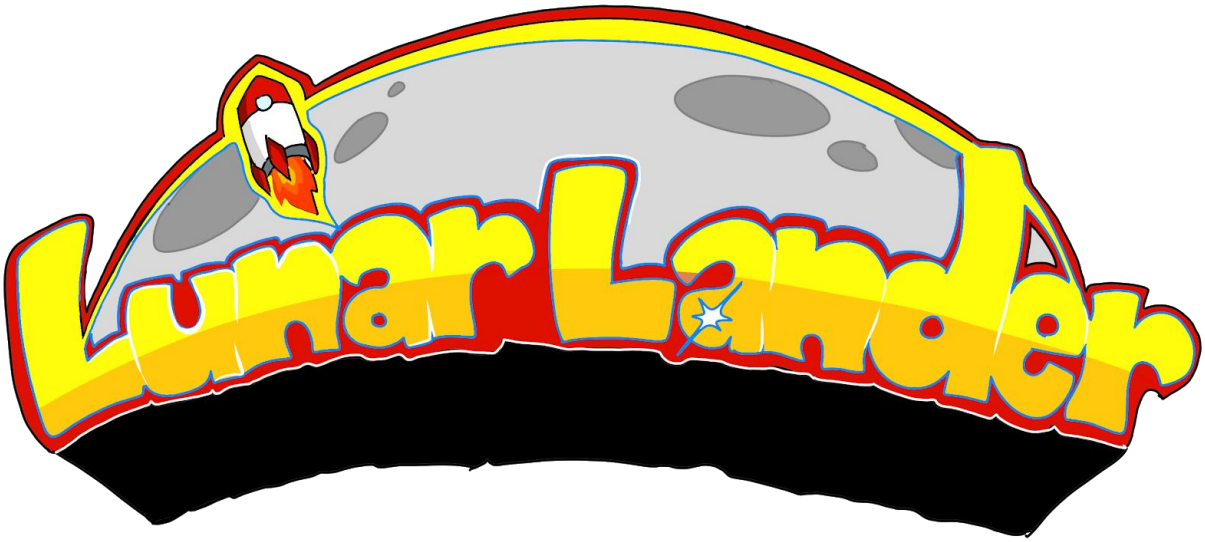
Loris Schirar Lysandre Macke

Objectif Lune



UPEM L1 Maths-Info

Année 2019/2020

TD f ∞ TP 9

Projet de fin de semestre

Janvier 2020

SOMMAIRE

Loris Schirar &
Lysandre Macke

I/ Introduction du sujet

A. Approche du sujet : le *Lunar Lander* ?

Lunar Lander (éponyme du petit module utilisé par les astronautes des missions Apollo pour se poser sur la Lune) est un petit jeu vidéo permettant au joueur de simuler **l'alunissage d'une petite fusée**. Ce dernier possède le contrôle des différents réacteurs de la machine et peut ainsi la déplacer dans un espace 2D supposé infini affiché à l'écran. La mission ne sera pas sans embûches: l'utilisateur sera confronté à diverses difficultés, à savoir la gestion du **carburant**, **l'attraction lunaire** sur la fusée qui accélère sa chute, mais également le **terrain abrupt** de l'astre empêchant l'alunissage sur certaines zones.

B. Ce que nous vous proposons

Nous avons implémenté toutes les caractéristiques originales citées ci-dessus, à l'exception du terrain supposé infini que nous avons remplacé par un système de **niveaux** : nous avons créé une série de terrains, plus ou moins hostiles. A mesure que l'utilisateur progresse dans le jeu, il découvre des décors de plus en plus atypiques. Nous avons ajouté des fonctionnalités supplémentaires, que nous développons dans la partie suivante (documentation utilisateur).

C. Manuel du Lunar-naute

Lorsque l'utilisateur lance le programme, une fenêtre s'ouvre, présentant le menu de chargement des sauvegardes. Il peut alors charger une sauvegarde existante, s'il y en a une. Il peut également en créer une nouvelle en sélectionnant un emplacement vide ou en supprimant une autre sauvegarde.
L'utilisateur doit utiliser les touches "gauche" et "droite" du clavier pour naviguer entre les différentes options et cliquer sur [entrer] pour confirmer sa sélection.
Une fois une partie chargée, il a accès à l'écran titre, second menu du jeu, contenant trois boutons. Il interagit cette fois avec le menu par l'intermédiaire de la souris. JOUER lance *Lunar Lander*, OPTIONS ouvre le menu Options, et enfin QUITTER permet simplement de fermer la fenêtre.

