Modéliser la structure musicale avec les tuiles hiérarchiques

Alice Rixte, David Jamin

Université de Bordeaux, Bordeaux INP

6 Mai 2024
Journées d'Informatique Musicale

Précédemment, dans les JIM ...

JIM 2023 : «LiveScaler : Contrôler en live l'harmonie d'un morceau de musique électronique»

- **Objectif** : changer l'harmonie à l'aide d'un contrôleur MIDI
- **Solution** : appliquer des transformations affines à des flux MIDI
- **Outils** : Ableton Live et Max MSP
- **Démo** : de la psytrance... C'était amusant !

Transformations rythmiques

Et si on appliquait en live des transformations rythmiques ?

- Inversion du temps
- Passage du binaire au ternaire
- Quantisation
- ...

Pas si facile

- Distinguer en temps et hors-temps
- Délimiter la portée des transformations

Live coding

- En live coding, **transformer des motifs** est bien plus aisé et courant que sur les **stations audionumériques** (DAW).

Live coding

Création musicale en live en utilisant un langage de programmation

- TidalCycles
- Chuck
- ixi lang

Live coding pour la musique électronique de danse

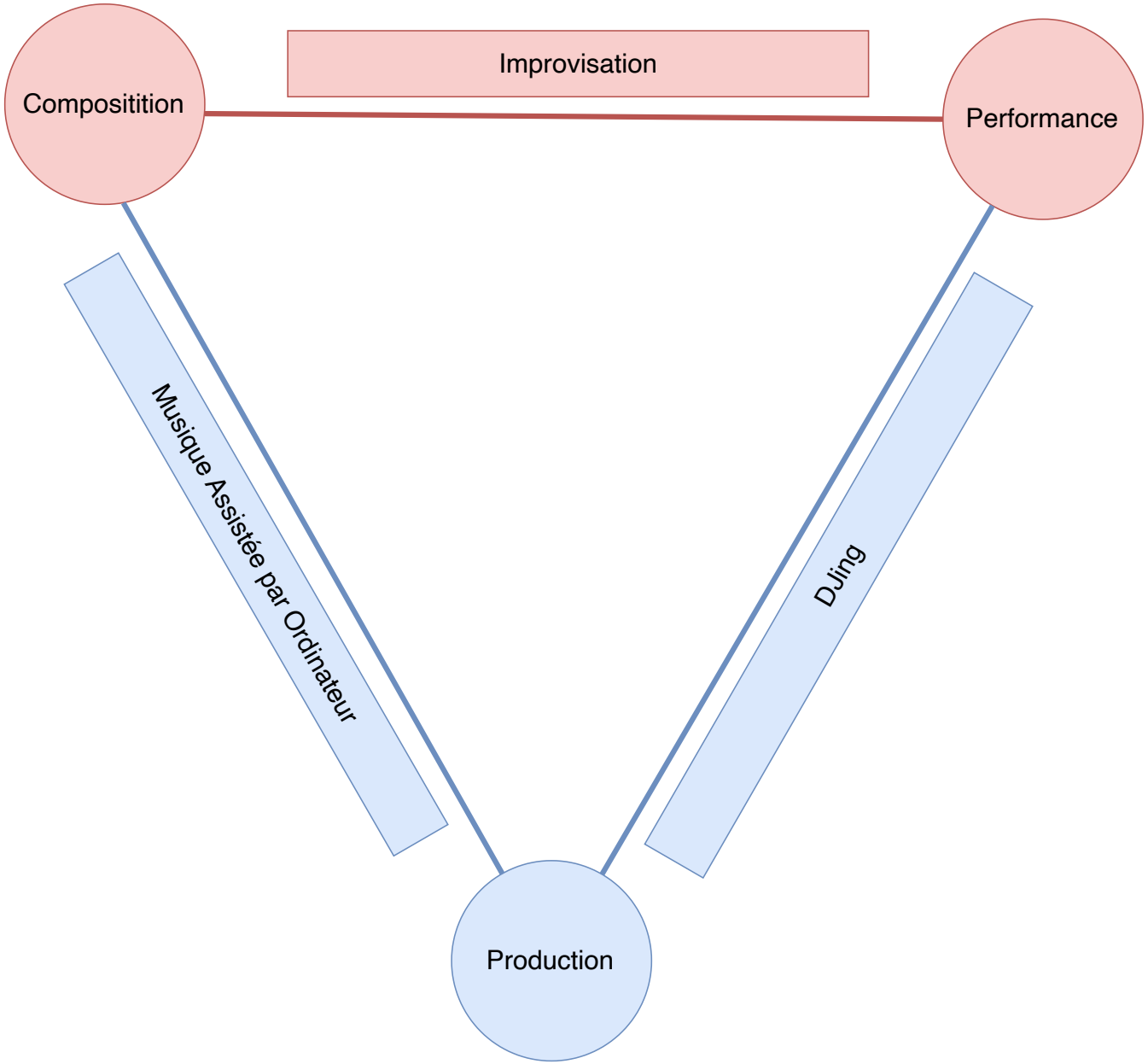
Electronic Dance Music (EDM)

- Terme parapluie recouvrant plusieurs genres musicaux : techno, house, trance, dubstep, trap ...
- Principalement faite pour la danse (boîtes de nuit, fesivals...)

