1. Introdução

Computação gráfica na ciência da computação é a área que lida com a geração, manipulação e exibição de imagens digitais. Envolve o uso de algoritmos e técnicas para criar gráficos 2D e 3D, animações, simulações e efeitos visuais. É uma área de estudo muito ampla, as pessoas sempre querem ver nas suas telas as imagens com as melhores qualidades, mais próximas do que elas veem na vida real e isso é um enorme desafio pois é preciso adaptar o mundo analógico à uma tela digital.

Ao longo da disciplina aprendemos conceitos, algoritmos e modelos matemáticos que são usados no processo de formação de imagem. Foi abordado conceitos como representação de pontos e vetores, sistemas de coordenadas, algoritmos de rasterização, transformações geométricas, recorte de planos, otimização na renderização de superfícies visíveis, cores texturas e muitas outras ferramentas com abordagem teórica e prática.

Durante a disciplina tivemos várias abordagens práticas dos conceitos aprendidos utilizando o OpenGL, que é uma biblioteca que permite o desenvolvedor construir interfaces gráficas de alta performance, o OpenGL é uma biblioteca de certa forma primitiva, que não te entrega telas prontas, gráficos pré-construídos, mas sim ferramentas mais cruas como linhas e formas, luzes cores e texturas, por isso é ideal para compreender como um gráfico é renderizado partindo do básico até chegar no resultado final. Mas em nenhuma dessas atividades tivemos a experiência de construir uma aplicação completa usando a renderização gráfica.

Portanto, nosso objetivo aqui é desenvolver um jogo que coloque os conhecimentos que adquirimos em prática. Este documento vai abordar os requisitos de desenvolvimento, funcionais e não funcionais, para deixar nosso escopo de trabalho bem definido agilizando o processo. Técnicas de codificação e tecnologias utilizadas, o que esperamos como resultado desse processo e por fim sintetizar o que podemos aprender e concluir com o desenvolvimento.

2. Requisitos

Nesta seção vamos especificar os requisitos do sistema, explicando como será a mecânica do jogo, funcionamento detalhado de cada funcionalidade e os requisitos não funcionais, como hardware e práticas que queremos que sejam abordadas no desenvolvimento.

2.1 Contextualização

Nós vamos desenvolver um jogo 2D onde o jogador controla uma nave/avião armado voando por um ambiente com inimigos, o jogador poderá direcionar a arma, atirar a acumular pontos eliminando os inimigos.

2.2 Identificação dos Requisitos

Requisitos funcionais, identificados por RF: Identificam os requisitos de funcionalidades do jogo.

Requisitos não funcionais, identificados por RNF: Identificam os requisitos não ligados diretamente a funcionalidade do jogo, mas do ambiente e do processo de desenvolvimento.

2.3 Requisitos Funcionais

RF001 - Menu

A primeira tela do jogo deve ser uma tela de menu com interação através do mouse, nesse menu o jogador terá as opções “Jogar”, “Instruções”, “Sair” e “Créditos”. Nessa tela de menu não deverá ter nenhuma outra interação além dos botões de opção.

Prioridade: Alta.

RF002 - Opção Sair

Opção acessada através do menu inicial deve fechar o jogo.

Prioridade: Alta.

RF003 - Opção Instruções

Ao acessar a opção instruções o jogo deve apresentar ao jogador uma tela com instruções de jogabilidade, indicando quais teclas controlam a nave, e como ativar as funções dela.

RF004 - Opção Créditos

Ao acessar a opção créditos pelo menu inicial o jogo deve apresentar ao jogador uma tela de créditos contendo o nome dos integrantes desenvolvedores e uma breve descrição do contexto em que foi desenvolvido descrevendo que foi um trabalho acadêmico.

Prioridade: Alta.

RF005 - Opção Jogar

Ao acessar no menu a opção jogar o jogador será redirecionado para a tela principal com ambiente do jogo e o jogo irá começar imediatamente quando carregar a tela.

Prioridade: Alta.

RF006 - Inicialização do Jogo

Ao iniciar a tela de jogo os objetos e texturas serão imediatamente carregados, com o objeto do jogador principal na parte inferior da tela centralizado horizontalmente e o indicador de vida no canto superior esquerdo da tela e o contador de pontos iniciado zerado no canto superior direito. A textura do fundo deve ser carregada também nesse momento.

Prioridade: Alta.

