Desenvolvimento de um editor de Lógica *Fuzzy* para o motor de jogo Unity

Francisco Diego Moreira Feitosa, Alysson Diniz dos Santos
Instituto UFC Virtual - Sistemas e Mídias Digitais
Universidade Federal do Ceará
Fortaleza, Brasil
chicodiegomoreira@gmail.com, alysson@virtual.ufc.br

***Resumo*—O desenvolvimento de jogos digitais é uma tarefa que envolve profissionais de diversas áreas e com diferentes competências. A Inteligência Artificial é um aspecto desafiador do desenvolvimento de jogos, pois pode envolver uma razoável quantidade de implementação de código e deve fornecer a ilusão de comportamento inteligente para os agentes do jogo. Ou seja, a Inteligência Artificial situa-se entre o design e a programação (áreas que nem sempre se comunicam da maneira mais fluida possível). Neste contexto, este trabalho detalha a concepção e implementação do *Fuzzy Logic Editor*, uma ferramenta para o auxílio da utilização da Lógica *Fuzzy* em jogos. Esta ferramenta possibilita ao projetista de IA estruturar as variáveis e regras sem a necessidade de codificação e é, até onde sabemos, a primeira ferramenta deste tipo totalmente grátis disponível para o motor de jogos Unity. A avaliação preliminar da usabilidade da ferramenta indica o sucesso em sua aplicação no desenvolvimento de um protótipo de jogo.**

***Keywords*-Jogos; Desenvolvimento; Lógica *Fuzzy*; Unity;**

I. Introdução

O desenvolvimento de jogos digitais (de agora em diante, jogos) é uma tarefa complicada que envolve profissionais de diferentes áreas, tais quais: programadores, roteiristas, animadores, designers de jogos (*game designers*), de interface, de som, dentre outros [1]. De forma geral, os jogos proporcionam experiências capazes de engajar os usuários em ambientes imersivos e interativos. Neste contexto, a Inteligência Artificial (IA) cumpre um papel fundamental nos jogos [2][3], sendo utilizada para: criar a ilusão de comportamento racional em personagens não jogáveis (NPCs), adaptar a dinâmica da experiência de jogo em tempo real, automatizar os ciclos de teste do projeto, dentre outras aplicações [11].

Frequentemente, ao implementar uma IA para jogos, ocorrem situações nas quais o desenvolvedor precisa modelar eventos cotidianos imprecisos. Por exemplo, se um NPC possui o nível de energia baixo, ele pode decidir fugir ao invés de atacar o jogador. Para definir se determinado nível de energia é baixo, pode-se utilizar da Lógica Booleana (ou Lógica Clássica) para estruturar sentenças que são classificadas em dois valores absolutos: "verdadeiro" ou "falso". Desta forma, no exemplo acima, a afirmação "O nível de energia do NPC é baixo" poderia ser considerada verdadeira se o nível fosse inferior a uma quantidade pré-determinada, ou falsa caso contrário.

Um problema surge na definição dos valores pré-determinados que definem os conjuntos. O que quer dizer energia baixa? E quanta variação seria necessária para dizer que a energia passa a ser alta? Uma solução parcial seria a utilização de vários intervalos, adicionando granularidade aos conjuntos (o nível de energia poderia, por exemplo, ser muito baixo, baixo, médio, alto ou muito alto). No entanto, mesmo neste caso, na vizinhança dos valores limítrofes uma variação de apenas uma unidade faria com que a energia fosse classificada como média (em um caso) ou baixa (no outro). No mundo real, a percepção humana é subjetiva e pequenas diferenças como essas podem até mesmo não ser percebidas.

Uma alternativa para a representação de informações imprecisas é a teoria da Lógica *Fuzzy* (proposta por Lofti A. Zadeh). Através desta lógica, sistemas de controle tratam informações imprecisas convertendo-as em valores numéricos [5]. Tais sistemas podem simular tomadas de decisão menos propensas à previsibilidade, possibilitando a elaboração de experiências interessantes para o jogador.

Este trabalho detalha a concepção, implementação e avaliação preliminar do *Fuzzy Logic Editor*, uma ferramenta para o auxílio da utilização da Lógica *Fuzzy* em jogos. Esta ferramenta possibilita ao projetista de IA estruturar as regras sem a necessidade de codificação, o que é particularmente interessante em projetos de jogos, onde (como apresentado anteriormente) profissionais de diversas áreas interagem.

Fuzzy Logic Editor é, até onde sabemos, a primeira ferramenta completamente gratuita para a criação de sistemas de Lógica *Fuzzy* no motor de jogo Unity. Desta forma, espera-se que a principal contribuição deste trabalho seja a produção de uma ferramenta que pode impactar positivamente principalmente equipes de desenvolvimento de jogos pequenas e com restrições orçamentárias.

II. Lógica *Fuzzy*

Como dito na Introdução, a Lógica *Fuzzy* [5] permite a representação de informações incertas (motivo pelo qual é também conhecida como Lógica Nebulosa ou Difusa). Enquanto na Lógica Clássica o pertencimento de um elemento a um conjunto só pode assumir dois valores (0 - ausência, e 1 - presença), na Lógica *Fuzzy* este pertencimento pode assumir um valor qualquer no intervalo [0, 1].

Em outras palavras, a Lógica *Fuzzy* permite a um elemento pertencer parcialmente a um determinado conjunto. Este pertencimento parcial de um elemento pode-se estender a múltiplos conjuntos, gerando o que denominamos de *variáveis linguísticas*. A Figura 1 representa graficamente a variável linguística "fome". Neste exemplo, a fome é composta por três conjuntos difusos (*fuzzy sets* [5]) que representam os possíveis estados que a variável pode assumir. Ou seja, a fome pode ser considerada baixa, média, ou alta.

Para compreender a Figura 1 é importante observar que o eixo x representa os possíveis valores a serem atribuídos à variável fome no sistema em questão (de 0 a 100), enquanto o eixo y representa a pertinência (de 0 à 1) de cada valor de x à cada um dos *fuzzy sets* que compõem a variável fome. Por exemplo, como está destacado na figura, um valor de 75 para fome, possui grau de pertinência 0 para baixa, 0.25 para media e 0.75 para alta. Desta forma, estabelece-se a principal diferença dos sistemas de Lógica *Fuzzy* com relação aos de Lógica Clássica: naqueles, os elementos podem, de maneira simultânea, pertencer parcialmente à diversos conjuntos.

