UNIVASSOURAS
ENGENHARIA DE SOFTWARE

LEANDRO LIMA CARDOSO (202323366)

PLANO DE GERENCIAMENTO DO PROJETO
DESENVOLVIMENTO DE *GAME ENGINE* 2D EM C++

PRÁTICAS EXTENSIONISTAS INTEGRADORAS VI

SAQUAREMA
2026

LEANDRO LIMA CARDOSO (202323366)

PLANO DE GERENCIAMENTO DO PROJETO
DESENVOLVIMENTO DE *GAME ENGINE* 2D EM C++

PRÁTICAS EXTENSIONISTAS INTEGRADORAS VI

Trabalho apresentado como requisito parcial para obtenção de nota da disciplina de **Práticas Extensionistas Integradoras VI** do curso de Engenharia de Software da Univassouras (campus universitário de Saquarema).

Prof. Dr. Altemar Sales

SAQUAREMA

2026

RESUMO

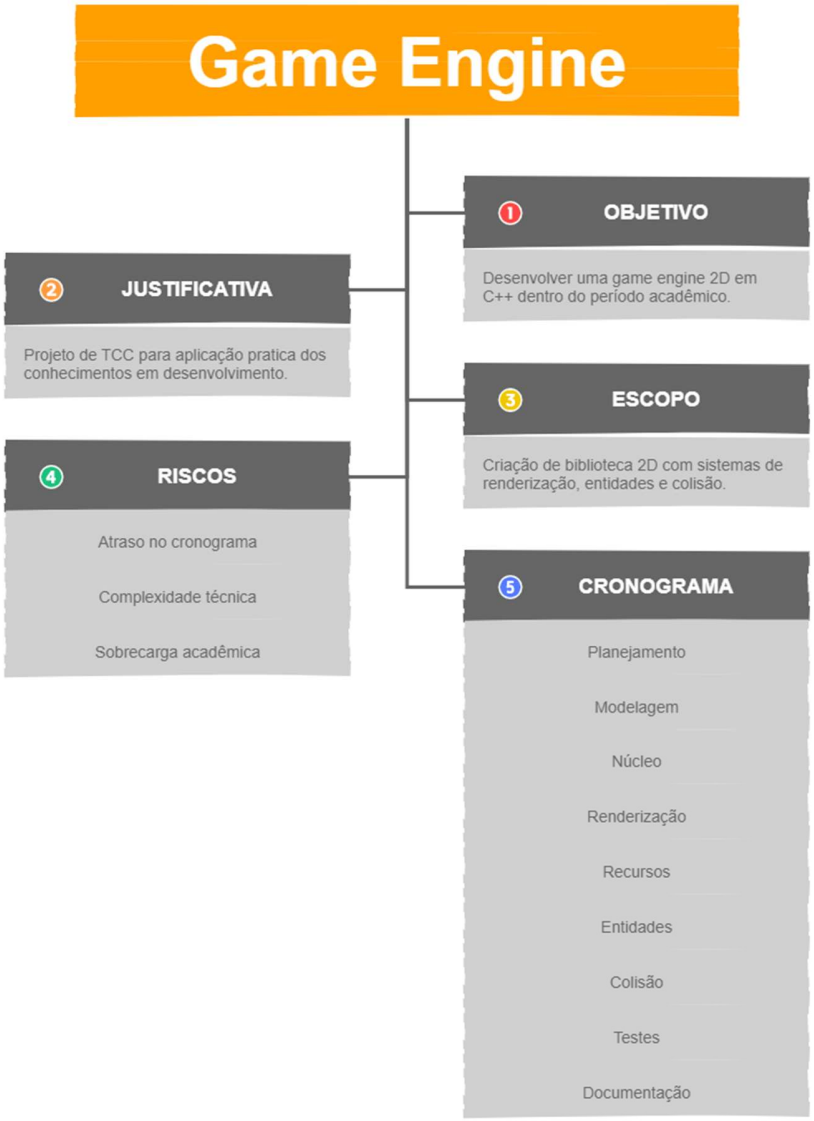
Este documento apresenta o Plano de Gerenciamento do Projeto referente ao desenvolvimento de uma *game engine* 2D em formato de biblioteca, implementada na linguagem C++, com foco em modularidade, desempenho e redução de dependências externas. O plano tem como objetivo estruturar e organizar as atividades necessárias para a concepção, modelagem, implementação, validação e documentação do produto final, garantindo controle sobre escopo, prazo, qualidade e riscos. São definidos os objetivos do projeto em formato SMART, as entregas previstas, a Estrutura Analítica do Projeto (EAP), o cronograma de execução, bem como as estratégias de gerenciamento de comunicação, partes interessadas e monitoramento do trabalho. O documento também contempla análise e tratamento de riscos associados à complexidade técnica e às restrições de tempo inerentes ao desenvolvimento individual. Por meio da aplicação de práticas formais de gerenciamento, busca-se assegurar a execução estruturada do projeto, minimizando incertezas e promovendo organização, rastreabilidade e controle ao longo de todo o ciclo de desenvolvimento da biblioteca.

