Universidade Paulista - UNIP

Bruno de Paula Silva - C992534
Daniel Sousa David De Oliveira - D137GC0
Gustavo Felipe De Santana Marques - C993AH8
Marcelo Bueno Silva - N805CA0
Wesley Luiz Carvalho Silva - C993077

Biblioteca de Tomada de Decisão com Lógica Paraconsistente para jogos de cartas

São Paulo

2019

Bruno de Paula Silva - C992534
Daniel Sousa David De Oliveira - D137GC0
Gustavo Felipe De Santana Marques - C993AH8
Marcelo Bueno Silva - N805CA0
Wesley Luiz Carvalho Silva - C993077

Biblioteca de Tomada de Decisão com Lógica Paraconsistente para jogos de cartas

Trabalho apresentado para aproveitamento da disciplina Trabalho de Curoso II , do curoso de Ciência da Computação, Da Universidade Paulista - UNIP Campus Cidade Universitária.

Orientadora: Profª. Dr.a Amanda Luiza S. Pereira

São Paulo
2019

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Este trabalho acadêmico demonstra a aplicação da Lógica Paraconsistente Anotada (LPA), em jogos do gênero *Tranding Card Games* (Jogos de Cartas Colecionáveis – TCG), através da criação de uma biblioteca de Tomada de Decisão que implementa a Lógica Paraconsistente Anotada Evidencial (LPA Eτ), além disso será criado um jogo de cartas que utiliza a biblioteca para demonstrar as suas funcionalidades.

Com parte do senso comum, as pessoas acreditam que os jogos têm como única finalidade entreter, ignorando as diversas opções que um jogo eletrônico pode trazer para auxiliar o desenvolvimento humano, de modo que a utilização dos jogos de forma educacional pode resolver problemas usando raciocínio lógico e trazer benefícios à saúde (LOPES et al., 2011).

A LPA é uma lógica não clássica que admite contradições e incertezas, é uma boa solução para fazer tratamento de situações reais, no qual a Lógica Clássica, por ser binária, se mostra ineficaz ou impossibilitada de ser aplicada (SILVA FILHO, 2006). Assim possibilita as mais variadas aplicações em áreas tais como computação, robótica, tráfego aéreo e de trens, distribuição de energia em grandes usinas, programação, redes neurais, pesquisa operacional entre outras (CARVALHO; ABE, 2011).

Uma biblioteca é uma coleção de subprogramas ou um programa que facilita o desenvolvimento de sistemas, no núcleo da biblioteca desenvolvida será utilizado a LPA. Dessa forma, a biblioteca implementada no jogo será responsável por tomar as decisões dos resultados de batalha, sendo o intuito de criar um software que pode ser reutilizável por outros, iniciando um estudo da aplicação da LPA em jogos TGC.

Será explicado como a biblioteca foi desenvolvida e implementada no jogo, juntamente com a sua documentação para utilização. Também será relatado como o jogo foi desenvolvido, quais ferramentas e metodologias foram utilizadas.

Este documento está estruturado nos seguintes tópicos, *1 - Introdução* apresenta o projeto, os objetivos e as justificativas. No capítulo *2 - Referência Teórica* é exposta a base conceitual do projeto. Na seção seguinte *3 - Materiais e Métodos* é retratada a metodologia utilizada para desenvolvimento da prototipagem além das ferramentas utilizadas no processo. No *4 - Desenvolvimento* Descrição das etapas de elaboração da biblioteca e da criação do jogo.

1.1 Justificativa

A lógica paraconsistente introduz duas novas categorias além do Verdadeiro e do Falso. Podemos ter proposições classificadas como Verdadeiras, Falsas, Inconsistentes ou Paracompletas. Para uma proposição ser classificada com Inconsistente tem que haver uma evidência que sugere que ela seja Verdadeira e outra evidência aponte que ela é Falsa, agora quando não tem evidência Verdadeira nem tampouco que ela seja Falsa a proposição é classificada como Paracompleta (ABE, 2013).

O presente trabalho destaca a importância da LPA como alternativa à LC em situações onde há necessidade de lidar com informações contraditórias ou incompletas.

No desenvolvimento de jogos de cartas, são encontrados diversas bibliotecas disponibilizadas na *Unity Asset Store*, além disso recursos de textura, *script* (roteiro), design e tutoriais, porém nenhuma delas traz um recurso de uma lógica capaz de trazer uma alternativa diferente das LC que aceitam apenas verdadeiro ou falso, que possibilitam novas dinâmicas em jogos e resolução de problemas de contradição de informações.

A proposta do projeto é desenvolver uma biblioteca aplicando TD com a LPA ao invés do LC, para auxiliar no processo decisório aceitando valores contraditórios, possibilitando a criação de novas dinâmicas nos jogos e expandindo a LPA no desenvolvimento de jogos.

1.2 Objetivos

O objetivo geral é desenvolver uma biblioteca de Tomada de Decisão que utilize a LPA com foco em jogos TCG com objetivo de, protótipo final, isso é, biblioteca com manual de utilização e jogo de demostração, com o intuito de ser utilizada por outros desenvolvedores de jogos de cartas

1.3 Objetivos Específicos

- Criar o modelo de paraconsistente a ser utilizado.
- Criar uma versão da biblioteca com valores fixo para uma implementação no jogo sem muitos problemas.
- Desenvolver uma segunda versão da biblioteca, permitindo que ela seja genérica o suficiente para atender diferentes regras de negócio em jogos TCG.
- Desenvolvimento da biblioteca.
- Construir um manual de utilização e documentação da biblioteca.
- Criar um jogo do gênero TCG.

- Implementar as funcionalidades da biblioteca no jogo.
- Gerar *Asset* e disponibilizar na plataforma *Unity Asset Store*.
- Criar a presente documentação descrevendo os materiais, métodos e referências utilizadas para a construção do projeto.

Finalizada a apresentação da estrutura do trabalho, prossegue para *capítulo 2 - Referência Teórica*.

2 REFERÊNCIA TEÓRICA

Neste capitulo serão apontados quais as referências que incentivaram a escolha do tema em questão.

2.1 Lógica Paraconsistente

A LPA teve como precursores o lógico russo N. A. Vasiliev e o lógico polonês J.Lukasiewicz. Os dois em 1910, publicaram trabalhos independentes, porém se restringiam a lógica aristotélica tradicional. Entre 1948 e 1954 o lógico polonês S.Jaskowski e o lógico brasileiro Newton C.A. da Costa, independentes construíram a LPA (CARVALHO; ABE, 2011, p. 27).

Segundo Silva Filho (2010) dentre as várias ideias no âmbito das Lógicas não-Clássicas criou-se uma família de lógicas que teve como fundamento principal a revogação do princípio da Não Contradição, a qual foi nomeada de Lógica Paraconsistente. Portanto, a LPA é uma Lógica não-Clássica que revoga o princípio da Não Contradição e admite o tratamento de informações contraditórias na sua estrutura teórica.

2.2 Teoria dos jogos

A Teoria dos Jogos segundo Sartini et al. (2004), pode ser definida como a teoria dos modelos matemáticos que estuda a escolha de decisões ótimas sob condições de conflito. E os elementos básicos dessa Teoria, segundo o mesmo, quando cada jogador escolhe sua estratégia, temos então uma situação ou perfil no espaço de todas as situações (perfis) possíveis. Cada jogador tem interesse ou preferências para cada situação no jogo.

As definições que constituem esta Teoria de acordo com Cardoso e Campos (2018), visa compreender a racionalidade das decisões tomadas pelos jogadores, sempre com base na ideia de que impera a racionalidade na busca da melhor estratégia, ou seja, daquela que dará ao jogador a maior vantagem, seja na forma de mais lucro ou de mais satisfação.

2.3 Biblioteca

A Biblioteca fará o uso da LPA , com intuito de ajudar ou facilitar a criação de jogos de carta mais especificamente para jogos TCG, a *Library* vem com o objetivo de entregar funções já desenvolvida sem ter a necessidade de ter que elaborar as funcionalidades do zero. Com relação a *Application Programming Interface* (Interface de programação de aplicações - API) e um *Framework* a uma *Library* pode ter uma certa similaridade, mas tem conceitos distintos.

Biblioteca é uma conjunto de implementações de ações escritos em uma linguagem e importadas no seu código. Nesse caso, há uma interface bem definida para cada comportamento invocado.

A API tem um conceito diferente da Biblioteca, que basicamente contém um conjunto de instruções, rotina e padrões do código que Contém acesso de um aplicativo especifico via conexão. Com tudo ele vem pra interpretar os dados e integra-los com outras plataformas e softwares. (RIBEIRO, 2016).

O *Framework* é a base sólida e padronizada de uma aplicação resumidamente é a unificação de Bibliotecas e API's de forma a oferecer uma estrutura ideal para desenvolver um software (FINZI, 2016).

2.4 Engenharia de Software

Visando melhorar a qualidade dos produtos de software e aumentar a produtividade no processo de desenvolvimento, surgiu a Engenharia de Software. A Engenharia de Software trata de aspectos relacionados ao estabelecimento de processos, métodos, técnicas, ferramentas e ambientes de suporte ao desenvolvimento de software (FALBO, 2014, p. 2).

2.4.1 Metodologia

Metodologia é um conjunto de passos para alcançar um determinado objetivo. Em engenharia de software é conjunto de práticas que abrange todo o ciclo de vida do software que pode ser divida em três partes (ALMEIDA, 2017):

- Definição é etapa que feito o levantamento de informações, das funcionalidades desejadas, restrições e validação do projeto.
- Desenvolvimento é etapa de estruturação dos dados e planejamento de como fazer o software.
- Manutenção é etapa focada em correções de erros e melhorias no software.

2.4.2 UML

Particularmente no tocante à Engenharia de Software, a *Unified Modeling Language* (Linguagem de Modelagem Unificada - UML) pode ser utilizada para modelar todas as etapas do processo de desenvolvimento de software, bem como produzir todos os artefatos de Software necessários à documentação dessas etapas (GUDWIN, 2015, p. 12).

Segundo o mesmo autor (2015), a linguagem UML, por meio de seus diagramas, permite a definição e design de *threads* (tarefas) e processos, que permitem o desenvolvimento de

sistemas distribuídos ou de programação concorrente. Da mesma maneira, permite a utilização dos chamados *patterns* são, a grosso modo, soluções de programação utilizadas devido ao seu bom desempenho e a descrição de colaborações esquemas de interação entre objetos que resultam em um comportamento do sistema.

2.4.3 RUP

O Modelo *rational unified process* (Processo Unificado da Rational - RUP) foi criado pela *Rational Software corporation* adquirida posteriormente pela *International Business Machines* (IBM), que pode ser customizado de acordo com as necessidades do projeto, deixando assim, RUP mais leve (ágil) ou mais pesado (tradicional).

Conforme Piske (2003) "[...] RUP é mais do que um softwares para auxiliar no desenvolvimento é uma metodologia de desenvolvimento, com uma estrutura formal e bem definida."

Os ciclos de desenvolvimento são divididos em 4 etapas:

1. Iniciação - levantamento de requisitos, funções desejáveis e criação do escopo do projeto.
2. Elaboração - analisar o escopo do projeto, estabelecer arquitetura, coletar os requisitos, desenvolver um plano para o projeto e mitigar os riscos do projeto.
3. Construção - desenvolvimento e testes do software.
4. Transição - validação e entrega do projeto.

Além disso, o RUP possui nove disciplinas que contém *templates* (modelos) que abrange todo o ciclo de vida de software, esses *templates* ajudaram no planejamento do projeto, alguns modelos continham algumas perguntas que não se enquadraram no tamanho do projeto, porém o RUP é adaptável a diversos tipos de projetos tanto grandes quanto pequenos. Além de após efetuar o preenchimento dos *templates*, foi descoberto várias situações não previstas antes do preenchimento do RUP que puderam ser planejadas. Na seção de apêndices será apresentado alguns *templates* do RUP preenchidos com base no projeto desenvolvido pelo grupo para conclusão de curso.

