UNIVERSIDADE FEDERAL DO CEARÁ

INSTITUTO UFC VIRTUAL

CURSO DE GRADUAÇÃO EM SISTEMAS E MÍDIAS DIGITAIS

RAONI COELHO VIEIRA

RELATÓRIO DO DESENVOLVIMENTO DO JOGO DIGITAL DE RPG LOST BORDERS NO GODOT

FORTALEZA

2022

RAONI COELHO VIEIRA

RELATÓRIO DO DESENVOLVIMENTO DO JOGO DIGITAL DE RPG LOST BORDERS NO GODOT

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e mídias digitais do Instituto UFC Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e mídias digitais.

Orientador: Prof. Dr. George Allan Menezes Gomes

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

V718r Vieira, Raoni Coelho.
Relatório do desenvolvimento do jogo digital de RPG Lost Borders no Godot / Raoni Coelho Vieira. – 2023.
45 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Instituto UFC Virtual, Curso de Sistemas e Mídias Digitais, Fortaleza, 2023.
Orientação: Prof. Dr. George Allan Menezes Gomes.

1. Jogos digitais de RPG. 2. Motor de jogo. 3. Godot. 4. Padrões de projeto. I. Título.

CDD 302.23

RAONI COELHO VIEIRA

RELATÓRIO DO DESENVOLVIMENTO DO JOGO DIGITAL DE RPG LOST BORDERS NO GODOT

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e mídias digitais do Instituto UFC Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e mídias digitais.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. George Allan Menezes Gomes (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Henrique Sergio Lima Pequeno
Universidade Federal do Ceará (UFC)

Prof. Dr. Antônio José Melo Leite Júnior
Universidade Federal do Ceará (UFC)

À minha família, por sempre acreditar e investir em mim.

AGRADECIMENTOS

Ao Prof. Dr. George Allan Menezes Gomes por me orientar em meu TCC.

Agradeço a minha família por sempre me apoiar, hoje cheguei aqui em grande parte graças a eles.

Agradeço a todos os professores do curso, por me proporcionarem o aprendizado e a chance de adquirir da melhor forma os conhecimentos da área.

Agradeço aos membros do estúdio Guildsight, ao qual em grande parte trabalhamos juntos desde que entramos do curso. Em especial, Amanda Milena Lima Pereira, Daniel Portela Bandeira, Tiago Caúla de Oliveira Maia, Caio Vinicius Bezerra Abreu e Angelo Ceccatto Cruz Santana. Juntos fizemos algo incrível!

Agradeço aos membros da empresa júnior TGD Studio, ao qual fui um dos fundadores. A experiência de fundar e trabalhar em um empresa no curso foi muito significativa para o meu crescimento.

“Você pode fazer um jogo incrível, mas não pode ser bem sucedido. Seus jogadores que farão o sucesso.”

(Irme Jele)

RESUMO

Jogos digitais de RPG são complexos por possuírem diversas características, estão entre os gêneros mais complexos do mercado de jogos. Para isso, a utilização de *game engines*, ferramentas de facilitação do desenvolvimento de jogos, é bastante útil para a criação de jogos do gênero. Recentemente, a *game engine* Godot tornou-se bastante usada por desenvolvedores independentes, por uma série de vantagens que ela apresenta. Outro facilitador do desenvolvimento de jogos é o uso de padrões de projeto da engenharia de *software*. Assim, esse trabalho tem o objetivo de apresentar um relatório sobre o desenvolvimento de um jogo de RPG no Godot. Nessa perspectiva, buscou-se a identificação de pontos positivos e negativo do Godot, a identificação de padrões de projeto, e os aprendizados e limitações adquiridas durante o desenvolvimento. É importante, para a validação do jogo com os jogadores, a realização de testes externos, algo que foi abordado no presente trabalho.

Palavras-chave: Jogos digitais de RPG. Motor de jogo. Godot. Padrões de projeto.

ABSTRACT

Digital role-playing games are complex because they have several characteristics, they are among among the most complex genres in the gaming market. For this, the use of game engines, tools that facilitate the development of games, is very useful for the the creation of games of this genre. Recently, the game engine Godot has become widely used by independent developers for a number of advantages that it presents. Another facilitator of game development is the use of design patterns from software engineering. Thus, this paper aims to present a report on the development of a on the development of a role-playing game in Godot. In this perspective, we sought to the identification of positive and negative points of Godot, the identification of design patterns and the learnings and limitations acquired during the development. It is important, for the validation of the game with the players, to perform external tests, something that was addressed in this work.

Keywords: Digital RPG games. Game engine. Godot. Design patterns.

SUMÁRIO

1 INTRODUÇÃO

Jogos de RPG (*role-playing game*) consistem na criação de uma narrativa construída por um mestre e um ou mais jogadores. O mestre irá organizar os detalhes gerais da narrativa e conduzi-la durante o jogo, enquanto que os jogadores irão vivenciá-la e tomar as decisões que podem encaminhar a história para diferentes caminhos. Dessa forma, o gênero de RPG para jogos digitais busca simular as características do jogo de mesa, podendo adaptar apenas mecânicas básicas ou até mesmo tentar recriar uma experiência similar. Além disso, segundo o site How To Market A Game[1], de ZUKOWSKI (2022), especializado em análise e marketing, jogos de RPG para computador (CRPG) estão entre os mais rentáveis na plataforma Steam[2], como visto na Figura 1, que apresenta o número de jogos lançados de cada gênero, representado pelas colunas e valores a direita, por a mediana dos ganhos de venda, representada pela linha e valores a esquerda. Assim, os gêneros a esquerda do gráfico são mais rentáveis, enquanto os a direita são menos. Isso demonstra uma tendência e busca dos jogadores pelo gênero de RPG, além de uma oportunidade para os desenvolvedores.

Figura 1 – Número de jogos lançados por ganho médio

Fonte: ZUKOWSKI (2022).

Jogos digitais de RPG comumente são complexos, por tentarem adaptar um jogo de mídia diferente e por normalmente possuírem diversas mecânicas e sistemas

[1] https://howtomarketagame.com/
[2] https://store.steampowered.com/

complexos. Como exemplo de mecânicas e sistemas, podemos citar a progressão de personagem, a simulação de um mundo de fantasia e o combate contra personagens não jogáveis (NPCs). Além disso, eles podem seguir diferentes vertentes e subgêneros, baseado em Ralph e Monu (2016), como exemplo:

- O JRPG (*Japanese Role Playing Game*), focado em apresentar uma narrativa rica;
- O WRPG (*Western Role Playing Game*), focado em entregar uma maior liberdade ao jogador;
- O ARPG (*Action Role Playing Game*), que normalmente utiliza o combate em tempo real, entre outros.

Assim, é recomendado o uso de uma *game engine* (ou motor de jogo), que é um *framework* com uma série de ferramentas que buscam facilitar o desenvolvimento de jogos, como a detecção de colisão e um motor gráfico de 2D ou 3D. Recentemente, a *engine* Godot[3] é bastante adotada pelos desenvolvedores independentes, como mostram levantamentos da GLOBAL GAME JAM (2020)[4]. Isso ocorre por sua facilidade de aprendizado, por possuir uma linguagem própria baseada em Python, a GDScript, a qual é simples de ser utilizada, e uma comunidade ativa e disposta a desenvolver tutoriais e *plugins*, além de ser gratuita.

Além disso, é importante seguir as boas práticas da engenharia de *software* com a utilização de padrões de projeto, que são formas de organização da arquitetura de *software*. Eles tem como objetivo facilitar o desenvolvimento do projeto, a realização de manutenção e mudanças no código sem a preocupação em prejudicar sistemas implementados. Além disso, proporcionam uma maior escalabilidade de crescimento das *features* e regras de negócio da aplicação.

Diante do exposto, o presente trabalho tem como objetivo geral a realização de um relatório sobre o desenvolvimento do jogo digital de RPG Lost Borders no Godot.

Para tanto, tem-se os seguintes objetivos específicos: relatar a experiência do desenvolvimento de um jogo digital de RPG no Godot utilizando padrões de projeto e identificar as facilidades e dificuldades de desenvolver jogos de RPG utilizando o Godot.

[3] https://godotengine.org/
[4] https://globalgamejam.org/

Esse projeto de jogo foi concebido por uma equipe de alunos do curso de Sistemas de Mídias Digitais que se reuniram para formar o estúdio de jogos Guildsight[5].

Esse documento é dividido em 5 capítulos. O segundo capítulo apresenta o referencial teórico, explicando as principais características do gênero de RPG, das *game engines*, com foco no Godot e uma lista de padrões de projeto. O terceiro capítulo explica a metodologia do trabalho, listando os padrões de projetos que foram utilizados, descrevendo as características do Lost Borders e citando as partes do desenvolvimento. O quarto capítulo demonstra os resultados obtidos durante o desenvolvimento do projeto, com a apresentação de aprendizados e limitações adquiridas durante o desenvolvimento. Finalmente, o quinto capítulo traz a conclusão do trabalho.

