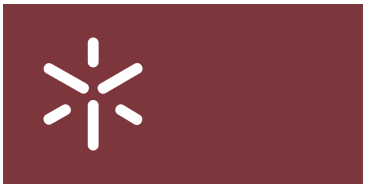

Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Unidade Curricular de Computação Gráfica

Ano Letivo de 2024/2025

Motor de Mini Cenas 3D Baseado em Grafos

Alex Sá a104257
José Vasconcelos a100763
Paulo Ferreira a96268
Rafael Fernandes a104271

27 de Abril, 2025

CG

Resumo

A crescente complexidade das aplicações gráficas modernas exige motores de renderização cada vez mais robustos e eficientes, capazes de integrar técnicas avançadas de iluminação, texturização e animação. Neste relatório, descreve-se o desenvolvimento da quarta e última fase de um motor gráfico 3D baseado em grafos, focando-se na implementação de iluminação dinâmica (suportando luzes pontuais, direcionais e spot), mapeamento de texturas e extensão do formato XML para definição de materiais e propriedades de luz. Esta fase culminou na demo final do sistema solar, agora com planetas texturizados, efeitos de luz realistas e animações suaves baseadas em curvas de Catmull-Rom, consolidando uma arquitetura escalável e otimizada através de Vertex e Index Buffer Objects (VBOs/IBOs).

Área de Aplicação: Computação Gráfica

Palavras-Chave: OpenGL; C++; 3D Engine; Iluminação; Mapeamento UV; Renderização 3D; Sistema Solar

Índice

1. Introdução **2**

2. Geração de Normais e Coordenadas de Textura **4**

3. Implementação de Iluminação **6**
3.1. Configuração XML de Iluminação e Materiais 6

4. Sistema de Texturas **8**

5. Demo Scene: Sistema Solar com Iluminação e Texturas **9**

6. Conclusão e Trabalho Futuro **11**

1. Introdução

No âmbito da unidade curricular de Computação Gráfica, o presente projeto visa o desenvolvimento progressivo de um motor gráfico tridimensional baseado em grafos. Até à fase anterior, foram estabelecidas as bases essenciais do sistema, incluindo a implementação de renderização otimizada através do uso de Vertex Buffer Objects (VBOs) e Index Buffer Objects (IBOs), a geração de superfícies complexas recorrendo a *Bezier patches* e a criação de animações dinâmicas fundamentadas em curvas de *Catmull-Rom*.

Nesta quarta e última fase, o trabalho centrou-se na integração de técnicas avançadas de iluminação e texturização, elementos cruciais para conferir realismo e profundidade às cenas tridimensionais. Com base no progresso alcançado previamente, o grupo enfrentou três desafios principais. O primeiro consistiu na extensão dos geradores de geometria, adaptando os algoritmos existentes para permitir a geração automática das coordenadas de textura (UV) e das normais por vértice, mesmo em modelos paramétricos como os *Bezier patches*. Destaca-se que o cálculo das normais, antecipado na fase anterior, revelou-se fundamental para garantir uma transição suave e correta para os modelos de iluminação aplicados.

