Rapport de projet

Université de Bordeaux - Projet de fin d'étude

« Amélioration des graphismes et des performances du prototype Dana »

Février - Mars 2025

Master Informatique, Spécialité Image et Son

Encadrant : Aymeric FERRON

Étudiants : Lucas BROUET
Timothé PAOLINI
Nicolas MOULIN

Table des matières

1 Introduction

Ce rapport décrit le travail effectué durant le *Projet de Fin d'étude*, un des deux modules du semestre 10 du master Informatique de l'université de Bordeaux. Ce projet reprend l'application **Dana** de Aymeric Ferron, équipe BIWVAC du centre de recherche Inria de l'université de Bordeaux. Les deux mois de février et mars 2025 sont consacrés à la réalisation de ce projet.

2 Contexte

2.1 Contexte général

Le changement climatique, ainsi que ses conséquences, est un problème qui importe à la communauté scientifique depuis un certain nombre d'années. Pour cause, le groupe d'experts intergouvernemental sur l'évolution du climat (GIEC) a publié de nombreux rapports expliquant le problème dans son ensemble. Le dernier en date expose concrètement les causes et les conséquences de ce dernier. [5]

Face à la nécessité de lutter contre le changement climatique, plusieurs formes d'action s'imposent : **l'action individuelle** et **l'action collective**. La première repose sur l'adoption de modes de vie plus durables afin de réduire son empreinte carbone, tandis que la seconde englobe les mesures prises par l'État et les entreprises. Toutefois, une réduction efficace des émissions ne peut être atteinte qu'avec une implication équilibrée de ces trois acteurs.

Ce défi est cependant freiné par un phénomène connu sous le nom de triangle de l'inaction : les individus estiment que leur impact est négligeable sans des mesures structurantes des entreprises et des gouvernements ; ces derniers, à leur tour, attendent une demande citoyenne claire avant d'agir ; et les entreprises privilégient souvent la rentabilité immédiate en l'absence de régulations incitatives. Ce cercle vicieux ralentit les progrès, alors même que des études montrent qu'un engagement massif des individus pourrait réduire leur empreinte carbone personnelle de moitié par rapport à la moyenne française . Sortir de cette inertie nécessite donc une prise de responsabilité conjointe, où chaque acteur stimule et soutient les autres dans une dynamique vertueuse. En effet, si tous les individus exerçaient un effort exemplaire concernant leur empreinte carbone, ils pourraient réduire leur empreinte personnelle de moitié (en comparaison à la moyenne française). [3]

C'est alors qu'on est en mesure de se poser la question : pourquoi n'y a-t-il pas de passage à l'acte de la part de la population ? Plusieurs réponses peuvent être apportées à cette problématique.
On a d'abord une **distance psychologique**, c'est à dire que plus le problème est éloigné par rapport à l'individu, moins il sera enclin à se saisir du problème. Cette "distance" peut

être spatiale, temporelle, sociale ainsi que le coté hypothétique de l'objet. [2].
Une autre raison est la **complexité** des données qui est inhérente au contexte scientifique. Une personne non avertie n'a pas la possibilité de comprendre la signification d'un chiffre précisément. C'est pour cela qu'il est pertinent d'adapter ces différentes données en *ordre de grandeur*, que l'on pourrait situer plus facilement. C'est dans ce contexte qu'entrerait en jeu la visualisation de données.

Quant à la question de l'urgence d'agir, cela n'est plus à prouver comme l'indique le GIEC, car chaque tonne de CO2 dans l'atmosphère contribue un peu plus au réchauffement de l'atmosphère. Parmi ces différents leviers d'action, on retrouve celui de consommer végétarien en abandonnant la viande : cela représente une diminution de 1.3 tonne de CO2 par personne et par an, ce qui n'est pas négligeable. [3] Mais un régime végétarien manque d'acceptabilité sociale [6], il y a donc un réel intérêt à faire de la pédagogie autour de l'impact environnemental de la consommation de nourriture.

2.2 Contexte du projet existant

C'est dans ce contexte qu'a été développée l'application **Dana**.[4]

Dana est une application permettant de visualiser l'impact environnemental sur trois limites planétaires (le changement climatique, l'utilisation de l'eau douce et la modification de l'usage des sols) à travers nos habitudes alimentaires. Cette application nous permet d'explorer un mini-monde interactif à travers un menu. Ce dernier permet de spécifier le pourcentage d'aliment et même le repas consommé pour chaque jour de la semaine. Les éléments qui composent ce repas seront alors convertit en chiffre permettant de quantifier les GES, ainsi que consommation d'eau et usage des sols. À partir de ces informations, on pourra potentiellement observer une montée des eaux, une fonte des glaces ainsi qu'un épuisement des ressources en eau. Ce projet a été mené afin de pouvoir sensibiliser de manière plus concrète et plus pédagogique sur l'impact environnemental en permettant à l'utilisateur d'explorer les différentes possibilités, tout en ayant des explications avec une visualisation plus concrète.

2.2.1 Mini monde 3D

C'est le moteur de jeu Unity 3D qui a été utilisé, ainsi que l'IDE Rider.

Les graphismes de l'application ont été réalisés grâce à l'application Blender. Toute la structure a été réalisée d'un bloc. Des éléments de paysage ont été ajoutés, à savoir : des montagnes, des glaciers, une rivière qui se jette dans l'océan, une petite île, et de manière générale du dénivelé pour simuler une montée des eaux. Certains assets ont été ajoutés également, comme des maisons, des arbres ainsi que des champs de blé.