Palavras-chave: *game engine*; C++; gerenciamento de projetos; planejamento; desenvolvimento de software.

SUMÁRIO

1 PROJECT CANVAS

Fonte: Elaborado pelo autor (2026).



1.1 Descrição do Project Canvas

O Project Canvas apresentado na Figura sintetiza os principais elementos estratégicos do projeto, permitindo uma visão estruturada dos objetivos, entregas, restrições e riscos envolvidos no desenvolvimento da game engine 2D em C++.

1.1.1 Problema

Desenvolvedores iniciantes e projetos acadêmicos que buscam criar jogos 2D baseados em *tiles* frequentemente utilizam motores generalistas que apresentam elevada complexidade estrutural, abstrações amplas e múltiplas dependências externas. Essa característica pode dificultar a compreensão aprofundada dos mecanismos internos de uma *engine* e limitar a experimentação arquitetural voltada ao desempenho e à modularidade.

1.1.2 Solução Proposta

Desenvolver uma game engine 2D estruturada como biblioteca em C++, especializada em jogos baseados em *tiles* com visão *top-down*, priorizando arquitetura modular, controle direto sobre recursos computacionais e redução de dependências externas.

1.1.3 Produto Final

O produto final do projeto consiste em:

- Biblioteca funcional compilável em C++;
- Sistema de renderização 2D baseado em *tiles*;
- Gerenciador de recursos gráficos;
- Sistema de entidades e detecção de colisão;
- Protótipo funcional de validação;
- Documentação técnica e acadêmica.

1.1.4 Público-Alvo

O projeto é direcionado a:

- Estudantes de Engenharia de Software;
- Desenvolvedores iniciantes;
- Projetos acadêmicos que demandem motor gráfico 2D simplificado;
- Desenvolvedores independentes interessados em soluções especializadas.

1.1.5 Objetivos do Projeto

- Desenvolver biblioteca modular e reutilizável;
- Garantir funcionamento consistente do sistema de renderização 2D;
- Implementar gerenciamento eficiente de recursos;
- Validar a *engine* por meio de protótipo funcional;
- Documentar arquitetura e decisões técnicas.

1.1.6 Entregas Principais

As entregas previstas incluem:

- Planejamento formal do projeto;
- Modelagem arquitetural;
- Implementação incremental da biblioteca;
- Protótipo funcional;
- Relatórios de acompanhamento;
- Trabalho acadêmico final.

1.1.7 Premissas

- Desenvolvimento será realizado individualmente;

- A linguagem principal utilizada será C++;
- Projeto será executado dentro do prazo acadêmico estabelecido;
- Ambiente computacional necessário estará disponível.

1.1.8 Restrições

- Tempo limitado ao período letivo;
- Recursos financeiros inexistentes;
- Escopo delimitado a jogos 2D baseados em *tiles*;
- Complexidade técnica inerente ao desenvolvimento de motor gráfico.

1.1.9 Riscos Iniciais

- Possível atraso na implementação devido à complexidade técnica;
- Dificuldade de integração entre módulos;
- Problemas de desempenho inesperados;
- Sobrecarga de atividades acadêmicas paralelas.

2 GERENCIAMENTO DO ESCOPO

O gerenciamento do escopo tem como finalidade definir claramente os limites do projeto, estabelecendo o que será desenvolvido, quais entregas serão realizadas e quais elementos não fazem parte da proposta. Essa definição busca evitar expansão não controlada do projeto (*scope creep*), assegurando foco nos objetivos previamente estabelecidos.

2.1 Objetivos SMART

Os objetivos do projeto foram definidos segundo o modelo SMART, garantindo que sejam específicos, mensuráveis, alcançáveis, relevantes e temporais.

