SmartChordGen v0.5 → v1.0 - issues (06-17-2020)

一、数据检验确认 Data double checking

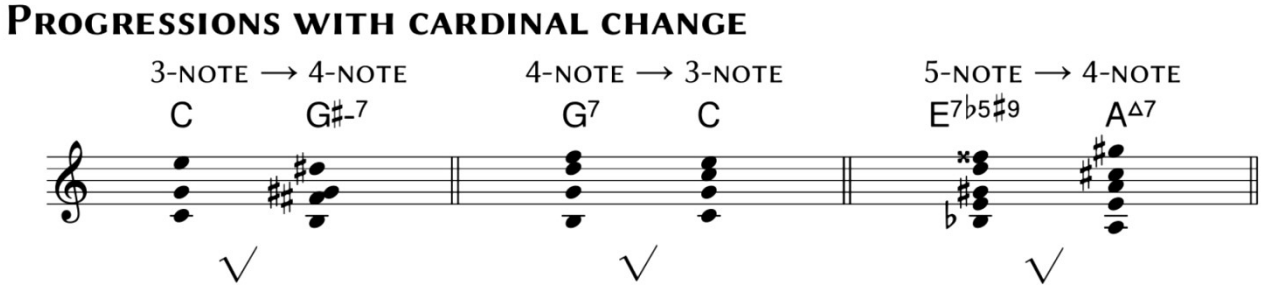
#1） 和弦表勘误（当前标记为“已完成”）

#2） 检查易错点：

#2a） 转位、排列、重复、省略音的所有情形是否已经覆盖完成。例如以下情形：

```
// original: 60 64 67
// inversed: 64 67 72, 67 72 76
// arranged: 60 67 76, 48 55 64 ...
// arranged-repeated: 60 64 67 76, 48 60 64 76 ...
// arranged-repeated-omitted: 48 64 88, 48 60 76 88 ...
// arranged-repeated-omitted-inversed: 64 88 96, 76 88 96 ...
// or, any combinations of inversion, arrangement, repetition and omission.
```

#2b） “前后音数不同，进行的向量不能做到一一对应，需以重复最接近的音的方式转化为相同的音个数，再算出进行向量”的功能是否工作正常。



二、增加功能 Function request

#1） 和弦对照库统一为 **.db（Database）**文件格式，用户在运行程序时选择库文件（输入 **.db** 文件名）。如果留空，则默认用 ***chord-popjazz.db*** ：

```
> Please input name of the database (.db) file: (press ENTER for default - Pop and Jazz):
  chord-rock
```

.db 文件（**Plain text** 格式）示例如下（***chord-popjazz.db*** ，可直接复制粘贴此格式）：

```
// SmartChordGen Chord Database: Pop and Jazz
// (c) 2020 Wenge CHEN, Ji-woon SIM .
// Database begins below.
0 3 6
0 3 7
```

```
0 4 7
0 4 8
0 5 7
... 。
```

统一格式后，这边将给出更多的对照库供选择（rock，pentatonic，dissonant，triads，等等）。

#2） *Charming Chords* 增加 vec 和 tension（基于 d_all）：

d_all - 原理与 d 一样，只是对所有的音程都计算一次。如：

```
a = [0, 2, 4];
// d(a) == [2 - 0, 4 - 2] == [2, 2]
// d_all(a) == [2 - 0, 4 - 0, 4 - 2] == [2, 4, 2]
```

d_all 的结果不需显示。

vec - 按 [1, 2, 3, 4, 5, 6] 的数目对 d_all 计数，输出为无逗号间隔的字符串：

```
a = [0, 2, 4];
// vec(a) == [020100]
```

tension（t）为和弦的“紧张度”，原算法定义为 vec 乘以 [16, 8, 4, 2, 1, 16] 后相加得一个整数：

```
a = [0, 2, 4];
// Because: vec(a) == [020100],
// t(a) == 0 * 16 + 2 * 8 + 0 * 4 + 1 * 2 + 0 * 1 + 0 * 16 == 18
```

