Musical modes

Musical modes provide a structure for generating nice sounding chords. There are seven modern musical modes:

- Ionian
- Dorian
- Phrygian
- Lydian
- Mixolydian
- Aeolian
- Locrian

The Ionian, Lydian, and Mixolydian modes are considered major modes and have a brighter and more cheery feel than the other listed modes, which are considered minor modes.

To read more about modes, see Musical Modes on Wikipedia.

Using modal chords

It is not actually necessary to understand modes to use them as a composition tool in songlib. By setting a particular mode, songlib allows you to generate modal chords compatible with that mode quite easily. The trichord generating function is named c, with the first argument specifying the chord number:

\[ c(1, \ldots), c(2, \ldots), c(3, \ldots), c(4, \ldots), c(5, \ldots), c(6, \ldots), \text{and } c(7, \ldots) \]

A trichord is simply a chord of three notes. If you are building a song in the key of C, then the \( c(1, \ldots) \) chord has a base (or root) note of C, the \( c(2, \ldots) \) chord has a root note of D, and so on through E, F, G, A, and B. If you are using the key of E, the \( c(1, \ldots) \) chord has a root note of E, the \( c(2, \ldots) \) chord has a root note of F#, and so on.

Thus, an Ionian riff in the key of D might look like the following in songlib:

```cpp
setKey(D);
setMode(IONIAN);
for (i = 0; i < 4; ++i) {
    c(1,H,guitar,octave);
}```
For simplicity, we will sometimes abbreviate the call \( c(1, \ldots) \) as \( c1 \), \( c(2, \ldots) \) as \( c2 \), and so on.

Notice the absence of F# in the above riff. This is because songlib figures out the root note and the other notes in the chord automatically when the \( c \) function is used.

## Chord progressions

When using the modal system, not only does songlib figure out which notes go with which chord, you also get, for free, a bullet-proof way to compose songs using chord progressions. A chord progression is simply two or more chords in a row. Some very smart people have figured out which chords sound ‘nice’ next to each other. Here are the general purpose charts for designing your own nice-sounding chord progressions.

For the **Ionian** or major mode, the chord progression generating chart looks like:

- c3
- c6
- c2 or c4
- c5 (then go to c1) or c7 (then go to c3)
- c1
- any chord

To generate a chord progression, you can start at any bullet. Say we start at c1. Then we go to the next bullet. It says any chord, so we can go to any bullet. Let’s go to the c2 or c4 bullet and choose c4. Again, we go to the next bullet and choose c5. Finally, we return to c1. It is common for major chord progressions to end at c1. Our little journey yielded the progression:

\[
c1 \rightarrow c4 \rightarrow c5 \rightarrow c1
\]

For the **harmonic minor** mode, the generating chart is slightly different:

- c7Major
- c3
- c6
- c2 (then go to c5 or c7) or c4 (then go to c7)
- c5 or c7
- c1
- any chord

Note that this chart introduces a new chord, c7Major, which stands for a Major c7. In the harmonic minor mode, c7 is a a diminished chord, so this chart features two versions of c7, one major and one diminished. The next section covers forcing chords in one mode to behave as if in another mode.
Playing chords out of mode

No matter the mode, some of the seven possible chords are major chords, some are minor, some are augmented, and some are diminished chords. Songlib figures this out automatically, so composing using the modal system is quite easy. However, the modal system described above is a little too confining. For example, in Ionian mode, c1 is a major chord. But perhaps you wish to throw in a minor version of c1. To do so, simply call the cm(1,...) function instead of c(1,...).

