

Documentație proiect SPG – *House Project*

Albert Hunor-Istvan gr8

2026

Cuprins

1 Prezentarea temei	3
2 Scenariul	3
2.1 Descrierea scenei și a obiectelor	3
2.2 Funcționalități	5
3 Detalii de implementare	8
3.1 Funcții și algoritmi	8
3.1.1 Randare în doi pași: depth pass + color pass (shadow mapping)	8
3.1.2 Alpha test pentru texturi cu transparență	9
3.1.3 Iluminare: direcțională + punctuală	9
3.1.4 Efect de ceată (fog)	9
3.1.5 Sky (fundal cer) randat separat	10
3.1.6 Încărcarea modelelor OBJ și a materialelor	10
3.1.7 Input și camera first-person	10
3.1.8 Fizică simplificată și restricții de mișcare	10
3.1.9 Coliziuni și interacțiuni	10
3.1.10 Soluții posibile	11
3.1.11 Motivarea abordării alese	11
3.2 Modelul grafic	11
3.3 Structuri de date	12
3.4 Ierarhia de clase	12
3.4.1 Clase	12
3.4.2 Fișier principal (controlul aplicației)	13
3.4.3 Resurse și module auxiliare	13
4 Prezentarea interfeței grafice utilizator / Manual de utilizare	13
4.1 Pornirea aplicației	13
4.2 Controlul camerei	13
4.3 Deplasare	13
4.3.1 Mod fără fizică (Fly Mode)	14
4.3.2 Mod cu fizică (Physics Mode)	14
4.4 Interacțiuni cu obiecte	15
4.4.1 Uși	15
4.4.2 Lampă (lumină punctuală în interior)	15
4.4.3 Editarea poziției canapelei	16

4.5 Efecte vizuale	16
4.5.1 Ceață (Fog)	16
4.5.2 Wireframe	17
4.6 Rezumat comenzi	17
5 Concluzii și dezvoltări ulterioare	18
5.1 Concluzii	18
5.2 Dezvoltări ulterioare	19
6 Referințe	19

1 Prezentarea temei

Scopul proiectului este realizarea unei scene 3D interactive folosind OpenGL (profil core 3.3), în care utilizatorul poate explora un mediu de tip *casă + exterior*. Scena include randare de modele 3D încărcate din fișiere .obj, texturi 2D, iluminare direcțională și punctuală, efect de ceată (*fog*), *shadow mapping* și un *skybox*.

2 Scenariul

Aplicația reprezintă o scenă 3D interactivă (tip *walkthrough*) în care utilizatorul explorează o casă (exterior + interior) folosind o cameră de tip *first-person*. Scena include obiecte texturate încărcate din fișiere .obj, iluminare direcțională (soarele), o lumină punctuală controlabilă (lampa din interior), umbre prin *shadow mapping* (cu filtrare PCF), un fundal de tip *sky* randat separat și efect de ceată comutabil.

2.1 Descrierea scenei și a obiectelor

Scena conține un teren exterior (iarbă) pe care este amplasată casa și mai mulți copaci. Casa este formată din componente separate încărcate ca modele OBJ: pereți exteriori, pereți interiori, podea și acoperiș, fiecare cu texturi aplicate. În interior există obiecte de mobilier (canapea, masă, lampă), iar utilizatorul poate intra în casă și privi în jur.



Figura 1: Vedere exterioară: casa, terenul texturat și copacii din jur.

Fundalul cerului este realizat printr-un cub (*skybox*) randat la final, folosind un shader separat. Textura cerului este o imagine 2D, iar coordonatele UV sunt obținute prin mapare sferică din direcția fragmentului (folosind funcțiile **atan** și **asin**).

Uși interactive. Casa are două uși încărcate ca modele separate. Ușile se rotesc în jurul balamalelor pentru animația de deschidere/închidere. Starea ușilor este vizibilă în imaginile de mai jos: ușă închisă și ușă deschisă.



Figura 2: Ușă închisă (stare inițială).



Figura 3: Ușă deschisă după interacțiune (rotație în jurul balamalei).

Interior și mobilier. Interiorul conține o canapea, o lampă și o masă. Pereții interiori, podeaua (lemn) și tavanul sunt texturate. Mobilierul este randat ca modele OBJ cu transformări (translație/rotație/scalare) aplicate în timpul randării.



Figura 4: Vedere interioară: podea din lemn, peretii/tavan texturate și mobilier.

Iluminare interioară. Pe lângă lumina direcțională (soarele), scena folosește o lumină punctuală plasată în poziția lămpii. Când lampa este activată, interiorul este iluminat suplimentar cu atenuare în funcție de distanță (cădere realistă a intensității).

Vegetatie și transparentă. Pentru obiectele care folosesc texturi cu canal alpha (de exemplu frunziș), se aplică un *alpha test*: fragmentele cu transparentă sub un prag sunt eliminate (*discard*). Aceeași regulă este aplicată și în randarea pentru harta de umbre (depth pass), astfel încât umbrele să respecte forma reală a frunzelor.

