Équipe no 3

Alexis Prud'homme
111186278

Joelle Lagadic
111239882

Kevin Loyer
111182046

Raphaël Paré
111139984

Programmation en animation
ANI-2012

Document de design TP2

Travail présenté à
Philippe Voyer

Faculté d'aménagement, d'architecture, d'art et de design
Université Laval
Automne 2019

Table des matières

Sommaire 2

Interactivité 2

Fonctionnalités 4

1. Scène Maya 4
2. Attribut 4
3. Sélection 5
4. Poses clés 5
5. Génération procédurale 5
6. Scène Unity 6
7. Intégration 7
8. Animation avec une ligne de temps 8
9. Animation avec une machine à états 9
10. Effets visuels 10
11. Autres fonctionnalités 10

Ressources 12

Builds 12
MayaProject 12
UnityProject 13

Présentation 16

Design 16
Animation de formes 16
Création de la scène 16
Intégration et programmation dans *Unity* 17

Sommaire

Dans le cadre du deuxième projet pratique du cours *Programmation en animation*, l'équipe 3 a réalisé un petit jeu vidéo qui se nomme *Vroom* avec l'aide de *Unity* et du langage *C#*. La majorité des objets en trois dimensions qui sont utilisés dans le jeu sont réalisés dans le logiciel *Maya*. Certains de ces objets sont générés par des scripts du langage *Python* qui ont été importés et exécutés dans une scène *Maya*.

Le jeu *Vroom* est un jeu de configuration d'avatar ayant la forme d'une voiture. Il est actuellement composé de deux scènes. La première scène sert à choisir la couleur de la voiture et à afficher celle-ci dans différents angles et différents environnements. Par exemple, la voiture peut tourner sur elle-même dans un mode d'affichage journée ou dans un mode d'affichage nuit. La seconde scène sert à afficher la voiture en mode plein écran. Le joueur peut donc contrôle l'angle d'affichage de la voiture en la faisant pivoter sur elle-même avec un cliquer-glisser (« drag »). De plus, dans le futur, il serait intéressant d'ajouter d'autres scènes au jeu offrant la possibilité aux joueurs de conduire la voiture dans un environnement irréaliste.

Selon les directives du projet, le programme devait contenir des générations procédurales d'objets 3D et inclure différents types d'animation. Les membres de l'équipe ont donc décidé de s'inspirer des projets de *Valléeduhamel*, une agence de design qui travaille en publicité, pour concevoir le jeu. Cette agence utilise un style graphique particulier qui exploite dans tous leurs projets. Dans leurs réalisations vidéos, elle effectue beaucoup d'animations avec des formes géométriques simples et des couleurs vives en aplat. Elle crée souvent des scènes irréalistes, mais très esthétiques, qui intègrent des éléments naturels comme de l'eau.

Interactivité

Le projet *Unity* contient toutes les interactivités possibles avec l'animation. Il est séparé en deux scènes distinctes: la scène principale et la scène plein écran. Ces deux scènes offrent des interactivités différentes avec la voiture.

Scène principale (*Unity*)

La scène principale contient un menu possédant beaucoup d'éléments interactifs. Il y a entre autres un groupe de boutons radio, qui sert à changer la couleur couleur du véhicule,

et une liste déroulante, qui permet de changer le mode d'affichage de la voiture qui possède trois modes : mode fixe, mode rotation, mode décomposé. Le menu permet également de changer le mode d'affichage du jeu en mode jour ou en mode nuit à l'aide du bouton ayant une icône de soleil ou de lune. Un autre bouton, qui est aussi intégré au menu, permet de naviguer à la scène plein écran. Les boutons annuler et enregistrer sont intégrés dans le coin inférieur droit de l'écran. Le bouton annuler permet de réinitialiser la dernière couleur de voiture enregistrée, d'appliquer le mode d'affichage fixe à la voiture, de repositionner la voiture à sa position initiale et définir le mode d'affichage en mode jour. Le bouton enregistrer permet de sauvegarder dans les préférences utilisateurs la couleur courante de la voiture.

D'ailleurs, la scène contient plusieurs formes animées flottantes. Au clic sur l'une de ces formes, l'animation de celle-ci est mise sur pause. Au second clic sur la même forme, l'animation se réactive.

De plus, il est possible d'effectuer un clic sur la voiture. Ce clic applique une force sur le composant « RigidBody » de la voiture ayant comme impact de faire flotter la voiture dans les airs.