[5] https://www.guildsight.com/

2 REFERENCIAL TEÓRICO

Esse capítulo tem o objetivo de explicar por que os jogos de RPG são complexos e como as *game engines* são utilizadas como facilitadoras para o desenvolvimento de jogos. Assim, serão exploradas as características do gênero de RPG, a utilidade das *game engines*, em particular a Godot, e os principais padrões de projeto úteis para o desenvolvimento desses jogos.

2.1 Características do RPG

O RPG de mesa se popularizou com o lançamento de Dungeons and Dragons (D&D)[1] em 1974, apresentando um lista de regras e mecânicas próprias. Pela alta popularidade e profundidade das mecânicas, os estúdios de jogos perceberam uma boa oportunidade de adaptarem as características do RPG de mesa para o digital, o que ocasionou a criação dos jogos digitais de RPG.

Um *videogame* do gênero RPG normalmente é caracterizado por simular elementos do jogo de mesa, como a presença de um sistema de regras, o desenvolvimento de um mundo com sua narrativa e a imersão do jogador dentro desse universo. Para cumprir esse objetivo, com base nos trabalhos de Stenström e Björk (2013) e Ralph e Monu (2016), o jogo digital de RPG pode ter as seguintes características:

- Atributos, como força, constituição e inteligência, que são utilizados para definir em que áreas o personagem do jogador tem vantagens e desvantagens. Por exemplo, um personagem com muitos pontos de inteligência e poucos de força e constituição poderia ser um poderoso conjurador de magias (inteligência), mas provavelmente teria deficiências em habilidades físicas (força) e pouca resistência contra ataques inimigos (constituição);
- Habilidades, que normalmente podem ser ofensivas ou defensivas, utilizam os valores dos atributos do personagem para definirem o seu nível de poder;
- Progressão de personagem, que acontece normalmente com o jogador acumulando pontos de experiência ao completar objetivos do jogo, po-

[1] https://dnd.wizards.com/

dendo resultar no incremento dos atributos e ganho de novas habilidades;

- NPCs (personagens não jogáveis), personagens que o jogador não controla, mas que tem algum tipo de importância na narrativa, podendo ser comerciantes, entregadores de missão, inimigos, ou até mesmo um simples aldeão que dará um conselho ao jogador;
- Combates, um dos principais momentos em que o jogador utilizará suas habilidades contra inimigos. Podem ser de diversos tipos, como turno, tempo real, ou tempo real com pausa;
- Itens, como uma poção para o jogador recuperar pontos de vida (item consumível), uma espada para o jogador utilizar para atacar os inimigos (item equipável), ou uma chave mágica que será usada para abrir uma porta secreta ao longo de uma missão (item de missão);
- Interação e escolhas narrativas, desde entrar em um diálogo com um NPC, até escolher opções de resposta durante o diálogo que podem alterar completamente a conclusão da narrativa;
- Missões, sequência de objetivos que o jogador adquire durante o jogo, podendo ser relacionadas diretamente à narrativa principal (missão principal), ou uma missão opcional que dá uma recompensa extra ao jogador (missão secundária);
- Exploração, que acontece no mundo ao qual o jogador se encontra, possibilitando encontrar NPCs, itens e inimigos;
- Quebra-cabeças, desafios que normalmente surgem durante a exploração e requerem sua conclusão para que o jogador possa prosseguir com a exploração ou até mesmo ganhar uma recompensa.

Cada característica tem suas próprias particularidades e normalmente estão interligadas, podendo seguir diferentes caminhos dependendo do design do jogo. Dois jogos de RPG podem ter mecânicas bastante diferentes, por exemplo no combate um utilizar turnos e o outro tempo real, mas normalmente apresentam características gerais em comum. Como exemplo, ambos podem utilizar atributos para balancear os seus combates distintos. Dois jogos que se encaixam nos exemplos anteriores são Baldur's Gate 2[2] e sua continuação Baldur's Gate 3[3]. Ambos são baseados no sistema

[2] https://store.steampowered.com/app/257350/Baldurs_Gate_II_Enhanced_Edition/
[3] https://baldursgate3.game/

de RPG Dungeons and Dragons, na 2ª edição e 5ª edição, respectivamente, e, apesar de utilizarem diferentes versões para desenvolverem seus sistemas de jogo, ambas têm várias semelhanças. Assim, seria natural que ambos os jogos apresentassem a maior parte de suas mecânicas parecidas, o que até certo ponto é verdade, divergindo a mecânica anteriormente exposta: o combate.

Figura 2 – Baldur´s Gate 3

Fonte: steampowered (2022).

O RPG de mesa Dungeons and Dragons apresenta um combate em turno, onde cada jogador tem sua vez para realizar suas ações, característica essa que se manteve em todas as versões posteriores. Todavia, ao ser adaptado para o digital, por meio da franquia Baldur's Gate, foi utilizada outra abordagem, o combate de tempo real com pausa, onde o jogador pode pausar a qualquer momento para pensar e escolher suas ações. O balanceamento é definido com o mesmo sistema de atributos, habilidades, itens, classes e inimigos do sistema de D&D. Já Baldur's Gate 3 busca se assemelhar com o RPG de mesa, utilizando assim o combate de turnos, como mostrado na Figura 2, onde é apresentada a ordem de ação dos personagens no canto superior esquerdo, e mantendo características de balanceamento similares aos jogos anteriores da franquia.

Dessa forma, este trabalho buscará focar nas características gerais que compõem um RPG, as quais normalmente são encontradas na maior parte dos jogos,

mas que podem ser utilizadas para o desenvolvimento de características específicas.

Além disso, segundo Onuczko *et al.* (2005), normalmente em um jogo do gênero são implementados quatro tipos diferentes de padrões de design: diálogos, encontros, comportamentos e enredo. Os padrões de diálogos podem ser utilizados para iniciar conversas com NPCs e escolher respostas durante o diálogo. Os padrões de encontros servem para iniciar eventos e interagir com objetos inanimados no cenário, como uma alavanca ou baú. O comportamento serve para criar vida aos elementos do mundo, principalmente os inimigos que enfrentarão o jogador. E os padrões de trama servem para guiar o jogador na narrativa do mundo, por exemplo implementando um sistema de missões.

2.2 *Game engine*

Os jogos de RPG são bastante complexos, exatamente por necessitarem implementar uma série de mecânicas que tem o potencial de seguirem por diferentes caminhos, normalmente precisando ter ligações entre elas. Além disso, existe a necessidade de balanceamento durante todas as partes, tornando difícil o desenvolvimento desse tipo de jogo. Essa é a importância de utilizar uma *game engine*, programa com o objetivo de facilitar o desenvolvimento e manutenção durante todo o processo de desenvolvimento. Elas possuem uma série de ferramentas disponíveis que servem para desenvolver elementos normalmente presentes na criação de jogos, segundo Valencia-García *et al.* (2016), como:

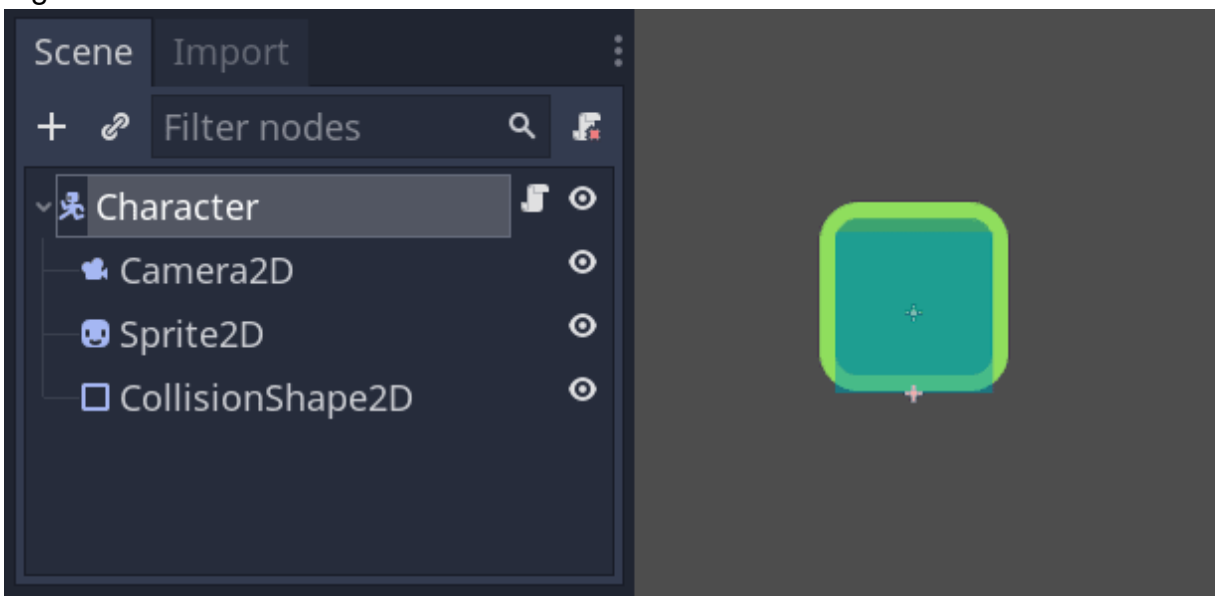
- Sistema de física e colisão;
- Renderização 2D e 3D;
- Suporte a som;
- Suporte a animação;
- Sistema de Inteligência Artificial;
- Suporte a programação em uma determinada linguagem;
- Representação visual das estruturas de programação orientada a objetos.

