COLEGIUL NAŢIONAL DE INFORMATICĂ

„GRIGORE MOISIL” BRAŞOV

LUCRARE PENTRU ATESTAREA COMPETENŢELOR PROFESIONALE

GLTF Viewer

Elev

Carlo Cotelea

Clasa

XII-E

Profesori îndrumător

Manuela Șerban

Laura Nițulescu

Mai 2023

Cuprins

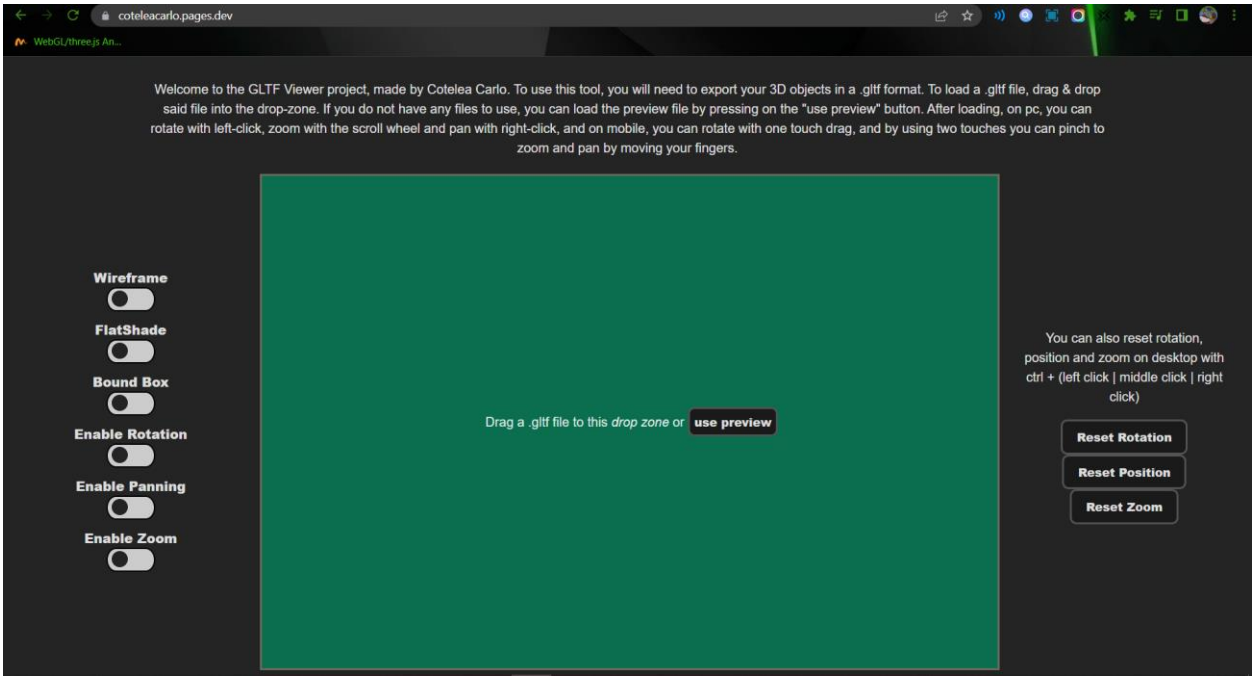
MOTIVAREA ALEGERII TEMEI

Acest proiect a fost conceput în urma unui internship la o companie specializată pe web design. S-a ivit un client care a cerut o componenta web care afișa un tablou interactiv pentru utilizatori, iar cum eu eram singurul developer care știa sa folosească 3D în browser, mi s-a oferit șansa să scriu cod pentru producție. Rezultatul versiunii inițiale poate fi accesat la website-ul ParrotPrint(https://parrotprint.com/canvasbuilder). După terminarea proiectului inițial au mai apărut două companii care au cerut această componentă, fapt care a dus la crearea componentei curente care deține o multitudine de opțiuni.