Scène plein écran (*Unity*)

L'interactivité de la scène plein écran est moins riche en interactivité que celle de la scène principale. Le menu offre moins de contrôles. Certains éléments n'ont pas été ajoutés à la scène tels que le podium et les formes animées. Les joueurs ne sont donc plus distraits par l'environnement autour du véhicule.

D'autre part, le menu de la scène plein écran est composé de seulement deux boutons. Le premier bouton sert à retourner à la scène précédente, la scène principale, et le deuxième bouton sert à positionner le véhicule à sa position initiale lorsqu'il a été déplacé par les joueurs.

De plus, les joueurs ont la possibilité de faire pivoter dans tous les angles la voiture avec les flèches du haut, du bas, de gauche et de droite du pavé numérique. La rotation 360 degrés de la voiture peut également être effectués avec l'action cliquer-glisser (« drag »).

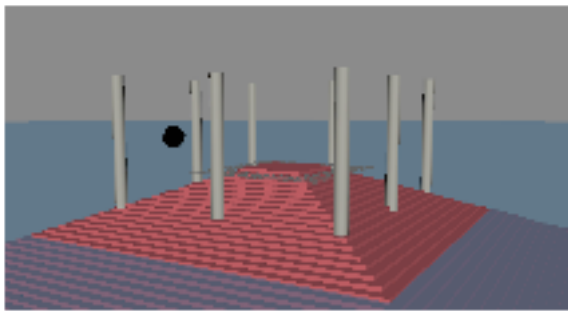
Fonctionnalités

1. Scène *Maya*

Dans le projet Unity, cinq scènes ont été créées à partir du logiciel Maya : « CubesRotation », « Diamonds », « DiamondsLarge », « RectanglesOnSquare » et « MayaScene ». L'entièreté de ces scènes a été générée par des scripts *Python*.

Les quatre premières scènes (« CubesRotation », « Diamonds », « DiamondsLarge » et « RectanglesOnSquare » sont un amalgame de forme primitive comprenant des fonctions d'interaction diverse. Ces scripts ont été intégrés au projet *Unity* de façon à ce qu'ils gravitent autour du podium pour créer un univers interactif intéressant.

La scène « MayaScene » présente quant à elle des éléments statiques qui n'impliquent aucune animation. Le premier script est celui que l'on retrouve dans la scène principale de *Unity*, le podium. Il implique la conception d'une pyramide flexible dont la hauteur varie selon le nombre d'étages souhaité. Le deuxième script est le script « DrawColomns », qui implique plusieurs fonctions pour générer des colonnes autour du podium. Par Ailleurs, ce dernier script n'a pas été intégré au projet Unity, puisqu'il devenait trop encombrant autour de l'élément central et il ne mettait pas en valeur la voiture.



2. Attribut

L'utilisation de la manipulation d'attributs a été employée à plusieurs reprises dans les scènes *Maya* pour altérer l'apparence de formes géométriques. Par exemple, dans le fichier Python « CubeRotation », une affectation d'attributs a été utilisée pour appeler les attributs « rotateX » et « rotateY ». En utilisant des valeurs aléatoires dans la configuration des attributs, un effet de vibrance et de profondeurs a été généré une fois que la lecture des attributs est effectuée. Grâce à la manipulation, l'affectation et la lecture d'attribut,

plusieurs scènes, qui s'affichent en harmonie avec l'esthétique générale du projet, ont été créés.

3. Sélection

Dans *Python*, la sélection par programmation est très utile pour appliquer des modifications sur des éléments. La sélection permet au programmeur de créer des changements à un grand ensemble d'éléments rapidement et efficacement avec quelques lignes de code. Cette option est employée dans la scène *Maya* « Cube Rotation » où plusieurs « polyCubes » sont instanciés. Les attributs de ces instances ont tous été modifiés grâce à une sélection sur une liste des noms de cubes. Une fois les cubes sélectionnés, des modifications à ceux-ci ont été appliquées comme l'ajout de poses clés. En tout, trois scripts *Python* utilisent la sélection soit les fichiers « CubesRotation.py », « DrawColumns » et « DrawPodium.py ».

