Galactic Crusade

Gabriel Fazenda[1]

[1] Universidade do Vale do Rio dos Sinos (UNISINOS), Escola Politécnica, Brasil

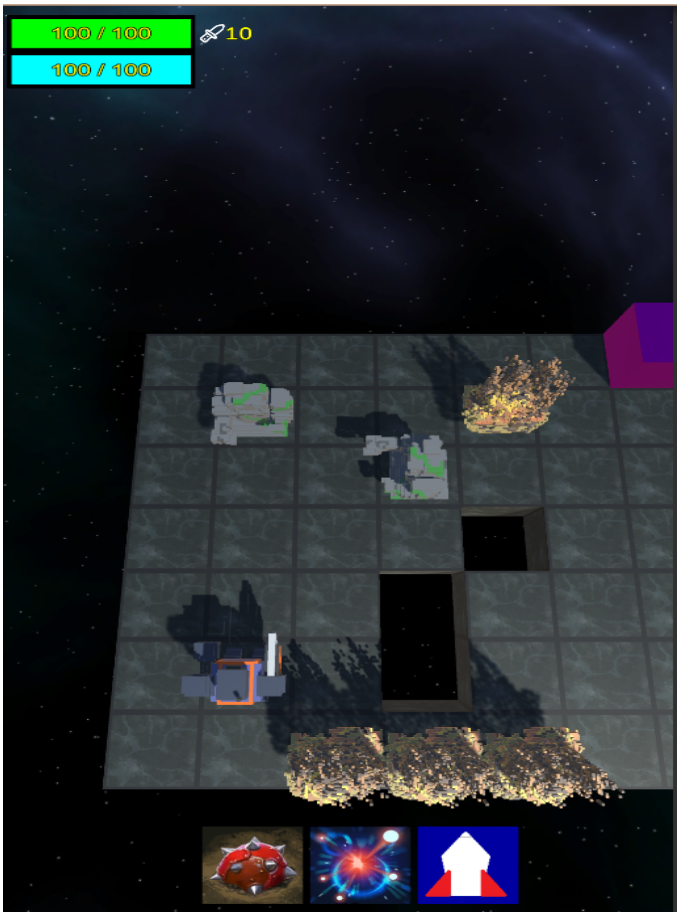

Figura 1: *Tela do Jogo*

Author's Contact:

gabriel.l.fazenda@gmail.com

Keywords: top-down, turn-based, roguelike, strategy.

1 Introdução

Galactic Crusade (working title) é um jogo tático de turnos onde o jogador controla um robô e deve sobreviver ao maior número de níveis que conseguir. A temática do jogo é espacial, cada inimigo é bem diferente e possui uma habilidade específica para dificultar a jornada de quem está jogando. O principal diferencial buscado é o de trazer uma experiência que se adapta ao jogador atual, analisando e coletando seus padrões, comportamentos e perfil de ataque/movimentação e utilizando-se destes com a finalidade de possibilitar tomadas de decisões mais precisas através da técnica Minimax (Inteligência Artificial), visando fazer com que ele se sinta sempre desafiado enquanto se diverte.

2 Inspirações

A ideia inicial do projeto partiu do objetivo de criar um jogo baseado principalmente em elementos de estratégia e que ao mesmo tempo trouxesse sempre a sensação de uma experiência única para cada jogador. Partindo disso, foram pensados e pesquisados vários jogos que trazem propostas semelhantes como base e inspiração para os elementos desenvolvidos: Crypt of Necrodancer, Sproggiwood, Dungeons of Dredmor, Hoplite, Hitman Go, entre outros. Cada um deles possui uma temática bem única, tanto como jogabilidade, arte e experiências proporcionadas. Considerando isso, busquei entender um pouco das mecânicas aplicadas por eles para trazer algo semelhante no meu projeto.

