PRESENTATION D'UN FORMALISME GRAPHIQUE POUR L'ECRITURE DE SCENARIOS INTERACTIFS

Théo de la Hogue — GMEA — theo.delahogue@gmea.net

Jean-Michaël Celerier — Blue Yeti — jeanmichael@blueyeti.fr

Pascal Baltazar — Compagnie Les Baltazars — pascal@baltazars.org

RÉSUMÉ

Cet article fait état d'un résultat des travaux réalisés dans le cadre du projet Open Scenario System for Interactive Application (OSSIA) dont l'objectif a été d'offrir des outils génériques pour le développement de logiciels d'écriture de scénarios interactifs. Après un rappel du contexte scientifique dans lequel s'est déroulé le projet, les notions du formalisme graphique d'écriture de scénarios interactifs sont présentées puis appliquées à un cas réel d'installation interactive.

1. CONTEXTE DE LA RECHERCHE

1.1. Enjeux et problématiques

Le cœur du projet OSSIA[1] répond à un manque de solutions d'écriture génériques dans l'état de l'art des logiciels créatifs pour la scénarisation interactive [1][2]. En effet, outre leurs intrications forte avec les medias utilisés, les logiciels de scénarisation existants se basent sur l'un ou l'autre des modèles de la *cue-list* et de la *time-line* :

- Le modèle *cue-list* permet de déclencher interactivement des blocs de temps statiques et prédéterminés, avec une grande souplesse dans l'agencement temporel et logique, mais sans la possibilité d'anticiper une cohérence globale du scénario – les différentes *cues* sont des éléments séparés et indépendants.
- Le modèle *time-line* permet de construire avec beaucoup de précision la cohérence globale et les agencements entre divers éléments de scénarisation, mais celle-ci reste statique dans son ensemble, avec pas ou peu de possibilités d'interaction avec un temps externe à celui qui meut l'avancée linéaire de la *time-line*.

De plus, le projet OSSIA a cherché spécifiquement à répondre à une autre dichotomie qui se surajoute à celle-ci, entre les modèles de scénarisation logiques et temporels, qui ne se mêlent généralement pas dans les logiciels de scénarisation existants :

- La scénarisation temporelle est généralement linéaire (sur le modèle du spectacle), ne se joue qu'une fois et toujours selon le même ordonnancement.
- La scénarisation logique (sur le modèle du jeu vidéo) peine à organiser des fragments de déroulement temporels linéaires.

Au commencement du projet le logiciel *i-score* proposait déjà un modèle de scénario interactif linéaire éditable à travers une *time-line*. L'enjeu scientifique du projet OSSIA a consisté à étendre ce modèle existant vers un modèle non-linéaire comportant des boucles et embranchements, à la fois de façon cohérente sur le plan théorique et satisfaisant les exigences des utilisateurs.

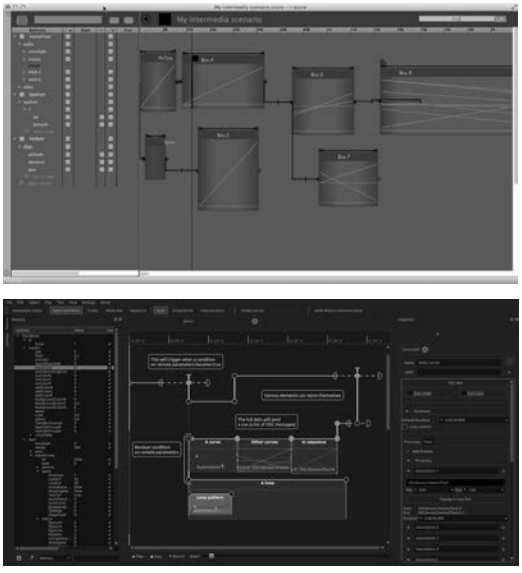

Figure 1. Évolution du logiciel *i-score* au cours du projet OSSIA (version ancienne et actuelle)

L'utilisation de la *time-line* d'*i-score* pour éditer de tels scénarios était une contrainte et aussi une originalité visant à mettre à la portée d'utilisateurs non programmeurs un système d'écriture complexe. La littérature scientifique associée aux domaines des nouvelles formes musicales, et des outils d'authoring multimédia montre qu'il existe des modèles et ontologies[3], des

[1] Le projet OSSIA s'est achevé fin octobre 2015 après 3 ans de collaboration financée par l'Agence Nationale de la Recherche. Le consortium réunissait deux laboratoires en informatique (LaBRI, CEDRIC) et une école (ENJMIN), un Centre National de Création Musicale (GMEA) et les sociétés Blue Yeti et RSF.

