Evolution de l'intelligence artificielle dans le cadre de la génération procédurale en jeux vidéo

Killian Mahé, Adrien Botrel et Jorick Célarier
Université du Québec à Chicoutimi
killian.mahe1@uqac.ca, adrien.botrel1@uqac.ca, jorick.celarier1@uqac.ca

I. Abstract

Dans cette revue de littérature, nous allons décrire les bases de la génération procédurale et montrer son importance dans le développement de jeux vidéo, en particulier au niveau de la rejouabilité d'un jeu mais aussi dans la simplification du travail des game designer. Avec la génération procédurale, deux grands axes existent: l'amélioration de l'esthétique d'un jeu afin de le rendre plus immersif et plus adapté à son univers et son histoire ainsi que l'amélioration des mécaniques de jeu afin d'améliorer et d'adapter la difficulté et les challenges proposés au joueur en fonction de ses actions et de ses compétences. Finalement, nous montrons que même si de nombreuses avancées ont eu lieu depuis la création des jeux vidéo, il y a encore beaucoup de difficultés et de problèmes à surmonter, notamment au niveau de la détection du degré d'amusement du joueur.

II. Introduction

Depuis longtemps, le rôle du game designer est très important dans le monde du jeu vidéo. C'est lui qui est chargé de réaliser l'environnement et le scénario et de s'assurer de la jouabilité d'un jeu mais il doit aussi faire en sorte que le joueur prenne du plaisir à jouer. C'est donc un métier difficile qui demande de plus en plus de temps, d'autant plus que les joueurs ont des attentes de plus en plus poussées au niveau du design et des mécaniques de jeu afin d'être plus en immersion dans le jeu. C'est là que l'intelligence artificielle (IA) intervient avec la génération procédurale. Cette technologie permet de générer des environnements ou des niveaux de jeu automatiquement en se basant sur des assets. Cela permet, entre autres, d'augmenter la rejouabilité d'un jeu en proposant un contenu nouveau à chaque partie. Bien que cela permette au game designer de gagner beaucoup de temps lors de la création d'un jeu, cette technologie possède certaines difficultés, notamment au niveau de la génération de plaisir et d'amusement chez le joueur.

Nous nous sommes alors posés la question suivante: En quoi les récentes évolutions de l'IA ont permis aux jeux vidéo de répondre aux attentes des joueurs toujours plus contraignantes grâce à la génération procédurale ? Pour y répondre, nous avons réalisé cette revue littéraire répartie de la manière suivante. Dans un premier temps, nous allons expliquer plus en détail ce qu'est la génération procédurale. Puis nous allons décrire, dans deux parties distinctes, les deux principaux axes de travail de la génération procédurale, soit l'esthétique et les mécaniques de jeu. Nous parlerons ensuite des limites de cette technologie pour finir par un résumé de ce que nous avons appris.

III. La génération procédurale

Dans cette section, nous allons présenter ce qu'est la génération procédurale ainsi que les principaux jeux vidéo développés avec. La génération procédurale se base sur l'utilisation d'algorithmes afin de générer les éléments d'un jeu. Un élément peut être à la fois un objet, un niveau ou même une carte ouverte de taille infinie. Des exemples connus de jeux utilisant ce principe sont Minecraft (Mojang, 2009)[1] ou No Man's Sky (Hello Games, 2016)[2] dans lesquels la carte est générée de manière procédurale, c'est-à-dire sans intervention humaine. La fonctionnalité et le but premier de la génération procédurale est de créer un jeu qui soit jouable et terminable, c'est-à-dire que le joueur doit avoir la possibilité, après une suite d'actions, d'arriver jusqu'à la fin d'un jeu ou d'un niveau. La

difficulté et l'enjeu de cette technologie restent néanmoins de faire en sorte que le joueur prenne du plaisir à jouer. Il faut donc que les graphismes et l'esthétique soient suffisamment développés pour correspondre à l'univers et à l'histoire du jeu. Il faut aussi une implémentation de mécanismes particuliers permettant au jeu d'évoluer en fonction des attentes et des compétences du joueur.