Objetivo Geral SMART

Desenvolver, até o final do período letivo de 2027-1, uma *game engine* 2D em formato de biblioteca, utilizando a linguagem C++, contendo sistema de renderização baseado em *tiles*, gerenciamento de recursos, sistema de entidades e mecanismo básico de colisão, validada por meio de protótipo funcional.

Especificidade (S – *Specific*)

O projeto consiste na criação de uma biblioteca especializada para jogos 2D com visão *top-down*, não abrangendo funcionalidades de motores generalistas completos.

Mensurabilidade (M – *Measurable*)

O sucesso do projeto será verificado por meio de:

- Compilação funcional da biblioteca;
- Execução do protótipo com renderização correta de *tiles*;
- Funcionamento consistente do sistema de entidades e colisão;
- Entrega da documentação técnica e acadêmica.

Atingibilidade (A – *Achievable*)

O projeto é viável considerando:

- Desenvolvimento individual;

- Escopo delimitado;
- Uso de ferramentas já dominadas pelo autor;
- Planejamento incremental das atividades.

Relevância (R – *Relevant*)

O projeto é relevante por:

- Aplicar conceitos de engenharia de software;
- Permitir aprofundamento em arquitetura e desempenho;
- Contribuir para formação acadêmica e técnica do autor.

Temporalidade (T – Time-bound)

O projeto será concluído dentro do cronograma acadêmico estabelecido para o semestre 2027-1.

2.2 Produto e Entregas

Produto Final

Biblioteca de game *engine* 2D implementada em C++, estruturada modularmente e validada por protótipo funcional.

Entregas Principais

1. Plano de Gerenciamento do Projeto.
2. Modelagem arquitetural da *engine*.
3. Implementação do núcleo da biblioteca.
4. Sistema de renderização baseado em *tiles*.
5. Sistema de gerenciamento de recursos.

6. Sistema de entidades e colisão.
7. Protótipo funcional de validação.
8. Documentação técnica.
9. Trabalho acadêmico final.

2.3 Matriz de Rastreabilidade dos Requisitos

A matriz de rastreabilidade tem como objetivo garantir que cada requisito identificado esteja vinculado a uma entrega específica do projeto, permitindo controle e verificação de conformidade.

ID	Requisito	Tipo	Entrega Relacionada	Critério de Validação
RF01	Renderizar mapa baseado em *tiles*	Funcional	Renderização	Execução correta no protótipo
RF02	Gerenciar carregamento de texturas	Funcional	*Resource Manager*	Carregamento sem duplicação
RF03	Criar e atualizar entidades	Funcional	*Entity System*	Atualização correta em execução
RF04	Detectar colisão com mapa	Funcional	*Collision System*	Bloqueio em áreas não transitáveis
RNF01	Manter arquitetura modular	Não funcional	Estrutura da biblioteca	Separação clara de módulos
RNF02	Minimizar dependências externas	Não funcional	Implementação geral	Uso restrito a bibliotecas essenciais

2.4 Premissas

- O desenvolvimento será realizado individualmente;
- O ambiente computacional será suficiente para execução do projeto;
- O autor possui conhecimento prévio em C++;
- O prazo acadêmico será mantido conforme calendário institucional.

2.5 Restrições

- Prazo limitado ao semestre letivo;
- Escopo restrito a jogos 2D baseados em *tiles*;
- Recursos financeiros inexistentes;
- Desenvolvimento sem equipe auxiliar.

2.6 Estrutura Analítica do Projeto (EAP)

A Estrutura Analítica do Projeto organiza o trabalho em níveis hierárquicos, facilitando planejamento e controle.

Nível 1 – Projeto *Game Engine* 2D

1. Planejamento
2. Modelagem Arquitetural
3. Implementação
   - 3.1 Núcleo (*Core*)
   - 3.2 Renderização 2D
   - 3.3 Gerenciamento de Recursos
   - 3.4 Sistema de Entidades
   - 3.5 Sistema de Colisão
4. Testes e Validação
5. Documentação
6. Entrega Final

2.7 Principais Atividades e Estratégia do Projeto

O projeto será conduzido por meio de desenvolvimento incremental, priorizando inicialmente a estrutura básica da biblioteca e posteriormente adicionando funcionalidades específicas.

A estratégia adotada inclui:

- Definição arquitetural prévia;
- Implementação modular;
- Validação progressiva por protótipo;
- Revisões periódicas de progresso;
- Documentação paralela ao desenvolvimento.

Essa abordagem busca reduzir riscos técnicos e permitir ajustes estruturais durante a execução do projeto.