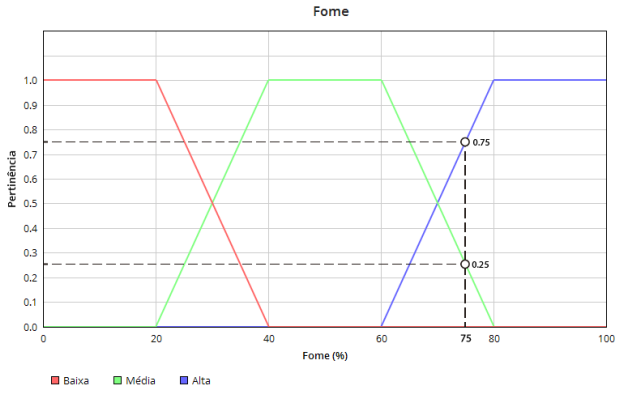

Figura 1. Variável linguística Fome, definida por três *fuzzy sets*: baixa (em vermelho), média (em verde), alta (em azul). Destaca-se que, por exemplo, um valor de entrada de 75 possui grau de pertinência 0 para baixa, 0.25 para media e 0.75 para alta.

É também importante observar que a soma das pertinências de 75 para cada um dos *fuzzy sets* é igual a 1 (0 + 0.25 + 0.75). Apesar de não ser obrigatório, é recomendável que os conjuntos sejam construídos em simetria, de forma a garantir esta propriedade [4]. Com esta conformação, é possível interpretar o valor de 75 em fome como sendo 0% pertencente à baixa, 25% pertencente à media e 75% pertencente à alta.

Matematicamente, os *fuzzy sets*, caracterizam-se por uma *função de pertinência* que mapeia o elemento observado para um grau de pertinência. Por exemplo, para conjuntos triangulares é utilizada a função de pertinência descrita na Figura 2.

Através da combinação de múltiplas variáveis linguísticas, pode-se formular regras em uma linguagem muito próxima

$$f\left(x;a,b,c\right) = max\left(min\left(\frac{x-a}{b-a},\frac{c-x}{c-b}\right),0\right)$$

Figura 2. Função de pertinência para *fuzzy set* triangular. Outros formatos são possíveis, como trapezoidais, triangulares, gaussianas, dentre outros.

da natural. Por exemplo, conjecturemos que estejam definidas outras duas variáveis para um agente inteligente de um jogo fictício: vida (baixa, media ou alta) e ir_caçar (pouco importante, importante ou muito importante). Desta forma, as seguintes regras podem ser estabelecidas:

- IF fome is baixa THEN ir_caçar IS pouco importante
- IF fome is media AND vida is baixa THEN ir_caçar IS importante
- IF fome is media AND vida is media THEN ir_caçar IS muito importante
- IF fome is media AND vida is alta THEN ir_caçar IS muito importante
- IF fome is alta THEN ir_caçar IS muito importante

Tais regras, denominadas *fuzzy*, devem ser sempre descritas com uma condição (IF) e uma consequência (THEN). A condição introduz as variáveis linguísticas de entrada (denominadas antecedentes, no exemplo: fome e vida) e a consequência apresenta as de saída (denominadas consequentes, no exemplo: ir_caçar) [8][10]. Como pode ser observado no conjunto de regras definido, os antecedentes podem ser combinados com expressões equivalentes às da Lógica Clássica, como: OR, AND e NOT (apesar de não constar no exemplo, o mesmo vale para os consequentes).

Uma vez definidas as regras, é possível definir um modelo de inferência (tomada de decisões) para sistemas especialistas (i.e., aqueles que modelam um conhecimento empírico). Este modelo (ver Figura 3) é composto basicamente por: um Fuzificador, um Motor de Inferência, e um Defuzificador, além das Regras que constituem a base de conhecimento do especialista [4][10]. O Fuzificador traduz os valores de entrada (dados precisos) para os *fuzzy sets* (dados difusos) através da aplicação da função de pertinência adequada. O Motor de Inferência determina os *fuzzy sets* de saída, ao processar as variáveis linguísticas junto às regras definidas na base de conhecimento. Finalmente, o Defuzificador traduz os dados difusos inferidos em valores precisos e os retornam [4][10].

A. *Lógica Fuzzy e a Programação de Jogos*

Desde que foi introduzida no desenvolvimento de jogos em 1996 na Game Developer Magazine por O'Brien [14][15], a Lógica *Fuzzy* tornou-se uma das principais técnicas no desenvolvimento de IA para jogos [2][3][4][12][13]. Muitas pesquisas ainda são realizadas sobre o uso desta técnica em jogos [11][16], desde a revisitação de jogos antigos [17][18] ao uso da Lógica *Fuzzy* como mecânica principal [14].

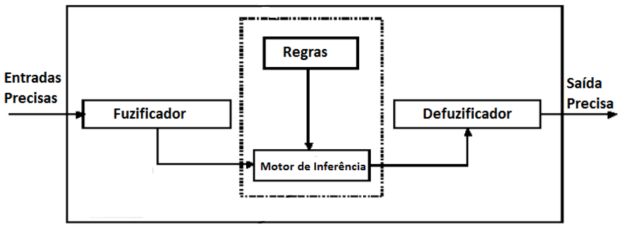

Figura 3. Modelo de Inferência *Fuzzy* - SANTOS, 2010, p. 66.

A natureza linguística da Lógica *Fuzzy* permite a formulação das regras por um especialista, papel que no desenvolvimento de jogos é regularmente atribuído a um *game designer* [11][14]. No entanto, se não tratada com uma ferramenta específica, a definição das regras fica intrincada no código, o que pode dificultar o acesso e a manutenção do sistema por profissionais sem conhecimentos de programação. Desta forma, para a adequada utilização de Lógica *Fuzzy* em um projeto de jogo digital, faz-se necessário o desenvolvimento de uma ferramenta específica, capaz de fornecer as principais funcionalidades desta técnica, abstraindo os cálculos matemáticos para o *game designer*. Com tal ferramenta, seria possível a criação de variáveis e regras lógicas sem a necessidade de alterações no código fonte da aplicação.

III. Metodologia

A abordagem metodológica utilizada neste trabalho contém três etapas principais: Ideação, Desenvolvimento e Avaliação. Tais etapas simplificam os passos comumente seguidos em outras metodologias no processo de desenvolvimento de softwares [19][20], para se adequar ao escopo do projeto.