outils de programmation visuelle[4][5], et des sémantiques d'exécution pour la musique et la scénarisation interactive. *i-score* se démarque en liant l'ensemble de ces fonctionnalités dans un seul outil d'écriture visuel axé sur une représentation en *time-line,* indépendamment du type de données manipulées à l'inverse d'INScore[6] ou d'Antescofo et Ascographe[7] qui sont orientés pour la manipulation d'objets musicaux. Le formalisme graphique présenté ci-après fait état des résultats d'une recherche menée de manière itérative.

Par ailleurs l'objectif initial du projet était de fournir des outils, à la fois pour les utilisateurs finaux et pour les développeurs de ces outils – sachant que les frontières entre ces deux catégories sont la plupart du temps mouvantes, et que les niveaux de compétence en programmation des utilisateurs/développeurs se distribuent de façon très variées suivant les domaines d'application et les divisions des métiers dans ces domaines. La problématique de la généricité a guidé le projet dans la production de briques logicielles combinables en fonction des contextes et besoins spécifiques qu'ils génèrent. Le projet OSSIA a abouti à la conception d'une API C++[2] qui permet d'implémenter informatiquement l'ensemble des notions du formalisme présenté dans cet article. Cette API sert désormais de base à *i-score* et est en train d'être portée dans d'autres langages (C, C#, bientôt Python).

1.2. Méthodologie

La recherche s'est déroulée en parallèle et en interaction entre une étude formelle, faisant évoluer les modèles scientifiques, et une étude plus pratique sur des questions d'ergonomie. Cette seconde étude concerne la gestion du temps et de la logique dans une seule interface graphique. L'objectif de cette interface est de créer des scénarios complexes tout en restant simple, claire, accessible et basée sur un petit nombre d'éléments de syntaxe.

Cette double étude s'est construite sur des échanges entre utilisateurs, scientifiques et ingénieurs [8]. Ainsi la confrontation constante entre les modèles scientifiques et les besoins des praticiens a permis de s'assurer de l'adéquation des résultats produits aux usages réels, tout en présentant des problématiques scientifiques imprévues.

Les débats concernant la séparation ou non des représentations logique et temporelle sont caractéristiques de cette démarche confrontant le pragmatisme des utilisateurs et la formalisation théorique des scientifiques. De multiples propositions ont été émises dont celle d'avoir des sous-scénarios dédiés à l'écriture logique. La solution choisie finalement a été de fondre les deux représentations. Cela repose sur le besoin qu'ont les utilisateurs de pouvoir penser le temps et la logique sur un même plan et non en créant une hiérarchie supplémentaire qui viendrait embrouiller l'acte d'écriture par une profusion d'éléments graphiques disjoints. Ainsi la représentation graphique que nous présentons dans cet article a en quelque sorte été profilée par des utilisations en situation de production.

[2] https://github.com/OSSIA/API

2. FORMALISME GRAPHIQUE

Le formalisme graphique présenté dans cette partie est celui sur lequel s'appuie désormais *i-score.* Cependant cette proposition est à considérer en dehors du logiciel en cela qu'elle permet une description visuelle de la structure logico-temporelle d'un scénario interactif sur papier.

1.3. Terminologie

Les différents éléments du formalisme pour l'écriture de scénarios interactifs s'organisent en trois catégories : le contenu, le temps et la logique.

1.3.1. Eléments de contenu

Figure 3.

- Un *état* contient un ensemble d'ordres, commandes ou messages à envoyer aux programmes à piloter.
- Un *processus* génère des *états* en fonction du temps. On peut imaginer toutes sortes de processus : une automation, un scénario interactif, un oscillateur, un générateur aléatoire, une fonction de transfert entre un paramètre d'entrée et de sortie, un programme *javascript* … Un *processus* n'a pas de durée intrinsèque.

1.3.2. Eléments temporels

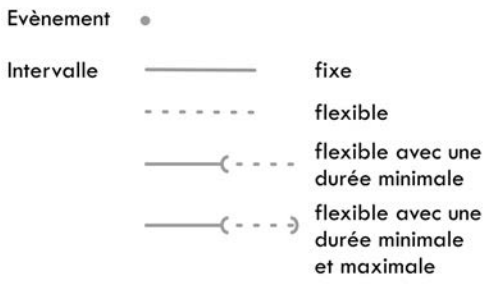

Figure 4.