（同度和任意八度音程紧张度定义为 0。）

为了适应实际的音乐应用，现对紧张度算法引入“**八度衰减**”和“**低音程限制**”：

· **八度衰减**：同一音程每隔开 n 个八度，该音程的紧张度乘以 1/(n+1)。

示例：60 64 的音程为 (64 - 60).mod(6) == 4，隔开了 floor((64-60)/12) == 0 个八度，紧张度为 2。

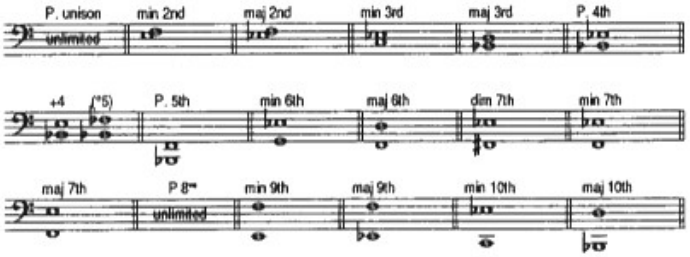
48 76 的音程为 (76 - 48).mod(6) == 4，隔开了 floor((76-48)/12) == 2 个八度，紧张度为.66。

· **低音程限制**：

根据：**Pease, T. and Freeman, B., 1996. Arranging 2: Workbook. Berklee College of Music, p.62.**

声学实验证实，音程在低过某个特定（界限）音时，音高会变得越来越不清，音响也会越来越浑浊、紧张。

各音程的界限音如下图所示（音程的低音为界限音）：



写成数字形式如下：

音程	0	1	2	3	4	5	6	7	8	9	(9*)	10	11	12	13	14	15	16
界限音	无	52	51	48	46	46	46	32	43	41	(42)	41	41	无	40	39	36	34

检查和弦中有否低音在相应界限音以下的音程，如有，取其界限音与其低音的键位号之比与原紧张度相乘。

示例：36 76 的音程为 (76 - 36).mod(16) == 4，隔开了 floor((76-36)/12) == 3 个八度；

又，音程 4 的界限音为 46，36 < 46，因此，将其紧张度乘以 46/36。

得：36 76 的紧张度为：2 * 1/(3+1) * (46/36) == 0.64 （取两位小数）。

对于音程为 17（含）以上的情况，所参考文献的声学实验并未提供界限音数据，即用如以上举例的方法，无论该音程多大，都统一对 16 取余后以隔开八度的数量进行“延拓”处理。

特别地，对音程 9*（减七度，与音程 9 —— 大六度半音数相等）由于目前无法判断，先不做处理。

总体效果：

```
Original chord #1:
[0, 2, 4],  d = [2, 2], vec = [020100], t = ...
3 progressions

// [1, -1, 0]
>> [1, 4],  d = [3], vec = [001000], t = ...
k = -2.5,  c = 1

... 。
```

在 SmartChordGen 中，依此增加 T% 的筛选功能（0 < Tmin <= Tmax <= 100）。

#3） *Charming Chords* 中对 c 值的计算改进：

（根据知乎 @叶小胖 指导）对于“6”（与根音相距三全音、增四减五度的音）的色差，最理想的方法是：**增四度取 +6，减五度取 -6（类似于调号）；允许在特定条件（十二平均律、等音转换）下互相转化。**

原程序中，该取值为 0；如今特定条件难以确定情况下，**先统一取为 +6**：

```
// k(1 .. 11) = [-5, 2, -3, 4, -1, 6, 1, -4, 3, -2, 5].
```

#4） 另写 5 个 utility 小程序：chord(check，sect，trans，stats)，writedb，可以重用部分代码：

功能示例（相信看到示例已经知道它做什么了）：

· **Utility - Chord check:**

```
> Please input chord A: 60 67 76
> Please input chord B: 60 68 75
>>  result: d(A) = ..., vec(A) = ..., t(A) = ...,
            d(B) = ..., vec(B) = ..., t(B) = ...,
            n(A) = ...,  n(B) = ...,   diff(n(A, B)) = ... //n 代表音数
            m(A) = ...,  m(B) = ...,   diff(m(A, B)) = ... //m 代表厚度（八度计数）

            k(A->B) = ...,    c(A, B) = ...

            A^B = ...,  set(A^B) = ...,

            d(A^B) = ...,  vec(A^B) = ...,  t(A^B) = ...

            AvB = ...,  set(AvB) = ...,
            d(AvB) = ...,  vec(AvB) = ...,  t(AvB) = ...

> Check isSmart? (Y/N) Y
>  Please input name of the database (.db) file: (press ENTER for default - Pop and Jazz):  chord-triads
>>  result: isSmart(chord-triads) == True .
```

（以上的 isSmart 检验了在指定的 db 在 SmartChordGen 中输出时是否包含该和弦进行。）

· **Utility - Chord sect**（对文件中两两相邻的和弦取“交”或“并”，并输出新文件）：

```
> Please input name of the chord data file (with extension, e.g. input.txt): input.txt
```

