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

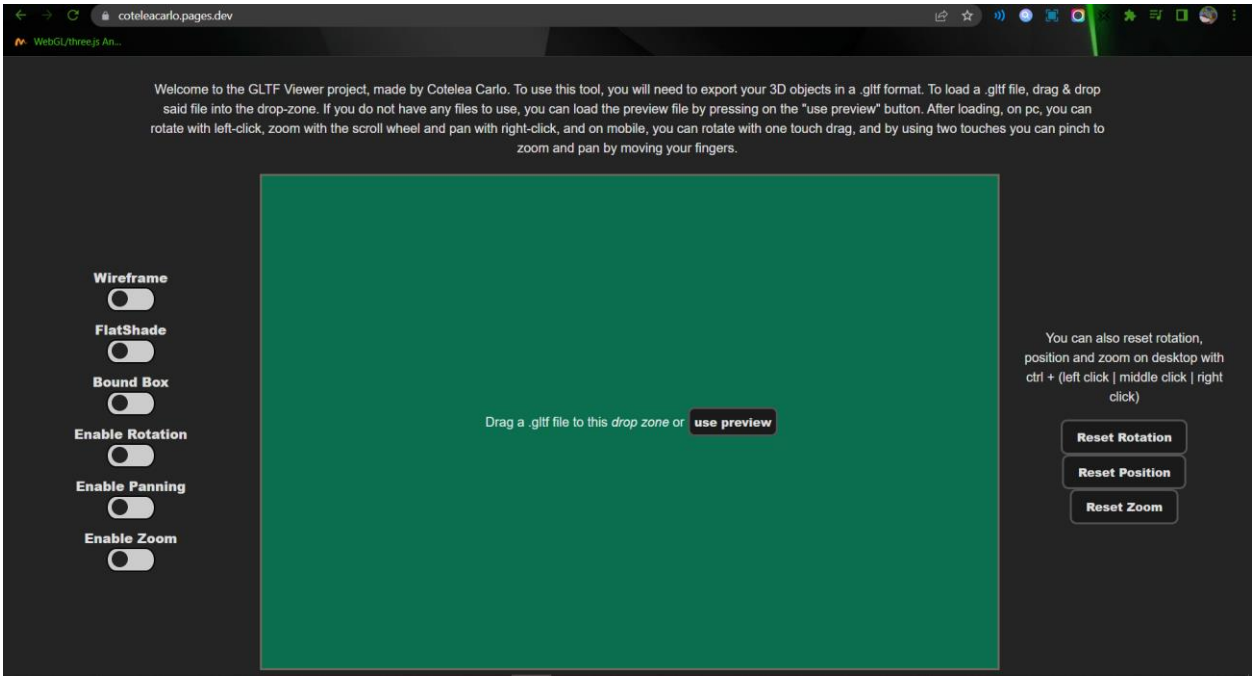
MOTIVAREA ALEGERII TEMEI

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.