Em relação a *game engine* Godot, é utilizada a linguagem de programação GDScript, a qual é baseada em Python, isto é, uma linguagem de tipagem dinâmica, que por definição significa que o próprio interpretador irá definir os tipos de variáveis, sem precisar que o programador as declare. Assim, é uma responsabilidade e boa

prática, mas não uma obrigação para o código funcionar, do programador organizar os tipos de variáveis e definir as suas permissões de acesso. As vantagens do GDScript são, segundo a GODOT COMMUNITY (2022), ser uma linguagem com fácil curva de aprendizado e, por ser integrada a *engine*, permite uma maior velocidade de desenvolvimento.

Além disso, o desenvolvimento no Godot é baseado em uma estrutura de *nodes*, objetos que podem conter diferentes atributos e métodos destinados a um funcionamento em específico. Eles podem ser filhos de outros *nodes* e assim formar uma árvore de elementos, a qual é nomeada de *scene*. Esses *nodes* podem ter diferentes tipos, conter diversas informações e normalmente representam os objetos da programação orientada a objetos (POO). O interessante é que as formações das árvores de *nodes* podem ser utilizadas para representar as hierarquias dos sistemas de RPG. Por exemplo, um *node* pai pode representar um personagem que tem habilidades e itens representados por seus *nodes* filhos.

Figura 3 – Estrutura de *nodes*



Fonte: GODOT COMMUNITY (2022).

A Figura 3 apresenta um exemplo de estrutura de *nodes*. Primeiro é criado o node pai "Character", que representa o personagem, e é atribuido a ele um script onde será programado o seu funcionamento primário. "Character" recebe como *nodes* filhos "Camera2D", que representa a câmera de visão do jogador que seguirá o personagem, "Sprite2D", a ilustração que representa o personagem, e "CollisionShape2D", que fica

responsável por detectar a colisão do personagem. Assim, é criada uma arvore de *nodes* fácil de visualizar, com algumas características de um node pai divididas entre os *nodes* filhos.

2.3 Padrões de projeto

O Godot funciona seguindo alguns padrões da engenharia de *software*, como a tríade da programação para jogos, apresentada por Nystrom (2014), a qual é:

- *Gameloop*, que significa a sequência de ações que serão executadas ao longo do jogo de forma repetida;
- *Update*, as atualizações de variáveis feitas durante o passar do tempo no jogo;
- Componentes, uma série de características e comportamentos que podem ser atribuídas a um objeto.

Assim, o Godot tem suas próprias formas de representar esses 3 elementos. O *gameloop* é representado por meio das funções _process(), processamento ocioso (*idle*) que atualiza o *node* a cada quadro com a maior frequência possível, e _physics_process(), processamento físico (*physics*) que ocorre em uma taxa fixa (por padrão 60 vezes por segundo). O *update* ocorre durante o processamento dos *nodes* e os componentes são representados por *nodes* filhos. Com isso em mente, uma boa prática é utilizar o *gameloop* de processamento físico para a implementação do funcionamento dos objetos e o *gameloop* ocioso para a renderização de texturas e animações. Isso evita *bugs* que podem ocorrer em *hardwares* com processamentos diferentes daquele onde o jogo foi desenvolvido.

Com base em Qu *et al.* (2013) e LOVATO (2021), os seguintes padrões de projeto são úteis para o desenvolvimento dessas estruturas presentes nos jogos de RPG: Entity Component System (ECS), Observer (Signal, na Godot) e Mediator. Para a criação da inteligência artificial dos inimigos, a Behavior Tree.

2.4 Entity Component System (ECS)

Entity Component System (ECS), arquitetura de *software* que segue o princípio de que cada elemento do jogo é uma entidade e cada característica, comportamento

ou funcionalidade é um componente. Esse padrão é representado no Godot pelo sistema de *nodes*. Segundo Lord (2012), o Entity Component System é útil para:

- Tornar as diversas estruturas do jogo independentes, permitindo elas funcionarem de forma simultânea;
- Salvar e carregar o estado do jogo, pois é facilitado por a maior parte das informações ficarem guardadas em componentes das entidades. Normalmente essas informações são repassadas para um arquivo JSON, durante o salvamento, e retornadas do arquivo para os componentes das entidades, durante o carregamento;
- Criar estruturas de código descentralizadas, permitindo elas serem mais flexíveis.

2.5 Observer

O Observer, chamado de Signal na Godot, é outro padrão da arquitetura de software que busca definir uma dependência entre objetos observadores ao ponto de fazer eles reagirem quando determinada condição de outro objeto for realizada, enviando assim um sinal para os observadores, segundo Nystrom (2014). O fluxo entre o objeto que envia o sinal e os observadores é essencial para o funcionamento desse padrão. O uso do Observer é essencial em todo tipo de sistema ao qual há extrema importância na comunicação entre objetos que for sucedida com respostas em sequência. Normalmente é utilizado na comunicação entre o sistema interno do jogo e a interface.

2.6 Mediator

O Mediator consiste na criação de um objeto que irá mediar a comunicação entre dois sistemas. Segundo Gamma *et al.* (1995), o Mediator é útil quando:

- Um conjunto de objetos se comunica de forma definida, mas complexa;
- É difícil reutilizar um objeto por ele se referir e se comunicar com vários outros objetos;
- Um comportamento presente em várias classes deve ser modificável sem muitas subclasses;

Um exemplo de situação em que ele é bastante usado é na comunicação entre dados de personagem e interface. Normalmente, vários dados do jogador são apresentados constantemente na interface, o que requer uma boa comunicação entre o objeto que armazena os dados e a interface, para haver uma atualização efetiva.

2.7 Behavior tree

A *behavior tree* é um modelo matemático acíclico que organiza uma árvore de tarefas, utilizado no desenvolvimento de inteligencia artificial em um jogo. Elas são construídas utilizando blocos que podem ser de diferentes tipos e que durante a sua execução podem retornar um resultado positivo ou negativo, segundo Colledanchise e Ögren (2018). Os tipos de bloco são os seguintes:

- Sequencer, todos os blocos filhos de um *sequencer* são executados em sequência, a medida em que eles retornem um resultado positivo. Caso um dos filhos retorne um resultado negativo, a sequência é interrompida e o *sequencer* irá retornar negativo. Caso todos os filhos retornem positivo, o *sequencer* irá completar a sequência e retornar positivo;
- Selector, todos os blocos filhos de um *selector* são executados em sequência, a medida em que eles retornem negativo. Caso um dos filhos retorne um resultado positivo, a sequência é interrompida e o *selector* irá retornar positivo. Caso todos os filhos retornem negativo, o *selector* irá completar a sequência e retornar negativo;
- Decorator, é uma ação ou condição especial que pode ser realizada baseada no retorno do bloco filhos. Por exemplo, um *decorator* do tipo *revert* irá reverter o retorno que receber do bloco filho, ou seja, se receber uma falha irá trocá-la para sucesso;
- Leaf, é a ação que será realizada. Nunca terá outro bloco como filho.

3 DESENVOLVIMENTO

Como discutido anteriormente, jogos de RPG são complexos, por possuírem uma vasta gama de características, e por isso o seu desenvolvimento se beneficia do uso de *game engines*. Uma da principais utilizadas recentemente é a Godot, que é gratuita, tem uma linguagem com fácil curva de aprendizado e uma comunidade ativa de desenvolvedores. Entendendo a complexidade do gênero de RPG e a utilidade das *game engines* e dos padrões de projeto, neste capítulo serão apresentados a metodologia do presente trabalho, a descrição do jogo que foi desenvolvido e detalhes do desenvolvimento da estrutura geral, dos menus e interfaces, da exploração, do uso de itens, do combate, do uso de habilidade, do comportamento dos inimigos utilizando *behavior tree* e das ilustrações.

3.1 Metodologia

Para desenvolver um jogo de RPG, foi utilizada a Godot, em sua versão mais recente, 3.5, com o padrão de arquitetura de *software* Entity Component System (ECS). A escolha da *engine* foi realizada por causa das vantagens que ela apresenta, além do interesse por parte da equipe de testar um motor de jogo que está crescendo na comunidade de desenvolvedores, para entender na prática as suas utilidades. Toda a implementação do jogo foi desenvolvida buscando utilizar as boas práticas da engenharia de *software*. Dessa forma, as partes relacionadas a funcionamento e comportamento físico foram construídas utilizando o *gameloop* físico (*physics*). Já as partes relacionadas a renderização de *sprites* e animação foram feitas utilizando o *gameloop* ocioso (*idle*).

Além disso, a notificação em relação às mudanças de variáveis e estados dos *nodes* foi realizada pela utilização do Observer. A fim de implementar a comunicação entre sistema e interface foi utilizado o padrão Mediator. Assim, utilizando o Observer e o Mediator, foram implementadas as interações de enredo, por meio da criação de um sistema de *quests*, e as de encontro, onde o jogador interage com um objeto do cenário.