A. Ideação

A fase de Ideação compreendeu uma pesquisa de soluções existentes e ferramentas similares, de forma a estruturar um arcabouço técnico sobre o qual a ferramenta foi idealizada. Nesta etapa realizaram-se *brainstorms* para identificar as principais características que a ferramente deve conter e foram produzidos esboços de interface.

B. Desenvolvimento

Etapa dividida em duas partes: projeto e implementação. Na primeira, as funcionalidades identificadas na Ideação foram estruturadas no projeto da ferramenta. O projeto técnico utilizou diagrama de classes (UML) para estruturar os elementos necessários. A segunda etapa do desenvolvimento foi a implementação. Ocorreu em ciclos iterativos, com a produção de protótipos semanais incrementais.

C. Avaliação

A ferramenta foi submetida a testes de usabilidade com usuários para avaliar a sua funcionalidade de criação de regras. O perfil dos participantes foram desenvolvedores/estudantes da área de jogos (programadores, *game designers*, projetistas de IA, etc) que detinham conhecimento prévio de Lógica *Fuzzy*, tendo em vista que o foco do teste não é a aprendizagem deste conceito, e sim a utilização da ferramenta. Os participantes foram apresentados à ferramenta, onde realizaram tarefas previamente definidas. Foram coletados dados quantitativos e qualitativos, obtidos por meio da observação, e coleta da opinião de cada participante acerca da ferramenta. Após a realização dos testes, os dados coletados foram analisados para averiguar se a ferramenta atendeu aos requisitos para o qual foi desenvolvida.

IV. Ideação

A ideação partiu das necessidades identificadas na Seção II: tornar acessíveis aos *game designers* as principais funcionalidades da Lógica *Fuzzy*, abstraindo os cálculos matemáticos e os aspectos específicos do desenvolvimento de software.

Inicialmente decidiu-se que seria necessário partir de um motor de jogo (*engine*). Uma *engine* provê funcionalidades basais do desenvolvimento de jogos [26] e permite que este trabalho seja focado apenas no conteúdo relacionado a Lógica *Fuzzy*. Neste contexto, escolhemos a Unity [6], uma das *engines* mais utilizadas atualmente, principalmente no contexto de jogos independentes [7]. A Unity conta com diversos recursos para a criação de jogos e sua licença gratuita permite o uso de todas as funcionalidades para a produção de jogos comerciais. Vale ressaltar também que ela permite a implementação de novos recursos ao seu editor, com vasta documentação e uma plataforma (Unity Asset Store) para a publicação de conteúdos produzidos pela comunidade desenvolvedora, os quais podem ser vendidos ou distribuídos gratuitamente.

Uma pesquisa realizada em maio de 2019 pelos termos "*Fuzzy Logic*" na Unity Asset Store, resultou em apenas duas entradas (um editor de preferências do jogador e um comparador de números de ponto flutuante) que não possibilitam a criação de sistemas de Lógica *Fuzzy*. Ampliando a pesquisa para o termo mais abrangente "*Artificial intelligence*", os 50 primeiros resultados mais relevantes também não apresentaram menção a Lógica *Fuzzy*. Então, considerada a importância desta técnica para o desenvolvimento de jogos, a popularidade da Unity, e a (aparente) ausência de ferramentas associadas, o desenvolvimento de uma extensão para facilitar a utilização de Lógica *Fuzzy* na Unity evidenciou-se bastante relevante.

Após a observação de soluções semelhantes para criação de sistemas *fuzzy* (como JFuzzyLogic [25]), em sessões de *brainstorm* foi definido que a extensão deveria: (i) permitir a criação de variáveis linguísticas (com as definições de

seus respectivos *fuzzy sets*), (ii) possibilitar a definição de regras que manipulem estas variáveis linguísticas e, (iii) prover uma interface gráfica para controlar este sistema. A extensão deveria ainda fornecer acesso direto ao código fonte do sistema, de forma a permitir a adaptação do mesmo às necessidades de cada usuário.

V. Desenvolvimento

A ferramenta *Fuzzy Logic Editor* é uma extensão da *engine* Unity que permite a definição de variáveis linguísticas e das regras que controlam o sistema de inferência *fuzzy* em uma interface visual. A ferramenta foi desenvolvida com a Unity 2019.1.0f2 e encontra-se disponível para download em https://github.com/chicodiegomoreira/fuzzylogiceditor.

Os cálculos referentes ao funcionamento da Lógica *Fuzzy* são adaptados de uma biblioteca contida no AForge.NET Framework [21], um *framework* C♯ de código aberto projetado para desenvolvedores de Inteligência Artificial e Visão Computacional.

De forma mais específica, as funcionalidades providas pelo *Fuzzy Logic Editor* podem ser classificadas em dois tipos: Definição e Codificação.

A. Funcionalidades de Definição

Relacionadas ao editor visual desenvolvido, as funcionalidades de definição são aquelas que devem ser acessadas pelo *game designer* (ou pelo projetista de IA) e permitem a definição de um sistema de inferência *fuzzy* de forma completamente independente da codificação em uma linguagem de programação. Tais funcionalidades possibilitam a estruturação dos elementos necessários para um sistema *fuzzy*: variáveis linguísticas e regras.

1) Fuzzy Logic Editor - aba de variáveis linguísticas: Como é possível observar na Figura 4, nesta aba é possível criar, excluir ou editar variáveis linguísticas. O usuário nomeia as variáveis e define o espaço de valores numéricos que ela pode assumir, bem como os *fuzzy sets* que a definem. Em sua versão atual, o *Fuzzy Logic Editor* trata os dois tipos mais comuns de *fuzzy sets*: os triangulares e os trapezoidais. Apesar de simples, estes são os conjuntos mais popularmente utilizados, e que atendem às necessidades dos sistemas de inferência de complexidade baixa ou regular [4]. Futuramente, outros modelos de conjuntos (como gaussianas, por exemplo) podem ser inseridos sem alterações significativas na estrutura da ferramenta.

2) Fuzzy Logic Editor - aba de regras fuzzy: Como é possível observar na Figura 5, nesta aba é possível inserir, excluir ou editar regras *fuzzy*. As regras manipulam as variáveis previamente definidas na aba de variáveis linguísticas, e permitem a utilização de operadores lógicos (OR, AND e NOT) e de bloco (parêntesis) para a definição das regras. Há ainda um campo de texto não editável (*Rule String*) que exibe o resultado da regra atual na forma escrita. Em sua versão atual, o *Fuzzy Logic Editor* disponibiliza apenas os conectores lógicos de Mamdani, que são os de implementação mais simples, mas que atendem às necessidades dos sistemas de inferência de pouca complexidade [9].

