UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

PROPOSTAS PARA O BALANCEAMENTO DINÂMICO DE JOGOS COM AJUSTE DE DIFICULDADE E PAYOFF LENTOS

TRABALHO DE GRADUAÇÃO

Autor: João Gabriel Gadelha Xavier Monteiro (jggxm@cin.ufpe.br)
Orientador: Geber Lisboa Ramalho (glr@cin.ufpe.br)

Recife, novembro de 2009.

UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

Propostas para o balanceamento dinâmico de jogos com ajuste de dificuldade e payoff lentos

Por

João Gabriel Gadelha Xavier Monteiro

Monografia apresentada ao Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Geber Lisboa Ramalho

Recife, novembro de 2009.

Resumo

O presente trabalho apresenta estudos que justificam a importância de um bom balanceamento e os requisitos existentes na literatura para a sua realização. Em seguida propõe duas técnicas para resolver o problema do balanceamento dinâmico de jogos em que não é possível realizar ações de ajuste de dificuldade a qualquer momento e que geralmente a dificuldade sentida pelo jogador para realizar um determinado desafio só pode ser observada em momentos muito específicos como *checkpoints* ou quando o ultimo desafio proposto é encerrado. Como estudo de caso as técnicas são aplicadas em um jogo com as características citadas. No final do trabalho são exibidos e interpretados os resultados da pesquisa realizada para a validação das técnicas.

Palavras chave: Balanceamento Dinâmico, Jogos, Adaptação ao usuário, Entretenimento Digital.

Abstract

This work presents studies that justify the importance of a good balance and requirements in the literature for achieve it. Then proposes two techniques to solve the problem of dynamic balancing of games that you can not undertake actions that adjust the difficulty at any time and generally the difficulty felt by the player to perform a particular challenge can only be observed at very specific times like checkpoints or when the last challenge proposed is finished. As a case study techniques are applied in a game with these characteristics. At the end of the work the results of research conducted for the validation of techniques are displayed and interpreted.

Keywords: Dynamic Game Balancing, Games, Adaptation to User, Digital Entertainment.

Agradecimentos

Agradeço muito a minha mãe Patrícia, meu pai Fernando e meus irmãos Rodrigo e Amanda por todo o apoio, carinho e paciência durante todo o curso e não só por este trabalho. Agradeço também a minha amada namorada Alice por toda sua compreensão, carinho e apoio nesses mais de três anos e meio juntos. Sem vocês talvez não tivesse tido forças para chegar até o fim.

Agradeço a todos que colaboraram realizando os testes do Pizzaiolo e ao pessoal da Manifesto por ter cedido gentilmente o jogo para o trabalho, em especial a Túlio Caraciolo e Felipe Maia pelo apoio técnico. Agradeço ao professor Geber Ramalho por sua paciência e dedicação na orientação do trabalho e a Gustavo Danzi pelo apoio técnico.

Sumário

Lista de Figuras

Lista de Gráficos

Lista de Algoritmos

1. Introdução

Jogos são softwares interativos que influem e são influenciados pelo usuário, diferentemente de outras formas de entretenimento como filmes ou música nos quais a influência é unidirecional, onde o fluxo não pode ser alterado. Na condição de softwares interativos, os jogos devem apresentar boa usabilidade, isso significa que eles devem ser capazes de atender as expectativas dos jogadores com os quais interagem [Crosby, 1979].

A principal expectativa de um jogador ao interagir com um jogo é que este seja capaz de entretê-lo. A capacidade de entretenimento influencia diretamente na qualidade do jogo [Tozour, 2002] e segundo [Andrade, 2006] pode ser definida por uma série de fatores, como enredo atraente, qualidade gráfica ou sonora e jogabilidade.

A jogabilidade pode ser descrita como a relação jogo/jogador [Lorenzon, 2008] e abrange o conjunto de elementos que tornam o jogo satisfatório do ponto de vista da usabilidade. Um dos principais elementos da jogabilidade é o nível de desafio enfrentado pelo usuário, que é definido diretamente pelo balanceamento do jogo. Na literatura vários autores enaltecem a importância de um bom balanceamento: Segundo [Sweetser e Wyeth, 2005] o nível de desafios influência na satisfação do jogador, para [Yannakais e Hallam, 2005] o balanceamento é uma das principais métricas do interesse do jogador no software e [Falstein, 2004] afirma que o balanceamento do jogo é considerado pela comunidade desenvolvedora como algo determinante para o seu sucesso.

A atividade de balanceamento dos jogos tradicionalmente é realizada de forma manual pelos *Game Designers* em um processo árduo e tedioso, envolvendo repetidas sessões de testes e ajustes, além de testes com grupos de usuários, consumindo cerca de 20% dos recursos destinados a produção do software. As técnicas tradicionais se baseiam em estudar grupos de jogadores e definir alguns níveis de dificuldade específicos [Andrade, 2006]. O resultado pode tornar-se muito difícil para jogadores casuais ou muito simples para jogadores habituais (*hardcore*). É muito complexo balancear um jogo que possa agradar todos os prováveis jogadores, porque mesmo com habilidades parecidas, jogadores possuem características pessoais diferentes e evoluem em ritmos distintos.

Diante das dificuldades para se fazer o balanceamento de um jogo e devido a sua enorme importância, faz-se necessário a pesquisa, o desenvolvimento e o aprimoramento de técnicas que possam tornar mais simples a execução dessa atividade e principalmente que façam com que o jogo sempre seja capaz de se adaptar ao nível do jogador, para que ele sempre receba desafios e um nível adequado fazendo com que sua vontade de interagir com o jogo permaneça. Essa adaptação contínua da dificuldade do jogo para o nível do jogador é chamada de balanceamento dinâmico [Andrade, 2006] ou ajuste dinâmico de dificuldade [Hunicke e Chapman, 2004].

Na próxima seção serão mostrados os principais problemas relacionados ao balanceamento dinâmico que serão atacados neste trabalho e na seção seguinte será apresentada a estrutura do documento.

1.1 Problemas a serem atacados

Na literatura sobre o assunto já existem inúmeras propostas para a realização do balanceamento dinâmico. Porém todas elas trabalham considerando que existe sempre a possibilidade de realizar ações de ajuste de dificuldade de forma muito rápida e a qualquer momento. Isso porque todos os jogos utilizados como estudo de caso apresentados para a validação das propostas possuem agentes autônomos chamados de NPCs (que podem possuir uma representação visual no jogo ou não) e essas técnicas se baseiam fundamentalmente em alterar o comportamento desses agentes ou seus atributos (como força ou velocidade). Para o ajuste de dificuldade nesses casos o agente pode ter sua habilidade aumentada ou diminuída, além de ter seus atributos alterados para que ele se adapte ao nível de habilidade do adversário e com isso o jogo se torne equilibrado. Essa possibilidade justifica a suposição de que o ajuste pode (e deve) ser muito rápido e pode ser realizado a qualquer momento, pois, desde que as ações de balanceamento sejam realizadas de uma forma moderada e inteligente, dificilmente o usuário irá perceber qualquer alteração e com isso sua experiência com o jogo não será prejudicada.

Além dessa característica, os jogos apresentados como estudo de caso sempre apresentam bons indicadores do desempenho atual do jogador no jogo (*pay-off)*. Isso é fundamental para a possibilidade do balanceamento rápido.

Existem, porém, jogos em que essa rápida execução de ações de balanceamento não é possível, pois só podem ser realizadas depois que o desafio apresentado por último é concluído ou algum outro evento específico acontece. Isso ocorre porque caso haja alguma modificação do comportamento do jogo durante o desafio, ela poderá ser percebida pelo usuário**.** Um exemplo desse caso pode ser um jogo em que o jogador deve encontrar a saída de um labirinto. Mesmo que a Inteligência Artificial perceba que ele sente dificuldades de encontrar a saída, não pode realizar nenhuma alteração no desafio, pois como o labirinto não é um agente, não existe comportamento e nem atributos para serem alterado e sua estrutura não pode ser modificada já que certamente isso seria perceptível para o jogador. O que é possível fazer é escolher um labirinto mais simples da próxima vez que o usuário for jogar, baseando-se em alguma grandeza, como pontos conquistados, para medir o seu desempenho no desafio anterior, que mesmo podendo ser obtida durante o desafio só terá utilidade para o balanceamento quando este for encerrado. Existem também casos de jogos em que o desempenho do jogador só pode ser medido de forma precisa no fim de um desafio ou em um *checkpoint.*

Esse exemplo citado acima é um caso diferente de um jogo de luta em que é possível fazer o lutador controlado pelo computador jogar de maneira mais ou menos habilidosa contra o jogador durante uma mesma luta, ou em um jogo do tipo FPS fazer um atirador errar ou acertar mais tiros, numa mesma partida. Em ambos os casos, se a técnica for bem aplicada, não haverá problemas em relação à experiência do jogador.

O objetivo desta pesquisa, diferentemente do que existe na literatura, é realizar o balanceamento dinâmico de jogos com essas características de lentidão no ajuste de dificuldade e que em geral só apresentam bons indicadores do desempenho do usuário no ultimo desafio quando este é concluído. Para isso são propostas duas técnicas.

1.2 Estrutura do documento

No segundo capitulo deste trabalho as técnicas tradicionais de balanceamento de jogos e os requisitos existentes na literatura para a realização do balanceamento dinâmico serão estudados. No terceiro capitulo haverá o estudo das

principais técnicas propostas na literatura sobre o assunto. No quarto capítulo, duas novas técnicas são propostas para a realização do balanceamento dinâmico nos jogos que possuem as características citadas previamente. No quinto capitulo será apresentado o jogo utilizado como estudo de caso e como as técnicas propostas foram aplicadas nele. No sexto capítulo serão apresentados os resultados obtidos nas pesquisas feitas com os usuários que jogaram o jogo usado como estudo de caso, o objetivo de realizar a pesquisa foi avaliar as técnicas propostas e definir qual das duas demonstra mais sucesso em realizar o balanceamento dinâmico. Finalmente o sétimo capítulo apresentará as conclusões gerais do trabalho e os trabalhos futuros.

2. Balanceamento de Jogos

O balanceamento de jogos pode ser definido como o processo que visa garantir um bom nível de desafios tentando evitar extremos tais como frustrar o jogador nos casos em que a dificuldade do jogo está muito elevada ou entediá-lo nos casos em que o jogo só apresenta desafios simples [Koster, 2004]. O objetivo final de balancear é garantir que o sucesso do jogador no jogo dependa somente das suas habilidades e o foco deve ser transmitir a experiência desejada.

De acordo com [Andrade, 2006] balancear não significa somente atuar em um nível mediano de desafio, é preciso considerar a evolução do usuário ao longo do tempo, ou seja, considerar o seu incremento de habilidades e adaptar os desafios para o seu novo nível. Ainda é preciso considerar que as habilidades dos jogadores podem regredir no decorrer do tempo, em virtude, por exemplo, de um longo período sem jogar. Um balanceamento eficiente deve ser capaz de acompanhar uma regressão de nível do jogador.

O inicio do jogo, que é onde se apresentam as maiores variações de habilidades, é crítico para o problema, pois, se o jogador considerar os desafios simples demais ou difíceis demais, simplesmente desistirá de jogar e dificilmente retornará. No decorrer do jogo o problema de balanceamento se atenua, pois passa a ser possível considerar níveis de habilidade mais próximos. Porém em jogos curtos e que possuem uma pequena curva de aprendizagem como os jogos casuais [Mendonça, 2009] o cuidado com o crescimento do nível de dificuldade deve ser ainda maior, principalmente porque geralmente quem busca esse tipo de jogo não possui o hábito de jogar jogos mais longos e que exijam grande habilidade.

O *game designer* é o profissional responsável pela elaboração dos desafios do jogo e seu balanceamento, além de possuir outras tarefas como definir a história, estudar o público alvo do jogo e escrever o documento de *game design,* que é o documento onde estarão escritos todos os requisitos para a construção do jogo.

. Para realizar o balanceamento não existem formalismos ou métodos científicos, apesar de ser possível usar ferramentas científicas para auxiliar no trabalho [Rollings e Adams, 2003]. O processo pode ser extremamente difícil. Além disso, a teoria dos jogos clássica não serve para jogos mais modernos e complexos, pois ela está direcionada para jogos discretos e em descobrir caminhos ótimos

[Rollings e Adams, 2003]. De modo geral as técnicas se resumem a sucessivas repetições de testes (feitos pela própria equipe de desenvolvimento ou por jogadores membros do público alvo) e ajustes que devem ser feitos pelos *game designers* [Tonietto, 2007], essa abordagem é bastante suscetível a erros, pois quanto mais complexo é o jogo, mais complexo é balancear. Um agravante ainda maior para a dificuldade do problema é que técnicas de balanceamento usadas por grandes desenvolvedoras, muitas vezes são mantidas em segredo, por isso ainda hoje a literatura disponível sobre o assunto é relativamente pequena [Cadwel, 2009].

Nas próximas seções serão analisadas as duas grandes classes de balanceamento, o estático e o dinâmico. Dentro da seção de balanceamento dinâmico, serão vistos requisitos relevantes a serem considerados nas técnicas propostas.

2.1 Balanceamento estático

O balanceamento estático é o mais amplamente utilizado na indústria de desenvolvimento jogos [Andrade, 2006]. Ele é também conhecido como o balanceamento clássico. Suas técnicas se concentram em definir as regras do jogo (como por exemplo, a força do tiro de um tanque ou a distância do pulo do Super Mario) e suas interações de uma maneira equilibrada, considerando que elas são invariantes no tempo e separando o jogo em começo, meio e fim [Rollings e Adams, 2003]. Através desse balanceamento pode-se certificar se o jogo é justo, o que quer dizer que não há estratégias claramente dominantes ou inúteis e nem existe complexidade desnecessária adicionada nas atividades do jogo.

