Andrea Leganza
MAT. 788513

Final homework

09/09/2022

1 Scene description

The scene is a diorama of an island placed in the middle of the ocean, the scene can be switched to day and night mode showing/hiding different elements.

2 Assets

Props were mainly dowloaded from sketchfab.org website in GLTF format:

- Castle
- Phoenix
- Boat

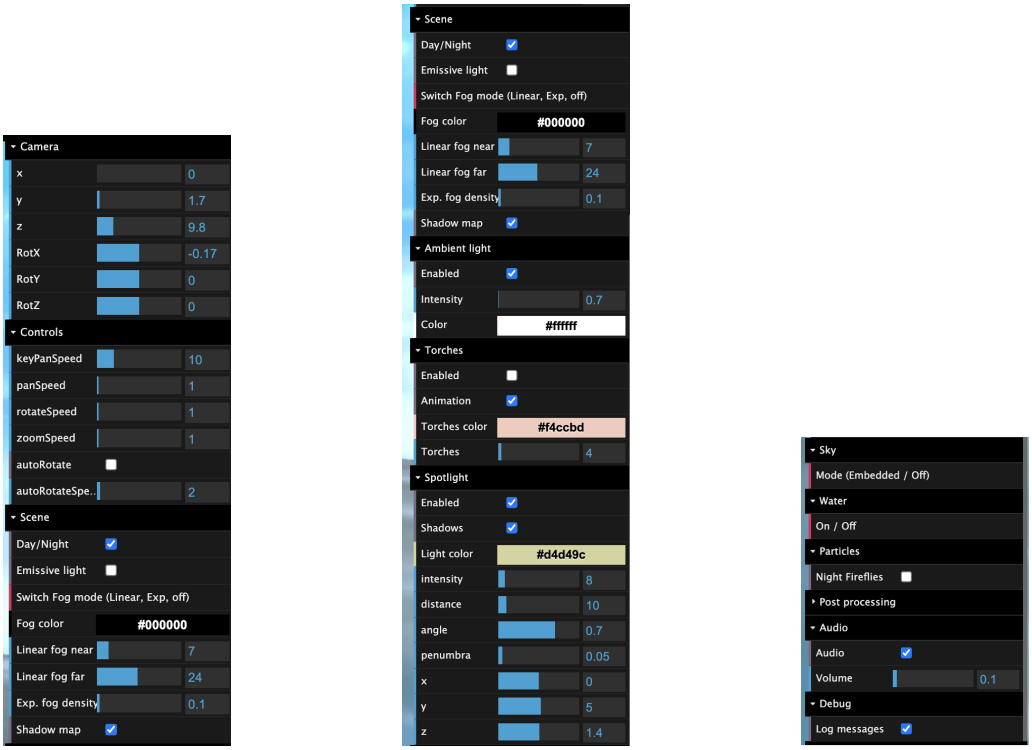
Most of the models required to be cleaned up, their structure and original materials to be modified too.

Leaves on the water were created using a simple transparent plane and applying on realtime downloaded textures, while fireflies where created using a plane with random color.

3 User interface

The GUI was developed using DAT.GUI, the whole interaction is controlled by javascript events. It provides controls to control all the aspect of the scene:

- Camera: position and rotation
- Controls: speed parameters, autorotate parameters
- Day/night switch, fog parameters, shadowmap on/off
- Ambient light: on/off, intensity and color
- Torches: on/off, animation, color, intensity
- Spotlight: on/off, shadows, color, intensity, distance, angle, penumbra, position
- Sky: on/off
- Water: on/off
- Particles: night fireflies on/off
- Postprocessing: on/off bloom/dof parameters
- Audio: on/off, volume
- Debug: on/off



