

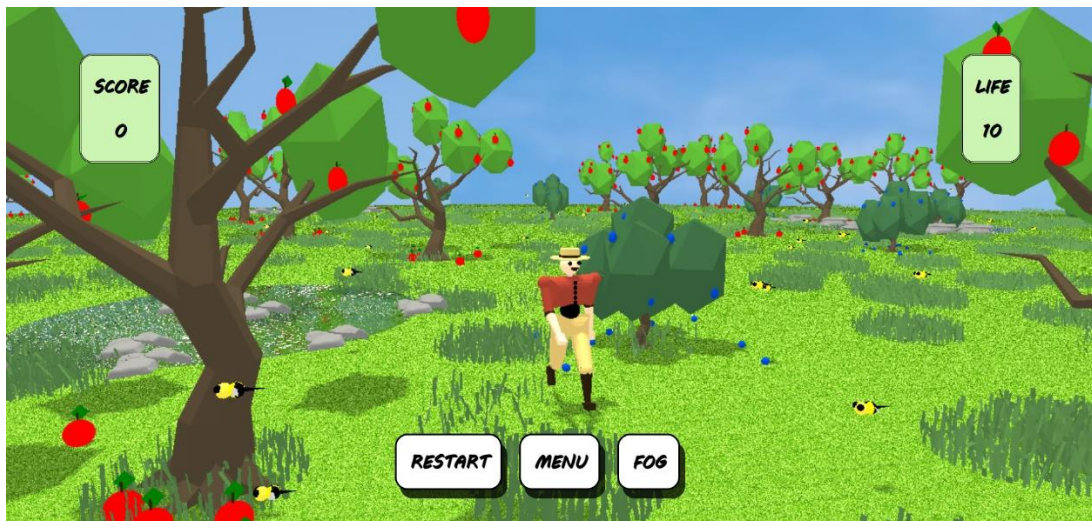
# Interactive Graphic

## Final Project Report – A.Y. 2021/2022

Francesca Andreotti

ID student number: 1696976

# *Pick Fruit!*



### ● Introduction

*Pick Fruit!* Is a single player game in which the objective is to reach the necessary score in order to win, without loose all the available life points.

### STORY

The user has to bring the farmer in the orchard toward the blueberry's brambles and apples trees in order to pick as much as possible fruit. Unfortunately, life is not always so easy .... The user has to beware from bees which will try to sting the farmer if it get

too close! Moreover, in the orchard, there are some lakes... and so there is the risk for the farmer of falling into the lakes!

## **MENU**

The game start in menu page, in which you can find the resume of the game story and two clickable buttons:

- **START GAME**, for starting directly to play
- **HOW TO PLAY**, which brings you to another page in which are explained the game rules

## **HOW TO PLAY**

When you start the game, you can move the farmer in order to reach the fruit and exploring the map using the keyboard arrows

- **Arrow up** moves the farmer forward;
- **Arrow left/right** change the direction of farmer motion, rotating him respectively to left and right

During the game, the user can change the camera view by using the mouse.

Moreover, for increasing the difficulty of the game, it is possible to add fog to the scene, by clicking the button **FOG**

If you want to restart the game, it is sufficient to press **RESTART** button

## **GENERAL RULES**

In the upper part of the screen, there are **two point counters**:

- **LIFE** (right upper corner), which represent farmer life points. It starts from 10 points and decreases of 1 each time a bee stings you

- SCORE (left upper corner), which represent the fruit quantity taken under the threes. It starts from 0 points and increases of 1 each time you approach a fruit on the ground

## HOW QUIT THE GAME

There are two possible endings:

- if SCORE counter reaches 20 points, then you win the game
- if LIFE counter reaches 0 points OR you fall in a lake, then you loose the game

## • Hierarchical models.

Almost all of the 3D models in the project are made in JS code, directly using THREE.JS library:

### Animated models

- “humanoid” Object3D is a complex model composed by different Meshes which represents a **humanoid structure** which is composed by all the main human body parts and joints. The realization of this model and its animations were be the main objective of the project and the more challenging part.

```
405 //////////////////////////////////////////////////HUMANOID//////////////////////////////////////
406
407 const humanoid = new THREE.Object3D();
408 humanoid.position.set(0,4.5,0);
409 scene.add(humanoid);
410
411 //waist
412 const waistGeo = new THREE.SphereGeometry(1,32,16);
413 const waistMat = new THREE.MeshLambertMaterial({color:"black"});
414 const waist = new THREE.Mesh(waistGeo, waistMat);
415 waist.receiveShadow = true;
416 waist.castShadow = true;
417 waist.position.set(2, 6, 0);
418 humanoid.add(waist);
```

