Chapitre 7

Modéliser le parcours du joueur

Sommaire

7.1 Objectif

Dans ce chapitre, notre objectif consiste à fournir un langage graphique de qui permette de définir la suite de challenges que propose un gameplay, ainsi que d'en calculer l'état lorsqu'un joueur progresse dans ce gameplay. Notre mesure de difficulté repose en effet sur le découpage d'un gameplay en challenges, que doit effectuer le game designer. Un tel langage nous permet tout d'abord de définir une interface graphique qui permette au game designer de définir plus aisément la suite de challenges et de la manipuler pour étudier les différents calculs de difficulté réalisés. Ensuite, le calcul de l'état des challenges nous permet de connaitre, étant donné les challenges déjà terminés par le joueur, les prochain challenges qui lui seront proposés, et d'alimenter ainsi l'algorithme chargé de tracer la courbe de difficulté du jeu présenté section **??**.

Nous avons définit dans le chapitre **??** la notion de *challenge*, qui correspond à un objectif que le joueur doit atteindre. Dans un jeu vidéo, les challenges sont présentés au joueur selon un enchainement pré-établit par le **game designer** [1] . La logique de cet enchainement permet de proposer des challenges adaptés, qui tiennent compte de l'apprentissage du joueur et du rythme que le game designer veut donner au gameplay. La courbe de difficulté d'un jeu vidéo dépend donc de cet enchainement : la suite d'objectifs que le joueur essaie d'atteindre détermine les niveaux de difficulté successifs auxquels il est confronté. Nous appellerons *scénario* la suite de challenges auxquels le joueur est confronté.

Nous employons ici le mot scénario avec une intention bien précise. Nous considérons que la plupart des jeux sont construits comme un enchainement de challenges, d'objectifs que le joueur essaie d'atteindre. Durant chaque challenge, le joueur tente d'atteindre l'objectif en interagissant avec le monde virtuel, grâce aux différentes actions autorisées par les règles du jeu. A l'issue du challenge, le scénario du jeu permet de déterminer l'ensemble de challenges auquel le joueur pourra être ensuite confronté. Le scénario du jeu est donc pour nous la logique d'enchainement des challenges, particulièrement pertinente dans le cadre d'une mesure de la difficulté. Cette logique est présente dans le jeu sous différents aspects. La construction de l'univers du jeu, c'est-à-dire l'agencement des différents objets de l'univers est une part de cette logique : de nombreux jeux rythment la suite de challenges par un système de porte et de clefs par exemple. Cette logique est également présente dans les scripts du jeu, qui permettent de définir le comportement des objets du jeu en fonction de la progression du joueur par exemple. Nous souhaitons représenter cette logique de manière synthétique, dans un modèle graphique.

Il apparait en effet particulièrement pertinent d'intégrer une représentation graphique du scénario d'un jeu vidéo à un outil de mesure de la courbe de difficulté. A l'issue des différents calculs réalisés par le modèle de difficulté (chap. **??**), chaque challenge sera annoté de diverses mesures de difficulté, et le designer pourra naviguer dans son scénario pour exploiter ces informations. Une représentation graphique du scénario facilite cette navigation, en offrant une vue d'ensemble plus facilement manipulable que sous forme purement textuelle. De plus, le scénario d'un jeu structure son déroulement temporel, et un tel déroulement se satisfait particulièrement d'une représentation graphique en deux dimensions.

Il est important de noter que notre modèle de scénario constitue une représentation *supplémentaire* de la logique du jeu. Il ne s'agit pas de remplacer les outils de scripting visuels déjà fournis par les moteurs de jeu, comme Kismet pour le moteur Unreal (Epic) ou le Flow Graph du CryENGINE (Crytek), mais de compléter ces systèmes en offrant une vue particulière de la logique de progression du joueur. Une vue synthétique de l'enchainement des challenges permet de considérer le rythme du gameplay, et en particulier sa difficulté, au niveau de détail le plus adapté, et sans ajouter à la complexité visuelle déjà importante

1. Le game designer concoit les règles du jeu, il a une vision générale du gameplay, qui sera instancié, contextualisé par le level designer. Une section spéciale lui est consacré dans la partie Vocabulaire.

des scripts visuels utilisés pour tout le reste du level design.

La première version de ce modèle a d'abord été mis au point au cours du projet DEEP [2] [Bosser 07], et constitue une extension de travaux précédemment réalisés au sein de CEDRIC pour modéliser le scénario d'un jeu vidéo [Vega 04]. L'objectif de ce projet a consisté à permettre aux personnages non joueurs d'un jeu vidéo, les **PNJ** [3] , de bénéficier d'une liberté d'improvisation. Cette liberté doit cependant rester contrainte au déroulement du jeu tel que prévu par les designers, ce dont notre modèle de scénario avait la charge. Nous avons adapté ce modèle à la problématique particulière du découpage d'un gameplay en une suite de challenges.

Ce chapitre est organisé de la façon suivante. Tout d'abord, nous reviendrons sur le principe de construction de la difficulté d'un jeu vidéo, et sur l'intérêt d'un modélisation des différents challenges et du scénario d'un jeu. Ensuite, nous définirons plus formellement le modèle de scénario, ainsi que le calcul de son état.

