Animal Life

Interactive Graphics
Università di Roma "La Sapienza"

Francesco Bovi 1803762
Valentina Ferrazzi 1756819

July 14, 2021

Contents

1 Introduction

The project consists in a game realized by using the Babylon.JS library. It is a powerful and simple open game and 3D rendering engine written in JavaScript. The Babylon.JS official documentation has been an important support during the project implementation. It offers us to use also the playground helps. Playground are Babylon.Js instances directly accessible from the website which allows us to quickly understand how the framework works.

1.1 General Idea and rules

The basic idea is to allow the user to move a T-rex in a forest. In the scene there are some hidden eggs and the goal of the game is to find all of them. Once the user has found one egg he has also to take it moving the T-rex in that position. Once the T-rex takes an egg, it disappears and the remaining eggs it has to take to win is updated.

The eggs have been placed in the scene in different ways: some of them are hidden behind other objects, some are placed inside bush, some float in the air at such height that the dinosaur has to jump over object or move them to catch the egg.

The game offers the user three levels of difficulty and he can select one of them from the main menu. The user can start to play by click on the play button which is placed on the earth planet near the Amazon rain forest in the main menu. The number of eggs the T-rex has to take and the time given to the user to reach the goal depends on the difficulty the user chooses. The easy level of difficulty proposes the user to catch four eggs in 90 seconds, the medium level proposes the user to take five eggs in 60 seconds while to pass the impossible level the user has to find six eggs in 45 seconds. Once the user passes one level he can choose to play again the same level or to go to the main menu and start another level. The same if the user doesn't pass the selected level.

1.2 Commands

In order to move the character the use of keyboard and mouse is needed. The game is expected to be played on a computer. The T-rex walks on by pressing the "w" letter of the keyboard while it walks back by pressing the letter "s". The T-rex can also jump by pressing the space bar. By clicking and moving the mouse the user changes the direction in which the T-rex should move. The direction of the T-rex movement can be done also by using the arrows of the keyboard.

2 Environment

The structure of the environment has been entirely created by us, only the decorative elements have been imported.

Figure 1: Game Scene

2.1 Skybox and ground

To have the illusion of an infinite and realistic environment we use skybox. So, we define at first a very big cube surrounding the scene and we apply to it a skybox forest image. Then we realise a very extended ground and we delimit the game perimeter by creating different cubic walls placed one next to each other, creating a polygon of 12 vertices. Also the ground texture is entirely crated by us: its structure helps the user to understand which is the game space where the T-rex can be moved highlighting the area enclosed in the playing polygon.

We used a physic engine Cannon.JS, applying to the scene a gravity force and providing to the ground and the perimetric walls a physics impostor through which the T-rex will collide and remain stuck in the game polygon. The physics impostor applied to the ground allows also to manage the jumping of the character in the environment.

2.2 Meshes and textures

Once the walls and the game polygon delimiter have been created, we focus on the scene elements. In order to fill the scene with the typical elements of a forest we imported different meshes, some with an already associated texture or color (in the .mtl file), others to which we selected and applied a proper texture.

In the scene there are several meshes which are used more then once such that trees, bushes, fences and eggs. The approach we use is to create a sample and then clone it as

many times we need changing its position or orientation.

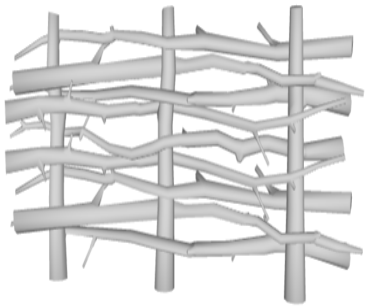
Only the model of the flower had an already associated texture, so we applied a proper one to all the others. Thanks to the diffuseTexture, bumptexture and specularTexture properties of the Standard Material we could use heat map and others texture images to have a very rough surface impression.

