

Christian Forget

Processus de planification pour l'expérience test de jeu

Guide de recherche

Le 24 avril 2023

Table des matières

Table des matières.....	2
Affordances et limitations humaines	3
Affordance	3
Limitations Humaines.....	9
Le cycle itératif, agile et incrémental.....	19
L' <i>onboarding</i> du joueur.....	22
Les outils d'analyse et de prise de donnée.....	29
Protéger face aux fuites de jeu (<i>Leaks</i>).....	29
Les outils et méthode pour prendre des notes.....	32
Trucs et astuces sur les outils.....	36
Les types de question à poser et les informations à conserver	38
Les étapes d'un test de jeu	43

Affordances et limitations humaines

Affordance

Il est **important de connaître les affordances pour la préparation de test de jeu parce que ce qui est intuitif n'a pas besoin d'être appris ou réappris par le joueur**. Le plus d'affordance perceptible dans le jeu, le moins de choses qui doivent être expliquées au joueur. Il est important de polir les signifiants dans le jeu pour que le joueur comprenne les mécaniques du jeu.

Rex Hartson, dans son article « Cognitive, physical, sensory, and functional affordances in interaction design », classe les affordances en quatre différents types :

- **Les affordances physiques**
 - Les **affordances physiques contiennent les caractéristiques qui facilitent l'action physique de faire quelque chose**. Comme exemple :
 - Pour un jeu mobile, avoir les boutons proches de la position de repos du pouce pour faciliter l'action d'appuyer dessus.
 - Avoir les boutons du menu ou dans le jeu assez large pour être facilement cliquable par le joueur ou la souris.

Les boutons dans l'image ci-dessous pourraient être plus larges pour faciliter le joueur à cliquer dessus.



- **Les affordances cognitives**

- Les **affordances cognitives aident le joueur à découvrir, apprendre, comprendre et ainsi décider ce qui peut être fait avec quelque chose.** Comme exemple :
 - Avoir du texte sur un bouton qui explique ce que le bouton fait.
 - Les formes des icônes dans l'interface utilisateur.
 - Le design des ennemis démontre leur fonctionnalité.
- Selon **Celia Hodent**, le pilier de convivialité *Form-follow-Fonction* est au sujet des affordances cognitives.



- **Les affordances sensorielles**

- Les **affordances sensorielles sont les caractéristiques aidant le joueur à ressentir quelque chose.** Ces affordances aident le joueur à voir, entendre et sentir des éléments de jeu. Les affordances sensorielles devraient être facilement vues, entendues ou senties. Comme exemple :
 - Avoir une taille de typographie assez large pour être capable de facilement être lue par le joueur.
 - Jouer un son précis lorsqu'un ennemi va faire une attaque précise permet au joueur de comprendre que cette attaque approche.

- Selon Celia Hodent, le pilier de convivialité *Clarté* est au sujet des affordances sensorielles.

Example *The Legend of Zelda: Ocarina of Time* (Nintendo, 1998):

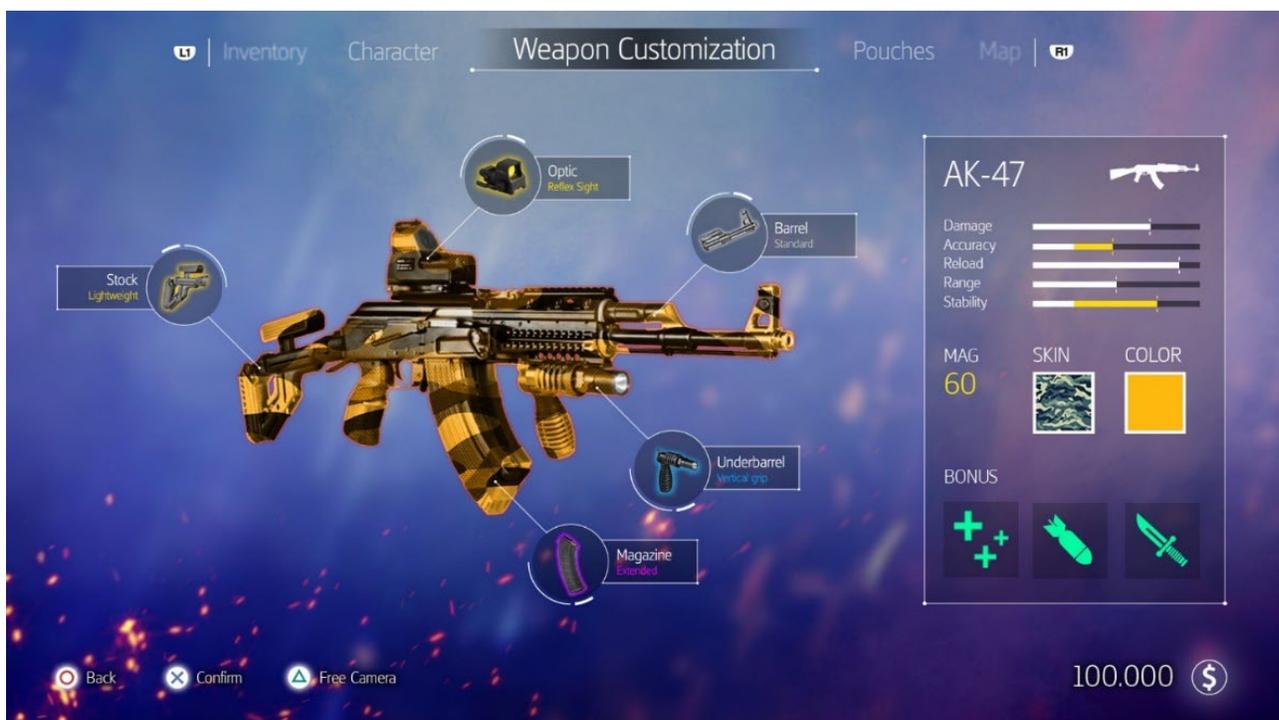
<https://www.youtube.com/watch?v=wOFVrjL-XBM>

Example *Five Nights at Freddy's* (Scott Cawthon, 2014):

<https://www.youtube.com/watch?v=9JLzR1hENNg>

- **Les affordances fonctionnelles**

- **Les affordances fonctionnelles sont les caractéristiques qui aident le joueur à accomplir des tâches.** Comme exemple :
 - Un bouton qui permet au joueur de filtrer les objets de son inventaire.
 - Une fonctionnalité qui permet de comparer les statistiques de deux objets.



Selon **William Gaver**, les **affordances** sont les **caractéristiques des objets qui sont construites pour l'interaction avec un humain**. Selon Gaver, il est possible de correspondre à la possibilité d'action apportée par l'affordance avec la perception humaine.

Une **affordance implique la «complimentarité» entre l'organisme qui joue et l'environnement**. Pour qu'une affordance fonctionne, il faut que les propriétés de l'objet rendent l'action possible avec l'organisme qui est équipé d'une certaine façon.

Comme exemple, **le fait que la poignée d'une tasse peut être prise va dépendre des caractéristiques de la personne qui la prend** (grandeur, taille de la main, etc.).

Dans le cas d'un jeu vidéo, **la force des affordances peut dépendre de la connaissance du testeur** face à ce type de jeu ou des jeux vidéo en général.

Pour les jeux vidéo, **Gaver mentionne le fait qu'un outil (clavier, souris, main, manette, etc.) offre beaucoup d'affordance possible d'interaction**.

Pour Gaver, des **affordances existent même si le joueur n'a pas perçu celle-ci**. Le fait que le joueur n'a pas soif ne change pas le fait qu'un verre sert à contenir de l'eau. Cependant, le fait que les affordances sont importantes est la raison pourquoi elles devraient être perçues.

	Affordance	Pas d'affordance
Information perceptible	Affordance perceptible	Fausse affordance
Information non perceptible	Affordance dissimulée	Rejet correct

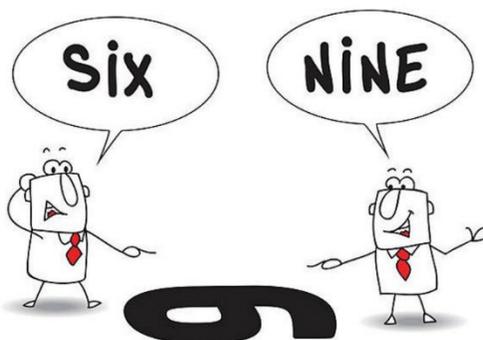
- **Affordance perceptible**
 - Une affordance existante est représentée par une information perceptible.
 - Ex. : Une porte visible est ouvrable

- **Affordance dissimulée**

- Si une affordance n'est pas représentée par de l'information perceptible, c'est une affordance dissimulée.
- S'il n'y a pas d'information qui démontre une affordance, le joueur devrait être capable de déduire l'existence de l'affordance via d'autres moyens.
 - Ex. : Une porte invisible est ouvrable
- **Fausse affordance**
 - Si de l'information perceptible suggère une affordance qui n'est pas existante, cela crée une fausse affordance.
 - Une fausse affordance pourrait frustrer le joueur si le jeu lui démontre régulièrement de la fausse information.
 - Ex. : Une porte visible n'est pas ouvrable
- **Rejet correct**
 - Le joueur ne va pas penser qu'une affordance existe s'il n'y a pas d'élément perceptible dans l'environnement.
 - Il n'y a pas de porte.

Selon Gaver, **utiliser des affordances perceptibles permet aux développeurs de désigner un système facile d'utilisation pour le joueur.**

Il est également important de comprendre que les affordances sont déterminées par la culture, le contexte social, les intentions et les expériences du joueur.



Selon **Celia Hodent**, il est **extrêmement important de polir les affordances rapidement dans le développement du jeu**. Il faut s'assurer lors de test que le joueur sache quelles fonctionnalités fonctionnent et lesquelles ne fonctionnent pas lors du test de jeu.

Éviter les fausses affordances est également recommandé pour ne pas frustrer le joueur lors des tests.

Les designers doivent travailler avec les chercheurs utilisateurs pour tester les affordances rapidement.

- *Est-ce qu'il est facile d'appuyer sur les boutons ?*
- *Les designs des ennemis démontrent-ils bien leur habileté ?*
- *Est-ce que les signes et rétroactions sont bien perçus par le joueur ?*
- *Est-ce que la forme des icônes et des objets explique leur fonctionnalité au joueur ?*
- *Est-ce que le joueur comprend qu'elle est son but facilement sans un paragraphe de texte ?*
- *Est-ce que les tâches des joueurs seraient plus faciles si d'autre fonctionnalité était disponible ?*
- *etc.*

Il faut **également désigner les systèmes et caractéristiques sous la lentille de l'affordance** pour aider à définir comment démontrer ces systèmes de la façon la plus intuitive possible.

