

Rapport PFE

Victor Andrault, Arthur Blondeau, Maxime Constant, Maël Galesne

27 mars 2023

Table des matières

1	Introduction	2
1.1	Contexte	2
1.2	Problématique	2
1.3	Plan	2
2	État de l’art et existant	3
2.1	Mot-clés	3
2.2	Visualisation des données et leur compréhension	3
2.3	Méthode d’apparition	3
2.4	Méthode de fondu	4
2.5	Méthode de modification de maillage	6
3	User Story	7
4	Implémentation	7
4.1	UI	7
4.1.1	Prototype de l’application	7
4.1.2	Implémentation de l’interface utilisateur	8
4.2	Algorithmes pour les transitions	13
4.2.1	Compression	13
4.2.2	Slicer	14
4.2.3	SizeReducing	15
4.2.4	Smooth	17
4.2.5	Launch	18
4.3	Transitions non retenues	19
4.3.1	FlyingCube	19
4.3.2	Fragment	20
5	Tests unitaires	20
5.1	Tests de Compression	20
5.2	Tests de Slicer	21
5.3	Tests de SizeReducing	22
5.4	Tests de Smooth & Launch	23
6	Test Utilisateur	24

7 Conclusion	27
7.1 Bilan	27
7.2 Perspective	28

1 Introduction

1.1 Contexte

Notre cliente, Ambre Assor, travaille sur un projet de sensibilisation de la production de déchets inutiles. Elle a implémenté un logiciel en réalité augmentée permettant de visualiser la quantité de déchets produits par une personne sur différentes périodes de temps, à partir d'un graphique. Cela permet de mieux se rendre compte de ce que les nombres qu'on a l'habitude de voir représentent vraiment. Avoir la possibilité de voir dans son salon les déchets que l'on a produit le mois précédent est bien plus parlant qu'un chiffre tel que 0,5 tonne.

1.2 Problématique

La transition qu'il y a entre une barre du graphique et les poches poubelle qui lui correspondent ne permet pas vraiment à l'utilisateur de bien faire le lien entre les deux. Nous avons donc travaillé sur différentes transitions, partant d'un objet 3D simple comme une barre de graphe, vers un objet 3D plus complexe comme une poche poubelle, afin que la simulation ait un effet plus marquant sur l'utilisateur.

1.3 Plan

Dans un premier temps, nous allons effectuer un état de l'art sur la visualisation et la compréhension des données par l'utilisateur ainsi qu'un existant sur les méthodes de transitions d'objets 3D. Ensuite nous vous présenterons une User Story. Par la suite nous vous parlerons de l'implémentation de nos transitions et de l'interface utilisateur. Enfin, nous vous montrerons les tests unitaires que nous aurions voulu effectuer, ainsi que les tests Utilisateurs que nous avons faits.

2 État de l’art et existant

2.1 Mot-clés

Transition, Fondue, Déformation de maillage, Visualisation des données.

2.2 Visualisation des données et leur compréhension

La visualisation des données est un point majeur de la compréhension et de l’affect des données statistiques pour l’utilisateur. Son intégration dans le domaine du *computer graphics* a permis, grâce à l’évolution des capacités des ordinateurs, d’avoir accès à toujours plus de données. Le problème étant qu’avec autant de données différentes, il peut être compliqué de les comprendre ou de les interpréter. C’est dans cette optique que [1] propose un état de l’art assez complet de différentes méthodes et algorithmes pour la visualisation de données.

L’importance de la visualisation et de la compréhension des données a un impact réel sur le temps et la difficulté du travail, c’est ce que démontre [2]. Son but est de créer une méthode pour avoir une métaphore graphique des données de patients neonatals en soins intensifs afin de faciliter le travail des médecins et infirmiers. Cette modification des données en un support graphique plus visuel a deux principaux buts ; le premier est de rendre le travail plus facile pour les professionnels de santé, le deuxième qui s’obtient grâce au premier est de permettre un meilleur traitement des patients en cas de changement drastique de leur état de santé.

2.3 Méthode d’apparition

Dans le cas d’une transition entre 2 objets différents, la méthode de faire disparaître le premier objet et apparaître le second objet est la plus naïve. Elle est de ce fait une bonne méthode pour introduire le concept, mais n’est pas bonne si le but est de créer une transition fluide, rapide et compréhensible pour l’utilisateur.

Elle est réalisable en temps constant et la transition ne peut être qu’instantanée. Il est néanmoins possible d’ajouter des effets visuels afin de tromper l’oeil et de rendre l’effet de transition plus vraisemblable.



Figure 1 : Effet d'Apparition / Disparition sur le jeu-vidéo Hydroneer.

On peut remarquer que dans le jeu-vidéo Hydroneer sorti en 2020 [3] , l'utilisation d'une méthode d'apparition permet de changer les objets tout en diminuant les impacts sur le temps de calcul, mais pour que l'effet soit réaliste, un contexte d'incinérateur à été mis en place. Vous trouverez dans le dossier *Annexe* un fichier .gif du nom de *Annexe1.gif* qui montrera plus en détail la transition.

2.4 Méthode de fondu

Il est également possible d'effectuer une transition en utilisant une méthode de fondu, soit par applatissement, soit comme dans le jeu-vidéo Elden Ring sorti en 2022 [4] par opacité.



Figure 2 : Apparition de l'objet A qui va se transitionner en objet B.

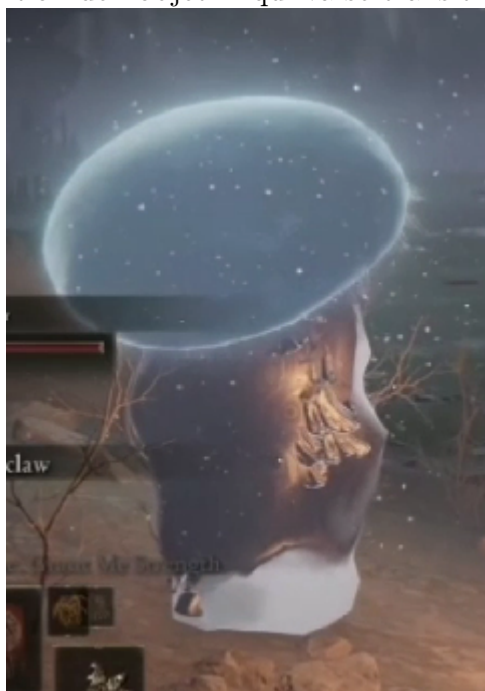


Figure 3 : L'objet B apparait pendant que l'objet A perd en opacité.

Vous trouverez dans le dossier *Annexe* un fichier .gif du nom de *Annexe2.gif* qui montrera plus en détail la transition.

L'avantage de ce type de technique est d'être fluide et facilement compré-

hensible, il est cependant recommandé d’appliquer un effet visuel en plus de la transition pour accentuer l’effet de transformation. Dans le cas de Elden Ring, pour faire apparaître une créature imitant les caractéristiques de notre personnage, la transformation se fait par opacité entre une créature liquide et une imitation de notre personnage, avec l’ajout de particules volantes pour tromper l’œil et rendre la transformation plus crédible.

2.5 Méthode de modification de maillage

Le premier papier notable sur la déformation de maillage [5] propose une approche permettant de modifier et déformer librement un maillage peu importe le type ou le degré de sa primitive. Cette méthode proposée en 1986 utilise des méthodes encore utilisées de nos jours pour la déformation de maillage, à savoir l’utilisation de courbes de Bézières et de matrices Jacobien.

Néanmoins, dans le cas de notre application, nous devons appliquer une transition très rapidement. Une méthode rapide est proposée avec [6]. Leur méthode promet de surpasser l’état de l’art sur des grands maillages aussi bien en temps de calcul qu’en coût de mémoire. Cet algorithme représente une méthode efficace pour la déformation de maillage. Cependant, il ne permet de modifier le maillage que de manière cinématique (directe ou inverse). Par conséquent nous ne pouvons pas implémenter cette méthode tel quel. En effet, nous avons besoin de deux maillages différents pour notre transition et la méthode proposée travaille sur un seul maillage.

Nous nous sommes également penchés sur des travaux ou des projets conçus sous Unity directement. Nous avons retenu un dépôt git [7] qui propose une méthode de modification de maillage d’un objet A vers un objet B, ainsi qu’un asset unity de déformation de maillage [8].