7.2 Challenges et construction de la difficulté

Comme nous l'avons montré dans le chapitre **??**, dans un jeu vidéo, le joueur poursuit en permanence un objectif. Nous avons définit la notion de challenge, qui encode la relation entre le joueur et un objectif, au travers de ses différents états. Pour construire la courbe de difficulté d'un jeu vidéo, c'est à dire les différents niveaux de difficulté auxquels le joueur va être confronté, le game designer va définir un ensemble de challenges ainsi que la logique avec laquelle ces challenges seront présentés au joueur, c'est à dire le scénario du jeu.

Le scénario d'un jeu permet de tenir compte de l'évolution du niveau du joueur. En effet, lorsque le joueur se confronte à un challenge, il développe ses capacités. Il découvre de nouvelles suites d'actions qui lui permettent de se rapprocher d'un objectif particulier, et sa pratique lui permet de réaliser ces suites d'actions plus rapidement, plus précisément. Lorsqu'un challenge est terminé par le joueur, ses capacités sont modifiées, et le challenge suivant doit être choisit de manière à tenir compte de cet apprentissage pour obtenir la difficulté souhaitée. C'est le scénario du jeu qui permet de tenir compte de l'évolution du niveau du joueur, afin d'obtenir la courbe de difficulté désirée.

D'une manière plus précise, nous pourrions définir la construction de la difficulté d'un jeu vidéo à partir de deux ensembles de challenges, les challenges *atomiques* et les challenges *composites*. Un challenge atomique introduit un contexte de jeu entièrement nouveau pour le

2. Dialogue basé sur les émotions, l'expérience et la personnalité

3. Personnage Non Joueur, personnage du jeu n'étant pas l'avatar d'un joueur humain, mais dirigé par une IA. Le plus souvent associé aux personnages peuplant l'univers des jeux de rôles ou d'aventures.

joueur. Sa seule expérience est celle de jeux différents mais au **gameplay**[4] similaire auxquels il a pu jouer. Les challenges composites, quant à eux, sont construits en tenant compte des autres challenges. Les challenges composites exploitent, combinent dans un nouveau contexte les savoir et compétences acquises dans les challenges précédents.

Un challenge atomique d'un jeu comme ceux de la série Zelda (Nintendo) (Figure **??**), par exemple, permet au joueur de découvrir comment se déplacer, ou comment manipuler un objet donné, comme l'épée ou le boomerang. Un challenge composite va placer le joueur face à un ennemi qui ne peut être blessé que dans le dos, en tenant compte du fait que le joueur a appris à utiliser le boomerang, très utile pour prendre un ennemi à revers. Ce challenge composite est construit en tenant compte du challenge atomique au cours duquel le joueur à appris à utiliser le boomerang, et lui permet d'exploiter et de combiner ses connaissances dans un nouveau contexte.

La construction des challenges est donc finement ciselée par le game designer, et il apparait donc tout à fait approprié de lui fournir un outil capable d'en offrir une vue synthétique. La difficulté de chaque challenge doit être perçue dans le contexte du parcours du joueur dans le scénario du jeu, et la mesure de difficulté que nous avons définie dans la section précédente sera d'autant plus pertinente qu'elle pourra être représentée dans le contexte scénaristique global du jeu.

Dans la section suivante, nous présentons les différents travaux de recherche à partir desquels nous avons élaboré un modèle du scénario d'un jeu vidéo.

7.3 Formalisations existantes

La formalisation du scénario pour les applications ludiques interactives est particulièrement étudiée dans le domaine de la narration interactive. Les recherches dans ce domaine ont pour objectif d'offrir un scénario adaptatif, capable de prendre en compte les envies du joueur tout en conservant les propriétés d'un *bon* scénario. Notre objectif est plus modeste, et consiste à fournir aux designers un outil pour encoder le scénario d'un jeu vidéo, du point de vue des challenges proposés par le gameplay. Nous allons donc passer brièvement en revue certains des modèles proposés dans le domaine de la narration interactive, et présenter ce que nous leur empruntons pour atteindre notre propre objectif.

Avant toute chose, nous tenons à préciser clairement que notre objectif n'est en aucun cas de réaliser une application de narration interactive, telles que la recherche dans ce domaine les conçoit actuellement. Etudiée sous le prisme de la narration interactive, notre démarche

4. Rollings et Adams définissent le gameplay de la façon suivante : *Les challenges, ainsi que les actions que le joueur peut entreprendre pour les réussir, constituent le gameplay. D'un manière plus large, on peut considérer le gameplay comme ce que fait le joueur lorsqu'il joue, c'est à dire le type de problèmes qui lui sont posés et la manière dont il les résout.*

peut même apparaitre comme rétrograde, s'inspirant de modèles aujourd'hui abandonnés par ces chercheurs, car ne permettant pas d'offrir la plasticité nécessaire à un algorithme générateur d'histoires. Mais notre objectif est tout autre, nous cherchons à fournir au designer de jeu vidéos la capacité d'encoder l'histoire de leur jeu, c'est à dire l'enchainement des challenges, pour pouvoir l'étudier. La narration interactive promet certes de grandes avancées sur le plan des histoires interactives, et peut sûrement permettre de structurer et d'enrichir les applications souhaitant offrir une *narration émergente*. Mais dans un jeu vidéo, les phases de narration émergentes sont en réalité des phases de gameplay, qui doivent satisfaire de nombreuses contraintes propres à leur aspect ludique, comme par exemple le fait de proposer un niveau de difficulté adapté. Ces contraintes particulières font qu'il est aujourd'hui extrêmement complexe d'avoir recours à un algorithme génératif pour organiser la progression du joueur dans l'univers d'un jeu. La très grande majorité des jeux conçus aujourd'hui s'appuient donc sur le travail de scénaristes et de game designers, qui garantissent l'expérience ludique du joueur en limitant fortement ses possibilités, en ponctuant un scénario global faiblement interactif avec des phases de gameplay pur, ou challenges, finement calibrées. C'est cette structure, cette suite d'étapes créées par les designers, que nous voulons représenter pour permettre d'étudier de la courbe de difficulté du jeu. Nous allons donc nous inspirer des travaux réalisés en narration interactive, tout en tenant compte du fait que nos objectifs divergent.

