Université Lumière Lyon 2

Rapport Programmation Créative

Thème choisi : Le Petit Prince

DENNEMONT Pierrick – ISAZADE Sabuhi
Année universitaire 2025 - 2026

Table des matières

Présentation

Il s'agit d'un projet universitaire dans le but d'utiliser la librairie p5.js de JavaScript. Dans le cadre de notre travail, nous avons décidé de nous appuyer sur le livre "Le Petit Prince", écrit par Antoine de Saint-Exupery. Cette œuvre, ayant marqué bon nombre d'individus, grâce à sa narration permettant une compréhension distinguée en fonction de la personne, était selon nous une source de contenu non négligeable. En effet, bon nombre des scènes de cet ouvrage sont notables, en espérant que celles-ci vous seront reconnaissables à travers notre travail.
Afin de mettre en avant l'expérience utilisateur, nous avons décidé d'illustrer le récit de l'ouvrage sous forme de livre intéractif. En effet, en laissant la possibilité à l'utilisateur de changer de scène de livre grâce à des boutons disposés à gauche et à droite du canva, on retrouve on se rapproche d'une vraie sensation de lecture. Pour renforcer l'immersion, on a décidé d'intégrer des scènes emblématiques du livre en tant que fond de scène, d'une part pour l'esthétique mais aussi pour rajouter des animations avec le premier plan (les boutons permettant de gérer la page du livre) et le fond. Enfin, pour rajouter de la clarté, un texte placé en haut du canva apportera plus de détails sur la scène. L'image ci-dessous permet d'illustrer nos propos :

Nous avons retenu trois scènes du livre dans notre projet, en fonction de la réalisabilité de leur esthétique et de leur message :

- Scène 1 : La Rencontre
- Scène 2 : Le Voyage (ou Les Planètes)
- Scène 3 : Le Secret (ou Le Renard)

Il est à noter que des scènes supplémentaires ont été implémentées, correspondant respectivement à la couverture du livre, comme le montre l'illustration ci-dessus, ainsi que le dos du livre.
Maintenant que le thème de notre projet a été décrit, détaillons plus en détail les fonctionnalités et les problèmes rencontrés.

Explications

Dans cette section, nous détaillerons davantage les interactions disponibles dans notre sketch, et cela à travers les différentes scènes de notre projet. Pour rappel, on entend par “scène” une page du livre correspondant à une scène du livre “Le Petit Prince”. On expliquera dans un premier temps les animations dites “statiques”, c’est-à-dire celles nécessitant aucune intervention de la part de l’utilisateur, et dans un autre temps celles déclenchées au clic de la souris.

Animations statiques

L’ensemble de nos animations statiques concernent nos fonds d’écran (“background”). De plus, toutes ont été utilisées dans plusieurs scènes de notre projet, ce qui démontre aussi la volonté de produire un code certes pour un projet créatif, mais qui reste factoriser et réutilisable.
Tout d’abord, on peut citer celles concernant l’animation des météorites pour les scènes 1 et 2. En effet, toutes deux se déroulent dans l’espace, ce qui renforce la pertinence de conserver le même fond. Disponible dans le fichier “draw_meteorites.js”, le code permet la génération de plusieurs de celles-ci avec un comportement aléatoire. En effet, la position, la vitesse, ou encore la durée de vie de la météorite est déterminée aléatoirement, permettant un effet plus réaliste à l’écran.
Il existe également dans les scènes correspondant à la page de couverture et du dos du livre des animations statiques. Concernant le dos du livre, on retrouve l’animation du texte, jouant sur l’opacité des lettres, permettant d’avoir un rendu progressif. Cela donne l’impression que les lettres apparaissent de façon indépendante et dissociée des autres.
Pour la gestion des transitions de façon générale, on a utilisé les fonctions mathématiques “sinus”, très utiles grâce à son intervalle continue [-1,1], l’usage de cette fonction nous a permis tout au long des différentes animations d’avoir un rendu plus fluide, évitant ainsi un effet de clignotement, grâce à sa plage de valeur très restreinte, permettant un changement de valeur léger. Reprenons les lettres par exemple de la scène 4 (dos du livre). Afin d’avoir un rendu progressif de l’opacité, on a rajouté progressivement une valeur calculée par le sinus.
De plus, cet effet de transition fluide se retrouve dans le fond des scènes 1 et 4 (couverture et dos du livre) lors de la création des étoiles dans le fond. En effet, le scintillement et la taille des étoiles ont aussi été gérés de cette façon (cf fichier draw_ciel_etoile.js).