```cpp
setKey(D);
setMode(IONIAN);
for (i = 0; i < 4; ++i)
{
    cm(1,H,guitar,octave);
    c(4,H,guitar,octave-1);
    c(5,H,guitar,octave-1);
    c(5,H,guitar,octave-1);
    c(4,Hd,guitar,octave-1);
}
```

In Ionian mode, the three major chords are c1, c4, and c5, and the three minor chords are c2, c3, and c6. The last chord, c7, is a diminished chord. In Aeolian mode, the major chords are c3, c6, and c7, while c1, c4, and c5 are minor chords and c2 is a diminished chord. All chords have major, minor, augmented, and diminished versions, obtained by appending the appropriate letter to the chord name; the major, minor, augmented, and diminished versions are named, cM, cm, ca, and cd, respectively.

Thus for the progression generating chart for the harmonic minor mode, shown in the previous section, the c7major placeholder would be implemented by a call to cM(7,...). Since c7 is a diminished chord in the harmonic minor mode, it could be implemented as:

```cpp
chord(beats,instrument,octave,C,+3,+6,(int) 0);
```

A diminished chord lowers both the second note and the third note one semitone each. A call to the harmonic minor chord cM(7,...) in the key of C is equivalent to the following call to chord:

```cpp
chord(beats,instrument,octave,C,+4,+7,(int) 0);
```

Augmenting a chord generally means to raise the third note in the chord one semitone. For example, the Aeolian c7 chord is a major chord, so calling ca(7,...) in that mode in the key of C is equivalent to:

```cpp
chord(beats,instrument,octave,C,+4,+8,(int) 0);
```

Degrees

In music theory, the degree of a note indicates its index with respect to the root note in the scale. The modal system of songlib uses a diatonic scale having seven notes. In the key of C, the index of C is 1, D is 2, E is 3 and so on. In the key of D, the index of D is 1, E is 2, F# is 3 and so on.

The degree function returns the root note of the associated modal chord. For example, if you are in the key of C, then degree(1) returns a C, degree(2) returns a D, and so on.

Since c1 is a major chord in Ionian mode, it is therefore equivalent, under this scenario, to all of the following calls:
chord(H, guitar, octave, C, +4, +7, (int) 0);
chord(H, guitar, octave, degree(1), +4, +7, (int) 0);
maj(H, guitar, octave, C);
maj(H, guitar, octave, degree(1));

when in the key of C. Likewise, c2 is a minor chord and is thus equivalent to all these calls:

chord(H, guitar, octave, D, +3, +7, (int) 0);
chord(H, guitar, octave, degree(2), +3, +7, (int) 0);
min(H, guitar, octave, D);
min(H, guitar, octave, degree(2));

Of course, you can always use chord and hardwire the notes, but then it is difficult to change the key of your composition. By using the degree function with chord, it is easy to change the key, but you have to keep track of which chords sound appropriate, i.e. when to play major and when to play minor chords. Songlib’s modal system takes care of all these problems automatically.

Modes and their chords

Note that every mode uses some combination of major and minor chords in its scale. For example, the Ionian mode actually uses the chords:

maj/degree(1), min/degree(2), min/degree(3), maj/degree(4), maj/degree(5), min/degree(6), and dim/degree(7)

so in Ionian mode, c1 is equivalent to maj/degree(1), and so on.

The following table lists the types of chords used for every mode:

<table>
<thead>
<tr>
<th>Mode</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ionian</td>
<td>maj</td>
<td>min</td>
<td>min</td>
<td>maj</td>
<td>maj</td>
<td>min</td>
<td>dim</td>
</tr>
<tr>
<td>Dorian</td>
<td>min</td>
<td>min</td>
<td>maj</td>
<td>maj</td>
<td>min</td>
<td>dim</td>
<td>maj</td>
</tr>
<tr>
<td>Phrygian</td>
<td>min</td>
<td>maj</td>
<td>maj</td>
<td>min</td>
<td>dim</td>
<td>maj</td>
<td>min</td>
</tr>
<tr>
<td>Lydian</td>
<td>maj</td>
<td>maj</td>
<td>min</td>
<td>dim</td>
<td>maj</td>
<td>min</td>
<td>min</td>
</tr>
<tr>
<td>Mixolydian</td>
<td>maj</td>
<td>min</td>
<td>dim</td>
<td>maj</td>
<td>min</td>
<td>min</td>
<td>maj</td>
</tr>
<tr>
<td>Aeolian</td>
<td>min</td>
<td>dim</td>
<td>maj</td>
<td>min</td>
<td>maj</td>
<td>maj</td>
<td>maj</td>
</tr>
<tr>
<td>Locrian</td>
<td>dim</td>
<td>maj</td>
<td>min</td>
<td>maj</td>
<td>maj</td>
<td>maj</td>
<td>min</td>
</tr>
<tr>
<td>Melodic Minor</td>
<td>min</td>
<td>min</td>
<td>aug</td>
<td>maj</td>
<td>maj</td>
<td>dim</td>
<td>min</td>
</tr>
<tr>
<td>Harmonic Minor</td>
<td>min</td>
<td>dim</td>
<td>aug</td>
<td>min</td>
<td>maj</td>
<td>maj</td>
<td>dim</td>
</tr>
</tbody>
</table>

To use Aeolian mode, for example, you could use the chords:

min/degree(1), dim/degree(2), maj/degree(3), min/degree(4), min/degree(5), maj/degree(6), and maj/degree(7)

Of course, songlib figures out which chords and degrees to use if you stick with calls to c(1, ... ) through c(7, ...).

Note the addition of two more modes in the table, the Melodic and Harmonic Minors. Furthermore, there are three alias modes, Major (which is the same as Ionian) and Minor and Natural Minor (which are aliases for Aeolian).
The `setMode` function is used to set the mode. To set the desired mode, capitalize the mode name and substitute underscores for spaces. For example, to set the *Harmonic Minor* mode, make this call:

```cpp
setMode(HARMONIC_MINOR);
```

To change the mode within a song, simply call `setMode` with the new mode. Any chords generated subsequently will use the new mode.

**Flattened and sharpened chords**

Sometimes, it is desirable to call a chord in the next lower (or next higher) key. One can **flatten** and **sharpen** the modal chords for this task. The chords are obtained by calling the `cf` function (for flatten) and the `cs` function (for sharpen) to the modal chord name. For example, suppose your `songlib` program is in E-Phrygian:

```cpp
setKey(E);
setMode(PHRYGIAN);
```

and you wish to play a `c3` chord in the key of D#. You would make a call similar to this:

```cpp
cf(3,beats,instrument,octave);
```

To play a `c4` chord in the key of F, the call would look similar to:

```cpp
cs(4,beats,instrument,octave);
```

**Seventh chords**

The `songlib` library also has functions for generating modal seventh chords. A seventh chord has an additional note one third higher than the highest note in a trichord, yielding a total of four notes, although, sometimes, the next to the highest note in the chord is omitted.

The seventh chord functions are exactly the same as the basic modal trichord functions, but the `c7th` function is called instead of the `c` function. To provide more control over seventh chords, `songlib` provides these additional functions:

<table>
<thead>
<tr>
<th>Common name</th>
<th>technical name</th>
<th>modal function</th>
<th>semitone deltas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major 7(^{th})</td>
<td>Major/Major</td>
<td>c7thM</td>
<td>4, 7, 11</td>
</tr>
<tr>
<td>Minor 7(^{th})</td>
<td>Minor/Minor</td>
<td>c7thm</td>
<td>3, 7, 10</td>
</tr>
<tr>
<td>Augmented 7(^{th})</td>
<td>Augmented</td>
<td>c7tha</td>
<td>4, 8, 11</td>
</tr>
<tr>
<td>Diminished 7(^{th})</td>
<td>Diminished/Diminished</td>
<td>c7thd</td>
<td>3, 6, 9</td>
</tr>
<tr>
<td>Dominant 7(^{th})</td>
<td>Major/Minor</td>
<td>c7thD</td>
<td>4, 7, 10</td>
</tr>
</tbody>
</table>

Recall that the degree function can be used to automatically derive the correct pitch. In *Ionian* mode, the seventh chord with a root of degree 3 would be a minor chord. Thus, in the non-modal system, one would call:

```cpp
chord(beats,instrument,octave,degree(3),+3,+7,+10,(int) 0);
```

Using the modal system, the call would be:
c7th(3,beats,instrument,octave);

in Ionian mode and

c7thm(3,beats,instrument,octave);

in modes where c3 is not a minor chord.

If the current key is the key of C and the mode is Ionian, the notes in both the above chords would be E, G, B, and D. Note that there are three semitones between E and G, seven semitones between E and B, and ten semitones between E and D.

Extended chords

The songlib system provides, ninth, eleventh, and thirteenth chords as well as seventh chords. Those functions are c9th, c11th, and c13th, respectively. One can also flatten and sharpen these chords with the functions c9thf, c9ths, c11thf, c11ths, c13thf, and c13ths.

With ninth chords, songlib omits the 3rd highest note. With eleventh chords, the 4th and 2nd highest notes are omitted, while with thirteenth chords, the 5th, 3rd, and 2nd highest notes are omitted. The next section discusses how to establish finer control over the modal system, including the omission of notes.

Fine control of the modal chord system

All the modal chords are based upon a general purpose function named cchord. If you wish to control the modal chord system at a lower level, you can call the cchord function directly. The function has the following signature:

\[
cchord(chordNumber,duration,instrument,octave,tweaks);
\]

where chordNumber is selected from the set \{1,2,3,4,5,6,7\} and and tweaks is an string of adjustments the sharpen, flatten, or omit the individual notes in the chord. For example, the c chord is simply a wrapper to the call:

\[
cchord(chordNumber,duration,instrument,octave,"xxx");
\]

In the tweaks string, the following encoding is used:
Thus, the chord \textit{c7th} is a wrapper to the call:

\texttt{cchord(chordNumber,duration,instrument,octave,"xxxx");}

If \textit{c4} is a major chord, then it can be tweaked into a dominant seventh chord using the following call to \texttt{cchord}:

\texttt{cchord(4,duration,instrument,octave,"xxp");}

which flattens the seventh note one step down the scale. If \textit{c4} is a minor chord, it can be tweaked into an augmented seventh chord with the call:

\texttt{cchord(4,duration,instrument,octave,"xnx");}

sharpening the middle two notes of the chord.

One should generally stick with tweaks from the set:

\texttt{X Y n N m M l L p P q Q r R}

Applying a pleasing sounding tweak using semitone shifts in one mode will likely sound unpleasant if the mode or key is changed.

In general, the \texttt{cchord} function is called directly if you wish to change the notes that are omitted in an extended chord. For example, a \textit{c9th} chord is a wrapper to:
This tweaks string omits the middle note of the chord; this note is known as the fifth. Music theorists use what seems, at first, a strange terminology for naming the notes of the chord. The following table gives the note names that correspond to the position in the tweaks string:

<table>
<thead>
<tr>
<th>string position</th>
<th>note name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the root</td>
</tr>
<tr>
<td>2</td>
<td>the third</td>
</tr>
<tr>
<td>3</td>
<td>the fifth</td>
</tr>
<tr>
<td>4</td>
<td>the seventh</td>
</tr>
<tr>
<td>5</td>
<td>the ninth</td>
</tr>
<tr>
<td>6</td>
<td>the eleventh</td>
</tr>
<tr>
<td>7</td>
<td>the thirteenth</td>
</tr>
</tbody>
</table>

If you wish to omit the third instead of the fifth, you would call `cchord` like this:

```
cchord(chordNumber,duration,instrument,octave,"x-xxx");
```

### Other modal chords

There are a few other modal chord functions of note:

<table>
<thead>
<tr>
<th>function</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci1</td>
<td>plays an inverted chord; the root note is played an octave higher</td>
</tr>
<tr>
<td>ci2</td>
<td>plays an inverted chord; the middle note is played an octave higher</td>
</tr>
<tr>
<td>ci3</td>
<td>plays an inverted chord; the top note is played an octave higher</td>
</tr>
<tr>
<td>ci4</td>
<td>plays an inverted chord; the root note is played an octave higher</td>
</tr>
<tr>
<td>ci5</td>
<td>plays an inverted chord; the root note is played an octave higher</td>
</tr>
<tr>
<td>ci6</td>
<td>plays an inverted chord; the root note is played an octave higher</td>
</tr>
<tr>
<td>cpower</td>
<td>plays a power chord; the middle note is replaced by the root note at one higher octave</td>
</tr>
<tr>
<td>cpower2</td>
<td>like <code>cpower</code>, but the middle note is played as well</td>
</tr>
<tr>
<td>csus2</td>
<td>plays a suspended chord; the middle note is replaced by the major second</td>
</tr>
<tr>
<td>csus4</td>
<td>plays a suspended chord; the middle note is replaced by the perfect fourth</td>
</tr>
<tr>
<td>cquart</td>
<td>plays some kind of obscure chord</td>
</tr>
<tr>
<td>b</td>
<td>plays a broken chord (arpeggio)</td>
</tr>
</tbody>
</table>

All these chords have the same arguments as `c`, except the broken chord function `b`. With a broken chord, the individual notes are played sequentially or in parallel, according to a set of given patterns, rather than all in parallel. For example, the call:

```
b(1,H,piano,octave,"-x-","x-x",(char *) 0);
```

would play the middle note of the chord followed by the root and the top note played simultaneously. The duration of each pattern is the duration of the chord divided by the number of patterns. In the above example, each pattern is played for a quarter note. Note that tweaks can be used in a broken chord pattern. Also, an “unbroken” broken chord is equivalent to `cchord`. The following calls are equivalent:

```
b(3,W,piano,octave,"xxx",(char *) 0);
cchord(3,W,piano,octave,"xxx");
c(3,W,piano,octave);
```

You can adjust the amplitude each pattern in a broken chord with a call to `setBrokenAmplitudes`: 8
setBrokenAmplitudes(1.0,0.5,DX);
b(1,H,piano,octave,"-x-","x-x",(char *) 0);

Each argument to setBrokenAmplitudes is a factor with which the current amplitude is multiplied before the pattern is played. In this example, the first pattern "-x-" is played at the current amplitude while the second pattern "x-x" is played at half the current amplitude. The factors sent to setBrokenAmplitudes persist until the next call to setBrokenAmplitudes.

Playing single notes in the modal system

You can play the individual notes in the current key, consistent with the current mode. These single note playing functions have the form:

\[ n(chordNumber,duration,instrument,octave,offset) \]

The offset is chosen from the set \{0..6\}. These numbers are the equivalent of do, re, mi, fa, so, la, and ti. For example, the c1 function is equivalent to playing the following notes simultaneously:

\begin{align*}
  n(1,H,piano,octave,0) ;
  n(1,H,piano,octave,2) ;
  n(1,H,piano,octave,4) ;
\end{align*}

You can flatten a note by calling nf instead of n. Likewise, you can sharpen a note by calling the ns function. For example, the chord cm is equivalent to the following notes played simultaneously:

\begin{align*}
  n(1,H,piano,octave,0) ;
  nf(1,H,piano,octave,2) ;
  n(1,H,piano,octave,4) ;
\end{align*}

A power chord omits the third and adds in the root note at an octave higher. You can play a power chord by combining calls to cchord and n. Here is one form of a power chord:

\begin{align*}
  cchord(chordNumber,duration,instrument,octave,"x-x") ;
  backwards(duration) ;
  n(chordNumber,duration,instrument,octave+1,0) ;
\end{align*}

This form has been encapsulated in the cpower function.

Finally, you can retrieve the actual note in a modal chord by using the getcnote function. These two calls are equivalent:

\begin{align*}
  n(3,H,piano,5) ;
  play(H,piano,getcnote(3,5)) ;
\end{align*}