Dans le cas du dépôt git, nous avons pu le tester, et vous retrouverez également, dans le dossier *Annexe*, une vidéo au format gif nommé *Annexe3.gif* qui montre le résultat de l’algorithme. Cette méthode propose une transition très rapide et précise. Malheureusement elle ne fonctionne que si les 2 objets de la transition possèdent un maillage avec exactement le même nombre de sommets, d’arêtes et de faces.

Concernant l’asset Unity, il propose une méthode de transformation de mesh en travaillant sur les shaders afin de calculer la bonne position des arêtes. Cette méthode semble fonctionner même si les deux objets n’ont pas un maillage avec un nombre similaire d’arêtes et de sommets. Néanmoins la

transition semble un peu lente et peu visuelle, ce qui ne correspond pas vraiment aux attentes pour notre application.

3 User Story

L'utilisateur doit pouvoir interagir avec un histogramme et observer une transition fluide, rapide et compréhensible entre un élément de l'histogramme et son équivalent en un objet complexe. Prenons un histogramme représentant le nombre de sacs de poubelle utilisés par mois, l'utilisateur doit pouvoir choisir un élément de l'histogramme correspondant à un mois et voir l'élément se transformer en sacs poubelle équivalent à la valeur de l'élément sélectionné.

4 Implémentation

4.1 UI

4.1.1 Prototype de l'application

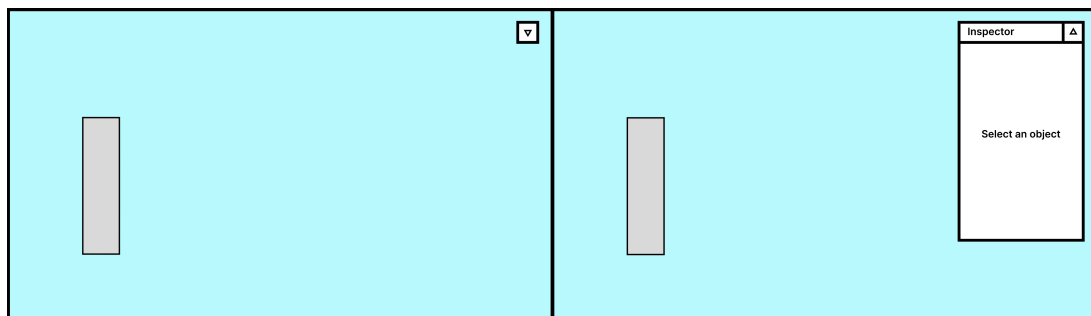


Figure 4 : Maquette Figma slide 1 et 2.

Dans un premier temps, nous avons créé une maquette à l'aide du site internet Figma afin d'anticiper les attentes de notre cliente pour l'interface utilisateur.

Dans cette première version, nous avons choisi d'ouvrir un menu "Inspector" avant toute interaction avec les barres du graphe, représentées ici par un simple rectangle gris. Une fois l'Inspecteur ouvert, l'utilisateur devait sélectionner un objet qui, dans notre cas, était toujours une barre.

Après avoir échangé avec le groupe et la cliente, plusieurs points négatifs ont été identifiés. Tout d'abord, l'utilisateur ne comprenait pas immédiatement qu'il devait ouvrir le menu avant de pouvoir interagir avec la scène. Ensuite, une fois le menu ouvert, l'intitulé "Select an object" était trop vague, car l'application allait contenir plusieurs types d'objets et seules les barres devaient être sélectionnables.

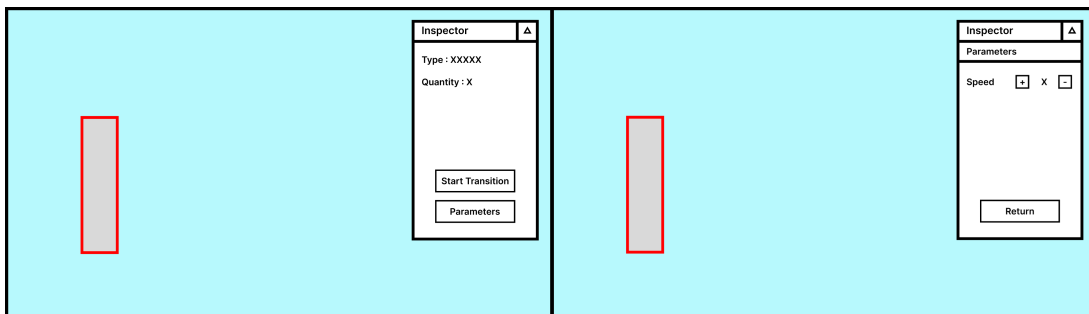


Figure 5 : Maquette Figma slide 3 et 4.

Une fois la barre sélectionnée, l'Inspecteur affiche le type de donnée qu'elle représente ainsi que son nombre. Deux boutons cliquables sont également présents, le premier "Start Transition" lance la transition actuellement implémentée dans la scène. Le deuxième ouvre le menu des paramètres, qui permet de modifier le facteur vitesse de la transition avant de la lancer.

Quelques éléments intéressants ont plu à l'équipe et à la cliente, notamment le contour rouge autour de la barre actuellement sélectionnée. Concernant la page principale de lancement de transition, il a été convenu d'ajouter un menu déroulant afin de pouvoir sélectionner directement dans le menu la transition à tester, puisqu'il y aura plusieurs transitions réalisées pour ce projet et non seulement une seule. Pour le menu des paramètres, la vitesse étant le paramètre principal et commun à toutes les transitions, elle est la seule à avoir été gardée pour la version finale, afin d'éviter tout conflit entre les différentes transitions.

4.1.2 Implémentation de l'interface utilisateur

4.1.2.1 Première version

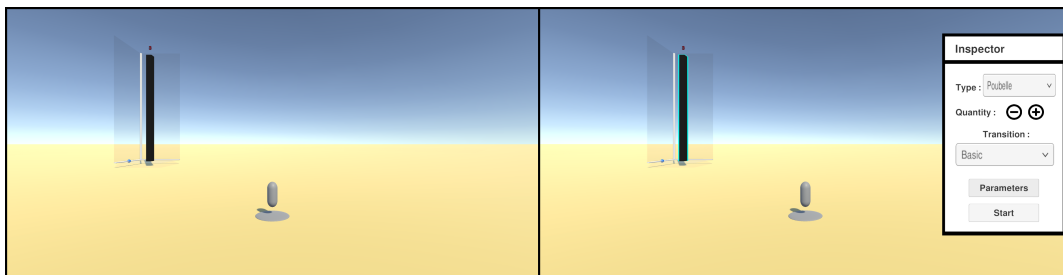


Figure 6 : Scène Unity V1

Dans cette première version de l'interface, le type de graphique utilisé par la cliente était affiché directement dans l'interface utilisateur (UI). Un repère gris au centre représentait le point d'apparition de toutes les transitions. L'icône pour ouvrir le menu afin d'interagir avec la scène a été supprimée par rapport au prototype initial. Désormais, il suffit de cliquer directement sur la barre pour afficher l'inspecteur.

Dans cet inspecteur, plusieurs éléments variaient par rapport à la maquette :

- Le type pouvait être modifié parmi les deux types d'éléments de base fournis par la cliente, à savoir "Poubelle" ou "CO2". Cela avait pour impact supplémentaire de changer la couleur de la barre pour différencier le type directement à l'œil nu.
- Pour fermer l'inspecteur, il suffisait de recliquer sur la barre déjà sélectionnée pour la désélectionner.
- La quantité était affichée uniquement au-dessus de la barre dans le graphique. Toutefois, il était possible de modifier cette quantité en cliquant sur les icônes "-" ou "+" du menu, mais sans pouvoir passer en dessous de 1 et au-dessus d'un maximum arbitraire de 10.
- Le menu déroulant discuté lors du prototypage a été implémenté avec une simple transition dite "Basic" afin de tester la bonne apparition des objets. À ce moment-là, aucune autre transition n'avait encore été raccordée à l'interface.

