Fiber Runaway

INTERACTIVE GRAPHICS PROJECT – LA SAPIENZA UNIVERSITY

De Iuliis Daniele - De Dominicis Mariacristina - Romano Remigio

2ND JULY 2018

Intro

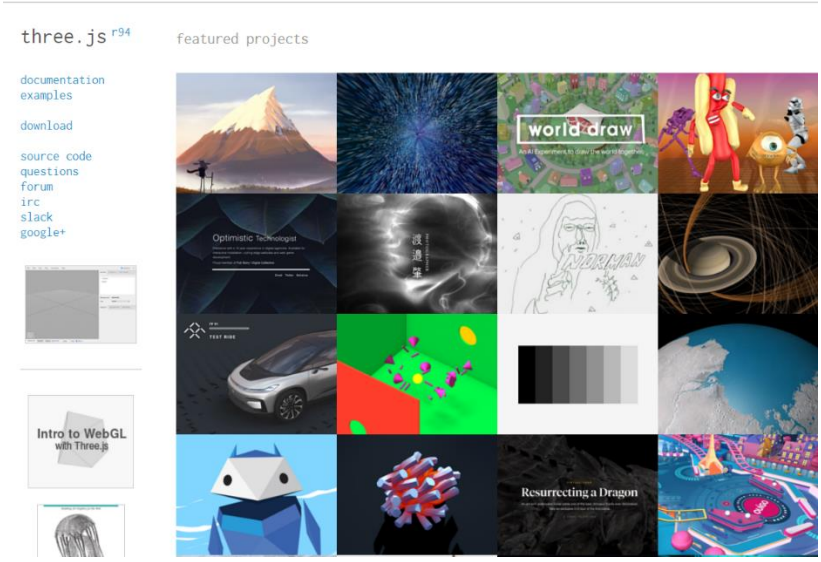
Fiber Runaway is an endless running game in which our character has to collect as many gears as possible, making sure to avoid bombs.

Environment

Three.js

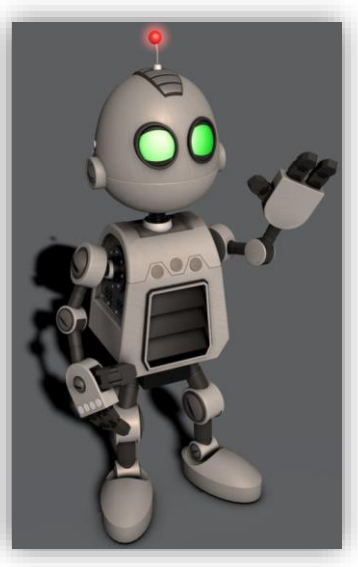
available at *https://threejs.org/*

Three.js is a cross-browser JavaScript library and Application Programming Interface (API) used to create and display animated 3D computer graphics in a web browser. Three.js uses WebGL. The source code is hosted in a repository on GitHub.



Libraries, tools and models used in the project but not developed by the team

In this project we took advantage of basic ThreeJS tools, such as scene, camera and light implementations, plus a few complex functions, like:



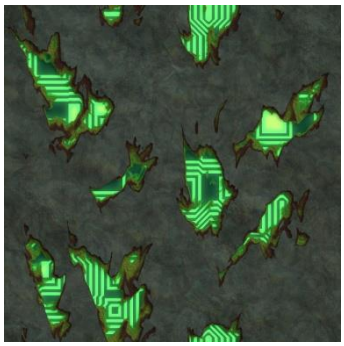
- MTLLoader.js and OBJLoader.js for the import of complex objects and their management in the application;
- OrbitControls.js to easily manipulate the camera with keyboard and mouse for debugging purposes.

Our Hero Model – Clank

We imported an external model to represent our character in the scene. We downloaded the global model (just one mesh) from *https://www.turbosquid.com/3d-models/clank-3ds-free/683985* . Then our job consisted of splitting it in several pieces, in order to obtain a complex model and to allow local control to implement the animation of each piece.

Textures and Audio

Now we list all textures used and the imported audios used for the project.



