Interactive Graphics Project

25 September 2022

Paola Fochetti (1759100)

1. Introduction

The game evolves around the adventures of an alien racoon that travels around the universe on a spaceship. The user controls the racoon by moving it around the world following the suggestion of a vocal interface that writes its messages on screen. The main goal of each level is to collect a treasure and then leave the planet, without dying. The game consists of three levels: one tutorial and two effective levels designed in different ways and with different environment and available operations. It was developed in ThreeJs, using TweenJs for the animations and models from sketchfab. Each model was transformed into a unique file containing textures, material, eventual armature and model.

2. The Levels

2.1. Main menu

The main menu of the game is composed by a very simple HTML page that uses an image as background. From the page there is the possibility to reach every level using a link. The link to the training is on first sight, while the links to the levels are hidden under a clickable paragraph and show only after interaction with said paragraph.

2.2. Basic level design

The game screen is divided in three parts:

A top bar that keeps visually tracks of the racoon's lives and of the collected object. It is modelled as a ThreeJs scene with a reduced size.

The effective game screen, in which the user can effectively play and move around. It is, again, a ThreeJs scene. Both the scenes use the same render, this can be done by performing a cut and by defining a perspective camera and a light for each one of them.

In the game screen there are some elements that are recurrent in each level, even if different: there is always an orbiting planet, usually with an appearance that recall the level's environment, a field in which the racoon can move, and a treasure.

A bottom bar destined to the messages from the vocal interface. This bar, unlike the first one, is designed as an HTML div with black background and white text. Technically, this bar is part of the game screen, even if it not influenced by it. It is possible to change the content of the bar from any javascript file, by simply extracting the div through its identifier. The content of the bar are the messages given by the vocal interface.

Each level is characterized by a different type of environment, sometimes a different set of operations available to move the racoon and different dangers. The main light used is an ambient light. To advert the user that a level is under construction I used a custom loading manager from ThreeJs, in this way the user will not be unsure of what is happening and will have a clear visual of the game screen. The loading manager is useful especially in the second level, since there are lots of models that need to be loaded and slow down the construction of the scene. The loading manager uses an HTML progress bar that is hidden after the loading is complete.

2.3. Tutorial

2.3.1. The setting

The tutorial level is set in a "controlled" environment where nothing can put the racoon in danger. It is built as a normal level: there is a field with a road texture, an orbiting planet, a treasure to collect and a spaceship to leave the level.

2.3.2. The learning

The user learns how to move in the world thanks to the messages of the vocal interface that appears at the bottom of the screen. In this level the racoon can't move for some seconds after a message has appeared so that the user has plenty of time to read the instructions. This can be achieved by setting a variable that controls the input keys to the racoon. When the interface is talking the variable is set to false and is put back to true after some seconds, this force the key buttons to be shut down for some seconds and the user to wait and read the instructions. Reading the explanation given by the interface is fundamental because the approach to learning chosen implies to make mistakes. As an example: to learn how to jump we first have to fall in the pit that is in the middle of the field. Once fallen the interface will show the user where he can see the racoon's lives and how to avoid pits. The messages given by the interface are managed by a counter, at each value corresponds a different message and the function to change the bottom bar content is triggered through if statement that check the actual value of the counter and if a condition is verified.

2.4. Level 1

2.4.1. Setting

The first level is characterized by two environments:

A field with texture mountains and completely empty, except for the treasure platform in the middle, is used for the first part of the level. This kind of field contains no dangers for the racoon.

The second part of the level starts when the treasure is collected. The action activates a trap that completely changes the environment transforming it in a lava lake with just some platforms, roads, and planes to walk on. The lava lake is modelled as a plane mesh that is raised to cover the rock ground. From an environment with zero dangers the game becomes fully dangerous. Now the goal of the user changes and he must go back to the starting point to reach the next level. In this part the racoon can move just on planes, roads, and platforms. Of the listed only planes moves following a trajectory in loop. One particularity is that when the racoon is on a plane it has to keep moving along with the plane's movement to avoid falling. When in this situation, in fact, a check is activated that finds out if the user is on lava or on the plane even if the racoon is at rest.