Uma das técnicas mais tradicionais aplicadas no balanceamento estático é definir níveis de dificuldade padrão, que geralmente são fácil, médio (normal) e difícil, para permitir que o usuário escolha aquele que ele julga mais se adequar as suas habilidades, como mostra a figura 2.1.

Figura 2.1: Tela de escolha do nível de dificuldade do jogo Spore (Maxis/EA Games)

À medida que o jogador progride no jogo, o nível de dificuldade dos desafios tende a aumentar, preferencialmente de maneira não linear, considerando que as habilidades do jogador também aumentam [Tonietto, 2007]. Geralmente são usadas fases (estágios) para demarcar o crescimento da dificuldade. Elas atuam como barreiras, para garantir um nível mínimo de habilidade antes de se aplicar um determinado desafio e servem para evitar que o jogo termine sem exigir grandes esforços do jogador [Andrade , 2006]. A figura 2.2 ilustra esse crescimento de dificuldade, dividido em fases.

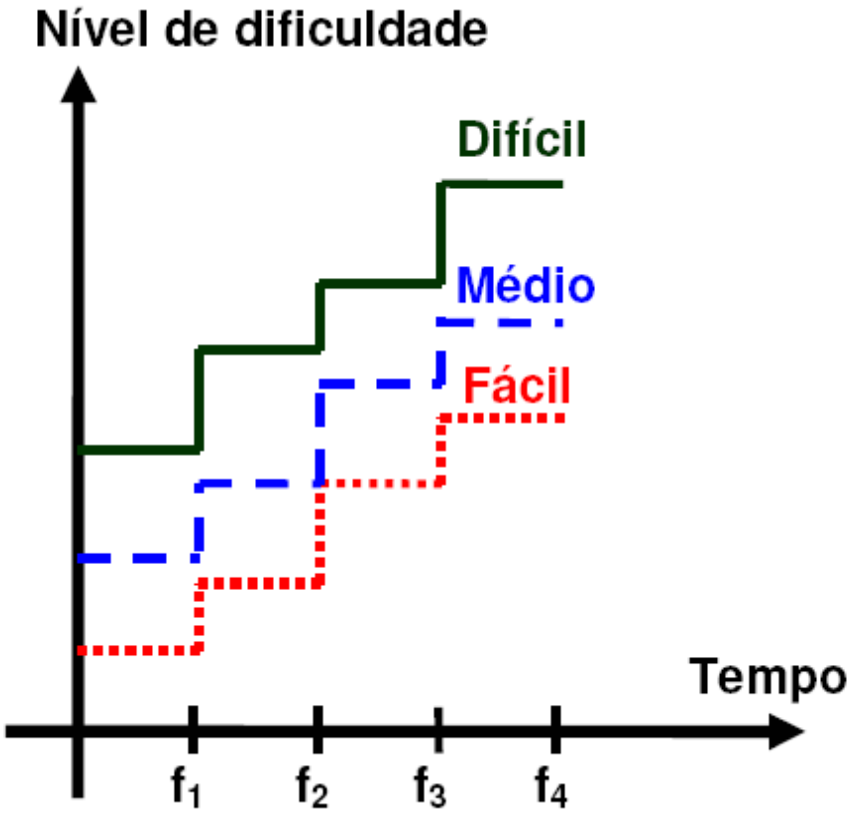


Figura 2.2: Crescimento do nível de dificuldade do jogo dividido em estágios [Andrade, 2006]

Para definir que tipos de desafios serão apresentados em cada um dos níveis de dificuldade, são definidos jogadores modelo, baseados no estudo do público alvo do jogo [Rollings e Adams, 2003]. Um grande problema dessa abordagem é que dificilmente todos os possíveis níveis de habilidade de jogadores em potencial serão abordados, pois cada pessoa evolui em um ritmo diferente, mesmo que o número de opções de escolha de níveis de dificuldade seja maior que os tradicionais fácil, difícil e médio. Isso pode fazer com que alguns jogadores desistam de jogar, pois não é possível considerar as características diferentes de cada jogador com essa abordagem de balanceamento [Andrade, 2006].

Como se vê, o balanceamento estático, apesar de ser necessário para a construção do jogo, geralmente não é suficiente para garantir equilíbrio de desafios durante todo ele. Sendo apenas bom para definir um equilíbrio inicial [Rollings e Adams, 2003], pois suas estratégias são fixas e se mantêm imutáveis ao longo do tempo.

2.2 Balanceamento dinâmico

O balanceamento dinâmico não divide o jogo nas etapas discretas, começo, meio e fim, ele considera o ciclo de vida do jogo como um todo. Os principais objetivos de realizá-lo são fazer com que o jogo mantenha-se sempre equilibrado, evitando trivialidades ou desafios intransponíveis, se mostre sempre justo, nunca colocando o jogador em clara desvantagem ou vantagem em alguma situação, e que a experiência do jogador seja sempre incrementada, nunca permitindo que o jogador se sinta perdido ou sem esperança de terminar seus desafios [Tonietto, 2007]. Para conseguir isso, as estratégias aplicadas para o balanceamento e suas heurísticas podem ser modificadas dinamicamente ao longo da vida do jogo.

Técnicas de inteligência artificial (IA) podem ser utilizadas durante o jogo para que o balanceamento dinâmico ocorra e é nelas que o estudo será focado na próxima subseção. Essas técnicas periodicamente devem avaliar o nível de dificuldade sentido pelo jogador e promover os ajustes necessários no jogo, para garantir um nível de desafio adequado [Andrade, 2006]. O uso da IA permite ainda, adicionar o fator de imprevisibilidade ao jogo, alternando momentos mais fáceis e mais difíceis, o que segundo [Falstein, 2004] é algo que agrada o jogador e aumenta o seu interesse pelo jogo.

A figura 2.3 ilustra os conceitos apresentados. Ela mostra que o jogo idealmente deve manter-se em uma região balanceada, em que a habilidade do jogador e a dificuldade apresentada se equilibram, podendo apresentar um incremento de desafios não linear, para adicionar o fator de imprevisibilidade.

Figura 2.3: Crescimento do nível de dificuldade de forma balanceada [Andrade, 2006]

2.2.1 Considerações para a implementação do balanceamento dinâmico

Todas as técnicas existentes na literatura possuem em comum o uso de uma função para medir o nível de dificuldade enfrentado pelo usuário a cada intervalo de tempo ou evento determinado por uma heurística. Essa função é chamada de função de facilidade [Demasi e Cruz, 2003] ou função desafio [Andrade et al, 2005]. Cada jogo específico deve definir como é a sua e levar em consideração o que é relevante para medir a dificuldade. Por exemplo, em um jogo de futebol a posse de bola e o placar podem ser bons indicadores do nível de dificuldade sentido pelo usuário e certamente podem ser usados para formular a função.

A função de facilidade deve ser rápida e eficiente computacionalmente, além de ter que ser confiável. Pode ser necessário sacrificar um pouco da confiabilidade em nome da velocidade, dependendo do domínio da aplicação, pois essa função, não deve de maneira nenhuma atrapalhar o desempenho da execução do jogo [Demasi e Cruz, 2003].

Após determinar o nível de dificuldade, é preciso definir que estratégias utilizar para realizar o balanceamento. Para essa questão, como já afirmado,

existem várias técnicas. Definir o que utilizar e quando, depende totalmente do domínio do jogo. Embora muitas técnicas sejam consideradas genéricas, podem surgir jogos em que elas não se aplicam como já citado na introdução.

Assim como para a função de facilidade são definidos alguns critérios (requisitos) para definir boas estratégias para o balanceamento dinâmico automático. [Andrade, 2006] propõe requisitos divididos no domínio de jogabilidade, que são aqueles que influenciam a percepção do jogador com relação ao jogo e no domínio da implementação, que são aqueles que influenciam nas características técnicas do jogo. Os requisitos citados no domínio da jogabilidade são:

- **Rápida adaptação inicial**: Significa que o jogo deve se adaptar rapidamente desde o inicio ao nível do jogador.
- **Rápida adaptação continua**: Durante a sua execução o jogo deve ser adaptar rapidamente a qualquer mudança de nível do jogador.
- **Inércia:** O jogo não deve apresentar mudanças bruscas de comportamento, nem exibir comportamentos estranhos.
- **Racionalidade:** O jogo não deve trapacear.

E no domínio da implementação são:

- **Desenvolvimento:** O esforço para a construção da técnica de balanceamento deve ser simples.
- **Execução:** Os algoritmos de balanceamento devem consumir poucos recursos de processamento e memória.

[Spronck et al, 2005] Define quatro requisitos computacionais, que são aqueles que os autores consideram obrigatórios para qualquer algoritmo de adaptação e quatro funcionais, que são considerados desejáveis. Os quatro requisitos computacionais são:

- **Velocidade:** Similar aos de rápida adaptação propostos por [Andrade, 2006]
- **Eficácia:** Os algoritmos devem sempre resultar em um comportamento razoavelmente bem sucedido.

- **Robustez:** Os algoritmos devem ser robustos com relação à aleatoriedade de muitos jogos.
- **Eficiência:** Os algoritmos devem aproveitar as oportunidades de aprender.

Os requisitos funcionais são:

- **Clareza:** Devem ser produzidos resultados fáceis de interpretar.
- **Variedade:** Devem ser produzidos vários comportamentos distintos
- **Consistência:** Os algoritmos de aprendizagem devem ser independentes do comportamento humano e do jogo.
- **Escalabilidade:** A Inteligência Artificial deve ser capaz de medir corretamente o nível do jogador.

Como se vê, a definição dos requisitos é algo subjetivo que varia entre autores, apesar de se tratar do mesmo problema. Na técnicas que serão propostas, esses requisitos apresentados serão, pelo menos parcialmente, cumpridos, outros como será mostrado terão de ser relaxados, devido ao domínio do problema que será atacado. No capitulo 4 serão apresentadas as justificativas dessa decisão.

3. Revisão da literatura sobre balanceamento dinâmico

Neste capítulo serão analisadas técnicas encontradas na literatura que propõem a resolução do problema de balanceamento dinâmico automático de jogos. Na ultima seção é apresentada uma justificativa de porque cada técnica não é muito eficiente para balancear dinamicamente os jogos que pertencem ao domínio estudado.

3.1 Balanceamento dinâmico baseado em manipulação de parâmetros

[Hunicke e Chapman, 2004] desenvolveram um sistema, chamado *Hamlet*, cujo objetivo é ajustar dinamicamente a dificuldade de um jogo. Para isso, utilizam uma abordagem baseada em economia, chamando de parâmetros de oferta àquilo que o jogador dispõe para intervir no ambiente, ou seja, seus atributos, como velocidade ou pontos de vida, e seus itens, como suas armas e quantidade de munição. Existe ainda o conceito de parâmetros de demanda, que são os desafios que o ambiente impõe, como por exemplo, a quantidade de inimigos em uma sala ou a força do ataque de um tipo de inimigo. A idéia é baseada no que propõe [Simpson, 2001]. Segundo ele jogos são planejados para manipular as trocas de recursos entre o jogador e o mundo. Como estudo de caso os jogadores utilizam um jogo de tiro em primeira pessoa FPS

O inventário é o local onde é possível ver os atributos do personagem, além de seus recursos (itens) disponíveis. Ele é utilizado para medir a defasagem entre a oferta e a demanda. A medição é feita periodicamente através de métodos estatísticos que detectam quando o jogador provavelmente irá falhar, e serve para descobrir a necessidade de ajuste do jogo para se adaptar ao nível de habilidade do usuário. Os parâmetros do jogo são então manipulados, de acordo com o nível de dificuldade desejado, para que o jogador consiga vencer mais facilmente ou sinta mais dificuldade. Por exemplo, para que um FPS se torne mais fácil é possível que o sistema faça com que as armas do jogador se tornem mais fortes (manipulação dos parâmetros da oferta) e os inimigos tenham mais chances de errar os tiros (manipulação dos parâmetros da demanda).

Um cuidado especial com essas manipulações deve ser tomado, pois as modificações exageradas nos parâmetros podem fazer com que o jogo não seja mais crível para os jogadores.

3.2 Balanceamentos dinâmico baseado em algoritmos genéticos coevolucionários

[Demasi e Cruz, 2002a] utilizam a abordagem baseada em algoritmos genéticos co-evolucionários, codificando características de comportamento como genes dos agentes aprendizes (CEAs). Nos CEAs, populações distintas devem evoluir simultaneamente, tanto para cooperarem como pra competirem [Demasi e Cruz, 2002b]. Como em qualquer abordagem utilizando algoritmos genéticos novos agentes são gerados a partir de cruzamentos (mistura de codificação dos genes) dos agentes considerados mais aptos ou através de mutações nos genes de agentes individuais. A heurística para medir a aptidão de um agente depende do domínio do problema. Como estudo de caso os autores utilizam um jogo de aventura.

Para a implementação do balanceamento dinâmico, os agentes evoluem ou mantém o nível atual quando são destruídos, dependendo do resultado da função de facilidade. Quatro abordagens são propostas para a evolução, visando adaptação ao nível do usuário.

