The Harmonization Project

By Selim Karaoglu and Kalani Sanidad

What our project is about

- This project is about music. More specifically, harmonization.
- In music, harmonization is the chordal accompaniment to a line or melody: "Using chords and melodies together, making harmony by stacking scale tones as triads"[1]

This project is designed to use the evolutionary programming to create a random melody and achieve a harmonization with user selected chord progression.

[1]: Schonbrun, M. The Everything Music Theory Book: A Complete Guide to Taking Your Understanding of Music to the Next Level, p.257. ISBN 1-59337-652-9, 2006

Audience

This web app is intended for

- Musicians
- Artists
- Anyone who’s into music.

Creating music one step at a time!

This project has **two** main elements; **melody** and the **chord progression.**

To set the chord progression, we need three different musical element

1. Scale
2. Root Note
3. Chord Progression

These values are presented the user with a <select> item, therefore there are no wrong choices… Just user taste.

How we designed it

The step-by-step approach has some designed features;

- Each step has it’s own help tooltip to clarify the process and inform users,
- After the selection, the next step appears automatically,
- If user wants to go back and change anything, clicking on the step name is enough, simple. But since this is about harmony between melody and chord progression, any change will result with created melody to be erased. Which makes sense in musical terms, since if a melody is designed for A minor scale won’t be fit for Db major scale.

Evolutionary Approach

This part focuses on random melody creation and harmonization of the melody using the selected variables.

The background information necessary to understand the musical math is too much to present here. Instead, we will present only how evolutionary process works.

This evolutionary process takes a list of random numbers (that forms the melody), focuses to make them better with three different evolutionary processes.

Mutations and Crossover

Our approach uses two different mutations and a crossover;

First mutation focuses on changing the **last note**. Takes a random good gene (green colored notes) and replaces it with the last note.

```
//Last Note mutation. This function runs a mutation that changes the last note of the individual.
function mutationLastNote(fit, rootn)
{
  var tmp = "";

  if (rootn === 'A3') {numeric = 57;}
  else if (rootn === 'Bb3') {numeric = 58;}
  else if (rootn === 'B3') {numeric = 59;}
  else if (rootn === 'C3') {numeric = 48;}
  else if (rootn === 'Db3') {numeric = 49;}
  else if (rootn === 'D3') {numeric = 50;}
  else if (rootn === 'Eb3') {numeric = 51;}
  else if (rootn === 'E3') {numeric = 52;}
  else if (rootn === 'F3') {numeric = 53;}
  else if (rootn === 'Gb3') {numeric = 54;}
  else if (rootn === 'G3') {numeric = 55;}
  else {numeric = 56;}
  var a = fit[fit.length - 2];
  if (a != numeric || a != (numeric + 12) || a != (numeric + 24) || a != (numeric + 36)) {
    for (var i = 0; i < fit.length - 1; i++) {
      if (fit[i] == numeric || fit[i] == (numeric + 12) || fit[i] == (numeric + 24) || fit[i] == (numeric + 36))
      {
        tmp = fit[i];
        fit[i] = a;
        a = tmp;
        tmp = 0;
        var fitnessTable = [];
        for (var i = 0; i < fit.length -1; i++) {
          fitnessTable[i] = noteFitness(fit[i]);
        }
        fit.pop();
        fit.push(averageFitness(fitnessTable));
        break;
      }
    }
  }
  else if (a != (numeric + 7)  || a != (numeric + 19) || a != (numeric + 31)) {
    for (var i = 0; i < fit.length - 1; i++) {
      if (fit[i] == (numeric + 7) || fit[i] == (numeric + 19) || fit[i] == (numeric + 31)) {
        tmp = fit[i];
        fit[i] = a;
        a = tmp;
        tmp = 0;
        var fitnessTable = [];
        for (var i = 0; i < fit.length -1; i++) {
          fitnessTable[i] = noteFitness(fit[i]);
        }
        fit.pop();
        fit.push(averageFitness(fitnessTable));
        break;
      }
    }
  }
  else if (a != (numeric + 5)  || a != (numeric + 17) || a != (numeric + 29)) {
    for (var i = 0; i < fit.length - 1; i++) {
      if (fit[i] == (numeric + 5) || fit[i] == (numeric + 17) || fit[i] == (numeric + 29)) {
        tmp = fit[i];
        fit[i] = a;
        a = tmp;
        tmp = 0;
        var fitnessTable = [];
        for (var i = 0; i < fit.length -1; i++) {
          fitnessTable[i] = noteFitness(fit[i]);
        }
        fit.pop();
        fit.push(averageFitness(fitnessTable));
        break;
      }
    }
  }
  return fit;
}
```