Certaines solutions de narration interactive reposent sur la création d'une base de règles logiques générales, permettant de générer ensuite le scénario à la volée, comme le proposent IDTension de Nicolas Szilas [Szilas 05, Szilas 07] ou DEFACTO de Nikitas Sgouros [Sgouros 99]. Ces travaux ne s'appuient pas sur une représentation graphique des différentes étapes du scénario mais sur la formalisation de règles de construction d'un scénario, et proposent donc des modèles purement génératifs, les plus éloignés de notre objectif. Néanmoins, ces travaux posent certaines bases communes aux modèles de représentation d'un scénario. Szilas, reprenant Propp [Propp 28] et Bremond [Bremond 74] pose le principe selon lequel une histoire peut être divisée en une suite de processus, chaque processus ayant plusieurs états possibles : virtualité, passage à l'acte, achèvement ([Bremond 74] p.131) [Szilas 99].

$$\text{éventualité} \begin{cases} \text{passage à l'acte} \\ \text{non passage à l'acte} \end{cases} \quad \begin{cases} \text{achèvement} \\ \text{inachèvement} \end{cases}$$

FIGURE 7.1 – Trois temps du développement d'un processus [Bremond 74]

La notion de processus, très générique, se retrouve dans plusieurs systèmes de Narration Interactive. Le système utilisé pour réaliser une des applications de narration interactive les plus connues, Façade, utilise une suite de *beats* narratifs, semblables aux processus de

Bremond [Mateas 03]. Ils sont au départ virtuels, puis sélectionnés si leur impact dramatique correspond aux besoins de la narration, et ensuite exécutés ou interrompus.

D'autres systèmes s'appuient également sur une représentation explicite des différentes étapes du scénario et de leur ordre. Le groupe Liquid Narrative, de l'université de Caroline du Nord, a par exemple travaillé sur le système Mimesis, qui génère et encode la narration sous la forme d'un plan en ordre partiel [Saretto 01, Young 04, Young 99]. Chaque étape du plan correspond à une action narrative qui doit être réalisée dans l'univers du jeu. Les plans sont générés à la volée, de manière à satisfaire un objectif scénaristique tout en respectant toute contrainte qualitative pouvant être exprimée de manière formelle.

L'objectif de Mimesis n'est pas d'encoder une fois pour toutes la narration du jeu, mais de générer cette narration au fur et à mesure. Néanmoins, l'utilisation d'un plan comme image d'un scénario possible contribue à notre réflexion. Chaque étape d'un plan respecte le principe de processus narratif de Bremond. Chaque action du plan doit satisfaire des préconditions (virtualité du processus), et peut s'exécuter avec ou sans succès (achèvement ou inachèvement du processus). Par contre, la notation sous forme de plan n'est pas assez ouverte pour nous permettre de réaliser notre modèle. En effet, les plans sont générés par le système Mimesis pour fournir une histoire possible. Le plan est en ordre partiel et ne spécifie pas totalement l'ordre d'exécution ce qui offre un légère plasticité. Mais si une des actions échoue, il faut alors construire un plan différent, car chaque plan n'encode qu'une seule histoire possible.

D'autres auteurs se sont appuyés sur l'utilisation de plans pour les applications de narration interactive. Brian Magerko, de l'université du Michigan, utilise également un plan en ordre partiel, à ceci près qu'il n'annote pas causalement les liens entre les actions du plan [Magerko 05]. Marc Cavazza, de l'université de Teeside, utilise des réseaux hiérarchique de tâches pour permettre à chaque personnage de planifier son comportement en fonction des ses objectifs [Cavazza 02]. Cette notation permet à chaque personnage de planifier à nouveau son comportement en cas d'erreur, car le réseau hiérarchique de tâches peut encoder plusieurs solutions pour atteindre le même objectif. Néanmoins, l'objectif de cette architecture consiste à faire émerger la narration de l'interaction entre personnages et ne représente pas la structure globale de l'histoire, qui nous intéresse particulièrement.

Riedl et Young ont montré l'équivalence entre la génération de plan par Mimesis et les graphes de scénario [Riedl 06]. Riedl et Young décrivent les graphes de scénario comme un ensemble de plans et de choix du joueur. Les noeuds du graphe correspondent aux parties linéaires de l'histoire, encodées sous forme de plan, et les arcs du graphe correspondent aux choix du joueur, donc d'une manière générale aux différentes éventualités qui vont guider le joueur entre les histoires possibles. Afin d'atteindre notre objectif, c'est à dire encoder toutes les histoires prévues par le game designer, nous pouvons donc choisir d'étendre la notation en plan, c'est à dire un ordre partiel entre actions, vers une notation sous forme de graphe, en attribuant une sémantique particulière à ses arcs.