Tunnel Texture

Ground and Buttons Texture

Side Border Texture

The first two textures are available at *http://spiralgraphics.biz/packs/browse.htm* , while the border texture can be found at

http://www.thepinsta.com/hazard-stripes_Mf1OueEPcxNXbCPqVAYH1UxI0sw6nBBxS5VO7LmZB1w/

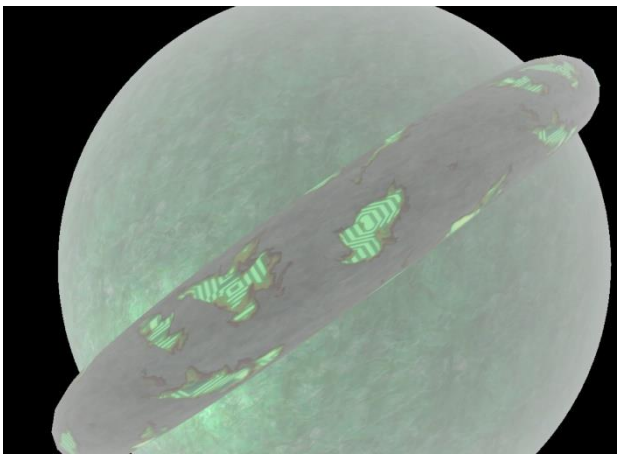
Finally, we defined four different audios, as follows:

- background.mp3 – the main game soundtrack, in continuous loop
- obstacles.ogg – it plays each time the hero collects a gear
- powerUp.ogg – it starts each time that hero collides with a powerUp object
- soundHit.ogg – this audio starts each time the hero hurts a bomb object

Description of all the technical aspects of the project

Path Generation – Path, Borders and Tunnel

We started with our endless path generation. In fact our path is nothing else than a huge sphere, upon which our hero runs. We also wanted to define some environment extras, so we added two path borders and a tunnel, all of them represented as TorusGeometry. For any doubt refer to the `function addWorld()` inside ***fiberRunaway.js*** file.



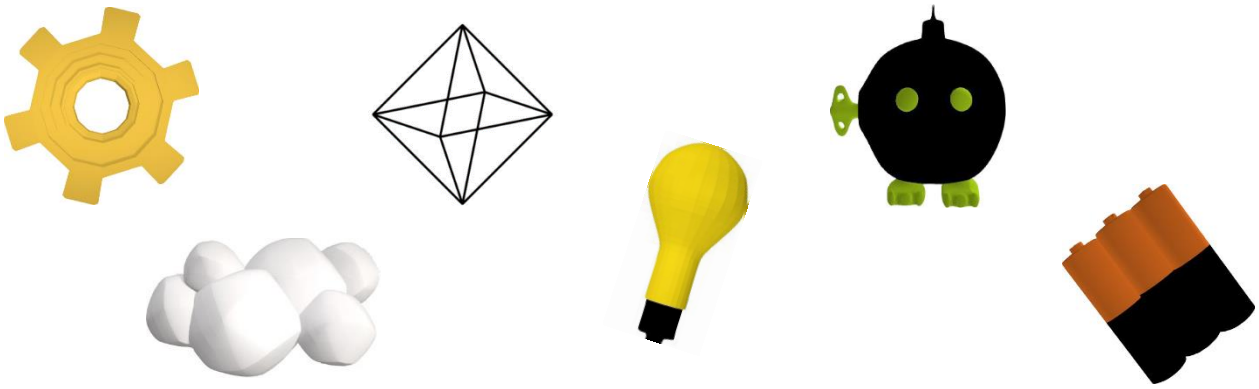
World seen from the outside, above the ground sphere

Object Modeling – Lightbulb, Bomb, Battery, Cloud, Gear, Octahedron

We created several models with the purpose of adding obstacles, coins and powerups along the path. We defined:

- Lightbulb – changes the environment lights and sets a dark mode
- Bomb – obstacle that our hero must avoid, otherwise the score decreases when hit
- Cloud – activates a stronger fog along the path, reducing visibility
- Gear – the game 'coin', that increases global score when collected
- Octahedron – in wireframe form, makes our hero invisible for a small amount of seconds

Each object import and manipulation is performed in a specific function inside ***fiberRunaway.js*** file, with the help of MTL and OBJ loader classes from Three.js library.