Na primeira agentes modelo (agentes que tem em seus genes a descrição do comportamento alvo dos agentes aprendizes) são definidos manualmente. Deve haver um agente modelo para cada nível de dificuldade desejado. Os agentes aprendizes podem evoluir através da técnica *Hamming Distance* (comparação entre os genes dos agentes aprendizes e modelo e substituição para descobrir diferenças [Teknomo, 2006]), tendo alguns genes substituídos para que sua codificação genética se aproxime da do modelo. Outra forma de evolução é através de cruzamentos dos agentes mais aptos entre si e com os agentes modelos. Os agentes mais aptos no caso desse problema são os que mais se aproximam do nível do usuário, não sendo necessariamente os mais difíceis e sim o que oferecem mais desafios equilibrados.

A segunda abordagem é bastante similar a primeira, a grande diferença está na criação dos agentes modelo. Ao invés de serem criados agentes manualmente, usando heurísticas do jogo, os modelos de vários níveis são criados através de treinamentos *offline,* que são treinamentos feitos previamente aos testes de validação do sistema, cujo objetivo é fazer com que ele tenha um bom comportamento inicial. Tais treinamentos podem ocorrer tanto contra jogadores humanos como contra outros agentes.

A terceira abordagem dispensa a criação de agentes modelo. A definição dos agentes mais aptos é feita em tempo de execução (*online*) e esses agentes são armazenados em um conjunto. Quando um agente morre, um novo agente é criado a partir do cruzamento entre um agente do conjunto e o agente morto. Se o agente criado se mostrar mais apto que um dos agentes do conjunto, esse agente substituirá o agente do conjunto com menos aptidão.

A quarta abordagem consiste em aplicar a terceira abordagem após a primeira ou a segunda. A intenção é acelerar a velocidade de adaptação ao usuário fazendo com que já haja algum conhecimento prévio dos agentes mais aptos.

3.3 Balanceamento dinâmico baseado em rtNEAT

A abordagem de [Olesen, Yannakakis e Hallam, 2008] é baseada no uso da técnica *Neuro-Evolution of Augmenting Topologies* (NEAT) desenvolvida por [Stanley e Miikkulainen, 2002] que consiste em evoluir redes neurais e através de algoritmos genéticos, gerando novas redes. Com essa técnica é possível modificar tanto pesos das conexões quanto a topologia da rede. No trabalho proposto, são usados agentes cujo comportamento é governado por redes neurais que evoluem através da NEAT. O uso das letras *rt* indica quando a técnica é aplicada em tempo real.

O balanceamento dinâmico é dividido em etapas. A primeira etapa consiste em identificar os fatores do jogo que influenciam no nível de desafio sentido pelo jogador. Em seguida a técnica NEAT é utilizada para treinar *offline* os agentes que se destacam em relação aos fatores descobertos na primeira etapa e o comportamento gerado é testado para ver se é adequado contra oponentes desafiadores. Na terceira etapa, é desenvolvida uma função de facilidade baseada

em todos os fatores encontrados na primeira etapa. A função de facilidade é usada como parte da função que mede a aptidão dos agentes. Os agentes considerados mais aptos, nesse problema, são aqueles que mais aproximam o seu valor gerado através da função de facilidade, do valor gerado pelo adversário nessa mesma função. Os agentes mais aptos são usados para gerar novos agentes. O objetivo é aproximar cada vez mais o nível dos agentes ao nível do jogador. Como estudo de caso para a proposta é utilizado um jogo do tipo estratégia em tempo real (RTS).

3.4 Balanceamento dinâmico baseado em scripts dinâmicos

Na abordagem de [Spronck et al, 2005] para descrever o comportamento dos agentes são utilizadas bases de regras do tipo condição-ação. Essa é uma prática bastante comum na indústria de jogos. As regras são escritas manualmente e possuem pesos associados. O peso indica o quão boa determinada regra é para ser aplicada quando sua condição for satisfeita e também a sua probabilidade de ser escolhida pela política de escolha das ações. Durante a execução do jogo e nos treinamentos *offline* os pesos das regras são atualizados através de técnicas de aprendizagem de máquina. Quando uma regra é escolhida seu peso é aumentado se o resultado de aplicá-la for bom e diminuído caso contrário. Isso faz com que as regras mais eficientes possuam os maiores pesos ao longo do tempo e haja correção de eventuais erros na avaliação da qualidade das regras feita inicialmente. Como estudo de caso para a avaliação da técnica foi utilizado um RPG (Role Playing Game).

Para o balanceamento dinâmico, chamado de dimensionamento de dificuldade (*difficulty scaling)* pelos autores, são sugeridas três abordagens.

A primeira, chamada de *High-Fitness Penalising,* faz com que regras tenham seu peso atualizado de acordo com a adaptação ao usuário, fazendo com que o peso seja maior nas regras mais adaptadas e não nas mais eficientes. Isso significa que se o jogador tiver um desempenho muito ruim, as piores regras terão os maiores pesos.

A segunda abordagem, chamada de *Weight Clipping* faz com que haja um limite máximo de valor do peso para as regras, fazendo com que o peso das melhores regras pare de aumentar ao atingir o limite. Para tornar o jogo mais fácil,

basta diminuir o valor do limite de peso, e para torná-lo mais difícil basta aumentar. Essa abordagem, após muitas iterações pode começar a apresentar um comportamento similar ao de escolha aleatória de ações.

A terceira abordagem, chamada de *Top Culling,* é, segundo os próprios autores, a mais eficiente para a tarefa de balanceamento dinâmico e consiste em criar um limite máximo de peso para escolha das regras, mas não impedir que as regras tenham seu peso aumentado acima desse limite. Para serem escolhidas, as regras têm que ter um peso menor do que o limite, sendo descartadas as regras melhores do que o nível desejado. De forma similar a segunda abordagem, o jogo pode ficar mais fácil ou mais difícil alterando-se o valor do limite.

3.5 Balanceamento dinâmico baseado em aprendizagem por reforço

No trabalhos de [Andrade, 2006] e [Andrade et al, 2005] é proposto um sistema de balanceamento dinâmico baseado em aprendizagem por reforço (RL). RL é uma abordagem de aprendizagem de máquina em que o agente deve aprender sozinho o que fazer, em outras palavras, aprender quais as melhores ações a tomar, em cada um dos seus estados possíveis, de modo a maximizar um sinal numérico de recompensa [Sutton e Barto, 1998]. Diferentemente de outras técnicas de aprendizagem de máquina, o agente não possui nenhum tipo de supervisão e deve descobrir, através de sucessivas tentativas, que ações maximizam sua recompensa. É como se o agente jogasse um jogo sem conhecer previamente suas regras [Russel e Norvig, 2004]. Nessa abordagem de balanceamento dinâmico a técnica utilizada é a *Q-Learning*, desenvolvida por [Watkins e Dayan, 1992]. A escolha de tal técnica se justifica pela suas características: a garantia de convergir sempre para a melhor solução (desde que todas as combinações de estados e ações sejam suficientemente visitadas) e simplicidade em relação a outras técnicas de RL. Como estudo de caso é utilizado um jogo de luta.

Para realizar o balanceamento dinâmico o autor divide o problema em duas dimensões chamadas de competência e desempenho, como nos estudos em lingüística propostos por [Chomsky, 1965]. A competência é definida como aquilo

que o agente sabe que são ações boas ou ruins. O desempenho é o nível do uso da competência.

Para adquirir competência os agentes utilizam a aprendizagem por reforço tradicional para descobrir as melhores ações a serem tomadas em cada um dos estados possíveis do jogo. Os valores de executar cada ação em cada estado, chamados de ação-valor, são armazenados em uma matriz bi-dimensional que relaciona todos os estados a todas as ações. Todas as ações-valor são sempre atualizadas tanto em treinamentos *offline* quanto durante a execução real do jogo, sendo aumentado se ação é boa e reduzido se a ação é ruim, da mesma maneira que seria feito em uma abordagem tradicional de RL.

O desempenho do agente depende do nível em que o jogador atua. O nível do jogador é medido periodicamente através da função de facilidade e o balanceamento dinâmico do jogo ocorre escolhendo as ações correspondentes a esse nível, ou seja, a melhor ação para um determinado estado não é a mais provável de ser escolhida, a mais provável é a ação cujo nível mais se aproxima do nível do jogador. Por exemplo, se o jogador é iniciante e possui pouca habilidade os agentes devem atuar em níveis fáceis, escolhendo geralmente ações que são consideradas ruins de acordo com o que foi aprendido na aquisição de competência. De maneira oposta, os agentes devem atuar escolhendo as melhores ações na maioria dos casos, se a função de facilidade indicar um jogador habilidoso.

3.6 Justificativas dos problemas das técnicas propostas na literatura com jogos no domínio estudado

Tomando novamente como exemplo o jogo em que o jogador deve encontrar a saída do labirinto e considerando que esse labirinto não tem NPCs inimigos ou amigos dentro e que as medidas para avaliar o sucesso do jogador dentro de cada labirinto são o tempo que ele leva para concluí-lo e a quantidade de itens escondidos que jogador consegue encontrar. Percebe-se que aqui o *pay-off* do jogo é obtido da forma mais precisa apenas quando o labirinto é concluído.

A abordagem proposta por [Hunicke e Chapman, 2004] teria problemas em balancear dinamicamente o jogo nesse caso. Pois caso o tempo fosse alterado, isso seria facilmente percebido pelo usuário. Fora essa media, alterar os atributos do

personagem controlado pelo jogador não iria influenciar muito no desempenho do jogador.

A proposta de [Demasi e Cruz, 2002a] é muito dependente da existência de agentes autônomos. Definir genes para codificar características de labirintos e ainda definir labirintos modelo através de genes seria um trabalho demasiadamente complicado. Além disso, no caso do jogo dos labirintos não há adversários que possam morrer, tornando mais difícil um critério de substituição, sem contar que a convergência para uma boa solução seria lenta.

A abordagem de [Olesen, Yannakakis e Hallam, 2008] também é muito dependente de agentes. Não há sentido em governar os labirintos através de uma técnica como redes neurais, a não ser que os labirintos sejam modelados como agentes, o que faz pouco sentido.

Definir regras de comportamento é uma coisa feita tipicamente para agentes. Por isso a proposta de [Spronck et al, 2005] teria de ser adaptada ao jogo exemplo. Talvez seja possível definir os labirintos através de regras, embora isso faça pouco sentido. O problema de conversão muito lenta para uma boa solução também ocorreria devido à dificuldade em se realizar treinamentos *offline* já que não existem agentes adversários.

A proposta de [Andrade, 2006] enfrenta dificuldades parecidas com as de [Spronck et al, 2005] já que como no caso do jogo do labirinto não há adversários e fazer um agente jogador é um outro trabalho complexo, também tendo problemas na velocidade de convergência para uma boa solução. Além disso, essa técnica de aprendizagem por reforço seria prejudicada devido a lentidão da avaliação da possibilidade de avaliação da dificuldade sentida pelo jogador e da possibilidade de realizar outras ações de balanceamento.

4. Abordagens Propostas

Nos capítulos anteriores vimos alguns requisitos para a implementação do balanceamento dinâmico e algumas técnicas encontradas na literatura. Como se viu, todas as técnicas apresentam dificuldades para realizar o balanceamento em jogos com as características que interessam neste trabalho.

Para a realização do balanceamento dinâmico dos jogos que pertencem ao domínio que é estudado aqui são propostas duas técnicas. Os passos inicias das duas são iguais, porém as heurísticas para a escolha do próximo desafio são diferentes.

Primeiramente, considerando os desafios são pouco suscetíveis a alterações, deve-se ordená-los por dificuldade de maneira crescente. Essa ordenação pode ser feita manualmente pelo *game designer,* assim como pode ser feita através do uso de técnicas de aprendizagem de máquina. Utilizando-se a abordagem automática, a probabilidade de haver erros diminui, principalmente porque essa ordenação pode apresentar um grande espaço de variáveis, ainda podendo haver o agravante de existir dependências entre elas. A escolha de uma boa técnica para realizar essa tarefa é um trabalho promissor, porém relativamente complexo por isso devido ao pouco tempo de pesquisa sobre esse assunto ainda não foi realizado. No estudo de caso apresentado a etapa de ordenação foi feita manualmente, aproveitando o que já havia sido feito pelo *game designer*.

Uma vez que essa ordem de dificuldade é estabelecida, deve-se dividir os desafios em grupos de dificuldade, como mostra a figura 4.1. Em cada grupo, os desafios devem possuir um nível parecido. A quantidade de grupos e desafios por grupo é variável e depende da decisão da equipe de desenvolvimento do jogo. Os grupos também são ordenados por dificuldade de maneira crescente.

Como em muitas abordagens de balanceamento dinâmico automático, é necessária a utilização de uma função de facilidade para periodicamente aferir o nível de dificuldade sentido pelo usuário. Uma vez que o valor dessa função é calculado, duas técnicas são propostas para a escolha do novo desafio. A primeira técnica é chamada de **Escolha linear de desafios** a segunda é chamada de **Escolha de desafios com possibilidade de saltos de níveis.** Os detalhes de cada uma delas serão vistos a seguir. Ambas possuem em comum a possibilidade de

aumentar o nível de dificuldade, escolhendo desafios num grupo de nível de dificuldade maior do que o grupo de onde último desafio foi escolhido. Nas duas técnicas também existe a possibilidade de diminuir o nível de dificuldade, fazendo com que o usuário tenha o próximo desafio escolhido a partir de um grupo cujos desafios apresentam dificuldade menor.

Depois de ver as técnicas propostas, serão justificados os cumprimentos de alguns requisitos propostos na literatura.