- **Le menu Options**

  Ce menu offre l'accès à certains paramètres de jeu, modifiables en quelques clics entre deux alunissages (ou crash !).

  Skin : Le joueur peut choisir de contrôler la fusée (par défaut) ou bien la pomme volante de Newton (cf la doc technique, *Ce bon vieux Newton*).
  Cette option fait également office de niveau de difficulté: la pomme n'effectue pas de rotations (niveau facile) tandis que les réacteurs latéraux de la fusée la font pivoter.
  Si nous avions disposé de plus de temps, nous aurions aimé dessiner d'autres véhicules, avec éventuellement chacun leurs caractéristiques. Il ne serait pas très compliqué de modifier le programme dans ce sens.

Niveaux : Les niveaux sont les différents terrains lunaires avec leurs particularités et leurs difficultés. Ils sont nommés de 0 à 15 au moment du rendu de ce rapport et ne sont pas spécialement classés par ordre de difficulté croissante. Il existe également un niveau final 'Terre', dont nous parlerons dans la section concernant les Succès.
Après avoir passé ou effectué le tutoriel (voire la section correspondante), le joueur peut choisir librement le décor dans lequel il évolue. S'il ne touche à rien, il passe au niveau suivant dès qu'il réussit un alunissage. Il ne peut pas revenir au niveau 0 (niveau tout plat du tutoriel) ni accéder au niveau secret final par l'intermédiaire de ce menu.
Une fois un niveau terminé (un alunissage réussi sur le niveau en question), le meilleur score s'affiche près du numéro de ce niveau. Le joueur peut essayer de l'améliorer ou essayer de compléter tous les niveaux.

Autres : les différents paramètres de jeu (vitesse maximale d'alunissage, force des moteurs, etc…) sont l'objet d'un équilibre fragile que nous n'avons pas toujours su trouver, rapport à la taille de la fenêtre de jeu, la maniabilité de la fusée et d'autres choses dans les succès. En tâtonnant, nous obtenions souvent quelque chose d'injouable
Nous avons donc préféré, contrairement à ce qui était proposé, ne pas donner au joueur la possibilité de les choisir lui-même.
Il est cependant possible de les modifier en touchant à une seule ligne du code pour la plupart (voire doc technique).

- **Les succès**

Nous proposons également aux Lunar-nautes en quête d'aventure un certain nombre d'objectifs à accomplir pour débloquer la fin du jeu (une liste de ceux-ci est disponible ici : Liste des succès).
Certains concernent les paramètres d'alunissage (vitesse, carburant, temps, etc…) et d'autres nécessitent de se trouver dans des situations particulières (tomber à court de carburant en altitude, sortir de la fenêtre à une certaine vitesse…). La plupart sont des références à des œuvres ou des évènements historiques où l'espace occupe une place importante.
Il s'agit d'un moyen d'enrichir l'expérience de l'utilisateur en incluant quelques surprises dans le jeu, mais il n'est pas obligatoire de les réaliser.
Une série de cases vides se trouve en bas de l'écran au cours du jeu. Dès que le joueur débloque un succès, la médaille correspondante apparaît à sa place. Il y a 16 succès en tout, dont un (au moins) est accompli lors du tutoriel.

Une fois les 16 médailles collectées et tous les niveaux terminés au moins une fois, le joueur accède à la fin du jeu: d'abord, le niveau spécial 'Terre' où il doit ramener sa fusée en sécurité au Lunar Center, puis un menu déroulant où nous le remercions d'avoir joué.
A noter que le niveau final présente un design unique, mais pas de difficulté particulière, mis à part le fait que la physique est très différente de celle à laquelle le joueur a été habitué (intensité de pesanteur plus importante, vent, frottements fluides…).