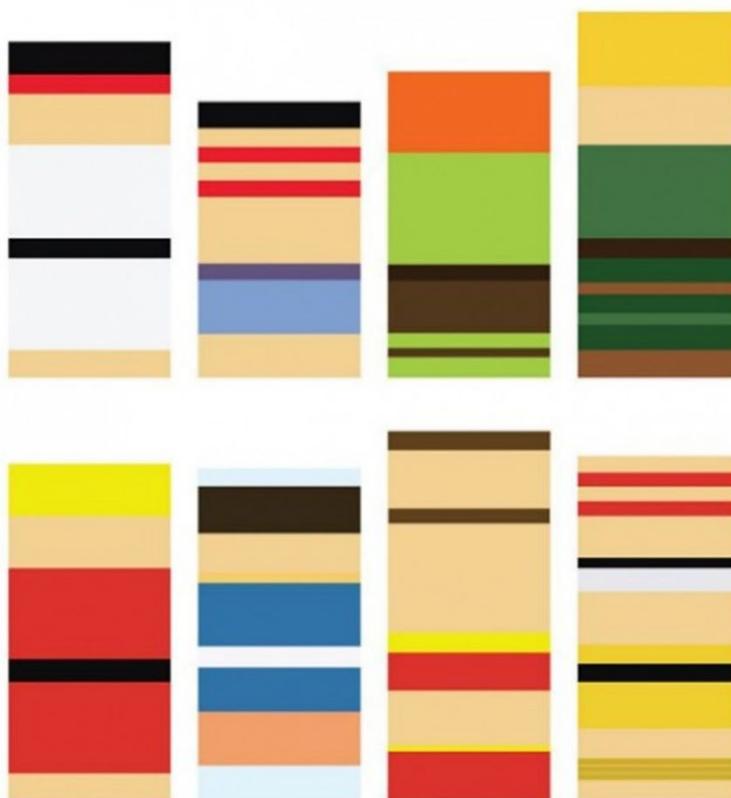
Finalement, durant le test de jeu, il est important de garder les différents types d'affordance en tête pour ainsi poser des questions utiles aux testeurs du jeu.

Limitations humaines

Perception humaine

Selon **Celia Hodent**, la perception humaine est limitée, car celle-ci est influencée par la cognition de l'être, ce qui la rend subjective. Elle est également influencée par les expériences passées du joueur. Dépendamment du public cible pour les tests, il se peut que le testeur ne comprenne pas une fonctionnalité, une icône, etc., vu la perception différente entre les développeurs et le joueur.

Il est également important de prendre en note les gens souffrants de handicap (les gens souffrants de daltonisme auront plus de difficulté de reconnaître les personnages plus bas) lors de test pour la perception.



La perception est également influencée par la situation, la culture et l'expectation du joueur dans le contexte du test.

Dans l'exemple plus bas, le centre peut être interprété comme un B ou un 13 dépendamment de la lecture en vertical ou en horizontal.



Célia Hodent utilise l'exemple du *floppy disk* pour l'icône de sauvegarde qui n'est pas aussi intuitive pour les jeunes joueurs contrairement aux joueurs plus vieux.

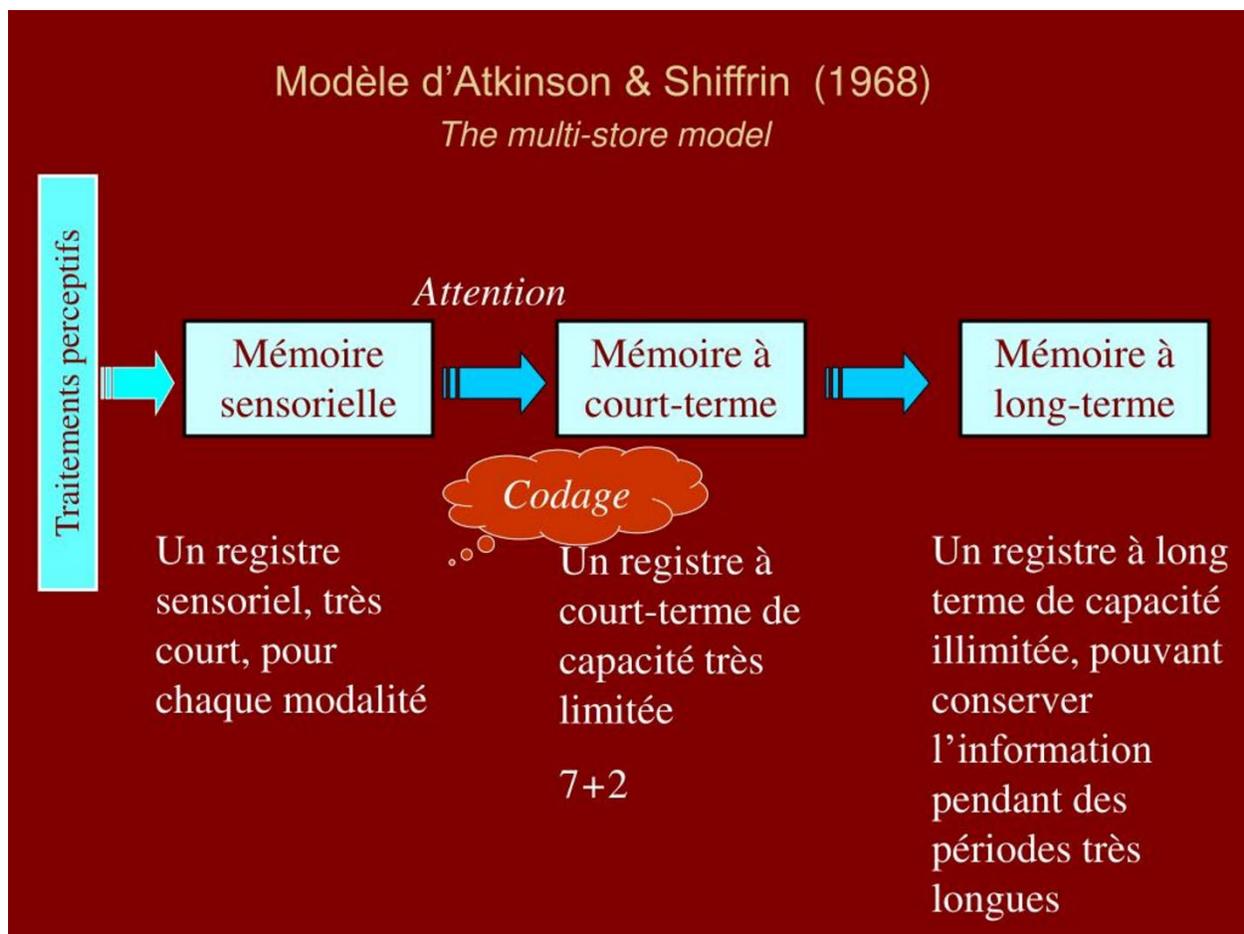


Enfin, la perception a également des limitations physiques, plus particulièrement dans le cas de la vue où les détails ressortent plus sur les éléments aux centres de l'œil. Les éléments du jeu dans la vision périphérique prennent plus de temps à être discernés pour le joueur, ce qui est à prendre en considération lors du développement de l'interface ou du monde.

Dans le cas de test de jeu, il est important de régulièrement tester l'iconographie, les sons, les effets visuels lors de tests avec de nouveau testeur pour s'assurer que le client cible comprenne les mécaniques et l'interface du jeu. Il est également important de se rappeler que *l'expérience console et PC n'est pas le même*, alors faire des tests sur les deux types de plateformes est primordial.

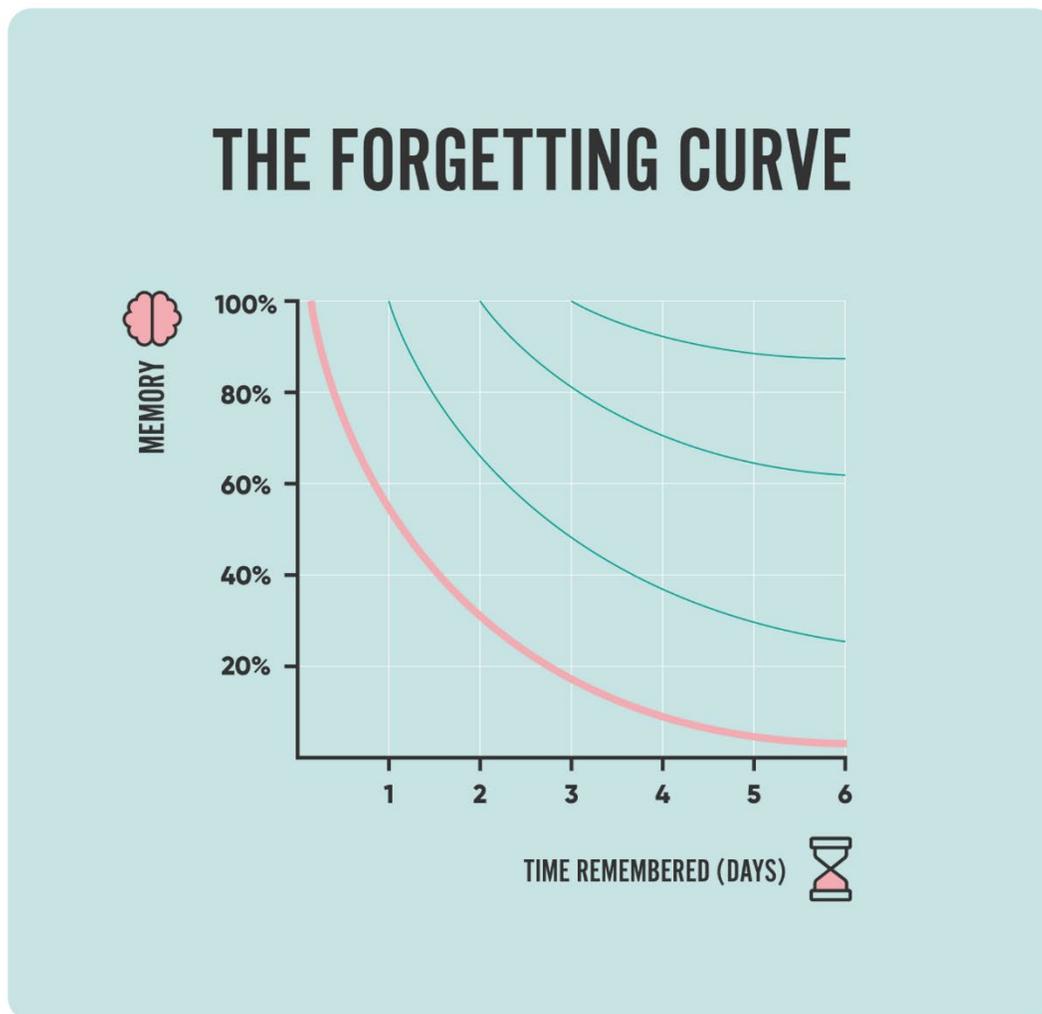
Mémoire

Selon le modèle d'Atkinson et Shiffrin, utilisé par Celia Hodent, il y a 3 types de mémoires :



Tout d'abord, la plus grosse limitation de la mémoire est le laps de mémoire. Il faut s'assurer que le joueur se rappelle régulièrement l'information lors de test de jeu. Si le jeu offre régulièrement des défis contextuels qui augmentent en difficulté, le joueur aura une plus grande facilité à se rappeler de l'information apprise.

