Documentação Técnica – Neural BVH

1. Visão Geral

O projeto Neural BVH implementa e compara dois métodos de construção de hierarquias de volume delimitador (Bounding Volume Hierarchies — BVH) para aceleração de renderização e detecção de colisões.

- BVH tradicional: baseada em heurísticas geométricas como SAH (Surface Area Heuristic)
- BVH neural: um modelo de rede neural supervisionada é treinado para prever pontos de divisão (splits) ótimos, buscando reduzir sobreposição (EPO) e custo SAH médio.

O pacote inclui geração de cenas sintéticas, treinamento neural rápido, avaliação automática e comparação visual dos resultados.

2. Estrutura de Diretórios

neural_bvh/
 main_demo.py → Script principal (execução completa)
 core/
  geometry.py → Geração de triângulos e primitivas
  bvh.py → Estrutura BVH e construção clássica
  metrics.py → Métricas EPO e SAH
  utils.py → Funções auxiliares
 neural/
  model.py → Definição da rede neural (PyTorch)
  train.py → Treinamento rápido do modelo
  evaluate.py → Avaliação e comparação (EPO/SAH + gráficos)
  dataset.py → Criação de datasets (opcional)
 data/
 scenes/ → Cenas geradas (OBJ)
 models/ → Modelos neurais (.pth)
 results/ → Resultados comparativos
  requirements.txt → Dependências

3. Fluxo de Execução

1. Geração da cena → cria triângulos e salva em data/scenes/
2. Treinamento neural → ajusta pesos do modelo MLP
3. Comparação BVH → mede métricas EPO e SAH
4. Geração de gráficos e relatórios → salva em data/results/

4. Módulos Principais

core/geometry.py → geração de triângulos
core/bvh.py → estrutura BVH tradicional
core/metrics.py → métricas SAH e EPO
neural/model.py → modelo MLP
neural/train.py → treinamento rápido
neural/evaluate.py → comparação e geração de gráficos

5. Execução Passo a Passo

Ativar ambiente virtual
source AmbFernando/bin/activate
Executar o pipeline completo
python main_demo.py
Saída esperada:
data/scenes/scene_toy.obj

data/results/compare_summary.txt
data/results/compare_plot.png
data/results/compare_plot.svg

6. Gráfico Gerado
O gráfico apresenta EPO (barras azuis) e SAH (linha rosa) normalizados.
Valores reais aparecem sobre as barras e marcadores. Ambos variam entre 0–1.

7. Ajustes e Extensões Possíveis
- Aumentar a profundidade da rede neural (mais camadas/neurônios)
- Usar cenas reais (CAD, objetos 3D)
- Adicionar novas métricas (IoU, número de nós)
- Suporte para BVH paralela (GPU)
- Comparar múltiplas cenas automaticamente

8. Boas Práticas
- Fixar semente aleatória (torch.manual_seed, np.random.seed)
- Testar modelos com diferentes escalas de triângulos
- Validar EPO em cenas com sobreposição proposital

9. Referências
Pharr, M., Jakob, W., & Humphreys, G. — Physically Based Rendering (3rd ed.)
Wald, I., et al. — State of the Art in Ray Tracing Animated Scenes (EG, 2003)
NVIDIA Research — Neural Bounding Volume Hierarchies (2022)

10. VS Code
ssh raimundo@10.147.17.35 (SantaMaria)
senha: Ra2023imundo#

Anexo A – Código de Execução
Exemplo de uso em main_demo.py:
```
from core.geometry import random_triangles
from neural.train import train_model
from neural.evaluate import compare
tris = random_triangles(500, scale=0.5)
model = train_model(tris, epochs=6, lr=1e-3)
compare(tris, model)
```