Universidade Federal de Pernambuco
Centro de Informática

Graduação em Engenharia da Computação

Análise e Predição de Dificuldade de Fases em Jogos

Délio Lustosa Cantarelli

Trabalho de Graduação

Recife
11 de Julho de 2017

Universidade Federal de Pernambuco
Centro de Informática

Délio Lustosa Cantarelli

Análise e Predição de Dificuldade de Fases em Jogos

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Graduação em Engenharia da Computação.

Orientador: *Geber Lisboa*
Co-orientador: *Túlio Caraciolo*

Recife
11 de Julho de 2017

Dedico este trabalho à minha mãe
que sempre apoiou minhas decisões
e ajudou no meu crescimento.

Agradecimentos

Agradeço primeiramente a toda minha família, não apenas minha mãe e meus irmãos, mas a todos meus tios, tias, primos e primas, que sempre estiveram comigo e me deram suporte durante toda a vida.

Agradeço a todos os meus amigos que estavam comigo tanto nos bons momentos quanto nos ruins. Em especial à minha amiga Daniele Soares, que sempre me ajudou em várias questões da faculdade com muita paciência e que compartilhou comigo várias experiências únicas.

Agradeço a minha namorada Gessianni Alves, que sempre me deu suporte e por todo esforço e compreensão que exigi durante esse tempo de desenvolvimento.

Agradeço a meu orientador Geber Ramalho, por toda dedicação e ajuda durante o trabalho nessa pesquisa.

Agradeço a meu co-orientador Tulio Caraciolo, pela confiança, paciência e ajuda necessárias para concluir essa pesquisa.

Por fim gostaria de agradecer ao Centro de Informática, por proporcionar uma excelente infra-estrutura de aprendizado aos estudantes.

Resumo

Todo jogo tem como objetivo fazer seus usuários aproveitarem a experiência o máximo possível. Para que o jogador não se sinta frustrado com o jogo, é essencial que a dificuldade experimentada não seja nem elevada demais, nem baixa demais para ele. Mas como as pessoas têm habilidades e preferências de dificuldades diferentes, configurar uma dificuldade que seja ideal para todos é inviável. Para que a configuração da dificuldade seja possível, é necessário que haja uma classificação da dificuldade que o jogador está sentindo durante o jogo, assim como uma classificação da dificuldade que o jogo possui. O objetivo dessa pesquisa é classificar automaticamente as fases de um jogo em diferentes classes de dificuldades, de acordo com a dificuldade que o jogador sentiu ao jogá-la. Para estimar a dificuldade que o jogador sentiu, foi proposto uma heurística e então avaliado se esta heurística é uma boa estimativa para a dificuldade. Para alcançar o objetivo, foi usado uma rede neural para classificar as fases do jogo Geppetto Builder, tendo como classificação prévia a dificuldade estimada pela heurística. O experimento demonstrou que a heurística serve como uma estimativa da dificuldade e possui uma taxa de acerto de 72.727% para a classificação das fases.

Palavras-chave: Video Games, Estimativa, Dificuldade, Aprendizado, Heurística

Abstract

Every game has as objective making its players enjoy the experience at most as possible. To make the player not feel frustrated with the game, it is essential that the difficulty experienced is not too high nor too low for him. But as every person has different skills and difficulty preferences, set a difficulty level that is ideal for everyone is too costly or impossible. For the difficulty setting to be possible, there should be a classification for the difficulty the player is feeling during the game, likewise a classification for the difficulty the game has. The aim of this research is to automatically classify the levels in a game in different difficulty classes, according to the difficulty the player felt when playing it. To estimate the difficulty the player felt, it was introduced an heuristic to then evaluate if this heuristic is a good estimative for the difficulty. To reach our goal, it was used a neural network to classify the levels in the game Geppetto Builder, having a previous difficulty estimative from the heuristic proposed. The experiment has shown that the heuristic serves as a good estimative for the difficulty and has a success rate of 72.727% for the levels classification.

