1 Description of proposed program structure

In this document I am aiming to describe in detail the proposed structure of the music generator program to be produced as part of my MSc dissertation project. The music generator program should produce musical fragments which are able to meet some of the criteria of creativity outlined in my discussion of the aims for the program. A high priority in the realisation of this program is to have constructed a system that, upon analysis of the processes whereby it produces the musical fragments, the program may be judged as employing a degree of skill and relative originality in the composition of the fragment, as opposed to be being based largely on random generation or mimicking the musical styles of other composers.

In designing the structure of the program, and important aim has been to promote modularity of the code in order to allow the potential for further developments and refinements at a later date, and to make the structure of the code as easily comprehended as possible by those unfamiliar with it. Thus, the program can be considered to constitute these core modules:

1. **A music generator module.**
2. **A grammar generation module.**
3. **A music analysis module.**

Further modularity involves the integration of JMusic as the means of representing the output of the music generator module audibly and visibly (as musical notation).

I will now proceed to explain how each of the proposed modules will function, and to provide implementation details.

2 Music Generator Module

The music generator module takes as its input an object that contains the grammatical structure of the musical piece to be produced. This grammar object will encapsulate the following details:

2.1 The grammar input

An ordered, sequential, and hierarchical list of beats within the fragment. This represents the metrical structure of the piece.

A segmentation of the metrical structure above into groups. This segmentation is hierarchical and represents the grouping structure analysis.

A segmentation of the metrical and grouping structure into prolongational groups, representing antecedent and consequent formations. In smaller analyses this may be omitted, and in such a case the whole fragment will be considered to represent a basic prolongational structure.

Relevant metrical beats will be associated with a timespan branch. The branches are structured as a tree structure, representing the time span reduction analysis of the piece.

Relevant metrical beats will be associated with a prolongational branch. The branches are structured as a tree structure, representing the prolongational reduction of the piece. Each branch identifies whether it represents a strong prolongation, weak prolongation, or progression of the pitch event associated with its parent branch.

For clarification:

Given the following structure:

	1	2	3	4	5	6	7	8	9
Metrical:	.	.	.	.	.	.	.	.	.
	.		.		.		.		.
	.				.				.
Grouping:	[			]	[				]

Time span reduction branches could be assigned in the following hierarchy:

The beat of 9 is associated with the top branch, which has as a child a branch associated with the beat 1. This branch could have as its child a branch associated with the beat 3. Finally, the original branch associated with 9 could have a further child branch associated with the beat 6. Beats 7, 8 4 and 2 may not be associated with any branch, indicating that they are not attack events.

Prolongational reduction branches will also be exhaustively allocated in this way, but each branch will represent either a strong prolongation, weak prolongation, or progression to/of the pitch event.

2.2 Generating music from the grammar

Taking this grammar as input, the music generator will create an ordered, sequential list of event objects, each representing a beat as described by the metrical structure. It will following the following general method in order to generate the musical fragment:

1. For each beat associated with a prolongational branch or time-span branch, set it as an attackevent object and set its pitch to be that of the tonic note.

2. Using the music analysis module, for each attack event establish candidate pitchs and durations. This is done in the following way:

(a) Establish for each attack event candidate pitch values that lie within the given key.

(b) Starting from the highest branch of the prolongational reduction, establish candidate pitch values based on ranking appropriate to level. Lower levels = less stable: generate appropriate pitches. In this implementation, the appropriate pitches will be generated by testing all the established candidate solutions for each attack event to see how they rank according to the PR rules, and their particular PR level. See the section of the music analyser for the overview of the ranking system.

(c) As in b, starting from the highest branch of the time span reduction work through the branches as manipulate the candidate pitch solutions associated with each attack event. This is done by consulting the music analyser time span rules ranking system. Lower branches = further away from tonic + less consonant.

(d) Using the grouping structure to identify which attack events lie on boundaries, establish candidate durations and pitches based on boundary ranking returned from the music analyser module.

(e) Using metrical structure, establish candidate durations for attack events, and candidate rest events based on rankings returned from music analyser.

3. After the potential pitch values has been reduced in this way, assign definite pitch events. This may be done by choosing them at random from candidate pitch and duration values, but could possibly be done by reaffirming their ranking according to the music analyser. These can be assigned starting with the first branch of the prolongational reduction.

After this process, the EventStream object can be passed to the music player for audible representation.

3 Music Analyser Module

The music analyser module has two potential purposes: firstly, it could be used to help analyse the grammatical structure of an existing piece of music, allowing the potential for that structure to be passed into the music generator. It also has the purpose of helping to establish the suitability of different pitches and duration within specified contexts to aid the music generator. The analyser works by ranking the suitability of given AttackEvents within their specified context. These rankings can be used by the music generator to establish their suitability within the broader grammatical structure of a piece. In this way, a low ranking may not mean that the attack event is not suitable in the given context: indeed, a low ranking may be used by the music generator to establish that the attack event is highly suited to that context.

Here I shall list the main components of this module, which comprise the different GTTM grammatical rules. This project will not try to implement all of the GTTM rules, but only the ones that I believe will be most effective in helping to generate musical fragments that exhibit musical structure.

3.1 Grouping Structure Rules

This component works to establish motivic groups by ranking whether a given note is suited as a boundary event for a group, or not. It takes an input for four attack events, n1, n2, n3, n4, and returns a ranking value based on the suitability of n2 as a boundary event.