3 GERENCIAMENTO DO TEMPO

O gerenciamento do tempo tem como objetivo planejar, estimar, organizar e controlar a duração das atividades do projeto, assegurando que a *game engine* 2D seja desenvolvida dentro do prazo acadêmico estabelecido para o semestre 2027-1.

O planejamento temporal foi estruturado considerando:

- Complexidade técnica das etapas;
- Desenvolvimento individual;
- Necessidade de paralelismo entre implementação e documentação;
- Períodos de revisão e ajustes.

3.1 Definição das Atividades

As atividades do projeto foram estruturadas a partir da EAP apresentada anteriormente.

Fase 1 – Planejamento

- Elaboração do Plano de Gerenciamento
- Definição da arquitetura inicial
- Levantamento de requisitos

Fase 2 – Modelagem Arquitetural

- Modelagem UML (classes e módulos)
- Definição de responsabilidades
- Estruturação dos pacotes da biblioteca

Fase 3 – Implementação do Núcleo

- Estrutura base do projeto
- *Loop* principal
- Sistema de inicialização

Fase 4 – Renderização 2D

- Sistema de *tiles*
- Carregamento de *sprites*
- Renderização em tela

Fase 5 – Gerenciamento de Recursos

- Implementação do *Resource Manager*
- Controle de carregamento e descarregamento
- Otimização de memória

Fase 6 – Sistema de Entidades

- Criação da classe base *Entity*
- Sistema de atualização
- Gerenciamento de múltiplas entidades

Fase 7 – Sistema de Colisão

- Implementação de *bounding box*
- Verificação de colisão com mapa
- Integração com entidades

Fase 8 – Testes e Validação

- Criação do protótipo funcional
- Testes de desempenho
- Correção de inconsistências

Fase 9 – Documentação Final

- Revisão do TCC
- Consolidação de diagramas
- Preparação para entrega

3.2 Sequenciamento das Atividades

As atividades seguem ordem lógica de dependência técnica:

1. Planejamento
2. Modelagem
3. Implementação do Núcleo

4. Renderização
5. Recursos
6. Entidades
7. Colisão
8. Testes
9. Documentação Final

A implementação ocorre de forma incremental, permitindo validação contínua.

3.3 Estimativa de Duração

A estimativa foi realizada considerando semanas acadêmicas.

Fase	Atividade	Duração
1	Planejamento	2 semanas
2	Modelagem	2 semanas
3	Núcleo	3 semanas
4	Renderização	3 semanas
5	Recursos	2 semanas
6	Entidades	2 semanas
7	Colisão	2 semanas
8	Testes	2 semanas
9	Documentação	2 semanas

Duração total estimada: aproximadamente 20 semanas.

3.4 Marcos do Projeto

Os marcos representam pontos críticos de verificação:

- M1 – Aprovação do Plano de Gerenciamento
- M2 – Conclusão da Modelagem Arquitetural
- M3 – Renderização funcional de mapa
- M4 – Sistema de entidades operacional
- M5 – Protótipo completo executando
- M6 – Entrega do TCC

3.5 Cronograma (Modelo para Gráfico de Gantt)

Fonte: Elaborado pelo autor (2026).

Atividade	Semanas																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Planejamento																				
Modelagem																				
Núcleo																				
Renderização																				
Recursos																				
Entidades																				
Colisão																				
Testes																				
Documentação																				

3.6 Controle do Cronograma

O acompanhamento será realizado por meio de:

- Revisões semanais de progresso;
- Comparação entre prazo planejado e prazo real;
- Ajustes pontuais de prioridade quando necessário;
- Registro de atrasos e suas causas.

Caso haja risco de atraso, será adotada reordenação de atividades ou simplificação de funcionalidades não críticas.

4 EQUIPE

O gerenciamento da equipe tem como objetivo definir responsabilidades, papéis, formas de comunicação e mecanismos de acompanhamento das atividades do projeto. Embora o desenvolvimento da *game engine* 2D seja realizado individualmente, a formalização da estrutura organizacional contribui para maior clareza na execução das tarefas e na prestação de contas acadêmica.

4.1 Estrutura Organizacional do Projeto

O projeto possui estrutura organizacional simplificada, caracterizada por modelo funcional individual, no qual o autor desempenha múltiplos papéis técnicos e gerenciais.