4. Poses clés

L'utilisation de poses clés était cruciale dans la création de nos scènes *Maya* où l'on retrouve des formes géométriques. L'animation de ces formes dépendait donc de la manipulation d'attributs (« rotate » et « translate »), mais aussi de poses clés, car chaque changement s'exécute dans le temps.

Deux classes ont été crées pour obtenir une meilleure sémantique et modularité de code. La première classe « Keyframe » a été créée pour définir les propriétés d'une pose clé (« time », « attribute » et « value »). Tandis que la deuxième class « Timeline » permet d'ajouter des poses clés en les sauvegardant dans une liste, de définir le début et la fin de la ligne du temps et de construire une ligne du temps à l'aide la liste de pose clé. Ces deux classes facilitent la manipulation de poses clés sur une ligne du temps en permettant d'attribuer plusieurs attributs de transformation sur un seul « timeframe ».

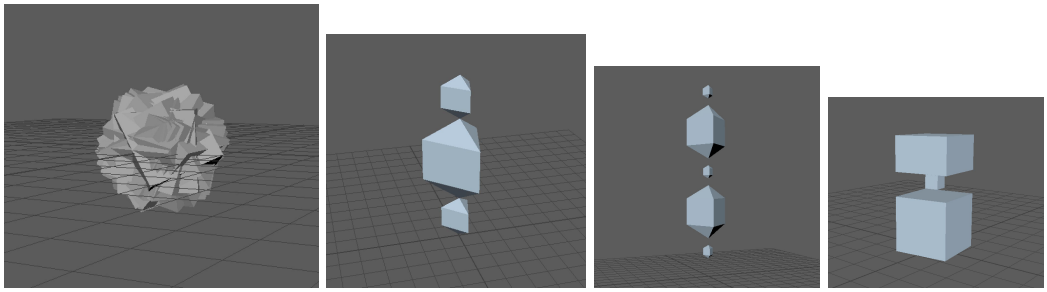
Dans le projet, quatre fichiers *Maya* utilisent des scripts *Pythons* qui manipulent des poses clés. Ces fichiers se trouvent dans ce répertoire:

« ANI2012A19_TP2_E03/MayaProject/AnimatedShapes ».

5. Génération procédurale

Les quatre scènes *Maya* (« CubesRotation », « Diamonds », « DiamondsLarge » et « RectanglesOnSquare ») utilisant les poses clés ont généré des formes primitives

géométriques par programmation. Pour créer ces formes, les « polyCube » et les « polySpheres » ont été utilisés.



D'autre part, les colonnes et le podium de la scène « MayaScene » sont également générés par programmation. La génération de ces formes est d'une plus grande complexité que celles des formes animées. C'est pourquoi deux classes ont été créées pour faciliter la génération de ces éléments.

La première classe « Columns » reçoit en paramètre dans son constructeur le nom des colonnes, le nombre de colonnes en x, le nombre de colonnes en y, le diamètre de la colonne, la hauteur de la colonne et la distance entre deux colonnes. Elle possède la méthode « draw() » qui dessine des « polyCylinder » à l'aide de deux boucles « for », la méthode « center_in_scene() », qui sélectionne les colonnes et les centre dans la scène, la méthode « move() » qui permet de déplacer les colonnes dans la scène et la méthode « delete() » qui supprime toutes les instances des colonnes à l'aide d'une sélection. Ces colonnes n'ont malheureusement pas été utilisées dans le projet Maya, car une fois intégré dans la scène *Unity*, les colonnes volaient la vedette à la voiture en surchargeant l'interface d'informations. Les colonnes ont donc été retirées du projet *Unity*.

La deuxième classe « Podium » reçoit en paramètre dans son constructeur le nom du podium, le nombre d'étages, la largeur du podium et la hauteur de podium. Elle possède la méthode « draw() » qui dessine des « polyCubes » à l'aide d'une boucle « for » et la méthode « delete() » qui supprime toutes les instances du podium à l'aide d'une sélection.

