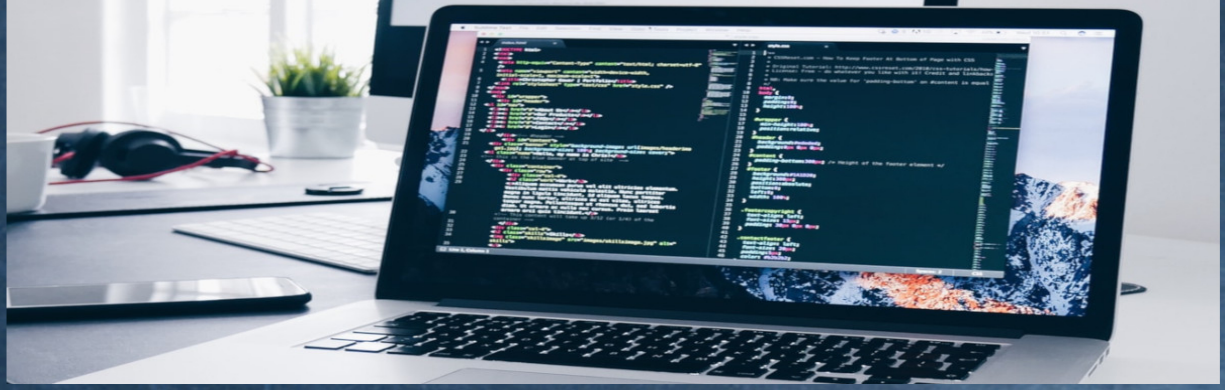


Banana Strike

Olivier Torrens

Jungle's Edition

Art. No SIN4U1T
ISBN 123–20–0106–21–1
Edité par 1.0

Design de la couverture par Cover Designer

Publié par Jungle's Edition
Imprimé en Dordogne represent

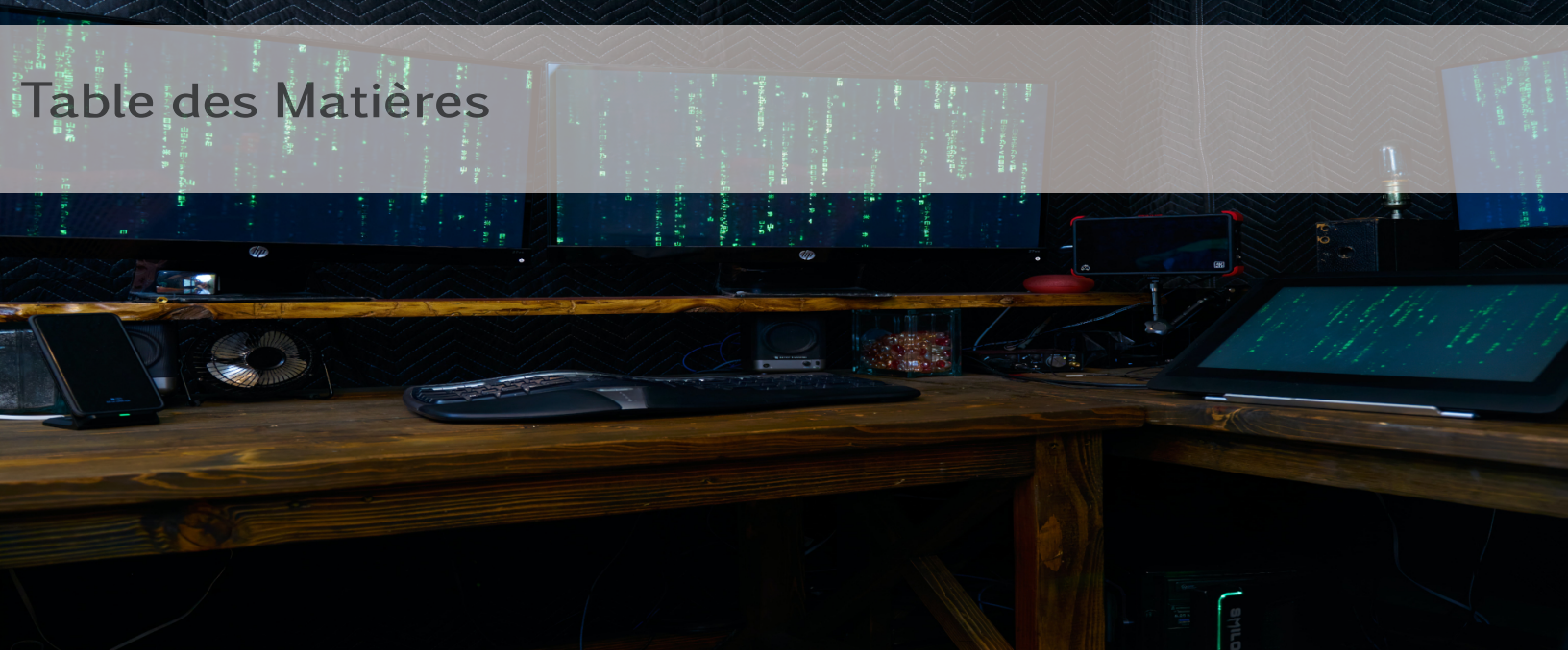
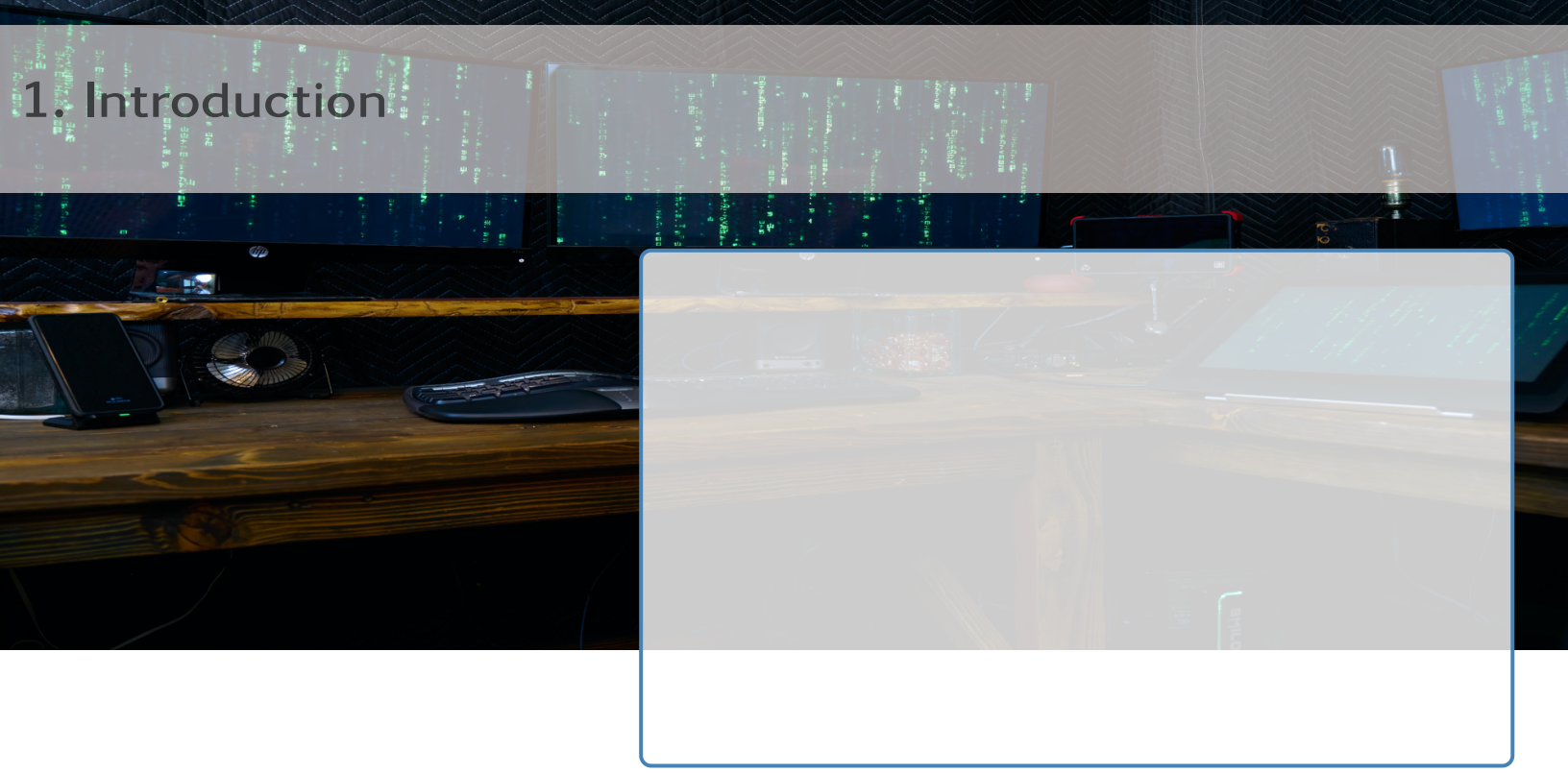


