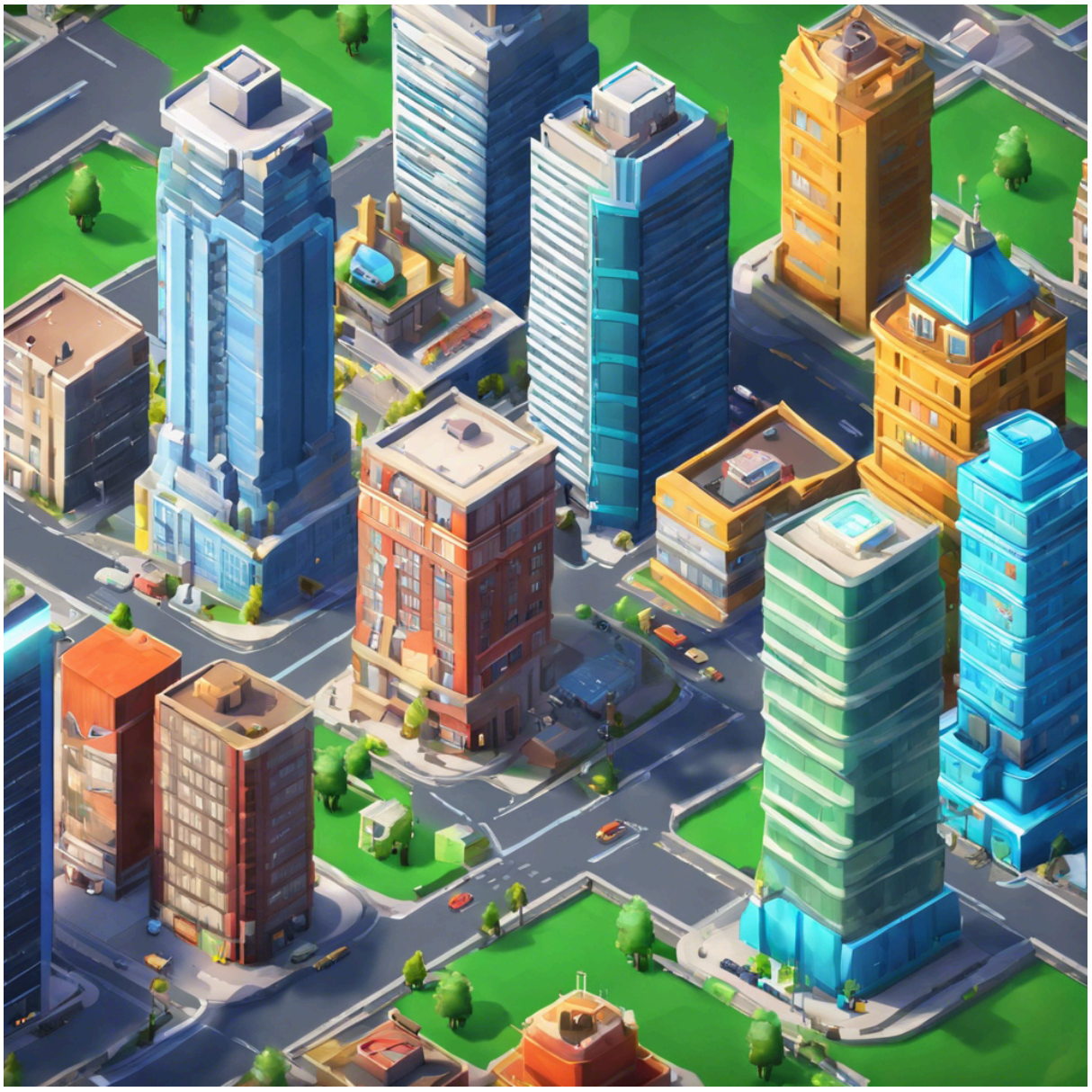
CITY TYCOON PROJECT

by Sullivan Bouvy, Kilyan Fontana and Olivier Legras



SOMMAIRE

I] Introduction / Explication du projet

II] Sullivan

III] Olivier

IV] Kilyan

V] Description approfondies des classes

A] Affichage

B] Backup

C] Building

D] BuildingType

E] Button

F] Game

G] GameStep

H] Launcher

I] MapStep

J] Player

VI] Création de maps

VII] Erreurs et améliorations possibles

VIII] Conclusion

I] Introduction / Explication du projet

Le jeu met en scène John, un personnage contrôlé par le joueur. John a un rêve : devenir riche. Le joueur doit donc tout faire pour qu'il obtienne le plus d'argent possible. John a une capacité spéciale : il peut gagner de l'argent en marchant et peut même parfois gagner des diamants.
Pour être riche, le joueur va devoir acheter des objets achetables qui lui permettront d'obtenir encore plus d'argent. Les objets achetables ont un système de niveau. Plus les niveaux sont élevés, plus le joueur gagne de l'argent. Une fois que le joueur aura acheté tous les bâtiments et les aura mis au niveau maximum, il pourra passer à la map suivante. Ainsi, plus le joueur gagne de l'argent, plus les bâtiments qu'il devra acheter seront chers.

Ce jeu combine la gestion des ressources, la construction et l'exploration dans un cadre varié et engageant, offrant aux joueurs une expérience de jeu riche et diversifiée.

- Campagne : Un joueur commence avec une petite somme de cash, achète des roches et un feu de camp. Au fur et à mesure qu'il génère du cash passif, il peut acheter des fleurs et commencer à améliorer ses maisons.
- Ville Médiévale : Une fois tous les bâtiments de la campagne améliorés, le joueur peut accéder à la ville médiévale, où il doit acheter et améliorer une forge et un terrain d'entraînement, parmi d'autres bâtiments.
- Ville Balnéaire : En atteignant cette map, le joueur peut acheter des maisons et explorer l'intérieur de l'une d'elles, ajoutant une nouvelle dimension d'exploration au jeu.

1. Mécaniques de Jeu

Collision : Le jeu inclut une mécanique de collision réaliste, ce qui signifie que les objets et les personnages réagissent physiquement lorsqu'ils entrent en contact les uns avec les autres.