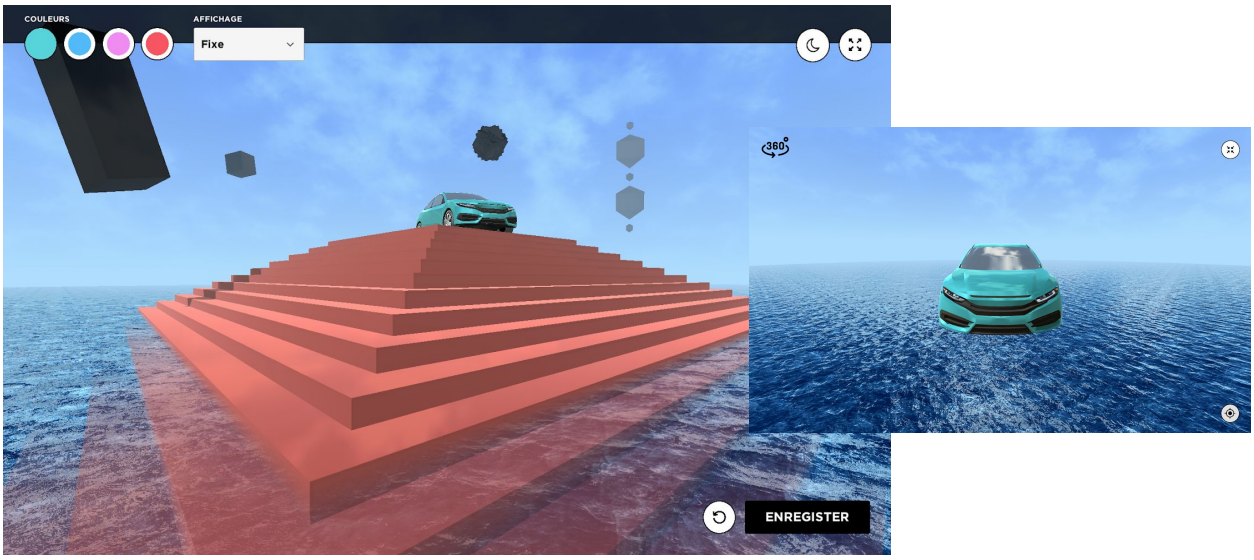
6. Scène *Unity*

Dans ce projet *Unity*, deux scènes ont été créées : la scène principale (« MainScene ») et la scène plein écran (« FullScreen »). Ces deux scènes contiennent des ressources communes telles que les « Prefabs » « Car », « IconButton » et « Atmosphere ».

La scène principale contient une caméra principale, une lumière directionnelle, un ciel, de l'eau, une voiture, des feux d'artifice, cinq formes flottantes animées, un podium

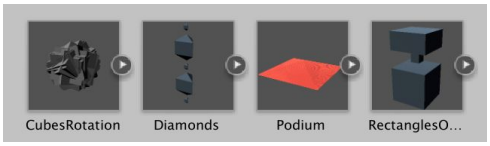
et un menu (UI) contenant une liste déroulante et des boutons. Le bouton plein écran, positionné sur le côté supérieur droit de l'écran, permet de naviguer vers la scène plein écran.

Par ailleurs, presque tous les éléments interactifs ont été retirés de la scène plein écran pour permettre d'attirer l'attention des joueurs sur la voiture. La scène principale contient une caméra principale, une lumière directionnelle, un ciel, de l'eau, une voiture et un menu contenant seulement deux boutons.

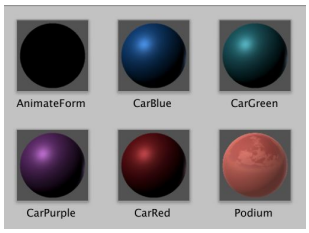


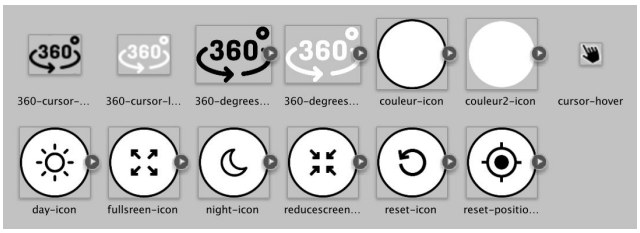
7. Intégration

Pour commencer, quatre éléments visuels produits par les membres de l'équipe dans *Maya* au format .fbx ont été importé dans le projet *Unity*. Ces éléments de type primitive géométrique sont contenus dans les fichiers « CubesRotation.fbx », « Diamonds.fbx », « Podium.fbx » et « RectanglesOnSquare.fbx ».

