

Techniques vidéo-ludiques pour logiciel auteur multimédia

Jean-Michaël Celerier

Laboratoire Bordelais de Recherche en Informatique, Blue Yeti

Problématique

Conception d'un logiciel d'écriture temporelle amené à être utilisé en production par des artistes tout en servant de plate-forme de recherche extensible pour des technologies multimédia.

Partitions interactives

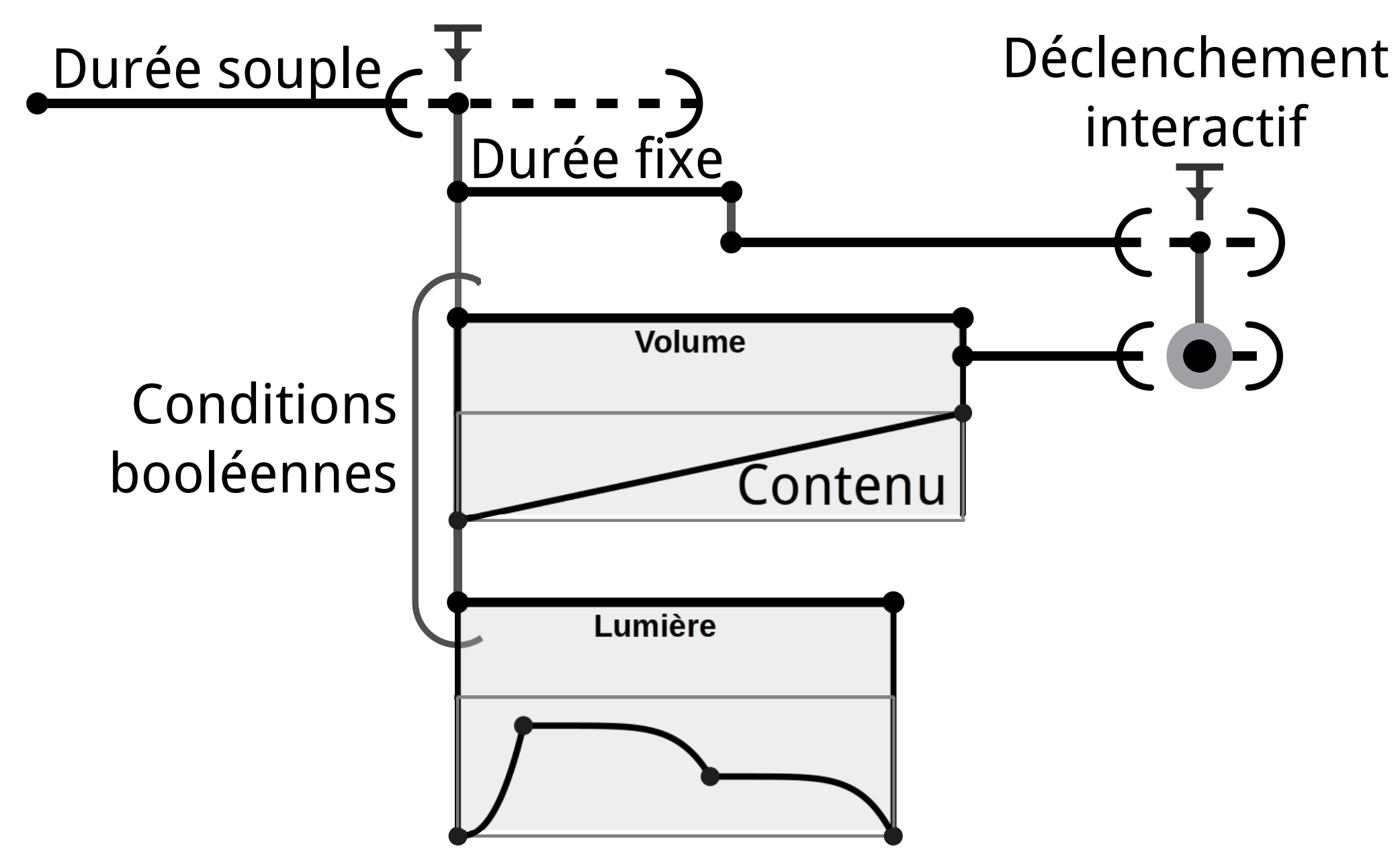


FIGURE 1 : Syntaxe d'une partition interactive

Possibilités d'écritures forment un langage de programmation structuré axé sur l'organisation temporelle. **Boucles** et **hiérarchie**, calcul instantané ou temporel possible via **Javascript**. Applications : musique interactive, scénographie et spectacle vivant, contrôle de robots. Autres approches graphiques : **OpenMusic** [3], **Antescofo** [5], **INscore** [6]; ainsi qu'approches programmatiques : **Abjad**, **Tuiles réactives**, ...