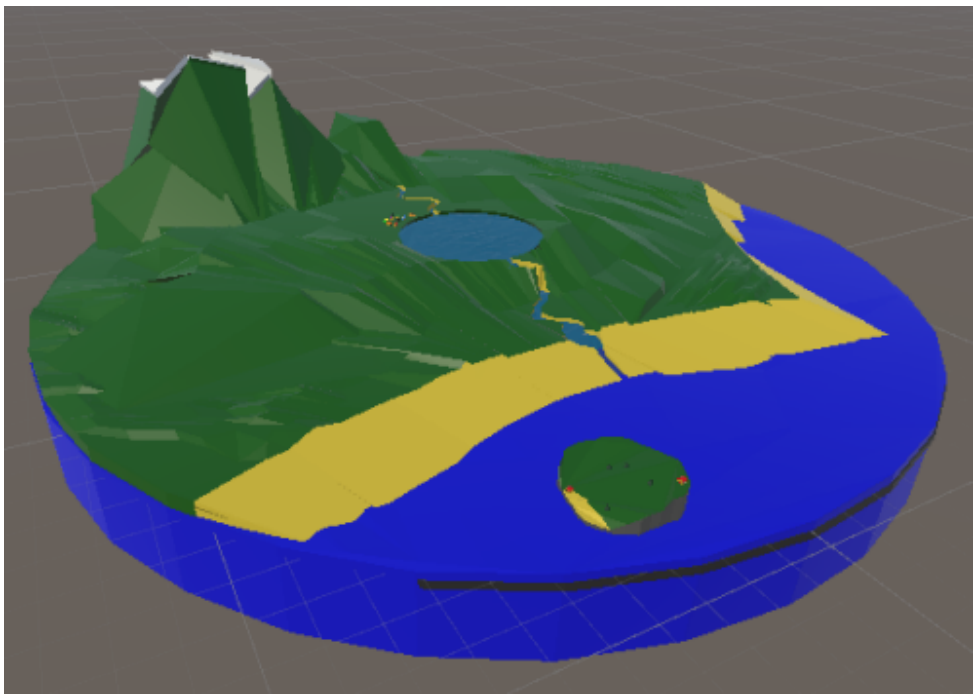


FIGURE 1 – Mini monde de Dana

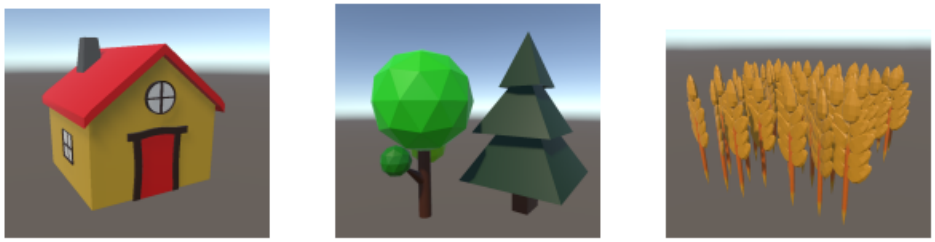


FIGURE 2 – Les assets

2.2.2 Les données

L'application permet de simuler l'impact environnemental d'une alimentation choisie. On peut ainsi sélectionner les différents types / catégories d'aliments que l'on consomme au cours d'une semaine. Cela modifiera le calcul des GES émit avec la consommation de ces aliments. La simulation sera alors impactée en par ces chiffres, et offrira une représentation du mini monde avec le taux de GES calculé. On verra alors un monde dont le climat et plus ou moins boulversé en fonction du taux de GES. Pour mesurer ces changements, les données utilisées sont celles d'**Agribalyse**. C'est une base de données développée par l'Agence environnementale et de la maîtrise de l'énergie (ADEME) et l'Institut national de la recherche agronomique (INRAE). [1]

3 Objectif

L'objectif de notre projet se concentre la refonte des graphismes, tout en maintenant les FPS stables.

FIGURE 3 – Choix des types de repas

3.1 Refonte des graphismes

Actuellement, les graphismes sont jugés minimalistes par le réalisateur du projet Dana, Aymeric Ferron. On peut citer les différents éléments manquent de réalisme quant à l'évolution du climat. Par exemple, l'eau, en cas d'alimentation émettrice de CO2, augmente de manière uniforme, ce qui ne représente pas vraiment la réalité. C'est le cas de la glace qui disparaît avec une direction uniforme. On peut aussi citer l'exemple des maisons encore présentes dans toute leur intégrité malgré leur submersion.

Notre mission concerne donc la refonte de ces graphismes avec 2 modes :

Le mode **photo-réaliste** qui comporterait des textures proches de la réalité. Cela se rapproche des textures que l'on pourrait avoir dans un jeu vidéo ou un simulateur.

Le mode **Cartoon** qui se rapprocherait plus des textures d'un film d'animation, ou d'un dessin animé (voir un jeu vidéo aux caractéristiques cartoon également).

3.2 Maintient des performances

Bien que l'on puisse considéré cette tâche comme annexe et secondaire, elle reste primordiale puisque l'objectif est également de permettre à l'application de pouvoir être utilisée sur toute sorte de PC.

Nous devons ainsi faire en sorte de maintenir un niveau de *FPS* acceptable.

4 Existant

Afin de réaliser l'application, la question de la méthode utilisée mérite réflexion. Voici donc un panorama des méthodes / projets existants concernant la simulation d'un mini-

monde dont l'environnement évolue.

4.1 Concept de simulation

Lorsqu'on parle de simulation d'écosystème, on retrouve plusieurs formats que l'on considère comme de la simulation. D'un coté une simple simulation de donnée, ou on expose des chiffres et leur signification, ainsi que des graphiques qui vont avec. D'un autre coté, il existe des simulations de mondes, ou l'on cherche à modéliser des phénomènes naturels.

4.1.1 Simulation de données

Certains modèles comportent uniquement des données chiffrées, relatant l'état du climat avec des indicateurs comme le CO2, des chiffres sur l'économie, la biodiversité et d'autres encore. Cela s'avère moins intéressant dans notre cas, car nous souhaitons une visualisation des conséquences du dérèglement climatique.

Il arrive que certains modèles soient représentés graphiquement à partir de ces données. Il s'agit en fait la plupart du temps de simulations à très grande échelle comme la Terre entière, représentée sous forme de carte et non sous forme de mini-monde.

4.1.2 Simulation de monde

La description de ce projet correspond beaucoup aux programmes d'écosystème. Mais la quasi-totalité de ces simulations se concentrent sur l'évolution des créatures organiques (et plus particulièrement, les animaux ou êtres ressemblants). Quasiment aucun ne se concentre ou inclue les conséquences d'un écosystème sur le terrain. Donc pas de dégradation des sols, pas de réchauffement climatique, pas de montée des eaux ni d'assèchement.

Afin de réaliser cela, il nous faut nous approcher de la réalisation de terrain.