Table des Matières

1. Introduction



Patience et longueur de temps valent mieux que force ni que rage

---

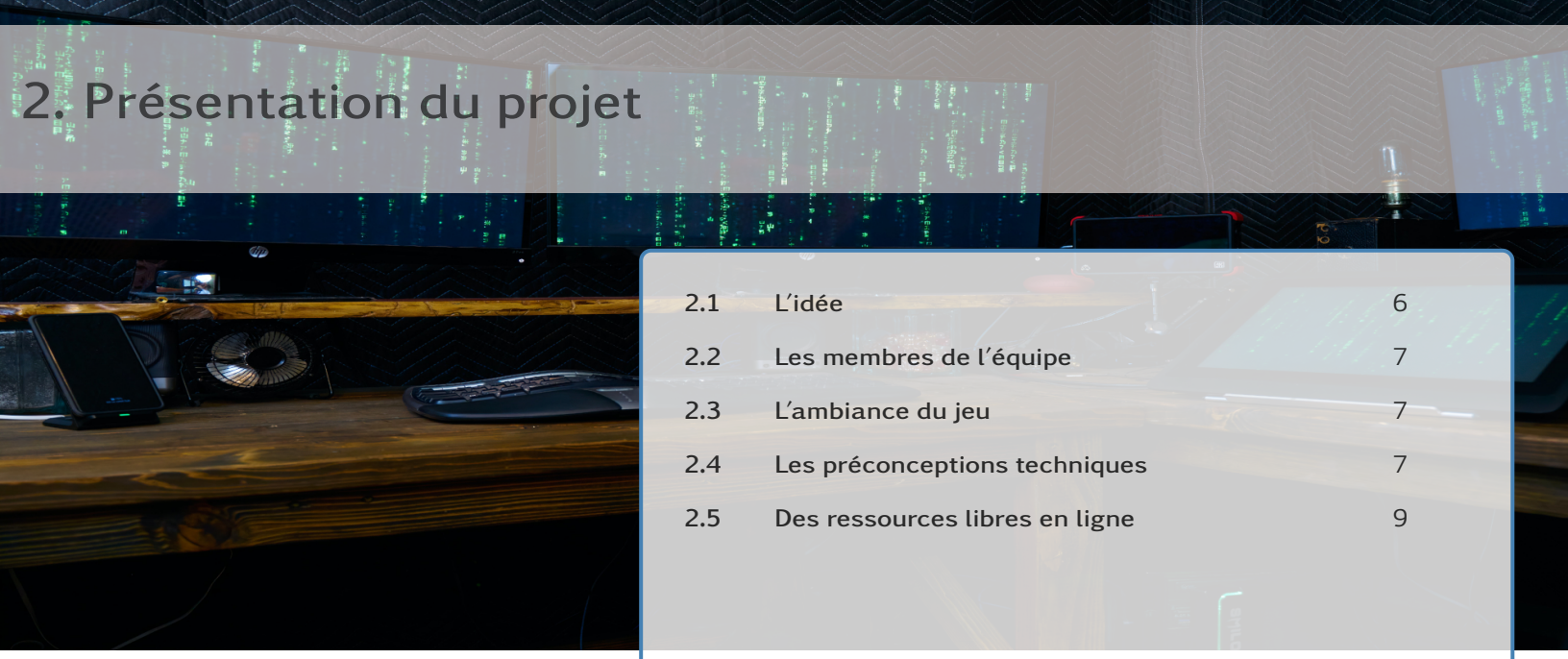
Ce document s'inscrit dans le cadre le la licence 3 de mathématiques-informatique (MI) proposée par l'université d'Aix-Marseille. Il s'agit de la rédaction d'un projet de développement informatique dont l'objectif est d'obtenir la validation de 6 crédits ECTS dans le cadre du sixième semestre du parcours licence MI.