Modèles pour logiciels auteurs

Standard : modèle-vue-contrôleur, modèle-vue-présentateur, document-présentation-instrument, modèle-vue-modèle de vue, présentation-abstraction-contrôle, programmation fonctionnelle réactive. Servent à la **séparation des responsabilités** lors du développement et principalement à la manière dont une interaction utilisateur affecte le modèle de données et du retour sur affichage.

Problématique de l'**édition temps-réel avec contraintes** [7].

Dans jeux-vidéo, modèle courant : entité-composant-système.

- **Entité** : objet ou identifiant trivial, auquel on associe des composants.
- **Composant** : contient des données.
- Une famille commune de composants (rendu, physique, son) est gérée par un même **Système**.

Méthode

Conception en entité-composant-système avec hiérarchies symétriques d'entités et de composants. Plusieurs moteurs opèrent en parallèle, avec une conception modulaire pour étendre le modèle.

Applications

Ce modèle de composants est utilisé pour gérer :

- **L'exécution** d'un scénario : des objets de l'API d'exécution développée indépendamment sont créés et fonctionnent de manière indépendante.
- Des contraintes de déplacement à l'édition : utilisation de **Cassowary** en tant que système.
- **L'affichage** : rendu graphique via la bibliothèque Qt, chaque objet possède un composant de présentation - vue.
- **L'arbre interne** : les objets sont introspectibles par le réseau et en local.
- **Un moteur audio** : un système surveille les processus audio ; on les ajoute et enlève du mixage quand nécessaire.
- **Des optimisations** : on peut partager le moteur d'exécution JS entre les processus JS.

Implémentation

Système d'entités adapté pour **hiérarchie fixée** dans le modèle : tout ne se compose pas avec tout. Identification unique fortement typée avec cache :

- Dans un document par chemins : nécessaire pour gestion undo - redo et identification sur réseau. Pattern Commande réparti pour **édition multi-utilisateurs**.
- À un niveau de hiérarchie donnée : performance pour itération.

Création de familles de composants fortement ou faiblement typés selon le besoin, et associés à un élément de modèle : fig. 3.

Contrairement à moteurs de jeu, **pas de synchro des tick rate** car tous les systèmes sont séparés ; certains systèmes peuvent fonctionner sur le même thread ou sur des threads différents. Les systèmes peuvent communiquer entre eux.

Résultats

Plusieurs moteurs sont implémentés en ECS : l'extensibilité de cette méthode permet d'itérer et de prototyper rapidement même dans un langage à la syntaxe complexe comme C++.