Papéis Identificados

Papel	Responsável	Responsabilidades
Gerente do Projeto	Autor do projeto	Planejamento, controle de escopo, prazo e riscos
Arquiteto de Software	Autor do projeto	Definição da arquitetura e modularização

Desenvolvedor	Autor do projeto	Implementação da biblioteca
Analista de Testes	Autor do projeto	Validação e verificação funcional
Documentador Técnico	Autor do projeto	Produção da documentação acadêmica
Orientador Acadêmico	Professor da disciplina	Acompanhamento, validação metodológica e avaliação

4.2 Responsabilidades

Autor do Projeto

- Elaborar o planejamento formal;
- Desenvolver a arquitetura da *engine*;
- Implementar os módulos previstos;
- Realizar testes funcionais;
- Produzir documentação técnica e acadêmica;
- Monitorar prazos e riscos.

Orientador Acadêmico

- Avaliar coerência metodológica;
- Validar entregas parciais;
- Fornecer direcionamento acadêmico;
- Avaliar o resultado final.

4.3 Matriz de Responsabilidades (RACI Simplificada)

Atividade	Autor do projeto	Orientador Acadêmico
Planejamento	R	A
Modelagem	R	C

Implementação	R	I
Testes	R	I
Documentação	R	C
Aprovação Final	-	A

Legenda:
R – Responsável
A – Aprovador
C – Consultado
I – Informado

4.4 Competências Necessárias

Para execução do projeto são requeridas as seguintes competências:

- Programação avançada em C++;
- Conhecimento em arquitetura de software;
- Modelagem UML;
- Noções de desempenho e gerenciamento de memória;
- Planejamento e organização de projetos;
- Escrita acadêmica conforme normas técnicas.

4.5 Estratégia de Desenvolvimento Individual

Considerando que o projeto é desenvolvido individualmente, será adotada estratégia de organização baseada em:

- Planejamento semanal de tarefas;
- Registro de progresso;

- Revisões periódicas de arquitetura;
- Desenvolvimento incremental;
- Testes contínuos após cada módulo implementado.

Essa abordagem busca minimizar riscos relacionados à sobrecarga de atividades e atrasos no cronograma.

4.6 Plano de Comunicação Interna

Mesmo sendo projeto individual, o controle será realizado por meio de:

- Registro de decisões técnicas;
- Organização de versões do código;
- Atualização periódica do documento acadêmico;
- Reuniões de acompanhamento com orientador.

5 GERENCIAMENTO DAS COMUNICAÇÕES

O gerenciamento das comunicações tem como finalidade assegurar que todas as informações relevantes do projeto sejam geradas, registradas, armazenadas e transmitidas de maneira adequada às partes interessadas, especialmente ao orientador acadêmico.

Este plano estabelece os canais, frequência e responsabilidades relacionados à comunicação durante o desenvolvimento da *game engine* 2D.

5.1 Objetivos da Comunicação

- Garantir alinhamento entre autor e orientador;
- Registrar decisões técnicas relevantes;
- Documentar progresso do projeto;
- Facilitar acompanhamento acadêmico;
- Manter histórico das versões e alterações realizadas.

5.2 Partes Interessadas na Comunicação

Parte Interessada	Interesse	Necessidade de Informação
Autor do projeto	Execução do projeto	Planejamento, progresso, riscos
Orientador acadêmico	Acompanhamento acadêmico	Status, entregas parciais, dificuldades
Instituição	Avaliação formal	Entrega final e documentação

5.3 Canais de Comunicação

Os seguintes canais serão utilizados:

- Reuniões presenciais ou virtuais com o orientador;
- Comunicação por e-mail institucional;
- Registro escrito no próprio documento do projeto;
- Controle de versões do código-fonte;
- Armazenamento digital organizado.

5.4 Frequência das Comunicações

Tipo de Comunicação	Frequência	Responsável
Reunião de acompanhamento	Quinzenal	Autor e Orientador

Atualização de progresso	Semanal	Autor do projeto
Entrega parcial de documentos	Conforme cronograma	Autor do projeto
Feedback técnico/metodológico	Conforme necessidade	Orientador Acadêmico

5.5 Registro e Armazenamento de Informações

Serão adotadas as seguintes práticas:

- Versionamento organizado do código-fonte;
- Armazenamento digital dos documentos em pasta estruturada;
- Registro de alterações significativas no projeto;
- *Backup* periódico dos arquivos.

Essa organização busca garantir segurança das informações e rastreabilidade das decisões técnicas.

