# Lexique

- Auto-scroller
- Boss rush
- Catégories
- Clip
- Cutscenes
- Damage boost
- Death abuse
- Emulateur
- Façons de mesurer le temps (IGT/RTA)
- Format d'image (PAL/NTSC)
- Frames (images)
- Genre d'un jeu
- Glitch
- Menuing
- Out of bounds
- Overflow / Underflow
- Personnal Best / World Record
- Pixel
- Rafraîchissement d'image (50hz/60hz)
- Random Number Generator
- Routing
- Sequence break
- Setup
- Skip
- Speedrun
- Splitter
- Stacking

- Storage
- Superplay
- Tool Assisted Speedrun
- Trigger
- Versions d'un jeu
- Zip

# Auto-scroller

L'auto-scroller est une phase de gameplay où le joueur voit ses déplacements plus ou moins limités par les mouvements automatiques et scriptés de la caméra. Le but recherché peut être de forcer le joueur à constamment avancer, créant ainsi parfois une certaine tension chez celui-ci. L'auto-scroller typique possède une caméra se déplaçant de gauche à droite. L'ascenseur est aussi une forme très classique d'auto-scroller (surtout dans les beat them up)

Vous en avez probablement déjà rencontrés si vous êtes féru de jeux vidéo, puisque beaucoup de licences et de genres de jeu ont utilisé cette mécanique, des jeux "classiques" (la licence Mario par exemple) aux jeux mobiles (Flappy Bird) en passant par les SHMUP (Ikaruga, DoDonPachi et bien d'autres) dont il sont généralement centraux (et qui en font, par nature, des jeux plus propice au superplay qu'au speedrun).

Ce sont généralement des phases que les runners apprécient peu en raison de leur répétitivité et du manque d'optimisation possible. Si certains jeux proposent des auto-scrollers très rigides, d'autres offrent des mécaniques très différentes comme en témoignent les exemples ci-dessous.

# Exemples d'auto-scroller standard

- Super Mario Bros 3 nous offre un exemple typique d'autoscroller *(à partir de 17:30) :*

https://www.youtube.com/embed/K_A7MX0ShYo

- Canabalt est un jeu ayant fait de l'auto-scroller le coeur de son gameplay :

https://www.youtube.com/embed/PynN-aZUGeA

- Dans Harry Potter et la Chambre des Secrets *(à partir de 0:41) :*

https://www.youtube.com/embed/P1ZWo8TBTqA

# Auto-scrollers dans le cadre de la pratique du speedrun

- Celles et ceux ayant joué à Ori and the Blind Forest se souviennent certainement de cette phase d'auto-scroller, qui a ceci de particulier qu'elle autorise le joueur à accélérer et n'est pas si restrictif dans les mouvements *(à partir de 13:40)*

https://www.youtube.com/embed/qnFAYKsta8M

- Tomb Raider Legend possède un auto-scroller plutôt long, fonctionnant avec un système de boucles : tant que vous n'avez pas tué un certain nombre d'ennemis dans une boucle, le jeu repasse la boucle en question. Il convient donc, en speedrun, de tuer le plus rapidement possible ces ennemis avant que le jeu ne recommence une boucle, ce qui peut s'avérer ardu en raison de l'aléatoire généré par le comportement des ennemis et la précision de l'arme. *(à partir de 3:50. On ne vous en voudra pas si vous ne regardez pas tout, promis)*

https://www.youtube.com/embed/rVKyTLFUVkI

- Dans Rayman, les auto-scrollers possèdent des triggers qui modifient la vitesse de défilement . La technique ici consiste à retarder le moment où l'on déclenche le ralenti de l'auto-scroller (quand le runner se place à gauche), et à anticiper le moment où l'on déclenche l'accélération de celui-ci (quand le runner se place à droite). Cette technique permet de gagner quelques secondes, là où la plupart des auto-scrollers ne permettent pas de tels gains de temps. *(de 03:51 à 04:51)*

https://www.youtube.com/embed/c-_MjYMAgdo

# Boss rush

# Catégories

Pour comparer leur performances entre eux sur une base équitable, les speedrunners ont besoin de fixer des règles qui ne bougent pas. Ces règles constituent des **catégories**.

Certaines de ces catégories sont récurrentes: on trouvera sur la quasi intégralité des jeux speedrunnés l'**any%**, mais également très fréquemment le **100%**, dont la définition va varier d'un jeu à l'autre.

Sur les jeux les plus populaires, on trouvera également des catégories qui restreignent l'usage de certains glitches ou tricks (Glitch Restriction), ou des catégories qui fixent des objectifs arbitraires (Arbitrary Goal). Il peut arriver que ces deux "catégories de catégories" se croisent (par exemple le 70 star dans Super Mario 64, dans lequel le Backward Long Jump est interdit).

L'idée de comparer les catégories par pourcentage provient du jeu Super Metroid, qui proposait justement un pourcentage d'objets récupérés une fois le jeu terminé.

## Any%

L'any%, ou "n'importe quel pourcentage", est généralement la catégorie qui fait le moins débat dans sa définition dans toutes les communautés.

Elle consiste à finir le jeu le plus rapidement en utilisant tous les moyens internes au jeu pour ce faire.

Attention: Malgré le fait que les glitches en tout genre soient autorisés, l'usage d'outils externes au jeu (par exemple Action Replay, ou le fait d'agir physiquement sur sa console par un autre biais que la manette et le bouton d'alimentation) sont proscrit.

## 100%

Le terme 100% est généralement un terme abusif pour signifier "tout le contenu proposé par le jeu". Il s'agit d'une catégorie arbitraire malgré les apparences.

En effet, si certains jeux proposent directement un pourcentage de complétion, ce n'est absolument pas une situation universelle. Et lorsque le jeu ne définit pas ce qu'est le 100%, c'est aux joueurs de ce concerter et de poser des règles.

Par exemple, dans le jeu The Legend of Zelda: Ocarina of Time, la catégorie 100% est définie comme "obtenir tous les objets présents sur l'écran d'inventaire en dehors d'un donjon". Cela exclue notamment de récupérer la carte, la boussole et la clé du boss de chaque donjon, ou encore de faire le mini-jeu qui permet d'obtenir une vache dans la maison de Link (car ce bonus n'apparaît nulle part sur l'écran de pause).

# Glitchless

Les catégories glitchless (sans glitch) se veulent, comme leur nom l'indique, être des catégories où le jeu est complété sans faire usages de défauts de programmation.