RF007 - Movimentação

A movimentação da nave/avião do jogador será feita através do teclado. A tecla W deve movimentar o jogador para cima, S para baixo, as teclas A e D movimentam o jogador horizontalmente para esquerda e direita respectivamente.

Prioridade: Alta.

RF008 - Atirar

A nave do jogador será equipada com uma arma para destruir os inimigos, os tiros da arma serão ativados pela tecla espaço e caso o jogador segure a tecla os tiros terão uma cadência de 5 tiros por segundo.

Prioridade: Alta.

RF009 - Sair do Jogo

Para sair do jogo em execução o jogador pode apertar ‘esc’ e voltar diretamente para o menu inicial, sem tela de pausa ou confirmação da ação. Nesse momento todos os parâmetros do jogo serão reiniciados.

Prioridade: Média.

RF010 - Inimigos

O jogo irá gerar inimigos aleatoriamente na parte mais alta da tela que caminharão para baixo em direção ao jogador. Esses inimigos poderão surgir em qualquer ponto horizontal dentro da tela, mas sempre no mesmo ponto vertical.

Prioridade: Alta

RF011 - Eliminação dos inimigos

Cada inimigo terá uma quantidade de vida que corresponde a quantos tiros ele precisa levar para ser eliminado. Essa quantidade de vidas não será exibida na tela para o jogador.

Prioridade: Baixa.

RF012 - Perda de Pontos

Para cada inimigo que atingir a parte de baixo da tela, onde se encontrará a base do jogador, o jogador irá perder uma vida. Quando o jogador não tiver mais nenhuma vida o jogo irá parar e mostrar uma mensagem de Game Over, indicando que o jogador deverá apertar ‘esc’ para voltar ao menu.

Prioridade: Média.

RF013 - Contagem de Pontos

Cada inimigo que o jogador eliminar o jogo irá incrementar na pontuação do jogador mais 50 pontos, quando o inimigo for eliminado os pontos serão atualizados no canto superior direito da tela.

Prioridade: Baixa.

RF014 - Dificuldade

A dificuldade do jogo será incremental, quanto mais pontos a pessoa fizer mais difícil será. A dificuldade será incrementada aumentando a velocidade com que os inimigos descem a tela até um certo limite a ser determinado em testes de jogabilidade e aumentando a quantidade de inimigos que podem aparecer simultaneamente.

Inicialmente poderão aparecer somente 4 inimigos simultâneos, e essa quantidade deve aumentar em 1 a cada 50 pontos que o usuário marcar.

Prioridade: Baixa.

RF015 - Iluminação

A nave terá uma luz que pode usar para iluminar os inimigos à sua frente e poderá ligar e desligar essa luz apertando a letra F.

Prioridade: Média.

2.4 Requisitos Não Funcionais

RNF001 - Interface Gráfica

O desenvolvimento da interface gráfica será feito em um ambiente 2D usando o OpenGL para renderização dos elementos visuais.

Prioridade: Alta.

RNF002 - Ambiente de Execução

O jogo deve ser capaz de ser executado em ambientes linux com as bibliotecas necessárias instaladas.

Prioridade: Alta.

RF003 - Colorização e Textura

O jogo deve utilizar colorização e texturas diferentes para distinguir os objetos.

Prioridade: Alta.

RF003 - Hardware

Para a execução correta do jogo o computador deve ter placa de vídeo, integrada ou dedicada, compatível com OpenGL.

Prioridade: Alta.

RF004 - Iluminação

O jogo deve ter alguma forma de utilização de iluminação.

Prioridade: Alta.

RF005 - Teclado

O jogo deve ter alguma forma de interação com teclado.

Prioridade: Alta.

RF006 - Mouse

O jogo deve ter alguma forma de interação com mouse.

Prioridade: Alta.

RF007 - Animação

O jogo deve ter alguma forma de animação em pelo menos 1 objeto.

Prioridade: Alta.

RF008 - Hierarquia

O jogo deve ter tratamento de objetos hierárquicos.

Prioridade: Alta.

RF008 - Transformações Geométricas

O jogo deve ter operações de transformação geométrica.

Prioridade: Alta.

3. Codificação

Esperamos em primeiro plano que o usuário tenha uma boa experiência e facilidade para usar o software, além disso, o software deve ser acessível para todos computadores da atualidade, desde computadores com poucos recursos de processamento até computadores mais eficientes.

O software será desenvolvido no Sistema Operacional Ubuntu 22.04.1 LTS. Ele será codificado na linguagem de programação C++, linguagem que, em conjunto com a linguagem de programação C, foi utilizada nesta disciplina de computação gráfica.