Exposto as Referencial teórico do trabalho, segue para o capítulo *3 - Materiais e métodos.*

3 MATERIAIS E MÉTODOS

Para a elaboração do presente trabalho foram adotadas técnicas e metodologias que serão mostrados nos próximos tópicos.

3.1 Linguagem C#

C# ou *C Sharp* é uma linguagem de programação desenvolvido pela Microsoft, que permite a criação de uma variedade de aplicativos executados no .NET Framework. Pode ser aplicada no desenvolvimento de aplicativos cliente-servidor, serviços Web XML, componentes distribuídos, aplicativos cliente-servidor, aplicativos de banco de dados entre outros.

Tem como aspecto ser uma linguagem fortemente tipada que utiliza o paradigma de orientação a objeto com sintaxe semelhante às linguagens C,C++ ou JAVA. Segundo os autores Edwin e Eugenio (2002) Algumas das características essenciais do C# que podem ser mencionadas:

- Simplicidade: facilidade de codificação com alta performance;
- Completamente orientada a objetos: Tudo é um objeto em C#;
- Fortemente tipada: atribuição de tipos para evitar manipulação imprópria ou incorreta;
- Flexibilidade: caso necessário C# permitir o uso de ponteiros, ms com custo de desenvolver um código não gerenciado, chamado de *unsafe*;
- Linguagem gerenciada: os programa desenvolvido em C# é executado em ambiente gerenciado, na qual, o gerenciamento de memória é feito pelo *Garbage Collector* (Coletor de lixo - GC);

A linguagem C# é utilizada nesse projeto devido sua integração com a plataforma *Unity*, na qual, será mais detalhada no próximo tópico.

3.2 Plataforma Unity

A plataforma Unity é conhecido como uma das melhores plataforma de desenvolvimento de jogos do mundo, justamente porque ela é potencializada em serviços e ferramentas sendo elas 2D e 3D. Segundo Dias (2018) “atualmente a plataforma domina 45% do mercado global de desenvolvimentos de games, segundo a própria empresa; 34% dos 1.000 maiores jogos mobile disponibilizado gratuitamente são feitos com a própria Unity.”

3.3 Visual Studio

O Visual Studio é um *Integrated Development Environment* (Ambiente de desenvolvimento integrado - IDE) de *open source* (código-fonte aberto) desenvolvido pela Microsoft, com recursos usados para auxiliar e simplificar no desenvolvimento de software.

De acordo com a Microsoft (2019), o software é composto por editor, compilador, ferramentas de preenchimento de código e designers gráficos, com o objetivo de facilitar a edição, criação, a depuração, o *build* e a publicação do software desenvolvidos, além de possuir uma grande biblioteca de *plugins* para interpretação de várias linguagens de programação com exemplo a linguagem Visual Basic, C, C++, C#, F#.

Terminado a apresentação do materiais e métodos em seguida e apresentado o capitulo *4 Desenvolvimento.*

4 DESENVOLVIMENTO

Este capitulo mostra as etapas de desenvolvimento da biblioteca e do jogo.

4.1 Modelo de paraconsistente utilizado

O modelo criado com a LPA foi implementado em uma dinâmica de ataque já adotada por jogos famosos como o *Teamfight tactics* e o *Dota Underlords*, se passando em um cenário onde todas as cartas devem atacar um único inimigo. Com a LPA é atribuído um benefício especial nessa dinâmica, possibilitando incluir valores contraditórios para os atributos das cartas, onde as regras de atribuição desses valores contraditórios podem vir por normas resultantes de mudanças de cenários, escolha de *skills* e armamentos diferentes entre outras regras, por exemplo: Uma carta de um lutador de boxe que tem o atributo de força com o valor 88 e sua idade é de 60 anos, o valor contraditório sobre esse atributo pode ser mais que 70 pelo motivo de uma pessoa normal com mais de 60 anos possuir uma perda de massa muscular comum ao envelhecimento, além da diminuição de força nos membros inferiores

Foi criado um modelo para ser usado de base na criação da biblioteca que será usado para decidir o quanto as cartas atacando juntas conseguem tirar de vida de um adversário, criamos o modelo com oito cartas e com os atributos de força, velocidade, cardio e experiência com valores favoráveis e desfavoráveis para os quatro atributos de cada carta conforme a tabela 1.

Tabela 1 – Visualização das Cartas

Carta	Atributos	Favoravel	Desfavoravel
Arqueiro	velocidade	80	67
	força	20	10
	cardio	70	68
	experiência	60	63
Bárbaro	velocidade	60	63
	força	75	15
	cardio	60	65
	experiência	40	30
Guerreiro	velocidade	23	60
	força	80	60
	cardio	34	70
	experiência	30	20
Cavaleiro	velocidade	68	50
	força	55	40
	cardio	64	60
	experiência	30	69
Espadachim	velocidade	72	50
	força	56	48
	cardio	58	5
	experiência	59	35
Ancião	velocidade	23	68
	força	44	31
	cardio	39	68
	experiência	77	0
Vikin	velocidade	26	50
	força	45	70
	cardio	34	80
	experiência	77	0
Mosqueteiro	velocidade	53	67
	força	53	10
	cardio	53	57
	experiência	53	52

Fonte: Produzido pelos autores.

Esses detalhes das cartas e seus atributos podem ter uma ampla variedade, mudando de acordo com o tema do jogo de cartas.

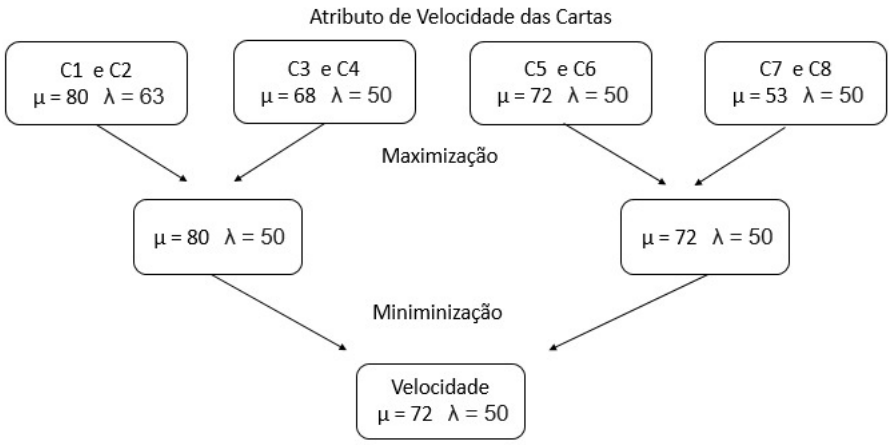
No primeiro passo do modelo é realizado o processo de maximização, a partir do qual se obtém os maiores valores das evidências favoráveis e os menores das evidências desfavoráveis, levando em consideração o primeiro atributo de todas as cartas, realizando a maximização entre as cartas arqueiro e bárbaro, repetindo o processo em relação às cartas guerreiro e cavaleiro, espadachim e ancião, viking e mosquiteiro.

Depois é realizado o processo de maximização novamente com os valores resultantes da

última maximização.

Na sequência, realiza-se o processo de minimização, no qual consiste na obtenção dos menores valores das evidências favoráveis e dos maiores valores das evidências desfavoráveis, as quais foram maximizadas anteriormente conforme a figura 1.

Figura 1 – Extração da contradição entre as cartas



Fonte: Produzido pelos autores.

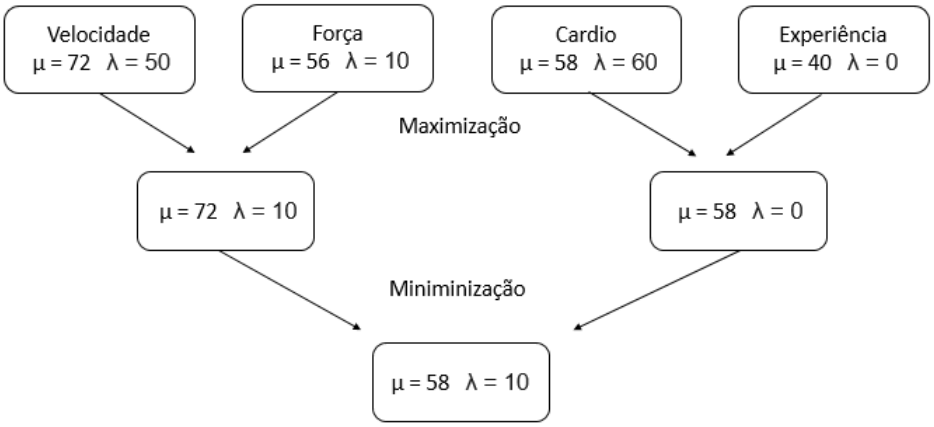
Após realizar os processos de maximização e minimização o processo deve ser refeito com todos os atributos das cartas, resultando no seguinte resultado exibido na figura 2:

Figura 2 – Atributos

Fonte: Produzido pelos autores.

Depois é realizado processo de maximização entre os atributos e depois minimização de acordo com a figura 3.

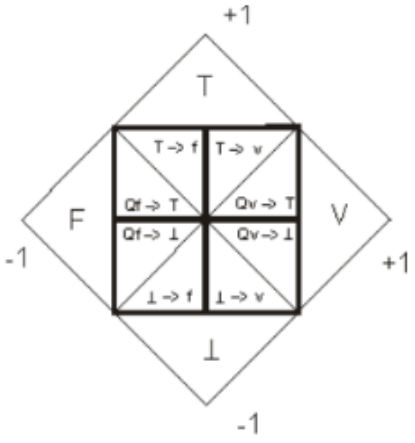
Figura 3 – Extração da contradição entre as cartas



Fonte: Produzido pelos autores.

Esses dois valores obtidos serão utilizados no algoritmo para-analisador. De maneira geral, para obter-se uma representação adequada da LPA e onde o resultado será inserido, utiliza-se um reticulado conforme a figura 4.

Figura 4 – Reticulado



Fonte: CARVALHO; ABE.

Como resultado das várias sentenças descritivas no reticulado representado no QUPC é proposto um algoritmo para implementação em um programa de computação convencional que possibilita a aplicação da LPA de anotação com dois valores LPA2v em Sistemas de Controle e

Especialistas de IA, logo abaixo foi feito o algoritmo já com a implementação do modelo visto neste capítulo (SILVA FILHO, 2006).

```
Valores de Entrada:
μ - Grau de Evidência Favorável  = 0.58
λ – Grau de Evidência Desfavorável = 0.10

Vscc – Valor Superior de Controle de Certeza = 0.9
Vscct – Valor Superior de Controle de Contradição = 0.5
Vicc – Valor Inferior de Controle de Certeza =  -0.5
Vicct – Valor Inferior de Controle de Contradição = -0.5

*/Variáveis de saída*
Saída discreta = S1



Algoritmo: /* Calcular graus de Certeza e Contradição: */
 Gc = μ - λ
 Gc = 0.58 - 0.10
 Gct = μ + λ – 1
 Gct = 0.58 + 0.10 - 1

/* Estados Lógicos Extremos */
 Se Gc ≥ Vscc então S1 = V
 Se Gc ≤ Vicc então S1 = F
 Se Gct ≥ Vscct então S1 = T
 Se Gct ≤ Vicct então S1 =  ⊥
 /*Estados Lógicos Não Extremos */
Para 0 ≤ Gc < Vscc e 0 ≤ Gct < Vscct
        Se Gc ≥ Gct
                Então S1 = Qv →T
                Senão S1 = T →v
Para 0 ≤ Gc < Vscc e Vicct < Gct ≤ Vscct
        Se Gc ≥ |Gct|
                Então S1 = Qv → ⊥
                Senão S1 = ⊥  →v
Para Vicc < Gc ≤ 0 e Vicct < Gct ≤ Vscct
        Se |Gc| ≥ |Gct|
                Então S1 = Qf → ⊥
                Senão S1 =  ⊥ → f
Para Vicc < Gc ≤ 0 e 0 ≤ Gct < Vscct
        Se |Gc| ≥ Gct
                Então S1 = Qf →T
                Senão S1 = T →f
 /* fim */
```

Abaixo na tabela 2 é demonstrado com tabelas a explicação de cada estado lógico.