Musique : Chaque map est accompagnée d'une musique d'ambiance unique qui enrichit l'expérience immersive.

Perspective : Les perspectives du jeu entre le joueur et les éléments des différentes cartes ont été pensés de sorte que l'affichage du joueur soit le plus réaliste possible.

Tourbillons révélateurs de prix : Dans chacune des différentes cartes, il y a des tourbillons bleus qui indiquent les différents objets achetables ou les différents objets importants. Lorsque le joueur se trouve sur l'un de ce tourbillon bleu, le prix, le niveau et l'objet en question est affiché pour que le joueur puisse l'acheter.

2. Maps

Le jeu propose plusieurs maps que les joueurs peuvent explorer et sur lesquelles ils peuvent acheter et améliorer des objets achetables :

Map 1 : Campagne

- Objets Achetables :
    - Roches
    - Feu de camp
    - Fleurs
    - 4 maisons

- Cette carte offre un cadre paisible et naturel où les joueurs peuvent commencer leur aventure en achetant des éléments basiques et en développant progressivement leurs ressources. On peut apercevoir une référence au jeu Pokémon avec la statue présente dans l'eau.

Map 2 : Ville Médiévale

- Objets Achetables :
    - Forge
    - Terrain d’entraînement
    - 6 autres bâtiments
- Inspirée de l'époque médiévale, cette carte propose des bâtiments plus complexes et un environnement riche en histoire et en culture. On peut apercevoir l’épée d’Arthur.

Map 3 : Ville Balnéaire

- Objets Achetables :
    - 3 maisons
    - Possibilité d’entrer dans une des maisons
- Cette carte offre un cadre côtier détendu où les joueurs peuvent acheter des maisons et explorer l'intérieur d'une maison spécifique.

3. Système de Monnaie

Le jeu utilise trois types de monnaies :

- Cash : Utilisé pour les achats de base et les améliorations.
- Cash Passif : Généré automatiquement par les objets achetables.
- Diamants : Monnaie premium, obtenue par des achats in-app, en regardant des publicités ou en marchant longtemps.

4. Bâtiments et Objets Achetables

Chaque map propose des objets et des bâtiments spécifiques que les joueurs peuvent acheter et améliorer. Les bâtiments ont des niveaux et des prix différents, et il est nécessaire de les améliorer pour progresser dans le jeu.

