

```
34 self.debug = debug
35 self.logger = logging.getLogger(__name__)
36
37 if path:
38     self.file = open(os.path.join(path, 'fingerprint.txt'), 'w')
39     self.file.seek(0)
40     self.fingerprints.add(fp)
41
42 @classmethod
43 def from_settings(cls, settings):
44     debug = settings.getboolean('DEBUG')
45     return cls(job_dir=settings.get('JOB_DIR'))
46
47 def request_seen(self, request):
48     fp = self.request_fingerprint(request)
49     if fp in self.fingerprints:
50         return True
51     self.fingerprints.add(fp)
52     if self.file:
53         self.file.write(fp + '\n')
54
```

Midpoint displacement

Midpoint displacement algoritme
(vaak toegepast in procedural content
generation in games)



Van strategie tot algoritme

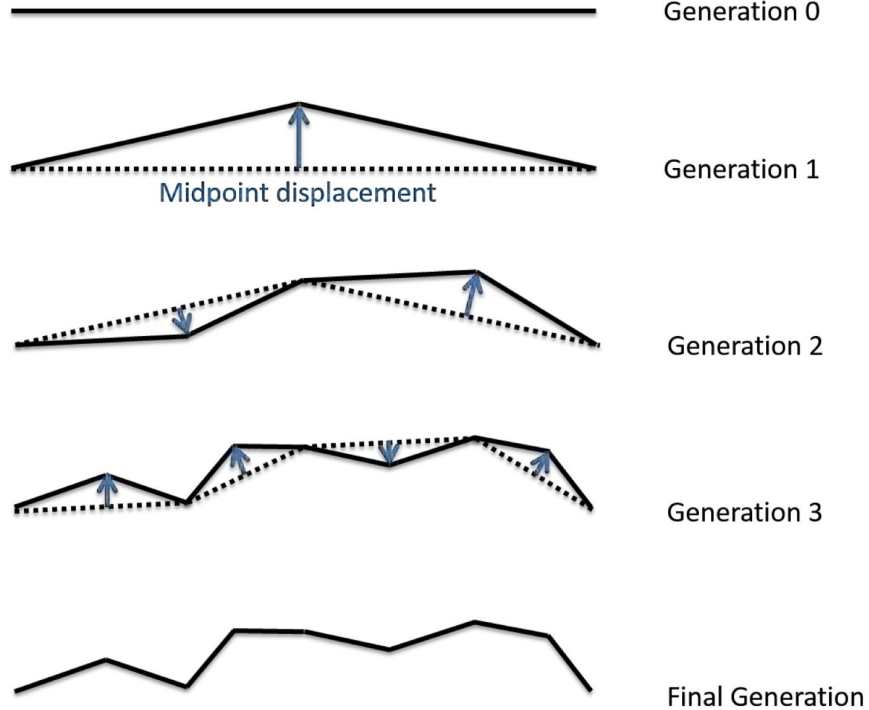


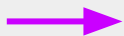
Fig. 4.8 The Midpoint Displacement algorithm visualized.

Midpoint displacement

uit **procedural content generation (PCG)** in games

1. start with a horizontal line.
2. find midpoint of this line
3. move midpoint up / down by a random amount
4. repeat 2 - 4 with lower range of random amount

Midpoint displacement



Applied freely to a sequence of notes

Unit **procedural content generation** (PCG) in games

1. start with a horizontal line.
2. find midpoint of this line / line fragment
3. move midpoint up / down by a random amount
4. repeat 2 - 4 with lower range of random amount

1. start with **two notes**
2. **insert new note between each set of 2 notes**
3. move **pitch of new note** up / down by a random amount
4. repeat 2 - 4 with lower range of random amount

Midpoint displacement



Applied freely to a sequence of notes

uit **procedural content generation** (PCG) in games

1. start with a horizontal line.
2. find midpoint of this line / line fragment
3. move midpoint up / down by a random amount
4. repeat 2 - 4 with lower range of random amount

1. start with **two notes**
2. **insert new note between each set of 2 notes**
3. move **pitch of new note** up / down by a random amount
4. repeat 2 - 4 with lower range of random amount



duration, velocity, ...