Keywords: Video Games, Estimate, Difficulty, Learning, Heuristic

Sumário

1 Introdução **1**
- 1.1 Motivação 1
- 1.2 Objetivo 4
- 1.3 Abordagem 5
- 1.4 Estrutura do Trabalho 6

2 O Problema **9**

3 Estado da arte **13**

4 Método **17**
- 4.1 Procedimento 17
- 4.2 Caso de Estudo 18
- 4.3 Categorização das Fases 19
- 4.4 Determinando as Heurísticas 22
- 4.5 Características da fase 23
- 4.6 Experimento 24
  - 4.6.1 Coleta dos Dados 24
  - 4.6.2 Construção do Classificador 25

5 Resultados e Análise **27**
- 5.1 Resultados 27
- 5.2 Análise dos Resultados 28

6 Conclusões **29**

Lista de Figuras

4.1 Exemplo de uma fase do jogo Geppetto Builder 20
4.2 Modelo da rede neural 26

Lista de Tabelas

5.1 Matriz de confusão gerada pela rede neural. 28

CAPÍTULO 1

Introdução

Neste capítulo será discutido a motivação inicial da pesquisa, o objetivo visado, a abordagem tomada para a realização desse objetivo e como esse trabalho está estruturado.

1.1 Motivação

Todo jogo tem como objetivo fazer seus usuários aproveitarem a experiência, enquanto o jogo conseguir despertar sensações boas no usuário, ele continuará jogando. Isso se deve ao fato de que todas as pessoas são movidas a fim de encontrar felicidade e prazer, então, para desenvolver e usar técnicas direcionadas para essas finalidades, é necessário entender primeiramente como funciona a felicidade [Csi13].

Para explicar a felicidade e como ela funciona, foi criado o conceito de fluxo (flow) pelo psicólogo e professor croata Mihaly Csikszentmihalyi na década de 1970. Fluxo, segundo ele, é o estado mental em que a pessoa está focada em uma atividade com alto grau de satisfação e prazer. Quando uma pessoa está nessa experiência, ela tem uma maximização do foco na tarefa e esquece preocupações ou a passagem do tempo [Che07].

O conceito de fluxo foi desenvolvido quando Csikszentmihalyi perguntou à diversas pessoas para descrever as experiências de quando elas achavam que estavam o mais

feliz possível. Ele então percebeu que as experiências possuíam características que ele definiu como o estado de fluxo [IDKP+07].

A experiência de quando uma pessoa está jogando e imersa no jogo é igual ao estado de fluxo, e isso faz com que esse conceito seja cada vez mais utilizado na área [Che07]. Fazer os jogadores entrar no estado de fluxo é essencial para que eles aproveitem ao máximo a experiência que o jogo tem a oferecer, por isso é importante saber como despertar o estado de fluxo nos jogadores.

Csikszentmihalyi descreve 8 componentes mais importantes do fluxo:

- Uma atividade desafiadora que requer habilidade;
- Uma mistura de ação e atenção
- Objetivos claros
- Feedbacks diretos e imediatos
- Concentração na tarefa atual
- Uma sensação de controle
- Uma perda de percepção própria
- Uma percepção de tempo alterada

Não é necessário todos os elementos para que uma pessoa entre no estado de fluxo [Che07], mas a indústria de jogos vem criando técnicas para usar mais elementos e assim melhorar a imersão do jogador. Nesse projeto, o elemento que mais interessa é a atividade desafiadora que requer habilidade. Quão desafiadora a atividade deve ser para que o fluxo seja estabelecido? Qual a dificuldade em que ela deixa de ser desafiadora? Ou com qual dificuldade ela passa a ser difícil demais?

Para se manter na zona de fluxo, é necessário que os desafios do jogo sejam adequados para a habilidade que o jogador possui. Caso o desafio seja muito maior que as

habilidades dele, ele sentirá ansiedade e irá sair da zona de fluxo. No caso de a dificuldade ser muito baixa, a pessoa não sentirá estímulo e com isso tédio. Ambos os casos levam o jogador à frustração e fazem com ele deixe de jogar.

Manipular a dificuldade de uma jogo para que ele mantenha o jogador na zona de fluxo é difícil, e ainda há o problema que diferentes pessoas possuem diferentes zonas de fluxo. Jogadores mais experientes possuem mais habilidades e por isso requerem desafios à altura de suas habilidades, já os jogadores iniciantes, que não possuem muitas habilidades, os desafios devem ter uma dificuldade mais baixa.