(a) Flowers

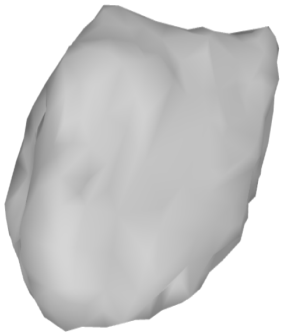
(b) Bush

(c) Little Trunk

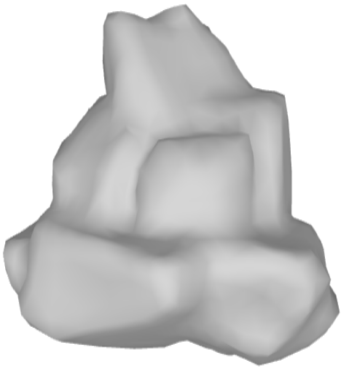
(a) Big Trunk



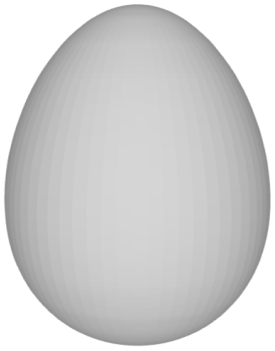
(b) Fence



(c) Little Rock



(d) Big Rock



(e) Egg

In addition to imported meshes we also create a fountain. To built it we specify an array of vertices which represents the fountain profile and then we use the CreateLathe method

over this array. We apply to the fountain mesh a proper material in order to have made of stones' fountain. About the spray water we exploited the particle system of Babylon.JS. Thanks to the ParticeSystem method and its properties we decide the number of particle the system has to be composed, we define the area in which the particle have to be emitted, the direction of the emission, the number of particle emitted per second and the particle colors. In this way we have a quite realistic water effect.

The T-rex can interact with the scene elements by hurting, moving or going through them. In order to manage the collision of the T-rex with objects we applied to all of them a box physics impostor. To allow the character to move object (Little Trunk) once hit we give to that object a very little mass, smaller than the one of the T-rex. Instead the objects the T-rex can not move have a very big mass. To reduce elastic force and avoid bounce after collision we set the restitution parameter to zero.

2.2.1 Eggs meshes

Eggs meshes are some of the most important meshes in the game. For these reason we decide to give them some particular details. First at all, eggs are highlight meshes. To create this effect we instantiate one highlight layer in our scene and then we add the meshes we want to highlight in it. The highlight color is assign when we add the mesh to the highlight layer and it is given as parameter to the addMesh() function.
Another characteristic, eggs meshes have is related to their animations. Indeed, to capture the player attention we decide to assign to each egg an animation: some eggs move up and down, some other eggs moved from right to left and vice-versa, some eggs rotate.

2.3 Lights and Shadows

To see meshes and other environment elements, there must be at least one light in the scene. We decide to use a Directional Light and an Hemispherical Light. The Directional light is emitted from everywhere in the specified direction, it has an infinite range and gives the appearance to came from far away. We decide to use it to simulate the sun light which comes from one single point and is spread everywhere. The Hemispheric light is used to simulate the environment light.

To produce a realistic scene we want each element has its own shadow. Shadows are easy managed in Babylon JS by the ShadowGenerator. The function uses a shadow map: a map of our scene generated from the light's point of view. To use the ShadowGenerator we have to specify as parameters the size of the shadow map and which light is used for the shadow map's computation. Then, we have to define which shadows will be rendered by pushing to the list of shadow casters.

3 Player

The leading actor of the game is the T-rex, that is a hierarchical model from the Babylon.JS library, with already associated textures and colors. The T-rex mesh is composed by a skeleton made of 72 bones.

Figure 4: T-rex model

To manage the interactions of the T-rex with the scene's elements, we created a little Bounding Box that we set as parent of the imported T-rex mesh. We exploited the Bounding Box to better handle collisions: the original bounding box of the T-rex would have been too big and so the mesh would have collided too easily. Having a small bound box is too easy to have more precise interaction, also the movement and the jumping of the character are improved by this choice. To allows the T-rex to hurt and jump over scene elements so we applied a box physics impostor to the Bounding Box. With the aim to change the direction of the T-rex with the dragging of the mouse we set the Bounding Box as target of the camera and set the direction of the Bounding Box equal to the one of the camera.