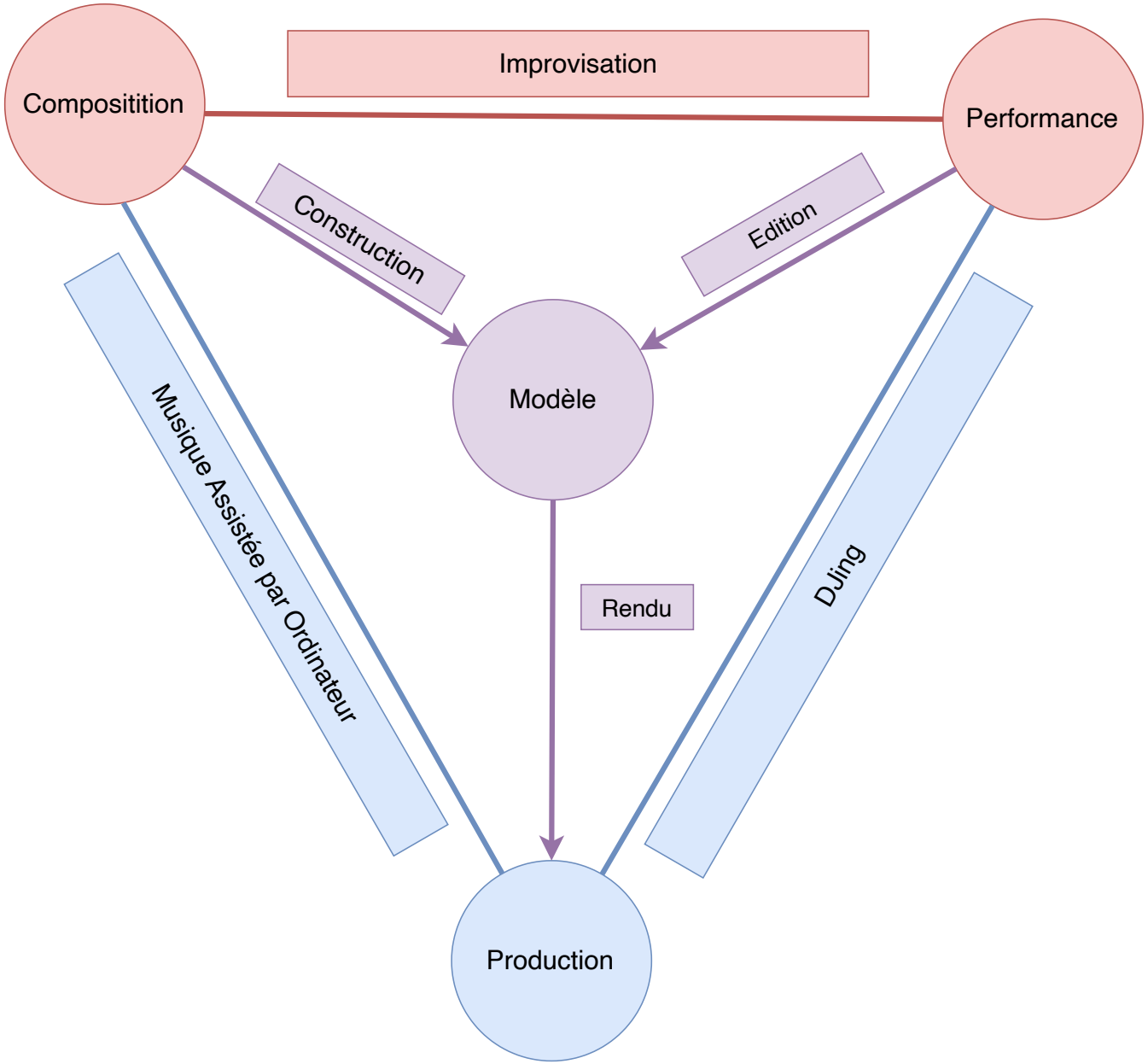
EDM et live coding

- Scène algorave (Alex McLean, Nick Colins ...)
- Reste largement minoritaire par rapport au DJing

