Interactive Graphics

Final Project
Luca Pierdicca 1633384

In the presented project a naturalistic scene is depicted. It is made of a small land made of ground where dandelium flowers bloom and a central bare tree is born in the middle. At the end of the blooming flowers give their petals to the tree turning it into a lush cherry blossom. All 3D models are developed exploiting the library `Three.js`, the animations are either physically driven or keyframed exploiting the library `GSAP.js` .

Structure

The structure of the code fully exploits the built-in Three.js scene graph framework. Therefore the internal representation of graphics concepts like camera, lights, geometry, material and meshes are completely transparent to the user.
Leveraging on this agile framework, the code is logically organized in these files:

- `main.js` - scene initialization and render loop
- `camera.js` - camera definition
- `lights.js` - light definition
- `meshes.js` - geometry, material and helper data structures definition
- `controls.js` - keyboard controls definition
- `index.html` - external libraries loading

In the following a detailed description of the files is given.

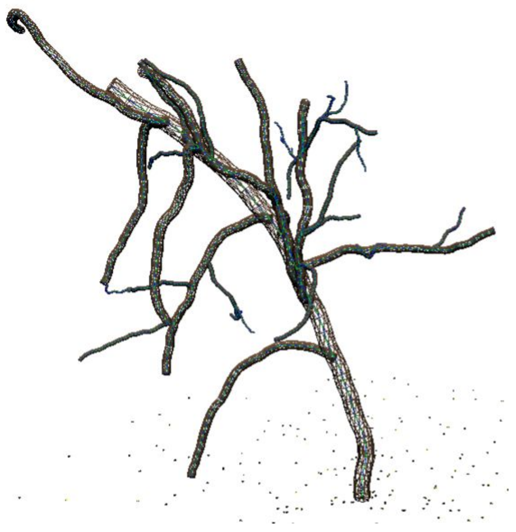
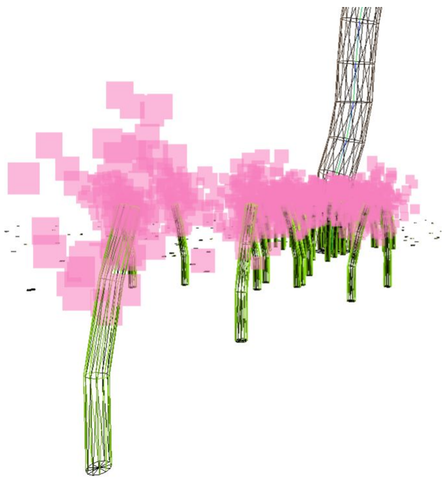
Modeling

The scene is made up of the models of a tree, a land, a flower and a sun.
At a geometric level **the tree** is modeled as a merged `CylinderGeometry` rigged with a `Skeleton` and meshed into a `SkinnedMesh`. The bones hierarchy and the bones parameters are procedurally built following ad-hoc rules and probability distributions that are (supposed to be) such that the resulting model recreates a realistic axial tree shape and branching pattern. Therefore they are based on parameters like the number of bones (the length of a branch), the number of expansion (the number of tree branches), the bone rotations (the wiggly pattern of branches) and so on.

The flowers are made up of a stem and several petals. **The stems** are modeled as a `CylinderGeometry` meshed into a `SkinnedMesh` and **the petals** are modeled as a custom `Geometry` and meshed into a `Points` mesh - this will constitute a particle system initially at rest attached to (children of) the tip bone of the stem. The bones hierarchy in this case is fixed a-priori whereas the petals geometry is procedurally built in such a way that petals are uniformly distributed into a spherical shape. The distribution of the flowers into the ground is procedurally driven as well, they are uniformly spread over the surface. The instancing technique is used in this case: the flower stem meshes are parameterized clones of a prototype mesh, however the petals are not, this is due to the fact that their geometry needs to be modified later in the render loop.

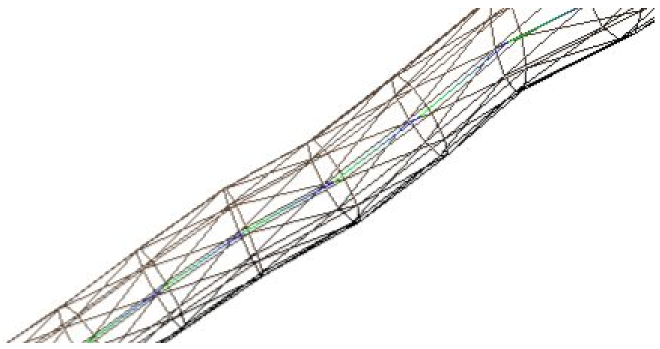
The land is made up of a merged top and bottom part. The top part is modeled as a `CircleGeometry` and the bottom part is modeled as a `LatheGeometry` half sphere whose vertices are procedurally shifted following an ad-hoc rule in order to recreate ground terrain depressions. Finally **the sun** is modeled as a `CircleGeometry`.