L'objectif de ce projet, basé sur le concept du serious game, est de faire du développement informatique sur un sujet libre de choix. Le travail se déroule en équipe, et reprend le cadre de ce que pourrait être un travail de développeur informatique dans une entreprise. Il y a un chef de projet responsable du développement, qui peut éventuellement jouer le rôle de "développeur sénior". Il manage et coordonne une équipe d'une ou deux personnes, aussi bien dans la technique que dans les choix inhérents au déroulé du projet. Il y a également *Ze Big Boss* qui occupe le statut de celui qui a commandé au chef de projet le développement. Il valide ou non les idées et les choix stratégiques de l'équipe. Il rappelle aussi que son argent est précieux et que celui-ci est investit à concurrence du temps passé sur le projet.

Comme pour le sujet du projet, les outils sont laissés libres. Mais comme il peut y avoir un temps d'adaptation aux outils techniques, *Ze Big Boss* est aussi un conseiller technique avisé. Les arcanes de la programmation multi-langage n'ont plus de secrets pour lui et il peut fournir, à la demande, des conseils judicieux pour débloquer tel ou tel point.

L'idée du projet de mon équipe a été de faire un jeu (rudimentaire) sur PC inspiré de la catégorie des *jeux balistiques*. Un joueur affronte des cibles qu'il doit détruire en lançant sur elles un projectile. Les jeux les plus connus de ce genre sont sûrement ceux de la série Worms et de leurs ancêtres les Lemmings.

2. Présentation du projet



Le projet est d'abord initialement proposé par un chef de projet. L'équipe se constitue ensuite de volontaires pour la rejoindre ou de personnes désignées par Ze Big Boss *car ces dernières ne possèdent pas d'équipe ou alors ont vu leur projet refusé.*

2.1 L'idée

Quand j'étais lycéen à la fin des années 1990, je possédais une calculatrice graphique. Cette calculatrice, une TI-89, faisait partie des premières générations de calculatrice qui pouvait être connectée à un ordinateur afin de récupérer des programmes glanés sur le Web. Certains de ces programmes étaient des utilitaires scientifiques, citons par exemple la classification périodique des éléments chimiques ou bien encore des aides à l'analyse de fonction mathématique. En effet la TI-89, comme d'autres modèles de la marque ainsi que d'autres marques (Casio, HP...) hébergeait un interpréteur de programme spécifique qui permettait de développer des applications écrite en basic. Il y avait aussi la possibilité de développer le programme sur une autre plateforme et de le rendre disponible ensuite au téléchargement. Ainsi on pouvait récupérer, à l'époque, des jeux imitant par exemple celui d'un plombier italo-japonais qui affrontait des tortues génétiquement modifiées. Ceci permettait d'occuper des heures d'études entre deux cours, à l'époque il n'y avait ni téléphone portable et internet était un luxe en application nomade mais aussi et parfois encore au domicile de chacun, où connexion sur AOL rimait avec monopolisation de la ligne téléphonique et source de conflit ado-parents.

Et notamment, par la multitude d'application de jeux disponibles, il y avait un jeu qui a occupé bon nombres de mes heures d'études à l'époque : deux gorilles de type king-kong se faisaient face dans un décor urbain symbolisant la ville de New York. Ils se lançaient des bananes au tout par tour ce qui permettaient de faire descendre leur barre de vie. Le premier à zéro de vie avait perdu. Le jeu se jouait face à un autre joueur humain ou alors le programme lançait en automatique les bananes sur le joueur humain, les prémices de l'intelligence artificielle de jeu de combat.
Ci-après une capture d'écran d'un portage en langage `Python` du jeu original. C'était là le point de départ de mon idée de projet.

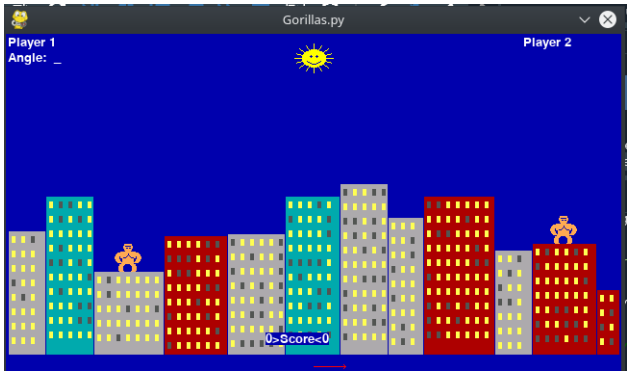


Figure 2.1: Capture d'écran du portage du jeu de calculatrice mentionné lors de la rédaction de la présentation du projet.

2.2 Les membres de l'équipe

Je suis l'auteur de la rédaction du projet initial. Je suis donc naturellement devenu sous l'impulsion de *Ze BigBoss* le chef de projet. Mes deux coéquipiers sont venus ensuite me rejoindre

- Olivier Torrens : je suis enseignant de sciences physiques dans un lycée de Périgueux. J'accueille des élèves de la classe de seconde à la classe de Terminale, mais j'interviens aussi en CPGE pour faire des colles de physique aux étudiants de première et deuxième années qui préparent les concours des grandes écoles. J'ai aussi vu l'apparition ces deux dernières années de l'enseignement de l'informatique au lycée, d'abord en seconde avec les S.N.T. (Sciences Numériques et Technologiques) puis en N.S.I. (Numérique Sciences de l'Informatique) en première et terminale.
- Anaëlle Aziza :
- Jean-Christophe Maceron :

2.3 L'ambiance du jeu

Plus ou moins consciemment, j'ai voulu m'inspiré du jeux vidéo Worms auquel j'ai également beaucoup joué avec des copains lors d'après-midis mémorables. J'ai donc initialement pensé à faire une ambiance du jeu que je voulais créer proche de celle-ci. A savoir des graphismes dans le style cartoon . Historiquement le décor pouvait être détruit par les explosions et les diverses armes de Worms, comme d'ailleurs les bâtiments étaient détruits dans Gorilla. Mais il m'est apparu plus simple d'avoir un décor statique dans le jeu que j'ai développé. J'ai toutefois essayé de garder l'esprit de Worms en créant un décor coloré et des personnages cartoonesques, une forme d'hommage à toute une génération de jeux et de joueurs.

