Documento de *design* de um jogo *roguelike*

Pedro Probst Minini

***Abstract*—Este documento tem como intuito demonstrar, em linhas gerais, o *game design* de um jogo *roguelike* que será elaborado com o intuito de explorar algoritmos de PCG (*Procedural Content Generation*), *data-driven design* e a arquitetura de software ECS (*Entity-Component System*).**

I. Introdução

COM o objetivo de explorar diversas facetas e técnicas de programação de jogos – em especial, algoritmos de geração procedural, *data-driven design* e a arquitetura *entity-component system* –, propõe-se o desenvolvimento de um jogo *roguelike* tradicional – um gênero de jogos conhecido pelos gráficos simples (na maioria das vezes, simplesmente caracteres ASCII ou *tile art* minimalista), mecânicas complexas e uso "abusivo" de geração procedural em mapas, equipamentos, etc., ou até mesmo cultura (ver *Ultima Ratio Regum* e *Dwarf Fortress*).

Por conta da simplicidade gráfica e do foco em mecânicas de jogo, *roguelikes* tipicamente estão na vanguarda em relação a diversas técnicas de programação e *game design*. *Caves of Qud*, por exemplo, foi o primeiro jogo comercial a utilizar o algoritmo *WaveFunctionCollapse* (Maxim Gumin, 2016) para a geração de níveis.

Portanto, é apenas natural que jogos *roguelike* se tornaram um grande *playground* para programadores experimentarem diversas técnicas de programação.

Março de 2020

II. Conceitos-chave

Nesta seção, alguns conceitos citados anteriormente serão explicados *grosso modo* (caso contrário, este documento ficaria exaustivo demais).

- **Procedural Content Generation (PCG)**:
  Por uma questão de simplicidade, aqui nos referimos a algoritmos de PCG como procedimentos que geram novos aspectos do jogo (níveis, equipamentos, inimigos, etc.) a cada nova "rodada". Por exemplo, o primeiro nível de uma *dungeon* é diferente a cada vez que o jogador perde no jogo e o reinicia. Algoritmos de geração de mapas funcionam tipicamente dentro de um *pipeline*, onde diversos algoritmos diferentes são usados em sequência para gerar *layouts* interessantes.
- **Data-Driven Design**:
  Em programação de jogos *design orientado a dados* é uma prática na qual várias entidades do jogo (monstros, equipamentos, itens, efeitos, etc.) são definidos fora do código principal do programa; ou seja, essas entidades estão tipicamente dentro de um arquivo JSON ou similar (chamados de RAWS). *Dwarf Fortress* é um jogo que faz uso massivo de RAWS para facilmente criar novo conteúdo para o jogo (ainda que DF não seja estritamente um *roguelike*, compartilha de vários aspectos comuns ao gênero).
- **Entity-Component System (ECS)**:
  É uma arquitetura de software que teve seu ressurgimento atrelado ao desenvolvimento de jogos. Enquanto não era tão popular há mais de 10 anos atrás, hoje esta arquitetura é comumente utilizada. No ECS, toda *entidade* do jogo (inimigo, equipamento, efeito, etc.) é composta de *componentes* que fornecem diversas funcionalidades a ela. Os *sistemas* são responsáveis pelo funcionamento da lógica do jogo. Como exemplo, podemos citar um sistema de "fome" num *roguelike* tradicional: neste sistema, todas as *entidades* com o *componente* "indicador de fome" são acessadas a cada *tick*, e o indicador de fome da entidade é decrementado; dependendo do nível do indicador, o status de fome da entidade é alterado.