Mutations and Crossover

Second mutation is for gene recreation. Takes a bad gene (red or blue) and creates another **random gene**. Just like life, there's no guarantee that mutation will get the individual **better**, but it's **not** going to make it **worse**.

Crossover takes the fittest and second fittest individual and makes a crossover at the calculated point (for crossover to be beneficial).

```
//Crossover Function. First part of the evolution process.
function crossover()
{
  var fit = getFittest();
  var sfit = getsecondFittest();
  var tmp = 0;
  for (var i = 0; i < fit.length; i++) {
    if (noteFitness(fit[i]) > noteFitness(sfit[i]))
    {
      tmp = fit[i];
      fit[i] = sfit[i];
      sfit[i] = tmp;
      tmp = 0;
      break;
    }
  }
  var fitnessTable = [];
  for (var i = 0; i < fit.length -1; i++) {
    fitnessTable[i] = noteFitness(fit[i]);
  }
  fit.pop();
  fit.push(averageFitness(fitnessTable));
  return (fit);
}
      ,
    }
  }
  return fit;
}

//Gene Flip Mutation function. This takes a bad gene and creates a new random gene to replace it.
function mutationGeneFlip(fit)
{
  for (var z = 0; z < fit.length - 1; z++) {
    if (noteFitness(fit[z]) == 5) {
      fit[z] = String(Math.floor((Math.random() * 25) + 60));
      var fitnessTable = [];
      for (var j = 0; j < fit.length -1; j++) {
        fitnessTable[j] = noteFitness(fit[j]);
      }
      fit.pop();
      fit.push(averageFitness(fitnessTable));
      break;
    }
  }
  return fit;
}
```

Fitness Function

Fitness function is used to determine the melody's fitness with the user selected chord progression. Takes every note independently, calculates the fitness of the note and assigns fitness values to an array. Average fitness function calculates the average fitness of the notes in array for each individual.

```
//This function calculates the fitness of a single note comparing it to the selected note's selected scale.
function noteFitness(note)
{
  var fitnessValue;
  var rootNote = document.getElementById("root").options[document.getElementById("root").selectedIndex].value;
  var scale = parseInt(document.getElementById("scale").options[document.getElementById("scale").selectedIndex].value);
  if (scale === 909) {
    chordScaleMap = [909,878,878,909,909,878,878];
    chordDistanceMap = [0, 0, 2, 4, 5, 7, 9, 11, 12];
  } else if (scale === 878) {
    chordScaleMap = [878,878,909,878,878,909,909];
    chordDistanceMap = [0, 0, 2, 3, 5, 7, 8, 10, 12];
  }
  var numeric;
  if (rootNote === 'A3') {numeric = 57;}
  else if (rootNote === 'Bb3') {numeric = 58;}
  else if (rootNote === 'B3') {numeric = 59;}
  else if (rootNote === 'C3') {numeric = 48;}
  else if (rootNote === 'Db3') {numeric = 49;}
  else if (rootNote === 'D3') {numeric = 50;}
  else if (rootNote === 'Eb3') {numeric = 51;}
  else if (rootNote === 'E3') {numeric = 52;}
  else if (rootNote === 'F3') {numeric = 53;}
  else if (rootNote === 'Gb3') {numeric = 54;}
  else if (rootNote === 'G3') {numeric = 55;}
  else {numeric = 56;}
  var noteDistance = note - numeric;

  //perfect consonants; unison, perf. IV, perf V, octave
  if (noteDistance == 12 || noteDistance == 24 || noteDistance == 36 || noteDistance == 5 || noteDistance == 17 || noteD
  { fitnessValue = 1; }
  //II and VII
  else if (noteDistance == 2 || noteDistance == 14 || noteDistance == 26 || noteDistance == 10 || noteDistance == 22 ||
  { fitnessValue = 3; }
  //84 is Rest
  else if (note == 84)
  { fitnessValue = 1; }
  //imperfect consonants; min and maj III and VI
  else if (scale == 909 && (noteDistance == 4 || noteDistance == 16 || noteDistance == 28 || noteDistance == 9 || noteDi
  { fitnessValue = 2; }
  else if (scale == 878 && (noteDistance == 3 || noteDistance == 15 || noteDistance == 27 || noteDistance == 8 || noteDi
  { fitnessValue = 2; }
  else { fitnessValue = 5; }

  return fitnessValue;
}

//This function takes an array of fitness values and calculates the average.
function averageFitness(fitnessTable)
{
  var avg = 0;
  for (var i = 0; i < fitnessTable.length; i++) {
    avg = avg + fitnessTable[i];
  }
  avg = avg / fitnessTable.length;
  return avg;
}
```

