UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GUILHERME TAKESHI SAEKI ABREU
LUIZ GUILHERME MONTEIRO PADILHA

INSERÇÃO DE UM SIMULADOR COM BASE EM AGENTES EM UM JOGO

PONTA GROSSA
2022

GUILHERME TAKESHI SAEKI ABREU
LUIZ GUILHERME MONTEIRO PADILHA

INSERÇÃO DE UM SIMULADOR COM BASE EM AGENTES EM UM JOGO

Inserting an agent-based simulation in a game

Trabalho de conclusão de curso de graduação apresentada como requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador(a): Prof. Dr. André Koscianski

PONTA GROSSA
2022

GUILHERME TAKESHI SAEKI ABREU
LUIZ GUILHERME MONTEIRO PADILHA

INSERÇÃO DE UM SIMULADOR COM BASE EM AGENTES EM UM JOGO

Trabalho de conclusão de curso de graduação apresentada como requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 26/maio/2022

André Koscianski
Doutor
Universidade Tecnológica Federal do Paraná

Erikson Freitas de Morais
Doutor
Universidade Tecnológica Federal do Paraná

Vinicius Camargo Andrade
Mestre
Universidade Tecnológica Federal do Paraná

PONTA GROSSA

2022

Dedicamos este trabalho às nossas famílias, pelos momentos de ausência e pela distância emocional.

AGRADECIMENTOS

Certamente estes parágrafos não irão atender a todas as pessoas que fizeram parte dessa importante fase de nossa vida. Portanto, desde já pedimos desculpas àquelas que não estão presentes entre essas palavras, mas elas podem estar certas de que fazem parte dos nossos pensamentos e de nossa gratidão.

Agradecemos ao nosso orientador Prof. Dr. André Koscianski, pela sabedoria com que nos guiou nesta trajetória.

Aos nossos colegas de sala, pois sempre que surgíamos com dúvidas eles nos guiavam da melhor forma possível, pois no fim estamos todos aprendendo.

A Secretaria do Curso, pela cooperação.

Gostaríamos de deixar registrado também, o nosso reconhecimento às nossas famílias, pois acreditamos que sem o apoio, a consideração e a paciência deles teria sido muito difícil vencer esse desafio.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

RESUMO

Muitos jogos de computador envolvem a simulação de experiências, objetos, e fenômenos do mundo real. Isso acontece mesmo que de forma fantasiosa como a série de jogos do Super Mario Bros, que retratam um indivíduo que anda, corre e salta sob influência de gravidade, e que tem objetivos e oponentes dentro de uma narrativa. Dentro desse nicho, o nível de detalhe da simulação e a dependência do jogo em relação a ela variam bastante; por exemplo F1 tem um modelo detalhado de Física, e Sid Meyer Civilization um modelo de desenvolvimento da civilização. Um problema de implementação de software consiste em tratar o acoplamento entre o jogo e o simulador, considerando que sejam dois componentes distintos. Esse trabalho visa a criação de um jogo em que o usuário enfrenta bandidos, sendo que as ocorrências de delitos são geradas em um modelo de simulação com base em estudo de criminologia. O jogo utiliza o simulador para obter dados sobre ocorrência de crimes, o que influencia no comportamento de personagens. Uma versão preliminar do conceito desse trabalho foi apresentada no volume 4 do Workshop de Pesquisa em Computação dos Campos Gerais (WPGCC).

Palavras-chave: Jogos. Modelos de Simulação. Arquitetura de Software. Simulação de Crimes.

ABSTRACT

Many computer games involve simulating real-world experiences, objects, and phenomena. This happens even in a fanciful way like the Super Mario Bros series of games, which portray an individual who walks, runs and jumps under the influence of gravity, and who has objectives and opponents within a narrative. Within this niche, the simulation's detail level and the game dependence on it vary widely; for instance, F1 has a detailed Physics model, and Sid Meyer Civilization a civilization development model. A software implementation problem consists of dealing with the coupling between the game and the simulator, considering them as two distinct components. This work aims to implement a game in which the user faces criminals, and the occurrences of law violations are generated in a simulation model based on a criminology study. The game uses the simulator to obtain information about the occurrence of crimes in a urban environment, which influence the behavior of characters. A preliminary version of the concept of this work was presented in volume 4 of the Campos Gerais Computing Research Workshop (WPGCC).

Keywords: Games. Simulation Models. Software Architecture. Crime Simulation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, ACRÔNIMOS E SIGLAS

ABM	Agent-based Modeling
CG	Computação Gráfica
DLC	Downloadable Content
F1	Formula 1
IA	Inteligência Artificial
NPC	Non-playable Character
UTFPR	Universidade Tecnológica Federal do Paraná
UX	User Experience

SUMÁRIO

1. INTRODUÇÃO

Jogos de computador são softwares com requisitos que podem tornar a sua implementação mais difícil quando comparada com outros tipos de aplicativos. Esses requisitos incluem áreas de estudo diversas, e que às vezes podem ir além do que é tratado em um curso de graduação. Entre as funções implementadas em jogos encontra-se a inteligência artificial (IA), em personagens não controlados pelo jogador (também referidos como *"Non-playable Characters"* ou *"NPCs"*); a simulação de modelos de Física para movimentação de objetos, e efeitos como líquidos e fumaça. Há funções de multimídia que compõem as partes mais artísticas de um projeto, porém muito prezadas para fazer com que os jogos consigam entregar uma sensação específica para o jogador (PICOLI, 2011). Outro tema são redes de computadores, que trata da conexão física ou digital entre máquinas com objetivo de compartilhar informações e se comunicarem em um ambiente, o que permite envio e recebimento de recursos. Finalmente há o usuário, que interage com esse tipo de software de uma maneira particular. Um jogo apresenta uma plataforma interativa que tem como objetivo trazer o jogador para um universo diferente ou para uma tentativa de cópia de experiências presentes na nossa realidade, mas que geralmente não são acessíveis ao público geral.

Muitos jogos de computador envolvem a simulação de experiências, objetos, fenômenos e emoções do mundo real. Elas podem ser observadas em jogos que demonstram fenômenos e ações mesmo que de forma fantasiosa como, por exemplo, a série de jogos do Super Mario Bros, o qual possui retratação de um indivíduo com um objetivo, ações e oponentes dentro de uma narrativa. Ou de modo mais realista em jogos como *"Kerbal Space Program"*, que demonstram consequências físicas da construção e lançamento de foguetes.

Dentro desse nicho existem exemplos em que a simulação existe intrínseca dentro das mecânicas do jogo. Assim há jogos como os da franquia da *"Formula 1"* (como o *"F1 2021"* desenvolvido pela *"Codemasters"* e publicado pela *"EA Sports"*) que tem toda a sua jogabilidade baseada na simulação de carros e a física atrelada à aceleração, movimento e impacto em entidades. Os jogos da franquia *"Sid Meier's Civilization"* (como o mais recente *"Sid Meier's Civilization VI"* desenvolvido pela *"Firaxis Games"*, publicado pela *"2K Games"* e distribuído pela *"Take-Two Interactive"*) utilizam de aspectos históricos do desenvolvimento da civilização da humanidade. Parte desses aspectos se entrelaça com a mecânica de turnos do jogo, que ocorrem em uma escala de tempo que começa com o alvorecer

da civilização nos milênios anteriores a 1d.C. e continua até os tempos atuais e além. Diferente dos jogos *"F1"*, em "*civilization"* não há a uma imposição do uso dessa simulação histórica de forma a impedir ou limitar a jogabilidade, mas há uma limitação temática e dos seus recursos mais internos do jogo. Dessa maneira jogos similares podem usar temáticas mais voltadas à fantasia, porém com o mesmo tipo de jogabilidade, combate e desenvolvimento por eras, assim como *"Stellaris"* (desenvolvido pela *"Paradox Development Studio"* e publicado pela *"Paradox Interactive"*).

"Dwarf Fortress" (desenvolvido pela *"Bay 12 Games"* e publicado pela *"Bay 12 Games"*) é outro bom exemplo; ele é categorizado como um simulador de construção e gestão. O jogo se passa em um universo com temática fantasiosa e seu principal modo de funcionamento consiste em selecionar uma região adequada em um mundo gerado de forma procedural e estabelecer uma colônia ou fortaleza bem-sucedida. No meio desse processo o jogador é confrontado com elementos de gerência de pessoal e recursos, desastres naturais e *"NPCs"* hostis.

A simulação é a imitação da operação de um processo ou sistema do mundo real ao longo do tempo. As simulações requerem o uso de modelos; o modelo representa as principais características ou comportamentos do sistema ou processo selecionado, enquanto a simulação em si é a execução, ou seja, a evolução do modelo ao longo do tempo.

Uma simulação de computador é uma tentativa de modelar uma situação da vida real ou hipotética em um ambiente virtual para que possa ser estudada para ver como o sistema funciona. Alterando variáveis na simulação, podem ser feitas previsões sobre o comportamento do sistema. É uma ferramenta para investigar virtualmente o comportamento de um sistema.

A união entre jogos de computador e softwares de simulação gera mais requisitos específicos, dependendo do sistema ou mecanismo que está sendo representado. Exemplificando, a simulação de Física em "F1" e a simulação histórica em "Civilization" são muito diferentes. No primeiro caso é preciso tratar items como atrito, aceleração e problemas de cálculo, enquanto no segundo tem-se o controle de NPCs individuais e em grupo que devem oferecer uma experiência de jogo, mas ao mesmo tempo retratar um episódio histórico. Nos dois exemplos, a programação do software deve tratar todas as funções necessárias a um jogo, mais o que for requerido pelo simulador.

1.1 Objetivos

Este TCC tem como objetivo projetar e implementar um protótipo de um jogo que contém um modelo de simulação.

Objetivos Específicos

A realização do TCC pode ser dividida nos seguintes objetivos específicos:

- construção de uma simulação baseada em agente simplificada;
- construção do jogo;
- a junção dos dois de forma coerente e modular.

O protótipo do jogo visa ter uma versão bruta do aplicativo, com um enfoque na jogabilidade e nos elementos básicos de *"user experience"* ou *"UX"*, assim criando um ambiente com elementos audiovisuais rústicos, que tem somente como objetivo uma ilustração simples do cenário e dos personagens.

2. REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico do trabalho. A Seção 2.1 discorre sobre a simulação computacional. A Seção 2.2 exibe um panorama geral sobre jogos de computador. Por fim a Seção 2.3 apresenta a visão geral sobre diagramas de classe.

2.1 Simulação computacional

Nesta seção são abordados os assuntos referentes à simulação computacional.

2.1.1 Visão geral

A simulação se tornou comum na modelagem de sistemas que podem ser observados na criação de motores de Física como o PhysX da NVIDIA, na Química, Biologia, e em sistemas humanos em economia e ciências sociais, podendo ser exemplificado pelo modelo de Schelling (SCHELLING, 1971). Porém também é muito útil e bem-visto em meio à engenharia de modo a obter informações sobre a operações desses sistemas (GAVIRA, 2003).

Simulações são frequentemente usadas como um complemento ou substituição para experiências reais. Sistemas que não permitem experimentos, como uma cidade, podem ser estudados com um modelo de computador. Em sistemas urbanos o comportamento humano atribui aleatoriedade, fazendo com que a compreensão mecânica de fenômenos não consiga ser propriamente refletidos por equações e modelos matemáticos para conseguir simular o seu comportamento com precisão (ROTH, 2019).

Outro exemplo são sistemas para os quais soluções analíticas de forma fechada não são possíveis; há vários casos assim, em Física, Química, Biologia e outras áreas. Simulações que utilizam entradas e elementos aleatórios inevitavelmente atribuem um grau de incerteza nos resultados e por isso é bom tratar os resultados como aproximações e utilizar em conjunto testes estatísticos (MCHANEY, 2009).

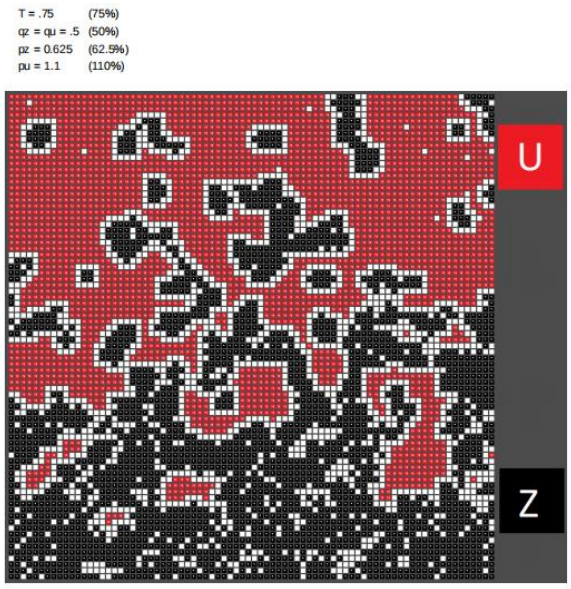
Mesmo com todas as limitações, há modelos que ajudam em um escopo geral, permitindo uma visão mais controlada de interações entre agentes em ambientes controlados, algo que se desenvolveu de uma forma mais acessível com o aprimoramento dos computadores.(MCCOWN, 2014).

Em campos como a Sociologia pode-se citar o modelo de segregação criado por Thomas Schelling (1971). É um modelo baseado em agentes, ou seja, um modelo que simula a

interação simultânea de várias entidades individuais, mas consegue demonstrar fenômenos complexos que resultantes de interações entre elas. O modelo sugere que decisões individuais também contribuem para a uma segregação entre grupos. Dada a simplicidade do modelo há trabalhos como o de Hatna e Benenson (2012) que busca adicionar variáveis ao modelo para que ele possa ser aplicado com mais precisão de forma a se encaixar melhor com uma situação urbana atual.

O modelo original de Schelling é definido em uma grade quadrada de dimensões arbitrárias. Há a introdução de agentes individuais e espaços vazios no ambiente, sendo que cada um deles pode ocupar um espaço da grade por vez, e são divididos em dois grupos. Os agentes desejam que os espaços adjacentes ao seu redor sejam ocupados por indivíduos pertencentes ao mesmo grupo, dentro de uma tolerância. A cada iteração os agentes verificam seus vizinhos e caso eles estejam ocupados por um número maior de agentes do grupo oposto do que o tolerado, eles se mudam para um local vago. Essas iterações continuam até que todos estejam satisfeitos. A Figura 1 apresenta um exemplo do resultado aplicado de modelo de segregação de Schelling.

Figura 1 - Modelo de segregação de Schelling



Fontes: Autoria própria

O modelo de Schelling é um bom ponto de partida para explicação, estudo e entendimento do conceito de modelos baseados em agentes pois abrange tópicos básicos além de ser utilizado pela comunidade acadêmica.

2.1.2 Simulação baseada em agentes

Modelos baseados em agentes, também referidos como *"ABM"* (*Agent-based modelling*) ou *"Micro modelling"* (GUSTAFSSON; STERNAD, 2010), são modelos de simulação que se baseiam no conceito de eventos e entidades, criando ambientes onde essas entidades podem interagir entre si ou com o ambiente, o que permite recriação do funcionamento dos sistemas. Esse tipo de modelo é geralmente usado para problemas que necessitam de uma simulação detalhada e fiel ao original. É possível representar domínios de aplicação variando de pacotes de rede até plantas e animais dentro de um sistema.

Modelos baseados em agentes possuem padrões que seguem regras de interação dinâmica, fazendo com que as interações dos agentes possam resultar em diferentes tipos de comportamentos complexos.

Estes modelos baseados em agentes complementam os métodos analíticos tradicionais, métodos estes que permitem caracterizar o equilíbrio de um sistema. Essa contribuição pode ser um dos principais benefícios desse tipo de modelagem. Exemplos de aplicação são Biologia (SITUNGKIR, 2004), resolução de uma variedade de problemas de negócios e tecnologia, comportamento organizacional e cognição (LIMA et al., 2009), crescimento e declínio de civilizações antigas, e em psicologia (HUGHES et al., 2012).

2.1.3 Simulação de crimes

O estudo de crimes com uso de modelos baseados em agentes é relativamente novo na comunidade da ciência criminal. Há diferentes teorias que tentam apresentar explicações para o problema. Por exemplo, a ocorrência de crimes em regiões específicas, como essas regiões evoluem em tamanhos e longevidade "macroscópicos", como são afetadas pelos comportamentos "microscópicos" e os fatores que permitem o seu surgimento. Geralmente essas teorias abrangem a relação entre infratores, vítimas, agentes de fiscalização e a geografia local. Em geral, o crime só pode ocorrer se um aspirante a criminoso encontrar a vítima ou o alvo certo sem medidas de segurança eficazes. Nesse contexto, a estrutura do ambiente urbano pode desempenhar um papel importante na limitação da circulação de criminosos e potenciais alvos. Por exemplo, características como volume de tráfego, riqueza abandonada, densidade populacional, e a distribuição de pessoas, influenciam nos padrões de criminalidade (TITA et al., 2007).

O simulador de Devia e Weber (2013) consiste de agentes, de um ambiente e algumas regras de comportamentos individuais determinadas pelas características da cidade ou do ambiente que se pretende simular. O modelo se apoia em estudos como o *"Enviromental Criminology"* (BRANTINGHAM; BRANTIGHAM, 1995) ao tratar de situações em como certas estruturas, como agências bancárias, atraem a atenção de grupos criminosos de forma diferente comparada com outros comércios espalhados pela cidade. Assim os ambientes simulados, fictícios ou reais, possuíam separações estruturais, contendo residências, onde agentes vivem ou passam tempo, empresas de varejo, que são lugares em que há comercio ativo por parte dos agentes, bancos, companhias de seguro e diversos serviços financeiros onde há presença de movimentação de dinheiro, escritórios, que categorizam qualquer lugar de trabalho, estabelecimentos educacionais, exemplificado por escolas e universidades, e acessos a transportes públicos.

Cada tipo de estrutura possui suas próprias características, localização dentro do ambiente e peso na decisão de tomada de crimes pelo agente. O ambiente é influenciado por um sistema de tempo, em que há períodos do dia relacionados diretamente com intervalos de horários, como observado na Tabela 1.

Tabela 1 - Horários e períodos do dia

Period	Hours
Early morning	00:00–05:59
Morning rush hour	06:00–08:59
Morning	09:00–11:59
Noon	12:00–14:59
Afternoon	15:00–17:59
Afternoon rush hour	18:00–20:59
Night	21:00–23:59

Fonte: Devia e Weber (2013)

Os agentes criados para popular a simulação consistem em civis, que são cidadãos que não cometem atos ilegais, aplicados no modelo como potenciais vítimas; criminosos, a classe de agentes que possui o potencial de cometer crimes; e policiais, que são agentes que se responsabilizam e se mobilizam para evitar crimes. Os agentes seguem regras de comportamentos dentro da simulação onde são definidas rotas, alcance de visão, velocidade de movimento, e o comportamento de cada tipo de indivíduo em relação ao ambiente (DEVIA; WEBER, 2013).

Um ponto importante é a análise feita pelo criminoso para definir se há ou não a realização do delito dadas certas condições. Essa análise depende de fatores de como presença

policial, número de civis próximos e característica pessoal, ou seja, se ele possui uma personalidade cuidadosa, neutra ou abusada. Um exemplo de função de probabilidade de crime pode ser observada na Tabela 2, em que *c* é o número de civis e *p* o número de policiais que estão próximos do criminoso.