IV. L'esthétique

Dans le domaine du jeu vidéo, l'esthétique et la création d'assets représentent une part très importante du travail de conception et de production d'un jeu. Les assets représentent les différentes ressources, éléments ou fichiers d'un jeu. Il peut s'agir d'éléments graphiques, audio, vidéo ou textuels. La création de ces différentes ressources nécessite un temps de travail et un nombre de désigner considérable. L'un des enjeux principaux de la génération procédurale est donc, non pas de remplacer le travail de designer, mais de soulager ces derniers en mettant en place des techniques de génération avancées permettant une variabilité élevée des assets proposés. En utilisant ainsi la génération procédurale de ressources esthétiques, les auteurs et les sociétés éditrices espèrent réduire grandement certains coûts de développement. Un intérêt majeur de cette génération est la possibilité de proposer aux joueurs un environnement virtuel diversifié et dynamique. La rejouabilité des jeux est un principe conducteur poussant de plus en plus d'éditeurs à adopter cette technique. Dans la plupart des jeux vidéos, l'esthétique comprend plusieurs types de contenu : les terrains, la végétation, les villes, les bâtiments et les routes. Nous allons essayer ici de décrire les principales techniques utilisées et les pistes de développement futur.

Dans la génération de terrains, il existe principalement trois familles de méthodes : les méthodes basées sur la simulation, sur l'édition et enfin la génération procédurale (Grosbellet, 2015) [3]. Contrairement à la simulation utilisant des modèles physiques, les méthodes de génération procédurale utilisent plusieurs types d'algorithmes permettant la génération automatique des terrains sur de larges surfaces. Les premiers travaux de génération automatique dans ce domaine remontent à 1988 par l'utilisation d'un modèle d'érosion (Kelley et al, 1988) [4]. La méthode utilisée permettait de représenter le terrain comme étant une surface sous tension sur laquelle il était possible d'effectuer des calculs d'érosion. Cette modélisation des terrains a été très utilisée, elle permet de modéliser un terrain à partir de rivières par le biais du phénomène d'érosion. Un autre modèle encore beaucoup utilisé de nos jours est le modèle fractal. Un fractal est un objet mathématique dont la structure reste similaire peu importe l'échelle. À l'image des poupées russes, il est possible de retrouver la structure de la figure en zoomant sur n'importe quelle partie de celle-ci. Le flocon de Koch est un exemple simple de ce qu'est une structure fractale (Lajoie, 2006) [5]. Ce modèle est en partie utile lors de la génération de texture ou bien de cartes de hauteurs (génération de terrain). Nous pouvons citer ainsi l'algorithme de Perlin noise (Perlin, 1985) [6]. D'autres modèles se basent plutôt sur des arbres de construction (Géneveaux et al, 2015) [7]. Dans ses travaux, l'auteur utilise des réseaux hydrographiques générés procéduralement afin de pouvoir créer des terrains. En contrôlant les pentes des rivières et des différentes vallées et en pilotant la génération du réseau hydrographique il est possible de créer des terrains faisant apparaître des bancs de sable ou d'autres attributs avancés.

Malgré toutes ces techniques de modélisation du terrain. Le domaine de prédilection de la génération procédurale reste la végétation et la création de structure en arbre comme les cavernes par exemple. Dans ce domaine de recherches de nombreuses études se basent sur les travaux de Lindermayer sur les L-System (Lindenmayer, 1968) [8]. Les L-System permettent de modéliser des processus d'évolution et de développement tels que la croissance de plantes ou la prolifération de bactéries. Une version plus complète a été réalisée, open L-System (Mech et Prusinkiewicz, 1996) [9]. Cette version a permis de prendre en considération l'environnement local afin de modéliser des végétaux plus réalistes. Plusieurs méthodes cherchent également à reproduire le développement naturel des plantes en utilisant du mimétisme. Ces méthodes auto-organisationnelles utilisent la lumière et l'environnement afin d'influencer le développement et la croissance de la structure des arbres par le phénomène de colonisation de l'espace (Palubicki et al, 2009) [10]. Plus récemment, l'utilisation des chaînes de Markov a suscité un intérêt croissant dans la création de

végétaux. C'est le cas par exemple des méthodes de Monte-Carlo par chaîne de Markov. Ces méthodes permettent de générer de la flore ayant une apparence naturelle et réaliste, mais respectant également certaines contraintes fournies comme paramètres d'entrée (Talton et al., 2011) [11].

