Musicol

Created by Lynn Hedegard

Top – Level Song Structure

A "**Song**" is made up of one or more "**Music Tracks**" …

Track-1
Track-2
Track-3

… and one and only one **Tempo Track**, which has global attributes;

- **Time signature:** (e.g. 2/4, 3/4, 4/4, 7/8, etc.)
- **Divisor:** Smallest note typically played (e.g. 1/8, 16, 1/32)
- **Tempo:** (e.g. 120 Beats per minute (BPM))

Piano-Track
Bass-Track
Drum Track
Tempo

Tracks & Sections

A “**Track**” is made up of one or more “**Sections**”

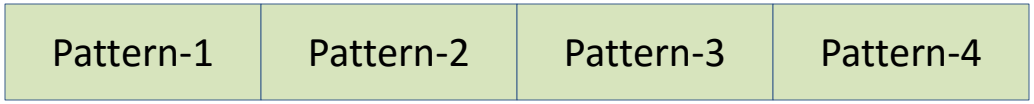
Sect-1	Sect-2	Sect-3	Sect-4	Sect-5	Sect-6

Example “**Tracks**” & “**Sections**”

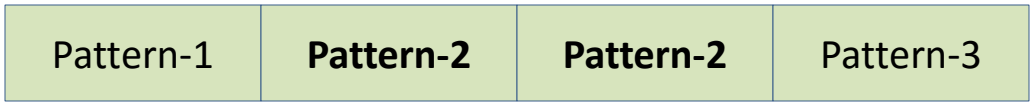
Piano:	P-Intro	P-Verse-1	P-Verse-2	P-Chorus	P-Verse-3	P-Ending
Bass:	B-Intro	B-Verse-1	B-Verse-2	B-Chorus	B-Verse-3	B-Ending
Drums:	D-Intro	D-Verse-1	D-Verse-2	D-Chorus	D-Verse-3	D-Ending
Tempo:	Tempo-1			Tempo-2	Tempo-3	

Sections & Patterns

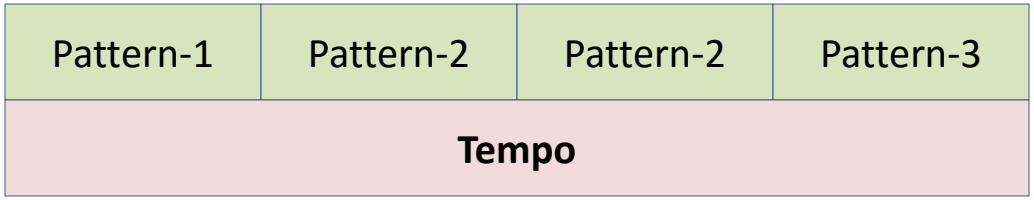
A “**Section**” is made up of one or more “**Patterns**”



Patterns can be repeated



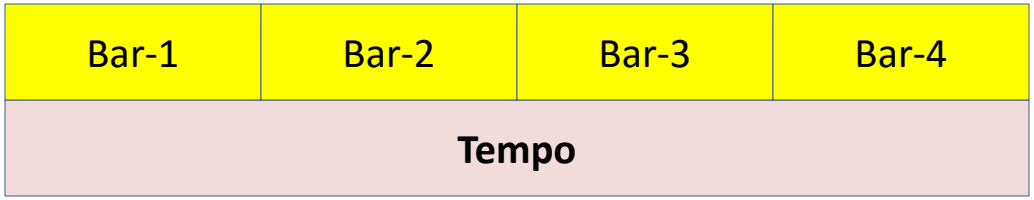
All **Patterns** are governed by the Global Temp Track



Patterns

A “**Pattern**” is made up of one or more “**Bars**”

All **Bars** are governed by the Global Temp Track



TIME Data Type

- **Time** data types are used to define when a musicol event occurs
  - **Bar number**: The bar number where the object BEGINS to sound (relative to the containing pattern). Bar numbers begin counting with “1”.
  - **Beat number**: The beat number (relative to the current bar mentioned above) where the object BEGINS to sound. Beat numbers begin counting with “1”.
  - **Divisor Number**: The “divisor offset” for the object -- based on an inherited global value for the smallest note to be played. For example, if I only want 1/8 notes or larger, then there are 2 values — (0,1). If I only want 1/16 notes or larger, then there are 4 values — (0,1,2,3). If I only want 1/32 notes or larger, then there are 4 values — (0,1,2,3,4,5,6,7). Devisor Numbers begin counting with “1”.

TIME Data Type – (cont)

- **Clock Offset**: An amount of time after the beat number mentioned above that the note BEGINS to sound. This gives us the ability to *"humanize"* a pattern (talented players will either *"push"* certain beats, or *"lay back"* — usually beats 2 & 4). Clock Offset begin counting with "0", which means no offset, and have a max of 119, which would be one clock pulse prior to the next bar.beat.devisor.