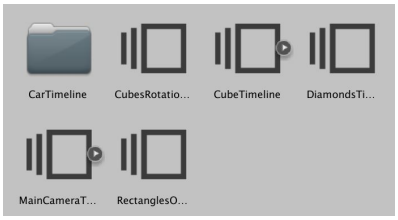


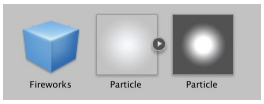
D'autre part, six matériaux ont été créés dans *Unity*. Ces matériaux sont appliqués sur le podium, les formes animées et sur la voiture. 13 images ont également été importées au projet *Unity*. Ces images sont des icônes utilisées pour construire le menu du jeu.





Par ailleurs, un système de particule ayant l'apparence de feux d'artifice et huit lignes du temps sont utilisés dans le jeu.





,

Pour terminer, deux « assets » du *Unity Asset Store* sont importés au projet. Le modèle 3D de la voiture provient de « Low poly car » et tout l'environnement incluant l'eau et le ciel est issu de « NVJOB Water Shader - simple and fast ».

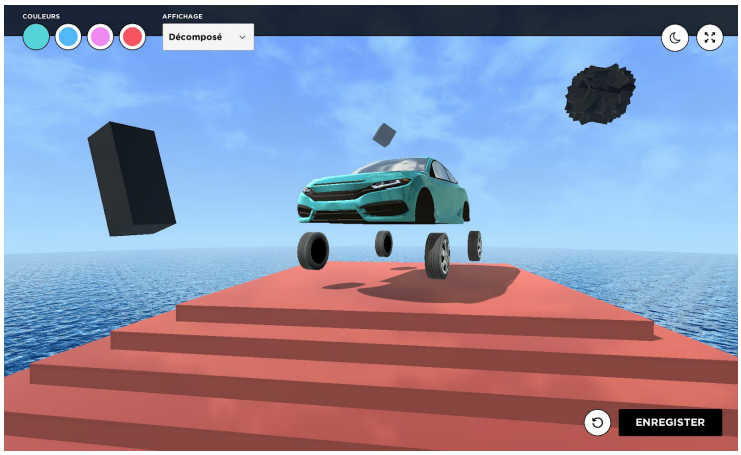
8. Animation avec une ligne de temps

Plusieurs éléments, dans la scène principale, utilisent des lignes du temps de *Unity*.

Pour débuter, la caméra de la scène principale possède une ligne du temps effectuant des animations sur les attributs de position et de rotation. À la première lecture de la scène, l'animation de la caméra est jouée une seule foi et elle ne sera plus jamais joué dans le cycle de vie du programme. Une variable « static » dans un script lié à la caméra est donc créée pour faire perdurer l'état de l'animation à travers les scènes *Unity*.

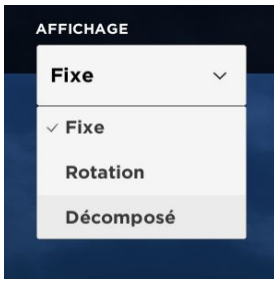
Par ailleurs, toutes les formes flottantes, qui ont été animées dans *Maya* avec des scripts *Python,* emploient une ligne du temps de *Unity*. Leur animation sur la ligne du temps est jouée en boucle. Lorsque nous cliquons sur une de ces formes, la tête de lecture est mise sur pause. Quand nous cliquons une seconde fois sur celle-ci, la ligne du temps est remise en lecture. Toutes ces formes provenant de Maya n'utilisent donc pas les propriétés d'animation de *Unity*. C'est pourquoi un des cubes en rotation a été entièrement créé dans *Unity*. Ce cube est animé sur les attributs « Rotation.x », « Rotation.y », « Rotation.z » et « Position.y ». Le « Prefab » qui se nomme « CubeRotation » a été créé pour ce cube.

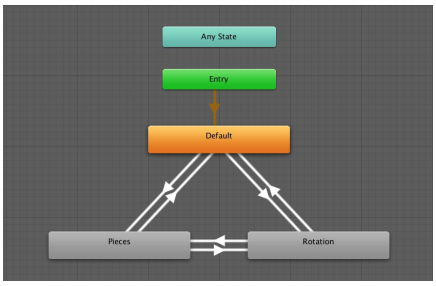
De plus, toutes les animations en lien avec la voiture ont été réalisées avec des lignes du temps de *Unity*. La voiture possède trois lignes du temps. La première n'effectue aucune animation. La seconde, fait pivot sur l'axe z la voiture en boucle. La dernière, anime le corps (« body ») de la voiture avec l'attribut « Position.y » et ses roues avec l'attribut « Position.x » et « Position.z ». Toutes les lignes du temps en lien avec la voiture sont gérées par une machine d'état. Les animations de la voiture sont déclenchées par la liste déroulante de la scène principale.



