

De Batman à Star Wars, de nombreuses franchises étendent leur univers au jeu vidéo. Il est donc tout à fait naturel que ma série sur les maths au cinéma engendre elle aussi son spin-off vidéoludique. Souhaitez donc la bienvenue au ChouxRom JV Club.

****générique avec une musique sympa****

Quand il est question de faire des liens entre jeux vidéo et mathématiques, j'ai tout de suite plusieurs approches qui me viennent en tête. Je pourrais bien sûr parler de l'aspect programmation. Si vous avez déjà essayé de coder des collisions entre des disques, vous savez que vous ne pouvez pas vraiment vous en sortir sans un minimum de maîtrise de trigonométrie ou d'algèbre linéaire. Pour maîtriser l'aléatoire dans un jeu vidéo, il faut connaître un minimum les probabilités, et pour comprendre comment cet aléatoire est généré, il vous faudra des bases d'arithmétique. Et je ne parle pas des maths que l'on peut trouver à l'intérieur des moteurs physique et graphique de n'importe quel triple A. Il y aurait d'ailleurs de quoi faire une vidéo complète sur la façon dont un algorithme qui calcule rapidement l'inverse d'une racine carrée a permis en son temps aux développeurs de Quake III d'améliorer sensiblement leur rendu visuel.

Mais ce n'est pas exactement ça qui va m'intéresser aujourd'hui. Ce dont j'ai envie de parler, c'est du dialogue qui peut exister entre les mathématiques et le jeu vidéo : les concepts mathématiques qui apparaissent dans les jeux, et, en réponse, ce que les mathématiciens ont d'intéressant à raconter sur les mathématiques présentes dans les mécaniques. Et, en tant que fanboy de Nintendo depuis ma première GameBoy en 1997, on va se concentrer uniquement à ce que l'on peut trouver de mieux dans les Mario, Pokemon et autres Zelda. Et, spoiler, certaines questions sont liées à un problème mis à prix à un million de dollars.

Partie I - Quand Nintendo s'intéresse aux maths

Bien sûr, dans leur immense majorité, les jeux vidéo prennent place dans des mondes en 2D ou en 3D, et les propriétés des figures géométriques du plan ou de l'espace font partie des premières découvertes mathématiques de l'histoire de l'humanité. Je ne vais donc pas vous faire l'affront de vous parler de la présence de cubes dans l'univers de Mario. Mais on peut malgré tout évoquer d'autres figures géométriques.

Tiens, par exemple, vous avez peut-être déjà remarqué que dans le logo de Mario Kart 8, le design du 8 n'a pas été choisi au hasard. Il s'agit d'un ruban de Mobius, un objet topologique que j'ai très souvent évoqué sur cette chaîne. Pour le construire, on part d'un rectangle allongé, comme un ruban de papier par exemple, et on va le recoller sur lui-même de manière à former une boucle, mais après lui avoir fait faire un demi-tour. L'objet que l'on obtient a alors la particularité de n'avoir qu'une seule face. En effet, quand on se déplace le long de la surface de ce ruban de Mobius, on finira par revenir à son point de départ, mais de l'autre côté de la surface. Il faut alors faire un tour supplémentaire pour vraiment revenir à son point de départ. Dans Mario Kart 8, le premier circuit de la coupe Fleur, c'est le circuit Mario, et le tracé de ce circuit a justement son tracé en forme de ruban de Mobius. L'une des nouveautés de Mario Kart 8 étant la présence d'antigravité, le ruban est alors le prétexte parfait pour créer un circuit où les personnages devront rouler aussi bien au-dessus qu'en dessous de la même route, tout en revisitant le traditionnel circuit en 8 que l'on retrouve dans de nombreuses autres courses.

La présence d'une référence mathématique aussi explicite dans le jeu le plus vendu de la Switch peut sembler surprenante. Ce n'est pourtant pas la première fois que Nintendo a voulu convertir les plus jeunes joueurs aux mathématiques. Mais pour vous en parler dans le détail, je vais plutôt laisser ma place à Mister Flech, le spécialiste Nintendo du Youtube francophone.

Le jeu d'arcade Donkey Kong, sorti en 1981, fut le tout premier jeu dans lequel est apparu Mario, et il était surnommé à l'époque Jumpman. Ce jeu a eu une suite : Donkey Kong Jr, dans lequel notre plombier joue cette fois-ci le rôle du méchant, et c'est à Donkey Kong Jr de délivrer son père emprisonné. Mais saviez-vous qu'il y a eu en 1983 sur NES un jeu ludo-éducatif avec ce personnage nommé Donkey Kong Jr. Math ? Dans le premier mode, Calculate A, Donkey Kong montre un nombre sur sa pancarte. A l'aide des chiffres et des opérateurs mathématiques sur le terrain, vous devrez réaliser un calcul vous permettant d'obtenir cette valeur. Dans Calculate B, les nombres sont complètement mélangés et vous commencez le calcul avec une valeur non nulle, ce qui rend les choses un peu plus difficiles. Quant au dernier mode, il vous propose de vous exercer sur différents calculs en allant chercher les chiffres vous permettant d'écrire le bon résultat. Si ce jeu a été développé officiellement par Nintendo, ce n'est pas spécialement le cas du prochain jeu éducatif avec Mario : Mario's Early Years! Fun with Numbers, et honnêtement, ça se sent. Sorti sur MS-DOS et sur NES, ce jeu fait parti de la trilogie de jeux éducatifs : Mario's Early Years ! Le principal but des différents mini-jeux est d'apprendre à reconnaître les nombres de 1 à 10, mais également des instructions spatiales ou des formes géométriques. On est donc sur un jeu totalement destiné aux enfants de moins de 6 ans.

Difficile également de passer à côté de Programme d'entraînement cérébral du Dr. Kawashima qui consacre une grande partie de ses exercices au calcul mental, avec parfois certaines variantes.