Outro fator importante é a implementação da inteligência artificial dos inimigos, que foi realizada pela utilização da Behavior Tree, pois tem a vantagem de

ser flexível e capaz de organizações complexas compostas por tarefas simples, mas sem a necessidade de saberem como as tarefas simples são implementadas. Para implementar as *behavior trees* no Godot, foi utilizado o *plugin* Godot Behavior Tree[1], criado pela comunidade, ao qual adiciona uma série de tipos de *nodes* que representam os elementos da *behavior trees*. Assim, foram criadas árvores de *nodes* com a mesma estrutura das *behavior trees* planejadas para o jogo.

Os diálogos foram implementados utilizando o *plugin* Dialogic[2], feito pela comunidade, que se baseia na criação de uma árvore de conversa, uma sequência de diálogos que permite a criação de diferentes caminhos de interação baseados nas respostas que os jogadores venham a escolher.

Ao final do desenvolvimento, foi realizada uma pesquisa utilizando o PANAS (Positive and Negative Affect Schedule), questionário que busca medir o efeito positivo e negativo do objeto analisado, para medir a aceitação com o público geral do jogo desenvolvido, baseado em Watson e Clark (1994). Os participantes respondem os níveis, em escala de 5, das emoções que tiveram durante o jogo, listadas no questionário. Depois é feita uma média das pontuações das emoções positivas e negativas, que podem variar de 10 a 50, para assim medir o nível de percepção positiva e negativa. Foi utilizada a tradução do PANAS apresentada por Galinha e Pais-Ribeiro (2005). O questionário usado é apresentado no Apêndice A.

3.2 O jogo Lost Borders

Já em relação ao Lost Borders, ao qual sua logo se encontra na Figura 4, foi desenvolvido inicialmente para PC pelo estúdio Guildsight. O jogo tem a seguinte sinopse: “Entre nessa jornada misteriosa e perigosa para derrotar um mal antigo com diferentes combinações de feitiços em combate de *grid* em tempo real e quebra-cabeças interessantes espalhados pelo mundo. Esteja pronto para enfrentar e superar grandes desafios nesta aventura de pixel *art* de RPG!”. Assim, ele é um *Action* RPG e tem o combate em tempo real com movimentação em *grid* baseado no jogo Megaman Battle Network[3], onde o jogador precisa usar suas habilidades para poder derrotar os adversários em um combate estratégico.

[1] https://github.com/kagenash1/godot-behavior-tree
[2] https://github.com/coppolaemilio/dialogic
[3] https://en.wikipedia.org/wiki/Mega_Man_Battle_Network

Figura 4 – Logo do Lost Borders

Fonte: Compilação do autor.

A exploração é *top-down* (visão da câmera de cima para baixo), baseada nos jogos de Zelda do mesmo estilo, como exemplo o Minish Cap[4], onde o jogador tem que completar quebra-cabeças, acionar mecanismos para liberar passagens e utilizar itens específicos para avançar em determinadas partes. O jogo tem um foco narrativo, inspirado por Undertale[5], em que o jogador precisa interagir com personagens e tomar decisões que podem impactar na narrativa principal. Basicamente, o jogador precisa interagir com personagens ao redor dos mundos que irão lhe entregar missões e para completá-las é necessário explorar o ambiente, buscar informações com outros NPCs, resolver enigmas e enfrentar as criaturas que surgirem no caminho. Uma vez que a missão é completada, ele receberá uma recompensa e voltará a explorar o mundo até receber uma nova missão. O jogo termina quando o jogador terminar a última missão, que será derrotar um inimigo final.

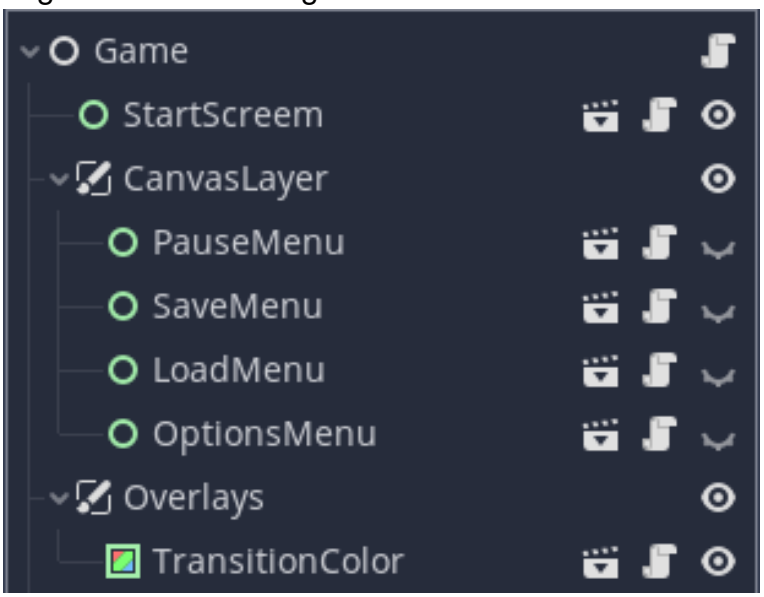
Os membros de equipe que trabalharam no projeto foram Raoni Coelho Vieira, na programação, Daniel Portela Bandeira, na produção e *game design*, Tiago Caúla de Oliveira Maia e Caio Vinicius Bezerra Abreu, na ilustração, Amanda Milena Lima Pereira, na UX e UI, e Angelo Ceccatto Cruz Santana, na música e *audio design*.

[4] https://en.wikipedia.org/wiki/The_Legend_of_Zelda:_The_Minish_Cap
[5] https://store.steampowered.com/app/391540/Undertale

3.3 Estrutura geral

A estrutura geral do projeto foi organizada seguindo o padrão de Entity Component System, como apresentado na Figura 5. O *node* raiz do projeto tem o nome de "Game", ele que guarda as variáveis e funções que precisam ser globais, como transições de cena. Os *nodes* que representam momentos do jogo, como o combate e a exploração, são adicionados como filhos de "Game"no momento em que precisarem ser usados.

Figura 5 – Estrutura geral



Fonte: Compilação do autor.

O *node* "StartScreen"representa o primeiro momento, a tela com a logo do estúdio desenvolvedor. Ao terminar o tempo de apresentação da logo, o *node* "StartScreen"é trocado pelo "StartMenu", e a tela do jogo apresenta o menu inicial como mostrado na Figura 6.

Os *nodes* filhos de "CanvasLayer"são menus que podem ser chamados em diversos momentos do jogo, e tem por característica compartilhada sempre ficam na camada mais a frente da tela. Os *nodes* filhos de "Overlay"são fundos de uma cor que aparecem e desaparecem, podendo ser utilizados em determinadas ocasiões, por exemplo um fundo escuro para a transição de tela.

As informações do jogador são guardadas no *node* "PlayerSheet", que será instanciado como filho de "Game"ao jogo ser iniciado.

Figura 6 – Menu inicial

Fonte: Compilação do autor.

3.4 Menu e interface

A interface foi projetada e montada seguindo cinco objetivos baseados em SPACEY (2018), sendo eles:

- Usabilidade. A interface precisa ser simples e intuitiva, para que o jogador consiga reconhecer os padrões em vez de pensar no que eles significam, tendo assim uma experiência agradável com o jogo;
- Apelo Visual. A interface deve ser visualmente atrativa e chamativa para o jogador com a sua estética voltada para a temática mística;
- Experiência do Jogador. Durante o jogo, a experiência que o usuário deve estar diretamente relacionada com a essência e a temática do jogo;
- Coesão. A interface deve seguir um padrão pré estabelecido dentro da sua temática, sem incongruências entre seus elementos e símbolos;
- Detectabilidade. Deve ser fácil para o jogador identificar o que ele precisa na interface, tendo seus elementos distintos e de fácil percepção.

No Godot, interfaces e menus são construídos utilizando os *node* do tipo "Control", representados na *engine* pela cor verde. Existem vários tipos diferentes de "Controls", que representam os diferentes elementos de uma interface, como *background*, botões, abas, *containers*, etc.

Figura 7 – Grimório de Magia

Fonte: Compilação do autor.

Um exemplo de interface é o grimório de magia apresentado na Figura 7. Os retângulos a esquerda são kits de magia, onde dentro deles estão botões em formato de quadrado. Ao selecionar um desses botões, o jogador precisa escolher uma das magias presentes em botões com ícones do lado direito. Ao selecionar um botão a direita, a magia presente é adicionada no espaço de kit correspondente.

De modo geral, a comunicação entre os elementos da interface e do sistema foi construída utilizando *signals* e o padrão Mediator, que foi representado por funções do *node* "PlayerSheet". Assim, sempre que a interface interagir com as informações do jogador, ela irá enviar um *signal* para o *node* "PlayerSheet", que irá atualizar e mediar as informações do jogador.

Os diálogos foram desenvolvidos com a ajuda do *plugin* Dialogic, que foi configurado para melhor representar o design de interface do jogo. A Figura 8 apresenta o resultado final da caixa de diálogo configurada e implementada.