（示例 *input.txt*：）

```
60 64 67 72
60 63 67 72
60 64 67 70
```

```
> Would you like to intersect (Y) or union (N) adjacent chords (Y/N)?  Y
> Please assign a name for the output file (with extension): intersect.txt

Please wait for the application to output the result to [intersect.txt]...
```

（示例 *intersect.txt*：）

```
60 67 72
60 67 70
```

> Would you like to intersect (Y) or union (N) adjacent chords (Y/N)? N
> Please assign a name for the output file (with extension): *union.txt*

Please wait for the application to output the result to [*union.txt*]...

（示例 *union.txt*:）

```
60 63 64 67 72
60 63 64 67 70 72
```

· **Utility - Chord trans**（对文件中的和弦取移位及以指定音为轴的翻转）：

> Please input name of the chord data file (with extension, e.g. *input.txt*): input.txt

（示例 *input.txt*:）

```
60 64 67 72
60 63 67 72
60 64 67 70
```

> Please input number of semitones to transpose (press ENTER for no trans): -2
> Please input the axis note for inversion (press ENTER for no inversion): 58

> Sort the result (Y/N)? Y（翻转后每个和弦音按从小到大排列）

> Please assign a name for the output file (with extension): *trans.txt*

Please wait for the application to output the result to [*trans.txt*]...

（示例 *trans.txt*:）

```
46 51 54 58
46 51 55 58
48 51 54 58
```

· **Utility - Chord stats**（和弦进行的统计工具，输出统计信息到屏幕/用户选择输出到文件）：

> Please input name of the chord data file (with extension, e.g. *input.txt*): ...
>> results:
>> Voice leading stats: （声部进行方向的统计）
>> movement instances: [0] .. （计数）
[+1] .. [+2] .. [+3] .. [+4] ..
[-1] .. [-2] .. [-3] .. [-4] ..
（固定顺序，0 单独一行，上下行即正负数各单独一行）
>> movement percentage (sorted H->L): （按出现频率从高到低排序，百分比取整数）
[x] ..% [x] ..% [x] ..% [x] ..% [x] ..% [x] ..% [x] ..% [x] ..% [x] ..%
>> cardinal change instances: ..（前后 n 变化的出现频次计数）（..%）（百分比取整数）

（空一行）

```
>> Other stats:
>> Cardinal (n): highest - ..  (@ # xx（和弦序号）)  ;  lowest - ..  (@ # xx)  ;
>> Tension (t): highest - ..  (@ # xx（和弦序号）)  ;  lowest - ..  (@ # xx)  ;
>> Chroma (|k|): highest - ..  (@ # xx（和弦序号）)  ;  lowest - ..  (@ # xx)  ;
>> Thickness (m): highest - ..  (@ # xx（和弦序号）)  ;  lowest - ..  (@ # xx)  ;
>>  Common tones (c):  most - ..  (@ # xx（和弦序号）)  ;  least- ..  (@ # xx)  ;
```

（注意：对色差的统计是取 k 的绝对值的最大/小值。）

```
> Copy the stats to a file (Y/N)? Y
> Please assign a name for the output file (with extension): stats.txt

Please wait for the application to output the result to [stats.txt]...
```

（输出统计内容到文件。）

· **Utility - Write database:**

```
> Please assign a name for the database: test
> Please input name of the author of the database: Test USER
```

（一种选择：用“vec 法”生成）

```
> Would you like to generate db file by vec(Y) or by subsets(N)? (Y/N) Y
> Please input desired number range for interval “1”: 0 3
> Please input desired number range for interval “2”: 0 3
> Please input desired number range for interval “3”: 0 3
> Please input desired number range for interval “4”: 0 3
> Please input desired number range for interval “5”: 0 3
> Please input desired number range for interval “6”: 0 3
```

（另一种选择：用“子集法”生成）

```
> Would you like to generate db file by vec(Y) or by subsets(N)? (Y/N) N
```

（输入一个和弦或者音阶，将其自动化简为 0 开头的一个八度内音集，并输出所有其含自身的子集。）

```
> Please input desired scale or chord as the original set: 60 64 69 74 79
  >> Please note that the input is normalized into a compact set: [0 2 4 7 9].

> Please input desired range of numbers of notes (n) for all subsets: 2 5


> Would you like to filter by tension value (T)? (Y/N) Y（选 N 则不按紧张度筛选）
> Please input desired range for tension value (T) in integer: 10 40

Please wait for the application to output the result to [test.db]...
```

