Projet de fin d'études

Réalité Augmentée au service d'une alimentation saine

Auteurs :

Thomas PIET et Thomas HEURTEL

Encadrant :

Arnaud PROUZEAU

En collaboration avec :

Antonio VERDEJO-GARCIA et Simon VAN BAAL
Monash University, Melbourne

Remerciements

Au terme de ce projet, nous tenons à remercier Arnaud Prouzeau, le professeur encandrant, pour son accompagnement, sa disponibilité et ses conseils.

Nous remercions également les clients , Antonio VERDEJO-GARCIA et Simon VAN BAAL, pour les échanges, idées et conseils lors de la réunion.

Enfin, nous souhaitons remercier Pascal Desbarats, responsable de l'UE MOCPI, pour l'organisation de cet enseignement et ses conseils pour la gestion et le bon déroulement de ce projet.

Table des matières

1 Introduction

Dans le cadre du parcours Master 2 d'Informatique pour l'Image et le Son, nous avons dû travailler pendant 2 mois sur notre Projet de Fin d'Études. Nous avons décidé de nous investir dans le projet intitulé "La réalité augmentée au service d'une alimentation saine". Le projet est proposé par Arnaud Prouzeau, en collaboration avec Antonio Verdejo-Garcia et Simon Van Baal de l'Université Monash à Melbourne.

Le projet traite de biais cognitifs étant à l'origine de choix alimentaires malsains. Précisement du biais d'approche, qui se traduit par une tendance à se diriger vers de la nourriture malsaine plutôt que l'éviter. Il existe donc un entraînement pour éviter ce biais d'approche, appelé entraînement d'approche-évitement, ou en anglais Approach-Avoidance Training (AAT). Il consiste à entraîner l'utilisateur à effectuer un mouvement d'évitement, dans notre cas face à des aliments malsains, et un mouvement d'approche face à des aliments sains. Récemment, des psychologues ont étudié la pratique de cet entraînement via une application pour smartphone afin de faciliter sa pratique à son domicile ou en déplacement, mais aussi dans l'objectif de le faire sur une plateforme plus réaliste et attrayante pour amener plus d'utilisateurs à s'en servir.

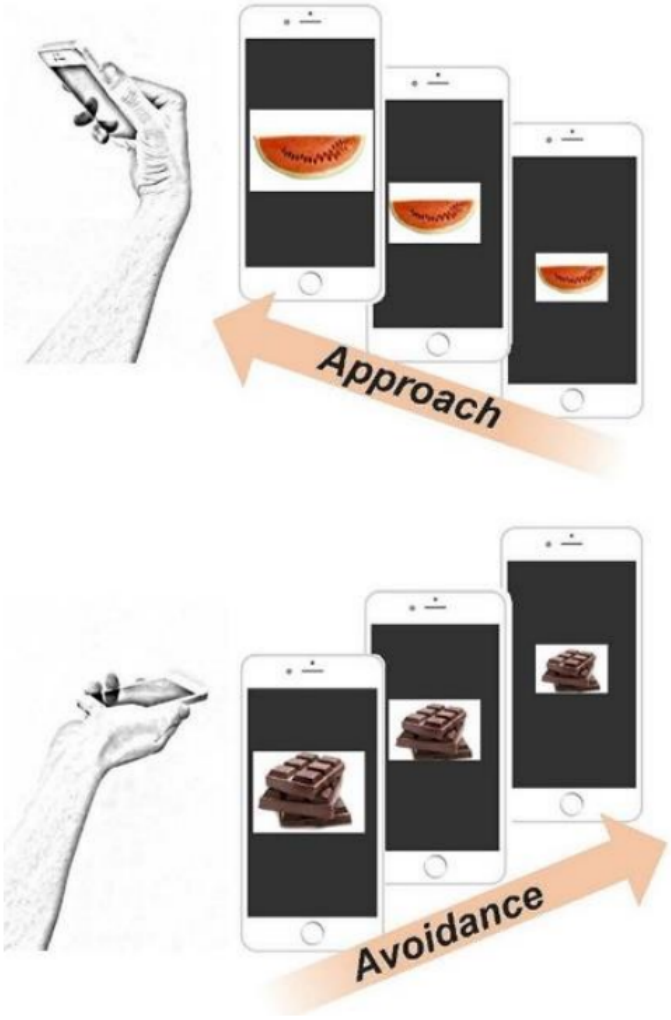


FIGURE 1 – AAT via une application mobile. L'utilsateur penche le télphone vers lui face à un aliment sain pour l'approcher. L'utilisateur penche le téléphone loin de lui pour éloigner l'aliment malsain.

Le but du projet (voir annexe, fig 8) est de réaliser un prototype d'application en réalité augmentée pour smartphone, permettant la mise en oeuvre de l'AAT à domicile sur une plateforme familière et ainsi tenter de rendre cet entraînement cognitif plus réaliste et attrayant pour permettre une utilisation par le grand public en recherche de solutions pour une alimentation saine.

L'application doit permettre aux utilisateurs de visualiser des aliments virtuels sur une surface plane (table de diner, plan de travail dans une cuisine ...) et leur permettre de les jeter ou non en effectuant l'AAT à l'aide de gestes physiques (mouvement du téléphone ou de la main).

On cherche donc à réaliser une application mobile en réalité augmentée implémentant l'AAT pour en faire un outil permettant aux psychologues d'évaluer l'impact de celle-ci dans l'immersion de l'utilisateur et l'efficacité de l'entraînement, tout en rendant l'application plus attrayante.

2 Etat de l'art

Dans un premier temps, nous avons commencé par nous renseigner sur l'utilisation de l'Approach-Avoidance Training pour aider la gestion de l'alimentation. D'après l'article "Smartphone-based cognitive bias modification training improves healthy food choice in obesity : A pilot study"[1] deux stratégies d'entraînement cognitif, l'AAT[2] et l'EFT[3] (pour Episodic Future Thinking) étaient particulièrement efficaces pour modifier les biais cognitifs qui pourraient orienter nos choix vers de la nourriture malsaine et la préférence d'une récompense immédiate. De plus, d'après les recherches de O'Neill et al [3], les individus qui pratiquent l'entraînement EFT à l'aide d'un smartphone dans des lieux de consommations naturels choisissaient des plats plus sains.