9. Animation avec une machine à états

Une machine d'état a été créée dans ce projet pour animer les différents modes d'affichage d'une voiture: mode fixe, mode rotation et mode décomposé. En état fixe, l'automobile ne se déplace pas, en état décomposé, elle exécute une rotation sur l'axe z en boucle, en état décomposé, le corps («body») de l'auto s'élève dans les airs tandis que les quatre roues s'éloignent du corps. Pour gérer ces modes d'affichage, le composant « Animator » de *Unity* est utilisé. Il permet de manipuler facilement la transition entre les différents états. La liste déroulante de la scène principale permet de lancer une des lignes de temps de la voiture selon le mode d'affichage sélectionné.



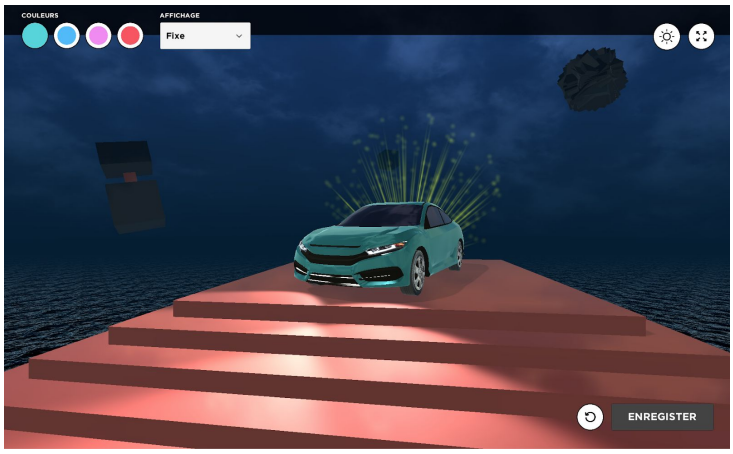


10. Effets visuels

Le projet contient des effets visuels tels qu'un système de particule (feux d'artifice) et des effets de lumière (lumière de l'automobile et lumière directionnelle d'ambiance).

Pour commencer, le système de particule est activé lorsque le joueur clique sur le bouton enregistrer. Des particules sous forme de feux d'artifice sont donc lancées à l'arrière de la voiture. Cet effet a été ajouté pour offrir une rétroaction visuelle positive aux usagers afin leur faire comprendre que l'enregistrement a bien été effectué. Ces particules ont été paramétrées à l'aide des outils offerts dans l'interface de *Unity*.

D'autre part, la lumière directionnelle d'ambiance est utilisée dans le « Prefab » « Atmosphère ». Ce « Prefab » contient également, le ciel et l'eau. L'intensité de la lumière directionnelle est modifiée lorsqu'un clic est appliqué sur le bouton de mode d'affichage jour et nuit. Quand le mode jour est activé, l'intensité de la lumière est égale à un et en mode nuit, l'intensité est à zéro. La texture du ciel et de l'eau et certains paramètres de la fenêtre « Lighting Settings » (ex.: FrogColor) sont également modifiés par programmation selon le mode jour et nuit.



11. Autres fonctionnalités

Dans ce projet, un certain nombre d'éléments d'interface (UI) offerte par *Unity* sont utilisés tels que les boutons et la liste déroulante. L'apparence visuelle (couleurs, dimension, effet de survol, etc.) de ces éléments a toute été modifiée pour s'adapter à la direction artistique de notre jeu. Par contre, certains éléments d'interface ne sont pas offerts par *Unity* comme les boutons radio. Nous avons donc créé nos propres groupes de boutons radio qui sont utilisés pour sélectionner les couleurs de l'automobile. Pour concevoir ces boutons

radio, un « Prefab » a été généré, contenant des images « sprites » et un script. Ainsi, cette fonctionnalité est facilement réutilisable dans d'autres contextes.