Plusieurs problèmes étaient survenus lors de ce premier jet :

- Beaucoup de temps avait été perdu à la compréhension de l'asset de génération de graphique fourni par la cliente, avec une navigation difficile entre les différents scripts importés.
- Il n'était possible d'interagir qu'avec une seule barre, notamment à cause du changement de type et de quantité depuis l'interface qui

générait des conflits lorsque plusieurs barres étaient sélectionnées.

Pour les pistes d'amélioration, la possibilité de déplacer le point d'apparition avec les touches "ZQSD" a été relevée, ainsi que l'idée de donner des couleurs plus vives à ce dernier.

4.1.2.2 Deuxième version

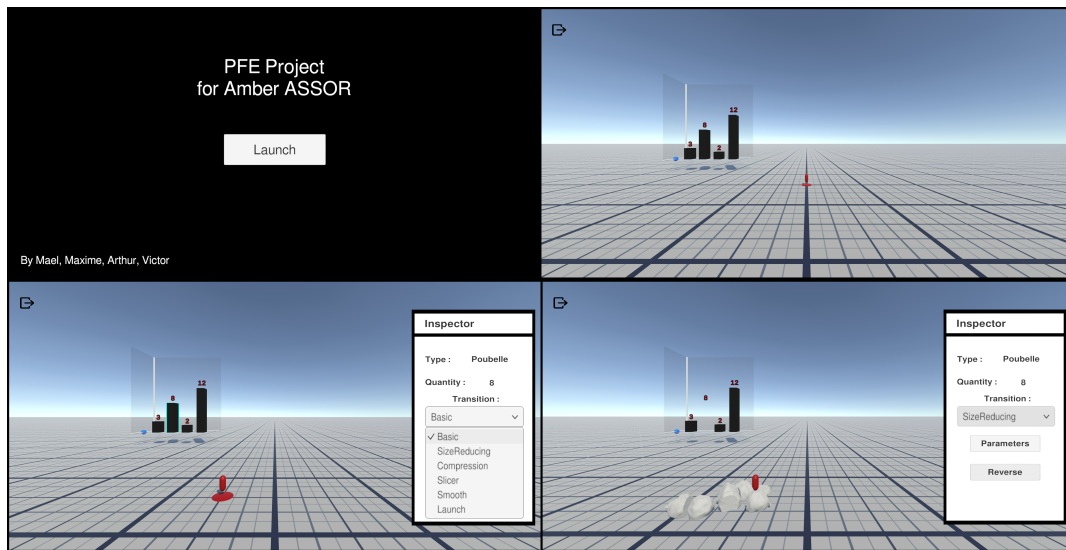


Figure 7 : Scène Unity V2

Dans cette deuxième itération de l'interface, des choix ont été faits par rapport à la précédente.

En commençant par les ajouts :

- Ajout d'un menu simple présentant notre projet et ses membres à l'ouverture de l'application.
- Dans la scène principale, le sol a été remplacé par un quadrillage pour une meilleure visibilité du placement du point d'apparition, qui est désormais déplaçable.
- Ajout d'une icône en haut à gauche de l'interface pour quitter l'application (il suffit de cliquer dessus).
- Le bouton "Start" qui permet de lancer la transition devient "Reverse" une fois lancé et vice-versa à chaque lancement.

Dans le cas des modifications :

- Afin de pouvoir interagir avec plusieurs barres et suite au problème rencontré lors de la précédente itération de l'interface, il n'est plus possible de modifier ni le contenu, ni la quantité d'une barre une fois dans l'application.
- Il n'est possible de sélectionner qu'une seule barre à la fois. Si l'on sélectionne une nouvelle barre, l'ancienne se désélectionne (visible par la perte de son contour bleu).

La plus grosse difficulté lors de l'implémentation de cette deuxième interface fut le raccordement de toutes les transitions réalisées par l'ensemble du groupe afin qu'elles coexistent toutes entre elles, sans s'impacter mutuellement de manière néfaste.

Nous avons fait tester cette version de l'interface auprès de nos utilisateurs, vous pouvez retrouver les tests utilisateurs dans la section 6. Il en est ressorti certains points d'amélioration qui seront apportés à la prochaine itération :

- Un manque d'information visuelle sur l'objectif et l'utilisation de l'interface une fois passé le menu.
- Rendre l'interface plus harmonieuse et attrayante pour l'utilisateur.

4.1.2.3 Troisième version

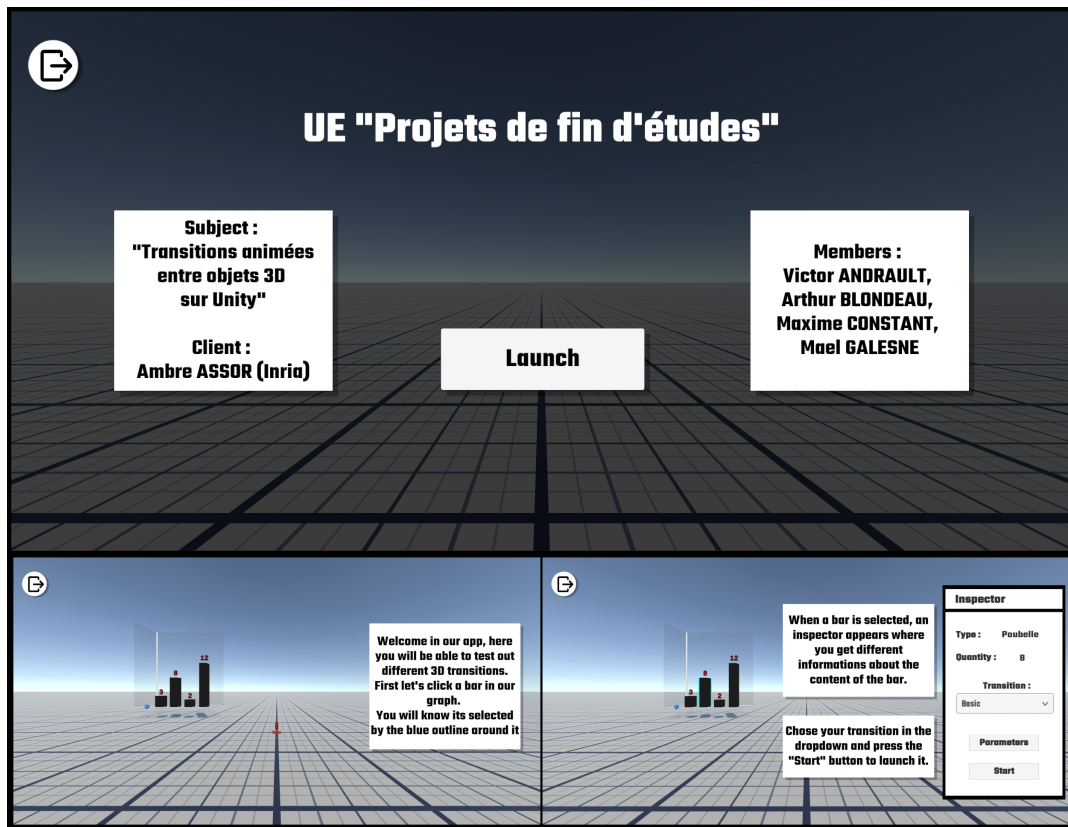


Figure 8 : Scène Unity V3

Dans cette version finale, l'accent a été mis sur l'expérience utilisateur avec l'ajout d'éléments pour faciliter la navigation dans l'application ainsi que pour améliorer ses éléments visuels.

Liste des nouveautés :

- Le menu, qui précédemment n'était qu'une ébauche, a été retravaillé pour présenter les différents actifs du projet, son sujet et la matière créditée.
- Un bouton pour quitter l'application, présent dans la scène dès le lancement, permet à l'utilisateur de quitter celle-ci à tout moment. Il est situé en haut à gauche de l'écran et est représenté par une icône de porte et de flèche.
- Ajout de plusieurs bulles de dialogue pour orienter l'utilisateur lors de sa première utilisation.
- La police a été changée pour une police plus attrayante et agréable à l'œil.

4.2 Algorithmes pour les transitions

4.2.1 Compression

La transition "Compression" consiste à aplatisir une barre du graphe tout en faisant apparaître les poubelles qui correspondent à cette barre les unes après les autres. Dans la première version, elle a été implémentée sur un pavé classique, et les poubelles apparaissaient soit à côté, soit à une position donnée en argument, mais s'empilaient pour faire une grande colonne avant de s'effondrer. Aussi, la transition commençait au moment de la collision avec le plan.

