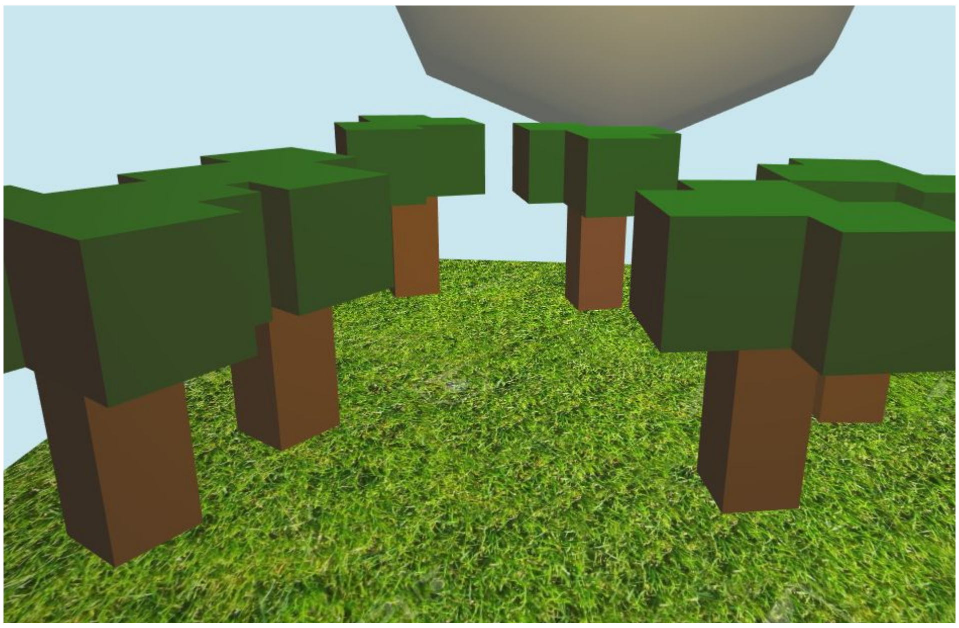
EE497: 3D Interface Technologies

Assignment 1



Enric Moreu

Introduction

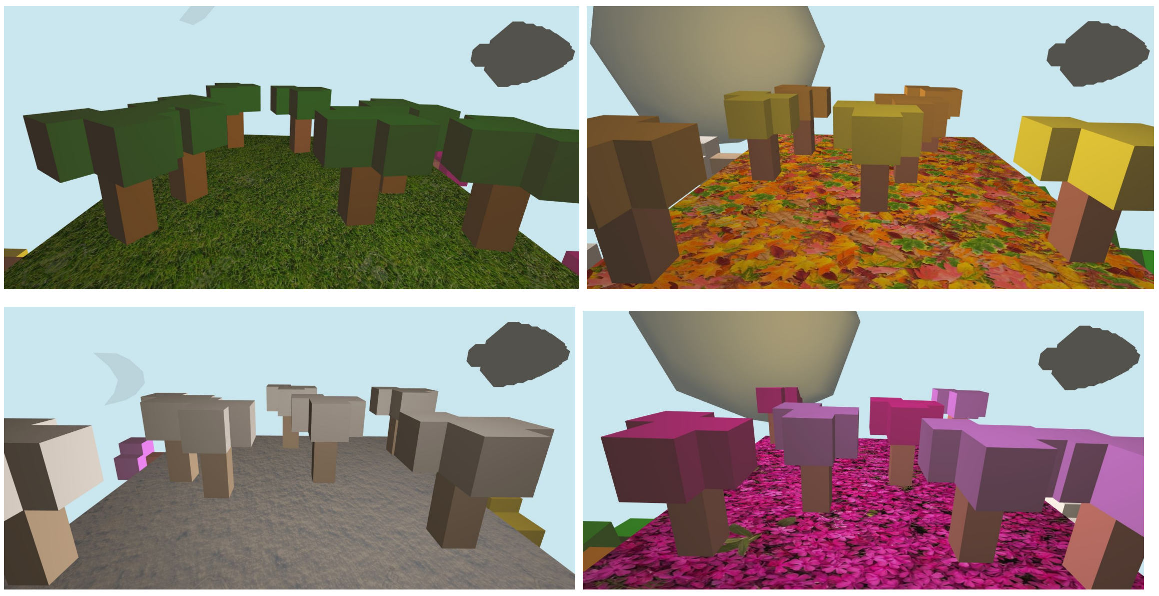
In the first assignment of the subject we will use OSGJS, a Javascript WebGL framework based on OpenSceneGraph. It allows us to create basic shapes and apply spatial transforms like rotations and translation. In addition, we can add materials, textures, lighting and interactions to generate complex scenes.