La plupart des approches abordées depuis le début sont dîtes constructives. Ces méthodes sont principalement basées sur un processus de génération à temps fixe, elles ne performent pas d'itérations. Elles utilisent des règles afin de construire un résultat. Cependant de plus en plus de recherches se tournent à présent vers des méthodes orientées vers des algorithmes basés sur la recherche. Il est ainsi possible de citer plusieurs familles d'algorithmes : les algorithmes évolutionnaires, l'apprentissage par renforcement, les méthodes basées sur l'optimisation par gradient ou encore les méthodes de recherches aléatoires. L'intérêt de ces méthodes réside dans leur fonctionnement. Contrairement aux approches classiques, les algorithmes de recherche se basent sur des fonctions d'évaluation afin de trouver un résultat correspondant aux contraintes demandées.

Certaines de ces méthodes ont été utilisées par exemple dans le cadre de la création de particule et de végétation. Dans ces travaux Hastings propose un nouvel algorithme, intitulé cgNEAT, faisant intervenir principalement des réseaux de neurones et permettant de générer automatiquement des assets durant le déroulement du jeu en réagissant dynamiquement aux actions et aux préférences du joueur (Hastings et al., 2009) [12]. Il a ensuite mis en pratique ce nouvel algorithme dans son jeu multijoueur Galatic Arms Race dans lequel les joueurs combattent des ennemis et font l'acquisition de nouvelles armes uniques à système de particule. Celles-ci sont générées automatiquement par l'algorithme cgNEAT. Ainsi, le jeu n'est plus seulement capable de proposer au joueur un contenu varié, mais également adapté aux préférences de ce dernier. Dans le même principe, le jeu multijoueur Petalz, propose aux joueurs de partager et de marchander les découvertes en matière de fleurs (Risi *et al*., 2015). Il est ensuite possible d'observer ses fleurs dans un environnement virtuel en trois dimensions. La génération de ces fleurs s'appuie sur l'utilisation d'un algorithme d'encodage CPPN-NEAT, de croisement et de classification (utilisant entre autres des réseaux de neurones artificiels) permettant ainsi une grande diversité dans l'esthétique des fleurs. La partie la plus importante dans les méthodes par recherche reste la fonction d'évaluation qui permet d'assigner un nombre à l'objet étudié basé sur son niveau de désirabilité. La plupart du temps, cette évaluation est réalisée en jouant au jeu et en assignant une note à l'expérience et au gameplay. D'autres essayent d'utiliser plutôt l'apprentissage machine pour estimer l'expérience du joueur.

V. Les mécaniques

Après avoir vu en quoi la génération procédurale était importante pour l'esthétique des jeux vidéo, il en est de même pour un autre aspect fondamental : les mécaniques. Nous allons voir en quoi et comment la génération procédurale sur ces mécaniques peut avoir un impact sur les jeux vidéo.

La génération peut à la fois se faire de manière semi-automatique, c'est-à-dire que le game designer crée les éléments permettant d'assurer la jouabilité d'un jeu et que le programme complète ensuite le niveau afin de développer sa difficulté et son contenu, soit elle peut être réalisée de manière entièrement automatique, le programme est alors chargé de développer entièrement le jeu de manière procédurale. Dans les deux cas, des agents peuvent être créés dans le but de tester la jouabilité du jeu.

Différents types d'agents existent et sont présentés lors du "GVGAI Level Generation Competition" de 2019 (Drageset, 2019)[14]. Il y en a des simples comme RANDOM, qui, comme son nom l'indique, va effectuer des actions aléatoires à chaque étape, ou bien DONOTHING, qui ne va rien effectuer à chaque étape. Il y a également One Step Lookahead (OSLA) : celui-ci va, à l'aide d'une heuristique, comparer et choisir l'état qui favorise la proximité parmi tous les états possibles accessibles à partir de l'état actuel. Mais l'un des agents le plus efficace est YOLOBOT. Il est une combinaison de 2 méthodes qui permet de gérer le cas de jeux déterministes (Best First Search guidé par heuristique) et le cas de jeux stochastiques (Arbre de recherche de Monte Carlo (MCTS)). Les jeux déterministes sont appelés ainsi car le prochain état de l'environnement est déterminé par l'état courant et l'action de l'agent. Au contraire, dans les jeux stochastiques, on ne sait pas

quel sera le prochain état, l'aléatoire est présent. Ainsi, ces agents sont indispensables pour vérifier si les niveaux générés sont jouables et valides.