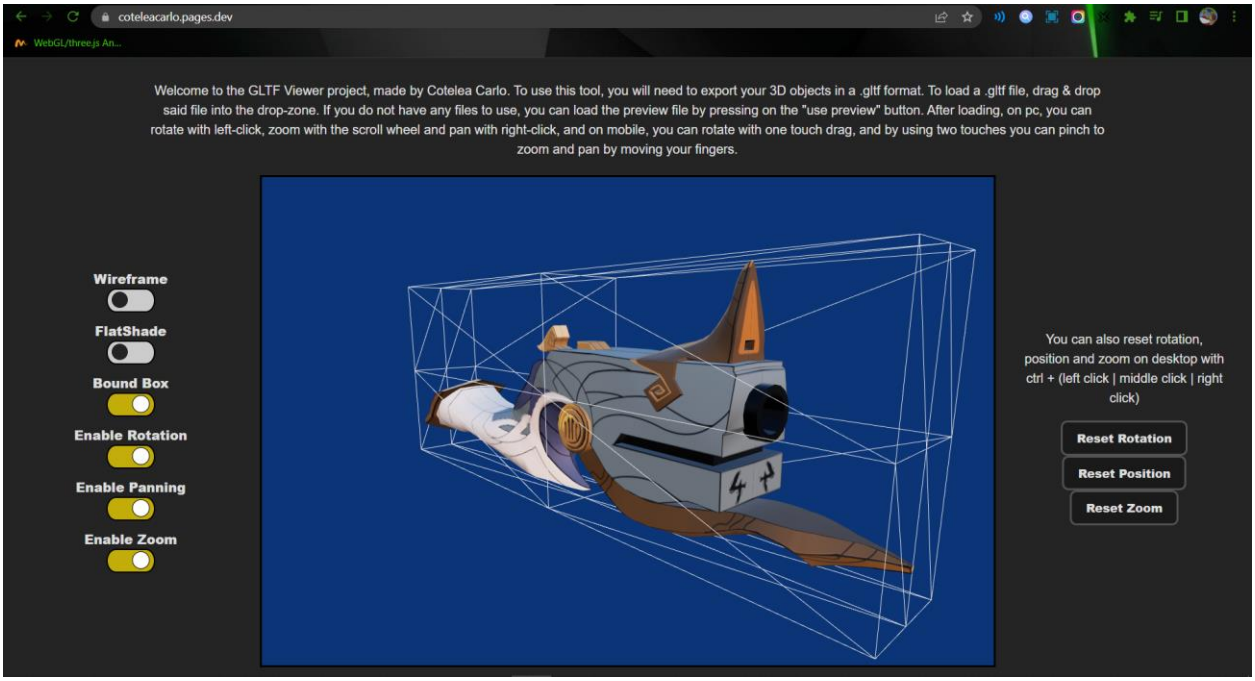
Prezentarea aplicației

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.

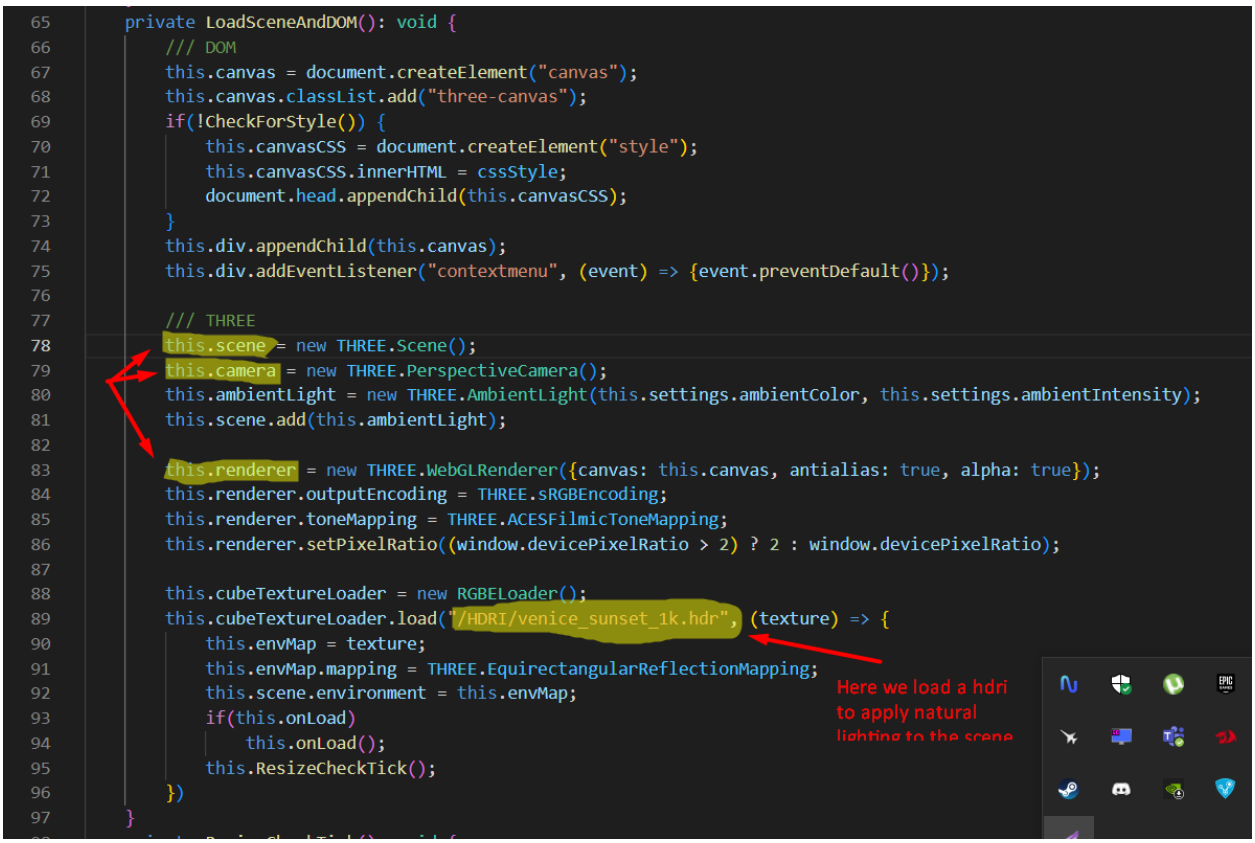
Componenta GLTF Viewer este partea complexă a aplicației. Aceasta este împărțită în 3 module.