Développé par **Herman Ebbinghaus**, la *forgetting curve* peut être utilisée pour visualiser la vitesse dont laquelle quelqu'un perd la mémoire sur un élément appris, que ce soit une phrase, une image, une information, etc.



Il y a plusieurs façons de rendre l'apprentissage de l'information mémorable pour le joueur lors de test de jeu :

- *En associant l'information récente avec des concepts déjà connus, il est plus facile pour le joueur de se rappeler de la nouvelle information.*

- *Les informations devraient être classées logiquement dans l'interface et dans les menus.*
- *Utiliser des icônes qui représentent bien les informations, mais ne pas être gêné d'utiliser des mots également.*
- *S'assurer que l'information est importante pour le joueur dans le contexte du jeu.*

Également, la façon dont les questions sont posées peut changer la perception dont le joueur va se rappeler l'information.

- *Ex. : Utiliser le mot « Anéantir » au lieu de « Tuer » dans le cas de la destruction d'un ennemi peut ajouter un biais de la part du joueur.*

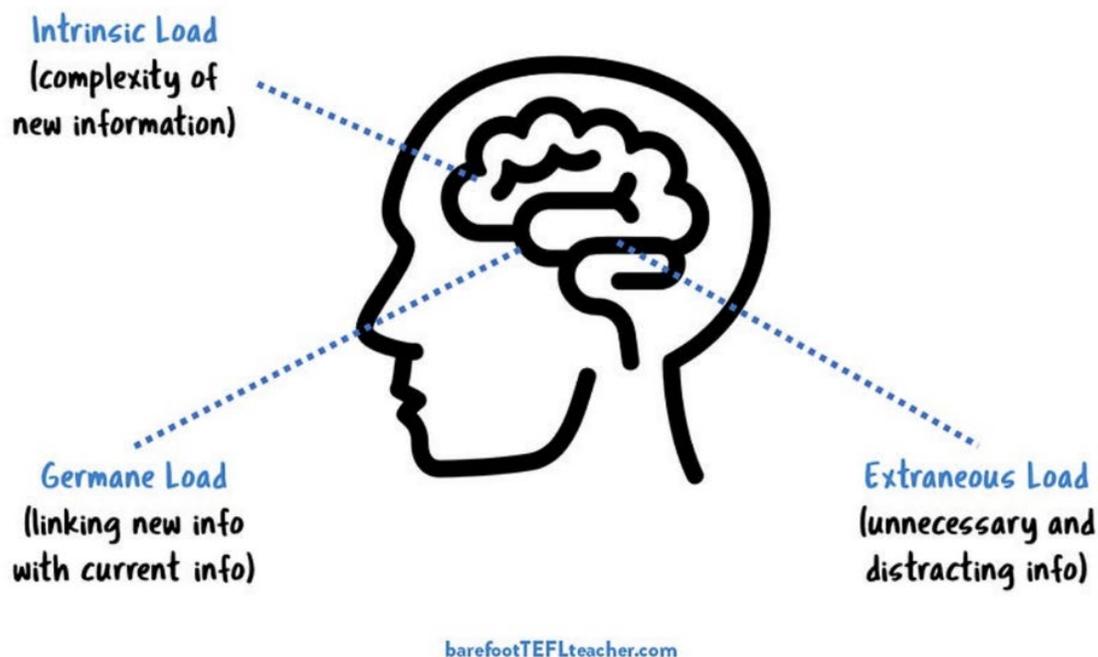
Dans le cas de test de jeu, il est important de s'assurer que le joueur se rappelle les informations cruciales lors des tests. Vu le temps court qui est pris avant qu'une information puisse être oubliée, mettre le joueur dans une situation que l'on veut tester rapidement permet à celui-ci d'avoir les informations claires en tête. Également, donner un peu de temps avant que l'information soit utile peut permettre de tester si l'information apprise est intuitive ou non.

Attention

L'être humain a très peu de ressource attribuée à l'attention, il est beaucoup plus efficace s'il travaille sur une seule tâche. Le plus de tâches que quelqu'un fait, le plus de chance de commettre une erreur et le plus grand temps pris pour faire la tâche en question.

Si une tâche est nouvelle ou complexe, le plus de ressource attentionnelle sera demandée de la part du joueur. S'il y a plusieurs tâches à faire en même temps, le joueur pourrait être distrait ou irrité face au jeu.

Cognitive Load Theory



L'attention, comme la mémoire et la perception, est influencée par les connaissances antérieures et les expertises du joueur.

Pour le test de jeu, limiter les demandes cognitives du joueur en ne créant pas de situation demandant du *multi-tasking* est une bonne façon de diriger l'attention du joueur vers la tâche qu'il est en train de faire. C'est parce que le *multi-tasking* requiert non seulement la coordination pour accomplir la tâche, mais également le management de temps pour coordonner les tâches (ceci est connu comme *underadditivity*).

Également, il faut prendre en considération l'*effet Stroop* qui représente la tâche du cerveau de filtrer l'information inutile montrée à l'écran pour mettre l'importance sur les informations importantes à la situation du jeu.

Red Green Purple
Brown Blue Red

Purple Red Brown
Red Green Blue

Enfin, il faut faire attention à l'**aveuglement non attentionnel** où quelqu'un est tellement concentré sur un élément qu'il manque d'autres informations importantes qui passent en face de lui.

L'exemple le plus connu de l'aveuglement non attentionnel est l'exemple recherché par Simons et Chabris.

<https://www.youtube.com/watch?v=vJG698U2Mvo>

Une autre information importante à propos de l'aveuglement non attentionnel est qu'ajouté plusieurs types de canaux (audio, visuel, haptique, etc.) pour démontrer qu'une seule information augmente l'aveuglement non attentionnel.

Pour un test de jeu, il faut s'assurer que le joueur ne soit pas distrait hors du jeu et que son attention ne soit pas divisée entre plusieurs tâches. Dans le cas d'un test sur une fonctionnalité en particulier, s'assurer que le joueur ne doit pas apprendre une autre fonctionnalité pour comprendre celle testée peut permettre au joueur d'être totalement dirigé vers les informations importantes.

Motivation

Selon **Celia Hodent**, il y a quatre types de motivations :

- **Les motivations implicites et les véhicules biologiques**
 - Le *but de ces motivations est spontané et intrinsèque, étant de garder une balance intérieure* (Ex. : Manger, boire de l'eau, dormir).
- **Les motivations environnementales et les véhicules appris**
 - Ici, nous parlons *de motivations qui sont créées par l'environnement ou apprises par erreurs ou réussite* (Ex. : Aller travailler ou ne pas mettre sa main sur un four en marche).
- **Les motivations intrinsèques ou des besoins cognitifs**
 - Les *motivations intrinsèques sont les besoins qui sont réalisés dans le but de plaisir personnel. Le but est de faire la tâche en elle-même.* (Ex. : Conduire un nouveau véhicule ou jouer à un jeu vidéo).
- **Les personnalités et besoins individuels**
 - Les différences entre chaque personne affectent ce que chaque personne considère intérieurement motivant sur un niveau cognitif.
 - *Il y a 5 traits de personnalités importants à chaque personne pour le test de jeu vidéo*, ces 5 traits sont utilisés dans le modèle de recherche **OCEAN** :
 - **L'ouverture à l'expérience** : À quel point est-ce que la personne est curieuse ? Les gens qui ont un bas score sont conventionnels et pratiques tandis que les hauts scoreurs sont créatifs et imaginatifs.
 - **La Conscience** : À quel point est-ce que la personne est organisée et efficace ? Les gens ayant un bas score sont spontanés et sans-soucis tandis que les hauts scoreurs sont organisés et se dirigent eux-mêmes.

- **L'Extraversion** : À quel point la personne est énergétique et sortante ? Les personnes qui scorent bas sont réservées et calmes tandis que les hauts scoreurs sont pleins d'enthousiasme et extraverties.
- **L'Agréabilité** : À quel point la personne est amicale et compassionnée ? Les personnes qui ont un bas score sont suspicieuses et hostiles tandis que les hauts scoreurs sont empathiques et font confiance.
- **Neurotoxique** : À quel point la personne est sensible et nerveuse ? Les personnes qui ont un bas score sont stables émotionnellement tandis que les hauts scoreurs sont cible de colère et d'anxiété.

