1. **Functionalitati de baza:**
    a. **API 3D : OpenGL (recomandat) sau Direct3D**
    Voi folosi OpenGL.
    b. **Format Intern propriu de Mesh-uri pentru obiecte**
        i. **Triunghiuri**
        Se va implementa folosind un VAO.
        ii. **Materiale**
        Se va implementa folosind shadere. Fiecare material se va realiza printr-un vertex shader, un fragment shader si eventual un geometry shader.
        iii. **Textura**
        Se va implementa folosind un VBO.
        iv. **Diverse (normale, coordonate textura, etc)**
        Se va implementa folosind VBO pentru fiecare dintre acestea. VBO-urile pentru pozitie, culoare, normale si coordonate de textura vor fi legate la VAO-ul mesh-ului.
    c. **Modul de 3D Math**
    Voi folosi GLM.
    d. **Gestiune interactiune cu utilizatorul**
        i. **Input/Output Events**
        Voi folosi GLFW si GLEW.
        ii. **Mouse**
        Voi folosi GLFW, GLEW si voi implementa folosind functiile aferente de tratarea a input-urilor de la mouse.
        iii. **Keyboard**
        Voi folosi GLFW, GLEW si voi implementa folosind functiile aferente de tratarea a input-urilor de la tastatura.
    e. **Implementare Camera de tip FPS**
    O voi implementa folosind GLFW, GLEW si GLM. Voi implementa folosind o clasa Camera care va trata updatarea, rotirea, translatia si care va permite calcularea matricilor de vizualizare, proiectie si a matricii globale.
    f. **Model iluminare, material si texturi**
        i. **Model de iluminare functional**
        Voi folosi modelul Phong. Il voi implementa folosind GLSL.
        ii. **Texturare**
        O voi implementa folosind GLFW, GLEW si STB.
        iii. **Suport pentru cel putin 1 format de textura clasic : .jpg, .tga, .png, etc**
        Voi implementa suport pentru fisiere PNG. Il voi implementa folosind STB, GLFW si GLEW.
    g. **Importer pentru un format cunoscut de obiecte 3D : .x , .3ds, .ms3d , etc**
    Voi implementa suport pentru fisiere OBJ. Il voi implementa folosind ASSIMP, GLFW si GLEW.
    h. **Scene Management**
        i. **Organizare ierarhica a obiectelor / entitatilor scenei 3D**
        Voi folosi o structura arborescenta bazata pe un XML.
        ii. **Mecanism de culling , implementare algoritm propriu sau clasic (de ex. OctTrees)**
        Voi folosi GLEW. Voi implementa folosindu-ma de algoritmul Frustum Culling.

iii. **Format XML propriu pentru scena 3D**
Voi crea formatul XML in asa fel incat sa contina caracteristici esentiale pentru fiecare instanta a obiectelor din scena, cum ar fi pozitie, rotatie, scalare, etc.

i. **2D GUI System : text, meniuri, butoane, liste, edit boxes, etc**
Voi folosi GLM, GLFW, GLEW si STB. O voi implementa folosindu-ma de texturi si eventual plane-uri bidimensionale.

j. **Gestiune evenimente si scripting**
   i. **Limbaje de scripting : Python, Lua, Ruby**
   Voi folosi Python.
   ii. **Declansare si procesare evenimente in scena 3D**
   O voi implementa folosind 2 functii – Init() si Run(). Init() se refera la tratarea logicii imediat ce s-a creat lumea, iar Run() se refera la logica repetitiva din fiecare frame si schimbul de buffere.

k. **Detectia coliziunilor : basics**
Voi folosi GLM si voi implementa folosind comparatii de intersectie a obiectelor.

l. **Bucla principala a motorului grafic : integrarea tuturor subsistemele motorului grafic**
O voi implementa folosind o functie Update(), care se va afla in Run(). Aceasta functie se va apela intre momentele de start si de sfarsit al unui frame.

2. **Functionalitate la alegere:**
   a. **Efecte speciale : sistem de particule, obiecte transparente, apa, ceata volumetrica, cer realist**
   Voi folosi GLM, GLFW, GLEW si STB. Voi crea un editor care permite crearea de sisteme de particule simple folosind quad-uri avand ca textura o culoare la creare si una la distrugere, putandu-se alege durata lor de viata si viteza cu care se misca catre exterior. Sistemul de particule va imprastia quad-urile de la un punct catre exterior.