- Un *évènement* est un point dans le temps. Sa date dépend de la durée et de la nature des *intervalles* qui le précèdent et peut être indéterminée. Sa position dans le scénario est hypothétique et peut ne jamais se produire.
- Un *intervalle* peut être fixe ou flexible. Un *intervalle* flexible peut avoir une durée minimale et/ou une durée maximale représentées par des bornes.

1.3.3. Eléments logiques

Figure 5.

- Une *condition* impose la vérification d'une expression logique impliquant les valeurs de paramètres des programmes à piloter. Cette vérification est réalisée une seule fois et retourne vrai ou faux au moment où elle a lieu.
- Un *déclencheur* impose la vérification d'une expression logique impliquant les valeurs de paramètres des programmes à piloter. Cette vérification est réalisée pendant l'*intervalle* (nécessairement flexible) qui le précède et tant qu'elle n'est pas devenue vraie.
- Un *motif* est un *intervalle* qui a vocation à être répété lorsqu'il se termine.

1.4. Syntaxe

Les règles de syntaxe entre les différents éléments sont aussi organisées selon trois catégories : contenu, temps et logique.

1.4.1. Règles de syntaxe de contenu

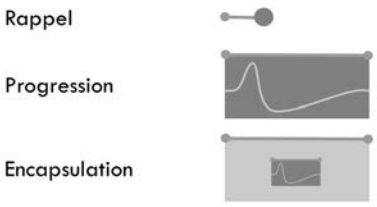

Figure 6.

- Un *rappel*[3] consiste à lier un *état* à un *évènement*. Le contenu de l'*état* est propagé vers les programmes à piloter lorsque l'*évènement* a lieu.
- Une *progression* consiste à lier un *processus* à un *intervalle.*
- Une *encapsulation* consiste à créer une hiérarchie temporelle en liant un *processus* de scénario interactif à un *intervalle* du scénario parent.

1.4.2. Règles de syntaxe temporelles

Figure 7.

- Une *synchronie* consiste à lier plusieurs *évènements* entre eux. Cela signifie que dès qu'un des *évènements* a lieu, tous les autres *évènements* ont lieu aussi et s'il sont soumis à des *conditions* celles-ci doivent être vérifiées.
- Une *séquence* consiste à lier deux *intervalles* successifs par un même *évènement*.

1.4.3. Règles de syntaxe logiques

Figure 8.

- Un *choix* consiste à lier un *évènement* à une *condition*. L'*évènement* n'a lieu que si l'expression logique de la *condition* est vérifiée sinon l'*évènement* est ignoré.
- Une *interaction* consiste à lier un *évènement* à un *déclencheur*. L'*évènement* a lieu lorsque l'expression logique du *déclencheur* devient vraie sinon l'*évènement* est en attente. L'*intervalle* précédant une *interaction* est nécessairement flexible et son *évènement* de fin (celui lié au *déclencheur*) aura lieu à l'échéance de sa durée maximale s'il y en a une. Il est possible de déclarer un *évènement* de fin alternatif (sur la borne de durée maximale) qui a lieu lorsque la durée maximale est atteinte. Dans ce cas l'*évènement* lié au *déclencheur* n'a pas lieu.
- Une *boucle* est un *processus* qui consiste à répéter un *motif* dans les limites d'un *intervalle*. Une boucle définit deux *évènements* liés au début et à la fin du *motif* : lorsque l'*évènement* de fin a lieu, l'*évènement* de début est de nouveau évalué. En combinant *choix* et *interaction* avec les *évènements* du *motif* ou de l'*intervalle* de la *boucle*, il est possible d'exprimer toutes sortes de répétitions.

[3] Le terme de *rappel* est emprunté au vocabulaire de la régie du spectacle vivant et notamment au métier de pupitreur lumière où il est courant de parler du *rappel* d'une mémoire de paramètre.

1.5. Exemple

Le formalisme OSSIA est ici utilisé pour décrire le comportement d'une installation réalisée par Blue Yeti[4] pour le Futuroscope : un jeu de course impliquant un visiteur à la fois. Si cet exemple ludico-sportif ne met pas l'accent sur une application musicale, il propose néanmoins un panel de situations d'interaction propice à illustrer l'ensemble des notions du formalisme.[5]

Figure 8. Photographie du projet Futuroscope Arena réalisé par Blue Yeti

Le dispositif mesure le temps réalisé par chaque visiteur pour parcourir une piste d'une vingtaine de mètres grâce à des capteurs positionnés au début et à la fin de la piste. Un bouton permet d'indiquer que l'on souhaite prendre part à une course. Un écran placé à la fin de la piste donne le top départ et rend compte du résultat de la course et des meilleurs temps. Le dispositif peut détecter un cas de triche où quelqu'un placé en fin de parcours déclencherait le capteur dans un temps aberrant. Le dispositif est aussi connecté au *back-end* du muséum et l'absence de connexion produit un message d'erreur à l'écran proposant de joindre un technicien.