Ces travaux de recherche en narration interactive fournissent les bases d'un modèle de scénario pour le jeu vidéo, tel que nous l'entendons, c'est à dire dans l'objectif de modéliser la suite de challenges proposés au joueur par un jeu vidéo :

- Tout d'abord, nos challenges peuvent s'appuyer sur la logique générale des processus décrits par Bremond, logique cohérente avec l'ensemble des représentations étudiées. Nous devons définir des conditions pour déterminer si le joueur a débuté un challenge, et des conditions permettant de savoir si le joueur à réussit le challenge.
- Ensuite, nous souhaitons représenter l'ensemble des parcours possibles du joueurs, c'est à dire décrire au maximum la logique d'enchainement des challenges. Nous avons choisit de nous appuyer sur une représentation en graphe, en attribuant une sémantique particulière aux arcs liant les challenges. Cette sémantique nous permet de définir visuellement des relations logiques et temporelles entre les différents challenges.

Les sections suivantes présentent notre modélisation du scénario d'un jeu vidéo. Nous situons tout d'abord le modèle en présentant son interface avec un moteur de jeu, puis décrivons plus précisément le graphe de scénario.

7.4 Modèle de scénario

Les sections suivantes présentent le modèle de scénario d'un jeu vidéo. Tout d'abord, la première section décrit l'interface entre le moteur de jeu et le modèle de scénario, de manière a expliquer plus précisément la place du modèle dans le cadre d'un jeu vidéo.

7.4.1 Interface avec le moteur de jeu

Le modèle de scénario permet d'encoder graphiquement la logique de présentation des challenges au joueur. Comme nous l'avons dit précédemment, cette logique est déjà encodée dans la topologie du jeu et dans les scripts qui gouvernent le fonctionnement des objets du jeu : la plupart des jeux vidéos proposent des suites de challenges au joueur sans avoir recours à un modèle explicite du scénario du jeu. A ce titre, le modèle de scénario est une vue particulière du code qui régit le fonctionnement de l'univers du jeu.

Nous avons donc conçu le modèle de scénario comme un module supplémentaire du moteur de jeu, qui ne vient pas remplacer des fonctionnalités déjà existantes. Dans un premier temps, notre modèle doit permettre au designer d'encoder cette logique, par ailleurs déjà présente à des multiples endroits du code du jeu, et d'observer son bon déroulement. Cette séparation permet de construire un modèle facilement adaptable à de nombreux moteurs de jeux, car son objectif n'est que d'observer le déroulement du jeu, sans intervenir spécifiquement. En effet l'objectif global de notre modèle est d'offrir un outil d'analyse de la difficulté

d'un jeu vidéo, et non pas un outil de conception d'un jeu vidéo.

Mais bien évidement, ce travail apparait redondant : le level designer va devoir encoder la logique du jeu deux fois. De manière exhaustive et effective dans le code et la topologie du jeu vidéo, et de manière rapide et synthétique dans le modèle de scénario pour pouvoir réaliser différentes analyses de difficulté. Il est donc évident que lors de l'intégration définitive d'un tel modèle dans un moteur de jeu, il sera particulièrement utile de limiter au maximum cette redondance. Un premier moyen consisterait par exemple à permettre au designer, à partir du graphe de scénario, d'accéder directement au code de chaque challenge, c'est à dire d'intégrer plus finement les différentes interfaces de création du graphe de scénario et du code de level design. De cette manière les deux vues seraient développées de manières conjointes et participeraient l'une et l'autre à l'élaboration du jeu.

Cependant, la description d'une telle interface est spécifique au moteur choisit et sort du cadre précis d'analyse de la difficulté que nous nous somme fixés. La figure 7.2 présente l'interface minimale du moteur de jeu est du modèle de scénario, nécessaire à l'analyse de la difficulté d'un jeu vidéo.

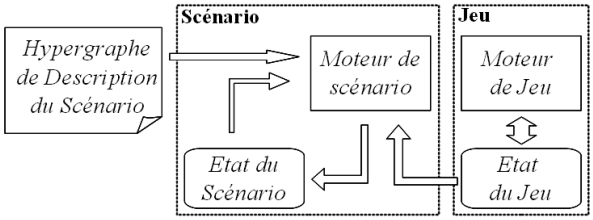

FIGURE 7.2 – Interface entre jeu est scénario.

La figure 7.2 met en évidence la séparation entre le modèle de scénario et le moteur de jeu. Le moteur de jeu calcule son propre état et suit sa propre logique mais communique au moteur de scénario tout évènement important. Nous avons définit ces évènement dans le chapitre précédent (sec. **??**). Le moteur de scénario se charge d'interpréter ces évènements pour calculer l'état du scénario, c'est à dire l'état des différents challenges qui le composent. Cette séparation a l'avantage de permettre au moteur de scénario de fonctionner sans faire appel au moteur de jeu, à partir d'une simple trace des évènements importants survenus pendant une session de jeu. De cette manière, un outil d'analyse de la difficulté peut être utilisé sans le moteur de jeu.

Bien que le modèle de scénario ne soit qu'un simple observateur du moteur de jeu, celui-ci doit être capable de transmettre au préalable une description des évènements particuliers qu'il souhaite observer. Le moteur de jeu n'est en effet pas capable de déterminer, a priori, les changements d'état du jeu susceptibles d'alimenter le calcul de l'état du modèle de scénario. Il doit donc exister une phase d'intégration du modèle de scénario au jeu en cours de déve-

loppement, qui permette de décrire, au niveau du moteur de jeu, les évènements importants que le moteur de scénario souhaite observer.