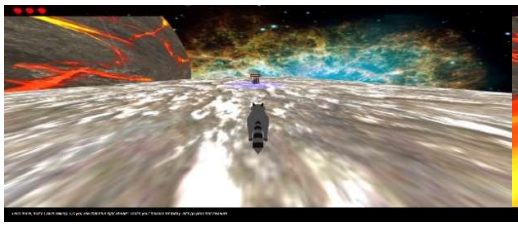

Figure 1: first part of level 1

Figure 2: second part of level1

2.4.2. Models

The models that characterize this level are the treasure, together with the platform on which it lies, and the platforms that can be used from the racoon to move around the lava lake. The treasure platforms is composed of many components (column, signal, text, treasure, platform), each one with its own parameters so that we can separately set their visibility and transforms the model at our own choice. At the beginning everything is visible except for the base platform, when the trap is activated the base platform is raised while column, text, signal and treasure disappear.

Figure 3: treasure platform model

The second model used are the platforms that are an alternative to the roads and plane on which the racoon can walk. In the level each platform appear after the trap is activated and is raised at the new ground level (the lava). Two of these assume the function of teleports and has a golden colour, if the racoon jumps on one of them it is teleported on the other one and vice versa. This is done by keeping track of which object the racoon is on and the identifier of the two teleports such that it is possible to change the racoon position immediately.

2.5. Level2

2.5.1. Settings

The second level completely changes the settings and put the racoon in an aquatic level. The main goal here is to find a ship, positioned at the end of the field, and sail with it. That's the main reason why the racoon is accompanied by the vocal interface, modelled as a crystal ball. Contrary to the first level, here all possible dangers are already in place and the user should avoid them to end the level. The field is modelled as a set of three box geometry with the same texture. To give the appearance of water the top geometries are transparent, while the bottom one is opaque. The racoon is forced to just swim, since in this level is not able to jump. This requires an initial different way of positioning the racoon that appears to be inclined such that the head is out of the water while the back is under the water. The arms and shoulder are already positioned to start the swimming operation.

2.5.2. Models and how they are used

This level contains much more model than the previous one, there are six copies for each model used: fish, shark, lifesaver, buoy. All the listed models, except the lifesavers, will cause the death of the racoon if they get in contact. Fish and shark models move in the field along the z axis in range [10,-10] such that they cross the racoon path. When they reach one of the two ends of the interval, they rotate along the y axis such that they are in the right position to move back to the other end of the field. Buoys moves on the spot, immerging themselves in the water and rotating along the x and z axis to resembles the movement given by the water. Lifesavers rotate on themselves along the z axis. All animations are in loop. Of the listed models only lifesavers are armless and act, again, as teleports. In this level there are six teleports all connected among each other and all collected inside an array, called *lifesavers*, such that each of them corresponds to an entry of the array. When the racoon detects a teleport, the latter will drag the racoon toward its centre and then teleport it. The dragging is performed by the racoon side, it is an animation that moves smoothly the racoon towards the underwater centre of the detected teleport and makes it reappear in another location. The new location is not certain, it depends from a constantly changing variable named *probabilityClock*. This variable is increased in the interval that

controls the movement of the racoon and determine the new location to which the racoon will be teleported as:

$$newLifesaver = lifesavers[round(probabilityClockMOD6)]$$

$$newLocation = (newLifesaver.position.x, 0.35, newLifesaver.position.z) + 1$$

In this way the racoon will not appear in the same location of the new lifesaver detected to be the terminal end of the teleport, but will appear on its right, avoiding any danger.

3. Collision detection

Collision detection is performed using one or multiple ray-caster at the same time, depending on the type of collision we are performing. In the game it is possible to distinguish three types of collisions:

3.1.Frontal collision:

Is used in any level and stops the racoon from hitting any object in the scene. It uses only a ray-caster that originates from the centre of the head of the racoon and goes toward the direction it is currently facing. The position of the centre of the head is displaced by 0.8, this means that when the racoon is at rest and facing forward the centre of the head is given by the following vector:

$$(racoon.position.x + 0.8, racoon.position.y, racoon.position.z)$$

The computation to find the centre of the head in any moment requires to know the actual displacement and the immediately previous displacement and add or subtract the constant 0.8 to the x or the z value, depending on the direction the racoon is facing. It can be formalized using a set of conditional statement:

Statement	Meaning	Computation
previousPosition.x<actualPosition.x	has moved forward and is facing forward	$x+= 0.8$
previousPosition.x>actualPosition.x	has moved backward and is facing back	$x-= 0.8$
previousPosition.z<actualPosition.z	the racoon has moved to the right and is facing right	$z+= 0.8$
previousPosition.z>actualPosition.z	the racoon has moved to the left and is facing left	$z-= 0.8$

A combination of these four statements gives us the centre of the head whichever the direction the racoon is facing.