  To simplify the notation, Clock values can be omitted

- **Example**s

  Assume: Time-Sig := 4/4 & Divisor = 1/16 (i.e. there are 4 subdivision to a beat)

  - `1.1.1.0` := Bar-1, Beat-1, Down-Beat
  - `2.3.2.0` := Bar-2, Beat-3, Up-Beat
  - `3.4.1` := Bar-3, Beat-4, Down-Beat

NOTE Data Type

- A **NOTE** is a basic data type in the language. A Note has three attributes
  - **Note Number**: This is defined as a simple integer, with the lowest note being 000, and the highest note being 999. The language will allow for an alpha-numeric alias, such as C4, D#3, etc., where C0 := 000, C#0 := 001, D0 := 002, C1 := 012, etc.
  - **Note-ON time stamp**: This is a time stamp that defines when the note begin to sound.
  - **Note-Length**: This is the time stamp the defines the **LENGTH** of the note.

NOTE Data Type

- The syntax for a note is;

```
MyNote = Note{Note-Num, On-Time, Length}
```

- Assume the Global Divisor is set 4, which means that the smallest “standard” note we will play is a 1/16 note.

- Example note declarations;
  - `N1 = Note{60, 1.1.1.0, 0.1.0.0}`

    This would be a middle C note played on the first beat of the first bar of the pattern, and lasting for one beat.

  - `N2 = Note{67, 1.1.3.0, 0.0.2.0}`

    This would be a G just above middle C played on the “up beat” (i.e “1 and”) of the first beat of the first bar of the pattern, and lasting for half of a beat (i.e. 1/8)

Pattern Object

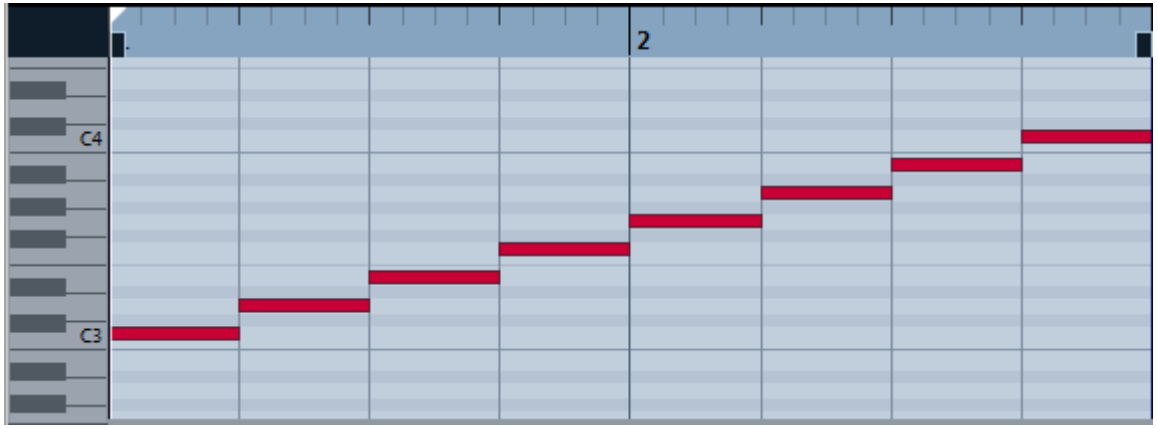
- Pattern Objects are collections of Note-Objects and/or other Pattern-Objects.

- Example of a simple pattern

```
P1 = Pattern{N1, N2, N13, N8, N21}
```

- It is interesting to observe that the order of the note-objects in the declaration has no effect on the order that they are played, because the note-objects define when a given note gets played within a given pattern.

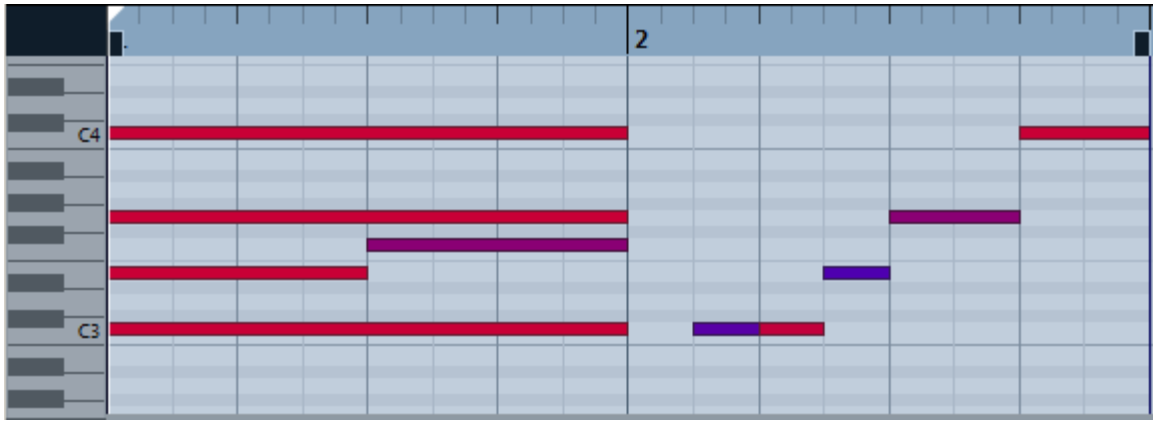
NOTES & PATTERNS (Ex. 1)




```
N1 = Note{36, 1.1.1.0, 0.1.0.0}
N2 = Note{38, 1.2.1.0, 0.1.0.0}
N3 = Note{40, 1.3.1.0, 0.1.0.0}
N4 = Note{41, 1.4.1.0, 0.1.0.0}
N5 = Note{43, 2.1.1.0, 0.1.0.0}
N6 = Note{45, 2.2.1.0, 0.1.0.0}
N7 = Note{47, 2.3.1.0, 0.1.0.0}
N8 = Note{48, 2.4.1.0, 0.1.0.0}

P1 = Pattern{N1, N2, N3, N4, N5, N6, N7, N8}
```