Animations déclenchées par l’utilisateur

Ce type d’animation est le plus intéressant du point de vue de l’utilisateur, grâce à la possibilité de pouvoir interagir avec.
La première développée concerne la gestion des pages du livre, grâce aux boutons situés aux extrémités du canva. En effet, grâce au clic sur ceux-ci, la page courante du livre change, afin de laisser place à une nouvelle scène. Concernant les détails techniques, l’ensemble du code concernant l’affichage du livre est disponible dans le fichier “class_livre.js”. Il regroupe les animations concernant le livre, l’utilisation d’une image pour la couverture, le dessin des pages, etc. Il suffit ensuite dans le script principal d’appeler la fonction adéquate et de changer le numéro de la page courante, grâce à l’instance de la classe Livre.

Scène 1

Pour la première scène, celle sur l'astéroïde, nous avons implémenté une animation sur l'avion. Celle-ci est décomposée en deux temps. Tout d'abord, l'avion oscille de haut en bas, pour simuler un déplacement. Enfin, l'hélice s'incline pour simuler l'avion qui vole.

Scène 2

Beaucoup d'animations ont été faites sur cette scène, notamment concernant les 6 monsieurs. Néanmoins, nous avons tout de même décidé d'en ajouter une autre, pour plus d'interactivité.
C'est ainsi que nous avons implémenté l'interaction de drag-drop des astéroïdes. La détection repose sur la fonction scene2SelectionnerAsteroide, qui permet de déterminer si l'on clique sur un astéroïde. Une fois sélectionné, son déplacement est géré en temps réel par scene2DeplacerAsteroide, en conservant le décalage initial et en contraignant la position aux limites de la fenêtre L'implémentation d'un système de drag and drop nous semblait plus original, surtout dans cette scène où il nous semblait plus compliqué de trouver des éléments sur lesquels interagir.

Scène 3

Pour la scène dans le désert, nous avons implémenté trois animations. Celles-ci se déclenchent au clic sur l'avion écrasé, le renard et le serpent. Détaillons-les brièvement :

- Au clic sur l'avion, de la fumée sort de l'avion pour simuler le crash
- Au clic sur le serpent, il bouge et sa langue sort et revient
- Au clic sur le renard, celui-ci se met à remuer la queue, et, à la fin de cette animation, une autre se déclenche automatiquement pour permettre au renard de sautiller de joie.

Problèmes rencontrés

On a essayé du mieux que possible d'avoir un résultat esthétique. Pour commencer, il est important de préciser que nous sommes satisfaits du résultat de notre travail. Cependant, cela a été accompagné de plusieurs problèmes lors du développement de notre sketch. Ainsi, cette section proposera une introspection sur les difficultés rencontrées tout en apportant la solution répondant à celles-ci.

Courbes

Bien que ce problème puisse rester anodin, l'utilisation des courbes a été et reste selon nous difficile à appréhender. En effet, les méthodes curves et bezier s'appuient sur des points d'ancrage, invisibles à l'écran pour dessiner les courbes, ce qui rend l'utilisation complexe à notre sens. Il ne s'agit pas vraiment d'un problème en soi mais nous avons tout de même décidé de le renseigner. Ainsi, la majorité du script utilise le moins possible ce type de fonction. Néanmoins, le dessin des courbes était important dans certains éléments de notre projet, comme le dessin des pages du livre, du serpent ou encore des manches du petit prince par exemple. En vue du temps donné pour la réalisation de ce travail, on a pris le parti de s'aider de l'intelligence artificielle afin d'expliquer au mieux l'utilisation de ces fonctions, par exemple à travers la génération d'exemple, en compléments de ceux

rapportés dans la documentation de p5.js, afin de mieux appréhender et comprendre ce type de fonctions.

Formes libres et redimensionnement

Une des autres problématiques concernait tout simplement la création de formes dite “libres”, c’est-à-dire ne se résumant pas à une forme géométrique de base, comme les cheveux du petit prince ou encore le corps du serpent. Pour se faire, on a utilisé la fonction “beginShape” et “endShape” de p5.js. Comme son nom l’indique, celles-ci permettent de créer une forme unique et de l'arrêter comme bon nous semble. Cela passe par l’utilisation de vertex (les segments de la forme), qui réunit, permet la création de formes originales, que l’on peut remplir avec la couleur de votre choix. Cela a été très utile notamment pour les pages du livre, afin de dessiner et de colorier les pages de façon efficace. En effet, avec l’effet de profondeur, les pages sont plus fines dans le fond de l’image, ce qui ne permet pas d’utiliser la fonction “rect” pour créer un rectangle classique.