Mais même dans les jeux sans promesses éducatives, on retrouve malgré tout à quelques occasions des références aux maths. J'aime beaucoup ce passage de Pokémon Écarlate et Violet, dans lequel Thaïm, professeur de mathématiques à l'université Orange ou Raisin, explique le principe des coups critiques et enseigne alors les bases des probabilités. Elle introduit alors son cours en évoquant un paradoxe bien connu, le paradoxe des anniversaires : dans un groupe de 40 personnes, la probabilité de trouver deux personnes qui partagent la même date d'anniversaire n'est pas de $40/365$ comme on pourrait le penser au premier abord, mais est en réalité de presque 90%. Ce résultat peut se calculer relativement facilement, en procédant par étape. Pour cela, on commence avec deux personnes. En ignorant les années bissextiles et en supposant que toutes les dates d'anniversaire sont équiprobables, la probabilité qu'elles aient la même date d'anniversaire est de $1/365$, et donc, la probabilité qu'elles n'aient pas la même date est de $364/365$. On reste dans ce cas-là, et on amène une troisième personne. La probabilité qu'elle n'ait aucun anniversaire commun avec les deux personnes déjà présentes est donc de $363/365$. La probabilité que trois personnes n'aient aucun anniversaire commun est alors de $364/365 * 363/365$. En poursuivant le raisonnement, on peut montrer que la probabilité que 40 personnes aient un anniversaire différent égale à $364/365 * 363/365 * ... 326/365$, soit 10.88%. La probabilité qu'il y ait donc au moins un anniversaire en commun dans ce groupe de 40 personnes est donc alors de 89.12%. On peut sentir intuitivement ce résultat par le fait que dans un groupe de 40 personnes, il y aura 780 couples possibles. Bien que chaque

couple ait une probabilité d'anniversaire commun très faible, leur grand nombre fait grandir très vite les probabilités. De la même façon, on peut montrer qu'à partir de 111 personnes, la probabilité qu'il n'y ait pas d'anniversaire en commun est grosso modo égale à la probabilité de gagner le gros lot à l'Euromillions.

Plus anecdotique, on peut retrouver des équations dans certains jeux.

C'est le cas par exemple dans le décor du premier niveau du premier monde de Super Paper Mario sur Wii. Même s'il y a quelques symboles fantaisistes liés à l'univers Mario, on y retrouve des symboles courant dans l'usage mathématique, comme les opérateurs de calcul, les symboles de comparaison, les barres de fraction, une racine et des puissances, même si certaines formules semblent totalement improbables.

Autre exemple dans le décor du Kollege Kong de Donkey Kong Country 2 dans lequel on peut voir différents calculs qui sont absolument tous faux ! D'ailleurs, si la version Super NES comporte l'intitulé EXAM, dans le remake sur GBA, il y aura marqué SUMS, sauf qu'il n'y a clairement qu'une seule addition au tableau !

Dans WarioWare Gold sur 3DS, le personnage 9-Volt éprouve beaucoup de difficultés en cours de maths. Heureusement, une petite créature nommée Fronk arrive à lui faire pratiquer des calculs en les transposant à des situations de jeux vidéo, comme par exemple un combat de RPG où l'attaque d'un ennemi enlève un certain nombre de PV au héros. Et ce n'est pas une si mauvaise idée d'illustrer des cas mathématiques dans un contexte plus concret lié à une passion.

Un dernier exemple avec Mario Party DS, dans lequel les personnages de l'univers Mario ont tous été miniaturisés. Dans le mini-jeu nommé La Brosse des maths, chaque personnage possède un aimant dans son dos. Le but est de s'accrocher au tableau le plus proche possible de la ligne rouge sans la dépasser. Et en guise d'habillage, le tableau est rempli d'inscriptions à la craie comme des formules physiques ($E = mc^2$), des représentations de molécules chimiques mais aussi un cercle trigonométrique, le théorème de Pythagore, et des approximations de racines carrées.

Et je passe sur de nombreuses autres références que l'on peut trouver dans Mario Party Advance, WarioWare: Touched! ou Yoshi's Island par exemple. Les mathématiques ont donc souvent inspiré les développeurs de Nintendo.

Quand les maths inspirent le JV : Mario Party 10

Que les jeux vidéo fassent des références au mathématiques, c'est une chose, mais que des mathématiciens fassent de Zelda ou de Pokémon le sujet d'un article de recherche, c'est déjà nettement moins courant. Il me vient alors le nom de Andrew Misseldine, professeur de mathématiques à la Southern Utah University. Au milieu de ses articles de recherche sans doute très intéressants sur les anneaux de Schur, on peut trouver un article de 17 pages appelé " Les mathématiques de Mario Party 10", publié en 2019. Forcément, ça dénote un peu. Ce petit article sans grande prétention ne parle pas vraiment de l'ensemble de Mario Party 10, mais d'un seul de ses mini-jeux, le loto destructeur de Bowser.

Dans ce mini-jeu, les différents joueurs doivent choisir une grille carrée de loto de 9 cases sur lesquelles figurent des ennemis de Mario : goomba, bob-omb, boo, etc. Une fois la grille sélectionnée, le joueur qui incarne Bowser lancera 5 fois le même dé à six faces sur lequel on retrouve les ennemis présents sur les grilles. Après chaque lancé, toutes les cases qui représentent l'ennemi obtenu sont rayées. Si Bowser obtient un ennemi déjà obtenu auparavant, rien ne se passe. A l'issue des 5 lancers, on fait les comptes : pour chaque ligne horizontale, verticale et diagonale complétée, le joueur perdra un cœur.

La question que se pose alors Misseldine est celle grille qu'il est préférable de choisir. Parmi les 6 grilles proposées aux joueurs, quelle est celle qui minimise le nombre moyen de cœurs perdus.

Par exemple, imaginons que j'ai le choix entre cette grille A, composée de 5 koopas et de 4 bob-ombs, et cette grille B, composée de 8 koopas et d'une seule bob-omb. On peut déjà remarquer que, quelle que soit la grille que je choisis, aucune ligne ne pourra être formée si aucun koopa n'est obtenu par Bowser, peu importe s'il obtient ou non des bob-omb.