Dezvoltarea acestei componente a însemnat și un drum de învățare pentru mine, un drum în care am făcut multe greșeli, le-am corectat, și am invățat practici care au să ma ajute in viitor.

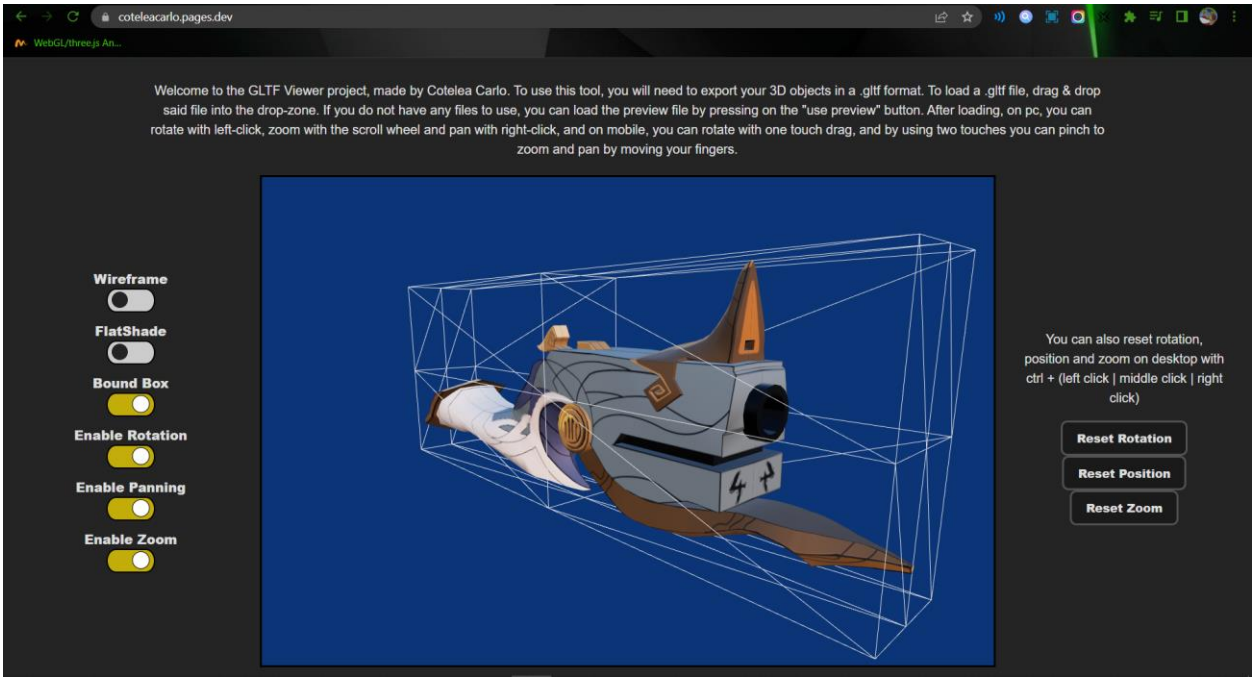
Prezentarea aplicației

Aplicația permite utilizatorilor să încarce fișiere de tip .gltf/.glb în browser pentru vizualizare prin interfața web. Pentru a încărca fișierul dorit, utilizatorul trebuie să folosească funcționalitatea drag & drop în zona corespunzătoare, după care obiectul va fi încărcat iar controalele devin disponibile. Aplicația dispune de un număr de butoane care prezintă opțiuni ale componentei cu care utilizatorul poate interacționa.



După încărcarea fișierului, utilizatorii pot controla camera în următoarele moduri:

- PC:
  - Left-Click: Rotația camerei
  - Scroll-Wheel: Zoom-ul camerei
  - Right-Click: Poziția camerei
- Mobil:
  - Un deget: Rotația camerei
  - Două degete pinch: Zoom-ul camerei
  - Două degete drag: Poziția camerei



Așa arată interfața după încărcarea fișierului.