NOTES & PATTERNS (Ex. 2)




```
N01 = Note{36, 1.1.1.0, 0.4.0.0}
N02 = Note{40, 1.1.1.0, 0.2.0.0}
N03 = Note{43, 1.1.1.0, 0.4.0.0}
N04 = Note{48, 1.1.1.0, 0.4.0.0}
N05 = Note{40, 1.3.1.0, 0.2.0.0}

N06 = Note{36, 2.1.2.0, 0.0.1.0}
N07 = Note{36, 2.2.1.0, 0.0.1.0}
N08 = Note{40, 2.2.2.0, 0.0.1.0}
N09 = Note{43, 2.3.1.0, 0.1.0.0}
N10 = Note{48, 2.4.1.0, 0.1.0.0}

P1 = Pattern{N1, N2, N3, N4, N5, N6, N7, N8, N9, N10}
```

PATTERN – Recursive

- Patterns can contain other patterns. When we do so, we include a time stamp to indicate where in the current pattern the sub-pattern is to begin. To keep things simple, we omit the Clock-Offset.

- `P2 = Pattern{P1:1.1.1, P1:2.1.1.1, P1:3.1.1.1, P1:4.1.1.1}`

  Pattern P2 contains 4 identical patterns (P1) beginning on beats 1, 2, 3, & 4.

- `P3 = Pattern{Repeat(P1, 4):1.1.1, Repeat(P2, 2):1.1.1)}`

  Play pattern P1 four times in sequence, beginning at bar 1, beat 1 of the current pattern AND play pattern P2 two times beginning at bar 1, beat 1 of the current pattern.

PATTERN Operators

Pattern can be modified using well defined operators, such as transpose, time-shift.

- `P2=ShiftPattern(P1, +0.2.1)`
- This will create a new pattern, P2, which is the same as P1 except for the fact that all notes will be shifted to the right by 2 beats.

- `P3 = ExpandPattern(P1, 2)`
- This will create a new pattern, P3, which is the same as P1 except for the fact that all notes will be twice as long, and start at the associated time that is two times later

- `P4 = ContractPattern(P1, 2)`
- This will create a new pattern, P4, which is the same as P1 except for the fact that all notes will be half as long, and start at the associated time that is two times sooner

PATTERN Operators

Pattern can be modified using well defined operators, such as transpose, time-shift.

- `P2 = TransposePattern(P1, 5, Absolute)`
- This will create a new pattern, P2, which is the same as P1 except for the fact that all notes will be increased by exactly 5 semi-tones.

- **Extra-Credit**
- `P3=TransposePattern(P1, 2, Relative, D-Major)`
- This will create a new pattern, P3, which is the same as P1 except for the fact that all notes will be increased by either 4 or 5 semi-tones, based on the input Key-Signature.

SECTION

- Sections are a collection of one or more patterns.
- Sections have a START Time relative to the Global Tempo track BAR NUMBER (Bar Numbers begin counting at “1”)

  `S1 = Section{<BN>, P1, P1, P1, P2}`

  When no time-stamp is included in the Pattern object, the patterns follow in sequential order.

- We may include a time stamp to indicate where the current pattern should begin. To keep things simple, we omit the Clock-Offset.
- `S2 = Section{9, P1:1.1.1, P1:5.1.1.1, P1:13.1.1.1}`

- `S3 = Section{25, Repeat(P1, 4):1.1.1, Repeat(P2,2):1.1.1,)}`

  Play pattern P1 four times in sequence, beginning at bar 1, beat 1 of the current pattern AND play pattern P2 two times beginning at bar 1, beat 1 of the current pattern.

- `S-Intro = Section{P-Intro, B-Intro, D-Intro)`

  S-Intro is a **MACRO** section which is a collection of the individual sections from piano, bass, and drums

SONG Data Types

- `MyHipJam = Song{Sect-Intro, Sect-V, S-End)`