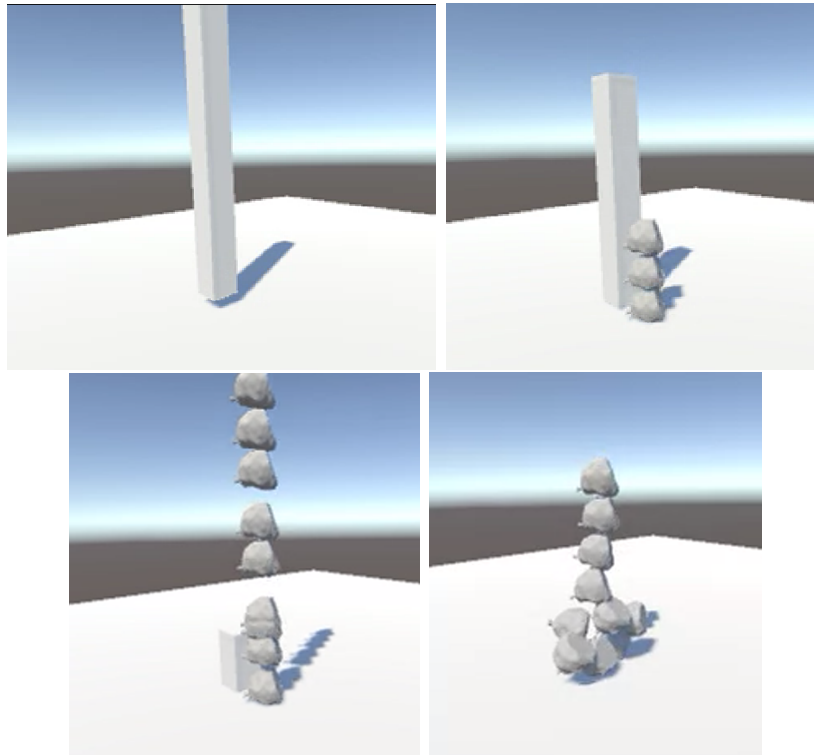


Figure 9 : Transition "Compression" avec un pavé simple.

Quand il a fallu faire la transition vers le modèle de graphe utilisé par la cliente, nous avons dû faire commencer la transition avec un bouton, et désigner l'emplacement d'apparition des poubelles par un objet (une capsule). Comme il n'y avait plus de collision nécessaire, le rigidbody a été retiré. On a aussi fait en sorte que les poubelles apparaissent autour du point désigné avec une part d'aléatoire afin de ne pas avoir une colonne de poubelles. La

transition peut être inversée. Le seul problème restant est que la barre ne s'applatit pas à sa base.

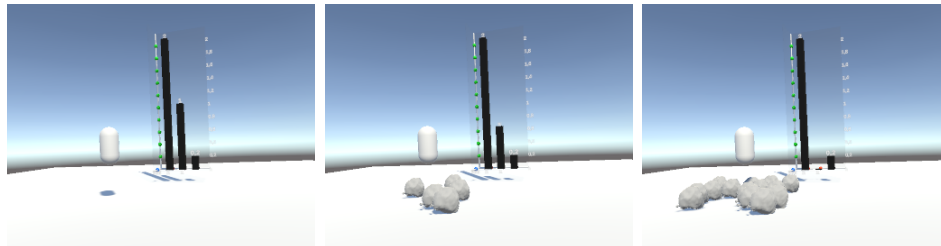


Figure 10 : Transition "Compression" avec la deuxième barre.

Vous trouverez en annexe une vidéo au format gif du nom de *Transition_Compression.gif* qui montre la transition.

4.2.2 Slicer

La transition "Slicer" consiste à projeter la barre dans une direction après un petit bond initiale, et de la découper en petits cubes au moment de l'impact avec le plan. Ces cubes se transforment en poubelles après un certain nombre de collisions. Dans la première version, cette transition se faisait sur un pavé classique. A la première collision, le pavé était remplacé par des cubes empilés, qui se chevauchaient légèrement afin qu'il y ai un effet de projection vers le haut. Après plusieurs collisions, ils se transformaient en poubelles. Le problème du nombre de collisions nécessaires pour initier le passage de cube vers poches poubelle est que, si il était trop bas, les poches poubelles apparaissaient sans qu'on ai le temps de voir les cubes. Au contraire, si il était trop grand, les cubes ne devenaient jamais des poches poubelles. Nous avons donc mis un système de minuterie, afin que la transition se fasse même si le nombre de collisions n'est pas atteint.

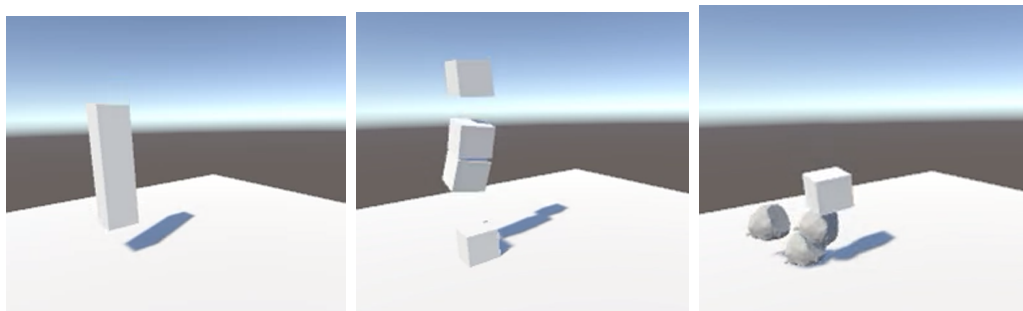


Figure 11 : Transition "Slicer" avec un pavé simple.

Avec la transition vers le modèle de graphe de la cliente, il a fallu faire en sorte que les cubes prennent la bonne taille, la bonne couleur et que le nombre s'adapte, au vu de l'épaisseur de la barre et de sa hauteur. On a eu des difficultés pour implémenter la transition inverse, car il fallait pouvoir faire disparaître les poubelles liées à la barre, et faire réapparaître cette dernière. Nous avons donc décidé de cacher la barre en la déplaçant plutôt que de la supprimer, afin de pouvoir ensuite la remettre à sa place, et de rendre les cubes transparents et intengibles, plutôt que de les supprimer aussi, pour pouvoir garder un lien avec les poubelles, et tous les supprimer lors de la transition inverse. En réappuyant sur le bouton, les poubelles disparaissent donc, et la barre réapparaît à sa position initiale.

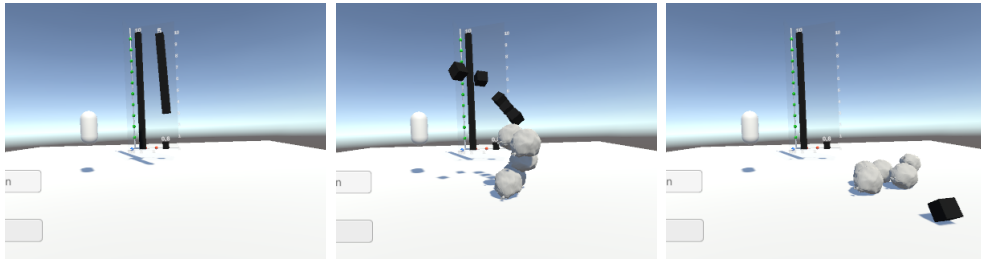


Figure 12 : Transition "Slicer" avec la deuxième barre.

Vous trouverez en annexe une vidéo au format gif du nom de *Transition_Slicer.gif* qui montre la transition.

4.2.3 SizeReducing

La transition "SizeReducing" fonctionne via un fondu par aplatissement. Le but de cette transition est de donner l'impression que les objets complexes, ici des sacs poubelle, semblent être stockés dans la barre et qu'ils sont libérés en même temps que l'applatissage de la barre. Pour que l'effet soit plus convaincant, il faut que la barre soit surélevée afin de laisser les poubelles tomber. Afin d'éviter une colonne de sacs poubelles comme avec le premier jet de la transition "Compression" 4.2.1, le premier sac poubelle est légèrement décalé par rapport aux autres.

