**Documento de Game Design**

**POOGame**

**Brendo Colaço**

**Jefferson Barboa**

**Léuson Mário da Silva**

**Rogério Carvalho**

**Wellington Lucas**

**22/03/2015**

**INTRODUÇÃO**

O jogo têm como objetivo estimular a pratica dos conceitos básicos de Programação Orientada a Objeto (POO).

**HISTÓRIA**

Apos sua família ser sequestrada por criaturas comandas pelo rei **tal**, um garoto parte em uma jornada em busca de seus pais. Porém esse caminho não será fácil, ele terá diversos desafios a sua frente. No meio de sua jornada ele acaba inconsciente devido a uma queda em uma buraco, deixado por caçadores, para capturar criaturas. No dia seguinte quando ele acorda percebe que está em uma gruta e logo tenta buscar uma saída. Após algumas horas de caminhada ele encontra um velho que fornece informações de onde poderiam estar seus pais e menciona que o garoto não possui força suficiente para enfrentar o exercito do rei e se propõe a ajudar a ensinar sobre a mágica desconhecida de POO. O velho mago fala que a gruta possui alguns pergaminhos e que apenas dois deles têm o poder necessário para invocar magias. O velho menciona também que muitos outros estão espalhados pelo reino de **tal** e que ele deve juntar todos eles para poder aumentar seu nível de magia e derrotar o exercito do rei **tal** e salvar sua família. Antes do garoto continuar sua jornada o velho deixa com ele o livro de boas práticas da POO e menciona que o livro deve ser seguido caso o garoto deseje maior eficiência de suas magias. agora em busca dos pergaminhos elementares da POO, medieval, um menino normal que aprende a ser mago ao longo do jogo, sua sobrevivência.

**CONTROLES**

|  |  |
| --- | --- |
| **CONTROLE** | **DESCRIÇÃO** |
| A | Corre/vira para a esquerda |
| D | Corre/vira para a direita |
| W | Subir ( em obstáculos específicos) |
| S | Descer (em obstáculos específicos) |
| Shift | Aterrissar/abaixar/soltar obstáculo |
| Barra de Espaço | Pular/agarrar obstáculo |
| Mouse esquerdo | Ataque |
| Mouse direito | Posição de defesa |
| (Em posiçao de defesa e na hora em que vai receber um ataque do inimigo) mouse esquerdo | Contra ataque |
| (Durante pulo para a frente) mouse esquerdo | Flying kick |
| Roda do mouse | Ativar adrenalina (câmera lenta + força extra) |

**ELEMENTOS DO JOGO**

**Obstáculos**

Pedras, cactus, areia, árvores, poças de água, larva de vulcão.

**Inimigos**

Ratos, cobras, lagartos, Unicórnio de duas cabeças,

**Barra de vida**

Marca a quantidade de “vida” do jogador. Quando ela zera o jogador perde o jogo e tem que continuar do último checkpoint salvo.

**SISTEMA DE PONTUAÇÃO**

**<descrição de como ganhar e perder pontos e/ou vida + como avança de fase>**

O jogador possui uma barra de “vida” e uma barra de “adrenalina”(câmera lenta) . Cada fase começa com a barra de vida cheia, perde-se vida com quedas ou recebendo ataques de inimigos, se a barra de vida zerar o jogador perde o jogo. É possível repor a vida coletando quites médicos que estão espalhados pelas fases.

Cada fase sempre começa com a barra de adrenalina vazia a qual é preenchida somente enquanto o jogador corre. O jogador então pode ativar a adrenalina acumulada para que o jogo fique em camera lenta e o jogador mais “forte”, neste momento, a barra de adrenalina começa a zerar rapidamente. O jogo fica em camera lenta até que o jogador desative a adrenalina ou a barra zere. Ativar a adrenalina é útil em momentos de luta, pois dessa forma se torna mais fácil usar o contra ataque e os ataques causam mais danos.

O jogador deve chegar aos checkpoints em um tempo pré-determinado. Esse tempo pode ser marcado por um cronômetro na tela ou marcado por eventos. Exemplo: o jogador deve perseguir e capturar um inimigo antes que ele escape. Quanto mais rápido o jogador terminar as fases maior será sua pontuação. Derrotar inimigos não agrega pontos.

**AÇÕES DO JOGO**

**Deslocamento**

O jogador pode se deslocar pelo cenário.

**Combinar classes**

O jogador pode combinar duas ou mais classes para formar novos movimentos.

**Visualizar Descrição**

O jogador pode visualizar a descrição de alguns objetos deixando o mouse sobre o mesmo.