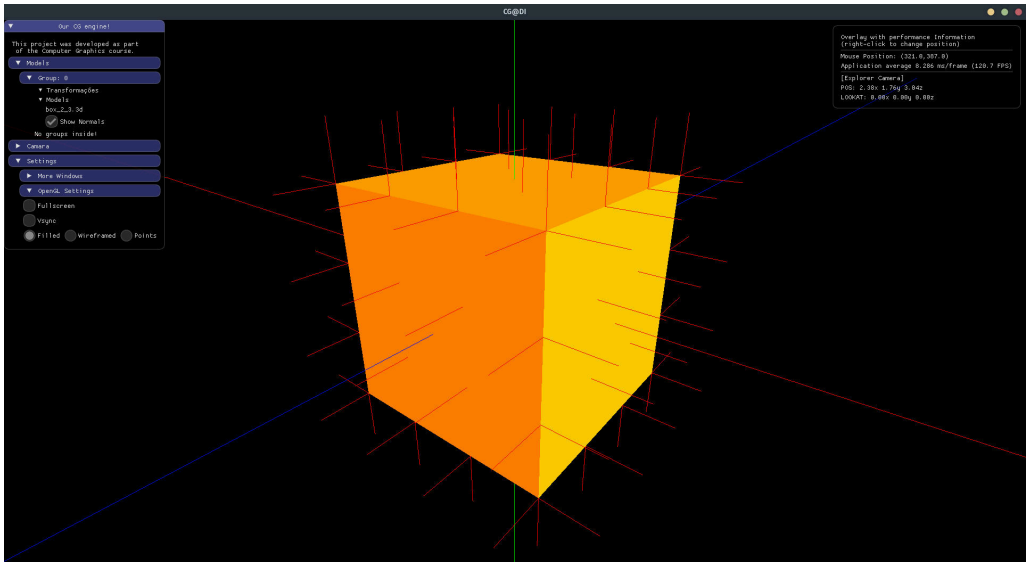

Figura 1: Exemplo de geração de normais

O segundo desafio incidiu na evolução do sistema de *parsing* XML, ampliando o formato de descrição das cenas para suportar propriedades materiais, tais como difusão (*diffuse*), especular (*specular*), emissão (*emissive*) e brilho (*shininess*), bem como a definição de diferentes tipos de luzes — pontuais, direccionais e *spotlights* — incluindo parâmetros essenciais como atenuação e orientação.

```xml
...
<world>
    <window width="512" height="512" />
    <camera>
        <position x="1.757135" y="1.799201" z="3.416921" />
        <lookAt x="0" y="0" z="0" />
        <up x="0" y="1" z="0" />
        <projection fov="60" near="1" far="1000" />
        <target targetId="cumeta">
            <model file="box_2_3.3d">
                <transform>
                    <rotate time="25" x="0" y="1" z="0" />
                    <scale x="10" y="10" z="10" />
                </transform>
            </model>
        </target>
    </camera>

    <lights>
        <light type="directional" dirx="1" diry="0.7" dirz="0.5"/>
    </lights>
    <group>
        <models>
            <model file="ring.3d" id="cumeta"> <!-- generator box 2 3 box_2_3.3d cone_1_2_4_3.3d -->

                    <texture file = "ring.jpg" />
            </model>
        </models>
    </group>
</world>
```

Figura 2: Esquema do sistema de parsing XML

Finalmente, procedeu-se à implementação efetiva da iluminação e das texturas no motor gráfico, processo que implicou a resolução de diversos desafios técnicos, nomeadamente a mitigação de artefactos visuais em texturas, como distorções evidenciadas em superfícies curvas.

O culminar deste trabalho traduziu-se numa demonstração final do sistema solar, enriquecida com texturas realistas aplicadas aos planetas (por exemplo, mapas da Terra e de Marte), iluminação dinâmica com o Sol a funcionar como fonte de luz pontual, e animações fluidas de translação e rotação herdadas das fases anteriores.

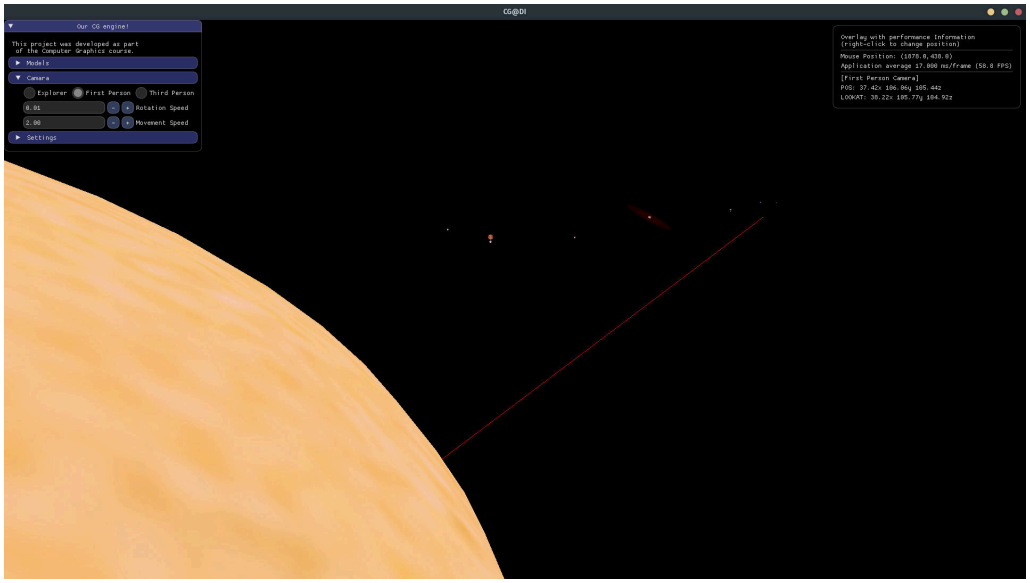

Figura 3: Demo do sistema solar

Esta fase constituiu não só uma consolidação dos conhecimentos teóricos adquiridos, como o modelo de iluminação de Gouraud e o mapeamento UV, mas evidenciou igualmente a importância de uma arquitetura modular que permita a evolução iterativa e escalável do motor gráfico.