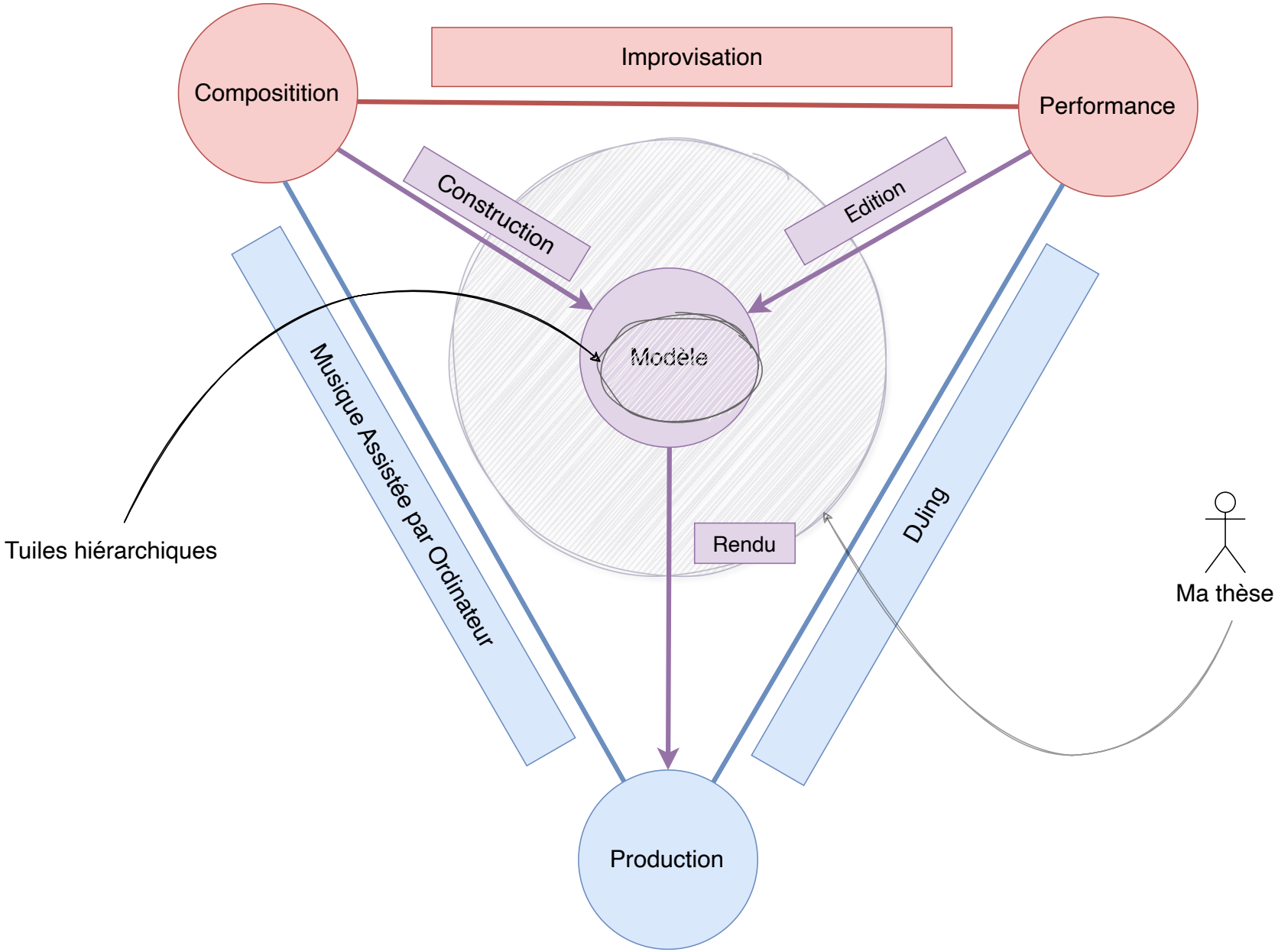
Pratiques musicales en Electronic Dance Music



Architecture d'un langage de live coding



Architecture d'un langage de live coding



Quel modèle utiliser pour un tel langage?

Nous avons besoin d'un modèle :

- Structuré
- Flexible
- Expressif

C'est ce que tentent de faire les tuiles hiérarchiques.

Les tuiles hiérarchique, en une diapo

1. Motifs hiérarchiques

- Des **atomes** placés dans un **espace** peuvent être combiner pour former des **groupes**.
- Les groupes peuvent eux-même être regroupés : c'est une **structure de donnée arborescente**.
- C'est une version algébrique des **hiérarchies de transformations** utilisées en 3D

2.Tuilage

- Le tuilage permet de **changer de repère**
- Base sémantique pour un **langage** de programmation **impératif**