2.4 Les préconceptions techniques

Je pars totalement de zéro dans le développement de jeux vidéos. Je ne sais pas pour quelle raison, mais je pensais qu'à chaque nouveau jeu l'équipe de développeurs, du graphisme au

scénario en passant par le décor et la musique, devait recréer un jeu *ex nihilo*. Hors je me suis très rapidement aperçu qu'en réalité un jeu fonctionne avec un moteur de jeu, nul besoin de réinventer systématiquement la roue. Il s'agit d'un logiciel ou d'un ensemble de logiciels autour duquel s'articule le jeu. C'est le moteur de jeu qui gère toutes les problématiques informatiques liés au monde virtuel dans lequel se déroule le jeu et qui lui donne vie, libérant ainsi du temps aux concepteurs du jeu pour se consacrer au contenu.
Quelques exemples pour illustrer mes propos, certes un peu naïfs, mais qui décrivent le rôle du moteur de jeu :

- pour l'animation liée au déplacement d'un personnage, le graphiste va créer des textures en nombre plus ou moins grand selon les détails qu'il veut apporter. Ces textures seront ensuite appliquées à des zones de l'espace correspondant aux différentes positions du personnage. Le "jeu" se contentant alors de récupérer d'afficher les différentes textures dans un sens ou dans l'autre suivant après l'interception des touches du clavier qui gèrent les directions du déplacement ;

- pour la physique du jeu, par exemple la gravité, c'est en temps réel que le moteur de jeu calcul l'effet de la gravité virtuelle sur les différents objets du jeu ;

- le fait que le personnage peut ou non passer à travers le décor est lui aussi gérer par le moteur de jeu. Des zones de l'espace sont définies comme décor, puis le moteur de jeu récupère des collisions entre ces zones et celles propres au joueur et calcule l'interaction qui en résulte. Simplement du laisse passer / ne laisse pas passer, ou bien alors une interaction plus complexe du type déformation ;

- l'éclairage et les jeux de lumière sont également gérés par le moteur de jeu. Le développeur choisi et calibre son ambiance, mais les ombres et les parties claires ou obscures sont déterminées en fonction des *desiderata* et des paramètres choisis par le développeur du jeu. En somme, le développeur choisi la couleur et la place de la source lumineuse, le moteur de jeu en calcul le rendu graphique en temps réel.

On comprend ainsi aisément pourquoi les jeux vidéos modernes, de plus en plus détaillés, qui plus en est en trois dimensions, demandent des cartes graphiques de plus en plus puissantes qui calculent à chaque instant les paramètres aussi bien physique que d'animation d'une scène et de tous les objets qui la composent. Un des exemples majeurs pour moi est le jeu Flight Simulator 2020[1] où le moteur de jeu calcule toute la physique du vol, de la machine à l'environnement, affiche tous ces paramètres à l'écran, le tout sur des décors dont certains sont en photo-réalisme 4K . Ajouté au fait que le jeu récupère en temps réel les informations du trafic aérien et météorologiques, le moteur de jeu, et vraisemblablement les moteurs, envoient des quantités de calculs phénoménales aux processeurs central et graphique.

Ainsi les considérations techniques inhérentes au développement d'un jeu sont, un peu caricaturalement, plus celles liées au contenu du jeu comme les graphismes ou le scénario, plutôt que celles de la partie technique qui va permettre à ce monde virtuel de prendre vie. Il y a donc des développeurs de moteur de jeu qui codent la physique ou l'intelligence artificielle du jeu et les développeurs du jeu qui finalement sont des créateurs de contenus. C'est finalement le même principe, si j'ose la comparaison, que les livres l'histoire qu'ils racontent est bien souvent créée par une personne totalement différente de celle qui gère les

[1] Et il vient de remporter plusieurs récompenses comme celui du meilleur jeu de l'année

rotatives d'imprimerie qui en assurent la création. Chacune de ses personnes ayant au final des problématiques qui lui sont propres.

2.5 Des ressources libres en ligne

Souvent en informatique, à un problème ou un objectif donné, il existe des solutions payantes et propriétaires et des solutions libres et gratuites. L'univers des moteurs de jeu de jeu ne fait bien évidement pas exception à cette règle. Il en existe énormément aux spécificités toutes différentes. Pour des raisons de coup évidente, mais aussi sur les conseils de *Ze Big Boss* je me suis tourné vers le moteur de jeu nommé Godot. D'une part pour sa praticité, il est multi-plateforme et fonctionne aussi bien sous Linux, Windows ou MacOS. Il ne nécessite même pas d'installation car il se présente sous la forme d'un exécutable. D'autre part, il possède un langage de programmation dont la syntaxe est assez proche de celle de Python, avec la même gestion de l'importance de l'indentation pour la mise en forme du code, ce qui a facilité sa prise en main.

Il est disponible à l'adresse suivante : `https://godotengine.org/`

- Cet article de blog recense une multitude de liens hypertextes renvoient vers des sites web qui hébergent des contenus libre de d'utilisation. J'ai donc suivi un des liens pour arriver au point suivant.

  `https://blog.felgo.com/game-resources/16-sites-featuring-free-game-graphics`

- Dans ce site, j'ai trouvé un fichier dont les graphismes me plaisaient et correspondaient à ce dont j'avais envie pour créer l'univers du jeu. J'ai donc choisi le fichier à l'adresse ci-après dont j'ai extrait des parties grâce à Gimp. Je les ai ensuite intégrées au projet directement sous Godot.

  `https://opengameart.org/content/donkey-kong-style-platform-set`

- Il existe aussi des sites de bibilitohèques de sons, aaussi bien des bruitages qu'aux bandes sonores de plusieurs minutes ce qui enrichit l'expérience d'utilisateur via l'interface multimédia du jeu.