Nous nous sommes donc renseignés pour savoir si des applications de l'AAT avec un smartphone chez soi ou dans des lieux de consommation étaient plus efficaces qu'en laboratoire. Encore une fois l'article [1] conclut que la portabilité de l'application implique un taux plus élevé de conformité à l'expérience et d'auto-évaluation. L'AAT sous forme d'application semble donc apporter de bons résultats, car le support utilisé rend l'entraînement plus agréable à réaliser, car faisable quand on le souhaite et quand on en a besoin, par exemple à la maison ou dans des lieux de consommation.

On cherche donc à savoir si une application en réalité augmentée pourrait renforcer cet aspect attrayant et si l'immersion que peut offrir la RA inciterait une utilisation plus courante et par un plus grand nombre. Aujourd'hui, la réalité augmentée est très présente dans le domaine de l'application mobile car attrayante de par son côté « nouvelle technologie » et sa promesse d'immersion accrue. De nombreuses applications utilisant la réalité augmentée ont connu un grand succès ces dernières années comme Pokémon Go, Google Arts Culture, ...

Dans notre cas, nous avons cherché des applications de l'AAT basées sur la réalité augmentée, mais nous n'en avons pas trouvé. Cependant, nous sommes tombés sur des articles sur l'utilisation de la réalité virtuelle associée à l'AAT [4] et l'AATP (Approach-Avoidance Training Program) [5] pour des troubles de la consommation d'alcool. Dans ces deux articles, ce qui revient en conclusion est que l'application virtuelle se rapprochant plus d'une situation réelle qu'un test AAT classique analogique, elle provoque un sentiment d'incarnation chez l'individu. De plus les gestes qu'il exécute lui paraissent plus réel et donc le sentiment de vraiment écarter ou rapprocher l'objet aussi. Les résultats semblent donc meilleurs avec une solution en VR qu'une solution classique.

Est-ce que ce sera aussi le cas pour une application en réalité augmentée ? D'après les psychologues qui nous ont proposé ce sujet, avec la réalité augmentée, nous avons aussi réellement une augmentation de la sensation de « réalité » grâce à la visualisation de l'objet virtuel superposé dans le monde réel. De plus, contrairement à la réalité virtuelle, l'application en réalité augmentée est plus facilement transportable et donc utilisable partout, à tout moment et par tout le monde, ce qui est un avantage considérable.

3 User stories

- En tant qu'utilisateur, je souhaite pouvoir placer l'objet sur un plan.
- En tant qu'utilisateur, je souhaite pouvoir modifier la position de l'aliment afin de le placer comme je le veux.
- En tant qu'utilisateur, je souhaite pouvoir "swipe" un aliment pour le "manger" ou le "jeter" puis passer à l'aliment suivant (nous avons ajouté ceci pour faire une première version simple à implémenter pour présenter un prototype aux psychologues tôt).
- En tant qu'utilisateur, je souhaite pouvoir rapprocher ou éloigner le téléphone d'un aliment pour le "manger" ou le "jeter" puis passer à l'aliment suivant.
- En tant qu'utilisateur, je souhaite pouvoir choisir entre commencer l'entraînement, accéder au réglages ou quitter l'application.
- En tant qu'utilisateur je souhaite pouvoir accéder à mes statistiques depuis le menu principal ou à la fin d'une session afin de voir mes résultats.
- En tant qu'utilisateur, je souhaite pouvoir revenir au menu principal afin de quitter l'application ou recommencer.

4 Les besoins

4.1 Besoins fonctionnels

- Détecter des plans horizontaux et les afficher.
- Bouton pour afficher ou cacher les zones détectées.
- Pouvoir placer les aliments sur un des plans détectés et modifier sa position en touchant l'écran.
- Pour la version swipe : afficher un bouton qui peut être glissé à droite pour "manger" l'aliment et à gauche pour le "jeter".
- Pour la version AAT : en appuyant sur un bouton, on peut déplacer l'aliment selon l'axe de la caméra en bougeant le téléphone. Éloigner le téléphone de l'utilisateur permet de "jeter" l'aliment et le rapprocher de l'utilisateur permet de "manger" l'objet.
- Récupérer depuis un fichier CSV pour chaque aliment un id et un statut - sain ou malsain.
- Lorsque l'utilisateur "mange" un aliment, si l'aliment est marqué comme sain, on affiche une image de validation, sinon on affiche une image d'erreur. Si l'aliment est marqué malsain, on fait l'inverse.

- Menu de démarrage avec un bouton pour lancer l'application, un bouton pour accéder aux paramètres et un bouton pour accéder aux skins.

- Lorsque l'application est en cours, il doit y avoir un bouton pour revenir au menu démarrage.

- Affichage d'un système de score, de streak et de golds qui sont sauvegardés même après fermeture de l'application. À chaque fois que l'utilisateur "mange" un aliment sain ou "jette" un aliment malsain son score augmente et ses golds aussi, sinon son score baisse.

- Récompenses journalières : chaque jour après une streak de 10, récompense de 1000 golds.

- Dans le menu paramètres, un bouton pour tester la version Swipe, un bouton pour réinitialiser les statistiques (score, gold, streak, ...) et un bouton pour revenir au menu de démarrage.

- Dans le menu Skins, possibilité d'acheter 3 skins à l'aide des golds gagnés et de passer d'un skin à l'autre à l'aide de boutons, et un bouton pour revenir au menu de démarrage.

4.2 Besoins non fonctionnels

- Proposer des menus, et une navigation qui permette d'accéder et de revenir à n'importe quel menu à tout moment.

- L'application doit être suffisamment performante pour pouvoir effectuer toutes les actions sans faire attendre l'utilisateur (chargement de scène, actions utilisateur)

- Présenter une interface ergonomique et intuitive pour l'utilisateur.