Néanmoins, il s'agit malgré tout dans beaucoup de jeux de catégories arbitraires, dans la mesure où certaines techniques font débat sur le fait qu'il s'agisse de glitches ou pas.

Par exemple, dans Final Fantasy VI, le fait d'exploiter une faille du système de combat pour fuir face à Ifrit est accepté en glitchless, alors qu'il est normalement impossible de s'enfuir d'un combat de boss.

# Glitch Restriction

Les catégories de type glitch restriction visent à interdire l'utilisation d'un (ou plusieurs) glitchs précis.

La raison à cela est généralement de proposer des catégories variées dans leur route. En effet, certains glitches sont parfois tellement puissant dans leur capacité à sequence break que la création d'une route alternative dans laquelle ce glitch est interdit devient pertinent pour créer de la variété dans le speedrun du jeu.

Par exemple, la catégorie 16 star dans Super Mario 64, ne correspond ni à l'any% (qui se fait à l'heure actuelle en 0 ou 1 star), ni à la 100% (120 star), et interdit spécifiquement l'usage du glitch appelé *Side Backward Long Jump*, qui permet justement de passer de 16 stars à 0.

# Arbitrary Goal

Les catégories de type arbitrary goal visent à fixer un objectif supplémentaire forcé au joueur. L'objectif en question peut être supplémentaire mais nécessiter malgré tout une complétion du jeu, ou être l'objectif final en lui-même.

Dans le premier cas, le but recherché par l'existence de la catégorie est là aussi une variation de la route. Par exemple: La catégorie *Stone Medallion Trials* dans Ocarina of Time.

Dans le deuxième cas, le but recherché est généralement de proposer une version courte d'un speedrun très long. En effet, certains jeux ont des speedruns any% qui peuvent être très longs (plusieurs heures), ce qui rend l'action de les speedrunner complexe. Des catégories "jusqu'à un certain point" sont alors créés. Par exemple, la catégorie *Get Shampoo* dans Baten Kaitos : Eternal Wings and the Lost Ocean (l'any% dure ~13h alors que la catégorie Get Shampoo dure ~1h).

# Low%

Malgré ce qui pourrait apparaître de prime abord, any% ne veut pas dire "le moins possible", mais bien "peu importe pourvu que c'est le plus rapide". Le low% (pourcentage bas), lui en revanche cherche vraiment à définir "le moins possible", ce qui peut être plus long (voir dans certains cas beaucoup plus long) qu'une catégorie any%.

Il peut arriver que, à la suite de certaines découvertes, ce qui était le low% ne le soit plus, ce qui peut donner naissance à des "anciennes catégories". Par exemple, le 22% dans Metroid Prime n'est pas le low% (c'est le 21%).

# Reverse Boss Order

Certains jeux sont tellement cassés qu'il est possible de fixer des objectifs arbitraires à priori absurdes et un peu humoristiques.

C'est le cas de la catégorie Reverse Boss Order, qui consiste à combattre les boss du jeu en partant du dernier pour finir sur le premier. Cette catégorie est possible par exemple sur Super Metroid, Ocarina of Time, ou encore Donkey Kong Country.

# Clip

# Cutscenes

# Damage boost

Beaucoup de jeux vont essayer de marquer/punir une erreur du joueur en faisant une animation pour marquer une prise de dégât, avec éventuellement une propulsion en arrière et des frames d'invulnérabilité pour éviter que la punition s'enchaîne avec d'autres attaques.

Il n'est pas rare qu'il soit possible d'exploiter ces animations de diverses façon pour en tirer un avantage, par exemple pour obtenir une allonge supplémentaire lors d'un saut, pour obtenir un gain de vitesse en jouant sur l'angle de propulsion, ou encore pour profiter des frames d'invulnérabilités pour passer un obstacle dangereux ou un groupe d'ennemis.

Par exemple, dans le jeu *Super Metroid*, l'utilisation efficace des damage boost sera un des facteurs déterminants pour distinguer les meilleurs runners (dans l'image, c'est un TAS, mais c'est pour l'idée).


Un damage boost dans Super Metroid

# Death abuse

# Emulateur

# Façons de mesurer le temps (IGT/RTA)

# Format d'image (PAL/NTSC)

Les jeux sur console étant généralement distribués par région (ou du moins ils l'étaient jusqu'à assez récemment), il convient de les adapter aux spécificités régionales afin de les rendre compatibles avec le marché local. Une de ces spécificités est le format de l'image.

## Formats historiques (PAL/NTSC)

Le NTSC (pour National Television System Comitee) est un standard de codage analogique pour la vidéo en couleur. Il a commencé à être vraiment utilisé pour la télévision aux États-Unis d'Amérique en 1953, puis qui s'est exporté vers d'autres pays (Canada, Japon, etc...). C'est un format d'image couleur 4:3, de résolution finale 640x480, et avec un taux de rafraîchissement de 60Hz.

Le PAL (Phase Alterning Line), comme son nom ne l'indique pas est également un standard de codage couleur. Il a été développé en Allemagne après le NTSC, en essayant de corriger les défauts de ce dernier (notamment les problèmes de fidélité des couleurs), et a commencé à être utilisé au début des années 1960, notamment en Europe et en Australie. Il s'agit d'un format d'image couleur 4:3, de résolution finale 768x576 et avec un taux de rafraîchissement de 50Hz.

On peut alors avoir jeux PAL qui sortaient sans aucune retouche, et donc qui tournent moins rapidement que leurs homologues NTSC, aussi bien au niveau de l'audio que de la vidéo. Il y a aussi des bandes noires en haut et en bas de l'écran pour combler la différence de résolution (576 – 480 = 96 lignes de différence). Sonic The Hedgehog sur Mega Drive cumule absolument tous ces défauts !

https://www.youtube.com/embed/Yi0OlParbyY

Parfois les versions PAL sont optimisées. Les musiques se jouaient à la bonne vitesse, l'image prenait bien tout l'écran, et l'optimisation pouvait aller jusqu'à la modification de la vitesse de certains éléments du jeu pour se rapprocher au mieux de l'expérience NTSC. C'est le cas de **Super Mario World** par exemple.

https://www.youtube.com/embed/xxvwRDF8pjQ

# Formats modernes

Si les formats mentionnés précédemment étaient valable jusque vers le milieu des années 2000, tout ce microcosme a été chamboulé avec l'arrivée de la haute définition qui a apporté ses propres formats.