3.1 Animation

T-rex mesh is an imported 3D model and it has an already implemented animation that we canceled and we create our own T-rex movement. To realize the T-rex animation we use the Inspector help. It helps us to understand the bone's id to move in order to have the right animation. Before to start any animations we focus on the T-rex initial position. At the beginning its legs are on the same position so we modify the legs bones orientation in order to reach our goal. We suppose the rest position of a T-rex to be something like the one in the following figure:

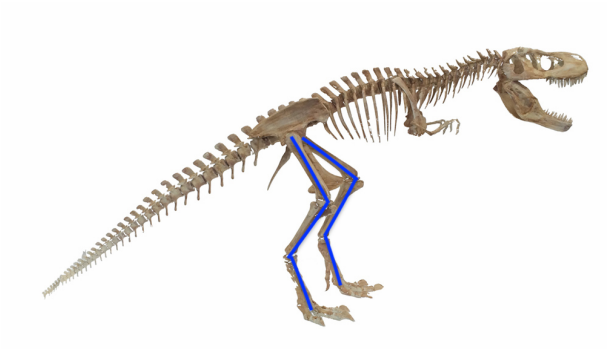


Figure 5: T-rex initial position

Once we have identified the bones which corresponds to these parts of the T-rex skeleton, by using the rotate function we change their orientation until the desire position is reached.

The T-rex animation we realized can be divided into three group: walk (on and back), jump and roar.

3.1.1 Walking animation

While walking the T-rex moves:
1. alternatively back and forth its legs;
2. alternatively up and down its arms;
3. alternatively left and right its head (only for the walking on animation);
4. alternatively right and left its tail (only for the walking on animation);

We would like to pay attention on the legs movement: it is realised not simply increasing and decreasing the upper leg orientation but also moving both the lower leg and the feet.

3.1.2 Roar animation

The roar animation is performed every time the T-rex catches an egg. It is realized by rotating the head along the Z axis and also opening and closing the mouth to better simulate the emission of the typical roar sound.

3.1.3 Jump animation

The jump animation is performed when the user click on the space-bar. It is useful to catch eggs which are on top. This animation is realized thanks to a physics impulse which is applied to the T-rex and pushes it up. The T-rex follows down due to the gravity force.

4 Structure of the game and information flow

In addition to the Forest scene we have described above, other scene have been created in order to allow the user interaction.

4.1 Main Menu Scene

As soon as the game is reached by clicking on the provided link, an initial scene which contains the main menu is opened.