3.5 Exploração

A partir de uma visão *top-down*, o jogador irá se movimentar em 8 direções para explorar o ambiente ao seu redor. Sempre que ele chegar perto de um elemento interativo (NPC ou objeto) poderá apertar o botão ENTER para interagir. No caso de

Figura 8 – Caixa de diálogo

Fonte: Compilação do autor.

NPC ou objeto, um pequeno ícone aparecerá na cabeça do personagem, indicando que ele está próximo de um objeto interativo. Exemplo demonstrado na Figura 9, onde o personagem do jogador está próximo de um baú. NPCs com missões terão um ícone diferente destacado em cima de suas cabeças.

Figura 9 – Objeto interativo na exploração

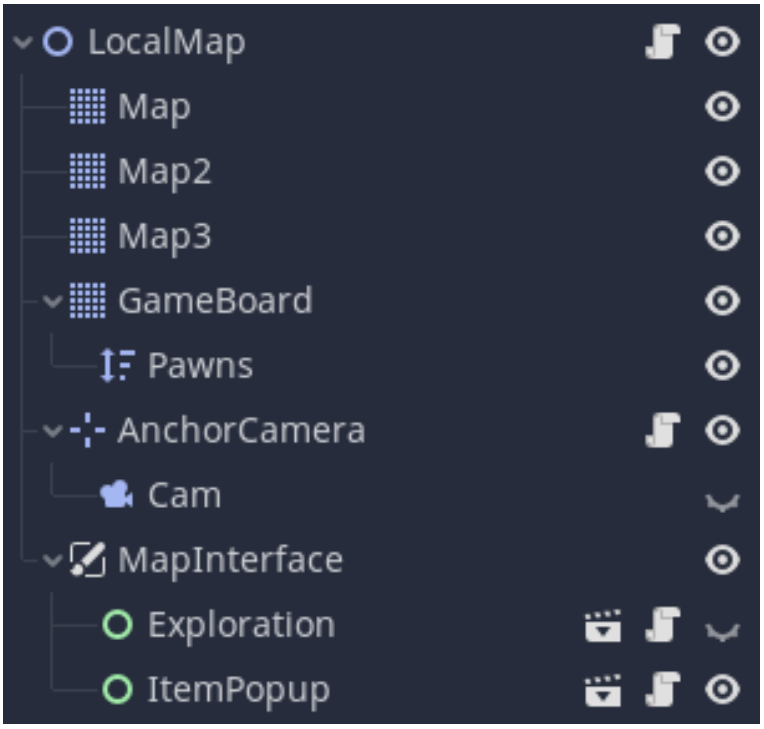
Fonte: Compilação do autor.

O mapa da exploração foi construído utilizando camadas de *tiles*, como apresentado na Figura 10. Nessa estrutura, os *nodes* de nome "Map"e "GameBoard"são do tipo TileMap, que servem para organizar um mapeamento de *tiles*.

Os *nodes* "AnchorCamera"e "Cam"representam a câmera de visão que o jogador terá, normalmente irá acompanhar o personagem principal. Os *nodes* filhos de "MapInterface"são interfaces e *popups* que podem surgir durante o momento de exploração.

Durante a exploração existem vários objetos que o jogador pode interagir, esses que receberam a nomenclatura de "InteractivePawn". Eles tem uma lista de ações, como coletar item, iniciar diálogo, iniciar combate, entre outras, que serão executadas em ordem no momento em que o jogador realizar a interação. Na estrutura da cena de exploração, eles são filhos do *node* "Pawns".

Figura 10 – Estrutura da exploração



Fonte: Compilação do autor.

Durante a exploração o jogador irá receber diferentes missões para cumprir ao longo da sua jornada, sendo elas:

- Missão principal. Missões relacionadas com a história do jogo;
- Missão secundária. Missões espalhadas pelo mundo que não impactam na jornada principal do jogador e possibilitam ele avançar paralelamente.

O jogador sempre irá ver a missão principal que foi dada a ele pela interface de exploração, enquanto pelo grimório ele poderá acessar as missões secundárias.

Ao finalizar uma missão, o jogador poderá ganhar: Pontos de Habilidade(PH)

para aumentar um *status*, dinheiro para gastar na loja, item, magia ou Ponto de Afinidade(PA) com o NPC específico. A recompensa das missões é distribuída de forma que os PH sejam dados principalmente pelas missões principais ou missões secundárias de relevância. Dinheiro e PA sendo recompensas comuns durante as missões, enquanto magias e itens sendo recompensas especiais e específicas. As missões são compostas por pequenas *sub-quests* (no máximo 4), que são sequências na sua progressão.

Além disso, o sistema de missões foi programado com cada missão podendo estar entre 4 estados:

- "Available", a missão pode ser iniciada pelo jogador ao realizar determinada ação;
- "Active", a missão já foi iniciada e será concluída ao jogador realizar os seus objetivos;
- "Completed", os objetivos da missão foram concluídos e o jogador precisa entregá-la para obter sua recompensa;
- "Delivered", a missão foi entregue e o jogador recebeu sua recompensa;

3.6 Itens

Durante a exploração, o jogador pode obter itens por meio de lojas com NPCs e encontrando pelo mapa em baús a serem coletados. A Figura 11 apresenta o inventário de itens com uma poção selecionado e um colar equipado, que acrescenta ao jogador um bônus de vida.

Os itens podem ser dos seguintes tipos:

- Equipáveis, tipo de item que concederá um bônus para o jogador e são únicos, podendo equipar no máximo 3 itens ao mesmo tempo no seu inventário;
- Consumíveis, tipo de item que pode ser acumulado pelo jogador, em um conjunto equivalente a no máximo 32 itens repetidos. Itens consumíveis podem ser usados durante o combate ou exploração;
- Especiais, itens de missão que não podem ser vendidos na loja e só saem do inventário do jogador no momento que ele completa a missão.

Com exceção de itens especiais, qualquer outro item pode ser vendido, a

Figura 11 – Inventário de itens

Fonte: Compilação do autor.

menos que sua descrição diga o contrário.

3.7 Combate

Enquanto o jogador explora o mapa, haverá zonas de perigo onde existe a chance de 10% x (multiplicador de combate) a cada passo que ele realizar dentro das zonas para entrar em combate. O combate do jogo é estruturado em uma *grid* padrão de tamanho 6x3, que é dividida ao meio entre área do jogador e inimigos, como demonstrado na Figura 12. A depender da fase, a *grid* pode variar ou ter elementos a mais. Quando um encontro de combate for gerado, o sistema do jogo irá inicialmente selecionar uma combinação de monstros de uma lista pré-estruturada, para logo em seguida montar a *grid*, definir os modificadores de combate e posicionar os monstros e o jogador na *grid*.

Durante o combate, o jogador poderá se movimentar nas 4 direções, apenas em quadrados da *grid* que possuem a cor do seu campo e que não estão ocupados por outras entidades ou objetos. Além disso, o jogador terá acesso a um *kit* de magias que ele irá montar previamente. Durante o combate, ele usará as magias alocadas nos botões Q, W e E, enquanto usa o botão R do teclado para poder alternar entre os *kits* em um menu circular que se expande do jogador. Enquanto estiver selecionando o

Figura 12 – Combate

Fonte: Compilação do autor.

kit, o jogador e a interface irão ficar em destaque e todos os elementos externos irão entrar em câmera lenta, como demonstrado na Figura 13. O jogador só poderá trocar de *kit* novamente em 1 segundo e terá 6 segundos para escolher, caso contrário o círculo fecha sozinho. Ao usar uma magia, o jogador tem uma quantidade de mana consumida equivalente ao nível da magia usada e sua mana irá se regenerar 0.25 a cada 1 segundo.

Ao fim do combate, caso o jogador vença, irá surgir a tela de resultado e haverá a chance de adquirir novos itens, como poções de cura. Em caso do jogador ser derrotado, ele irá retornar para o último *checkpoint* do jogo.

Em relação a estrutura no Godot, o *node* "Combat"controla as principais funções da cena, como inicialização e encerramento. O *grid* é dividido em 2, um para o jogador e um para os inimigos, onde os *nodes* de ambos são filhos do *node* "Grids". O *node* "Map", que é do tipo Sprite, é utilizado para carregar a imagem de fundo da cena. Ao final estão presentes os *nodes* de interface.

Cada personagem do combate é tratado como uma unidade. Cada unidade tem um conjunto de atributos, principalmente seu valor de vida, que indica o quanto de dano a unidade precisa adquirir para morrer. Além disso, as unidades têm o seu próprio conjunto de animações de movimento e ataque. Os *nodes* de unidade tem vários nodes

Figura 13 – Círculo de troca dos *kits* de magia

Fonte: Compilação do autor.

filhos que representam suas características. As ações que a unidade pode realizar durante o combate ficam presentes em *nodes* filhos do *node* da unidade "Actions"e normalmente invocam habilidades anteriormente armazenadas no *node* da unidade "Skills". Essas habilidades são salvas inicialmente em objetos do Godot chamados de "Resources", que são basicamente objetos que armazenam dados reutilizáveis, e em seguida armazenadas em *nodes* filhos de "Skills".