Here follows some images depicting the wireframed models along with their skeletons:

Animation

All animations leverage on the skeleton of the models. Skinning is a very useful technique however here it is used in a very simple fashion: the cylinder models are made up of a number of vertical segments that is equal to the number of bones, and all the vertices of the geometry that pertains to a particular segment of the cylinder are assigned to one and only one bone (the nearest one) with full weight value. In other words, modifying the transformation of a bone influences only its nearest vertices and not other vertices so the transformations are separated - they are linearly interpolated with a weight equal to 1 for the nearest bone and with a weight equal to 0 for the others. Looking at the image it is possible to see the bones that are positioned at the center of each ellipse and the links that go from one bone to the other. Ellipses represent vertices, each vertex can be mapped to maximum four bones with different weights. In the skinned models used in this project each vertex is assigned to only one bone, the nearest bone, with a weight value of 1. Usually skinning is done with more complex weighting schemes in order to reach realism for the animation of a mesh, however even this sample scheme gives a nice result for the presented scene.



The flowers and **the tree** are animated in the following ways: the growth of the stems is (bone) keyframed in a way that resembles a realistic sequential internode growing developmental model and expansion. The effect of gravity is simply simulated by consistently arching the bone structure along its length. The petals will appear almost at the end of the stem growing time resembling the blooming - this is realized keyframing the opacity parameter of the material. The flowers react to light so the growth is triggered only if they are illuminated.
The growth animation of the tree instead does not model a realistic development, it is based on (bone) keyframing as well but it resembles a sequential appearing ordered by axis level so firstly the central axis will appear and then all the axes of level 1 and so on. This is aesthetically appealing anyway. The tree animation starts right at the beginning of the execution.
At the implementation level all the animations are made possible and easy by exploiting the `timeline` structure and an ad-hoc tweens time positioning scheme in the timeline. In particular each flower has its own timeline which is advanced independently during the rendering loop depending on the illumination.

The petals animation is physically based. They constitute a particle system which at the end of the blooming starts moving towards the branches of the tree - actually towards the tip bone position of each axis but the central one, the tree trunk. The mapping between each petal (particle) and each branch is fixed a priori in a random fashion. In details, the petals move according to a proportional reactive control law:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{u}(t)$$

$$\boldsymbol{u}(t) = K(\boldsymbol{x}_d - \boldsymbol{x}(t)) + \boldsymbol{\epsilon}$$

Where the input **u** is the velocity of the particle, K is a scalar, **x** is the state of the particle (the position in space), $\mathbf{x}_d$ is the desired position (the bone tip position of a branch) and **ε** is a random noise vector. Everything is tuned in such a way to obtain a somewhat random motion towards the branches and to obtain the effect that particles when they arrive in the vicinity of their desired position continue moving around it - they can be seen also as a swarm of butterflies flying over the branches.

Materials - Lights - Textures

There are three white colored lights in the scene: two `DirectionalLight` up above the land pointing towards the tree and one `SpotLight` directly on top of the tree.
Depending on the model, `MeshLambertMaterial`, `MeshPhongMaterial` or `MeshBasicMaterial` have been used. Where lights shining effect are needed the Phong material is exploited as for the ground land whereas where lights should not affect the material Basic is exploited as for the sun.
To obtain a more realistic model of the wrinkles of the ground and of the tree bark as well as of the paper sun, different textures are used exploiting (only) Bump Mapping. For this reason this the final mesh material is not affected by the image resulting in a less artificial aesthetic.

Interactions

The possible interactions are the camera position and the spotlight direction and activation/deactivation.
They are all keyboard driven:

- WASD - moves camera up/left/down/right always looking at the tree
- directional arrows - moves the spotlight target on the surface of the ground
- space bar - turn the spotlight on/off

Conclusions

This project presents a first approach with the library Three.js. It is a pretty versatile library to develop 3D contents w.r.t. WebGL where all the useful data structures are to be done from scratch. However the modeling part is in general much easier using an external tool like Blender or Unity.
In particular, the mesh skinning operation turned out to be rather difficult at first, the more a model grows in structure complexity the more difficult it is the assignment of bones and weights to its vertices. This is one of the reasons why a simple assignment has been used for the used skinned models.
Another experimented limitation of Three.js is the impossibility to use non-uniform scaling in a hierarchical model. Scaling is inherited between parent-children objects in hierarchical models, and this is very handy in the case of the realized tree and stem models, however, the scaling must be uniform otherwise rotating children results in a shear-like effect. This fact did not allow the modeling of the axes and the stems as a cylindrical shape with variable radius along its full length and a constant height (not affected by scaling) between the cylinder segments.
Difficulties arose also in the particle system animation indirectly due to the use of a skinned mesh for the stems, a detaching operation of the point mesh turned out to be necessary for the correct final motion.

Finally as a possible improvement I'd like to add a simple model of an insect moving/flying interactively between branches of the tree exploiting a simple kinematic model and a real time procedure to plan its path.

References

Three.js site - https://threejs.org/
GSAP.js site - https://greensock.com/gsap/
A. Lindenmayer, P. Prusinkiewicz - The algorithmic beauty of plants