- “bee” Object3D is a model representing a bee and its wings. In the implementation, many of them are rendered in order to build the game structure.

```
723 ///////////////////////////////////////////////////BEES MODEL//////////////////////////////////////
724 var positionBeesX = [];
725 var positionBeesZ = [];
726
727 const numBees = 200;
728 const beeRadius = 2;
729
730 for (let i = 0; i < numBees; ++i) {
731   const bee = new THREE.Object3D();
732   bee.position.set(10,4,0);
733   bee.scale.set(0.25,0.25,0.25);
734   scene.add(bee);
735   const dist1 = Math.round(getRandomArbitrary(-350,350));
736   const dist2 = Math.round(getRandomArbitrary(0,15));
737   const dist3 = Math.round(getRandomArbitrary(-350,350));
738
739   positionBeesX[i] = [dist1];
740   positionBeesZ[i] = [dist3];
741
742   bee.position.set(dist1, dist2 , dist3);
743   scene.add(bee);
744
745   const beeTexture = loader.load('./images/beeTexture.png');
```

## Environment models

- “sun” model, representing the sun in the scene
- “plane” model, representing the grass field (by setting a texture) on which the game takes place

```
92 ///////////////////////////////////////////////////PLANE//////////////////////////////////////
93 const planeSize = 500;
94
95 const loader = new THREE.TextureLoader();
96 const texture = loader.load('./images/grass.jpg');
97 texture.wrapS = THREE.RepeatWrapping;
98 texture.wrapT = THREE.RepeatWrapping;
99 texture.minFilter = THREE.NearestFilter;
100 const repeats = planeSize / 2;
```

- “sky” model, which is a sphere geometry representing the sky in the scene and containing the whole scene itself
- “lakes” model, which is made by using makeLakes() function. In order to mimic the water behaviour, in the function is

included the water model included in THREE.JS library. The bottom of the lake is made by using a texture and some rocks (obj model) are added randomly.



```
150
151 //////////////////////////////////////////////////LAKES////////////////////////////////////
152 let water;
153 var waterRadius = 30;
154 var numLakes;
155
156 function makeLakes(p1,p3, scale) {
157     const waterGeometry = new THREE.CircleGeometry(waterRadius, 30);
158     water = new Water( waterGeometry, {
159         color: '#aabbcc',
160         scale: 4,
```

- In the scene, are implemented two three models: apples tree and blueberry brambles. Both models use the tree.obj in order to implement the trunk, while the leaves and the fruits on the branches and on the ground are models which are made directly as 3D objects. In particular, the apples are composed by a leaf and the stalk meshes. Apples and blueberries models are implemented respectively in makeApples() and makeBluberries() functions.

```
259 //APPLES
260 function makeApples(x,y,z, par, scal) {
261     const appleObj = new THREE.Object3D();
262     //appleObj.position.set(0.15,getRandomArbitrary(0.1,0.09),getRandomArbitrary(0.1,0.09));
263     appleObj.position.set(x,y,z);
264     par.add(appleObj);
265     const appleGeo2 = new THREE.SphereGeometry(1,32,16);
266     const appleMat2 = new THREE.MeshStandardMaterial({color:"red"});
267     const apple2 = new THREE.Mesh(appleGeo2, appleMat2);
268     apple2.receiveShadow = true;
269     apple2.castShadow = true;
270     apple2.scale.set(scal,scal,scal);
271     appleObj.add(apple2);
272     const stalkGeo = new THREE.BoxGeometry(0.1,1.2,0.1);
273     const stalkMat = new THREE.MeshPhongMaterial({color:"#341E07"});
274     const stalk = new THREE.Mesh(stalkGeo, stalkMat);
275     stalk.receiveShadow = true;
```

## • Lights and shadows

- The lights implemented are two: an ambient light and a point light, miming the behaviour of the sun light

```
67 ///////////////////////////////////////////////////AMBIENT LIGHT//////////////////////////////////////
68 {
69   const ambientcolor = 0xFFFFFF;
70   const ambientintensity = 1;
71   const ambientlight = new THREE.AmbientLight(ambientcolor, ambientintensity);
72   scene.add(ambientlight);
73 }
74 ///////////////////////////////////////////////////POINT LIGHT//////////////////////////////////////
75 {
76   const pointcolor = 0xFFFFE0;
77   const pointintensity = 0.5;
78   const pointlight = new THREE.PointLight(pointcolor, pointintensity);
79   pointlight.position.set(0, 200, 30);
80   pointlight.castShadow = true; //we have to tell the light to cast a shadow
81   scene.add(pointlight);
82 }
```

- In particular, the point light is implemented in order to cast shadows in all the scene. All the models which are implemented in the project are set in order to receive and cast shadows

## • Fog Option

During the game, the user can choose if add in the scene some fog, in order to increase the difficulty of the game

```
41 var fogOn = 0;
42 var density;
43 fogButton.addEventListener("mousedown", function () {
44   {
45     if(fogOn==0){
46       density = 0.008;
47       fogOn = 1;
48     } else {
49       density = 0;
50       fogOn = 0;
51     }
52     const color = 'lightgray';
53     scene.fog = new THREE.FogExp2(color, density);
54   }
55 }
```

## • Camera

The camera implemented is a perspective one. By using OrbitsControl.js library, during the play it is let the user to spin or to orbit the camera around the farmer character. This function is implemented by updating the camera controls and the target position in the render() function, in order to follow properly the farmer movement during the changes of its position.