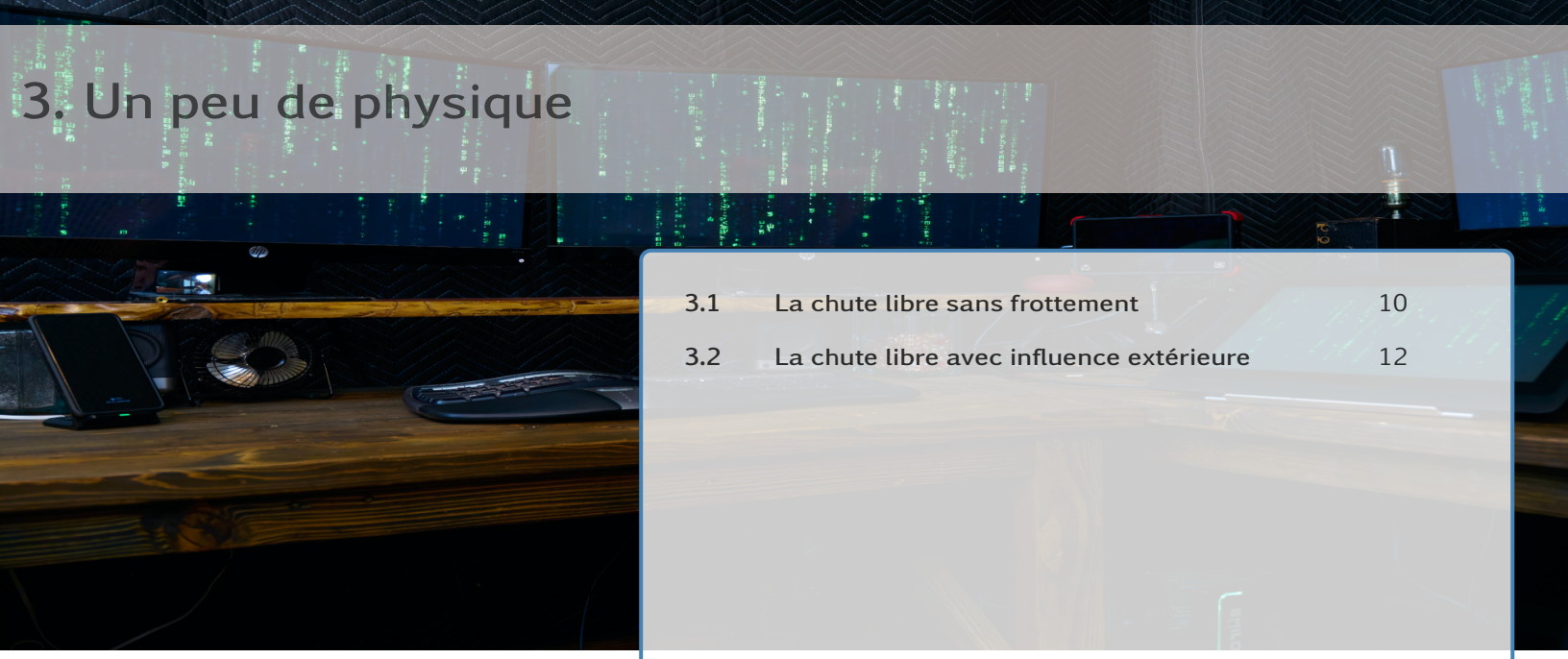
  `https://jeux.developpez.com/medias/#LVI`

- Enfin, les quelques images qui décorent ce document sont elles aussi issues de bibliothèques libres qui sont trouvables sur le site web ci-après.

  `https://unsplash.com/s/photos/jungle`

Nous nous retrouvons ainsi avec tous les éléments nécessaires pour faire un jeu, le moteur de jeu, les graphismes et l'ambiance autour du jeu.

3. Un peu de physique



Quelques points sur la physique de la chute libre et des trajectoires balistiques, ce que le moteur de jeu permet de calculer de façon transparente pour déterminer la trajectoire de la banane et la gestion des collisions.

3.1 La chute libre sans frottement

Par définition, un mouvement de chute libre est un mouvement pour lequel le système mécanique qui se déplace n'est soumis qu'à l'effet de son poids, modélisé par un vecteur $\vec{P}$.

Ainsi en appliquant la 2ème loi de Newton à notre système, pour nous c'est la banane-projectile, nous obtenons la relation vectorielle dans un référentiel galiléen, pour nous la fenêtre du jeu :

$$m_{inerte}\vec{a} = m_{grave}\vec{g}$$

Où

- le vecteur accélération et $\vec{a}$ est le vecteur modélisant l'accélération de la banane ;
- le vecteur accélération et $\vec{g}$ le vecteur modélisant le champ de pesanteur local ;
- m_{inerte} est la masse inerte de la banane c'est-à-dire sa résistance intrinsèque à être mis en mouvement ;
- m_{grave} est la masse grave de la banane $\vec{a}$ c'est-à-dire sa caractéristique à interagir avec le champ de pesanteur local.

Un des piliers de la physique, proche du dogme est l'égalité entre masse inerte et masse grave. Cette égalité est connu sous le principe d'équivalence. Des dizaines d'expériences toujours plus raffinés ont montrés cette égalité jusqu'à une précision relative de l'ordre de 10^{15}. Si jamais cette égalité n'était plus vérifiée, quelques théories deviendraient alors caduques comme celle de la relativité générale dont le principe d'équivalence est le point de départ car central ou bien encore la mécanique quantique, dont le modèle actuel le plus sophistiqué et

complet est la théorie quantique des champs, théorie basée sur ... le principe d'équivalence.

Ainsi, eu égard au principe d'équivalence, le PFD devient simplement :

$$\vec{a} = \vec{g}$$

En utilisant les coordonnées (X,Y) du centre de gravité de la banane quand elle est affichée à l'intérieur d'une fenêtre d'affichage 2D. L'égalité vectorielle précédente impose les égalités scalaires suivantes :

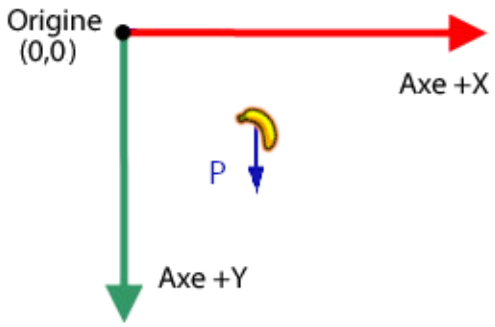


Figure 3.1: Les axes représentent le système de coordonnés 2D d'une fenêtre graphique. La flèche en bleu représente le vecteur $\vec{P}$ modélisant l'effet du poids sur la banane

On obtient ainsi que :

$$\begin{cases} \ddot{X}(t) &= 0 \\ \ddot{Y}(t) &= g \end{cases}$$

Accélération, vitesse et coordonnées sont des fonctions paramétriques du temps toutes reliées entre elles. L'accélération est la dérivée de la vitesse, elle-même la dérivée de la position. Le moteur de jeu a besoin de connaître le vecteur vitesse, comme nous le verrons dans la partie suivant, le moteur physique calcule à chaque nouvelle image (*frame*) la nouvelle orientation et la nouvelle norme de ce vecteur.

Nous obtenons donc le vecteur vitesse en fonction de la vitesse initiale v_0 et de l'angle qu'il fait avec l'axe X. Attention, sous le moteur de jeu les angles sont orientés et sont en quelques sortes en *sens inverse* du sens trigonométrique auquel nous sommes habitués, en effet l'axe Y n'est pas orienté comme le classique axe des ordonnées, et comme $\sin(-\alpha) = \sin\alpha$

$$\begin{cases} v_x(t) &= v_0 cos\alpha \\ v_x(t) &= gt + v_o sin\alpha \end{cases}$$

Pour le dire autrement, à chaque instant qui sera une image pour le moteur de jeu, le vecteur vitesse initiale $\vec{v_0} = \begin{pmatrix} v_0 \cos\alpha \\ v_0 \sin\alpha \end{pmatrix}$ est affecté d'une correction g au cours du temps qui n'est n'est

plus ni mois que l'intensité du champ de pesanteur local, autrement appelée pour simplifier intensité de la gravité. C'est ainsi que la physique de la balistique de la banane sera codée sous Godot par la suite. Cela correspondant au développement du projet dans sa phase dite *alpha*.