5. Progression et Changement de Map

Pour accéder à une nouvelle map, les joueurs doivent avoir amélioré tous les bâtiments de la map actuelle au niveau maximum. Cela incite les joueurs à compléter entièrement chaque map avant de passer à la suivante, assurant ainsi une progression structurée et gratifiante.

II] Sullivan

Lors du développement du projet, j'ai pu effectuer différentes tâches. Voici un compte rendu détaillé de mes contributions et des tâches accomplies au cours de ce projet.

1. Mise en place du launcher
    - Pour lancer le programme, je me suis dit que c'était bien de créer un fichier Python à part. Je l'ai donc appelé launcher.py, puis j'ai mis dedans le strict minimum pour pouvoir lancer le jeu. L'idée de faire un fichier exprès pour le launcher permet que le joueur ne voit pas toutes les lignes de codes. On pourrait imaginer que le joueur n'a seulement accès à ce fichier, et que les autres sont « bloqués » ou même protégés.
2. Création de la backup
    - J'ai développé un système de sauvegarde automatique. Ce système de sauvegarde permet au joueur de garder son argent, ses diamants, ses bâtiments ainsi que de garder la position dans laquelle il était lorsqu'il a quitté le jeu.
3. Interface utilisateur (UI) et Utilisation de Photoshop
    - J'ai conçu et implémenté l'interface utilisateur du jeu, à l'exception des cartes Map1 et Map2. J'ai donc utilisé Photoshop pour améliorer et créer des éléments graphiques tels que les boutons pour acheter/améliorer les bâtiments, le panneau de tutoriel, les tourbillons qui permettent au joueur de savoir les interactions possibles, l'affiche pour la pub, les affiches pour les transitions vers la prochaine ville (ou non si impossibilité), et les boutons.
4. Mise en place des différentes classes de jeu
    - MapStep: Gestion des étapes de la carte.
    - GameStep: Gestion des étapes du jeu.
    - BuildingType: Définition des types de bâtiments disponibles dans le jeu.
    - Button: Création et gestion des boutons interactifs.

5. Système de collision
    - J'ai mis en place un système de collisions pour éviter que les personnages et objets ne se traversent.

6. Affichage du joueur
    - Le joueur est affiché avec des images différentes selon la direction de son déplacement.
7. Image de profil
    - En effet, grâce à une intelligence artificielle de création d'image, j'ai pu ajouter une image unique pour notre jeu.
8. Mouvement du joueur et gain d'argent
    - J'ai programmé le mouvement du joueur et le système permettant de gagner de l'argent pendant les déplacements. De plus, en fonction de la direction du joueur, l'image de John change.
9. Début de strDollars et strDiamonds
    - J'ai commencé à développer un système de gestion des monnaies virtuelles du jeu (strDollars et strDiamonds), avec un format à un chiffre après la virgule.
10. Essai de vidéo de lancement
    - J'ai essayé d'intégrer une vidéo de lancement directement dans le jeu, sans passer par un site web.
11. Création des sites internets
    - J'ai créé des sites internets qui permettent de voir des pubs
12. Map3 et intérieur de la maison
    - J'ai conçu et mis en place la carte Map3 ainsi que l'intérieur d'une des maisons

III] Olivier

1. Création de la Bêta de la Map 1

La première étape du projet de mon côté a été de concevoir la bêta de la première carte. Cela a impliqué :

-La mise en place de la structure de base de la carte.

-L`ajout des éléments essentiels pour un premier test du jeu.

-La vérification des fonctionnalités et de la navigation sur la carte.

2. Création de la Map 2

Suite à la bêta de la Map 1, la deuxième carte a été développée :

-Conception et mise en œuvre des éléments graphiques et structurels.

-Intégration des fonctionnalités apprises et optimisées à partir de la Map 1.

-Tests initiaux pour assurer la stabilité de la map et la non présence de bug

3. Possibilités d'Achat de Certains Bâtiments

Une des fonctionnalités majeures ajoutées au projet est la possibilité d'acheter certains bâtiments :

-Implémentation des mécanismes d'achat pour divers bâtiments.

-Tests pour garantir que les achats fonctionnent correctement et sont enregistrés.