2. Geração de Normais e Coordenadas de Textura

A geração correta das normais e das coordenadas de textura representa um aspeto fundamental para garantir a qualidade visual e a eficiência computacional do motor gráfico desenvolvido. Nesta secção, é descrita a abordagem adotada para as diferentes primitivas geométricas, com especial ênfase na parametrização matemática e nas decisões técnicas implementadas.

Relativamente ao cálculo das normais, foram aplicadas duas metodologias distintas, adequadas à natureza de cada primitiva. Para as superfícies paramétricas, nomeadamente os *Bezier patches* desenvolvidos na fase anterior, as normais são calculadas através do produto vetorial das derivadas parciais da função paramétrica que define a superfície. Formalmente, considerando a superfície paramétrica $\boldsymbol{P}(u, v)$, o vetor normal $\boldsymbol{n}$ é obtido por:

$$\boldsymbol{n} = \frac{\partial \boldsymbol{P}}{\partial u} \times \frac{\partial \boldsymbol{P}}{\partial v}$$

seguido de uma normalização do vetor resultante para garantir unidade. Esta abordagem permite obter normais precisas e contínuas ao longo da superfície parametrizada, fundamentais para a correta aplicação dos modelos de iluminação.

Nas prestante primitivas, como cubo, esfera, cone, cilindro,... adotou-se uma estratégia distinta. No caso particular do cubo, as normais são constantes em cada face e definidas antecipadamente conforme a orientação da mesma. Por exemplo, para a face superior do cubo, a normal é dada pelo vetor

$$\boldsymbol{n}\{+Y\} = (0, 1, 0)$$

Esta abordagem, simples mas eficiente, é adequada para superfícies planas e contribui para o efeito visual de arestas nítidas quando aplicado o modelo de iluminação de Gouraud.

No que respeita ao mapeamento UV, este foi implementado considerando as propriedades geométricas específicas de cada primitiva. Para o cubo, foi utilizada uma projeção planar independente para cada face, definindo as coordenadas UV no intervalo $[0, 1] \times [0, 1]$. Formalmente, para uma face genérica com origem $\boldsymbol{O}$ e vetores direcionais $\boldsymbol{u}$ e $\boldsymbol{v}$, a parametrização de um ponto $\boldsymbol{P}$ na face é dada por:

$$\mathbf{uv} = ((\boldsymbol{P} - \boldsymbol{O}) \cdot \boldsymbol{u}, (\boldsymbol{P} - \boldsymbol{O}) \cdot \boldsymbol{v})$$

garantindo uma distribuição uniforme da textura em cada face sem distorções, conforme exemplificado na implementação do *boxGenerator.cpp*, onde cada face é processada independentemente com vetores UV próprios. (foto com exemplo do mapeamento UV no cubo)

Para as demais primitivas, o mapeamento UV segue princípios análogos adaptados às suas geometrias específicas. No caso da esfera, cuja superfície curva implica desafios adicionais, implementou-se uma parametrização baseada em coordenadas esféricas, traduzida numa projeção equiretangular. As coordenadas de textura (u, v) são calculadas segundo as expressões:

$$u = 1.0 - \frac{\varphi}{2\pi} \quad e \quad v = 1.0 - \frac{\theta + \frac{\pi}{2}}{\pi}$$

onde φ é o ângulo azimutal, variando entre 0 e 2π, e θ é o ângulo polar, no intervalo $-\frac{\pi}{2}$ a $\frac{\pi}{2}$. Esta parametrização, implementada no código através das variáveis *sliceStep* e *stackStep*, assegura a continuidade da textura, particularmente no meridiano inicial, onde é corrigido o valor de u para coincidir $u = 1.0$ com $u = 0.0$. Também garante que v varie entre 0 (polo sul) e 1 (polo norte), correspondendo às divisões em *stacks*.