- **Le Tutoriel**

  Quand l'utilisateur lance le jeu pour la première fois sur la sauvegarde chargée, une boîte de dialogue s'ouvre à l'écran. Un clic de souris sur la fenêtre permet d'accéder à la suite du dialogue. Le joueur peut passer le tutoriel ou l'effectuer. S'il le passe, il n'aura aucun moyen d'y revenir par la suite, sauf en chargeant une nouvelle sauvegarde. S'il accepte, il sera guidé par les conseils d'un petit Lunar-naute au long des 2 premiers niveaux. Celui-ci explique brièvement les mécaniques décrites dans ce chapitre (succès, options, affichage etc…)

- **Affichage**

  Au début de chaque niveau, la fusée apparaît à un endroit aléatoire en haut de la fenêtre; avec une vitesse dépendant de sa position par rapport au centre. Elle commence dès lors à chuter sous l'effet de la gravité. Sa trajectoire est représentée par de petits points blancs.
  Au sol, on peut identifier deux types de terrain : les zones rocailleuses (en blanc) et les zones propices à l'alunissage (en vert). En haut à droite de l'écran se trouve le tableau de bord de l'utilisateur, affichant en temps réel les informations dont il a besoin pour diriger son engin: le temps, l'altitude, la quantité de carburant, les vitesses horizontales et verticales et l'angle formée avec la verticale (qui ne varie que pour la fusée et non pour la pomme).
  Lorsque la fusée touche le sol (ou dans d'autres circonstances exceptionnelles liées aux succès), le jeu s'arrête. Un message indique au joueur s'il a obtenu un succès ou si son alunissage est réussi. Les boutons JOUER, OPTIONS et QUITTER réapparaissent alors.
  Au cours d'une partie, un clic sur le bouton de fermeture permet de quitter. La sauvegarde est automatique.
  Si on ne quitte pas le jeu, il est possible de continuer à jouer jusqu'à ce qu'on ait fini tous les succès. Le cas échéant, on ne laisse pas au joueur la possibilité de rester sur la Lune et de continuer à jouer normalement : il passe au niveau 'Terre' automatiquement.

- **Mécanique du jeu**

  L'utilisateur contrôle la fusée grâce aux touches directionnelles du clavier ([haut], [gauche], [droite]). Chacune de ces touches est associé à un réacteur : le réacteur central permet de propulser la fusée dans la direction où elle pointe son nez, et les réacteurs gauche et droite permettent de modifier l'orientation et la trajectoire de la fusée (ou de déplacer latéralement la pomme).
  Pour un alunissage réussi, la fusée doit se poser sur une plateforme verte, perpendiculaire au sol et à une vitesse globale (norme du vecteur vitesse) inférieure à 100 m/s.

II/ Notre programme

A. Pour que notre fusée atteigne les étoiles

a. Ce bon vieux Newton !

Selon le sujet, le principal enjeu était de simuler des trajectoires physiquement crédibles. C'est donc par-là que nous avons commencé. "Newton" est une petite bibliothèque contenant une batterie de fonctions permettant de déplacer un objet ponctuel dans un champ gravitationnel uniforme. On peut choisir aussi le nombre de dimensions de l'espace dans lequel s'effectuent les simulations.
Il est plutôt court et ses fonctions sont pour la plupart bien documentées.
Il contient également un prototype du Lunar Lander démuni d'interface et de menus, utilisé pour tester les différentes fonctions lors de leur développement. Mais ici, c'est une pomme qu'on contrôle (cette même pomme que nous avons plus tard ajouté parmi les skins disponibles).

La position et la vitesse instantanée du point sont représentées par des listes plutôt que des tuples car celles-ci sont mutables. Celles-ci peuvent être modifiées tout en restant globales. On cherche à simuler une trajectoire en deux dimensions pour pouvoir l'afficher dans la fenêtre, ces listes ont donc deux éléments chacune.
La variable globale `Dt` représente le temps d'attente en secondes entre deux prises de coordonnées. Tous les `Dt` secondes donc, l'accélération de la pomme (également une liste) est calculée : on ajoute le poids de celle-ci, et (éventuellement), la force de poussée des moteurs, un vent fixe et des frottements fluides proportionnels à la vitesse. Il n'y a pas de limite au nombre de forces qu'on peut appliquer sur le système.
Ce programme contient donc également les fonctions permettant de déplacer la pomme avec le clavier : on distingue la fonction `direction()`, qui fait agir un moteur principal et deux moteurs latéraux (niveau facile), de la fonction `rotation()`, qui permet de faire pivoter la fusée (niveau difficile). Nous n'avons qu'une seule image de la pomme, c'est pour cela qu'elle n'utilise que le niveau facile. Certains détails présents dans le jeu font défaut à ce prototype (par exemple, dans le programme final, la masse de carburant brûlé est déduite de celle de la fusée). Il s'agit de celles que nous avons implémentées plus tard.