(a) Menu **(b)** Menu **(c)** Menu

Figure 1: Menu views

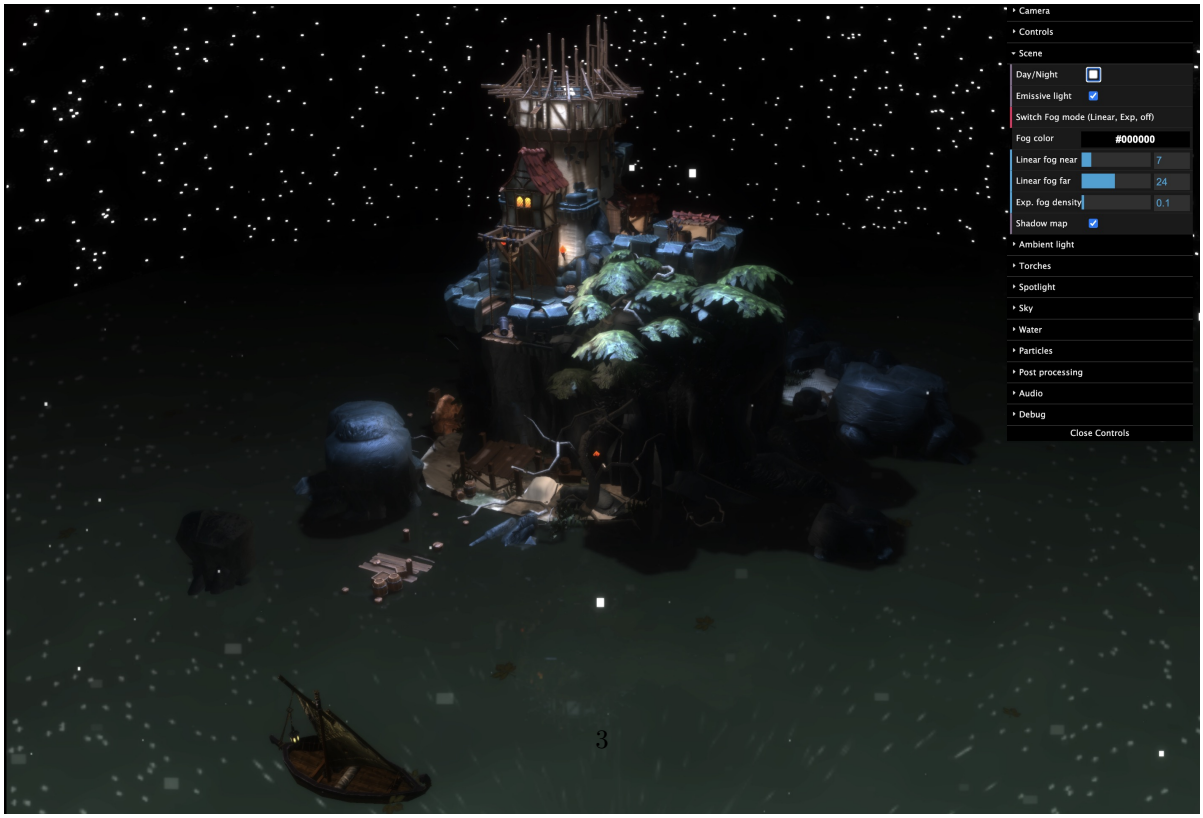
4 Day/Night mode

Switching to day/night mode using the GUI button lets to switch to different look of the scene, showing/hiding different elements:

- Day: Phoenix flies around the castle, sky rotates around, leaves float
- Night: A boat travels around the castle, night sky rotates, fireflies fly around, torches lights up and animate their flame
- The sky changes its texture from a cloudy day to a night sky.
- Fog changes to exp mode during night to increase the night look darkening the whole scene.
- Switching to night/day also customizes the lights with different colors and intensities to improve the scene look.

