UNIVERSITÉ PARIS NANTERRE

RAPPORT DU PROJET

JEU DU RÔLE

DOAN Thi Mai Chi - 40016084
THIRUGNANAKUMAR Shageen - 40005408
JARDIN Ryan - 40012419

Remerciements

Le rapport du sujet " Jeu du rôle " est le résultat de notre propres efforts constants et du soutien et des encouragements d'enseignants, d'amis, de collègues et de personnes. A travers cette page, nous tenons tient à remercier ceux qui nous ont aidés au cours de nos études et recherches.

Pendant le projet, d'être confus à inexpérimenté, nous avons rencontré de nombreuses difficultés, mais avec l'aide dévouée de tout le monde autour, tout a été complété.

Je tiens à remercier les professeurs : Monsieur François Delbot et Monsieur Bouquet Valentin pour leur soutien enthousiaste dans la recherche et l'enseignement spécialisé pour mener à bien ce rapport.

Remerciements spéciaux à mon relecteur et correcteur qui a contribué, grâce à ses conseils et recommandations, à l'élaboration et au bon déroulé de mon rapport.

I Introduction

Notre projet est réalisé en python, axé sur la création d'un générateur de cartes de mondes fictifs, se concentre particulièrement sur le domaine du jeu de rôle.

Ce projet vise à créer un outil qui permet aux joueurs de rôle de créer des mondes fictifs pour leurs aventures. L'outil exploite des algorithmes de génération procédurale pour permettre aux utilisateurs de créer facilement des territoires inexplorés avec des topographies uniques.

En mettant l'accent sur la flexibilité, notre générateur offre des options de personnalisation pour intégrer diverses caractéristiques géographiques, comme des montagnes, des rivières, des forêts, des déserts, des villages et des villes, afin de répondre aux besoins spécifiques des créateurs de jeux et d'histoires fantastiques. Ils pourraient, par exemple, définir la taille de la carte, la répartition des caractéristiques géographiques, et l'emplacement des éléments spécifiques, tels que des villes ou des villages.

II Objectif et Fonctionnalités

Objectif

- Générer des mondes fictifs uniques et réalistes
- Permettre aux utilisateurs de personnaliser les mondes
- Stocker les mondes dans une base de données

Fonctionnalités

- Génération de topographies uniques et réalistes, comprenant diverses caractéristiques géographiques, telles que des montagnes, des rivières, des forêts, des déserts, des villages, et des villes.
- Options de personnalisation pour les utilisateurs, telles que la taille de la carte, le climat, la densité de la population, etc.
- Interface utilisateur intuitive et conviviale.

III Travail réalisé

Réalisé : Nous avons réussi à créer une carte du monde en utilisant le bruit de Perlin, fournissant ainsi une méthode de génération de paysages fictifs. Les utilisateurs ont la possibilité d'explorer la carte, de zoomer, de dézoomer et de la déplacer selon leurs préférences, offrant ainsi une personnalisation de leur expérience. De plus, le code permet de stocker et de récupérer le bruit de Perlin dans une base de données, offrant aux utilisateurs la possibilité d'organiser et de retrouver facilement leurs mondes fictifs.