A partir de là, la différence régionale s'estompe pour laisser la part belle aux différents HD Ready, Full HD, etc.

On trouvera en vrac:

- le 720p (p pour *progressive scan*, signifiant que l'affichage de l'image se fait continuellement de haut en bas (en général) jusqu'à arriver au bout et recommencer du début) qui fait donc 1280x720 pixels
- 1080i (i pour *interlaced scan*, qui consiste à dessiner une ligne sur deux d'une image et une ligne sur deux de l'image suivant, mais en alternance, ce qui permet en jouant sur la vitesse de rafraichissement de donner l'impression d'une image complète), et qui fait donc du 1920x540 (mais 540 qui oscillent une image sur deux pour former 1080 lignes distinctes)
- 1080p, qui fait donc 1920x1080 pixels
- et plus récemment la 4K, qui fait donc 3840x2160 (en progressif)

# Frames (images)

# Genre d'un jeu

# Glitch

Un glitch, ou bug, décrit un défaut de programmation. Il n'existe pas de différence inscrite dans le marbre entre un glitch et un bug, mais on a tendance à utiliser le terme "glitch" pour décrire un bug exploitable de façon positive pour le speedrunner. C'est évidemment très arbitraire comme distinction.

Il existe une multitude de types de glitches. Dès qu'un système a une faille, il a un glitch, et comme un jeu vidéo est un ensemble de systèmes et de règles, il existe autant de glitches différents. Certains sont très récurrent dans leur logique et ont donc droit à un nom qui leur est propre et qu'on retrouvera entre plusieurs jeux, malgré le fait que les jeux en question aient été développés pas des gens différent qui ne se connaissent probablement même pas. Attention toutefois, ce n'est pas parce que deux glitches appartiennent au même genre (et donc, a fortiori, ont la même logique globale) qu'ils vont s'exécuter de façon identique ni avoir le même résultat.

La difficulté d'exécution d'un glitch (ainsi que sa faisabilité) va varier grandement en fonction des setup, ainsi que des versions du jeu utilisées.

Ce qui est considéré en général ou non comme un glitch va varier grandement du contexte, des communautés. Si certaines techniques laissent peu la place au débat, d'autres sont dans une zone nettement plus grise. Par exemple, l'exploitation d'une mécanique comme le damage boost pour parvenir à ses fins est rarement considéré comme un glitch, malgré qu'il permette parfois de résoudre des situation ou éviter des segments du jeu de manière absolument non prévue et souhaitée par les développeurs du jeu.

Parmi les glitches les plus récurrents, on trouvera notamment les out of bounds, les zip, les clip, les overflow/underflow, ou encore le storage.

Il existe bien entendu de nombreux autres glitches propres à certains jeux, dont certains ont leur propre nom comme le Reverse Bottle Adventure (RBA) de *The Legend of Zelda: Ocarina of Time* ou l'Accelerated Back Hopping (ABH) de *Half-Life 2* et *Portal*. Ci dessous, un exemple d'Hyper Extended SuperSlide (HESS) dans Ocarina of Time

Un Hyper Extended Superslide d'Ocarina of Time

# Menuing

Le menuing désigne le fait de se déplacer dans les menus d'un jeu. Cela peut impliquer des actions diverses comme équiper un sort, une compétence ou autres, sauvegarder, réorganiser sa formation, etc.

C'est un terme qu'on retrouve la plupart du temps naturellement dans le speedrun de RPG, même si techniquement, à partir du moment où un jeu a un menu, se déplacer dedans est du menuing (indépendamment de son genre, donc). Notons que le menuing fait partie intégrante du routing, puisqu'il faut planifier et optimiser chaque objet à prendre / équiper ou non.

# Exemples

- Dans certains jeux, passer du temps dans les menus empêche le runner d'avancer et c'est donc forcément une perte de temps. Il est alors requis de travailler ces menuings afin d'y passer le moins de temps possible. Cela peut être très impressionnant à regarder comme dans ce run de **Final Fantasy IX** *(de 7h49m40s à 7h50m45)*
- Dans **Dark Souls** à l'opposé, le temps ne s'arrête pas et vous pouvez même continuer d'avancer. Cela peut donc être dangereux si fait au mauvais moment. Le menuing est alors effectué lorsque le speedrunner se trouve dans une zone sans risque, dans un ascenseur, monte une échelle ou, plus globalement, lorsqu'il a le temps de le faire sans compromettre son avancée. *(de 0:36 à 0:41 et de 0:47 à 0:54) :*

https://www.youtube.com/embed/IYFu0b7VROs

- Nous pouvons observer la même chose dans le jeu **Black Messiah of Might and Magic** *(de 13:45 à 13:52)* :

https://www.youtube.com/embed/4gDPilODlw4

# Out of bounds

Lorsque le joueur est amené à sortir des limites définies par les développeurs, on dit qu'il est "out of bounds" (oob), ou en français, "en dehors des limites" (le terme "out of map" s'utilise également occasionnellement).

Sortir out of bounds permet en général lorsque c'est possible de sauter des portions entières d'une carte, et donc par exemple d'éviter des portes (ou des portes déguisées en obstacles), d'atteindre des sections tôt (amenant parfois à du sequence break), d'aller chercher des trigger (ou au contraire de les éviter).

Il s'agit d'un des types de glitches qu'on retrouve le plus fréquemment tous jeux confondus, en raison de sa grande simplicité.

En général, les développeurs ne prévoient pas que le joueur sorte des limites imposées, et donc ce qu'on trouve le plus souvent dans ces "non zones" est du vide pour les jeux 3D (un fond noir ou coloré, ou une skybox quand il y en a une), et de la corruption pour les jeux 2D (le jeu essaye de charger des éléments de décor qui correspondent à des segments de la mémoire dédiés à d'autres choses et essaye de faire le lien avec les images qu'il a dans sa mémoire).

Il arrive néanmoins, par accident ou plus rarement volontairement, que des choses se trouvent en dehors des limites, comme par exemple des murs invisibles, ou des éléments oubliés (par exemple, des décors non terminés).

Une vue out of bounds de Journey

Parmi les jeux qui contiennent de surprises out of bounds, on citera par exemple *Batman Arkham City* et , ou encore *Journey* dont certaines zones out of bounds décèlent des effets visuels étonnants, ainsi que des bâtiments modélisés mais non utilisés.

