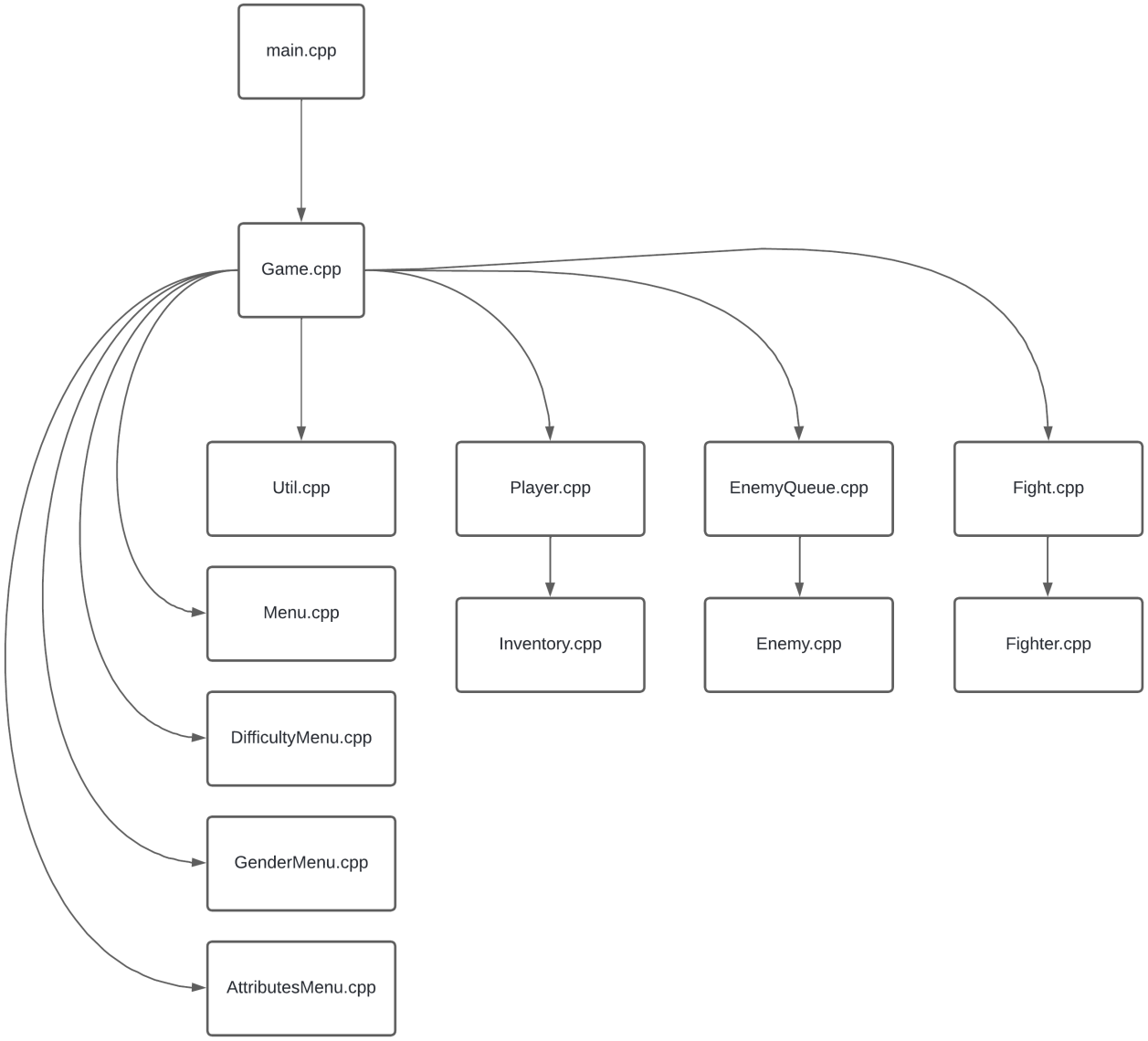
Game 2 - Estrutura de Dados



Comentários gerais

O game é um side-scroller, foi desenvolvido utilizando orientação à objetos (por que os tutoriais usavam) e a parte visual foi feita com SFML.

Possíveis Perguntas do Bruno e respostas plausíveis

O que você melhoraria no game?
Colocaria um combate mais dinâmico onde o usuário teria mais escolhas e funcionasse 100% na tela do jogo, adicionaria mais animações, sistema de pontos, possibilidade de pulo, plataformas, entre outros.

Por que vocês escolheram a estrutura de fila de inimigos?
Porque assim podemos adicionar mais inimigos no fim da fila e ao mesmo tempo ir removendo os inimigos que o personagem derrotar, isso abre ainda mais possibilidades caso desejássemos aumentar o escopo do game

Por que uma lista encadeada simples no inventário?
Para facilitar a remoção de itens independentemente da posição que ele se encontra.

Onde usaríamos pilha?
Talvez no sistema de turnos, onde o usuário poderia voltar e jogar um turno anterior

Onde usaríamos uma fila de prioridade?
Em um possível sistema de buffs e debuffs aprimorado, onde um buff poderia ter prioridade sobre um debuff, e vice versa.

