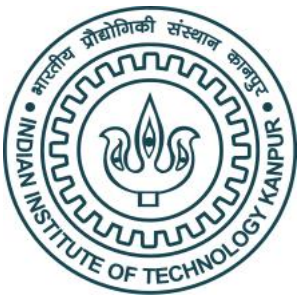

Indian Institute of Technology Kanpur

Metronome

Mentor:

Prof. Vipul Arora,
Department of Electrical Engineering
IIT Kanpur

Project Members :

Gyanendra Kumar
Nitin Gupta
Abhishek Verma

Contents

1 Introduction **2**

2 Basic Terminologies of Metronome **2**
2.1 Tempo 2
2.2 BPM 2
2.3 Beat Cycles 2
2.4 Strokes 2
2.5 BPM velocity 2
2.6 Transition 2

3 Working of Metronome **3**
3.1 Technologies used 3
3.2 General Working 3
3.3 Favourites Playlist 4
3.4 Backend Processes for compiling and playing beat cycle 5

4 References **9**

1 Introduction

A metronome is a device that plays sounds at regular intervals which can be set by the user in BPM (beats per minute).

Musicians practise with metronomes to improve their timing, especially the ability to stick to a regular tempo. Metronome practice helps internalize a clear sense of timing and tempo. Composers and conductors often use a metronome as a standard tempo reference and may play, sing, or conduct to the metronome. The metronome is used by composers to derive beats per minute if they want to indicate that in a composition. Conductors use a metronome to note their preferred tempo in each section.

2 Basic Terminologies of Metronome

2.1 Tempo

Tempo is the speed at which a piece of music is played. It is measured in BPM.

2.2 BPM

This method involves assigning a numerical value to a tempo. "Beats per minute" (or BPM) is self-explanatory: it indicates the number of beats in one minute. For instance, a tempo notated as 60 BPM would mean that a beat sounds exactly once per second. A 120 BPM tempo would be twice as fast, with two beats per second.

2.3 Beat Cycles

It is the musical structure that repeats itself in the song. For example – ['tin', 'tin', 'dha', '-'], this is one beat cycle.

2.4 Strokes

The notes played in the beat cycle are called strokes. Eg.- 'tin'. Every stroke has an anchor point marked by the tap of the musician.

2.5 BPM velocity

Sometimes we need a gradual increase in the tempo. The user can set BPM velocity as per need to increase/decrease tempo after every beat cycle automatically.

2.6 Transition

A transition is a passage of music composed to link one section of music to another.

3 Working of Metronome

3.1 Technologies used

- HTML, CSS, Bootstrap and Javascript

3.2 General Working

In order to play a particular configuration, User needs to select style (beat cycle) and set the BPM and BPM velocity and then on clicking 'Start' button the following configuration will set playing. The configuration (like BPM , BPM velocity, Style) may be changes at any moment of time. But any changes will reflect once the current playing beat cycle played up to last beat. This is done to ensure that there will be smooth change between the configuration.

If the use want to play any beat cycle from the playlist they have created, they will do so by just clicking the 'Play' button. Again, the clicked beat cycle will played once the current playing beat cycle played up to last beat.

Currently Present Beat cycles

We have used 4 beat cycles namely:

1) **Kartaal** is an ancient instrument mainly used in devotional / folk songs. It has derived its name from Sanskrit words 'kara' meaning hand and 'tala' meaning clapping.

Kartaal(3) - [[tin] [dha] [-]]
Kartaal(4) - [[tin] [tin] [dha] [-]]

2) The **mridangam** is a percussion instrument of ancient origin. It is the primary rhythmic accompaniment in a Carnatic music ensemble.

Mridanga(16) - [[khi] [- khi] [na] [dha] [ghe_soft] [dha] [ghe_soft] [dha] [ghe] [-] [-] [na] [te] [te re] [te re] [na]]
Mridanga(8) - [[ghe] [-] [na] [pa] [-] [ghe] [dha] [-]]

The transition cycle for Mridanga(16) is:

[[te re] [khe ta][dha] [te re] [khe ta] [dha] [te re] [khe ta] [dha] [-] [-] [na] [te] [te re] [te re] [na]]
(The first 11 beats have been changed rest is same as Mridanga(16))