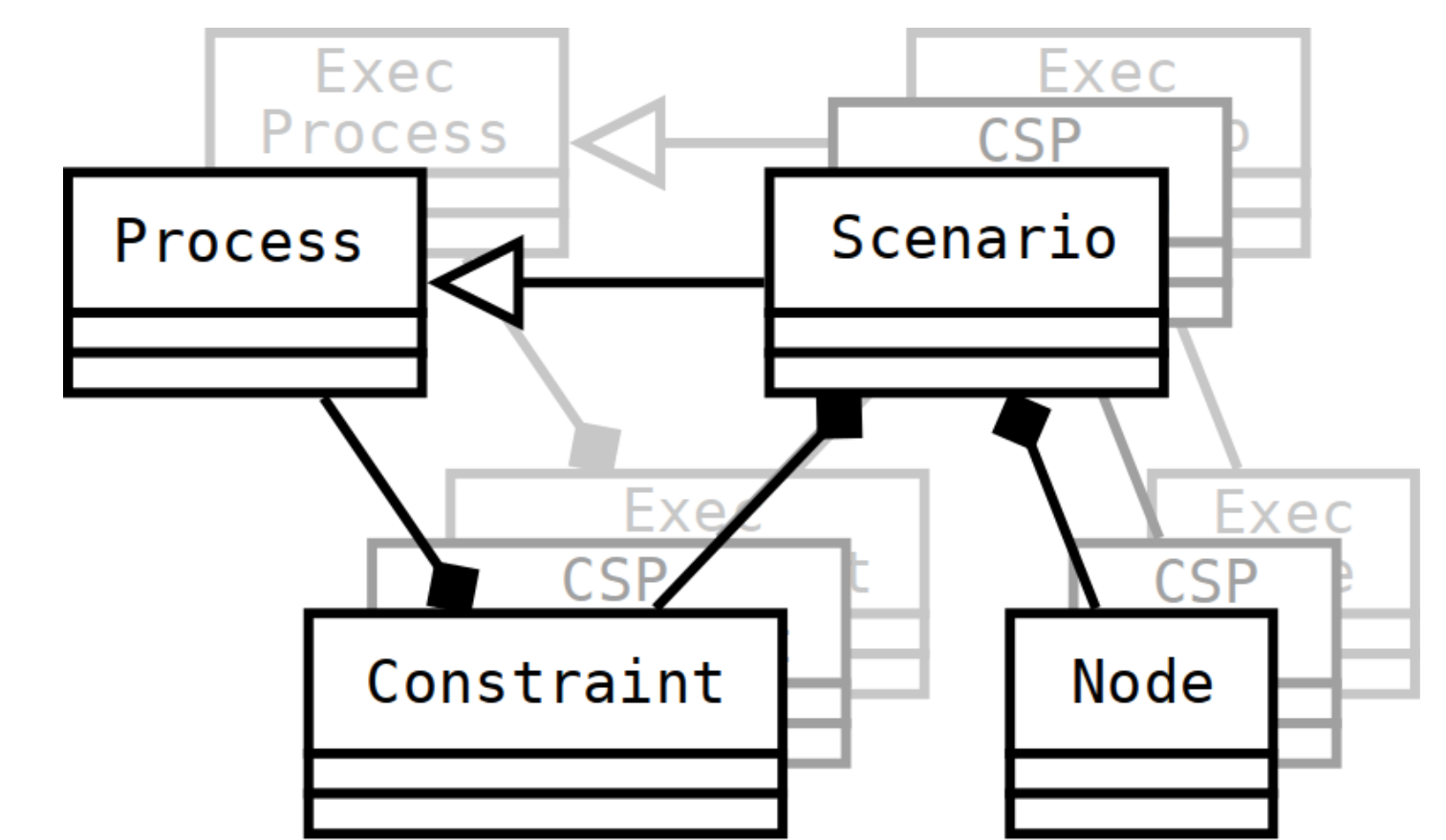


FIGURE 3 : Organisation des graphes d'entités - composants

Prochaines étapes

- Implémentation audio plus poussée via **libaudiostream**[8] qui crée un graphe semblable à celui d'i-score.
- Travail en cours sur modèles et **processus spatiaux** : écriture avec des données multi-dimensionnelles.

Informations complémentaires

Articles sur ce sujet :

- Modèles formels sur lesquels se base i-score [1, 2].
- Paradigme graphique OSSIA : [4].

i-score peut être téléchargé librement sur

- www.i-score.org

Références

- [1] Antoine ALLOMBERT, Gérard ASSAYAG et Myriam DESAINTE-CATHERINE. "A system of interactive scores based on Petri nets". In : *4th Sound and Music Computing Conference (SMC07)*. 2007, p. 158–165. (Visité le 02/11/2015).
- [2] Jaime ARIAS, Myriam DESAINTE-CATHERINE et Camilo RUEDA. "Modelling Data Processing for Interactive Scores Using Coloured Petri Nets". In : IEEE, juin 2014, p. 186–195. ISBN : 9781479942817. DOI : 10.1109/ACSD.2014.23. URL : <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7016342> (visité le 02/11/2015).
- [3] Jean BRESSON, Carlos AGON et Gérard ASSAYAG. "OpenMusic : Visual Programming Environment for Music Composition, Analysis and Research". In : *Proceedings of the 19th ACM International Conference on Multimedia*. MM '11. Scottsdale, Arizona, USA : ACM, 2011, p. 743–746.
- [4] Jean-Michaël CELERIER et al. "OSSIA : Towards a unified interface for scoring time and interaction". In : *TENOR : First International Conference on Technologies for Music Notation and Representation, Paris, France*. 2015. (Visité le 02/11/2015).
- [5] Arshia CONT. "ANTESCOFO : Anticipatory Synchronization and Control of Interactive Parameters in Computer Music." In : *International Computer Music Conference (ICMC)*. 2008, p. 33–40.
- [6] Dominique FOBER, Yann ORLAREY et Stéphane LETZ. "An environment for the design of live music scores". In : *Proceedings of the Linux Audio Conference, CCRMA, Stanford University, California, US*. 2012, p. 47–54.
- [7] Chris LAFFRA et al. *Object-oriented programming for graphics*. Springer Science & Business Media, 2012.
- [8] Stéphane LETZ. "Spécification de l'extension LibAudioStream". In : (2014). URL : <https://hal.archives-ouvertes.fr/hal-00965269/> (visité le 22/07/2015).