III. Design

Nesta seção, exploraremos alguns aspectos do *design* de um jogo *roguelike* tradicional com temática de ficção científica. De início, este jogo não pretende ser um *roguelike* extenso e complexo como *Caves of Qud* ou *Cataclysm: Dark Days Ahead* – estes jogos receberam anos de desenvolvimento. Como o *timeframe* é limitado, optamos pelo desenvolvimento de um *coffeebreak roguelike*, tal como *Brogue*, no qual há um número menor de mapas a serem explorados, progressão baseada puramente nos itens carregados pelo jogador e ausência de criação de personagens estilo RPG. Em suma, o personagem principal inicia com itens *default* e seu estilo de jogo vai sendo alterado a medida que encontra itens diferentes (e aleatórios!).

A. Inspirações

O jogo pretende ser uma miríade de tudo o que o autor aprecia em termos de história, temática e jogabilidade. Nominalmente:

- A tetralogia de livros *The Book of the New Sun*, de Gene Wolfe. Dos livros partirá a maior parte da inspiração da *lore* e temática do jogo (de modo similar em como a grande maioria dos *roguelikes* é inspirado pelas obras de J.R.R. Tolkien).
- O filme soviético *Stalker*, de Andrei Tarkovski.
- *Caves of Qud*, *roguelike* do qual partirá a *atmosfera* do jogo, que contará com plantas sencientes e cibernéticos. O aspecto de exploração de ruínas de civilizações passadas também será replicado a partir do *pipeline* de geração de mapas.
- Brogue, *roguelike* que possui um sistema de progressão similar ao que será implementado.

- A melancolia do mangá *Girls' Last Tour*, de Tsukumizu em conjunção com a atmosfera aconchegante do mangá *Yokohama Kaidashi Kikou*, de Hitoshi Ashinano. "Melancólico" e "aconchegante" podem ser dois termos antagônicos, mas ao mesmo tempo em que a realidade do jogo pode ser dura, o jogador pode encontrar breves momentos de descanso em meio às ruínas de civilizações antigas.

B. Objetivos

- Explorar PCG para geração de maps e itens diferentes.
- Aplicar *data-driven design*.
- Aplicar ECS.
- Explorar aspectos diversos relacionados ao desenvolvimento de jogos, com a implementação de sistemas de inteligência artificial (IA), interface de usuário (UI), inventário, turnos, etc.

Em trabalhos futuros, talvez seja interessante explorar geração procedural de culturas, sociedades e história, tal como é feito em *Ultima Ratio Regum* e *Dwarf Fortress*. Entretanto, são tarefas árduas e muito experimentais. Se geração procedural, no geral, é uma área de nicho, geração procedural de civilizações inteiras é o nicho do nicho. *Ultima Ratio Regum* é um projeto ambicioso e de longo termo do PhD Mark R. Johnson, enquanto *Dwarf Fortress* está em desenvolvimento desde *antes* de 2002 até os dias de hoje.

C. Mecânicas

Um dos maiores objetivos de jogos *roguelike* é a implementação de *gameplay* tático. Para atingir isso, vários sistemas são implementados; por essa razão é dito que *roguelikes* são mecanicamente complexos. Por baixo dos caracteres ASCII que assustam jogadores casuais, há sistemas ainda mais assustadores.