5.6 Tipos de Informação Comunicada

Durante o projeto, serão comunicadas:

- Alterações no escopo;
- Ajustes no cronograma;
- Identificação de riscos;
- Dificuldades técnicas;
- Conclusão de marcos importantes.

5.7 Fluxo de Comunicação

O fluxo comunicacional do projeto segue estrutura simples:

1. Autor do projeto
2. Registro formal
3. Orientador acadêmico
4. *Feedback*
5. Ajustes no projeto

Esse fluxo assegura que decisões relevantes sejam validadas e documentadas.

6 GERENCIAMENTO DAS PARTES INTERESSADAS

O gerenciamento das partes interessadas tem como objetivo identificar indivíduos ou grupos que possam influenciar ou ser impactados pelo projeto, analisando seu nível de interesse e influência, bem como definindo estratégias adequadas de engajamento.

No contexto deste projeto, as partes interessadas estão relacionadas principalmente ao ambiente acadêmico e ao próprio autor.

6.1 Identificação das Partes Interessadas

As principais partes interessadas do projeto são:

- Autor do projeto
- Orientador acadêmico
- Instituição de ensino
- Banca avaliadora
- Comunidade acadêmica e desenvolvedores interessados

6.2 Análise das Partes Interessadas

A análise considera dois fatores principais: nível de interesse e nível de influência.

Parte Interessada	Interesse	Influência	Classificação
Autor do projeto	Muito alto	Alto	Principal interessado
Orientador acadêmico	Alto	Alto	Influenciador direto
Instituição	Médio	Alto	Regulador formal
Banca avaliadora	Alto	Alto	Avaliador final
Comunidade técnica	Médio	Baixo	Beneficiário indireto

6.3 Matriz Interesse x Influência

A partir da análise, estabelece-se a seguinte estratégia:

- Alta influência / Alto interesse → Gerenciar de perto
  - Orientador acadêmico
  - Banca avaliadora
- Alta influência / Médio interesse → Manter satisfeito
  - Instituição
- Alto interesse / Alta responsabilidade → Controle direto
  - Autor do projeto
- Baixa influência / Médio interesse → Monitorar
  - Comunidade técnica

6.4 Estratégias de Engajamento

Autor do projeto

- Planejamento estruturado;
- Registro contínuo de progresso;
- Controle disciplinado do cronograma.

Orientador acadêmico

- Reuniões periódicas;
- Entregas parciais para validação;
- Ajustes conforme orientação metodológica.

Instituição

- Atendimento às normas acadêmicas;
- Cumprimento dos prazos estabelecidos.

Banca Avaliadora

- Produção de documento consistente;
- Clareza técnica na defesa do projeto.

Comunidade Técnica

- Possível disponibilização futura da biblioteca;
- Documentação clara e organizada.

6.5 Monitoramento das Partes Interessadas

O acompanhamento será realizado por meio de:

- *Feedback* do orientador;
- Cumprimento dos marcos acadêmicos;
- Avaliação contínua do alinhamento do projeto com os objetivos institucionais.

Caso haja alteração no nível de interesse ou influência de alguma parte interessada, o plano poderá ser ajustado.

7 GERENCIAMENTO DOS RISCOS

O gerenciamento dos riscos tem como objetivo identificar, analisar e propor respostas para eventos que possam impactar negativamente o desempenho do projeto, seja em termos de prazo, escopo ou qualidade.

Considerando que o desenvolvimento da *game engine* 2D é realizado individualmente e envolve complexidade técnica relevante, a gestão de riscos torna-se fundamental para garantir a viabilidade do projeto.

7.1 Identificação dos Riscos

Os principais riscos identificados para o projeto são:

- Complexidade técnica superior à estimada;
- Atraso no cronograma;
- Problemas de desempenho da biblioteca;
- Dificuldades na integração entre módulos;
- Sobrecarga acadêmica;
- Perda de dados ou falhas no ambiente de desenvolvimento.

7.2 Análise Qualitativa dos Riscos

A análise qualitativa considera probabilidade e impacto.

ID	Risco	Probabilidade	Impacto	Nível
R1	Complexidade técnica elevada	Média	Alta	Alto
R2	Atraso no cronograma	Média	Alta	Alto
R3	Baixo desempenho da *engine*	Baixa	Alta	Média
R4	Falha na integração de módulos	Média	Média	Média
R5	Sobrecarga acadêmica	Alta	Média	Alto
R6	Perda de arquivos	Baixa	Alta	Média

7.3 Estratégias de Resposta aos Riscos

R1 – Complexidade Técnica Elevada

Estratégia: Mitigação

- Dividir implementação em etapas menores;
- Validar cada módulo isoladamente;
- Realizar testes incrementais.

