Guia de Evolução do Projeto

Este documento descreve como evoluir o Motor Procedural Educacional de forma estruturada, alinhado com o ecossistema EZ Studios.

🎯 Visão Geral

O projeto está estruturado para evolução contínua através de **5 versões principais**:

Versão	Foco	Timeline	Status
V1.0	Núcleo + Roblox	✅ Completo	Released
V1.1	Performance + UX	2026-Q1	Planejado
V2.0	Multi-engine	2026-Q2	Planejado
V2.5	Search-based PCG	2026-Q3	Planejado
V3.0	IA Generativa	2026-Q4	Planejado

📊 Próximos Passos Priorizados

🔴 Alta Prioridade (V1.1 - Q1 2026)

1. Interface Web de Geração

Objetivo: Permitir alunos customizar parâmetros visualmente

Esforço: Médio (5 dias)

Impacto: Alto

Dependências: Nenhuma

Tarefas:

TypeScript

```
// client/src/pages/GenerateMap.tsx
- [ ] Criar formulário com campos de intenção
- [ ] Validar entrada do usuário
- [ ] Chamar API educacional
- [ ] Mostrar progresso de geração
- [ ] Exibir mapa gerado
```

```
- [ ] Permitir download de JSON
- [ ] Permitir copiar código Luau
```

Arquivos a Criar:

- `client/src/pages/GenerateMap.tsx`
- `client/src/components/IntentionForm.tsx`
- `client/src/components/MapPreview.tsx`

Exemplo de Uso:

TypeScript

```
// Aluno acessa /generate
// Preenche: quantidadeAreas=5, temBossRoom=true
// Clica "Gerar"
// Vê mapa em tempo real
// Download JSON ou copiar código Luau
```

2. Dashboard de Alunos

Objetivo: Rastrear histórico de mapas gerados

Esforço: Médio (5 dias)

Impacto: Alto

Dependências: Interface Web de Geração

Tarefas:

TypeScript

```
// client/src/pages/StudentDashboard.tsx
- [ ] Listar mapas gerados
- [ ] Mostrar estatísticas (total, tempo médio)
- [ ] Permitir re-gerar com mesma seed
- [ ] Mostrar logs de cada geração
- [ ] Exportar histórico
```

Arquivos a Criar:

- `client/src/pages/StudentDashboard.tsx`
- `client/src/components/MapHistory.tsx`
- `client/src/components/GenerationStats.tsx`

3. Otimização de Performance

Objetivo: Reduzir tempo de geração

Esforço: Alto (8 dias)

Impacto: Médio

Dependências: Nenhuma

Tarefas:

```
TypeScript

// src/core/wfc/wfc-parallel.ts
- [ ] Implementar WFC paralelo com Worker Threads
- [ ] Benchmarking antes/depois
- [ ] Otimizar BSP com memoização
- [ ] Reduzir alocações de memória
- [ ] Adicionar cache de resultados
```

Métrica de Sucesso: Reduzir tempo de 250-600ms para <200ms

🟡 Média Prioridade (V2.0 - Q2 2026)

4. Adaptador Unity/C#

Objetivo: Suportar Unity além de Roblox

Esforço: Alto (10 dias)

Impacto: Alto

Dependências: Núcleo V1.0

Tarefas:

```
TypeScript

// src/adapters/unity/
- [ ] Criar estrutura de adaptador
- [ ] Implementar geração de C# MonoBehaviour
- [ ] Gerar scripts de construção
- [ ] Criar exemplo de integração
- [ ] Documentar em UNITY_INTEGRATION.md
```

Arquivos a Criar:

- `src/adapters/unity/unityAdapter.ts`

- `examples/UnityMapBuilder.cs`
- `examples/UnityExample.unity`
- `UNITY_INTEGRATION.md`

5. Suporte a Godot/GDScript

Objetivo: Suportar Godot além de Roblox/Unity

Esforço: Alto (10 dias)

Impacto: Alto

Dependências: Núcleo V1.0

Tarefas:

```TypeScript
// src/adapters/godot/
- [ ] Criar estrutura de adaptador
- [ ] Implementar geração de GDScript
- [ ] Gerar scripts de construção
- [ ] Criar exemplo de integração
- [ ] Documentar em GODOT_INTEGRATION.md
```

6. Geração em Tempo Real

Objetivo: Gerar mapas dentro do jogo

Esforço: Alto (12 dias)

Impacto: Médio

Dependências: Otimização V1.1

Tarefas:

```TypeScript
// src/core/realtime/
- [ ] Implementar geração incremental
- [ ] Adicionar cancelamento de operações
- [ ] Implementar streaming de tiles
- [ ] Adicionar feedback visual
```

🟢 Baixa Prioridade (V2.5+ - Q3+ 2026)

7. Search-based PCG

Objetivo: Otimizar mapas usando busca

Esforço: Muito Alto (20 dias)

Impacto: Alto

Dependências: Geração em Tempo Real

Tarefas:

TypeScript

```
// src/core/search/
- [ ] Implementar algoritmo de busca (GA, PSO)
- [ ] Definir função de fitness
- [ ] Otimizar para dificuldade
- [ ] Adicionar constraints
```