Pour les tests de jeu, **avoir une motivation intrinsèque de la part du joueur ou extrinsèque de la part de la compagnie augmente la performance du testeur. La motivation intrinsèque va dépendre du contexte de la tâche et la motivation extrinsèque va dépendre du contrôle de l'autonomie du joueur face à la tâche en question.**

Cependant, il y a certaines choses à savoir lors de la formation de test de jeu :

- *N'importe quelle récompense est mieux qu'aucune récompense.*
- *Si la tâche testée est répétitive, avoir une récompense extrinsèque peut augmenter la performance du testeur.*
 - *Plus la tâche est compliquée et répétitive, plus la récompense devrait être élevée.*
- *La performance d'une tâche complexe peut être améliorée s'il y a une motivation intrinsèque.*
 - *Laisser le testeur être autonome et le laisser tester ses compétences peut lui permettre de monter la motivation intrinsèque.*
- *Quand la qualité du test des tâches est évaluée ou quand la créativité, le travail d'équipe et le comportement éthique, la récompense extrinsèque ne devrait pas être saillant.*
- *Avoir un bon flow est la façon optimale de monter la motivation intrinsèque du testeur, les tests ne devraient pas être trop faciles ou trop difficiles*

Le cycle itératif, agile et incrémental

Selon **Celia Hodent**, le cycle itératif débute pendant la période de préproduction et n'arrête pas tant que le jeu n'est pas livré, dans le cas d'un jeu « live-service », le cycle itératif ne finit jamais. Pour qu'un cycle itératif puisse débiter, l'équipe de développement doit avoir décidé :

- *Les piliers du jeu*
- *Les boucles de jouabilités*
- *Les caractéristiques marquantes du jeu*
- *L'audience cible*
- *Le modèle d'économie (Free-to-play, pay-to-play, online, single player, etc)*

Pour Celia Hodent, le cycle itératif d'une **caractéristique du jeu** consiste à :

1. *Désigner la caractéristique*
2. *Prototyper ou implémenter la fonctionnalité*
3. *Tester la fonctionnalité*
4. *Analyser les résultats des tests*
5. *Définir les changements de design à faire face aux rétroactions des tests de jeu*

Les premières boucles itératives devraient être sur papier, ensuite, c'est la transition vers un prototype interactif et c'est uniquement dans ce prototype que la caractéristique devrait être implémentée pour de futurs tests. Ceci est important parce que si une fonctionnalité doit être coupée ou modifiée tard dans la production, cela peut causer des problèmes sur le restant de la production (*ripple-effect*). C'est pour **cela que même si un prototype papier semble inutile, il peut permettre d'économiser énormément de temps et d'argent pour le développement d'un jeu.**

Ensuite, il faut **faire attention à la vitesse de la production et du temps limité**. Si une fonctionnalité est faite trop rapidement ou le scope n'est pas baissé suivant l'avancement de la production, une fonctionnalité peut ne pas avoir été bien testée et peut être mal implémentée dans un projet.

Éventuellement (préférentiellement lors de la fin de la préproduction, mais souvent lors de la production), le cycle itératif n'impactera pas uniquement une simple caractéristique du jeu, mais le jeu en entier. C'est dans cette éventualité que l'équipe **découvre les fonctionnalités qui ne sont pas utiles pour l'expérience désirée rapidement** et puisse les retirer ou modifier pendant la production.

Pour le *test de jeu*, **il est important de regarder si le joueur non seulement comprend comment la fonctionnalité marche, mais s'il interagit avec la fonctionnalité en question**. Il est extrêmement important de bien analyser les informations recueillies lors de test pour regarder quelles fonctionnalités devraient être gardées et quelles fonctionnalités ne sont pas importantes pour l'expérience visée ou qui dérange le joueur. Si une fonctionnalité ne peut pas être coupée, alors, il faut s'assurer qu'elle est compréhensible pour le joueur lors de test.

Il est important de garder en tête qui est le public cible, alors, l'équipe pourra avoir ainsi une approche « joueur premier » et comprendre quelles fonctionnalités sont à garder et quelles fonctionnalités sont à retirer.

Victor Rouvroy de *TheTribe*, explique que le cycle itératif débute avec un MVP (version minimale du produit) qui sera amélioré petit à petit selon le retour des utilisateurs. Selon lui, le design itératif consiste à faire une série d'opérations de manière cyclique.

Chez *TheTribe*, un cycle prend entre une à deux semaines.

Pour Victor, **l'approche itérative fonctionne pour 3 raisons principales :**

- *Le cycle itératif permet de dé-risquer la production, en ajoutant et testant rapidement des fonctionnalités avec les joueurs, il est possible de rapidement valider les hypothèses et repérer les défauts dans le produit.*
- *Le cycle itératif permet d'économiser du temps et de l'argent en validant rapidement les solutions auprès de la clientèle, permettant de savoir si le produit sera populaire.*
- *Le cycle itératif permet de mettre au point un produit centré sur l'utilisateur, à la fin de chaque itération, il est possible de recevoir de la rétroaction des clients en eux-mêmes. Permettant de s'assurer que le produit répond aux demandes de la clientèle.*

Finalement, Victor Rouvroy sépare un cycle itératif en 5 étapes :

1. **La délimitation du périmètre et des objectifs** : La toute première étape du sprint consiste à dessiner les contours du projet avec l'ensemble de l'équipe. Pour cela, il faut définir la cible et les objectifs, ainsi qu'identifier le problème.
2. **L'analyse et conception** : C'est l'heure du *brainstorming* avec toutes les parties prenantes. Lors de cette phase, l'équipe va récolter le maximum d'idées pour trouver des solutions au problème identifié.
3. **Mise en œuvre** : Une fois qu'une solution sera trouvée, l'équipe devra construire un prototype, une première version du produit avec les fonctionnalités minimales.
4. **Les tests de jeu** : Pour valider une hypothèse, il faudra tester le prototype auprès des utilisateurs en récoltant le maximum de rétroactions. Pour cela, tous les moyens sont bons : *social listening*, analyse des métriques d'une application, entretiens utilisateurs, etc.
5. **Feedback et révisions** : Grâce aux informations collectées, il sera possible d'ajuster le produit afin de l'améliorer et faire qu'il soit davantage en adéquation avec les besoins des utilisateurs.

Selon [Julia Martins](#) du site Asana, la différence entre le cycle itératif et la conception incrémentale est :

- **Cycle itératif** : une approche essai-erreur (par tâtonnements) : avancez pas à pas et le projet se bonifie au fil du temps grâce aux ajustements effectués.
- **Conception incrémentale** : l'équipe ajoute de nouvelles fonctionnalités et améliore le produit à partir de la version initiale de ce dernier ou du premier livrable. La mise en œuvre de ce type de démarche invite les équipes à produire à dessein une ébauche de leur livrable final afin de pouvoir lancer au plus vite son perfectionnement. L'équipe se lance donc dans une phase d'itération pour enrichir son livrable initial en ajoutant des incréments, chacun d'eux apportant davantage de fonctionnalités à la version initiale. L'équipe répète ce processus jusqu'à ce que le produit livré dispose de toutes les fonctionnalités nécessaires.

Également Julia Martins mentionne les avantages et les inconvénients du cycle itératif :

Avantages :

- **Une efficacité accrue.** Les processus itératifs permettent d'approcher par tâtonnement et aideront à atteindre le résultat souhaité plus rapidement.
- **Une collaboration optimisée.** Les membres de l'équipe peuvent travailler plus efficacement en équipe sous un cycle itératif.
- **Une adaptabilité renforcée.** Il est plus facile de changer la taille du livrable dans un cycle itératif, également, les membres de l'équipe apprennent de nouvelles capacités et acquis lors des nombreux cycles.
- **Une solution plus efficace.** Il est facilement possible de modifier la taille du projet dans un cycle itératif.
- **La possibilité de travailler sur des tâches connexes en parallèle.** L'équipe peut travailler sur plusieurs tâches en même temps vu que celle-ci n'est pas dépendante les unes des autres.
- **Des risques limités à l'échelle du projet.** Les risques sont gérés et identifiés lors de chaque cycle itératif du projet.
- **Des retours utilisateur plus fiables.** Le cycle itératif permet d'avoir plus de livrables qui sont montrables aux utilisateurs, ce qui leur permette de donner leurs opinions plus souvent.

Inconvénients :

- **Un risque accru de souffrir d'une dérive des objectifs.** Il se peut que le projet se développe d'une manière inattendue et ainsi la taille du projet augmentera.
- **Une rigidité de la planification et des exigences.** Changer les objectifs du projet pendant le processus itératif peut causer la perte de rythme de travail et faire entrer l'équipe dans des phases d'itération qui ne rentre pas dans le cadre des objectifs fixés au départ.
- **Des délais vagues.** Il est difficile d'avoir un calendrier fixe lors de l'utilisation de processus itératif vue que la durée des tests effectués pour les livrables partiels peut varier, avec là aussi des répercussions sur la chronologie générale du processus itératif en question.

***L'onboarding* du joueur**

L'onboarding est le processus où un nouvel employé ou joueur (dans le cas du jeu vidéo) est introduit et intégré au rôle et aux fonctionnalités de l'organisation ou du jeu.

Selon *SteamSpy*, **20 % des joueurs qui jouent à un jeu arrêtent de jouer après seulement 1 heure de jeu.** Alors, la première heure de jeu est cruciale pour le *First-Time-User-Experience*, où la majorité de l'onboarding cette passe.

Cependant, **il est important de mentionner que du onboarding est nécessaire à chaque fois qu'une nouvelle fonctionnalité, mécanique ou système est montré au joueur.** L'onboarding doit être intéressant pour le joueur, il faut exciter la curiosité de celui-ci.

Il faut que le joueur se sente compétent et autonome, il faut aussi que le joueur puisse améliorer ses compétences sur le long et court terme tout en s'assurant que les demandes cognitives ne soient pas trop intenses pour le joueur.

Une façon de rendre l'onboarding à la fois engageant et efficient est de créer un plan d'onboarding.

La façon de créer un plan d'onboarding est :

- ***Créer une liste de tous les éléments que le joueur doit apprendre***
- ***Catégoriser les éléments par les systèmes généraux***
- ***Définir les éléments les plus importants pour que le joueur apprenne l'expérience principale.***
- ***Mettre la liste dans un tableau avec les éléments tous dans une colonne, chaque élément sur une ligne différente.*** Les autres colonnes devraient avoir les informations suivantes :
 - **Catégorie primordiale** (Système)
 - **Niveau de priorité** : Définir les mécaniques, systèmes et fonctionnalités les plus importantes pour l'expérience du jeu.
 - 0 = Fonctionnalité critique
 - 1 = Fonctionnalité importante
 - 2 = Nice to Have
 - 3 = C'est correct si le joueur ne comprend jamais la mécanique

- **Quand** : À quel moment dans le jeu est-ce que l'élément devrait être appris ? Utiliser des chiffres *integer* pour quand l'élément devrait être appris (ex. : 1 est la première mission/15 premières minutes, 2 est la deuxième mission/30 premières minutes). Il peut y avoir plusieurs éléments sur le même chiffre.
- **L'ordre de tutoriel** : L'ordre dans lequel l'élément doit être appris, aucun élément ne devrait avoir le même chiffre, mais il peut y avoir des décimales, ainsi, il est possible de sortir de la table selon l'ordre (ex. : 1,1, 1,2, 1,3, 2,1, 2,2, 2,3, etc.).
- **Difficulté** : À quel point la fonctionnalité est difficile à apprendre (simple, modéré, difficile)
- **Pourquoi** : Définir pourquoi la fonctionnalité est importante à apprendre et comment cela va rendre le joueur plus compétent. Donner un exemple de comment la fonctionnalité va être utilisée va aider à créer des situations importantes pour enseigner la mécanique (ex. : Si je n'apprends pas à fabriquer une arme, je ne pourrais pas vaincre les ennemis).
- **Comment** : Comment la fonctionnalité sera apprise (Ex. : Via l'interface, en effectuant un exercice de saut, avec des tutoriels dynamiques)
- **Emballage narratif** : Lorsque les éléments sont placés chronologiques en utilisant l'ordre de tutoriel, il sera plus facile d'écrire une histoire qui supporte le plan d'*onboarding*.
- **Feedback UX**: s'il y a un spécialiste UX dans l'équipe, il peut mentionner les résultats tôt dans le développement ou il peut anticiper les difficultés que les joueurs vont rencontrer.