Puisqu'à chacun des 5 lancers, la probabilité de ne pas obtenir un koopa est de $5/6$, la probabilité de n'obtenir aucun koopa sur les 5 lancers sera donc de $(5/6)^5$, soit environ 40%. Là où les deux grilles sont différentes, c'est dans le cas où Bowser obtient au moins un koopa mais aucun bob-omb. Dans ce cas, la première grille fera perdre seulement 2 cœurs, quand l'autre en fera perdre le double. Pour calculer la probabilité de cet événement, il va falloir envisager tous les scénarios, et on va représenter ça par un tableau. Sur l'ensemble de ses lancers, Bowser peut obtenir ou bien 0 koopas, ou bien au moins 1. De même, il peut obtenir ou bien 0 bob-ombs, ou bien, au moins 1 bob-omb. Ce que l'on a déjà calculé, c'est la probabilité d'obtenir 0 koopa, qui est de 40%. Par un raisonnement strictement identique, on peut aussi calculer que la probabilité d'obtenir 0 bomb-omb est aussi de 40%. Mais ce que l'on veut plutôt calculer, c'est la probabilité de cette case-là, celle d'avoir 0 bob-omb et au moins 1 koopa. Pour obtenir ce résultat, on va pour cela plutôt calculer la probabilité de la case d'à côté. Sur chaque lancé, la probabilité de n'obtenir ni koopa, ni bob-omb est de $4/6$. Sur 5 lancers, cette probabilité devient donc $(4/6)^5$, soit environ 13%. On en déduit alors que la probabilité d'obtenir aucun bomb-omb mais au moins un koopa est de $40\% - 13\%$, soit 27%.

En complétant le tableau, on peut aussi calculer que la probabilité d'obtenir au moins un koopa et au moins un bomb-omb est d'environ 33%, ce qui correspond au cas où les grilles sont entièrement complétées, et font perdre 8 cœurs.

Si on résume, la probabilité d'avoir 0 koopa est de 40%, celle d'avoir au moins 1 koopa et 0 bomb-omb est de 27%, et celle d'avoir au moins un koopa et au moins un bob-omb est de 33%. Pour la grille A, le nombre de cœurs perdus sera respectivement de 0, 2 et 8, ce qui donne une espérance, une moyenne, de 3.2 cœurs perdus. Pour la grille B, le nombre de cœurs perdus sera respectivement de 0, 4 et 8, soit une espérance de 3.7 cœurs perdus. La grille A est donc légèrement plus intéressante que la grille B.

On peut faire les mêmes calculs pour des grilles avec davantage de symboles différents, et c'est ce à quoi s'attelle Misseldine dans son article. Grâce à cette horrible formule qu'est le principe d'inclusion-exclusion, il est possible de calculer les probabilités de toutes les configurations de remplissage de grille. On peut alors en déduire l'espérance du nombre de cœurs perdus pour chacune des grilles du jeu. La conclusion de l'article après plusieurs pages de raisonnements et de calculs, c'est que les meilleures grilles à choisir sont celles qui ont un maximum de symboles différents. Ce résultat ne défie pas vraiment l'intuition que l'on pouvait avoir, mais au moins, les calculs sont là pour le confirmer.

7 - Les travaux de Erik Demaine

Mais en réalité, ce n'est pas sur les travaux de Andrew Misseldine que je voulais m'appesantir, mais sur ceux d'un autre mathématicien, Erik Demaine. Avant de faire mes recherches sur les liens entre Nintendo et les mathématiques, je ne connaissais Demaine que de par son théorème sur la découpe d'origami, le théorème fold-and-cut. Petite parenthèse. Dans son livre Paper Magic publié en 1922, le célèbre magicien Harry Houdini décrit une méthode qui permet de découper une étoile à 5 branche dans une feuille de papier et à l'aide d'un seul coup de ciseaux. L'astuce est de plier la feuille de façon ingénieuse et de bien choisir la ligne de découpe. D'autres magiciens perfectionnent la technique, jusqu'à ce que, en 1960, Martin Gardner découvre ces travaux et pose la question suivante : quelles sont les formes qu'il est possible d'obtenir après avoir découpé d'un seul coup de ciseaux une feuille préalablement pliée. La réponse sera donnée en 1998 par trois mathématiciens canadiens : Erik Demaine, alors âgé de 17 ans, Martin Demaine, son père, et Anna Lubiw, directrice de thèse de Erik. Parce que, oui, à 17 ans, il était déjà en train de faire sa thèse, vu qu'il a décroché son bachelor à 14 ans. Bref. Ils démontrent alors que par de simples pliages et un unique coup de ciseau, il est possible de découper n'importe quelle forme, tant que celle-ci est composée de segments de droites. Les formes peuvent être non convexes, non connexes ou même imbriquées, le découpage restera quoi qu'il arrive toujours réalisable. On n'est cependant pas à l'abri d'une construction qui demande des centaines de pliages. Demaine ne s'arrêtera pas à cet article sur le pliage, puisqu'il a aussi écrit à propos de la construction de labyrinthes en origami, ou sur la difficulté mathématique qu'il existe à replier une carte Michelin ou une notice de médicament.

Aujourd'hui, Erik Demaine est professeur de sciences informatiques au MIT, et parmi les sujets auxquels il s'intéresse, les jeux vidéo ont une place de choix. Il a ainsi à son actif des articles sur Tetris, the Witness, Cookie Clicker, SnipperClips, Bust a Move ou même Celeste. Peut-être en reparlerai-je un jour dans une autre vidéo, car je voudrais détailler un peu plus son article de 2014 "les jeux classiques de Nintendo sont difficiles, algorithmiquement parlant". Dans cet article coécrit avec Greg Aloupis, Alan Guo et Giovanni Viglietta, Demaine détaille en quoi, d'un point de vue de la théorie de la complexité, on peut considérer comme difficiles les jeux 2D des franchises Pokémon, Zelda, Donkey Kong, Metroid et bien sûr Mario.