Descobrir, então, como estimar a dificuldade é de importância para que seja possível manter um jogador na zona de fluxo. Sabendo a dificuldade que o jogador está sentindo no momento, é possível ajustá-la para que o desafio fique à nível das habilidades do jogador, assim ele não sentirá ansiedade por ser muito difícil, ou tédio caso seja muito fácil, fazendo a experiência se manter na zona de fluxo.

Para fazer ajustes de dificuldades, existem várias técnicas aplicadas atualmente, a mais comum é permitindo os jogadores escolher a dificuldade e dividi-las em níveis, como fácil, médio e difícil. O problema dessa divisão é que dificilmente um nível de dificuldade será exatamente a dificuldade necessária para que o jogador se mantenha no fluxo, ou seja, ele pode ter as habilidades necessárias para completar no fácil, sendo baixa a dificuldade para as habilidades dele, mas na dificuldade média, a dificuldade pode ser muito elevada, não havendo um meio termo. Para resolver esse problema, é possível aumentar a faixa de dificuldades presentes no jogo, mas assim se torna frustrante para o jogador achar a dificuldade perfeita dentre um grande número de graus de dificuldades [MG09].

Uma maneira eficiente de adaptar a dificuldade do jogo para a mais indicada para o jogador é usando técnicas de *Dynamic Difficulty Adjustment* (DDA). Essas técnicas

propõem modificar os parâmetros do jogo automaticamente, de forma que os desafios fiquem próximos ao nível das habilidades do jogador. Para vários jogos esses parâmetros são conhecidos e facilmente manipuláveis, como exemplo o dano causado por uma arma, a vida máxima que um adversário possui ou o tempo limite para completar uma fase [Hun05].

No caso de um jogo de puzzle sem condições de perda, não há esses parâmetros para serem manipulados, pois uma fase é imutável. Então a estimação da dificuldade é essencial para ser usada como parâmetro, manipulando assim qual fase a ser indicada, escolhendo aquela mais indicada de acordo com a dificuldade possuída.

Nosso grupo de pesquisa propôs com sucesso um modelo para classificar a dificuldade de desafios para jogos de *endless running*, que são jogos cujo único objetivo é conseguir a pontuação mais elevada possível. Mas o modelo proposto é limitado ao tipo de jogo do caso de estudo [dC12].

Esse estudo visa estender o método proposto por Carvalho [dC12], aplicando a técnica de classificação proposta para outro caso de estudo, que será os jogos do tipo puzzle. Esse estudo também

1.2 Objetivo

O objetivo dessa pesquisa é formular uma maneira automática para estimar a dificuldade de uma fase em um jogo de puzzle a partir das características que a fase possui. As fases usadas para a estimação serão as fases do jogo Geppetto Builder.

Para estimar a dificuldade, é necessário avaliar as características que um level possui

e estudar o impacto de cada uma delas na dificuldade da fase. Também é necessário ter uma avaliação prévia da dificuldade das fases, e para isso dados foram coletado das crianças jogando de uma maneira não intrusiva, pois esse dado é importante para a indicação de uma fase de acordo com as habilidades dela, e durante o jogo é necessário ser o menos intrusivo possível.

A estimação automática da dificuldade de uma fase é uma ferramenta poderosa quando associada com geração procedimental de níveis ou indicação automática de fases de acordo com o desempenho do jogador. Assim que um level é gerado procedimentalmente, é necessário que ele seja classificado, e uma maneira de fazer isso é ordenando-os por dificuldade associada. Para a indicação automática, é necessário ter uma função que leve do desempenho que o jogador está tendo no jogo para um valor de dificuldade, assim é possível indicar qual a fase que tenha uma dificuldade mais próxima desse valor. Ambos os pontos estão fora do escopo desta pesquisa, são apenas importantes utilizações de uma estimação automática de dificuldade.

1.3 Abordagem

Propomos estender a técnica usada por Carvalho [dC12], para a classificação da dificuldade. Essa técnica consiste em uma maneira automática de estimar a dificuldade que o jogador teve com elementos do jogo. A dificuldade é etiquetada com um valor estático que corresponde a uma estimativa do quão difícil a fase é para um jogador e pode assumir os valores de fácil, médio ou difícil.