Figure 2: The DAY mode look

Figure 3: The NIGTH mode look



5 Torches

Figure 4: One of the torches

Torches are detected during mesh loading and to them are added:

- A spotlight is added with related animation tween
- A sphere with some faces to mimic a Platonic solid to simulate the flame

To set the color of the flame a custom shader was made, which interpolates between two colors during the time (an uniform updated during the animation loop using a deltatime value) :

```
const sphere = new THREE.SphereGeometry( 3, 4, 2 );

            const mat = new THREE.ShaderMaterial({
                uniforms: {
                    u_time: { type: "f", value: 0 }
                },
                vertexShader: `
                    varying vec2 vUv;
```

```
            uniform float u_time;

            void main() {
                vUv = uv;

                mat4 scale = mat4(vec4(1.0+sin(u_time)*0.1,0.0,0.0,0.0),
                                  vec4(0.0,1.0+sin(u_time)*0.3,0.0,0.0),
                                  vec4(0.0,0.0,1.0+sin(u_time)*0.5,0.0),
                                  vec4(0.0,0.0,0.0,1.0));
                gl_Position = projectionMatrix * modelViewMatrix * scale * vec4(position,1.0
            }
        `,
        fragmentShader: `
            uniform float u_time;
            varying vec2 vUv;

            void main() {
                gl_FragColor = vec4(mix(vec3(1,0,0), vec3(1,0.65,0), vUv.y+sin(u_time)), 1.0
            }
    `
    });
```

6 Leaves

Leaves were created with a simple double side transparent plane, loading a random local stored leaf texture; their float around the scene with a simple Math.sin movement which starts from a random generated value written/readen using mesh userData in the animation loop.

Figure 5: The leaves on the surface

```
const geometry = new THREE.PlaneGeometry( 1, 1 );
      const material = new THREE.MeshBasicMaterial(
      {color: 0x263529, side: THREE.DoubleSide, depthTest: true, transparent: true} );

      const textureName = 'Textures/leaf'+(Math.floor(Math.random() * 3) + 0)+'.png';

      material.map = new THREE.TextureLoader().load( textureName , function  (texture) {
          texture.wrapS = texture.wrapT = THREE.RepeatWrapping;
      });

      var leavesCoordPosition = [];

      for (var i= 0; i <50;i++){
          var ranXNum = Math.ceil(Math.random() * 7) * (Math.round(Math.random()) ? 1 : -1)
          var ranYNum = Math.ceil(Math.random() * 7) * (Math.round(Math.random()) ? 1 : -1)

          const pos = new THREE.Vector3(ranXNum,ranYNum,0.1)
          leavesCoordPosition.push(pos);
      }

      leavesCoordPosition.forEach(position => {

          if (debugMode){
              console.log(position);
          }

          const mesh = new THREE.Mesh( geometry, material );
          mesh.rotation.z = Math.random() * Math.PI * 2;
          mesh.position.set(position.x,position.y,position.z);

          let s = Math.random() * 0.1 + 0.1;
          mesh.scale.set(s, s, s);
          mesh.userData.delta = Math.random() * Math.PI * 2;

          floatingObjects.push(mesh);

          if (water!=null){
              water.add(mesh);
          }
          else {
              sea.add(mesh)
          }

      });
```

7 Fireflies

Fireflies were developed as simple squares of different sizes an colors embbeded inside a BufferGeometry with a PointsMaterial .

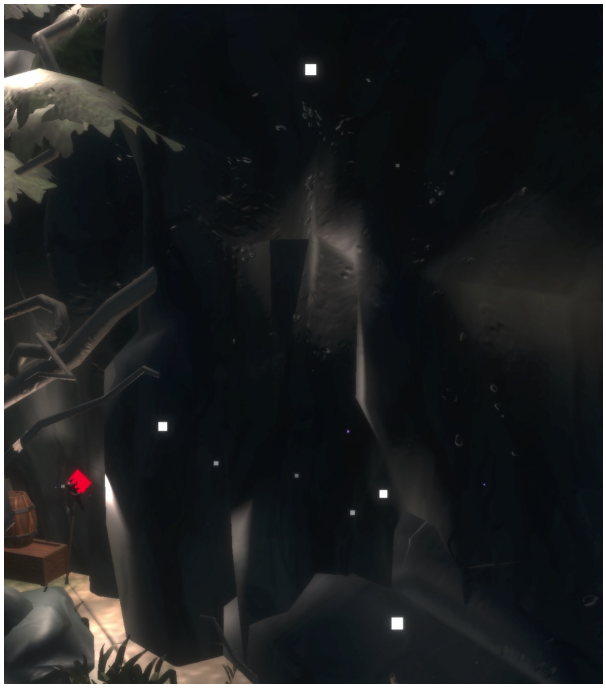


Figure 6: Fireflies flying around the night scene

8 Animations

Animations were implemented with different techniques:

- Updating rotation/position properties of mesh directly
- Updating UV offset
- Using Tween.JS
- Updating shader properties

The following elements were animated:

- Sky: rotates on z axis on day, on z and y axis during night
- Water: translates up/down to simulate low/high tide
- Torches: flame scales and rotates, their lights glow randomly

- Leaves on the water: oscillate on the surface
- Phoenix (day mode) flies and rotate around a spline (CatmullRomCurve3)
- Boat (night mode) rotate around castle and floats up/down
- Boat lantern: oscillates
- Fireflies particles: randomly move

9 Phoenix

The original model was rigged and was a sigle mesh, the first operation that was done on it was to remove the whole rigging hierarchy and split the different segments in Blender, then a proper hierarchy was created starting from the body. To speed up the animation process "custom properties" were added to each of the mesh segments to pass axis, easing, delay and movement direction which were read in three.js by accessing the userdata property; these properties were then used by Tween.js to animate as needed. All the meshes required to set an ad hoc pivot to be properly aimated.

Figure 7: The phoenix structure

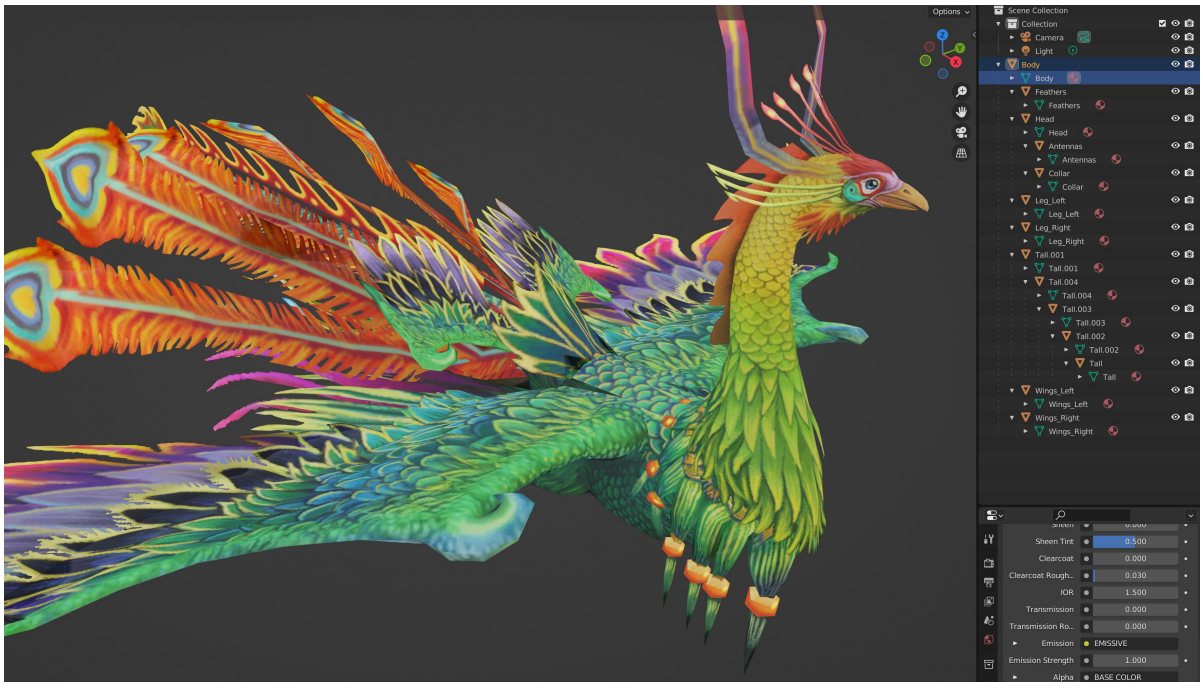
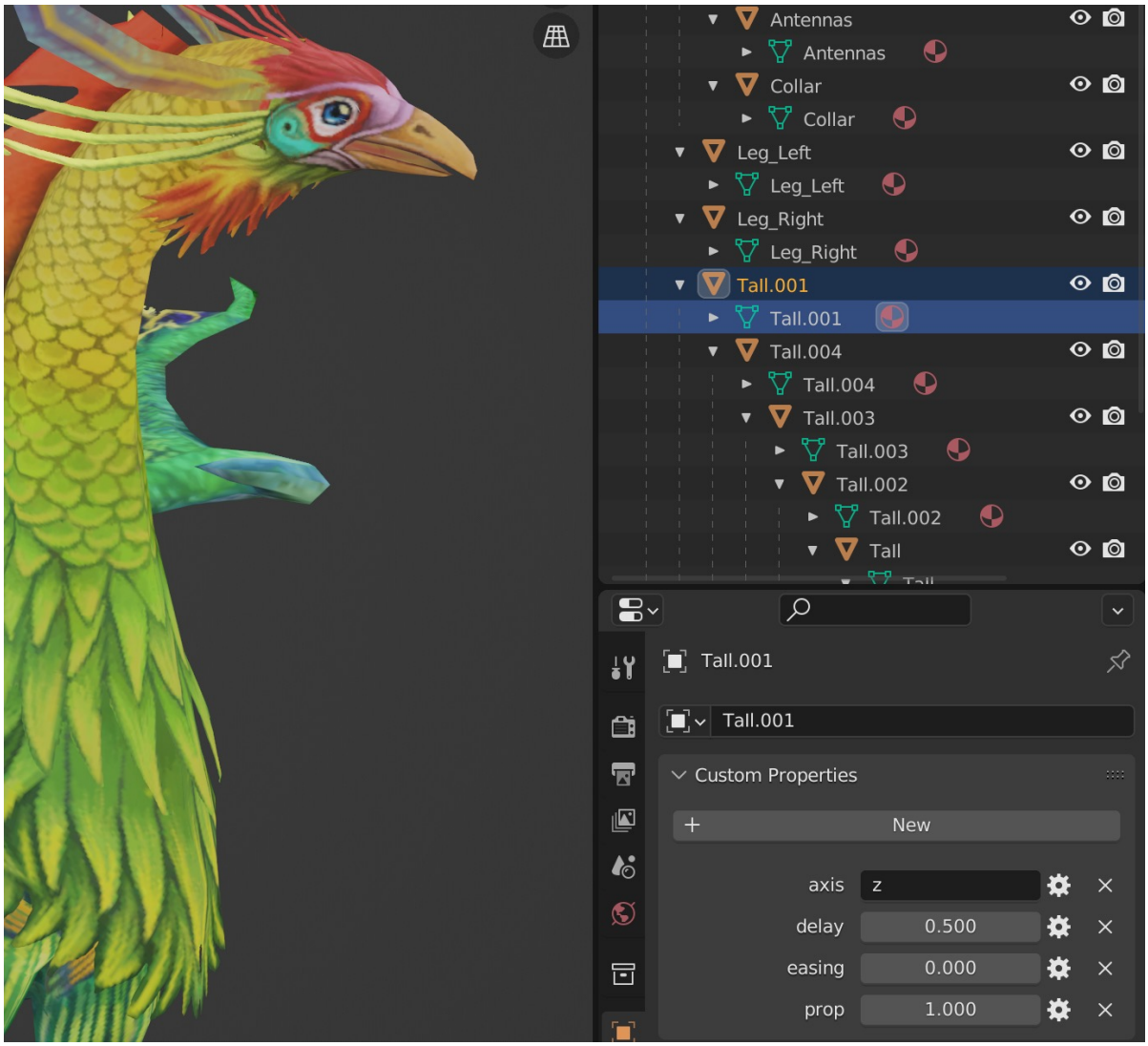