4. Ajout d'Images des Bâtiments sur le Panneau d’Achat

Pour améliorer l'expérience utilisateur, des images des bâtiments ont été ajoutées sur le panneau d’achat :

Sélection et intégration d'images appropriées pour chaque bâtiment.

Mise à jour de l'interface du panneau d’achat pour inclure ces visuels.

Vérification de l'affichage correct et de la réactivité de l'interface.

5. Recherche de Certains Tileset :

La recherche et l’intégration de nouveaux tilesets ont été effectuées pour enrichir les cartes :

Identification de tilesets compatibles avec le style visuel du projet.

Téléchargement et intégration de ces tilesets dans le projet.

Ajustement des cartes existantes pour inclure les nouveaux éléments graphiques.

6. Correction de Certains Bugs

Une partie cruciale du développement a été la correction de divers bugs :

-Identification des bugs impactant la jouabilité et la performance.

-Application de correctifs et optimisation du code.

-Tests pour s’assurer que les corrections ne crée pas d’autres problèmes

7. Rédaction de la Partie “Les Maps” du Rapport de Projet

La section du rapport de projet concernant les cartes a été rédigée :

-Description détaillée des processus de création des cartes.

8. Proposition d’Idées

Tout au long du projet, diverses idées ont été proposées pour améliorer le développement et les fonctionnalités :

-Suggestions d'améliorations pour les cartes et les mécanismes de jeu.

-Innovations potentielles pour l'interface utilisateur et l'expérience de jeu.

-Réflexions sur les futures extensions et ajouts au projet.

IV] Kilyan

Création de la Map 1 Finale

- **Description :** J'ai terminé la conception et l'implémentation de la première carte du jeu (Map 1). Cela inclut la disposition des éléments visuels, la définition des limites de la carte et la mise en place des fonctionnalités spécifiques à cette carte tel que les collisions, les différents bâtiments, leurs coûts et la réflexion sur l'avancée du joueur.

- **Détails :** La carte a été conçue en utilisant Tiled. Les éléments graphiques ont été disposés de manière à créer un environnement cohérent et immersif pour les utilisateurs et leur donner la meilleure expérience de jeu possible.

Changement de Map avec Conditions et Messages

- **Description :** J'ai implémenté la fonctionnalité permettant de changer de carte en fonction de certaines conditions prédéfinies comme le fait d'être obligé d'avoir tous les bâtiments pour pouvoir accéder à la prochaine map ou bien le fait qu'il fallait avoir une certaine somme d'argent avant de pouvoir accéder à la prochaine map (même si cette idée a été abandonné) ainsi que des messages informatifs pour guider l'utilisateur lors de ces transitions.

- **Détails :** Les conditions de changement de carte peuvent inclure des événements spécifiques, l'atteinte de certains points de la carte, ou des interactions avec des objets ou des personnages. Les messages ont été conçus pour être clairs et informatifs afin d'améliorer l'expérience de l'utilisateur.

Ajustement de Détails Visuels et Pratiques

- **Description :** J'ai apporté divers ajustements visuels et pratiques pour améliorer l'apparence et la fonctionnalité des cartes, comme le fait d'avoir modifié l'emplacement des différents money et ajouté des chiffres après la virgule pour une plus grande précision.

- **Détails :** Cela inclut la correction de bugs graphiques comme pour les différents calques, l'affichage du cout des bâtiments, les messages si le joueur n'a pas assez d'argent, l'esthétique de l'affichage et la mise à jour des visuels pour une meilleure esthétique. Les ajustements pratiques peuvent inclure des modifications de la disposition des éléments pour améliorer la jouabilité.

Bêta de la sauvegarde et sauvegarde de la Carte Actuelle

- **Description :** J'ai implémenté la bêta d'un système de sauvegarde qui a été terminé par mon camarade Sullivan permettant, j'y ai ensuite ajouté une sauvegarde pour conserver l'état actuel de la carte.

- **Détails :** Le système de sauvegarde enregistre le niveau des bâtiments, l'état des interactions et les progrès du joueur comme leur argent ou leurs positions. Cela permet aux utilisateurs de reprendre leur partie là où ils l'avaient laissée.