Et maintenant, les tuiles hiérarchiques ...
en 27 diapos

1. Motifs hiérarchiques

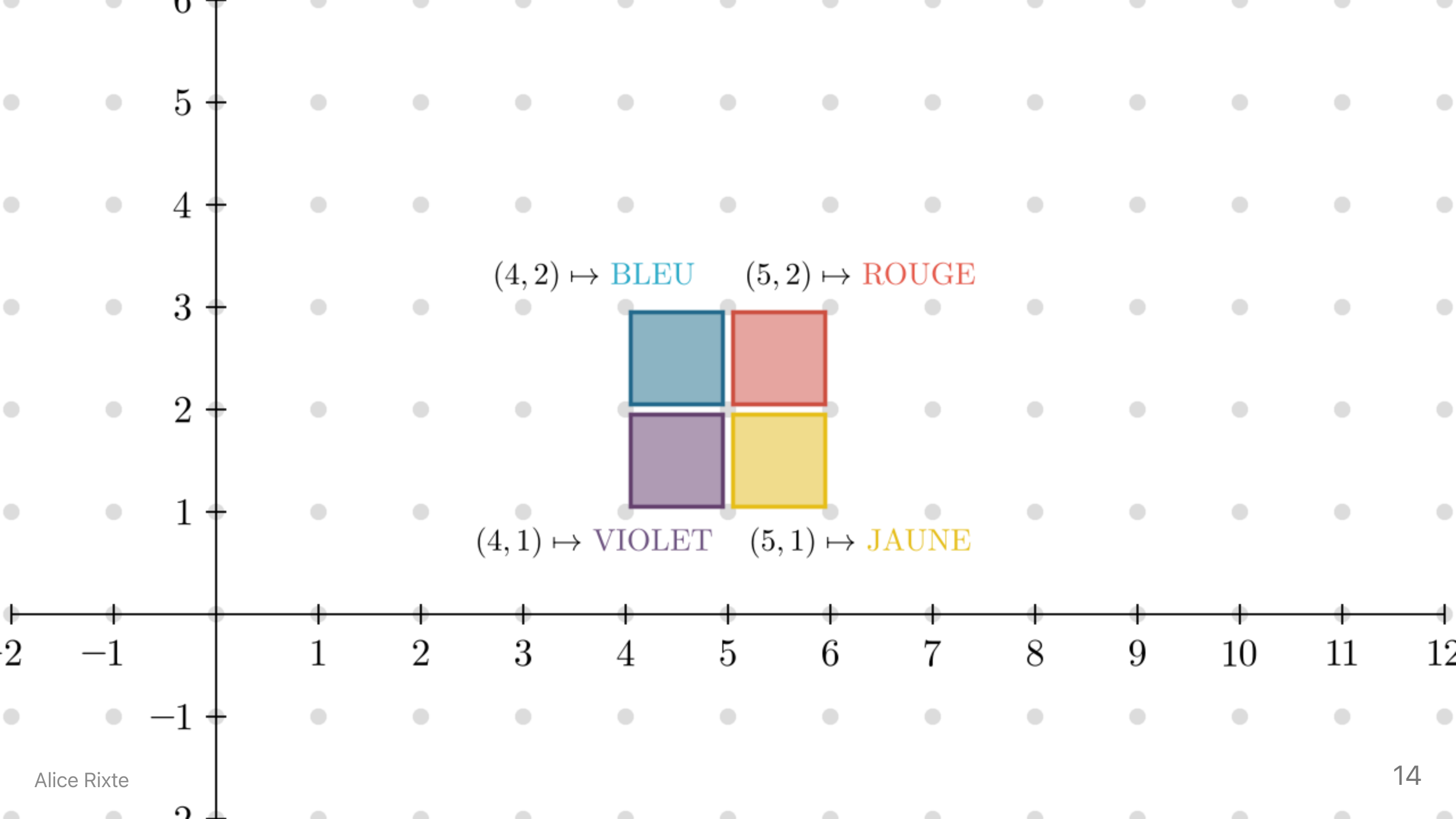
a) Espaces et atomes

b) Hiérarchie et transformations

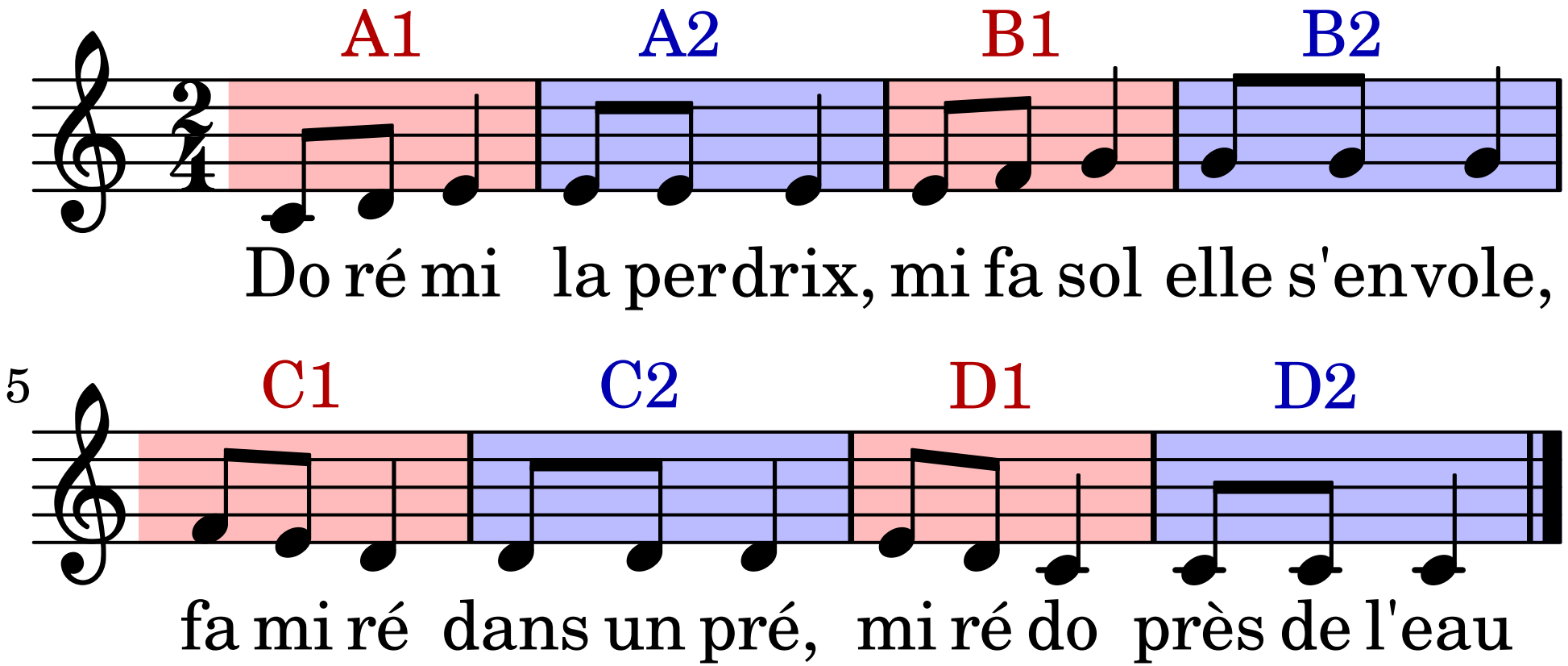