# Overflow / Underflow

# Personnal Best / World Record

# Pixel

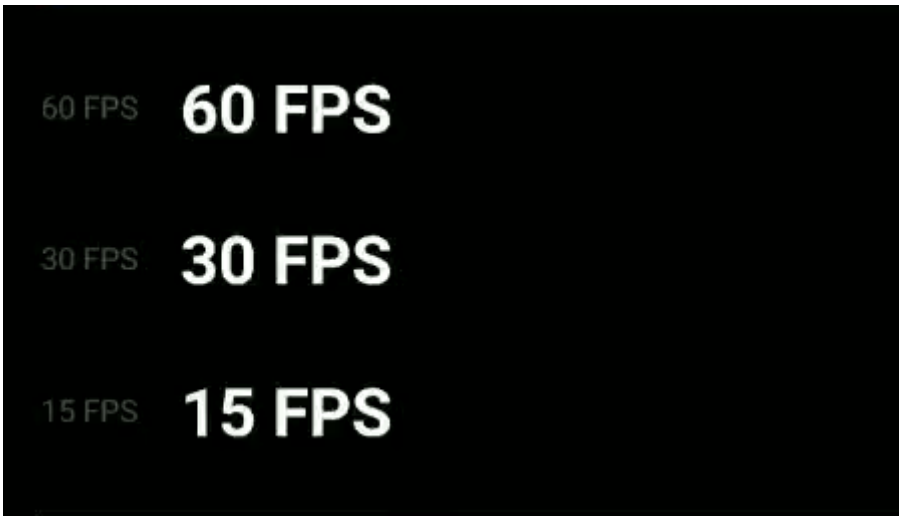
# Rafraîchissement d'image (50hz/60hz)

La fréquence de rafraîchissement d’un écran correspond au nombre de fois que l’image va être mise à jour par seconde. Un rafraîchissement de 60Hz veut donc dire que l’écran sera capable d’afficher 60 images différentes en 1 seconde.

Il ne faut pas confondre la fréquence de rafraîchissement et le nombre d'images par seconde (les FPS). Que votre machine envoie 12 ou 4500FPS, l'image sur votre écran 60Hz ne sera rafraîchie que 60 fois par seconde. Et même si les images ne sont pas toutes affichées sur votre écran, elles ont quand même existé, et vos actions ont quand même toutes été prises en compte. Par exemple, l'elevator skip de Dishonored fonctionne plus simplement à partir de 200FPS, car du coup le nombre de frames sur lequel on peut l'activer augmente.

De nos jours, tous les écrans sont compatibles 60Hz, mais cela n’a pas toujours été le cas. En effet, jusqu’à la moitié des années 1990, la très grande majorité des télévisions de la zone PAL ne supportaient que les signaux 50Hz. Les constructeurs ont donc été contraints d’adapter leurs consoles à cette norme en modifiant, entre autres, la fréquence à laquelle la machine va fonctionner.

Les jeux portés sans aucune modification sur une console de la zone PAL seront donc généralement plus lents que leurs homologues NTSC. Prenons un exemple simple pour expliquer ceci. Imaginons que votre personnage possède une animation de saut qui s'étend sur 60 frames. L'animation durera donc 1 seconde sur un écran 60Hz. Sur un écran 50Hz, par contre, votre animation sera toujours découpée en 60 frames, et elle s'étalera donc sur 1,2 secondes (50 images en 1 seconde + 10 images en 0,2 secondes).




On peut aussi noter que pour les consoles portables, l'écran n'étant pas branché sur le secteur, il n'y a pas de différence de taux de rafraîchissement entre les consoles de différentes régions.

A noter également, avec l'arrivée de la 3D sur PC (et ses lunettes qui allaient avec), il s'est trouvé sur le marché des écrans 120hz (deux fois 60hz, la carte graphique calculant une image sur deux pour l'œil gauche et l'autre pour l'œil droit, les lunettes se chargeant de faire le tri).

Vous trouverez peut-être sur le marché des téléviseurs vous annonçant des taux de rafraichissement surprenant (par exemple 100hz), qui ne correspondent pas à un standard. Il s'agit en fait d'un travail sur l'image effectué par le processeur graphique intégré à la télévision pour calculer des images intermédiaires pour fluidifier le mouvement.

Si la pertinence d'une telle technologie est discutable dans le cadre de la télévision (qui envoie en général 24 images par secondes), elle est en revanche non souhaitable dans le cadre du jeu vidéo. En effet, les images ajoutées étant calculées, elles manquent souvent de précision, et surtout, le travail sur l'image effectué par la télévision induit un décalage dans le temps de réponse du jeu (input lag), qui peut aller jusqu'à rendre le jeu injouable.

# Random Number Generator

En informatique, l'aléatoire n'existe pas (expliquer pourquoi serait long et compliqué). Tout est déterministe et reproductible. Pour pallier à ça, On génère des nombres *pseudo-aléatoires*, en se basant sur des éléments impossibles à prévoir et manipuler sur une échelle humaine.

