

# Computer Graphics Final Assignment

2023-24



**Figure 1.** The 3D platform version (above) and the 2.5D version of the game (2D control, 3D rendering - below).

## Overview

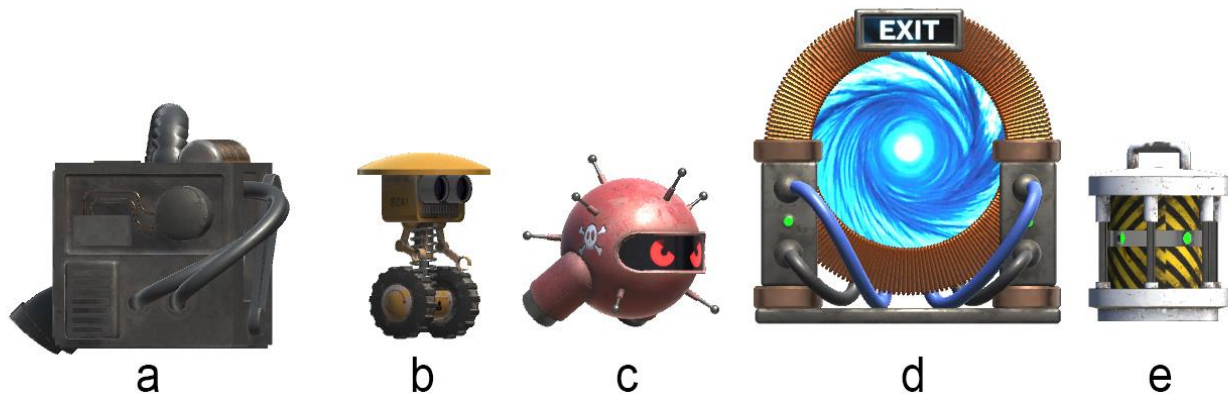
The goal of the assignment is to design and implement a first- or third-person 3D/2.5D arcade game in the Unity game engine (see Figure 1). The user controls a small robot, called Boing, to negotiate obstacles and platforms and reach the exit portal of a level, avoiding enemies and hazardous areas in the process. Depending on the concept, the robot can be floating in space (no gravity) or be controlled by gravity and must thus jump to move along disconnected platforms.

## Elements of the Game

The game has 5 primary elements, accompanied with respective 3D models as building blocks:

- **Platforms/obstacles.** They comprise the static part of the environment and define the navigation pattern through the maze-like environment. For the implementation of the level maze, you are given a multi-faceted platform block (Figure 2a) that you can rotate to create variations.
- **Boing the robot.** This is your main character that you can use to instantiate the game's avatar (in a third-person version – Figure2b). In the case of a first person 3D implementation, this is not needed. The provided asset includes an animation timeline which you can use. The animation segments are explained in Table 1 below.
- **Enemies.** They constitute predatory adversaries that either try to collide with our character or pass idly by. In both cases, contact with them should lead to some damage to the character. A premade enemy object has been provided for this purpose (Figure2c).
- **Portal.** This is the exit of the level. A proximity trigger should be placed on or near the portal to enable the successful termination of the level. A suitable multi-part and multi-material model has been provided with the option to hide elements of the geometry that may be too heavy to render on low-end devices. There is also a “vortex” geometric element that can be animated (Figure2d).
- **Power-ups.** You can place various power-ups throughout the level to assist the character by replenishing lost energy, enhancing its mobility, pass through obstacles for a limited time, auto-kill all nearby enemies, etc. A model has been provided for this purpose (Figure2e) with separate parts that can be painted differently to represent variations.

*You are free to introduce or replace assets at will to better serve the intentions of the game.*



**Figure 2.** The provided assets for the game. a) platform block, b) Boing, c) enemy, d) portal, e) power-up.

**Table 1.** Animation segments for the Boing robot.

Frames	Description
18-22	Orientation reversal transition. Obsolete - you can use a transformation animation instead.
0-10	Forward motion to the right
22-32	Forward motion to the left
10-18	Jump. The upper part and the spring are extended a little. Optionally enable this animation when not in contact with the ground. This is the full extend-contract cycle. You can also split the segment in half to represent the expansion and compression of the spring.

## 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 character 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. Collision with the environment via Unity's physics engine is also advised for gravity-affected environments.
- **Basic 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. Place shadowless sources where necessary for lighting effects and remember that shadows come with an additional rendering pass and respective cost.

## 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, sparks, etc.
- **Multiple reflection probes**, to more accurately represent specular indirect lighting on surfaces (see Figure1).
- **Baked indirect lighting**, for static geometry.
- **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. **The deadline for the assignment is 11 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 shown in Figure 2 and two background images (one HDR panorama and one simple image) at:

[https://auebgr-my.sharepoint.com/:u:/g/personal/gepap\\_aueb\\_gr/EWGEmRbJPPJli76Ba0HxZ4gB8mZwzLLTXgwNe532ola5hg?e=i8BVGm](https://auebgr-my.sharepoint.com/:u:/g/personal/gepap_aueb_gr/EWGEmRbJPPJli76Ba0HxZ4gB8mZwzLLTXgwNe532ola5hg?e=i8BVGm)