Tabela 2 - Função de probabilidade de ação do criminoso

Function	Prob. of bag snatching	Prob. of robbery
Bold	$prob_L = \begin{cases} \frac{c}{30} & \text{if } p = 0 \\ \left[\frac{c}{30 \cdot (p+1)}\right]^{\frac{3}{2}} & \text{if } p \geqslant 1 \end{cases}$	$prob_A = \frac{0.1}{(p+1)^3 \cdot c^3}$
Normal	$prob_L = \begin{cases} \frac{c}{20} & \text{if } p = 0 \\ \left[\frac{c}{20 \cdot (p+1)}\right]^{\frac{3}{2}} & \text{if } p \geqslant 1 \end{cases}$	$prob_A = \frac{0.1}{(p+1)^3 \cdot c^2}$
Cautious	$prob_L = \begin{cases} \frac{c}{10} & \text{if } p = 0 \\ \left[\frac{c}{10 \cdot (p+1)}\right]^{\frac{3}{2}} & \text{if } p \geqslant 1 \end{cases}$	$prob_A = \frac{0.1}{(p+1)^2 \cdot c}$
Factor	$f_L \sim U[0.9; 1.1]$	$f_A \sim U[0.8; 1.2]$

Fonte: Devia e Weber (2013)

2.2 Jogos de computador

Jogos de computadores interativos surgiram como um conceito por volta de 1961, com a criação do *"Spacewar";* nele dois jogadores possuem o controle de duas naves e se enfrentam. O jogo se passa em uma perspectiva 2D com uma visão *"top-down"* do cenário (GONSALVES, 2018). O jogo foi criado por Steve Russell, estudante do MIT (Instituto de Tecnologia de Massachusetts) (KENT, 2001). Jogos, sendo uma forma de mídia, tem como objetivo permitir o usuário vivenciar experiências, sensações, aventuras e um universo que pode ou não se diferenciar da realidade. Diferente das outras plataformas de mídia de entretenimento, os jogos dependem diretamente da interação entre usuário e sistema para que a evocação desses sentimentos seja possível (ZAWALSKI, 2019).

Esse tipo de software costuma ter requisitos que podem vir a tornar a sua implementação mais complexa quando comparada com outros tipos de aplicativos. Isso acontece principalmente pela reunião de várias técnicas diferentes, como IA, computação gráfica, requisitos de tempo real, juntamente com equipes multidisciplinares de artistas.

A simulação física geralmente vista em jogos é utilizada para controle e direção de objetos, efeitos como líquidos e fumaça, que compõem partes mais artísticas de um projeto, porém muito prezadas para fazer com que os jogos consigam entregar uma sensação específica para o jogador. Pode-se citar como exemplo a utilização do motor de física da *"NVIDIA"*, o

"PhysX", no jogo *"Kerbal Space Program"*, o qual foi elogiado por sua mecânica orbital precisa dinâmica Newtoniana.

Jogos de Computador podem, também, se aproveitar muito de interações com redes distribuídas e uso de servidores. A comunicação entre redes toma vantagem de múltiplos clientes poderem se comunicar entre si por meio de um servidor, o que abre algumas oportunidades para armazenamento de informações pessoais e interação entre usuários, o último sendo utilizado quando há foco em multijogador (PICOLI, 2011). A inclusão de um sistema distribuído em jogos mais elaborados permite também a distribuição de atualizações e *patches* de uma forma mais rápida, sem depender de meios físicos ou da utilização de fóruns para divulgação.

A implementação de uma interface gráfica depende do gênero do jogo, gosto dos usuários e as mecânicas que se pretende usar. Com isso tem-se a implementação de técnicas e métodos de Computação Gráfica (CG) para tornar a idealização do jogo do programador em uma realidade. A computação gráfica é a reunião de conjuntos de técnicas que permitem a geração de imagens a partir de modelos computacionais de objetos e dados.

Em CG, modelos são usados para representar entidades e fenômenos do mundo físico real no computador. Existem várias categorias ou métodos de construção de modelos tridimensionais, cada uma contém suas vantagens e desvantagens, assim tendo usos que podem ser mais especializados dependendo das suas exigências de processamento e facilidade de gerenciamento. Modelagem consiste em todo o processo de descrever um modelo, objeto ou cena, de forma que se possa desenhá-lo (MANSSOUR, 2006).

Com todas essas especializações e focos de programação que são utilizadas de forma conjunta para a implementação de um jogo, o processo de desenvolvimento se torna complexo, pois há grande número de particularidades que se entrelaçam.

Jogos com simulações podem apresentá-las de uma forma intrínseca ao jogo ou de forma modular ou separada. Jogos como os da franquia da *"Formula 1"* tem toda a sua jogabilidade baseada na simulação de carros e a física atrelada à aceleração, movimento e impacto em entidades. Esse tipo de jogo costuma utilizar de motores de física para realizar a parte mais bruta da simulação, ou seja, utiliza-se um conjunto de bibliotecas que trata dos problemas relacionados com cálculo de velocidade, aceleração e colisões entre entidades. A precisão desses cálculos visa demonstrar um comportamento realista para que a experiencia desejada seja a mais próxima do real possível.

Há casos como os jogos da franquia *"Sid Meier's Civilization"* que utilizam a simulação para representar aspectos históricos sobre o desenvolvimento de civiçizações. Os turnos ocorrem em uma escala de tempo iniciada com o alvorecer da civilização nos milênios anteriores a 1 d.C. e expandida até os tempos atuais e além, dependendo da duração de uma partida. Diferente do jogo *"F1"*, em "*Civilization"* não há a imposição da simulação histórica na jogabilidade, mas na temática e nos seus recursos internos. Isso abre portas para que jogos similares possam usar temáticas mais voltadas à fantasia, porém utilizando do mesmo tipo de jogabilidade, combate e desenvolvimento por eras. Um exemplo similar é *"Stellaris"*, que compartilha da categoria 4X com *"Civilization"*. Essa classe de jogos é um subgênero de jogos de estratégia que envolvem construção de impérios com ênfase em desenvolvimento econômico e tecnológico bem como uma série de rotas militares e não militares para a supremacia.

"Dwarf Fortress" é outro bom exemplo de jogo que possui uma simulação intrínseca; ele é categorizado como um simulador de construção e gestão. Diferente do caso que é apresentado pelo *"Civilization"*, *"Dwarf Fortress"* leva o usuário a um mundo fantasioso, no qual se tem a necessidade de controlar recursos e uma população de indivíduos, assim direcionando a atenção da simulação ao comportamental de cada indivíduo, dado que eles possuem características individuais, gostos, preferências e limitações.

2.2.1 Visão geral sobre desenvolvimento de jogos

O projeto de um jogo é um processo que começa com uma ideia ou conceito que, normalmente, é desenvolvido a partir de um ou mais gêneros já existentes dentro da comunidade e do mercado de jogos (BATES, 200). Isso é muito comum na indústria de jogos e dado que alguns títulos ficam extremamente populares e acabam sendo marcos dentro do gênero. Exemplos disso podem ser observados com o lançamento de "DOOM", que influênciou o mercado de jogos de *"first person shooters"*(FPS) da época, mesmo não sendo o primeiro do seu tipo. Um exemplo mais recente se tem com a série de jogos "Dark Souls" que definiu um gênero e criou um nicho próprio dentro da comunidade de jogos.

As concepções de *design* e desenvolvimento, podendo pertencer a um ou mais gêneros, dão a liberdade para que projetistas experimentem com diferentes combinações entre os gêneros de forma a tentar criar um jogo que se destaque dentro da comunidade, ou a fim de dar uma individualidade ao seu produto. Com isso, é comum que os desenvolvedores de jogos se desviem dos métodos formais de desenvolvimento e produção levados por uma falsa racionalização de que jogos são apenas arte, não ciência. (BETHKE, 2003).

Um projetista de jogos geralmente escreve um documento inicial de proposta de jogo, que descreve o conceito básico, jogabilidade, lista de recursos, cenário e história, público-alvo, requisitos e cronograma e, finalmente, estimativas de equipe e orçamento. Não existe um método único de desenvolvimento, porém existem algumas diretrizes básicas (MCGUIRE; MORGAN, 2009). Essa diretriz do processo de desenvolvimento geralmente se divide em pré-produção, produção e pós-produção, cada uma delas com suas próprias etapas e metas de desenvolvimento.

A pré-produção ou fase de design é o estágio de planejamento do produto, focada no desenvolvimento de ideias e conceitos e na produção de documentos iniciais de design. O objetivo do desenvolvimento de conceito é produzir um plano de produção (BETHKE, 2003), que é a documentação clara e de fácil compreensão que descreva todas as tarefas, cronogramas e estimativas para a equipe de desenvolvimento (BATES, 2004).

Geralmente há a divisão da pré-produção em estágios. O *"high concept"*, é o estágio em que se tem o desenvolvimento inicial da ideia do que será produzido e poderia ser classificado como o processo de *"brainstorming"*. Em seguida *"pitch"* que é descrito em um pequeno documento de resumo que trata das questões envolvendo vendas, marketing e monetização do produto (MOORE; NOVAK, 2010). Depois *"concept",* que é uma junção dos dois estágios anteriores, porém aprofundando nos pontos que remetem gênero do jogo, descrição da jogabilidade, características, cenário, história, público-alvo, plataformas de hardware, cronograma estimado, análise de marketing, requisitos da equipe e análise de risco (BATES, 2004). O *"Game Design Document"* é um documento altamente descritivo do design de um videogame (BATES, 2004), e pôr fim, tem-se a prototipação do jogo.

A produção é o estágio mais extenso do desenvolvimento. Nela acontece o desenvolvimento principal do código, personagens, criaturas, adereços, ambientes e toda a fundação programável. Esse estágio do desenvolvimento pode ser dividido da seguinte maneira:

- *"Design",* que é o trabalho em conjunto com os artistas, renderiza modelos de personagens, criam designs e ambientes de níveis dinâmicos e imersivos, iteram nas interfaces e assim por diante (BATES, 2004). Geralmente esse processo ocorre em conjunto com o início da prototipação e programação pois a programação somente ocorre quando a base intelectual do jogo está sólida e o design avançado do jogo necessita saber como as mecânicas do jogo estão funcionando.

- *"Programming"*, etapa de prototipação de ideias de trabalho, incorporando novos recursos e corrigindo os bugs introduzidos ao longo do caminho.

- *"Art production"*, etapa que envolve a produção de gráficos e áudio. A maioria dos jogos requer muitos ativos criativos, dependendo de Artistas, compositores de música e dubladores. Para garantir que todos os elementos se encaixem há o trabalho conjunto desses profissionais com os profissionais de design (BETHKE, 2003).