Lors de cette compétition, des générateurs ont d'ailleurs pu être créés de manière automatique. En effet, les 3 types différents de générateurs ne prennent que quelques paramètres et peuvent créer des niveaux, plus ou moins réalisables. Il y a le Random Generator, qui vient placer un sprite de chaque type et rempli ensuite l'espace aléatoirement. Il a l'avantage d'inclure tous les sprites susceptibles d'atteindre l'objectif. Le Constructive Generator, quant à lui, repose sur le fait qu'il construit des murs connectés. Il s'assure que tous les espaces libres sont connectés et que les ennemis sont loin par exemple, mais ne garantit pas que tous les sprites soient utilisés. Le dernier générateur, Genetic Generator, est l'un des plus avancés. Il utilise un des générateurs au-dessus puis va évoluer les niveaux générés à l'aide d'une fonction de fitness. On voit donc qu'il est possible de créer différents types de générateurs de niveaux, mais dans des cas précis (ici, des niveaux en 2D format VGDL).

La génération procédurale est utilisée depuis longtemps dans les jeux vidéo. Le premier grand succès a eu lieu en 1980 avec Rogue (Toy et al, 1980) [15]. Dans ce jeu, le joueur doit parcourir les salles une à une en combattant les monstres s'y trouvant jusqu'à atteindre la salle finale. Les développeurs, voulant pouvoir y jouer tout en étant surpris par leur propre jeu, ont alors décidé de générer les salles de manière procédurale. D'autres jeux ont ensuite repris ce principe comme le fameux jeu Diablo (Blizzard, 1997)[16] ou encore Spelunky (Mossmouth, 2008)[17]. D'autres développeurs ont aussi utilisé la génération procédurale dans le but de modifier la difficulté du jeu ou la fréquence d'apparition d'objets en fonction des compétences et des statistiques du joueur. Un des premier jeu à avoir utilisé ce principe est Galactic Arms Race (Evolutionary Games, 2010)[18]. Dans ce jeu se déroulant dans l'espace, le joueur a la possibilité de ramasser des armes afin de pouvoir tirer sur les vaisseaux ennemis. Le jeu analyse alors le pourcentage de ramassage de chaque arme pour modifier les fréquences d'apparition en faisant en sorte de faire apparaître plus souvent les armes préférées du joueur afin que ce dernier passe un bon moment. Un autre jeu bien connu est Left 4 Dead (Valve Corporation, 2008)[19] dans lequel le jeu fait en sorte de modifier la fréquence d'apparition des zombis en fonction de la rapidité du joueur à vaincre les vagues précédentes. Le but est alors de trouver le meilleur équilibre entre la difficulté et l'amusement afin d'atteindre le meilleur "niveau de tension" possible.

Il existe ainsi plusieurs manières d'utiliser la génération procédurale dans un jeu. D'un côté, il y a la génération par recherche qui se base sur l'utilisation de fonctions d'évaluation et sur l'approche de recherche et d'optimisation stochastique afin de permettre d'améliorer la qualité du jeu. Cette méthode est très courante, surtout sur les jeux nécessitant la génération de niveaux en temps réel tels que StarCraft (Blizzard, 1998)[20], Super Mario Bros (Nintendo, 1983)[21], Doom (id Software, 1993)[22] ou encore Angry Birds (Rovio Entertainment Corporation, 2009)[23]. De plus, même si cela ne concerne pas directement notre étude, Browne et Marie ont utilisé en 2010 cette génération par recherche dans le but de créer des jeux de plateau tout en décrivant les règles pour y jouer. D'un autre côté, la génération procédurale peut être utilisée en collaboration avec l'apprentissage par renforcement.

