Christian Ximenes, Ruan Pereira, Ericson Teófilo, Jefferson Rodrigues

Trabalho em Grupo 4: Algoritmos de Divisão e Conquista em Aplicações de Jogos

1. Definição do Trabalho

Nesta atividade, a equipe deverá implementar e avaliar dois módulos principais para uso em jogos 2D/3D, baseados em algoritmos de divisão e conquista já estudados:

1. **Detecção de Colisões em Tempo Real**:
   - Implementar o algoritmo **Closest Pair** (divisão e conquista em duas dimensões) para identificar pares de objetos próximos.
   - Adaptar a solução para cenários de até 50–100 *sprites* móveis, exibindo frames por segundo (FPS) médios.
2. **Geração Procedural Simplificada de Terreno**:
   - Usar o algoritmo de **Seleção Linear** (Média de Médianas) para extrair subconjuntos de pontos representativos de uma nuvem inicial de alturas.
   - Aplicar divisão e conquista para subdividir recursivamente a malha de terreno, refinando detalhes em pontos selecionados.

O projeto deve ser implementado preferencialmente em **Python** (por exemplo, com Pygame para visualização 2D) ou em **C++** (usando SDL ou SFML). **Não usar Java.**

2. Conceitos e Fundamentação Teórica

2.1 Closest Pair em 2D

- Problema: dado um conjunto de n pontos no plano, encontrar o par cuja distância Euclidiana é mínima.
- Algoritmo de divisão e conquista:
  1. Ordenar todos os pontos por coordenada x.
  2. Dividir o conjunto em duas metades de tamanho $\lfloor n/2 \rfloor$.
  3. Recursivamente, encontrar o par mínimo em cada metade.

4. Combinar resultados verificando pares que atravessam a linha de divisão (criar região δ-faixa e ordenar por y para comparar apenas até 7 vizinhos).

- Complexidade teórica: $O(n \log n)$.
- Aplicação em jogos: detectar colisões de forma mais rápida do que verificar todos os pares ($O(n^2)$).

2.2 Seleção Linear (Média de Médianas)

- Objetivo: encontrar o k-ésimo menor elemento em tempo linear $O(n)$ no pior caso.
- Procedimento:
  1. Dividir o vetor em grupos de 5 elementos.
  2. Calcular a mediana de cada grupo (ordenando cada grupo de 5).
  3. Recursivamente, encontrar a mediana das medianas (chamado de pivô "médio").
  4. Particionar o vetor em menores e maiores que esse "pivô médio" e recursivamente buscar no lado que contém o k-ésimo elemento.
- Complexidade: $T(n) = T(n/5) + T(7n/10) + O(n) = O(n)$.
- Aplicação em geração procedural: selecionar subconjuntos representativos de pontos para reduzir o número de vértices antes de subdividir recursivamente.

2.3 Divisão e Conquista em Jogos

- Em colisões, a divisão dá suporte a dividir o campo de jogo (pontos) em regiões, filtrando comparações a partir de limites locais.
- Em geração procedural, a seleção linear ajuda a escolher pontos-chave para subdivisão, enquanto divisão e conquista gera níveis de detalhe incrementalmente.
- Esses métodos reduzem o custo computacional em relação a abordagens ingênuas ($O(n^2)$ em colisões e subdivisão exaustiva de terreno).

3. Objetivos Específicos

1. **Implementar o módulo de Detecção de Colisões com Closest Pair:**
   - Carregar posições de até $n = 50$–100 objetos (*sprites*) em um espaço 2D.
   - Executar o algoritmo Closest Pair recursivamente para identificar o par de sprites mais próximos a cada frame.
   - Medir *frames por segundo* (FPS) para cenários de $n = 50$, $n = 75$ e $n = 100$.
   - Comparar com abordagem *brute force* ($O(n^2)$) e reportar FPS para cada método.

