OBSIDIA GAME

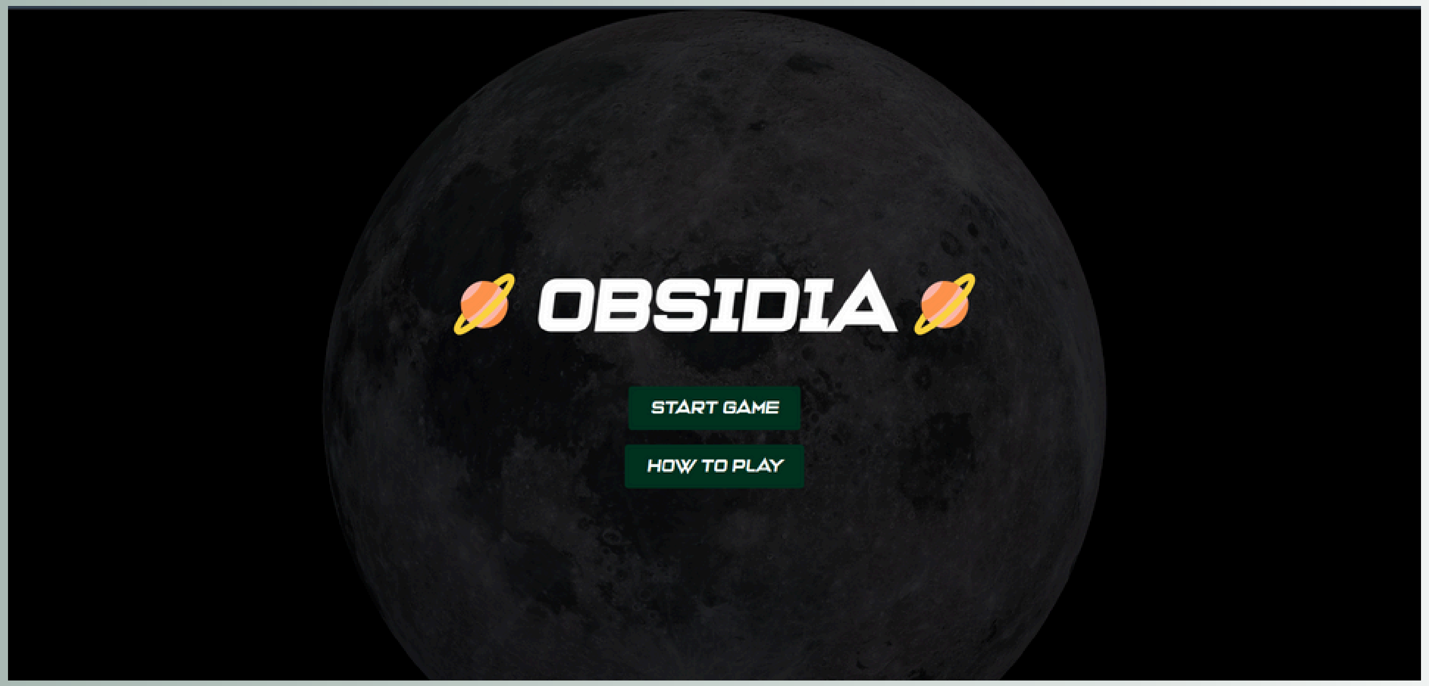
Gloria Marinelli - 2054014

Interactive Graphics Project

Overview

The project is a **3D action-based arcade survival game** where the player controls a **robot** on a platform in space, avoiding dangerous balls thrown by a giant **tentacled monster**.

The **goal** is to **survive** as long as possible while **avoiding** the deadly **projectiles,** otherwise the robot will lose a life until it reaches zero.



Tecnology

It is a JavaScript library for creating 3D graphics and animations in the browser using WebGL. It simplifies complex tasks and supports lighting, shadows, textures, and physics, making it ideal for games and simulations.

Project structure

/fonts	Contains font files
/models	Stores 3D model
/music	Contains background music and sound effects
/style	Holds CSS files
/textures	Stores image textures used for the elements
game_over.html	HTML file for the "Game Over" screen
game.js	Main JavaScript file that implements the game logic
index.html	HTML file that loads the game

Character & Monster Design: 3D Model by Sketchfab

Robot defined as “**Character**”

Monster with tentacles defined as “**Monster**”

Character & Monster Design: Animations

Character

manually moved by the player

- Jump
- Walking to the right or left

Bones used:

- leftLeg
- rightLeg
- leftArm
- rightArm
- rootBone

Monster

automatically moved

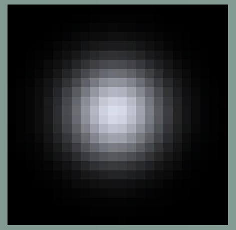
- Moving tentacles
- Throw the balls

Bones used:

- every stalk

Other elements

STARS



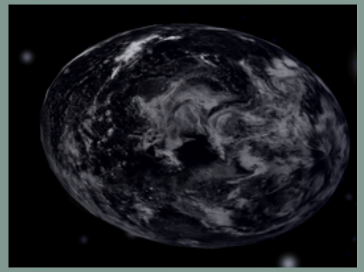
The **stars** are represented as points scattered randomly across a 3D space.

Creation Process: **BufferGeometry**, a representation of mesh, line, or point geometry

Material and Appearance: **PointsMaterial** which maps a texture of a star image

Environment: random distribution of **1800** stars

PLANETS



The **planets** are larger objects created dynamically and randomly positioned around the environment, each with unique sizes and textures.

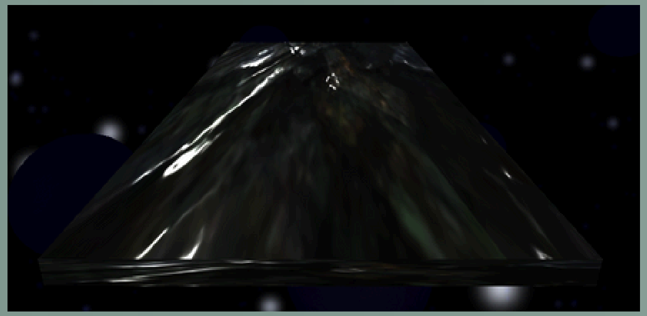
Creation Process: **SphereGeometry**, class for generating sphere geometries

Material and Appearance: **MeshStandardMaterial** which maps one of the textures from an array of planet textures

Positioning: distribution through **THREE.MathUtils.randFloatSpread**

Rotation: random rotation speed across the x, y, and z axes

BASE



The **base** is a box with specific width, height and depth.

Material and Appearance: **MeshStandardMaterial** which maps a texture

Let’s play!