De plus, divers effets de survol ont été ajoutés sur les éléments interactifs pour indiquer aux joueurs les interactions possibles avec l'interface. Lorsque le curseur de la souris passe au-dessus des boutons, de la liste déroulante ou des formes animées interactives, le curseur change pour afficher une image « sprite » de main. Lorsque nous naviguons vers le mode plein écran (scène « FullScreen »), le survol de l'automobile affiche une icône 360 degrés qui signale aux usagers qu'ils peuvent cliquer-glisser (« drager ») la voiture pour la faire pivoter sur elle même. Dans la même scène, quand le mode nuit est activé, l'icône de 360 degrés change de couleur pour le blanc pour assurer un bon niveau de contraste entre le curseur et l'arrière-plan.

D'autre part, plusieurs options de sauvegarde ont été implémentées. À chaque fois, que l'utilisateur clique sur le bouton enregistrer, la couleur courante de l'auto est sauvegardé dans « PlayerPref s» de *Unity*, c'est-à-dire lorsque nous arrêtons le programme, les données seront stockées. L'autre option de sauvegarde est temporaire. Elle agit quand nous naviguons d'une scène à l'autre. Avec cette méthode, les valeurs des variables de la couleur de l'auto et du mode d'affichage nuit et jour sont conservées grâce à l'emploi des propriétés « static ». Cependant, quand le programme s'arrête, les données ne sont pas sauvegardées.

Ressources

À la racine du répertoire du projet, il y a trois dossiers. Le premier dossier « Builds » contient les versions compilées *Mac* et la *Windows*. Le deuxième dossier « MayaProject » contient un dossier « AnimatedShapes » et un dossier « MayaScene ». Le dossier « AnimatedShapes » contient un fichier *Python*, une scène *Maya* et une exportation .fbx pour chacune des formes animées. Le même principe est appliqué pour le dossier « MayaScene ».
Le troisième dossier « UnityProject » à la racine du répertoire du projet contient le projet *Unity*. À la racine du dossier *Unity* « Assets », il y a de dossier « Ressources », qui contient toutes les objets 3D, les lignes du temps, les « Prefabs », les fonts, les matériaux, les textures, les images et les « assets » du *Unity Asset Store*, et le dossier « Scripts », qui contient les fichiers *C#*.

Builds

- Mac
    - Vroom
- Windows
    - Vroom.exe

MayaProject

- AnimatedShapes
    - CubesRotation
        - CubesRotation.fbx
        - CubesRotation.mb
        - CubesRotation.py
    - Diamonds
        - Diamonds.fbx
        - Diamonds.mb
        - Diamonds.py
    - DiamondsLarge
        - DiamondsLarge.fbx
        - DiamondsLarge.mb
        - DiamondsLarge.py
    - RectangleOnSquare

            - RectangleOnSquare.fbx
            - RectangleOnSquare.mb
            - RectangleOnSquare.py
- MayaScene
    - DrawColumns.py
    - DrawPodium.py
    - MayaScene.mb

UnityProject

- Assets
    - Ressources
        - Fonts
            - Gotham-Bold.otf
        - Materials
            - AnimateForm,mat
            - CarBlue.mat
            - CarGreen.mat
            - CarPurpules.mat
            - CarRed.mat
            - Podium.mat
        - MayaFBX
            - CubesRotation.fbx
            - Diamonds.fbx
            - RectangleOnSquare.fbx
            - Podium.fbx
        - Particles
            - Fireworks.prefab
            - Particle.mat[1]
            - Particle.png[2]
        - Prefabs
            - Atmospere.prefab

[1] Provient des exemples du cours

[2] Provient des exemples du cours

        - Car.prefab
        - ColorButton.prefab
        - CubeRotation.prefab
        - CubesRotation.prefab
        - Diamonds.prefab
        - IconButton.prefab
        - RectanglesOnSquare.prefab
        - SceneLoader.prefab
    - Scenes
        - FullScreen.unity
        - MainScene.unity
    - Sprites
        - 360-cursor-dark.png[3]
        - 360-cursor-light.png[4]
        - 360-degrees-dark.png[5]
        - 360-degrees-light.png[6]
        - couleur-icon.png
        - couleur2-icon.png
        - cursor-hover.png[7]
        - day-icon.png[8]
        - fullscreen-icon.png[9]
        - night-icon.png[10]
        - reducescreen-icon.png[11]
        - reset-icon.png[12]
        - reset-position-icon.png
    - Timelines