Ce dispositif met en œuvre un patch Max qui prend en charge l'image à l'écran, les capteurs et le bouton. L'ensemble des paramètres, messages et retours d'informations du patch sont adressables et/ou observables via le protocole OSC et sont organisées selon le *namespace* suivant :

/connexion : état de la connexion au *back-end*
/display
  /init : affichage de l'écran de démarrage
  /error : affichage de l'écran d'erreur
  /run : affichage de l'écran de course
  /failed : affichage de course non terminée
  /cheated : affichage de course trichée
  /time : affichage du temps de la course
  /scores : affichage des meilleurs scores
/button : démarrage d'une nouvelle course
/sensor.start : détection du départ
/sensor.end : détection de l'arrivée

Dès lors nous avons tout le nécessaire pour formaliser le comportement temporel et logique du dispositif. Celui-ci s'organise selon quatre niveaux de hiérarchie temporelle.

Figure 9. Scénario du dispositif

Le premier niveau consiste en la répétition un nombre indéfini de fois du scénario interactif de la routine principale dont la durée est inconnue. Ce fonctionnement en boucle infinie est typique des installations muséographiques à la différence du scénario d'un spectacle où le temps ne s'écoule qu'une seule fois. Le formalisme OSSIA permet d'appréhender les deux situations.

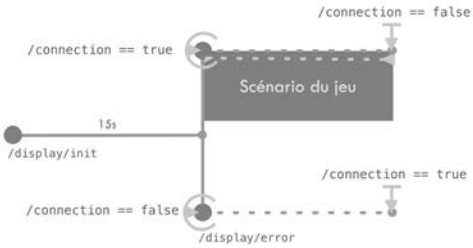

Figure 10. Routine principale

La routine principale s'occupe d'ordonner l'affichage de l'écran d'initialisation et, après un *intervalle* fixe de 15 secondes, vérifie si la connexion au *back-end* est établie. Les deux *choix*, dont les expressions logiques sont ici mutuellement exclusives, définissent deux alternatives possibles. Si la connexion est établie, le scénario interactif du jeu est exécuté en boucle tant que la connexion est valide. Sinon, l'affichage de l'écran d'erreur est ordonné et la routine principale attend le rétablissement de la connexion. Dans les deux cas, lorsque le *déclencheur* est activé, le scénario de la routine se termine et, comme il est encapsulé dans une boucle infinie, il redémarre.

Figure 11. Scénario du jeu

Le scénario du jeu consiste à afficher la liste des

[4] http://www.blueyeti.fr

[5] En parallèle des JIM 2016 plusieurs œuvres et démonstrateurs conçus avec *i-score* illustrent la diversité des médias que le formalisme peut appréhender : Nebula (Compagnie Les Baltazars, Larsen (Renaud Rubiano), INALAYA (Blue Yeti) et une pièce musicale interactive développée par le Scrime et le GMEA.

scores en attendant que quelqu'un presse le bouton pour participer à une course. L'*intervalle* de temps d'une course correspond à un scénario interactif dont la durée est indéterminée mais qui ne s'exécute qu'une fois. Après la course, le scénario du jeu attend 5 secondes et, comme il est encapsulé dans une boucle infinie, il redémarre.

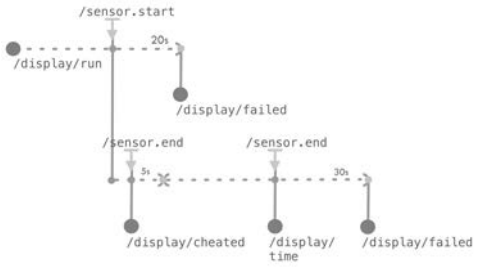

Figure 12. Scénario d'une course

Le scénario d'une course commence par afficher l'écran de course et attend 20 secondes que le capteur placé au début de la piste détecte la présence de quelqu'un. Après 20 secondes, l'écran d'échec est affiché et le scénario de la course se termine. Si le capteur de départ est déclenché, le scénario vérifie que le capteur de fin n'est pas déclenché pendant un *intervalle* de 5 secondes. Si c'est le cas, l'écran de triche est affiché. Sinon un *intervalle* de 30 secondes attend le déclenchement du capteur de fin. Si le capteur est déclenché dans le temps imparti on accède à l'écran d'affichage du temps effectué. Sinon l'écran d'échec est affiché. Dans tout les cas, dès que l'affichage d'un écran est sélectionné le scénario se termine.