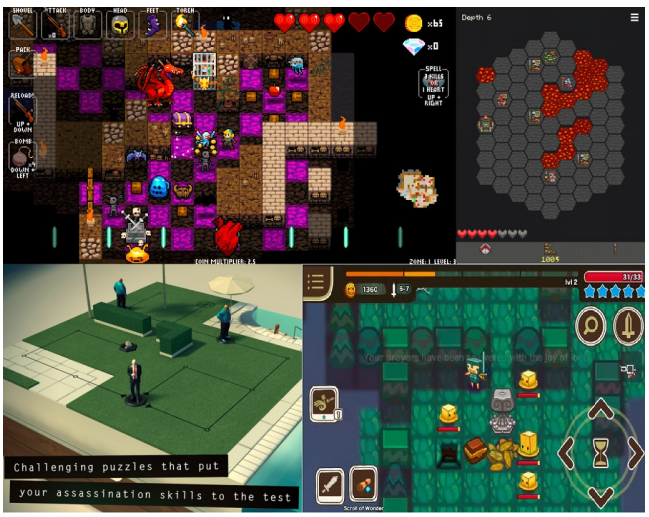

Figura 2: *Inspirações*

3 Desenvolvimento

Desde o começo, o objetivo foi de sempre buscar primeiramente as mecânicas fundamentais do jogo, visando possibilitar o teste de cada elemento desde sua fase inicial. Tendo isso em mente, o processo se iniciou com a montagem do gerador de níveis, para que já fosse possível ver o grid visualmente e, consequentemente, possibilitasse e criação do personagem principal e sua movimentação por turnos.

Após a movimentação do jogador, o pathfinding com A* dos inimigos foi implementado. Visto que os níveis são formados por tiles quadrados e não hexagonais, a escolha foi de não possibilitar a movimentação diagonal. Com isso, foram realizados vários testes incrementais para sempre buscar a melhor maneira de se desenvolver estes elementos fundamentais, com foco na movimentação do jogador, para que a mesma aconteça de uma forma que faça sentido e seja fácil/intuitiva para quem quer que esteja jogando. Nesta fase, foram estudados 3 métodos de movimento (no caso da plataforma Android): swipe, clique sobre jogador+clique em tile desejado e setas direcionais permanentes na tela. Após realizar testes e observar pessoas jogando, o método vencedor foi o swipe, então os outros foram retirados. Juntamente com isso, também foi adicionada uma movimentação estendida, que permite com que o jogador 'programe' uma movimentação de até 5 casas em linha reta, visando facilitar e agilizar o movimento em casos onde os inimigos já tenham sido derrotados e o portal para o próximo nível ainda esteja distante.

A terceira fase do projeto foi a mais extensa, onde foram pensados e trazidos novos elementos para o gameplay como um todo. Primeiramente, veio o desenvolvimento e implementação dos poderes do jogador para que cada um tivesse uma certa estratégia envolvida, juntamente com um balanceamento inicial. Para aumentar as possibilidades e dificuldades nas fases, foram também desenvolvidas armadilhas/obstáculos de cenário, que não atacam como os inimigos, mas possuem efeitos que visam fazer com que o jogador tenha que planejar melhor a sua movimentação. Juntamente com isso, nesta parte comecei a fazer os modelos 3D, que serão melhor explicados a seguir, e também habilidades secundárias para cada um dos inimigos para complementar ainda mais o gameplay.

Por último, foi implementado o leitor de níveis, para possibilitar a utilização de níveis pré-montados, além dos procedurais. A ideia, a partir disso, é de que os 10 primeiros níveis não serão procedurais pois servirão para coletar informações sobre padrões e comportamentos do jogador. Os motivos para tal serão explicados logo abaixo. Além disso, foram adicionados menus, feedbacks visuais, sons e vários bugs foram corrigidos.

4 Geração de Níveis

De início, a ideia era de implementar uma rede neural com o gerador de níveis procedural, para que ele fosse capaz de criar níveis funcionais de maneira eficaz. Porém, após algumas pesquisas e testes, esta implementação foi descartada considerando a grande quantidade de dados e testes necessários para que este algoritmo fosse bem sucedido. Devido a estes fatos, a nova ideia é utilizar o algoritmo MiniMax, que é definido como 'método para minimizar a possível perda máxima', mais popularmente utilizado em jogos baseados em turnos, como criação de bots para jogo da velha e xadrez. Além disso, já possuo conhecimentos prévios de implementação desta técnica, utilizada na disciplina Inteligencia Artificial do curso para definir tomadas de decisão de um bot para o jogo Pokémon Showdown.

```
Data: node, depth, maximizingPlayer
Result: o valor da heurística do nodo
if depth = 0 || node is a terminal node then
    retorna o valor da heurística do nodo
end
if maximizingPlayer then
    bestValue = - (infinite);
    forall childs of node do
        v = minimax(child, depth - 1, FALSE);
        bestValue = max(bestValue, v);
    end
    return bestValue
end
else
    bestValue = + (infinite);
    forall childs of node do
        v = minimax(child, depth - 1, FALSE);
        bestValue = min(bestValue, v);
    end
    return bestValue
end
```

Figura 3: *Pseudocódigo do Algoritmo de Minimax*

Para tal tarefa, a principal ferramenta que servirá de auxílio na decisão de como montar os níveis (e também de como os inimigos se comportarão ao longo das fases) será a coleta e classificação dos dados do jogador e seu comportamento ao longo do jogo. Nesses dados, estarão contidas informações como:

- Quantidade de dano causado e levado por nível;
- Forma mais utilizada para causar dano;
- Tipo de inimigo que mais causou dano ao jogador;
- Número de inimigos mortos por nível (0-100, de acordo com os inimigos gerados no nível em questão);
- Número de rodadas necessárias para completar cada nível.