The objective of the assignment is apply the techniques explained at class in a custom 3D scene. The scene have to demonstrate the following aspects of OSGJS:

- Group nodes
- Custom geometry
- Lighting
- Texture mapping
- Animation
- Interaction

My custom 3D scene: the 4 seasons

My OSGJS implementation is a representation of the 4 seasons that illustrates the changes of a set of trees during the year.
The trees grow in a cube that represents the Earth, which is surrounded by the Sun and the Moon.

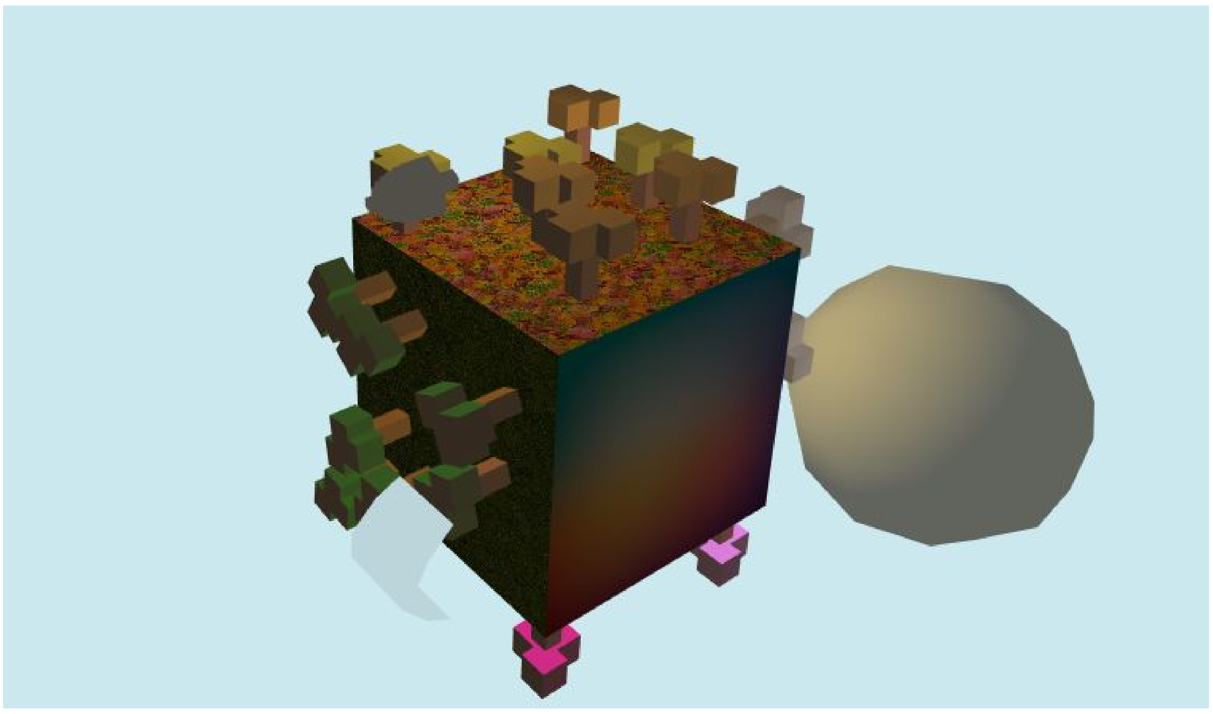


The custom scene is focused on animation and grouping different objects to move synchronously. It also includes, interaction, dynamic lighting and textures.