**Coletar itens**

O jogador pode coletar pergaminhos que municiarão seu arsenal de magias. Para coletar os pergaminho basta pressionar a tecla de ação.

**Atacar**

O jogador pode selecionar um dos métodos de ataque a partir das instancias de classe**.**

**Defender**

O jogador pode selecionar uma classe de defesa de acordo com sua necessidade e instanciá-la para realizar sua defesa.

**Conversar**

O jogador pode conversar com personagens do jogo pressionando a tecla de ação.

**DIFICULDADE**

A dificuldade do jogo é marcada pelo o nível dos monstros que aumentam de acordo com o avanço das fases.

**REGRAS DE BATALHA**

Ao transitar pelo terreno do jogo, o personagem principal pode se deparar com monstros, quando isso acorrer uma batalha se iniciará. A câmera se deslocará para melhor apresentar a batalha.

As regras são descritas a seguir:

* **Configuração geral:**

1. O personagem fica fixo no mapa;
2. A batalha será dividida em rodadas;
3. Cada personagem, tanto o principal quanto o monstro terão seus momentos de ataque;
4. O personagem principal invocará monstros para efetuar seus ataques.
5. O personagem principal será o primeiro a atacar, onde ele pode fazer um ataque e fazer um movimento de defesa;
6. O monstro ataca após a finalização da vez do personagem principal, podendo atacar e ficar em modo de defesa;
7. Os ataques/defesas são métodos das classes que representam os monstros que o personagem ganhar após vencer as batalhas.
8. Os comandos de ataques e defesa seguirão os comandos de batalha definidos.

* **Ataque:**

1. Para efetuar um ataque o jogador deve escrever um método, que esteja contido na sua classe/pergaminho escolhida, no console apresentado.
2. O jogador pode combinar métodos ou classes de modo a aumentar o dano dos ataques, uma combinação é permitida por vez.
3. As classes poderão apresentar vantagens sobre tipos específicos de monstros. Ex.: Uma classe onde a natureza do Objeto gerado é água, terá vantagens sobre monstros cuja natureza é fogo.

água > fogo > vento > raio > terra > água

1. Para o ataque ser efetivado o jogador deve escrever corretamente o método desejado. Caso erre na sintaxe do método deve ser apresentado um aviso e a opção de ataque bloqueada, só podendo ser feito movimento de defesa.

* **Defesa:**

1. Para efetuar um movimento de defesa o jogador deve escrever um método, que esteja contido na sua classe/pergaminho escolhida, no console apresentado.
2. As classes poderão apresentar vantagens sobre tipos específicos de monstros. Ex.: Uma classe onde a natureza do Objeto gerado é água, terá vantagens sobre monstros cuja natureza é fogo.

água > fogo > vento > raio > terra > água

1. Para a defesa ser efetivada o jogador deve escrever corretamente o método desejado. Caso erre na sintaxe do método deve ser apresentado um aviso e a opção de defesa bloqueada, passando a vez do jogador.

* **Final:**

1. Caso o jogador vença a batalha ele receberá uma nova classe, de acordo com a natureza do monstro.
2. Caso o monstro vença o jogado inicia a fase novamente e procura melhores classes para enfrentar o monstro.

**CLASSES**

1. **Correta**

public class Aranha {

private float poder;

private float custo;

public Aranha() {

poder = 10;

custo = 15;

}

public float atacar(){

return poder;

}

public void defender(float dano){

custo = custo - dano;

}

}

**2) Erro, falta ponto e vírgula no construtor**

public class Aranha {

private float poder;

private float custo;

public Aranha() {

poder = 10

custo = 15;

}

public float atacar(){

return poder;

}

public void defender(float dano){

custo = custo - dano;

}

}

**3) Erro, método atacar() não tem retorno, exige um retorno do tipo float.**

public class Aranha {

private float poder;

private float custo;

public Aranha() {

poder = 10;

custo = 15;

}

public float atacar(){

poder = poder + 1;

}

public void defender(float dano){

custo = custo - dano;

}

}

**4) Erro, método defender() retorna uma atribuição;**

public class Aranha {

private float poder;

private float custo;

public Aranha() {

poder = 10;

custo = 15;

}

public float atacar(){

return poder;

}

public void defender(float dano){

return custo = custo - dano;

}

}

**5) Erro, método float atacar() retorna uma String.**

public class Aranha {

private float poder;

private float custo;

public Aranha() {

poder = 10;

custo = 15;

}

public float atacar(){

return “ataque”;

}

public void defender(float dano){

custo = custo - dano;

}

}