Enfin, ce modèle permet de se rendre compte que seulement deux paramètres règlent une fois pour toute la trajectoire de la banane une fois tirée, la vitesse initiale et l'angle de tir par rapport aux axes de la fenêtre. Dans la suite du développement du projet, la vitesse initiale de la banane, vrai paramètre physique, sera improprement nommé *puissance du tir*. En effet, il est plus logique pour un joueur de régler la *puissance* de son lanceur plutôt que de régler la *vitesse initiale* avec laquelle la banane sera envoyée. De même avec des considérations similaires, il sera plus commode de paramétrer un angle de tir entre 0 et 90 degré alors qu'en réalité, dans le système de coordonnée des l'affichage, c'est un angle compris entre 0 et $-\frac{\pi}{2}$ radian.

3.2 La chute libre avec influence extérieure

Ce titre est partiellement erroné puisque la définition d'une chute libre est que le système n'est soumis qu'à l'influence de son poids.
Pour autant, l'idée étant d'ajouter au mouvement de chute libre une perturbation externe au système nommé *vent*. Concrètement, au mouvement précédent, il suffit d'ajouter une contribution extérieure au vecteur vitesse de la banane, puisque le moteur physique du jeu ne s'intéresse qu'à cette donnée pour générer le déplacement de la banane.

Mon idée a donc été de tirer au sort des coordonnées d'un vecteur entre deux bornes minimales et maximales. Ce tirage au sort se déroule après chaque tir de banane, ainsi, le tir d'avant est toujours un peu différent du tir d'après à l'exception peu probable de garder tous les paramètres du tir constant, à savoir l'angle de tir la puissance du tir et exactement le même tirage aléatoire pour les coordonnées du vecteur vent.

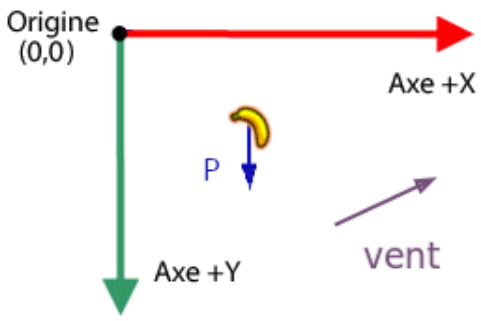


Figure 3.2: Ajout d'un vecteur vent pour modéliser les effets d'éléments extérieurs à la banane.

Pour les équations horaires de la trajectoire nous obtenons donc avec cette approche :

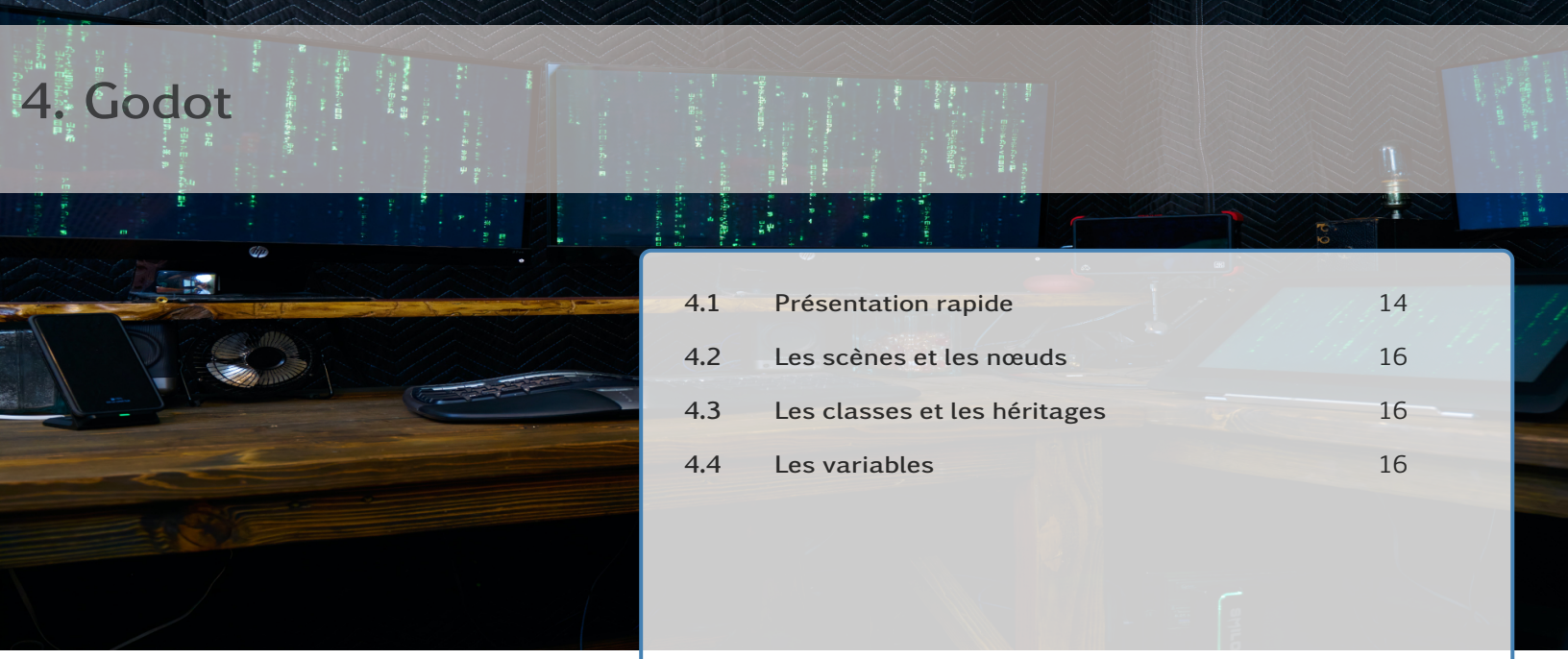
$$\begin{cases} v_x(t) &= v_0 cos\alpha + vent_x \\ v_x(t) &= gt + v_o sin\alpha + vent_y \end{cases}$$

Où le vent est modélisée par son vecteur vitesse de coordonnées $\overrightarrow{vent} = \begin{pmatrix} vent_x \\ vent_y \end{pmatrix}$.

La norme du vecteur vent devient la *vitesse du vent* et l'angle que fait le vent avec las deux axes de la fenêtre graphique devient la *direction du vent*.