Ajout de certaines Collisions et bâtiments dans la Map 2

- **Description :** J'ai ajouté des mécanismes de collision dans la deuxième carte 2 pour gérer les interactions physiques entre les éléments du jeu et les avatars des joueurs.

- **Détails :** Les zones de collision ont été définies pour empêcher les joueurs de traverser certains obstacles et pour créer des interactions réalistes avec l'environnement ainsi que pour permettre au jeu de comprendre que le joueur se trouve dans la zone pour acheter le bâtiment.

Recherche de Tileset et Mise en Forme des Éléments

- **Description :** J'ai recherché et sélectionné des ensembles de tileset pour l'utilisation dans les différentes cartes, puis j'ai organisé et mis en forme les éléments visuels.

- **Détails :** Les tilesets sont des ensembles de petites images utilisées pour créer les graphismes des cartes. J'ai choisi ceux qui correspondent le mieux à l'esthétique et le game design souhaitée et je les ai disposés de manière cohérente.

Ajout d'Attributs dans la Classe Buildings

- **Description :** J'ai enrichi la classe Buildings en ajoutant de nouveaux attributs pour gérer des caractéristiques supplémentaires des bâtiments.

- **Détails :** Les nouveaux attributs peuvent que j'ai implémenté ont été ajouté pour que chaque bâtiment ait un niveau maximum à lui et un nouveau prix après l'augmentation de son niveau propre à lui et à son prix de base ce qui rend les bâtiments uniques et permet au joueur de comprendre l'importance de certains bâtiments dans l'évolution du jeu

Première Version du Déplacement du Joueur

- **Description :** J'ai implémenté la première version du système de déplacement du joueur qui a ensuite été complété à l'aide des différents sprite (images du joueur) qui ont été implémentés à la suite de cette version pour que le joueur est une position différente selon sa direction.

- **Détails :** Cette version permet au joueur de se déplacer sur la carte en utilisant les commandes de base (par exemple, les touches fléchées pour les déplacements directionnels). Le système gère les mouvements fluides et réactifs du personnage, en prenant en compte les collisions avec l'environnement. Des tests ont été effectués pour s'assurer que le déplacement est intuitif et fonctionne correctement dans de différents scénarios.

V] Description approfondies des classes

A] Affichage

prend en charge l'affichage pour le joueur

1) draw: gère l'affichage du jeu en général
2) switchMap : fonction qui permet de changer de map : création des collisions des bâtiments et enlever celles des anciens, création d'autres objets, et changement de la taille du joueur

B] Backup

permet de sauvegarder la progression du joueur

1) load : charge les données les plus récentes
2) loadMap : permet de charger directement le dernier emplacement du joueur

C] Building

détermine le prix d'un bâtiment et de ses différent niveau ainsi que les revenu qu'ils donnera au joueur

1) str pour le prix du building ce qui permet de transformer les nombreux 0 en différente lettre tels que "K" pour milles ou bien "M" pour un million

D] BuildingType

détermine quel type est un bâtiment et son image associée

1) permet donc de savoir a quel batiment doit etre associé quel image sous la forme : “NOMDUBATIMENT = photobatiment.png”

E] Button

permet de faire fonctionner les boutons

1) touched : fonction permettant de vérifier si le joueur appuie sur le bouton
2) draw : cette fonction correspond a la fonction d’affichage du bouton

F] Game

prend en charge des éléments de base du jeu

1) start : fonction pour initialiser le jeu
2) createPlayer : fonction pour créer le joueur sur la map
3) createBuildings: fonction permettant de créer un batiment dans le code en lui attribuant la valeur d’achat ainsi que son buildingtype pour son image
4) createBuildingsCollideArea : permet de faire les zone de collision du batiment
5) onInputJoueur : fonction qui permet de donner de l’argent en marchant au joueur ainsi que des diamants
6) printGame: permet de rendre l’affichage fonctionnel
7) RestoreMap :fonction qui change self.mapStep en fonction de la map enregistré dans la backup
8) restoreBuilding : restaure les buildings au lancement du jeu grâce à la backup
9) restorePlayer : restaure le joueur en fonction des enregistrements fait avec la backup
10) restoreData : prend les élements de la Backup et les utilise dans les fonctions permettant de les restaurer