The Demo

Harmonization Project

Step 1

Please Select a scale

Scale

Project Source Structure

i. main
   1. Soundfont (Midi note audio files)
      a. acoustic_grand_piano-mp3 (89 sound files)
      b. acoustic_grand_piano-mp3.js
      c. acoustic_grand_piano-ogg.js
   2. banner.png
   3. harmonizing.html (Main HTML file)
   4. main.css

ii. js
   1. midi (javaScript files for midi)
   2. util (javaScript utility files)
   3. evo.js
   4. main.js
   5. musicsheet.js
   6. test.js

iii. inc
   1. shim (midi.js package)

Outer Sources

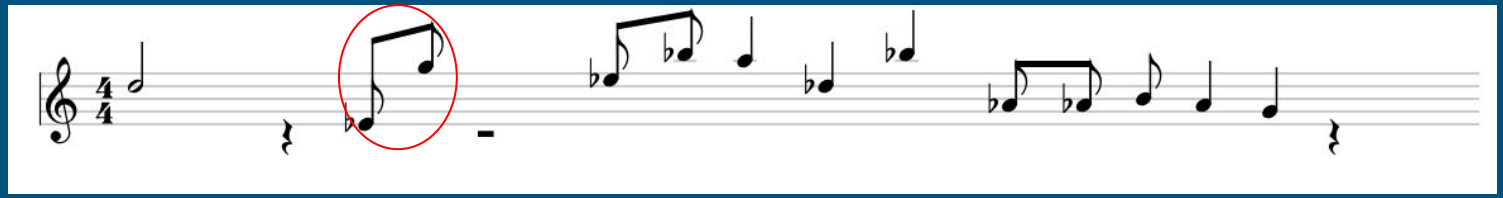
i. Midi.js
ii. VexFlow
iii. Bootstrap
iv. Popper.js

Also 000webhostapp.com is used to publish the project on the web. Since this is a free hosting service, there are limitations for free features.

Future Features

The stave used in the project can be improved.

If you are to look at the stave, making those notes look more presentable is possible;



These notes still have tails on them which are not supposed to have in other musical composure

Future Features

The harmonization process works on a short melody (4 whole notes on a 4-4 bar settings), in the future the melody size can be increased.

There can be several other mutations, and current mutations can be modified.

Several chord progressions can and should be implemented in the future. Since we focused on delivering a fully functional project, we couldn’t focused on musical depth.

Just like chord progressions, number of scales can be increased for this project. Similar to current scales, some other scales like Blues, Pentatonic, Harmonic etc. scales can be added to the context of this work.