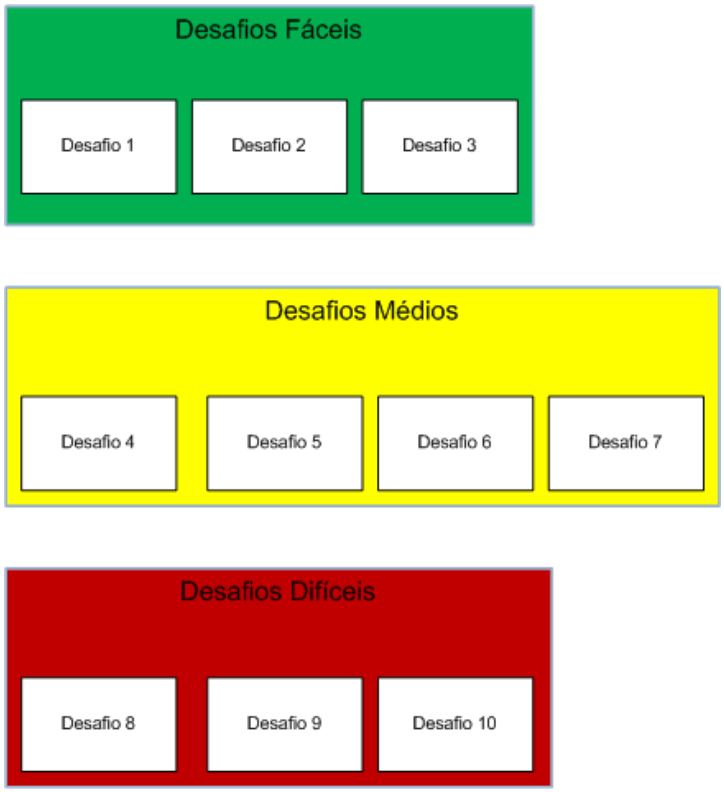


Figura 4.1: Divisão de desafios de níveis similares por grupos de dificuldade

4.1 Escolha Linear de desafios

Na escolha linear de desafios, não existem saltos entre os grupos de dificuldade. Ao encerrar um desafio, o sucesso do usuário é avaliado pela função de facilidade. Se o valor retornado pela função for considerado suficientemente bom o usuário será promovido para um grupo de desafios mais elevado. Este próximo grupo é o imediatamente posterior ao grupo do último desafio escolhido. O próximo desafio será sorteado aleatoriamente deste novo grupo. Se o contrário ocorrer, ou seja, se a função de facilidade indicar que o jogador não foi bem no último desafio, o

jogador será rebaixado para um grupo de desafios de menor dificuldade. Esse grupo deve ser imediatamente anterior ao ultimo grupo em que o usuário se encontrava. Novamente a escolha do próximo desafio deve ser aleatória dentro do novo grupo de desafios. Existe ainda a possibilidade do jogador se manter no mesmo grupo de desafios, se a o resultado da função de facilidade for intermediário entre o que é considerado um bom resultado e o que é considerado um mau resultado.

Algumas considerações a serem feitas são: O jogador iniciante deve começar no primeiro grupo de desafios, em outras palavras, no grupo que apresenta os desafios mais simples, se for um jogador mais experiente, o uso de perfis para salvar o progresso permite iniciar o jogador em um grupo mais avançado da próxima vez que ele for jogar. Outra consideração é que a possibilidade de repetição em um mesmo desafio depende da escolha do *game designer,* pois em alguns jogos ela pode ser considerada interessante ou até pode ser necessária. Mais uma consideração é que a quantidade de intervalos para ponderar a mudança de grupos também depende das decisões da equipe desenvolvedora, sendo possível ainda, dividir os grupos em subgrupos. Por último, por uma questão lógica, quando o jogador se encontra no primeiro grupo de desafios e não consegue sair, não haverá possibilidade de baixar o nível de dificuldade, a não ser que novos desafios sejam criados. Isso caracteriza claramente um problema na elaboração dos desafios. O caso contrário, pode ser um problema, mas também pode caracterizar que o jogador atingiu o limite máximo de dificuldade dos desafios do jogo.

Abaixo, será apresentado um pseudocódigo do algoritmo 4.1 relativo a essa proposta, considerando apenas três intervalos para a interpretação dos resultados enviados pela função de facilidade e que é possível repetir desafios já realizados anteriormente.

Algoritmo 4.1: Escolha linear de desafios.

Entradas:
Grupo Atual de desafios, grupoAtual.
Valor da função de facilidade, f(.)
Valor Maximo para fracasso no desafio, fracassoMaxValor
Valor Mínimo para sucesso no desafio, sucessoMinValor.

Saida:
Um novo desafio a ser executado

Algoritmo:

1. Se f(.) <= fracassoMaxValor
   Então grupoAtual.Decrementar()
2. Senão Se fracassoMaxValor < f(.) < sucessoMinValor
   Então grupoAtual.Manter()
3. Senão grupoAtual.Incrementar()
4. ProximoDesafio =
   EscolherProximoDesafioAleatoriamente(grupoAtual)
5. Retorne ProximoDesafio

4.2 Escolha de desafios com possibilidade de saltos de níveis

No caso dessa técnica existe a possibilidade de escolher desafios em um grupo que não seja imediatamente anterior ou posterior ao ultimo grupo de desafios utilizado.

Para a escolha do próximo grupo de desafios, deve-se definir uma escala. O tamanho dessa escala deve ser definido pela equipe desenvolvedora. Por uma questão de simplificação recomenda-se que esse tamanho seja um numero de valor alto. Em seguida, divide-se esse valor definido pelo número de grupos de dificuldade também definido pela equipe. Isso serve para definir um intervalo de valores para cada grupo de dificuldade. Por exemplo, pode-se definir uma escala de tamanho 100 e dividir os desafios em 4 grupos de dificuldade, fazendo com que cada grupo tenha um intervalo de tamanho 25, como ilustra a figura 4.2. Caso a equipe deseje, é

possível fazer intervalos de tamanhos diferentes, baseando-se, por exemplo, no número de desafios em cada grupo.

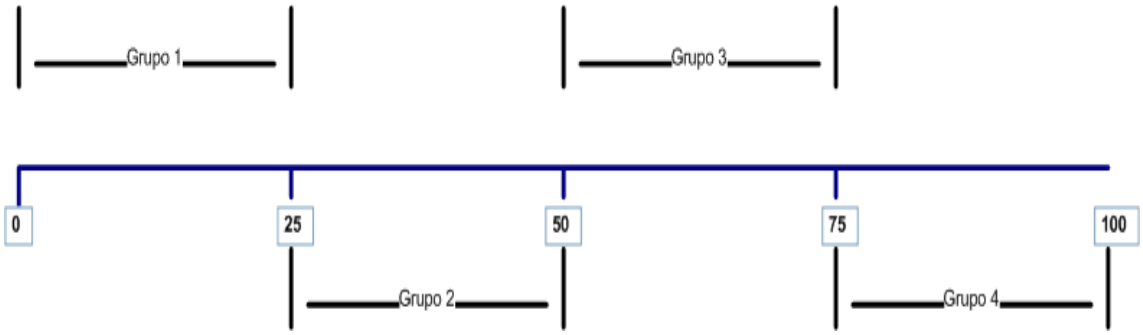


Figura 4.2: A escala de desafios de tamanho 100 com 4 grupos de dificuldade, cada um tem um intervalo de tamanho 25.

Essa escala funciona como um termômetro da habilidade do jogador. Para medir essa habilidade, inicialmente deve-se definir um valor entre zero e tamanho da escala. Recomenda-se que este valor esteja localizado dentro do intervalo do grupo mais fácil se o jogador for iniciante, caso haja o sistema de perfis pode-se guardar o ultimo valor da escala atingido pelo usuário. Por exemplo, para um jogador iniciante, na escala acima, é possível indicar que ele começa com o valor de habilidade 20.

À medida que os desafios são concluídos esse valor de habilidade é atualizado. A função de facilidade deve definir a atualização desse valor. Em caso de um valor pequeno retornado por ela, o valor da habilidade pode ser reduzido. Caso a função desafio retorne um valor grande, o valor da habilidade pode ser aumentado. Em caso de um valor intermediário retornado pela função pode-se manter o valor da habilidade constante, ou alterar levemente seu valor para mais ou para menos. A quantidade de intervalos possíveis para dividir o valor de retorno da função desafio e que ações tomar em cada intervalo também dependem das decisões da equipe desenvolvedora.

É importante frisar que o valor da habilidade não deve sair da escala, ou seja, esse valor não deve se tornar menor que zero ou maior que o tamanho da escala, pois isso causaria comportamentos incorretos da escolha de desafios. Por exemplo, caso um usuário fosse mal consecutivas vezes para então começar a conseguir os objetivos, o valor da habilidade acumulado muito pequeno faria com que ele tivesse que repetir desafios fáceis inúmeras vezes, podendo causar frustração ao usuário.

Após a atualização do valor da habilidade, usa-se a escala para saber em que grupo de dificuldade o próximo desafio deve ser escolhido. Novamente utilizando o exemplo ilustrado na figura 4.2, se o valor da habilidade antes do desafio ser cumprido era 42, ou seja, o último desafio escolhido foi do grupo 2 e depois de utilizar a função desafio para atualizar o valor da habilidade o novo valor for 78, o próximo desafio será escolhido aleatoriamente do grupo 4. Isso justifica porque é possível saltar de níveis. Da mesma maneira é possível que regressão do nível seja maior do que apenas um grupo de dificuldade.

Um cuidado especial é preciso ser tomado, para evitar saltos grandes demais que possam exibir desafios de níveis muito diferentes consecutivamente. Pois se isso ocorrer certamente poderá ferir os requisitos de **Inércia, Eficácia** e **Clareza.** Além dessa consideração, as considerações explanadas na seção 4.1 também devem ser averiguadas.

Abaixo será exibido o pseudocódigo do algoritmo 4.2, relativo a essa proposta. Nessa descrição do algoritmo, será considerado que a cada grupo de dificuldade terá o mesmo tamanho de intervalo, que haverão 4 grupos de dificuldade e que só existem três possibilidades de alteração do valor da habilidade dependendo do valor retornado pela função facilidade.

Algoritmo 4.2: Escolha de desafios com possibilidade de saltos de níveis

Entradas:

Valor da habilidade atual, habilidadeAtual.
Valor da função de facilidade, f(.)
Tamanho da escala de desafios, tamanhoEscala
Numero de grupos de dificuldade, nGruposDificuldade
Valor Maximo para fracasso no desafio, fracassoMaxValor
Valor Mínimo para sucesso no desafio, sucessoMinValor.

Saida:

Um novo desafio a ser executado

Algoritmo:

1. valorIntervalo = tamanhoEscala/nGruposDificuldade
2. Se f(.) <= fracassoMaxValor
   Então habilidadeAtual.Decrementar()
3. Senão Se fracassoMaxValor < f(.) < sucessoMinValor
   Então habilidadeAtual.Manter()
4. Senão HabilidadeAtual.Incrementar()
5. Se habilidadeAtual <= valorIntervalo
   Então grupoAtual = 1
6. Se valorIntervalo < habilidadeAtual <= valorIntervalo * 2
   Então grupoAtual = 2
7. Se valorIntervalo * 2 < habilidadeAtual <= valorIntervalo * 4
   Então grupoAtual = 3
8. Senão grupoAtual = 4
9. ProximoDesafio =
   EscolherProximoDesafioAleatoriamente(grupoAtual)
10. Retorne ProximoDesafio

4.3 Justificativa do cumprimento de requisitos

Com os cuidados necessários na realização das ações de balanceamento os requisitos chamados de **Inércia, Eficácia** e **Clareza** serão cumpridos.

Já os requisitos **Rápida adaptação inicial, Rápida adaptação continua** e **Velocidade** terão de ser relaxados devido as características dos jogos do domínio estudado, já que não existem comportamentos e atributos a serem modificados e alterações no ambiente prejudicariam a experiência do jogador, por isso o balanceamento ocorre de forma lenta.

Quanto aos outros requisitos propostos na literatura, considera-se que **Racionalidade** é cumprida, pois não há nenhuma forma de trapaça. É possível considerar também o cumprimento da **Robustez, Execução** e **Desenvolvimento** devido à simplicidade dos algoritmos apresentados.

Os requisitos **Variedade** e **Escalabilidade** dependem muito das decisões tomadas pela equipe de desenvolvimento. Pois em caso de elaboração de uma função de dificuldade pouco eficiente ou de um número reduzido de desafios, esses requisitos podem ser cumpridos apenas parcialmente ou não serem cumpridos.

Por último os requisitos chamados de **Eficiência** e **Consistência** não são cumpridos, pois ainda não há nenhuma aprendizagem nos algoritmos propostos.

5. Implementação do estudo de caso

Para a avaliação das técnicas propostas no capitulo anterior foi utilizado como estudo de caso o primeiro protótipo funcional de um jogo casual chamado **Pizzaiolo** que está sendo desenvolvido pela Manifesto Game Studio [Manifesto, 2009]. Esse é um jogo do tipo *puzzle* (Quebra-Cabeça). Esse tipo é caracterizado por apresentar problemas que exigem raciocínio do jogador para que este consiga resolve-los [Wikipedia, 2009].

Na próxima seção será mostrada a descrição do jogo e na seção seguinte será justificado o porquê da escolha desse jogo como estudo de caso e como as técnicas apresentadas no capítulo anterior foram implementadas nele.

5.1 Descrição do jogo

Como o próprio nome do jogo sugere, o **Pizzaiolo** é ambientado em um restaurante e o objetivo é fazer pizzas. As pizzas a serem feitas são escolhidas pelo computador e variam de acordo com tamanho (sendo 5 possíveis) e receitas (sendo 18 possíveis). O sabor de pizza escolhida e o seu tamanho são exibidos no canto superior esquerdo da tela. Ao receber o pedido o jogador deve clicar na massa correspondente ao tamanho pedido, como ilustra a figura 5.1. Em seguida o jogador deve encontrar os ingredientes que correspondem à receita pedida, isso é ilustrado na figura 5.2. Se o jogador não houver decorado os ingredientes da receita pode consultar o livro de receitas, que fica no canto inferior esquerdo da tela.