Cependant, cet apprentissage n'est pas simple à mettre en place. En effet, différents paramètres rentrent en compte comme les quêtes ou les niveaux par exemple qui font que la génération n'est pas aussi triviale que pour la création d'images. Dans ce cas, on entraîne un GAN (Generative Adversarial Networks) qui utilise principalement le voisinage pour apprendre. Mais dans les jeux vidéo, les attributs sont éloignés les uns des autres. Il existe cependant plusieurs solutions pour résoudre ce problème de jouabilité, l'une d'elles étant le bootstrapping. Le principe est que toutes instances générées et satisfaisants les contraintes de jouabilité sont ajoutées au jeu de données d'entraînement du modèle. Mais la méthode la plus utilisée est d'utiliser l'apprentissage par renforcement pour générer un premier espace général et d'ensuite utiliser des méthodes de recherche sur cet ensemble.

VI. Les limites actuelles de la génération procédurale

Au travers de cet article nous avons essayé de rendre compte des différentes méthodes qui ont existées, qui

existent et les nouvelles approches émergentes. Cependant, la génération procédurale dans le domaine du jeu vidéo a encore quelques obstacles à surmonter dans le futur. En effet, certaines technologies liées à l'intelligence artificielle telles que l'apprentissage machine par réseaux de neurones ont du mal à s'adapter aux contraintes du domaine. Les contraintes de jouabilités comme le niveau, la carte ou bien les quêtes donnent lieu à une génération complexe et coûteuse. Ainsi entraîner certains algorithmes d'apprentissage comme les réseaux adverses génératifs sur un large ensemble d'instances fonctionnelles ne garantit par la jouabilité des niveaux qui seront produits (Risi et Togelius, 2019) [26]. En effet, contrairement à d'autres domaines, la plupart des fonctionnalités et des contraintes dépendent d'attributs éloignés les uns des autres. Actuellement, beaucoup d'éditeurs préfèrent rester sur des méthodes de génération procédurale classique afin de laisser plus de contrôle aux designers. Très peu de recherches se penchent sur des moyens de créer des jeux qui soient plus interactif. Quelques recherches se posent sur la création de contenu par génération procédurale qui pourrait être contrôlé directement par le concepteur, mais peu de choses du côté des joueurs (Smith, 2014) [25].

VII. Conclusion

Dans cette revue littéraire, nous avons vu que la génération procédurale était le fait de générer de manière automatique ou semi automatique des objets, des niveaux ou des cartes entières dans les jeux vidéo. Au cours de cette revue, on a remarqué qu'il y a eu une évolution dans les objectifs de cette génération procédurale. En effet, au début, il y avait surtout la volonté de créer des jeux se générant tout seul afin de permettre sa rejouabilité. De plus, cela permet de diminuer la charge de travail des game designer et ainsi diminuer le temps nécessaire à la réalisation d'un jeu. Un défi s'est ensuite petit à petit imposé. La volonté de créer des jeux jouables s'est peu à peu dirigée vers la volonté de créer des jeux toujours jouables mais permettant de provoquer de nombreuses sensations chez le joueur comme l'amusement, l'excitation, la tension, la peur … Ce défi peut être résolu de deux manière: soit en améliorant le côté esthétique d'un jeu en faisant en sorte que le design se rapproche le plus possible de l'univers et de l'histoire du jeu, soit en faisant en sorte de modifier la difficulté du jeu en analysant les compétences et les actions du joueur. Cependant, bien que l'évaluation de ces compétences se fait de plus en plus précisément, cela reste difficile pour un ordinateur de mesurer précisément le degré d'amusement du joueur et d'adapter le jeu pour produire un challenge ajusté au niveau du joueur. Néanmoins, la génération procédurale est une technologie qui sera intéressante à suivre et qui connaîtra encore de nombreuses évolutions dans les années à venir, d'autant plus que son utilisation peut être élargie à d'autres domaines comme le cinéma ou encore le machine learning avec la génération de sets de données par la machine elle-même qu'elle pourra utiliser lors de son apprentissage.

Références

1. Mojang. (2009). *Minecraft*. Video Game. Mojang: Stockholm, Suède.

2. Hello Games. (2016). *No Man's Sky*. Video Game. Hello Games: Guildford, Royaume-Uni, 2016

3. Grosbellet, F. (2015, November 20). (thesis). *Génération de détails dans les mondes procéduraux*.

4. Alex Kelley, Malin, M., & Nielson, G. (1988). Computer Graphics. In *Terrain simulation using a model of stream erosion* (pp. 263–268). Computer Graphics.