2.2 Funcționalități

Interacțiunea utilizatorului cu scena este realizată prin tastatură și mouse, aplicația oferind următoarele funcționalități:

- **Control cameră first-person:** orientarea camerei se face cu mouse-ul (yaw/pitch), iar deplasarea cu tastele (W/A/S/D).
- **Două moduri de deplasare:**
 - **Mod cu fizică** (gravitație + săritură): utilizatorul este menținut pe podele/-panete, cu viteză verticală și gravitație; săritura se face cu Space.
 - **Mod liber (fly)**: deplasare fără gravitație, inclusiv urcare/coborâre pe axa verticală.



Figura 5: Interior cu lampa pornită: contribuția luminii punctuale este vizibilă pe podea și peretei.

- **Coliziuni:** mișcarea este restrictionată astfel încât utilizatorul să nu treacă prin peretei și uși și să rămână în zona permisă a scenei. Pentru pante se calculează înălțimea suprafetei și se corectează poziția camerei astfel încât să „stea” pe rampă.
- **Interacțiune uși (tasta E):** când utilizatorul este suficient de aproape de o ușă, aceasta poate fi deschisă/închisă. Deschiderea este animată prin rotație, iar direcția (sensul) poate depinde de poziția utilizatorului față de ușă.
- **Lampă ON/OFF (tasta L):** lampa poate fi comutată doar în proximitate, activând/dezactivând lumina punctuală din interior.
- **Mod editare canapea (tasta M):** canapeaua poate fi mutată cu săgețile, iar rotația se poate ajusta cu scroll-ul.



Figura 6: Mod liber (fizică dezactivată): explorare rapidă, inclusiv pe verticală.



Figura 7: Mod editare canapea: reposiționare în interior (deplasare și rotație).

- **Ceață ON/OFF (tasta F):** se aplică un efect atmosferic calculat în shader în funcție de distanța cameră–fragment.

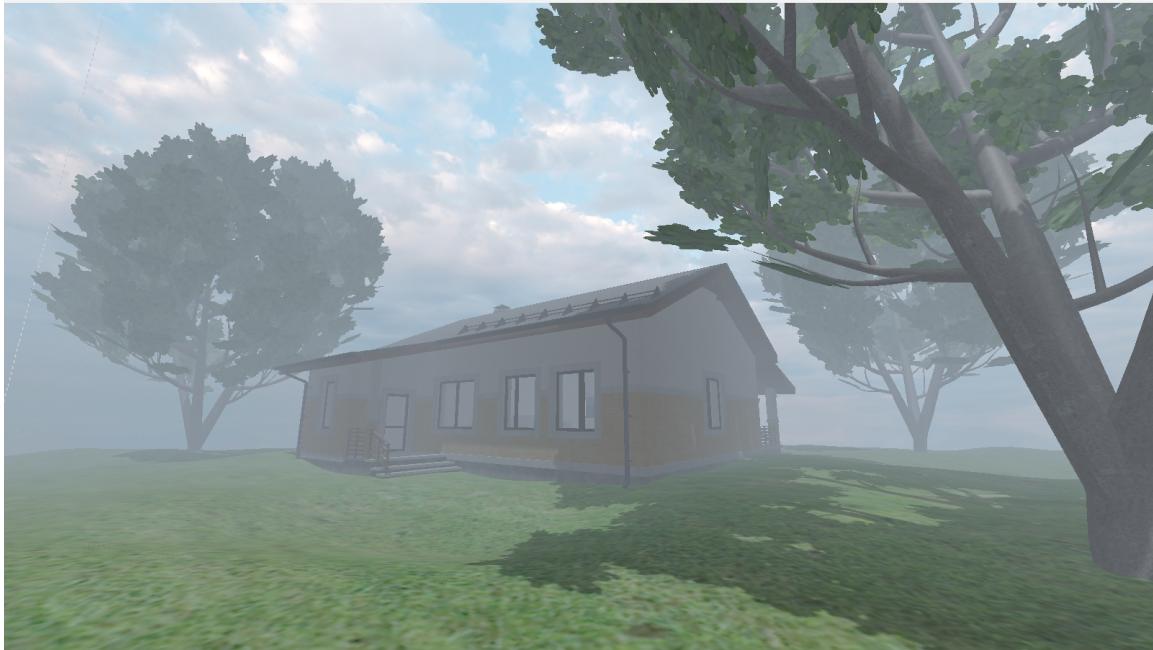


Figura 8: Efect de ceață activat: reduce contrastul la distanță și creează atmosferă.

- **Wireframe (tasta F2):** comutare între randare normală și randare în mod sârmă pentru inspectarea geometriei.

3 Detalii de implementare

Aplicația este implementată în C++ folosind **OpenGL 3.3 Core**, **GLFW** (fereastră + input), **GLEW** (încărcare extensii), **GLM** (matrici și vectori), **stb_image** (încărcare texturi) și **tinyobjloader** (încărcare modele .obj cu materiale).