3. CONCLUSION

Le formalisme OSSIA possède plusieurs propriétés importantes qui encouragent à le comparer avec d'autres langages informatiques [9]. La possibilité de combiner plusieurs éléments selon des règles simples et sans exceptions aboutit à un langage modulaire et cohérent. Le fait que ce soit un formalisme graphique offre une bonne lisibilité et assure une simplicité dans la maintenance du code. Cependant OSSIA n'est pas un langage expressif car il est spécialisé pour l'écriture de scénario logico-temporel et semble peu enclin à être détourné dans son utilisation. Il est par contre intéressant de s'interroger sur la capacité d'abstraction qu'il serait possible d'offrir dans des versions futures du langage en autorisant par exemple l'écriture de « patrons » temporels réutilisables dans d'autres projets et organisables en bibliothèques de structures propres à certains domaines d'applications. Enfin, sa résistance aux changements (ou capacité de viscosité [9]) est variable selon que les changements soient de nature temporelle ou logique. La représentation en *time-line* assure une cohérence visuelle qui fait comprendre immédiatement les conséquences de la modification d'un élément temporel. Par contre, les conséquences de la modification d'un élément logique nécessitent une phase de compilation pour relever des incohérences.

L'implémentation principale du formalisme OSSIA est incarnée par le logiciel *i-score* en permettant l'édition et l'exécution de scénarios interactifs. Cet environnement de travail conçu parallèlement au formalisme soulève beaucoup de problèmes d'ergonomie notamment parce qu'il faut envisager son utilisation en relation avec les programmes à piloter. D'autres questions émergent aussi selon la nature des paramètres à contrôler et en particuliers lorsqu'il s'agit de paramètres spatiaux (édition de trajectoire, détection de collision, analyse de distance ou de recouvrement, …).

Cependant l'existence d'une bibliothèque C++ multiplateforme et *open-source* font espérer la constitution d'une communauté de réflexions et de développements partagés pour intégrer OSSIA dans d'autres environnements logiciels (Max, Unity, Raspberry Pi) agglomérant ainsi des approches expertes et variées dans le domaine de scénarisation des interactions.

4. REFERENCES

[1] Baltazar, P., Allombert, A., Marczak, R., Couturier, J-M., Roy, M., Sèdes, A., Desainte-Catherine, M. « Virage : Une réflexion pluridisciplinaire autour du temps dans la création numérique », Actes des 14e Journées d'Informatique Musicale, Grenoble, 2009.

[2] de la Hogue, T., Desainte-Catherine, M., Chao, J. « OSSIA : Open Scenario System for Interactive Application », Actes des 19e Journées d'Informatiques Musicales, Bourges, 2014.

[3] Raimond, Y., Abdallah, S. A., Sandler, M. B., & Giasson, F. (2007, September). The Music Ontology. In *ISMIR* (pp. 417-422).

[4] Hirzalla, N., Falchuk, B., Karmouch, A. « A Temporal Model for Interactive Multimedia Scenarios », IEEE Multimedia, Vol.2 No.3, pp.24-31, Automne 1995.

[5] Bulterman, D., Hardman, L. « Structured Multimedia Authoring », ACM Transactions on Multimedia Computing, 2005.

[6] Fober, D., Orlarey, Y., & Letz, S. (2014). Augmented Interactive Scores for Music Creation. In Korean Electro-Acoustic Music Society's 2014 Annual Conference (p. 85–91).

[7] Echeveste, J-M. Un langage de programmation pour composer l'interaction musicale: la gestion du temps et des événements dans Antescofo. Thèse. Université Pierre et Marie Curie-Paris VI, 2015.

[8] Meyssonier T. « Analyse des relations entre création artistique, modélisation mathématique et implémentation logicielle dans la problématique de l'écriture de structures temporelles », Stage de recherche effectué au LaBRI sous la direction de Desainte-Catherine M.

[9] Orlarey, Y. « Entre Calcul Programmation et Création », GRAME, 2009.