Steps to develop the AR Project

Development Process

Creation of 3d models

The 3D models created for this project aim to visualise the historical evolution of the Two Towers of Bologna through augmented reality. Existing models of the towers uploaded on on UltiMaker Thingiverse have been enriched and modified in Blender to create seven distinct 3D models, each representing the towers during different historical periods: the original construction in the 1100s, the addition of the *Corridore* and the resizing of both towers in the 1300s and 1400s, and the more modern additions of the late 1800s and early 1900s.

Extensive research was conducted using historical documents, photographs, and period drawings to ensure all models accurately reflect their respective era.

AR Marker Tracking

The goal is to use AR.js, an open-source collaborative library for Augmented Reality on the Web, to implement marker-based AR tracking. AR.js is known for its efficiency and compatibility with web-based applications which allows the creation of easily accessible AR experiences without requiring users to download an app.

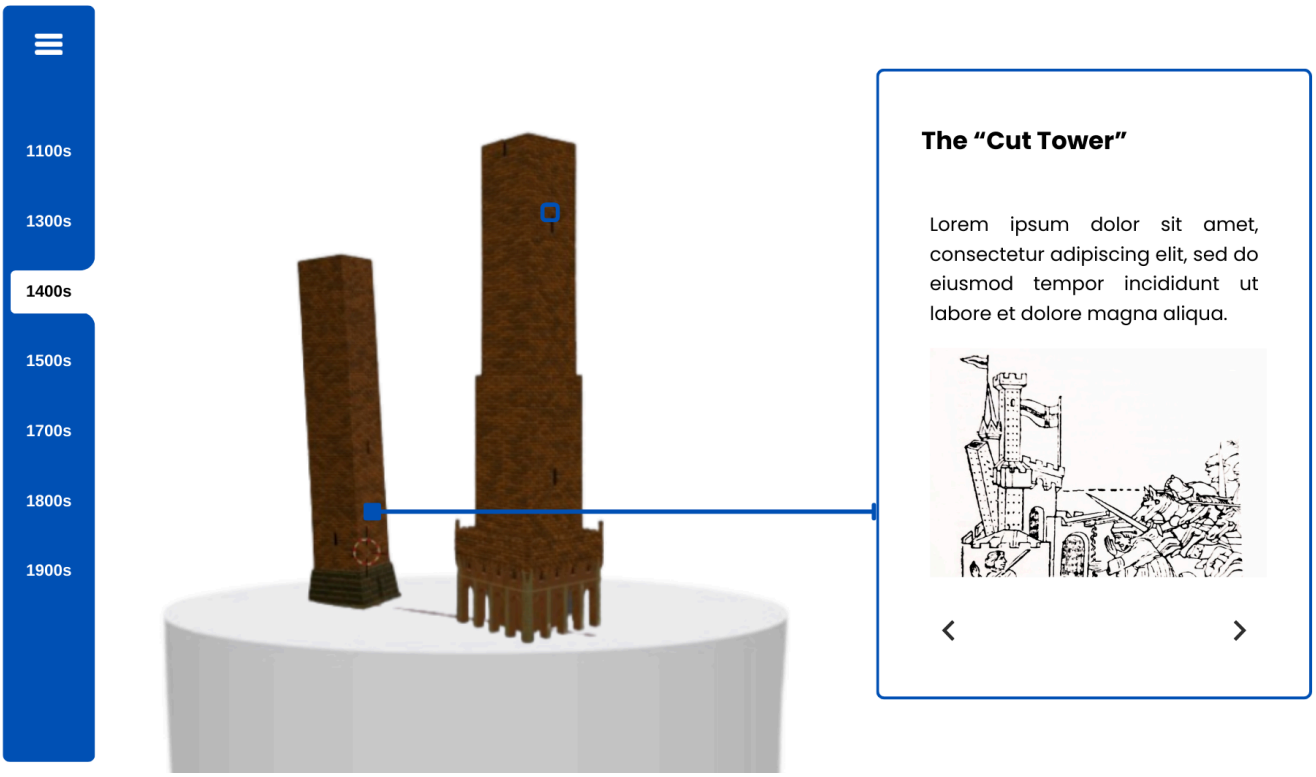
The markers that should be used are simple black and white patterns that are easily recognizable by the AR.js framework. To set up marker tracking, AR.js is integrated into the web application and *A-scene* and *a-marker* elements are added to the HTML structure to define the AR environment and markers respectively.

Model Superimposition

These models are exported in formats compatible with AR.js, *.glb*. Each 3D model is linked to a specific AR marker using the *a-marker* element in AR.js. When the marker is detected by the camera, the corresponding 3D model is superimposed on top of the marker in real-time. The position, scale, and orientation of the 3D models were adjusted to ensure they align correctly with the physical markers.