4.2 Réalisation de terrain

Il y plusieurs cas pour lesquels réaliser un terrain s'avère particulièrement intéressant. Bien souvent, on retrouve se concept dans les jeux vidéos. On peut ainsi modélisé ce terrain pour augmenter au maximum l'immersion du joueur. Dans notre cas, on cherche à conserver l'immersion, mais d'un point de vu extérieur, l'utilisateur n'étant que spectateur. On relève alors plusieurs outils pour créer nos terrains.

4.2.1 Blender

A l'origine la structure du mini monde de dana a été réalisée grâce Blender. Ce logiciel a l'avantage d'être un outil de modélisation puissant et possède de nombreuses possibilités d'animation. Nous pourrions donc l'utiliser à nouveau et améliorer le design de la carte avec des modules comme ANT Landscape [1] ou GAEA [2]. Cependant, ce n'est pas le meilleur

1. https ://docs.blender.org/manual/fr/2.92/addons/add_mesh/ant_landscape.html
2. https ://docs.quadspinner.com/

outil pour réaliser un mini monde 3D **en évolution**. En effet, ces derniers forment de paysages tout à fait convenable, mais assez statiques. De notre côté, nous souhaitons un paysage flexible qui peut évoluer sans complexité majeure.

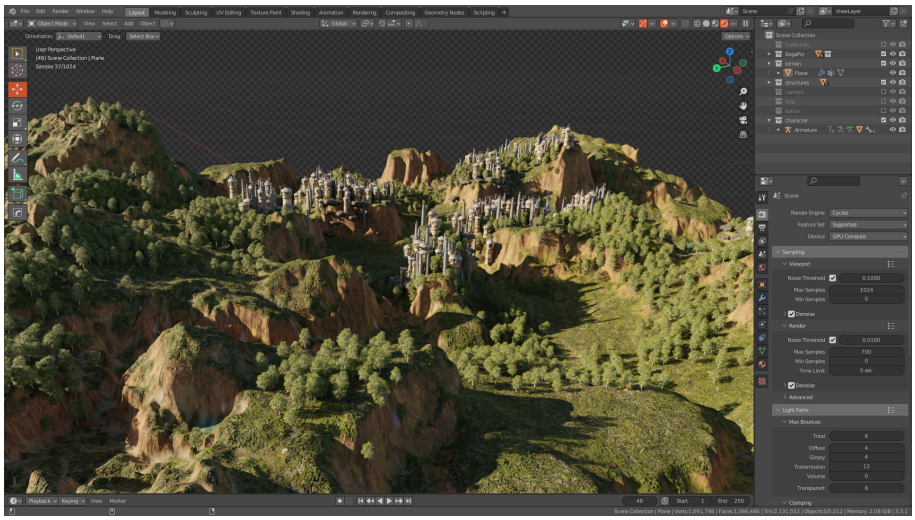


FIGURE 4 – Terrain avec *Blender*

4.2.2 Unity Terrain

L'éditeur de terrain d'Unity 3D [3] est un outil intéressant. Il permet la réalisation d'un terrain dans le cadre d'un jeu par exemple. Unity Terrain peut gérer des terrains de très grande taille, et son utilisation est majoritairement pour des paysages naturels comme des prairies ou des montagnes.

L'inconvénient majeur de Unity Terrain est tout comme Blender la complexité de modification de ces terrains. L'utilisation de l'outil est souvent dans un contexte de jeu vidéo, où l'on souhaite créer un terrain immense dans lequel le joueur pourrait se déplacer. Mais à nouveau, nous souhaitons une modification profonde du paysage et il serait complexe de modifier des terrains tout entier avec Unity 3D.

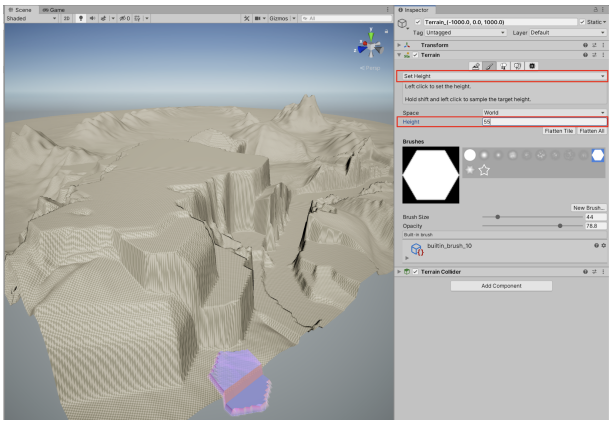


FIGURE 5 – Unity Terrain

3. https ://docs.unity3d.com/Manual/script-Terrain.html

4.2.3 Les tuiles

On peut retrouver le système de tuile dans de multiples projets, mais le meilleur domaine pour en parler reste celui du jeu vidéo. Les jeux basés sur un système de tuiles sont des types de jeu dont l'environnement est constitué de formes régulières, formant une grille. Ce format a l'avantage d'avoir un système simple de gestion divisé. On peut attribuer des propriétés ainsi que des caractéristiques pour chaque case, ce qui permet d'avoir une vue claire sur l'état de chacune d'entre elles. La forme des grilles peut prendre plusieurs formes, à savoir **carrée**, **hexagonale**, ou moins souvent **rectangulaire** ou **parallélogramme**. Le choix du format de la tuile est régulièrement adapté en fonction du contexte.

Le format carré

Le format carré est un format qui se veut simple, efficace. Il est plutôt simple de gérer un système de grille avec les 4 directions orthogonales. C'est pourquoi on le retrouve très souvent dans les jeux, anciens comme plus récents. Parmi les jeux utilisant des tuiles carrées, on en retrouve de nombreux exemples, ce format étant le plus répandu au sein du répertoire.

FIGURE 6 – Exemple de jeu utilisant des tuiles carrées : à gauche *Civilization 4* et à droite *Into The Beach*

Le format hexagonal

Le système de tuiles hexagonales quant à lui est de plus en plus populaire et ainsi utilisé dans les jeux. On peut le reconnaître plus pratique pour le déplacement car il permet de se déplacer plus librement autour de nous, et de se séparer de la notion de "diagonal" qui peut être contraignant. C'est une méthode qui est plus agréable pour représenter des terrains également. On peut donc le considérer comme plus complexe que le système carré, mais plus intéressant pour représenter un monde réaliste.