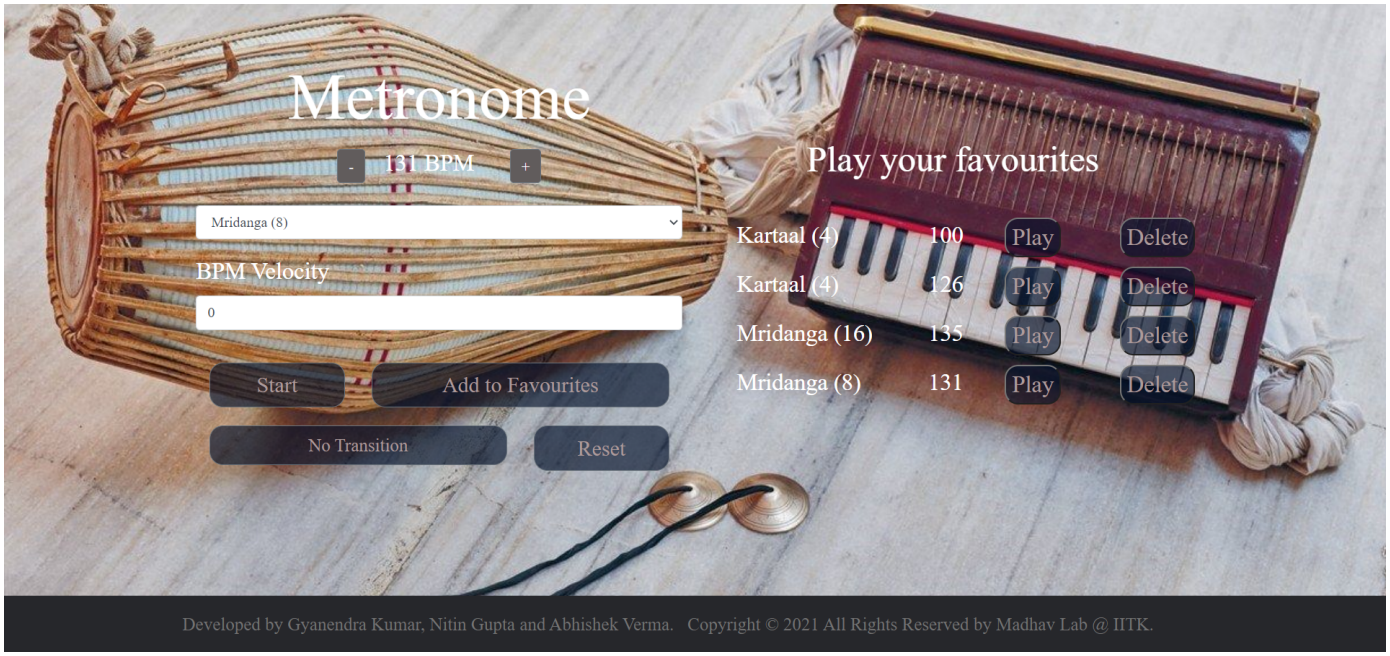

Figure 1: User Interface of Online Metronome

3.3 Favourites Playlist

There is a section in a online metronome where the users can create there own personalized playlist. This is helpful for the musicians as in many instances they need to switch between two beat cycles or BPM instantly. So, Creating playlist would be helpful during live performance i.e., they create there playlist before hand and they can play any saved beat cycles with desired configurations at one click. This ensures that the flow of the musics would not get interrupted.

Steps to Add Particular Beat cycles and Configuration in Playlist:

- Choose your style (beat cycle) from a dropdown.
- Enter BPM Velocity and adjust BPM according to the needs.
- Click on Add to Favourites button.
- If you want to add transition to playlist then you need to play a song and configure a transition then click on Add to Favourites Button.
- After clicking the Add to Favourites button, the selected beat cycle and the configuration will be saved and shown in 'Play your Favourites'. Now you can play this configuration by clicking play on this button.
- If you want to remove particular configuration from playlist then you simply need to click 'Delete' button of particular configuration.

3.4 Backend Processes for compiling and playing beat cycle

Loading of Audio beats

With the loading of webapp the audio files will get loaded so that there would be no lag due to poor internet connectivity. In Javascript, audio file can easily be loaded through passing the source url of beat in new Audio function as shown below:

```
var beat_name = new Audio('beat_source_url');
```

Creation of Beat Cycle and speed rate of each Beat

Two Arrays would be made for each beat cycle in which first array show the sequence of beats in which they need to play and in second array there is an speed array. This array would be helpful in the case when we need to play two beats in one interval.

Sample code for Beatcycle = [[A][B C][D]] :

```
var beatCycle1 = [[A],[B],[C],[D]] // beat array
var beatCycle1t = [1,2,2,1]; // speed array
```

In this example, we can see in the 2nd beat interval of beatcycle1 two beats need to be played so it simply means that beats should plays twice faster than the other beats.

If this beatcycle contains any type of transition then we need to create another 2 arrays named tran1 and tran1a. tran1 stores the sequence of beat that need to play when transition is enabled and tran1a keeps a track of the speed of beat at which they are playing.

So, For ex. lets say we have transition for above beat cycle = [[B][A C][A D]] then,

```
var tran1 = [[B],[A],[C],[A],[D]]
var tran1a = [1,2,2,2,2];
```

If we don't have tansition for particular beatcycle then,

```
var tran1 = beatcycle1
var tran1a = beatcycle1t;
```

We need to create such array for each beat cycle we want to add in our metronome.