Espace et atomes

Une représentation multimedia

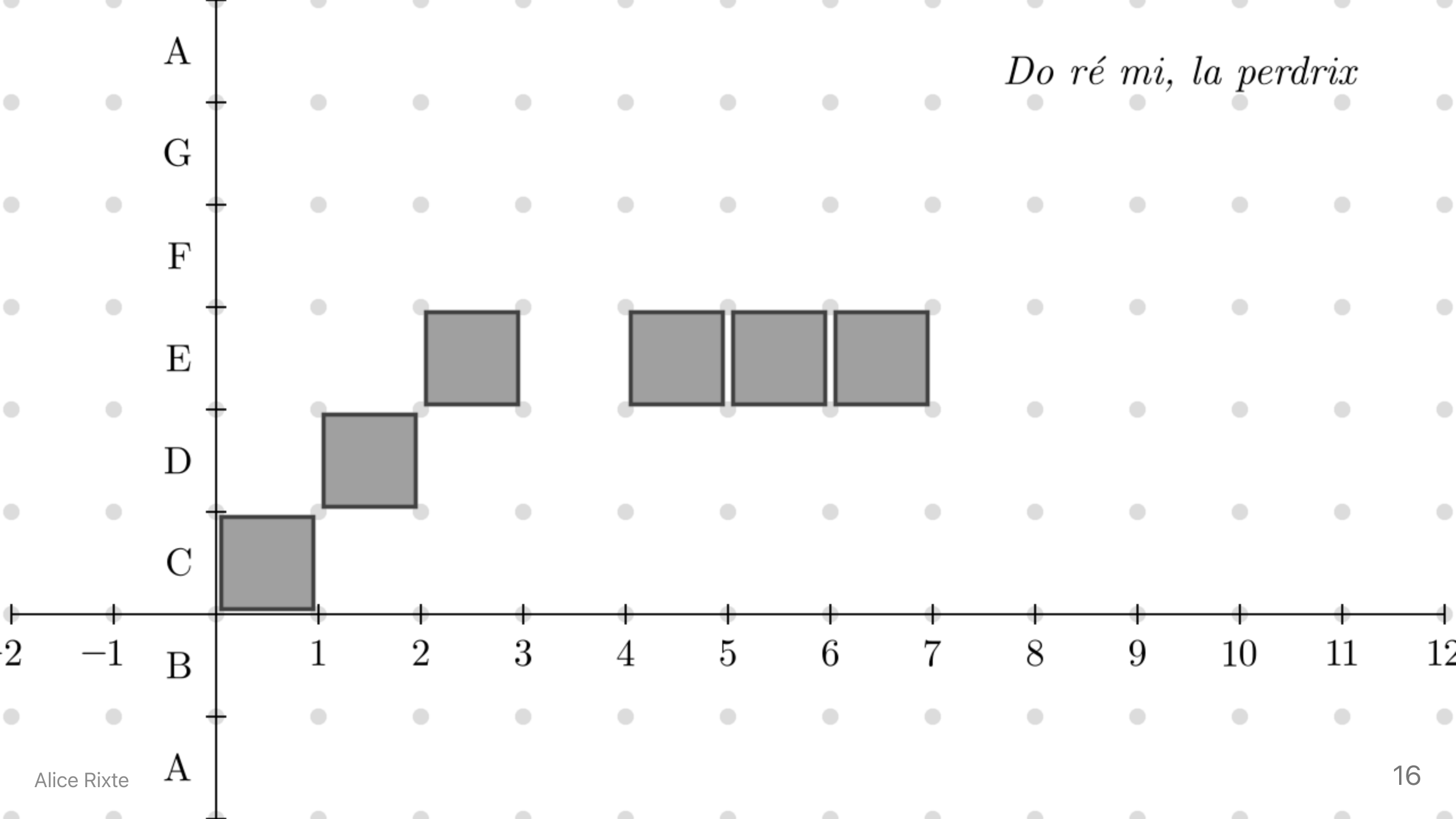
Un objet multimedia peut être représenté comme une liste de valeurs positionnées dans un certain espace