Nous définissons ainsi un ensemble de tests T. Ces tests ont valeur dans $\mathbb{B}$ et sont définis par le game designer sous forme textuelle. Il peut s'agir de savoir si le joueur est vivant ou mort, ou s'il possède tel ou tel objet. Ces tests sont ensuite intégrés au moteur de jeu. Lorsqu'un de ces tests change d'état, par exemple si le joueur meurt, ou ramasse un objet, un évènement correspondant est enregistré dans la trace et communiqué au moteur de scénario. Ce dernier conserve ainsi l'état courant de tous les tests utiles sur l'état du jeu, qu'il met à jour à chaque fois que le moteur de jeu lui communique un nouvel évènement. Grâce à l'ensemble T, le moteur de scénario dispose d'une abstraction de l'état du jeu, sous forme d'un ensemble de valeurs booléennes. Chaque état du jeu correspond à une configuration des valeurs prises par les différents tests de T. On notera U l'ensemble de toutes les combinaisons de valeurs possibles des éléments de T, c'est à dire l'ensemble des états du jeu possibles, tels que perçus par le moteur de scénario.

- T est un ensemble de tests sur l'état du jeu, réalisés par le moteur de jeu et dont la valeur est fournie au moteur de scénario sous forme d'évènements
- U correspond aux différentes valeurs attribuables simultanément aux éléments de T. U est l'ensemble des états possibles du jeu, tels que perçus par le moteur de scénario.

Nous avons donc conçu le modèle de scénario comme une vue particulière de la logique du jeu et de son exécution, parallèle à la spécification et à l'exécution de la logique globale du jeu. Le moteur de scénario est donc pour l'instant extérieur au moteur de jeu et permet simplement d'en étudier le fonctionnement. L'intégration de notre outil à un moteur particulier pourra permettre de limiter la redondance lié à l'encodage consécutif de la logique du jeu dans le moteur de jeu puis dans le moteur de scénario, mais ne fait pas partie des objectifs de cette thèse.

Dans cette section, nous avons décrit l'interface entre le moteur de jeu et le moteur de scénario. Dans les prochaines sections, nous décrivons plus précisément le modèle de scénario, en débutant par les challenges, dont la formalisation est présentée dans la section suivante.

7.4.2 Les challenges

Nous représentons un scénario sous forme d'un hypergraphe. Les sommets de cet hypergraphe sont les challenges auxquels le joueur pourra être confronté, et les arêtes de l'hypergraphe permettent de définir un ordre partiel et une partie de la logique d'enchainement des challenges. Dans cette première section, nous décrivons plus précisément les challenges, et leur représentation dans le modèle de scénario.

Nous avons déjà introduit les challenges précédemment, dans la section **??**. Les challenges représentent un objectif que le joueur doit atteindre, et leur état décrit la progression du

joueur vers cet objectif. Un challenge peut être à l'état *initial*, auquel cas le joueur n'a encore aucun rapport avec lui. Lorsque le challenge est $actif$, le joueur est effectivement en train de résoudre le challenge. Finalement, si le challenge passe à l'état *valide*, c'est que le joueur a réussit à atteindre l'objectif, mais s'il passe à l'état *echec* c'est que le joueur à échoué. L'automate de la figure 7.3 présente les différents états du challenge ainsi que le transitions possibles entre ces états.

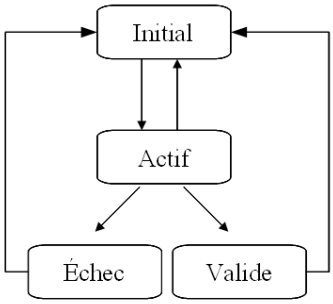

FIGURE 7.3 – Automate d'un challenge.

A tout instant, le moteur de scénario doit être capable de calculer l'état du modèle de scénario, c'est à dire, au niveau des challenges, être capable de décider d'une transition entre deux états. Ces transitions dépendent de l'évolution de l'état du jeu : pour savoir si le joueur à validé un challenge, le moteur de scénario s'attend à observer un changement dans l'état du jeu traduisant cette victoire. Comme nous l'avons définit précédemment, toute modification de l'état du jeu susceptible d'aider le moteur de scénario à calculer l'état du scénario est communiqué sous forme d'évènement. Nous avons définit U comme l'ensemble des différents états du jeu perçus par le moteur de scénario.

Pour chaque challenge, le game designer va construire en ensemble de conditions, qui régissent le fonctionnement de son automate. Ces conditions définissent quels éléments de U, c'est à dire quels états du jeu, entrainent une transition de l'automate. A chaque challenge correspondent ainsi quatre conditions :

- La condition d'activation, $\omega_a : U \rightarrow \mathbb{B}$, permet de déterminer si un challenge peut passer de l'état initial à l'état actif[5].
- La condition d'échec, $\omega_e : U \rightarrow \mathbb{B}$, permet de déterminer si un challenge peut passer de l'état actif à l'état échec.
- La condition de validation, $\omega_v : U \rightarrow \mathbb{B}$, permet de déterminer si un challenge peut passer de l'état actif à l'état valide.

5. Cette transition ne dépend pas que de ω_a mais également de l'état des liens entrants du challenge, que nous définissons par la suite, section 7.4.3.