R2 – Atraso no Cronograma

Estratégia: Mitigação

- Planejamento semanal;
- Priorização de funcionalidades essenciais;
- Redução de funcionalidades não críticas, se necessário.

R3 – Baixo Desempenho

Estratégia: Mitigação

- Testes de desempenho periódicos;
- Revisão de código e otimização de memória.

R4 – Falha na Integração
Estratégia: Mitigação

- Definição clara de *interfaces*;
- Testes de integração progressivos.

R5 – Sobrecarga Acadêmica
Estratégia: Aceitação controlada com monitoramento

- Organização antecipada de tarefas;
- Ajustes no cronograma quando necessário.

R6 – Perda de Arquivos
Estratégia: Prevenção

- Backup periódico;
- Versionamento do código.

7.4 Plano de Contingência

Caso ocorra materialização de riscos críticos:

- Redução do escopo não essencial;
- Reorganização do cronograma;
- Simplificação de funcionalidades avançadas;

- Priorização da estabilidade do núcleo da biblioteca.

7.5 Monitoramento dos Riscos

O acompanhamento será realizado por meio de:

- Revisão semanal do progresso;
- Atualização do status dos riscos identificados;
- Registro de novos riscos que surgirem durante o desenvolvimento.

O plano de riscos poderá ser revisado sempre que houver alteração significativa no escopo ou cronograma.

8 MONITORAMENTO E CONTROLE

O monitoramento e controle do projeto têm como objetivo acompanhar o desempenho das atividades planejadas, verificando desvios em relação ao escopo, prazo e qualidade, e promovendo ajustes quando necessário.

No contexto do desenvolvimento da *game engine* 2D em C++, o controle será realizado de forma contínua, com foco na disciplina organizacional e na validação incremental das entregas.

8.1 Controle do Escopo

O controle do escopo será realizado por meio de:

- Verificação periódica dos requisitos definidos;

- Conferência entre funcionalidades implementadas e entregas previstas;
- Registro formal de qualquer alteração no escopo;
- Avaliação de impacto antes da inclusão de novas funcionalidades.

Qualquer solicitação de modificação será analisada quanto a:

- Impacto no prazo;
- Impacto na complexidade técnica;
- Necessidade real para os objetivos do projeto.

8.2 Controle do Cronograma

O acompanhamento do prazo será realizado por meio de:

- Revisão semanal das atividades planejadas;
- Comparação entre duração estimada e duração real;
- Identificação antecipada de atrasos;
- Replanejamento quando necessário.

Em caso de desvio significativo, poderão ser adotadas medidas como:

- Priorização de funcionalidades essenciais;
- Simplificação de módulos secundários;
- Ajuste na sequência das atividades.

8.3 Controle da Qualidade

A qualidade do projeto será monitorada por meio de:

- Testes funcionais após implementação de cada módulo;

- Execução do protótipo de validação;
- Revisão de código;
- Verificação da aderência à arquitetura definida;
- Conferência da documentação técnica.

Critérios de qualidade incluem:

- Funcionamento correto do sistema de renderização;
- Estabilidade da execução;
- Organização modular do código;
- Clareza da documentação.

8.4 Indicadores de Desempenho

Serão utilizados os seguintes indicadores:

- Percentual de atividades concluídas no prazo;
- Número de funcionalidades implementadas conforme requisitos;
- Número de erros identificados durante testes;
- Cumprimento dos marcos do projeto.

8.5 Relatórios de Acompanhamento

O acompanhamento do projeto será documentado por meio de:

- Atualizações semanais de progresso;
- Registro de dificuldades técnicas;
- Atualização da matriz de riscos;
- Consolidação de marcos atingidos.

Esses registros servirão como base para análise do desempenho do projeto ao final do período acadêmico.

8.6 Processo de Ajustes

Caso sejam identificados desvios relevantes:

- problema será analisado;
- Será avaliado o impacto no escopo e no prazo;
- Definir-se-á uma ação corretiva;
- ajuste será registrado formalmente.

Essa sistemática garante controle estruturado e reduz a probabilidade de decisões improvisadas.

9 CONTROLE INTEGRADO DE MUDANÇAS

O controle integrado de mudanças tem como finalidade assegurar que qualquer alteração no escopo, cronograma, requisitos ou arquitetura do projeto seja analisada de forma estruturada antes de sua implementação.