- *"Testing"*, etapa que tem como foco o teste e certificação de que as mecânicas implementadas estão funcionando como esperado. Esses testes ocorrem durante todo o processo de desenvolvimento e programação do jogo, no início geralmente ocupa um período relativamente pequeno, dado que não há muito conteúdo implementado a nível de protótipo, porém à medida que o desenvolvimento se aproxima do fim, maior se torna a quantia de conteúdo pelo qual testadores tem que passar. À medida que o código amadurece e os recursos de jogabilidade se solidificam fazendo com que seja necessário inclusão de controles de teste mais rigorosos e amplos, como testes de regressão, para garantir que novas atualizações na base do código não alterem as partes funcionais do jogo.

A pós-produção pode ser resumida como o processo de manutenção do jogo após o seu lançamento. Ela geralmente se divide na correção de bugs e implementação de *"patches"* e no lançamento de novos conteúdos geralmente na forma de *"updates", "game-balancing patches"* e *"DLCs"*. O processo de manutenção de jogos pode ser visto como algo mais recente em vista da tecnologia que se tinha na época de produção dos jogos mais antigos, no caso a entrega de um *"patch"* de correção não era tão simples quanto é atualmente, não havia essa conexão e facilidade provida pelo acesso à internet.

De forma a desenvolver e expandir um dos tópicos tocados durante a explicação do desenvolvimento, o *"indie development"* é os desenvolvimentos de jogos de forma independente. Jogos independentes ou jogos *"indie"* são produzidos por indivíduos e pequenas equipes sem afiliações de desenvolvedor ou editor em grande escala. Muitos desenvolvedores amadores começam suas carreiras como criadores de *"mods"* para jogos já existentes.

Desenvolvedores independentes geralmente confiam em esquemas de distribuição na Internet. Plataformas como a Steam costumam abrir espaços específicos para jogos desse tipo e permitem um modelo de monetização mais generalizado. Outro caminho é o modelo de

doações. como incentivo para continuação do desenvolvimento ou como ajuda para manutenção e aprofundamento na pós-produção do jogo.

3. DESENVOLVIMENTO

O projeto foi desenvolvido primariamente na plataforma Godot 3.4.4, um motor de jogo multiplataforma gratuito e *"open-source"*, aberto e lançado sob a licença do MIT. O Godot 3.4.4 visa oferecer um ambiente de desenvolvimento de jogos totalmente integrado. A rigor ele não necessita de outras ferramentas além daquelas usadas para artes visuais e músicas.

Em conjunto com o Godot 3.4.4, particularmente para a criação de conteúdo visual, foi utilizado o Aseprite e o Tilesetter. Com o Aseprite foi possível a criação de alguns *"sprites"* animados em 2D assim como *"tilesets"* simples para uso pessoal. A importação de ambos somente foi possível com a utilização do tilesetter, que é um gerador de *"tilesets"* e editor de mapas.

3.1 Implementação do simulador

O conceito do simulador dentro do jogo começou como uma intenção de cooperação com um projeto de mestrado no Programa de Pós-Graduação em Ciência da Computação, PPGCC, da UTFPR de Ponta Grossa. A implementação do simulador se baseou no estudo realizado por Nelson Devia e Richard Weber em *"Generating crime data using agent-based simulation"* (DEVIA; WEBER, 2013). O modelo apresenta situações e características que tornavam natural o processo de criar um jogo *"standalone"*.

O modelo implementado no TCC trata de uma escala menor do estudo e não estão implementados *"NPCs"* do tipo policial. Foi criado um protótipo de civis na simulação para um resultado mais realista e há a presença de prédios policiais espalhados pelo mapa, que podem impactar na presença e nos instanciamentos. Foram utilizados dados relacionados a personagens com características neutras (personagens dados como civis que não teriam um impacto significativo no jogo) para que fosse possível ter a incidência de crimes em dadas localizações com o fator de atividade de civis na região.

A decisão de cometer um crime é tomada com uma certa probabilidade que difere para os tipos de delitos, definidos como roubos ou assaltos. Eles têm o processo de decisão mais complexo do modelo pois englobam aspectos como quantidade e presença de civis, impacta na probabilidade de o criminoso cometer um delito. A Tabela 3 mostra fórmulas que são adaptações do estudo de Devia e Weber e que não consideram agentes policiais.

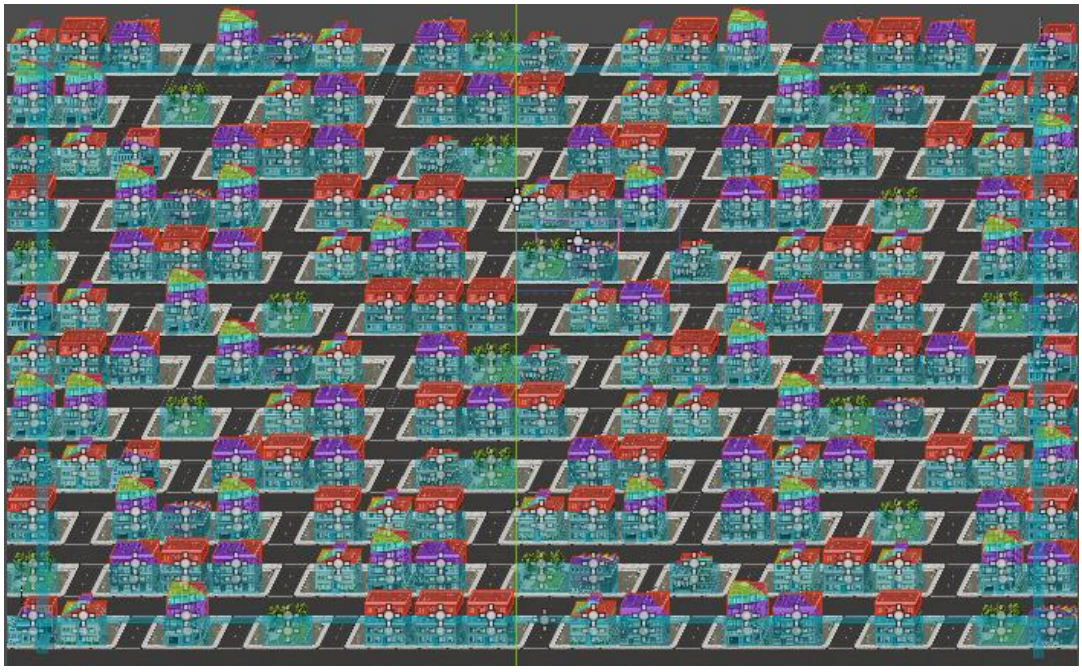
Tabela 3 - Probabilidade de infrações por tipo e por personalidade

Tipos	Roubos	Assaltos
Atrevido	$prob = \frac{c}{10}$	$prob = \frac{0,1}{c}$
Normal	$prob = \frac{c}{20}$	$prob = \frac{0,1}{c^2}$
Cuidadoso	$prob = \frac{c}{30}$	$prob = \frac{0,1}{c^3}$

Fonte: Autoria própria (2022)

A ação dos agentes e a área de influência de crimes na cidade são influenciadas pela divisão do mapa geral, que é constituída em uma grade que define a cidade. O mapa geral foi montado utilizando uma projeção *"top-down"* e possui naturalmente uma grade para orientação espacial no ambiente 2D. Isso facilita tanto a construção do mapa quanto a categorização de regiões, de maneira similar ao mapa *"Gridville"* utilizado por Devia e Weber (2013). O mapa do jogo e sua grade podem ser observado na Figura 2.

Figura 2 - Mapa Geral com grid, limite e hitboxes



Fonte: Autoria Própria (2022)

Com a divisão da cidade pronta foi feita a categorização arbitrária das áreas. Elas foram definidas dentro dos seguintes tipos: residencial, lazer, comercial, socio-comercial e

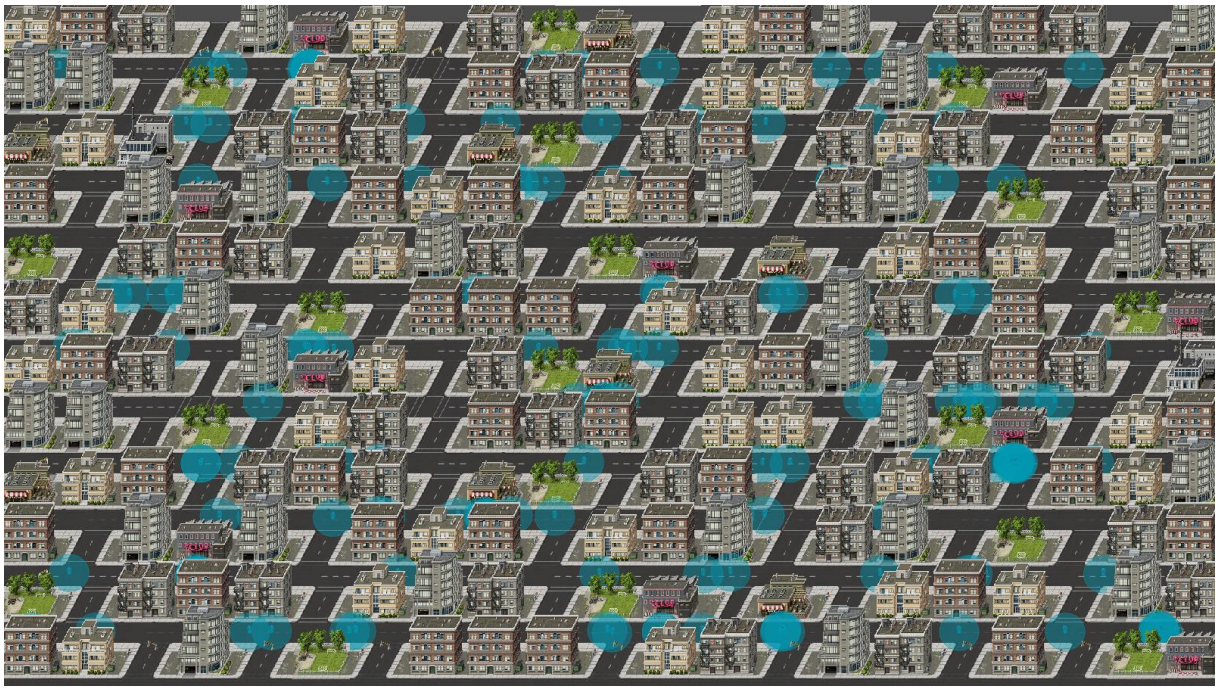
supervisionados. Essa categorização tentou se manter aleatória. Com as localizações da cidade definidas foi dado início ao desenvolvimento dos agentes da simulação, sendo eles divididos em civis e criminosos.