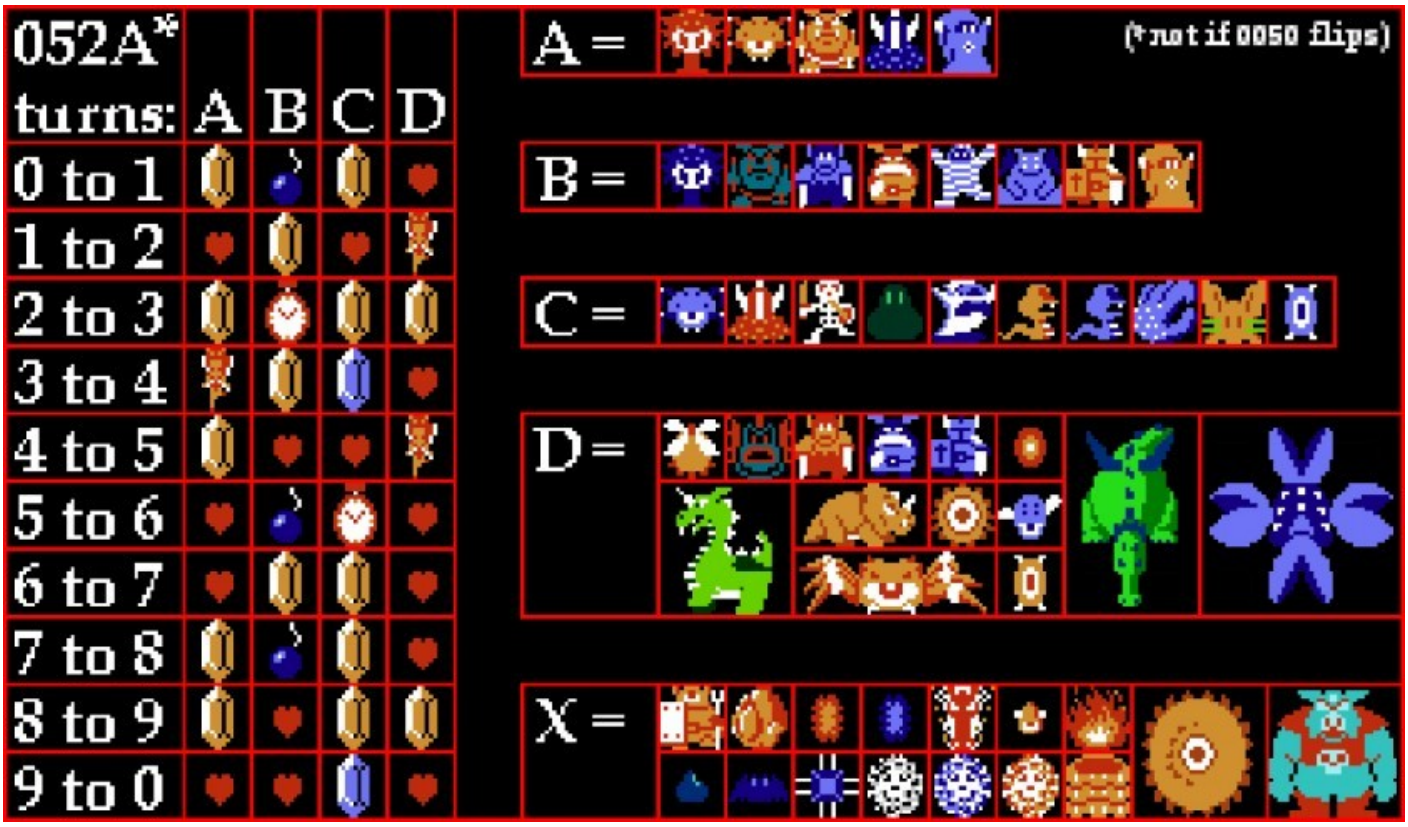
Ce processus de génération de nombre aléatoire est appelé le **Random Number Generator** (souvent appelé "La RNG"). Par vulgarisation, quand on parle de RNG, on parle d'aléatoire.

## Seed

Une seed ("graine" en Anglais) est, dans la RNG, l'élément "clé" qui va déterminer toute la génération aléatoire. En général (l'immense majorité du temps), l'algorithme lui-même ne change pas. Seul la seed va changer.

Les critères sur lesquels sont générés ces seed vont varier en fonction des choix des développeurs. On retrouvera notamment:

- l'heure interne de la console
- les touches pressées par le joueur
- le nom du fichier ou les noms donnés aux personnages dans le jeu
- sur PC, les mouvements de souris du joueur pendant une fenêtre de temps
- des "planifications aléatoires" (qui ne sont évidemment pas aléatoires mais qui essayent d'en donner l'impression, comme par exemple la table des loots du jeu *The Legend of Zelda* sur NES)




La seed peut soit être unique pour toute une partie (ce qui arrive parfois ou peut survenir volontairement sur certains romhack, par exemple sur certains randomizer), soit changer régulièrement (le plus souvent, surtout sur les jeux les plus récents). Le fait que la seed soit unique ou non va être grandement influencé par ses critères de génération et va déterminer la faisabilité d'une **RNG Manipulation**, trick qui consiste à être maître de la génération aléatoire du jeu.

# Routing

Le routing désigne le fait de prévoir un speedrun, au sens macroscopique comme sur des éléments précis. C'est un exercice qui demande une connaissance théorique exhaustive de l'état du speedrun sur un jeu, et qui implique réflexion, expérimentation, et test des différentes stratégies afin d'optimiser un speedrun.

Le but est de prévoir une route impliquant donc un ordre des évènements, les tricks utilisés, et de prévoir les besoins en ressources (argent, vie, etc.), qu'on saura ainsi optimale avant de commencer à la pratiquer.

Par essence, le routing se remet en permanence lui-même en question, pour aboutir à des évolutions notables : gain de temps, nouvelles découvertes de glitches, facilités de nouveaux setups, etc. La découverte de nouveaux glitches influence en général grandement le routing, qu'il faut alors repenser pour optimiser toujours plus.

La pratique du routing ne concerne pas que le speedrun. Elle est également pertinente dans le cadre du TAS et du superplay.

# Exemples

- Dans Half-Life, les runners avaient pour habitude de parler au scientifique à un moment précis afin de le faire venir vers la porte (ce qui n'est pas censé être possible). Si l'on poussait le scientifique vers la porte, ce dernier l'ouvrait, nous laissant accéder à un interrupteur nous autorisant à finir un chapitre aussi vite qu'il n'avait commencé. *(de 8:52 à 9:10)*

https://www.youtube.com/embed/oEeFzIC5FDU

Après un travail de routing, les runners ont conservé ce principe de manipulation du scientifique mais ont opté pour une nouvelle stratégie, bien plus rapide : effrayer le scientifique en lui tirant dessus et bloquer le passage (derrière le runner) permet de le manipuler pour qu'il aille ouvrir la porte en courant. La grenade quant à elle permet d'éviter qu'il ne bloque le passage. *(de 6:53 à 7:05)*

https://www.youtube.com/embed/LzLZ-hYNmL4

- Dans **Dishonored**, énormément d'améliorations ont été apportées. Entre 2013 et 2016, les runners ont gagné 7 minutes, notamment grâce au simple achat de potions de mana assez tôt dans la run *(dans la vidéo ci-dessous, de 9:18 à 9:30)*, ce qui implique de récolter suffisamment de pièces d'or plus tôt dans la run. L'achat de ces potions de mana a permis d'utiliser la compétence Blink (ou "téléportation) bien plus souvent, et la gestion du mana est dès lors devenue beaucoup plus facile.

https://www.youtube.com/embed/LDGrV_J09ws

- **Quake** est un speedrun comportant peu de glitchs, mais propose une richesse en terme de routing. Le tuto d'Elgu, ancien recordman de Quake et excellent runner de fps, montre bien cela. Il propose différentes façons d'atteindre la clé, condition requise pour finir le niveau.