5 Réalisation

Nous avons commencé en reprenant la base du projet Dana, avec un objectif dès le départ, à savoir **une amélioration graphique pour le mini monde**. Actuellement, la structure du monde a été réalisée via Blender mais n'est pas jugée satisfaisante dans le

FIGURE 7 – Exemple de jeu utilisant des tuiles hexagonales : à gauche *Civilization 6* et à droite *The Battle for Wesnoth*

rendu visuel par son réalisateur. Notre mission est donc d'améliorer la qualité des graphismes de l'application. De surcroît, nous devrons améliorer les performances globales autant que possible. Afin de réaliser cette tâche nous avons décomposé le projet en plusieurs étapes. Dans un premier temps, nous implémentons le mini monde 3D depuis zéro, avec Unity 3D. Ensuite, nous choisissons des textures adéquates. Puis, nous relions notre projet à celui de Dana.

5.1 Choix du modèle

Parmi les différents modèles possibles que nous avons explorés, nous avons fait le choix de partir sur un modèle hexagonal, pour des raisons de simplicité et de praticité, deux mois étant assez courts pour un tel projet. Le système hexagonal se veut pratique pour modéliser le changement d'état de l'environnement. Subdiviser le terrain en tuiles permet d'associer des paramètres précis à une case : est-elle inondée ? Possède-t-elle des features comme une forêt, ou une montagne ? Avec un tel système, il est aisé de modifier l'environnement case par case, cela devient une histoire de changement de paramètre. Et bien que ce système semble moins réaliste que la réalisation d'un terrain car moins "continue", il n'en reste pas moins visuellement agréable. Le défi est de choisir et d'implémenter des textures adéquates.

5.2 Modélisation du mini monde 3D

La réalisation de ce mini monde 3D s'est effectuée en plusieurs étapes. Dans un premier temps, nous avons implémenté le mini monde 3D depuis zéro, avec Unity 3D. Ensuite, nous avons choisi des textures adéquates. Puis, nous avons relié notre projet à celui de Dana.

Dans un premier temps, l'objectif était de développer un mini monde 3D. Nous avons alors trouvé un tutoriel en ligne détaillant la création d'un jeu de gestion utilisant des cases diagonales. Afin d'accélérer le processus, nous nous sommes basés dessus pour réaliser le début du projet. Cela nous a permis un gain de temps considérable.

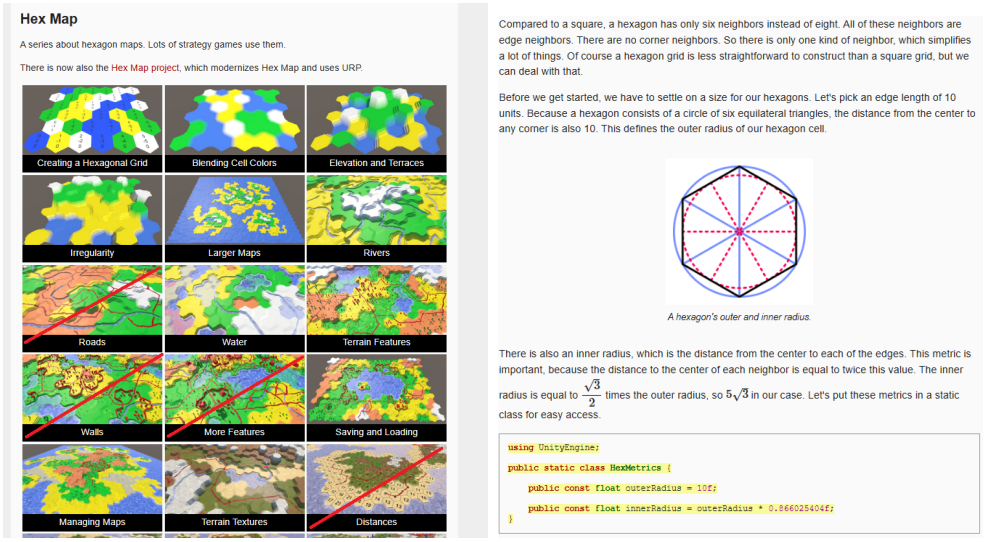
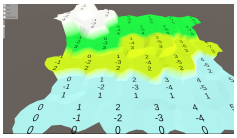


FIGURE 8 – Tutoriel pour réaliser des cartes hexagonales

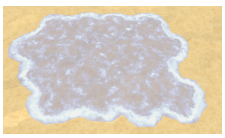
Bien entendu, nous ne souhaitions pas réaliser un jeu de gestion. Il fallait donc sélectionner les étapes qui nous intéressaient, ce qui comprenait :

Le terrain



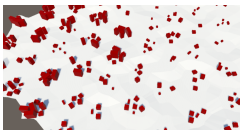
Les rivières

L'eau



Les textures

Les features



De plus, nous avons implémenté la **sauvegarde** et le **chargement** de mondes afin de pouvoir conserver un monde type représentatif avec plusieurs biomes, comme l'était Dana.

A présent, voici une explication de l'implémentation de ces différentes parties.

5.2.1 Le terrain

Nous avons commencé l'implémentation avec ce qui va être la base du terrain : les hexagones. Cette forme peut être divisée en 6 triangles, ce qui nous permet de trianguler simplement.

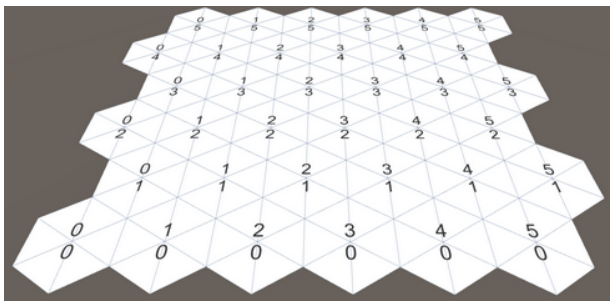


FIGURE 9 – Hexagones avec triangles

Ce modèle de tuile n'est en revanche pas optimale, les transitions de tuiles étant brutes. L'idée est alors de revoir le modèle de la tuile, et de la décomposer en bien plus de "régions", ce qui permettra de réaliser un terrain viable. Nous avons alors ajouté des transitions entres les cotés, ainsi que le coins pour chaque hexagones.