B. Funcionalidades de Codificação

Além da definição do sistema *fuzzy* via interface gráfica, o *Fuzzy Logic Editor* também permite que o sistema de inferência seja acessado via código. Desta forma, a ferramenta permite o acesso de baixo nível ao sistema, permitindo controle total ao programador para utilização e extensão da ferramenta.

As funcionalidades de codificação estão disponíveis através de uma classe estática denominada FIS, abreviação de Fuzzy Inference System. Esta classe permite:

- inicializar o sistema *fuzzy*: o que pode ser feito automaticamente no início da aplicação, ou em determinado ponto que o programador desejar;
- inserir entradas: com a função SetInput, que faz o papel do Fuzificador (ver Figura 3), atribuindo valores às variáveis linguísticas, e;
- ler o valor de uma variável: através da função Evaluate, que faz o papel do Motor de Inferência e do Defuzificador (ver Figura 3), retornando o resultado da inferência com base nas regras definidas.

Além das funcionalidades de codificação, é importante lembrar que o *Fuzzy Logic Editor* é uma solução de código aberto. Em uma utilização típica do sistema, caso o projetista de IA identifique alguma funcionalidade que é desejada e não está disponível na interface gráfica do *Fuzzy Logic Editor*, ele simplesmente requisita que o programador implemente esta funcionalidade. Portanto, é permitido (e encorajado) que o usuário estenda as funcionalidades disponíveis na classe FIS ou em quaisquer outras partes do sistema.

VI. Avaliação

O objetivo do *Fuzzy Logic Editor* é auxiliar a utilização da Lógica *Fuzzy* em jogos. Para tanto, é importante que projetista de IA consiga compreender e utilizar a interface da extensão para que possa estruturar as variáveis linguísticas e regras sem a necessidade de codificação. Para investigar preliminarmente se a ferramenta atende essas necessidades, decidiu-se pela realização de um teste de usabilidade.

Usabilidade pode ser definida como a extensão em que um produto pode ser utilizado por usuários para atingir metas específicas com eficácia, eficiência e satisfação em um determinado contexto de uso [23]. Portanto, os testes de usabilidade possuem o objetivo de verificar a usabilidade do produto em mãos [24].

O teste de usabilidade deste experimento utilizou um protótipo de jogo onde existem dois tipos de agentes inteligentes: presas (várias) e (um) caçador. Utilizando o *Fuzzy Logic Editor*, foi requisitado que os testadores realizassem todas as atividades possíveis relacionadas à definição de

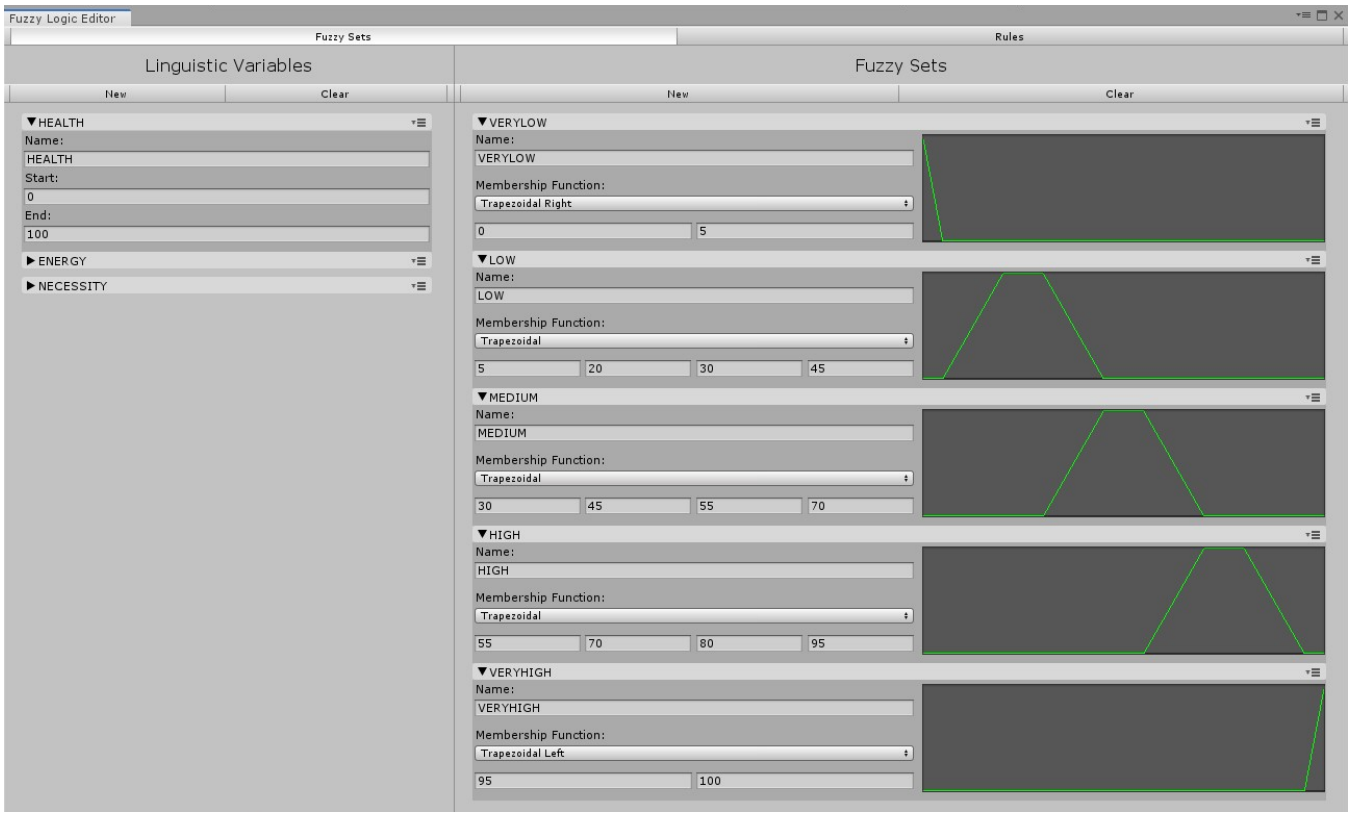

Figura 4. *Fuzzy Logic Editor* - aba de variáveis linguísticas. No lado esquerdo é possível inserir, excluir ou selecionar variáveis linguísticas. No lado direito é possível editar a variável selecionada. Na imagem são apresentados os 5 *fuzzy sets* que compõem a variável Health: VeryLow, Low, Medium, High e VeryHigh.

regras *fuzzy* que modelassem o comportamento do Caçador. Nos subtópicos seguintes, detalha-se o perfil dos participantes, o protótipo desenvolvido e os instrumentos utilizados na coleta de dados.

