Milestones

Deadline
Vendredi 08/05/2020 – 18h00

Sujet

Version 0 :

- Un joueur, représenté par un modèle non animé de votre choix, doit pouvoir se déplacer et sauter de plateforme en plateforme avec les touches du clavier et la barre espace (pour sauter).
- Pour le calcul des collisions, le joueur doit être représenté par une sphère
- Étant soumis à la gravité, le joueur peut tomber dans le vide
- Chaque plateforme étant pour l'instant un parallélépipède dont les côtés sont parallèles avec les axes du monde, et qui doit être texturé (texture de votre choix)
- Il ne doit y avoir aucun mur ou obstacle sur lequel le joueur puisse glisser (donc il n'est pas demandé d'implémenter la glissade sur les murs).
- La caméra doit suivre le joueur tout en restant à une distance fixe.
- Vous devez utiliser une machine à états pour gérer le joueur
- Le jeu doit se lancer directement dans un niveau contenant plusieurs plateformes de textures différentes
- Lorsque le joueur appuie sur la touche ESC, l'application se ferme
- Vous porterez un soin particulier à ce que les contrôles du joueur soient très agréables, aussi bien pour les déplacements que pour le saut.
- Vous devez utiliser une projection perspective

Version 1 : Menu

- Vous devez rajouter un menu au début du jeu avec les options suivantes :
  - Nouvelle partie
  - Charger partie
  - Options (binding des touches)
  - Quitter
- Nouvelle partie doit pouvoir charger et lancer le niveau que vous avez déjà créé
- Lorsque le joueur est en train de jouer dans le niveau, il peut sauvegarder à tout instant la partie en appuyant sur la touche F5
- Lorsque le joueur appuie sur ESC en jeu, il revient au menu
- Charger partie doit permettre de charger la partie sauvegardée s'il y en a une (l'option ne doit pas être disponible s'il n'y a pas de partie sauvegardée).
- Le menu Options doit permettre de rebinder les touches du clavier de façon très intuitive et pratique, comme cela se fait dans les jeux actuels
- Le menu doit être navigable au clavier ET à la souris
- Le menu doit être rendu avec une caméra ortho

Version 2 : Glide & Camera

- Rajouter des plateformes inclinées (toujours des parallélépipèdes) et des murs sur lequel le joueur doit pouvoir facilement glisser (regardez des vidéos de Mario 64 en guise d'exemple)
- Rajouter 4 touches du clavier pour contrôler la caméra verticalement et horizontalement. La caméra doit tourner autour du joueur (comme dans n'importe quel jeu à la 3ème personne)
- Le joueur doit se déplacer dans le repère de la caméra, c'est à dire que si le joueur appuie sur la touche "gauche" du clavier, le personnage joueur doit se déplacer vers la gauche de l'écran (jouez à Resident Evil 1 puis à Mario 64 si vous ne comprenez pas)

Version 3 : Gameplay

- Développer un ennemi dont le comportement est le suivant :
  - C'est un ennemi dont la physique est exactement la même que le joueur
  - Par défaut il tourne en rond, c'est à dire qu'il se déplace en suivant un cercle au sol
  - Si le joueur se rapproche trop près de l'ennemi, celui-ci se met à pourchasser le joueur (sachant qu'il peut donc tomber dans le vide, étant soumis à la gravité).
  - Si l'ennemi touche le joueur sur les côtés, sa barre de vie (qu'il faut donc coder) diminue
  - Si le joueur touche l'ennemi en sautant dessus (donc en le touchant par le haut), celui-ci meurt et disparaît du niveau.
- Rajouter plusieurs ennemis dans le niveau
- Lorsque la barre de vie du joueur atteint 0, la partie est rechargée (si aucune partie n'est sauvegardée, le niveau est simplement rechargé)

- Les ennemis doivent être spawnable et respawner au bout d'un certain temps après leurs morts.
- Lorsque le joueur sauvegarde, l'état des ennemis doit donc être sauvegardé également

Version 4 : UI/UX

- Développer une GUI in-game, rendue après le jeu avec une caméra ortho, contenant les éléments suivants :
  - Barre de vie du joueur
  - Lorsque le joueur appuie sur echap in-game, le jeu ne retourne plus au menu principal, mais se met en pause, et un menu est affiché par dessus le jeu qui est en pause (donc on doit toujours voir le jeu en pause quand on est dans le menu)
- Le menu pause doit contenir les entrées suivantes :
  - Reprendre partie
  - Retourner au menu
  - Sauvegarder partie
  - Charger partie

Version 5 : Editeur

- Développer un éditeur de niveau permettant de positionner et tourner des plateformes, et de placer des ennemis
- L'éditeur doit être le plus intuitif possible
- L'éditeur doit être sélectionnable depuis le menu principal (option Editeur de niveau)
- Il doit également y avoir une option dans le menu principal permettant de choisir le niveau à charger

Il est impératif que votre code soit le mieux architecturé possible :

- *choix de classes, de variables et de fonctions*
- *respect de la POO, encapsulation et protection au maximum des variables membres (à l'exception des structures mathématiques)*
- *pas de memory leak, code le plus optimisé possible*
- *pas de "nombres magiques" en dur*
- *une classe par fichier avec .h et .cpp*
- *Utiliser **const** chaque fois que c'est possible*
- *taille des fonctions minimale*
- *format et nommage des variables et fonctions normalisés sur l'ensemble du projet (m_ …)*
- *indentation et aération du code consistantes dans l'ensemble du code.*

Liens utiles

---

https://github.com/bulletphysics/bullet3

https://pybullet.org/Bullet/phpBB3/index.php

https://github.com/bulletphysics/bullet3/blob/master/docs/BulletQuickstart.pdf

http://www.cs.kent.edu/~ruttan/GameEngines/lectures/Bullet_User_Manual

https://pybullet.org/Bullet/BulletFull/index.html

https://www.packtpub.com/game-development/learning-game-physics-bullet-physics-and-opengl

https://github.com/ocornut/imgui

https://www.glfw.org/

http://www.assimp.org/

https://en.cppreference.com/w/

http://gameprogrammingpatterns.com/contents.html