Iteration #1

input: 

INS #1:

output:

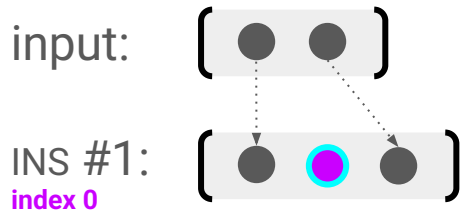
Midpoint displacement

Applied freely to a sequence of notes

1. start with **two notes**
2. **insert new note between each set of 2 notes**
3. move **pitch of new note** up / down by a random amount
4. repeat 2 - 4 with lower range of random amount

Iteration #1

index: 0



output:

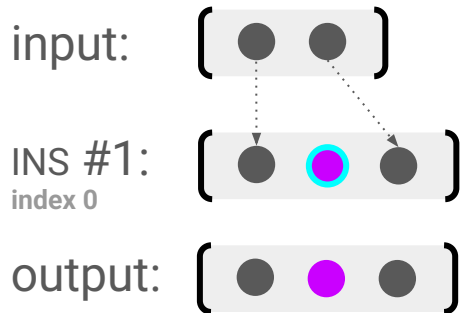
Midpoint displacement

Applied freely to a sequence of notes

1. start with **two notes**
2. **insert new note between each set of 2 notes**
3. move **pitch of new note** up / down by a random amount
4. repeat 2 - 4 with lower range of random amount

Iteration #1

index: 0



Midpoint displacement

Applied freely to a sequence of notes

1. start with **two notes**
2. **insert new note between each set of 2 notes**
3. move **pitch of new note** up / down by a random amount
4. repeat 2 - 4 with lower range of random amount

Iteration #1

index: 0

input: [● ●]

INS #1: [● ● ●]
index 0

output: [● ● ●]

Iteration #2

input: [● ● ●]

INS #1:

INS #2:

output:

Midpoint displacement

Applied freely to a sequence of notes

1. start with **two notes**
2. **insert new note between each set of 2 notes**
3. move **pitch of new note** up / down by a random amount
4. repeat 2 - 4 with lower range of random amount

Iteration #1

index: 0

input: [● ●]

INS #1: [● ●● ●]
index 0

output: [● ●● ●]

Iteration #2

indices: 0, ...

input: [● ● ●]

INS #1: [● ●●● ●]
index 0

INS #2:

output:

Midpoint displacement

Applied freely to a sequence of notes

1. start with **two notes**
2. **insert new note between each set of 2 notes**
3. move **pitch of new note** up / down by a random amount
4. repeat 2 - 4 with lower range of random amount

Iteration #1

index: 0

input: [● ●]

INS #1: [● ●● ●]
index 0

output: [● ●● ●]

Iteration #2

indices: 0, 2

input: [● ● ●]

INS #1: [● ●● ● ●]
index 0

INS #2: [● ●● ● ●●●]
index 2

output:

Midpoint displacement

Applied freely to a sequence of notes

1. start with **two notes**
2. **insert new note between each set of 2 notes**
3. move **pitch of new note** up / down by a random amount
4. repeat 2 - 4 with lower range of random amount

Iteration #1

index: 0

input: [● ●]

INS #1: [● ●● ●]
index 0

output: [● ●● ●]

Iteration #2

indices: 0, 2

input: [● ● ●]

INS #1: [● ●● ● ●]
index 0

INS #2: [● ●● ● ●●●]
index 2

output: [● ●● ●●●●]

Midpoint displacement

Applied freely to a sequence of notes

1. start with **two notes**
2. **insert new note between each set of 2 notes**
3. move **pitch of new note** up / down by a random amount
4. repeat 2 - 4 with lower range of random amount

Iteration #1

index: 0

input: [● ●]

INS #1:
index 0 [● ●● ●]

output: [● ●● ●]

Iteration #2

indices: 0, 2

input: [● ● ●]

INS #1:
index 0 [● ●● ● ●]

INS #2:
index 2 [● ●● ● ●●●]

output: [● ●● ● ●●●]

Iteration #3

input: [● ● ● ● ●]

INS #1:
index 0

INS #2:
index 2

INS #3:
index 4

INS #4:
index 6

output:

Iteration #1

index: 0

input: [● ●]

INS #1: [● ● ●]
index 0

output: [● ● ●]

Iteration #2

indices: 0, 2

input: [● ● ●]

INS #1: [● ● ● ●]
index 0

INS #2: [● ● ● ● ●]
index 2

output: [● ● ● ● ●]

Iteration #3

indices: 0, ...

input: [● ● ● ● ●]

INS #1: [● ● ● ● ● ●]
index 0

INS #2:

INS #3:

INS #4:

output:

Iteration #1

index: 0

input: [● ●]

INS #1:
index 0 [● ●● ●]

output: [● ●● ●]

Iteration #2

indices: 0, 2

input: [● ● ●]

INS #1:
index 0 [● ●● ● ●]