$(4,2) \mapsto$ BLEU
$(5,2) \mapsto$ ROUGE
$(4,1) \mapsto$ VIOLET
$(5,1) \mapsto$ JAUNE

Do ré mi, la perdrix



Do ré mi, la perdrix



Un peu de syntaxe

```
type Duration = Rational
type Time = Rational
type Interval = Int
type Pitch = Int

type Space = (Time,Pitch)

pattA1 :: Media Space c d Duration
pattA1 = do
    atom ( 0 ,0) (1/2) ;   -- Do
    atom (1/2,1) (1/2) ;   -- Re
    atom ( 1 ,2)   1   ;   -- Mi
```

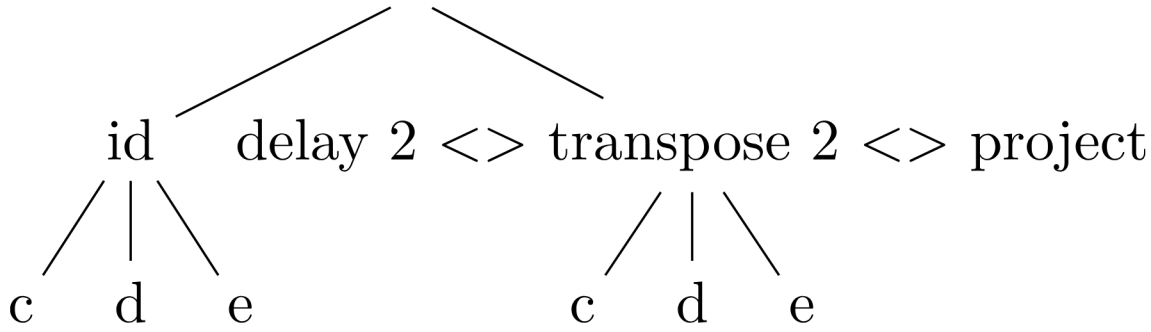
Hiérarchies de transformations

Représentation arborescente

En posant

```
c = atom ( 0 ,0) (1/2) ;   -- Do
d = atom (1/2,1) (1/2) ;   -- Re
e = atom ( 1 ,2)   1   ;   -- Mi
```

Le motif «Do ré mi, la perdrix» être représenté par



Syntaxe

Pour coder un tel arbre, on utilise la syntaxe suivante :

```
pattA = do
  -- do ré mi
  idle                              |> pattA1 ;
  -- la perdrix
  transp 2 <> proj <> delay 2 |> pattA1
```