8. Editor Visual

Objetivo: Node graph para composição

Esforço: Muito Alto (20 dias)

Impacto: Alto

Dependências: Interface Web V1.1

Tarefas:

TypeScript

```
// client/src/pages/Editor.tsx
- [ ] Criar canvas para nodes
- [ ] Implementar node types
- [ ] Adicionar conexões entre nodes
- [ ] Compilar graph para código
```

9. Integração IA Generativa

Objetivo: Usar IA para gerar conteúdo

Esforço: Alto (15 dias)

Impacto: Alto

Dependências: API educacional

Tarefas:

TypeScript

```
// src/ai/
- [ ] Integrar com LLM (GPT, Claude)
- [ ] Gerar descrições naturais
- [ ] Converter descrição em intenção
- [ ] Validar output
```

🔄 Processo de Evolução

1. Planejamento

YAML

```yaml
# Adicionar ao project.json
evolutionGuidance:
  nextSteps:
    - priority: high
      title: "Seu Feature"
      description: "Descrição"
      effort: "medium"
      impact: "high"
      targetVersion: "v1.1"
      estimatedDays: 5
```

2. Implementação

Bash

```bash
# Criar branch
git checkout -b feature/seu-feature

# Implementar
# - Adicionar código em src/
# - Adicionar testes em tests/
# - Atualizar documentação

# Validar
pnpm check
pnpm test

# Commit
```

```
git commit -m "feat: adicionar seu feature"
git push origin feature/seu-feature
```

3. Review

Bash

```
# Abrir Pull Request
# - Descrever mudanças
# - Referenciar issue
# - Pedir review

# Resolver comentários
# Merge quando aprovado
```

4. Release

Bash

```
# Atualizar version em package.json
# Atualizar CHANGELOG
# Criar tag
git tag v1.1.0
git push origin v1.1.0
```

📈 Métricas de Progresso

Acompanhe evolução com estas métricas:

Métrica	V1.0	V1.1	V2.0	V2.5	V3.0
Linhas de Código	2.5K	4K	6K	8K	10K
Módulos	7	10	15	18	20
Engines Suportadas	1	1	3	3	3
Tempo Geração	250-600ms	<200ms	<200ms	<100ms	<100ms

Cobertura Testes	85%	90%	92%	95%	95%
Documentação	5 docs	7 docs	10 docs	12 docs	15 docs

🎓 Papéis e Responsabilidades

Aprendiz

- Usar interface web para gerar mapas
- Explorar exemplos
- Fornecer feedback
- Sugerir melhorias

Ações:

- Acessar `/generate`
- Preencher formulário
- Download/export
- Comentar em issues

Engenheiro

- Implementar features
- Otimizar performance
- Escrever testes
- Revisar código

Ações:

- Criar branches
- Implementar features
- Rodar `pnpm check && pnpm test`
- Abrir PRs

Fundador

- Priorizar features
- Planejar roadmap
- Monetizar conteúdo
- Escalar infraestrutura

Ações:

- Atualizar `project.json`
- Planejar releases
- Integrar marketplace
- Analisar métricas

🔗 Integrações Futuras

Marketplace UGC

TypeScript

```
// Permitir venda de mapas gerados
interface MapaVenda {
  id: string;
  preco: number;
  descricao: string;
  thumbnail: string;
  downloads: number;
}
```

Datastore

TypeScript

```
// Persistir mapas gerados
interface MapaArmazenado {
  id: string;
  autorId: string;
  mapaJson: string;
  criadoEm: Date;
  acessosPublicos: number;
}
```

Roblox Monetization

```TypeScript
// Integrar com sistema de monetização Roblox
interface MonetizacaoConfig {
  premiumFeatures: string[];
  cosmetics: string[];
  analyticsEnabled: boolean;
}
```

📚 Documentação de Evolução

Cada versão deve incluir:

1. **CHANGELOG.md** - O que mudou
2. **MIGRATION.md** - Como migrar de versão anterior
3. **NEW_FEATURES.md** - Features novas
4. **PERFORMANCE.md** - Benchmarks

✅ Checklist de Release

Antes de cada release:

- [ ] Todos os testes passam
- [ ] TypeScript sem erros
- [ ] Documentação atualizada
- [ ] Exemplos funcionam
- [ ] Performance validada
- [ ] Changelog completo
- [ ] Version bumped
- [ ] Tag criada
- [ ] Release notes escritas

🚀 Como Contribuir

1. Escolha um item de `evolutionGuidance` em `project.json`
2. Crie issue descrevendo a feature
3. Crie branch: `git checkout -b feature/seu-feature`
4. Implemente com testes
5. Abra PR com descrição clara
6. Aguarde review e merge

📞 Suporte

- **Dúvidas**: Abra Discussion no GitHub
- **Bugs**: Abra Issue com `bug` label
- **Features**: Abra Issue com `enhancement` label
- **Roadmap**: Consulte `project.json`

Última atualização: 10 de Janeiro de 2026

Versão: 1.0.0

Próxima revisão: Q1 2026