A. *Participantes*

Para a avaliação foi definido um perfil específico de usuários: desenvolvedores/estudantes da área de jogos (programadores, *game designers*, projetistas de IA, etc) que já possuam conhecimento básico do motor de jogos Unity e da Lógica *Fuzzy*. A escolha por este público-alvo particular pretende minimizar possíveis erros causados por incompreensão da teoria das variáveis linguísticas e das regras *fuzzy*, ou por inexperiência com a Unity. Por esta razão, a coleta foi realizada com um número limitado de usuários (7 usuários e 1 teste piloto que foi descartado) porém, apesar disso, espera-se conseguir dados mais acurados acerca da usabilidade da ferramenta.

Todos os participantes são do sexo masculino, residentes em Fortaleza, e possuem entre 20 e 28 anos. Entendemos que este perfil pode inserir um viés nos dados coletados, porém foi o possível para o tempo disponível para a realização deste trabalho.

B. *PreyHunt: O protótipo*

Para a avaliação foi desenvolvido um protótipo de mecânica denominado PreyHunt (ver Figura 6). PreyHunt possui dois tipos de agentes autônomos: as Presas e o Caçador.

As Presas aparecem no mundo periodicamente, e possuem uma propriedade numérica denominada saúde, que diminui ao longo do tempo. Cada Presa se comporta de acordo com o seu nível de saúde e proximidade com o Caçador, podendo vaguear, fugir ou descansar (para recuperar a saúde).

O Caçador, por sua vez, recupera saúde apenas quando consegue capturar uma presa. Ele possui ainda uma outra propriedade numérica denominada energia, que diminui quando o Caçador não está descansando. Ou seja, o comportamento do Caçador deve variar entre caçar (para recuperar saúde) e descansar (para recuperar energia).

O Caçador é um agente que tem seu comportamento definido por um sistema *fuzzy*. Suas duas propriedades (saúde e energia) são as variáveis linguísticas de entrada no sistema. A variável de saída, denominada necessidade, assume os valores dos possíveis estados do Caçador: assustar (procurar por Presas descansando, e se aproximar delas para obrigá-las a fugir), caçar (alimentar-se de uma presa que está muito

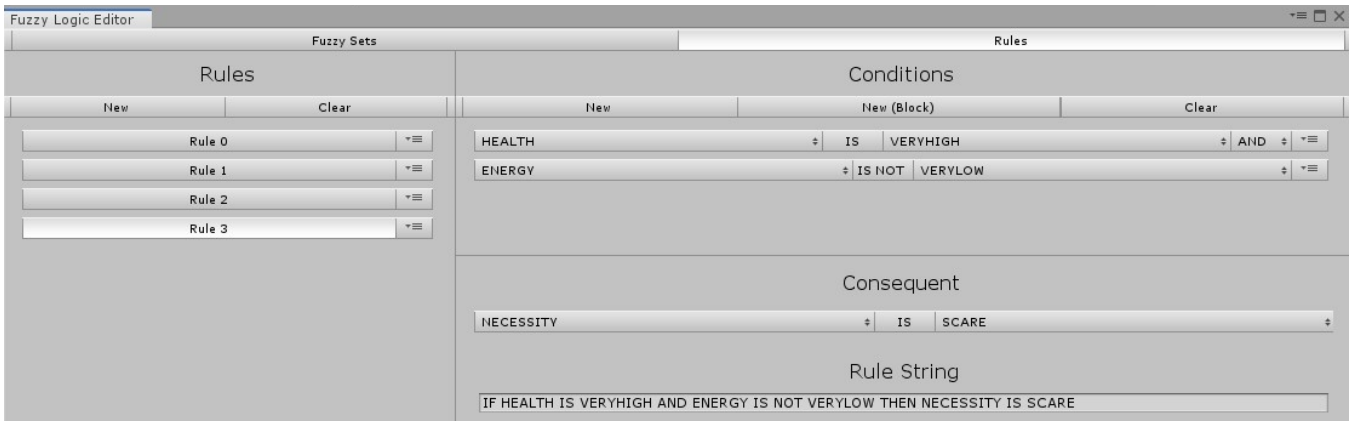

Figura 5. *Fuzzy Logic Editor* - aba de regras *fuzzy*. No lado esquerdo é possível inserir, excluir ou selecionar regras. No lado direito é possível editar a regra selecionada. Na imagem a regra 3 está definida como "IF Health is VeryHigh AND Energy is NOT VeryLow THEN Necessity is Scare".

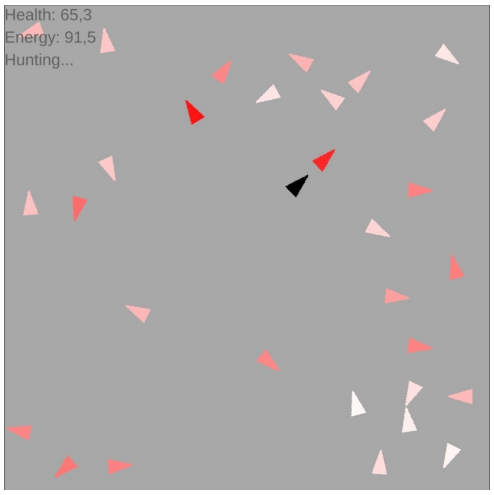

Figura 6. PreyHunt - Protótipo desenvolvido. O triângulo preto indica o caçador, enquanto todos os outros são presas. As diferentes tonalidades das presas indicam seu nível de saúde. Quanto mais perto do vermelho, menos saúde restante. No canto esquerdo superior exibe-se a saúde, a energia e o estado atual do caçador.

cansada para fugir, o que recupera a saúde do Caçador), ou descansar (o que recupera a energia do Caçador).