Voyons dans les parties suivantes comment nous allons implémenter cela dans le moteur de jeu Godot.

4. Godot



En guise d'introduction, voilà ce qui est marqué sur la page d'accueil de godot :

`https://godotengine.org/`

The game engine you waited for.

Godot provides a huge set of common tools, so you can just focus on making your game without reinventing the wheel.

Godot is completely free and open-source under the very permissive MIT license. No strings attached, no royalties, nothing. Your game is yours, down to the last line of engine code.

4.1 Présentation rapide

Godot est un moteur de jeu multiplateforme. Sa première version disponible date de 2007 et en 2014 il est devenu complètement libre. Dans l'esprit du logiciel libre, il est développé et maintenu par des contributeurs bénévoles, seule une poignée de contributeurs est rémunérée.

Il est développé en `C` et en `C++`. Godot possède son propre langage de Programmation le `GDScript`. Ce langage a une syntaxe proche de celle de `Python` et en possède quelques caractéristiques. La programmation fonctionnelle est y omniprésente et l'indentation joue un rôle fondamental. Le typage des variables n'est pas dynamique. La gestion des allocations mémoire et du ramasse-miette est totalement transparente et gérée par Godot, de fait il y a beaucoup moins de difficultés techniques à s'approprier ce nouveau langage dont la relative simplicité permet en quelques heures de test de commencer d'obtenir quelques résultats techniques. Notons que Godot permet aussi une programmation en `C++` qui permet de réaliser de la programmation de plus bas niveau dans le but d'optimiser les performance *In-Game* lors des projets lourds. Godot permet de mêler dans le même projet `C++` et `GDScript` ce qui permet d'allier simplicité et fonctionnalités à la demande. Enfin il est possible de faire une partie du développement en `C #` (prononcer à l'anglaise *C sharp*, langage dérivé du du `C++`, oritenté objet et développé par Microsoft.

La facilité d'appropriation de Godot est d'autant plus facilité que la partie graphique permet de faire du glisser-déposer des objets à intégrer au jeu. L'interface utilisateur à ce niveau-là est revendiquée WYSIWYG (*what you see is what you get*) ce qui permet là encore à

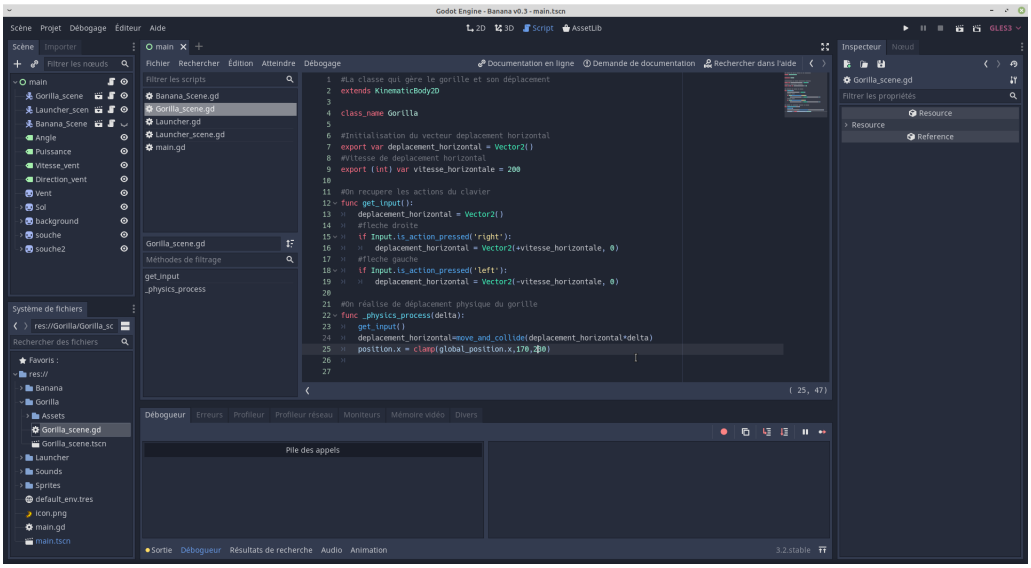


Figure 4.1: Godot avec l'affichage de l'environnement de développement (IDE) ici en GDScript

des débutants en graphisme d'obtenir des résultats assez rapidement. Ainsi toute la gestion des lumières, caméras, positionnement des objets de décor ou d'animation se fait directement. De plus, la communauté Godot propose une bibliothèque conséquente d'outils pour le développement, aussi bien des les objets graphiques que dans des classes de Programmation déjà réalisées dans d'autres projets qui ont hérités de la licence libre de Godot à condition que ses développeurs l'aient autorisé.

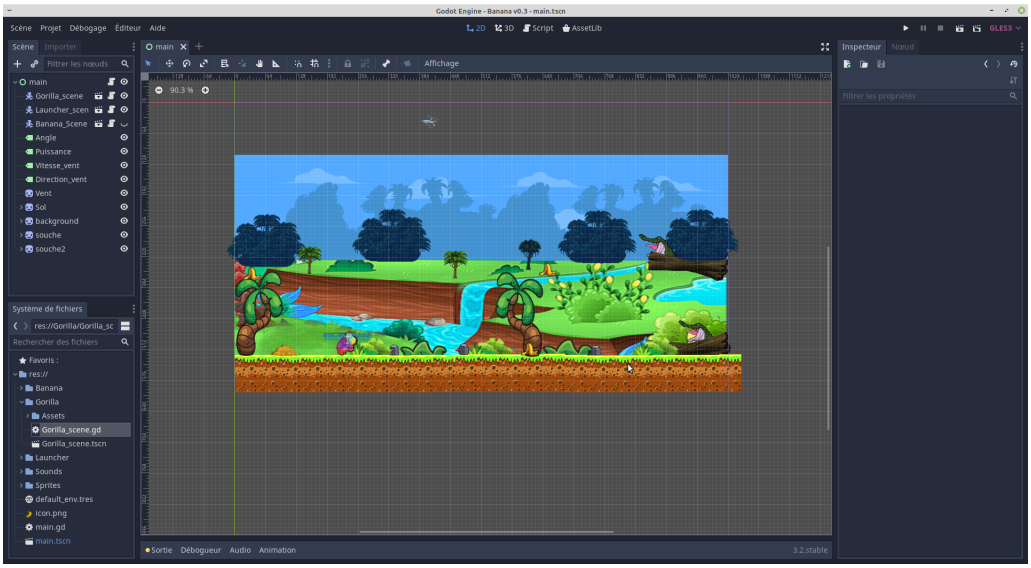


Figure 4.2: Godot avec l'affichage de la gestion WYSIWYG des scènes, ici en 2D

4.2 Les scènes et les nœuds

Godot tire son nom de la pièce de théâtre *En attendant Godot* de *Samuel Beckett* publiée en 1952. La particularité de ce livre vient du fait que le nombre de scènes n'est ni décompté ni annoncé.