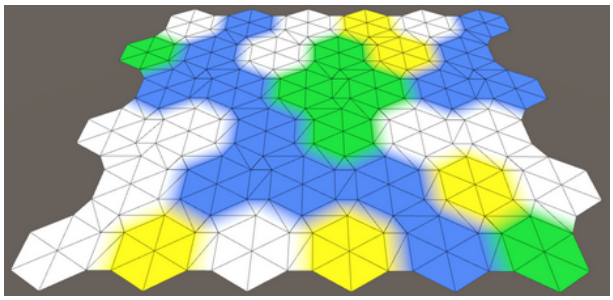


FIGURE 10 – hexagones avec plus de régions

Une fois cela réalisé, nous avons pu ajouter l'élévation. On peut ajuster la hauteur de cette élévation via un paramètre, et donc via l'interface. Cette hauteur est en fait celle du centre de la case que l'on peut appeler plateau, que l'on peut considérer simplement comme une surface 2D que l'on peut rehausser ou abaisser. Les rectangles de transitions de la tuile ainsi que ses triangles de coins seront alors automatiquement triangulés en fonction de la hauteur du plateau.

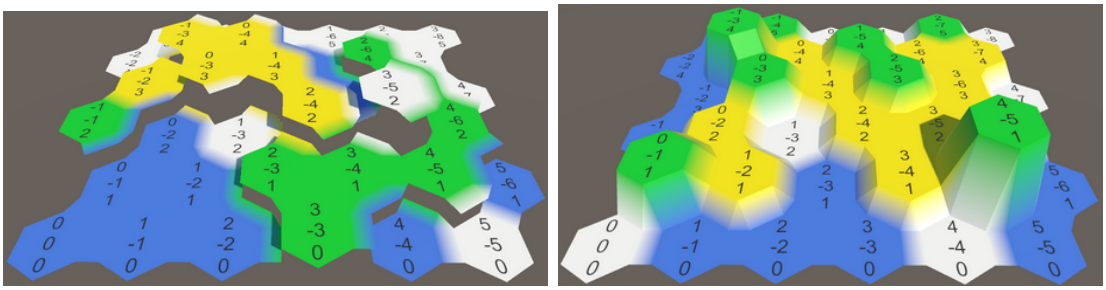


FIGURE 11 – Elevation avec et sans transition

Ceci représente la base du terrain, avec un élément supplémentaire : l'irrégularité des cases. Actuellement, elles ont toutes la même taille et font toutes la même dimension. Nous avons donc ajouté un perturbateur de terrain afin de le rendre plus réaliste. Cela se fait grâce à une texture de bruit, que l'on peut importer directement depuis Unity. Grâce à cette texture que l'on a échantillonné, on peut directement venir modifiez les positions

des sommets du maillage hexagonal en appliquant les vecteurs de perturbation obtenus, créant ainsi des irrégularités naturelles.

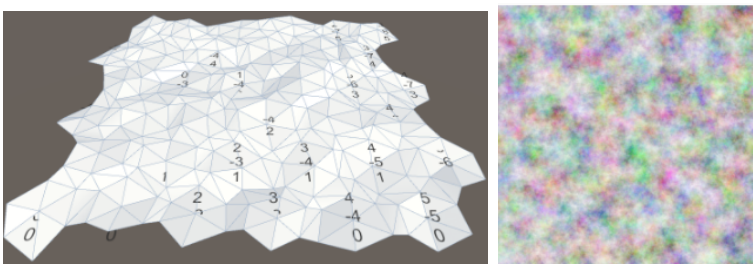


FIGURE 12 – Terrain avec des sommets perturbés (gauche), à l'aide d'une texture de bruit (droite)

5.2.2 Les rivières

La réalisation des rivières se fait en 2 parties : D'abord nous avons modélisé le lit de rivières, ensuite nous avons ajouté la texture de l'eau.

Chaque case possède deux attributs : *hasIncomingRiver* et *hasOutgoingRiver* qui sont des booleens. Si la case possède bien des rivières entrantes ou sortant, alors deux autres paramètres indiquent quelle direction sont concernées. Ces deux paramètres, *Incoming-Direction* et *OutGoingDirection* sont des *HexDirection*, utilisé pour manipuler les voisins dans le système de grille hexagonal.

Plusieurs configurations sont à prendre en compte pour le rivières, indiquées comme ci dessous :

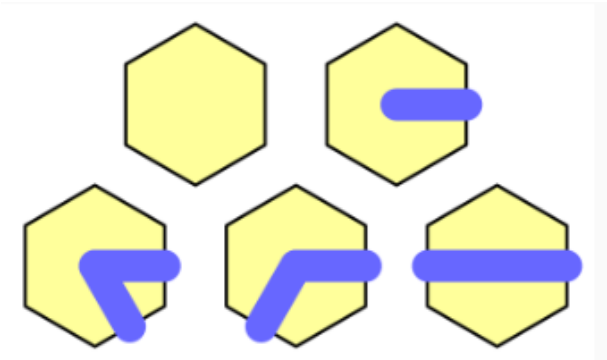


FIGURE 13 – Configurations des rivières

Chacune d'entre elle nécessite une traitement particulier afin d'avoir une modélisation correcte, d'où la nombreuse présence de fonction pour trianguler les rivières. Par la suite, on peut rajouter l'eau grâce à la triangulation de quadrilatères.

FIGURE 14 – Quadrilatères pour représenter la rivière

D'abord, on crée un nouveau matériel ainsi qu'un nouveau shader, qui sont propres à la rivière. Nous avons ensuite cherché à définir le mouvement de l'eau dans cette dernière. Chaque segment est texturé avec une UV map, et le shader de rivière applique une translation des coordonnées UV pour simuler un écoulement. Cette translation s'effectue en fonction d'un paramètre de temps afin de faire défiler la texture. Nous avons pu ensuite définir la réelle apparence de l'eau. Pour cela, nous avons utilisé une texture de bruit, à laquelle on viendra changer la couleur et la transparence.

FIGURE 15 – Texture de la rivière

5.2.3 L'eau

Le premier paramètre que l'on donne pour l'eau et qui a toute son importance pour la suite est le *WaterLevel*. Chaque cellule *HexCell* possède ce dernier et indique ainsi à quelle hauteur dessiner la texture.

Comme pour la rivière, la création de l'eau implique un nouveau materiel, ainsi qu'un nouveau shader. De plus, l'eau constitue une mesh différentes des terrains ainsi que de la rivières. Elle constitue donc une *HexMesh* différente. Mise à part cela, le concept pour réaliser l'eau est le même que pour le terrain au départ.