- A 6:22, il montre comment l'atteindre grâce au bunnyhop, une technique permettant de gagner de la vitesse en enchainant des sauts.

- A 6:42, il propose d'utiliser le wall strafing, une technique permettant de gagner de la vitesse en courant contre le mur.

- De 7:05 à 7:30, le runner effectue une solution alternative, qui est aussi le moyen classique d'atteindre la clé : appuyer sur l'interrupteur, ce qui fait apparaître un pont menant directement à la clé.

https://www.youtube.com/embed/i__AApGprog

# Sequence break

Un jeu linéaire va proposer une séquence d'évènements au joueur qu'il s'agira de suivre dans un ordre plus ou moins défini par les développeurs. Le fait, dans le speedrun, d'aller chercher un évènement plus tôt que ce que la chaîne d'évènement prévue par les développeurs aurait normalement autorisé (par le moyen de tricks ou de glitches) s'appelle du *sequence break*.

Le but du sequence break dans le cadre du speedrun va généralement être de skip certains morceaux du jeu (des objets, voir des niveaux entiers), ou alors d'obtenir un avantage dans un niveau avancé (par exemple un objet très puissant) pour faire plus facilement les niveaux précédents.

Par exemple, dans le jeu *The Legend of Zelda: Majora's Mask*, le grappin est prévu pour être obtenu entre le 2e et le 3e temple du jeu, et l'obtenir avant casse complètement les deux premiers donjons. Autre exemple: Le Backward Long Jump dans *Super Mario 64* permet de sauter des pans entier du jeu.

un exemple de BLJ dans Mario 64

Image not found or type unknown

Attention toutefois à ne pas confondre les notions de linéarité et d'ouverture dans un jeu. Un jeu peut être ouvert et linéaire (par exemple les *Assassin's Creed*), tout comme il peut être peu ou non linéaire tout en étant fermé (par exemple les jeux musicaux, qui ouvrent l'entier de leur contenu dès le début, mais qui n'offrent aucune liberté d'interprétation au cours d'une chanson). La notion de sequence break est liée à la linéarité d'un jeu. Pas à son ouverture.

# Setup

# Skip

Effectuer un skip signifie outrepasser un élément ou une phase de jeu, avec l'utilisation de glitchs ou non, ce qui permet au runner d'obtenir un gain de temps plus ou moins important. Parmi les plus courants, on notera les skips de dialogues, de cutscenes, voire de chapitres entiers d'un jeu.

## Exemples

- Dans **Super Meat Boy,** le Brownie Skip consiste à ne pas effectuer la course contre Brownie, mais à contourner le décor par la droite et atteindre rapidement le trigger de fin de course. *(de 7:25 à 7:38)*

https://www.youtube.com/embed/Wq9MQ8Y9Wo0

- Dans **Half-Life**, poser une mine sur l'obstacle permet de monter par-dessus ce dernier et de skip un chapitre entier ! *(de 17:26 à 17:38)*

https://www.youtube.com/embed/b_Qd6dNslno

- Dans **Soleil**, sauter en bas ou en haut d'une corde puis presser droite permet de passer en OoB et de skip une partie du niveau. *(de 9:33 à 10:00)*

https://www.youtube.com/embed/OIKZPLmwrCI

- Dans **Dishonored**, le simple fait de rester appuyé sur la touche action (F par défaut sur PC) pendant la cutscene annule le dialogue qui est censé se dérouler après être arrivé (notez comme les gardes s'en vont aussitôt alors qu'ils devraient parler avec vous) *(de 20:48 à 21:10)*

https://www.youtube.com/embed/LDGrV_J09ws

# Speedrun

Le speedrun est l'action de compléter une tâche dans un jeu vidéo le plus rapidement possible.

Cette tâche peut être simplement de finir le jeu (en se fixant ou non des objectifs intermédiaires), ou parfois d'obtenir un objet particulier (souvent sur les jeux longs) ou encore d'effectuer un niveau précis du jeu (on parle alors d'*Invidual Levels*, ou IL).

Il s'agit d'une pratique compétitive, contre d'autres joueurs mais également contre soi-même (notre premier concurrent est notre meilleure performance). Qui dit compétition, dit comparaison, et pour cela, on passe par des tableaux de scores (leaderboard).

Le leaderboard le plus globalement utilisé est **speedrun.com**, mais il en existe d'autres.

# Ça vient d'où ?

L'idée de compétition dans les jeux vidéos n'est pas nouvelle. On pensera notamment aux tableaux des scores dans les bornes d'arcade qui incitaient au superplay, mais aussi plus simplement aux jeux qui mettent en compétition directe des joueurs, par exemple les jeux de combat, et qui peuvent donner lieu à des tournois avec des prix à la clé.

Néanmoins (et bien que les développeurs aient mis longtemps à le comprendre), le scoring ne se prête pas à tous les genres de jeux. On peut mettre un compteur de score un peu sur n'importe quoi dans n'importe quel jeu, mais cela ne transcende pas pour autant le gameplay en offrant une nouvelle profondeur (dans certains cas, évidemment, oui, mais pas toujours).

Notamment, les joueurs de jeux solo post-arcade qui proposaient une progression linéaire ont rapidement eu besoin d'une meilleure base de comparaison que le score. Et quoi de plus naturel que la capacité à finir vite un jeu ?

Il est probable que le speedrun existe depuis le premier jeu disposant d'un début et d'une fin, car naturellement quelqu'un a du se demander à quel point il était capable de finir un jeu rapidement. Néanmoins, il a fallu attendre 1993 pour trouver la première trace de compétition ouverte sur le speedrun, avec le site COMPET-N. En effet, ce dernier proposait de comparer des speedruns de Doom, alors probablement le premier jeu qui proposait d'enregistrer sa performance (au moyen d'un fichier contenant les touches pressées et leur timing, à la façon d'un TAS aujourd'hui).

# Splitter

Les splitters sont des logiciels visant à améliorer la pratique du speedrun. Il s'agit d'un cas intéressant de logiciel conçu dans un cadre précis pour un usage ultra spécifique, et qui a peu sinon aucun sens en dehors de son contexte.

Il s'agit simplement d'un chronomètre segmenté. Il mesure le temps, et le divise en sous-catégories (on parle de "splits") avec un timer global. L'idée étant de pouvoir comparer ses performances sur des plus petites sections qu'une run entière, à la fois pour suivre la performance globale de la run, mais aussi pour pouvoir déterminer quels sont les segments de la run qui peuvent être améliorés.