Tabela 2 – Estados lógicos extremos

Estado	Definição
V	Verdadeiro
F	Falso
$\perp$	Indeterminado
T	Inconsistente

Fonte: Produzido pelos autores.

Além dos estados lógicos "extremos" a LPA2v permite a determinação de outros estados lógicos paraconsistentes listados abaixo conforme a tabela 3:

Tabela 3 – Estados lógicos não extremos

Estado	Definição
$T \rightarrow F$	Inconsistente tendendo ao falso
$T \rightarrow V$	Inconsistente tendendo ao verdadeiro
$\perp \rightarrow F$	Indeterminado tendendo ao falso
$\perp \rightarrow V$	Indeterminado tendendo ao verdadeiro
$Qf \rightarrow T$	Quase falso tendendo ao inconsistente
$Qf \rightarrow \perp$	Quase falso tendendo ao indeterminado
$Qv \rightarrow T$	Quase verdadeiro tendendo ao inconsistente
$Qv \rightarrow \perp$	Quase verdadeiro tendendo ao indeterminado

Fonte: Produzido pelos autores.

Aplicando o modelo criado no algoritmo paranalisador obtivemos o resultado: $\perp \rightarrow F$.

O resultado lógico obtido foi indeterminado tendendo ao falso e através do estado lógico retornado foi produzido o parecer analítico com uma tabela 4, cujo resultado refere-se a porcentagem que as oito cartas conseguem tirar de vida do adversário, nesse caso o ataque das oito cartas tirou 20% da vida do adversário.

Tabela 4 – Parecer analítico

Status	Parecer Analítico
V	100%
$Qv \rightarrow T$	90%
$Qv \rightarrow \perp$	80%
$T \rightarrow V$	70%
$\perp \rightarrow V$	60%
T	50%
$\perp$	40%
$T \rightarrow F$	30%
$\perp \rightarrow F$	20%
$F \rightarrow T$	10%
$F \rightarrow \perp$	5%
F	0%

Fonte: Produzido pelos autores.

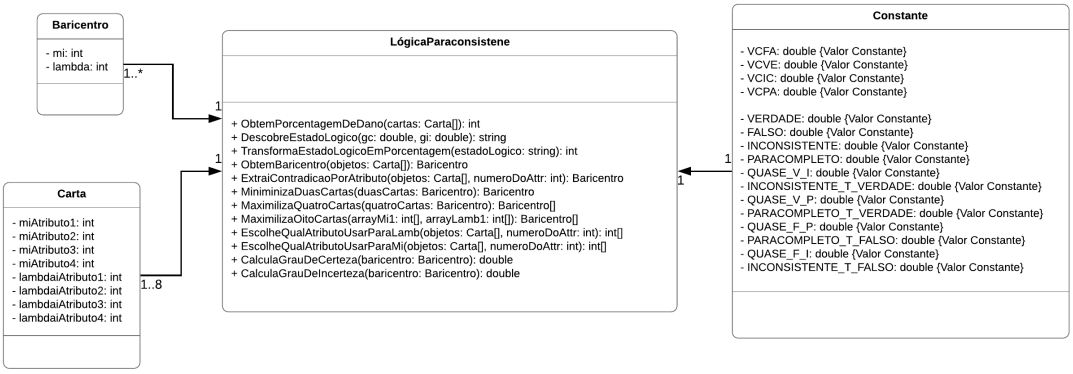
O parecer analítico sobre os estados lógicos foi desenvolvido, com o pensamento de que quanto mais o resultado fosse verdade maior a porcentagem de vida que as cartas conseguem tirar do adversário.

Os resultados obtidos através do uso do algoritmo podem então ser utilizados em processos de tomada de decisão nas mais variadas áreas incluindo robótica e engenharia de controle (ABE, 2013).

4.2 Criar da biblioteca para jogos de cartas

Após a criação do modelo foi criado a primeira versão da biblioteca cuja sua implementação foi feita na IDE Visual Studio criando uma projeto do tipo *console application* (aplicação console) devido sua facilidade e rapidez no desenvolvimento para realizar as validações e testes. Valendo do modelo conceitual já apresentado, a biblioteca foi criada seguindo a modelagem de classes UML na figura 5 cujo detalhes estão no anexo do RUP.

Figura 5 – Diagrama de Classe



Fonte: Produzido pelos autores.

A primeira versão da biblioteca foi construída pensada em somente receber os parâmetros fixos das oito cartas com quatro atributos cada, assim a sua integração com o jogo seria mais rápida, e o protótipo do jogo já poderia ser desenvolvido.

Essa versão é composta de quatro classes, classe principal Lógica paraconsistente que é responsável pelo fluxo de extrair os valores da cartas, processar e apresentar na saída o valor em porcentagem do estado lógico resultante. As classes baricentro e carta são classes auxiliares para extração de valores, e a constante contém os valores fixo utilizada de parâmetro na lógica.

4.3 Criação do jogo que integra a biblioteca

O jogo foi desenvolvido com o intuito de provar a efetividade da biblioteca reunindo valores contraditórios retirando as contradições e tomando a decisão sobre o resultado da batalha.

O nome do jogo é *Fighting Against Monsters*, o tema enquadrado se passa em diferentes cenários e as cartas são uma espécie de *crossover* entre inúmeros personagens de quadrinho, desenhos animados e personagens criados pela equipe.

Com o jogo criado é possível demonstrar que a integração com a biblioteca é realizável. Em um primeiro momento o jogo contaria com oito cartas atacando um adversário. Foi alterado para entre as 8 cartas o jogador escolher quatro cartas para atacar assim adicionado uma maior jogabilidade, isso foi possível devido a segunda versão da biblioteca que será apresentado no próximo capítulo.

O método que foi chamado da biblioteca é o ObtemPorcentagemDeDano onde é passado como parâmetro um *array* (Arranjo) de cartas e é retornado um inteiro com o número da porcentagem que as cartas que foram passadas como parâmetro conseguem tirar de vida do

adversário. Esse método chama todos os outros métodos da biblioteca, extraindo a contradição e depois aplicando o algoritmo da paraconsistente, os outros métodos também podem ser chamados separadamente dependendo da aplicabilidade no jogo desenvolvido. Na figura 6 está a tela inicial do jogo.

Figura 6 – Menu do Fighting Against Monsters

Fonte: Produzido pelos autores.

Todas as cartas atacam o inimigo ao mesmo tempo e a lógica de ataque com o resultado é realizada pela biblioteca. Na figura 7 e 8 está a tela de combate.

Figura 7 – Tela de combate Fighting Against Monsters

Fonte: Produzido pelos autores.

Figura 8 – Ataque das cartas

Fonte: Produzido pelos autores.

4.4 Segunda versão da biblioteca

Na segunda versão da biblioteca agora sendo nomeada com Decision Maker LPA, foi feito realizado uma refatoração na modelagem de classe, distribuindo a responsabilidade que antes estava exclusivamente na classe de lógica paraconsistente, adicionado melhor legibilidade ao código. Além disso, houve um aperfeiçoamento na implementação, possibilitando utilizar a biblioteca com quantidade de cartas da base dois.

O objetivo do desenvolvimento da biblioteca, além de expandir a LPA no mercado de jogos, é ajudar o desenvolvedor no unity a economizar horas em um projeto. Toda a construção da

biblioteca foi mensurada em horas do início ao fim, no total foram 50 horas de desenvolvimento, ou seja, 50 horas que o desenvolvedor que baixar a biblioteca obterá de tempo economizado e conseguirá focar em outros pontos importantes, como nas regras de negócio do jogo. Essas horas foram mensuradas apenas para a biblioteca e não para o jogo, demonstrando que a biblioteca obteve maior tempo de desenvolvimento, devido ao fato de criação de design e script da lógica de jogos de cartas.

4.5 Disponibilização no Assets Store

Como dito em capítulos anteriores o *Unity* é a maior plataforma de desenvolvimento de jogos e a *Unity Asset Store* é a plataforma para obter recursos pagos ou de graça para ajudar no desenvolvimento, esses recursos podem ser designs prontos, texturas, tutoriais, *scripts*, ou jogos betas para serem usados de modelo no desenvolvimento.A *Unity Asset Store* é uma biblioteca crescente de recursos, que são criados pela *Unity Technologies* e pelos membros da comunidade e publicados na loja.

Por isso o *Unity Assets Stores* é um dos lugares que a biblioteca e o jogo será disponibilizado

Em um primeiro momento foi pensado em cobrar pelo download do projeto, pois todos os recursos bons de design tem um preço razoável e os recursos de *scripts* são geralmente bem mais caros, por estarmos disponibilizando os dois tipos de recursos, seria interessante cobrar por isso, porém a decisão da equipe foi não cobrar, pois para expandir a lógica paraconsistente no mercado de jogos a maneira mais rápida seria disponibilizar a biblioteca e o jogo de graça.

REFERÊNCIAS

ABE, J. M. ***Aspectos de Computação Inteligente Paraconsistente***. [S.l.: s.n.], 2013. Citado 2 vezes nas páginas 7 e 20.

ALMEIDA, G. A. M. D. **Dissertação de mestrado Fatores de escolha entre metodologias de software tradicionais e ágeis**. 2017. Citado na página 10.

CARDOSO, M. J. R.; CAMPOS, C. R. ***A Teoria dos Jogos a mente brilhante de John Nash***. 2018. Disponível em: <https://www.researchgate.net/publication/315635843_A_TEORIA_DOS_JOGOS_E_A_MENTE_BRILHANTE_DE_JOHN_NASH>. Citado na página 9.

CARVALHO, F. R. de; ABE, J. M. ***Tomadas de Decisão com Ferramentas de Lógica Paraconsistente Anotada***. [S.l.: s.n.], 2011. Citado 3 vezes nas páginas 6, 9 e 17.

DIAS, R. ***Unity – Guia Completo sobre a Game Engine***. 2018. Disponível em: <https://producaodejogos.com/unity>. Citado na página 12.

EDWIN, L.; EUGENIO, R. ***C# E .NET Guia do Desenvolvedor***. [S.l.: s.n.], 2002. Citado na página 12.

FALBO, R. de A. ***Engenharia de Software***. 2014. Disponível em: <http://www.inf.ufes.br/~falbo/files/ES/Notas_Aula_Engenharia_Software.pdf>. Citado na página 10.

FINZI, E. ***API,biblioteca e Frameworks: entenda a diferença entre eles.*** 2016. Disponível em: <https://blog.cedrotech.com/api-bibliotecas-e-frameworks-entenda-diferenca-entre-eles/>. Citado na página 10.

GUDWIN, R. R. ***Engenharia de Software Uma Visão Prática***. 2015. Disponível em: <http://faculty.dca.fee.unicamp.br/gudwin/sites/faculty.dca.fee.unicamp.br.gudwin/files/ea975/ESUVP2.pdf>. Citado na página 10.

LOPES, D. D. et al. **Jogos de gestão e estratégia voltados para educação e apoiados em lógica paraconsistente anotada evidencial Eτ**. *XL IGIP*, p. 1, 2011. Citado na página 6.

MICROSOFT. *Visão Geral Visual Studio - Microsoft Docs*. 2019. Disponível em: <https://docs.microsoft.com/pt-br/visualstudio/get-started/visual-studio-ide?view=vs-2019>. Citado na página 13.

PISKE, O. R. **RUP – Rational Unified Process**. 2003. Citado na página 11.

RIBEIRO, M. ***O que é API e como ele aumenta a produtividade nas empresas.*** 2016. Disponível em: <https://pluga.co/blog/api/o-que-e-api/>. Citado na página 10.

SARTINI, B. A. et al. **Uma Introdução a Teoria dos Jogos**. *Universidade Federal da Bahia*, 2004. Citado na página 9.