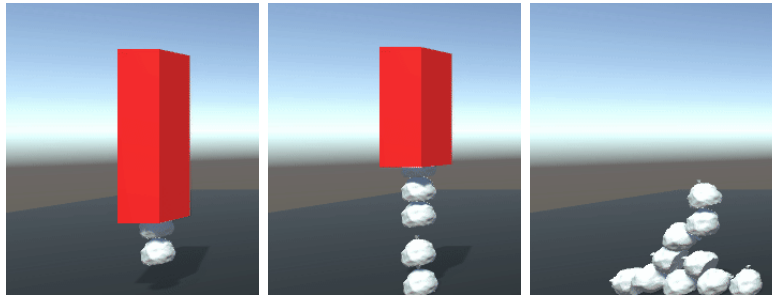


Figure 13 : Transition "SizeReducing" avec un pavé simple.

Pour un meilleur aperçu de la transition. Vous trouverez en annexe un fichier au format .gif intitulé *transition_SizeReducing_early.gif*.

Lors de l'ajout de la transition à l'asset Unity de graphe utilisé par la cliente, il a fallu modifier l'appel de la transition. Tout d'abord lorsque l'on clique sur une barre sur graphe, la barre va être remplacé par des cubes contenant la transition, les cubes étant de taille x , 1 , z . Avec x et z les valeurs x et z de la barre du graphe. Les cubes ont une hauteur de 1 car une valeur supérieur ne correspond pas fidèlement à la hauteur originelle de la barre du graphe. Il faut ensuite leur attribuer le même matériau. Après ces deux étapes, la barre n'est plus active mais l'utilisateur ne le voit pas. Si jamais il clique sur une autre barre, l'ancienne barre redevient active et la nouvelle est remplacée par des cubes et se désactive.

Un bouton a été implémenté pour permettre à l'utilisateur de choisir quand il veut lancer la transition. Un point d'apparition est également implémenté. Quand l'utilisateur appuie sur le bouton, la barre va voler jusqu'au-dessus du point d'apparition avant de commencer la transition. Il est possible ensuite de réappuyer sur le bouton pour que les poubelles, ou autre objets complexes voulu, soit retirés de la scène et que la barre réintègre le graphe avec sa position et sa taille d'origine.

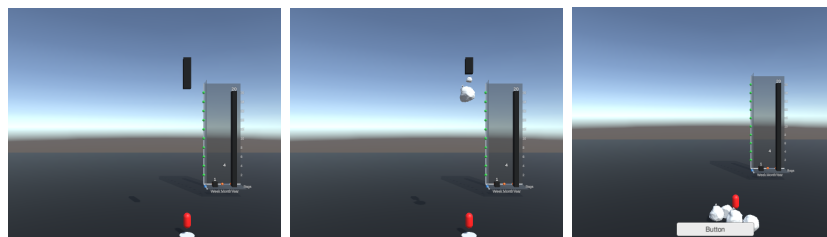


Figure 14 : Transition "SizeReducing" avec la deuxième barre.

Vous trouverez en annexe une vidéo au format gif du nom de *transition_SizeReducing.gif* qui montre la transition.

4.2.4 Smooth

La transition "Smooth" divise la barre sélectionnée en plusieurs parties qui vont chacune se changer en sac poubelle de manière fluide vers l'endroit souhaité. Pour cela, chaque partie se change d'abord en sphère puis rapidement en sac poubelle pour donner cet effet de fluidité. La transition peut s'effectuer sur toutes les parties de la barre à la fois ou bien l'une après l'autre. Le délai entre deux transitions peut être modifié à souhait ainsi que la vitesse de ladite transition. Celle-ci s'effectue également en sens inverse, en passant de sac poubelle à sphère puis cube pour reformer la barre d'origine dans le graphe.

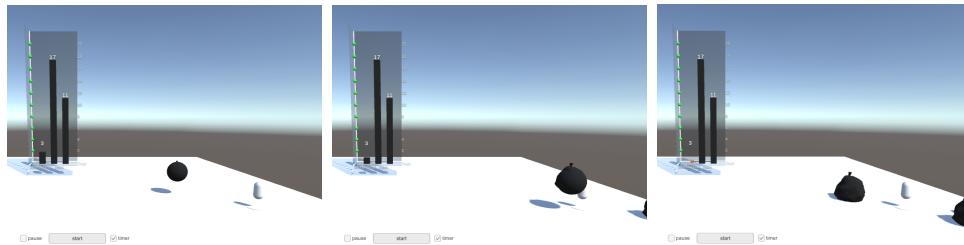


Figure 15 : Transition "Smooth" avec la première barre.

Vous trouverez en annexe une vidéo au format gif du nom de *transition_Smooth.gif* qui montre la transition.

L'idée à l'origine de cette transition était de voir la barre prendre la forme des sacs poubelle qu'elle représente, il fallait donc trouver un moyen simple et rapide pour créer cette transformation. Notre choix s'est donc dirigé vers une transformation en trois étapes : division de la barre en autant de cubes que de sacs poubelles nécessaire, transformation des cubes en sphères et enfin transformation des sphères en sacs. La première étape était plutôt simple à mettre en place, la seule difficulté était de mettre les cubes à la bonne position pour qu'ils prennent la place exacte de la barre. Les deux étapes suivantes se déroulent en réalité en même temps. Les cubes, les sphères et les sacs existent au même moment au même endroit, seulement les sphères et

les sacs sont plus petits que les cubes. Pour passer d'une forme à une autre, les sphères et les sacs vont simplement grossir petit à petit, les sacs plus lentement que les sphères, les faisant ainsi apparaître l'un après l'autre. Cette transformation se passant en l'espace d'une seconde, elle paraît très fluide à l'écran. La transition inverse se déroule de la même manière, cette fois-ci les sphères rétrécissant plus lentement que les sacs.

4.2.5 Launch

La transition "Launch" utilise le même principe que la transition Smooth pour transformer un cube en sac poubelle mais au lieu d'aller en ligne droite les différentes parties sont envoyées vers l'endroit souhaité. La transition peut s'effectuer sur toutes les parties de la barre à la fois ou bien l'une après l'autre. Le délai entre deux transitions peut être modifié à souhait. Celle-ci s'effectue également en sens inverse, où les sacs poubelle sont renvoyés dans le graphe en ligne droite cette fois-ci.

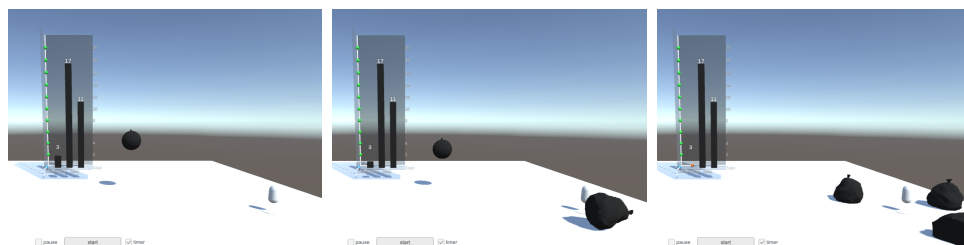


Figure 16 : Transition "Launch" avec la première barre.

Vous trouverez en annexe une vidéo au format gif du nom de *transition_Launch.gif* qui montre la transition.

Pour réaliser cette transition, nous sommes donc partis de la transition "Smooth" et avons simplement changé la trajectoire en ligne droite vers le lieu cible en un lancement de l'objet vers la cible. Cela paraît relativement simple en apparence mais fait apparaître de nouveaux problèmes, parmi lesquels de nouvelles collisions à gérer et le choix de la force appliquée à l'objet pour qu'il se retrouve au bon endroit. Pour ce qui est des collisions, le problème était que l'objet rentrait en contact avec la barre d'où il provenait ou bien avec les autres barres du graphe. La solution a simplement été de désactiver temporairement le collider pour empêcher les collisions et de le réactiver

lorsque l'objet arrive à une certaine de distance du point cible ou bien juste au-dessus du sol pour éviter qu'il ne passe au travers. Pour la force appliquée à l'objet, nous avons décidé de la diriger vers un point à mi-distance de la cible et légèrement au-dessus de la position initiale, ainsi l'objet est projeté légèrement en cloche donnant un effet de lancer.