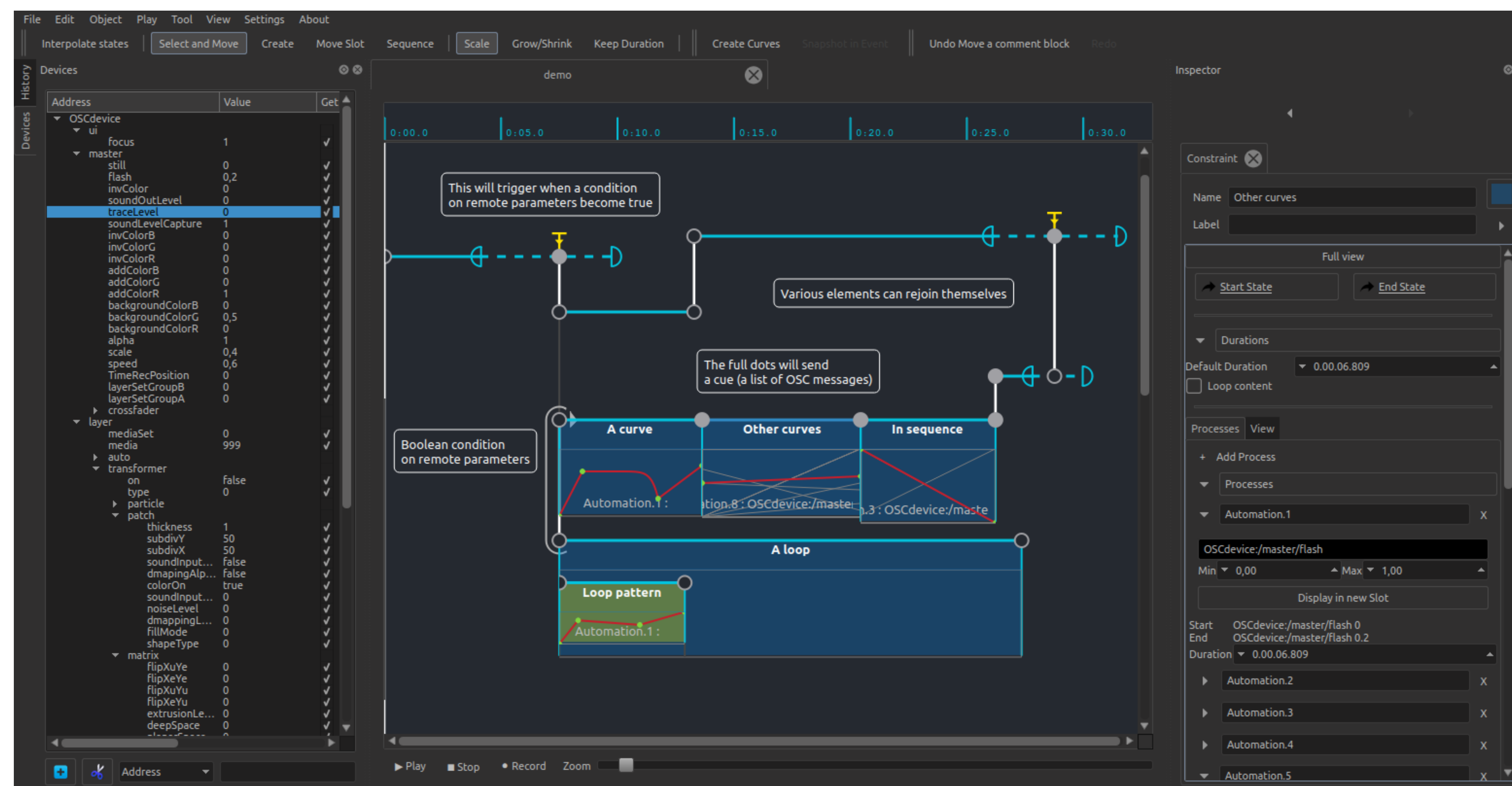


FIGURE 2 : Un scénario d'exemple i-score