SAE501 - Salle de classe en 3D

Alexandre Hamon - Théo Chaput - Yanis Buhot - Marin Vandelet
TD1

OBJET

- La salle
- Les portes
- Les fenêtres
- Rideaux
- Radiateur
- Lumières de la salle (néon + lumières du bureau)
- Chaise
- Chaise du bureau
- Table
- Bureau
- Tableau + Effaceur
- Horloge
- Ecran (PC) + cablage
- Clavier
- Souris
- Prises + Interrupteurs
- Barre blanche de prises
- Enceinte
- Affiche/Poster
- Vidéo-projecteur + câblage
- Poubelle
- Détails de la salle (panneaux à trous, barre en métal, rebord proche de la fenêtre, poutre et aération)

Modélisation faites à 100% sur blender avec utilisation de 2 addons :

- Looptools
- Boltfactory

Textures et matériaux

Pour les textures, quatre textures PBR (base color, roughness et normal) ont été utilisées, importées depuis le site Ambient CG en 4K, ainsi qu'une texture procédurale :

- Béton : pour les murs
- Revêtement au sol : pour le sol
- Bois : pour les chaises
- Tissu : pour les rideaux
- Horloge (photo) : pour l'horloge
- Verre (procédural) : pour les fenêtres et le verre

Pour le verre, j'ai utilisé le moteur de rendu EEVEE, avec le raytracing activé et les samples augmentés à 500.

Ensuite, seuls des matériaux classiques ont été appliqués aux autres éléments de la scène. Un dépliage UV a également été réalisé pour les textures.

Lumières

Pour la version web, les lumières n'ont pas été utilisées. Elles ont servi uniquement pour la réalisation de renders photoréalistes de la scène :

- Area lights pour l'éclairage intérieur de la salle
- Sun pour simuler la lumière globale du soleil

Ajustements pour le web

Hébergement du modèle 3D en ligne : sur Cloudflare R2 qui a une limite de 300 Mo par fichier.

Réduction de la taille du fichier GLB (initialement +700 Mo) :

1. Redimensionnement des textures :
    - Passage des images de 4K à 1024×1024 pour la color map, roughness, metallic et normal
    - Remplacement des anciennes images par les versions allégées → 488 Mo
2. Réduction du nombre de faces avec le modificateur Decimate :
    - Principalement pour les objets lourds (rideaux, panneaux)
    - Taux de decimate utilisé : 0,3 → 167 Mo
3. Duplication des éléments (tables, chaises, etc.) :
    - Duplication avec Shift+D → taille 421 Mo
    - Duplication avec Alt+D → taille 171 Mo
4. Compression Draco → taille finale : 12 Mo

Problème WebGL :

- J’avais ajouté une fonctionnalité de highlight des éléments sélectionnés sur Three.js, mais cela provoquait des bugs WebGL importants, donc elle a été supprimée.
- Les panneaux blancs du mur du fond et du plafond avaient plus d’1 million de faces, le modificateur Decimate détruisait trop la géométrie, donc j’ai simplement supprimé la partie problématique et rempli avec F (faces).
- Réduction globale de la taille de la salle et repositionnement des éléments pour éviter les flickers sur les murs.

Intégration du modèle pour le web

Le modèle 3D a été intégré dans une page web avec Three.js, permettant à l'utilisateur d'interagir directement avec la scène.

Fonctionnalités disponibles :

- Navigation dans la scène : déplacement avec la souris ou le clavier grâce à des event listeners.

```
// Avancer / Reculer
if (keys.current["KeyZ"] || keys.current["KeyW"]) {
  camera.position.add(forward.clone().multiplyScalar(velocity));
}
if (keys.current["KeyS"]) {
  camera.position.add(forward.clone().multiplyScalar(-velocity));
}
```

- Activation/désactivation du backface culling.

```
export default function BackfaceController({ scene, backfaceCulling }) {
  useEffect(() => {
    if (!scene) return;

    scene.traverse((obj) => {
      if (obj.isMesh) {
        obj.material.side = backfaceCulling ? THREE.FrontSide : THREE.DoubleSide;
        obj.material.needsUpdate = true;
      }
    });
  }, [scene, backfaceCulling]);

  return null; // invisible
}
```

- Changement de la couleur de fond de la scène.

```
export default function BackgroundController({ color = "#cccccc" }) {
  const { scene } = useThree();

  useEffect(() => {
    if (scene) {
      scene.background = new THREE.Color(color);
    }
  }, [color, scene]);

  return null; // composant invisible
}
```

- Masquage des éléments de la scène selon les besoins.

```
export default function VisibilityController({ scene, visibleElements }) {
  useEffect(() => {
    if (!scene) return;

    scene.traverse((obj) => {
      const model = ModelDataVisi.find((m) => m.objectNames.includes(obj.name));
      if (model && obj.isMesh) {
        obj.visible = visibleElements?.[model.id] ?? true;
      }
    });
  }, [scene, visibleElements]);

  return null;
}
```

- Zoom ou focus sur un élément spécifique de la scène.

```
export default function ZoomController({ controlsRef, scene, targetName }) {
  const { camera } = useThree();

  useEffect(() => {
    if (!scene || !controlsRef.current || !targetName) return;

    const object = scene.getObjectByName(targetName);
    if (!object) return;

    const box = new THREE.Box3().setFromObject(object);
    const center = box.getCenter(new THREE.Vector3());
    const size = box.getSize(new THREE.Vector3()).length();
    const direction = new THREE.Vector3(1, 1, 1).normalize();
    const distance = size * 2.5;
    const targetPosition = center.clone().add(direction.multiplyScalar(distance));

    const startPos = camera.position.clone();
    const startTarget = controlsRef.current.target.clone();
    let startTime = null;
    const duration = 1000;
    const easeOutQuad = (t) => t * (2 - t);

    const animate = (time) => {
      if (!startTime) startTime = time;
      const t = Math.min((time - startTime) / duration, 1);
      const eased = easeOutQuad(t);

      camera.position.lerpVectors(startPos, targetPosition, eased);
      controlsRef.current.target.lerpVectors(startTarget, center, eased);
      controlsRef.current.update();

      if (t < 1) requestAnimationFrame(animate);
    };

    requestAnimationFrame(animate);
  }, [scene, controlsRef, targetName]);
```