- ***Permadeath***. Quando o jogador morre, ele realmente morre e precisa iniciar um novo jogo. Este é o grande pilar do gênero *roguelike*.
- *Gameplay* baseado em **turnos**, similar ao estilo *D&D* no qual a maioria das ações do jogador (e dos inimigos) tem seu sucesso/fracasso definido estatisticamente a partir da "rolagem de dados". Por exemplo, um certo rifle pode ter o dano definido como "*1d9 + 4*", ou seja, *1* dado de *9* lados é jogado e o resultado é somado a *4*.
- Grande uso de **algoritmos de PCG** para a geração procedural de mapas e itens com efeitos diversos, além de posicionar apropriadamente *mobs* e *props* no mapa.
- **UI** completa, incluindo um sistema de ***logging*** que exiba ações importantes realizadas em cada turno (e.g. *"Player attacks Man-Ape for 20 HP."*, *"Player is stunned for the turn!"*, etc.)
- Sistema de **facções**, com pontos em cada facção definidos aleatoriamente a cada novo jogo. Por exemplo, a facção dos robôs pode ser aliada/hostil à facção das plantas sencientes, os bandidos podem ser hostis/aliados aos manapes, etc. No entanto, para assegurar que haja facções antagonistas ao jogador, sempre haverá internamente a facção *player hater*, que engloba várias facções antagonistas diferentes. Na verdade, a maioria dos *roguelikes* implementa facções em código, mas como normalmente não são aspectos a serem manipulados pelo jogador, são omitidos totalmente do jogo – afinal, num *roguelike* típico, todos querem acabar com a aventura do jogador. Uma das ideias do jogo é que o jogador manipule o sistema de facções para obter aliados e que também haja disputas entre elas.
- Sistema de **inventário** de tamanho limitado, no qual o jogador poderá armazenar os itens encontrados. Não haverá sistema de "peso", ou seja, o jogador poderá carregar quantos itens o inventário permitir.
- Como não há magia, as **habilidades especiais** terão uso limitado (*charges*) ao invés de utilizarem pontos de *mana*, como é tipicamente feito em *roguelikes*.
- **IA relativamente avançada** para os inimigos, contendo diferentes comportamentos: atacar o jogador, preferir atacar à distância (se tiver arma balística), fugir e reagrupar, avisar aliados que o jogador foi avistado, etc. É preferível que os *mobs* sejam tão simulados quanto o jogador, contendo seus próprios equipamentos, cibernéticos, etc.
- Sistema de **cibernéticos e modificação de armas**. O jogador pode ter até "X" cibernéticos instalados no corpo, conferindo a ele habilidades novas. Armas podem ser modificadas para gerarem diferentes efeitos ao atingirem o inimigo.
- Sistema de **iluminação** e furtividade (este último *talvez*!).
- Entre outros!

Tipicamente, *roguelikes* têm *layouts* de mapas diferentes em cada nível (ao subir/descer escadas), mas temática similar. Neste jogo, para garantir uma variabilidade grande de temáticas de mapas, ao invés de subir/descer escadas para explorar diferentes níveis numa *dungeon*, o jogador ativa teletransportadores e pode ser transportado para um mapa radicalmente diferente do que ele estava anteriormente, contendo arquitetura, inimigos e *layout* próprios da temática.

Por conta do tempo limitado de desenvolvimento do jogo, uma história não estará presente; entretanto, aspectos do mundo (*lore*) poderão ser explorados a partir das diversas entidades encontradas pelo jogador ao decorrer do jogo.

D. Estética

O projeto final será similar à figura 1, contida na próxima página.

E. Software

O jogo será desenvolvido na lingugem de programação **Rust** utilizando a biblioteca **RLTK** (*Roguelike Toolkit*) de Herbert Wolverson em conjunto com a biblioteca ***specs***, que fornece suporte a ECS.

F. Lore (opcional)

O jogo se passa em um tempo indeterminado em relação a hoje, no qual o "apocalipse" já ocorreu e a civilização humana passou por diversas etapas fracassadas de reconstrução.