5. Josiane, Lajoie. (2006). *La géométrie fractale*. Université du Québec à Trois-Rivières.

6. Ken Perlin. (1985). *An image synthesizer*. In *Siggraph Computer Graphics*, (pp. 287-296).

7. Jean-David Génevaux, Éric Galin, Adrien Peytavie, Éric Guérin, Cyril Briquet, François Grosbellet et Bedrich Beneš. (2015). *Terrain modeling from feature primitives*. Computer Graphics Forum – CGF.

8. Aristid Lindenmayer. (1968). *Mathematical models for cellular interactions in development i. filaments with one-sided inputs*. Journal of Theoretical Biology, (pp. 280 – 299).

9. Radomír Mech et Przemyslaw Prusinkiewicz. (1996). *Visual models of plants interacting with their environment*. In *Proceedings of the 23rd Annual*

Conference on Computer Graphics and Interactive Techniques, (pp. 397 – 410).

10. Wojciech Palubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomír Mech et Przemyslaw Prusinkiewicz. (2009). *Self-organizing tree models for image synthesis*. ACM Transactions on Graphics - TOG, (pp .58–67).

11. Jerry O Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Mech et Vladlen Koltun. (2011). *Metropolis procedural modeling*. Transactions on Graphics – TOG, (pp. 1 - 14).

12. Hastings EJ, Guha RK, Stanley KO. (2009). *Automatic content generation in the galactic arms race video game*. (pp. 245-263). IEEE Trans Comput Intell AI Games 1(4). New York: IEEE Press.

13. Tobias Joppen, Miriam Moneke, Nils Schroder, Christian Wirth, and Johannes Fürnkranz. (2018). *Informed Hybrid Game Tree Search for General Video Game Playing*. (pp. 78–90). IEEE Trans. on Games, 10(1).

14. Olve Drageset, Mark H.M. Winands, Raluca D. Gaina and Diego Perez-Liebana. (2019). *Optimising Level Generators for General Video Game AI*. IEEE Conference on Games (CoG).

15. Toy, M, Wichman, G., Arnold, K., and Lane, J. (1980). *Rogue*. PC Game.

16. Blizzard Entertainment. (1997). *Diablo*. Video Game. Blizzard Entertainment: Irvine, Californie, États-Unis.

17. Mossmouth. (2008). *Spelunky*. Video Game. Mossmouth: San Francisco, Californie, États-Unis.

18. Evolutionary Games. (2010) Galactic Arms Race. Video Game.

19. Valve Corporation. (2008). *Left 4 Dead*. Video Game. Valve Corporation: Bellevue, Washington, États-Unis.

20. Blizzard Entertainment. (1998). *Starcraft*. Video Game. Blizzard Entertainment: Irvine, Californie, États-Unis.

21. Nintendo. (1983). *Super Mario Bros*. Video Game. Nintendo: Kyoto, Japon.

22. id Software. (1993). *Doom*. Video Game. id Software: Richardson, Texas, États-Unis.

23. Rovio Entertainment Corporation. (2009). *Angry Birds*. Video Game. Rovio Entertainment Corporation: Espoo, Finlande.

24. S. Risi, J. Lehman, D B D'Ambrosio, R. Hall, and Kenneth O Stanley. (2015). *Petalz: Search-based procedural content generation for the casual gamer*. Compu-tational Intelligence and AI in Games, IEEE Trans-actions on.

25. Gillian Smith. (2014). *The future of procedural content generation in games*. Notheastern University. AIIDE Workshop.

26. Sebastian Risi and Julian Togelius. (2019). *Procedural Content Generation: From Automatically Generating Game Levels to Increasing Generality in Machine Learning*. University of Copenhagen.

27. Ripamonti L.A., Mannalà M., Gadia D., Maggiorini D. (2016). *Procedural content generation for platformers: designing and testing FUN PLEDGE*, Springer Science and Business Media, New York

28. Martinho C., Bicho F. (2018). *Multi-dimensional Player Skill Progression Modelling for Procedural Content Generation.* FDG '18: Proceedings of the 13th International Conference on the Foundations of Digital Games