8 - P vs NP

Avant de détailler plus précisément ce qu'ils ont bien ou raconter sur ces jeux Nintendo, il va falloir parler dans les grandes lignes de théorie de la complexité. Ça va être l'occasion de parler d'un problème du millénaire que je n'ai encore jamais vraiment approfondi sur cette chaîne : le problème P vs NP. C'est un problème que l'on résume parfois par la question : existe-t-il des algorithmes rapides pour résoudre les problèmes les plus difficiles. Cette formulation est plutôt réductrice, donc je vais essayer de préciser un peu plus les contours de la question.

Pour parler du problème P vs NP, il faut introduire la notion de problème de décision. Un problème de décision, c'est simplement une question mathématique dont la réponse est soit oui, soit non. Par exemple : sur cette carte de restaurant, y a-t-il un plat adapté à un régime végétarien ? Ce niveau de Mario peut-il être terminé ? Peut-on résoudre ce sudoku ? Ce nombre est-il un nombre composé ? La question qui intéresse les chercheurs, c'est de savoir à quel point ces problèmes pourront être rapides à résoudre, de les classer en fonction

de leur complexité à être efficacement par un bon algorithme. Pour comprendre ça, prenons le premier problème.

Vous avez une carte de pizzeria avec 10 pizza, et chaque pizza n'a jamais plus que 8 ingrédients. Pour savoir si au moins l'une des pizza est adaptée à votre régime, la procédure à suivre, l'algorithme à suivre, c'est de lire la composition de chaque pizza, et de s'arrêter dès que l'une d'elle convient. On va alors compter le nombre d'instructions élémentaires à faire pour mener à bien cette mission. Ici, on va appeler instruction élémentaire la lecture d'un ingrédient. Dans le pire des cas, pour une carte non végétarienne avec des pizzas un peu trop fournies, il faudra lire 10×8 soit 80 ingrédients. Maintenant, si, au lieu d'une carte avec 10 pizzas, on donne une carte avec 100 pizzas, toujours avec au plus 8 ingrédients. En suivant la même procédure, le pire des cas demandera la lecture de 800 ingrédients. On peut alors généraliser : pour la donnée d'une carte de N pizzas, il faudra lire $8 \times N$ ingrédients. Le nombre d'instructions élémentaires pour résoudre ce problème de décision est donc proportionnel à la taille de la donnée.

Quand la résolution d'un problème de décision passe par un algorithme qui, dans le pire des cas, demande un nombre d'instruction élémentaires proportionnel à la taille de ses données ou à une de ses puissances, on rangera ce problème dans ce que l'on appelle la classe P : P comme polynomial. On y range donc tous les problèmes qui peuvent être résolu par un algorithme polynomial. On peut, d'une certaine façon, considérer que ce sont des problèmes qui sont faciles à résoudre, ou, du moins, que l'on peut résoudre d'une façon efficace même même lorsque la taille des données augmente. Mais ce n'est pas le cas de tous les problèmes.

Prenons cet autre problème de décision : on considère un nombre entier X dont la taille est de, disons, 4 chiffres. Ce nombre est-il un nombre composé, c'est-à-dire, peut-il être écrit sous la forme d'un produit de deux nombres plus petits. Pour ce nombre, la réponse est oui, puisqu'il est divisible par 17, et pour celui-là, la réponse est non, puisqu'il n'a aucun diviseur, c'est un nombre premier. La procédure naïve à suivre, c'est de prendre chaque entier plus petit que X , et de tester par une division si il s'agit d'un diviseur de X . Puisque le nombre a 4 chiffres, on aura dans le pire des cas, une dizaine de milliers de divisions à faire. Si le nombre avait 5 chiffres, il aurait fallu une centaine de milliers de divisions, et, de façon plus générale, si le nombre à tester s'écrit avec N chiffres, le nombre de divisions requises par cet algorithme naïf sera dans le pire des cas proportionnel à 10^N , ce qui augmente très vite. On peut facilement améliorer cet algorithme, mais quoiqu'il arrive avec cette approche, le nombre d'instructions élémentaires finira par exploser. Cet algorithme-là n'est donc pas assez efficace, il ne permet pas de classer le problème dans la classe P . Cependant, même si le problème est difficile à résoudre, il est facile de vérifier une solution quand la réponse est oui. Si je vous donne ce grand nombre X , vous pourrez facilement vérifier que ce nombre p est un diviseur de X , en faisant simplement une division. Le problème est donc d'un côté difficile à résoudre, mais de l'autre, il est facile de vérifier une réponse. Un problème qui vérifie ce deuxième critère, être facile à vérifier, est à ranger dans ce que l'on appelle la classe NP . NP , ça veut dire non-déterministe polynomial. La définition précise de la classe NP est en réalité un peu plus subtile que ça, mais on va garder en tête qu'il s'agit des problèmes faciles à vérifier quand la réponse est oui, faciles dans le sens où il existe un algorithme polynomial qui y répond.

Savoir si un nombre est composé est donc un problème NP , et l'algorithme que j'ai présenté ne permet pas de le ranger dans les problèmes de classe P . Deux précisions importantes cependant. Savoir si un nombre est composé ou premier, c'est un problème que l'on sait

aujourd'hui résoudre de façon plutôt rapide, mais seulement grâce à des algorithmes non-déterministes, qui font appel au hasard. La présence de hasard n'est pas autorisée dans la définition d'un problème de classe P. Malgré tout, trois chercheurs indiens ont découvert en 2002 qu'il existe un algorithme déterministe et polynomial qui permet de décider si un nombre est composé ou premier. Le problème de savoir si un nombre est composé est donc en réalité de classe P. Seulement, l'algorithme qui a été découvert est inutilisable en pratique, puisque le nombre d'instructions élémentaires est proportionnelle à la 17^e puissance de la taille des données. Pour savoir si un nombre est premier, on préférera l'approche probabiliste.