A normal associada a cada vértice da esfera é inicialmente calculada como o vetor radial normalizado:

$$\boldsymbol{n} = \frac{\boldsymbol{P}}{|\boldsymbol{P}|}$$

seguido de uma suavização obtida pela média ponderada das normais das faces adjacentes, garantindo transições suaves de iluminação conforme o modelo de Gouraud. Esta suavização implica, contudo, um tratamento cuidadoso dos vértices situados no meridiano, onde as normais e coordenadas UV necessitam de correção explícita para evitar artefactos visuais.

A figura abaixo exemplifica o resultado do mapeamento UV da esfera, evidenciando a distribuição uniforme da textura e as distorções características da projeção equiretangular, especialmente nas regiões polares.

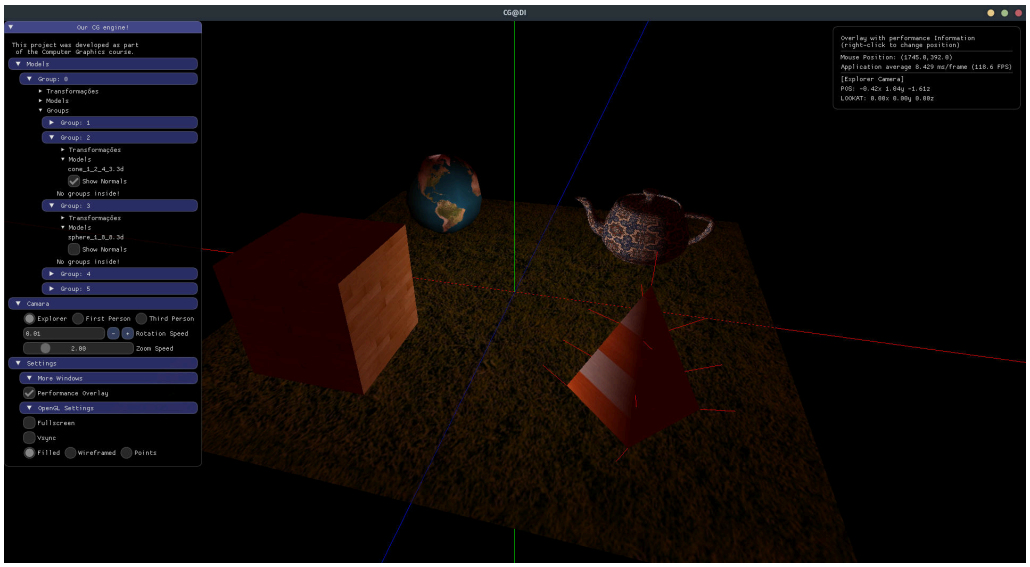

Figura 4: Mapeamento de UVs!

3. Implementação de Iluminação

O sistema de iluminação implementado no motor gráfico baseou-se no modelo de **Gouraud shading**, uma técnica nativa do pipeline gráfico do OpenGL/GLUT, que calcula os efeitos luminosos nos vértices da malha geométrica para posteriormente interpolar linearmente esses valores ao longo das superfícies poligonais. Esta opção arquitetural assentou em quatro fundamentos técnicos essenciais.

Em primeiro lugar, a eficiência computacional foi um fator decisivo, pois o modelo de Gouraud calcula a equação de iluminação apenas nos vértices, ao contrário do modelo de Phong, que o faz por fragmento, reduzindo assim significativamente a carga sobre a GPU. Formalmente, a intensidade da luz num vértice I_v é dada por:

$$I_v = k_a I_a + \sum_{i=1}^{N_{\text{lights}}} (k_d(\boldsymbol{l}_i \cdot \boldsymbol{n}_v) + k_s(\boldsymbol{r}_i \cdot \boldsymbol{v})^{\alpha}) I_i$$

onde k_a, k_d e k_s representam os coeficientes de luz ambiente, difusa e especular respetivamente; I_a é a intensidade da luz ambiente; $\boldsymbol{l}_i$ o vetor direção da luz i; $\boldsymbol{n}_v$ a normal no vértice; $\boldsymbol{r}_i$ o vetor refletido; $\boldsymbol{v}$ o vetor observador e α o coeficiente de brilho ou *shininess*.