2. Tuilage

a) Intuition

b) Syntaxe

c) Sémantique

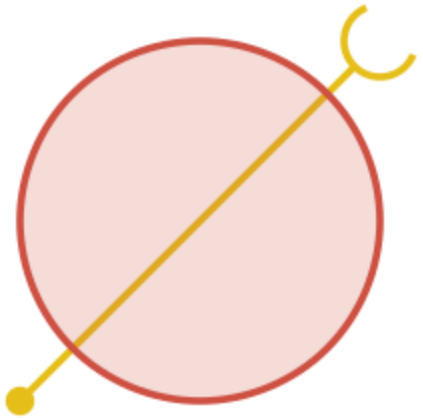
Tuiles

- Combiner motif et changement de repère permet le positionnement relatif

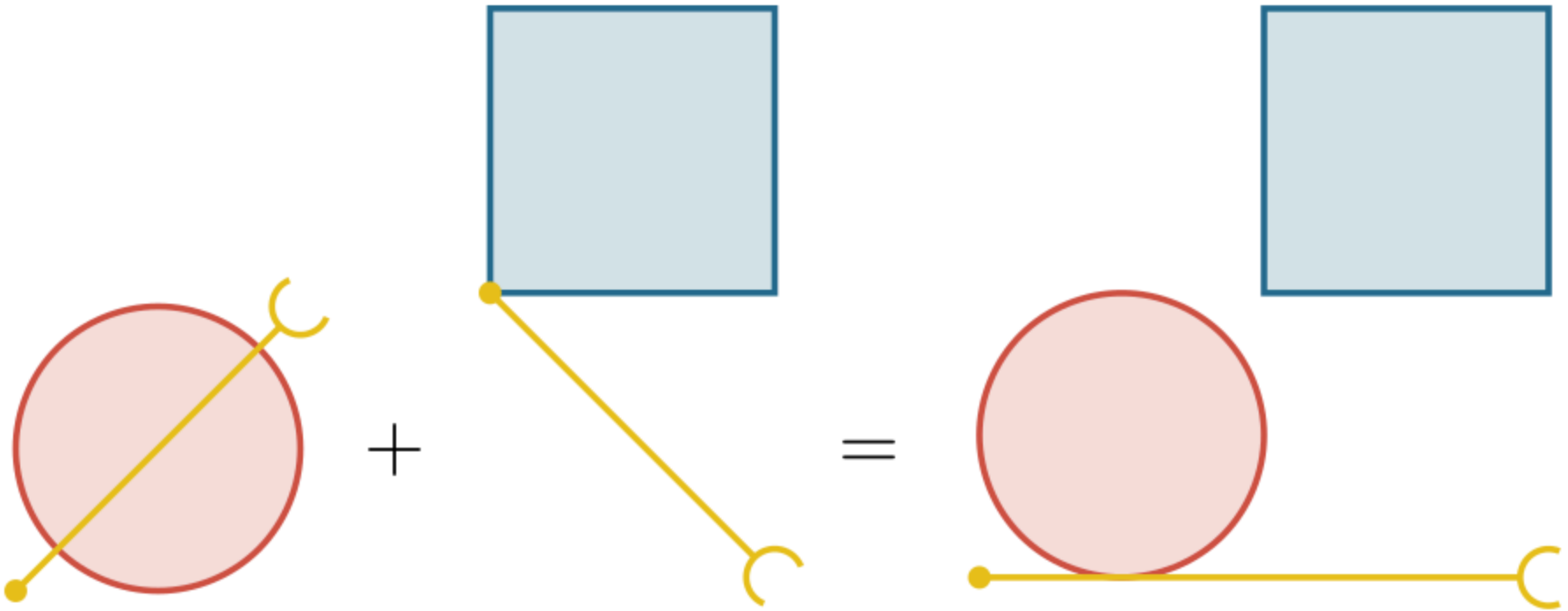
Définition formelle

Une tuile $t = (m, c)$ est une paire composée d'un motif m et d'un changement de repère c.