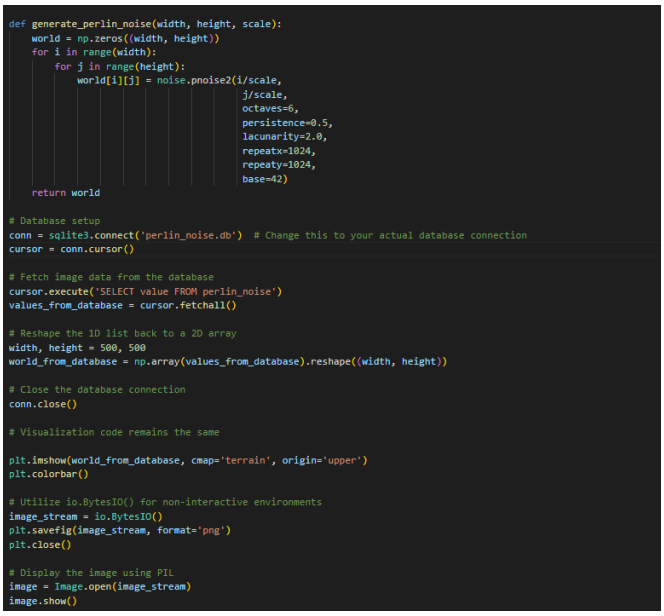
On a utilisé la bibliothèque numpy et matplotlib pour créer une carte de terrain aléatoire, applique un filtre gaussien pour lisser les variations, puis permet à l'utilisateur (ou fixe par défaut au centre) d'ajouter un point sur la carte. La figure résultante est affichée avec des variations d'altitude représentées par une carte de couleur.

```
# Paramètres pour la taille de la carte et l'échelle du bruit
largeur, hauteur = 100, 100
echelle = 20

# Générer un terrain aléatoire
terrain = np.random.rand(largeur, hauteur)
terrain = gaussian_filter(terrain, sigma=echelle)
print(terrain)

# Position du point à ajouter (par exemple au centre)

print('Entrer X : ')
point_x = input()
print('Entrer Y : ')
point_y = input()

# point_x, point_y = largeur // 2, hauteur // 2

# Visualiser le terrain
plt.figure(figsize=(10, 10))
plt.imshow(terrain, cmap='terrain')
plt.colorbar(label='Altitude')
plt.scatter(point_x, point_y, c='red', s=100)  # Ajout d'un point rouge pour marquer un emplacement
plt.title('Carte de Terrain Générée Proceduralement')
plt.show()
```

Après les coordonnées initiales du point sont définies, une figure est créée avec un seul axe, et un point rouge est positionné à ces coordonnées initiales. Le script comporte également une fonction appelée on_key, qui réagit lorsque l'utilisateur presse une touche du clavier. En fonction de la touche pressée, les coordonnées du point sont ajustées. Le processus de mise à jour implique l'effacement du point existant, le dessin du nouveau point, et la mise à jour de la figure. Enfin, l'événement de pression de touche est lié à la fonction on_key. L'objectif principal de ce code est de fournir une interface graphique interactive permettant à l'utilisateur de déplacer un point rouge en utilisant les touches fléchées du clavier. La fenêtre graphique est actualisée en temps réel pour refléter ces déplacements.

```python
# Initialisation des coordonnées du point
point = [50, 50]  # Commencer au centre

# Initialisation de la figure et de l'axe
fig, ax = plt.subplots()
ax.plot(point[0], point[1], 'ro')  # Point rouge

# Fonction pour mettre à jour le point
def on_key(event):
    if event.key == 'up':
        point[1] -= 1
    elif event.key == 'down':
        point[1] += 1
    elif event.key == 'left':
        point[0] -= 1
    elif event.key == 'right':
        point[0] += 1

    ax.clear()  # Efface l'ancien point
    ax.plot(point[0], point[1], 'ro')  # Dessine le nouveau point
    plt.draw()  # Met à jour la figure
    plt.colorbar(label='Altitude')


# Connecte l'événement de la touche à la fonction on_key
fig.canvas.mpl_connect('key_press_event', on_key)

plt.show()
```