La liste va être massive, mais il sera plus facile de comprendre quels éléments sont nécessaires pour une bonne intégration du joueur.

Exemple fait ici avec le jeu « *Claws* » développé pour le projet commun :

Éléments à apprendre	Catégorie primordiale (Systémique)	Niveau de priorité	Quand	L'ordre de tutoriel	Difficulté	Pourquoi	Comment	Emballage narratif	Feedback UX
Se déplacer avec Claws	Mouvement	0	1	1,1	Simple	Si je n'apprends pas à déplacer le personnage,	Apprendre en le faisant. Image	Claws vient de se réveiller et doit se dégoûter les jambes.	

						je ne peux pas explorer le niveau.	tutoriel contextuel		
Entrer dans le sol	Sol	0	1	1,3	Simple	Si je n'apprends pas à entrer dans le sol, je ne pourrais pas traverser la roche qui bloque le chemin.	Apprendre en le faisant Image tutoriel contextuel	Une lignée d'arbre bloque le chemin de Claws, il doit creuser sous celle-ci pour traverser.	
Sortir du sol	Sol	0	1	1,4	Simple	Si je n'apprends pas à sortir du sol, je ne pourrais pas atteindre certaines plateformes.	Apprendre en le faisant Image tutoriel contextuel	Claws est dans le sol et doit en ressortir pour continuer son chemin.	
Sauter avec Claws	Mouvement	1	1	1,2	Simple	Si je n'apprends pas à sauter, je ne pourrais pas monter les marches dans la section tutorial.	Apprendre en le faisant Image tutoriel contextuel	Une petite roche/arbre bloque le chemin de Claws, il doit sauter par-dessus pour continuer son chemin.	
Attaquer	Mouton	0	2	1,7	Modéré	Si je n'apprends pas comment attaquer, je ne saurais pas comment éliminer les moutons.	Apprendre en le faisant Image tutoriel contextuel	Un mouton est sur le chemin de Claws et Claws a très faim, il est temps pour lui de manger le mouton.	Il ne faut pas que le mouton puisse s'échapper de Claws.
Traquer les moutons/interagir avec les couronnes	Mouton	2	2	1,6	Simple	Si je n'apprends pas comment traquer les moutons, il sera plus difficile de les trouver dans le niveau.	Apprendre en le faisant Image tutoriel contextuel	Claws entre en contact avec une couronne et cela lui permet de sentir un mouton à proximité	Il faut que le joueur comprenne la liaison entre le talisman et l'apparition du mouton jaune.
Gagner de la vitesse en sortant et entrant dans le sol	Sol	0	2	1,5	Difficile	Si je n'apprends pas comment gagner de la vitesse en entrant et en sortant du sol, il sera difficile de	Apprendre en le faisant Image tutoriel contextuel	Claws doit sauter par-dessus un gros trou pour continuer. Cependant, il n'est pas assez vite normalement, en creusant et	Il se peut que cette mécanique nécessite plusieurs essais de la part du joueur pour bien comprendre.

						traverser les gros trous dans le niveau.		sortant régulièrement du sol, il gagne assez de vitesse pour passer par-dessus le trou.	
--	--	--	--	--	--	---	--	--	--

Lors de la création du plan d'*onboarding*, l'équipe commence en listant une catégorie primordiale (systémique) et un niveau de priorité. Il est également important de connaître le public cible et leur connaissance avant le test de jeu va être importante pour connaître cette information.

Une mécanique qui est commune avec de nombreux jeux sera facile à apprendre et une mécanique unique à mon jeu sera beaucoup plus difficile à apprendre.

Il est important de comprendre qu'un apprentissage distribué est plus facile qu'un apprentissage en masse, alors, il est important de distribuer l'apprentissage pour que toute l'information ne soit pas apprise d'un coup.

Si possible, évitez de devoir apprendre deux mécaniques difficiles l'une à côté de l'autre.

Une fois que toutes les informations sont insérées dans le tableau, **filtrer la colonne « L'ordre de tutoriel » en ascendant va révéler le plan d'*onboarding*.**

Pour apprendre les mécaniques, **le joueur devrait l'apprendre dans un contexte important et qui a du sens.**

Le tableau démontre également quelle fonctionnalité nécessite le plus grand nombre de ressources, les fonctionnalités simples peuvent être apprises avec une boîte de texte alors que les autres mécaniques nécessiteront de l'apprentissage dans le monde du jeu.

Suivant les tests de jeu, il se peut que les éléments changent de place vue que les mécaniques que l'équipe pensait simples peuvent se retrouver beaucoup plus difficiles que prévu pour le joueur.

Il est également possible d'utiliser le tableau pour savoir quels tutoriels peuvent être affichés sur les écrans de chargement. Quoiqu'il soit important de ne pas mettre trop d'information sur les écrans de chargement (selon [Celia Hodent](#), 3 astuces devraient être le maximum).

Si un écran de chargement est utilisé pour parler des règles du niveau, mettre l'emphase sur pourquoi (objectif du niveau principal) du niveau plutôt que le comment (comment me rendre à l'objectif) ou le quoi (objet avec lequel le joueur doit interagir). Apprendre des mécaniques via des écrans de chargement n'est pas une façon optimale de faire apprendre les mécaniques cependant, c'est une bonne façon d'occuper le joueur pendant le chargement du jeu. Il y a d'autre façon de faire passer l'écran de chargement plus vite :

- *Barre de progression de chargement avec animation*
- *La barre de progression accélère au lieu de ralentir*
- *Si le joueur peut faire quelque chose pendant le chargement du niveau/jeu*

Se rappeler des limitations humaines en perception, attention et mémoire lors du design des astuces de tutoriel pendant l'écran de chargement.

Selon [Page Laubheimer](#), un Senior UX spécialiste du Norman Nielsen Group, **il faut s'assurer que le joueur puisse retourner voir le tutoriel ou puisse accéder aux astuces facilement dans le jeu.**

Il ne faudrait pas également que le *flow* du jeu soit brisé par des fenêtres surgissant ou un tutoriel qui explique une nouvelle fonctionnalité du jeu. Cependant, une nouvelle information devrait être révélée dans un contexte où elle serait utile pour l'utilisateur ET quand le joueur est prêt à recevoir l'information en question.

Selon [Page Laubheimer](#), **les didacticiels intrusifs et les listes de modifications/mise à jour qui s'affichent lorsqu'une application est lancée** ou à des moments aléatoires pendant la session du joueur sont appelés des révélations poussées.

Pour Page, les révélations poussées ont 3 problèmes principaux :

1. **Les utilisateurs veulent utiliser le produit/jeu le plus rapidement possible, ils ne veulent pas perdre du temps à étudier comment utiliser le jeu.** Ceci est connu comme les paradoxes de l'utilisateur actif vu que le tutoriel aiderait le joueur sur le long terme. Cependant, le joueur va ignorer le tutoriel vu que celui-ci n'est pas voulu.
2. **La guidance apportée par les tutoriels est difficile à être rappelée par le joueur lorsque celui-ci en a besoin.** Les tutoriels en fenêtres surgissant montrent l'information hors contexte et le joueur devra s'en rappeler lorsque l'information serait applicable. Cependant, avec la mémoire limitée de l'être humain, il est difficile de se rappeler de toutes les informations qui apprennent dans un tutoriel.
3. **Ignorer le tutoriel nécessite un effort de l'utilisateur.** Par exemple, les révélations poussées (même s'il ne s'agit pas de didacticiels complets) sont souvent conçues pour être visuellement saillantes et perceptibles, ce qui les rend malheureusement également plus susceptibles d'être distrayantes ou ennuyeuses.

Selon **Alita Joyce** du Norman Nielsen Group, il y a 3 composants régulièrement utilisés pour l'*onboarding* de nouveau utilisateur :

1. **La promotion de fonctionnalité :** Ce *component* sert à éduquer l'utilisateur sur ce que l'application/jeu/fonctionnalité puisse faire. Cependant, cela peut ressembler à de la commercialisation, alors, les joueurs qui jouent pour la première fois risquent d'ignorer ce *component*. C'est uniquement lorsque la fonctionnalité est nouvelle ou avancée qu'elle devrait être promue.
2. **La personnalisation :** Ce *component* nécessitant de l'information de la part du joueur permet d'utiliser de l'information personnalisée pour promouvoir du contenu à l'utilisateur. Si un *component* est utilisé cependant, il faut expliquer au joueur comment l'information sera recueillie et pourquoi l'information sera utilisée.
3. **Les instructions :** Ce *component*, quoique pas souvent nécessaire, contiennent l'aide contextuelle comme les fenêtres surgissant ou les tutoriels. Il faut que les instructions soient brèves et pas gênantes.

Il est cependant important que l'*onboarding* soit le plus simple possible.

Les outils d'analyse et de prise de donnée

Protéger face aux fuites de jeu (*Leaks*)

La première étape pour la création de la session de test de jeu est de s'assurer qu'il n'y a pas de fuite (*leaks*) à propos du jeu hors de test de jeu.

Pour les fuites de jeu lors de test de jeu, il y a plusieurs façons de protéger les secrets des jeux lors de possible test de jeu.