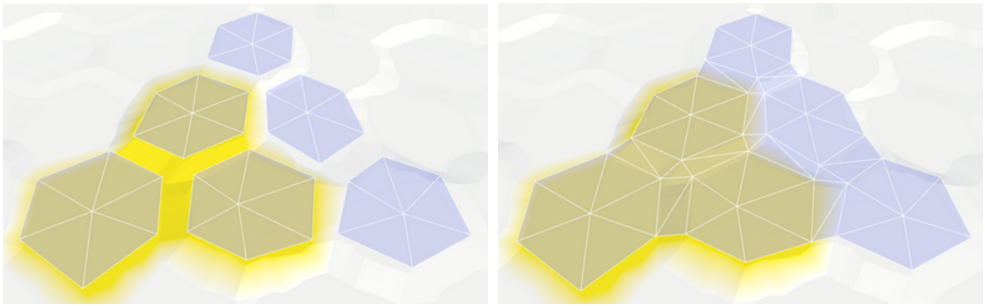


FIGURE 16 – Triangulation des hexagones d'eau

On peut aussi définir le niveau de l'eau grâce au WaterLevel. Comme on peut le voir nous nous ne sommes pas occupé de créer une cohérence avec les différents niveau d'eau. Il faut indiquer manuellement le bon niveau.

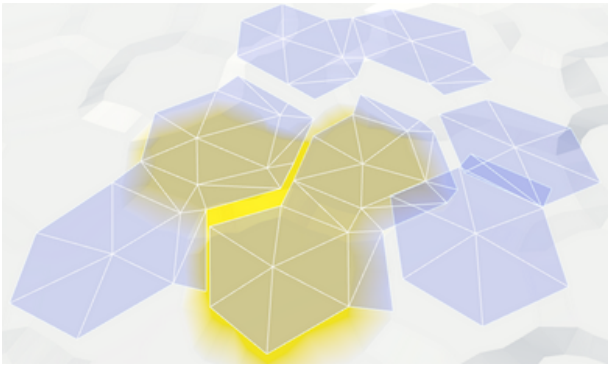


FIGURE 17 – Différents niveaux d'eau

L'enjeu principal de l'eau est son animation. Pour ce faire, nous avons ajouté des perturbations régulières à notre textures plates (des textures de bruit encore un fois). Ainsi on peut former des vagues régulières se propageant dans un sens, mais très dispersées, comme l'océan. On va retrouver une deuxième animation sur les rivages (ou *WaterShore*), consistant à reproduire l'effet de vague se cassant sur la plage.

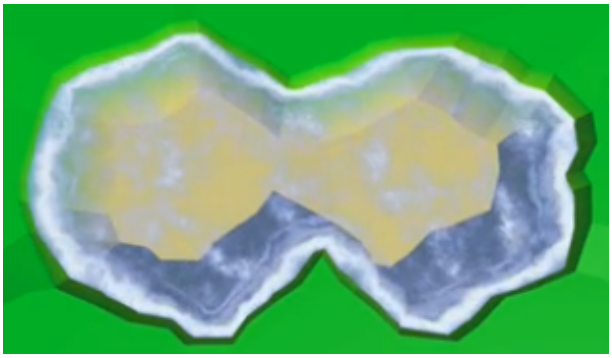


FIGURE 18 – Animation des rivages

La dernière étape d'animation fut celle des rivières se déversant dans l'eau, créant ainsi un estuaire. Il a fallu trianguliariser correctement la partie de l'eau concernée

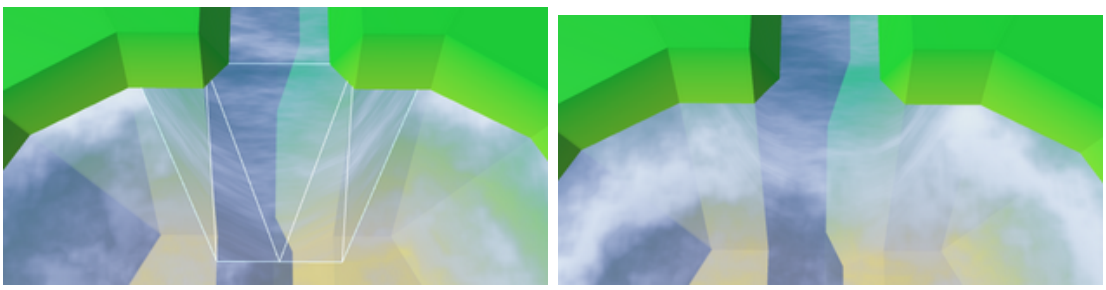


FIGURE 19 – Triangularisation d'un estuaire

5.2.4 Les textures

Nous avons ensuite implémenter les textures, ce qui nous permet d'obtenir un réel terrain. La première étape de cette implémentation fut l'utilisation des couleurs des sommets comme une *Splat Maps*. Cela signifie que l'on va sommer les 3 valeurs de la couleurs afin d'en obtenir une seule. Cette méthode nous permet en fait de simplifier le passage de la couleur à la texture.

Afin d'implémenter les textures, il faut prendre plusieurs éléments en compte, à savoir : l'adaptation à la structure de la rivière, les "bordures" ou l'on doit ajouter des transitions, puis les coins

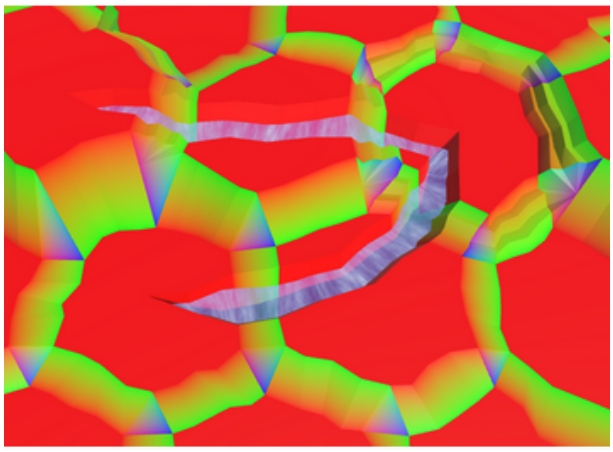


FIGURE 20 – Pattern des textures

Après cela, nous avons pu ajouter des textures, en créant d'abord un texture array, puis en ajoutant les *patterns* de textures. Nous avons directement récupéré les textures du tutoriel à savoir :

5.2.5 les Features

Afin de rajouter le décor originellement présent dans Dana (les villages, les champs, les forêts, etc), nous ne pouvions juste rajouter rajouter un ensemble d'assets statique par dessus le terrain, il nous fallait les lier directement avec les tuiles du terrains. C'est pourquoi nous avons implémenter sur les tuiles des features.