3.1 Funcții și algoritmi

3.1.1 Randare în doi pași: depth pass + color pass (shadow mapping)

Umbrele sunt generate prin *shadow mapping*, folosind un framebuffer dedicat cu o textură de adâncime (`depthMap`). Procesul are două etape:

- **Depth pass:** scena este randată din perspectiva luminii direcționale într-o hartă de adâncime (`shadowMap`). Se folosește o matrice `lightSpaceMatrix` (proiecție ortografică + `lookAt`).
- **Color pass:** scena este randată normal din perspectiva camerei; fragment shaderul compară adâncimea curentă cu valoarea din `shadowMap` pentru a decide dacă fragmentul este în umbră.

Pentru a obține umbre mai plăcute vizual, se aplică **PCF (Percentage Closer Filtering)**: se eșantionează mai multe valori din `shadowMap` în jurul coordonatei curente și se face media rezultatelor.

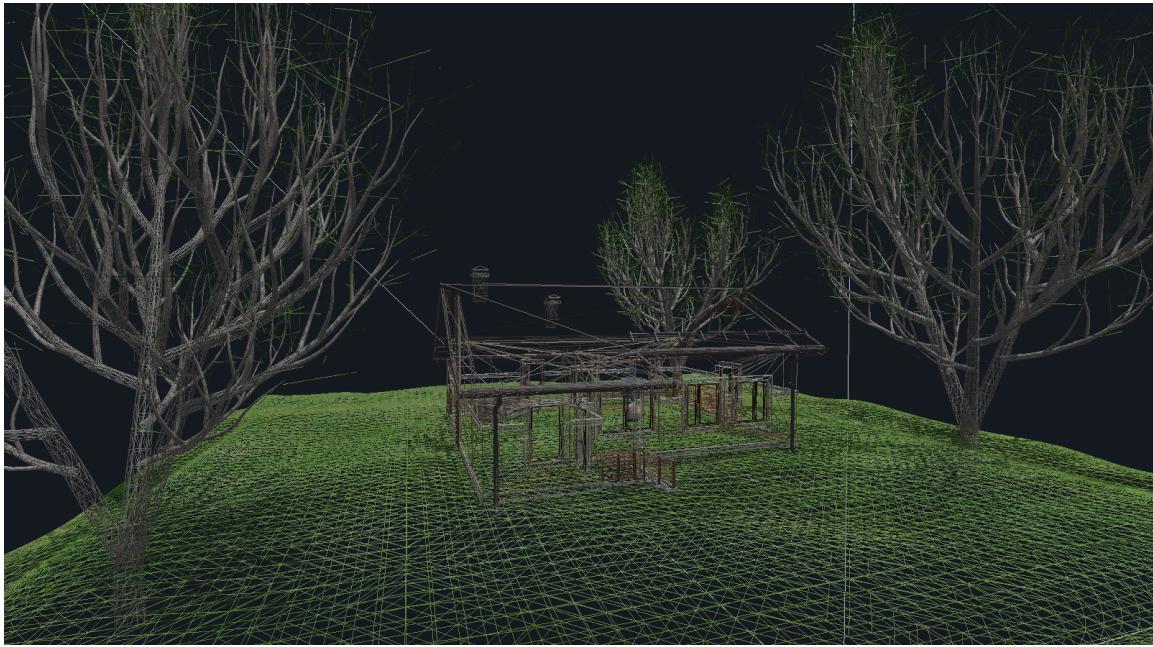


Figura 9: Mod wireframe: vizualizarea structurii geometrice a scenei.

3.1.2 Alpha test pentru texturi cu transparentă

Pentru obiecte cu texturi care conțin canal alpha (de ex. frunziș), se aplică un prag (*alpha test*) în shader:

- în **color pass**, fragmentele cu alpha sub un prag sunt eliminate (**discard**);
- același comportament este aplicat și în **depth pass**, astfel încât umbra să respecte conturul real al obiectelor transparente.

3.1.3 Iluminare: direcțională + punctuală

Scena folosește două surse de lumină:

- **Lumină direcțională** (soarele): componentă difuză + speculară (Blinn-Phong), afectată de umbră (**shadowMap**).
- **Lumină punctuală** (lampa): difuză + speculară, cu atenuare în funcție de distanță (*falloff* realist). Este comutabilă din input (ON/OFF).

În plus, este adăugat un termen de **ambient** relativ ridicat pentru a simula lumină indirectă în scenă.

3.1.4 Efect de ceată (fog)

Ceața este calculată în fragment shader în funcție de distanță dintre cameră și fragment. Când este activată, culoarea finală este amestecată cu o culoare de ceată, cu o intensitate controlată de **fogDensity**.

3.1.5 Sky (fundal cer) randat separat

Fundalul cerului este randat la final cu un shader separat și cu **depth func** setat la **GL_EQUAL**, pentru a evita problemele de *depth testing*. Textura cerului este o **imagină 2D**, iar coordonatele UV sunt obținute prin **mapare sferică** din direcția fragmentului (vectorul **TexCoords**).