Bref. Ce qui est facile à affirmer, c'est que lorsqu'un problème est de classe P, il est forcément de classe NP : si un problème est facile à résoudre, il est facile de vérifier sa réponse. La question inverse, c'est sa réciproque. Un problème de classe NP est-il toujours de classe P : si un problème a une solution que l'on peut facilement vérifier, est-il toujours possible de trouver un algorithme qui le résout de façon efficace ? C'est ça la question que pose le problème P vs NP.

Il y a alors deux possibilités. Soit P est différent de NP, soit $P = NP$. Pour prouver la première hypothèse, il faudrait trouver un problème de classe NP qui ne soit pas dans P, trouver donc un problème pour lequel il faudrait rigoureusement démontrer qu'aucun algorithme polynomial n'est capable de le décider. C'est aujourd'hui l'hypothèse privilégiée parmi les chercheurs en informatique.

L'autre hypothèse, c'est que P est en réalité égal à NP. Et pour le prouver, il faudrait montrer que chaque problème NP est aussi un problème P, c'est-à-dire, trouver un algorithme polynomial pour chaque problème NP qui pourrait exister. Il est bien sûr inenvisageable de trouver des algorithmes au cas par cas, puisque des problèmes NP, il en existe une infinité. Heureusement, il y a une solution : des problèmes de classe NP particuliers auquel n'importe quel autre problème NP peut se ramener. On parle de réduire un problème à un autre, mais je ne détaillerai pas la définition formelle. Ces problèmes particuliers sont donc des problèmes qui portent donc en eux l'essence de l'ensemble des problèmes NP. On les appelle les problèmes NP-complets. Si on parvient à trouver un algorithme polynomial qui résout un seul problème NP complet, alors on pourra en déduire une solution pour tous les autres problèmes NP, et ça montrerait que $P=NP$.

Des problèmes NP qui sont NP-complets, on en connaît beaucoup. Un exemple simple est celui du sudoku : est-il possible de résoudre une grille de sudoku de taille $n^2 \times n^2$. Ainsi, si vous trouvez un algorithme efficace pour résoudre n'importe quel sudoku, alors vous aurez résolu le problème P vs NP.

9 - Le problème 3SAT

Le problème NP-complet qui va nous intéresser pour comprendre les travaux de Demaine sur les jeux Nintendo est un peu plus abstrait, il s'agit du problème 3-SAT. SAT comme satisfaisabilité. Il pose la question suivante : étant donné une forme normale conjonctive, est-il possible d'assigner des valeurs booléennes à ses variables qui la rende logiquement vraie ? Je vais détailler un peu. On dispose de plusieurs variables booléennes, c'est-à-dire, des variables qui ne peuvent prendre que deux valeurs, VRAI et FAUX. Pour l'exemple, on prendra 4 variables A, B, C et D. A l'aide des opérateurs logiques ET, OU et NON, on peut construire des expressions, comme par exemple $A \text{ ET } (B \text{ OU Non } C)$. Pour rendre cette expression soit vraie, il faut donc que la variable A prenne la valeur vraie et que, soit B soit vrai, soit C soit faux, ou éventuellement les deux), et on peut assigner à D n'importe quelle

~~valeur~~. On dit qu'une expression logique est sous forme normale conjonctive quand les variables ou leur négations sont regroupées dans des parenthèses de "ou" entrecoupées par des "et". L'expression suivante

$(A \text{ ou } B \text{ ou } \neg D) \text{ et } (\neg A \text{ ou } \neg C \text{ ou } \neg D) \text{ et } (A \text{ ou } C \text{ ou } D) \text{ et } (\neg A \text{ ou } B \text{ ou } \neg C) \text{ et } (B \text{ ou } \neg C \text{ ou } \neg D)$

est un exemple de forme normale conjonctive. Elle est construite à partir de 4 variables. Une paire de parenthèse est appelée une clause. L'expression est donc constituée de 5 clauses avec au plus 3 variables par clauses. Le problème 3-SAT demande justement à ce qu'il n'y ait jamais plus de 3 variable par clause. Est-il alors possible de trouver une valeur logique pour chacune des 4 variables de manière à rendre l'expression vraie ?

Dans ce cas, c'est effectivement possible, en prenant par exemple les variables $A=\text{Vrai}$, $B=\text{Vrai}$ et $C=\text{Faux}$ et $D=\text{Faux}$. Avec ce choix-là, on peut vérifier que dans chacune des 5 clauses, il y aura au moins un VRAI. Pour qu'une expression sous forme normale conjonctive soi vérifiée, il faut que chaque clause soit vérifiée, ce qui est ici le cas.

L'expression complète est donc bien vérifiée. Pour aboutir à cette solution, la façon la plus simple de procéder est d'essayer les 16 combinaisons possibles jusqu'à tomber sur celle qui fonctionne. Lorsque le nombre de variables augmente, le nombre de combinaisons explose, ce qui laisse penser que le problème n'est pas de classe P. Cependant, la solution est simple à vérifier, ce fait de ce problème 3-SAT un problème NP. Et c'est même un problème NP-complet, puisqu'on peut montrer que tout problème NP peut se réduire à celui-ci. ~~Ce qui nous amène donc à parler des travaux sur les jeux classique Nintendo.~~

10 - Les jeux classique Nintendo sont difficiles

Revenons donc à l'article de nos quatre mathématiciens. Le problème de décision qui va maintenant nous intéresser est le suivant : étant donné un donjon créé à partir du moteur de The Legend of Zelda : A link To the past, est-il possible d'aller d'un point de départ A jusqu'à un point d'arrivée B ? Cette question pourra s'adapter aux niveaux des Mario Bros 2D, aux autres jeux Zelda, aux RPG Pokémon 2D, aux jeux Metroid et aux trois premiers Donkey Kong Country. Ce que démontrent Aloupis, Demaine, Guo et Viglietta, c'est que ces problèmes de décision peuvent tous être classés comme étant des problèmes NP-difficiles. On dit qu'un problème est NP-difficile quand il est au moins aussi difficile à résoudre qu'un problème NP-complet. Un problème NP-difficile peut être un problème NP, mais rien n'empêche qu'il soit encore plus complexe qu'un problème NP. Oui, parce que les classes de complexité sont loin de ne s'arrêter qu'aux classes P et NP. Bref.