Detalii de implementare

Aplicația este construită pentru browser cu ajutorul framework-ului Vite cu Typescript. IDE-ul folosit pentru proiect este Visual Studio Code, iar pentru încărcarea, afișarea și interactibilitatea componentei a fost folosit pachetul three.js.

Aplicația este o prezentare a componentei GLTF Viewer, pagina web fiind una simpla, cu câteva butoane pentru prezentarea celor mai importante funcționalități.

Componenta GLTF Viewer este partea complexă a aplicației. Aceasta este împărțită în 3 module.

1. SceneLoader.ts
2. GLTFObjectLoader.ts
3. CameraControlsLoader.ts

Aceste module sunt inițiate in clasa părinte care este apelată pentru încărcarea fișierelor.

SceneLoader.ts

Această componentă crează elemente HTML din cod, care apoi sunt puse in pagină, pentru afișarea obiectului. De asemenea, este creată in memorie scena, renderer-ul și camera.

```
private LoadSceneAndDOM(): void {
        /// DOM
        this.canvas = document.createElement("canvas");
        this.canvas.classList.add("three-canvas");
        if(!CheckForStyle()) {
            this.canvasCSS = document.createElement("style");
            this.canvasCSS.innerHTML = cssStyle;
            document.head.appendChild(this.canvasCSS);
        }
        this.div.appendChild(this.canvas);
        this.div.addEventListener("contextmenu", (event) => {event.preventDefault()});

        /// THREE
        this.scene = new THREE.Scene();
        this.camera = new THREE.PerspectiveCamera();
        this.ambientLight = new THREE.AmbientLight(this.settings.ambientColor,
this.settings.ambientIntensity);
        this.scene.add(this.ambientLight);

        this.renderer = new THREE.WebGLRenderer({canvas: this.canvas, antialias: true,
alpha: true});
        this.renderer.outputEncoding = THREE.sRGBEncoding;
        this.renderer.toneMapping = THREE.ACESFilmicToneMapping;
        this.renderer.setPixelRatio((window.devicePixelRatio > 2) ? 2 :
window.devicePixelRatio);
```

```
        this.cubeTextureLoader = new RGBELoader();
        this.cubeTextureLoader.load("/HDRI/venice_sunset_1k.hdr", (texture) => {
            this.envMap = texture;
            this.envMap.mapping = THREE.EquirectangularReflectionMapping;
            this.scene.environment = this.envMap;
            if(this.onLoad)
                this.onLoad();
            this.ResizeCheckTick();
        })
    }
```

GLTFObjectLoader.ts

Această componentă încarcă fișierul de tip .gltf/.glb.

```
private LoadObjectFromPath(objectPath: string | ArrayBuffer): void {
        var onLoad = (loaded: GLTF) => {
            while(loaded.scene.children.length)
                this.object.add(loaded.scene.children[0]);
            var traverseIndex = 0;
            this.object.traverse((child) => {
                if(child.type !== "Mesh") return;
                const standardMat = ((child as THREE.Mesh).material as
THREE.MeshStandardMaterial);
                const basicMat = new THREE.MeshBasicMaterial();
                THREE.MeshBasicMaterial.prototype.copy.call(basicMat, standardMat);
                this.materials[0][traverseIndex] = standardMat;
this.materials[1][traverseIndex] = basicMat;
                this.materials[0][traverseIndex].wireframe =
this.materials[1][traverseIndex].wireframe = this.settings.wireframe!;
                (child as THREE.Mesh).material = this.settings.flatShade ?
this.materials[1][traverseIndex] : this.materials[0][traverseIndex];
                traverseIndex++;
            })
            this.CreateBoundingBox();
        }
        if(typeof objectPath === "string")
            gltfLoader.load(objectPath, onLoad);
        else
            gltfLoader.parse(objectPath, "", onLoad);
    }
```