[3] https://www.flaticon.com/free-icon/360-degrees_1758455?term=360%20degree&page=1&position=3
[4] https://www.flaticon.com/free-icon/360-degrees_1758455?term=360%20degree&page=1&position=3
[5] https://www.flaticon.com/free-icon/360-degrees_1758455?term=360%20degree&page=1&position=3
[6] https://www.flaticon.com/free-icon/360-degrees_1758455?term=360%20degree&page=1&position=3
[7] https://www.flaticon.com/free-icon/clicker_99162?term=hand&page=1&position=4
[8] https://www.flaticon.com/free-icon/sun_606795?term=sun&page=1&position=2
[9] https://www.flaticon.com/free-icon/expad-arrows_81267?term=full%20screen&page=1&position=18
[10] https://www.flaticon.com/free-icon/moon-phase-outline_25665?term=moon&page=1&position=1
[11] https://www.flaticon.com/free-icon/expad-arrows_81267?term=full%20screen&page=1&position=18
[12] https://www.flaticon.com/free-icon/undo-arrow_82004?term=undo&page=1&position=19

            - CarTimeline
                - CarAnimatorController.controller
                - CarDefaultTimeline.playable
                - CarPiecesTimeline.playable
                - CarRotationTimeline.playable
            - CubesRotationTimeline.playable
            - CubeTimeline.playable
            - DiamondsTimeline.playable
            - MainCameraTimeline.playable
            - RectanglesOnSquareTimeline.playable
        - UnityAssetStore
            - low poly car[13]
            - #NVJOB Water Shader - simple and fast[14]
    - Scripts
        - AnimatedShapeTimeline.cs
        - Atmosphere.cs
        - Car.cs
        - CarDisplayDropdown.cs
        - ColorButton.cs
        - ColorButtonGroup.cs
        - CursorHover.cs
        - DayNightButton.cs
        - Fireworks.cs
        - IconeSceneFullScreen.cs
        - MainCameraTimeline.cs
        - ResetButton.cs
        - SceneLoader.cs

[13] https://assetstore.unity.com/packages/3d/vehicles/low-poly-car-149312
[14] https://assetstore.unity.com/packages/vfx/shaders/nvjob-water-shader-simple-and-fast-149916

Présentation

Tous les membres de cette équipe étudient en Design graphique. C'était donc la première fois qu'ils ouvraient un logiciel de 3D. La courbe d'apprentissage en début de projet a donc été très ardue, car ils devaient apprendre deux nouveaux logiciels (*Maya* et *Unity*) en plus de découvrir deux nouveaux langages de programmation (*Python* et *C#*). Ils ont dû faire plusieurs heures de formation *Maya* sur *YouTube* pour comprendre comment naviguer dans une scène *Maya*, comment ajouter des formes et comment manipuler les textures. Tous ces éléments semblent simples, mais peuvent s'avérer ardus pour des débutants.

Pour ce deuxième travail pratique avec les mêmes membres de l'équipe, les tâches se sont réparties plus facilement qu'au dernier projet. Alexis Prud'homme, Joëlle Lagadic, Kevin Loyer se sont concentrés plus sur la partie *Maya* et *Python* tandis que Raphaël Paré a travaillé majoritairement sur *Unity* avec *C#*.

Design

Tous les membres de l'équipe ont participé activement au design des interfaces graphiques. Nous avons donc tous effectué une recherche de stylistique chacun de notre côté.
Par la suite, nous avons mis en commun nos idées et nous avons décidé de nous inspirer des ambiances graphiques de *Valléeduhamel*, un groupe de designers qui travaillent en publicité.

Animation de formes

Toutes les formes animées qui flottent dans la scène principale ont été réalisées par Joëlle Lagadic. Dans le logiciel *Maya*, elle a généré les formes et a créé des poses clés sur des lignes du temps par programmation.

Création de la scène

La création de la scène *Maya* qui inclut la génération procédurale du podium et des colonnes, qui ont malheureusement été retirées du projet *Unity* pour ne pas surcharger l'interface, ont été exécutés par Alexis Prud'homme et Kevin Loyer. Ils se sont également occupés de trouver les modèles 3D qui sont utilisés dans *Unity* provenant de *Unity Asset Store*.

Intégration et programmation dans *Unity*

L'intégration des modèles 3D et leur modification pour s'adapter à ce projet ont été effectuées par Raphaël Paré. Il s'est occupé de créer les deux scènes, d'animer la voiture sur une ligne du temps, de créer l'interface utilisateur (UI) et de programmer l'interactivité.