G] GameStep

détermine l'étape du jeu

H] Launcher

permet de lancer le jeu

I] MapStep

détermine sur quelle map est le joueur

J] Player

gère les propriétés monétaire et les batiments ainsi que les mouvement du joueur

1) sauvegardelocation : fonction de sauvegarde de location du joueur
2) animation: fonction qui permet l'animation de l'affichage du joueur
3) update : met a jour la position du joueur
4) revenirEnArriere : fonction qui permet de revenir en arrière, c'est à dire de mettre le joueur dans la position d'avant
5) get_image : renvoie l'image du joueur (pour map1 et map2)
6) get_imageMap3 : renvoie l'image du joueur (pour map3)
7) getVerificationFinVille1 : retourne self.conditionCar (True or False)
8) verificationFinVille1 : fonction de verification pour le changement de map1 à map2
9) getVerificationFinVille2 : retourne self.conditionBoat (True or False)
10) verificationFinVille2 : fonction de verification pour le changement de map2 à map3
11) strDollars : str pour le print les dollars du joueur
12) strDiamonds : str pour print les diamants du joueur
13) bougerDroite : permet le mouvement du joueur vers la Droite
14) bougerGauche : permet le mouvement du joueur vers la Gauche
15) bougerBas : permet le mouvement du joueur vers le Bas
16) BougerHaut : permet le mouvement du joueur vers le Haut
17) RevenuPassif : fonction str pour afficher le revenu passif par seconde du joueur
18) ownBuilding : fonction qui permet de savoir si le bâtiment est possédé ou non

VI] Création de Maps

1. Installation de Tiled

Tout d'abord, j'ai téléchargé et installé Tiled depuis le site officiel (https://www.mapeditor.org/).

2. Création d'un Nouveau Projet

Pour débuter, j'ai ouvert Tiled et créé un nouveau fichier en suivant les étapes suivantes :

- Création d'une nouvelle carte
- Définition des paramètres de la carte : J'ai précisé l'orientation (orthogonale), la taille des tuiles (16x16 pixels) et les dimensions de la carte (nombre de tuiles en largeur et hauteur, ce qui correspond 50x50 dans notre cas).

3. Importation des Tilesets

J'ai ensuite importé les tilesets nécessaires :

- Ajout d'un tileset : J'ai utilisé `Map > New Tileset...` pour ajouter des tilesets externes trouver sur google et autre site telle que itch.io

- Chargement des images de tileset : J'ai sélectionné les images appropriées a la map que l'on veut créer et défini les paramètres comme la taille des tuiles, l'espacement et la marge.

4. Placement des Tuiles

Ensuite, j'ai procédé au placement des tuiles sur la carte :

- Sélection de la couche : J'ai travaillé sur la bonne couche en la sélectionnant dans le panneau des calques de manière a ce que le joueur marche sur l'herbe et non sur les murs.
- Choix des tilesets : J'ai choisi le tileset nécessaire au type de ville que l'on a voulu créer

5. Ajout d'Objets

Pour faire fonctionner la carte, j'ai ajouté divers objets :

- ajout des objets : J'ai placé des objets sur la carte et défini leurs propriétés telles que les objets nommés collision qui permettent de bloquer le passage du joueur, ainsi que l'objet joueur pour faire apparaitre le joueur à cet endroit-là. Il y a également les objets pour chaque bâtiment que l'on souhaite être achetables.

6. Définir les Propriétés des tuiles et objets

J'ai pris soin de définir les propriétés des tuiles et des objets :

- Propriétés des tuiles et des objets : J'ai sélectionné chaque tuile ou objet pour définir ses propriétés dans le panneau correspondant, incluant des données spécifiques comme les points de collision ou de spawn.

7. Tests et Ajustements

Enfin, j'ai testé la carte dans notre code et effectué les ajustements nécessaires :

- Tests : J'ai importé la carte pour vérifier son fonctionnement.
- Ajustements : J'ai corrigé et ajusté les tuiles, les objets et les propriétés en fonction des résultats des tests.