Em segundo lugar, o sistema beneficiou da sinergia com a infraestrutura previamente desenvolvida, nomeadamente o cálculo de normais por vértice efetuado nos geradores geométricos, que permitiu a reutilização dos *Vertex Buffer Objects* (VBOs). Por exemplo, no ficheiro `sphereGenerator.cpp`, a suavização das normais foi implementada através da acumulação das normais das faces adjacentes, seguida da normalização final, conforme ilustrado:

O terceiro pilar corresponde ao equilíbrio entre qualidade visual e desempenho. Em malhas com densidade moderada — comuns em contextos educacionais e de prototipagem — o modelo de Gouraud shading proporciona uma suavização eficiente das normais por vértice, com um custo computacional proporcional ao número de vértices, $O(V)$, em contraste com o modelo de Phong cujo custo é aproximadamente proporcional ao número de fragmentos ou polígonos, $O(P)$, sendo usualmente $PggV$.

Por fim, reconhecem-se limitações intencionais ao adotar o modelo de Gouraud, nomeadamente em situações que envolvam malhas de densidade extremamente elevada ou onde sejam críticos efeitos especulares muito precisos, como em materiais metálicos polidos, onde o modelo de Phong seria preferível devido à sua capacidade de calcular a iluminação por fragmento, mesmo que o número de píxeis possa ser inferior ao número de vértices.

3.1. Configuração XML de Iluminação e Materiais

O motor gráfico incorpora um esquema de configuração hierárquica em XML que permite o controlo detalhado das fontes luminosas e das propriedades materiais dos objetos. Esta estrutura segue um paradigma simplificado de Rendering Baseado em Física (PBR), suportando três tipos de luzes: direcional, pontual e **spot lights**, bem como cinco componentes materiais principais.

No que concerne às fontes de luz, o sistema possibilita a inclusão de luzes direcionais, pontuais e focais (**spot lights**), cada uma parametrizável em termos de intensidade, direção e atenuação. Para os materiais, é possível definir as componentes RGB para os valores de luz difusa, ambiente, especular e emissiva, bem como um valor inteiro para o brilho, designado por **shininess**, que influencia o destaque especular dos objetos.

4. Sistema de Texturas

Para o sistema de texturização do motor gráfico, as imagens utilizadas foram armazenadas numa pasta designada por **Texturas**, estando todas no formato JPEG (.jpg). A gestão do carregamento destas imagens e a posterior alocação das mesmas na memória da placa gráfica foram realizadas através da biblioteca DevIL (Developer's Image Library).

Esta biblioteca permite a leitura eficiente e compatível com múltiplos formatos de imagem, facilitando o processamento e upload das texturas para o GPU, garantindo assim uma integração eficaz das texturas nas primitivas geométricas do motor.

5. Demo Scene: Sistema Solar com Iluminação e Texturas

A conceção da cena de demonstração exigiu soluções criativas para integrar de forma coerente os sistemas de iluminação e texturização. O principal desafio residia no Sol, que deveria funcionar simultaneamente como fonte luminosa e como objeto visível na cena. Para solucionar esta dualidade, implementou-se uma esfera com normais invertidas, renderizada como uma superfície emissiva de tonalidade amarelada. Esta estratégia evitou o escurecimento dos planetas quando estes se posicionavam entre o Sol e a câmara, mantendo, simultaneamente, a simplicidade física do modelo.

Os planetas e as suas luas foram texturizados recorrendo a mapas com proporção 2:1, obtidos do Planet Pixel Emporium, tendo sido adaptado o mapeamento UV às características geométricas de cada primitiva. Para as esferas, utilizou-se a projeção equiretangular padrão, enquanto que o anel de Saturno — modelado como um anel (*annulus*) — exigiu ajustes específicos. A textura alongada do anel (915×96 pixels) foi reamostrada através de um fator de correção vertical (0,21) no gerador geométrico, garantindo uma distribuição uniforme sem distorção visível.

Este processo evidenciou a importância do alinhamento entre as coordenadas UV e a topologia geométrica. No caso dos planetas, o mapeamento natural por coordenadas esféricas produziu resultados satisfatórios desde o início. Contudo, para o anel de Saturno, foi necessário redefinir manualmente os parâmetros UV, demonstrando como primitivas não convencionais podem requerer tratamentos personalizados.

Os resultados finais evidenciam um equilíbrio rigoroso entre realismo visual e desempenho. A iluminação segundo o modelo de Gouraud, aliada à suavização das normais, produziu transições graduais de luz e sombra adequadas à escala planetária. As seguintes figuras ilustram varios planetas do sistema solar, apresentando as os efeitos cumulativos das opções técnicas descritas.