Componenta poate încărca obiectul în două moduri. Printr-un string care conține path-ul către fișier sau printr-un ArrayBuffer care conține fișierul în sine.

```
function CenterObject(object: THREE.Group, cbFunc?: Function): void {
    const objectBox = new THREE.Box3().setFromObject(object);
    const objectOffset = {
        x: (objectBox.max.x + objectBox.min.x) / 2,
        y: (objectBox.max.y + objectBox.min.y) / 2,
        z: (objectBox.max.z + objectBox.min.z) / 2,
    }
    object.children.forEach((mesh) => {
        mesh.position.x -= objectOffset.x;
        mesh.position.y -= objectOffset.y;
        mesh.position.z -= objectOffset.z;
    })
    if(cbFunc) cbFunc();
}
```

Odată încărcat, obiectul trebuie centrat în pagină. Această funcție primește ca parametru variabile de tip grupă three.js care conține obiectul și un parametru opțional prin care se poate chema o funcție la finalizarea centrării.

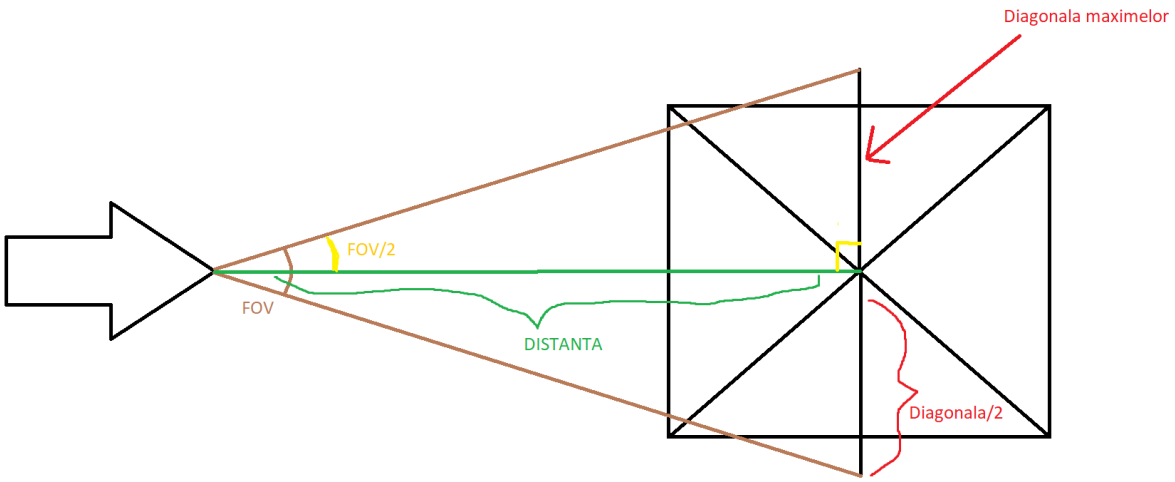
CameraControlsLoader.ts

Această componentă este cea mai complexă, fiind necesară calcularea distanței de la cameră la obiect, cât și implementarea interacțiunilor cu utilizatorul.

```
private CameraCalculateDistance() {
        const objectBox = new THREE.Box3().setFromObject(this.object);
        const size = {
            x: Math.abs(objectBox.max.x - objectBox.min.x),
            y: Math.abs(objectBox.max.y - objectBox.min.y),
            z: Math.abs(objectBox.max.z - objectBox.min.z),
        }
        /// Calculate the max and min length of the object
        this.cameraSettings.maxObjectLength = Math.max(size.x, size.y, size.z);
        this.cameraSettings.minObjectLength = Math.min(size.x, size.y, size.z);
        /// Calculate average object length
        this.cameraSettings.averageObjectLength = this.Average([size.x, size.y,
size.z]);
        /// Calculate the recommended camera distance
        const x = Math.sqrt(Math.pow(Math.sqrt(size.x*size.x + size.y*size.y), 2) +
size.z*size.z)/2;
        const delta = Math.tan((25/180)*Math.PI);
        this.cameraSettings.recommCameraDist = (x / delta)*1.125;
        /// Set the camera near and far based of the max and min object length
        this.camera.near = this.cameraSettings.minObjectLength/100;
        this.camera.far = this.cameraSettings.maxObjectLength*100;
        /// If not already set in the settings, calculate max zoom and current zoom
```

```
        if(this.settings.maxZoom?.in !== undefined && typeof this.settings.maxZoom.in
=== "string")
            this.cameraSettings.maxZoom.in =
this.cameraSettings.recommCameraDist*this.CheckZoomString(this.settings.maxZoom.in)/10
0;
        if(this.settings.maxZoom?.out !== undefined && typeof
this.settings.maxZoom.out === "string")
            this.cameraSettings.maxZoom.out =
this.cameraSettings.recommCameraDist*this.CheckZoomString(this.settings.maxZoom.out)/1
00;
        else
            this.cameraSettings.maxZoom.out = this.cameraSettings.recommCameraDist*3;
        /// ----------
        if(this.settings.currentZoom === undefined)
            this.cameraSettings.currentZoom = this.cameraSettings.recommCameraDist;
        if(this.settings.currentZoom && typeof this.settings.currentZoom === "string")
            this.cameraSettings.currentZoom =
this.cameraSettings.recommCameraDist*this.CheckZoomString(this.settings.currentZoom)/1
00;

        if(this.onLoad)
            this.onLoad();
    }
```