4.3 Transitions non retenues

4.3.1 FlyingCube

La transition "FlyingCube" divise le pavé en $y*4$ mini-cube, avec y la taille du pavé principal. Le but de cette transition est de découper le pavé simple et de faire voler les mini-cubes jusqu'à la destination avant de les faire se réassembler en objets complexes, ici des sacs poubelles.

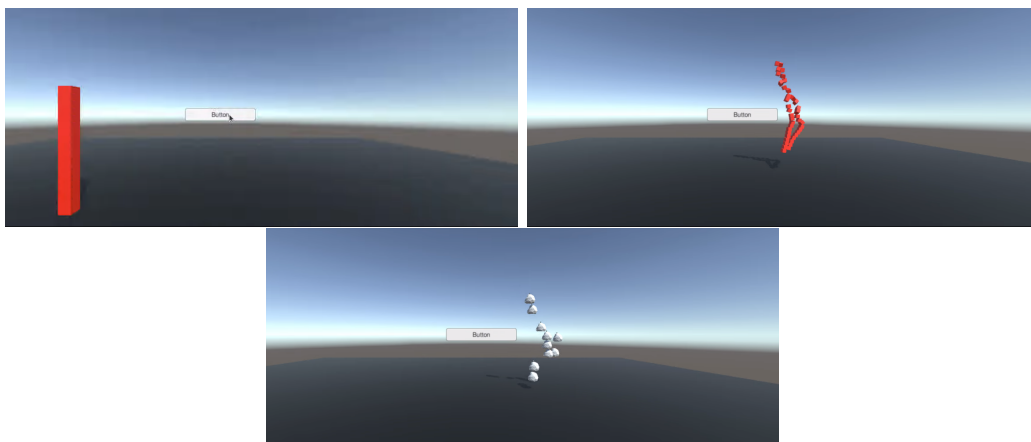


Figure 17 : Transition "FlyinCube" sur un pavé simple.

Une vidéo est disponible dans le dossier Annexe au format .gif avec le nom de *transition_FlyingCube.gif*

Cette transition devait être capable de fonctionner sur plusieurs barre en même temps et chaque barre pouvait avoir un point d'arrivée différent. Nous n'avons pas conservé cette transition dans la version final de l'application car après test et vérification auprès de la cliente, la transformation et l'effet de déplacement ne sont pas assez réalistes.

4.3.2 Fragment

La transition "Fragment" divise la barre sélectionnée en plusieurs parties qui vont chacune se fragmenter en une multitude de petits cubes qui vont reformer les poubelles à l'endroit souhaité. La transition peut s'effectuer sur toutes les parties de la barre à la fois ou bien l'une après l'autre. Le délai entre deux transitions peut être modifié à souhait ainsi que la vitesse de ladite transition. Celle-ci s'effectue également en sens inverse, en renvoyant tous les petits cubes reformer la barre d'origine dans le graphe. Nous n'avons pas inclus cette transition dans l'application dû à des complications liées à son implémentation qui la rendent trop compliquée à lier avec l'interface.

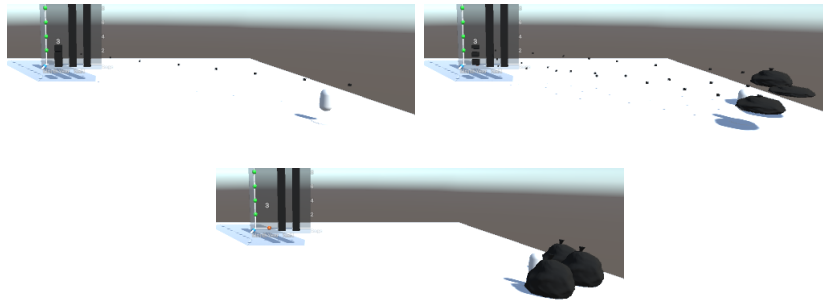


Figure 18 : Transition "Fragment" avec la première barre.

Vous trouverez en annexe une vidéo au format gif du nom de *transition_Fragment.gif* qui montre la transition.

5 Tests unitaires

A cause de l'architecture de notre logiciel, nous ne pouvons pas faire de tests sans avoir d'Assembly cyclique, qui n'est pas autorisé par Unity. Nous allons donc décrire les tests que nous aurions voulu implémenter.

5.1 Tests de Compression

1) Il faut qu'à la fin de la transition, la barre ait atteint la taille de 0. Il faut donc démarrer la transition, et une fois que qu'elle est terminée, comparer sa taille à 0. Si ce n'est pas égal, alors le test est raté.

2) Après la transition inverse, il faut que la barre soit de la même taille que la barre initiale, et à la même position. Il faudrait donc mettre la taille initiale de la barre dans une variable, faire la transition Compression, puis lancer la transition inverse, et comparer la taille de la barre avec celle qu'on a mis dans la variable. Si ce n'est pas égal, alors le test est raté.

3) Le bon nombre de poubelles doit apparaître au moment où la barre atteint la taille de 0. Pour cela, il faut récupérer le nombre de poubelles, démarrer la transition, et comparer le nombre de poubelles sur la scène au moment où la barre atteint la taille de 0. Si le nombre de poubelles est trop grand ou trop petit, alors le test est raté.

4) A la fin de la transition inverse, il faut qu'il n'y ai plus de poubelles actives dans la scène. Il faut donc sélectionner une barre, lancer la transition, puis lancer la transition inverse à la fin de la transition initiale. Une fois la transition inverse finit, on cherche dans la scène si il y a au moins une poubelle active. Si c'est le cas, alors le test est raté.

5) Une fois la transition inverse terminé, le script "Compression" doit être supprimé de la barre. Pour tester ça il faut lancer la transition sur une barre, faire la transition inverse, puis vérifier si le script est toujours présent parmi les composants de la barre. Si c'est le cas, alors le test est raté.

5.2 Tests de Slicer

1) Cette transition a un petit bond initiale sur place, afin d'éviter certains problème. Il faut donc vérifier qu'au moment de la deuxième collision, La position de la barre est la même qu'au départ. Si ce n'est pas le cas, alors le test est raté.

2) La barre se transforme en cubes de la même couleur que la barre lors de sa deuxième collision. Il faut donc vérifier qu'au moment de la deuxième collision, Le nombre de cubes présents sur la scène est égal à la taille de la barre, et qu'il sont tous de la bonne couleur. Si les valeur de ces deux éléments ne sont pas les bonnes, alors le test est raté.

3) Les poubelles doivent être réparti équitablement entre les cubes. Il faut donc vérifier au moment où sont créés les cubes, c'est à dire la deuxième col-

lision, que le plus petit nombre de poubelles contenues dans un cube a une différence au maximum de 1 par rapport au plus gros nombre de poubelles contenues dans un cube. Si la différence est supérieure à 1, alors le test est raté.

4) Quand on lance la transition inverse, la barre doit retourner à sa position initiale. Il faut donc garder sa position de départ dans une variable, et la comparer à sa position une fois la transition inverse terminée. Si c'est n'est pas la même position, alors le test est raté.

5) Quand on lance la transition inverse, les cubes et les poubelles de la scène doivent être supprimés. Il faut donc lancer la transition slicer, puis lancer la transition inverse, et en ensuite chercher dans la scène si il reste des cubes ou des poubelles. Si il reste au moins un exemplaire de ces deux objets dans la scène, alors le test est raté.

6) Une fois la transition inverse terminé, le script "Slicer" doit être supprimé de la barre. Pour tester ça il faut lancer la transition sur une barre, faire la transition inverse, puis vérifier si le script est toujours présent parmi les composants de la barre. Si c'est le cas, alors le test est raté.

5.3 Tests de SizeReducing

1) Au début de la transition, un certain nombre de cubes remplacent la barre du graphe. Il faut tester si le nombre de cube est strictement égal à la valeur y de la barre sélectionnée. Si ce n'est pas le cas le test est raté.

2) Comme la barre du graphe ne bouge pas, il faut la désactiver afin qu'elle ne soit plus visible pour le joueur mais toujours présente dans le jeu pour être ré-activée plus tard. Donc, on doit tester que la barre sélectionnée est bien désactivée et que les autres sont toujours activées, si ce n'est pas le cas, le test est raté.