b. Dessinons la Lune

L'éditeur de terrain permet de créer et de visualiser les niveaux.
On affecte d'abord à la variable `lvl` le nom du niveau qu'on veut éditer (type: `int` ou `str`).
Ensuite, on lance le programme. Le terminal demande une confirmation pour créer ou écraser le niveau choisi. Pour simplement le visualiser, saisir "non".
Une fenêtre s'ouvre alors, avec le terrain dessiné automatiquement par les fonctions d'upemtk. Des tests de contrôle affichent des détails supplémentaires, afin de s'assurer que la hitbox du sol est bien placée aux bons endroits: une ligne bleue pour la surface et des points verts et roses pour (respectivement) l'extérieur et l'intérieur.

Pour créer un nouveau niveau où en refaire intégralement un déjà existant, entrer "oui" dans le terminal. Une fenêtre vierge s'ouvre alors, où on peut saisir une série de points à la souris. Un clic gauche place un point, un clic droit termine le tracé. Le programme relie ensuite automatiquement les points entre eux et dessine le nouveau terrain.
Les coordonnées de ces sommets sont en fait converties vers leurs quotients respectifs par les dimensions de la fenêtre. Cela permet de changer la taille de la fenêtre sans altérer l'apparence globale du niveau ou provoquer une erreur.
La liste des quotients est stockée dans un fichier .txt, créé automatiquement dans le dossier "__niveaux__".
Il est donc possible de modifier manuellement le contenu de ce fichier. Certains niveaux, comme le numéro 13 (où une partie du tracé est en dehors de la fenêtre, donc impossible à saisir avec des clics), ont été réalisés de cette manière. Cela sert également à ajuster manuellement la position des plateformes (bien qu'une légère inexactitude soit tolérée lors de la saisie au clic).
Ce programme est très simple à manipuler (tant qu'on ne va pas jusqu'à ajuster manuellement dans le fichier) ; nous avions pensé le mettre à la disposition de l'utilisateur à l'intérieur du menu Options afin qu'il puisse réaliser ses propres niveaux (stockés bien sûr dans un autre dossier). Nous ne l'avons pas fait par manque de temps.

B. Au cœur du module lunaire

a. Répartition des tâches

Lorsqu'on travaille sur la construction d'une fusée, il est essentiel d'être méthodique et bien organisé.
Comme nous ne pouvions pas nous voir pendant les vacances, nous avons opté pour la création d'un Drive commun, disponible ici: LUNAR_LANDER.
Celui-ci était mis à jour à chaque nouvelle avancée ou modification, même les moins significatives. On peut y retrouver les images, les textes et les programmes en différentes versions. On y trouve aussi le Journal de Bord, où nous nous tenions mutuellement informé du contenu des mises à jour et du travail restant.
La charge de travail au sein du binôme a été plutôt équilibrée. Bien que chacun ait un peu travaillé sur tout, Lysandre s'est occupée surtout de dessiner les images et les animations, et a créé la majeure partie de l'affichage des menus, la gestion des clics, la mise en place des sauvegardes et du tutoriel et l'éditeur de terrain.
Loris, quant à lui, s'est occupé de Newton et de la physique du jeu, de la gestion des game_over et des succès et de l'assemblage de différentes parties pour créer un tout cohérent. Il a également passé en revue le code à chaque étape pour vérifier la syntaxe et le rendre le plus concis possible.

b. Fonctions, variables, structure

Nous avons cherché un compromis entre efficacité et simplicité lors de l'implémentation des fonctionnalités. Notre programme principal fait plus de 700 lignes. Les deux programmes annexes en font presque 300 chacun et les trois contiennent un grand nombre de variables et de fonctions. L'explication présentée ici sera donc loin d'être exhaustive et de rendre compte de tout ce qu'il se passe

durant une partie. Nous essaierons d'aller à l'essentiel. Pour de plus amples détails, se référer aux commentaires et aux docstrings.

La structure principale de notre *Lunar Lander* est constituée de deux boucles imbriquées, à savoir « `while not Quitter:` » et « `while Jouer:` ». La première permet de répéter autant de parties qu'on le souhaite tant qu'on ne quitte pas le jeu (`Quitter` est le booléen dépendant du bouton de même nom du menu principal). La seconde, plus petite, est celle qui gère tout ce qu'il se passe au cours d'une partie (caractère continu du déplacement de la fusée). `Jouer` est aussi le booléen lié au bouton JOUER du menu principal.