Redimensionnement d’objets

De plus, une part importante de réflexion s’est portée sur la réutilisabilité de nos éléments graphiques, comme par exemple le dessin du petit prince. Pour éviter de le redessiner dans chaque scène, on a décidé de le créer une seule fois mais en faisant varier sa taille grâce à la gestion de l’échelle (scale).
Pour ce faire, on utilisé la fonction “scale” prenant en paramètre une échelle. Comme son nom l’indique cela permet de redimensionner automatiquement tout le contenu présent en fonction de la valeur renseignée. Pour bien délimiter l’objet à redimensionner, on s’est servi des fonctions “push” et “pop”, qui permettent respectivement de sauvegarder et de restaurer l’état du contexte de dessin.
Ainsi, tout ce qui est dessiné entre ces deux appels reste isolé et n’impacte pas le reste du canvas.
Combiné à la méthode “translate”, qui permet de modifier les coordonnées x et y du repère, cela a permis de travailler sur un dessin réutilisable plus facilement, tout en améliorant la lisibilité et l’optimisation du code. L’utilisation de valeurs absolues dans le dessin de certains éléments n’étaient donc plus un problème, grâce à la possibilité de modifier l’échelle et les coordonnées du dessin.
Cette méthode se retrouve dans beaucoup de nos illustrations : le prince, le renard, le serpent, l’avion, etc.

Animations

Enfin la dernière partie ayant posé le plus de problème à nos yeux concerne les animations. Quand on parle de cela, on ne prend pas en compte celles de fond, comme les étoiles ou les météorites. Celles-ci restaient relativement faciles à créer, grâce à la création d’objets gérant les différentes variables de l'animation.
Le plus complexe relève des animations au clic de l’utilisateur. Deux problématiques ont été constatées : comment savoir si l’endroit cliqué correspond bien à un élément cliquable et comment le faire bouger ? Pour les boutons servant de navigation, cela restait assez naturel à appréhender, parce que leur position était la même quelque soit la scène dans laquelle on se trouvait. Mais pour les éléments précis des différentes pages, cela nécessite de procéder de façon différente.

Tout d'abord, reprenons la structure de notre projet. Le canva est décomposé en 3 plans :

- 1e plan : Les boutons
- 2e plan : Le livre
- 3e plan : La scène en fonction de la page courante

Cela permet de ne pas redessiner sur les éléments dissociés les uns des autres. De plus, chaque scène de projet est écrite dans un fichier JavaScript unique (scene0.js jusqu'à scene5.js). Partageant tous le même canva, il fallait dans un premier temps bien nommer le nom de nos fonctions et de nos variables globales, pour éviter des conflits entre les fichiers.
Concernant les animations présentes dans les scènes 1, 2 et 3, sur les objets, la méthode a été un peu différente. En effet, le code pour les animations est écrit directement dans les scripts des scènes adéquates et est appelé dans le fichier "sketch.js". Pour bien déterminer la position de l'objet, et cela pour chaque objet animé, on utilise des tableaux de données pour déterminer la position des éléments sur lesquels cliquer. Cela permet de déterminer si, en fonction de la scène sur laquelle on se trouve, la zone sur laquelle on a cliqué lance un événement ou non.
Celles-ci ont majoritairement été construites en manipulant les fonctions "translate" et rotate sur une partie de l'objet, voir l'objet entier. Pour gérer l'avancement de la fonction, on a dû utiliser plusieurs variables, gérant l'avancement de l'animation, en incrémentant au fur et à mesure. La principale difficulté réside dans le temps que cela nous a consacré. En effet, répéter la même logique pour plusieurs objets prenait un temps non négligeable.

Conclusion

Malgré ces différents problèmes, nous sommes très satisfaits du résultat et de l'amélioration de l'usage de p5.js comparé aux précédents TDs, malgré le temps imparti. On a réalisé au mieux les différentes scènes, tout en les rendant reconnaissables et c'était l'effet voulu. La création du livre était certes complexe elle aussi mais apporte une touche d'interaction dans le sketch que nous trouvions très importante, d'autant plus le sujet du projet. Le fait de travailler par binôme nous a vraiment permis d'avoir un résultat plus riche en contenu, d'avoir poussé les fonctionnalités implémentées au maximum et d'avoir un résultat plus complet qu'un travail réalisé individuellement.
A noter aussi que la structuration du code a été effectuée avec un fichier .prettierc, formatant automatiquement les lignes du fichier, expliquant parfois la structure un peu particulière de celui-ci.

Annexe

Lien de l'éditeur p5.js : https://editor.p5js.org/pierrick.dennemont/sketches/erKoS_Qyd
A noter que les performances ont l'air moins efficaces que de lancer le sketch en dehors de l'éditeur en ligne. Il est donc préférable de l'exécuter en démarrant la page « index.html » dans le navigateur.
Comme demandé, le dossier zip contient uniquement les fichier .js, les autres fichiers sont cependant disponible dans l'éditeur en ligne