Le projet a été développé en utilisant le langage de programmation Python et la bibliothèque Pygame pour l'interface utilisateur. La génération de bruit a été réalisée en utilisant l'algorithme de Perlin pour créer des cartes de hauteur réalistes. Grâce à chatgpt, on a généré du bruit de Perlin en utilisant la fonction noise.pnoise2 de la bibliothèque noise. La fonction noise.pnoise2 prend cinq paramètres : x, y, octaves, persistence, lacunarity, et repeatx et repeaty. Les paramètres octaves et persistence contrôlent la fréquence et la rugosité du bruit. Le paramètre lacunarity contrôle la distance entre les octaves. Les paramètres repeatx et repeaty contrôlent la taille de la texture de bruit. Le code visualise le bruit de Perlin généré en utilisant la bibliothèque matplotlib.pyplot. Le code utilise la fonction imshow pour afficher le bruit array. Le code peut également stocker et récupérer du bruit de Perlin dans une base de données en utilisant la bibliothèque sqlite3. En résumé, le code permet de générer et de visualiser du bruit de Perlin de manière procédurale.



```python
def generate_perlin_noise(width, height, scale):
    world = np.zeros((width, height))
    for i in range(width):
        for j in range(height):
            world[i][j] = noise.pnoise2(i/scale,
                                        j/scale,
                                        octaves=6,
                                        persistence=0.5,
                                        lacunarity=2.0,
                                        repeatx=1024,
                                        repeaty=1024,
                                        base=42)
    return world

# Database setup
conn = sqlite3.connect('perlin_noise.db')  # Change this to your actual database connection
cursor = conn.cursor()

# Fetch image data from the database
cursor.execute('SELECT value FROM perlin_noise')
values_from_database = cursor.fetchall()

# Reshape the 1D list back to a 2D array
width, height = 500, 500
world_from_database = np.array(values_from_database).reshape((width, height))

# Close the database connection
conn.close()

# Visualization code remains the same

plt.imshow(world_from_database, cmap='terrain', origin='upper')
plt.colorbar()

# Utilize io.BytesIO() for non-interactive environments
image_stream = io.BytesIO()
plt.savefig(image_stream, format='png')
plt.close()

# Display the image using PIL
image = Image.open(image_stream)
image.show()
```

On continue de travailler sur la map3, ainsi qu'une map supplémentaire en utilisant Pygame. Une liste lobja est créée pour stocker les images associées à chaque type d'objet. Les images sont chargées depuis des fichiers PNG dans le dossier "images". Une salle vide est créée, puis des rectangles sont créés pour chaque type d'objet. La fonction afficher affiche la salle et les objets.

```python
screen = display.set_mode((hors_map + 100, TAILLE_Y * PIXELS_Y))
lobj = ["fond", "mur", "porte"]

lobja = []
for nom in lobj:
    try:
        img = image.load("images/" + nom + ".png").convert_alpha()
        loaded_images.append(img)
    except Exception as e:
        print(f"Erreur lors du chargement de l'image {nom}: {e}")

print(3)

salle = [[1] * TAILLE_X] + [[1] + [0] * (TAILLE_X - 2) + [1] for i in range(TAILLE_Y - 2)] + [[1]
l_rect = [Rect(hors_map + 10, y, 90, 30) for  y in range(100, len(lobj) * 30 + 100, 30)]
select = None
print(4)
```

Elle parcourt chaque case de la salle et affiche l'image correspondante. Elle affiche également les rectangles pour les types d'objets avec leurs noms. L'utilisateur peut sélectionner un type d'objet en cliquant sur son nom. Il peut ensuite placer l'objet sélectionné dans la salle en cliquant sur une case. La salle mise à jour est affichée après chaque modification.

```
def ecrire(txt, taille, posx, posy, couleur = (0, 0, 0)):
    display.get_surface().blit(font.Font (None, taille).render (txt, 1, couleur), (posx, posy))
# problemme ici
def afficher ():
    for y, ligne in enumerate(salle):
        for x, case in enumerate(ligne):
            display.get_surface().blit(lobja[case], (x * PIXELS_X, y * PIXELS_Y))
    for i, r in enumerate(l_rect):
        ecrire(lobj[i], 30, hors_map + 10, r[1], (255, 255, 255))
    display.flip()

print(1)

afficher()
print(2)

while 1:
    ev = event.wait()
    if ev.type == QUIT or ev.type == KEYDOWN:
        quit(); break
    if ev.type == MOUSEBUTTONUP:
        if ev.pos[0] > hors_map or select == None:
            for i, r in enumerate(l_rect):
                if r.collidepoint(ev.pos): select = i
        else:
            x, y = ev.pos[0] // PIXELS_X, ev.pos[1] // PIXELS_Y
            salle[y][x] = select
            display.get_surface().blit(lobja[select], (PIXELS_X * x, PIXELS_Y * y))
            display.update((PIXELS_X * x, PIXELS_Y * y, PIXELS_X, PIXELS_Y))

with open(name + ".txt", "a") as f:
    for i in salle: f.write(" ".join(map(str, i)) + "\n")
```