3) Il faut ensuite tester que chaque cube possède bien 1 objet complexe qui doit apparaître à la fin de la transition. Si cette valeur est différente, le test est raté.

4) Afin d'assurer une meilleur compréhension de cette transition, il faut

que les cubes s'envolent au-dessus du point d'arrivée. Il faut donc tester que les valeurs y des position des cubes soient strictement supérieures à la valeur y de la position du point d'arrivée. Si ce n'est pas le cas, le test est raté.

5) Une fois arrivé au dessus du point d'arrivée, il faut tester que chaque cube ne soit pas à la même position. En théorie, le cube le plus bas est à la position x, y, z et le cube juste au dessus à la position $x, y + 0,5, z$. Si la différence de position entre 2 cubes voisins est différente de $x, y \pm 0,5, z$, alors le test est raté.

6) Afin de donner une impression que la barre s'applatie vers le haut, il faut que les cubes s'applatissent les uns après les autres. Pour ce faire un timer permet de faire chaque cube les uns après les autres. Il faut tester que le timer s'exécute normalement, on doit donc tester que la valeur du timer augmente en permanence et que lorsqu'elle atteint la valeur demandée pour activer la transition d'un cube, elle retourne bien à la valeur 0. Si non, le test est raté.

7) Pour ne pas déranger la chute des objets complexes servant pour le résultat final, les cubes maintenant de hauteur 0, sont téléportés hors de la zone de jeu. Il faut donc vérifier que leurs nouvelles positions sont bien différentes des anciennes. Si ce n'est pas le cas, alors le test est raté.

8) Une fois la transition terminée, on peut en ré-appuyant sur le bouton qui a servi à lancer la transition, revenir à l'état initiale. Pour ce faire, on désactive les objets résultats et les cubes, puis on ré-active la barre du graphe. Si un seul élément n'est pas correctement désactivé ou ré-activé, le test est raté.

5.4 Tests de Smooth & Launch

Partageant la même base de code, les mêmes tests peuvent être conduits pour les deux transitions.

1) Le nombre de parties créées en cliquant sur une barre doit être égal au nombre de sacs que celles-ci représentent. Si ce n'est pas le cas, le test échoue.

2) Chaque partie créée pour une barre doit contenir un cube, une sphère

et un sac poubelle. S'il manque au moins l'un des trois, le test échoue.

3) A la fin de la transition, aucune partie ne doit être à sa position initiale. Si au moins une partie n'a pas bougé, le test échoue.

4) A la fin de la transition, tous les sacs poubelle doivent avoir leur échelle à 1. Si au moins un des sacs n'est pas à la bonne échelle, le test échoue.

5) A la fin de la transition, tous les cubes et sphères ainsi que la barre doivent être désactivés. Si au moins un est encore actif, le test échoue.

6) Pour respecter la physique, le rigidbody et le collider de chaque sac doivent être actifs à la fin de la transition. Si tel n'est pas le cas, le test échoue.

7) A la fin de la transition inverse, toutes les parties doivent être désactivées et la barre activée. Si au moins une partie est encore active ou bien la barre désactivée, le test échoue.

8) A la fin de la transition inverse, toutes les parties doivent être à leur position et leur rotation initiales. Le test échoue dans le cas contraire.

9) Le délai entre deux transitions et leur vitesse (seulement pour Smooth) doivent être strictement supérieurs à 0. Si au moins l'un des deux est égal ou inférieur à 0, le test échoue.

6 Test Utilisateur

Pour faire tester notre application, nous avons proposé aux utilisateurs le scénario suivant :

"Vous allez tester une application ayant pour but de créer un lien visuel entre des données statistiques et leur équivalent dans le monde réel. Vous devrez interagir avec les éléments du graphes et les boutons disponibles pour lancer et observer différentes méthodes de transitions animées. Vous devrez ensuite remplir un formulaire de manière anonyme vous demandant votre avis sur les différentes transitions, l'interface utilisateur et l'application en général." Nous avons pu faire tester notre application sur une vingtaine de personne, voici les résultats que nous avons pu récupérer :

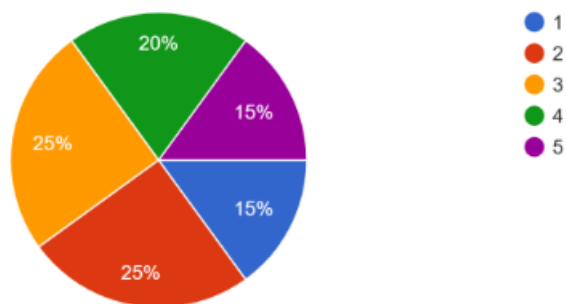


Figure 19 : Résultat à la demande "Sur une échelle de 1 à 5, à quel point avez-vous trouvé attractive l'interface de l'application."

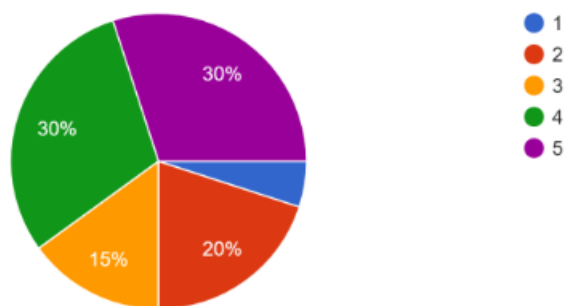


Figure 20 : Résultat à la demande "Sur une échelle de 1 à 5, à quel point l'interface de l'application est facile à prendre en main."

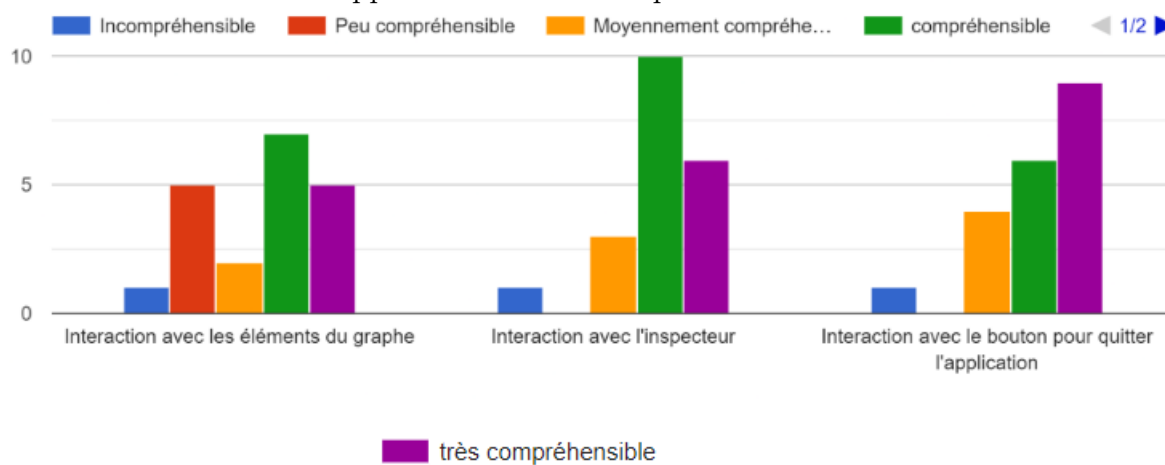


Figure 21 : Résultat à la demande "Veuillez noter la compréhension des

différents éléments de l'interface."

Note sur la Figure 21 : la flèche permettant de voir la suite, ne montre pas plus de donnée statistique, elle ne montre que la légende pour la couleur violette.

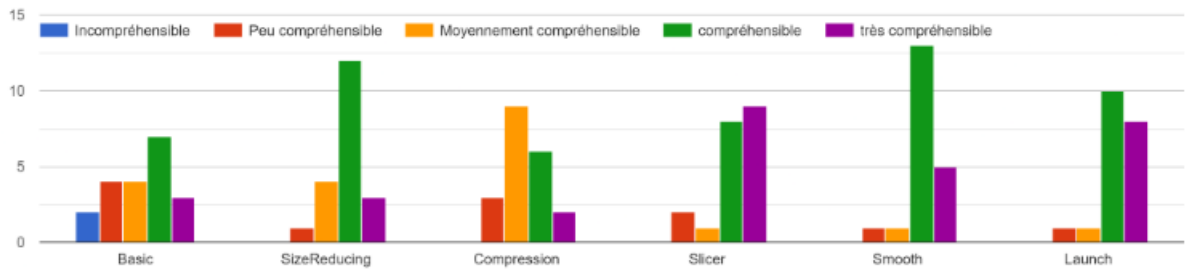


Figure 22 : Résultat à la demande "Veuillez noter à quel point vous avez trouvé chaque transition compréhensible."