SILVA FILHO, J. I. da. **Métodos de Aplicações da Lógica Paraconsistente Anotada de anotação com dois valores-LPA2v**. *seleção documental do GLPA*, p. 20–24, 2006. Citado 2 vezes nas páginas 6 e 18.

SILVA FILHO, J. I. da. **Introdução ao conceito de estado Lógico Paraconsistente $\epsilon\tau$** . *seleção documental do GLPA*, p. 20–24, 2010. Citado na página 9.

Apêndices

APÊNDICE A – VISÃO DE NEGÓCIO

BGMDW

DecisionMakerLPA
Visão do Negócio

Versão 6.0

Histórico da Revisão

Data	Versão	Descrição	Autor
04/05/2019	1.0	Preenchimento dos tópicos	BGMDW
11/05/2019	2.0	Preenchimento dos tópicos 6 e 7	Bruno de Paula
31/05/2019	3.0	Revisão do documento	Bruno de Paula, Gustavo Marques
01/06/2019	4.0	Revisão do documento	Bruno de Paula
12/10/2019	5.0	Revisão do documento	Bruno de Paula
30/10/2019	6;0	Reformulação do documento	Gustavo Marques

Índice Analítico

Visão do Negócio

1. Introdução

Apresentação da documentação com foco na visão de negócio.

1.1 Finalidade

Este documento tem como objetivo apresentar a área de negócio do projeto, permitido à equipe de desenvolvimento uma visão sobre o negócio.

1.2 Escopo

Desenvolver uma biblioteca que auxiliará no desenvolvimento de jogos de cartas na plataforma Unity aplicando a lógica paraconsistente para tomada de decisão.

1.3 Definições, Acrônimos e Abreviações

LPA – Lógica Paraconsistente
TCG – Trading Card Game

1.4 Referências

Não se aplica.

1.5 Visão Geral

Nesse documento expõem sobre problema que produto se propõem a resolver, as definições do produto, as pessoas envolvidas e uma visão demográfica do mercado cuja o produto está inserido.

2. Posicionamento

Detalha a relação entre o produto e problema que mesmo é proposta a resolver.

2.1 Oportunidade de Negócios

Criar novas mecânicas de jogar para gênero TCG, que se beneficia da lógica da paraconsistente para tomada de decisão e resolver conflito de valores contraditórios.

2.2 Descrição do Problema

O problema de	Tomada de decisão para saber qual conjunto de carta é mais forte.
afeta	Nos jogos do gênero TCG.
cujo impacto é	Decisão de quais cartas são mais fortes no combate.
uma boa solução seria	Utilização da biblioteca que implementa a lógica paraconsistente.

2.3 Sentença de Posição do Produto

Para	Desenvolvedores de jogos que usa a plataforma *Unity.*
Quem	Cria jogos do gênero TCG.
O DecisionMakerLPA	É uma biblioteca que aplica a lógica paraconsistente para tomada de decisão.
Que	Trata os valores contraditórios na hora de decidir quais cartas são mais fortes.
Diferente de	Aplicações que utiliza método básico de ataque contra defesa.
Nosso produto	Criação de novas mecânicas de jogos.

3. Descrições dos Envolvidos e dos Clientes

Descreve sobre empresa, os envolvidos diretamente, os usuários do produto e clientes.

3.1 Demografia do Mercado

A BGMDW é uma nova empresa de tecnológica que tem como foco no desenvolvimento de jogos para plataforma mobile e em soluções que de suporta nas etapas de desenvolvimento. Tem como meta ser uma empresa que cria produtos que agilize e melhores os processos de desenvolvimento de jogos. O DecisionMakerLPA é primeira criação que visa disponibilizar aos desenvolvedores uma biblioteca que usufrui da lógica paraconsistente.

3.2 Resumo dos Envolvidos

Nome	Descrição	Responsabilidades
Bruno de Paula	Gerência do projeto	Garantir que o projeto seja concluído e os objetivos alcançados, definindo objetivo geral do projeto, objetivos individuais, cronograma de atividades, responsabilidades e recursos.
Daniel Oliveira	Desenvolvedor Unity	Preencher documentação e desenvolver um jogo que servirá para demonstrar as funcionalidades da biblioteca.
Gustavo Marques	Analista de Desenvolvimento	Desenvolver a biblioteca DecisionMakerLPA, prestar suporta na integração com o jogo. Documentar as funcionalidades da biblioteca.
Marcelo Bueno	Designer e desenvolvedor	Criar as artes do para jogo de demonstração, além de auxiliar no desenvolvimento.
Wesley Luiz	Assistente do projeto	Toda documentação referente ao projeto.

3.3 Resumo dos Usuários

Nome	Descrição	Responsabilidades	Envolvido
Desenvolvedores de jogos	Criador de jogos mobile do gênero TCG utilizando a plataforma Unity.	Mecânica do jogo e suas interações com os usuários.	Usuário direto.

3.4 Ambiente do Usuário

Usuário-alvo trabalha utilizando a plataforma *Unity* para desenvolvimento de jogos mobile, essa plataforma tem uma loja de produto que contém funções relativas a criação de jogos já implementadas com física, artes, áudio, entre outros.

3.5 Perfis dos Envolvidos

3.5.1 BGMDW

Representante	Bruno de Paula, Daniel Oliveira, Gustavo Marques, Marcelo Bueno, Wesley Luiz
Descrição	Equipe responsável pelo projeto DecisionMakerLPA.
Tipo	Gerente, desenvolvedor, designer, assistente
Responsabilidades	Captar as necessidades dos desenvolvedores de jogos que desejam criar jogos de cartas e criar uma biblioteca que auxilie nas tomadas de decisões.
Critérios de Sucesso	Desenvolver um produto de qualidade que supra a necessidade dos desenvolvedores de jogos de cartas no quesito qual conjunto de cartas é mais forte.
Envolvimento	Analistas de desenvolvimento do software, abrangendo todas as etapas do processo
Produtos Liberados	Não
Comentários e Problemas	Dificuldade na gerência do projeto.

3.6 Perfis dos Clientes

3.6.1 Desenvolvedor de Jogos

Representante	Não possui
Descrição	Desenvolvedor de jogos do tipo TCG na plataforma Unity
Tipo	Usuário final
Responsabilidades	Desenvolver a mecânicas do jogo.
Critérios de Sucesso	O sucesso será que o framework a ser desenvolvido supra suas necessidades e ajude na tomada de decisão para saber qual conjunto de cartas venceu o combate.
Envolvimento	Informar os requisitos necessários.
Produtos Liberados	Pode ser utilizado em produtos já criados ou não, desde que seja jogo do tipo TGA e desenvolvido na plataforma Unity.
Comentários e Problemas	Não possui comentário.

3.7 Necessidades dos Principais Envolvidos ou dos Clientes

Necessidade	Prioridade	Preocupações	Solução Atual	Soluções Propostas
Lidar com valores contraditórios na toma de decisão de quais cartas são mais fortes no campo.	Alta	Facilidade de utilização do framework	Utilizar o conceito básico de atributo ataque contra defesa para decidir qual carta é mais forte.	Utilizar a DecisionMakerLPA para implementar novas mecânicas de jogos

3.8 Alternativas e Concorrência

As alternativas e concorrência ao produto são, utilizar o modelo básico de combate em jogos de cartas, cuja a decisão é feita através do maior entre os atributos de ataque e defesa. Outra opção é utilizar a lógica *Fuzzy* para implementar uma nova mecânica de jogo.

4. Objetivos da Modelagem de Negócios

Descreve de ponto de vista de negócios os objetivos que envolve o projeto.

4.1 Entendimento do Cliente

Com a modelagem de negócios, a intenção principal é identificar a necessidade do cliente para desenvolver um framework que supra as suas necessidades

4.2 Excelência no produto

A partir do desenvolvimento da modelagem de negócios, após ter identificado o problema, e assim possamos entregar um framework de qualidade, para que possamos deixar um legado para o desenvolvimento de jogos

5. Restrições

Um jogo do gênero TCG desenvolvido na plataforma Unity.

6. Faixas de Qualidade

A biblioteca precisa simplificar o uso da lógica paraconsistente e ter um mínimo de falhas na hora de processar e apresentar o resultado.

7. Precedência e Prioridade

A prioridade é criar um framework que vai auxiliar na tomada de decisão de qual conjunto de cartas é maior forte, possibilitando explorar novas mecânicas de jogo.

8. Outros Requisitos

Os requisitos de hardware do framework estão diretamente ligados com requisitos do Unity. Os requisitos para utilização do framework são ser um jogo do tipo TCG e ser desenvolvido na plataforma Unity.

8.1 Padrões Aplicáveis

Não se aplica.

8.2 Requisitos do Sistema

Os requisitos para utilizar a DecisionMakerLPA está ligado com requisitos da plataforma *Unity. Segue a lista:*

- OS: Windows 7 SP1+, 8, 10, 64-bit versão somente; macOS 10.12+; Ubuntu 16.04, 18.04, e CentOS 7.
- CPU: Suporte para o conjunto de instruções SSE2
- GPU: placa gráfica com recursos DX10 (shader model 4.0).
- Android: Android SDK e Java Development Kit (JDK); O scripting backend IL2CPP necessita do NDK de Android.
- OS: Mac computador executando no mínimo macOS 10.12.6 e Xcode 9.4 ou superior
- Plataforma Universal do Windows: Windows 10 (64 bits), Visual Studio 2015 com componente C++ Tools ou posterior e SDK para Windows 10.

8.3 Requisitos de Desempenho

Não se aplica.

8.4 Requisitos Ambientais

Não se aplica.

Apêndice 1 – Atributos dos Objetivos

- Biblioteca – Critico – Aprovado
- Jogo de demonstração –Importante – Aprovado

APÊNDICE B – AVALIAÇÃO DA ORGANIZAÇÃO

BGMDW

DecisionMakerLPA
Avaliação da Organização-alvo

Versão 6.0

Histórico da Revisão

Data	Versão	Descrição	Autor
04/05/2019	1.0	Preenchimento dos tópicos	BGMDW
11/05/2019	2.0	Preenchimento dos tópicos 6 e 7	Bruno de Paula
31/05/2019	3.0	Revisão do documento	Bruno de Paula, Gustavo Marques
01/06/2019	4.0	Revisão do documento	Bruno de Paula
30/10/2019	5.0	Reformulação do documento	Gustavo Marques

‘

Índice Analítico

Avaliação da Organização-alvo

1. Introdução

Apresentação da documentação com foco na organização alvo.

1.1 Finalidade

Apresentar os pontos da organização alvo e informações sobre a BGMDW e seu negócio.

1.2 Escopo

Uma avaliação da empresa BGMDW que está desenvolvendo o projeto DecisionMakerLPA.

1.3 Definições, Acrônimos e Abreviações

TCG – Trading Card Game.

1.4 Referências

Não possui referências.

1.5 Visão Geral

Esse documento informa o contexto do negócio, as ideias e estratégias da empresa. Lista os fatores externos e interno.

2. Contexto do Negócio

A BGMDW utilizando a lógica paraconsistente como tomada de decisão em um Framework para jogos de cartas na Unity, na linguagem C#. Captando empresas de desenvolvimento com o objetivo de aplicar a lógica paraconsistente de maneira que facilite o uso e compreensão.

3. Ideias e Estratégias de Negócios no Contexto do Projeto

A ideia de oferecer uma maneira fácil e objetiva de utilizar a lógica paraconsistente em tomada de decisão no mercado de desenvolvimento de jogos, especificamente do gênero TCG desenvolvido na plataforma *Unity.*

4. Fatores Externos

Descrição das pessoas envolvidas no projeto.

4.1 Clientes

Desenvolvedores de jogos de carta do gênero TCG, desde o desenvolvedor independente, os pequenos e médios estúdios até os grandes estúdios.

4.2 Concorrentes

Utilizar modelo básico para decidir qual carta é mais forte ou implementar a lógica *Fuzzy.*

4.3 Outros Envolvidos

A plataforma e game engine *Unity* desenvolvida pela Unity Technologies.