Le code est un générateur de carte basé sur le bruit. Il génère une carte du monde en générant un tableau bidimensionnel de valeurs de bruit, où chaque valeur représente l'altitude d'un point de terrain. Le code est organisé en trois modules Le module noisepy contient des fonctions pour générer des valeurs de bruit. Le module map_generator.py génère les cartes du monde. Le fichier teststest_noise.py contient des tests unitaires pour la fonction noise2 du module noise.py.

```
def noise2(x, y):
    # ... (Votre implémentation du bruit simplex pour 2D)
    pass

# //Créez le module de génération de carte (generator/map_generator.py)://

# generator/map_generator.py
from noise import noise2

def generate_world_map(width, height, scale):
    world_map = []

    for y in range(height):
        row = []
        for x in range(width):
            sample_x = x / scale
            sample_y = y / scale
            noise_value = noise2(sample_x, sample_y)
            row.append(noise_value)
        world_map.append(row)

    return world_map

# //Créez un fichier de test (tests/test_noise.py)//
# tests/test_noise.py

import unittest
# from generator.noise import noise2

class TestNoiseFunctions(unittest.TestCase):
    def test_noise2(self):
        # Test your noise2 function here
        pass

if __name__ == '__main__':
    unittest.main()
```

En conclusion, nous avons développé une application de visualisation de cartes de base qui offre aux utilisateurs la possibilité de zoomer, de faire défiler la carte et de sélectionner des zones en cliquant et en glissant.

L'application démarre en chargeant l'image de la carte et en créant une fenêtre de la taille

correspondante. La carte est ensuite redimensionnée pour s'ajuster à la taille de la fenêtre. Elle surveille en permanence les événements de la souris. Lorsqu'un utilisateur clique sur la carte, l'application commence à suivre le mouvement de la souris. Dès que l'utilisateur relâche le bouton de la souris, l'application enregistre la zone sélectionnée.

Chaque fois qu'un événement est traité, l'application met à jour l'affichage pour refléter les modifications effectuées.

Voici le résultat lors de l'exécution du programme:

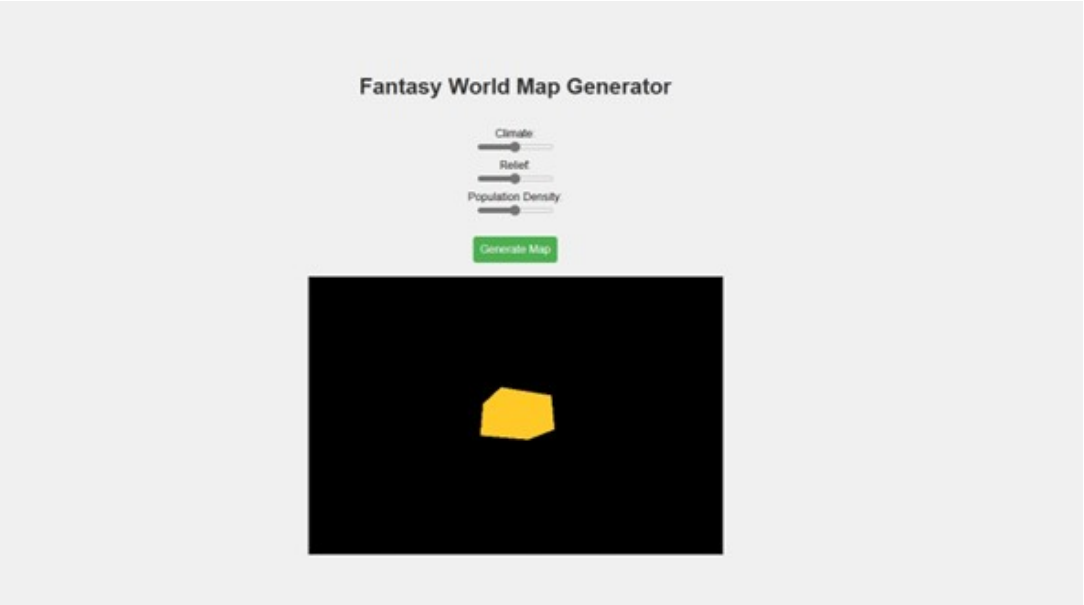
IV Difficultés rencontrées

La mise en œuvre du rapport est pour nous l'occasion de synthétiser et de systématiser ce que nous avons appris, et en même temps de les appliquer dans la pratique pour améliorer nos connaissances dans ce domaine. Bien qu'il ne s'agisse que d'un petit projet, grâce à ce temps, j'ai élargi mes horizons et absorbé beaucoup de connaissances pratiques. Elle aide les étudiants à construire une base théorique plus solide.

Notre première décision a été de choisir le langage de programmation pour notre projet. Nous avons opté initialement pour JavaScript en utilisant Visual Studio Code, puis avons décidé d'utiliser three.js pour générer un plan 3D avec le terrain. Shageen a effectué des recherches en ligne sur la génération de terrains, et nous avons trouvé des ressources utiles, notamment

des vidéos en anglais sur le sujet. Pendant ce temps, Chi a organisé un plan de travail pour maximiser l'efficacité.

À un moment donné, nous avions un programme très avancé. Cependant, une erreur de codage commise par Shageen le 14 décembre 2023 a provoqué un crash complet de l'ordinateur, entraînant une certaine panique. Une des erreurs à ne pas reproduire est de n'avoir pas utilisé GitHub pour le programme, principalement en raison d'une connaissance limitée du réseau, nous avons pratiquement tout partagé sur Discord.



Suite à ces problèmes, nous avons décidé de recommencer le projet en optant cette fois pour un générateur en 2D plutôt qu'en 3D, utilisant Anaconda (Spyder). L'étape suivante consistait à télécharger la bibliothèque "noise" dans la console Spyder. Malheureusement, le téléchargement échouait, probablement en raison d'une détection antivirus, ce qui nous a amenés à désinstaller l'antivirus sur les ordinateurs pour permettre au module "noise" (basé sur l'algorithme de Perlin) de fonctionner correctement. Nous avons suivi des tutoriels sur YouTube pour résoudre ces problèmes. Après ces épisodes, Shageen a émis l'idée que nous aurions dû entamer le projet en 2D dès le départ, considérant que la 3D s'est révélée excessivement compliquée.

Durant le processus de création d'une carte de terrain aléatoire et d'ajout d'un point sur la carte, nous avons été confrontés à l'un des principaux obstacles: le compilateur refusait de fonctionner correctement. En parallèle, nous avons également abordé la problématique du dézoomage, qui s'est avérée complexe en raison d'indications peu claires. Pour résoudre ces problèmes, nous avons dû effectuer des recherches sur plusieurs sites, désactiver les pare-feux et même

désinstaller les antivirus.

Malheureusement, malgré nos efforts, le résultat obtenu ne répondait pas à nos attentes, ce qui nous a poussés à intervenir manuellement en faisant appel à ChatGPT. Dans le processus de correction, ChatGPT nous a suggéré d'utiliser la commande "Cherche-moi une alternative", ce qui a entraîné une série de recherches sans fin.