Em resumo, o sistema *fuzzy* em questão possui as seguintes variáveis:

- saúde (de entrada): decai com o tempo, e cresce quando uma Presa é caçada com sucesso;
- energia (de entrada): decai com o tempo, e cresce quando o Caçador descansa; e
- necessidade (de saída): indica a ação do Caçador: assustar as presas, caçá-las ou descansar.

No experimento conduzido neste trabalho, foi requisitado que os testadores, dadas as variáveis linguísticas já definidas, definissem as regras *fuzzy* que modelam o comportamento do Caçador. A escolha por avaliar apenas a aba de regras *fuzzy* do *Fuzzy Logic Editor* foi devido às restrições de tempo disponível para a realização deste trabalho. Futuramente pretende-se testar também a usabilidade aba de variáveis linguísticas.

C. Instrumentos

Os dados acerca da interação foram coletados através de dois instrumentos: formulário de tarefas e o formulário de avaliação da ferramenta.

O formulário de coleta foi utilizado pelo moderador do teste para armazenar os dados acerca da interação dos testadores. O moderador utilizou um dispositivo móvel (*smartphone*) com acesso a um formulário online, onde constavam: 16 tarefas, um campo para assinalar se a tarefa foi realizada com sucesso ou não, e um campo aberto para eventuais anotações. As 16 tarefas definidas são todas as interações possíveis de serem realizadas com a aba de criação de regras do *Fuzzy Logic Editor*:

- T1: Adicionar um nova regra
- T2: Adicionar uma nova condição para a regra
- T3: Alterar a variável da condição
- T4: Alterar o valor da variável da condição
- T5: Negar a condição
- T6: Duplicar a condição
- T7: Alterar o operador lógico da condição
- T8: Adicionar um Bloco de Condições
- T9: Adicionar condições ao Bloco e alterá-las a vontade
- T10: Alterar o valor da variável da consequência
- T11: Negar a variável da consequência
- T12: Duplicar a regra
- T13: Deletar todas as condições
- T14: Definir a seguinte regra: IF health is NOT verylow AND energy is high OR energy IS veryhigh THEN necessity is hunt
- T15: Deletar todas as regras
- T16: Definir regras para o protótipo PreyHunt

O segundo instrumento de coleta foi o formulário de avaliação da ferramenta. O mesmo foi respondido pelo par-

ticipante ao final do teste. Este simula a sessão de avaliação da Unity Asset Store, dando uma nota de 0 a 5 relacionada a sua satisfação com a ferramenta, seguido por um comentário que era facultativo ao usuário.

D. Procedimento

Todos os testes foram realizados em um mesmo laboratório na Universidade Federal do Ceará, durante o mês de Outubro de 2019 e contando com a presença de um moderador. Tal metodologia, apesar de retirar o contexto real da utilização da ferramenta, possibilitou que fossem mitigados eventuais problemas relacionados à versões diferentes da Unity, ou à não compreensão das tarefas.

Os materiais necessários foram um computador utilizado pelo testador para manipular o *Fuzzy Logic Editor*, um *smartphone* utilizado pelo moderador para preencher o instrumento de coleta de dados, e uma folha de papel e uma caneta que foram disponibilizados para o usuário anotar alguma observação, caso julgasse necessário.

Cada participante foi testado individualmente e assinou um termo de consentimento, declarando ciência dos objetivos da avaliação, e da garantia do sigilo de suas informações pessoais. O experimento foi estruturado em quatro etapas principais: apresentação do cenário de teste, adaptação à ferramenta, definição de regras e crítica pessoal.

Na primeira etapa, o participante era apresentado ao documento "Descriçao do Protótipo" (que pode ser acessado em http://bit.ly/preyhuntdescription). Este documento contém a descrição do cenário do teste: "Você é o responsável por implementar a inteligência artificial de um novo projeto de jogo da empresa *indie* que você participa com outros dois colegas. Você resolveu testar a ferramenta *Fuzzy Logic Editor* para, possivelmente, adicionar Lógica Fuzzy ao seu projeto. Para testar a ferramenta você criou o protótipo PreyHunt que está em execução na sua frente". O documento ainda contém a explicação acerca das variáveis já definidas na aba de variáveis linguísticas e do comportamento delas no protótipo PreyHunt.

Em seguida, durante a etapa de adaptação à ferramenta, o usuário era solicitado a executar as quinze tarefas iniciais do formulário de tarefas (de T1 à T15). Tais tarefas são relacionadas à interface de criação de regras. Durante todo o tempo o participante era observado pelo moderador do teste que, neste momento, cumpria o papel de um observador silencioso. O moderador anotava se cada tarefa tinha sido concluída com sucesso ou falha, anotava algum comentário ou ação inesperada do participante, e só interagia com o participante se diretamente solicitado pelo mesmo.

Na etapa de definição de regras, era requisitado que o participante realizasse a tarefa T16: definir regras para o protótipo PreyHunt. Nesta etapa o usuário ficava livre para planejar e implementar na interface as regras para definir a tomada de decisão do Caçador, sendo possível executar o protótipo para verificar o comportamento do Caçador e retornar ao conjunto de regras quantas vezes ele julgasse necessário.

Ao final, na etapa de crítica pessoal, o participante preenchia o formulário de avaliação da ferramenta, que simulava a avaliação da Unity Asset Store.

VII. Resultados

As Figuras 7 e 8 sintetizam os resultados obtidos em termos de taxas percentuais de sucesso e falha de acordo com as tarefas e os participantes (respectivamente). Estes resultados estão compilados em uma tabela e disponíveis em http://bit.ly/results-fuzzylogiceditor.

É importante observar que a quantidade de participantes no teste de usabilidade (7), bem como o possível viés dos usuários (já que são todos homens e moradores da mesma cidade) não nos permite obter dados conclusivos acerca da eficácia do *Fuzzy Logic Editor*. Entretanto, acreditamos que os resultados podem indicar preliminarmente algumas conclusões interessantes que podem ser, em momento posterior, referendadas por estudos mais completos.

De forma geral os usuário assimilaram as funcionalidades possíveis com facilidade, executando as tarefas corretamente em sua maioria, demonstrando compressão da ferramenta e de suas possibilidades. Das 16 tarefas definidas, apenas 4 apresentaram falhas (T3,T4,T12 e T16) e, ainda assim, o percentual de erro (14,3% para cada uma das tarefas) correspondia a apenas um dos 7 participantes (ver Figura 7). É interessante perceber que os erros foram esparsos (cada um dos quatro participantes que cometeu algum erro o fez em tarefas diferentes), o que nos indica que não parece haver problemas severos em alguma regra em específico.