Chaque tuile comporte deux types de feature, une features centrale et des features nivelé. Une features centrale est juste un asset posé sur le centre d'une tuile avec une rotation aléatoire. Elles sont utilisé pour des assets soit de grande taille soit visuellement important. Les features nivelés sont un ensemble d'assets que l'on partitionne sur une

tuile. Lorsqu'on utilise une features nivelé, on décompose une tuile en sept section, six pour chaque direction vers une extrémité de la tuile et une pour le centre de la tuile. Nous plaçons ainsi des assets de petite taille (exemple : un arbre) dans chaque section de la tuile selon plusieurs paramètres offrant ainsi une grande variété de tuile différentes avec les même assets.

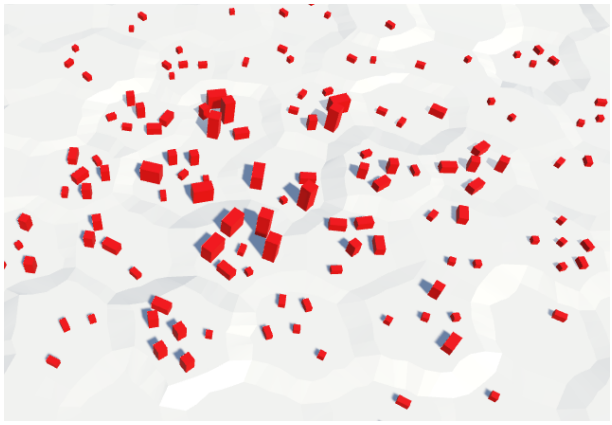


FIGURE 21 – Features nivelé urbaine, forte concentration au centre, mais faible présence sur les tuiles éloigné

Les features nivelés servent principalement à visualiser un biome / un environnement, ou une frontière entre deux biome. Nous avons actuellement trois type d'environnement proposé par les features nivelés : urbain, champ et forêts. Représentant donc les trois environnement de Dana : village (urbain), sols en bon état (forêts), sols dégradé (champs).

Le noms des features nivelés vient du fait que pour chaque type type d'environnement nous avons quatre niveau représentant la présence de l'environnement sur une tuile : niveau 0 correspond à l'absence complète de l'environnement, et niveau 3 correspond à la surprésence d'un environnement. À savoir, chaque feature nivelé est indépendante les unes des autres, ainsi nous pouvons avoir sur une même case les features urbaines, champêtres et boisés chacune au niveau 3.

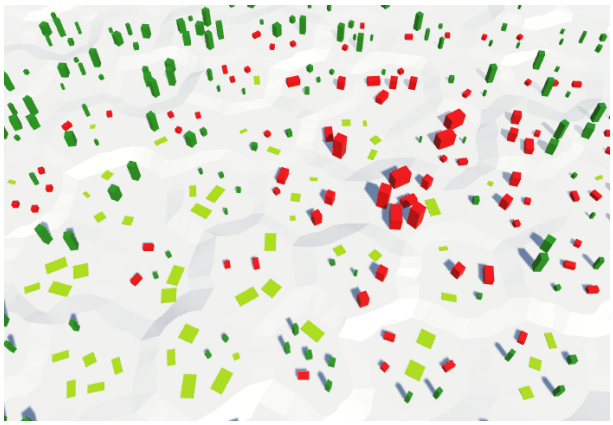


FIGURE 22 – Mélange de feature prototype avec en rouge les assets Urbain, en vert les forêts, en jaune les champs

5.2.6 Sauvegarde du monde

Lors du développement du monde, un problème s'est imposé, à chaque fois que nous compilions le projet, nous démarions toujours avec un monde vierge. En effet, le monde est créé à chaque compilation du projet et n'est sauvegarder à aucun moment. Nous avons donc du implémenter un système de sauvegarde du monde. Pour ce faire, nous avons encodé en binaire principalement les donnés de chacune des tuiles l'une à la suite des autres dans un même fichier qui est chargé au lancement de l'application.

5.3 Intégration du monde

Afin d'implémenter notre modèle de monde en tuile à Dana, nous devions résoudre quelque problèmes, consistant à recoder et adapter quasiment entièrement la partie *miniWorld* du code du projet.

Le premier problème est la gestion des champs et des forêts de Dana. Originellement, le monde de Dana comporte juste des forêts et champs que l'on a préalablement placé avant le lancement de l'application. Ils sont par la suite progressivement affiché et caché en fonction des besoins. Or notre monde qui est généré à chaque lancement de l'application ne peut bénéficier du même artifice. Pour trouver une solution à la gestion des forêts et des champs qui doivent apparaître et disparaître progressivement en fonction du régime alimentaire de l'utilisateur, nous avons implémenté un parcours de graphe similaire à l'algorithme de *pathfinding* A*. L'objectif consiste, à partir d'un point d'origine surnommé "spawn", nous faisons apparaître les forêts ou les champs sur les cases les plus proches de ce point jusqu'à ce qu'il y est autant de forêts ou de champ que voulu. Le choix de s'inspirer de l'algorithme de pathfinding A* est justifié de un par l'efficacité de l'algorithme mais aussi par l'aisance de notre équipe à utiliser ce dernier et à le modifier sur Unity 3D.

Un second problème rencontré est la gestion de l'eau. Originellement sur Dana, l'eau était juste un bloc que l'on monté de façon linéaire en fonction des années et du régime alimentaire. Nous aurions pu user du même artifice, mais le client n'était pas satisfait par ce rendu. Nous avons donc implémenter une nouvelle représentation de l'eau sur notre modèle en tuile (modèle de l'eau présenté précédemment dans ce même rapport). Quant à sa gestion, nous avons commencé à concevoir plusieurs algorithmes d'égalisation des eaux pour simuler un fluide permettant de monter de façon progressive et non linéaire, ce qui aurait pu avoir un impact sur le terrain. Malheureusement aucun de ces algorithmes ne nous satisfaisaient, en raison de leur implémentation soit trop complexe, soit trop lourd à l'exécution.

