats": "8/8", "notes": [ "Ka_Ke", "Ta_Te", "Ga", "Ghe", "Na", "Ga", "Ghe", "Na" ] },
        "TeReKeTe": { "beats": "4/16", "notes": [ "Ta_Te", "Te_Re", "Ka_Ke", "Ta_Te" ] },
        "KeNa": { "beats": "8/8", "notes": [ "Ka_Ke", "Na", "Ka_Ke", "Ka_Ke", "Na", "Ka_Ke", "Ka_Ke", "Na" ] },
        "_KeNa": { "beats": "8/8", "notes": [ 0, 0, 0, "Ka_Ke", "Na", "Ka_Ke", "Ka_Ke", "Na" ] },
        "DhinTa": { "beats": "8/8", "notes": [ "Dhin", 0, "Na", "Dhin", 0, "Dhin", "Na", "Te_Re" ] },
        "TinTa": { "beats": "8/8", "notes": [ "Tin", 0, "Na", "Tin", 0, "Tin", "Na", "Te_Re" ] },
        "GaGhe": {
            "beats": "4/4",
            "notes": [
                "Ga",
```

Markov Chains

- Transition probability distribution, calculated from JSON of patterns (actually done with python! #PyData)


```
{
    "Ghe": {
        "Ghe": 0.021392190152801367,
        "Ga": 0.021392190152801367,
        "Ka_Ke": 0.021392190152801367,
        "Na": 0.1045840407470289,
        "Tun_Di": 0.021392190152801367,
        "Ta_Te": 0.18947368421052638,
        "Te_Re": 0.021392190152801367,
        "Ne": 0.4492359932088287,
        "Tin": 0.021392190152801367,
        "Dhin": 0.021392190152801367,
        "Dha": 0.021392190152801367,
        "Thun": 0.021392190152801367,
        "0": 0.06417657045840411
    },
```

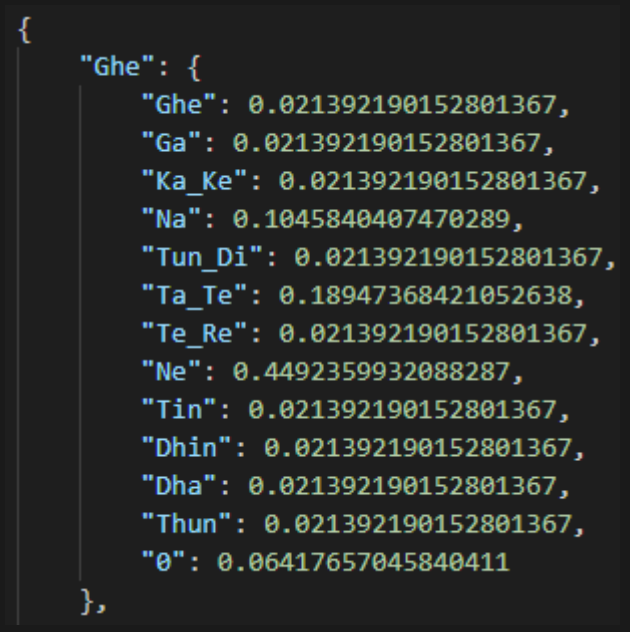
Markov Chains

- Transition probability distribution, calculated from JSON of patterns (actually done with python! #PyData)
- P(Note A => Note B) = P(from JSON) + P(injected randomness)
  - Additional chance of randomly selecting any note, and greater chance of being a stop note