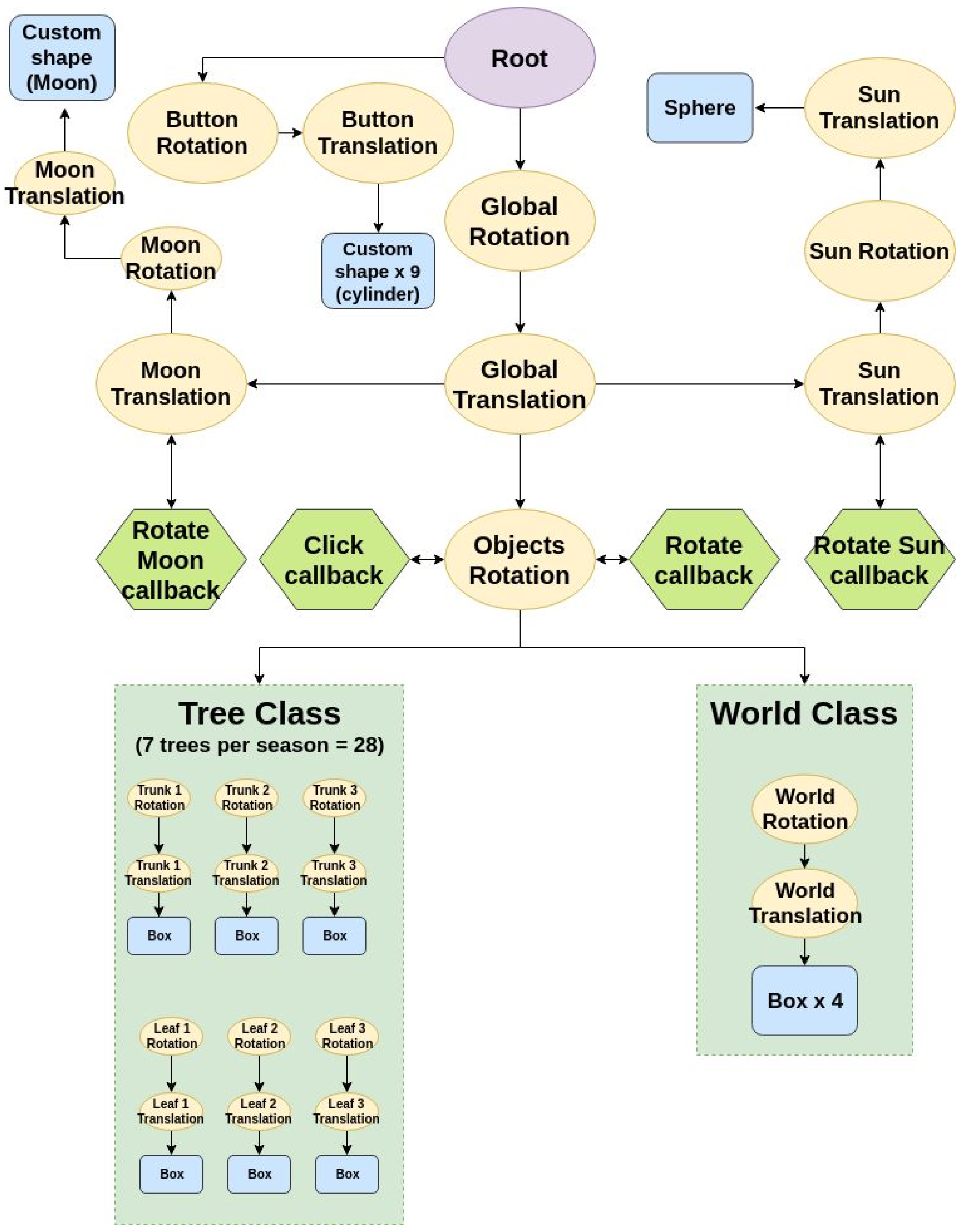
Scene graph design:
The scene is orchestrated by a global variable called status, it indicates which state is displayed (0 for summer, 1 for autumn, 2 for winter and 3 for spring).

