Computer Graphics Final Assignment

2025-26

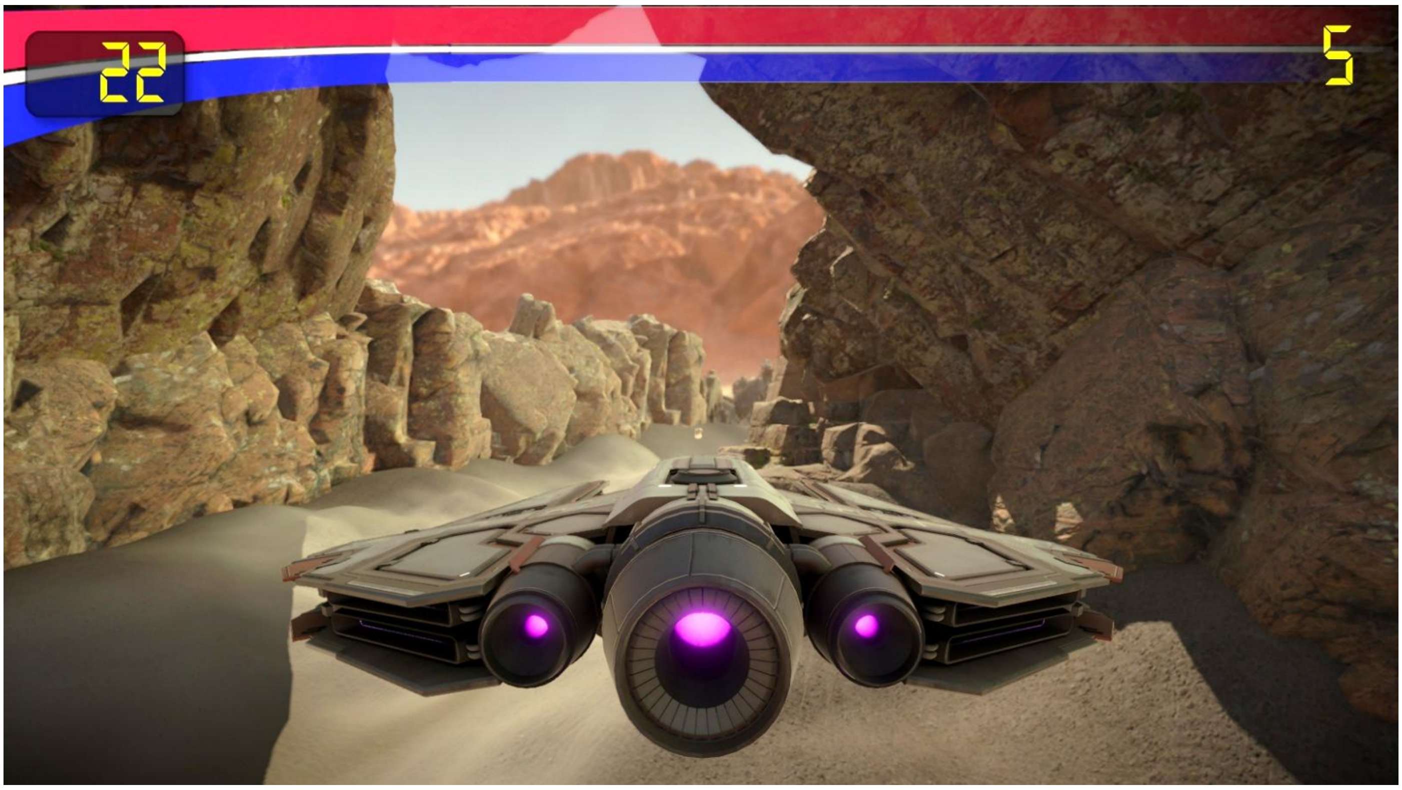

Figure 1. Example of a 3D racing game.

Overview

The goal of the assignment is to design and implement a first- or third-person racing game variant in the Unity game engine (see Figure 1). The user pilots a craft/vessel/humanoid/animal to negotiate one or more racing tracks and potential obstacles or even face enemies and gather trophies and power-ups to achieve a high score. The race could take place in a specific amount of time or rounds. Conversely, the goal could be to gather a fixed amount of trophies or pass through all required checkpoints in the shortest possible time.

Elements of the Game

The game is set in at least one pre-designed track, consisting of individual elements (e.g. here rock formations) and optionally, a terrain Unity element. The starting point is an arbitrary location in the track and additional non-static assets can be used as animated proxies for power-ups (e.g. speed, time, extra weapons or auto-pilot boosters) and pickable tokens (trophies/checkpoints). A user-specified background is also advised

A selection of pre-defined assets are provided, but students are encouraged to improvise and create their own style and premise for their racing track.

Lighting

Static global lighting should be computed based on all unmovable elements of the game. This includes baked light-probe diffuse global illumination ("adaptive light probes") and reflection probes manually positioned throughout the environment to adequately capture position-dependent reflections.

Compulsory Implementation Elements

The game must include the following:

- **Environment synthesis**, either in Unity or using a third-party modelling tool. The first is advised to allow Unity to efficiently perform frustum culling. Prefer a modular design.
- **Camera and craft/avatar control**, using the mouse, the keyboard or a combination of both. There are ready-made assets/packages in the Unity package repository to assist in this.
- **Collision detection**. Basic collision response is mandatory since it is required for triggering events in the game and keeping the moving objects in the track. Collision with the environment via Unity's physics engine is also advised for gravity-affected environments and natural collision response.
- **Lighting**. Move beyond the default lighting by introducing some form of background (back plate or environment map) and using at least one light source with shadows. Make engaging and authentic environments using global illumination effects.

Optional Implementation Elements

In addition to the above, you can use the following elements, or any other that seems appropriate:

- **Particle systems**, to render smoke, dust, sparks, debris, etc.
- **Music and other sound effects**.
- **Intro/outro screen** (separate scenes).

Teams, Deadline and Evaluation

You can implement the assignment in teams of up to 2 persons. Erasmus students can form larger groups (up to 3 persons). After the assignment deadline, you will be asked to present the project either via teleconference or physical presence. You build and present the game on your own computer, therefore you do not need to upload your executable in an external repository, unless instructed so by the examiner. However, you must **post a captured video of your game on YouTube** and optionally (yet preferably) make it public. The link must be submitted to e-class as the project deliverable (text field). **The deadline for the assignment is 13 Feb 2023**. Erasmus students can opt to present their work earlier, if this date conflicts with their own program.

Assets

You can find the assets used for the creation of the racing example of Figure 1 at:

https://cloud.aueb.gr/index.php/s/nzg5FF3P2626ojm

Prof. Georgios Papaioannou