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.



GLTFObjectLoader.ts

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

```
49  private LoadObjectFromPath(objectPath: string | ArrayBuffer): void {
50      var onLoad = (loaded: GLTF) => {
51          while(loaded.scene.children.length)
52              this.object.add(loaded.scene.children[0]);
53          var traverseIndex = 0;
54          this.object.traverse((child) => {
55              if(child.type !== "Mesh") return;
56              const standardMat = ((child as THREE.Mesh).material as THREE.MeshStandardMaterial);
57              const basicMat = new THREE.MeshBasicMaterial();
58              THREE.MeshBasicMaterial.prototype.copy.call(basicMat, standardMat);
59              this.materials[0][traverseIndex] = standardMat; this.materials[1][traverseIndex] = basicMat;
60              this.materials[0][traverseIndex].wireframe = this.materials[1][traverseIndex].wireframe = this.settings.wireframe
61              (child as THREE.Mesh).material = this.settings.flatShade ? this.materials[1][traverseIndex] : this.materials[0][t
62              traverseIndex++;
63          })
64          this.CreateBoundingBox();
65      }
66      if(typeof objectPath === "string")
67          gltfLoader.load(objectPath, onLoad);
68      else
69          gltfLoader.parse(objectPath, "", onLoad);
70  }
```

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.

```
7   function CenterObject(object: THREE.Group, cbFunc?: Function): void {
8       const objectBox = new THREE.Box3().setFromObject(object);
9       const objectOffset = {
10          x: (objectBox.max.x + objectBox.min.x) / 2,
11          y: (objectBox.max.y + objectBox.min.y) / 2,
12          z: (objectBox.max.z + objectBox.min.z) / 2,
13      }
14      object.children.forEach((mesh) => {
15          mesh.position.x -= objectOffset.x;
16          mesh.position.y -= objectOffset.y;
17          mesh.position.z -= objectOffset.z;
18      })
19      if(cbFunc) cbFunc();
20  }
21
22  interface ObjectLoaderParams {
23      objectPath: string | ArrayBuffer,
24      settings?: ObjectSettings
25  }
26
27  class ObjectLoader {
```

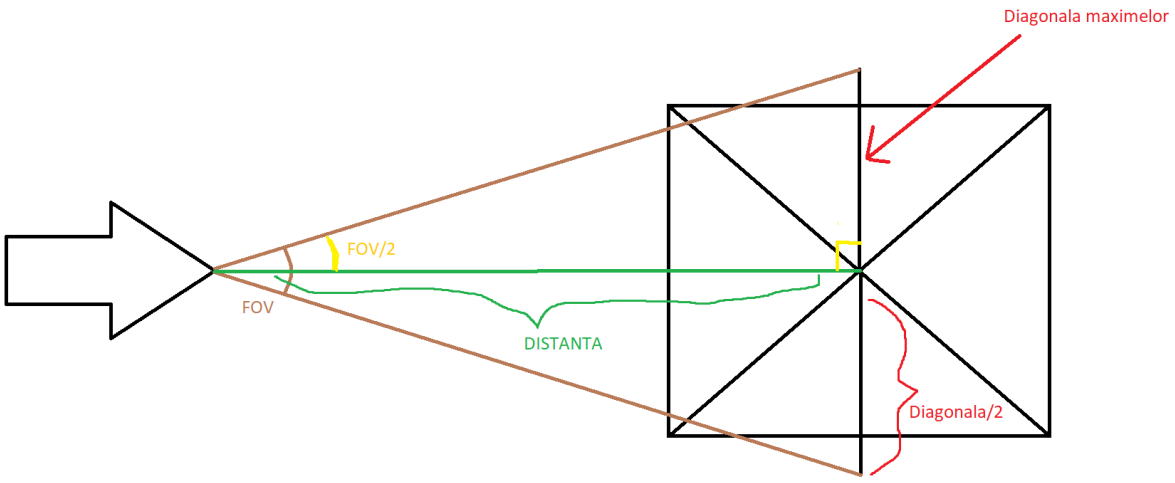
Odată încărcat, obiectul trebuie centrat în pagină. Această funcție primește ca parametru variabile de tip grupă three.js care conține obiectul și un parametru opțional prin care se poate chema o funcție la finalizarea centrării.