Avem la dispoziție valoarea in grade a unghiului FOV(Field Of View) si cunoaștem diagonala maximelor, iar Distanța împarte unghiul FOV in jumătate și este perpendiculară cu diagonala maximelor. Putem calcula Distanța folosind funcția de trigonometrie a tangentei.

```
private ControlsInit() {
        this.pivot = new THREE.Group();
        this.scene.add(this.pivot);
        this.pivot.rotation.order = "ZYX";
        this.pivot.add(this.camera);
        this.CameraCalculateDistance();
        if(this.isMobile) this.LoadTouchControls();
        else this.LoadMouseControls();
        this.Tick();
    }
```

Funcţia de iniţializare.

```
private LoadMouseControls() {
        var oldRot: Rotation = {horizontal: 0, vertical: 0}, deltaRot: Rotation =
{horizontal: 0, vertical: 0};
        var oldMouse: MousePos = {x: 0, y: 0}, deltaMouse: MousePos = {x: 0, y: 0};
        var isDown: boolean = false, isInside: boolean = false, normalizeSize: number;
        var horizontalCorrection: 1 | -1 = 1, bodyOverscrollSave: string =
document.body.style.overflow;
        this.canvas.addEventListener("mouseenter", () => {
            isInside = true;
            if(this.cameraSettings.focusDisplay) document.body.style.overflow =
"hidden";
        })
        this.canvas.addEventListener("mousedown", (event) => {
            event.preventDefault();
            /// Event: 0 - left click | 1 - middle click | 2 - right click
            isDown = true;
            normalizeSize = Math.min(this.canvas.clientWidth,
this.canvas.clientHeight);
            oldMouse.x = event.clientX;
            oldMouse.y = event.clientY;
            /// Based on mouse button
            if(event.ctrlKey) {
                if(event.button === 0)
                    this.ResetRotation();
                else if(event.button === 1)
                    this.ResetZoom();
                else if(event.button === 2)
                    this.ResetPosition();
            }
            if(event.button === 0) { /// Deprecated but WHY IS IT DEPRECATED LITERALLY
MOST USEFUL
                oldRot.horizontal = this.cameraSettings.currentRotation.horizontal;
                oldRot.vertical = this.cameraSettings.currentRotation.vertical;
                var verticalCheck = Math.abs(oldRot.vertical)%2;
```

```
                horizontalCorrection = (verticalCheck > 0.5 && verticalCheck < 1.5) ?
-1 : 1;
            }
            else if(event.button === 2) {
                document.body.style.cursor = "grab";
            }
        })
        window.addEventListener("mousemove", (event) => {
            if(!isDown) return;
            deltaMouse.x = event.clientX - oldMouse.x;
            deltaMouse.y = event.clientY - oldMouse.y;
            /// Based on mouse button
            if(event.which === 1) {
                if(!this.cameraSettings.canRotate) return;
                deltaRot.horizontal = deltaMouse.x/normalizeSize;
                deltaRot.vertical = deltaMouse.y/normalizeSize;
                this.cameraSettings.currentRotation.vertical = oldRot.vertical -
deltaRot.vertical;
                this.cameraSettings.currentRotation.horizontal = oldRot.horizontal -
deltaRot.horizontal*horizontalCorrection;
                this.ClampRotation();
            }
            else if(event.which === 3) { /// event.button is 0 for both left and right
click in this event for some reason
                if(!this.cameraSettings.canPan) return;
                this.pivot.translateX(-
(deltaMouse.x/normalizeSize)*this.cameraSettings.averageObjectLength*2*(this.camera.po
sition.z/this.cameraSettings.recommCameraDist)*this.CalculatePanFalloff());
                this.pivot.translateY((deltaMouse.y/normalizeSize)*this.cameraSettings
.averageObjectLength*2*(this.camera.position.z/this.cameraSettings.recommCameraDist)*t
his.CalculatePanFalloff());
                oldMouse.x = event.clientX;
                oldMouse.y = event.clientY;
            }
        })
        this.canvas.addEventListener("mouseleave", () => {
            isInside = false;
            if(isDown) return;
            document.body.style.overflow = bodyOverscrollSave;
        })
        window.addEventListener("mouseup", () => {
            isDown = false;
            if(!isInside) document.body.style.overflow = bodyOverscrollSave;
            document.body.style.cursor = "auto";
        })
        const ZoomFalloff = (): number => {
            if(!this.cameraSettings.enableZoomFalloff) return 1;
```

```
            return ((this.cameraSettings.currentZoom-
this.cameraSettings.maxZoom.in)/this.cameraSettings.recommCameraDist)
        }
        this.canvas.addEventListener("wheel", (event) => {
            if(!this.cameraSettings.canZoom) return;
            this.cameraSettings.currentZoom +=
(this.cameraSettings.averageObjectLength*10/100) * ZoomFalloff() * (event.deltaY > 0 ?
1 : -1);
            this.ClampZoom();
        })
    }
```