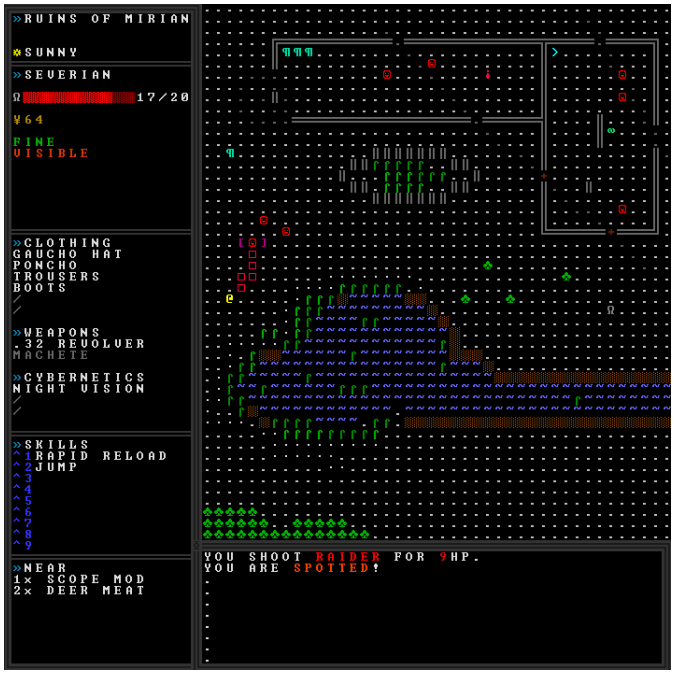

Fig. 1. *Mock* produzido com o software *RexPaint*.

Basicamente, a sociedade é estruturada como uma ”idade das trevas” onde certas relíquias do passado (equipamentos altamente tecnológicos como robôs, computadores, armas e implantes cibernéticos) estão presentes, ainda que em grau mínimo.

Décadas antes do tempo do jogador, a autarquia ecoludista que dominava o Cone Sul do planeta foi reduzido a uma pequena fração do que era, e a maioria das cidades da região tornaram-se independentes a custos extremamente árduos: as regiões agora são marcadas por guerras e monstruosidades que antes estavam presentes em regiões além das fronteiras.

A motivação para o jogador explorar diversas regiões ainda não foi trabalhada. Entretanto, provavelmente estará relacionada a realizar um certo pacto com seres extradimensionais para subverter a autarquia enfraquecida e descobrir os segredos mantidos por ela, que podem potencialmente alterar o rumo da civilização. O jogador é escolhido para essa função pois ele é capaz de acessar diferentes linhas do tempo, retendo a informação obtida nas linhas passadas. Ou seja, a cada vez que o jogador inicia um novo jogo, *lore-wise* ele está acessando uma linha do tempo diferente.

Como não haverá tempo de descrever toda a história do jogo em tempo, prevê-se que o jogador consiga inferir os diferentes aspectos do mundo que o cerca enquanto joga.

IV. Desenvolvimento

O desenvolvimento será realizado em etapas graduais, onde semana a semana serão feitos pequenos progressos até as diferentes metas serem atingidas.

V. Assets

Por enquanto, não está prevista nenhuma forma de arte externa presente no jogo, como trilha sonora ou *tile art*.

VI. Conclusão

Ainda que não ataque um problema específico, creio que seja relevante o desenvolvimento de um jogo como *playground* para diferentes algoritmos e técnicas de programação. As áreas abordadas são nichos interessantes de serem explorados, e, no Brasil, quase inexistentes. Há pouquíssimos *papers* e trabalhos científicos brasileiros envolvendo geração procedural, por exemplo – o que torna a possibilidade de fazer um TCC englobando a área uma proposta interessante, ainda que de fato não traga nenhuma grande inovação, a não ser um possível despertar de interesses e um sentimento de satisfação própria ao autor.

Não pretendo parar de trabalhar neste projeto ainda que termine o TCC (se é que será aceito como tal), pois ele envolve também uma recente curiosidade que desenvolvi relacionado ao desenvolvimento de *roguelikes*, por conta do grande aspecto de *simulação* presente no gênero.

Creio que haverá ainda um grande chão a ser percorrido para que o jogo aproxime-se do ideal que visiono; a expansão de um *coffeebreak roguelike* para um *roguelike* realmente expansivo e de mundo aberto como *Caves of Qud* é algo que almejo para o futuro. *Ultima Ratio Regum*, apesar de longe de estar completo, também serve de grande inspiração para projetos futuros, com sua geração procedural de sociedades inteiras – cada uma com estilos arquitetônicos e governamentais próprios, entre outros aspectos humanos.