CameraControlsLoader.ts

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

```
133     private CameraCalculateDistance() {
134         const objectBox = new THREE.Box3().setFromObject(this.object);
135         const size = {
136             x: Math.abs(objectBox.max.x - objectBox.min.x),
137             y: Math.abs(objectBox.max.y - objectBox.min.y),
138             z: Math.abs(objectBox.max.z - objectBox.min.z),
139         }
140         /// Calculate the max and min length of the object
141         this.cameraSettings.maxObjectLength = Math.max(size.x, size.y, size.z);
142         this.cameraSettings.minObjectLength = Math.min(size.x, size.y, size.z);
143         /// Calculate average object length
144         this.cameraSettings.averageObjectLength = this.Average([size.x, size.y, size.z]);
145         /// Calculate the recommended camera distance
146         const x = Math.sqrt(Math.pow(Math.sqrt(size.x*size.x + size.y*size.y), 2) + size.z*size.z)/2;
147         const delta = Math.tan((25/180)*Math.PI);
148         this.cameraSettings.recommCameraDist = (x / delta)*1.125;
149         /// Set the camera near and far based of the max and min object length
150         this.camera.near = this.cameraSettings.minObjectLength/100;
151         this.camera.far = this.cameraSettings.maxObjectLength*100;
152         /// If not already set in the settings, calculate max zoom and current zoom
153         if(this.settings.maxZoom?.in !== undefined && typeof this.settings.maxZoom.in === "string")
154             this.cameraSettings.maxZoom.in = this.cameraSettings.recommCameraDist*this.CheckZoomString(this.settings.maxZoom.in)/100;
155         if(this.settings.maxZoom?.out !== undefined && typeof this.settings.maxZoom.out === "string")
156             this.cameraSettings.maxZoom.out = this.cameraSettings.recommCameraDist*this.CheckZoomString(this.settings.maxZoom.out)/100;
157         else
158             this.cameraSettings.maxZoom.out = this.cameraSettings.recommCameraDist*3;
159         /// ----------
160         if(this.settings.currentZoom === undefined)
161             this.cameraSettings.currentZoom = this.cameraSettings.recommCameraDist;
162         if(this.settings.currentZoom && typeof this.settings.currentZoom === "string")
163             this.cameraSettings.currentZoom = this.cameraSettings.recommCameraDist*this.CheckZoomString(this.settings.currentZoom)/100;
164
165         if(this.onLoad)
166             this.onLoad();
167     }
```