Os civis são considerados agentes passivos que somente vagam pela cidade de forma pré-definida. No caso da nossa simulação eles surgem nas áreas residenciais, criando suas rotas tendo como prioridade o rumo a áreas de categoria comercial, lazer e socio-comercial. Eles vagam por essas áreas por um tempo também aleatório e depois retornam para as áreas residenciais. Dado que não há um sistema de tempo implementado na simulação não existe um conceito de dia ou noite, então os tempos são aleatórios com um limite no período máximo de permanência fora da sua área de início. Eles não cometem crimes e são considerados como as vítimas em potencial dos crimes que ocorrem na cidade.

Os criminosos agem de forma similar aos civis, porém possuem a chance de cometer crimes e suas rotas têm uma preferência que leva em consideração, o acúmulo de civis em certos pontos da cidade e a presença dos prédios policiais espalhados pela cidade. Os criminosos escolhem suas rotas aleatoriamente; a presença de delegacias não impede que ocorra a presença deles nesses locais, levando em consideração a atividade de civis e se houve ou não crimes recentes cometidos na área. A ocorrência de múltiplos delitos seguidos em uma área faz com que os criminosos evitem uma rota que passe por ela.

A divisão do mapa foi feita a partir do conteúdo dos seus sprites. Os clubes, tem pesos maiores para a instanciação de personagens. Locais comerciais como pizzarias, ficaram com um peso médio, e lazer que foram representados como os parques, os quais foram atribuídos com peso mais baixo, porém não nulo. A atribuição desses pesos teve influência nos estudos de Devia e Weber (2013) e Cohen e Felson (1979).

Figura 3 - Instanciamento de *"NPCs"* no mapa geral



Fonte: Autoria própria (2022)

O Instanciamento dos *"NPCs"* utiliza cálculos de probabilidade de sucesso. Essa abordagem faz com que as áreas residenciais não sejam inteiramente seguras ou sem a presença de criminosos.

3.3 Jogo

Nesta seção são abordados os aspectos referentes ao Jogo.

3.3.1 Projeto geral do jogo

O jogo consiste em um personagem principal que é controlado pelo jogador. Esse personagem possui elementos básicos de movimentação, ataque e vida. O protagonista tem o papel de um vigilante dentro do jogo, tendo como seu objetivo o confronto contra *"NPCs"* hostis.

Esses *"NPCs"* também possuem movimentação, pontos de ataque e vida, porém sua movimentação e padrão de comportamento é programada dentro do jogo e seus Instanciamentos pelo mapa são controlados pela simulação. O objetivo dos inimigos é se opor ao personagem principal e, se necessário, perseguir e eliminar ele.

O universo em que se passa o jogo se divide em um mapa geral, denominado de *"over world"* e de mapas instanciados dinamicamente ("salas") acessíveis ao completar objetivos específicos. Esses mapas separados corresponderiam ao esconderijo das personagens agressivas, consistindo então de inimigos mais fortes e um inimigo final, denominado de "BOSS".

Com o foco desse projeto consiste em demonstrar a simulação em ação com o apoio da plataforma que é o jogo, não há um aprofundamento na parte estética ou no desenvolvimento de uma história.

3.3.2 Descrição detalhada

O vigilante, personagem controlado pelo jogador, possui uma matriz de movimentos e ações que consiste em movimentação nas 8 direções básicas, um ataque direcional e um rolamento com frames de invencibilidade. O vigilante possui, também, um contador de vida e animações de *"feedback"* para dano.

Tanto os inimigos quanto o vigilante possuem uma quantia fixa de vida e de dano, sendo ela, no caso dos inimigos, dependente da dificuldade deles. O dano aplicado ou recebido por qualquer entidade no jogo é definido pelas caixas de colisão dos seus modelos e a condição de morte se define pela quantia de vida atual menos a quantia de dano recebido, sendo que a vida de cada entidade não pode ser menor do que 1 para ela continuar existindo.

O mapa geral é fixo e consiste em ruas que definem espaços possíveis de movimentação para as entidades do jogo e prédios que definem os quadrantes do mapa. Diferente do *"over world"*, os mapas instanciados são gerados de forma procedural e possuem uma geração de entidades por sala[1].

Ambos os mapas possuem objetivos distintos. O mapa geral é uma área urbana onde é aplicado o principal aspecto de simulação para ocorrências de crimes e circulação de inimigos. O mapa instanciado se apresenta como uma forma de desafio para a conclusão de uma interação do jogo, nesse espaço se localizariam inimigos mais difíceis com o intuito de simular uma base de operações sendo protegida. No seu fim existe o *"BOSS"* do jogo, um inimigo com uma dificuldade maior em comparação aos apresentados até o momento e que define o objetivo final.

---

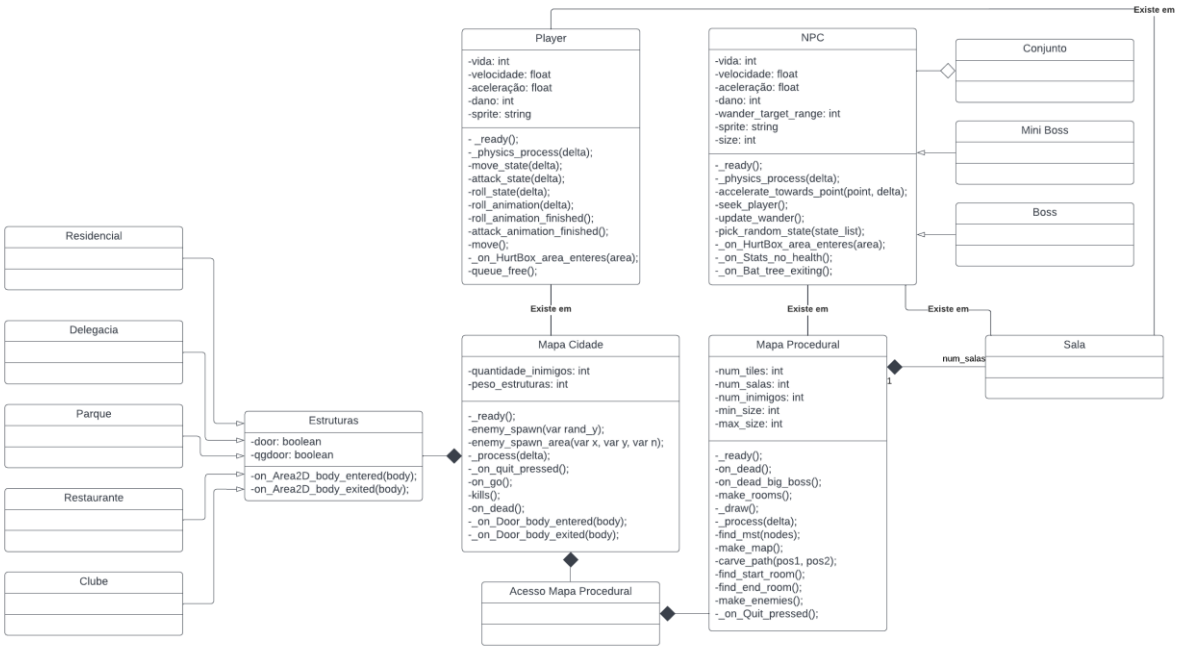
[1] Código baseado na solução de Chris Bradfield, do site Kids Can Code.

3.4 Implementação do jogo

A implementação do jogo se iniciou com o desenvolvimento do mapa principal, que foi feito a partir de *"sprites"* pré-existentes de uso livre criados por David Baumgart e publicados em *"opengameart.org"*. Em paralelo foi realizado o desenvolvimento do personagem principal, sua vida, velocidade, ataques, dano, movimentação, *"hitboxes"*. Logo depois do personagem principal estar terminado foram feitos os *"NPCs"* hostis, que continham somente um *"hitbox"* para registro de dano e vida. O desenvolvimento do jogo pode ser organizado, no seu início, o diagrama de Classe da Unified Modeling Language (UML) apresentado na Figura 4. Como um jogo é programado de forma a utilizar múltiplos objetos como atributos, o diagrama apresenta somente a abstração inicial do conceito desenvolvido.

No diagrama as salas individuais, que são criadas pelo mapa procedural, foram separadas em objetos individuais, pois elas existem somente nessa instancia do mapa, podendo conter um número de entidades de diferentes tipos. Os tipos de "NPCs" que são diferentes do básico, no caso o "Mini Boss" e o "Boss" somente podem existir dentro dessas salas, ou seja, não podem aparecer no "Mapa Cidade". O player pode existir em ambos a cidade e labirinto, porém não simultaneamente. O objeto "Acesso Mapa Procedural" permite haver a ligação e transferência do jogador entre os dois tipos de mapa, sendo uma entidade que necessita de interação para funcionar.