Cada ingrediente que recebe um clique gera um conjunto de *slots* com o seu formato na pizza. Após todos os ingredientes da receita serem clicados eles são colocados na pizza. Inicialmente os ingredientes são postos em *slots* que possuem um formato diferente do seu. Nessa etapa o jogador deve trocar os ingredientes dois a dois até que todos fiquem em um *slot* cujo formato é igual ao seu. Isso é ilustrado pela figura 5.3.

Essas três etapas do jogo devem ser realizadas no menor tempo possível. Ao terminar de montar uma pizza, uma gorjeta é calculada indicando quão bem o jogador se saiu para realizar a tarefa. A gorjeta é inversamente proporcional ao tempo consumido pelo usuário para realizar o desafio. Após esse cálculo uma nova

pizza é escolhida pelo jogo. Esse processo pode se repetir infinitamente, por isso nesse jogo a repetição das pizzas é necessária.

Figura 5.1: Seleção da massa

Figura 5.2: Seleção dos ingredientes

Figura 5.3: Realizando as trocas dos ingredientes para colocá-los nos *slots* corretos.

5.2 Descrição da implementação do balanceamento dinâmico no jogo.

Pela descrição do Pizzaiolo vista na seção anterior é possível perceber que ele tem as características que o tornam um jogo que está dentro do domínio problema estudado. Pois uma vez que as pizzas são escolhidas pelo computador, não é possível realizar nenhuma ação de balanceamento que seja imperceptível até que o jogador consiga concluir o desafio. Além disso, só é possível saber se a última pizza escolhida foi de fato boa para o nível de habilidade do jogador depois que o desafio termina, através do cálculo da gorjeta.

Para a implementação das técnicas propostas a IDE (*Integrated Development Environment*) utilizada foi o Microsoft Visual C# 2005 *express edition*, utilizando a versão 2.0 da plataforma .NET. O uso dessa IDE se justifica pela facilidade de integração com a *engine* de jogos TorqueX 2D [TorqueX, 2009] e com a biblioteca XNA [XNA, 2009] que foram utilizadas na construção do jogo.

As modificações no jogo foram feitas num modo chamado "*Free Play*". Isso foi feito para que nos testes com usuário o jogador só tivesse contato com a construção

das pizzas, sem outras características que o modo "*Campaing*" apresenta como, por exemplo, a compra de ingredientes e estágios bônus.

Três classes principais foram desenvolvidas para a realização do balanceamento, elas foram chamadas de **BalancedWaiter, BalancedClient** e **FreePlayLevel.** A seguir serão descritas as responsabilidades em relação ao balanceamento dinâmico de cada uma delas. Além dessas, foram desenvolvidas mais classes de apoio para a geração de logs, configuração do balanceamento e leitura de definições de *game design*.

A classe **BalancedWaiter** é responsável por dividir as pizzas por grupo de dificuldade de acordo com o que foi definido pelo *game designer.* A **BalancedClient** é responsável por escolher a próxima pizza de acordo com uma das duas técnicas de balanceamento proposta. A técnica a ser utilizada é definida através da classe chamada **BalanceConfig**. Além disso, a classe **BalancedClient** possui um método que faz o cálculo da gorjeta e possui os atributos que indicam o nível de dificuldade atual. As classes citadas anteriormente, exceto a **BalanceConfig**, herdam alguns métodos e atributos de classes do jogo original. Isso foi necessário para que a arquitetura do jogo se mantivesse, com o objetivo de não haver nenhum problema de integração durante a implementação das técnicas. A classe **FreePlayLevel** é uma modificação da classe original homônima, sua responsabilidade é calcular o valor da função de facilidade, ela ainda cuida do *pipeline* do jogo no modo "*Free Play*". A figura 5.4 ilustra o que foi explicado através de um diagrama de classes simplificado, focado somente no balanceamento dinâmico.

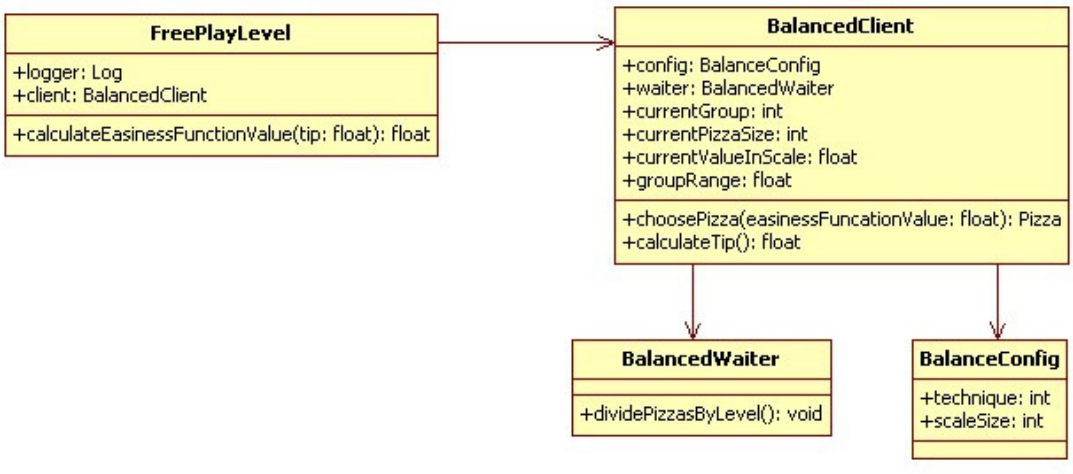


Figura 5.4: Diagrama das classes de balanceamento simplificado.

Para o cálculo da gorjeta (G) foi utilizada a fórmula abaixo, onde GM é a gorjeta máxima definida pelo *game designer,* TR é a porcentagem de tempo usado para concluir o desafio sobre o tempo máximo e N é o nível do desafio, ou seja, o número do grupo de dificuldade.

$$G = GM * (1 – TR) * N$$

A função de dificuldade é mostrada a seguir, ela é simplesmente a porcentagem da gorjeta máxima conseguida, dada por:

$$f(.) = G/GM$$

Definiu-se que nesse jogo, as pizzas são divididas em cinco grupos de dificuldade e que nos testes a pizza inicial é sempre a mais fácil de todas.

Cada vez que uma pizza é concluída a função dificuldade é calculada e a próxima pizza é escolhida com base nesse valor. Cada uma das técnicas propostas escolhe a próxima pizza de uma maneira diferente. A seguir será mostrado, como se dá essa escolha de acordo com cada técnica.

5.2.1 Escolha Linear

Para a escolha linear, os resultados da função desafio foram divididos em sete intervalos. O algoritmo 5.1 abaixo é uma modificação do algoritmo 4.1 adaptado especificamente para o Pizzaiolo. Ambos possuem as mesmas entradas (exceto pelos valores de fracasso e sucesso) e saída, por isso o pseudocódigo só exibirá o algoritmo.

Algoritmo 5.1: Escolha linear da próxima pizza

Algoritmo:

1. Se f(.) <= 0.1

   Então: grupoAtual.Decrementar();
   tamanhoPizzaAtual.Decrementar()
2. Senão Se 0.1 < f(.) <= 0.3

   Então: grupoAtual.Decrementar()
3. Senão Se 0.3 < f(.) <= 0.5

   Então:tamanhoPizzaAtual.Decrementar()
4. Senão Se 0.5 < f(.) <= 0.65

   Então: //Não há modificações
5. Senão Se 0.65 < f(.) <= 0.75

   Então: tamanhoPizzaAtual.Incrementar()
6. Senão Se 0.75 < f(.) <= 0.9

   Então: grupoAtual.Incrementar()
7. Senão

   grupoAtual.Incrementar(); tamanhoPizzaAtual.Incrementar()
8. ProximoDesafio = EscolherProximoDesafioAleatoriamente(grupoAtual)
9. Retorne ProximoDesafio

5.2.2 Escolha com possibilidades de saltos de níveis

Para essa técnica definiu-se uma escala de tamanho 100 e como são cinco grupos cada um ficou com um intervalo de tamanho 20. O valor inicial na escala foi definido como sendo 20 que é o valor máximo de habilidade (chamado de H no pseudocódigo) para se escolher um desafio do grupo 1.

Duas fórmulas básicas foram definidas para o cálculo do novo valor da habilidade, após cada desafio concluído. Uma fórmula decrementa esse valor e a outra incrementa.

Para o cálculo do decremento (Dh) definiu-se a equação apresentada abaixo, onde R é o valor do tamanho do intervalo do grupo atual na escala.

$$Dh = (1 - f(.)) * R$$

Para o cálculo do incremento (Ih) definiu-se a fórmula apresentada a seguir:

$$\mathbf{Ih = f(.) * R}$$

Uma constante relevante a ser considerada é chamado de Ft e o seu valor nessa implementação é 6, vinda da conta simples valor do tamanho máximo (5, já que são 5 tamanhos de pizza) acrescido de 1. Essa constante será usada para aumentar os valores de decremento, para o caso de resultados muito ruins indicados pela função de facilidade.

O aumento ou redução do tamanho da pizza é estocástico e é baseado no valor de f(.).

O pseudocódigo que será apresentado abaixo divide a função facilidade em 5 intervalos. Esse algoritmo, chamado de algoritmo 5.2 é a adaptação do algoritmo 4.2 apresentado no capítulo anterior para o Pizzaiolo. Ambos possuem as mesmas entradas (exceto pelos valores de sucesso e fracasso) e a mesma saída por isso o pseudocódigo está concentrado no algoritmo.

Algoritmo 5.2: Escolha com saltos de níveis da próxima pizza

Algoritmo:

1. valorIntervalo = tamanhoEscala/nGruposDificuldade
2. Se f(.) <= 0.3

   Então: H = H – (Dh + ((Ft - tamanhoAtual) * 2)

   tamanhoPizzaAtual.Decrementar()
3. Senão Se 0.3 < f(.) <= 0.5

   Então: H = H – Dh

   Se randomFloat > f(.)

   Então: tamanhoPizzaAtual.Decrementar()
4. Senão Se 0.5 < f(.) <= 0.7

   Então: H = H + (tamanhoPizzaAtual * 2)
5. Senão Se 0.7 < f(.) <= 0.85

   Então: H = H + Ih

   Se randomFloat < f(.)

   Então: tamanhoPizzaAtual.Incrementar()
6. Senão

   H = H + Ih + (tamanhoPizzaAtual * 2)
7. Se H <= valorIntervalo

   Então: grupoAtual = 1
8. Senão Se valorIntervalor < H <= valorIntervalo*2

   Então: grupoAtual = 2
9. Senão Se valorIntervalor* 2 < H <= valorIntervalo*3

   Então: grupoAtual = 3
10. Senão Se valorIntervalor * 3 < H <= valorIntervalo*4

    Então: grupoAtual = 4
11. Senão Se valorIntervalor * 4 < H <= valorIntervalo*5

    Então: grupoAtual = 5
12. ProximoDesafio =

    EscolherProximoDesafioAleatoriamente(grupoAtual)
13. Retorne ProximoDesafio

6. Experimentos com usuários

Para uma avaliação das técnicas propostas e para definir qual das duas apresenta a maior eficiência em escolher desafios no nível de dificuldade correto para a habilidade dos jogadores foi realizada uma pesquisa com acompanhamento de vinte jogadores com diferentes perfis. O jogo utilizado para validar as técnicas propostas foi Pizzaiolo, apresentado no capitulo anterior.

A pesquisa foi dividida em duas partes, a parte 1 serviu para identificar o perfil dos jogadores que participaram dos testes, somente através das respostas dadas por eles no questionário. A parte 2 para avaliar as técnicas propostas, nessa parte foram levados em consideração as respostas dadas no questionário e os *logs* gerados automaticamente pelo sistema. Os resultados obtidos em cada uma dessas partes serão mostrados nas seções seguintes, bem como a sua interpretação.

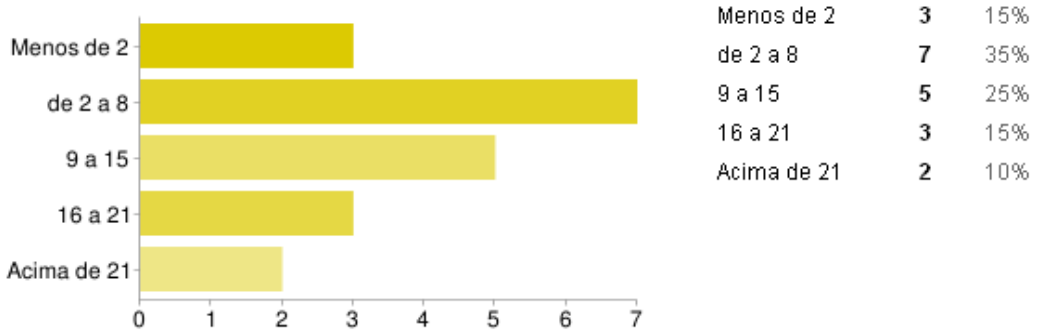
O avaliador que acompanhou os testes teve a responsabilidade de esclarecer os objetivos da pesquisa e explicar como jogar, já que o tutorial não estava acessível no modo de jogo usado ("*Free Play*"). Tais explicações também estavam acessíveis no questionário aplicado, que pode ser observado no Apêndice A.