Prima instanță de trigonometrie in programare

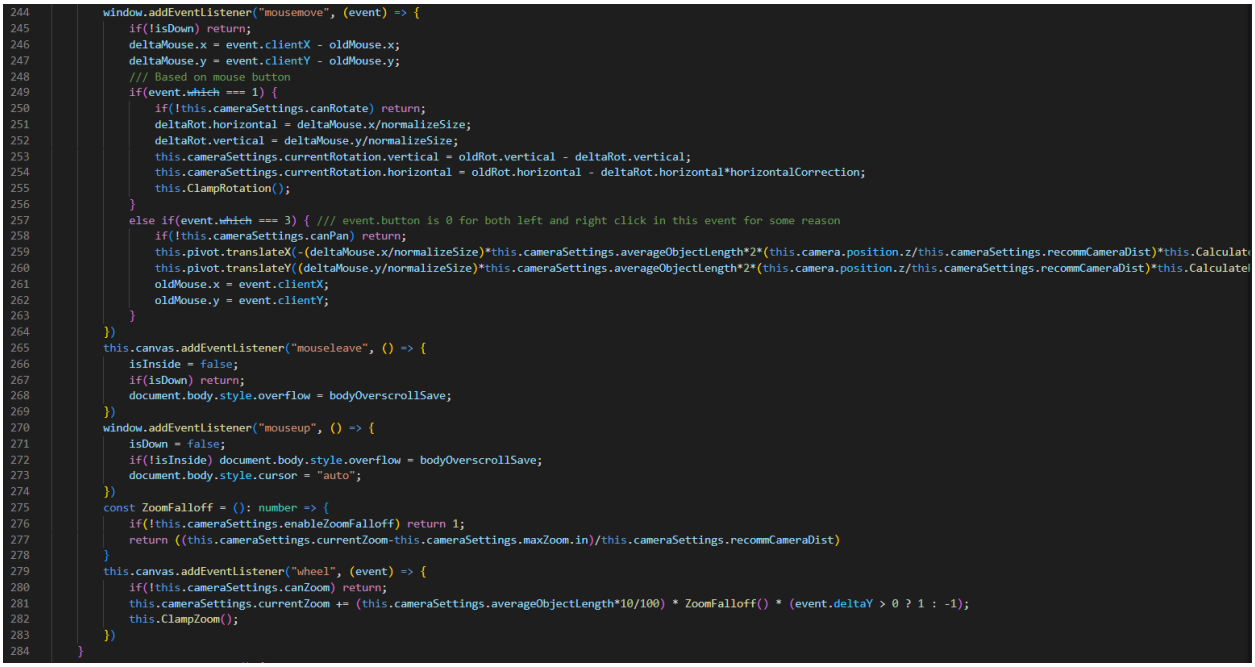


Avem la dispoziție valoarea in grade a unghiului FOV(Field Of View) si cunoaștem diagonala maximelor, iar Distanța împarte unghiul FOV in jumătate și este perpendiculară cu diagonala maximelor. Putem calcula Distanța folosind funcția de trigonometrie a tangentei.

```
181     private ControlsInit() {
182         this.pivot = new THREE.Group();
183         this.scene.add(this.pivot);
184         this.pivot.rotation.order = "ZYX";
185         this.pivot.add(this.camera);
186         this.CameraCalculateDistance();
187         if(this.isMobile) this.LoadTouchControls();
188         else this.LoadMouseControls();
189         this.Tick();
```

Funcţia de iniţializare.

```
209     private LoadMouseControls() {
210         var oldRot: Rotation = {horizontal: 0, vertical: 0}, deltaRot: Rotation = {horizontal: 0, vertical: 0};
211         var oldMouse: MousePos = {x: 0, y: 0}, deltaMouse: MousePos = {x: 0, y: 0};
212         var isDown: boolean = false, isInside: boolean = false, normalizeSize: number;
213         var horizontalCorrection: 1 | -1 = 1, bodyOverscrollSave: string = document.body.style.overflow;
214         this.canvas.addEventListener("mouseenter", () => {
215             isInside = true;
216             if(this.cameraSettings.focusDisplay) document.body.style.overflow = "hidden";
217         })
218         this.canvas.addEventListener("mousedown", (event) => {
219             event.preventDefault();
220             /// Event: 0 - left click | 1 - middle click | 2 - right click
221             isDown = true;
222             normalizeSize = Math.min(this.canvas.clientWidth, this.canvas.clientHeight);
223             oldMouse.x = event.clientX;
224             oldMouse.y = event.clientY;
225             /// Based on mouse button
226             if(event.ctrlKey) {
227                 if(event.button === 0)
228                     this.ResetRotation();
229                 else if(event.button === 1)
230                     this.ResetZoom();
231                 else if(event.button === 2)
232                     this.ResetPosition();
233             }
234             if(event.button === 0) { /// Deprecated but WHY IS IT DEPRECATED LITERALLY MOST USEFUL
235                 oldRot.horizontal = this.cameraSettings.currentRotation.horizontal;
236                 oldRot.vertical = this.cameraSettings.currentRotation.vertical;
237                 var verticalCheck = Math.abs(oldRot.vertical)%2;
238                 horizontalCorrection = (verticalCheck > 0.5 && verticalCheck < 1.5) ? -1 : 1;
239             }
240             else if(event.button === 2) {
241                 document.body.style.cursor = "grab";
242             }
243         })
244         window.addEventListener("mousemove", (event) => {
245             if(!isDown) return;
246             deltaMouse.x = event.clientX - oldMouse.x;
247             deltaMouse.y = event.clientY - oldMouse.y;
248             /// Based on mouse button
249             if(event.~~which~~ === 1) {
250                 if(!this.cameraSettings.canRotate) return;
```



```
        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.position.z/this.cameraSettings.recommCameraDist)*this.Calculat
                this.pivot.translateY((deltaMouse.y/normalizeSize)*this.cameraSettings.averageObjectLength*2*(this.camera.position.z/this.cameraSettings.recommCameraDist)*this.Calculate
                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 şi verificarea tipului de click apăsat. Pentru zoom, se foloseşte event-ul pentru 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)

Carte 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/)