```
{
    "Ghe": {
        "Ghe": 0.021392190152801367,
        "Ga": 0.021392190152801367,
        "Ka_Ke": 0.021392190152801367,
        "Na": 0.1045840407470289,
        "Tun_Di": 0.021392190152801367,
        "Ta_Te": 0.18947368421052638,
        "Te_Re": 0.021392190152801367,
        "Ne": 0.4492359932088287,
        "Tin": 0.021392190152801367,
        "Dhin": 0.021392190152801367,
        "Dha": 0.021392190152801367,
        "Thun": 0.021392190152801367,
        "0": 0.06417657045840411
    },
```

Markov Chains

- Transition probability distribution, calculated from JSON of patterns (actually done with python! #PyData)
- P(Note A => Note B) = P(from JSON) + P(injected randomness)
  - Additional chance of randomly selecting any note, and greater chance of being a stop note
- Simulate the Markov Chain!



How does it hook to p5.js?

- p5.loadSound()
- p5.loadJSON()
- draw()

p5.js code-architecture

- preload()
- setup()
- draw()

p5.js code-architecture

- preload()
  - Load all files that are needed for the application to run, like one big Promise
  - the tabla samples are loaded here

p5.js code-architecture

- setup()
  - Called once before draw() to define all variables and settings

p5.js code-architecture

- draw()
  - This code is executed every frame, which is controlled by frameRate()
  - “draw” from the distribution every frame

p5.js code-architecture

- listeners: mousePressed, mouseDragged, …
- p5.Sound:
  - p5.Phrase
  - p5.Part

p5.js code-architecture

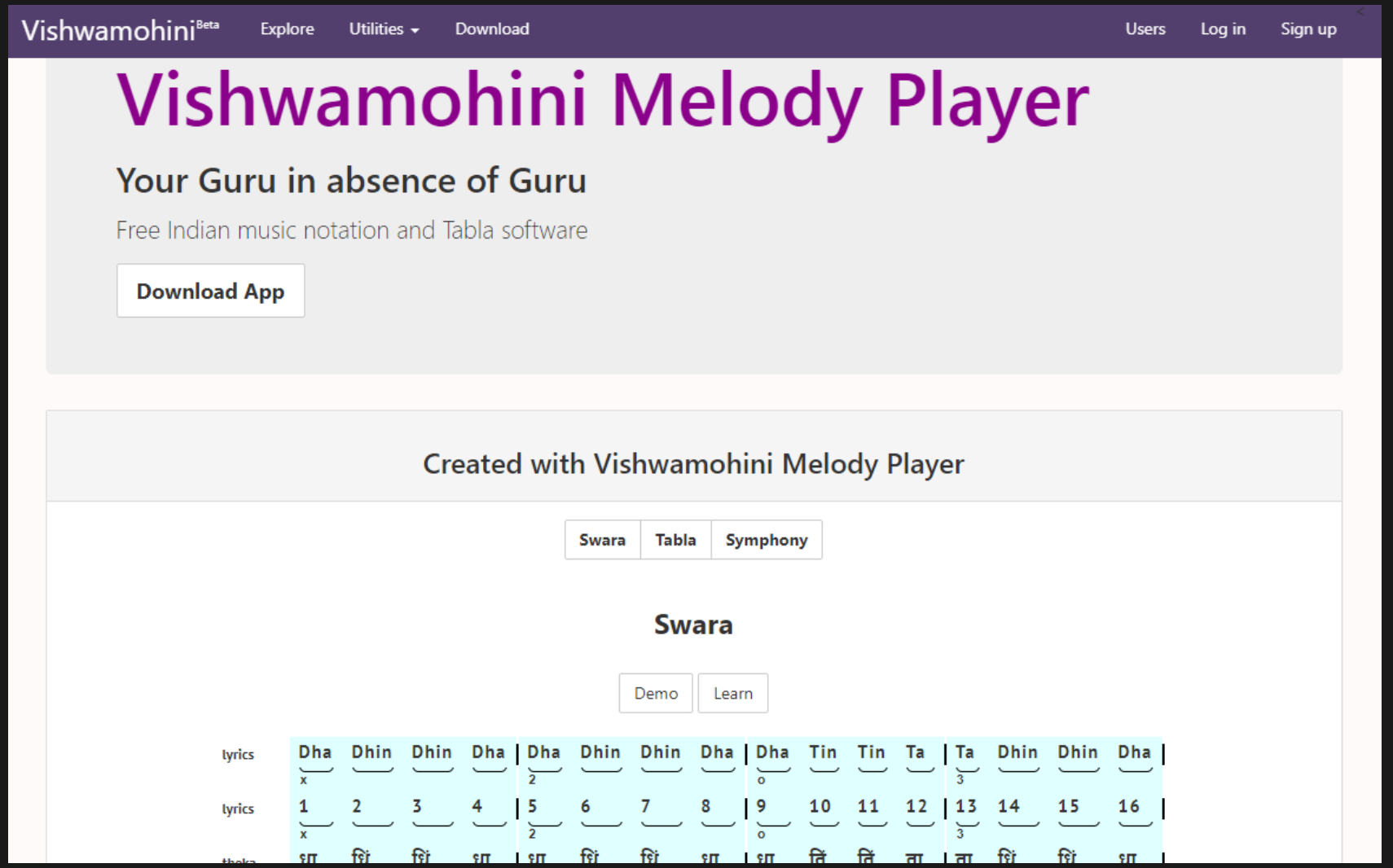
- Queueing up samples into a phrase for playback
  - ```
    function playList(toPlay) {
      try {
            myPart.stop();
      }
      catch {}
      myPhrase = new p5.Phrase('tabla', playNote, toPlay);
      myPart = new p5.Part();
      myPart.setBPM(defBPM);
      myPart.addPhrase(myPhrase);
      myPart.start();