6.1 Perfil dos jogadores

Na pesquisa realizada, a faixa etária predominante foi dos 16 aos 27 anos, sendo responsável por 75% do total. Com menos de 16 anos foram 5%, 10% de 28 a 39 anos e 10% com mais 39 anos. Dos vinte jogadores, quatorze são homens e seis são mulheres.

Na auto-avaliação de que tipo de jogador as pessoas se consideram, 55% dos usuários afirmaram ser jogadores casuais e 45% habituais (*hardcore).* Apesar disso, quando foi perguntado quantas horas por semana essas pessoas costumam jogar, surpreendentemente alguns que se consideram casuais afirmaram jogar pelo menos nove horas nesse período e outros que se consideram habituais afirmaram jogar 8 horas ou menos. O gráfico 6.1 exibe as porcentagens de horas jogadas por semana afirmadas por cada pessoa.

Gráfico 6.1: Quantidade horas que cada jogador costuma jogar por semana

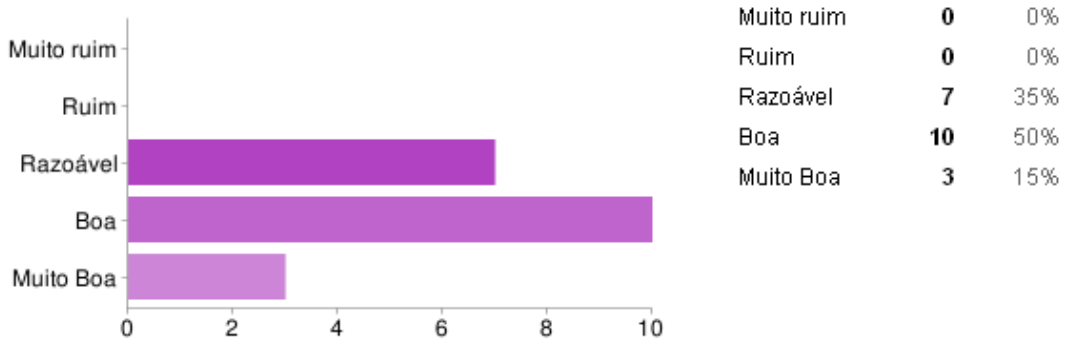


Fonte: Dados de campo, 2009.

A pesquisa indicou ainda que os jogos do tipo puzzle são jogados costumeiramente por 55% dos que responderam a pesquisa. O estilo de jogo mais procurado por quem respondeu a pesquisa é ação, 70% dos jogadores afirmaram gostar desse tipo de jogo.

Quando foi pedido para que as pessoas avaliassem a própria habilidade em jogos de uma maneira geral, todos consideraram sua habilidade pelo menos razoável. O gráfico 6.2 mostra a distribuição de como os jogadores avaliam a própria habilidade.

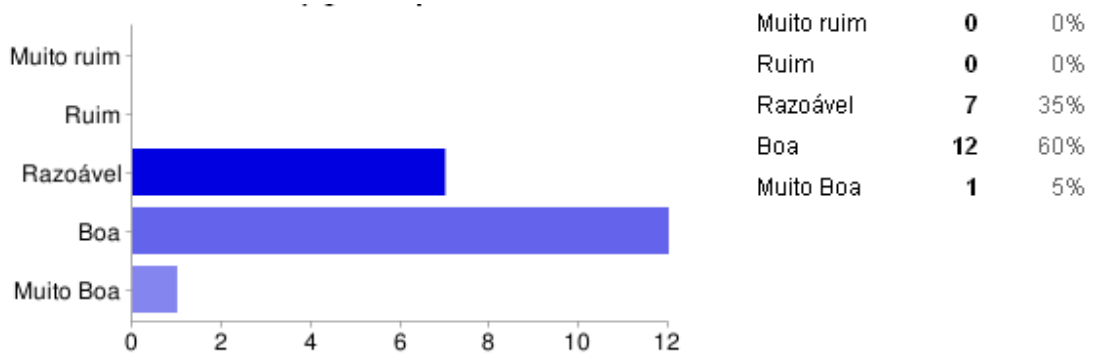
Gráfico 6.2: Auto-avaliação da habilidade jogando de maneira geral



Fonte: Dados de campo, 2009.

A mesma auto-avaliação foi solicitada especificamente para jogos do tipo puzzle. Isso foi solicitado, porque esse é exatamente o tipo de jogo do Pizzaiolo. Novamente todas as pessoas se consideraram sua habilidade pelo menos razoável. Diferentemente dos resultados da pergunta anterior, apenas uma pessoa se considerou muito boa nesse tipo de jogo. O gráfico 6.3 ilustra esses resultados.

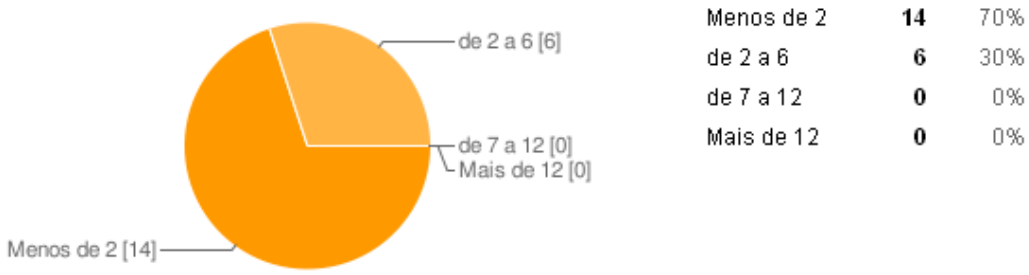
Gráfico 6.3: Auto-avaliação da habilidade jogando puzzles



Fonte: Dados de campo, 2009.

Por último, todas as pessoas afirmaram jogar esse tipo de jogo menos que 6 horas por semana. Sendo que 70% jogam menos de 2 horas e 30% de 2 a 6 horas por semana. O gráfico 6.4 ilustra essa situação.

Gráfico 6.4: Horas dedicadas por semana a jogos do tipo puzzle.



Fonte: Dados de campo, 2009.

6.2 Análise das técnicas

Para a realização da segunda parte do teste, cada pessoa jogou seis pizzas. Para cada pizza terminada a pessoa deveria responder qual o nível de dificuldade sentido para realizar o desafio. O nível de dificuldade foi dividido nos três valores clássicos e discretos, **fácil, médio e difícil**. Isso foi feito porque os significados desses valores são facilmente compreendidos por quem joga.

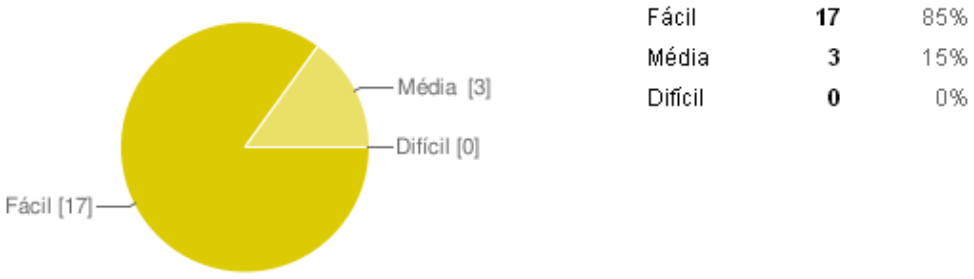
Como o objetivo das técnicas aplicadas é fazer o jogo propor desafios equilibrados ao usuário, assumiu-se que o resultado ideal ocorre quando o jogador considerar que a pizza apresentou uma dificuldade **média**. Assumiu-se também, que

quando o mesmo considerar que uma pizza foi **fácil** é um resultado melhor do que quando ele considerar **difícil**. Isso porque, como a duração do teste é muito pequena, dificilmente qualquer jogador conseguiria atingir um nível de habilidade bom o suficiente para que o sistema pudesse propor desafios mais complexos.

Como já afirmado, além de validar as técnicas, o objetivo da pesquisa também é tentar descobrir qual delas se mostra mais eficiente, por isso, nessa etapa os jogadores foram divididos em dois grupos de dez pessoas. O grupo 1, jogou com a escolha da próxima pizza sendo feita pela técnica chamada de **escolha linear de desafios** e o grupo 2 pela técnica chamada de **escolha de desafios com possibilidade de saltos de níveis**. Para os jogadores, a técnica utilizada foi completamente transparente, no questionário elas eram simplesmente chamadas de abordagem 1 e abordagem 2.

A pizza inicial em ambas as abordagens foi aquela que é considerada pelo *game designer* como a pizza mais simples de todas. O gráfico 6.5 abaixo mostra que para a maioria das pessoas ela foi de fato considerada fácil. O objetivo de colocar essa pizza inicialmente era fazer com que o jogador se familiarizasse com o jogo. Além disso, caso a pessoa quisesse, durante a explicação de como jogar, poderia fazer uma pizza exemplo para fixação das regras sem ter que responder o questionário.

Gráfico 6.5: Avaliação de dificuldade da pizza inicial padrão



Fonte: Dados de campo, 2009.

Para a avaliação das técnicas definiu-se que os dados apresentados serão divididos por abordagem da seguinte maneira: serão consideradas quantas pizzas foram votadas em cada dificuldade, depois serão mostradas as curvas de crescimento da quantidade de votos de cada uma das dificuldades ao longo das

cinco pizzas e por último serão exibidos como cada jogador avaliou a dificuldade geral do jogo e seu desempenho.

Além de mostrar esses dados de maneira geral, eles serão exibidos dividindo os jogadores de cada grupo em subgrupos, separando os jogadores **casuais e habituais**. Essa escolha de divisão possui a melhor distribuição entre dois grupos principais. Para decidir a que subgrupo pertence cada jogador, foi definido que será usado o que cada um informou na quantidade de horas jogadas por semana e não como o jogador se vê. Essa decisão se baseia no fato de que se o jogador joga mais de uma hora em média todos os dias, isso pode ser considerado um hábito, logo esse jogador deve ser mais experiente. Com isso, considera-se que os jogadores casuais são aqueles que jogam até 8 horas por semana e habituais os outros.

As próximas subseções mostram os dados explicados acima para cada uma das técnicas bem como a interpretação desses resultados.

6.2.1 Avaliação da técnica de escolha linear de desafios

Ao longo das 50 pizzas propostas por essa abordagem (cinco para cada jogador), a **dificuldade média** foi votada 25 vezes. Isso significa que metade das pizzas apresentadas foram julgadas como estando em um nível de desafio ideal**.** Dezenove pizzas foram consideradas **fáceis** e seis pizzas apenas foram consideradas **difíceis**. O gráfico 6.6 exibe as curvas de variação da quantidade de cada dificuldade para cada pizza.

Gráfico 6.6: Curvas de variação da quantidade de votos em cada nível de dificuldade para cada pizza na abordagem 1.

Fonte: Dados de campo, 2009.

A quantidade de pizzas consideradas **médias** foi alta no começo da avaliação. As pizzas consideradas **difíceis e fáceis** apresentam certa simetria no comportamento, que na análise profunda dos dados se justifica pelo fato de que algumas pessoas quando vão bem numa pizza a consideram fácil e assim que o nível aumenta e o desempenho diminui consideram a pizza seguinte difícil. A avaliação da troca de dificuldade rápida com saltos grandes não é um bom comportamento. Esse fenômeno foi inesperado já que essa abordagem é mais gradual.

Dividindo-se a análise dos resultados em jogadores causais e habituais, percebe-se que os jogadores casuais, quase todos afirmaram ter puzzle como preferência e que em geral eles consideraram o jogo de médio para fácil, com raras pizzas consideradas difíceis. Isso pode ser justificado pela prática que eles apresentam no estilo. Os jogadores habituais apresentaram uma avaliação com mais extremos, considerando mais pizzas fáceis e difíceis. Esses jogadores ainda apresentam comentários informando que algumas vezes a troca de níveis é nítida e até demonstrando certa insatisfação considerando que jogam mal, apesar de um desempenho razoável de acordo com os logs.

Ao analisar como os jogadores avaliaram a dificuldade do jogo de forma geral, 5 pessoas consideraram o jogo **médio**, 3 **difícil** e 2 **fácil**. A maioria dos que considerou fácil foram jogadores habituais, o jogadores casuais consideraram o jogo com média dificuldade. O gráfico 6.7 ilustra essa divisão. O gráfico 6.8 mostra como cada pessoa avalia o próprio desempenho no jogo. Com essa análise é possível inferir que em geral os jogadores habituais são mais críticos em relação ao próprio desempenho.