3.1.6 Încărcarea modelelor OBJ și a materialelor

Modelele sunt încărcate cu **tinyobjloader**. Pentru fiecare triunghi:

- se citesc pozițiile, normalele și coordonatele UV dacă există;
- dacă modelul nu are normale, se calculează normalul triunghiului și se atribuie tuturor celor 3 vârfuri;
- triunghiurile sunt grupate pe **material** (material id), astfel încât fiecare grup să poată avea propria textură (**diffuse_texname**).

La randare, dacă există grupuri de materiale, se face **glDrawArrays** pe intervalele corespunzătoare fiecărui grup, legând textura materialului (sau o textură fallback setată manual).

3.1.7 Input și camera first-person

Camera folosește yaw/pitch pentru orientare:

- mouse-ul actualizează yaw/pitch, iar vectorul **camFront** se recalculează din acestea;
- tastatura controlează mișcarea în plan (W/A/S/D), iar în modul liber se permite și mișcare pe verticală.

3.1.8 Fizică simplificată și restricții de mișcare

Aplicația include un mod cu gravitație și săritură:

- se acumulează viteza verticală (**verticalVel**) sub acțiunea gravitației;
- camera este menținută deasupra podelei folosind o înălțime a „ochilor” (**eyeHeight**);
- pentru pante, se calculează înălțimea suprafetei prin ecuația planului definit de 3 puncte, apoi poziția camerei este corectată dacă „picioarele” ar intra sub pantă.

În modul fără fizică, camera se mișcă liber (fără gravitație), dar rămân active verificările de coliziune cu ușile/pereții.

3.1.9 Coliziuni și interacțiuni

- **Zonă de deplasare**: mișcarea este permisă doar în anumite poligoane 2D (plan XZ), folosind un test de tip *point-in-polygon* (ray casting).
- **Pereți**: coliziunea se face prin verificare de tip *bounding* pe un „perete” definit de 4 colțuri (se folosește un prag de grosime).

- **Uși:** coliziunea ține cont de unghiul de rotație; colțurile ușii sunt rotite în jurul balamalei, apoi se verifică intersecția poziției camerei cu volumul rezultat.
- **Interacțiune uși:** când utilizatorul este în proximitate, apăsarea tastei de interacțiune comută starea ușii; animația se face prin interpolarea unghiului curent către un unghi țintă cu o viteză (`doorSpeed`).
- **Interacțiune lampă:** în proximitate, lampa comută lumina punctuală (culoare ON/OFF).
- **Editare canapea:** poziția canapelei se poate modifica din input, iar rotația se ajustează din scroll (în modul de editare).

3.1.10 Soluții posibile

- **Umbre:** alternativ la shadow mapping + PCF se puteau folosi VSM/ESM sau cascaded shadow maps (pentru scene mari), însă complexitatea ar crește.
- **Coliziuni și fizică:** alternativ se putea integra un motor (Bullet/PhysX), însă pentru un proiect educațional, o fizică simplă + coliziuni personalizate este suficientă și ușor de controlat.
- **Sky:** alternativ la textura 2D cu mapare sferică se putea folosi un cubemap real (`samplerCube`), însă varianta 2D reduce numărul de resurse și simplifică încărcarea.

3.1.11 Motivarea abordării alese

Abordarea implementată urmărește un echilibru între:

- **calitate vizuală** (umbrare PCF, iluminare combinată, ceată, sky randat separat),
- **complexitate moderată** (fără biblioteci grele pentru fizică),
- **control explicit** asupra interacțiunilor (uși/lampă/mobilier) și asupra comportamentului camerei.

3.2 Modelul grafic

Modelul de randare este bazat pe:

- **matrici standard:** `model`, `view`, `projection`;
- **normal matrix:** în vertex shader se folosește `mat3(transpose(inverse(model)))` pentru transformarea corectă a normalelor;
- **shadow mapping:** `FragPosLightSpace` este calculat în vertex shader și folosit în fragment shader pentru determinarea umbrei;
- **pipeline:** (1) depth pass în `depthMapFB0` → (2) randare normală cu texturi + iluminare + umbre → (3) randare sky la final.

3.3 Structuri de date

Implementarea folosește structuri simple și eficiente:

- **Geometrie OBJ:** `std::vector<ObjVertex>` unde fiecare vârf conține poziție, normal și UV.
- **Materiale:** `std::vector<MaterialGroup>` pentru a reține intervalele de vertecși asociate fiecărui material și textura lui.
- **Resurse OpenGL:** pentru fiecare model: VAO, VBO + un `textureID` (fallback).
- **Stare scenă (runtime):** variabile globale pentru cameră (poziție/orientare), timp (`deltaTime`), starea ușilor (unghi current/țintă), starea luminii, starea ceții, poziția/rotația canapelei etc.
- **Coliziuni:** poligoane 2D pentru zone de podea (plan XZ) și seturi de colțuri 3D pentru pante și uși.

