TP de projet de méthodologie 2

Niveau 3

2020 - 2021

Le niveau 3 du projet consiste à compléter votre code pour atteindre la version de base du projet. Il s'agira, dans un premier temps, de placer et de gérer les déplacements de plusieurs murs de météorites. Puis, il conviendra d'implémenter les règles du jeu. Enfin, vous devrez ajouter l'affichage du temps écoulé et différents messages sur l'écran de jeu.

Le sujet est désormais un peu moins guidé que pour les précédents niveaux. À vous de bien organiser votre code pour réaliser ce qui est demandé. Vous êtes libres d'ajouter des champs supplémentaires dans le monde et de créer/utiliser de nouvelles constantes, de nouvelles fonctions, ...

1 Environnement de travail

Pour faire ce niveau, si ce n'est pas déjà fait, installer l'extension `SDL2_TTF` qui va servir à faire l'affichage de texte sur l'écran de jeu.

Sur Arche, vous trouverez des ressources supplémentaires à placer au bon endroit dans votre projet.

- un module `sdl2-ttf-light` pour vous faciliter la tâche pour l'affichage de textes sur l'écran de jeu (les fichiers .c et .h sont à placer avec votre code).
- la police `arial.ttf` (à placer dans votre répertoire contenant vos ressources).

2 Gestion de plusieurs murs de météorites

Jusqu'à présent, nous n'avons considéré qu'un seul mur de météorites. Nous allons maintenant en considérer plusieurs pour constituer notre parcours. Pour cela, nous allons les stocker dans un tableau de `sprite_t`.

1. Ajouter un champ dans la structure du monde correspondant au tableau de murs de météorites.
2. Créer une fonction `init_walls` qui prend le monde comme paramètre et qui donnera leur position initiale aux murs de météorites dans le monde pour former des couloirs. Pour commencer, nous aurons six murs avec les caractéristiques suivantes (celles de la vidéo de démonstration):

	0	1	2	3	4	5
x	48	252	16	188	48	252
y	0	0	-352	-352	-672	-672
largeur	96	96	32	224	96	96
hauteur	192	192	160	160	192	192

Vous remarquerez que les ordonnées de la plupart des murs sont négatifs, car ils sont situés au-dessus de l'écran au départ.

3. Tester maintenant la fonction `init_walls` dans votre programme de test mis en place dans le niveau 2.

4. Appeler la fonction `init_walls` depuis `init_data`.

5. Tester le jeu. Vous remarquerez que les murs que vous venez d'initialiser n'apparaissent pas. Vous remarquerez aussi que le mur correspondant au champ `wall` du monde est toujours présent. La ligne d'arrivée est, quant à elle, mal placée. Nous nous en occuperons plus tard.

6. Afin de permettre à tous les murs de se déplacer verticalement vers le bas, il faut mettre à jour la fonction `update_data`. Pour cela, vous écrirez une fonction `update_walls` qui mettra à jour la position des murs de la même manière que vous l'avez fait pour le mur correspondant au champ `wall` du monde.

7. Ne pas oublier de tester la fonction `update_walls` dans votre programme de test.

8. Maintenant, afin de visualiser sur l'écran tous les murs, nous allons mettre à jour la fonction `refresh_graphics`. Pour cela, vous allez créer une fonction `apply_walls` qui appliquera chaque mur à sa position dans le renderer. Tester. Vous remarquerez encore que le mur correspondant au champ `wall` du monde est toujours présent.

9. Commenter maintenant le champ `wall` de la structure `struct world_s`, ainsi que toutes les opérations impliquant ce champ dans votre code. Tester. Normalement, le mur originel a disparu.

10. Placer maintenant la ligne d'arrivée au bon endroit (son ordonnée est de -960 dans la vidéo de démonstration). Tester.

3 Implémentation des règles du jeu

Désormais, vous pouvez implémenter les règles du jeu. Pour cela, vous allez devoir

- gérer les collisions du vaisseau avec les murs et la ligne d'arrivée
- gérer le temps qui s'est écoulé

Les messages de fin de jeu seront, pour l'instant, affichés sur la sortie standard. Quand le joueur a perdu, le message est "You lost!". Quand le joeur a gagné, le message est "You finished in xx s!" avec xx le temps écoulé depuis le début du jeu en secondes.

L'affichage du temps écoulé au fur et à mesure sera réalisé plus tard.

Comment récupérer le temps écoulé? Vous pouvez utiliser la fonction `SDL_GetTicks` qui renvoie le temps écoulé en millisecondes depuis l'initialisation de la SDL. Allez voir sa documentation: `https://wiki.libsdl.org/SDL_GetTicks`.

4 Affichage de texte à l'écran

Nous souhaitons maintenant faire l'affichage du temps écoulé et du message final du jeu sur l'interface graphique. Pour cela, nous allons utiliser l'extension `SDL2_TTF`. Il existe un tutoriel disponible en ligne ici.

Pour vous simplifier la vie, nous vous mettons à disposition le module `sdl2-ttf-light` qui est une sur-couche de `SDL2-TTF`.

1. Dans la fonction `init`, appeler la fonction `init_ttf` pour initialiser l'environnement TTF. Afin de préparer la suite, penser à placer l'appel à `init_ttf` avant l'initialisation des textures dans `init`.

2. Afin de stocker la police utilisée pour l'affichage, nous allons rajouter un champ `font` de type `TTF_Font *` dans la structure `struct textures_s`.

3. Modifier la fonction `init_textures` afin de charger et stocker la police `arial.ttf` avec une taille de 14. Pour cela, utiliser la fonction `load_font` du module `sdl2-ttf-light`.

4. Modifier la fonction `clean_textures` afin de nettoyer la police avec l'appel à la fonction `clean_font` du module `sdl2-ttf-light`.

5. Notez que le nom de la structure `struct textures_s` n'est plus très approprié. Vous pouvez changer le nom afin qu'il soit plus représentatif de son contenu (ex. `resources`). Les noms de certaines fonctions manipulant cette structure pourraient aussi être modifiés. Ne pas oublier de recompiler votre code pour vérifier que vous avez fait correctement ce renommage.Ne pas oublier non plus de réviser la documentation des fonctions affectées.

6. Afficher maintenant le temps écoulé sur l'écran de jeu en haut à gauche. Pour cela, vous devez appeler, depuis `refresh_graphics` (pour commencer), la fonction `apply_text` du module `sdl2-ttf-light`. Bien lire sa documentation pour l'utiliser. À vous de déterminer les bonnes coordonnées et dimensions du texte. Conseil: vous pouvez la fonction `sprintf` pour construire la chaîne de caractères à afficher.

7. Dans le cas où l'on se trouve à la fin de la partie, ajouter l'affichage du message final au milieu de l'écran selon l'état du jeu.

5 C'est presque fini...

- Nous souhaitons que l'application se ferme 2-3 secondes après la détection de la fin de la partie. Implémenter une solution.

- Ce serait bête de terminer l'implémentation du niveau de base du jeu sans avoir nettoyé votre code.

- Est-ce que votre découpage en fonctions est satisfaisant? Si non, modifier votre code en conséquence.

- Avez-vous documenté votre code? Si non, améliorer votre documentation.

- Avez-vous commenté votre code aux endroits pertinents? Si non, ajouter des commentaires.

- Avez-vous placé vos nouvelles fonctions dans les bons modules? Si non, vous pouvez les changer de module.

- Avez-vous mis à jour le dépôt `git`?

Bravo ! Vous avez terminé le niveau 3 ! Vous êtes maintenant autorisé à faire évoluer librement le jeu.