Gráfico 6.7: Análise da dificuldade geral do jogo (abordagem 1)

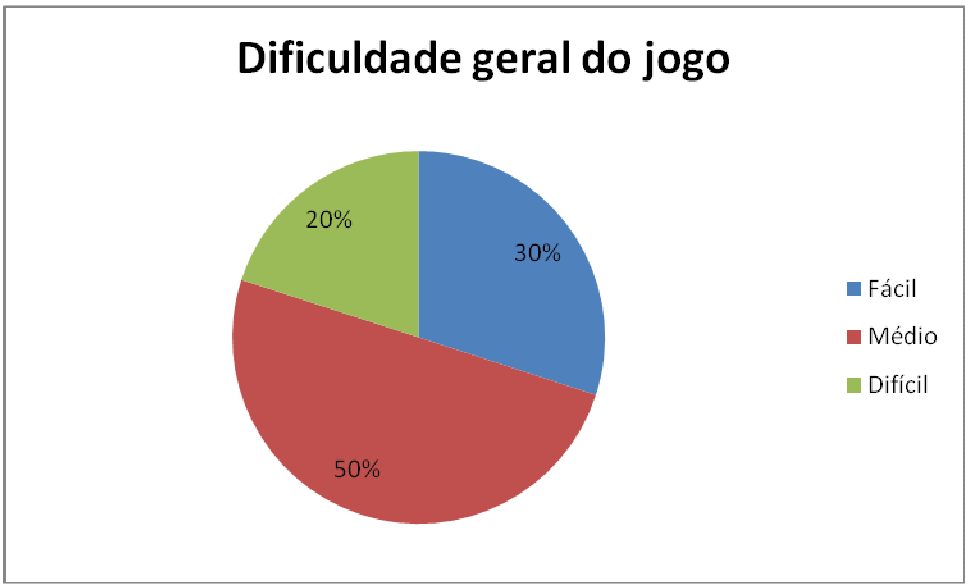
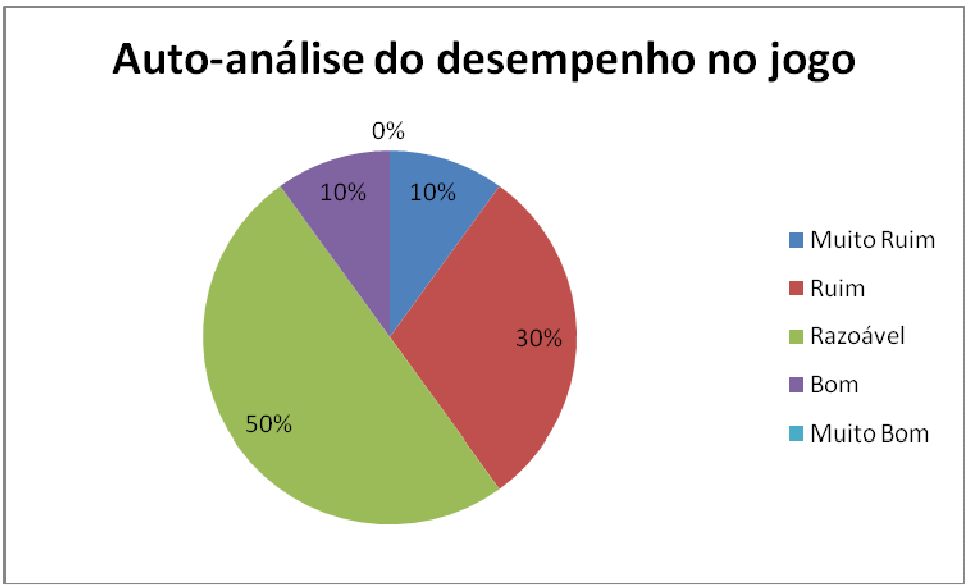


Gráfico 6.8: Auto-análise do desempenho geral no jogo (abordagem 1)



Fonte: Dados de campo

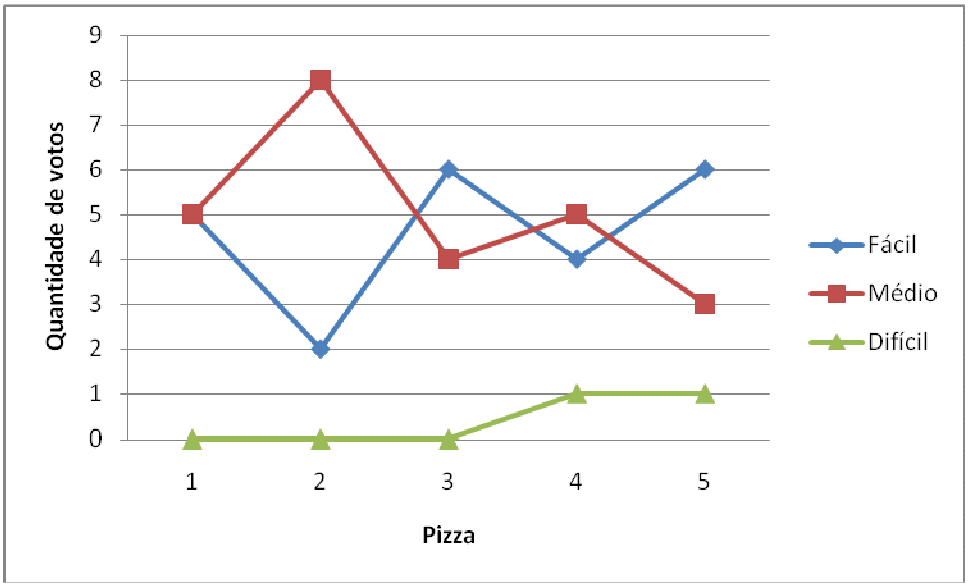
De uma maneira geral o desempenho da técnica pode ser considerado razoável**,** pois apesar de metade dos desafios apresentarem dificuldade mediana, segundo os jogadores, as rápidas variações entre pizzas fáceis e difíceis e a percepção da troca de níveis por alguns usuários foram prejudiciais.

O estudo dessa técnica torna claro que é necessária uma avaliação mais precisa da real dificuldade de cada desafio, o surgimento de algumas pizzas mesmo consideradas fáceis pelo *game designer* não foram avaliadas da mesma maneira pelos jogadores, isso pode justificar em parte o problema da clara variação de dificuldade apresentado anteriormente, além de confirmar que uma ordenação automática de desafios é um trabalho promissor.

6.2.2 Avaliação da técnica com possibilidade de saltos de níveis

Nessa segunda técnica também foram propostas 50 pizzas. Coincidentemente também 25 pizzas foram eleitas **médias**, apenas 2 **difíceis** e 23 **fáceis**. Esse resultado pode ser considerado melhor que o da primeira abordagem, pois como já afirmado, considera-se o resultado fácil melhor que o difícil. O gráfico 6.9 exibe as curvas de variação da quantidade de cada dificuldade para cada pizza para essa segunda abordagem.

Gráfico 6.9: Curvas de variação da quantidade de votos em cada nível de dificuldade para cada pizza na abordagem 2.



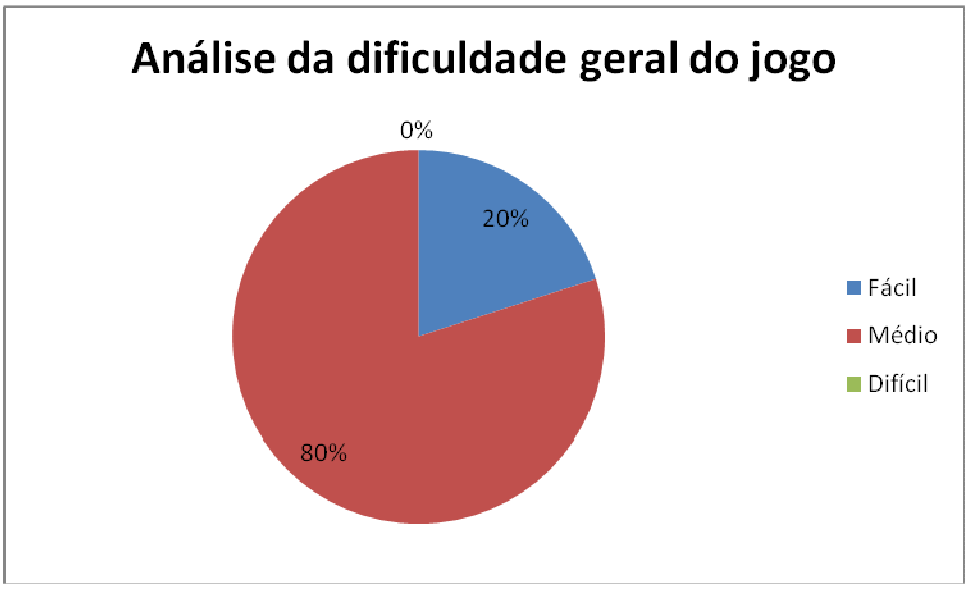
Fonte: Dados de campo, 2009.

Nessa segunda abordagem existe uma alternância maior entre as pizzas consideradas fáceis e as consideradas médias, o que pode ser um bom resultado. A igualdade entre valores no começo se justifica pelo fato de que alguns não vão tão bem na pizza inicial mais simples e recebem uma pizza também fácil em seguida, sendo normal considerá-la assim, outros que se dão melhor, dependendo de quão bem foram pode dar saltos nos níveis, já considerando a pizza média. Esse comportamento se mostra repetidamente, ou seja, em geral quando o jogador se da bem em uma pizza tende a se dar um pouco pior na seguinte que é mais dificil. O inverso também se mostra verdadeiro, pizzas mais dificeis indicam desempenhos piores que fazem o sistema escolher pizzas mais fáceis em seguida. Com o tempo o nível dos jogadores vai melhorando, aqueles que mantém um bom desempenho acabam no futuro enfrentando pizzas de dificuldades maiores, isso justifca aquele pequeno crescimento do nível dificil no final.

Quando se divide novamente os jogadores em causais e habituais percebe-se que diferentemente do primeiro grupo o puzzle foi pouco citado como um estilo muito jogado entre os jogadores considerados casuais. Alguns jogadores habituais sentiram mais dificuldade no começo, por isso em seguida continuaram recebendo pizzas fáceis. Houve então a alternância entre as pizzas consideradas fáceis e médias em ambos os grupos. Os jogadores de melhor desempenho foram os considerados casuais, sendo membros desse grupo que chegaram a enfrentar dificuldades maiores.

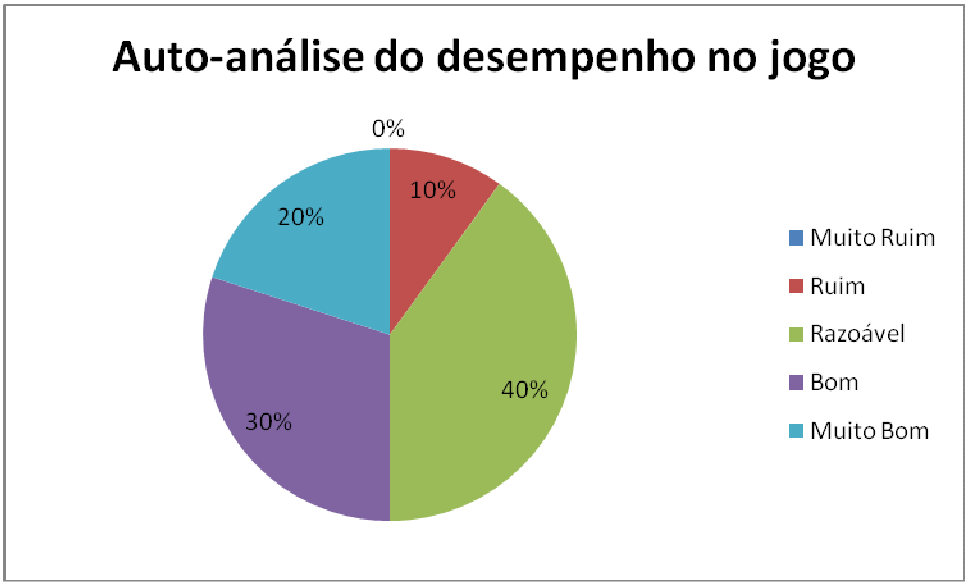
Na análise da dificuldade de forma geral 8 pessoas consideraram o jogo **médio** e 2 pessoas f**ácil**. Dos que consideraram o jogo fácil um é um jogador considerado casual e o outro habitual e seus desempenhos gerais foram equivalentes e bons de acordo com os logs. Ambos têm puzzles como um dos estilos preferidos, o que pode justificar o bom desempenho. O gráfico 6.10 ilustra a divisão de opiniões. O gráfico 6.11 expõe a auto-avaliação do desempenho geral do jogador durante o jogo, novamente os jogadores habituais foram mais rigorosos em relação ao próprio desempenho, porém o a auto-avaliação nessa abordagem obteve resultados mais otimistas.

Gráfico 6.10: Análise da dificuldade geral do jogo (abordagem 2)



Fonte: Dados de campo, 2009

Gráfico 6.11: Auto-análise do desempenho geral no jogo (abordagem 2)



Fonte: Dados de campo, 2009

Essa segunda abordagem apresentou de maneira geral dados mais animador, podendo ser considerada boa. A alternância entre pizzas de dificuldades médias e fáceis é esperada, sendo considerada algo bom [Falstein, 2004], pois os saltos de dificuldade são relativamente discretos. O artifício da escala então parece ser um medidor melhor de desempenho do que uma ascensão gradual que pode levar os jogadores a sentirem mais dificuldades mais rápido do que deveriam.

Essa abordagem também confirma a necessidade de um ordenamento de dificuldade mais automatizado, pois mesmo sendo mais sutil, apesar de poder saltar níveis de dificuldade, alguns jogadores se sentiram frustrados ao considerar que voltaram para um nível de dificuldade menor, quando na verdade de acordo com a classificação manual das pizzas por dificuldade o nível havia sido mantido.

7. Conclusões e trabalhos futuros

Este trabalho apresentou um estudo sobre as principais considerações de projeto para realizar o balanceamento dinâmico de jogos presentes na literatura. Além disso, um estudo das principais técnicas existentes para realizá-lo.

Na pesquisa é levantada a questão de como fazer o balanceamento dinâmico no caso de jogos em que as ações para o balanceamento só podem ser realizadas em momentos muito específicos, geralmente após o fim de um desafio anterior e que os resultados do jogador no desafio só podem ser avaliados também de maneira lenta, através de *checkpoints* ou quando o desafio é encerrado. Essas limitações existem para evitar que a experiência do usuário seja prejudicada, caso ele perceba ajustes de dificuldade durante o jogo.

Para resolver o problema foram propostas duas técnicas com características similares, porém com diferentes processos de escolhas de desafios e avaliação do grau de dificuldade sentido pelo jogador.