- Le code de l'application doit être lisible et compréhensible afin d'assurer son état évolutif et extensible pour pouvoir introduire d'autres fonctionnalités.

- L'application a été réalisé pour des appareils sous Android 7.0 au minimum disposant d'une caméra.

- L'application a été réalisé sous Windows 10 avec Unity 2020.3.24f1.

5 Choix logiciels

Pour ce projet, il nous était demandé de développer un application Android à l'aide du moteur de jeu Unity en C# et d'utiliser le toolkit "AR Foundation" qui permet de travailler sur de la réalité augmentée sur Unity. Nous avons utilisé le package ARCore XR Plug-in qui permet de supporter AR Foundation pour les appareils Android. Pour la partie collaborative nous avons utilisé la gestion de projet collaboratif inclue dans Unity.

6 Gantt prévisionnel

FIGURE 2 – Gantt prévisionnel

L'objectif était d'avoir une détection de plans simple et pouvoir placer un prefab après 2 semaines. Puis réaliser une première version avec un swipe pour pouvoir la présenter au rendez-vous avec les clients. Après ce rendez-vous, nous devions donc réaliser la version AAT pour début mars, pour avoir le temps d'implémenter des éléments de ludification (score, skins, récompenses, ...), le menuing et les effets sonores et visuels. Puis nous comptions garder les deux dernières semaines pour le rapport et la préparation de la soutenance.

7 Architecture

7.1 Description

Ce projet ayant été réalisé sous Unity, il a une architecture d'application Unity classique. Les paramètres d'utilisateur et de projet ainsi que les logs et les bibliothèques sont placés par défaut dans des dossiers gérés par Unity. Tout ce que nous avons modifié se trouve dans le dossier Assets. Nos scènes sont dans le dossier Scenes, les classes sont dans le dossier Scripts et nos images, prefabs et fichiers CSV se trouvent dans le dossier Resources.

Le script principal est SpawnableManager, qui gère l'écran de jeu principal de l'application. Il est donc au centre du projet et dépend de plusieurs autres scripts, appelés Handlers, qui gèrent chacun une tâche particulière.

ButtonHandler est appelé pour la gestion du bouton d'interaction. Il permet de savoir quand celui-ci est pressé et peut changer dynamiquement son apparence.

CSVHandler permet de lire les fichiers CSV stockés dans le dossier Resources. Il y en a deux : un pour les aliments et un pour les skins. Ils contiennent chacun quatre colonnes, une pour les noms des ressources associées (prefab/sprite), une pour leur index, une qui permet de savoir si l'aliment est sain ou non (inutile pour le fichier skins, cette colonne y est remplie de False) et la dernière pour les prix des skins (inutile pour les aliments, colonne remplie de 0). Le script permet donc de récupérer facilement une donnée stockée dans ces fichiers, auxquels on peut donc rajouter des lignes facilement si on le souhaite.

DataHandler gère toutes les données sauvegardées dans les PlayerPrefs du projet (une fonctionnalité proposée par Unity qui permet de sauvegarder ses préférences facilement). Cela comprend le score de l'utilisateur, son argent actuel, la date de sa dernière récompense journalière, son skin actuellement équipé et tous ses skins débloqués. Il permet de charger, modifier et sauvegarder facilement l'une de ces valeurs.

SkinHandler n'est jamais appelé par un autre script. Il est géré par Unity lors du clic sur l'un des boutons de choix de skin. Il permet de bloquer le changement de skin si celui-ci n'est pas débloqué et de gérer l'achat. Il appelle CSVHandler et DataHandler.

Certains scripts ne sont pas liés aux autres et sont appelés individuellement par Unity. On peut compter MainMenu, qui permet de changer de scène (il est appelé lors de clics sur boutons), PlaneDetectionToggle, qui active ou désactive l'affichage des plans, et LongClickButton, utilisé pour le reset des données, qui force à rester appuyé un certain temps (5 secondes) sur le bouton pour l'activer, et affiche une animation dessus. Nous n'avons pas codé nous-mêmes ce dernier, il a été écrit par Jason Weimann et est disponible en libre accès sur internet[6].

Nous avons également décidé d'ajouter « en bonus » à cette version de l'application le prototype que nous avions réalisé au début du projet pour pouvoir proposer rapidement une première démo à nos clients, et qui fonctionne par swipe au lieu du mouvement du téléphone. Il contient deux scripts différents de la nouvelle version : SwipeEffect, qui gère le déplacement du bouton, et SwipeSpawnableManager, le script principal qui correspond au SpawnableManager actuel et appelle CSVHandler et SwipeEffect.

