FASE 2: SISTEMA BÁSICO DE PARTÍCULAS

CONTEXTO

Continuar desenvolvimento do Simulador de Partículas e Áudio 3D. A FASE 1 (estrutura base + áudio) está funcionando perfeitamente. Agora implementar sistema de partículas reagindo ao áudio.

OBJETIVO DA FASE 2

Criar 500 partículas esféricas coloridas que reagem ao áudio WAV carregado, mantendo 30-60fps estável.

IMPLEMENTAÇÕES NECESSÁRIAS:

1. SISTEMA DE PARTÍCULAS (no script.js)

```javascript
// Adicionar após as variáveis globais existentes:
let particleSystem, particles = [];
let frequencyBands = new Float32Array(8); // 8 bandas de frequência
const PARTICLE_COUNT = 500;
```

Criar classe ParticleSystem:

- Geometry instanciada (InstancedMesh) para performance
- 500 partículas esféricas iniciais
- Cores do arco-íris usando HSL mapping
- Posições aleatórias distribuídas em esfera 3D (raio 10-15)
- Buffer geometry reutilizável para otimização

2. ANÁLISE DE ÁUDIO AVANÇADA

Implementar função `analyzeAudio()`:

- Dividir espectro FFT em 8 bandas de frequência
- Mapear bandas para grupos de partículas (62-63 partículas por banda)
- Calcular amplitude média por banda
- Aplicar smoothing temporal para evitar jittering

3. REAÇÃO VISUAL AO ÁUDIO

Implementar função `updateParticles()`:

- Escala das partículas baseada na amplitude da banda

- Movimento/oscilação proporcional à intensidade
- Cores reagindo à frequência dominante (HSL shift)
- Throttle de update para máximo 30fps

4. CONTROLES DA INTERFACE (adicionar ao HTML)

Adicionar após os controles existentes:

```html
<div id="particleControls">
    <label for="particleCount">Partículas:</label>
    <input type="range" id="particleCount" min="200" max="2000" value="500">
    <span id="particleCountValue">500</span>

    <label for="audioSensitivity">Sensibilidade:</label>
    <input type="range" id="audioSensitivity" min="0.1" max="3.0" step="0.1" value="1.0">
    <span id="sensitivityValue">1.0</span>

    <button id="toggleParticles">Ocultar Partículas</button>
</div>
```

5. OTIMIZAÇÕES OBRIGATÓRIAS

- **InstancedMesh** para renderização eficiente
- **Buffer geometry** reutilizável
- Update apenas partículas visíveis (frustum culling básico)
- **Throttle** análise de áudio para 30fps máximo
- **Object pooling** para gerenciamento de memória

6. ESTRUTURA DE INICIALIZAÇÃO

Modificar função `init()`:

- Remover cubo de teste da FASE 1
- Inicializar sistema de partículas
- Configurar iluminação ambiente suave
- Integrar análise de áudio ao loop de animação

Modificar função `animate()`:

- Adicionar chamada para `analyzeAudio()`
- Adicionar chamada para `updateParticles()`

- Manter 60fps estável

7. MAPEAMENTO DE CORES

Implementar função `getColorFromFrequency(bandIndex, amplitude)`:

- Banda 0-1 (graves): Vermelho/Laranja (HSL: 0-30°)
- Banda 2-3 (médios-graves): Amarelo/Verde (HSL: 60-120°)
- Banda 4-5 (médios-agudos): Azul/Ciano (HSL: 180-240°)
- Banda 6-7 (agudos): Roxo/Magenta (HSL: 270-330°)

REQUISITOS DE PERFORMANCE:

- ✅ 30fps mínimo com 500 partículas
- ✅ Escalável até 2000 partículas
- ✅ Memória estável (sem vazamentos)
- ✅ Análise de áudio em 30fps máximo

TESTE DA FASE 2:

1. Carregar música WAV com ritmo marcado
2. Verificar partículas dançando sincronizadas
3. Testar mudança de cores conforme frequências
4. Confirmar controles funcionando
5. Verificar performance estável (usar stats.js se necessário)

DEBUGGING:

- Console.log do número de partículas ativas
- Console.log das bandas de frequência
- Console.log do FPS médio
- Avisos de performance se FPS < 30

COMPATIBILIDADE:

- Manter funcionalidade da FASE 1 intacta
- Fallback para menos partículas se performance baixa
- Graceful degradation para hardware limitado

IMPLEMENTAR SOMENTE A FASE 2 - NÃO AVANÇAR PARA FASE 3