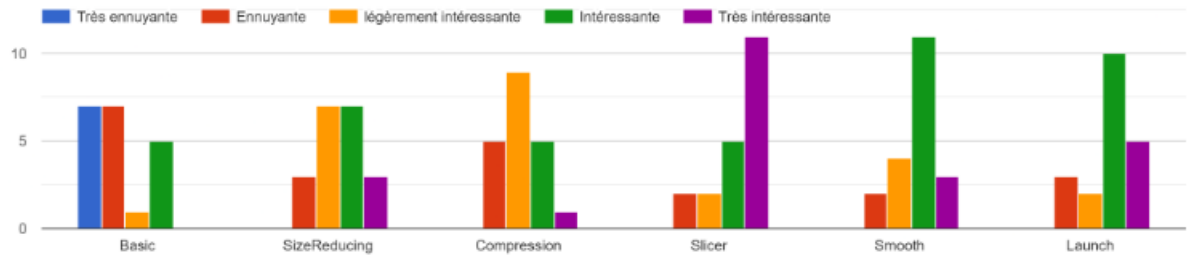


Figure 23 : Résultat à la demande "Veuillez noter l'intérêt ressenti pour chaque transition."

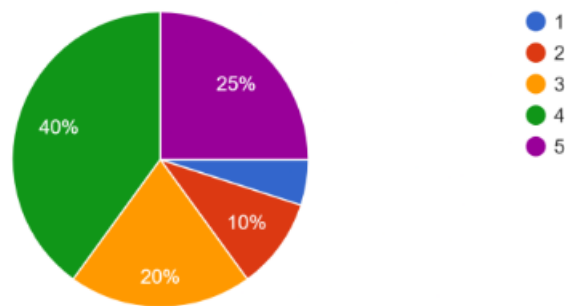


Figure 24 : Résultat à la demande "Sur une échelle de 1 à 5, à quel point trouvez-vous cette application et son but utile."

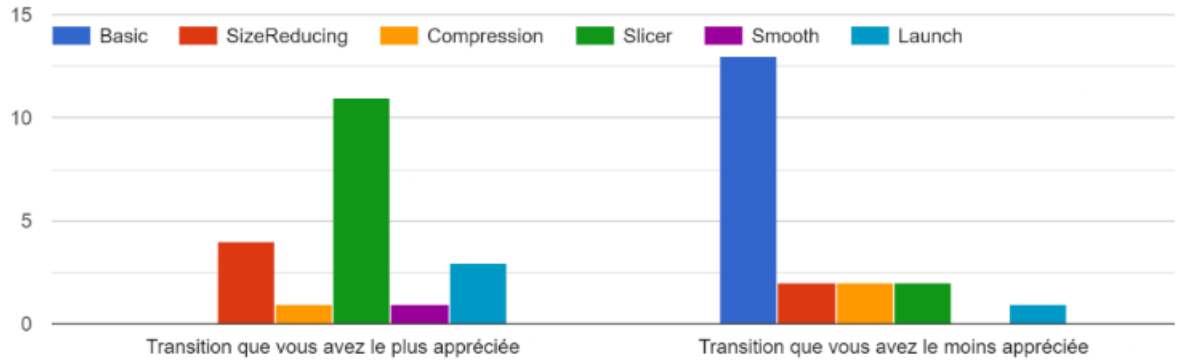


Figure 25 : Résultat à la demande "Quelle a été la transition que vous avez le plus / le moins appréciée."

Ces résultats nous montre comment les utilisateurs ont ressentis l'application et les transitions proposés ainsi que la marche à suivre pour améliorer l'application. Même si l'interface est facile à prendre en main pour les utilisateurs, elle n'est pas considéré comme vraiment attractive pour presque la moitié d'entre eux. Cela nous montre qu'il y a une différence notable entre l'attractivité de l'application et sa prise en main. C'est grâce à ses résultats que nous avons pu modifier l'interface et son attractivité dans la troisième version 4.1.2.3.

On observe également des disparités dans les différentes transitions. Malgré cela, on observe un intérêt plus prononcé pour "Slicer" et moindre pour "Basic". Cela nous montre vers quel type de transition les utilisateurs sont le plus sensible et comprennent le mieux le résultat.

Finalement, dans 65% des cas, les utilisateurs ont plutôt été favorables à l'application et l'ont trouvé utile. Ce résultat est important car il montre l'intérêt que peut avoir cette application dans des conditions réels.

7 Conclusion

7.1 Bilan

Nos transitions fonctionnent correctement, sont fluides et peuvent être choisis depuis le menu. L'interface est plutôt facile à prendre en main d'après les tests utilisateurs, et l'application est jugé utile. De par les différents rendez-vous avec la cliente, il est sensé de conclure que l'objectif principal à

été atteint.

Cependant, il y a divers points qui pourraient être améliorés. Tout d'abord, pour les transitions, il manque l'option des paramètres, tels que la vitesse. Pour les transitions, slicer à des problèmes d'affichage avec les cubes, et va parfois trop loin, et Compression ne s'applatit pas à la base de la barre. Nous aurions voulu faire le passage vers la réalité augmentée mais nous n'avons pas eu le temps. Nous voulions aussi que la transition se fasse vers des objets indépendamment de leur taille, pour que les poches poubelles ne fassent pas forcément la même taille que les barres du graphe. Également, certaines transitions, comme SizeReducing, n'ont pas de méthode pour effectuer la transition dans le sens inverse, ce qui perd en qualité et en confort visuel.

7.2 Perspective

Avec la réalité augmentée, nous aurions pu faire en sorte que les transitions commencent au moment où l'utilisateur relâche une barre qu'il a attrapé plutôt qu'en appuyant sur le bouton. On peut aussi imaginer des poubelles de différentes couleurs en fonction de la barre à laquelle elles appartenaient à l'origine. Il serait aussi intéressant de pouvoir remettre les poubelles dans la barre d'origine "à la main" plutôt qu'avec un bouton.

Nous aurions pu également réfléchir à une technique de transition utilisant un système de mesh morphing avec des interpolations harmoniques, technique de transition plus complexe et pouvant être plus soumise à la complexité algorithmique, mais également la technique la plus fluide et compréhensible pour une transition animée 3D entre objets.

Références

- [1] F.H. Post, G.M. Nielson and G.P. Bonneau *Data Visualization : The State of the Art*, publisher : Springer US, ISBN : 9781461511779
url : <https://books.google.fr/books?id=PZnqBwAAQBAJ>
- [2] W. Horn, C. Popow and L. Unterasinger *Support for Fast Comprehension of ICU Data : Visualization using Metaphor Graphics*, Methods Inf Med 2001 ; 40(05) : 421-424
url : <https://www.thieme-connect.com/products/ejournals/abstract/10.1055/s-0038-1634202>

- [3] Foulball Hangover *Hydroneer*,
url : <http://foulballhangover.com/hydroneer.html>
- [4] Bandai Namco and From Software *Elden Ring* ,
url : <https://en.bandainamcoent.eu/elden-ring/elden-ring>
- [5] T. W.Sederberg and S. R.Parry *Free-form deformation of solid geometric models*, dl.acm.org, abs/15922.15903
url : <https://doi.org/10.1145/15922.15903>
- [6] L. Shi, Y. Yu, N. Bell and W-W. Feng *A fast multigrid algorithm for mesh deformation*, dl.acm.org, abs/1141911.1142001
url : <https://doi.org/10.1145/1141911.1142001>
- [7] M. Ha *unity-meshmorphing*, github.com
url : <https://github.com/minhhh/unity-meshmorphing>
- [8] SzPro *Mesh Morpher*, assetstore.unity.com
url : <https://assetstore.unity.com/packages/tools/modeling/mesh-morpher-159822>