## • Animations

Most of the objects are animated and all the animation are implemented by using TWEEN.JS library.

The humanoid performs walking animation, moving the rigid body components by rotating the various joints which represent the human articulations (ankles, wrists, elbows and hips). For smoother and realistic animation, are used different tweens (chain of tweens) and Quadratic easing function (InOut mode).



```
835 ///////////////ANIMATION: WALKING ////////////////////////////
836 var start = {
837   /*rotations*/ sir:0, s2r:0, e1r:0, e2r:0, tr:0, hir:0, h2r:0, kir:0, k2r:0};
838 var target1 = { sir:Math.PI/8, s2r:Math.PI/8, e1r:Math.PI/6, e2r:Math.PI/12, tr:Math.PI/12, hir:Math.PI/4, h2r:Math.PI/12, kir:Math.PI/6, k2r:Math.PI/6};
839 var target2 = { sir:Math.PI/6, s2r:Math.PI/6, e1r:Math.PI/4, e2r:Math.PI/6, tr:0, hir:Math.PI/6, h2r:Math.PI/6, kir:0, k2r:Math.PI/6};
840 var target3 = { sir:Math.PI/8, s2r:Math.PI/8, e1r:Math.PI/6, e2r:Math.PI/6, tr:Math.PI/12, hir:Math.PI/6, h2r:Math.PI/4, kir:Math.PI/6, k2r:Math.PI/6};
841 var tween1 = new TWEEN.Tween(start).to(target1, 500).easing(TWEEN.Easing.Quadratic.InOut)
842 var tween2 = new TWEEN.Tween(start).to(target2, 200).easing(TWEEN.Easing.Quadratic.InOut)
843 var tween3 = new TWEEN.Tween(start).to(target3, 500).easing(TWEEN.Easing.Quadratic.InOut).chain(tween1)
844
845 tween1.chain(tween2);
846 tween2.chain(tween3);
847
848 const update = function () {
849   hip1.rotation.x = start.hir;
850   hip2.rotation.x = start.h2r;
851   torso.rotation.y = start.tr;
852   shoulder1.rotation.x = start.sir;
853   shoulder2.rotation.x = start.s2r;
854   elbow1.rotation.x = start.e1r;
855   elbow2.rotation.x = start.e2r;
856   knee1.rotation.x = start.kir;
857   knee2.rotation.x = start.k2r;
858 }
859 tween1.start();
```



The bee model is animated in the same way, but this time they go up and down, moving their wings.



## • Score/life System implementation

The assignment and remotion of point (score and life) is implemented during render time, in render() function, by increasing/decreasing counters and sending results and consequent messages to the HTML file

```
393 const time = performance.now() * 0.001;
394 document.getElementById("life").innerHTML = life;
395 document.getElementById("score").innerHTML = score;
396
397 ///////////////////////////////////////////////////HIT BEES
398 for(let i = 0; i < numBees; i++){
399   var dxb = Math.abs(humanoid.position.x - positionBeesX[i]);
400   var dzb = Math.abs(humanoid.position.z - positionBeesZ[i]);
401
402   if ((dxb + dzb) <= beeRadius+1) {
403     life -= 1;
404     if (life > 0){
405       setTimeout(function(){
406         document.getElementById("sting").innerHTML = "BEE STING!";
407       }, 0);
408       setTimeout(function(){
409         document.getElementById("sting").innerHTML = "";
410       }, 5000);
411     }
412     if (life == 0) {
413       window.location.href = "loose.html";
414       life = 10;
415     }
416   }
417 }
418 ///////////////////////////////////////////////////FRUITS////////////////////////////////////
419 for(let i = 0; i < numTrees; i++){
420   var dxa = Math.abs(humanoid.position.x - positionTreesX[i]);
421   var dza = Math.abs(humanoid.position.z - positionTreesZ[i]);
```

## • External Objects

The few external objects which are not implemented are the following:

- Stones.obj (which are the rocks in the water lakes)
- Tree.obj (which is the trunk of the trees)
- Grass.obj (which are the blades of grass randomly putted in the scene)



The objects are imported and loaded using OBJloader.js, which is included in three.js library.

- **Libraries and References**

- three.js
- tween.js
- <https://threejs.org/manual/>
- <https://threejs.org/>

- **Personal Conclusions**

Since this was the first time for me about implementing such kind of application, it was been very challenging at the beginning to take experience in programming this game. But now I am very glad to have take part to this course. I will go on working on this application