No momento, a técnica ainda não está sendo empregada na versão atual do jogo, este será o próximo passo após trazer a mecânica de gravação dos dados do jogador, que se dará a partir da classe GameLogs, que terá funções públicas que permitirão com que todos os objetos se comuniquem com esta classe e informem sobre os eventos do jogo e valores essenciais como dano, HP, posições, etc.

Estes dados serão coletados e atualizados constantemente, afim de se entender a melhor forma de trazer um desafio interessante a quem está jogando. Para classificação inicial, serão utilizados 10 níveis pré-configurados, visando criar uma média de valores e grupos de perfis a partir disso, para que seja possível dar uma classificação inicial para cada jogador.

5 Estrutura

O projeto em si baseia-se principalmente nos scripts 'managers': Um que controla o campo de batalha (matriz) e todos objetos que se movimentam nele, outro responsável pela progressão de níveis, um terceiro que faz o controle de todos os elementos da interface e por último o gerenciador de inimigos, que utilizara a técnica MiniMax para montar suas jogadas.

- **BoardManager** - Controla a matriz de todos os objetos que compõem cada nível, bem como todas as validações para movimentação de cada personagem, etc;
- **GameManager** – Responsável por controlar o carregamento de novas fases e limite de FPS;

- **UXManager** – Controla todos os botões e as chamadas feitas pelos mesmos. Também controla as barras de HP e mana do jogador, bem como ativação/desativação dos botões dos poderes;
- **EnemyCoordinator** – Controle de quando cada um dos inimigos age na partida e como se comportam, a partir do perfil específico do jogador.

Além disso, para realizar as trocas de turno entre jogador e inimigos, foi implementado um sistema de eventos conforme o tutorial 'Events: Creating a simple messaging system' disponibilizado no site da Unity. Com isso, todos os outros objetos relacionados tanto a ações do jogador, como minas remotas quanto dos inimigos, como tiles venenosos, sabem quando realizar suas funções/efeitos no momento certo, sem a necessidade de utilizar o loop Update ou de serem chamados separadamente.

Sobre a estrutura das classes, há duas principais das quais o jogador e inimigos fazem uso (através de polimorfismo): Character, que é a classe que possibilita a movimentação do objeto na matriz e Entity, que possui as informações de pontos de vidas do objeto. Ou seja, todos os Characters são Entities, mas Entities não necessariamente são Characters. Como os obstáculos que utilizam apenas a classe Entity.

Fora isso, outra classe bastante utilizada foi a SpecialTile, que serve para situar elementos no cenário, fazendo com que cada um saiba sua posição na matriz, caso esta informação seja necessária durante o jogo. A partir desta, outras classes são derivadas, como a Obstacle, que gera tiles ao redor dos obstáculos para mostrar suas áreas de efeito e a PoisonTile, que envenena o jogador caso o mesmo passe por cima do veneno.

6 Arte

Visto que arte e modelagem não são áreas com as quais me identifico (ou tenho experiencias previas), um dos maiores desafios deste projeto foi pensar na arte e identidade visual que eu gostaria de trazer para o jogo. Depois de pesquisar várias opções, como assets e modelos open-source, etc. foi então que encontrei a ferramenta MagicaVoxel, que permite modelagem 3D baseada em voxels. Devido a sua fácil curva de aprendizagem, juntamente com o fato de que a própria arte baseada em voxels pode ser considerada mais prática para alcançar resultados rápidos, decidi ir por este caminho e nele encontrei o modo do qual quero que meu jogo seja visualmente.

Além do fato desse estilo de arte estar em alta ultimamente, a possibilidade de modelar todos os objetos do jogo auxilia para manter a coerência entre todos os elementos visuais e também trás mais liberdade se comparado a necessidade de encontrar modelos já prontos que combinem entre si.