Qual a principal dificuldade que tivemos ao desenvolver o jogo?
A integração do combate ao jogo, onde não conseguimos manter a mesma dinamicidade do restante do game.

main.cpp

- Inicia uma instância de Game
- Abre um while (game.getWindow().isOpen()), que é basicamente um while infinito enquanto a janela do jogo estiver aberto e fica atualizando o game (chamando as funções render e update do Game.cpp)

Game.cpp

- Inicializa o estado do jogo como PLAYING
- Inicializa o arquivo Util.cpp que é basicamente um ajuste de teclado para os Menus
- Inicializa a janela
- Inicializa a câmera
- Inicializa o Menu e aguarda sua execução
- Inicializa o DifficultyMenu e aguarda sua execução
- Inicializa o GenderMenu e aguarda sua execução
- Inicializa o AttributesMenu e aguarda sua execução
- Inicializa o Player
- Inicializa a EnemyQueue informando qual será a dificuldade
- Inicializa, carrega e dá play na música de fundo em loop

Update

- Atualiza o posicionamento da câmera conforme o jogador anda, escuta o evento de teclado e movimenta o player, atualiza a animação do inimigo mais próximo e calcula a distância entre o player e ele
- Caso a distância <= 950, pausa a movimentação do jogador, inicia o combate, verifica o inventário do jogador, se vai ser usado algum item, realiza a luta criando uma nova instância de Fight, após ela acontecer verifica o ganhador
    - Caso seja o Player, verifica se o inimigo derrotado foi o BOSS,
        - Se for, troca o estado para VICTORY, pausa a música do jogo, carrega a música de vitória e dá play nela
        - Se for um inimigo normal, faz o sorteio do drop de item, remove o inimigo da fila e libera novamente a movimentação do personagem
    - Caso seja o Inimigo, troca o estado do jogo para DEFEAT, pausa a música do jogo, carrega a música de derrota e dá play nela

Render

- Verifica o estado do game
    - Caso PLAYING, renderiza em tela o background, o player e renderiza em suas posições no mapa os inimigos que já foram inicializados
    - Caso VICTORY, volta a câmera para a posição inicial, remove o background anterior e coloca o background de vitória
    - Caso DEFEAT, volta a câmera a posição inicial, remove o background anterior e coloca o background de derrota

Menu.cpp

- Inicializa o background do menu e a fonte, posiciona duas opções e escuta as teclas W, S e Enter
    - Quando o usuário aperta Enter e a posição for 1, o menu se fecha, deixando o código do game continuar a rodar
    - Quando o usuário aperta Enter e a posição for 2, a janela se fecha, saindo do While em main.cpp, encerrando o game

DifficultyMenu.cpp

- Inicializa o background do menu e a fonte, posiciona três opções e escuta as teclas W, S e Enter
    - Quando o usuário aperta Enter e a posição for 1, a dificuldade é setada como 1 (Fácil)
    - Na posição 2 (Médio)
    - Na posição 3 (Difícil)

- Após qualquer uma dessas opções o menu se fecha e o game continua a rodar

GenderMenu.cpp

- Inicializa o background do menu e a fonte, posiciona duas opções e escuta as teclas W, S e Enter
  - Quando o usuário aperta Enter e a posição for 1, o gênero do personagem é setado como 1 (Masculino)
  - Quando a posição for 2, o gênero é 2 (Feminino)
- Após qualquer uma dessas opções o menu se fecha e o game continua a rodar

AttributesMenu.cpp

- Inicializa o background do menu e a fonte, posiciona as opções + e - para cada atributo e o botão de confirmar, escuta as teclas W, A, S, D e Enter, esse menu altera diretamente na memória do player os valores, usando referências
  - Quando o usuário aperta enter em algum "+", e ele possui atributos restantes, ele aplica um ponto a esse atributo
  - Quando o usuário aperta enter em algum "-", e ele possui pelo menos 1 ponto aplicado no atributo selecionado, esse ponto é retirado
  - Quando o usuário aperta enter no botão confirmar, a o menu se fecha e o game continua a rodar, o botão confirmar apenas realiza sua ação se todos os pontos forem distribuídos

Player.cpp

Carrega as variáveis de velocidade, o sprite e o posicionamento, cuida também das animações e fornece métodos para o gerenciamento do inventário

EnemyQueue.cpp

Inicializa uma fila, sorteia uma quantidade de inimigos e coloca na fila, e por fim sorteia um boss e também coloca na fila, permite visualizar a fila completa e remover os inimigos conforme forem sendo derrotados

Fight.cpp

Recebe o Player e o inimigo que irão lutar, verifica o item em uso e aplica o buff concedido ao jogador, transforma o jogador e o inimigo em uma instância de Fighter, entra em um while de no máximo 10 turnos:

- Sorteia o primeiro a atacar
- Verifica e aplica o sangramento no perdedor do sorteio
- Verifica e aplica o sangramento no vencedor do sorteio
- Faz o vencedor do sorteio performar o ataque
- Faz o perdedor do sorteio performar o ataque

A cada aplicação de dano é verificado se o sofredor foi derrotado.
Após o combate são removidos os bufs ao jogador, o item tem sua duração decrementada e as informações do combate são exibidas (dano, vida restante, turnos ocorridos, ataques especiais, esquivas)

Inventory.cpp

Inicializa uma lista encadeada para gerenciamento dos itens no inventário, permite adicionar um item ao fim da lista, remover itens tanto pelo nome quanto pela posição, visualizar o inventário completo, verificar se está vazio, encontrar um item pela posição, e decrementar a duração de um item

Enemy.cpp

Carrega as sprites de todos os inimigos, pelo id e tipo, também os status de cada um, além de atualizar a animação também

Fighter.cpp

Carrega a vida máxima de um personagem, o tanto de dano que ele causou no combate e quantos rounds ele precisa sangrar, possui o método de performar ataque que verifica primeiro se o defensor vai desviar

- Caso ocorra, aumenta o número de desvios em Fight e finaliza o ataque
  Depois calcula qual será o dano de ataque e se será ataque especial
- Caso seja ataque especial, aplica a lógica de sorteio de sangramento
  Por fim, aplica o dano no defensor