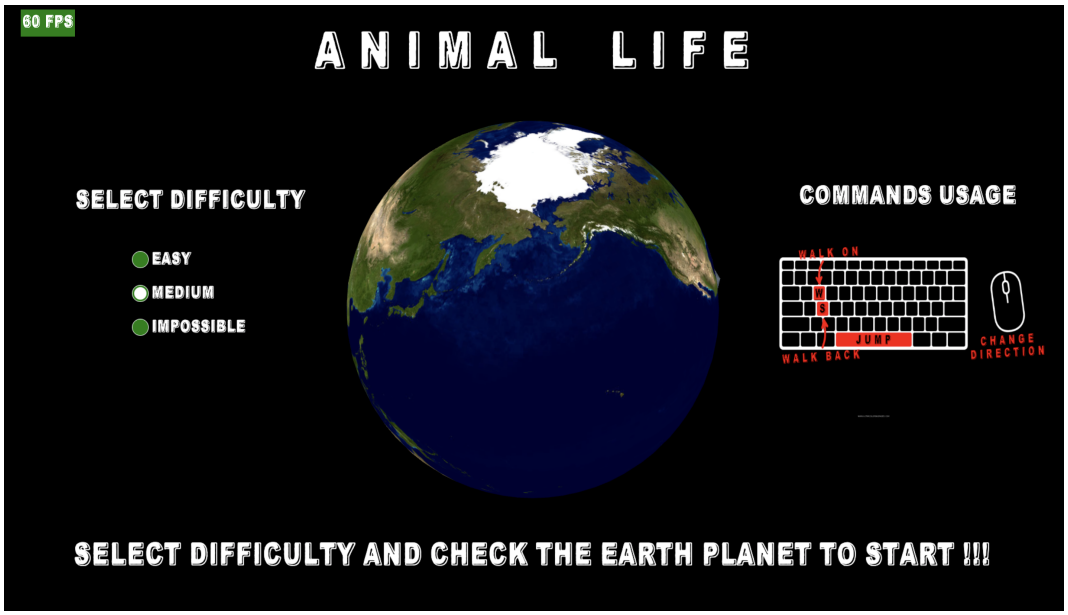


Figure 6: Main Menu Scene

This first scene is characterized by its own camera and its own Hemispheric Light (since we want an ambient light). The camera is an ArcRotateCamera which acts like a satellite in orbit around a target and always points towards the target position. The target is set to be the earth planet. We also set lower beta limit to be equal to upper beta limit and lower radius limit to be equal to upper radius limit. In this way, since the center of the planet is fixed during its rotation and the camera can not increase the distance or even change the angle among the Y axis, the camera orientation is locked and we can just rotate around the earth planet.

Other elements we have in this first scene are the GUI elements. We use TextBlock, StackPanel, Button, RadioButton, and also Sphere Panel and Holographic Button which came from the 3D GUI Babylon.JS elements. We need these two last elements to build a Sphere Panel around the earth and to attach to it a Button in order to have the illusion the button is placed on the earth surface. Thanks to the the Main Menu Scene, the user can choose the level of difficulty and start to play. The user can also understand which are the commands to use during the game thanks to an illustrative image that we have created and added to this first Scene.

4.2 Loading Scene

By clicking the Play button from the Main Menu Scene, the Loading Scene is rendered while the Menu Scene is disposed. The Loading scene is a very simple scene where the same earth planet of the Main Menu Scene rotates until all the items of the Forest Scene are loaded. Light and Camera are set as in the previous scene while the rotation animation is added in the registerAfterRender.

4.3 Forest Scene

The forest scene is where the user can play and move the T-rex character to find the hidden eggs. We have described above the structure of this scene: the enviroment is full of meshes, there are two lights and the shadowGenerator is used to have the right light-meshes interactions. In this scene the camera we use is an ArcRotateCamera which has the T-rex as target. In this way when the player moves, the camera follows it. Arrows keys and the mouse movement control the camera orientation and also its radius parameter. Since the walking direction of the player is linked to the camera orientation, by changing camera orientation we can control the T-rex movement.

In addition to the environment elements, the Forest Scene shows some GUI Elements which provide to the user important information related to the game state. Among this information there are the remaining eggs and the remaining time. Both are dynamically updated. The dynamism of remaining eggs number is realized thanks to a function which is called once the T-rex bounding box intersects the egg bounding box and it updates the relative path of the image to be shown.

4.4 Winning Scene

The user arrives to the Winning scene when he is able to find all the hidden eggs provided for the selected difficulty level. This scene is really minimal, it is just composed by GUI Elements such that Buttons and TextBlocks. The Winning Scene is used to notify the user about the win and to allow him to restart the same level again or to visit the Main Menu to change the game difficulty level.

4.5 Losing Scene

The user arrives to the Losing scene when he fails to find all the hidden eggs provided for the selected difficulty level in time. Also this scene as the previous one is really minimal, it is just composed by GUI Elements such that Buttons and TextBlocks. The Losing Scene is used to notify the user about the time out and to allow him to restart the same level again or to visit the Main Menu to change the game difficulty level.