Signature de NDA

Le plus basique est possiblement les accords de non-divulgence ou *non disclosure agreement* (NDA). **Un accord de non-divulgence est un accord légal pour garder quelque chose de secret.** Il est typique de le signer lorsque l'on travaille sur des projets de jeux et tout aussi typique de demander aux testeurs de les signer au début des tests de jeu.

Il existe de nombreux modèles de document d'accord de non-divulgence disponible en ligne, cependant, beaucoup de compagnies auront accès à un avocat qui sera capable d'écrire le document pour l'équipe de développement.

Cependant, **il est difficile d'enforcer un accord**, si un joueur décide de mettre des images du jeu en ligne, non seulement est-ce qu'entrer en contact avec un avocat pour agir sur la situation prend du temps et de l'argent, cela met de l'emphase sur la fuite et, dans le cas l'Internet, la poursuite sera faite trop tard vu qu'il est impossible de cacher quelque chose une fois que celle-ci est sur le web.

C'est pour cela que le point d'un accord de non-divulgence est de créer une peur face à la possible action de poursuite plutôt que d'enforcer pour de vrai l'accord.

La meilleure chose à faire dans le cas d'un accord de non-divulgence est de donner une copie du document signé au début de la session au joueur lorsque celui-ci a fini les tests. Également, rappeler au joueur l'existence du document à la fin de la session peut faire sembler le processus officiel et formel, ainsi, le joueur a plus de chance de le prendre sérieusement.

Utiliser des méthodes de recherches et test de jeu en personnes

Un test de jeu optimal se fait en personne dans un laboratoire, cela permet de garder un œil sur les testeurs et de les prévenir, de prendre des photos ou une capture d'écran hors de la session. Il est aussi possible de demander aux testeurs de mettre leur téléphone dans un casier pour s'assurer qu'il n'est pas accès à un appareil photo.

Si le test doit être fait à distance

Quoiqu'un test en personne est la meilleure condition possible, il se peut qu'il soit nécessaire de faire des tests à distance. Dans ce cas, il y a quand même des étapes possibles qui peuvent prévenir des fuites.

Tout d'abord, il est possible de regarder le joueur tester le jeu et de demander au joueur de parler en temps réel pour ainsi lui offrir moins de possibilités de capturer de l'information qui ne doit pas être montrée au public.

La vidéo de *Bob Tilford* de *RockStar* ci-dessous explique comment avoir un rapport avec le testeur peut aider à réduire le risque de fuite :

<https://www.youtube.com/watch?v=J8zT5MNIL0I>

Pour certains jeux, **il peut être judicieux de diffuser (*stream*) le jeu au joueur, plutôt que de lui envoyer directement une version.** Bien que cela ne soit pas possible pour les jeux qui nécessitent une réactivité extrême, pour les jeux plus lents, cela signifie qu'il est possible de ne pas laisser le joueur avoir un accès non surveillé au jeu et qu'il est possible de contrôler quand et où il peut accéder au jeu.

L'un des outils utilisés pour diffuser le jeu est *Parsec*, *Parsec* est un outil qui autorise les joueurs distants à diffuser l'accès au jeu sur un appareil à distance.

<https://parsec.app/solutions/user-research-test-de-jeu-jeu-ing-quality-assurance>

Mettre des filigranes (*watermark*) dans le prototype

Il est possible, si cela ne dérange pas le but des tests de jeu, de mettre un filigrane ou des informations dans le prototype du jeu.



Il est possible de rendre les filigranes invisibles au joueur, mais visible à l'équipe. Le but de cela est de faire savoir au joueur que l'information qui fait partie d'une fuite peut être retracé jusqu'à lui.

Certaines informations intéressantes à mettre peuvent être le temps courant, la date et des informations identifiables qui peuvent reliés la capture à une devise spécifique. Démontrer qu'une capture peut être retracé à un individu spécifique peut baisser le risque de fuite.

Il est important de se rappeler que cette fonctionnalité peut être difficile et peut prendre beaucoup de temps à implémenter dépendamment de l'équipe.

Retirer les assets importants des tests de jeu

Lorsque l'équipe travaille avec une propriété intellectuelle établie, il peut être utile de masquer la propriété dans la version de test. Cela pourrait ressembler à changer l'écran titre ou à renommer le jeu dans la version afin qu'il ne mentionne pas ou n'affiche pas d'images de la marque. Ceci peut-être plus pertinent pour les jeux mobiles simples, car il est peu probable que le remplacement de modèles de personnages entiers soit possible dans des jeux plus complexes.

La suppression d'assets peut être réalisée soit en créant une version personnalisée, soit en évitant d'inclure les assets finaux dans n'importe quelle version jusqu'à l'annonce officiel. Le remplacement des assets nécessite également du temps de développement, mais peut protéger un jeu contre les fuites.

Filtrer les participants des tests de jeu

Quand l'équipe recherche des testeurs pour le jeu, il est important de regarder et vérifier durant l'interview :

- *Que le testeur n'est pas un journaliste*
- *Que le testeur n'est pas un créateur de contenu/streamer*
- *Qu'ils n'ont pas de connexion avec les autres participants*
 - *Si des testeurs se connaissent, cela peut augmenter leur courage de collaborer et de causer une fuite sur le jeu.*

Il y a très peu de participant malicieux, mais il est tout de même important de poser les questions et de retirer les testeurs qui ne peuvent pas donner de réponses convaincantes.

Les outils et méthode pour prendre des notes

Une fois que les testeurs ont été filtrer, il faut installer les outils nécessaires pour la prise de note et d'informations. **Normalement, un outil qui ne requiert pas la présence de l'interviewer est préférable.**

Rapport de Bogue et Engagement du joueur

La meilleure façon de calculer l'engagement d'un joueur est de faire un questionnaire. Il faut poser des questions sur l'expérience de jeu au joueur pour ainsi lui faire penser fort sur l'expérience qu'il vient de passer.

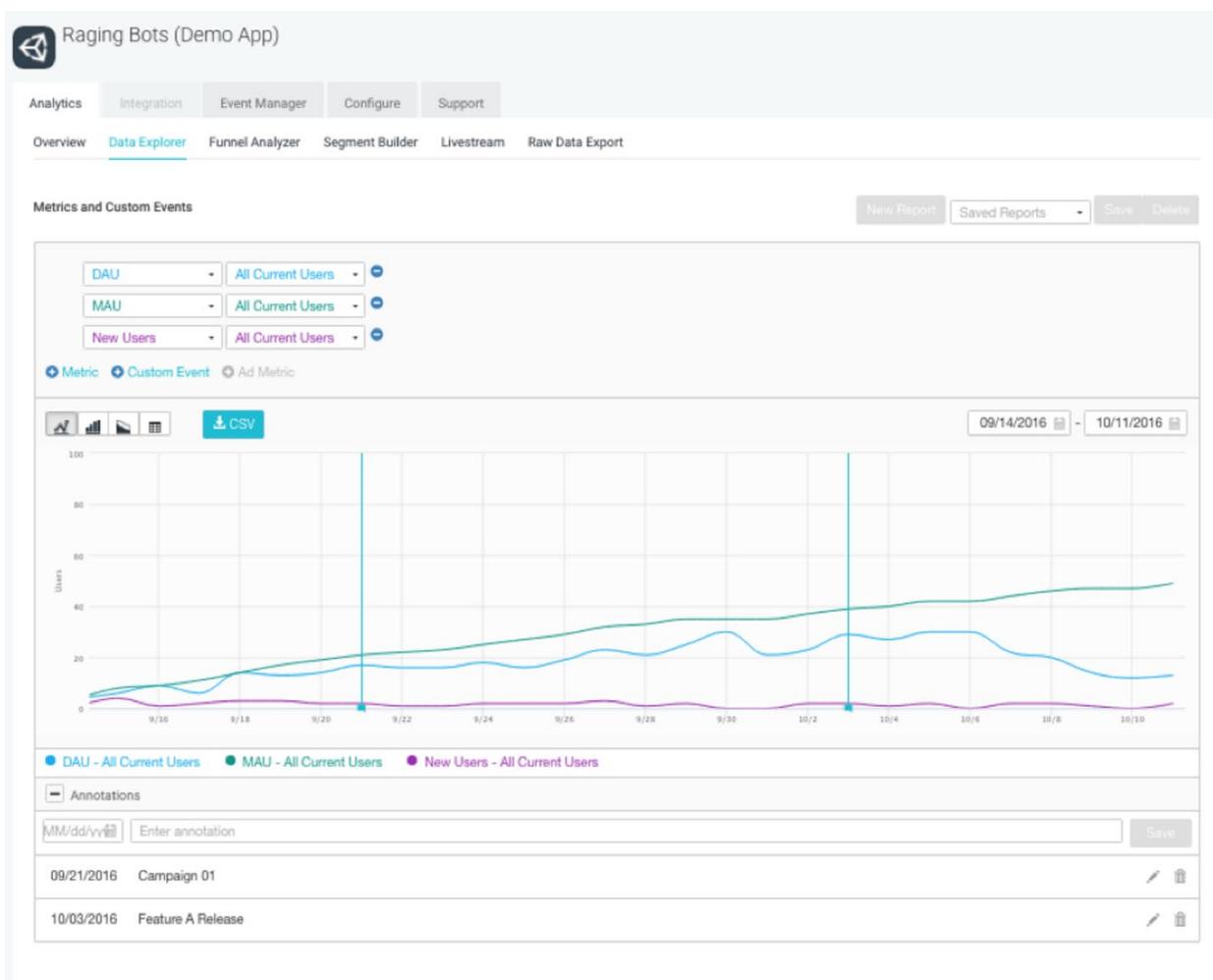
Selon *Cypress Reeves* de GameAcademy.org, le logiciel recommandé est Google Forms, il y a cependant d'autres équivalents si préféré :

- [Microsoft Forms](#)
- [Wufoo](#)
- [JotForm](#)
- [Formstack](#)
- [Typeform](#)
- [Paperform](#)
- [Formsite](#)
- [123FormPrototypeer](#)
- [Cognito Forms](#)
- [Zoho Forms](#)

Pour la prise de données et statistiques

Le moyen le plus simple de collecter des données importantes pour le jeu consiste à utiliser un logiciel d'analyse. Cela vous permet d'équilibrer et de tester des mécanismes individuels, de cartographier les tendances du comportement des joueurs et de s'assurer que les joueurs voient tout le contenu que l'équipe veut qu'ils voient.

