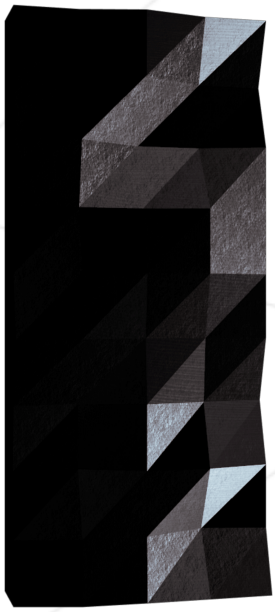

42run

Projet graphique OpenGL

Résumé: Vous avez déjà joué au jeu mobile Temple Run ?

Version: 4.0

Table des matières

I Préambule **2**

II Sujet **4**

II.1 Vous codiez ? Eh bien, courez maintenant. 4

II.2 Ce qu'il faut réaliser 4

II.3 Ce que vous pouvez ou pas faire 4

III Bonus **6**

IV Démo **7**

Chapitre I

Préambule

Nom de Zeus ! Marty, tu ne réfléchis pas en 4 dimensions !

Oh là, moi, déjà avec la 3D je suis perdu, alors...

Chapitre II

Sujet

II.1 Vous codiez ? Eh bien, courez maintenant.

Vous avez aimé “Temple run” ? Alors on va refaire la même chose. Le pitch est un peu différent : vous avez malencontreusement touché à la moto de votre directeur de campus, et donc elle/il n’est vraiment pas content-e. Bref, vous devez courir ! Et vite. Mais attention : de nombreux obstacles parsèment votre chemin, vous devez les éviter. Jusqu’où irez-vous ?

II.2 Ce qu’il faut réaliser

Votre objectif est de réaliser un petit programme qui affiche une course sans fin en 3D, en reprenant les codes du gameplay de “Temple run”, et qui se déroule dans les locaux de votre campus. Le programme doit à minima fournir les éléments suivants :

- Une vision du décor en perspective
- Le décor défile pour nous donner l’impression d’avancer
- Un décor généré aléatoirement en réutilisant un jeu limité d’objets 3D mis bout à bout
- Un décor qui reprend les éléments architecturaux et visuels du campus
- Un personnage qui reste au premier plan et que l’on peut déplacer latéralement et faire sauter
- Des obstacles à éviter et à sauter, sinon la partie s’arrête.
- Un compteur de distance

II.3 Ce que vous pouvez ou pas faire

Les contraintes techniques sont les suivantes :

- Vous pouvez choisir le langage de programmation que vous souhaitez

- Le binaire s'appelle *42run*
- Vous devez avoir un mécanisme de compilation et de création du binaire (une forme quelconque de Makefile)
- Utiliser de l'OpenGL moderne : en version 4.0 minimum, avec des shaders obligatoirement
- Vous pouvez utiliser des librairies en respectant les contraintes suivantes :
  - Vos meshes 3D doivent être créés ou chargés par votre propre code, pas une librairie
  - Vous devez créer et utiliser vos propres shaders
  - Vous devez impérativement utiliser OpenGL et son API. Vous ne pouvez pas utiliser une autre librairie qui vient chapeauter OpenGL
  - Vous devez faire vous même vos calculs de transformations matricielles (et ne pas utiliser des librairies qui le font comme glut, glfw, ...)
  - Vous pouvez utiliser la MinilibX-OpenGL (MacOS seulement) ou les fonctions équivalentes d'une autre librairie qui gère le fenêtrage et les évènements (sdl, glfw, ...)
  - Vous ne pouvez pas utiliser de librairie qui fait le gameplay (bref le boulot) à votre place
  - Vous pouvez utiliser une librairie de chargement d'un format d'image (libjpeg, libpng, ...) mais pas de librairie générique qui prend en charge plusieurs formats
- Le jeu doit être jouable sur les machines du campus

Chapitre III

Bonus

Quelques bonus sont proposés :

- Un décor particulièrement travaillé (avec de la vraie 3D, pas comme dans la démo moche)
- Des pièces de monnaie (ou des chatons) à ramasser, en plus des obstacles à éviter
- Des power-up à attraper qui donnent des pouvoirs particuliers
- Des missions ou des quêtes à effectuer
- Glisser sous des obstacles en hauteur
- Proposer des personnages différents avec compétences différentes
- Tout plein d'autres ajouts qui existent dans ce type de jeux
- Et il y a sûrement encore d'autres bonus que vous pouvez implémenter

Bon projet !

`La partie bonus ne sera évaluée que si la partie obligatoire est parfaite. Cela signifie qu'elle est complète et fonctionne sans problème. Autrement, les bonus ne seront pas évalués.`

Chapitre IV

Démo

FIGURE IV.1 – Marvin dans l'entrée

FIGURE IV.2 – Marvin passe en salle de jeux vidéo

FIGURE IV.3 – Marvin doit éviter les obstacles en cluster