3.8 Magias

As magias são as habilidades que o jogador adquire durante a sua aventura, usando-as nos combates para derrotar seus inimigos. O jogador irá adquirir novas magias quando completar *quests*, sendo recebidas como recompensa, ou então a partir de baús que ele encontra durante a exploração. Os baús terão magias pré-selecionadas dentro deles para serem coletadas.

Cada magia e criatura dentro do universo do jogo possui um elemento atrelado a ele, contendo resistência ou fraqueza a determinado elemento, sendo eles: Fogo, Água, Natureza, Vento e Terra. Na Figura 14 o jogador utiliza a magia Tornado de Veneno, que tem o elemento da natureza.

Figura 14 – Jogador usando a magia de Tornado de Veneno

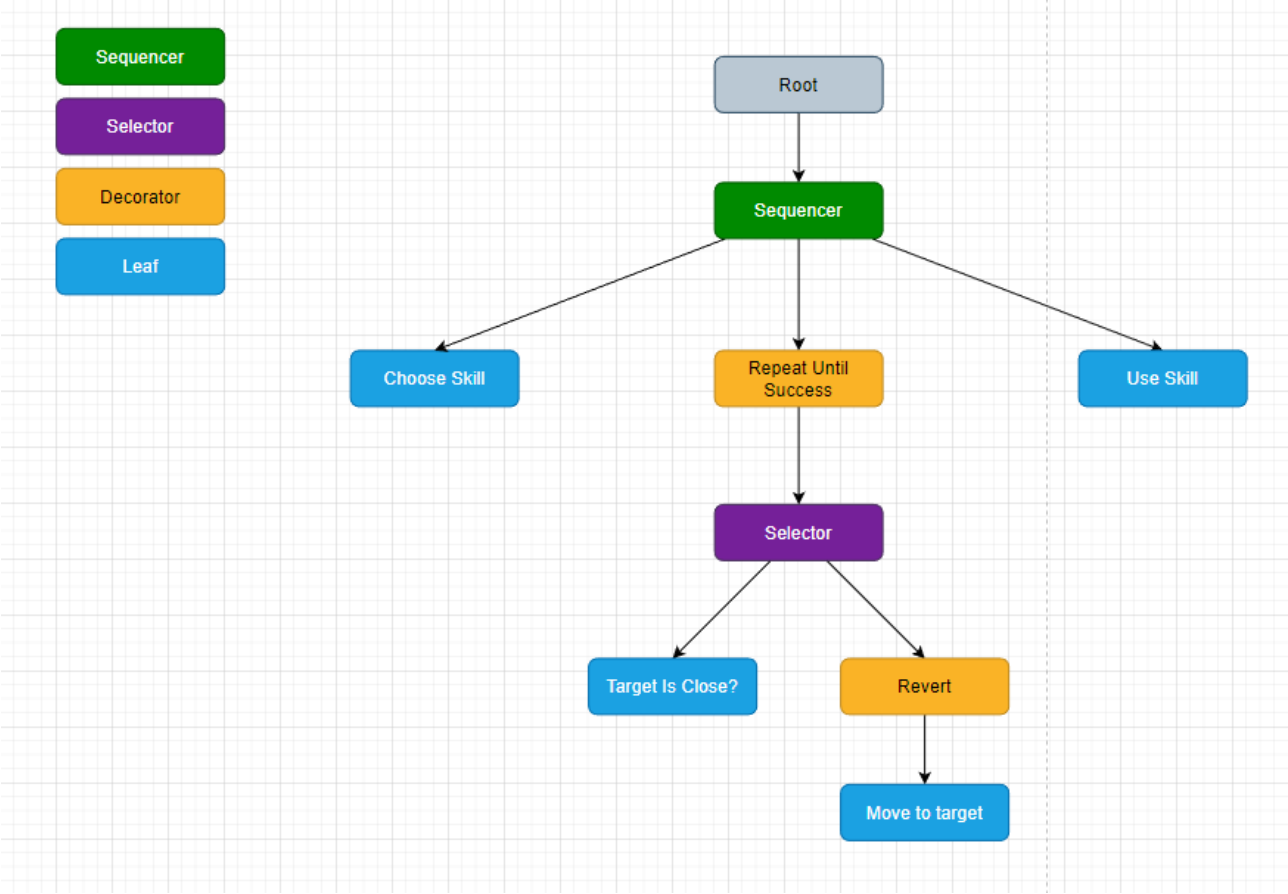
Fonte: Compilação do autor.

3.9 Comportamento de inimigos

Inicialmente, a *behavior tree* dos inimigos foi esquematizada fora do Godot, como apresentado na Figura 15. A primeira ação da sequência será o inimigo escolher uma habilidade, durante a *leaf* "Choose Skill", que é realizada de forma aleatória. Após a habilidade ser escolhida com sucesso, o inimigo entrará em uma sequência para verificar se ele está a uma distância miníma de acertar a habilidade no alvo (por meio do leaf "Target Is Close?") e, caso não esteja, se aproximar do alvo (por meio do leaf "Move to target"). Ele só sairá dessa sequência quando conseguir ficar na distância certa do inimigo graças ao *decorator* "Repeat Until Success", ou seja, quando a ação de "Target Is Close?" retornar positivo. Caso a ação de "Move to target" retorne positivo, não será tratado como um retorno positivo pelo *decorator* "Repeat Until Success", por causa do outro *decorator* "Revert", que reverte o resultado do bloco filho, resultando na repetição da sequência. Ao estar na distância miníma, o inimigo irá usar a habilidade na *leaf* "Use Skill".

Um exemplo de monstro que usa esse comportamento é o *goblin*, presente na Figura 16. É uma pequena criatura humanoide verde que carrega em suas costas uma cesta de bombas e possui uma aparência agressiva e feroz. Possui 60 pontos de vida e se movimenta apenas no eixo Y de forma agressiva, tentando sempre se alinhar

Figura 15 – *Behavior tree* dos inimigos



Fonte: Compilação do autor.

Figura 16 – *Goblin* preparando para arremessar uma bomba

Fonte: Compilação do autor.

com o jogador para acertá-lo com seus ataques. O *goblin* possui o ataque de bomba explosiva, que tem 3 segundos de *cooldown* e é arremessada na posição do jogador,

causando 25 de dano caso acerte.

3.10 Ilustração

O estilo do jogo seguiu a linha de 32bits inspirado nos jogos de GameBoy Advance (GBA), usando esse valor como o módulo para calcular a proporção e tamanho de outros elementos.

As principais referências visuais para a exploração do jogo são Zelda Minish Cap, Pokémon e Stardew Valley, onde os personagens possuem um tamanho de 1 x 1 módulo, onde os mais simples se destacam com alguma característica marcante, já as árvores do jogo sendo mais largas e ocupando 2 x 3 módulos e com uma distribuição de cores bem vibrante e colorida. Enquanto que para o combate, afs principais referências usadas são os jogos Megaman Battle Network e One Step From Eden, por seu estilos trazerem personagens mais detalhados, deixando uma região no fundo em torno de 25% para um background do ambiente onde o combate está ocorrendo.

Figura 17 – *Moodboard* do mundo

Fonte: Compilação do autor.

Para criar o mundo, foi montado um *moodboard*, como mostrado na Figura 17. Nele é bem destacado o ambiente de floresta, com algumas construções, personagens e monstros. Todos os assets utilizados no jogo foram desenvolvidos pela equipe ou comprados em pacotes na internet.

4 RESULTADOS

Este capítulo de resultados será dividido em quatro partes. A primeira com os resultados dos testes utilizando o PANAS (Positive and Negative Affect Schedule), com o gráfico das respostas dos usuários e uma análise. A segunda com pontos positivos e negativos do Godot observados ao longo do desenvolvimento do jogo, focando em aprendizados adquiridos e limitações observadas. A terceira demonstrando como foi a participação da equipe, com foco para o que ajudou e atrapalhou o desenvolvimento. A quarta e última com algumas decisões e mudanças importantes de design.

Ao final do desenvolvimento do jogo, foi gravado um vídeo de demonstração[1].