3.4 Ierarhia de clase

Proiectul are o structură simplă, cu o singură clasă centrală pentru gestionarea modelelor 3D, iar logica aplicației este organizată procedural în fișierul principal.

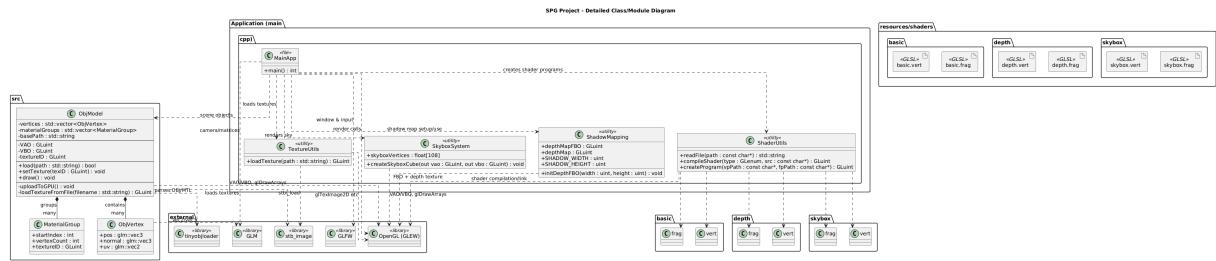


Figura 10: Diagrama de clase și module a proiectului (fără modulul de debug).

3.4.1 Clase

- **ObjModel (src/ObjModel.h, src/ObjModel.cpp)**
 - `bool load(const std::string& path):` încarcă un model .obj folosind `tinyobjloader`, citeste poziții/normale/UV și grupează vertecșii pe materiale; dacă lipsesc normalele, le calculează.
 - `void uploadToGPU():` creează și încarcă buffer-ele OpenGL (VAO, VBO) și definește atributele de vertecși (poziție, normal, coordonate UV).
 - `void setTexture(GLuint texID):` setează o textură implicită (fallback) pentru model.
 - `void draw() const:` rindează modelul fie cu textura implicită, fie pe grupuri de materiale (când există materiale/texuri în .mtl).

3.4.2 Fișier principal (controlul aplicației)

- `main.cpp`: initializează fereastra și contextul OpenGL (GLFW/GLEW), încarcă shaderele, texturile și modelele, gestionează input-ul (cameră + interacțiuni) și rulează bucla principală de randare (depth pass pentru umbre + randare normală + sky).

3.4.3 Resurse și module auxiliare

- **Shadere** (`resources/shaders/`): `basic.vert/.frag` (randare principală), `depth.vert/.frag` (shadow map), `skybox.vert/.frag` (fundal cer).
- **Modele și texturi** (`resources/models/`): organizate pe categorii (`house`, `interior`, `furniture`, `ground`, `sky`, etc.).
- **Biblioteci externe** (`external/`): `stb_image.h` pentru imagini/texturi, `tiny_obj_loader.*` pentru `.obj`, iar `glm/` pentru matematică (vectori/matrice).
- **Build**: `CMakeLists.txt` (configurare CMake), `BUILD_INSTRUCTIONS.md` și `build.bat` (instructiuni/script de compilare).

4 Prezentarea interfeței grafice utilizator / Manual de utilizare

Aplicația rulează într-o fereastră OpenGL (GLFW), cu cameră tip *first-person*. Utilizatorul poate explora scena (exterior + interior), poate interacționa cu ușile și lampa, poate activa/dezactiva efecte vizuale (ceată, wireframe) și poate comuta între moduri de mișcare (*fly* vs. *physics*).

4.1 Pornirea aplicației

Resursele (modele, texturi, shadere) sunt încărcate din folderul `resources/`. Din acest motiv, aplicația trebuie rulată cu **working directory** setat astfel încât căile relative de forma `resources/...` să fie valide (ex.: rulare din directorul rădăcină al proiectului).

4.2 Controlul camerei

- **Mouse**: rotire cameră (yaw/pitch).
- **Scroll**: zoom (modifică FOV) *când nu este activ modul de editare al canapelei*.
- **ESC**: închide aplicația.

4.3 Deplasare

Aplicația are două moduri principale de mișcare:

4.3.1 Mod fără fizică (Fly Mode)

În acest mod utilizatorul se poate deplasa liber, inclusiv pe verticală.

- **W/A/S/D**: înainte / stânga / înapoi / dreapta
- **SPACE**: urcare
- **Q**: coborâre



Figura 11: Exemplu de explorare a scenei în modul fără fizică (deplasare liberă).

4.3.2 Mod cu fizică (Physics Mode)

Acet mod simulează o mișcare mai realistă:

- gravitație (cădere controlată)
- coliziune cu podeaua (zone definite) și cu elementele structurale (pereti/uși)
- **săritură** (jump) atunci când utilizatorul are suport sub picioare

Controale:

- **G**: activează/dezactivează fizica
- **W/A/S/D**: deplasare pe planul XZ
- **SPACE**: săritură (doar dacă există suport)



Figura 12: Mișcare cu fizică activată: gravitație + coliziuni (interior/exterior).