En réalité, nous avons sous-estimé le temps qu'il a fallu pour terminer le projet, pour le réaliser. Au début, nous avons également sous-estimé la difficulté de produire le spectacle. Ignorance du programme et notre capacité à utiliser les fonctions est limitée, on a rencontré plusieurs difficultés pour atteindre des objectifs du projet. De plus, mettre en place un projet informatique dans les premières années va enrichir d'un point de vue personnel mais aussi d'un point de vue professionnel : à l'avenir, quel que soit le domaine dans lequel on travaille, on devrait assurément faire le travail d'équipe : l'organisation et la notion de gestion du temps que on a pu acquérir dans ce projet vont donc s'avérer très utiles.

V Bilan

Le choix d'un sujet de projet peut parfois être délicat. Sur les conseils de M. Bouquet Valentine, une connaissance, on a orienté mes recherches vers le Jeu du Rôle.

Ce projet s'est avéré bénéfique à plusieurs niveaux. Tout d'abord, il a renforcé notre travail d'équipe et a été une opportunité d'approfondir nos connaissances en informatique, en particulier dans le langage Python. Les bases acquises en classe ont servi de fondation, mais ce projet nous a permis d'explorer davantage et de mieux comprendre des concepts tels que Python, SQL ou l'algorithme de génération de terrain procédurale, l'algorithme de bruit aléatoire,...

En outre, cette expérience pratique a été une occasion précieuse d'acquérir de l'expérience dans notre domaine d'études. Ces compétences pratiques seront certainement utiles dans notre future carrière de développeurs Web ou Game développer.

La mise en œuvre d'un projet informatique dès les premières années se révèle enrichissante tant sur le plan personnel que professionnel. Les compétences en organisation et en gestion du temps développées au cours de ce projet seront des atouts précieux, quel que soit le domaine

dans lequel nous évoluerons à l'avenir.

VI Conclusion

C'est la première fois que nous travaillons en équipe sur un projet avec un objectif clairement défini. L'opinion générale est que nous avons défini nos connaissances communes et appris comment rendre les applications plus attrayantes. Lors de la préparation de notre projet, nous avons essayé de mettre en pratique les connaissances acquises lors des études universitaires et cela en plus de faire une application pratique nous aide à effectuer des tâches au lieu de.

Dans ce rapport, nous avons recherché et mis en œuvre différents algorithmes cryptographiques pour créer un jeu algorithmique simple capable d'exécuter des commandes de base.

Nous devions encore résoudre les problèmes d'importation manquante et travailler sur la sauvegarde. Cela s'est avéré complexe, en particulier pour les installations de maps. Nous avons résolu manuellement ces problèmes avec l'aide de supports en ligne, après une semaine vraiment rude. On a donc terminé le projet, même si Shageen souhaite le poursuivre en raison de son intérêt prononcé pour l'univers fantaisiste et de son orientation de carrière vers le gaming.

Les leçons à retenir sont une meilleure organisation avec plus de communication, mais aussi de débuter de manière simple avant de complexifier, évitant ainsi de se lancer directement dans une idée de projet ambitieuse mais aussi développer son talent du codage. Chi a pour résolution, d'une meilleure communication mais aussi avoir plus de présence dans le groupe et Ryan a pour résolution de développer plus ses problèmes dans le codage, mais aussi d'être plus précis et cohérent. Shageen veut refaire un générateur full fantasy en 3D, si ça prend du temps du coup.

Bien que quelques problèmes subsistent, dans l'ensemble, nous sommes satisfaits des réalisations accomplies jusqu'à présent. En termes de gestion de projet en équipe, nous avons réussi à livrer efficacement les tâches, respectant nos objectifs dans les délais impartis. L'ambiance au sein de l'équipe est particulièrement positive.

VII Bibliographie

Les sites de journaux en ligne

Youtube

Wikipedia

VIII Webographie

(CAT) savoircoder.fr/cat

(Manpage) manpagesfr.free.fr

(Stack overflow) stackoverflow.com

(GeeksforGeeks) geeksforgeeks.org

(Youtube) https://www.youtube.com/watch?v=IKB1hWWedMk

(Youtube playlist)

https://www.youtube.com/watch?v=wbpMiKiSKm8&list=PLFt_AvWsXl0eBW2EiBtl_sxmDtSgZBxB3&index=1