（以上输出：）

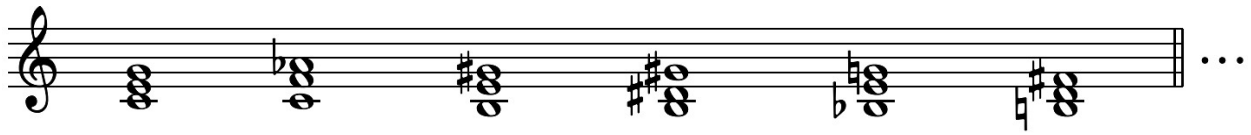
```
// SmartChordGen Chord Database: test
// (c) Test USER.
```

// Database begins below.

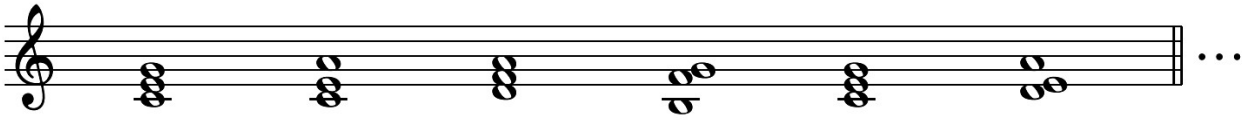
0 ... 。(满足以上条件的和弦列表)

#5) 对主程序进行两项功能增补，如下：

·“宏和声”(整体音阶)： 在音乐的和声学术语中,“宏和声”(macro-harmonony)是所有和弦并集所得大音集，俗称“整体音阶”(the overall scale)。对其不加限制时，所生成的结果会在十二个半音之间自由地游荡：



而我们把宏和声（整体音阶）限制在自然音阶（CDEFGAB，音集：[0 2 4 5 7 9 11]）以内，则得到：



这有助于利用程序的创作者在自己喜欢的音阶范围内进行创作。

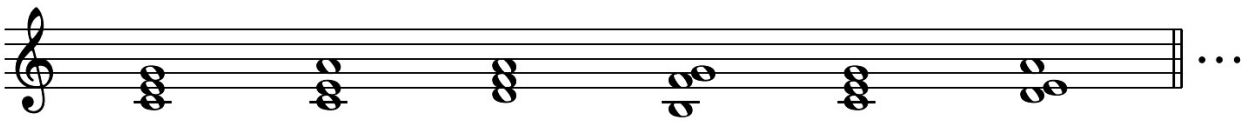
引入这一条件相当于在查验第一个条件时增加一个额外的条件，**在开始生成之前询问**。界面效果示例如下：

> Would you like to constraint the overall scale (Y/N) ? Y （选 N 则不限制）

> Please input desired constraining scale set (0~11, with space): 0 2 4 5 7 9 11

限制可能导致生成结果无解；此时跟别的无解情况一样，输出“ERROR - no solution found!”即可。

· 去除重复和弦/不重复和弦类型的生成： 也是在开始生成之前询问。界面效果示例如下（仍以上例为例)：



> Remove duplicate chords (Y/N) ? Y （**第 5 个和弦将被去除，因为跟第 1 个相同，总保留最先的。**）

> Generate in unique chord types (Y/N) ? Y（**选 Y 则不会产生如第 2、3 个和弦的情况，因为它们都是小三和弦，属于同一个音集，并非“唯一”；总保留最先的。**）

>> Please note that you have chosen ‘unique chord types’. When chord (set) types are all used up, the generation will be automatically terminated.

（在连续生成模式中，当和弦类型被用完，则自动终止。）

#6) 用户输入提供多一种选择：音名的输入方式，见文件：*midinote.db.xls* 。

#Last）争取实现写出 MIDI 文件的功能。

三、界面的细枝末节 Interface

1．程序名定为：SmartChordGen。程序启动时加入如下文字：

```
[[  SmartChordGen v1.0 [Build: 07/01/2020]  ]]
[[  (c) 2020 Wenge CHEN, Ji-woon SIM .      ]]

> Welcome to SmartChordGen !
```

2．原有的 *Charming Chords(2019.11.12)* 变为其中的一个 utility（即更新 *Charming Chords(2019.11.12).cpp/exe* 后改名：*SmartChordGen-utility-charmingchords.cpp/exe* ）。