4.4 Interacțiuni cu obiecte

4.4.1 Uși

Ușile pot fi deschise/închise doar dacă utilizatorul este suficient de aproape. Interacțiunea declanșează o animație de rotație (deschidere progresivă).

- **E:** deschide/închide ușa (când utilizatorul este în proximitate)

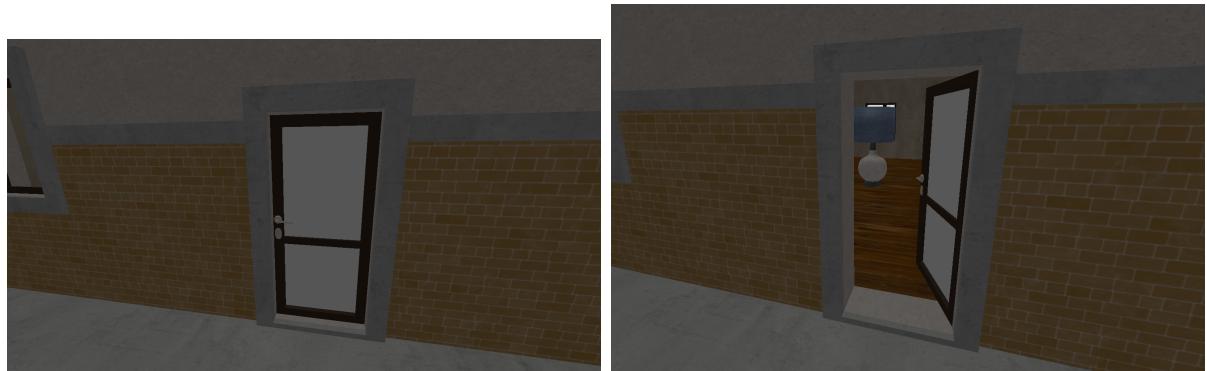


Figura 13: Interacțiune cu ușa: stare închisă vs. deschisă.

4.4.2 Lampă (lumină punctuală în interior)

În interior există o lampă care poate fi pornită/oprită în funcție de proximitatea utilizatorului. Pornirea lămpii activează o sursă de lumină punctuală (point light) ce contribuie la iluminarea scenei.

- **L:** pornește/oprește lumina lămpii (când utilizatorul este în proximitate)

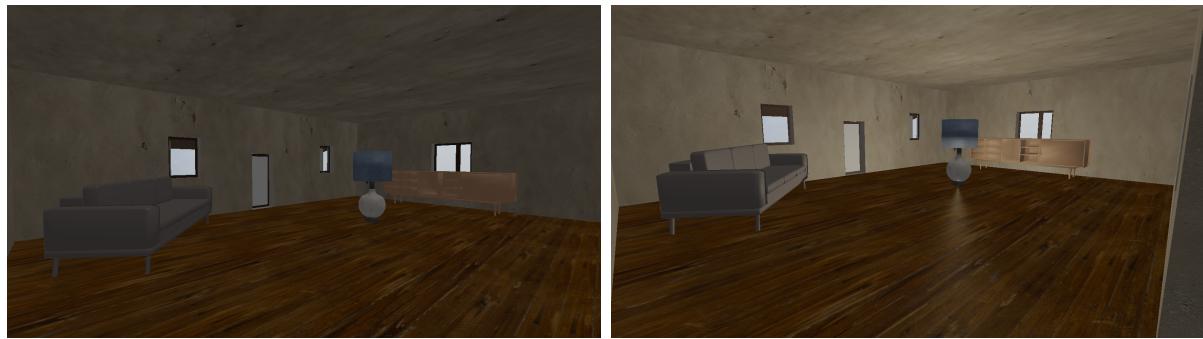


Figura 14: Interior: iluminare implicită vs. iluminare cu lampă pornită.

4.4.3 Editarea poziției canapelei

Aplicația permite mutarea canapelei în scenă, ca o funcționalitate interactivă. Când modul este activ, rotița mouse-ului nu mai controlează FOV, ci rotația canapelei.

- **M:** activează/dezactivează *Sofa Edit Mode*
- **Săgeți** ($\uparrow \downarrow \leftarrow \rightarrow$): mută canapeaua pe planul XZ
- **Scroll:** rotește canapeaua



Figura 15: Mod de editare: reposiționarea canapelei în interior.

4.5 Efecte vizuale

4.5.1 Ceață (Fog)

Ceață poate fi activată/dezactivată pentru a modifica atmosfera scenei și profunzimea percepută.