Ceci est bien une tuile

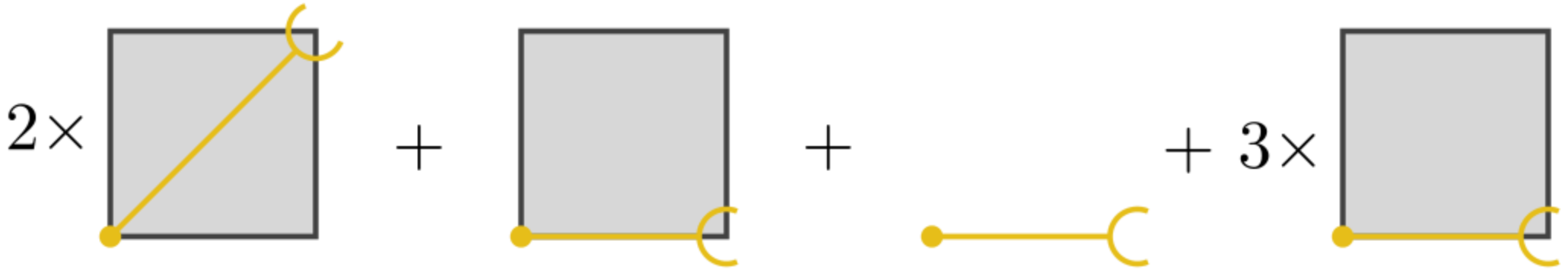


Opération algébrique sur les tuiles

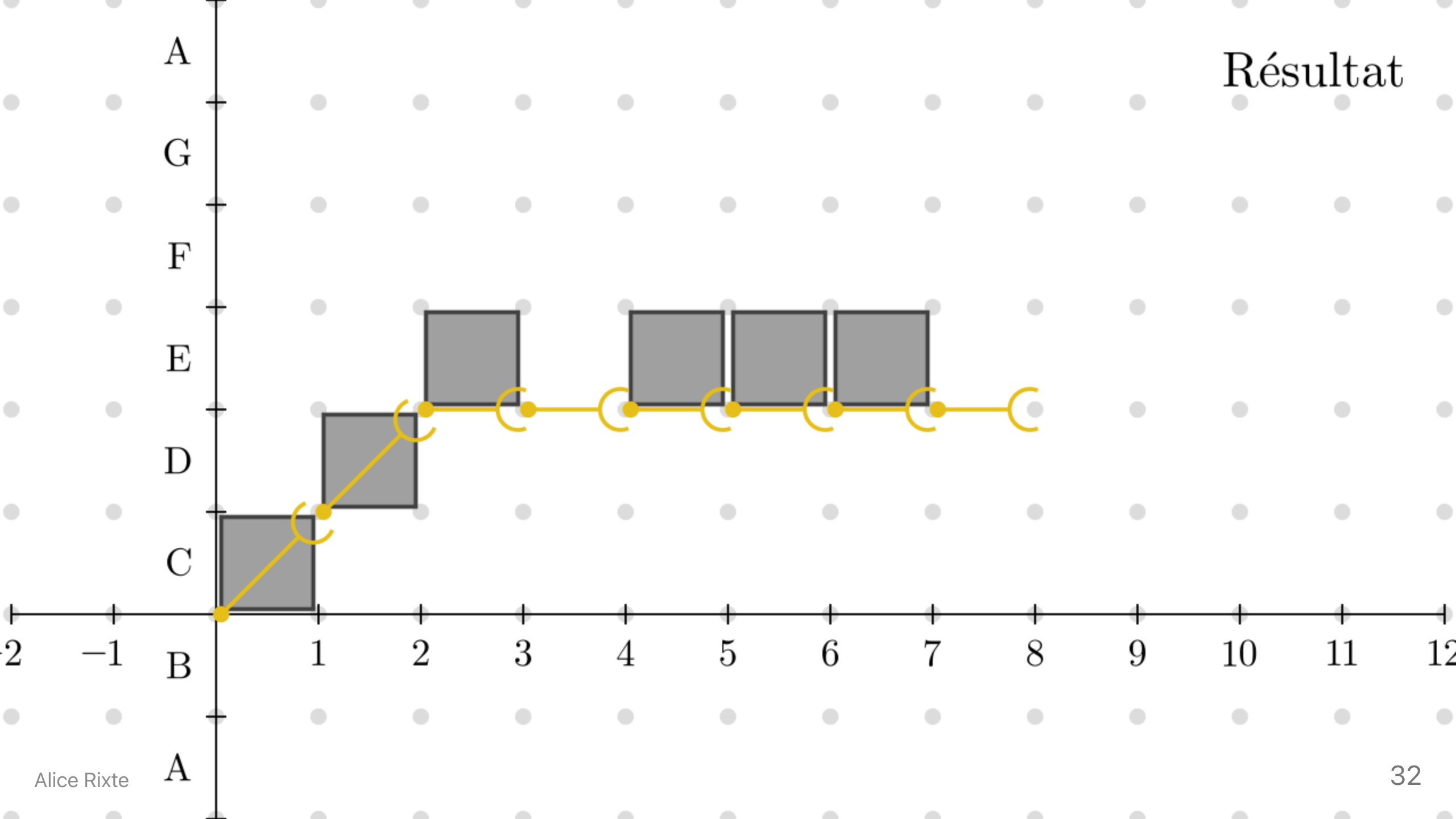


$$(m_1, c_1) \quad + \quad (m_2, c_2) \quad = \quad (m_1 \cdot c_1(m_2), c_2 \circ c_1)$$

Do Ré Mi, La perdrix, la version tuilée



Résultat



Tuiles hiérarchiques

Le même principe de tuilage s'applique lorsque le motif est hiérarchique.

Syntaxe

```
eighth = do
    atom (0,0) (1/2);
    change (delay (1/2))

quarter = do
    atom (0,0) 1;
    change (delay 1)

tileA1 = do
    eighth ; --do
    change (transpose 1) ;
    eighth ; -- re
    change (transpose 1) ;
    quarter -- mi

tileA = do
  idle |-> tileA1
  proj |-> tileA1
```

Do ré mi, la perdrix en entier

```
fullSong = do
      idle |-> tileA;     -- A
      idle |-> tileA;     -- B

      change (transpose (-1));
      inverse |-> tileA   -- C

      change (transpose 1);
      inverse |-> tileA   -- D
```

Conclusion

Les tuiles hiérarchiques sont bien

- **Flexibles** : on peut encoder n'importe quel media avec
- **Structurées** : de part leuraspect hiérarchiques
- **Expressives** :
  - on peut définir les transformations qu'on souhaite
  - les tuiles permettent d'ajouter du sucre syntaxique

L'année prochaine, dans les JIM ...

Un langage de live coding et une démo !

Merci pour votre attention !