La façon la plus simple de montrer qu'un problème est NP-difficile, c'est de montrer qu'il est possible d'y modéliser un problème NP-complet, et c'est pour cela que j'ai parlé du problème 3-SAT. Si on arrive à fabriquer un problème 3-SAT à l'aide du moteur de Zelda, ça prouvera que le jeu est NP-difficile.

Eh bien, allons-y. Transformons un problème 3-SAT en donjon de Zelda, un donjon dont le level design obligera Link à assigner des variables et à vérifier des clauses. On va prendre pour cela l'exemple de la forme normale conjonctive suivante $(A \text{ ou } \neg B \text{ ou } C) \text{ et } (\neg A \text{ ou } B \text{ ou } C) \text{ et } (A \text{ ou } \neg B \text{ ou } \neg C)$. Trois variables, trois clauses.

Le schéma à suivre est le suivant. Il nous faut déjà un point de départ et un point d'arrivée. On va avoir besoin de quelque chose pour modéliser les variables A, B et C, ce que je vais appeler des modules 'assignation de variable, et il faudra aussi modéliser l'expression

logique, ce que l'on va appeler des modules de clause. Dans un premier temps, notre avatar doit passer successivement, et sans possibilité de retour en arrière, par trois modules d'assignation de variables. Chacun de ces modules consiste à choisir un chemin, ce qui correspondra à assigner une valeur logique Vrai/Faux à chacune des variables A, B et C. Lorsqu'une valeur "Vraie" est choisie pour une variable, elle doit donner l'accès aux modules de clauses correspondants. Par exemple, si je choisis le chemin VRAI de la variable A, je dois avoir accès au premier et au dernier module de clause. Résoudre le problème 3-SAT, c'est donc trouver un chemin qui visitera au moins une fois chaque module de clause. Une fois que les variables ont été affectées, il va falloir que Link traverse les trois modules de clauses, ce qui ne peut être possible qu'à la condition que chaque module de clause ait été visité.

Ca, c'est la théorie. En pratique, il nous faut donc construire, à l'aide des éléments de gameplay de Zelda, chacun de ces modules. Pour cela, on utilisera essentiellement deux mécaniques : celle qui permet à Link de pousser des blocs et celle du grappin, qui permet à Link de rejoindre en ligne droite un coffre ou un bloc. On considère de plus que Link n'a aucun moyen de sauter au-dessus d'un vide, et que le grappin ne traverse pas les murs.

Pour les modules de départ et d'arrivée, pas de problème. On considère que Link entre dans la salle depuis une porte, et cherche à atteindre l'autre porte. On veillera cependant à placer un sol destructible au début de la pièce. En effet, si Link tombe dans un précipice, il sera téléporté en début de salle mais ne doit pas être en mesure de retenter la salle sans la réinitialiser.

Pour les modules d'assignation des variables, on doit faire faire à Link un choix binaire, qui correspondent aux valeurs booléennes Vrai ou Faux. Pour modéliser cela, on oblige Link à devoir choisir entre deux chemins qui s'offrent à lui, sans possibilité de retour en arrière. Le grappin va permettre cela. Quand Link arrive en haut, il doit choisir, grâce au grappin, l'un des deux chemins proposés en s'agrippant à l'un des deux coffres. Disons que le chemin de gauche correspond à assigner à la variable la valeur Vraie, et le chemin de droite la valeur Faux.

Ce module donne également une façon de construire des passages à sens unique, en plaçant un coffre de l'autre côté d'un précipice.

Il ne reste qu'à construire les modules de clause, qui ont une entrée, une sortie et 3 accès qui correspondent aux variables A, B et C. On va construire ce module à l'aide des blocs à pousser. Link entre dans ce module par la droite, et doit atteindre la sortie à gauche. Pour passer, il faut que au moins l'un des blocs ait été poussé, afin qu'elle puisse être atteinte à l'aide du grappin. On doit seulement s'assurer que chaque bloc correspond à l'une des trois variables de la clause concernée, les chemins y menant étant reliés à leur variable correspondante.

On a donc tout ce qu'il faut pour transformer l'expression logique en niveau de Zelda. Il reste un petit détail : les chemins qui relient les modules de variables aux modules de clauses ont le mauvais goût de se croiser. Il faut donc trouver un moyen de construire des intersections qui empêche Link de passer d'un chemin à l'autre. On doit donc construire un module de croisement à placer dès que deux chemins se croisent. Grâce à quelques coffres bien placés et l'utilisation du grappin, on peut vérifier que si Link entre à l'ouest, il ne pourra que ressortir à l'est. Et inversement.

Il ne reste plus qu'à associer tout cela, et vous avez la modélisation d'un problème 3-SAT sous la forme d'un donjon Zelda. Ainsi, quand Link entre dans la pièce, il doit commencer par choisir le chemin Vrai ou le chemin Faux. En choisissant Vrai, il a accès à la première clause et à la dernière, dans lesquelles il va pouvoir pousser un bloc. Link doit ensuite se rendre au module de la variable B, et il choisit par exemple le chemin Faux. Ce chemin donne accès là aussi à la première et à la dernière clause. Enfin, en arrivant au module de la variable C, il choisit le chemin VRAI, ce qui lui donne l'accès à la première et à la deuxième clause. Link vient donc d'assigner aux variables A B et C les valeurs Vrai, Faux et Vrai. On remarque alors que chaque clause a été visitée au moins une fois, et les blocs qui ont été poussés donnent bien l'accès à la porte d'arrivée.

On peut généraliser la méthode à des expressions logiques avec autant de variable et autant de clauses que l'on veut, ce qui implique donc que n'importe quel problème 3-SAT imaginable pourra être modélisé par un donjon Zelda. Le jeu peut donc être aussi difficile qu'un problème NP-complet : il est NP-difficile.

