

Projeto Cucco Race

Especificação de Requisitos de Software

Versão 1.2

22/06/2016

Versão	Autores	Data	Ação
1.0	Diego Feijó	07/04/2016	Estabelecimento dos requisitos.
1.1	Diego Feijó	26/04/2016	Revisão sobre requisitos conforme avaliação da primeira entrega.
1.2	Diego Feijó	22/06/2016	Retirado conceito de NPCs e reestruturado o conceito de eventos. Atualizado esboço da GUI.

Conteúdo:

1. Introdução;
2. Visão geral;
3. Requisitos de software;
4. Esboço Interface Gráfica;
5. Referências.

1 Introdução:

Objetivo: desenvolvimento de um software distribuído que emule uma disputa entre dois jogadores em um tabuleiro que propõe uma corrida por turnos, permitindo que jogadores se enfrentem em confrontos eliminatórios.

Nomenclatura utilizada neste documento:

- **Mapa:** representação do tabuleiro no sistema, composto de casas e *tiles* de cenário.
- **Casas:** parte jogável do mapa, onde os jogadores poderão interagir com outros personagens e itens de cenário. Cada casa é indicada como uma fração do mapa, chamada de *tile*, nomenclatura padrão de jogos baseados em tiles. Uma casa pode ser ocupada por apenas um personagem.
- **Entidade:** generalização para inimigos e jogadores.
- **Jogador:** entidade que responde a comandos e representa o usuário do sistema.
- **Inimigos:** entidades que disparam eventos de confronto.
- **PdV:** pontos de vida da entidade. Determinam a saúde e podem ser alterados durante o jogo. Uma entidade sem PdV é eliminada do mapa.
- **Eventos:** ações disparadas ao fim do movimento de um jogador, que podem ser de coleta de joias, atalho, informações, lojas ou confrontos.
 - **Joias:** o jogador adiciona a quantidade correspondente a cor da joia ao seu número de joias.
 - **Atalho:** casas especiais podem servir de atalho para os jogadores mudando sua posição para o fim do atalho.
 - **Informações:** algumas casas apresentam dicas ou mensagens interessantes aos jogadores.
 - **Lojas:** casas que representam lojas podem vender bônus de movimento ao jogador em troca de uma certa quantidade de suas joias.
 - **Confronto:** quando um jogador finaliza seu movimento em uma casa ocupada por um inimigo, acontece um confronto que decidirá quem ocupará a casa em questão. O jogador atual pode decidir entre enfrentar ou conceder a vitória. No primeiro caso, o atacante rolará dois dados de seis faces para decidir seu ataque e o defensor rolará um dado de seis faces para cada ponto de defesa. Quem conseguir o maior número, realiza o ataque, fazendo com que o defensor perca uma quantidade de seus PdV. O confronto se repetirá até que uma das partes não tenha PdV restante, ou até que o jogador desista do confronto, concedendo a vitória e regressando uma casa.

Regras do jogo:

Ao iniciar o jogo os jogadores são posicionados na casa inicial do mapa. O jogador que iniciou a partida será o primeiro e os próximos jogadores respeitarão a ordem de conexão. A cada turno os jogadores poderão executar as seguintes ações:

- **Rolar** dados para determinar o número de casas que pode movimentar no turno;
- **Movimentar** exatamente o número de casas sorteado, sendo que as regras de evento devem ser respeitadas ao fim do movimento.

- **Decidir** sobre eventos disparados, entre eles:
 - Escolher enfrentar inimigos, se estiver na mesma casa;
 - Interagir com lojas;

Cada jogador inicia com uma quantidade de PdV que pode ser alterada em confrontos. Os jogadores sem pontos de vida são eliminados da partida.

Vence o primeiro jogador que chegar ao fim do mapa ou o jogador remanescente.

2 Visão Geral

Arquitetura do programa: programa orientado a objetos, distribuído.

Premissas de desenvolvimento:

- O programa deve apresentar uma interface gráfica bidimensional, dividida em uma matriz onde cada posição será referenciada como um *tile*, um quadrado de 32 pixels de lado, com uma paleta de 256 cores, similar aos jogos 16 bits.
- O programa fará uso da linguagem de programação Java em sua versão SE8, com o paradigma de Orientação a Objetos, do software para cliente-servidor NetGames.
- A estrutura básica do jogo será gerida pelo software TiledGame2D, um *framework* que utiliza as bibliotecas do Java para controlar as regras básicas, comuns a jogos bidimensionais baseados em *tiles*.