Heureusement nous avons trouvé un algorithme qui nous convenait en répondant à toute nos attentes (complexité, facilité d'implémentation, et interaction organique avec le monde possible). Il s'agit simplement de l'algorithme utilisé pour la gestion des champs et des forêts. La seul différence et que ce dernier ne s'arrête pas quand il rempli le nombre

de tuile voulue, mais simplement quand il ne trouve plus de case à remplir à la hauteur voulue. Ainsi nous relançons donc l'algorithme à chaque année qui passe avec une hauteur d'eau (hauteur donné par la gestion des données déjà géré par Dana) différentes.

5.4 Difficultés rencontrées

Certaines étapes nous ont particulièrement ralentit dans le développement du projet, nous faisant prendre du retard dans l'avancement général. Souvent, la méthode que nous employions en cas de blocages était de passer la main à quelqu'un d'autre, car il est selon nous compliqué de voir un problème après avoir passé des heures dessus. Un oeil neuf pouvait alors s'avérer utile.

Parmi l'ensemble de difficultés, voici une sélection de ceux qui nous ont le plus ralenti.

5.4.1 Exemple spécifique : Le développement de l'eau

Le début du développement a la particularité d'être fortement rattaché à un tutoriel. Ce dernier est très linéaire, ce qui signifie que pour faire une étape, il faut dans la plupart des cas avoir réalisé l'ensemble des étapes précédentes. Si ce n'est pas fait, on risque de s'embrouiller assez facilement, ce qui rendrait la tâche plus ardue.

Ainsi, nous avions dans un premier temps décidé de ne pas développer la rivière, car ce n'était pas, selon nous, quelque chose de nécessaire dans le cadre du projet. Nous étions dans l'optique de gagner du temps. Cependant, après réalisation, certains bugs graphiques sont apparus, sans exactement que nous sachions pourquoi, puisque nous avons suivi l'ensemble des étapes indiquées. Même après une recherche active de l'erreur, cette dernière demeurait

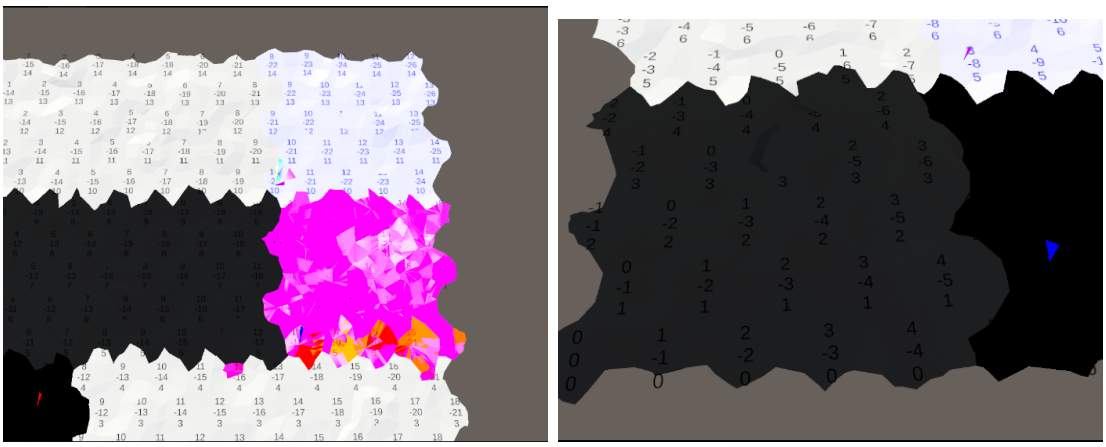


FIGURE 23 – Bug Graphique de l'eau

C'est par la suite que nous nous sommes rendu compte de l'importance d'implémenter la rivière. Son code était nécessaire pour la réalisation de l'eau. Nous sommes donc revenus sur l'implémentation de la rivière, qui s'est faite sans soucis majeurs par ailleurs. Nous aurions simplement pu gagner du temps en l'implémentant avant l'eau.

6 Conclusion

L'objectif principal de ce projet était la refonte graphique de Dana, en veillant à conserver des performances optimales. Dans l'ensemble, nous avons atteint la quasi-totalité de nos objectifs.

L'implémentation du mini-monde est désormais concrète et fonctionnelle, offrant des graphismes nettement améliorés et agréables. Cependant, l'intégration de Dana s'est révélée plus complexe que prévu, entraînant un retard sur cet aspect essentiel du projet. De plus, certains éléments n'ont pas pu être finalisés faute de temps, bien que leur développement ait été bien avancé. C'est notamment le cas de la fonte des glaces et de l'égalisation des eaux, qui restent à intégrer.

Malgré ces défis, nous sommes globalement satisfaits des avancées réalisées. Ce projet a permis une amélioration significative et constitue une base fonctionnelle pour de futures évolutions de l'application.

Références

[1] ADEME, *Agribalyse : méthodologie acv*, 2025. https ://doc.agribalyse.fr/documentation/methodolo
acv.

[2] P. Breves and H. Schramm, *Bridging psychological distance :- the impact of immersive media on distant and proximal environmental issues*, Feb 2021. https ://www.sciencedirect.com/science/article/pii/S0747563220303538.

[3] C. Dugast and A. Soyeux, *Faire sa part*, 2022. https ://www.carbone4.com/publication-faire-sa-part.

[4] A. Ferron, M. Hachet, and Y. Jansen, *Visualizing the environmental impact of dietary choices : Exploring an interactive mini-world as a proxy to communicate three planetary boundaries : Visualiser l'impact environnemental des choix alimentaires : exploration d'un mini-monde interactif comme visualisation mandataire pour communiquer trois limites planétaires*, in Proceedings of the 35th Conference on l'Interaction Humain-Machine, 2024, pp. 1–16.

[5] GIEC, *"c'est établi : nous pouvons réduire de moitié les émissions d'ici à 2030, mais il faut agir aujourd'hui"*. Communiqué de presse, 2022. Consulté le 17 mars 2025.

[6] T. L. P.-A. Cinquin, P. Giraudeau and L. Cousin, *Une interface tangible pour la réduction*, https ://hal.archives-ouvertes.fr/hal-01900054, (Oct. 2018).