After this, we have created a master beatcycle array and beatcycle speed array in which all the current beatcycle were pushed. This is done to access each beatcycle with O(1) time complexity.

```
var beatCycle = [];//In this array we push all beatcycle beat array
var beatCyclet = [];//In this array we push all beatcycle speed array
```

Similarly for transition, these two arrays can be make easily.

Assigning Onset for each beats

The next step is to make the onset array of each beat. Through this array we can easily get an onset time for each beats in O(1) time complexity.

Playing Beats for each configuration

At this stage we have created a database for our webapp. Once user click on 'Start' button the playing function is executed.

The main logic of playing function is that we are creating a temporary playlist array that need to be played. If the playlist size is 0 then it will add the beatcycle and there speed on the configuration set by user at that time. If the playlist size is not equal to 0 then it will play the first beat of playlist and then this first beat is removed from the playlist. This was set on loop such that playing function will be called after specific interval.

Playing Function

```
function playing(){
    if(playlist.length==0){
        addcycle(); // add the new beatcycle based on user configuration
    }
    beat=playlist[0]; //accessing first beat of playlist
    k=playlistt[0]; //accessing speed rate of first beat of playlist
    playlist.shift(); //deleting first beat of playlist
    playlistt.shift();//deleting speed rate of first beat of playlist
    bpmcurr=bpm*k; // determining the speed of beat
    temp+=1/k;
    clearInterval(t);
    audio[i] = new Audio();
    audio[i].volume = 1.0;
    audio[i].loop = false;
    audio[i]=beat[0];
    if(beat[0]==blnk || beat[0]==blnk0)audio[i].volume=0.0;
    console.log(k);
    audio[i].play(); //Playing audio
    i++;
    loop();//playing function will be called after specific interval.
}
```

```
/* This function will excetue the playing function after specific set of
time such that the difference between the onset time of two beat is equal to
bpm/60*/
    function loop(){
        if(playlist.length>0) t =
        setInterval(playing,60.0*1000/(bpmcurr)+1000*dict[beat[0]]-
                    1000*dict[playlist[0][0]]);
        else t = setInterval(playing,60.0*1000/(bpmcurr)-1000*dict[beat[0]]
                             +1000*dict[beatCycle[x][0][0]]);
    }
```

Calculation of Tap Cycle and Incorporation in Playlist

For some of the styles in which we have a transition , the transition button are enabled.
For Setting a particular tap cycle :

- The button will be marked as 'Mark Transition 1'. Once user click on this button,the button changes to Mark Transition 2' and the tap cycle value increases by 1 after each beatcycle will be played.
- When user clicks on the 'Mark Transition 2', the tapcycle value at that moment of time will be kept in memory and after playing tapcycle value basic playlist the transition playlist will be played.
- If user click on 'Reset' button then it simply nullifies the effect of transition.

The below function will show how we are adding beat cycles after each beats of playlist gets played.

```
function addcycle(){
    if(bpm+bpm_velocity<10){
        bpm =10;  // min_value of bpm is set to be 10
    }
    else if(bpm+bpm_velocity>300){
        bpm =300; // max_value of bpm is set to be 300
    }
    else{
        bpm = bpm+bpm_velocity; // adding bpm velocity after each beat cycle
    }
    // if beatcyles played to be next is selected from favourites.
    // The whole code is about retriveing the details of selected favourites
    if(flag1==1){
        bpm = temp_bpm;
        bpm_velocity = temp_bpm_velocity;
        if(x==2&&temp_tapbut==2){
```

```
            tapcyc =temp_tapcyc;
            document.getElementById("tap").value =tapcyc;
            tapbut =2;
            tp =0;
        }
        else{
            tapbut=0;
            if(x==2)document.getElementById("tap").value ="Mark Transition 1"
            else document.getElementById("tap").value  ="No Transition"
        }
        flag1=0;
    }
    var i =0;
    // If transition button is tapped once then after each beatcycle the
//tapcycle value increase by 1.
    if(tapbut==1){
        tapcyc++;
        document.getElementById("tap").value ="Mark Transition 2 at beatcycle"
    }
    //If transtion button is tapped twice then tapcyc gets stored in local
    //cache.
    if(tapbut==2){
        tp++;
        tp=tp%tapcyc;

        if(tapcyc<=1){
            //adding transtion beats to playlist
            playlist = playlist.concat(trana[x]);
            //adding tansition beats speed rate to playlistt
            playlistt= playlistt.concat(tranat[x]);
            return;
        }
        if(tp==0){
            playlist = playlist.concat(trana[x]);
            playlistt= playlistt.concat(tranat[x]);
            return;
        }
    }
    //If there is no transtion then the normal playlist will concat and played
    playlist = playlist.concat(beatCycle[x]);
    playlistt=playlistt.concat(beatCyclet[x]);
}
```

4 References

- *Online Metronome Link*
- *Source Code*