– La condition de reset, $\omega_r : U \rightarrow \mathbb{B}$, permet de déterminer si un challenge peut passer de l'état valide ou échec à l'état initial.

Ces différents conditions permettent de calculer l'état d'un challenge en fonction de l'état du jeu. Toutefois, le calcul de l'état d'un challenge ne dépend pas uniquement de ces conditions. En effet, nous souhaitons encoder explicitement les relations d'ordre entre challenges, au moyen d'un hypergraphe. Le calcul de l'état du challenge doit donc tenir compte de la place du challenge dans l'hypergraphe de scénario. Les sections suivantes décrivent l'hypergraphe de scénario ainsi que sa prise en compte dans le calcul de l'état des challenges.

7.4.3 Hypergraphe de scénario

Dans la section précédente, nous avons définit les challenges ainsi que le calcul des transitions entre leur différents états. Dans cette section, nous nous préoccupons plus particulièrement de modéliser l'ordre de présentation des challenges au joueur. Nous allons construire un hypergraphe de scénario, dont les sommets sont les challenges, et dont le hyperarcs permettent de définir des conditions logiques qui contraignent l'activation des challenges. Par exemple, nous pourrons écrire qu'un challenge ne peut être débuté par le joueur que si deux autres challenges sont au préalable validés, et qu'un autre n'est pas encore débuté. Ce challenge ne pourra débuter que si l'hyperarc entrant qui définit cette condition est *satisfait*, c'est à dire si les challenges à l'origine de cet arc satisfont cet état particulier.

En effet, comme nous l'avons dit précédemment, les challenges sont présentés au joueur selon un ordre et une logique particulière. Les conditions de transition d'état définies dans la section précédente pourraient permettre de prendre en compte cet ordre : il suffirait de spécifier dans la condition d'activation des challenges ω_a, l'état que doivent respecter les challenge précédents. Néanmoins, nous pensons qu'il est beaucoup plus lisible et intuitif d'encoder cet ordre sous la forme d'un hypergraphe. De cette manière, la condition d'activation ω_a est beaucoup plus simple à écrire car elle ne doit pas tenir compte des challenges précédents, et surtout, la logique de déroulement du scénario peut être représentée sous forme graphique et gagner ainsi en lisibilité. Cette représentation graphique permet de faire coïncider le déroulement temporel et logique du scénario avec sa représentation dans l'espace, offrant une vision globale et intuitive du scénario qui ne pourrait être obtenue au moyen d'une suite de formules logiques textuelles. Nous décrivons donc, dans cette section, de quelle manière l'hypergraphe de scénario encode l'ordre des challenges, et son impact sur le calcul de l'activation d'un challenge.

Nous définissons un hypergraphe de scénario comme un tuple $(C, L, \phi_L, \lambda_L)$, avec :

– C : l'ensemble des challenges, sommets de l'hypergraphe, définis dans la section précédente.

– L : l'ensemble des *étiquettes de liens* (ou *étiquettes d'hyper-arcs*), caractérisés par les deux fonctions ϕ_L et λ_L suivantes :

- $\phi_L : L \rightarrow \mathbb{P}(C) \times \mathbb{P}^1(C)$ est une fonction de lien, qui à toute étiquette de lien fait correspondre un hyper-arc orienté, c'est à dire une paire ordonnée $(O, \{c\})$ où O est un ensemble non vide de challenges nommé *origine du lien*, et c est un challenge nommé *cible du lien*, et tel que $c \notin O$. Nous ne considérons que des liens dont le challenge cible est un singleton.
- $\lambda_L : C \times L \nrightarrow \{=, \neq\} \times \{actif, valide, echec\}$ est la *précondition d'un lien, vis à vis d'un challenge d'origine*, qui fait correspondre, à tout couple (c, l) tels que c appartient à l'origine de l, un couple (op, e) ou op est un opérateur d'égalité ou d'inégalité, et e une étiquette correspondant à l'un des trois états $\{actif, valide, echec\}$ d'un challenge.

Un *lien* peut être vu comme un triplet $(\{c_i\}, (op_i, e_i), r)$, qui relie l'*origine* du lien, un ensemble de challenges $\{c_i\}$, à la *cible* du lien, un challenge unique r. La *précondition* (op_i, e_i) spécifie l'état que chaque challenge c_i doit respecter pour *satisfaire* le lien. Nous notons un lien l, tel que $\phi_L(l) = (\{c_1, \dots c_n\}, r)$ et $op_i, e_i = \lambda_L(c_i, l)$, de la façon suivante : $\{c_i \overset{op_i, e_i}{\rightsquigarrow} r\}$.

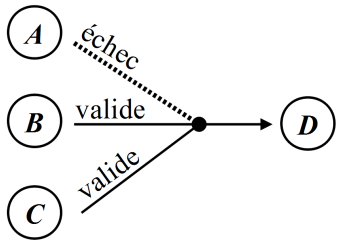

FIGURE 7.4 – Exemple de lien

Les pointillés indiquent une négation.

La figure 7.4 présente un exemple d'hypergraphe, avec un seul lien. On a donc, pour ce lien l, $\phi_L(l) = (\{A, B, C\}, D)$, c'est à dire que l'origine de l est l'ensemble de challenges $\{A, B, C\}$ et que le challenge D est sa cible. Le lien l décrit l'état que chaque challenge d'origine de l doit respecter. Par exemple, $\lambda_L(B, l) = \{=, valide\}$, ce qui signifie que pour que l soit satisfait, il faut le challenge B soit valide. Le lien l exprime aussi le fait que pour pouvoir activer D, il faut que A ne soit pas en échec, et que B et C soient valides.