Figura 4: *Modelos Criados*

7 Estado Atual

Durante o semestre, foi possível desenvolver toda a fundamentação do jogo e elementos essenciais para que a versão atual seja completa no sentido da jogabilidade, com um início, meio e fim. No momento, o jogo está completamente funcional para a plataforma Android e possui os seguintes elementos:

- Gerador de níveis procedural e pré-configurados;
- 10 níveis pré-montados para introduzir os elementos do jogo e capturar informações do jogador;
- Personagem principal com 3 poderes, ataque básico e movimentação estendida/configurável de até 5 casas;
- Menus;
- Sons;
- Efeitos especiais (explosões, partículas, etc.);
- 3 Inimigos que se movimentam a partir do algoritmo A*, onde cada um possui uma característica/habilidade secundaria única, para trazer mais elementos que façam o jogador pensar melhor na sua próxima jogada:

  1 – **RockMonster**: Poderoso inimigo feito de pedra com alta resistência e poder de ataque. Seus punhos são tão fortes que possuem a possibilidade de nocautear o jogador por 1 rodada;

  2 – **Veno-alien**: Mesmo não sendo tão letal na luta corpo-a-corpo, este inimigo deixa sempre um rastro de veneno capaz de causar um alto dano ao jogador;

  3 – **LavaThing**: Inimigo altamente perigoso por ser capaz de atacar de distancias maiores.
- 2 Obstáculos estáticos: um que drena dano e outro que bloqueia a utilização de poderes.

8 Próximos Passos

Considerando o estado atual do jogo, os próximos objetivos são de trazer o refinamento do que já foi desenvolvido (principalmente balanceamento) e inclusão de mais elementos de gameplay em si, além de elementos visuais e sonoros para complementar a experiencia do jogador. Outro ponto com qual pretendo trabalhar também é a otimização do jogo como um todo para a plataforma Android, para que seja possível rodá-lo em um número maior de dispositivos sem maiores problemas de performance. Isso incluirá maior controle sobre objetos criados/instanciados e também otimização dos modelos 3D.

Além disso, serão conduzidas sessões de gameplay com diferentes pessoas afim de recolher feedbacks e também para se estabelecer e reconhecer os diferentes padrões de comportamento e decisão, visando caracteriza-los e agrupa-los em perfis. Com isso, será possível implementar e utilizar a técnica MiniMax de forma mais precisa, considerando a possibilidade de pesar melhor as jogadas, a partir de cada perfil determinado. Juntamente com isso, os inimigos terão também movimentações mais avançadas além de só seguir o jogador, como possuir a habilidade de defender uma área do cenário especifica/patrulhar e também atacar de maneira mais eficiente (quando o ataque for a distância).

Olhando para trás, vejo que consegui aprender muito com este projeto e também pude utilizar muitos ensinamentos adquiridos ao longo do curso. Mantendo a mesma mentalidade que tive na disciplina de motores: Projeto de Jogos, foi possível me organizar de maneira efetiva e desta forma pude desenvolver todos os elementos que esperava/necessitava para o jogo nesta primeira etapa. Com certeza muitos aprendizados novos virão nesta segunda parte e o jogo ficará ainda mais robusto.

Referências

BRACEYOURSELFGAMES. *Crypt of the Necrodancer*. http://necrodancer.com/.

CORNELL-CS312-RECITATION-21. *Minimax search and Alpha-Beta Pruning*. https://www.cs.cornell.edu/courses/cs312/2002sp/lectures/rec21.htm.

EPHTRACY. *MagicaVoxel*. https://ephtracy.github.io/.

FREEHOLDGAMES. *Sproggiwood*. https://play.google.com/store/apps/details?id=com.freeholdgames.sproggiwood&hl=en.

GASLAMPGAMES. *Dungeons of Dredmor*. https://dungeonsofdredmor.com/.

LUO, G. *Pokemon Showdown*. https://pokemonshowdown.com/.

MAGMAFORTRESS. *Hoplite*. https://play.google.com/store/apps/details?id=com.magmafortress.hoplite&hl=en.

MILLINGTON, I., AND FUNGE, J. 2009. *Artificial Intelligence for Games*. CRC Press.

NEVERSTOPBUILDING. *Tic Tac Toe: Understanding The Minimax Algorithm*. https://neverstopbuilding-dropblog.herokuapp.com/minimax.

RUSSELL, S. J., AND NORVIG, P. 2003. *Artificial Intelligence: A Modern Approach*, 2 ed. Pearson Education.

SQUAREENIX. *Hitman GO*. https://play.google.com/store/apps/details?id=com.squareenixmontreal.hitmango&hl=en.

UNITY. *Creating a simple messaging system*. https://unity3d.com/learn/tutorials/topics/scripting/events-creating-simple-messaging-system.

WIKIPEDIA. Novembro, 2017. *Minimax*. https://en.wikipedia.org/wiki/Minimax.