Avant de rentrer dans ces boucles, on crée la fenêtre et on importe le fond, puis on procède à la sélection d'une sauvegarde grâce aux fonctions `menu_sauv()` et `retablir_save(partie)`.
Le chargement d'une sauvegarde met à jour une série de variables :

- `Tuto`, un entier qui représente le stade du tutoriel où se trouve le joueur (0 si le joueur a fini ou passé le tutoriel) : on ne charge donc le tutoriel que si nécessaire

- `lvl`, un entier correspondant au niveau actuel où se trouve le joueur

- `succes`, une liste de booléens. Pour chaque succès obtenu, ce booléen vaut `True`. Ainsi, on récupère sa progression en chargeant une sauvegarde.

-Niveaux, une liste d'entiers. Le principe est identique à celui de la liste succès, mais associe à chaque niveau le nombre d'alunissage réussi dans celui-ci.
- `best_niveaux`, une liste de triplets contenant les meilleurs scores (carburant – vitesse – temps) des niveaux réussi (trois fois `None` sinon)

Si le fichier est vide, ces variables sont remplies par des valeurs de début de partie par défaut. On commence au niveau 0, avec le tutoriel actif, sans avoir réussi aucun succès ni aucun niveau.

Les premières lignes de la boucle `while not Quitter` sont une série de contrôles des booléens `Menu`, `Options` et `Jouer`, dont un seul vaut `True` (c'est `Menu` qui est initialisé à `True`) qui permettent l'affichage des principaux menus du jeu.
Les fonctions `menu_titre()` et `menu_options()` sont responsables de la gestion des clics de leurs menus respectifs. Les fonctions `ecran_titre()` et `affiche_options()` leurs sont associées et gère les affichages.

Les variables `lvl` et `skin` sont modifiables dans le menu Options.
La fonction `affiche_setup()`, appelée dès le début de la partie, récupère les données liées au terrain (variables Sol et pistes) en faisant appel aux fonctions de l'éditeur de terrain.
Elle affiche le terrain, les médailles de succès et tout ce qui va avec.

Tant qu'on est dans la boucle `while Jouer`, les informations du tableau sont mises continuellement à jour grâce aux fonctions de Newton, de la manière décrite dans la section correspondante.
Chaque frame, on appelle `affiche_fusee()`, `affiche_compteurs()` et `affiche_reaction()`, pour mettre à jour l'image également.
Lorque la fusée touche le sol, `Jouer` devient `False` et on revient au Menu.

Une batterie de petites variables récoltent tout un tas d'informations, comme la sortie de l'écran ou la distance parcourue, et les communiquent à la fonction `game_over()`, qui agit en conséquence comme mentionné plus haut.

Les variables `Masse`, `Moteurs` (force de poussée des moteurs), `g` (intensité de pesanteur), `k` (forces de frottements fluides) et `v_max` (vitesse maximum d'alunissage autorisée) sont modifiables au début du code.
`k` ne sert que pour le niveau Terre, où il y a effectivement des frottements fluides.

- **Comment le faire fonctionner ?**

  Pour fonctionner correctement, Lunar Lander doit être placé dans un dossier contenant également :

  - Les programmes annexes Newton_v3, éditeur_de_terrain et upemtk

  - un dossier '__niveaux__' contenant des fichiers .txt avec les informations nécessaires à la génération d'un terrain. Ils doivent être nommés 'niveau_i', pour i allant de 0 à un entier quelconque, sans laisser d'entier entre 0 et i. La seule exception est le niveau spécial Terre, nommé 'niveau_144'
  Dans le dossier de rendu, nous avons également inclus un 'niveau_test', que l'on peut utiliser pour tester l'éditeur de terrain, mais pas pour jouer.

  - un dossier 'textes', contenant des fichiers .txt avec les textes des options, du tutoriel, du menu de fin de jeu et du menu game_over, ainsi qu'un dossier où sont rangées les sauvegardes. Les textes affichés dans la fenêtre sont tous encodés en utf-8.