2. **Implementar o módulo de Geração Procedural Simplificada:**

   - Gerar uma nuvem inicial de $n = 10^4$ pontos de altura (por exemplo, usando ruído simples).
   - Aplicar Seleção Linear para extrair $m = n/5$ pontos representativos (subconjunto).
   - Em cada subconjunto, usar divisão e conquista para subdividir a malha em quadrantes, refinando até profundidade máxima de 3 níveis.
   - Renderizar o resultado final em 2D (mapa de altura) ou 3D (*wireframe* simples).
   - Medir tempo de geração para cada profundidade (nível 1, 2 e 3) e reportar complexidade prática.

3. **Testar e comparar desempenho:**

   - Para colisões: comparar Closest Pair vs. *brute force* em termos de FPS.
   - Para geração procedural: comparar tempo de geração entre uso de Seleção Linear + D&C vs. subdivisão exaustiva (dividir sem seleção, $O(n \log n)$ vs. $O(n^2)$).

4. **Analisar qualitativamente:**

   - Em quais cenários Closest Pair sustenta FPS adequados (30 FPS) e onde *brute force* falha.
   - Como a Seleção Linear impacta o custo de subdivisão de terreno em comparação à abordagem ingênua.
   - Aderência empírica à complexidade teórica: $O(n \log n)$ vs. $O(n^2)$.

4. Formato do Relatório de Entrega

O relatório final deve ser entregue em **PDF** e conter, no máximo, **8 páginas**, distribuídas da seguinte forma:

1. **Capa:** título, nomes dos integrantes (já apresentados na primeira linha), disciplina e data.

2. **Resumo (abstract):** até 150 palavras, descrevendo a proposta de colisão e geração procedural e principais resultados.

3. **Introdução:** motivação para usar algoritmos de divisão e conquista em jogos, desafios de colisão em tempo real e geração procedural (meia página).

4. **Fundamentação Teórica:**

   - Descrição de Closest Pair em 2D e sua complexidade $O(n \log n)$.
   - Descrição de Seleção Linear (Médiana de Médianas) e subdivisão recursiva do terreno.

- Breve comparação teórica com abordagens ingênuas ($O(n^2)$).

5. **Metodologia:**
   - Ambiente de desenvolvimento (ex.: "Python 3.9 + Pygame em máquina com Intel i5 e 8 GB RAM" ou "C++17 + SDL2 em Linux Ubuntu").
   - Organização do código: arquivos ou módulos para colisão e geração procedural.
   - Descrição de como os pontos e sprites são gerados/armazenados.
   - Métodos de medição de desempenho:
     - FPS (usar `pygame.time.Clock()` ou `std::chrono`).
     - Tempo de geração (tempo de CPU ou `time.time()`).
6. **Resultados e Discussão:**
   - Tabelas ou gráficos mostrando FPS de *Closest Pair* vs. *brute force* para $n = 50, 75, 100$.
   - Gráficos de tempo de geração procedural comparando Seleção Linear + D&C vs. subdivisão ingênua para níveis 1, 2 e 3.
   - Comentários sobre limites práticos (quando FPS cai abaixo de 30 ou tempo de geração torna-se excessivo).
   - Observações sobre conformidade com $O(n \log n)$ vs. $O(n^2)$.
7. **Conclusão:** principais achados, limitações (por exemplo, saturação de CPU ou memória), e sugestões de melhorias (ex.: usar quadtrees ou kd-trees, aplicar subseleção de pontos por clusters).
8. **Referências:** ao menos 3 fontes (artigos ou tutoriais sobre Closest Pair, Seleção Linear, e geração procedural em jogos), formatadas em ABNT ou similar.

5. Apresentação de Resultados

A equipe deverá preparar uma **apresentação oral de 20 minutos** para mostrar:

- Demonstração ao vivo (ou gravação) do módulo de colisões comparando *Closest Pair* e *brute force*.
- Demonstração da geração procedural simplificada com Seleção Linear + divisão e conquista versus subdivisão ingênua.
- Explicação teórica sucinta de cada algoritmo e discussão dos resultados empíricos.

Boa codificação e bons estudos!