A análise qualitativa indica que os usuários consideraram o *Fuzzy Logic Editor* uma interessante ferramenta auxiliar para a utilização da Lógica *Fuzzy* em jogos, o que corresponde ao objetivo principal deste trabalho. Isto pode ser percebido em algumas afirmações extraídas do formulário de avaliação da ferramenta:

- "Muito interessante para abstrair a lógica pra game design."
- "Estabelecer uma regra é rápido e fácil."
- "A criação da regra da lógica *fuzzy* estão funcionando quase que perfeitamente."
- "A ferramenta permite definir comportamentos para lógica *fuzzy* com grande praticidade (...) A funcionalidade da ferramenta agradou bastante e facilita muito a utilização da lógica *fuzzy*."

Além disso, durante a realização da tarefa T16 (definir as regras para o protótipo PreyHunt) foi possível perceber interesse em revisar e melhorar as regras para atingir o comportamento adequado do Caçador.

Apesar destes interessantes resultados, também foram perceptíveis possíveis pontos de melhoria na utilização da ferramenta: na utilização de parêntesis para agrupar condições e na organização das regras.

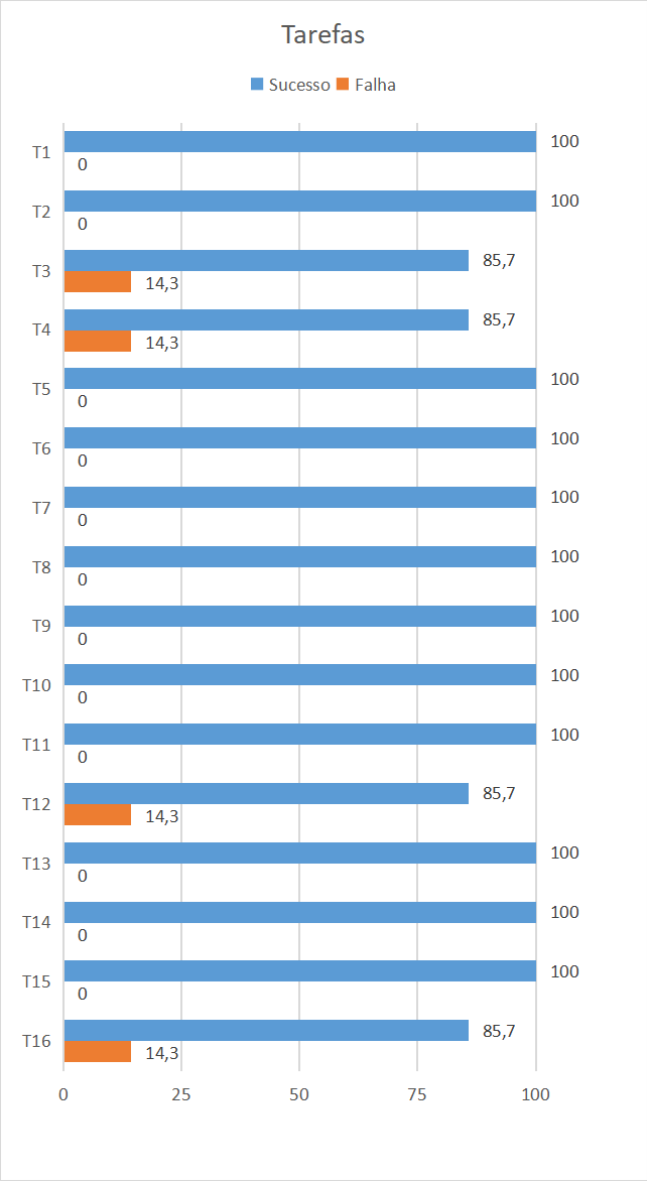

Figura 7. Gráfico de taxa de erros e acertos por tarefas

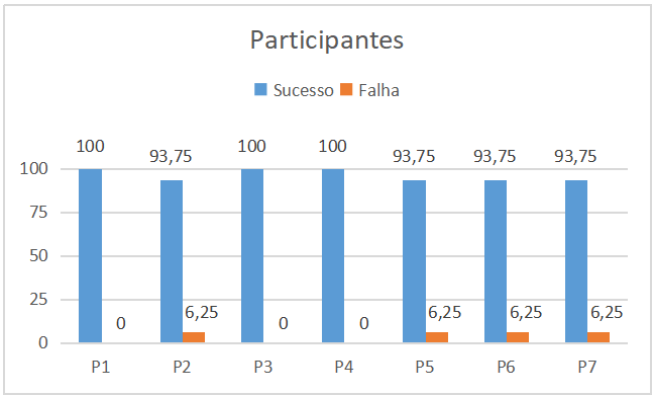

Figura 8. Gráfico de taxa de erros e acertos por participante

A. *Deixar mais clara a utilização de parêntesis para agrupar condições*

A funcionalidade de blocagem das condições, que representa adicionar parênteses para agrupar elementos de uma regra, apesar de corretamente realizada por todos os usuários (tarefa T8), representou problemas quando da definição das regras para o protótipo (tarefa T16). Os usuários relataram dificuldade em entender quais condições eram agrupadas quando um bloco era definido. Apesar disso, os próprios usuários apontaram que o campo *Rule String* atenua o problema. Este campo mostra a regra definida em sua forma escrita. Porém, eles não percebiam este campo a priori, ou seja, identificando-o apenas após investigarem a interface ao se sentirem perdidos.

Acreditamos que o campo *Rule String* é o caminho para a solução deste problema e que deve-se deixar mais explícita a sua existência desde o princípio da interação. Uma possibilidade seria mover o campo para a parte superior da interface, onde seria um dos primeiros elementos a serem observados.

B. *Facilitar a organização das regras*

Quando da realização da tarefa T16, os usuários relataram dificuldades em estabelecer ordem de prioridades para as regras e entender quais regras já haviam alterado previamente. Acreditamos que estas preocupações referem-se à organização das regras de um modo geral, que poderia ser melhorada. Sugerimos:

- possibilitar a renomeação das regras, e
- permitir arrastar as regras para reorganizá-las.

A renomeação ajudaria o usuário a estabelecer uma identidade para regra definida, enquanto reordená-las (através do arrasto do mouse) permitiria estabelecer ordens de prioridade. Neste ponto, é importante ressaltar que estas funcionalidades estavam previstas no projeto da ferramenta e que, entretanto, não foram implementadas devido à dificuldades inesperadas com o Editor da Unity que apresenta algumas limitações.