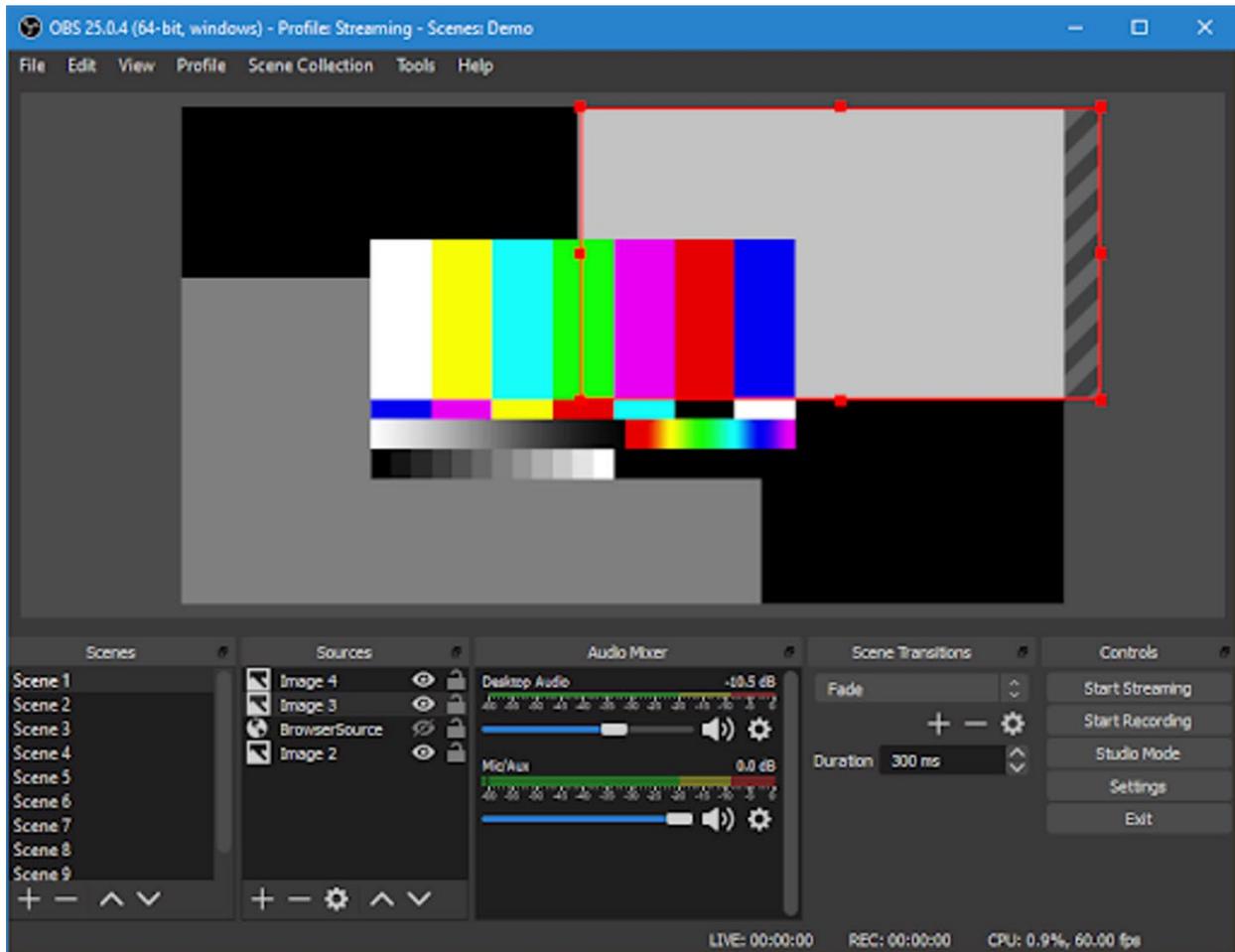
La majorité des moteurs de jeu utilise des logiciels d'analyse intégré.



Jouabilité, capture d'écran et vidéos

Enregistrer des séquences de jeu des testeurs peut être incroyablement utile. Les membres de l'équipe passent tellement de temps avec le jeu que les membres y jouent d'une manière qu'aucune personne normale ne le ferait, donc être capable de voir ce que les joueurs font réellement lorsqu'ils ont le contrôle et qu'ils sont laissés à eux-mêmes peut ouvrir les yeux des membres sur des choses que l'équipe n'aurait jamais repérées. L'équipe peut même potentiellement utiliser les images qu'elle collecte pour le marketing.

L'outil recommandé par Cypress Reeves est OBS.



Dans le cas où le test de jeu se fait à distance et le test nécessite que le joueur télécharge une application, il est nécessaire de mettre les instructions sur comment installer le logiciel dans le questionnaire ou rentre les instructions disponibles d'une autre façon.

If you recorded gameplay, please provide the video link here

Short answer text

Are you okay with your recording being used for marketing purposes?

Yes

No

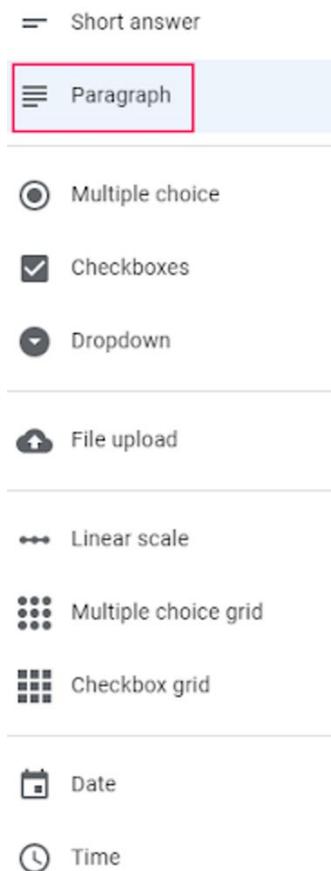
If you wish to be featured in the credits, please provide the name you would like to be listed as.

Short answer text

Trucs et astuces sur les outils

Google Forms

Google Forms vous propose une gamme de types de questions différents, allant des choix multiples aux échelles d'un à cinq, ce qui rend leur service personnalisable pour une variété de cas d'utilisation différents. **Le seul qui est intéressant cependant, selon Cypress Reeves, est le type de réponse paragraphe.**



Ceci n'est pas une règle enforcé, dans le cas d'avoir des centaines de testeurs, il est possible de demander aux testeurs de noter leur engagement sur une balance de 1 à 5 et quand même avoir des réponses signifiantes.

Cependant, **dans le cas d'avoir 5 ou 10 testeurs, il est important de poser des questions qui interrogent en particulier l'expérience vécue par les joueurs.** L'équipe veut mettre le joueur à qui la question est posée dans la position d'être incapable de donner une réponse paresseuse.

Le truc magique de Cypress Reeves est de prendre la question que l'équipe pense vouloir poser sur le jeu et la reformuler comme une demande d'un élément sur l'expérience du joueur.

Exemple :

« Quel était le moment préféré de votre jeu ? »

Devient :

« Si possible, décrire un moment ou vous sentiez inspiré ou énérgisé de participer à l'expérience de jeu. »

Cela invite le joueur à interroger sa propre expérience et à donner une réponse plus détaillée.

Cela les incite également à considérer les choses qu'ils ont faites, plutôt que les choses qui viennent de se passer dans le jeu.

En raison de la spécificité et des détails que ces questions demandent, il est préférable, selon Cypress Reeves, **de généralement viser entre 3 à 5 questions de ce type afin de ne pas épuiser les testeurs.**

Voici un exemple produit Cypress Reeves : <https://forms.gle/Cr8TcTDWdrErQ7NZA>

Les types de question à poser et les informations à conserver

1. *Jesse Schell sur les tests de jeu : Pourquoi*

Selon **Jesse Schell** dans la section sur le test de jeu dans son livre « The Art of Game Design », **il explique que faire un test de jeu sans avoir de but spécifique en tête est une perte de temps.** Cependant, avoir une question générique comme « Est-ce que mon jeu est amusant » n'est pas assez pour avoir des informations importantes. **Pour des questions de test de jeu, il faut être le plus précis possible.**

Voici quelque exemple de question possible à se poser, certaines sont assez génériques, d'autres sont très précises :

- *Est-ce que les hommes et les femmes jouent à mon jeu différemment ?*
- *Est-ce que les enfants préfèrent mon jeu face aux adultes ?*
- *Est-ce que les joueurs comprennent comment jouer ?*
- *Est-ce que les joueurs veulent jouer une deuxième fois ? Une troisième fois ? Une vingtième fois ?*
- *Est-ce que les joueurs pensent que le jeu est équitable ?*
- *Est-ce que les joueurs sont ennuyés/frustrés/confus à n'importe quel moment dans le jeu ?*
- *Il y a-t-il des stratégies dominantes ou échappatoires (loophole) ?*
- *Il y a-t-il des bogues cachés dans le jeu ?*
- *Quelles stratégies est-ce que le joueur trouve par eux-mêmes ?*
- *Quelles parties du jeu est le plus amusantes ?*

- *Quelles parties du jeu est le moins amusantes ?*
- *Est-ce que le bouton A ou B devrait être utilisé pour la mécanique de saut ?*
- *Est-ce que le niveau 3 est trop long ?*
- *Est-ce qu'une énigme est trop difficile ?*

1. *Jesse Schell sur les tests de jeu : Qui*

Une fois que l'équipe à trouver la ou les raisons pour avoir un test de jeu, il faut que l'équipe pense qui sera testé. Et les testeurs seront déterminés par ce que l'équipe voudrait apprendre. D'habitude, l'équipe voudra choisir des testeurs qui font partie du client cible, mais même dans cette situation, il y a des choix possibles. Voici des choix communs :

1. **Les développeurs** : Les premières personnes qui ont une chance de testés sont les développeurs du jeu.
 1. Pros : Les développeurs sont toujours disponibles et peuvent donner des commentaires significatifs et réfléchis. Ils n'ont également pas besoin de signer des accords de non-divulgaration vue qu'ils connaissent déjà toute les informations et aspect du jeu.
 2. Cons : Les développeurs sont les personnes les plus proches du jeu et cela va déformer leurs opinions sur le jeu. Cependant, ne pas tester avec des développeurs peut faire manquer la chance de recevoir des commentaires signifiants. Dans le cas de test de jeu avec des développeurs, il est important de tester avec des développeurs, mais il faut prendre les commentaires avec des doutes.