3.2. Under collision

Is used in any level and stops the racoon from walking on deadly grounds or from walking wherever there isn't any ground. It uses a ray-caster that points from the centre of the racoon's body towards down. The main idea is that if the ray-caster does not intersect anything, then the racoon is out of the field and this causes its death, on the other side, if the ray-caster intersects something this may be dangerous or not. If the nearest object intersected is a platform, then there is no need to worry, but if it is lava ground then this will cause the death of the racoon. To avoid intersecting all scene's children, the objects of interests are pushed inside an array and the intersections are find among the elements of the array. The under collision is performed by the function *whereAmI* and is called each time we end a skeleton animation that moves the racoon.

3.3. Side collision

Is used only in the second level and controls if the racoon gets hit by a fish or shark when is not moving. It consists of two ray-casters, both originating from the racoon's centre and one pointing left while the other pointing right. Again, it is used an array containing all elements for which we need to look for a possible intersection. If no intersection is found this means that no danger is approaching the racon from the sides, if there is an intersection this will cause the death of the racoon.

4. The animations

All animations, except for the camera motion, are implemented with TweenJs. The trickiest of all is the racoon animation. Each of these animation moves the racoon also in space and it requires a different time from the other to be completed; the longest one is the jump. The racoon can move in three different ways: it can walk, jump, or swim. Each of these animations can be subdivided in the animation of each component of the racoon's skeleton. The main useful bones are seventeen and are grouped in this way:

- 1 for the head
- 2 for the spine
- 4 for each arm
- 2 for each leg
- 2 for the tail

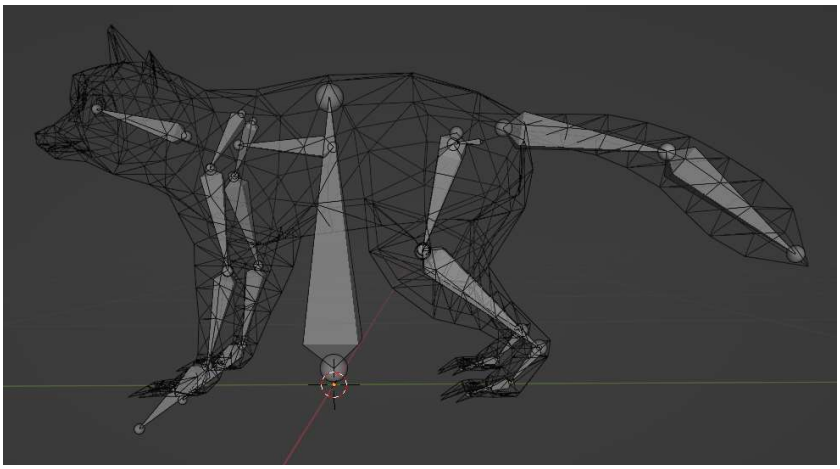

Figure 4: racoon's skeleton

4.1.Walk

The walk animation moves the racoon, in the direction it is currently facing, by 0.3. If the racoon is not performing another animation, it moves the bones too. The walk is composed by a left step and a right step and then it stops. It resembles a dog's walk: arms and legs are moved alternatively, the head and the tail perform a small rotation to the right and left at each step.

4.2.Swim

The swim animation it's like the walk, but it requires a different initial positioning of the racoon. The starting position has all body inclined and it is maintained through all the level. Arms and legs rotate in the same way as before along the x direction, but they also rotate along the z direction in such a way that they seem to be more open and gives a frog style to the swim. In addiction there is another movement of all body along the y direction that moves the racoon a little above and under water as result of the thrust given by the movement of arms and legs.