De nombreux splitters ont étés développés et se sont succédés (WSplit, Llanfair, etc.), mais il en est un qui domine largement aujourd'hui de par sa polyvalence et sa puissance. Il s'agit de **LiveSplit**.

Le travail du splitter est de permettre la mesure segmentée du temps de la run, mais il propose en général de nombreuses autres fonctions, comme une comparaison par rapport au personnal best, ou un suivi des performances (donnant naissance à la notion de "best split", des fois appelés "gold split" en raison de leur couleur dorée sur les splitters). Il peut également proposer des graphiques, ou une intégration vidéo du personnal best.

Accessoirement, il s'agit d'un logiciel généralement hautement personnalisable, qui s'adapte pour correspondre au mieux au besoins du speedrunner (dans le cas de LiveSplit, il peut s'afficher verticalement ou horizontalement, s'intègre très bien au stream avec un peu d'huile de coude, et si vous speedrunnez un jeu en 4:3, il peut remplir avantageusement l'espace à pour faire un 16:9e).

En bref, il s'agit d'une trousse à outil indispensable pour le speedrunner.

# Stacking

# Storage

# Superplay

# Tool Assisted Speedrun

Si vous avez déjà regardé un speedrun en vous disant "mais ça ressemblerait à quoi si c'était effectué absolument parfaitement ?", eh bien vous avez fait le premier pas dans le monde du Tool Assisted Speedrun.

Cette discipline (communément appelée "TAS"), n'est pas compétitive, contrairement au speedrun. Elle est collaborative. En partie parce que le travail demandé à la réalisation d'un TAS est tellement grand que la compétition n'a pas sa place, mais aussi parce qu'elle n'aurait pas vraiment de sens.

Le principe d'un TAS, c'est de faire un speedrun, mais en remplaçant l'aspect "performance" par un aspect de "travail perfectionniste". Dans un TAS, on peut se permettre d'utiliser des outils externes au jeu pour en avoir une meilleure compréhension, et un meilleur contrôle. Attention néanmoins, il ne s'agit pas de tricher ! Aucune modification préalable du code du jeu n'est tolérée (par des codes de triche ou autres). Le TAS vise à être "vrai", et à pouvoir être exécuté sur n'importe quelle version du jeu, sans intervenant autre que la manette (des fois, plusieurs manettes).

Parfois, le but du TAS n'est pas de finir le jeu le plus vite possible, mais de montrer que quelque chose est possible. On parlera alors de **Tool Assisted Demonstration**, ou TAD.

Le site de référence pour les TAS est **tasvideos.org**.

Pour faire un TAS, on va utiliser des outils perfectionnés (comme des émulateurs ultra précis), qui vont nous permettre notamment:

- de ralentir le jeu jusqu'à avancer image par image (et dans certains cas, encore plus lentement)
- de ré-enregistrer des segments autant de fois que nécessaire (retour en arrière)
- de modifier librement la séquence des touches sur lesquelles on va appuyer et leur timing comme on modifierait le script d'une pièce de théâtre
- de voir le contenu de la mémoire interne au jeu (pas la modifier directement, mais la consulter oui)

De plus, le fait d'avoir un contrôle total sur la précision des touches rentrées sur la manette permet généralement l'exécution de tricks normalement absolument impossible à effectuer par des humains. On parle alors de tricks *TAS-only*.

La constitution d'un TAS est un travail de minutie et de patience. Certains projets peuvent prendre plusieurs mois pour être complétés, voir plus, au point que certains vont parfois proposer des **LOTAS** (pour *Low Optimization TAS*) ou **LOTAD** (pour *Low Optimization TAD*), afin d'obtenir un résultat non optimal mais donnant une bonne idée d'un TAS ou d'une démonstration dans un

temps de constitution raisonnable.

Un exemple célèbre de TAS est le 1 key de *Super Mario 64*.

https://www.youtube.com/embed/TkOkJvLKxUY

Un exemple célèbre de TAD est le no door de *The Legend of Zelda: Ocarina of Time* (qui consiste à faire tout le jeu sans ouvrir de porte):

https://www.youtube.com/embed/S9dLpmXR0kw

# Arbitrary Code Exectution (ACE)

L'ACE, parfois appelé "Total Control" est dans le domaine du TAS généralement perçu comme le nanan du nanan, le Saint Graal, l'objectif final de tout TASeur sur un jeu.

Il s'agit en fait de pouvoir effectuer une exécution arbitraire de code uniquement en se basant sur le jeu choisi et sur les boutons pressés sur la manette. Cela permet de littéralement recoder ce que vous voulez directement en passant par le jeu, et donc à partir de là faire réellement ce que vous voulez.

La méthode pour exécuter un ACE (lorsque c'est seulement possible) va varier grandement d'un jeu à un autre, donc il est difficile d'écrire une méthodologie. Néanmoins, lorsqu'un ACE est trouvé, il devient généralement très célèbre.

Un exemple très connu d'ACE est celui de *Pokemon Red* qui recode un chat Twitch avec une interface réseau fonctionnelle en utilisant un Super Gameboy (la démonstration démarre vers 6m40) :

https://www.youtube.com/embed/P28kp66XMw4

# Trigger

Trigger signifie déclencheur. Vous l'avez forcément déjà expérimenté en tant que joueur : vous avancez dans une zone, et une cinématique apparaît ou vous fait passer au niveau suivant : vous avez touché/franchi un trigger. On le représente souvent comme une ligne ou une zone invisibles pour le joueur, qui permettent d'initier de nouveaux événements.