Controalele sunt adăugate prin event-uri de tip MouseEvent și verificarea tipului de click apăsat. Pentru zoom, se folosește event-ul de la scroll-wheel.

Posibilități de dezvoltare

Componenta GLTF Viewer permite accesul din cod la elementele HTML create și la variabilele din pachetul three.js, oferind posibilitatea de expansiune a componentei, crearea unei componente noi, sau interacționarea cu alte pachete si elemente.

Un exemplu de expansiune al componentei se poate observa la link-ul acesta(https://ploom3d.pages.dev/), unde este prezentată o nouă componentă pentru un website temporar care nu mai există. Prin crearea unei noi clase numite HotspotLoader care primește ca si parametru componenta GLTF Viewer, în scena componentei au fost adăugate Sprite-ul interactive care conțin informații legate despre produs (informațiile de pe link sunt placeholder, neavând acces la textul final).

Resurse hardware și software necesare

Resurse hardware:

- Un calculator/laptop/telefon
- Pentru calculator/laptop, Nvidia/AMD/Apple/Intel GPU (trebuie să poată fi accesat de browser)
- Pentru calculator/laptop, minimum un dual-core

Resurse software:

- Sistem de operare Windows/Linux/MacOs/Android/IOS
- Orice browser modern post-2011 (> Internet Explorer 9) (Ex: Chrome, Firefox, OperaGX, Safari)

Bibliografie

Tutorial HTML/CSS (https://www.w3schools.com/html/default.asp)

Tutorial Javascript (https://www.w3schools.com/js/default.asp)

Marjin Haverbeke – Eloquent Javascript (https://eloquentjavascript.net/)

Documentație Javascript (https://developer.mozilla.org/en-US/docs/Web/JavaScript)

Tutorial three.js (https://threejs-journey.com/)

Documentație three.js (https://threejs.org/docs/)