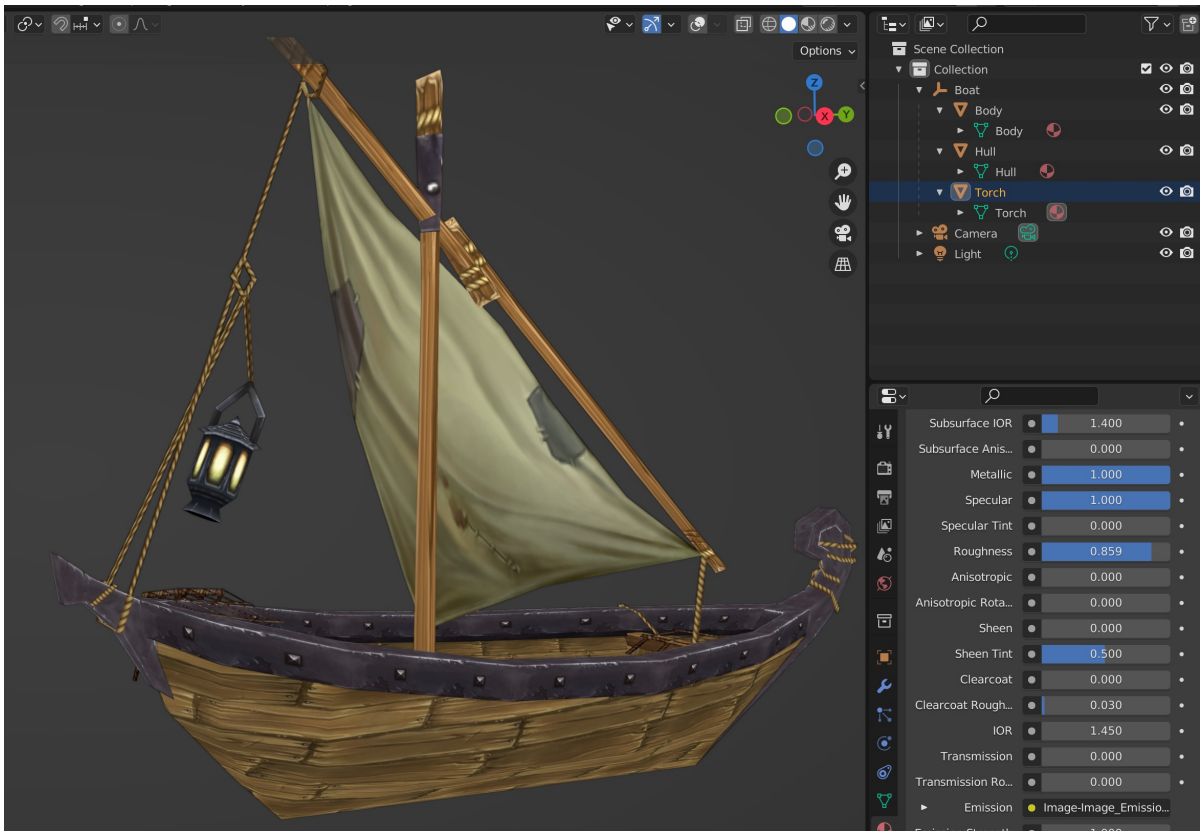
Figure 8: Custom properties used to pass data to Three.js



10 Boat

The boat was cleaned up, the pivot was changed to a proper position, the lantern was separated from the boat and the pivot set in the correct position for both of them to proper support the animation process.