User Interaction

UI buttons are added to the application to allow users to switch between different eras and different evolutions during the same era. When a button is clicked, either on the model or in the timeline on the left, the application updates the 3D model displayed on the marker to show the corresponding historical period. Interactive hotspots are added to the models to provide historical information and explain the changes. When users tap on these hotspots, text or multimedia content appears, offering more details about specific features of the towers.



The prototype of the interface

Code snippets

```
<!DOCTYPE html>
<html>
<head>
    <title>TimeTowers</title>
    <script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>
    <script
src="https://cdn.rawgit.com/jeromeetienne/AR.js/1.7.2/aframe/build/aframe-ar.js">
</script>
</head>
<body style="margin: 0; overflow: hidden;">
    <a-scene embedded arjs>
        <a-assets>
            <a-asset-item id="model1" src="path/to/model1.gltf"></a-asset-item>
            <a-asset-item id="model2" src="path/to/model2.gltf"></a-asset-item>
            <!-- Add more models as needed -->
        </a-assets>
        <a-marker preset="hiro" id="marker">
```

```
            <a-entity id="model" position="0 0 0" scale="0.1 0.1 0.1"
gltf-model="#model1" visible="true">
                <a-entity id="hotspot1" position="0 1 0" geometry="primitive:
sphere; radius: 0.1" material="color: red" class="clickable"></a-entity>
                <a-text id="info1" value="Historical Information 1" position="0
1.2 0" visible="false"></a-text>
                <a-entity id="hotspot2" position="0 1 0" geometry="primitive:
sphere; radius: 0.1" material="color: blue" class="clickable"
visible="false"></a-entity>
                <a-text id="info2" value="Historical Information 2" position="0
1.2 0" visible="false"></a-text>
                <!-- Add more hotspots and texts as needed -->
            </a-entity>
        </a-marker>
        <a-entity camera></a-entity>
    </a-scene>

    <div style="position: fixed; top: 10px; left: 10px; z-index: 1;">
        <button onclick="showModel('model1')">Era 1</button>
        <button onclick="showModel('model2')">Era 2</button>
        <!-- Add more buttons for each era -->
    </div>

    <div id="timeline" style="position: fixed; top: 50px; left: 10px; z-index:
1;">
        <button onclick="showModel('model1')">Timeline Era 1</button>
        <button onclick="showModel('model2')">Timeline Era 2</button>
        <!-- Add more timeline buttons as needed -->
    </div>

    <script>
        function showModel(modelId) {
            const model = document.getElementById('model');
            model.setAttribute('gltf-model', `#${modelId}`);
            updateHotspots(modelId);
        }

        function updateHotspots(modelId) {
            // Hide all hotspots initially
            const hotspots = document.querySelectorAll('[id^=hotspot]');
            hotspots.forEach(hotspot => hotspot.setAttribute('visible',
```

```
'false'));

            const infos = document.querySelectorAll('[id^=info]');
            infos.forEach(info => info.setAttribute('visible', 'false'));

            // Show hotspots specific to the modelId
            if (modelId === 'model1') {
                document.getElementById('hotspot1').setAttribute('visible',
'true');
            } else if (modelId === 'model2') {
                document.getElementById('hotspot2').setAttribute('visible',
'true');
            }
            // Add more conditions for additional models
        }

        AFRAME.registerComponent('clickable', {
            init: function () {
                var el = this.el;
                el.addEventListener('click', function () {
                    var infoId = el.getAttribute('id').replace('hotspot',
'info');
                    var info = document.getElementById(infoId);
                    var isVisible = info.getAttribute('visible');
                    info.setAttribute('visible', !isVisible);
                });
            }
        });
    </script>
</body>
</html>
```

Testing and Validation

After having been developed the AR application will be tested extensively to ensure that both functionality and user experience are optimal.

The following must be tested:

- Markers: it must be ensured that all the markers are recognized correctly by the AR.js library
- Model switching: each button must correspond to the correct 3d model, the transition between models and any associated content, like hotspots and information bubbles, must be smooth

- Hotspots: the correct information is associated with the hotspot and that it appears and disappears as expected
- Overall performance: the performance of the application must be monitored to ensure smooth rendering and interaction. Testing it on different devices and screen sizes is required.

Usability tests with potential users (students, locals) will be conducted to gather feedback on navigation, clarity of information, and overall user experience. Feedback in the form of short questionnaires will be incorporated in the testing procedures.

Key Features for the AR Application (Based on the Personas):

User-Friendly Interface: Ensure the app is intuitive and easy to navigate, with clear instructions for tourists who may be using it for the first time.

Multilingual Support: Offer translations in multiple languages to cater to international tourists.

Photo Sharing: Enable users to take screenshots or photos of the AR experience to share on social media and travel blogs.