7.2 Diagramme UML

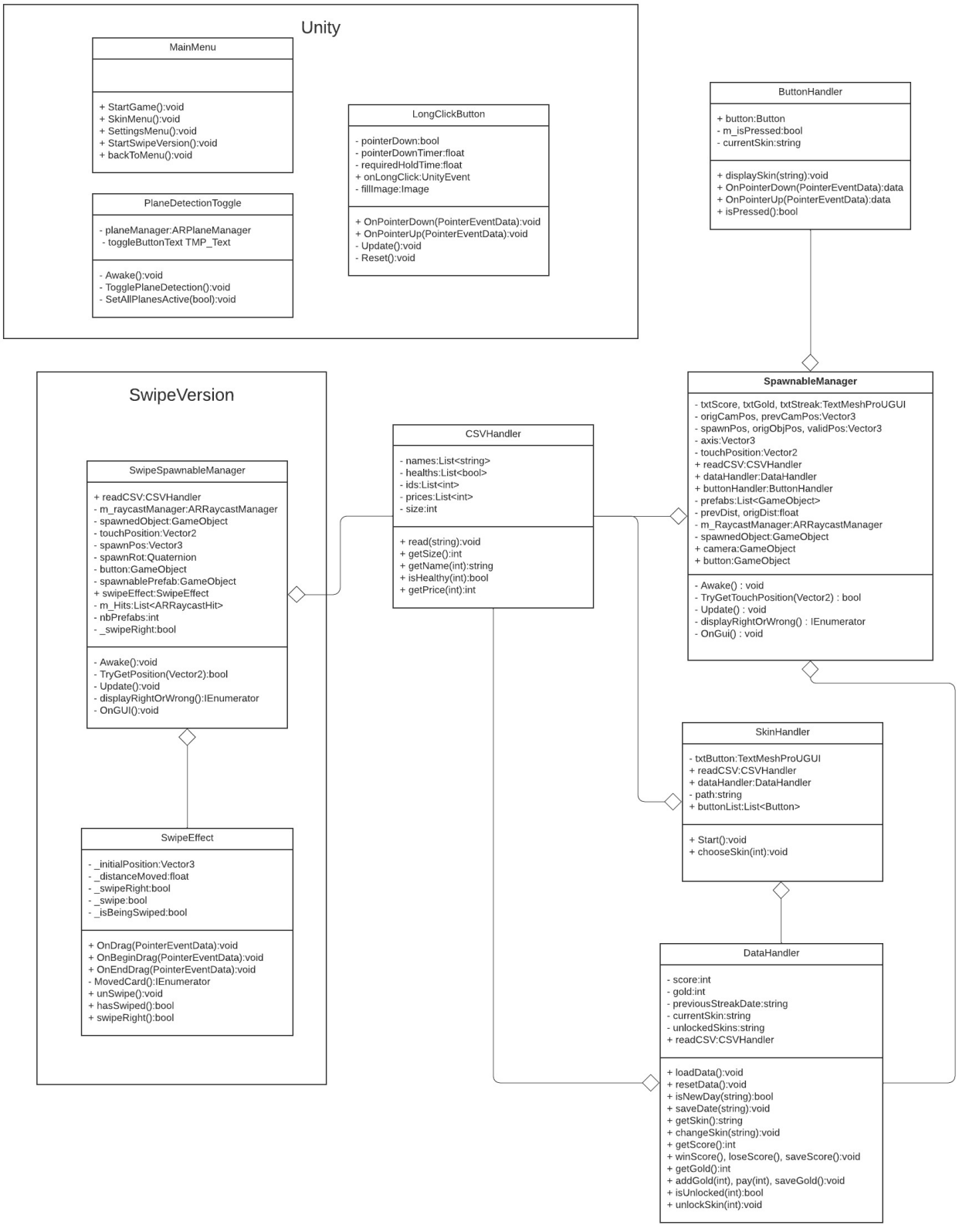


FIGURE 3 – Diagramme de classe UML

8 Réalisation

8.1 Implémentation

L'application est séparée en plusieurs écrans différents, correspondant chacun à une scène Unity. Chaque scène contient au minimum un Canvas, qui représente la disposition de l'écran du téléphone, et un objet EventSystem, qui permet de gérer les événements liés aux boutons. Pour passer d'une scène à l'autre, on utilise des boutons Unity auxquels on attache le script MainMenu. Celui-ci permet de charger la scène choisie grâce à la méthode $LoadSceneAsync()$ de la bibliothèque SceneManager de UnityEngine.

SampleScene est la scène principale de l'application, là où l'utilisateur effectue des actions en réalité augmentée. Cette scène contient une AR Session, un objet qui contient les scripts principaux de ARFoundation, et une AR Session Origin et son AR Camera, qui gèrent la caméra du téléphone, sa position dans l'espace et les plans. Le script SpawnableManager est lié à AR Session Origin et il peut donc s'exécuter au lancement de la scène.

Lors de son initialisation, sa méthode $Awake()$ s'active et charge les données sauvegardées grâce à la méthode $loadData()$ du DataHandler (cette méthode utilise les getters existants des PlayerPrefs de Unity et stocke les valeurs dans ses attributs - de même quand on voudra sauvegarder on utilisera les setters). Puis elle récupère les textes affichés à l'écran avec $GameObject.Find().GetComponent < Text > ()$ afin de pouvoir les modifier en temps réel. Elle lit ensuite le fichier list.csv contenant les informations sur les aliments grâce à la méthode $read()$ du CSVHandler (un simple StreamReader qui stocke les valeurs du CSV dans les attributs). Cela permet de récupérer les noms des prefabs correspondants et ainsi de les charger et les ajouter à la liste des prefabs du programme. Elle charge aussi un ARRaycastManager, qui permettra de gérer la plupart des actions liées à la caméra.

La méthode $TryGetPosition()$ retourne true si l'écran est touché, false sinon.

La méthode $displayRightOrWrong()$, si l'utilisateur fait le bon ou le mauvais choix, change le booléen correspondant pour activer l'affichage d'image dans $OnGUI()$ pendant un temps défini.

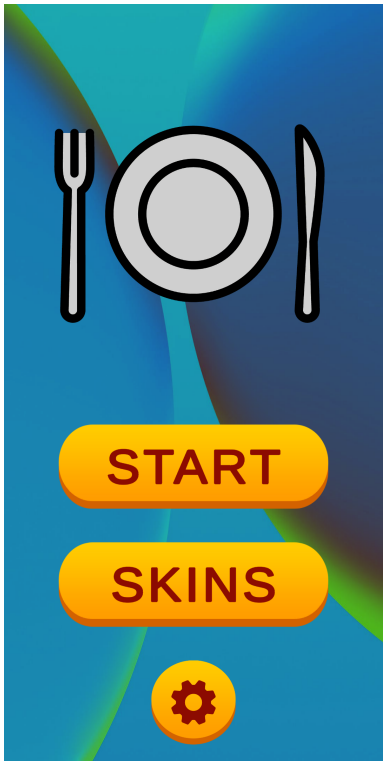
La méthode $OnGUI()$ affiche une image selon le résultat right/wrong. $OnGUI()$ est héritée de MonoBehaviour et ne dépend pas des frames ou d'$Update()$. Dès que ses conditions sont validées, elle s'active. Ainsi dès que le booléen de $displayRightOrWrong()$ passe en true, elle affiche l'image et dès qu'il repasse en false elle l'efface.