Pour qu'un lien soit satisfait, sa précondition doit être satisfaite, c'est à dire que l'ensemble des challenges d'origine doivent respecter un état particulier. Si on note $\pi(c)$ l'état d'un challenge, alors l'ensemble des liens satisfait L_S^π peut être exprimé de la façon suivante :

$$L_S^\pi = \{\{p_i \overset{op_i, e_i}{\rightsquigarrow} r\} \in L \mid \bigwedge_i op_i(\pi(p_i), e_i)\} \tag{7.1}$$

Les liens permettent d'ordonner les challenges en spécifiant une condition supplémentaire à l'activation d'un challenge. En effet, pour qu'un challenge passe de l'état initial a l'état

actif, il faut à la fois que ω_a soit satisfaite, mais également que *si le challenge est la cible de liens, au moins un de ces liens soit satisfait.* Si on appelle la condition générale de transition de l'état initial à l'état actif ω'_a, celle ci s'écrit :

$$\omega'_a(c) = \omega_a \wedge [(\exists l / \phi_L(l) = (X, c) \wedge l \in L_S^{\pi}) \vee \nexists l / \phi_L(l) = (X, c)] \tag{7.2}$$

En attribuant une sémantique particulière aux hyper-arcs de l'hypergraphe de scénario, nous sommes ainsi en mesure de spécifier un pré-ordre sur les challenges, assorti de conditions logiques. Dans la section suivante, nous présentons le calcul complet de l'état du graphe, à partir des conditions de transition et de la satisfaction des liens.

7.4.4 Calcul de l'état du graphe

L'algorithme suivant est appelé à chaque fois que le moteur de scénario reçoit un nouvel évènement, lu dans une trace ou reçu en temps réel, ou à l'initialisation du moteur. Cet algorithme montre le calcul de l'état du scénario, c'est à dire de l'état des liens et des challenges de l'hypergraphe de scénario. On considère qu'à chaque fois que le moteur reçoit un évènement, il met à jour son abstraction de l'état du jeu, et la fournit à cet algorithme.

```
1  fonction MajScenario(GameState u, Graphe g)
2  {
3
4    var etatModifie:Booleen = true;
5
6    tant que(etatModifie)
7    {
8      var c:Challenge;
9      var l:Lien;
10
11     //Initialisation
12     etatModifie = false;
13
14     //On met a jour tous les challenges
15     pour chaque challenge c de g
16     {
17       //On applique les conditions de transition
18       si(MajChallenge(c,u))
19         etatModifie = true;
20     }
21
22     //On met à jour tous les liens
23     pour chaque lien l de g
```

```
24      {
25        //On vérifie la satisfaction du lien
26        si(MajLien(l,u))
27          etatModifie = true;
28      }
29    }
30  }
```

L'algorithme `MajScenario` montre les différentes étapes de mise à jour du graphe de scénario. Tout d'abord, les conditions de transition des challenges sont évaluées de manière à calculer leur nouvel état. Les challenges sont uniquement autorisés à effectuer une transition dans leur automate d'état, si les conditions de transitions le permettent. L'état des challenges obtenu n'est pas forcément stable, c'est à dire qu'une nouvelle exécution de la boucle pourrait déclencher d'autres transitions. Mais il est nécessaire de répercuter tout changement d'état d'un challenge sur ses liens sortants, aussi les challenges ne sont ils mis à jour qu'une seule fois, avant de mettre à jour les liens.

Le nouvel état des challenges est ensuite utilisé pour calculer le nouvel état des liens. Tant qu'il existe un challenge ou un lien dont l'état change, ces deux mises à jour sont répétées, jusqu'à obtention d'un graphe de scénario stable. Bien sûr, on remarque que dans le cas d'un hypergraphe cyclique, ce calcul peut être infini. Nous n'avons toutefois pas limité le graphe de scénario à un modèle acyclique, car les scénarios de jeux s'appuient par essence sur des cycles, lorsqu'ils demandent par au joueur d'effectuer plusieurs fois un même challenge. Mais si le scénario est bien construit, le cycle demandera forcément une intervention du joueur, et donc une variation de l'état du jeu. Dans un scénario bien construit, la boucle de l'algorithme `MajScenario` ne sera donc jamais infinie. L'implémentation de cet algorithme dans notre application limite néanmoins le nombre de mises à jour successives de manière a détecter une boucle infinie, signe d'une erreur dans la modélisation du scénario.

7.5 Synthèse

Dans ce chapitre, nous avons présenté comment encoder le scénario d'un jeu vidéo, au sens ou nous l'entendons, c'est à dire la logique d'enchainement des challenges d'un jeu vidéo. Cette suite de challenges est primordiale pour un outil d'analyse de la difficulté d'un jeu vidéo. Elle permet de faire apparaitre la structure sur laquelle s'appuie le game designer pour construire la difficulté du jeu. Nous nous sommes inspirés des recherches effectuées en Narration Interactive pour construire notre propre formalisme. De ces recherches, nous avons principalement retenu la notion de processus et celle d'ordre, dont une représentation graphique est particulièrement pertinente.