  - un dossier 'images', contenant lui-même d'autres dossiers : du fait du grand nombre d'images utilisées par le programme, nous ne les détaillerons pas ici mais nous vous invitons à jeter un coup d'œil à son organisation (dans le dossier de rendu)

  Nous travaillions tout deux sous Windows, chaque occurrence d'un chemin d'exploitation est donc de la forme '<u>images\\logo\\sprite_logo_14.png</u>'. Nous n'avons pas eu la possibilité de tester notre programme sous d'autres systèmes d'exploitation, et ignorons si les mêmes chemins seront valides sous Linux.

III/ Conclusion : rapport des Lunar-nautes

A. Ce que nous aurions pu améliorer/ajouter

Au début de la réalisation du projet, nous avions beaucoup d'idées en tête, toutes plus ou moins réalisables et/ou pertinentes. Nous avons déjà fait référence à certaines de celles que nous avons dû abandonner. En voici une liste plus exhaustive :

- L'implémentation d'un éditeur de niveau basique, accessible au joueur, dans le programme principal (cf *Dessinons la Lune*)
- Le remplissage du sol lunaire par une image dessinée plutôt que par un fond gris
- Une animation pour le crash de la fusée et une pour l'obtention d'un succès
- Une caméra dynamique (idée proposée par l'énoncé) effectuant des zooms et travellings
- Un terrain « infini » (à explorer avec la caméra dynamique)
- Un système permettant au joueur d'accéder aux informations concernant les succès sans quitter le jeu, en cliquant sur les emplacements de médailles

Enfin, certains design ont été terminés un peu rapidement pour être dans les temps et mériteraient sans doute d'être repris.
La différence de fonctionnement entre les différents menus est assez gênante (certains menus fonctionnent par clic de souris et d'autres entièrement au clavier), et aurait dû être réglée.
Enfin, toutes les fonctions du programme principal ne sont pas munies de docstrings ni de tests. Nous espérons que cela n'impactera pas trop la lisibilité du code, et avons essayé de commenter aux endroits nécessaires.

B. Les difficultés rencontrées

Dans la section précédente, nous présentions une liste des modifications que nous avons abandonnées. Si dans la plupart des cas il s'agissait d'une question de temps, parfois nous n'avons tout simplement pas été capables de trouver une solution réalisable au problème posé (c'est-à-dire qui n'aurait pas supposé de multiplier la longueur de notre code par deux). C'est ce qu'il s'est passé pour la gestion de la caméra et le terrain infini, par exemple : il existe certainement u moyen de le faire, mais il devait être incompatible avec la manière dont nous avions choisi de générer notre terrain.

Le calcul du score est un calcul global : il prend en compte la vitesse d'alunissage, le carburant restant et le temps de vol. Il n'est pas aisé de choisir le poids de ces trois éléments pour définir un score unique. Nous le calculons comme ceci, faute d'une meilleure méthode:

$$Score = \frac{carburant}{10} + \frac{100}{vitesse} + \frac{100}{temps}$$

Travailler séparément n'a pas toujours été simple : nous avions parfois besoin de documents ou modifications que seul l'autre possédait pour continuer à travailler. Une fois que le Drive a été mis en place, il était bien plus simple de s'organiser. Nous avons également pu nous téléphoner à deux ou trois reprises pour tout mettre au point.

Enfin, dernier détail : lors des parties trop longues, il arrive que le jeu crash et que la fenêtre se ferme brusquement pour une raison inconnue. Nous n'avons pas identifié l'origine de ce problème. S'il n'a pas été réglé, ses conséquences sont minimisées par l'ajout des sauvegardes automatiques, qui permettent de reprendre à l'endroit où l'on s'est arrêté.

C. Ce que nous avons appris

« Travailler sur *Lunar Lander* m'a pour ma part permis de travailler sur des affichages proches de ce qu'on peut trouver dans les jeux vidéo, que ce soit au niveau des images et animations que nous avons inséré dans le jeu ou encore (et surtout) au niveau des affichages interactifs (tutoriel, menu de chargement de sauvegardes). Je pense également que l'expérience d'un travail de groupe "séparé", quasiment entièrement à distance fut pour le moins... enrichissante. Aujourd'hui je suis fière du jeu que nous avons produit, mon partenaire et moi, bien qu'il présente des choses qui auraient pu être améliorées. »