Para aplicar a técnica em outro caso de estudo, é necessário adaptar a maneira de etiquetagem e classificação da proposta, pois a jogabilidade e as características que

influenciam a dificuldade dos jogos tem elementos diferentes.

Para etiquetar as fases em classes de dificuldades, é necessário encontrar uma heurística que melhor represente a dificuldade que o jogador teve ao jogar uma fase. Já para a classificação, é necessário encontrar quais as características da fase que influenciam na dificuldade.

Usamos uma rede neural para fazer a classificação da dificuldade de uma fase, e analisamos a porcentagem de fases em que, após dadas como entrada na rede, resultaram em uma predição equivalente à heurística associada a fase.

1.4 Estrutura do Trabalho

Esse trabalho está dividido em 6 capítulos. No capítulo 2, discutimos o problema existente em selecionar uma dificuldade mais indicada para o jogador, também discutimos a importância de saber uma estimativa da dificuldade nos jogos. O capítulo 3 faz uma revisão da literatura, para entender as abordagens já seguidas para resolver os problemas mencionados.

No capítulo 4, será detalhado o método seguido para realizar essa pesquisa. Falaremos sobre o jogo usado, como foi determinado a heurística que servirá como uma estimativa da dificuldade e como o valor dela foi obtida durante o jogo. Também discutiremos as características de uma fase que podem influenciar na dificuldade. Em seguida será detalhado o experimento, o processo da coleta de dados para estimar a dificuldade e então como a rede neural usada foi implementada.

Em seguida, no capítulo 5, será apresentados os resultados obtidos e então uma aná-

lise destes resultados. No capítulo 6 será feita as conclusões finais e possíveis trabalhos futuros.

CAPÍTULO 2

O Problema

Este capítulo apresenta o problema presente atualmente, sobre a dificuldade em aplicar técnicas de ajuste dinâmico de dificuldade e também para a estimação de dificuldades em um jogo. Em seguida avaliamos avaliamos a dificuldade no processo de geração procedimental de fases e o porquê a estimação de dificuldades é importante para ele.

Manter a dificuldade dos desafios a par com as habilidades do jogador é um desafio para o desenvolvedor. Com o ajuste dinâmico de dificuldade, é possível fazer o jogador seguir uma curva de dificuldade que o *game designer* ache que seja a melhor para o jogo, mas fazê-la não é uma tarefa simples. Modificar a curva exige entender qual parâmetros pertencentes ao jogo influencia na dificuldade e quais desses parâmetros, assim que alterados, oferecem a maior diversão para o jogador, o que não é uma tarefa fácil.

Outro problema para manter o jogador na curva de dificuldade, é que diferentes pessoas com mesma experiência no jogo podem preferir níveis de dificuldade diferentes para essa experiência. Por exemplo, uma pessoa que quer descansar pode preferir uma dificuldade que não exija total concentração para jogar, enquanto uma pessoa que queira testar suas habilidades vai preferir uma dificuldade que exija diversos erros e perdas para ganhar [KL12].

Então, mesmo que ambos os jogadores estejam na curva definida pelo *game designer* com o valor da dificuldade para a habilidade possuída, eles não terão a experiência que

desejam.

Para o uso de técnicas de ajuste dinâmico de dificuldade, é necessário que haja uma maneira de estimar a dificuldade do jogo, para que assim a dificuldade possa ser ajustada. Mas estimar a dificuldade é algo que depende de vários fatores no jogo, já esses fatores são dependentes do gênero do jogo. Aplicar um método de ajuste dinâmico de dificuldade em um outro gênero de jogo exige estudar as características nele, não podendo ser aplicada diretamente.

Dentre os vários gêneros de jogos, essa pesquisa se limita apenas a estudar ao gênero de puzzle. Puzzle se refere à jogos não apenas digitais, em que o objetivo é completar fases usando conhecimento e ingenuidade, ou seja, aplicar ideias para resolver problemas e desafios. Uma fase de um jogo de puzzle é composta por um conjunto de desafios, onde é necessário que o jogador descubra os passos para resolvê-los, e assim completar a fase. Após completá-la, é possível ir para uma outra fase, de preferência com desafios mais difíceis.