5. Fatores Internos

5.1 Processos de Negócios

A biblioteca e jogo de demonstração será disponibilizado na loja de *assets store* (loja de ativos) da *Unity,* essa loja pode ser acessada por qualquer desenvolver que desenvolva na engine *Unity* e possua uma conta na loja.

5.2 Ferramentas de Suporte

O suporte será através da assets store respeitando suas políticas.

5.3 Organização Interna

Equipe de desenvolvimento - responsável pelo desenvolvimento pela lógica de tomada de decisão paraconsistente aplicada no framework.

Equipe de designer - responsável pelo estilo artístico do jogo Competências, Habilidades e Atitudes

Equipe de Gerência – responsável por gerenciar e garantir a entrega do produto.

5.4 Competências, Habilidades e Atitudes

Bruno de Paula

- Gerência o projeto.
- Desenvolver a biblioteca.
- Documentação do projeto.

Daniel Oliveira

- Criação do modelo do jogo.
- Desenvolver jogo de demonstração.
- Documentação do projeto.

Gustavo Marques

- Desenvolver a biblioteca
- Suporte com a integração da biblioteca com o jogo.
- Documentação da biblioteca e do projeto.

Marcelo Bueno

- Criação de artes para jogo
- Suporte com a integração da biblioteca com o jogo.
- Documentação do jogo e do projeto.

Wesley Luiz

- Criação dos modelos de venda do projeto
- Gerenciar as entrega do jogo.
- Documentação da empresa.

5.5 Capacidade de Mudança

BDGDW é empresa nova disposta a encarar novos desafios.

6. Resultados da Avaliação de Desempenho

Não se aplica ao projeto.

7. Desempenho da Organização-Alvo

Não se aplica ao projeto.

8. Conclusão da Avaliação

Não se aplica ao projeto.

8.1 Áreas com Problemas

Não se aplica ao projeto.

8.2 Novas Tecnologias Aplicáveis

Não se aplica ao projeto.

APÊNDICE C – DOCUMENTO DE ARQUITETURA DE NEGÓCIOS

BGMDW

DecisionMakerLPA
Documento de Arquitetura de Negócios

Versão 3.0

Histórico da Revisão

Data	Versão	Descrição	Autor
04/05/2019	1.0	Preenchimento dos tópicos	BGMDW
01/06/2019	2.0	Revisão do documento	Bruno de Paula
30/10/2019	3.0	Reformulação do documento	Gustavo Marques

Índice Analítico

Documento de Arquitetura de Negócios

1. Introdução

Documenta as arquiteturas de negócios.

1.1 Finalidade

Listar e descrever as visões da empresa BGMDW.

1.2 Escopo

Compreender de maneira objetivas as perspectivas da empresa.

1.3 Definições, Acrônimos e Abreviações

Não contém Definições, acrônimos ou abreviações.

1.4 Referências

Não possui referência para outro documento.

1.5 Visão Geral

Apresenta as visões de processo de negócio, estrutura organizacional, cultural e recursos humanos.

2. Representação Arquitetural

A empresa ainda não possui um modelo arquitetural definido para expor suas visões.

3. Metas e Restrições da Arquitetura

Aplicar de soluções que agilize a criação de jogos, utilizar lógica não clássica com exemplo a lógica paraconsistente para ligar com situações reais que exige tomada de decisão.

4. Visão do Processo de Negócios

Desenvolver soluções que facilite o processo de criação de jogos.

5. Visão da Estrutura Organizacional

Apresenta o modelo de negócio da empresa.

5.1 Realizações de Casos de Uso de Negócios

Criação de jogos que são vendidas nas lojas de aplicativos com exemplo Google Play e criação de biblioteca que ajude no desenvolvimento de jogos disponibilizadas na loja assets store da *Unity*.

6. Visão Cultural

BGMDW tem uma cultura de pessoas novas com espirito para encarar desafios com foco e dedicação.

7. Visão dos Aspectos de Recursos Humanos

Ainda não a processo para desenvolvimento de recursos humanos.

8. Visão de Domínio (opcional)

Empresa BGMDW tem como especificidade a área de desenvolvimento de jogos.

9. Metas de Tamanho e Desempenho

Ser uma biblioteca robusta que através da lógica paraconsistente possibilite tomada decisão para vários cenários, até mesmo com os que possui contradições.

10. Metas de Qualidade

Qualidade do produto está relacionado com a usabilidade e confiabilidade, ou seja, que produto é fácil de ser utilizado e ele entrega resultado conforme foi planejado.

APÊNDICE D – PLANO DE GERENCIAMENTO DE REQUISITOS

BGMDW

Decision Maker LPA
Plano de Gerenciamento de Requisitos

Versão 4.0

Histórico da Revisão

Data	Versão	Descrição	Autor
25/05/2019	1.0	Preenchimento do Plano de Gerenciamento de Requisitos	Wesley Luiz
28/05/2019	2.0	Refatoração de todos os capítulos	Bruno de Paula
12/10/2019	3.0	Revisão de todos os capítulos	Bruno de Paula
16/11/2019	4.0	Atualização de todos os capítulos	Gustavo Marques

Índice Analítico

Plano de Gerenciamento de Requisitos

1. Introdução

Apresentar um plano de ação para levantar, analisar e gerenciar os requisitos do projeto.

1.1 Finalidade

Descrever os requisitos funcionais e não funcionais como serão obtidos, analisados, documentados e gerenciados do início ao fim do projeto.

1.2 Escopo

Identificar os requisitos, classifica-los em categoria, documentar para criar métricas para rastreabilidade possibilitando criação de estratégias para gerenciar os requisitos.

1.3 Definições, Acrônimos e Abreviações

LPA – Lógica Paraconsistente

TCG – Trading Card Gane

1.4 Referências

Solicitações dos Principais Envolvidos (STR) - APS-SPE-19-10-2019

Visão (VIS) - APS-VS-19-10-2019

Modelo de Casos de Uso - APS-UC-19-10-2019

1.5 Visão Geral

Mapear os requisitos do sistema, definindo status e prioridade. Elabora estratégia para gerenciar e garantir que mesmo sejam cumpridos.

2. Gerenciamento de Requisitos

2.1 Organização, Responsabilidades e Interfaces

A equipe foi dividida em dois grupos, um responsável pela biblioteca e outro pelo jogo de demonstração. Primeiro grupo é liderado pelo Bruno de Paula, o segundo grupo é liderado por Daniel Oliveira.

2.2 Ferramentas, Ambiente e Infraestrutura

A ferramenta utilizada foi o Trello no ambiente de web, o mesmo é um aplicativo de gerenciamento de projeto que utiliza o paradigma Kanban popularizada pela empresa Toyota.

3. O Programa de Gerenciamento de Requisitos

3.1 Identificação de Requisitos

Artefato (Tipo de Documento)	Item de Rastreabilidade	Descrição

Solicitações dos Principais Envolvidos (STR)	Solicitação do Envolvido (STRQ)	Documento com objetivo de entrevistar o usuário alvo para levantar os requisitos, problemas e alinhamento e entendimento com o usuário.
Visão (VIS)	Necessidade dos Envolvidos (NEED)	documento fornece uma visão sobre os usuários que utilizarão o produto, dos envolvidos que de alguma maneira participarão no processo de criação, implementação e manutenção do produto e dos produtos, informando os requisitos, recursos e benefício do mesmo.
Modelo de Casos de Uso	Caso de Uso (UC)	Descreve utilizando o diagrama de caso de uso todas as interações com o sistema.

3.2 Rastreabilidade

Desenvolvimento da biblioteca como lógica paraconsistente e jogo de demonstração do gênero TCG.

3.2.1 Critérios de Biblioteca.

A biblioteca deve implementar a lógica paraconsistente para lidar com valores contraditórios ou indefinidos, recebendo quantidade de cartas que seja da base dois, cada um com quatro atributos favoráveis e quatro desfavoráveis.

3.2.2 Critérios do Jogo de demonstração

Jogo de demonstração tem que ser desenvolvido na engine *Unity* do gênero TCG para plataforma mobile Android. Além disso, deve ser implementar na mecânica de combate a biblioteca Decision Maker LPA.

3.3 Atributos

3.3.1 Atributos de Biblioteca

Status

Proposto	Desenvolver uma biblioteca que aplique lógica paraconsistente em jogos do gênero TCG.
Aprovado	Desenvolver uma biblioteca que aplique lógica paraconsistente em jogos do gênero TCG.
Rejeitado	Nenhuma rejeição
Incorporado	Nenhuma incorporação

Benefício

Crítico	O Funcionamento correto da lógica paraconsistente.
Importante	Facilidade de utilização da biblioteca, Desempenho e qualidade.
Útil	Lidar com quantidade de cartas que abrange além da base dois.

Esforço

Todo fim de semana a equipe se reúne para discutir sobre o projeto e planeja o que será desenvolvido no próprio fim de semana e durante a semana, aumenta ou diminui o esforço em caso de atrasos do projeto.

Risco

A falta de comunicação entre os envolvidos da equipe pode ser um risco, caso alguém realize uma atividade ou desenvolvimento no jogo que não é um requisito funcional da biblioteca, por não consultar a equipe responsável.

Estabilidade

A prioridade de desenvolvimento é a biblioteca implementando a lógica paraconsistente.

Release-alvo

Primeira versão vai ser no *console application* (aplicação em console) do Visual Studio, as versões seguintes já serão integradas no jogo de demonstração.

Atribuído a

Bruno de Paula, Gustavo Marques.

Motivo

Pilar do projeto, aplicar lógica paraconsistente na área de jogos.

3.3.2 Atributos de Jogo

Status

Proposto	Desenvolver um jogo para demonstra aplicação da biblioteca.
Aprovado	Criar um protótipo que mostre a biblioteca implementada e integrada no jogo.
Rejeitado	Nenhuma rejeição
Incorporado	Nenhuma incorporação

Benefício

Crítico	Um jogo do gênero TCG que utilizar a biblioteca.

Importante	Jogo possui oitos fases e loja de compra de cartas.
Útil	Possuir todas das funcionalidades de um jogo de gênero TCG.

Esforço

Todo fim de semana a equipe se reúne para discutir sobre o projeto e planeja o que será desenvolvido no próprio fim de semana e durante a semana, aumenta ou diminui o esforço em caso de atrasos do projeto.

Risco

A falta de comunicação entre os envolvidos da equipe pode ser um risco, caso alguém realize uma atividade ou desenvolvimento no jogo que não é um requisito funcional da biblioteca, por não consultar a equipe responsável.

Estabilidade

A prioridade de desenvolvimento é o jogo que utilize a biblioteca e facilite o entendimento da lógica paraconsistente.

Release-alvo

As versões do jogo serão disponibilizadas nas lojas Assets Store e no Google Play.

Atribuído a

Daniel Oliveira, Marcelo Bueno e Wesley Luiz.

Motivo

Através do jogo, divulgar os benefícios de aplicar lógica paraconsistente.

3.4 Relatórios e Medidas

Atividade, conteúdo, atribuído a quem, prazo de entrega.

3.5 Gerenciamento de Mudanças de Requisitos

3.5.1 Processamento e Aprovação de Solicitações de Mudança

Os problemas e mudanças são lista e categorizado no software Trello e precisa da autorização da gerência para entra na linha de produção.

3.5.2 Comitê de Controle de Mudança (CCB)

Processo de mudança exige a autorização do gerente do projeto e aprovação da equipe.

3.5.3 Baselines do Projeto

Não se aplica no projeto.

3.6 Fluxos de Trabalho e Atividades

O fluxo é divido em dois, um responsável pela atividade de desenvolvimento e documentação da biblioteca e outro responsável pelo desenvolvimento do jogo, design e documentação.

4. Marcos

Desenvolver a biblioteca em *console application (*aplicação em console), realizar teste de funcionalidades, desenvolver a base do jogo do gênero TCG, realizar integração da biblioteca no jogo, efetuar melhorias na biblioteca e no jogo e realizar os testes para verificar o se os requisitos estão sendo cumprido.