      return myPart
    }
    ```
    ```

p5.js code-architecture

- Callback as follows:
  - ```
    function playNote(time, note){
      // logic to play compound notes
      if (typeof note == 'object’) {…}
      else {
            notes[note].play(time);
      }
      // logic to stop each pad from ringing an old sound
      if (leftNote & pad[note]=='left’) {…}
      if (rightNote & pad[note]=='right’) {…}
    }
    ```
    ```

Related Work:
http://vishwamohini.com/music/home.php



TODO: sequence generation

INPUT: JSON of tabla note sequences

⇒generative model: Markov Chains, **RNNs, LSTMs, etc.**

⇒OUTPUT: generative tabla note sequences

RNNs and other sequence models do away with the Markovian assumption, and will ideally maintain some longer-term structure

TODO: sequence generation

Audio-based generation models:

- direct audio => audio completion
  - Needs training dataset of lots of recording
- audio => note transcription => note completion => audio conversion
  - Needs training dataset of recording labeled by notes or audio => notes classifier. also metadata on time-signatures

TODO: sequence generation

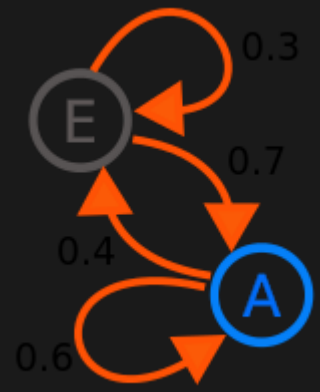
Latent space of tabla sequences

- UI to explore the latent space a la
  - https://glitch.com/edit/#!/incredible-spinners
  - https://teampieshop.github.io/latent-loops/

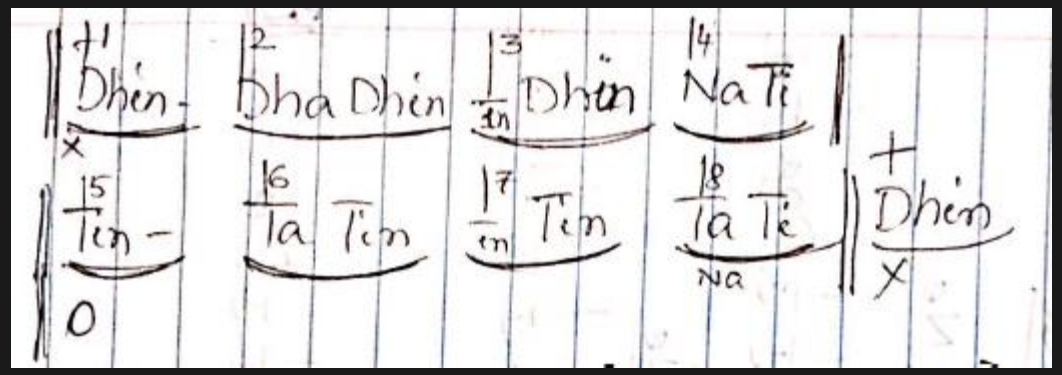
TODO: features

- More note-sequences in JSON
- Saving patterns to disk
- Writing out custom patterns to play
- Incorporating different time-signatures into generation

Recap

p5.js + 

+



= FREE ENDLESS TABLA ??

Thank You

Contact @
Email: kjoshfree@gmail.com
GitHub: kmjoshi
LinkedIn: keshavmjoshi
Twitter: keshavahsek