L'avantage de la méthode que je viens de vous présenter, c'est qu'on peut la réutiliser pour démontrer la NP-difficulté de n'importe quel jeu, pourvu que l'on réussisse à créer un module de variable, un module de clause, un module de croisement et des chemins à sens unique. Et ça, nos quatre chercheurs montrent que c'est faisable dans tous les Mario 2D, dans les jeux Pokemon, dans les 3 premiers Donkey Kong Country et dans les jeux Metroid. Je vous renvoie à leur article si vous voulez en saisir toutes les subtilités.

Sauf que Demaine ne s'est pas arrêté à ça. L'année suivante, il publie avec quatre autres chercheurs l'article "Mario Kart est difficile", dans lequel ils démontrent la NP-difficulté du mode contre-la-montre de tous les jeux Mario Kart.

Le summum sera atteint dans un article publié en 2021, The Legend of Zelda: The Complexity of Mechanics. Avec l'aide de 6 autres chercheurs, il décortique et classe l'intégralité des mécaniques qui apparaissent dans les 19 jeux de la saga Zelda, du Zelda Nes de 1989 jusqu'à Breath of the Wild. On y apprend par exemple qu'une énigme de A Link to the Past basée sur les Kodongos sera NP-difficile, tandis que le grappin de Oracle of Ages permet de créer des énigmes de complexité P. Une telle exhaustivité ne peut que forcer le respect et l'admiration.

Mario est indécidable + conclusion

Mais on peut aller encore plus loin, et c'est l'objet de l'article écrit en 2024 par le MIT Hardness Group duquel Erik Demaine fait parti, un article titré "Vous ne pouvez pas résoudre ces niveaux de Super Mario". Il y est question des niveaux que l'on pourrait construire dans un Mario Maker idéalisé qui permettrait de placer d'ennemi que l'on voudrait. Ce que le groupe du MIT démontre, c'est que non seulement ces niveaux sont NP-difficiles, ça, on est habitué, mais qu'ils sont en plus RE-complet. Bon. Je ne vais pas rentrer dans les détails de ce que cela signifie exactement, parce qu'il faudrait pour cela que je détaille les machines de Turing, le théorème de Gödel et ses implications, et vous n'avez vraiment pas envie que je face ça maintenant, cette vidéo est déjà bien assez longue comme ça. Si on doit résumer ce que ça implique, on peut dire qu'il est rigoureusement impossible d'imaginer un algorithme qui prend en entrée un niveau de Mario Maker et qui retourne vrai si il est possible de terminer le niveau, et faux lorsque c'est impossible. Ce problème de décision est

ce que l'on appelle indécidable. ~~Le théorème est vrai pour tous les styles de niveaux créé dans Mario maker 1 et 2, et s'applique aussi aux jeux de la série des New Super Mario Bros.~~

Bref. On peut raisonnablement se demander si tout ça sert vraiment à quelque chose. Bien sûr que ça ne sert à rien de savoir que les énigmes à base de dalles colorées dans Zelda The minish Cap sont NP-complètes ! Enfin, pas complètement non plus. L'étude des jeux vidéo n'est ici qu'un prétexte pour étudier finement comment des mécaniques simples peuvent générer de la complexité. Bien sûr, les chances sont infimes pour que ces recherches fassent progresser les grandes questions ouvertes de la théorie de la complexité dont le problème P vs NP est la partie émergée. Ces question sur ce qu'un algorithme peut ou ne pas faire sont très théorique, mais on n'est jamais à l'abri d'une application pratique, et vu que les applications pratiques en question peuvent éventuellement remettre en cause la sécurité bancaire internationale, ça serait dommage de ne pas se pencher dessus un minimum.

Un grand merci à MisterFlech pour sa participation. Sur sa chaine, il parle de jeux vidéo en général, et des jeux Nintendo en particulier. Je vous invite à regarder sa vidéo sur la récursivité dans les jeux vidéo dans laquelle je fais une petite apparition, et à vous abonner à sa chaine.

Sources

Mathématiques de Quake 3

[Le plus gros hack de l'histoire du jeu vidéo \(1/2\)](#)

Equations chez Nintendo :

[Lineland - Super Mario Wiki, the Mario encyclopedia](#)

[Kong Kollege- Super Mario Wiki, the Mario encyclopedia](#)

[Accountant Assault - Super Mario Wiki, the Mario encyclopedia](#)

[Study Fall - Super Mario Wiki, the Mario encyclopedia](#)

[Roulette - Super Mario Wiki, the Mario encyclopedia](#)

[Maths of Glory - Super Mario Wiki, the Mario encyclopedia](#)

[Mathletes - Super Mario Wiki, the Mario encyclopedia](#)

[Mathemagician! - Super Mario Wiki, the Mario encyclopedia](#)

[Bowser's Bogus Bingo - Super Mario Wiki, the Mario encyclopedia](#)

▶ SNES Longplay [511] Mario's Early Years! Fun with Numbers

▶ WarioWare Gold - Full Story Mode Walkthrough (All Leagues)

Sprites des jeux

- [The Spriters Resource](#)

- [The Textures Resource](#)

Articles :

[The Mathematics of Mario Party 10](#)

Articles de Erik Demaine (et al.) évoqués dans la vidéo

- [Folding and cutting paper.](#)

- [Folding and Unfolding](#)

- [Folding Any Orthogonal Maze](#)

- [When Can You Fold a Map?](#)

- [Classic Nintendo Games are \(Computationally\) Hard](#)

- [Mario Kart is Hard](#)

- [The Legend of Zelda: The Complexity of Mechanics](#)

- [You Can't Solve These Super Mario Bros. Levels: Undecidable Mario Games](#)

- [Snipperclips: Cutting Tools into Desired Polygons using Themselves](#)

- [Bust-a-Move/Puzzle Bobble is NP-Complete](#)

- [Who witnesses The Witness?](#)

- [Cookie Clicker](#)

- [Tetris is NP-hard even with \$O\(1\)\$ rows or columns](#)