A técnica chamada escolha linear de desafios apresentou um desempenho razoável, conseguindo fazer com que muitos desafios fossem classificados como de dificuldade mediana, considerada a ideal. Porém nessa técnica pizzas consecutivas tiveram suas dificuldades avaliadas com extremos. Isso ocorreu porque o bom desempenho em um desafio já colocava o jogador imediatamente em um nível mais elevado, mesmo que seu histórico fosse ruim. Às vezes nesse nível mais elevado o jogador se dava mal e com a pouca quantidade de desafios do jogo do estudo de caso, o retorno a um nível fazia com que um desafio pudesse ser repetido, o que geralmente o fazia ser classificado como fácil. Esse comportamento ás vezes frustrava jogadores mais experientes.

A outra técnica foi chamada de escolha de desafios com possibilidade de saltos de níveis. Apesar do nome essa técnica se mostrou mais eficiente em frear o crescimento da dificuldade, pois o uso da escala de medida da habilidade fazia com que os jogadores demorassem um pouco mais em níveis mais baixos devido a um mau histórico inicial, mesmo se dando bem no último desafio. Os desafios mais difíceis só começaram a aparecer mais tardiamente. O número limitado de desafios também desagradou alguns jogadores nessa abordagem, assim como o perceptível decréscimo na dificuldade em alguns casos.

Um problema comum em ambos foi que desafios considerados com um nível similar de dificuldade pelo *game designer* às vezes não foram considerados da mesma maneira pelos jogadores. Isso mostra que a ordenação automática de desafios é um bom trabalho futuro para o aprimoramento dessas técnicas. As técnicas apresentadas na literatura talvez possam sim ser aproveitadas nesse sentido, porém serão necessárias modificações além de estudos mais profundos.

Além da ordenação automática, outras técnicas para escolha de desafios e avaliação das dificuldades podem ser propostas, inclusive considerando o uso de técnicas mais sofisticadas de inteligência artificial.

Por fim, de uma maneira geral o trabalho apesar de simples apresentou bons resultados, mostrando-se promissor no sentido de melhorar a experiência do usuário ao jogar, considerando que o nível adequado de desafios proporciona mais diversão.

Referências Bibliográficas

[Andrade et al, 2005] ANDRADE, G., RAMALHO, G. L., SANTANA, H., CORRUBLE, V. **Challenge-Sensitive Action Selection: an Application to Game Balancing.** In: IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games, 2005. v. 1. pp. 7-12, Edinburgh, 2005.

[Andrade, 2006] ANDRADE, G. D. **Balanceamento Dinâmico de Jogos: Uma A bordagem Baseada em Aprendizagem por Reforço**. Dissertação (Mestrado em Ciência da Computação) - Centro de Informática. Universidade Federal de Pernambuco. Recife, 2006.

[Cadwel, 2009] CADWEL, T. **Techniques for achieving play balance.** Disponível em: <http://www.gamedev.net/reference/design/features/balance/>. Acesso em: Novembro de 2009.

[Chomsky, 1965] CHOMSKY, N. **Aspects of the Theory of Syntax**, Massachusetts, The MIT Press, 1965.

[Crosby, 1979] CROSBY, P. **Quality is Free: the art of making quality certain**. McGraw-Hill, Nova York, 1979.

[Demasi e Cruz, 2002a] DEMASI, P.; CRUZ, A. **Online Coevolution for Action Games**. In: *Proceedings of The 3rd International Conference on Intelligent Games And Simulation*, pp.113-120, Londres, 2002.

[Demasi e Cruz, 2002b] DEMASI, P.; CRUZ, A. **Algoritmos Coevolucionários Cooperativos em Jogos**. In: WJogos 2002, 1st Brazilian Workshop in Games and Digital Entertainment, Fortaleza, 2002.

[Demasi e Cruz, 2003] DEMASI, P. e CRUZ, A. **Evolução de Agentes em Tempo Real para Jogos Eletrônicos de Ação**. In: II Workshop Brasileiro de Jogos e Entretenimento Digital, 2003.

[Falstein, 2004] FALSTEIN, N. **The Flow Channel**. *Game Developer Magazine*, Vol. 11, N. 05, 2004.

[Hunicke e Chapman, 2004] HUNICKE, R.; CHAPMAN, V. **AI for Dynamic Difficulty Adjustment in Games**. In: *Challenges in Game Artificial Intelligence AAAI Workshop*, pp. 91-96, San Jose, 2004.

[Koster, 2004] KOSTER, R. **Theory of Fun for Game Design**, Phoenix: Paraglyph Press, 2004.

[Lorenzon, 2008] LORENZON, F. **Jogabilidade I – Introdução,** 2008. Disponível em <http://www.cubagames.com.br/jogabilidade-i-%E2%80%93-introducao/>. Acesso em: 13/08/2009.

[Manifesto, 2009] **Manifesto Game Studio**. Disponível em: <http://manifestogames.com.br/>. Acesso em: Novembro de 2009.

[Mendonça, 2009] MENDONÇA, V. G. **Jogos Casuais.** Disponível em: <http://www.pontov.com.br/site/index.php?option=com_content&view=article&id=62:jogos-casuais-versus-hardcore&catid=44:gdbasico&Itemid=59>. Acesso em: Novembro de 2009.

[Olesen, Yannakakis e Hallam, 2008] OLESEN, J. K.; YANNAKAKIS, G. N.; HALLAM, J. **Real-time challenge balance in an RTS game using rtNEAT**, In: *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pp. 87-94, Perth, 2008.

[Rollings e Adams, 2003] ROLLINGS, A.; ADAMS, E.; **On Game Design,** New Riders Publishing, 2003.

[Russel e Norvig, 2004] RUSSEL, S.; NORVIG P. **Artificial Intelligence A Modern Approach.** Upper Saddle River: Prentice Hall, 2004. 1081 p.

[Simpson, 2001] SIMPSON, Z. ***The In-game Economics of Ultima Online.*** In: Game Developers Conference: 2000, San Jose, 2000.

[Spronck et al, 2005] SPRONCK, P.; PONSEN, M.; SPRINKHUIZEN-KUYPER, I.; POSTMA, E. **Adaptive Game AI with Dynamic Scripting.** *Machine Learning*, Vol. 63, No. 3, pp. 217-248. 2005.

[Stanley e Miikkulainen, 2002] STANLEY, K.; MIIKKULAINEN, R. **Evolving neural networks through augmenting topologies,** Evolutionary Computation, vol. 10, n. 02, pp.99–127, 2002.

[Sutton e Barto, 1998] SUTTON, R.; BARTO, A. **Reinforcement Learning: An Introduction**. Massachusetts: The MIT Press, 1998. 322 p.

[Sweetser e Wyeth, 2005] Sweetser, P., e Wyeth, P. **GameFlow: a Model for Evaluating Player Enjoyment in Games**. *ACM Computers in Entertainment*, Vol. 3, No. 3, Julho, 2005.

[Teknomo, 2006] TEKNOMO, K. **Similarity Measurement.** Disponível em <http:\\people.revoledu.com\kardi\ tutorial\Similarity\>. Acesso em: Novembro de 2009.

[Tonietto, 2007] TONIETTO, L. **Técnicas de Balanceamento de Jogos.** Disponível em <http://www.inf.unisinos.br/~ltonietto/jed/taj/tbj2008_01.html>. Acesso em: Novembro de 2009.

[TorqueX, 2009] **TorqueX 2D.** Disponível em: <http://www.garagegames.com/products/torquex-2d>. Acesso em: Novembro de 2009.

[Tozour, 2002] TOZOUR, P. **The Evolution of Game AI.** In S. Rabin (ed.): AI Game Programming Wisdom. Charles River Media, Inc. 2002.

[Watkins e Dayan, 1992] WATKINS, C.; DAYAN, P. **Q-learning**. *Machine Learning*, 8(3):279–292, 1992.

[Wikipedia, 2009] Wikipedia. **Quebra-Cabeça.** Disponível em: <http://pt.wikipedia.org/wiki/Quebra-cabe%C3%A7a>. Acesso em: Novembro de 2009.

[XNA, 2009] **XNA.** Disponível em: <http://creators.xna.com/en-US/>. Acesso em: Novembro de 2009.

[Yannakakis e Hallam, 2005] YANNAKAKIS, G. N. e HALLAM, J. **A Generic Approach for Generating Interesting Interactive Pac-Man Opponents**. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pp. 94-101, Essex University, 2005.

Apêndice A

Questionário do Pizzaiolo

O seguinte questionário tem o único objetivo de avaliar as técnicas aplicadas no jogo Pizzaiolo e não o jogador!

Antes de começar a jogar, por favor, responda as questões relativas as suas caracteristicas pessoais.

Como jogar:

Um sabor de pizza e seu tamanho serão escolhidos e indicados no canto superior esquerdo da tela. Você deve clicar no tamanho da massa correspondente ao que foi pedido e em seguida olhar no livro de receitas (canto inferior esquerdo) que ingredientes pertencem aquele sabor de pizza e selecioná-los em qualquer ordem.

O objetivo do jogo é montar um pizza colocando todos seus ingredientes que estão espalhados nos lugares corretos no menor tempo possível. O lugar correto para um ingrediente é aquele cujo formato é exatamente o formato do ingrediente. Colocando o mouse sobre um ingrediente que está no lugar incorreto você pode ver o formato do ingrediente que deveria estar ali.

Para trocar os ingredientes você deve clicar em dois deles. Se pelo menos um deles for o ingrediente se encaixa no lugar do outro a troca é realizada e o ingrediente que foi para o lugar correto desaparece da pizza, gerando uma pontuação.

---

*Obrigatório

Parte 1: Características Pessoais

O objetivo dessa parte é identificar o perfil dos jogadores que realizaram o teste.

Qual a sua idade? *

- ☐ menos de 16 anos
- ☐ de 16 a 27 anos
- ☐ de 28 a 39 anos
- ☐ mais de 39 anos

Qual o seu sexo? *

- ☐ Masculino
- ☐ Feminino

Você se considera um(a) jogador(a): *

- ☐ Casual
- ☐ Habitual (Hardcore)

Quantas horas por semana costuma jogar? *

- ☐ Menos de 2
- ☐ de 2 a 8
- ☐ 9 a 15
- ☐ 16 a 21
- ☐ Acima de 21

Que tipos de jogos costuma jogar? *

- ☐ Ação
- ☐ Esportes
- ☐ Luta
- ☐ Puzzles (Quebra cabeças)
- ☐ Aventura
- ☐ RPG

- ☐ FPS
- ☐ Simulação
- ☐ Outro: _____

Como você avalia sua habilidade jogando? *

- ○ Muito ruim
- ○ Ruim
- ○ Razoável
- ○ Boa
- ○ Muito Boa

Como é sua habilidade em jogos do tipo Puzzle? *

- ○ Muito ruim
- ○ Ruim
- ○ Razoável
- ○ Boa
- ○ Muito Boa

Quantas horas por semana costuma jogar esse tipo de jogo? *

- ○ Menos de 2
- ○ de 2 a 6
- ○ de 7 a 12
- ○ Mais de 12

Parte 2: Avaliação do jogo

Para jogar, escolha a opção play e em seguida clique no modo free Play. Você deve jogar seis pizzas, para cada pizza concluída você deve responder a

pergunta correspondente, indicando o nome da pizza (escrito no canto superior esquerdo da tela) incluindo seu tamanho e em seguida responder o que você achou dessa pizza. Sinta-se a vontade para jogar mais de seis se desejar, no entanto não é necessário para o questionário. IMPORTANTE: Você NÃO deve responder todas as questões de 1 a 11. Se a sua abordagem for 1, responda as questões de 1 a 6. Se for 2, responda a 1 e em seguida de 7 a 11.

Abordagem (A abordagem é indicada pelo avaliador)

- ☐ Abordagem 1
- ☐ Abordagem 2

1- Nome da pizza: *

1- Você considerou a pizza: *

- ☐ Fácil
- ☐ Média
- ☐ Difícil

2- Nome da pizza:

2 - Você considerou a pizza:

- ☐ Fácil
- ☐ Média
- ☐ Difícil

3 - Nome da pizza:

3 - Você considerou a pizza:

- ☐ Fácil

- ☐ Média
- ☐ Difícil

4- Nome da pizza: _____

4- Você considerou a pizza:

- ☐ Fácil
- ☐ Média
- ☐ Difícil

5- Nome da pizza: _____

5- Você considerou a pizza:

- ☐ Fácil
- ☐ Média
- ☐ Difícil

6- Nome da pizza: _____

6- Você considerou a pizza:

- ☐ Fácil
- ☐ Média
- ☐ Difícil

7- Nome da pizza: _____

7- Você considerou a pizza:

- ☐ Fácil

- ☐ Média
- ☐ Difícil

8- Nome da pizza: _____

8- Você considerou a pizza:

- ☐ Fácil
- ☐ Média
- ☐ Difícil

9- Nome da pizza: _____

9- Você considerou a pizza:

- ☐ Fácil
- ☐ Média
- ☐ Difícil

10- Nome da pizza: _____

10- Você considerou a pizza:

- ☐ Fácil
- ☐ Média
- ☐ Difícil

11- Nome da pizza: _____

11- Você considerou a pizza:

- ☐ Fácil
- ☐ Média

- Difícil

Como você avalia seu desempenho no jogo? *

- Muito ruim
- Ruim
- Razoável
- Bom
- Muito Bom

Como foi a dificuldade geral do jogo? *

- Fácil
- Médio
- Difícil

Como foi a experiência de jogar o Pizzaiolo? *

- Muito ruim
- Ruim
- Regular
- Boa
- Muito Boa

Sinta-se a vontade para fazer qualquer comentário que desejar