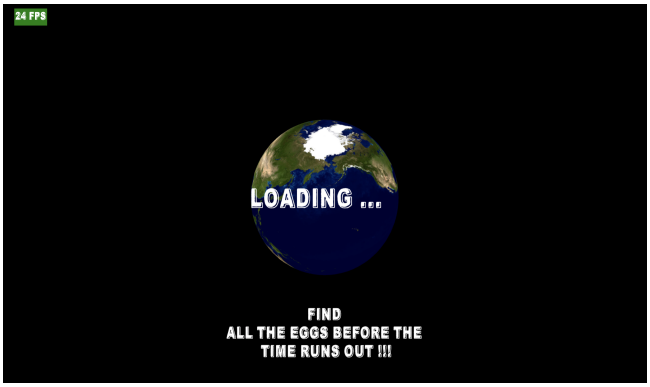


Figure 7: Loading Scene

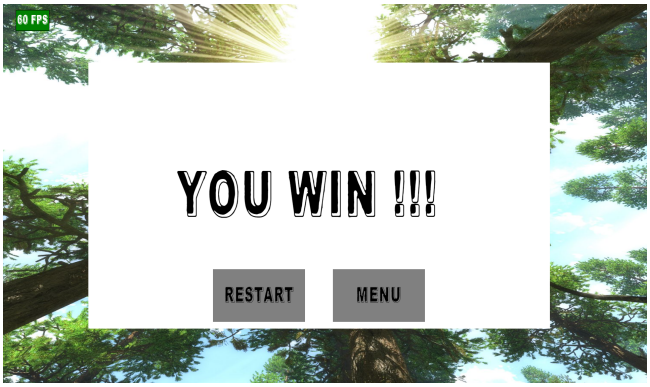


Figure 8: Winning Scene

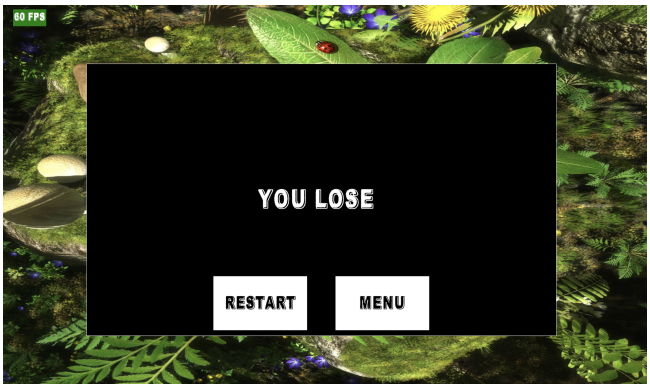


Figure 9: Losing Scene

5 Interactions

During the execution of the game the T-rex can interact with all the imported meshes and these interactions have been developed thanks to the Physics engine and the use of callbacks for mesh collisions.

5.1 Physics Interaction

Using Box Impostor we give to all the meshes of the game some physical parameters, like mass, restitution, friction and so on. Obviously we set them in order to allow the T-rex to move only the Little Trunk and prevent the movement of Big trunks, fountain and big rock if hit by the character. So, movable object has low mass while not movable objects have height mass like in the real world.
Thanks to the physics engine we also avoids the T-rex passes through objects in the environment while walking. In order to implement the jump we used a variable *jump* that is initially set to 0, then when we click on the space-bar is set to 1 (to avoid more character jumps in the air), the value of the variable is set again to 0 when the T-rex Bounding Box collide on the ground, rocks or trunks, allowing the player to jump again.

5.2 Mesh collision Interaction

The collision between T-rex and each egg is handled using the **intersectMesh()** function provided by Babylon.JS. The function takes the mesh to which check the collision as first parameter. We use intersectMesh() in the RegisterBeforeRender and since we want an egg disappears once cached, we use flags to have the interesectMesh called only once for each egg. Each InteresectMesh() is executed only if the corresponding flag is set to 0. When the T-rex catches an egg, the InteresectMesh registers the event, the number of remaining eggs is updated, the egg disappears, the roar sound and the roar animation start and the flag is set to 1 to ensure no more intersections with the same egg occur.