2. **Les amis et familles des développeurs** : Les prochaines personnes à tester le jeu sera probablement les familles et amis des développeurs.
 1. Pros : Les familles et amis sont grandement disponibles et confortables à vous parler. S'ils pensent à une bonne idée après les tests, il y a de grande chance que vous l'entendez quand même.
 2. Cons : Les amis et familles ne voudront pas blesser vos émotions, surtout qu'ils vont vous voir sur une base régulière. Alors, ils vont déformer la vérité, également,

vu qu'ils vous apprécient, ils seront prédisposés à aimer le jeu et vont essayer de l'aimer, ce qui ne se passera pas dans le vrai monde.

3. **Les joueurs experts** : Chaque genre de jeu a les « experts » — les joueurs *hardcore* qui ont joués a beaucoup de variation du type de jeu que l'équipe est en train de faire. Ces personnes aiment faire des tests de jeu de jeu en progrès parce que ça donne un coup de pouce à leur titre « d'expert ».
 1. Pros : Après avoir joué à beaucoup de jeu de même type et genre que le jeu que l'équipe produit, les joueurs experts peuvent communiquer des informations détaillées en utilisant de la terminologie technique et des exemples spécifiques et ainsi comparer comment le jeu de l'équipe se compare aux autres jeux qui lui ressemblent.
 2. Cons : Seulement un petit pourcentage de la population fait partie de joueurs experts. Également, ces joueurs demandent des défis et mécaniques plus difficile et complexes que ce que le joueur ordinaire demanderait.

4. **Les « testeurs tissus »** : Ce sont des testeurs qui n'ont jamais entendus parlés du jeu de l'équipe. Le terme « testeurs tissus » fait référence aux tissus *Kleenex* car, comme un mouchoir, ils ne peuvent utilisés qu'une seule fois.
 1. Pros : Les testeurs de cette catégorie vont voir le jeu avec des yeux frais et peuvent remarquer des choses ou émettre des commentaires sur des éléments dont l'équipe c'était habitué. Si l'équipe veut répondre à des questions d'usabilité, de communications ou des questions d'attrayantes initiales (*initial appeal*), ces testeurs peuvent être très utile.
 2. Cons : Vu que des jeux vidéo sont souvent joués plusieurs fois, sur plusieurs sessions. Si l'équipe utilise uniquement des « testeurs tissus » pour tester le jeu, cela pose un risque de créer un jeu qui démontre une forte attraction lors de la première

session de jeu, mais qui peut devenir ennuyant lors de plusieurs sessions de jouabilité.

Encore, les testeurs choisis vont dépendre des questions et des informations que l'équipe recherche. Il faut faire correspondre les testeurs aux questions que l'équipe veut faire répondre, c'est uniquement de cette façon qu'il sera possible d'avoir des informations pertinentes et utiles. La majorité des jeux vont utiliser tous les types de testeurs mentionnés plus haut, ce qui est important est d'avoir les bons testeurs aux bons moments pour répondre à le plus de questions possibles soigneusement.

Lors des retours de test de jeu, il est important de diriger la discussion et de prendre l'initiative, au départ, c'est une bonne idée de demander quelles étaient les premières impressions du chaque testeur. Ensuite, posé des questions précises auxquelles l'équipe veut des réponses. Finalement, demander de dernières réflexions sur l'expérience de jeu.

Voici 20 questions ouvertes qui sont écrites pour permettre aux testeurs de donner de bonnes réponses spécifiques. Les questions ne sont pas tous appropriés pour tous les jeux ou groupes de testeurs. Il est alors recommandé de prendre 1 à 2 questions par catégorie ou mettre de l'emphase sur des éléments de jeu qui nécessite le plus de recherche. **Ici, l'important est de trouver des problèmes et non des solutions :**

- **Règles/explication**
 - *Qu'est-ce qui aurait pu être expliqué mieux ou plus tôt lorsque j'enseignais le jeu ?*
 - *Qu'auriez-vous aimé savoir lorsque vous avez commencé à jouer ?*
 - *Y a-t-il quelque chose que vous avez l'impression de ne pas comprendre, même après avoir terminé le jeu ?*
 - *Qu'est-ce qui manquait dans les tableaux de joueurs/cartes de référence ?*
- **Jeu/agence**
 - *Que vouliez-vous faire, mais que vous n'avez pas pu ou n'avez pas pu faire ?*
 - *Vos décisions étaient-elles significatives ou avaient-elles l'impression qu'elles n'avaient pas d'importance ?*

- *Qu'est-ce qu'on vous a demandé de mémoriser/intérioriser/rappeler ? Était-ce trop pour un jeu comme celui-ci ?*
- *Stratégie/tactique*
 - *Comment décririez-vous votre stratégie ?*
 - *Quelles autres stratégies avez-vous vues ou auxquelles vous avez pensé ?*
 - *Avez-vous remarqué un avantage de premier ou de dernier joueur ?*
 - *Avez-vous remarqué une stratégie dominante ?*
- *Émotions/sentiments*
 - *Quel genre d'émotions ou de sentiments avez-vous ressenti/remarqué pendant que vous jouiez ?*
 - *Qu'est-ce qui a provoqué ces émotions, le jeu ou un autre joueur ?*
 - *Quels éléments vous ont aidé à vous sentir immergé dans le thème du jeu ?*
 - *Quels éléments vous ont fait sortir du thème du jeu ?*
 - *Quels moments ont été les plus amusants ?*
 - *Quels moments ressemblaient à du travail ou étaient ennuyeux ?*
 - *Qu'est-ce qui vous a semblé équilibré ou déséquilibré ? (Se rappeler que nous nous concentrons sur les sentiments ici, pas sur les calculs bruts des choses.)*
- *Globale*
 - *Avez-vous remarqué des problèmes d'accessibilité ? (Exemples : daltonisme, connaissances culturelles requises, charge cognitive élevée, beaucoup d'éléments de mémoire, etc.)*
 - *Combien de temps le jeu a-t-il semblé (trop court, trop long, juste ce qu'il fallait) ?*
 - *La mécanique a-t-elle bien fonctionné avec le thème ?*

Les étapes d'un test de jeu

1. **Identifier les temps pour les tests de jeu, à quel intervalle l'équipe veut elle tester le jeu avec les différents types de testeur ?**
 - a. Test par les développeurs
 - b. Test par les amis et familles
 - c. Test par des joueurs expérimentés
 - d. Test par des testeurs tissus
2. **Identifier les éléments à tester, quelles sont les différentes fonctionnalités que l'équipe veut tester lors des tests à venir ? Le faire avec un *Onboarding Sheet*.**
 - a. Définir les éléments de jeu qui seront à tester
 - i. Catégorie primordiale
 - ii. Niveau de priorité
 - iii. Quand
 - iv. L'ordre de tutoriel
 - v. Difficulté
 - vi. Pourquoi
 - vii. Comment
 - viii. Emballage narratif
 - ix. Feedback UX
 - b. Est-ce que l'équipe veut tester des mécaniques
 - c. Est-ce que l'équipe veut tester des fonctionnalités
 - d. Est-ce que l'équipe veut tester l'expérience utilisateur
 - e. Est-ce que l'équipe veut tester l'affordance visuel et sonore
3. **Identifier le style de test. Est-ce que ce sera un test à distance ou en présentiel ?**
 - a. Test à distance ?
 - i. Comment est-ce que le joueur doit-il installer les logiciels ?
 - b. Test en présentiel ?
 - i. Test avec supervision d'un membre de l'équipe ?
 - ii. Test entre plusieurs testeurs ?
 - c. Test sur PC ou sur console ?

- i. Si sur PC, sa sera avec Clavier-souris ou manette ?
- ii. Sur console, sur quelle console le test se passera-t-il ?

4. Créer les questions qui seront posés aux testeurs suivants les tests.

- a. Combien de question l'équipe veut elle posé ?
- b. Combien y aura-t-il de testeur ?
 - i. L'Équipe veut elle poser des questions ouvertes ou des questions simples avec une réponse comme oui/non ?
- c. Quelle information est-ce que l'équipe priorise ?
- d. Quel problème est-ce que l'équipe aimerait découvrir ?

5. Identifier les outils qui seront utilisés pour le test de jeu.

- a. Est-ce que le test est en présentiel ou en distance ?
- b. Comment le testeur sera-t-il enregistrer ?
- c. Où est-ce que les informations present seront-elle écrite ?
- d. Décider si une récompense sera offerte au testeur (que ce soit monétaire ou autre)
- e. Comment les informations seront t'elles filtrer ?

6. Créer un prototype jouable du jeu

- a. Mettre des filigranes dans le prototype du test
 - i. Il est possible des cachés au testeur
- b. Changer les éléments visuels si la propriété est connue et que le jeu n'a pas encore été annoncé

7. Créer un accord de non-divulgation.

- a. Contacter un avocat pour la rédaction si la compagnie y a accès.
- b. S'assurer que le testeur signe le document.
- c. Lui expliquer que la signature est importante et lui faire comprendre que des actions seront prises dans le cas de fuite de document.
- d. Donner une copie du document au signataire lors de la fin des tests.

8. Filtrer les testeurs

- a. S'assurer de ne pas engager des journalistes
- b. Regarder avec les possibles streamer/créateur de contenu.
- c. Ne pas engager de groupe d'ami.

- d. S'informer sur les possibles handicaps des testeurs, est-ce que le testeur est daltonien, etc. ?
- e. Confisquer le téléphone du testeur et tout autre appareil qui pourrait servir pour prendre des captures d'écran.

9. Tester le jeu

- a. S'assurer de faire parler les testeurs lors du test
- b. Enregistrer ou regarder les expressions du joueur en temps réel
 - i. Enregistrer le testeur en tout temps
- c. S'assurer que le joueur ne prend pas de capture d'écran.
- d. Écrire des informations en temps réel

10. Recueillir l'informations

- a. Demander au testeur de répondre à un questionnaire
- b. Poser les questions au testeur en personne
 - i. Questions à développement sont priorisés

11. Finaliser le test

- a. Donner une copie de l'accord de non-divulgence au testeur
 - i. Lui rappeler de l'importance du document et du poids de celui-ci.
- b. Donner le téléphone au testeur s'il a prêté son téléphone à l'équipe

12. Retour sur les tests

- a. Quelle information ont été recueillie par l'équipe ?
 - i. Mettre l'emphase sur la découverte de problème et non de solution.
- b. Faire un *brainstorm* sur les possibles solutions aux problèmes en équipe.
- c. Créer un prototype jouable avec les changements.
- d. Retourner à l'étape 1