4.1 Resultado do PANAS

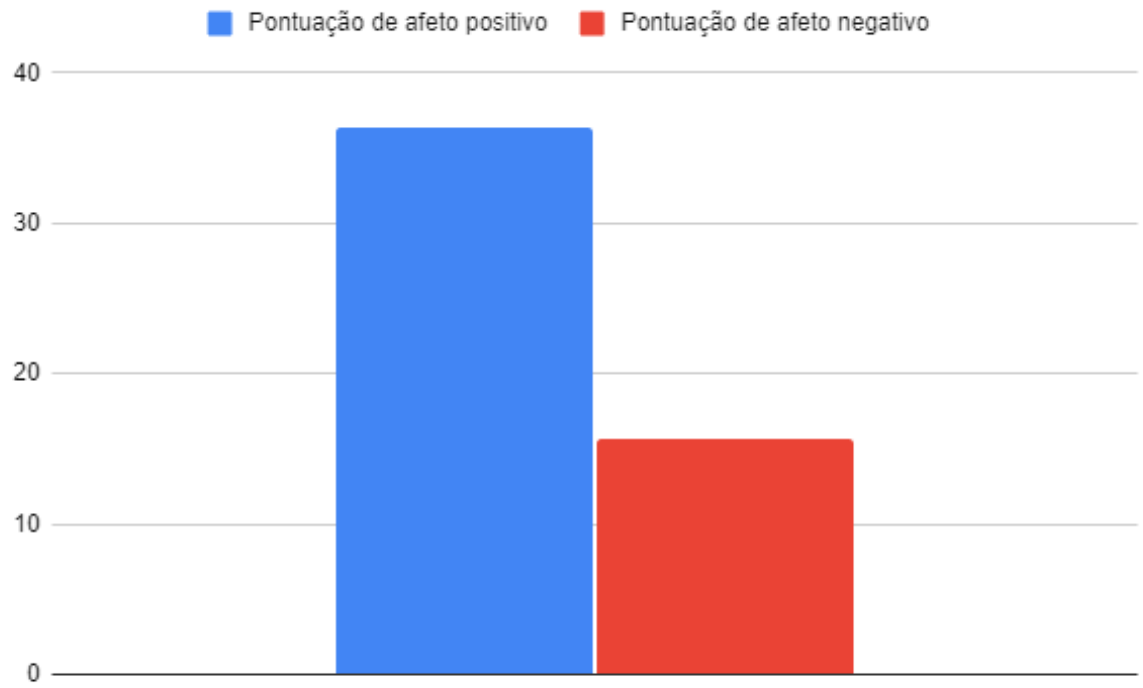
Foram realizados testes com 9 usuários, a maioria jovens na faixa etária entre 18 e 26 anos, com uma parcela de alunos do curso de Sistemas e Mídias Digitais. O recrutamento foi realizado de forma presencial e online. Ao final, o PANAS apresenta uma pontuação positiva e uma negativa, ambas podendo variar de 10 a 50. Quanto maior a pontuação positiva, maior foi a percepção positiva média dos usuários, e quanto menor for a pontuação negativa, menor foi a percepção negativa média dos usuários. O objetivo ideal que indica um resultado geral positivo é a pontuação positiva ser igual ou maior que 33,3 e a pontuação negativa ser igual ou menor que 17,4, segundo Watson e Clark (1994). Os resultados apresentados estão presentes na Figura 18, com a média da pontuação positiva igual a 36,3 e a média da pontuação negativa igual a 15,6. Assim, essa pontuação reflete de modo geral uma aceitação positiva do jogo. Muitos jogadores ficaram atraídos pelo visual do jogo, o que ocasionou no interesse inicial para jogar.

Foi identificado alguns usuários com altos valores em emoções negativas específicas, como de irritação. Conversando de forma aberta com os entrevistados e observando eles jogarem, chegou-se a conclusão que isso ocorreu por causa de algumas falhas presentes nessa versão do jogo, como um tutorial ineficiente, a alta dificuldade no inicio e a excessiva aparição de inimigos durante os momentos de exploração.

Assim, os testes com o PANAS validaram que o jogo tem uma aceitação

[1] https://drive.google.com/file/d/1E9NXGIQVIZtdrBRYNn8dQP3euuMGuRui/view?usp=share_link

Figura 18 – Resultado do teste PANAS



Fonte: Compilação do autor.

positiva com o público em geral, mas que existem pontos específicos que causam emoções negativas e que devem ser melhorados. Algo interessante é que mesmo os usuários com um maior grau de emoções positivas marcavam também algumas emoções negativas, algo que era mais difícil de detectar simplesmente observando o usuário jogar, mas graças ao questionário PANAS isso foi possível. Dessa forma, é recomendado o uso do PANAS para catalogar, de forma mais fácil e detalhada, as emoções dos jogadores.

Uma observação a se atentar em relação ao PANAS é se sua classificação de emoções positivas e negativas sempre será condizente para todo tipo de contexto relacionado a jogos. Por exemplo, a emoção "assustado" não é necessariamente negativa em um contexto de jogo de terror. Assim, durante os testes, vários jogadores que se envolveram positivamente com o jogo, e se sentiram desafiados, responderam que tiveram altos índices da emoção "nervoso", que no PANAS é considerada como negativa. Dependendo do jogo, o objetivo pode ser causar algum nível de nervosismo.

4.2 Pontos positivos e negativos do Godot

De modo geral, o uso do Godot, dos padrões de projeto e dos *plugins* da comunidade ajudaram bastante no desenvolvimento do jogo. Algo a destacar é a estrutura de *nodes* do Godot, que realmente ajudou graças a criação de estruturas visuais similares às organizações das fichas de RPG. Assim, fica claro no Godot quais as características de determinado personagem de forma visual. Outro fator é o sistema de *resource* do Godot, ele é realmente útil para armazenar dados reutilizáveis do RPG, como itens e classes.

Entre os pontos negativos, o sistema de *tileset* da versão 3.5 do Godot ainda tem certas limitações, como exemplo a limitação da edição e manipulação dos *tiles* dentro da *engine*. Isso justifica a comunidade do Godot estar desenvolvendo um *rework* para a próxima versão a ser lançada. O principal problema do sistema de *tiles* da versão 3.5 é em relação a interação e usabilidade. Várias configurações significativas, como criação e colisão de um *tile*, não tem *feedbacks* suficientes que deixem claro a situação dos *tiles* para o desenvolvedor. Além disso, o sistema de seleção dos *tiles* na *grid* é bastante limitado, o que pode acarretar uma dificuldade de selecionar determinados *tiles* entre diversas camadas de *grid*.

Outro problema é a limitação para identificar os *tiles*, já que não existe um sistema de indexação de *tiles* efetivo, o que dificulta a identificação de alguma característica específica de determinado *tile*. Um exemplo concreto, o sistema atual dificulta a implementação de diferentes sons de passos em diferentes *tiles* (grama e pedra, por exemplo). Isso ocorre em razão do atual sistema de *tiles* ter uma forma limitada de adicionar informações relevantes a *tiles* em específico.

Outra questão é a forma pouco intuitiva que o Godot trabalha com a *grid*, pois ele tem uma *grid* geral e uma específica da camada de *tile*, onde a segunda serve para criar organizações específicas da região que será criada, porém essa diferença não fica clara.

Já na versão 4.0, todo o sistema será refeito, onde a prioridade será melhorar a usabilidade, apresentando as informações de forma mais clara, adicionar um sistema de *tags* e melhorar a disposição dos *tiles* entre as camadas, deixando assim mais fácil criar, identificar e organizar os *tiles*[2].

[2] https://godotengine.org/article/tiles-editor-rework

Outra aspecto que tem limitação é a manipulação e organização dos *sprites* das animações na *engine*. Além disso, algo que causou confusão em momentos do desenvolvimento foi a duplicação de objetos, que no Godot objetos duplicados ficam ligados por padrão, ou seja, ao alterar um atributo de um é automaticamente alterado o do outro. Assim, se o objetivo for duplicar um objeto e utilizar a cópia de forma independente, é necessário quebrar a ligação após a duplicação. Isso criou dificuldades durante o desenvolvimento pois não fica claro na interface do Godot quando dois objetos duplicados estão ligados.

4.3 Participação da equipe

Em relação à equipe de desenvolvimento do estúdio Guildsight, todos os desenvolvedores são universitários. Assim, desenvolveram o Lost Borders em paralelo a atividades da universidade e trabalhos assalariados. Durante o desenvolvimento, a equipe adquiriu vários aprendizados e teve contato com elementos e decisões que contribuíram para facilitar o processo, como:

- Definitivamente a organização e alinhamento da equipe, o que permitiu entender o andamento do desenvolvimento de forma geral, trocar informações e agilizar os processos;
- O uso de metodologias ágeis, como o Scrum, com reuniões semanais de alinhamento. Isso permitiu uma melhor organização e comunicação. Para executar o Scrum, foi utilizado o HacknPlan[3], site de Scrum focado no desenvolvimento de projetos relacionados a jogos;
- Reconhecimento e validação do desenvolvimento, por meio do recebimento de *feedbacks* positivos e testes com terceiros, que ajudaram a equipe a ter mais engajamento e energia;
- O estúdio recebeu um aporte financeiro da Prefeitura de Fortaleza, do projeto Sebrae Developers, ao ser vitorioso em um edital, o que possibilitou aos membros dedicarem mais tempo no trabalho do projeto;
- A especialização de cada membro em sua área, em que cada um trabalhou de maneira mais focada, apesar de alguns acúmulos de funções quando necessário. Ajudou bastante a experiência multidisciplinar adqui-

[3] https://hacknplan.com/

rida pelos membros no curso de Sistemas e Mídias Digitais;

- Confiança dos membros uns nos outros, adquirida por diversos trabalhos que o grupo realizou anteriormente.