This component aims to implement GPR 2 and GPR 3. These rules assess the suitability of n2 as a boundary based on such things as whether n1-n2 and n3-n4 is of a smaller duration than n2-n3, whether the intervalic distance between n1-n2 and n3-n4 is smaller than the distance between n2-n3, etc. The more a pitch event meets the criteria stated in these rules, the higher its ranking.

This can be used in the music generator, for example, by firstly establishing an attack event candidate for a boundary that occurs on a high level of the grouping hierarchy. This will demand a strong ranking. On lower levels, an attack event on a boundary which is not a boundary at a higher level will require a relatively smaller ranking to be judged as suitable. Attack events that are not on boundaries should receive very low rankings.

Two methods might be implemented for this: one assessing the pitch suitability, and one assessing the duration suitability.

3.2 Metrical Structure Rules

The metrical structure ranking system will be implemented to provide a strong ranking if a given attack event as an input is suited to positioning on a strong beat. This is particularly important for establishing candidate durations for an attack event, and will largely be focused on implementing parts of MPR 5.

3.3 Time Span Reduction Rules

This component will be particularly focused on providing a ranking system that makes use of the TSPR 2 rule, as well as TSPR 3 rules (which will have a weaker influence in recommending an attack event candidate). The ranking associated with TSPR 2 will return a strong ranking if the given attack event is consonant with the parent attack event, and/or close to the tonic pitch value. Conversely, the music generator can use this ranking to establish candidate pitches for those attack events at the bottom of the time-span reduction hierarchy.

3.4 Prolongational Reduction Rules

The prolongational reduction ranking system will focus on the PR3 rules to establish relative stability of an attack event candidate. It takes the prolonga-

tional region end attack event and the candidate attack event and branch type as inputs. Using details about the branch type and pitch value it will return a ranking that indicates the stability of the candidate pitch in relation to the end pitch. The music generator can use this to establish candidate attack events which represent high stability when at the top of the prolongational reduction hierarchy, and attack events which exhibit low stability when at the bottom of the hierarchy.

4 The grammar constructor module

This module will be used to construct different grammatical structures to be input into the musical generator. The construction of this module will take a naive form, producing only basic, uncomplicated structures. The module should allow for the potential for a more complex implementation to be created at a later date for the construction of more interesting grammar structures. At this stage the grammar constructor module will only be constructing a grammatical structure that represents one musical phrase.

The constructor will generate grammars in the following way:

1. Produce a prolongational normal form that represents the prolongational structure of a musical phrase; that is, with a segmentation into an antecedent and consequent form. Thus the grammatical structure will have two prolongational segments as prolongational regions.

2. Create a starting prolongational branch that represents the prolongational end point.

3. Create a second branch as a child of the starting branch, representing the prolongational starting point.

4. Establish a right branching progression from the prolongational beginning.

5. Establish a left branching progression to the prolongational ending.

6. Provide subdivisions of this hierarchy (probably established at random).

7. Using PRPR 1, establish corroborating time span reduction branches to the event objects contained within the prolongational branching objects.

8. Identify the cadenced two attack events and assign them the cadence time-span reduction branch.

9. Using established attack events, construct metrical structure low level list.

10. Using the fact that a cadenced region should have a stable metrical structure, ascertain a candidate metrical structure for that region and work backwards to establish a suitable metrical structure over the attack events.

Add extra event objects where needs be to promote a strongly stable metrical structure. (Such extra events may represent that fact that the previous attack event has a longer duration, or that a rest event exists on this beat).

11. Establish the grouping structure, ensuring that groups do not consist of a single element (not too short), and that non attack event events are preferred at grouping boundaries. Attack events at strong metrical beats are preferred as the beginning of a group.

4.1 Things left unclarified

How to deal with cadences.

Exactly how things will be ranked.

Rank candidate pitches according to how well they are suited to their context?

How to get around problem of prolongational reduction ranking relying on solutions to the endpoints of the piece (in naive implementation, just give the end point one pitch?). I'm going to look into constraint logic programming for Java for solutions to this.

5 Proposed schedule

The priority of this project will be to develop the music generator module. The grammar generator module is an important aspect of the project, but if worst comes to worst the grammar inputs to the generator module can be written by hand, or partially by hand. Therefore I will work firstly on developing the music analyser module followed by the music constructor module.

25th June - 30th June
Create basic classes for basic elements. Create the grammar container for use as input to the music generator.

July 1st - July 7th
Develop the grouping structure and metrical structure rules component. If time permits, and if necessary, refactor, else implement JUnit tests.

July 8th - 14th
Develop the time span reduction rule component (1-2 days?). Develop the prolongational rules component. JUnit tests and refactoring as required.

July 15th - 20th
Develop the music generator module. Making use of the established ranking system to generate musical fragments from an input grammar.

July 21 - 27th
Develop the naive grammar constructor. Link it with the music generator module.

July 28th - 30th

Further testing and refactoring. Generate a number of examples from the generator, based on hand written grammars, and those generated by the naive generator for comparison purposes.

First week of August: Develop a survey to test the generated pieces of music.

Second week of August: distribute survey. (May potentially have a friend coming to stay for a few days this week.)