La méthode $Update()$ s'active à chaque frame. Elle appelle $TryGetPosition()$ et si cela retourne true, le Raycast Manager va lancer un rayon dans la direction du toucher sur le plan. Si aucun objet n'est instancié, elle choisit un prefab aléatoirement dans la liste puis l'instancie à la position du toucher avec la méthode $Instantiate()$. Si un objet est déjà instancié, elle va seulement changer sa position. Une fois l'objet instancié, la méthode vérifie si le bouton est enfoncé grâce à ButtonHandler (celui-ci fonctionne avec des simples PointerEventData de la bibliothèque EventSystems de Unity). Si c'est le cas et que c'est la première frame où le bouton est pressé, on stocke la position de départ de la caméra, la distance et l'axe (=vecteur normalisé) de départ entre l'objet et la caméra. Ensuite on cherche à maintenir la distance de départ entre la caméra et l'objet : si la distance actuelle est inférieure, c'est qu'on se rapproche de l'objet donc va l'éloigner le long de l'axe (et inversement). Puis si l'objet a parcouru une certaine distance, on active la routine de validation $displayRightOrWrong()$ et on prépare le mouvement

de rapprochement/éloignement de l'objet sur le référentiel caméra objet actuel (on passe donc sur un nouvel axe). Ce mouvement se fait simplement en changeant la position de l'objet le long de l'axe à chaque update, jusqu'à arriver à la caméra, ou à une distance équivalente au loin.

8.2 Résultats

Le premier écran, le menu principal, s'affiche au lancement de l'application. Il permet de lancer la scène principale ou d'accéder aux autres menus. Tous les autres écrans ont un bouton en haut à gauche qui permet de retourner au menu principal.



((a)) Menu principal

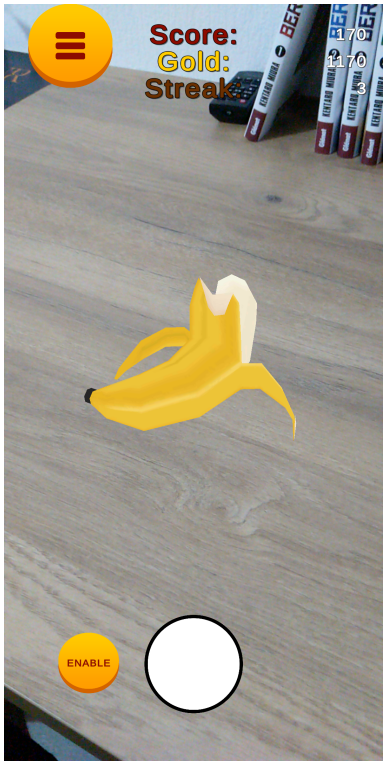
((b)) Aliment posé sur un plan

L'écran de jeu principal propose d'utiliser notre implémentation de l'Approach-Avoidance Training en réalité augmentée. Il commence par détecter les plans sur lesquels on pourra ensuite poser un aliment aléatoire par une pression du doigt. Une fois l'aliment posé, on peut le déplacer sur le plan. En pressant le bouton en bas de l'écran, on va pouvoir « l'attraper », puis l'approcher ou l'éloigner le long l'axe établi entre sa position de départ et la position de départ du téléphone (plus précisément sa caméra). Si on lâche le bouton, l'aliment retourne à sa position de départ. Sinon, une fois qu'il a parcouru une certaine distance, il va valider notre choix et effectuer des actions en conséquence.

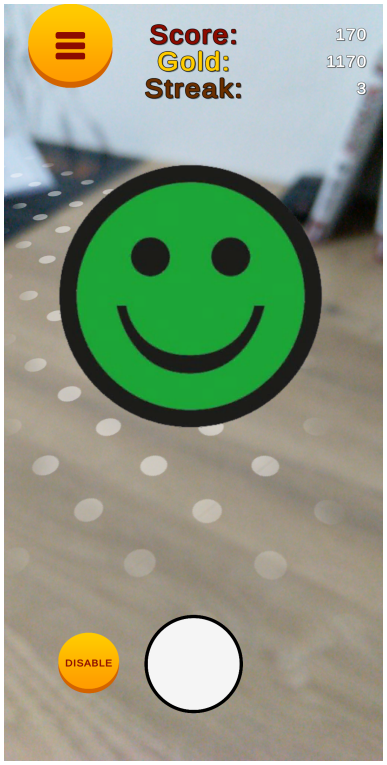
Si on l'a rapproché de soi pour le « manger », l'objet va continuer de se rapprocher jusqu'à la caméra (en suivant cette fois l'axe entre sa position lors de la validation et la position de la caméra au même moment). Si on l'a jeté au loin, il va s'éloigner de la caméra et disparaître au bout de quelques frames. Puis un visage souriant ou en colère s'affiche selon notre choix. Au bout de deux secondes, le visage disparaît et un nouvel aliment apparaît là où le précédent se trouvait sur le plan (si on touche le plan avant la disparition de l'image, l'aliment s'y place immédiatement).

Il est recommandé d'approcher le téléphone de l'objet avant d'appuyer sur le bouton, la distance à parcourir sera moins grande et l'utilisateur ne risque pas de devoir reculer. Cela permet

de mieux simuler le fait d'attraper l'objet : on approche sa main, une fois qu'il est à portée on appuie sur le bouton pour l'attraper, puis on le ramène pour le manger ou on le jette au loin.