4.3.Jump

The jump is the longest animation, requiring two seconds to be performed and a completely different movement of the bones. Despite real racoons not being able to jump the animation is inspired by a flying squirrel. First, the displacement is along the y axis, the racoon moves up, stays up for some milliseconds and then begin to land. The landing is managed by an entire function that uses a ray-caster to detect the current distance from the ground and the final value for the y axis depending on the object that is right under the racoon. It is possible to jump while we walk, in this case the racoon moves towards the direction it is facing and along the y axis, describing a curve. The animation moves arms and legs in the same way: they start from the resting position, open up, reach the maximum opening when they reach the maximum height and then start to close to go back to resting position. The back of the racoon and the head rotate to make the entire movements more realistic.

Figure 5: screenshot of intermediate racoon's jump

4.4. The mechanism

All these operations are controlled by a set of keys. The jump can be obtained by pressing spacebar and it is performed only if the racoon is not moving its body for another animation. The walk and swim are controlled through a mechanism that include two events: an *onkeydown* and an *onkeyup*. The two operations can be performed through the W, A, S, D keys. To allows the user to move in any direction the pressed buttons have been modelled as an array with four entries all initially set to false. When a key is pressed its entries in the array is set to true, the racoon's body is rotate according to the entry and the value is maintained until the key is not released. In this way it is possible to keep a key pressed and to keep multiple keys pressed at the same time and move the racoon also in diagonal. The walk and swim movement are then based on an interval that, through a series of if statements, decide if the racoon has to move, check where it is currently and if it has to change position.

5. User manual

The manual is realized as follows: for each of the possible operations in a page there is a small description of what it does, all grouped as a table.

5.1.Main menu

Figure 6: screenshot of main page

Figure 7: screenshot of expanded main page

Start training	Click here to start the tutorial
Start Hunting	Click here to visualize the levels you can choose
Lava Mountains	Click here to start the first level
The Deep	Click here to start the second level

5.2. Levels

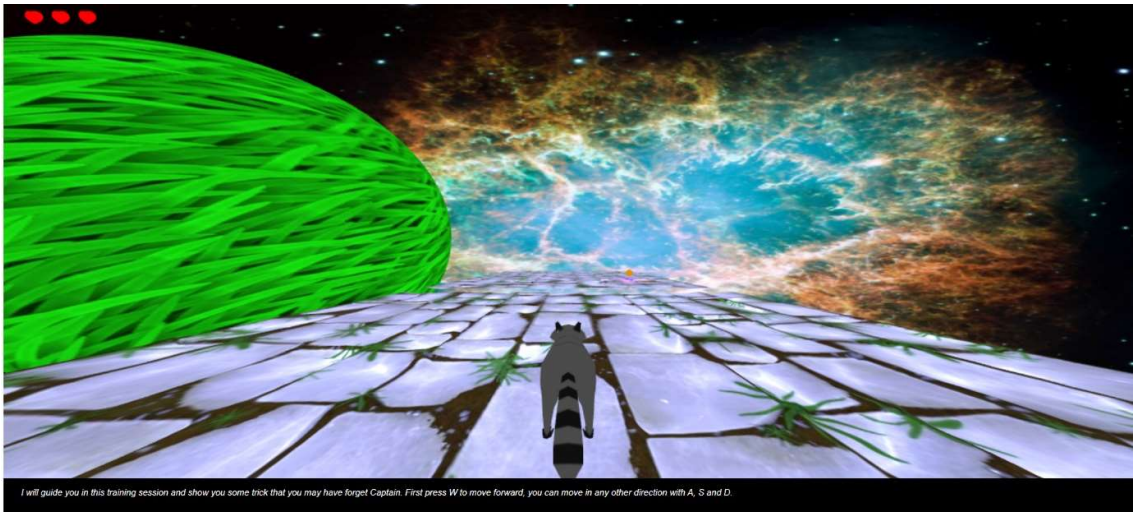

Figure 8: screenshot of training level

Upper left corner	Here you can visualize the remaining life
Bottom bar	Here you can visualize tips and suggestions that change during the game
W	Walk/Swim forward
A	Walk/Swim left
S	Walk/Swim Backward
D	Walk/Swim Right
G	Grab a treasure (available only if are near a treasure)
Spacebar	Jump (available only during training and first level.
Left arrow	Move camera forward
Right arrow	Move camera backward
R	Restart the game after game over
N	Go to next level once finished
M	Go to main menu once finished the level or after game over