Além disso, tiveram elementos e situações que atrapalharam e dificultaram o desenvolvimento do jogo, como:

- Gerenciamento de tempo com as atividades da universidade, o que consumiu consideravelmente energia e tempo da equipe, além de causar estresse em muitos membros;
- O escopo completo do jogo foi fechado tardiamente, por conta da demora no processo de validação de design, causado pela dificuldade anterior;
- Em casos bem específicos, o baixo comprometimento de algum membro para com as atividades, por conta da universidade e outros trabalhos, ocasionando com que o desenvolvimento se tornar-se algo "secundário";
- Pequenos ruídos e falhas de comunicação, que ocasionou com que partes do jogo, desenvolvidas inicialmente de forma errada, fossem modificadas.

4.4 Decisões e mudanças de design

Várias decisões e mudanças de design foram validadas durante o desenvolvimento, com testes internos e externos. Assim, algumas escolhas foram bem aproveitadas, enquanto outras foram pouco usadas ou substituídas. Uma decisão positiva foi a reorganização do sistema de elementos e fraquezas, o que aumentou as possibilidades no combate, criando situações mais estratégicas. De decisão ruim foi a forma de utilizar o *grid* modular, algo que, apesar de ser prototipado e pensado para criar possibilidades interessantes, não foi muito usado e ficou defasado comparado aos outros elementos do jogo.

5 CONCLUSÃO

Nesse trabalho foi apresentado um relatório do desenvolvimento do jogo de RPG Lost Borders no Godot. Esse trabalho foi realizado com esse objetivo visto que jogos de RPG são complexos por apresentarem uma série de características específicas do gênero, o que demanda a busca de facilitadores para o desenvolvimento desses jogos. Nessa linha, uma game *engine* é um *framework* com várias ferramentas para a criação de jogos, com objetivo de facilitar o desenvolvimento e o uso de padrões de projeto em busca uma melhor criação, organização, manutenção e escalabilidade do código.

Assim, foi desenvolvido o jogo Lost Borders, um RPG de ação com exploração *top-down* e combate em *grid*, utilizando os padrões de projeto Entity Component System, Observer e Mediator, além de Behavior Tree para o desenvolvimento da inteligencia artificial dos inimigos.

Ao final do desenvolvimento, foi realizado testes de usuário utilizando o PANAS. De modo geral, os testes demonstraram uma aceitação positiva do jogo pelos usuários, com ressalvas em relação ao balanceamento e dificuldade. Ao final, foram indicados pontos positivos e negativos observados ao longo do desenvolvimento. De positivo o visual da organização dos *nodes* do Godot, de negativo algumas limitações das ferramentas de *tileset* e animação.

Para trabalhos futuros é recomendado:

- Analisar como as dificuldades referentes ao Godot podem ser superadas, principalmente em relação a edição de *sprites* e tiles dentro da *engine*;
- Identificar como os diversos tipos de RPGs podem diferenciar o desenvolvimento;
- Realizar um comparativo do uso do Godot com outras *engines*, em relação ao desenvolvimento de jogos de RPG;
- Realizar testes de balanceamento da dificuldade do Lost Borders.

REFERÊNCIAS

COLLEDANCHISE, M.; ÖGREN, P. **Behavior trees in robotics and AI: An introduction**. [S.l.]: CRC Press, 2018.

COMMUNITY, G. **Godot Docs**. 2022. Disponível em: <https://docs.godotengine.org/>. Acesso em: 18 mai. 2022.

GALINHA, I. C.; PAIS-RIBEIRO, J. L. Contribuição para o estudo da versão portuguesa da positive and negative affect schedule (panas): Ii–estudo psicométrico. **Análise psicológica**, ISPA, p. 219–227, 2005.

GAMMA, E.; JOHNSON, R.; HELM, R.; JOHNSON, R. E.; VLISSIDES, J. **Design patterns: elements of reusable object-oriented software**. [S.l.]: Pearson Deutschland GmbH, 1995.

JAM, I. G. G. **Global Game Jam**. 2020. Disponível em: <https://globalgamejam.org/>. Acesso em: 22 mai. 2022.

LORD, R. Why use an entity component system architecture for game development? 2012.

LOVATO, N. **GDQuest**. 2021. Disponível em: <https://www.gdquest.com/tutorial/godot/design-patterns/intro-to-design-patterns/>. Acesso em: 02 jul. 2022.

NYSTROM, R. **Game programming patterns**. [S.l.]: Genever Benning, 2014.

ONUCZKO, C.; CUTUMISU, M.; SZAFRON, D.; SCHAEFFER, J.; MCNAUGHTON, M.; ROY, T.; WAUGH, K.; CARBONARO, M.; SIEGEL, J. A pattern catalog for computer role playing games. **GameOn North America**, p. 33–38, 2005.

QU, J.; SONG, Y.; WEI, Y. Applying design patterns in game programming. In: THE STEERING COMMITTEE OF THE WORLD CONGRESS IN COMPUTER SCIENCE, COMPUTER .... **Proceedings of the International Conference on Software Engineering Research and Practice (SERP)**. [S.l.], 2013. p. 1.

RALPH, P.; MONU, K. Enduring design challenges in western roleplaying video games. **The Journal of Creative Technologies**, n. 6, 2016.

SPACEY, J. **Simplicable: 39 Examples of Design Goals**. 2018. Disponível em: <https://simplicable.com/design/design-goals>. Acesso em: 02 mar. 2023.

STEAMPOWERED. **Baldur's Gate 3**. 2022. Disponível em: <https://store.steampowered.com/app/1086940/Baldurs_Gate_3/>. Acesso em: 05 dez. 2022.

STENSTRÖM, C. D.; BJÖRK, S. Understanding computer role-playing games: A genre analysis based on gameplay features in combat systems. In: **Second Workshop on Design Patterns in Games (FDG 2013)**. [S.l.: s.n.], 2013.

VALENCIA-GARCÍA, R.; LAGOS-ORTIZ, K.; ALCARAZ-MÁRMOL, G.; CIOPPO, J. del; VERA-LUCIO, N. Technologies and innovation: Second international conference, citi 2016, guayaquil, ecuador, november 23-25, 2016. **Proceedings. Communications in Computer and Information Science**, v. 658, 2016.

WATSON, D.; CLARK, L. A. The panas-x: Manual for the positive and negative affect schedule-expanded form. University of Iowa, 1994.

ZUKOWSKI, C. **How To Market A Game**. 2022. Disponível em: <https://howtomarketagame.com/>. Acesso em: 18 mai. 2022.

APÊNDICE A – QUESTIONÁRIO PANAS

A seguir, um anexo do questionário PANAS utilizado nos testes do jogo Lost Borders. Apresenta um termo de consentimento, as perguntas do PANAS e algumas questões extras.

PANAS (Positive and Negative Affect Schedule)

Termo de Consentimento Livre e Esclarecido

1. Você está sendo convidado para participar de um formulário feito pelo aluno Raoni Coelho do curso de Sistemas e Mídias Digitais para a realização de um Trabalho de Conclusão de Curso.
2. Você foi selecionado para ser voluntário e sua participação não é obrigatória.
3. A qualquer momento você pode desistir de participar e retirar seu consentimento.
4. Sua recusa não trará nenhum prejuízo em sua relação com a equipe de desenvolvimento do jogo.
5. Esse formulário tem por objetivo saber como foi a experiência e engajamento dos entrevistados com as atividades, coletando "feedbacks".
6. Sua participação neste formulário consistirá em ler as perguntas e respondê-las sinceramente.
7. As informações obtidas através desse formulário serão confidenciais e asseguramos o sigilo sobre sua participação.
8. Os dados não serão divulgados de forma a possibilitar sua identificação.
9. Por falta de versão do questionário PANAS em português do Brasil, foi utilizada uma versão em português de Portugal.

☐ Concordo

Nome Completo:_____

Esta escala consiste num conjunto de palavras que descrevem diferentes sentimentos e emoções. Leia cada palavra e marque a resposta adequada. Indique em que medida sentiu cada uma das emoções ao jogar o Lost Borders.

	Nada ou muito Ligeiramente	Um Pouco	Moderadamente	Bastante	Extremamente
Interessado	☐	☐	☐	☐	☐
Perturbado	☐	☐	☐	☐	☐
Excitado	☐	☐	☐	☐	☐
Atormentado	☐	☐	☐	☐	☐

Agradavelmente surpreendido	☐	☐	☐	☐	☐
Culpado	☐	☐	☐	☐	☐
Assustado	☐	☐	☐	☐	☐
Caloroso	☐	☐	☐	☐	☐
Repulsa	☐	☐	☐	☐	☐
Entusiasmado	☐	☐	☐	☐	☐
Orgulhoso	☐	☐	☐	☐	☐
Irritado	☐	☐	☐	☐	☐
Encantado	☐	☐	☐	☐	☐
Remorsos	☐	☐	☐	☐	☐
Inspirado	☐	☐	☐	☐	☐
Nervoso	☐	☐	☐	☐	☐
Determinado	☐	☐	☐	☐	☐
Trémulo	☐	☐	☐	☐	☐
Activo	☐	☐	☐	☐	☐
Amedrontado	☐	☐	☐	☐	☐

Gostaria de enviar alguma observação extra?

_____

_____

_____