- **F:** activează/dezactivează ceață

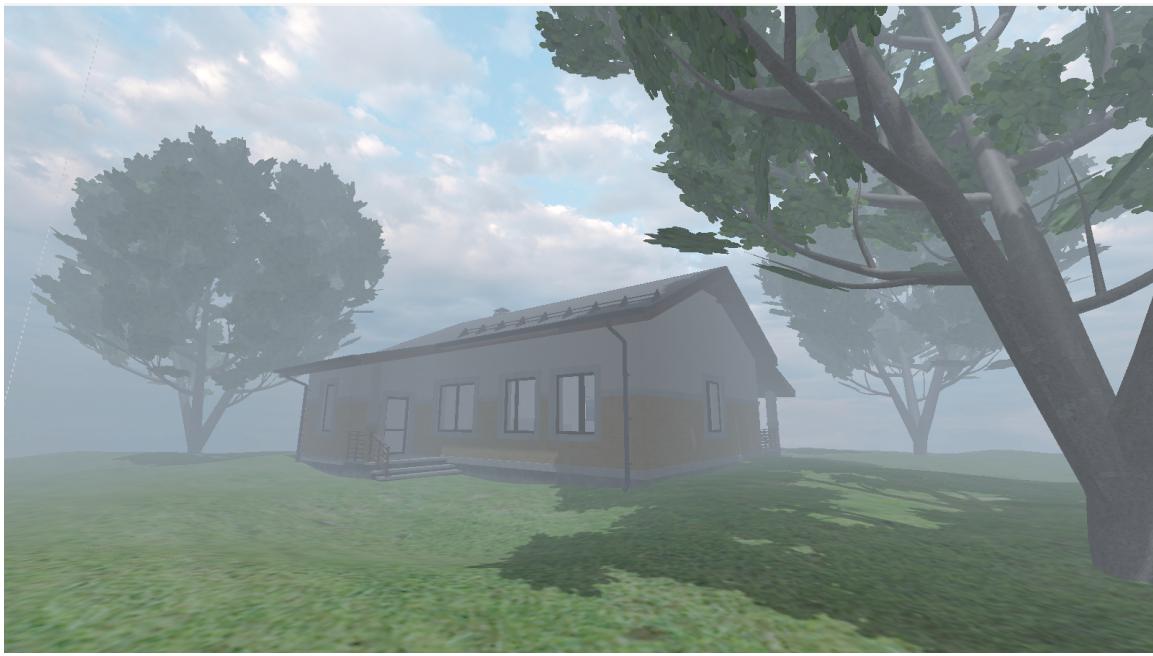


Figura 16: Efect de ceată activat: scade vizibilitatea la distanță și schimbă atmosfera scenei.

4.5.2 Wireframe

Wireframe este util pentru a vizualiza structura geometrică (triangulația) a obiectelor.

- **F2:** activează/dezactivează randarea wireframe

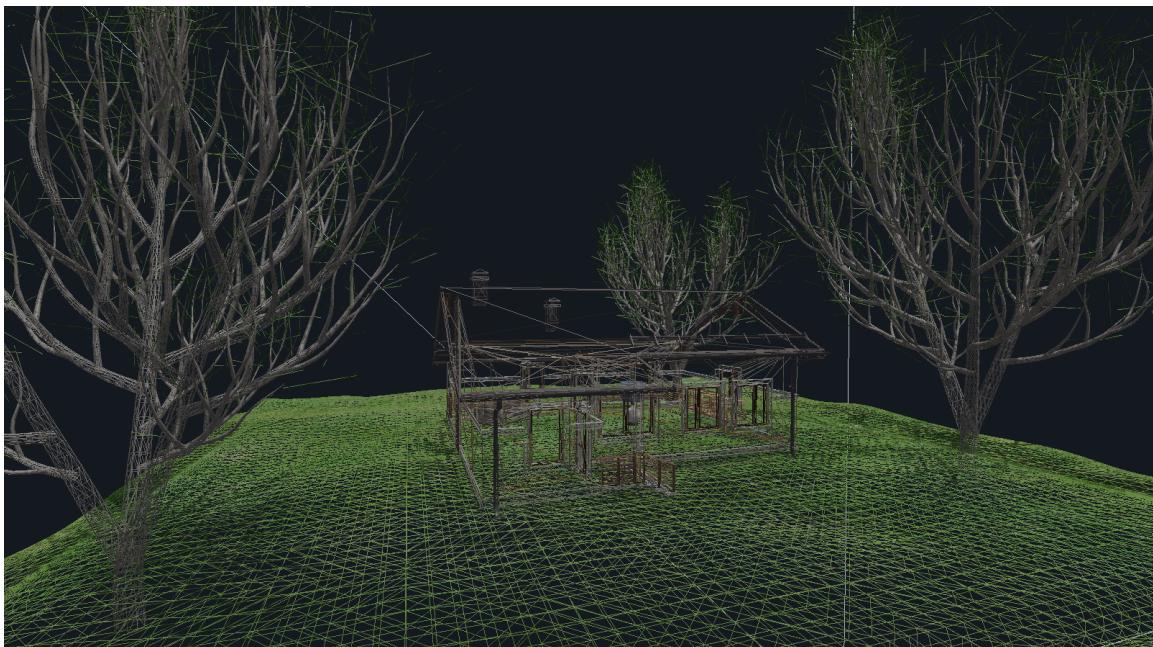


Figura 17: Mod wireframe: evidențierea geometriei scenei.