## Exemples

- En speedrun, les runners cherchent constamment à atteindre certains triggers le plus rapidement possible. Par exemple, dans **Tomb Raider 2**, lorsque le joueur atteint le trigger situé dans l'avion, le jeu change de niveau : *(de 10:20 à 10:25)*

https://www.youtube.com/embed/ENG-p4oXm1o

- Dans d'autres cas en revanche, les runners cherchent plutôt à éviter ces triggers. Ces derniers étant invisibles, il est parfois assez ardu d'effectuer cette opération. Dans **Dishonored**, le joueur est supposé parler à Emily dès lors qu'il franchit le pont. Dès le début le runner effectue l'Elevator Skip, un glitch qui lui permet de prendre de la hauteur puis de passer au-dessus du trigger de ce dialogue. L'interaction avec Emily n'a alors pas lieu et elle reste plantée à vous attendre. *(de 0:00 à 0:14)*

https://www.youtube.com/embed/LDGrV_J09ws

- Quake est composé d'un certains nombres de techniques et skips. Dans le niveau nommé e3m2, lorsque le joueur avance la clé disparaît dans le sol *(dans la vidéo ci-dessous, de 39:23 à 39:27)*. Nous pouvons éviter cela en effectuant un damage boost, qui fait gagner un peu de hauteur et propulsera le runner au-dessus du trigger faisant disparaître la clé. (de 39:27 à 39:38). Il propose même une solution alternative (voir routing) (de 39:51 à 40:05)

https://www.youtube.com/embed/i__AApGprog

# Versions d'un jeu

"Oh non, encore une mise à jour de mon jeu" est une phrase qu'on entend assez fréquemment de nos jours. Pourtant, le fait de modifier le code d'un jeu après sa sortie initiale ne date pas de l'ADSL. Déjà à l'époque des cartouches et des consoles non connectées, les jeux pouvaient être sortis plusieurs fois en apportant des modifications plus ou moins importantes.

A cela s'ajoute d'autres types de différence de versions: Les portages, basés ou non sur de l'émulation, et les différences régionales.

Bien évidemment, toutes ces modifications peuvent avoir une importance dans le speedrun, dans la mesure où les glitches sont fréquemment au centre des routes et donc que le fait qu'ils soient patchés ou que de nouveaux tricks soient trouvés influencera grandement le développement des runs.

# Révisions internes d'un jeu

Le premier type de différence de version qui vient en tête est simplement le numéro de version (v1.0, v1.1, etc.) et qui signifie généralement que le code du jeu a été modifié dans le but d'améliorer son expérience.

Les détails modifiés par les développeurs ne concernent en général pas tous le speedrun. Il peut s'agir d'éléments graphiques modifiés, de contenu ajouté/retiré, etc. Ci dessous, Morpha dans le jeu *The Legend of Zelda: Ocarina of Time*, à gauche en version 1.0, et à droite en version 1.1 et 1.2.

A gauche, Morpha dans la version 1.0 d'OoT, à droite, dans la version 1.2

Mais il arrive aussi que des glitches soient corrigés, ou, dans de plus rares cas, ajoutés par accident.

# Différences régionales

Un type de différence de version qui prend une importance très grande dans le cadre du speedrun est la différence régionale.

La première raison à cela est naturellement la langue. Toutes les langues ne s'écrivent pas avec la même compacité, et bien souvent (et historiquement) le japonais est la langue qui s'écrit le plus rapidement à l'écran, en raison justement de son texte très compact (un seul kanji pouvant

contenir un mot, voir plusieurs dans certains cas).

Néanmoins, cela n'est pas toujours le cas. En effet, il arrive que d'autres langues tirent leur épingle du jeu. Par exemple, dans *The Legend of Zelda: Breath of the Wild*, la langue la plus rapide pour le speedrun any% est l'allemand. En effet, si l'allemand a tendance à écrire de très longs mots en comparaison d'autre langues, à l'oral la situation est tout autre (par opposition par exemple au japonais, où une idée exprimé en quelques kanji peut prendre un grand nombre de syllabes pour être prononcé). Le jeu intégrant des doublages, le fait de jouer en allemand permet de gagner un peu de temps.

Une autre différence régionale qui a son importance dans le speedrun (surtout pour les jeux pré-HD ), c'est la différence de rafraichissement d'image. En 60hz, les jeux vont en général un peu plus vite qu'en 50hz (voir l'article lié pour plus d'explications).

Enfin, et plus rarement, les différences de régions peuvent s'accompagner de différences plus ou moins grande au niveau du gameplay. On citera notamment le niveau de difficulté un cran au dessus des autres régions pour la version japonaise de *Catherine*, ou encore la version japonaise de *The Legend of Zelda: Majora's Mask* qui possède de grandes différences avec la version NTSC ou la version PAL, comme par exemple le fait que sur la version japonaise on ne puisse pas sauvegarder aux hiboux, avec en compensation trois fichiers de sauvegarde à disposition au lieu de deux par chez nous.

# Rééditions d'un jeu et imprécision de l'émulation

Entre la popularité grandissante du retro gaming et le fait que les jeux ont subi avec le passage à l'ère HD une forte augmentation de leur coût de production, le fait de ressortir un jeu est une pratique courante au sein de l'industrie du jeu vidéo d'aujourd'hui.

Il existe plusieurs variantes de rééditions:

- Le portage, où le jeu est réédité sans aucune modifications (tout au plus quelques adaptations, par exemple changer l'aspect des boutons pour correspondre avec l'interface de la console), voir émulé. C'est le cas par exemple de la *Virtual Console* sur Wii, WiiU et 3DS.
- Le remaster, où le jeu est réédité avec des modifications mineures (résolution modifiée, petits contenus ajoutés, textures. modifiées, etc.). C'est le cas de *The Legend of Zelda: The Wind Waker HD* ou encore *Shadow of the Colossus* (qui a eu droit à deux remaster différents).
- Le remake, où le jeu est refait intégralement de zéro mais en se basant en plus ou moins grande partie sur le cahier des charges et intentions artistiques du jeu de base. Par exemple, *Metroid Zero Mission* est un remake du premier *Metroid* sur NES.

Le cas des portages crée généralement un débat et des recherches au sein de la communauté de speedrun pour déterminer quelle est la version la plus intéressante pour speedrunner le jeu. On pensera notamment au lag qui peut être différent d'une plateforme à l'autre, ou à l'exclusivité de certains glitches, en raison d'une émulation imparfaite (par exemple, le *Get Item Manipulation* sur *The Legend of Zelda: Ocarina of Time* n'est pas possible sur toutes les plateformes).

Dans le cas du remaster, il peut arriver que certains glitches soient corrigés, parfois même au profit de nouveaux glitches. Le cas le plus célèbre probablement est *The Legend of Zelda: The Wind Waker HD* qui a vu la disparition du storage présent sur la version SD, mais au profit de l'apparition de l'*Item Slide* qui a rendu possible le *Barrier Skip*, jusqu'alors une des plus grandes arlésiennes du speedrun *(Item Slide* qui n'est pas possible sur la version SD car c'est un des gimmick ajoutés sur la version HD qui permet de le faire).

Enfin, les remakes étant à considérer comme de nouveaux jeux intégralement, il est rare qu'ils partagent des glitches avec leurs anciennes versions. Néanmoins, cela ne les empêche généralement pas d'avoir leur propres glitches.

# Zip