Lysandre

« Ce qui rend ce type de travaux passionnant, c'est le fait que ce soit une des applications les plus directes de ce que nous avons appris en programmation. Nous avons dû faire appel à tout le spectre de connaissance que nous développons depuis plusieurs mois pour confectionner notre programme. Le travail en groupe est également une bonne expérience ; cela permet de mieux dispenser la charge de tout ce qu'il reste à faire, et d'entretenir par la discussion un renouvellement dans nos idées. Nous avons tous les deux pris du plaisir à développer ce jeu, et je suis assez satisfait du rendu final, que nous avons tenté de faire ressembler à un vrai jeu vidéo : sauvegardes, fin du jeu, etc… Nous avons dû faire preuve de créativité et d'organisation pour venir à bout de certains des défis que nous avons rencontrés. J'étais également heureux de pouvoir retravailler mes connaissances en physique, une matière qui me manque un peu parfois. »

Loris

Sources et liens

Lunar Lander:

Difficile de créer notre jeu sans une vague idée de ce à quoi l'original ressemblait. Voilà où nous avons puisé notre inspiration:

-Lunar Lander — un simulateur en ligne du jeu sur Atari !
-https://youtu.be/McAhSoAEbhM — une vidéo du jeu sur Atari
-https://youtu.be/Jo3PwrWAPMc — la vidéo fournie par l'énoncé
-Lunar lander — la page Wikipédia du jeu

Images :

Nous avons réalisé la plupart de nos images nous-mêmes, mais pour gagner du temps nous avons utilisé ces images issues du web pour créer les médailles de succès:

-https://history.nasa.gov/apollo/images/apo11.gif
-https://upload.wikimedia.org/wikipedia/commons/thumb/0/05/Apollo_13-insignia.svg/480px-Apollo_13-insignia.svg.png
-http://www.gabuzo38.fr/shads/shadoko3.gif
-http://www.brickshelf.com/gallery/Okay/Movies/Disney/ToyStory/Misc/star_command_logo.png
-http://4.bp.blogspot.com/-bPAYvpqGUPQ/VRcpBboZm4I/AAAAAAACHO0/c7ykzzx1Tas/s1600/17.gif
-http://kunisan.jp/planets/jupiter.gif
-https://media.istockphoto.com/vectors/cartoon-gas-can-vector-id472391063?k=6&m=472391063&s=170667a&w=0&h=WE7oTjAuMWERt1H4GRTmLjT_2a9ZGO1xiKBdLGxAmnc=
-http://pngimg.com/uploads/snails/snails_PNG13216.png
-https://2.bp.blogspot.com/-gZ-0hlJ7F4E/WMrH0d8JBWI/AAAAAAAApSc/fmqmrKG13oQTBV3QGFYifn2Wh2mqVKlTwCLcB/s1_600/ship-large.png
-https://upload.wikimedia.org/wikipedia/commons/thumb/0/07/3-4_star.svg/647px-3-4_star.svg.png
-https://upload.wikimedia.org/wikipedia/commons/thumb/6/66/Oxygen480-actions-chronometer.svg/1024px-Oxygen480-actions-chronometer.svg.png
-http://citronelle.c.i.pic.centerblog.net/pr52y4xe.gif
-http://www.pouce-et-compagnie.com/casque-sonore-astronaute-ferme.gif
-https://thumbs.gfycat.com/SandyPotableBarbet-size_restricted.gif
-https://preview.redd.it/wiga0fsgors11.png?width=248&auto=webp&s=fb46db274487ffcab4fd7316d6e576fbf20ae3d5
-http://www.pngmart.com/files/8/Exclamation-Mark-Background-PNG.png

Et pour redimensionner / convertir des images:

-https://en.bloggif.com/resize?id=b1312973663f49276f1ddd4103bd1af3

Python:

Notre principale source d'information était la plateforme E-learning où se trouvent nos cours. Cependant, nous nous sommes également aider d'autres documents (principalement Internet) pour certaines choses (par exemple l'encodage en utf-8 et la bibliothèque os). La liste proposée ici n'est probablement pas exhaustive, car il difficile de retrouver tous les sites ou forums croisés durant ces recherches. En voici du moins quelques-uns qui nous été bien utiles:

-Python Code Checker - Online syntax check
-Zeste de Savoir
-Welcome to Python.org
-Club des développeurs Python : actualités, cours, tutoriels, faq, sources, forum
-Python
-Python Tutorial
-Pythonforbeginners.com - Learn Python by Example
-Supprimer un fichier ou un répertoire avec python
-Python Tutorial
-Alphabet range in Python