3 Requisitos de Software

3.1 Requisitos Funcionais:

1. **Conectar:** como será um software distribuído, deve apresentar uma interface gráfica que permita ao usuário conectar-se ao servidor NetGames que fará a gestão dos usuários conectados.
2. **Desconectar:** deve permitir ao usuário que possa se desconectar de um servidor via interface gráfica.
3. **Iniciar partida:** uma vez conectado, o programa deve possibilitar a inicialização de uma nova partida, onde será indicado o número de jogadores desta partida. O programa deve permitir partidas contendo dois jogadores.
4. **Composição do mapa:** o mapa deve ser composto de tiles jogáveis (casas) e tiles de cenários. O conjunto de casas servirá como o caminho entre o ponto de partida até o fim do

mapa, podendo haver mais de um caminho possível. Os tiles de cenário servem apenas como imagem de fundo, não permitindo que os jogadores caminhem por eles.

5. **Rolar Dados:** para decidir a quantidade de casas que será possível andar no turno, o programa deve simular ao usuário uma rolagem de dados, mostrando na interface gráfica o resultado desta rolagem e guardar este valor como movimentação restante do personagem.
6. **Movimento do jogador:** o jogador deve ter a possibilidade de se movimentar no tabuleiro utilizando as teclas direcionais do teclado, sendo que cada movimentação corresponde a um tile e deve respeitar a quantidade restante de movimentações possíveis. Um movimento bem-sucedido segue as seguintes condições:
 - O jogador deve ter movimentação disponível;
 - O tile ao qual deseja mover-se deve ser uma casa do tabuleiro.
 - Ao fim de cada movimento as seguintes ações devem ser executadas:
 - Decrescer a movimentação disponível do jogador;
 - Conferir se existem movimentos a serem utilizados por esse jogador;
 - Se não houverem movimentos restantes, conferir se existem eventos a serem disparados;
 - Se um evento foi disparado, decidir se executará ou não o evento. A opção pela não execução deve fazer com que o jogador regresse uma casa;
 - Conferir se a condição de vitória foi satisfeita e atribuir vitória ao jogador que executou o movimento em caso positivo;
 - Se não houverem movimentos restantes e a condição de vitória não foi satisfeita, finalizar o turno do jogador;
7. **Atualizar imagens:** a cada *tick* de jogo – controlado pelo framework TiledGame2D – as imagens devem ser atualizadas, fazendo com que as animações ocorram sutilmente.
8. **Atualizar regras:** as regras de jogo devem ser conferidas a cada *tick* – controlado pelo framework TiledGame 2D – o que garante o *loop* do jogo. Assim, os elementos que controlam regra e interface devem implementar o *tick* do framework e ali aplicar suas regras.

3.2 Requisitos não funcionais

1. **Especificação de projeto:** além do código Java, deve ser produzida especificação de projeto baseada em UML segunda versão.
2. **Identificação dos jogadores:** os jogadores devem ser identificados por um nome inserido ao iniciar uma nova partida.
3. **Tecnologia da GUI:** *graphic user interface* baseada na biblioteca nativa *Graphics2D*, auxiliado pelo software TiledGame2D.

4 Esboço da Interface Gráfica

A interface gráfica do jogo (*GUI*), será composta de recursos disponíveis nas comunidades de desenvolvedores de jogos na web, sendo suas respectivas fontes referenciadas ao fim deste documento. A ferramenta Tiled será utilizada para criação dos mapas.



Figura 1 - Esboço da GUI

5 Referências

MORGAN, Leonardo. **NetGames**. Disponível em: <<http://www.inf.ufsc.br/~netgames/>>. Acesso em: 07 abr. 2016.

LINDEIJER, Thorbjørn. **Tiled Map Editor**. Disponível em: <<http://www.mapeditor.org/>>. Acesso em: 07 abr. 2016.

THEVGRESOURCE. **The VG Resource**. Disponível em: <<http://www.sprisers-resource.com/>>. Acesso em: 07 abr. 2016.