Figura 4 - Diagrama UML

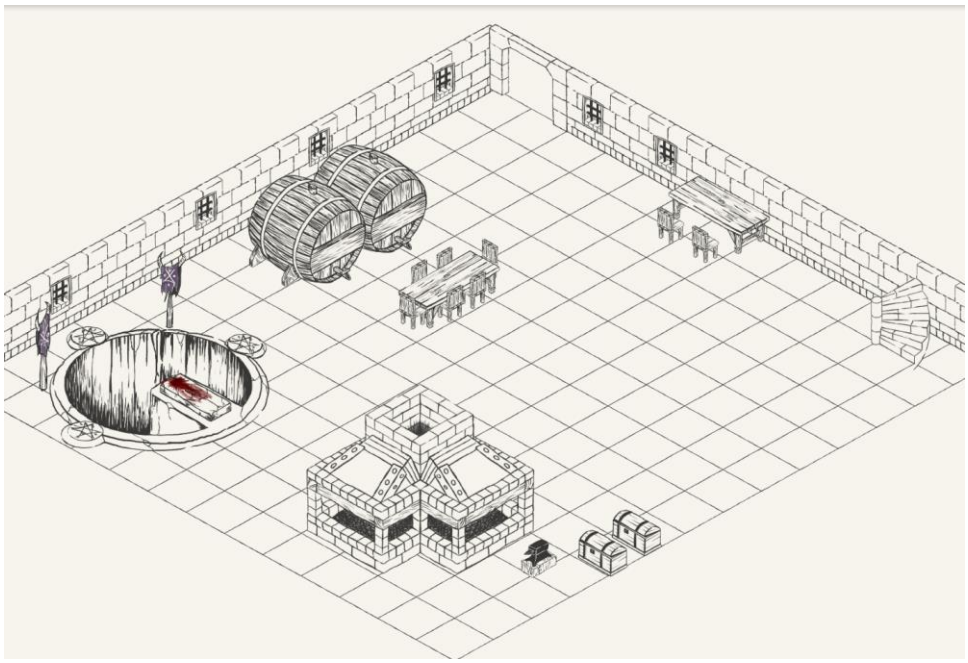


Fonte: Autoria própria (2022)

3.4.1 Implementação do mapa

A implementação do mapa teve início com a definição da sua orientação e mecânicas relacionadas, no caso a movimentação de entidades e o instanciamento delas pelo mapa. No início do desenvolvimento houveram conflitos com o tipo de mapa, orientação da câmera e a movimentação do personagem. A primeira versão do mapa geral apresentava uma câmera com orientação isométrica, conforme ilustrado na Figura 5, porém o desenvolvimento do protótipo utilizava de sprites que não possuíam uma compatibilidade boa com essa orientação, então foi decidido por refazer o mapa e utilizar uma visão 2d mais parecida com um *"top-down view"* do cenário. Os *"sprites"* utilizados para o uso dessa perspectiva são ilustrados pela Figura 6.

Figura 5 - Exemplo de plano isométrico



Fonte: Autoria Própria (2022)

Figura 6 - Tiles utilizados para criação do mapa geral

Fonte: https://opengameart.org/content/turfwars-pixel-city-tile-set

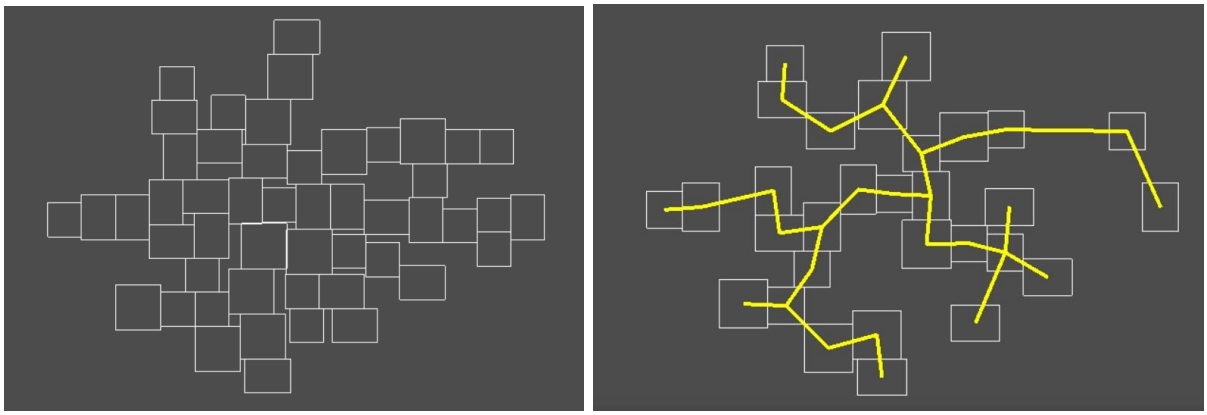
Com o estilo do mapa aberto definido, foi necessária somente a criação da grade do mapa e a distribuição dos *tiles*. Junto com essa distribuição foram definidas as ruas e as partes dos prédios que poderiam ser atravessadas, dado que os prédios definem as partes sólidas e impassáveis do mapa, o que permite definir o Instanciamento dos *"NPCs"* e sua movimentação dentro desses limites. Esse ajuste de design foi necessário para que fosse possível o jogador transitar pelo mapa de forma natural.

Em contraste com a estaticidade do mapa geral, os mapas instanciados são individuais e gerados no momento em que o jogador interage com a entrada dele. A criação do mapa começa com a criação de "X" quadrados, sendo "X" um número arbitrário definido no código de criação do mapa, que são possíveis salas dentro desse labirinto. Após a definição e criação dessas salas é utilizado um algoritmo de *"pathfinding"* "A*" (A-estrela) para a definição das salas e dos caminhos que serão gerados. Tais processos podem ser observados na figura 7.

Com o mapa pronto os inimigos são divididos por entre as salas e o "BOSS" é colocado na sala final, o jogador então volta ao controle do personagem principal e é colocado na sala inicial desse labirinto. O algoritmo "A*" já vem implementado dentro do Godot 3.4.4 para uso na forma da função *"AStar.new"*.

A Figura 7 apresenta a criação de possíveis salas que compões o mapa procedural e em seguida o "A*" em funcionamento para definir o melhor caminho da origem até as extremidades e assim filtrar as salas que serão selecionadas para a formação do mapa final.

Figura 7 – Criação das possíveis salas e escolha do caminho



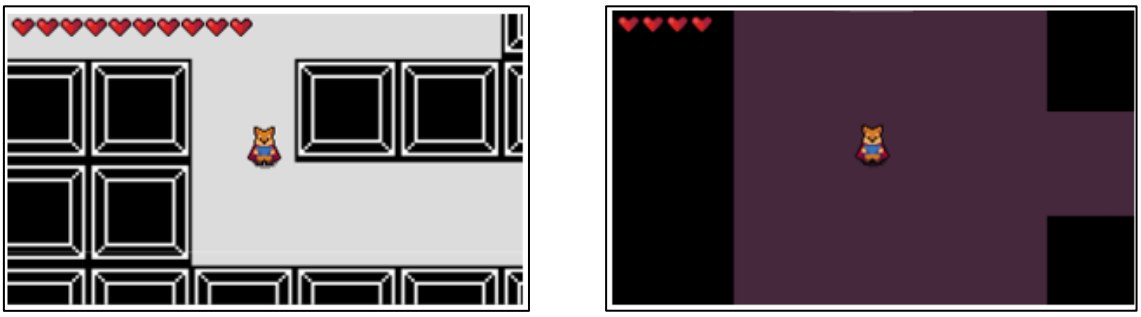
Fonte: Autoria Própria (2022)

Foram construídos alguns *"tilesets"* para uso dentro dos mapas procedurais do jogo, porém dado a um erro desconhecido no momento em que as paredes eram geradas o Godot 3.4.4 apenas utilizada o primeiro quadrante do *"tileset"* para todas as paredes e para os espaços "vazios" que deveriam ser preenchidos pelo tile central do conjunto. Não foi achada uma solução que deixasse viável o uso dos *"tilesets"*, então foi criado um tipo de *"tile"* customizado para evitar que esse problema se repetisse. A criação desse *"tile"* foi feita de forma mais simplista em relação ao *"tileset"* tendo em vista as necessidades que deveriam ser atendidas para o funcionamento próprio da geração procedural.

Foi então necessária a implementação do mínimo de detalhes nos tiles novos pois, ao realizar testes com cores sólidas o jogador poderia cair em uma ilusão de ótica de que o personagem não estava saindo do lugar. Essa ilusão só se fortalecia dada a natureza procedural de criação do mapa e a possibilidade de encontros com corredores mais longos do que o normal.

A Figura 8 representa essa mudança de design entre as texturas. A Figura 8a apresenta a solução para o problema de percepção de movimento que se apresentava na situação representada pela figura 8b.

Figura 8 – Comparação das mudanças de textura labirinto procedural



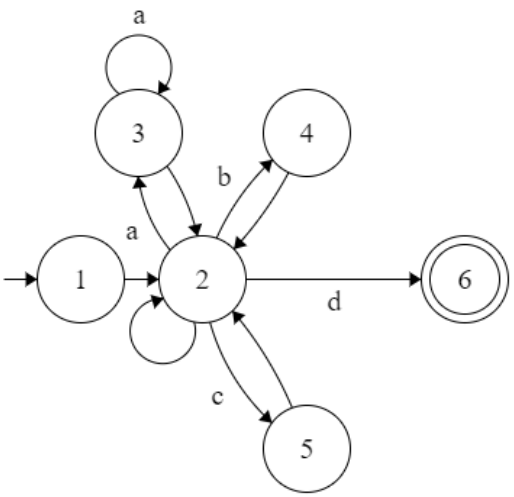
a) **b)**

Fonte: Autoria Própria (2022)

3.4.2 Implementação do personagem principal

Com a implementação dos mapas concluída, começou o processo de implementação do protagonista, suas características e as possíveis interações com o ambiente e com outras entidades. Foram atribuídos atributos ao objeto personagem, que consistem em quantidade de vida, velocidade, *"hitboxes"* para ele e para seus ataques, e mecânicas de movimentação. Com as definições base do personagem implementadas foi então definido seu padrão de comportamento e interação com base nas escolhas possíveis para ele, algo que pode ser melhor representado pela máquina de estado apresentada na Figura 9.