INS #2:
index 2 [● ●● ● ●● ●●]

output: [● ●● ● ●● ●●]

Iteration #3

indices: 0, 2, ...

input: [● ● ● ● ●]

INS #1:
index 0 [● ●● ● ●● ●●]

INS #2:
index 2 [● ●● ● ●● ●● ●●]

INS #3:

INS #4:

output:

Iteration #1

index: 0

input: [● ●]

INS #1:
index 0 [● ●● ●]

output: [● ●● ●]

Iteration #2

indices: 0, 2

input: [● ● ●]

INS #1:
index 0 [● ●● ● ●]

INS #2:
index 2 [● ●● ● ●●]

output: [● ●● ●● ●]

Iteration #3

indices: 0, 2, 4, ...

input: [● ● ● ● ●]

INS #1:
index 0 [● ●● ● ● ●]

INS #2:
index 2 [● ●● ●● ●●]

INS #3:
index 4 [● ●● ●● ●●●]

INS #4:

output:

Iteration #1

index: 0

input: [● ●]

INS #1:
index 0 [● ●● ●]

output: [● ●● ●]

Iteration #2

indices: 0, 2

input: [● ● ●]

INS #1:
index 0 [● ●● ● ●]

INS #2:
index 2 [● ●● ● ●●●]

output: [● ●● ● ●●●]

Iteration #3

indices: 0, 2, 4, 6

input: [● ● ● ● ●]

INS #1:
index 0 [● ●● ● ●● ●]

INS #2:
index 2 [● ●● ● ●●●●]

INS #3:
index 4 [● ●● ● ●●●●●]

INS #4:
index 6 [● ●● ● ●●●●●●]

output:

Iteration #1

index: 0

input: [● ●]

INS #1:
index 0 [● ●● ●]

output: [● ●● ●]

Iteration #2

indices: 0, 2

input: [● ● ●]

INS #1:
index 0 [● ●● ● ●]

INS #2:
index 2 [● ●● ● ●● ●]

output: [● ●● ● ●● ●]

Iteration #3

indices: 0, 2, 4, 6

input: [● ● ● ● ●]

INS #1:
index 0 [● ●● ● ● ● ●]

INS #2:
index 2 [● ●● ● ●● ● ● ●]

INS #3:
index 4 [● ●● ● ●● ● ●● ● ●]

INS #4:
index 6 [● ●● ● ●● ● ●● ● ●● ●]

output: [● ●● ● ●● ● ●● ● ●● ●]

Midpoint displacement

Applied freely to a sequence of notes

How to generate the necessary indices?

Iteration	indices of notes to split	# INS	# INS expressed as power of 2
1			
2			
3			
4			
5			
6			

Iteration #1

index: 0

input: [● ●]

INS #1: [● ●● ●]

output: [● ●● ●]

Midpoint displacement

Applied freely to a sequence of notes

How to generate the necessary indices?

Iteration	indices of notes to split	# INS	# INS expressed as power of 2
1	0	1	2^0
2			
3			
4			
5			
6			

Iteration #1

index: 0

input: [● ●]

INS #1: [● ●● ●]

output: [● ●● ●]

Midpoint displacement

Applied freely to a sequence of notes

How to generate the necessary indices?

Iteration	indices of notes to split	# INS	# INS expressed as power of 2
1	0	1	2^0
2	0, 2	2	2^1
3			
4			
5			
6			

Iteration #2

indices: 0, 2

input: [● ● ●]

INS #1: [● ● ● ●]

INS #2: [● ● ● ● ●]

output: [● ● ● ● ●]

Midpoint displacement

Applied freely to a sequence of notes

How to generate the necessary indices?

Iteration	indices of notes to split	# INS	# INS expressed as power of 2
1	0	1	2^0
2	0, 2	2	2^1
3	0, 2, 4, 6	4	2^2
4			
5			
6			

Iteration #3

indices: 0, 2, 4, 6

input: [● ● ● ● ●]

INS #1: [● ●● ● ● ●]

INS #2: [● ●● ● ●●●]

INS #3: [● ●● ● ●●●●]

INS #4: [● ●● ● ●●●●●]

output: [● ●● ● ●●●●●]

Midpoint displacement

Applied freely to a sequence of notes

How to generate the necessary indices?