Além disso, utilizaremos a API OpenGL com suas bibliotecas GLUT e GLU. Essas bibliotecas nos ajudarão a construir este projeto do Guerra Especial em 2D. Durante o processo de codificação, iremos utilizar o editor de texto Visual Studio Code para implementarmos e testarmos nosso software. A biblioteca GLUT fica responsável pela criação de janelas e também pelo tratamento de eventos de dispositivos de entrada como mouse, touchpad e teclado. O paradigma de programação em GLUT é orientado a eventos do tipo, ou seja, captura, por exemplo, quando uma tecla for pressionada e quando for solta.

Os meios de interação do usuário com o jogo serão mouse e teclado. Para acessar as opções do menu, o usuário utilizará o mouse, e para jogabilidade o teclado que será utilizado.

Uma aplicação simples pode ser implementada diretamente em um único algoritmo. Contudo, na construção deste projeto, a maioria das tarefas necessitam de um planejamento mais eficiente para a implementação correta. Sendo assim, faz-se necessário a decomposição das funcionalidades do sistema em aplicações menores.

Conforme as funcionalidades crescem e se tornam mais complexas, surge uma série de situações que precisam ser tratadas, como por exemplo a implementação das diferentes transformações geométricas. Isto posto, cada

situação pode ser resolvida isoladamente por meio de um algoritmo particular. Como resultado, espera-se que as soluções parciais de cada situação possam ser combinadas e juntas serem capazes de produzir os resultados esperados. Além disso, com esta modularização almejamos uma clareza e objetividade maior no software criado, e uma manutenção de código mais eficiente e eficaz, haja visto que será empregado na construção do projeto as técnicas como subprogramas, métodos, rotinas, módulos, unidades, bibliotecas, entre outras.

Toda biblioteca possui muitas funções e métodos prontos que são disponibilizados para facilitar que o desenvolvimento seja mais rápido e fácil. Na biblioteca GLUT, todos os nomes de funções começam com o prefixo glut. Ao longo da disciplina, foram utilizados uma boa parte dessas funções e também serão utilizados com grande importância para o funcionamento do nosso projeto.

Por conseguinte, as estratégias descritas nos trazem benefícios para construir este projeto de um modo decisivo e palpável. Considerando que os requisitos estabelecidos possam ser cumpridos, as propostas abaixo, assim como outras não mencionadas, estabelecem os procedimentos a serem adotados. Abaixo terá uma lista das principais funções que utilizamos no desenvolvimento do projeto.

glutInit(&argc, argv): Utilizada para inicializar a biblioteca GLUT e recebe como parâmetros as opções de execução via linha de comando (command line options). Ela deve ser chamada ANTES de qualquer rotina GLUT que não seja de inicialização.

glutInitDisplayMode( ): Define o modo de exibição inicial da tela.

glutInitWindowSize( ): Especifica o tamanho inicial da janela.

glutInitWindowPosition( ): Especifica a posição da janela na tela.

glutCreateWindow( ): Cria uma janela principal com o nome recebido como parâmetro.

glutDisplayFunc( ): Registra a rotina para exibição da janela.

glutReshapeFunc( ): Registra a rotina para reexibição da janela.
glutKeyboardFunc( ): Registra a rotina para tratamento de eventos gerados pelo teclado.

glutMouseFunc( ): Registra a rotina para tratamento de eventos originados pelo clique do mouse.

glutMainLoop( ): Esta rotina deve ser chamada após a inicialização. O controle do programa é entregue a GLUT que passa a monitorar os eventos e chamar as rotinas de callback apropriadas.

As transformações geométricas são essenciais para a construção de um programa interativo e animado em OpenGL. Por meio dos métodos **glRotatef()**, **glTranslatef()** e o **glScalef()** é possível aplicar rotações, translações e operações de escala. Apoiados por variáveis de controle que são atualizadas nas funções de interação por teclado e mouse, estes métodos permitiram as animações e movimentos de câmera implementados no projeto, além do suporte às operações para renderizar a cena.

Duas importantes funções do OpenGL que são responsáveis por empilhar e desempilhar matrizes de desenho são as **glPushMatrix()** e **glPopMatrix()**. Essas chamadas permitem isolar transformações geométricas. E para a implementação das cores dos objetos será utilizado o método de definição de cor **glColor3f()**, que recebe três parâmetros do tipo float que integra a cor no formato RGB.