VII] Erreurs et améliorations possibles

1. Problème de calque
    - Problème : Lors de changements de maps, le joueur se retrouve parfois en dessous de certains calques, rendant le jeu difficile. Il est donc nécessaire de redémarrer le jeu pour corriger ce problème.
2. Map avec bâtiments achetables avec des diamants
    - Objectif : Créer une carte dans laquelle les bâtiments seraient achetables uniquement avec des diamants.
    - Problème : La map n'a pas été créée. Sinon, cela aurait été très facile à faire et cela aurait ajouté une meilleure expérience de jeu pour le joueur.
3. Événements aléatoires
    - Objectif : Ajouter des événements aléatoires (neige, tornade, fisc) qui influenceraient l'expérience de jeu.
    - Problème : Le système d'événements nécessitait une logique complexe et une gestion dynamique du gameplay, ce qui a été jugé trop ambitieux pour le temps disponible. De plus, il aurait fallu ajouter d'autres affiches qu'il aurait fallu créer avec Photoshop
4. Système d'eau et d'électricité
    - Objectif : Ajouter un système de gestion de l'eau et de l'électricité.
    - Problème : Ce système nécessitait une infrastructure de simulation détaillée, qui n'a pas pu être développée en raison de la complexité et du temps limité. On aurait quand même pu imaginer le fait que le joueur doive constamment ajouter de l'électricité pour ses maisons, ce qui aurait amélioré l'expérience de jeu.
5. PNJ vendeurs dans la map3
    - Objectif : Ajouter des PNJ (personnages non-joueurs) qui vendraient des objets dans la carte Map3 tel qu'un boost d'argent pendant un certain temps ou encore un boost de vitesse.
    - Problème : Malgré le fait que cela n'aurait pas été difficile, cela n'a pas pu être développé à cause du temps limité exigé.
6. Léviator a diamants passif
    - Objectif : Lorsque le joueur possède tous les bâtiments de la map 3 au niveau maximum, lorsqu'il retourne dans la ville 1, un pont donnant accès au léviator qui, si on lui donne une certaine somme d'argent, nous donnera des diamants passifs (sans rien faire).
    - Problème : L'outil graphique utilisé pour créer les cartes du jeu ne permet pas de faire cela, car il n'est pas possible d'ajouter des éléments ainsi que de retirer/ajouter certaines collisions sous certaines conditions.
    - Ce qui peut être fait à la place : Directement mettre le pont et les collisions adéquates dès le début du jeu, mais laisser le prix d'achat du léviator pour que le joueur sache qu'il doit revenir ici quand il aura cette somme d'argent.

Conclusion

Le projet City Tycoon a été une enrichissante et un excellent exercice de développement de compétences en programmation, conception de jeux et collaboration en équipe.
À travers les différentes étapes de développement, nous avons pu transformer notre vision initiale en un jeu engageant et interactif.

Points Forts du Projet :
travail d'équipe : La répartition des tâches entre nous a permis de développer nos compétences respectives et de progresser rapidement.

Diversité : Les diverses mécaniques de jeu, telles que les achats de bâtiments, les niveaux, et les différents types de monnaies, ont enrichi l'expérience utilisateur.

Développement Technique : Nous avons mis en place des systèmes techniques avancés comme le launcher, les sauvegardes automatiques, les collisions réalistes et les interfaces utilisateur, augmentant ainsi la qualité et la fonctionnalité du jeu.

Défis et Limites :
Problèmes de Calque : La gestion des calques lors des changements de maps a posé des problèmes techniques nécessitant des redémarrages du jeu.

Systèmes de jeu non Implémentés : Des fonctionnalités comme les événements aléatoires, la gestion de l'eau et de l'électricité, et les interactions avec les PNJ n'ont pas pu être développées en raison du temps limité.

Complexité Graphique : La création et l'intégration des éléments graphiques ont nécessité des efforts supplémentaires, notamment pour éviter les bugs et optimiser l'expérience visuelle.

En conclusion, le projet City Tycoon a atteint ses objectifs principaux en offrant une expérience de jeu immersive et riche en fonctionnalités. Les défis rencontrés ont été autant d'opportunités d'apprentissage, et les succès obtenus témoignent de notre capacité à concevoir et réaliser un projet de jeu complet. Ce projet nous a non seulement permis de développer nos compétences techniques et créatives, mais aussi de renforcer notre capacité à travailler efficacement en équipe.