Figura 9 - Máquina de estados



Fonte: Autoria própria (2022)

A máquina de estados apresentada utiliza o seguinte conjunto de regras. Quando há a instanciação do personagem no mapa (1) ele entra obrigatoriamente em um estado de espera,

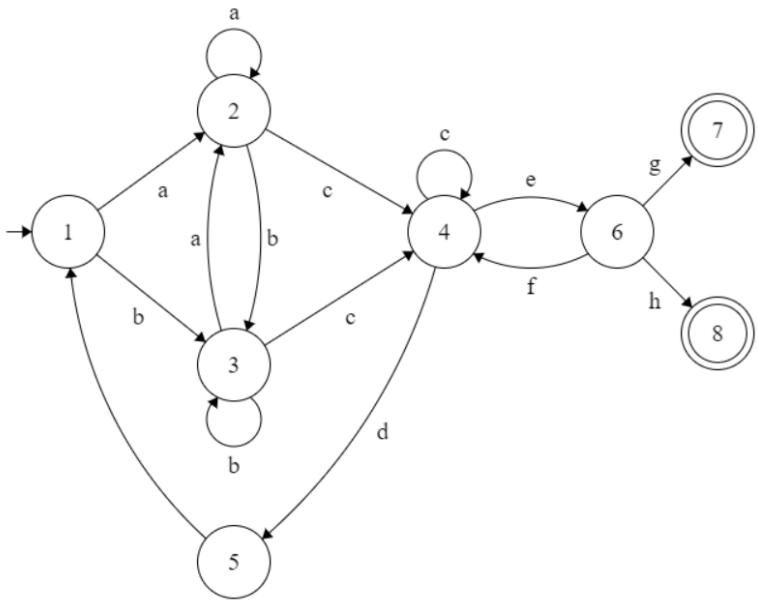
ou *"idle"*, para que sejam executadas ações. As ações possíveis para o personagem consistem em comando de movimentação (a) que faz com que o estado em movimento (3) aconteça e comandos contínuos permitem que ele se mantenha. Rolamento (4) é ativado ao apertar o comando de rolamento (b) e a ação de ataque (5) é ativada na ação de ataque (c). A máquina de estados é dada como desativada quando o personagem morre (6) ou quando o software é terminado. Com a implementação das características e da máquina de estados, o personagem possui capacidade de interagir com o ambiente e com outras entidades.

Durante a implementação do sistema de vidas e o registro de ataques das *"hitboxes"* dos personagens do jogo houve erros que envolviam falha no registro dos *"I-frames"* (*"Invincibility Frames"*) tanto em ataques consecutivos quanto durante a animação de rolamento. Não foi possível chegar em uma solução com relação à falta de *"I-frames"* no rolamento do personagem, porém foi adicionado um novo efeito para tratar disso. Quando os personagens se movem eles possuem um valor de movimento relacionado à fricção que teria com o cenário. Isso aumentaria ou diminuiria o momento (inércia), então foi atribuído ao movimento de desvio (*"dodge"*) uma diminuição desse valor, no caso um aumento no momento relacionado ao movimento. Assim resulta uma maior distância percorrida pela ação. O estado *"dodge"* é representado pelo número 4 da Figura 9.

3.4.3 Implementação dos *"NPCs"*

A implementação dos *"NPC's"* se baseou em um autômato finito (demonstrado na Figura 10) para que ele tivesse uma quantia de autonomia sobre ações e comportamentos pelo mapa e perante outras entidades. A movimentação foi aplicada aos *"NPCs"* de forma a que eles tivessem um ponto de Instanciamento fixo e conseguissem se locomover dentro de rotas aleatórias nas proximidades de sua área inicial. Essa movimentação é referida como comportamento *"idle"* da entidade. Foi implementado também um alcance de reconhecimento para os *"NPCs"* que, ao detectar a presença do jogador dentro dessa área, iniciava o padrão de movimentação de perseguição, onde há a utilização de um *"Pathfinding"* direcionando a entidade ao jogador enquanto ele se encontrasse próximo o suficiente.

Figura 10 - Automato NPC



Fonte: Autoria própria (2022)

As ações apresentadas podem ser definidas da seguinte maneira; Instanciamento/Comportamento Inicial (1), escolha da sua próxima ação pelo *"NPC",* sendo ela *"idle"* (a) ou vagar (b). Essa escolha de ações acontece novamente após ser terminada, podendo entrar em loop ou modificar o estado. A ação atual somente é interrompida quando há a detecção do jogador dentro da sua área de visão (c), assim assumindo a instancia de perseguição (4). Enquanto há presença do jogador dentro da área de detecção do *"NPC"*, ele se mantém em loop nesse comportamento. Caso o personagem escape da área, o agente entra no estado de término de perseguição e retorna ao estado de escolha de Comportamento Inicial (1). Caso haja um confronto (e), há engajamento no modo de combate (6) que pode resultar em fuga do jogador (f), morte do jogador (g), que finaliza o jogo (7) ou morte do inimigo (h) que finaliza o autômato (8).

A ação de perseguição pode resultar em duas situações, o encontro ou a fuga do jogador. No caso de fuga do jogador, há o retorno da entidade para o seu local de instanciamento e se reinicia o processo de comportamento *"idle"*. Caso haja encontro com o jogador o resultado depende das ações realizadas por ele, sendo que caso a *"hitbox"* do inimigo entre em contato com a *"hitbox"* do jogador, o jogador leva x pontos de dano e entra em um estado imune durante pouco tempo, para que seja impedido múltiplos ataques seguidos. Esse conceito de

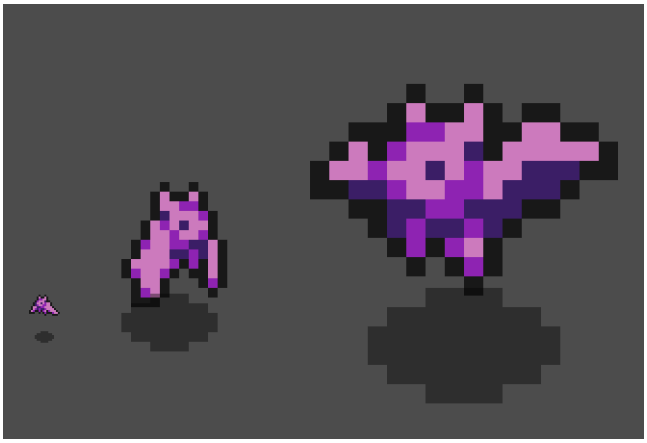
invulnerabilidade temporária é comumente referido como *"I-frames"* e é usado, também, em ações como rolamentos que podem ser realizados pelo personagem.

Caso o jogador reaja e ataque o *"NPC"*, será avaliado se houve contato entre a *"hitbox"* do ataque e a entidade, podendo resultar em um ataque bem-sucedido, assim dando x pontos de dano diretamente na vida do *"NPC"* e o jogando minimamente para trás. Um ataque falho resulta na entidade podendo avançar em direção ao jogador e dar dano no intervalo entre a animação do ataque já executado e no início de um próximo ataque.

É nesse momento do desenvolvimento, no qual se tem ambiente e agentes finalizados, que há a implementação dos dados obtidos na simulação dentro do cenário do jogo. Esses dados são implementados de forma a dar uma visão mais realista para a distribuição de inimigos pelo cenário geral do jogo, assim atribuindo uma camada de realidade para a localização e encontro do jogador com inimigos durante seu tempo explorando a cidade.

Os *"NPCs"* possuem atributos da mesma forma que o jogador, contendo vida, velocidade, dano, hitboxes e a inclusão de uma característica distinta, sendo ela tamanho. O tamanho foi usado para diferenciar e escalar o tipo de *"NPC"* que o jogador estaria enfrentando, deixando visivelmente aparente que o indivíduo apresentaria uma dificuldade mais elevada. Essas entidades com tamanhos variados aparecem nos mapas instanciados e possuem mais vida e mais dano. Nessa situação se houvesse um artista gráfico na equipe de desenvolvimento essas alterações do "sprite" padrão dos *"NPCs"* seria desnecessária pois inimigos diferentes poderiam ser representados com visuais e características próprias e individuais. A diferença de tamanho entre "NPCs" pode ser observada na Figura 11.

Figura 11 - *"NPCs"* com diferentes tamanhos



Fonte: Autoria Própria (2022)

Por conta do tamanho do inimigo *"BOSS"*, caso o jogador se mantivesse em contato com a *"hitbox"* do *"NPC"* após sofrer danos ele ficaria imune aos ataques desse inimigo até que se afastasse da *"hitbox"*. Esse problema se dava por conta de a mecânica de dano somente computar um ataque por contato. A solução adotada consiste na implementação de um efeito de *"knockback"* no jogador, que cria momentum de forma a mover ele na direção contrária do contato por uma distância fixa. Essa solução permite a criação de espaço entre as entidades assim impedindo um *"loop"* de invencibilidade do jogador ao ficar dentro do que seria considerada a zona de *"hitbox"* do *"NPC"*.

4. RESULTADOS OBTIDOS

Os resultados obtidos na inserção da simulação como complemento para o jogo foram satisfatórios, ou seja, foi possível fazer com que a instanciação dos *"NPC's"* hostis pelo mapa seguisse o sistema de pesos atribuídos as estruturas e divisões do mapa. O comportamento dos criminosos em relação ao jogador e a *"NPC's"* neutros também seguiu o que foi projetado.

O jogo ficou responsivo e foi possível implementar todos os comandos e máquinas de estado desejadas, tanto para os *"NPC's"* quanto para o personagem controlado pelo jogador. As mecânicas de criação de mapas, instanciamento de entidades e *"hitboxes"* se mantiveram funcionais em todos os testes e execuções do aplicativo.

Para avaliar de maneira informal o quesito de jogabilidade, o software foi apresentado para amigos e familiares. Os indivíduos que testaram o jogo tiveram abordagens diferentes e apresentaram experiências distintas, geralmente relacionadas ao seu nível de familiaridade com jogos de computador. Observou-se dificuldade relcionada à falta de um tutorial dentro do jogo, porém de maneira geral o resultado foi bom. É importante lembrar que o objetivo do trabalho não era implementar um jogo no sentido completo do termo, o que também envolve arte finalização e outros ítens como balanço de dificuldade que podem alterar muito o resultado final. Em relação aos objetivos, o trabalho alcançou o esperado.

5. CONCLUSÃO

Com o desenvolvimento e conclusão desse trabalho foi possível a observação de como simulações podem afetar a maneira com que o jogo funciona, assim tendo um impacto tanto técnico, afetando a jogabilidade e a distribuição de recursos e entidades pelo mapa, quanto no aspecto social, adicionando valor e peso para as experiências e ambientação do jogador no cenário apresentado.

Comparado com o que havia sido planejado no projeto preliminar houveram mudanças. O desenvolvimento desse trabalho utilizou uma abordagem diferente e mais simples no tópico de simulação, comparada ao que se pretendia inicialmente, e que seria integrar um modelo mais detalhado.