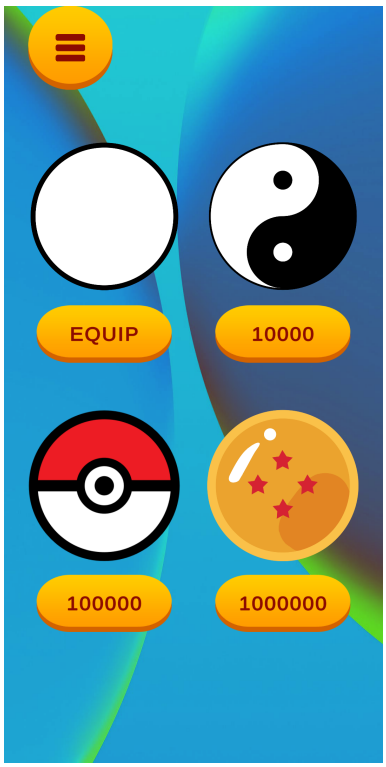
((a)) Plan caché



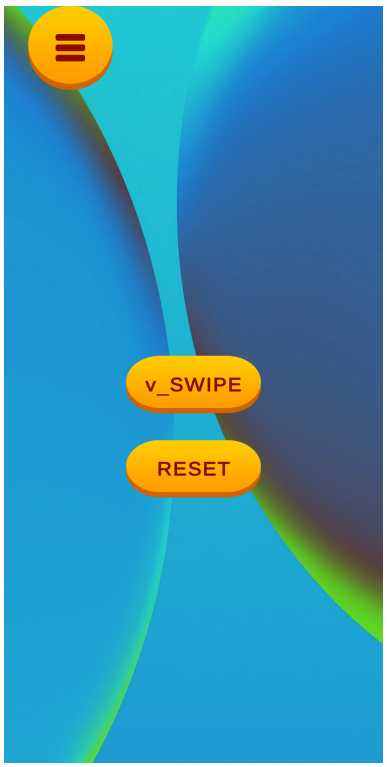
((b)) Validation

Au niveau de l'interface, on a un bouton en bas à gauche qui permet d'activer ou de désactiver l'affichage des plans, et trois compteurs se trouvent en haut à droite de l'écran : un pour le score qui s'incrémente ou se décrémente de dix à chaque choix, un pour l'argent, qui augmente de dix pour chaque bon choix, et un pour la chaîne de succès actuelle. Une fois par jour, comme récompense de connexion journalière, réaliser une chaîne de dix permet de gagner mille golds d'un coup. L'argent et le score sont sauvegardés, mais la chaîne se réinitialise en retournant au menu ou en quittant l'application.

L'écran de skins permet à l'utilisateur de changer l'apparence du bouton. L'apparence par défaut est gratuite, puis il faut payer avec l'argent gagné pour obtenir les autres. Une fois achetées, elles sont débloquées définitivement et au lieu d'afficher leur prix, leurs boutons respectifs proposent de les équiper directement.



((a)) Menu Skins



((b)) Menu Settings

L'écran d'options permet deux choses. On peut y réinitialiser les données sauvegardées (score, quantité de golds, skins débloqués). Pour l'activer il faut maintenir le bouton reset enfoncé pendant cinq secondes. En choisissant l'option swipe, on bascule sur un nouvel écran qui permet d'essayer la première version de l'application : il suffit de swiper sur l'écran au lieu de déplacer son téléphone. Cette version ayant été rajoutée après coup en bonus, elle n'est pas liée à toutes les autres fonctionnalités et ne permet pas de changer son score, ses golds ou son skin.

9 Tests

9.1 Test du singe

Nous avons effectué un test du singe en plusieurs temps : dans les menus et dans l'écran d'AR.

9.1.1 Menus

Nous avons commencé par enchaîner des aller-retours rapides entre les différents menus (et entre l'écran AR et le menu principal) afin de noter d'éventuels ralentissements. Nous n'avons rien remarqué de significatif. L'application n'étant de plus pas prévue pour ce type d'usage, il y a peu de chances qu'elle soit mise à si rude épreuve en conditions réelles.

Nous avons également tenté d'appuyer sur tous les boutons en même temps : les différentes scènes se contentent de se charger les unes à la suite des autres rapidement sans ralentir particulièrement l'application.

9.1.2 Écran principal

En appuyant à répétition sur les boutons, en déplaçant l'objet erratiquement et très rapidement nous ne remarquons aucun dysfonctionnement. Le seul problème rencontré, assez logiquement, est une difficulté à détecter les plans pendant plusieurs secondes après avoir secoué le téléphone. Mais notre détection de plans étant déjà très approximative en temps normal, cela ne change pas grand chose.

9.2 Chargement, mise à jour et sauvegarde des données

Pour vérifier que l'or ou le score s'incrémentent correctement, il suffit d'effectuer des actions et regarder les compteurs en haut de l'écran. En changeant de menu ou en quittant et relançant l'application, on peut également voir qu'ils sont correctement sauvegardés et chargés, tous comme les skins débloqués.

Pour les données non affichées directement à l'écran (comme la date de la précédente récompense journalière par exemple), nous avons utilisé des Debug.Log() afin de les voir dans la console sur Unity. Ainsi, après la lecture du fichier CSV nous avons pu voir que tous les noms d'aliments étaient affichés et que tous les prefabs correspondants étaient chargés.