Além das funções principais de um programa em OpenGL utilizando a GLUT, também iremos aproveitar, como estratégia, algumas funções/implementações que realizam o procedimento que esperamos, disponibilizados nos exemplos do redbook.

4. Resultados Esperados

Como resultado, esperamos conseguir representar o ambiente com características sucintas do espaço sideral. Dessa forma, estará presente uma iluminação que lembre um ambiente majoritariamente escuro com alguns pontos de luz clara, para representar estrelas, e outros tons de cores mais escuras. Nos tópicos que serão apresentados abaixo, indicaremos de forma mais objetiva e específica os objetivos que queremos alcançar.

4.1. Modelagem

O primeiro passo a ser desenvolvido seria o menu do jogo, nesse menu o jogador terá as opções “Jogar”, “Instruções”, “Sair” e “Créditos”. O menu será uma tela simples, apenas com as opções e um fundo de cor neutra. O jogador poderá acessar cada opção através do mouse. A opção jogar abrirá a tela principal do jogo, onde o jogador encontrará uma simulação do espaço, com luzes bem definidas, sons, animação de objetos, etc. Na opção instruções, será mostrado ao jogador instruções de como controlar a nave para movimentação e para atirar. A opção sair fecha a janela do jogo e a opção créditos mostrará os nomes dos integrantes do grupo.

4.2. Ambiente

O ambiente de jogo terá uma cor escura como fundo, predominantemente preta e com luzes claras para representação de estrelas, nesse ambiente, cada objeto terá uma cor e alguns terão iluminação, a figura abaixo apresenta uma ideia de como será o fundo da tela principal do jogo.

Figura representando o espaço sideral

Fonte:
https://br.freepik.com/fotos-premium/estrelas-e-galaxia-espaco-sideral-ceu-noite-universo-preto_6953256.htm

4.3. Interação com teclado e mouse

A interação com o mouse será exclusivamente no menu, não existindo na parte da jogabilidade. O usuário poderá clicar nas opções do menu na qual deseja. Dentro do jogo o jogador utilizará as seguintes teclas: W deve movimentar o jogador para cima, S para baixo, A e D movimentam o jogador horizontalmente para esquerda e direita respectivamente e a tecla Espaço faz a arma da nave disparar tiros. Dentro do jogo, ao apertar a tecla esc o jogador será redirecionado para o menu.

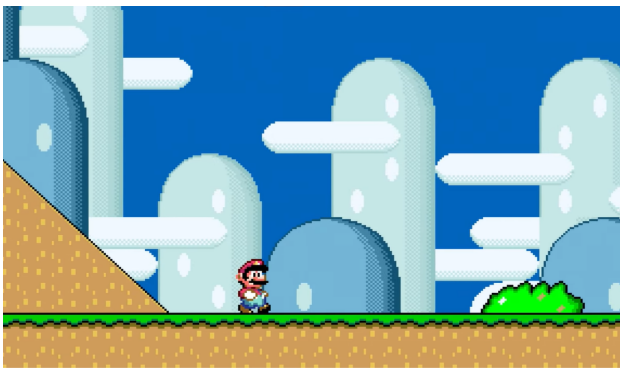
Além disso, a nave terá uma opção de acender uma luz ao apertar a tecla F, dessa forma, os inimigos ficarão iluminados.

4.4. Operações de transformação geométrica

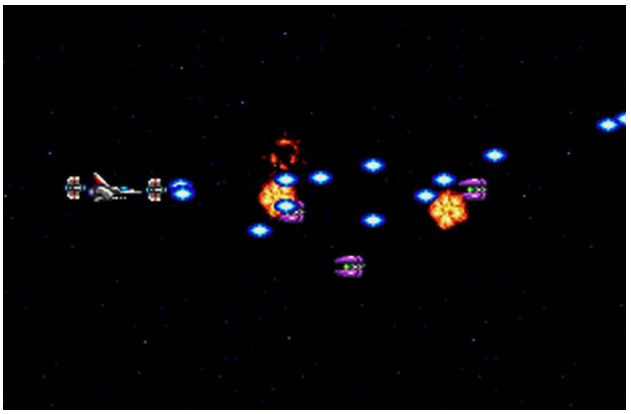
Durante o tempo de vida da aeronave, sua hélice, que ficará na parte superior do corpo, terá o movimento de rotação continuamente, dessa forma vai ser possível notar o giro da hélice enquanto a nave estiver vagando.