4.6 Rezumat comenzi

- **W/A/S/D:** deplasare

- **Mouse:** rotire cameră
- **Scroll:** zoom (FOV) / rotație canapea (în Sofa Edit Mode)
- **G:** physics ON/OFF
- **SPACE:** urcare (fly) / săritură (physics)
- **Q:** coborâre (fly)
- **E:** uși ON/OFF (în proximitate)
- **L:** lampă ON/OFF (în proximitate)
- **M:** Sofa Edit Mode ON/OFF
- **F:** ceață ON/OFF
- **F2:** wireframe ON/OFF
- **ESC:** ieșire

5 Concluzii și dezvoltări ulterioare

Proiectul a avut ca scop realizarea unei scene 3D interactive, randată în timp real cu OpenGL, care să includă atât elemente de exterior (teren, vegetație, cer), cât și un interior navigabil (casă + obiecte de mobilier), împreună cu interacțiuni și efecte vizuale specifice.

5.1 Concluzii

În forma curentă, aplicația îndeplinește obiectivele propuse prin:

- **Randare 3D completă a scenei** folosind modele .obj încărcate din fișiere, cu texturi aplicate pe obiecte (teren, casă, interior, podea, tavan, mobilier, copaci).
- **Navigare first-person** cu rotație din mouse și deplasare cu tastatura, oferind două moduri de explorare:
 - deplasare liberă (fără fizică),
 - deplasare cu fizică (gravitație, suport, săritură) și restricționare pe zonele de podea.
- **Interacțiuni în scenă** care cresc gradul de realism și utilitatea aplicației:
 - uși animate (deschidere/închidere) în funcție de proximitate,
 - comutarea unei surse de lumină punctuală (lampă),
 - editarea poziției și rotației canapelei în runtime.
- **Efecte vizuale** care îmbunătățesc aspectul final:
 - *shadow mapping* pentru umbre dinamice,
 - ceață (fog) controlabilă din tastatură,

- skybox texturat (cer) pentru un fundal natural,
- mod wireframe pentru vizualizarea structurii geometriei.

Pe ansamblu, proiectul rezultă într-o aplicație stabilă, coerentă ca scenă și funcționalități, care demonstrează integrarea completă a unui pipeline de randare (shader-e, texturi, modele, iluminare, umbre) împreună cu interacțiuni și logică de navigare.

5.2 Dezvoltări ulterioare

Pe baza implementării existente, aplicația poate fi extinsă în mai multe direcții:

- **Skybox îmbunătățit:** înlocuirea mapării sferice pe un *cubemap* real (6 fețe), pentru o proiecție mai corectă și mai flexibilă.
- **Sistem de coliziuni mai general:** trecerea de la verificări pe poligoane/plane la volume de coliziune dedicate (AABB/OBB), sau chiar o bibliotecă de fizică (ex: Bullet) pentru coliziuni robuste.
- **Iluminare avansată:** suport pentru mai multe *point light*-uri în interior, intensități variabile, culori diferite și un sistem de „switch” pentru fiecare sursă.
- **Materiale PBR:** introducerea texturilor de tip normal/roughness/metallic și implementarea unui shader PBR pentru realism crescut.
- **Interacțiuni suplimentare:** alte obiecte interactive (sertare, geamuri, comutatoare), animații mai complexe și feedback vizual la selecție (highlight).
- **UI simplu în overlay:** afișarea pe ecran a controalelor, a modului curent (fog/-physics/sofa edit), sau a unor indicatori (FPS, poziție).
- **Optimizare:** frustum culling, instancing pentru obiecte repetitive (copaci), LOD pentru vegetație și optimizarea încărcării modelelor mari.

Acstea îmbunătățiri ar transforma proiectul dintr-o scenă interactivă demonstrativă într-o bază solidă pentru un mini-engine sau pentru o aplicație mai complexă (simulator de explorare, walkthrough arhitectural, prototip de joc etc.).

6 Referințe

Bibliografie

- [1] Poly Haven – Wall / Brick Textures (accesat pentru texturi PBR și diffuse).
<https://polyhaven.com/textures/brick/wall>
- [2] Sketchfab – Căutare modele descărcabile (folosit pentru identificarea/descărcarea unor modele 3D).
https://sketchfab.com/search?features=downloadable&q=wall+texture&sort_by=-likeCount
- [3] Free3D – Colecție modele 3D (surse de modele compatibile Blender/OBJ).
<https://free3d.com/3d-models/blender>

- [4] Document de suport / cerință proiect (descriere structură documentație).
https://docs.google.com/document/d/1njtWPMm0QNIaD_z9ve8iPRUqQTWdIV_P0-NvPD0n0uM/edit
- [5] Stb Image.h.
https://github.com/nothings/stb/blob/master/stb_image.h