10 Comparaison gantt prévisionnel et effectif

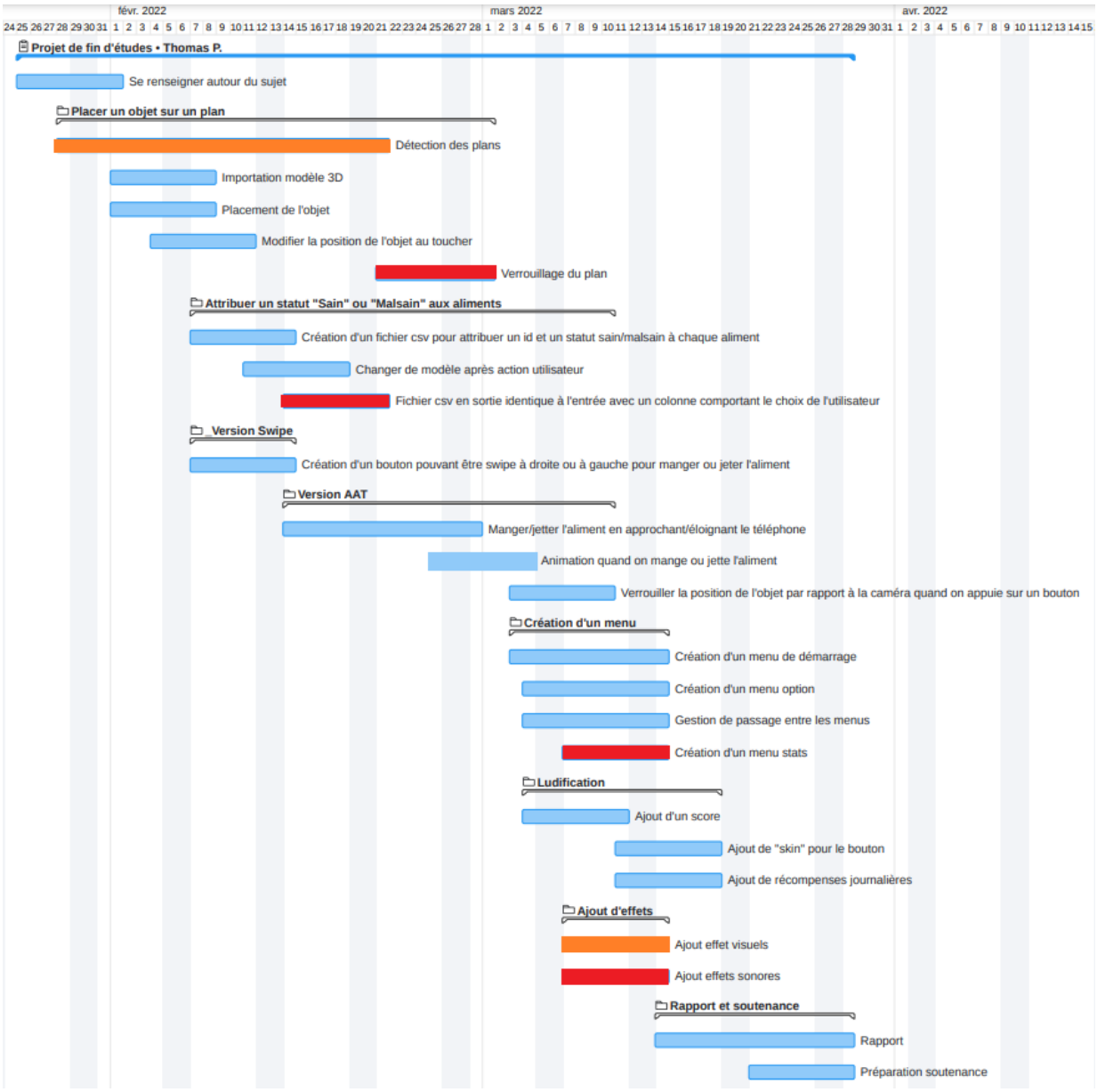


FIGURE 7 – Gantt prévisionnel modifié avec, en rouge, les tâches que nous n'avons pas eu le temps de faire et en orange, celles qui ne sont pas complètement terminées

Nous avons rencontré plusieurs difficultés lors du projet qui ont ralenti notre avancée par rapport au planning initial. De plus, après le premier rendez-vous avec les clients, nous avons ajouté plusieurs nouvelles fonctionnalités à implémenter.

Par rapport à ce que nous avions prévu au début en termes de planning, la partie détection de plan nous a pris beaucoup plus de temps. Nous avions rapidement une détection simple qui fonctionnait, mais nous avons perdu beaucoup de temps à chercher des solutions pour une meilleure détection, plus robuste, que nous n'avons pas réussi à faire finalement pour privilégier l'implémentation de la version AAT ainsi que le menuing. De plus, nous n'avons pas réussi à verrouiller le plan. À cause du retard pris pour la détection, et l'ajout de petites fonctionnalités au cours du projet, la plupart des tâches se sont donc retrouvées décalées dans le planning et

nous avons donc dû en oublier certaines.

Nous n'avons donc pas de fichier CSV en sortie avec une colonne contenant le choix de l'utilisateur, ni de menu statistiques, les seules statistiques que nous avons étant le score, les golds et la streak affiché sur l'écran principal. De plus, nous n'avons pas eu le temps d'ajouter d'effets sonores. Pour les animations, nous avons juste les aliments qui se rapprochent de la caméra lorsqu'on les mange et s'éloigne lorsqu'on les jette, mais pas d'effets particuliers pour les autres actions utilisateur.

Au final, toutes les tâches se sont retrouvées décalées d'une semaine ou deux, et donc pour le rapport il ne nous restait que la dernière semaine, durant laquelle nous avions encore quelques modifications de code à faire.

Mais à part cette détection de plans, l'ajout d'effets visuels et sonores, les statistiques et le fichier log csv de sortie, nous avons dans l'ensemble réussi à implémenter tout le reste des fonctionnalités prévues.

11 Conclusion

11.1 Bilan

Le sujet s'est avéré plus difficile que prévu au niveau de la détection de plans, qui nous a posé quelques difficultés : parfois il faut bouger dans toute la pièce pour trouver un plan, parfois on en trouve beaucoup trop et ils ne sont pas bons, et parfois cela fonctionne sans problème. Nous avons perdu pas mal de temps sur cela, c'est ce qui nous a le plus ralentis dans le projet et nous a empêché de réaliser certaines tâches qu'on s'était fixées au début.

Cependant, la plupart des autres fonctionnalités que nous avons eu le temps d'implémenter fonctionnent bien. Les prefabs se chargent et s'affichent correctement, le chargement, la sauvegarde et l'édition des données de l'utilisateur fonctionnent. Tous les boutons et menus font leur travail, et l'interface est visuellement plutôt propre. On a un bon début de gamification avec un score, un monnaie qui permet d'acheter des apparences et un système basique de récompenses journalières.

Mais surtout la fonctionnalité la plus importante fonctionne : l'utilisateur peut attraper un aliment et le rapprocher ou l'éloigner de lui pour le manger ou le jeter, et l'application lui fait un retour positif ou négatif. Nous avons donc réussi à apporter un prototype qui fait ce qui était attendu dans le sujet.

Ce projet nous a permis de découvrir plus en détail le fonctionnement d'un projet équipe réduite, avec un client que l'on voit régulièrement et un délai court, ce qui peut arriver dans le monde professionnel.