4.5. Câmera

A câmera do jogo mostrará ao usuário uma perspectiva de cima, o ambiente será 2D e portanto não haverá mudanças na câmera. As imagens a seguir mostram alguns exemplos de jogos desenvolvidos em um ambiente 2D.



Exemplo de jogo em 2D.



Exemplo de jogo em 2D.

4.6. Iluminação

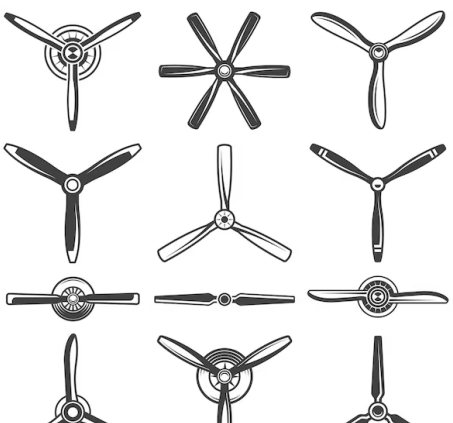
Aplicamos uma iluminação com opção do usuário ativar e desativar. A iluminação da nave funciona como uma espécie de farol, podendo deixar o ambiente mais claro ou mais escuro, dependendo da escolha do usuário. Isso traz mais dinamicidade e personalização para o ambiente.

Exemplo de iluminação de uma aeronave

Fonte: ***https://www.rbsdirect.com.br/imagesrc/25655801.jpg?w=700***

4.7. Animação

A animação empregada no jogo é a de rotação da hélice da aeronave, ela não poderá ser desligada, ficará rotacionando durante todo o tempo de vida da nave. Para fazer a rotação da hélice iremos aplicar um ponto fixo no meio dela para que possa ser feita a rotação sem nenhuma alteração, rotacionado de forma correta apenas as hélices e não a nave.



Animação: representação da rotação das hélices da nave.

4.8. Movimentação

A movimentação será responsável pela interação do usuário com a nave, ele poderá escolher em qual direção movimentá-la, podendo ser vertical ou horizontalmente. Dessa forma o jogador poderá escolher o melhor posicionamento para se esquivar dos ataques dos inimigos e também se posicionar da forma mais adequada para atingi-los.

4.9. Textura

Para a textura, pegamos algumas já definidas para representação do ambiente e dos objetos presentes nele. Dessa forma, vai ficar mais parecido com um jogo e não tendo o intuito de parecer realista, afinal é uma simulação. Além disso, o jogador não poderá fazer modificações nas texturas existentes dentro do jogo.

5. Conclusão

Neste documento, delineamos os requisitos para o desenvolvimento de um jogo 2D que incorpora os princípios da computação gráfica. O jogo envolve o controle de uma nave/avião armado, a eliminação de inimigos e a acumulação de pontos. Através dos requisitos funcionais, definimos a mecânica do jogo, incluindo opções de menu, instruções, controles de jogabilidade, geração e eliminação de inimigos, sistema de pontuação e progressão de dificuldade.

Além disso, especificamos requisitos não funcionais relacionados à interface gráfica, ambiente de execução, compatibilidade de hardware, iluminação e dispositivos de entrada. Esses requisitos garantem que o jogo possa ser executado em um ambiente adequado e ofereça uma experiência envolvente e visualmente atraente para os jogadores.

Por meio do desenvolvimento deste jogo, nosso objetivo é aplicar os conhecimentos e conceitos adquiridos no campo da computação gráfica. Utilizaremos a biblioteca OpenGL para renderizar os visuais do jogo e incorporaremos colorização, texturas e efeitos de iluminação para aprimorar a experiência visual.

Por meio desse processo de desenvolvimento, esperamos obter experiência prática na implementação dos algoritmos, modelos e técnicas aprendidas, além de aprimorar nossa compreensão das práticas de codificação, renderização gráfica e desenvolvimento de jogos como um todo. Este projeto servirá como uma oportunidade para consolidar nosso conhecimento e demonstrar nossas habilidades na criação de uma aplicação completa usando renderização gráfica.

Em conclusão, o desenvolvimento deste jogo não apenas colocará em prática nosso conhecimento adquirido, mas também proporcionará uma experiência emocionante e interativa para os jogadores. Estamos confiantes de que, por meio do cumprimento dos requisitos especificados e da aplicação de nossas habilidades, criaremos um jogo envolvente que demonstra nossa compreensão dos princípios de computação gráfica e desenvolvimento de jogos.