Foram cumpridos os objetivos propostos nesse documento, sendo a de implementação de um protótipo de jogo e de uma simulação de crimes, ambos de forma separada e modular, e foi possível a junção de ambos de forma a apresentar um jogo. O jogo consegue demonstrar aspectos da simulação de uma forma orgânica dentro do cenário apresentado por ele e que possuí elementos interativos coerentes com os resultados obtidos.

Conclui-se que uma simulação não necessita de um jogo para ser compreendida, porém sistemas que provem interação como os jogos facilitam a acessibilidade e no alcance que um estudo pode ter fora do seu meio de pesquisa. Jogos que apresentam cenários mais próximos da realidade não necessitam de uma simulação para serem bons, porém o uso de simulação pode adicionar muito na experiência e facilitar a demonstração de um cenário mais realista.

6. TRABALHOS FUTUROS

Planejamos seguir com o desenvolvimento da simulação nesse mesmo cenário, adicionando mais tipos de *"NPC's"*, no caso mais criminosos, mais tipos de civis e policiais, de forma a deixar o cenário mais realista. Está nos planos, também, seguir com um aprimoramento visual do jogo, a implementação de um tutorial e um *"hub"* para o jogador.

REFERÊNCIAS

BATES, B. ***Game design***. Boston, MA: Thomson Course Technology, 2004.

BETHKE, E. ***Game development, and production***. Plano, Tex.: Wordware Pub, 2003.

BONABEAU, E. *Agent-based modeling: Methods and techniques for simulating human systems.* ***Proceedings of the National Academy of Sciences***, v. 99, n. Supplement 3, p. 7280–7287, 14 maio 2002.

BOURG, D. M.; HUMPHREYS, K.; BYWALEC, B. ***Physics for game developers***. Sebastopol, Calif.: O'Reilly Media, 2013.

BRANTINGHAM, P. L.; BRANTINGHAM, P. J. ***Environmental criminology***. Prospect Heights, Ill.: Waveland Press, 1991.

BRATHWAITE, B.; SCHREIBER, I. ***Challenges for game designers: non-digital exercises for video game designers***. Boston, Ma: Course Technology, 2009.

EDMONDS, B.; HALES, D. *Computational Simulation as Theoretical Experiment.* ***The Journal of Mathematical Sociology***, v. 29, n. 3, p. 209–232, jul. 2005.

COHEN, L. E.; FELSON, M. *Social Change, and Crime Rate Trends: a Routine Activity Approach.* ***American Sociological Review***, v. 44, n. 4, p. 588–608, 1979.

ROTH, E. ***Urban Growth Forecast Using Segmented and Complete Maps with the Sleuth Simulator***. [s.l: s.n.]. Disponível em: <https://repositorio.utfpr.edu.br/jspui/bitstream/1/4173/1/PG_PPGCC_M_Roth%2C%20Ellen%20Cristina%20Wolf_2019.pdf>. Acesso em: 27 abr. 2022.

CROOKS, A. T.; HEPPENSTALL, A. J. *Introduction to Agent-Based Modelling.* ***Agent-Based Models of Geographical Systems***, p. 85–105, 19 out. 2011.

DEVIA, N.; WEBER, R. *Generating crime data using agent-based simulation.* ***Computers, Environment and Urban Systems***, v. 42, p. 26–41, Nov. 2013.

DYTHAM, C.; DEANGELIS, D. L.; GROSS, L. J. *Individual-Based Models and Approaches in Ecology.* ***The Journal of Animal Ecology***, v. 63, n. 2, p. 496, abr. 1994.

EDMONDS, B.; HALES, D. *Computational Simulation as Theoretical Experiment.* ***The Journal of Mathematical Sociology***, v. 29, n. 3, p. 209–232, Jul. 2005.

F1® 2021. Disponível em: <https://www.codemasters.com/game/f1-2021>. Acesso em: 25 abr. 2022.

GAUDENCIO, Alex. **Desenvolvimento de Jogo com Balanceamento de Dificuldade Dinâmico.** Orientador: Prof. Dr. André Koscianski. 2017. Trabalho de Conclusão de Curso (bacharelado em ciência da computação) - Universidade Tecnológica Federal do Paraná, [S. l.], 2017.

GAVIRA, M. D. O. **Simulação computacional como uma ferramenta de aquisição de conhecimento.** Tese de Doutorado, Universidade de São Paulo. 2003.

GONSALVES, Lucas G. **Um Estudo de Aplicação de Inteligência Artificial em Jogos.** Orientador: Prof. Dr. André Koscianski. 2018. Trabalho de Conclusão de Curso (tecnologia em análise e desenvolvimento de sistemas) - Universidade Tecnológica Federal do Paraná, [S. l.], 2018.

GROFF, E. R. *Simulation for Theory Testing and Experimentation: An Example Using Routine Activity Theory and Street Robbery*. ***Journal of Quantitative Criminology***, v. 23, n. 2, p. 75–103, 22 fev. 2007.

GUSTAFSSON, L.; STERNAD, M. *Consistent micro, macro, and state-based population modeling*. ***Mathematical Biosciences***, v. 225, n. 2, p. 94–107, Jun. 2010.

HATNA, E.; BENENSON, I. *The Schelling Model of Ethnic Residential Dynamics: Beyond the Integrated - Segregated Dichotomy of Patterns*. ***Journal of Artificial Societies and Social Simulation***, v. 15, n. 1, 2012.

HUGHES, H. P. N. et al. *Agent-based modelling and simulation: The potential contribution to organizational psychology*. ***Journal of Occupational and Organizational Psychology***, v. 85, n. 3, p. 487–502, 5 Mar. 2012.

JANSSEN, M. ***Agent-Based Modelling Entry prepared for the Internet Encyclopaedia of Ecological Economics***. [s.l: s.n.]. Disponível em: <https://www.isecoeco.org/pdf/agent_based%20_modeling.pdf>. Acesso em: 25 abr. 2022.

KENT, S. L. ***The ultimate history of video games***. New York: Random House International; London, 2002.

LIMA, F. W. S.; HADZIBEGANOVIC, T.; STAUFFER, D. *Evolution of ethnocentrism on undirected and directed Barabási–Albert networks*. ***Physica A: Statistical Mechanics and its Applications***, v. 388, n. 24, p. 4999–5004, Dez. 2009.

LINIETSKY, J; MANZUR, A. **Godot Engine - License**. Disponível em: <https://godotengine.org/license>.

LIU, H.; BROWN, D. E. *Criminal incident prediction using a point-pattern-based density model*. ***International Journal of Forecasting***, v. 19, n. 4, p. 603–622, out. 2003.

Lample. Disponível em: <https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14456/0>. Acesso em: 25 abr. 2022.

MANSSOUR, Isabel H; COHEN, Marcelo. **Introdução à computação gráfica**. RITA, v. 13, n. 2, p. 43-68, 2006.

MCHANEY, R. ***Understanding computer simulation***. [s.l.] Ventus Publishing, 2009.

MCCOWN**,** F. ***Schelling's Model of Segregation***. Disponível em: <http://nifty.stanford.edu/2014/mccown-schelling-model-segregation>. Acesso em: 25 abr. 2022.

MCGUIRE, M.; ODEST CHADWICKE JENKINS. ***Creating games : mechanics, content, and technology***. Wellesley, Mass.: Ak Peters, 2009.

MAXFIELD, M. G. *Lifestyle and routine activity theories of crime: Empirical studies of victimization, delinquency, and offender decision-making.* ***Journal of Quantitative Criminology***, v. 3, n. 4, p. 275–282, dez. 1987.

MEIER, R. F.; KENNEDY, L. W.; SACCO, V. F. ***The Process and Structure of Crime: Criminal Events and Crime Analysis***. [s.l.] Transaction Publishers, [s.d.].

PICOLI, Ivan L. **Arquitetura Cliente-Servidor em Jogos Multijogador.** Orientador: Prof. Dr. André Koscianski. 2011. Trabalho de Conclusão de Curso (tecnologia em análise e desenvolvimento de sistemas) - Universidade Tecnológica Federal do Paraná, [S. l.], 2011.

POLITOWSKI, C.; PETRILLO, F.; GUÉHÉNEUC, Y.-G. *A Survey of Video Game Testing*. **arXiv:2103.06431 [cs]**, 10 mar. 2021.

ROGERS, T.; MCKANE, A. J. *A unified framework for Schelling's model of segregation.* ***Journal of Statistical Mechanics: Theory and Experiment***, v. 2011, n. 07, p. P07006, 12 jul. 2011.

ROSS, D. ***Game Theory (Stanford Encyclopedia of Philosophy)***. Disponível em: <https://plato.stanford.edu/entries/game-theory/>.

SCHELL, J. ***The Art of Game Design: A book of lenses***. Massachussets, USA. Elsevier. 2008.

SHORT, M. B. et al. *A STATISTICAL MODEL OF CRIMINAL BEHAVIOR.* ***Mathematical Models and Methods in Applied Sciences***, v. 18, n. supp01, p. 1249–1267, ago. 2008.

SITUNGKIR, H. *Epidemiology Through Cellular Automata: Case of Study Avian Influenza in Indonesia*. **arXiv:nlin/0403035**, 17 mar. 2004.

TITA, G. E.; COHEN, J.; ENGBERG, J. *An Ecological Study of the Location of Gang "Set Space".* ***Social Problems,*** v. 52, n. 2, p. 272–299, maio 2005.

TITA, G.; RIDGEWAY, G. *The Impact of Gang Formation on Local Patterns of Crime.* ***Journal of Research in Crime and Delinquency***, v. 44, n. 2, p. 208–237, maio 2007.

WANG, F. ***Geographic information systems and crime analysis***. Hershey, Pa: Idea Group Pub, 2005.

WILSON, J.; KELLING, G. ***Broken Windows The police and neighborhood safety***. [s.l: s.n.]. Disponível em: <http://personal.psu.edu/exs44/597b-Comm&Crime/Wilson-Kelling-Broken%20Windows.pdf>.

ZAWALSKI, Gabriel L. **Implementação de Controle Automático de Dificuldade em um Jogo.** Orientador: Prof. Dr. André Koscianski. 2019. Trabalho de Conclusão de Curso (bacharelado em ciência da computação) - Universidade Tecnológica Federal do Paraná, [S. l.], 2019.