5. Treinamento e Recursos

Para auxiliar no desenvolvimento do jogo foi comprado dois cursos de desenvolvimento de jogos no Unity no portal da Udemy.

APÊNDICE E – VISÃO

BGMDW

Decision Maker LPA
Visão (Projeto Pequeno)

Versão 3.0

Histórico da Revisão

Data	Versão	Descrição	Autor
18/05/2019	1.0	Preenchimento do modelo Visão	Gustavo Marques
19/10/2019	2.0	Revisão de todos os capítulos	Marcelo Bueno
16/11/2019	3.0	Revisão de todos os capítulos	Bruno de Paula

Índice Analítico

Visão (Projeto Pequeno)

1. Introdução

Esse documento fornece uma visão sobre os usuários que utilizarão o produto, dos envolvidos que de alguma maneira participarão no processo de criação, implementação e manutenção do produto e dos produtos, informando os requisitos, recursos e benefício do mesmo.

1.1 Referências

Esse Documento não faz referência a outro documento.

2. Posicionamento

2.1 Descrição do Problema

O problema	A aplicabilidade de lógica não clássica (Paraconsistente) em jogos de cartas.
Afeta	Desenvolvedores de jogos
cujo impacto é	Não pode lidar com contradição ou inconsistências
uma boa solução seria	Encapsulamento da lógica paraconsistente, facilidade da utilização da mesma e criação de dinâmicas de jogos diferentes dos habituais.

2.2 Sentença de Posição do Produto

Para	Desenvolvedores de jogos da plataforma Unity
Quem	Precisa lidar com contradição em jogos de cartas
O (nome do produto)	É uma biblioteca para plataforma Unity
Que	Facilita a utilização da lógica paraconsistente no confronto de duas cartas ou mais pares de cartas, decidindo a cartas vencedoras
Diferente de	Utilizar lógica clássica (Verdadeiro/Falso) ou implementar a lógica manualmente para desenvolver o jogo de cartas.
Nosso produto	Isolar a lógica Paraconsistente, exigindo o mínimo de conhecimento da mesma para utilização

3. Descrições dos Envolvidos e Usuários

3.1 Resumo dos Envolvidos

Bruno de Paula	Gerente de Projeto	Gerências demandas e distribuir entre os membros da equipe, monitora entrega e garantir prazo estabelecido no cronograma sejam cumpridas e que escopo seja seguido.

Gustavo Marques	Analista e Desenvolvedor	Desenvolver as funcionalidades core da biblioteca DecisionMakerLPA, documentação, arquitetura Implementação, teste, qualidade, manutenção e melhorias do software.
Marcelo Bueno	Designer e Desenvolvedor Unity	Responsável pelo concept art, stroryboard, game design document e estética do jogo de demonstração. Desenvolver as funcionalidades do jogo e aplicar as funcionalidades do framework Cards Game Decision.
Wesley Luiz	Desenvolvedor	Responsável pelo desenvolvimento do framework
Daniel Oliveira	Desenvolvedor Unity	Desenvolver o jogo que utilizará as funcionalidades da biblioteca DecisionMakerLPA, responsável de garantir a usabilidade, confiabilidade, desempenho e testes da biblioteca.

3.2 Resumo dos Usuários

Programadores Unity	Desenvolvedor de jogos	Responsável na parte de programação do jogo na plataforma Unity	Daniel Oliveira, Marcelo Bueno

3.3 Ambiente do Usuário

A plataforma utiliza é Unity que é um motor de jogo proprietário criado pela Unity. A Unity oferece três versões para os usuários, o Personal que é gratuito focado para iniciantes, o Plus que é pago para entusiastas e o Pro que é o mais caro focado para profissionais e estúdios. A biblioteca será um assets de biblioteca que precisa da plataforma Unity no desenvolvimento de jogos de cartas.

3.4 Resumo das Principais Necessidades dos Envolvidos ou Usuários

Aplicar tomada de decisão em jogos de cartas	Alta	Lógica de confronto de cartas	Desenvolver manualmente a lógica de tomada de decisão	Biblioteca que já tenha essas funcionalidades implementadas e exigi um mínimo de conhecimento da lógica de tomada de decisão para usa lá

3.5 Alternativas e Concorrência

Alternativas seria ter que implementar a lógica de tomada de decisão (Paraconsistente) manualmente. Concorrência seria utilizar software que aplica a lógica de maneira que especificamente focado em desenvolvimento de jogos.

4. Visão Geral do Produto

4.1 Perspectiva do Produto

DecisionMakerLPA é uma biblioteca que será disponibilizado no assets do Unity e oferece componentes na loja Unity store.

4.2 Suposições e Dependências

Não se aplica a esse produto.

5. Recursos do Produto

O produto fornece recurso de tomada de decisão para decidir quanto de porcentagem de vida todas as cartas atacando juntas conseguem tirar de vida do adversário.

6. Outros Requisitos do Produto

Não se aplica a esse produto.

APÊNDICE F – ESPECIFICAÇÃO DE CASO DE USO

BGMDW Ltda

DecisionMakerLPA
Especificação de Caso de Uso: Utilização da biblioteca

Versão 3.0

Histórico da Revisão

Data	Versão	Descrição	Autor
18/05/2019	1.0	Preenchimento do documento	Bruno de Paula
19/10/2019	2.0	Revisão de todos os capítulos	Bruno de Paula
16/11/2019	3.0	Revisão dos casos de uso	Bruno de Paula

Índice Analítico

Especificação de Caso de Uso: Utilização da biblioteca

1. Nome do Caso de Uso

Utilização da biblioteca.

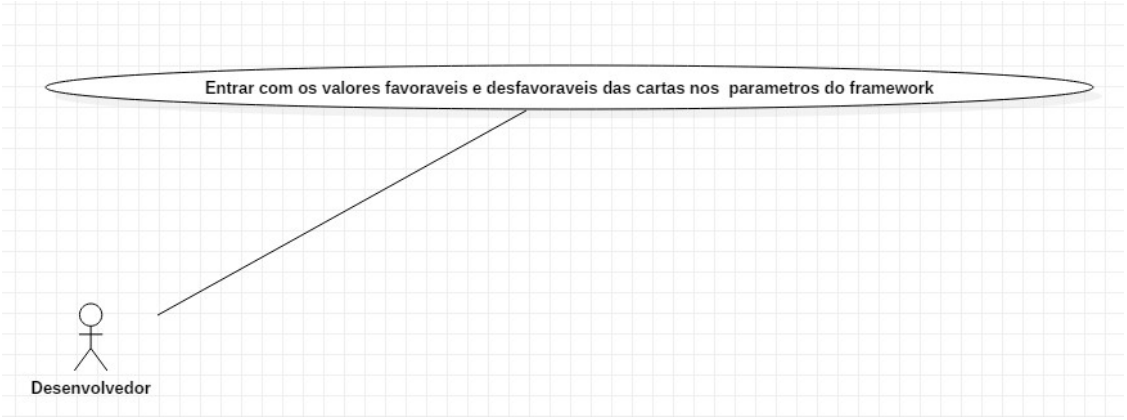
1.1 Breve Descrição

A principal utilização da biblioteca.

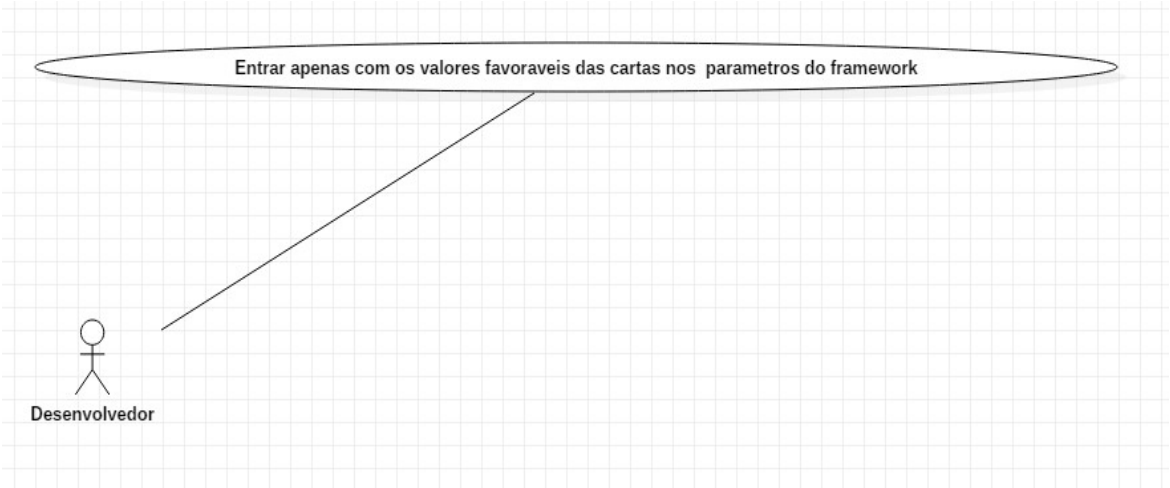
2. Fluxo de Eventos

2.1 Fluxo Básico

2.2 Fluxos Alternativos



2.2.1 < Primeiro Fluxo Alternativo >



2.2.1.1 < Um Subfluxo Alternativo >

Não se aplica.

2.2.2 *< Segundo Fluxo Alternativo >*

Não se aplica.

3. Requisitos Especiais

No final a biblioteca não passara de um asset que só poderá ser importado no Unity Asset Store.

3.1 < Primeiro Requisito Especial >

4. Precondições

Não se aplica.

4.1 < Precondição Um >

5. Pós-condições

Não se aplica.

5.1 < Pós-condição Um >

6. Pontos de Extensão

Não se aplica.

6.1 <Nome do Ponto de Extensão>

Não se aplica.

APÊNDICE G – ARQUITETURA DE SOFTWARE

BGMDW Ltda

DecisionMakerLPA
Documento de Arquitetura de Software

Versão 6.0

Histórico da Revisão

Data	Versão	Descrição	Autor
18/05/2019	1.0	Preenchimento do documento de arquitetura de software.	Marcelo Bueno
20/05/2019	2.0	Preenchimento do documento de arquitetura de software.	Marcelo Bueno
25/05/2019	3.0	Preenchimento do documento de arquitetura de software.	Bruno de Paula
31/06/2019	4.0	Revisão dos documentos	Marcelo Bueno
19/10/2019	5.0	Revisão de todos os capítulos	Bruno de Paula
16/11/2019	6.0	Revisão de todos os capítulos	Bruno de Paula

Índice Analítico

Documento de Arquitetura de Software

1. Introdução

Este documento fornece em detalhe a arquitetura utilizada para o desenvolvimento da biblioteca.

1.1 Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.

1.2 Escopo

Este documento trata da implementação arquitetura da biblioteca de tomada de decisão para jogo de cartas no Unity, com base nos diagramas UML acerca dos casos de uso.

1.3 Referências

Esse documento não faz referência a outro documento.

1.4 Visão Geral

Neste documento estará definindo a arquitetura de software.

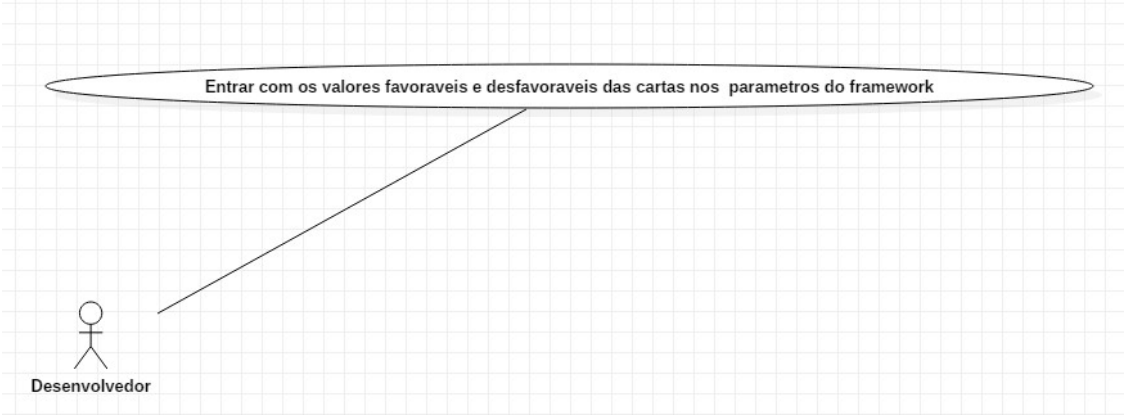
2. Representação Arquitetural

A biblioteca será desenvolvida na linguagem de programação C#.