- [Celeste is PSPACE-hard](#)

Pliage de papier

[The Fold-and-Cut Problem \(Erik Demaine\)](#)

[MATHEMATICAL GAMES - Recreations involving folding and cutting paper](#)

[Fold-and-cut theorem - Wikipedia](#)

[File:FoldedAntiKoch.gif - Wikimedia Commons](#)

P et NP

[Chapitre 8 : Classes de complexité P et NP](#)

[NP Entertainment System](#)

[Graph of NP-Complete Problems](#)

Indécidabilité

 [A Universal Register Machine in New Super Mario Bros. Wii](#)

Je n'en parlerai pas

Analyse de Twitch Play Pokemon : <https://arxiv.org/pdf/1408.4925>

Musiques

CJVC Theme - Tam

[Bamboo Island \(The Legend of Zelda: Skyward Sword\)](#) - Dj Cutman & Helynt

[Brawl \(Super Smash Bros. Brawl\)](#) - Coffee Date

[Coconut Mall \(Mario Kart\)](#) - Lofi 64

[Dedede theme \(Kirby\)](#) - Tudd & General Offensive

[Dire Dire Docks \(Super Mario 64\)](#) - Lofi 64

[Honeylune Ridge \(Super Mario Odyssey\)](#) - Helynt

[Lost Woods \(The Legend of Zelda: Breath of the Wild\)](#) - Tune in with Chewie

[Planet Wisp \(Sonic ?!\)](#) - Coffee Date

[Pokémon Center Lullaby \(Pokémon\)](#) - Coffee Date

[Rito Village \(The Legend of Zelda: Breath of the Wild\)](#) - Jokabi

[Rosalina in the Observatory \(Super Mario Galaxy\)](#) - Coffee Date

[Spiral Mountain \(Banjo & Kazooie\)](#) - Snore Lax

[Story of the stolen spirits \(Paper Mario\)](#) - Snore Lax

Coupé au montage

Dans le micro-jeu Assaut comptable de WarioWare: Touched! il vous faudra simplement écrire le plus vite possible sur une calculatrice le calcul affiché. Mais si le joueur le souhaite, il peut aussi directement écrire le bon résultat sur la calculette et appuyer sur égal.

Dans la quête Mathémagicien de Mario Party Advance, un Mécha-Koopa nous posera 3 questions de calcul mathématique impliquant une addition et deux multiplications, et sur le tableau derrière lui, on peut voir un triangle sous-entendant un cours de géométrie.

Dans Yoshi's Island, un challenge bonus vous propose de parier un certain nombre de vos vies pour peut-être en gagner plus. Les deux roulettes vont ensuite défiler et s'arrêter sur un opérateur à gauche : soit +, soit x, et sur un nombre à droite pour savoir l'issue du calcul. Si l'addition est le signe le plus sûr, puisque le pire des cas : +0 ne vous fait perdre aucune vie pariée, la multiplication peut autant vous rapporter gros avec un x3 que tout vous faire perdre avec un x0.

Dans le mini-jeu Mathémattaque de Super Mario Party sur Switch, les joueurs s'affrontent en équipe de 2. Chacun doit stopper la roulette de 1 à 6 sur le chiffre le plus haut, et les 2 chiffres sont ensuite multipliés. Le château en pièce de puzzles de l'adversaire reçoit alors autant de projectiles que le produit du calcul. La première équipe qui détruit la fortification de l'autre gagne. C'est donc plus un mini-jeu de réflexes et de chance que de calcul.

On pouvait déjà retrouver un concept similaire dans un mini-jeu 1 vs 3 de Mario Party 5, où, là encore, il fallait stopper 3 dés, et le résultat du calcul correspondait au nombre de pièces que les joueurs gagnaient.

Mais si l'on remonte beaucoup plus loin, en 1980, Nintendo présentait déjà dans son jeu Game & Watch nommé Judge, un concept très intéressant. Il s'agit ici d'un jeu de réflexe où il faut identifier qui de nous ou l'adversaire a le plus grand nombre, et réagir en conséquence. Si notre nombre est le plus petit, il faut appuyer sur le bouton qui permet d'esquiver le coup adverse. Mais si on a le plus grand, il faudra attaquer rapidement avant qu'il esquive !

On peut d'ailleurs retrouver cette attaque dans les jeux Super Smash Bros., où chaque chiffre aura son propre effet sur l'adversaire, le 9 étant, bien évidemment, une attaque critique.

Existe-t-il des maths dans les jeux Nintendo ? Les mathématiciens ont-t-il déjà fait d'un jeu Nintendo leur objet d'étude ? Étonnamment, la réponse est oui !

Animations : Camtasia, Geogebra, Processing

Musiques : Tam

Script, sources, musiques additionnelles :

<https://docs.google.com/document/d/1nhmHvZ6GyKfln6E88kHEdsnb9kZgbrGA4X91Lu1YhI/edit?usp=sharing>

Sous-titres : TBA

Article de E. Demaine : <https://arxiv.org/pdf/1203.1895>

Article de Misseldine :

https://www.researchgate.net/publication/332530498_The_Mathematics_of_Mario_Party_10

Chapitrage :

00:00 Prégénérique

00:12 Générique

00:37 Introduction

01:47 Nintendo et les maths

04:38 Le paradoxe des anniversaires

07:48 Mario Party 10

11:42 Le théorème Fold-and-Cut

14:01 P vs NP

20:37 Les problèmes 3-SAT

22:42 Les classiques Nintendo sont difficiles

29:18 Conclusion

30:56 Abonnez-vous à Mister Flech !

31:11 Générique de fin

31:38 Teaser

Si vous voulez me soutenir :

Mon bouquin : Les maths font leur cinéma -

<https://www.dunod.com/sciences-techniques/maths-font-leur-cinema-will-hunting-imitation-game>

Mon tipeee : <https://fr.tipeee.com/el-jj>

Mon KissKissBankBank : <https://www.kisskissbankbank.com/fr/projects/el-jj>