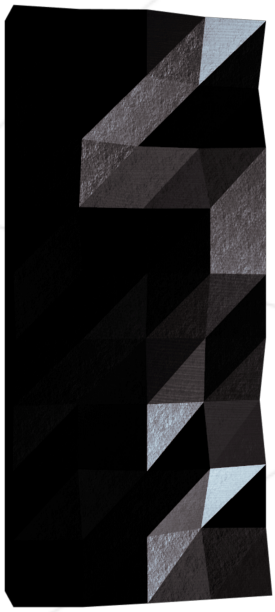

42run

OpenGL graphic project

Summary: Have you ever played the mobile game "Temple Run"?

Version: 4.0

Contents

Chapter I

Foreword

Great Scott! Marty, you're just not thinking fourth-dimensionally!!

Right, right. I have a real problem with that.

Chapter II

Mandatory Part

II.1 You were coding? Well, time to run now...

Did you enjoy "Temple run"? Well, we are going to do the same thing. The cover story is a bit different: you have accidentally touched your campus director's beloved motor bike on the street in front of the campus, and she/he is so angry at you! No other choice, you need to run! And fast. But careful, many obstacles are in your way and you need to avoid them. How far will you get?

II.2 What you need to create

Your goal is to create a small program that will present an endless run (within the school walls, to match the cover story) in 3D, using a similar gameplay as in "Temple run". The program needs at least to show the following elements:

- A set with a cool perspective.
- A set that moves forward to give an impression of movement.
- A randomly generated set using a limited number of 3D elements and obstacles glued all together.
- A set inspired by the architectural elements of your campus.
- A character that stays in the foreground, and that can move laterally and jump.
- Obstacles to avoid, and to jump over, otherwise the game stops.
- A distance meter.

II.3 General instructions

The technical constraints are as follows:

- You can choose the coding language you want for this project.
- The binary is called *42run*.

- You will have a compilation mechanism that create the *42run* binary file (some kind of Makefile).
- Use modern OpenGL, at least the 4.0 version with shaders, it's mandatory.
- You can use various libraries, as long as you follow these constraints:
    - Your 3D meshes will be created by your own code or loaded by your own code, not a library.
    - You need to create and handle your own shaders.
    - Obviously, you will use OpenGL library and its API directly. You can't use any library that comes on top of the OpenGL library.
    - Manage and compute your transformation matrixes yourself (not with libraries like glut, glfw, ...).
    - You can use the functions of the MinilibX-OpenGL (MacOS only) or their equivalent in another graphic library that handles windowing and events (sdl, glfw, ...).
    - You can't use a library that is providing the gameplay of your software.
    - You are allowed to use a regular image library for a specific format (libjpeg, libpng, ...) but not a generic library that handles multiple formats.
- The game must be playable on the cluster's computers.

Chapter III

Bonus part

Here are some ideas of possible bonuses:

- A particularly cool set (with complex 3D elements, not like the demo video).
- Coins (or kittens) to be picked-up in addition to the obstacles to avoid.
- Some power-ups to be picked-up that give special abilities or protections.
- Specific missions or quests to complete.
- Slide under some obstacles that are higher.
- Different characters with different skills.
- All sorts of add-ons that exist in these type of games.
- There are probably way more bonuses that could be implemented.

Enjoy!

The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.

Chapter IV

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.

Chapter V

Demo

Figure V.1: Marvin in the entrance

Figure V.2: Marvin nearby the arena

Figure V.3: Marvin must avoid obstacles