Nous avons ensuite décrit plus précisément l'interface entre le moteur de scénario et un

moteur de jeu. Notre modèle propose une vue particulière de la logique du jeu, et nous l'avons conçu comme un module supplémentaire, parallèle au moteur de jeu. Nous somme conscients que lors de l'intégration définitive de notre moteur de scénario à un moteur de jeu particulier, il sera pertinent de permettre une plus forte interaction entre la logique décrite par le scénario et celle décrite au sein des scripts de level design. Néanmoins, notre objectif est de fournir ici un modèle adaptable à n'importe quel moteur de jeu, et son intégration au sein d'une plateforme particulière est une étape supplémentaire qui sort du champ de cette thèse.

Nous avons ensuite décrit le calcul de l'état du scénario, c'est à dire de l'état des challenges et des différents liens qui composent l'hypergraphe de scénario. Ce calcul permet de décrire précisément la sémantique du graphe de scénario, et plus particulièrement la manière dont l'état des liens interagit avec l'état des challenges, en ajoutant une précondition d'activation. Ce calcul, réalisé par le moteur de scénario, permet au designer de visualiser, à partir d'une trace d'évènements ou pendant une session de jeu, l'évolution de l'état du scénario.

Le fait d'utiliser une représentation graphique pour le modèle de scénario permet de fournir une interface particulièrement pertinente pour la description des différents calculs liés à l'évaluation de la difficulté. Nous avons développé une application exploitant cette interface, que nous présentons dans le chapitre suivant.

Glossary

game designer Le game designer concoit les règles du jeu, il a une vision générale du gameplay, qui sera instancié, contextualisé par le level designer. Une section spéciale lui est consacré dans la partie Vocabulaire.. 1

gameplay Rollings et Adams définissent le gameplay de la façon suivante : *Les challenges, ainsi que les actions que le joueur peut entreprendre pour les réussir, constituent le gameplay. D'un manière plus large, on peut considérer le gameplay comme ce que fait le joueur lorsqu'il joue, c'est à dire le type de problèmes qui lui sont posés et la manière dont il les résout.*. 3

PNJ Personnage Non Joueur, personnage du jeu n'étant pas l'avatar d'un joueur humain, mais dirigé par une IA. Le plus souvent associé aux personnages peuplant l'univers des jeux de rôles ou d'aventures.. 2

Bibliographie

[Bosser 07] Anne-Gwenn Bosser, Guillaume Levieux, Karim Sehaba, Axel Buendia, Vincent Corruble, Guillaume de Fondaumière, Viviane Gal, Stéphane Natkin & Nicolas Sabouret. *Dialogs Taking into Account Experience, Emotions and Personality.* In ICEC, pages 356–362, 2007.

[Bremond 74] Claude Bremond. Logique du recit. Seuil, Paris, 1974.

[Cavazza 02] Marc Cavazza, Fred Charles & Steven J. Mead. *Character-Based Interactive Storytelling.* IEEE Intelligent Systems, vol. 17, no. 4, pages 17–24, 2002.

[Magerko 05] Brian Magerko. *Story Representation and Interactive Drama.* In AIIDE : First Artificial Intelligence and Interactive Digital Entertainment Conference, June 1-5, 2005, Marina del Rey, California, USA, pages 87–92, 2005.

[Mateas 03] Michael Mateas & Andrew Stern. *Façade : An Experiment in Building a Fully-Realized Interactive Drama.* In Game Developers Conference (GDC'03), 2003.

[Propp 28] Vladimir Propp. Morphologie du conte. Seuil, 1928.

[Riedl 06] Mark O. Riedl & R. Michael Young. *From Linear Story Generation to Branching Story Graphs.* IEEE Comput. Graph. Appl., vol. 26, no. 3, pages 23–31, 2006.

[Saretto 01] C.J. Saretto & Michael Young. *Mediation in mimesis liquid narrative.* In ACSME, 2001.

[Sgouros 99] Nikitas M. Sgouros. *Dynamic generation, management and resolution of interactive plots.* Artif. Intell., vol. 107, no. 1, pages 29–62, 1999.

[Szilas 99] Nicolas Szilas. *Interactive drama on computer : beyond linear narrative.* In Narrative Intelligence, Association for the Advancement of Artificial Intelligence (AAAI), Fall Symposium. AAAI Press, 1999.

[Szilas 05] Nicolas Szilas. *The future of interactive drama.* In IE2005 : Proceedings of the second Australasian conference on Interactive entertainment, pages 193–199, Sydney, Australia, Australia, 2005. Creativity & Cognition Studios Press.

[Szilas 07] Nicolas Szilas, Jason Barles & Manolya Kavakli. *An implementation of real-time 3D interactive drama.* Computers In Entertainment (CIE), vol. 5, no. 1, page 5, 2007.

[Vega 04] Liliana Vega. *Modélisation et analyse spatiale et temporelle des jeux vidéo basées sur les réseaux de Pétri.* PhD thesis, Conservatoire National des Arts et Métiers, 2004.

[Young 99] R. Michael Young. *Notes on the Use of Plan Structures in the Creation of Interactive Plot.* In Narrative Intelligence, Association for the Advancement of Artificial Intelligence (AAAI), Fall Symposium. AAAI Press, 1999.

[Young 04] R. Michael Young. *An architecture for integrating plan-based behavior generation with interactive game environments.* Journal of Game Development, vol. 1, no. 1, pages 51–70, 2004.