Environment setting – Global Parameters, Object Placement, Audio

In the `function createScene()` we set our environment and initialize a pool with our objects. So we define lights, path, hero hierarchy, object pool, textures and the audio of both game and effects. We also provided several methods, like `function addExplosion()`, to handle object collision and its logic.

Obstacles and PowerUps generation – ObjectPool

In our environment the objects, both obstacles and powerups, are placed with the following mechanism. We created a predefined object pool, with gears, bombs and powerups. Then we used the `function addPathCone()` to define a specific ratio between object categories and to generate on a random lane a new object.

This way we ensure a random factor that helps game playability and provides slightly different games every time.

Graphic aspects – Start Menu, Game Over, Restart Button, Score, Textures

By binding together different variables between ***index.html*** and ***fiberRunaway.js*** we were able to implement different screens. In particular we added:

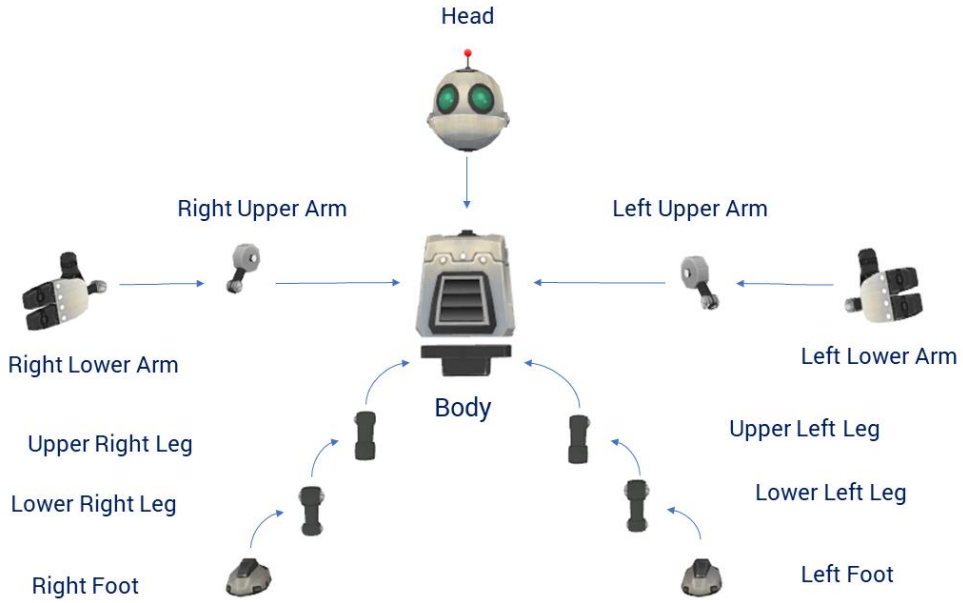
- A *Start menu*, in which we have a button to play the game and several graphic elements that describe objects and inputs. This menu disappears when the player clicks on the Play button;
- A *Game Over* screen, that appears when the score reaches a negative amount;
- A *Restart button*, that allows the player to start another run;
- A *Score* panel, in the top right side of the screen, always on and displaying of course the current score of the player

The most of this part was implemented via CSS adjustments inside ***index.html*** and via textures imported from the created models. The Start screen is displayed below.

Description of the implemented interactions

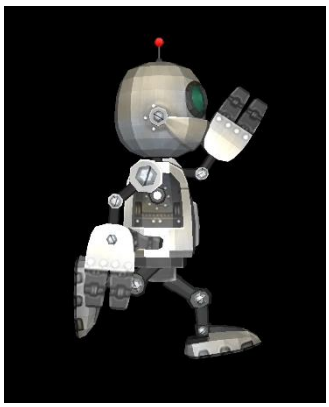
Hero movements – Keyboard Input

We added a separate file in order to process our hero's movements. In this file, ***loadRobot.js***, we defined a periodic movement for our hero, in such a way that it simulates a perpetual running towards the objective. In particular, we defined a hierarchy in which all components are represented in a relational scheme, as follows:



Then, we wrote a new function in order to control each relevant piece, following the hierachy. So we defined:

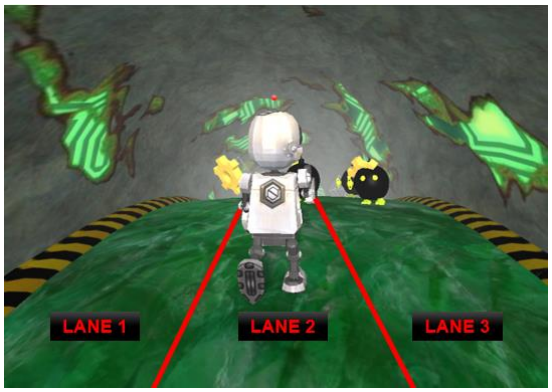
- `function moveHead()`
- `function moveRightUpperLeg()`
- `function moveLeftUpperLeg()`
- `function moveRightLowerLeg()`
- `function moveLeftLowerLeg()`
- `function moveRightArm()`
- `function moveLeftArm()`



Hero movement from the side…

… and front

Hero movements



The world sphere is split into three vertical lanes, so besides the standard hero movement, we implemented the possibility of switching to an adjacent lane and to jump obstacles.

This actions are performed through the keyboard input via either directional arrows or WASD keys.

Collision detection

Besides the hero movement, the other important implementation in the matter of environment interactions is collision detection mechanism.

We implemented collisions via a switch/case block inside the ***fiberRunaway.js*** file. Our hero collides with an object when the distance between them is less than a small amount. When this happens, the method for collision handling activates. Here we have the switch/case block introduced before. This method uses the name attribute of the object parameter, so the method always knows on which object the hero ran into.

The switch then is able to provide a different behaviour for each object. These will be described in the next sections.

Obstacles – Cloud, Lightbulb, Bomb, Octahedron

Cloud – Fog effect

Cloud collision will cause a massive increase of the fog, for a determined amount of time. After this interval the fog will return to its original value.

Lightbulb – Night Mode

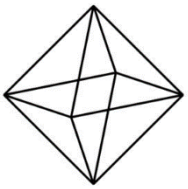
When the hero runs onto a lightbulb, this will activate night mode. This implementation consists of reducing the intensity of both ambient and directional lights. After a certain amount of time, the lights come back again as before.

Bomb – Score Decrease

The bomb hit will reduce our hero's score. Be careful: if the score reaches a negative amount, you will have to start again!

Octahedron – Transparent Mode

By reaching the wireframe octahedron, our hero will become invisible for a determined interval, so it will be harder for the player to know in which lane the hero is running.



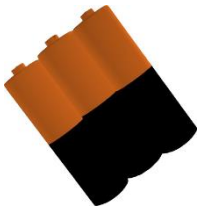
PowerUps – Battery, Gear

Gear – Score increase

The gears are our game coins: by collecting gears the player can increase its global score. So be sure to collect as many gears as possible!

Battery – Fast Mode

The battery powerup will allow our hero to run twice as fast. This means the game becomes harder, but the player can collect gears faster.



Difficulty

We also provided a further interaction with the Start menu: the player can select a difficulty level among three different ones, via click on one of the three buttons.

Tutorial – Readme.txt

Start Instructions:

1. To start the game locally you must install npm and then run a server locally.

Tutorial:

https://www.npmjs.com/package/http-server

2. Then, open a terminal and create a server in the home repository of the game

E.g. the game is in C:/Desktop/fiberrunaway

execute

cd C:/Desktop/fiberrunaway

http-server

3. Now the server is on. You can now access the repository via localhost at the port specified in the terminal after server creation.

E.g. the server is on http://127.0.0.1:8080

then go to localhost:8080 and click Enter. Enjoy.

Authors

De Iuliis Daniele
De Dominicis Mariacristina
Romano Remigio

02nd July 2018
Sapienza University, Rome
Interactive Graphics