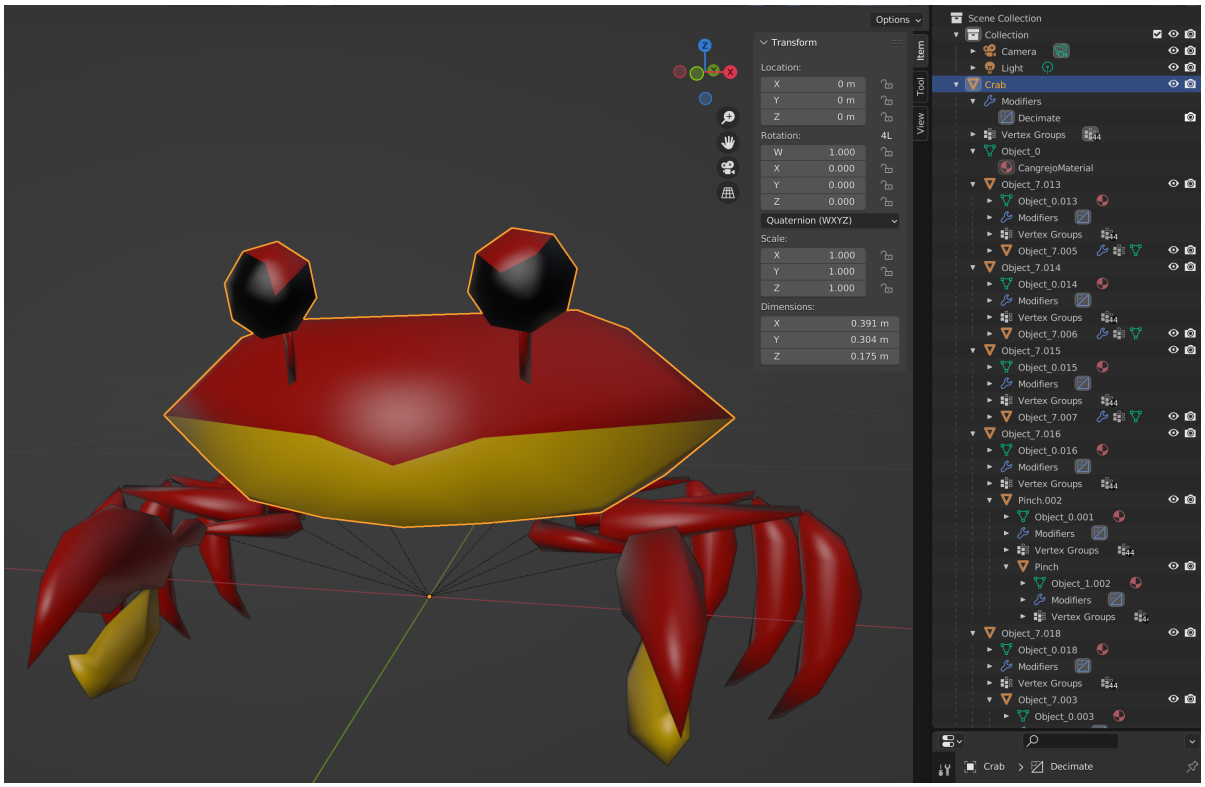
Figure 9: The boat structure



11 Crab

The crab was cleaned up, the mesh was reduced using Blender modifier, the pivot was changed to a proper position for each segment, crab legs are animated using Tween.js usig the same approach of the Phoenix usig model custom properties.

Figure 10: The crab structure



12 Textures

The following textures were used:

- Base/Albedo map
- Normal map
- Emissive map

For some models the normal map was generated from base/albedo map using an online tool; for emission some maps were generated by hand modifying in Krita the base map; the emissive intensity effect was increased during the load of the models and a GUI option is available to change its value.

13 Lights

The lights used in the scene are different:

- An Ambient Light to light the whole scene
- A Spotlight to light from the top the castle
- Point lights for each torch lights to simulate the light of the torches

14 Audio

A seashore audio loop sound effect is provided, during the day mode a seagull sound is played too, while during the night a crickets soud is played.

15 Render code