3. Metas e Restrições da Arquitetura

A biblioteca deverá ser desenvolvendo na linguagem C#, utilizando plataforma Unity.

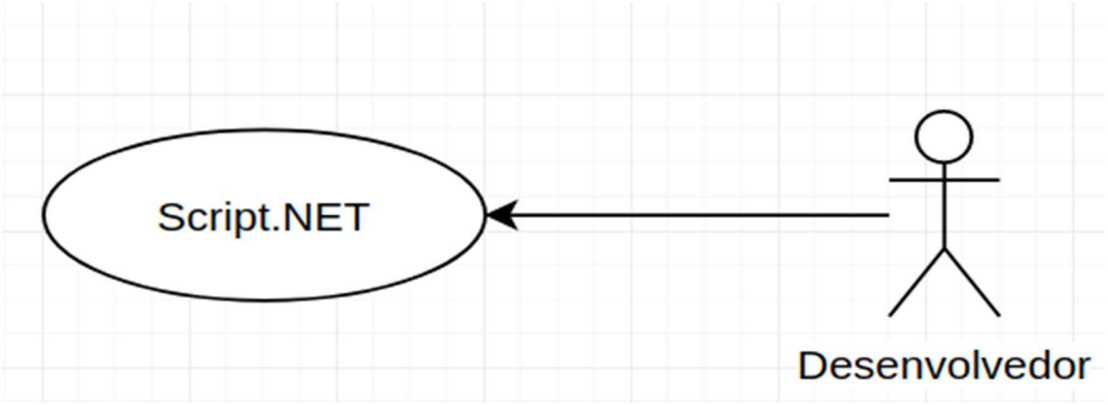
4. Visão de Casos de Uso



4.1 Realizações de Casos de Uso

O desenvolvedor irá entrar com valores favoráveis e desfavoráveis das cartas nos parâmetros da biblioteca.

5. Visão Lógica



5.1 Visão Geral

Não se aplica.

5.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura

Não se aplica.

6. Visão de Processos

O método principal receberá valores por parâmetro, e o mesma irá executar outros métodos que no final retornará qual a porcentagem de vida que todas as cartas juntas conseguem tirar das outras.

7. Visão da Implantação

Iremos publicar na lógica da Unity um conjunto de arquivos, que no caso seria um jogo e a biblioteca, que serão convertidos para assets.

8. Visão da Implementação

Estrutura de classe:

```
Public Integer ObtemPorcentagemDeDano (cartas[]){
}
```

8.1 Visão Geral

Quando o método Principal for executado após ter passado os parâmetros, será executado diversos métodos e retornado o resultado.

8.2 Camadas

Não se aplica.

9. Visão de Dados (opcional)

Não se aplica.

10. Tamanho e Desempenho

O Objetivo é que os usuários consigam utilizar do framework sem impactar no desempenho do jogo.

11. Qualidade

Todo o código será documentado facilitando a sua manutenção.

APÊNDICE H – DOCUMENTAÇÃO DA BIBLIOTECA

DecisionMakerLPA Reference

Table of Contents

DecisionMakerLPA Reference

Namespaces

<default> Namespace

Classes

Baricentro[5], Carta[7], Constante[11], EstadoLogico[15], LogicaParaconsistente[16]

Baricentro Class

Classe modelo que contém os valores favoráveis (Mi) e desfavoráveis (Lambda).

C#

```
public class Baricentro
```

Requirements

Namespace:<default>[5]

Assembly: DecisionMakerLPA (in DecisionMakerLPA.dll)

Properties

Lambda[6], Mi[6]

Methods

CalculaGrauDeCerteza[6], CalculaGrauDeIncerteza[6], Normalizar[7]

Baricentro Constructor

Método Construtor.

C#

```
public Baricentro(
   int mi,
   int lambda
)
```

Parameters

mi

Valor Favorável.

lambda

Valor Desfavorável.

See Also

Applies to: Baricentro[5]

Baricentro.Lambda Property

C#

```
public int Lambda {get; set;}
```

Property Value

Getter e Setter do valor Lambda.

See Also

Applies to: Baricentro$_5$

Baricentro.Mi Property

C#

```
public int Mi {get; set;}
```

Property Value

Getter e Setter do valor Mi.

See Also

Applies to: Baricentro$_5$

Baricentro.CalculaGrauDeCerteza Method

Calcula o grau de Certeza.

C#

```
public double CalculaGrauDeCerteza()
```

Returns

Retorna valor double no intervalo de 0 até 1.

See Also

Applies to: Baricentro$_5$

Baricentro.CalculaGrauDeIncerteza Method

Calcula o grau de incerteza.

C#

```
public double CalculaGrauDeIncerteza()
```

Returns

Retorna valor double no intervalo de 0 até 1.

See Also

Applies to: Baricentro5

Baricentro.Normalizar Method

Normalizar o valor para intervalo de 0 até 1.

C#

```
public static double Normalizar(
   int atributo
)
```

Parameters

atributo

Returns

Retorna o valor do Mi ou Lambda divido por cem.

See Also

Applies to: Baricentro5

Carta Class

Classe Modelo Carta que contém oito valores.

C#

```
public class Carta
```

Requirements

Namespace:<default>5

Assembly: DecisionMakerLPA (in DecisionMakerLPA.dll)

Properties

LambAtributo18, LambAtributo29, LambAtributo39, LambAtributo49, MiAtributo19, MiAtributo29, MiAtributo310, MiAtributo410

Methods

GetAtributoLamb10, GetAtributoMi10

Carta Constructor

Método Construtor.

C#

```
public Carta(
   int mi1,
   int lamb1,
   int mi2,
   int lamb2,
   int mi3,
   int lamb3,
   int mi4,
   int lamb4
)
```

Parameters

mi1

lamb1

mi2

lamb2

mi3

lamb3

mi4

lamb4

See Also

Applies to: Carta7

Carta.LambAtributo1 Property

C#

```
public int LambAtributo1 {get; set;}
```

See Also

Applies to: Carta7

Carta.LambAtributo2 Property

C#

```
public int LambAtributo2 {get; set;}
```

See Also

Applies to: Carta7

Carta.LambAtributo3 Property

C#

```
public int LambAtributo3 {get; set;}
```

See Also

Applies to: Carta7

Carta.LambAtributo4 Property

C#

```
public int LambAtributo4 {get; set;}
```

See Also

Applies to: Carta7

Carta.MiAtributo1 Property

C#

```
public int MiAtributo1 {get; set;}
```

See Also

Applies to: Carta7

Carta.MiAtributo2 Property

C#

```
public int MiAtributo2 {get; set;}
```

See Also

Applies to: Carta7

Carta.MiAtributo3 Property

C#

```
public int MiAtributo3 {get; set;}
```

See Also

Applies to: Carta7

Carta.MiAtributo4 Property

C#

```
public int MiAtributo4 {get; set;}
```

See Also

Applies to: Carta7

Carta.GetAtributoLamb Method

Pega o atributo de acordo com índice passado por parâmetro.

C#

```
public int GetAtributoLamb(
   int i
)
```

Parameters

i

Índice

Returns

Retorna o atributo selecionado.

See Also

Applies to: Carta7

Carta.GetAtributoMi Method

Pega o atributo de acordo com índice passado por paramêtro.

C#

```
public int GetAtributoMi(
   int i
)
```

Parameters

i

Índice

Returns

Retorna o atributo selecionado.

See Also

Applies to: Carta7

Constante Class

Classe estática que possui as constantes da biblioteca.

C#

```
public static class Constante
```

Requirements

Namespace:<default>5

Assembly: DecisionMakerLPA (in DecisionMakerLPA.dll)

Fields

FALSO11, INCONSISTENTE11, INCONSISTENTE_T_FALSO12, INCONSISTENTE_T_VERDADE12, PARACOMPLETO12, PARACOMPLETO_T_FALSO12, PARACOMPLETO_T_VERDADE12, QUASE_F_I13, QUASE_F_P13, QUASE_V_I13, QUASE_V_P13, VCFA13, VCIC14, VCPA14, VCVE14, VERDADE14

FALSO Field

See Also

Applies to: Constante11

INCONSISTENTE Field

C#

```
public const string INCONSISTENTE = @"INCONSISTENTE"
```

See Also

Applies to: Constante11

INCONSISTENTE_T_FALSO Field

C#

```
public const string INCONSISTENTE_T_FALSO = @"INCONSISTENTE_TENDENDO_A_FALSO"
```

See Also

Applies to: Constante$_{11}$

INCONSISTENTE_T_VERDADE Field

C#

```
public const string INCONSISTENTE_T_VERDADE =
@"INCONSISTENTE_TENDENDO_A_VERDADE"
```

See Also

Applies to: Constante$_{11}$

PARACOMPLETO Field

C#

```
public const string PARACOMPLETO = @"PARACOMPLETO"
```

See Also

Applies to: Constante$_{11}$

PARACOMPLETO_T_FALSO Field

C#

```
public const string PARACOMPLETO_T_FALSO = @"PARACOMPLETO_TENDENDO_A_FALSO"
```

See Also

Applies to: Constante$_{11}$

PARACOMPLETO_T_VERDADE Field

C#

```
public const string PARACOMPLETO_T_VERDADE = @"PARACOMPLETO_TENDENTO_A_VERADADE"
```

See Also

Applies to: Constante11

QUASE_F_I Field

C#

```
public const string QUASE_F_I = @"QUASE_FALSO_TEDENDO_A_INCONSISTENTE"
```

See Also

Applies to: Constante11

QUASE_F_P Field

C#

```
public const string QUASE_F_P = @"QUASE_FALSO_TENDENDO_PARACOMPLETO"
```

See Also

Applies to: Constante11

QUASE_V_I Field

C#

```
public const string QUASE_V_I = @"QUASE_VERDADE_TENDENDO_A_INCONSISTENTE"
```

See Also

Applies to: Constante11

QUASE_V_P Field

C#

```
public const string QUASE_V_P = @"QUASE_VERDADE_TENDENDO_A_PARACOMPLETO"
```

See Also

Applies to: Constante11

VCFA Field

C#

```
public const double VCFA = -0,5
```

See Also

Applies to: Constante11

VCIC Field

C#

```
public const double VCIC = 0,5
```

See Also

Applies to: Constante11

VCPA Field

C#

```
public const double VCPA = -0,5
```

See Also

Applies to: Constante11

VCVE Field

C#

```
public const double VCVE = 0,9
```

See Also

Applies to: Constante11

VERDADE Field

C#

```
public const string VERDADE = @"VERDADE"
```

See Also

Applies to: Constante11

EstadoLogico Class

Classe estática que realizar a verificação do estado Lógico no reticulado.

C#

```
public static class EstadoLogico
```

Requirements

Namespace:<default>[5]

Assembly: DecisionMakerLPA (in DecisionMakerLPA.dll)

Methods

DescobreEstadoLogico[15], TransformaEstadoLogicoEmPorcentagem[15]

EstadoLogico.DescobreEstadoLogico Method

Confere o estado lógico pelo grau de certeza e incerteza.

C#

```
public static string DescobreEstadoLogico(
   double gc,
   double gi
)
```

Parameters

gc

Grau de Certeza

gi

Grau de Incerteza

Returns

Retorna o estado lógico.

See Also

Applies to: EstadoLogico[15]

EstadoLogico.TransformaEstadoLogicoEmPorcentagem Method

Transforma a estado lógico em porcentagem.