The status is updated by a Button, that allows the user to navigate through the scenes. Note that the button is the only item which is not affected by any Transform, it is static because his direct parent is the Root.

In order to spin the trees and the Earth when we switch the scene, I implemented a Global Rotation, followed by a Global Translation that is inherited by the Objects Rotation. This “objects” local rotation is used by all the trees and Earth. The rotation is updated by a Rotation Callback, that follows the global variable status. Note that the Sun and the Moon are not affected by the Objects Rotation transform:

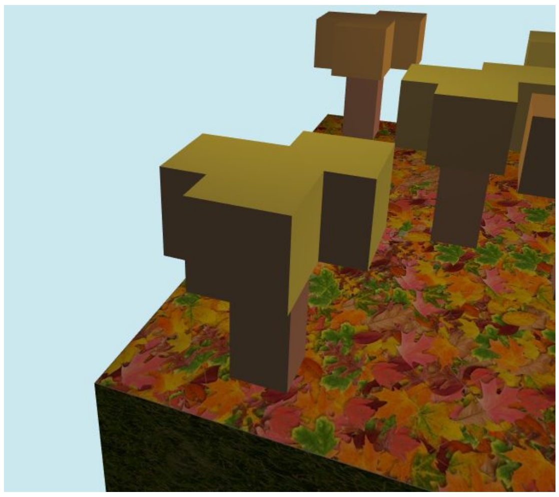


All the shapes in the sketch (excepting the Button), are childs of the Objects Rotation, that allows them to rotate at the same time, and in their own Y axis. All the interactions and callbacks are illustrated in the following figure:


Custom shape (Moon)
Root
Button Rotation
Button Translation
Sphere
Sun Translation
Moon Translation
Moon Rotation
Custom shape x 9 (cylinder)
Global Rotation
Sun Rotation
Moon Translation
Global Translation
Sun Translation
Rotate Moon callback
Click callback
Objects Rotation
Rotate callback
Rotate Sun callback
Tree Class
(7 trees per season = 28)
Trunk 1 Rotation
Trunk 2 Rotation
Trunk 3 Rotation
Trunk 1 Translation
Trunk 2 Translation
Trunk 3 Translation
Box
Box
Box
Leaf 1 Rotation
Leaf 2 Rotation
Leaf 3 Rotation
Leaf 1 Translation
Leaf 2 Translation
Leaf 3 Translation
Box
Box
Box
World Class
World Rotation
World Translation
Box x 4

The key item of the scene is the Tree object. It is composed by three blocks to build the trunk and three more blocks to simulate the leafs. It is encapsulated in the Tree class, that allow us to modify the position, rotation, color, and size in a very efficient way:

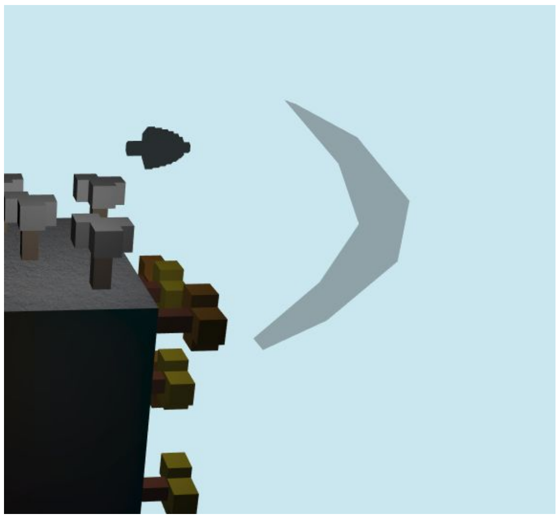
```
function Tree(localTransform,
 x, y, z,
 rx, ry, rz, angle,
 trunkColorR, trunkColorG,  trunkColorB,
 leafColorR, leafColorG, leafColorB)
{
 this.localTransform = localTransform;
 this.x = x;
 this.y = y;
 this.z = z;
 this.rx = rx;
 this.ry = ry;
 this.rz = rz;
 this.angle = angle;
 this.trunkColorR= trunkColorR;
 this.trunkColorG= trunkColorG;
 this.trunkColorB= trunkColorB;
 this.leafColorR= leafColorR;
 this.leafColorG= leafColorG;
 this.leafColorB= leafColorB;
 this.trunkSize = 2;
 this.leafSize = 3;
}
```



