m

A language for music generation.

Ethan Hann (eh2413)
Yiling Hu (yh2378)
Monica Ramirez-Santana (mir2115)
Jiaying Xu (jx2129)

Language Description

The language will allow a composer to write algorithms that generate music. The fundamental type is *Integer*. All other types are derived from this type. The derived types are based on traditional music concepts. *Notes* consist of a pitch, intensity (volume), and duration. *Chords* are defined as a list of *Notes*. *Notes* and *Chords* are assigned to *Staffs*. All of the *Notes*/*Chords* on a staff are played at a specific tempo, in the order that they were added. Multiple *Staffs* can be included into a *Part.* This allows songs to have different sections of varying tempo, akin to traditional music.

Type Summary:

Types	Derived Types
Integer	Note
	Chord
	Staff
	Part

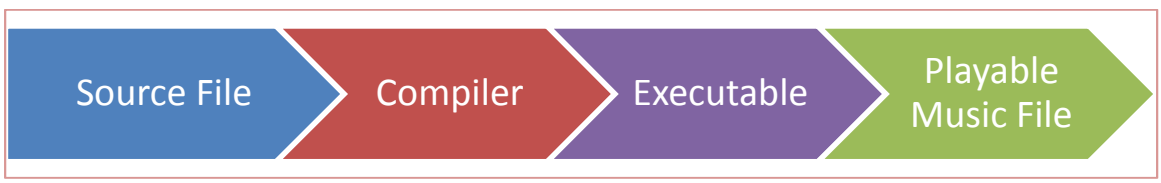
When a program written in the language is compiled, all *Parts* are played simultaneously. This feature of the language can be used to mimic the right hand and left hand sections of a piano piece or the different instruments of a multi-instrument ensemble.

Problem Language Solves and How It Should Be Used

Many interesting uses for the language exist. This is due to the fact that all musical phenomena are derived types of the *Integer* type. The language makes possible the mathematical manipulation of music through integer arithmetic, randomization, and classical programming control structures. Traditionally, composers would have to write out every note of their composition, but using this language the composer could simply specify sets of notes and chords that are used in conjunction with control structures, and a randomization function, to algorithmically generate music. Algorithms designed by the composer, and in the standard library, can generate music in a particular genre, tempo, and key.

An executable program is produced when an *m* source file is compiled. When the program is executed a MIDI/WAV file, which can be played by any variety of music players, is generated. The resultant music file may be different each time the program is executed depending on how the composer utilized the randomization functionality of the language.

Language Compilation Process



Interesting, Representative Program

The following sample code demonstrates how an algorithmically generated music scale based on the Fibonacci sequence is generated using the *m* language.

7 Note Fibonacci Music Scale

```
Part $guitar_part = (nothing, ElectricGuitar); /* (Staff, Instrument) */
Staff $fib_scale = (nothing, 120, 4); /* (Note/Chord, BPM, Time Signature) */
Note $temp_note = (Cn, 4, 100); /* (Pitch, Duration, Intensity) */
Integer $i;

For ($i = 2; $i <= 9; $i++)
{
	Integer $n = $i, $first = 0, $second = 1, $tmp;
	While ($n--)
	{
		$tmp = $first + $second;
		$first = $second;
		$second = $tmp;
	}
	$fib_scale.notes.add($first);
}

$guitar_part.staves.add($fib_scale);
```

A real world application of the language could be to generate music for Pandora's (www.pandora.com) user radio stations. The generated music would add novelty and diversity. Stations would no longer be limited to the realm of recorded songs. Instead, each song would be different from any that the user had heard before. This would solve the problem of users becoming bored with their radio stations.