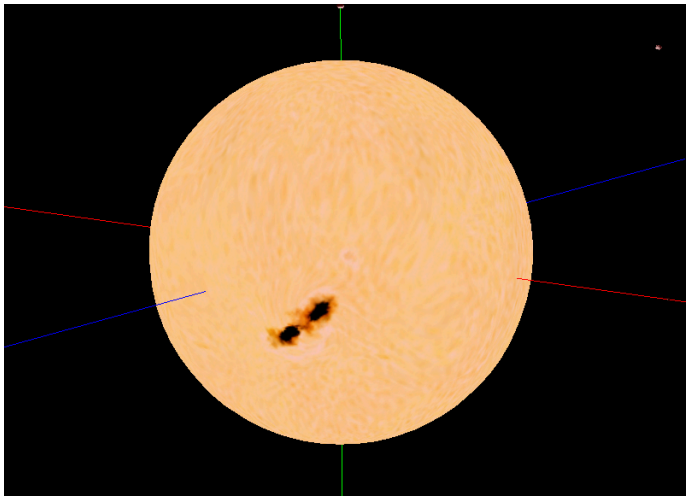

Figura 5: O sol no sistema solar

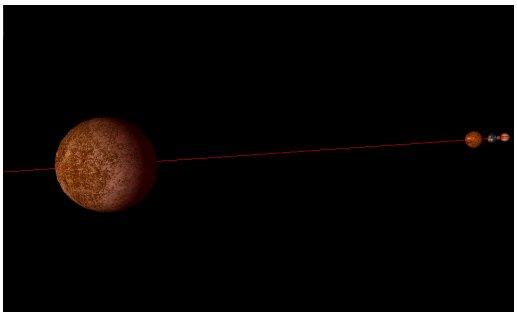

Figura 6: Conjunto 1 de planetas

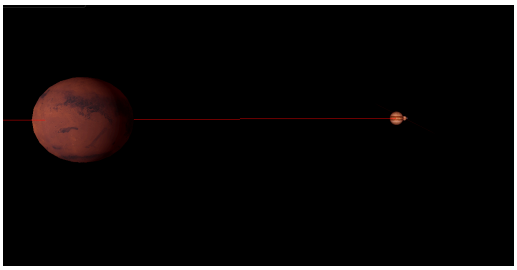

Figura 7: Conjunto 2 de planetas

6. Conclusão e Trabalho Futuro

O desenvolvimento deste motor gráfico cumpriu integralmente os objetivos inicialmente definidos, resultando numa engine funcional capaz de renderizar tanto cenas estáticas como animadas, com suporte completo a iluminação dinâmica e mapeamento de texturas através de coordenadas UV. A demonstração do sistema solar evidencia a eficácia das soluções implementadas, apresentando um equilíbrio satisfatório entre qualidade visual e desempenho.

Para além dos requisitos obrigatórios, foram incorporados diversos elementos adicionais que enriqueceram significativamente o projeto:

- **Primitivas Geométricas Estendidas**: inclusão de torus, anel plano (*flatRing*), cilindro e icosaedro subdividido, com geradores parametrizados suportando cálculo de normais e coordenadas UV;
- **Parser de Ficheiros OBJ**: implementação básica e funcional, com estrutura preparada para futura expansão, incluindo suporte à leitura de normais e UVs;
- **Sistema de Câmara Avançado**: estrutura XML atualizada para suporte a visão em terceira pessoa, embora com limitação atual relacionada com o acesso dinâmico à posição do alvo.

O trabalho futuro deverá incidir sobre três eixos principais:

- **Melhorias Imediatas**: conclusão do parser OBJ para suporte completo a texturas e refinamento do sistema de câmara em terceira pessoa;
- **Otimizações**: implementação de *frustum culling* e desenvolvimento de um sistema adaptativo de *Level of Detail* (LOD).

A arquitetura modular do motor, com clara separação entre componentes geométricos, de renderização e de parsing, assegura que estas expansões possam ser realizadas progressivamente, sem comprometer a estabilidade do sistema existente. O código-fonte encontra-se organizado de modo a facilitar tanto a manutenção contínua como a adição de novas funcionalidades, preservando o equilíbrio entre complexidade técnica e usabilidade pedagógica que caracterizou todo o desenvolvimento.