<u>Custom Geometry</u>

The Moon class is defined using custom geometry and parametrized by two attributes: the width and the height.

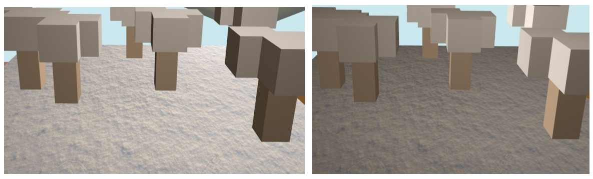
It has been defined using indexed geometry using 18 points and 34 triangles.



Lighting

The lighting of the scene is composed by three lights:

- An ambient light
- A spot light facing to the top face of the scene
- A spinning point light that “follows” the sun (the LightSource is a child of the Sun Translation). This moving light creates an effect of day/night when the Sun is hidden behind the Earth:



Texture Mapping

Four different texture mappings can be seen, one texture for each season (stored in the “res” folder). They are defined in the World object, where we place them in the ground.

Animation

Animation has been the more complicated part of this project. There are several rotation and translation matrixs on top of each object.

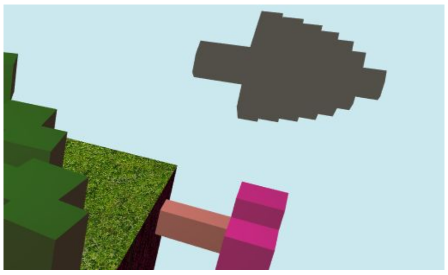
- **World and Trees rotation**
  All the trees and the Earth are childs of the Objects Rotation Transform, that rotate them together on the same axis.
- **Sun and Moon rotation**
  It uses the parametric formula to define a circumference:
  *x=radius*cos(t), y=radius*sin(t)*

```
osg.Matrix.makeTranslate(35*Math.sin(currentTime/4),
20,
35*Math.cos(currentTime/4),
matrix);
```

Interaction

The interaction is handled by the button, that is composed by 9 cylinders with different radius. It allows the user to spin the Earth to the next season.

```
var ClickCallback = function() {};
ClickCallback.prototype = {
    update: function(node) {
        node.onpick = function(){
            state += 1;
        }
    }
}
```



Note that the callback is launched by the center cylinder of the button. If the user click on the external part of the button, the callback won't be triggered.

Development environment

In order to create the custom scene I used the following tools:

- **Interface development environment (IDE):**
  I used the Visual Studio Code IDE with the JavaScript extension in order to highlight the code and have additional suggestions.
- **Docker:**
  The JavaScript code has been developed inside an isolated Docker container using the NGINX web server image. The container expose the port 80 to my local machine and also a shared volume containing the source code.
- **Control Version System (CVS):**
  To track all the changes, I used the GitHub hosted control version. It also allowed me to work in different computers with the same code.
  The code is available on: https://github.com/enric1994/EE497-1

Deployment

The OSGJS project is deployed on Google Cloud Platform using Kubernetes. It is accessible at http://seasons.enricmor.eu

Additional information

The aim of this project changed during the second week of development. The initial objective was a 3D curriculum using 3D models created with Blender and translated to OSGJS model using the *trigrou/osg* Docker image. However, the complexity of the initial project was too much for the project's time.

Enric Moreu
3D resume

Conclusion

In conclusion, a 3D scene representing the 4 seasons on a set of trees can be designed. The OSGJS framework is very flexible and supported the implementation of lighting, grouping, texturing and animation.
The usage of tools like Docker and Git improved the development and deployment.