C#

```
public static int TransformaEstadoLogicoEmPorcentagem(
   string estadoLogico
)
```

Parameters

estadoLogico

Um dos doze estados lógicos da paraconsistente.

Returns

Porcentagem em Danos.

See Also

Applies to: EstadoLogico[15]

LogicaParaconsistente Class

Classe Responsável por implementar a Lógica Paraconsistente.

C#

```
public class LogicaParaconsistente
```

Requirements

Namespace:<default>[5]

Assembly: DecisionMakerLPA (in DecisionMakerLPA.dll)

Methods

ObtemBaricentro[16], ObtemPorcentagemDeDano[17]

LogicaParaconsistente.ObtemBaricentro Method

Adiquire o Baricentro através da lista de cartas.

C#

```
public Baricentro ObtemBaricentro(
   List<Carta> cartas
)
```

Parameters

cartas

Índice do atributo da carta.

Returns

Retorna a classe Baricentro.

See Also

Applies to: LogicaParaconsistente[16]

LogicaParaconsistente.ObtemPorcentagemDeDano Method

Recebe uma lista "Carta" com seus valores favoráveis e desfavoráveis, aplicar a Lógica Paraconsistente.

C#

```
public int ObtemPorcentagemDeDano(
   List<Carta> cartas
)
```

Parameters

cartas

Índice do atributo da carta.

Returns

Retorna porcentagem de Dano.

See Also

Applies to: LogicaParaconsistente[16]

Index

APÊNDICE I – DOCUMENTAÇÃO DO GAME DESIGN

Fighting Against Monsters

Game Design Document

Versão: 2.0

Autores:

Daniel Sousa David de Oliveira
Marcelo Bueno Silva
Bruno de Paula Silva
Gustavo Felipe Marques
Wesley luis

São Paulo,
Dezembro de 2019

Índice

1 História

Em uma época muito distante, grandes heróis se juntaram em middletown na grande taverna dragão para tramar uma batalha épica, que para muitos é chamada de **Fighting Against Monsters.** Dizem em muitos lugares, que o indivíduo que é derrotado em **Nome do jogo**, terá de lhe entregar a alma ao campeão, e será devoto a ele eternamente, e hoje finalmente acontecerá o torneio do grande **Fighting Against Monsters** na taverna do rei de middletown, aquele que vencer se tornará imortal e receberá todos os prazeres da vida.

2 Gameplay

Basicamente irá ter 8 cartas na mão e o jogador poderá escolher 4 cartas entre as 8 e todas as 4 cartas atacam juntas e tiram uma porcentagem de vida do adversário.

O jogador precisa pensar qual carta jogar em campo, e se vale a pena usar ele naquele momento.

A cada duelo ganho, o jogador irá ganhar cartas novas, e usar em seu deck nos próximos duelos.

Para ganhar o jogo, o jogador precisa ganhar todos duelos zerando a vida dos adversários, e matando os chefes das fases, caso não mate os chefes perdi a fase atual e volta para a inicial.

3 Personagens

Foram criados inúmeras cartas todas parecidas com personagens conhecidos porem com uma certa mudança nas características e o nome alterado por conta de direitos autorais.

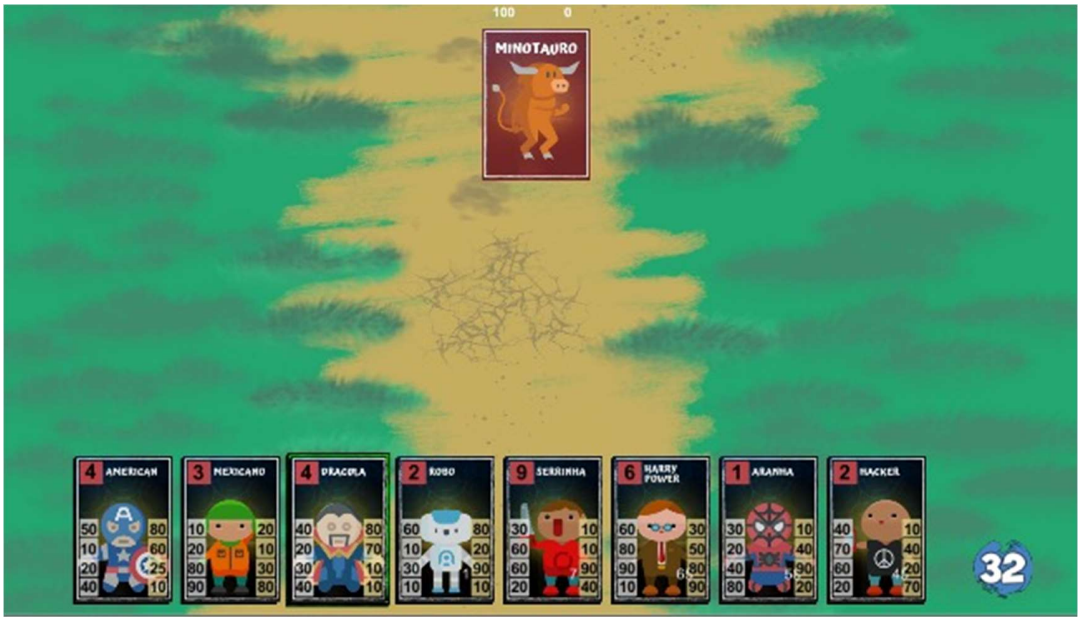
4 Controles

Para controlar as ações que o personagem irá executar no duelo, ele precisa pressionar os botões que irá estar no visor.


PAUSE
MINOTAURO
100
ATTACK!
12

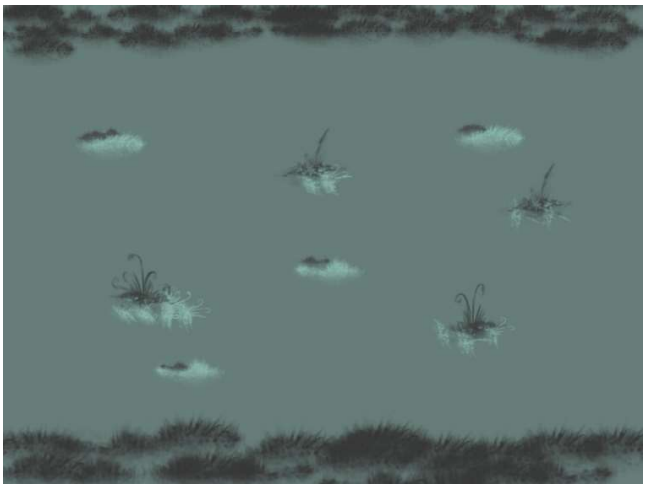
5 Câmera

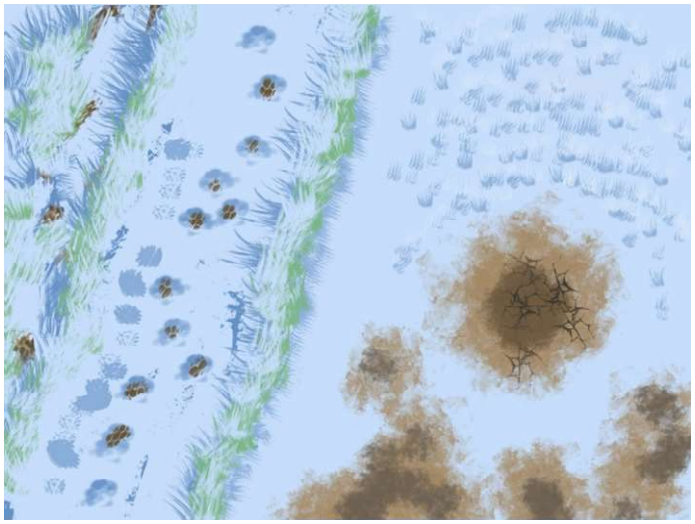
Será visualizado as cartas em mãos, e o campo de duelo, e podemos visualizar as cartas do adversário.

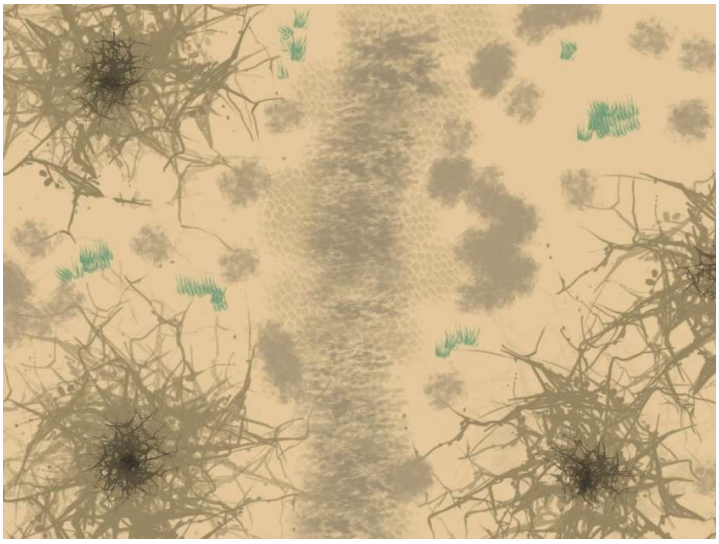


6 Universo do Jogo

Todos os cenários do jogo foram desenhados pela equipe, cada um com um tema diferente possibilitando a implementação de dinâmicas diferentes no jogo ou vantagens ou desvantagens nas cartas.







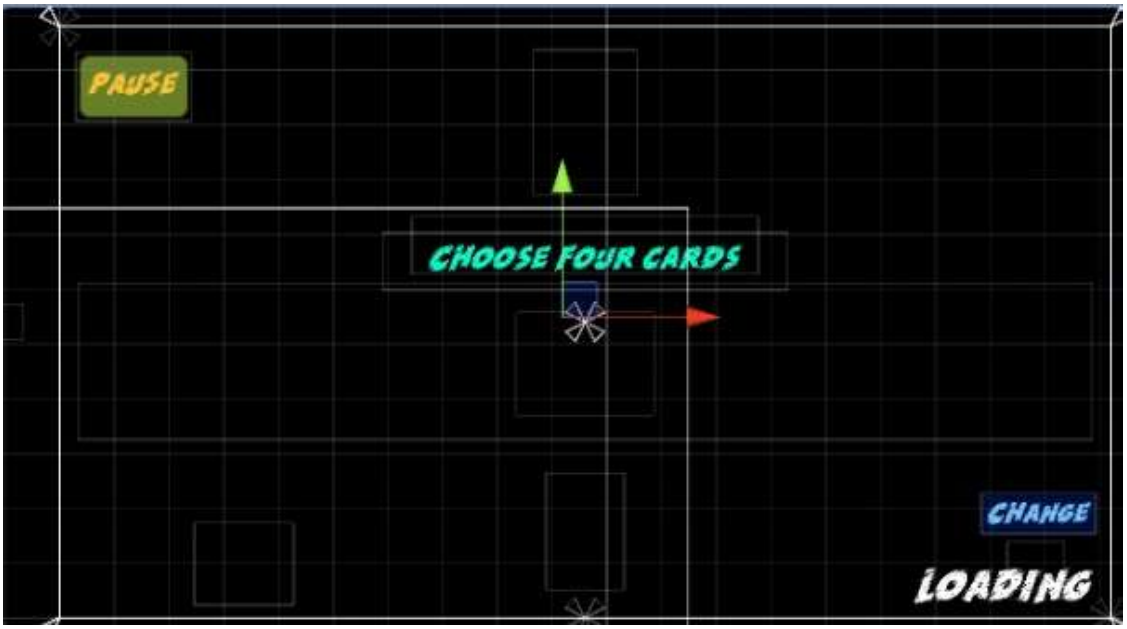
7 Inimigos

Os inimigos, são monstros, são humanos, são entidades por ser qualquer coisa.

O jogador está em um campeonato de cartas, e para ganhar precisa duelar com oponentes.

8 Interface

Todas as interfaces foram desenhadas pela própria equipe.



9 Cutscenes

O jogo não terá cutscenes.