Iteration	indices of notes to split	# INS	# INS <i>expressed as power of 2</i>
1	0	1	2^0
2	0, 2	2	2^1
3	0, 2, 4, 6	4	2^2
4	0, 2, 4, 6, 8, 10, 12, 14	8	2^3
5	0, 2, 4, 6, 8, 10, 12, 14, ..., 30	16	2^4
6	...	32	2^5

Midpoint displacement

Applied freely to a sequence of notes

How to generate the necessary indices?

pseudo code



output

```
0
0
2
0
2
4
6
0
2
4
6
8
10
12
```

Iteration	indices of notes to split	# INS	# INS expressed as power of 2
1	0	1	2^0
2	0, 2	2	2^1
3	0, 2, 4, 6	4	2^2
4	0, 2, 4, 6, 8, 10, 12, 14	8	2^3
5	0, 2, 4, 6, 8, 10, 12, 14, ..., 30	16	2^4
6	...	32	2^5

Midpoint displacement

Applied freely to a sequence of notes

How to generate the necessary indices?

Iteration	indices of notes to split	# INS	# INS expressed as power of 2
1	0	1	2^0
2	0, 2	2	2^1
3	0, 2, 4, 6	4	2^2
4	0, 2, 4, 6, 8, 10, 12, 14	8	2^3
5	0, 2, 4, 6, 8, 10, 12, 14, ..., 30	16	2^4
6	...	32	2^5

pseudo code

```
for i in range(num_iterations):  
    num_splits = 2 ^ i  
    for j in range(num_splits):  
        index = j * 2  
        print(index)
```

output

```
0  
0  
2  
0  
2  
4  
6  
0  
2  
4  
6  
8  
...
```

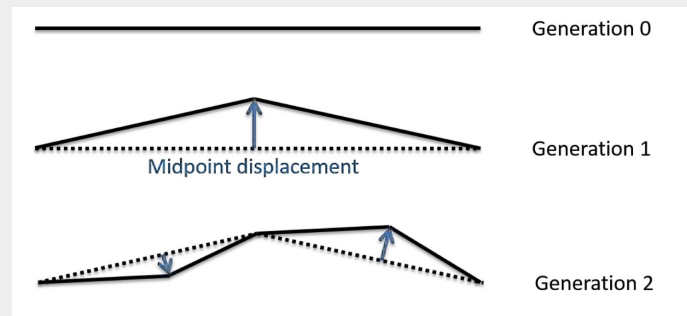

Midpoint displacement

Applied freely to a sequence of notes

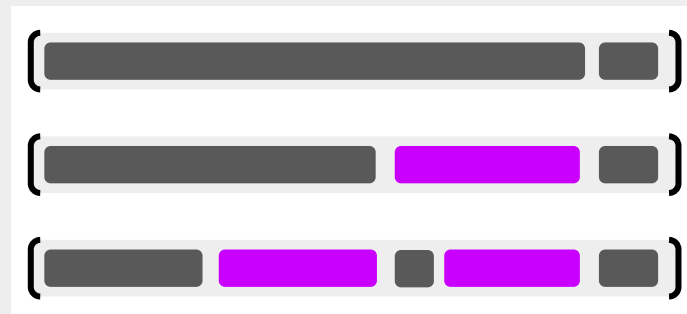
How to **insert** / **split** a note?

1. start with two notes
2. **insert** new note between each set of 2 notes
3. move pitch of new note up / down by a random amount
4. repeat 2 - 4 with lower range of random amount

insert



split



Midpoint displacement

Applied freely to a sequence of notes

How to **split** a note?

Example 1

split_amount = 0.5

input: [[qnote_dur = 4] [qnote_dur = 2]]

split: [[qnote_dur = 2] [qnote_dur = 2] [qnote_dur = 2]]

Example 2

split_amount = 0.25

input: [[qnote_dur = 4] [qnote_dur = 2]]

split: [[1] [3] [2]]

Midpoint displacement

Applied freely to a sequence of notes

How to **split** a note?

Example 1

split_amount = 0.5

input: [[qnote_dur = 4] [qnote_dur = 2]]

split: [[qnote_dur = 2] [qnote_dur = 2] [qnote_dur = 2]]

Example 2

split_amount = 0.25

input: [[qnote_dur = 4] [qnote_dur = 2]]

split: [[1] [3] [2]]

pseudo_code

```
// split amount → divide in half, in a quarter, in an eighth
split_amount = random.choice([0.5, 0.25, 0.125])

// retrieve current note and its duration
note = notes[index]

dur = note["qnote_dur"]

// calculate new note duration and rest value
new_dur = dur * split_amount
rest_dur = dur - new_dur

// store new note duration to current note
note["qnote_dur"] = new_dur

// generate new note with rest duration value
new_note = {"qnote_dur": rest_dur}

// insert new note
notes.insert(new_note, index + 1)
```