Le logiciel de développement s'appelle ainsi d'une part car la gestion des projets se fait par scène (avec ce vocable là, y compris en anglais) qui s'imbriquent les unes dans les autres. Tout ceci n'apparaît pas aux yeux de l'utilisateur qui ne voit que le résultat final. Juan Linietsky, un des créateurs argentins du moteur, justifie *a posteriori* le choix de garder officiellement ce nom. En effet, les personnages de la pièce attendent en permanence un individu nommé Godot sans que personne ne sache s'il viendra effectivement un jour. Juan compare ces personnages aux utilisateurs de Godot, qui sont en permanence dans l'attente de nouvelles fonctionnalités pour le moteur. (`https://fr.wikipedia.org/wiki/Godot_(moteur_de_jeu)#Choix_du_nom`)

A l'intérieur d'une scène on peut placer autant de nœuds que voulu. Un nœud est un objet qui peut être un sprite (*lutin en français*), un objet qui se déplace à l'écran, une source de lumière, un objet physique déformable ou non, qui peut générer une collision ou non..

Il est aussi possible d'utiliser en tant que nœuds dans une autre scène une scène annexe. Ce système d'arbre de scène permet ainsi de construire un projet avec une scène principale qui contient les scènes des personnages ou des animations ainsi que la gestion de leurs interactions, de leur mouvement ou de leur physique spécifique.

Dans l'interface graphique, cette gestion de l'arbre de scène est encore graphique ce qui permet de visualiser les liens de parenté ou d'héritage. Surtout il permet de re-parenter (terme employé sous Godot) un sous-nœud à un autre nœud de la scène. Fonctionnalité très utile dans la création d'un décor par exemple pour gérer les apparitions devant/derrière d'un texture, ou alors de déplacer massivement un ensemble de texture.

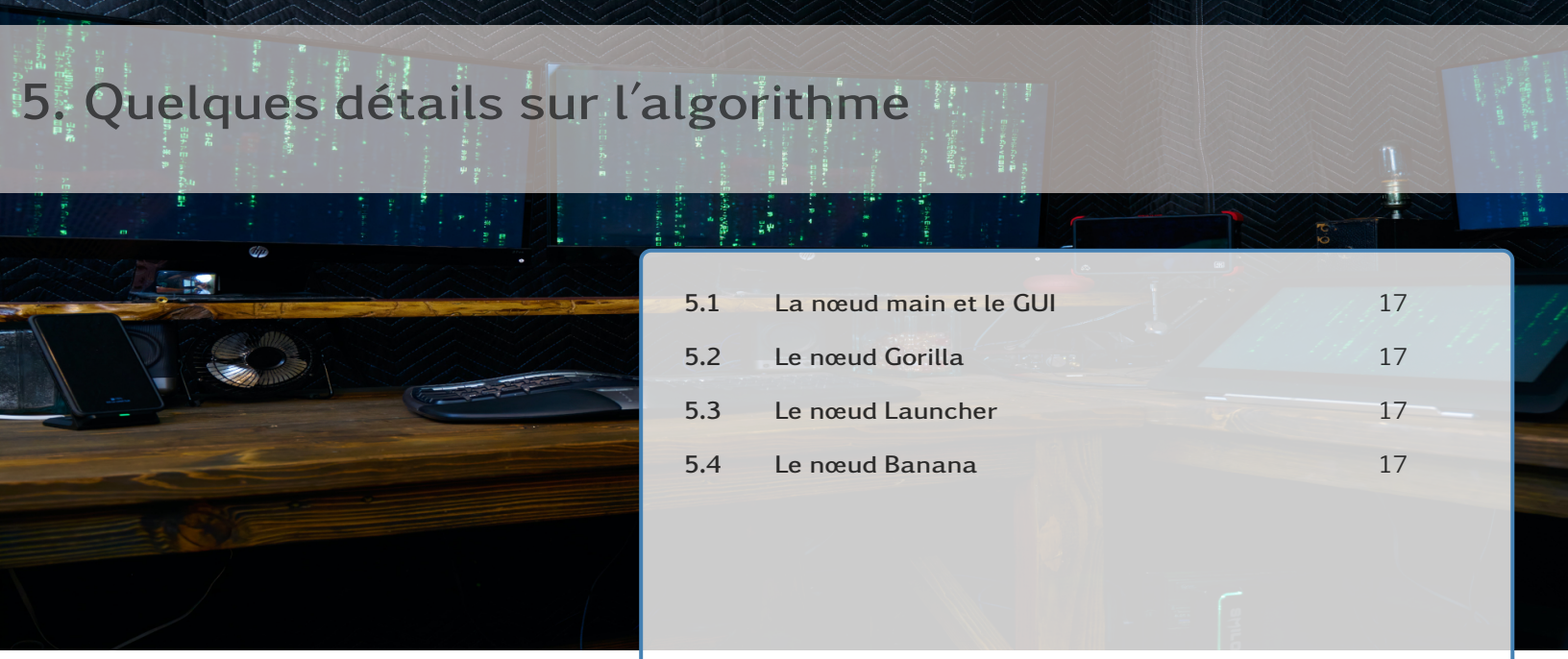
4.3 Les classes et les héritages

Les classes pré-intégrés, les méthodes dont elles héritent et les extensions liées au développement spécifique du projet.

4.4 Les variables

Si dans le script la variable à été déclaré en export, l'inspecteur permet de modifier la valeur en live d'une variable pour en voir l'incidence directement dans le gameplay. Très utile pour gérer les vitesses de déplacement, rotation ou l'intensité de la gravité par exemple.

5. Quelques détails sur l'algorithme



Quelques points techniques spécifiques au projet.

5.1 La nœud main et le GUI

5.2 Le nœud Gorilla

5.3 Le nœud Launcher

5.4 Le nœud Banana

6. A propos de cette documentation Latex

La commande explicite de Ze Big Boss *pour réaliser cette documentation était de la faire en utilisant le langage LaTeX.*

6.1 Les templates

Il y en a à foison sur le net.

6.2 Le découpage

Dans un premier temps j'ai voulu faire un fichier tex par chapitre, puis les appeler un par un dans le document principal.

7. Conclusion

Bibliographie

Cette vilaine ne veut pas s'intégrer pour l'instant...