11.2 Perspectives

Dans le futur, la première chose à faire serait d'implémenter une détection de plans plus efficace, par exemple en créant un plan étant la moyenne des plans alentours. Ou alors en utilisant un QR Code posé sur la table pour avoir le niveau et placer l'aliment dessus. Cette dernière solution implique que l'aliment ne peux plus être déplacé et surtout que l'application est utilisable seulement à la maison, ou là où le QR Code est collé, c'est pourquoi cette solution ne serait que temporaire.

On pourrait aussi, pour améliorer l'application, faire des animations plus complètes lorsqu'un aliment est mangé ou jeté et ajouter des effets sonores et effets visuels.

De plus, nous pourrions améliorer la gamification, en ajoutant par exemple la possibilité de viser et jeter, à l'aide des gestes sur l'écran, l'aliment dans un panier de basket/une poubelle pour gagner plus de points. On pourrait aussi améliorer le système de récompenses journalières en augmentant la récompense si les jours d'utilisation s'enchaînent. Un autre ajout utile serait un menu de statistiques pour connaître ses statistiques globales et suivre son évolution et ajouter un mode en ligne pour comparer et partager sa progression.

Nous avions aussi pensé à faire des dons à des associations en lien avec les problèmes de santé liés à la nourriture, à l'aide de points ou en débloquant des récompenses par exemple. On pourrait même complètement changer la méthode et attraper les aliments en passant sa main devant la caméra au lieu de bouger le téléphone par exemple. Enfin, on pourrait évidemment augmenter la base de donnée des aliments et y ajouter des boissons par exemple (sodas et eau).

12 Annexe

Projet de fin d'étude

Réalité Augmentée au service d'une alimentation saine

Encadrant:
Arnaud Prouzeau, Équipe Potioc, arnaud.prouzeau@inria.fr

En collaboration avec:
Antonio Verdejo-Garcia, Monash University, Melbourne, antonio.verdejo@monash.edu
Simon Van Baal, Monash University, Melbourne, simon.vanbaal1@monash.edu

Contexte:
L'obésité est en partie due à des choix alimentaires malsains eux-mêmes provoqués par des biais cognitifs, notamment le biais d'approche (une tendance à se diriger vers la nourriture malsaine) et une préférence pour les récompenses immédiates. L'entraînement à l'évitement du biais d'approche (en anglais "Approach-Avoidance Training - AAT) est un entraînement cognitif dans lequel les utilisateurs sont entraînés à effectuer un mouvement d'évitement lorsqu'ils voient des aliments malsains, il s'est révélé efficace dans plusieurs contextes similaires [1]. Récemment, des chercheurs ont étudié l'utilisation d'une application pour smartphone afin de permettre aux participants de s'entraîner lorsqu'ils en ont le temps, chez eux ou en déplacement, mais aussi de le faire sur une plateforme plus réaliste et plus attrayante [2]. Les résultats sont encourageants et suggèrent, en raison de sa disponibilité, qu'un tel entraînement pourrait également bénéficier aussi à la population générale intéressée par une meilleure alimentation.

Projet:
Dans le cadre de ce projet, les étudiants devront développer un prototype en réalité augmentée (RA) sur un smartphone permettant la mise en œuvre de l'AAT à domicile. L'application doit permettre aux utilisateurs de visualiser des aliments virtuels dans leur propre maison (sur la table du dîner, le comptoir de la cuisine, etc.) et leur permettre de les jeter ou non, en utilisant des gestes physiques (soit des mouvements du téléphone, soit des gestes de la main). Les étudiants seront également invités à étudier comment fournir des articles virtuels qui sont familiers à chaque utilisateur.
L'application sera développée à l'aide de Unity [3] en utilisant C# et la toolkit 'AR Foundation' [4]. Ce projet est réalisé en partenariat avec l'université Monash (Melbourne, Australie), mais les réunions se dérouleront en général pendant les heures de travail françaises.

References:
[1] Schumacher, S. E., Kemps, E., & Tiggemann, M. (2016). Bias modification training can alter approach bias and chocolate consumption. Appetite, 96, 219-224.
[2] Becker, D., Jostmann, N. B., Wiers, R. W., & Holland, R. W. (2015). Approach avoidance training in the eating domain: Testing the effectiveness across three single session studies. Appetite, 85, 58-65.
[3] https://unity.com/
[4] https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html

FIGURE 8 – Sujet officiel

13 Bibliographie

Références

[1] Hawker C Castine B de Courten B Verdejo-Garcia A Kakoschke, N. Smartphone-based cognitive bias modification training improves healthy food choice in obesity : A pilot study,.
Disponible sur : https://onlinelibrary.wiley.com/doi/10.1002/erv.2622, consulté le 25/03/2022.

[2] Kemps E. Tiggemann M. Kakoschke, N. Approach bias modification training and consumption : A review of the literature.
Disponible sur : https://www.sciencedirect.com/science/article/abs/pii/S0306460316302787?via%3Dih consulté le 25/03/2022.

[3] Daniel T. O. Epstein L. H O'Neill, J. Episodic future thinking reduces eating in a food court.
Disponible sur : https://www.sciencedirect.com/science/article/abs/pii/S1471015315300143?via%3Dih consulté le 25/03/2022.

[4] Schmidt S Degner J, Steep L and Steinicke F. Assessing automatic approach-avoidance behavior in an immersive virtual environment.
Disponible sur : https://doi.org/10.3389/frvir.2021.761142, consulté le 26/03/2022.

[5] Nielsen A.S. Ascone L. et al. Mellentin, A.I. A randomized controlled trial of a virtual reality based, approach-avoidance training program for alcohol use disorder : a study protocol.
Disponible sur : https://doi.org/10.1186/s12888-020-02739-1, consulté le 26/03/2022.

[6] J Weimann. Unity3d UGUI Hold to Click Buttons.
Disponible sur : https://unity3d.college/2018/01/30/unity3d-ugui-hold-click-buttons, consulté le 27/03/2022.