Considerando que o desenvolvimento da *game engine* 2D envolve decisões técnicas contínuas, este processo busca evitar modificações impulsivas que possam comprometer prazo, estabilidade ou coerência arquitetural.

9.1 Objetivo do Controle de Mudanças

- Garantir que alterações sejam justificadas;

- Avaliar impacto técnico e temporal;
- Manter rastreabilidade das decisões;
- Preservar alinhamento com os objetivos do projeto.

9.2 Tipos de Mudanças Possíveis

As mudanças podem envolver:

- Inclusão de novas funcionalidades;
- Remoção de funcionalidades previstas;
- Alterações na arquitetura;
- Ajustes no cronograma;
- Modificações nos requisitos funcionais ou não funcionais.

9.3 Processo de Solicitação de Mudança

Toda mudança deverá seguir as seguintes etapas:

1. Identificação da necessidade de alteração;
2. Registro da solicitação de mudança;
3. Análise de impacto (escopo, prazo e risco);
4. Decisão de aprovação ou rejeição;
5. Atualização formal do plano e documentação.

9.4 Critérios para Aprovação de Mudanças

Uma mudança poderá ser aprovada quando:

- Contribuir diretamente para o objetivo principal do projeto;

- Não comprometer o prazo final;
- Não aumentar significativamente o risco;
- For tecnicamente viável dentro do tempo disponível.

Mudanças que representem ampliação significativa de escopo poderão ser adiadas ou descartadas.

9.5 Registro das Mudanças

Será mantido um controle simplificado de mudanças conforme o modelo:

ID	Descrição da Mudança	Justificativa	Impacto	Decisão
M01	Exemplo: otimização do sistema de tiles	Melhorar desempenho	Médio	Aprovada

Esse registro garantirá rastreabilidade das decisões técnicas tomadas durante o desenvolvimento.

9.6 Autoridade de Aprovação

Como o projeto é individual, o autor atua como responsável pela análise técnica das mudanças. Entretanto, alterações que impactem o escopo acadêmico deverão ser discutidas com o orientador antes da implementação.

9.7 Controle de Versões

O controle de mudanças será complementado por:

- Versionamento organizado do código;
- Identificação clara das versões da biblioteca;

- Registro das principais alterações realizadas em cada etapa.

Com isso, o projeto passa a ter um mecanismo formal de governança técnica, reduzindo riscos de desorganização ou expansão não planejada do escopo.

10 LIÇÕES APRENDIDAS

Durante o planejamento e desenvolvimento do projeto de criação da *game engine* 2D em C++, foram identificadas as seguintes lições aprendidas:

1. A definição clara e detalhada do escopo desde o início do projeto reduz significativamente o risco de expansão não controlada de funcionalidades (scope creep).

2. A elaboração prévia da Estrutura Analítica do Projeto (EAP) facilita a visualização das etapas técnicas e melhora a organização do trabalho.

3. A divisão do desenvolvimento em fases incrementais contribui para redução da complexidade e facilita a validação progressiva dos módulos implementados.

4. A estimativa inicial de prazos tende a subdimensionar a complexidade envolvida no desenvolvimento de arquitetura de software, exigindo margem de segurança no cronograma.

5. acompanhamento sistemático e semanal das atividades é essencial para evitar acúmulo de atrasos.

6. A priorização de funcionalidades essenciais aumenta a probabilidade de cumprimento do prazo acadêmico.

7. A definição arquitetural antes da implementação reduz retrabalho e inconsistências estruturais.

8. A modularização adequada do código facilita testes, manutenção e integração entre componentes.

9. A realização de testes incrementais após cada módulo implementado previne a propagação de erros estruturais.

10. A redução de dependências externas amplia o controle técnico sobre a aplicação, mas eleva o nível de responsabilidade e complexidade do desenvolvimento.

11. A antecipação e análise formal de riscos contribuem para tomada de decisão mais estratégica e menos reativa.

12. registro estruturado das mudanças melhora a rastreabilidade e evita decisões impulsivas.

13. A organização documental paralela ao desenvolvimento técnico reduz a sobrecarga ao final do projeto.

14. Projetos individuais demandam elevado nível de disciplina, organização e autogerenciamento.

15. A aplicação de práticas formais de gerenciamento de projetos agrega maturidade técnica e organizacional ao desenvolvimento de software.