VIII. Conclusão

O desenvolvimento de jogos é uma tarefa multifacetada, que envolve diferentes competências e que pode se tornar um desafio, especialmente para equipes de tamanho reduzido. Neste contexto, este trabalho apresentou o *Fuzzy Logic Editor*, uma extensão para o motor de jogo Unity, cujo objetivo é auxiliar a aplicação da Lógica *Fuzzy* em jogos. O *Fuzzy Logic Editor* é, até onde sabemos, a primeira extensão do Unity grátis para esta finalidade. Desta forma, espera-se que a ferramenta auxilie principalmente equipes de desenvolvimento pequenas e que possuam recursos financeiros limitados.

Foi apresentado o design e o desenvolvimento da ferramenta, bem como um teste de usabilidade com usuários da Unity, que utilizaram o *Fuzzy Logic Editor* para estruturar regras para um protótipo previamente desenvolvido. Apesar do número de testadores não ser elevado (7), o perfil deles era especializado (conhecimentos prévios da *engine* e da Lógica *Fuzzy*), o que, espera-se, tenha trazido contribuições interessantes.

Os resultados dos testes foram majoritariamente positivos, o que indica preliminarmente a eficácia da ferramenta. A opinião dos usuários ainda embasou a sugestão de futuras alterações referentes à usabilidade da definição das regras.

Como trabalhos futuros pretende-se: (i) aplicar as melhorias sugeridas na avaliação; (ii) realizar nova avaliação, abordando todas as funcionalidades da ferramenta; (iii) estruturar a documentação completa, para publicação da ferramenta na Unity Asset Store; (iv) desenvolver um jogo que utilize a ferramenta; e (v) inserir funcionalidades adicionais na ferramenta, como diferentes *fuzzy sets* e conectores lógicos.

REFERÊNCIAS

[1] C.E. Crooks II. *Awesome 3D Game Development*. Charles River Media, Hingham, Massachusetts, 2004.

[2] D. M. Bourg, G. Seeman. *AI for Game Developers*. O'Reilly Media Inc., 2004.

[3] G. N. Yannakakis, J. Togelius. *Artificial Intelligence and Games*. Springer, 2018.

[4] M. Buckland. *Programming Game AI by Example*. Plano, Texas: Wordware Publishing Inc., 2005.

[5] L. A. Zadeh. *Fuzzy Sets*. Information and Control, Vol. 8, p. 338-353, 1965.

[6] UNITY TECHNOLOGIES. *Unity*. c2019. Página Inicial. Disponível em: https://unity.com/. Acesso em: 29 de mar. de 2019.

[7] ITCH CORP. *itch.io*, c2019. Most used Engines. Disponível em: https://itch.io/game-development/engines/most-projects. Acesso em: 29 de mar. de 2019.

[8] L.A. Zadeh. *Outline of a New Approach to the Analysis of Complex Systems and Decision Processes*. IEEE Transactions on Systems, Man and Cybernetics, Vol. 3, No. 1, pp. 28-44, 1973.

[9] E. H. Mamdani. *Application of fuzzy algorithms for control of simple dynamic plant*. Proceedings of the Institution of Electrical Engineers, vol. 121, no. 12, pp. 1585-1588, 1974.

[10] A. D. Santos. *Simulação Médica Baseada em Realidade Virtual para Ensino e Treinamento em Ginecologia*. 2010.

[11] M. Pirovano. *The use of Fuzzy Logic for Artificial Intelligence in Games*. 2012.

[12] I. Millington, J. Funge. *Artificial intelligence for games*, Second Edition 2nd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. 2009.

[13] T. N. Swe, C. Peters, A. S. Kyaw, R. Barrera. *Unity AI Game Programming - Second Edition*. Packt Publishing, 2015.

[14] M. Pirovano, P. L. Lanzi. *Fuzzy Tactics: A scripting game that leverages fuzzy logic as an engaging game mechanic*. 2014.

[15] L. O'Brien. *Fuzzy logic in games*. Game Developer Magazine, 1996.

[16] U. Köse. *Developing a fuzzy logic based game system*. Computer Technology and Application. 3. 510-517, 2012.

[17] Y. Li, P. Musilek, L. Wyard-Scott. *Fuzzy logic in agent-based game design*. Vol.2. 10.1109/NAFIPS.2004.1337393, 2004.

[18] A. Shaout, B. W. King, L. A. Reisner. *Real-time game design of pac-man using fuzzy logic*. Int. Arab J. Inf. Technol., 3(4):315–325, 2006.

[19] V. M. Teles. *Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade*. Novatec Editora, 2004.

[20] Guia do Scrum. *scrumguides.org*, c2019. Disponível em: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Portuguese-Brazilian.pdf. Acesso em: 04 de jun. de 2019.

[21] AForge.NET::Framework. *aforgenet.com*, c2008-2012. Disponível em: http://www.aforgenet.com/framework/. Acesso em: 02 de nov. de 2019.

[22] Unity - Scripting API: Editor. *unity.com*, c2019. Disponível em: https://docs.unity3d.com/ScriptReference/Editor.html. Acesso em: 02 de nov. de 2019.

[23] ISO 9241-11. Ergonomic requirements for office work with visual display terminals (vdts). *The International Organization for Standardization*, 45, 1998.

[24] J. S. Dumas, J. C. Redish. *A practical guide to usability testing*, Intellect books, 1999.

[25] jfuzzylogic. *jfuzzylogic.sourceforge.net*. Disponível em: http://jfuzzylogic.sourceforge.net/html/index.html. Acesso em: 04 de jun. de 2019.

[26] J. Ward. *What is a Game Engine?*. Games Career Guide, 2008. Disponível em: https://www.gamecareerguide.com/features/529/what_is_a_game_.php?page=1. Acesso em: 20 de nov. de 2019.

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

F336d Feitosa, Francisco Diego Moreira.
Desenvolvimento de um editor de Lógica Fuzzy para o motor de jogo Unity / Francisco Diego Moreira Feitosa. – 2019.
9 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Instituto UFC Virtual, Curso de Sistemas e Mídias Digitais, Fortaleza, 2019.
Orientação: Prof. Dr. Alysson Diniz dos Santos.

1. Jogos. 2. Desenvolvimento. 3. Lógica Fuzzy. 4. Unity. I. Título.

CDD 302.23