Após o jogador descobrir os passos ou conhecimentos necessários para se resolver uma fase, não é ideal que ela seja indicada para o jogador novamente, pois ele já saberá como resolver e não haverá mais desafio, ou seja, ele sentirá tédio. Por essa razão que a geração de fases é um processo importante para o jogo, é necessário ter um amplo repertório de fases para que os jogadores possam se entreter pelo máximo de tempo possível.

A geração de fases hoje em dia é normalmente feita manualmente por um *game designer*, são eles que fazem cada uma e estimam qual a dificuldade que a fase representa, colocando um a um os desafios da fase de forma que fique o mais interessante que ele achar. Os problemas desse processo é que ele é ocupa muito tempo do *game designer*, principalmente na estimação da dificuldade, que compõe maior parte do custo, e tam-

bém que esse processo apresenta falhas por ser baseado na intuição de quem gera as fases [VKLM15] [dC12].

Como maneira de facilitar o processo de geração de fases, existe a geração procedimental de conteúdo. Com isso é possível gerar mais conteúdo do que seria possível apenas com a geração manual. Outra vantagem é que o conteúdo não é tão engessado quanto o conteúdo gerado a mão, ou seja, é mais facilmente adaptável para o perfil de diferentes jogadores [SPD11].

Após a geração de uma fase, é essencial a estimação da dificuldade que ele representa, pois é necessário que ele seja classificado, seja para ordenação das fases ou indicação para diferentes perfis de jogadores. Isso quer dizer que para a automação do processo de gerar fases, ainda é de importância ter técnicas para saber uma estimativa da dificuldade. Alguns processos de geração procedimental também são dependentes de uma métrica de dificuldade nas fases para que o algoritmo diferencie o difícil de trabalhoso [TP11].

CAPÍTULO 3

Estado da arte

Este capítulo apresenta técnicas existentes para a estimação de dificuldade em um jogo. O foco é em técnicas para jogos do gênero puzzle, mas também será apresentado soluções para jogos de outros gêneros.

Horn e Volz estabelecem uma relação entre dificuldade e o desempenho de vários tipos de técnicas de inteligência artificial, para jogar vários tipos de jogos diferentes, chamados de *controllers*. Eles estimam a dificuldade dos jogos de acordo com várias características que eles definem, mas a estimação é feita manualmente. Com essas características da dificuldade, eles preveem a chance de vitória de um controller para um jogo e descobrem qual seria o *controller* mais indicado para esse jogo [HVPLP16].

Ashlock e Schonfeld estudam a dificuldade de diferentes fases do jogo de Sokoban. Eles estabelecem uma relação entre o tempo médio e a chance de falha do algoritmo usados com a dificuldade que a fase representa. Eles utilizam essa medida de dificuldade encontrada para então ordenar as fases do jogo [AS10].

Jaru¨ek e Pelánek também trabalham com dificuldade no jogo de Sokoban, mas eles utilizam dados coletados de pessoas jogando para ter uma medida da dificuldade e então poder estimá-la [JP10].

Mantere e Kojinen usam um algoritmo genético com três finalidades, que são de resolver um configuração de um jogo de sudoku, gerar uma nova configuração, e avaliar a dificuldade que essa configuração gerada possui. Eles propõem que uma configuração

de sudoku considerada difícil para um humano, também é difícil para o algoritmo, e estabelecem uma relação entre a dificuldade da configuração com o tempo que o algoritmo leva para resolvê-la. Eles concluem que os resultados confirmam a teoria deles [MK07].

Tijs, Brokken e IJsselsteijn afirmam que para adaptar a dificuldade de um jogo, não se deve apenas depender da performance que o jogador está tendo, pois cada jogador tem suas próprias razões para jogar e as emoções deles devem ser levadas em conta. Então, para encontrar a dificuldade ideal do jogo para cada indivíduo, eles avaliam o comportamento através de sensores e o jogo modifica a dificuldade de acordo, para que o jogador não sinta frustração ou tédio [TBI08].

Guid e Bratko estimam a dificuldade em um jogo de xadrez a partir da profundidade na busca pela melhor solução, levando em conta a configuração do tabuleiro de xadrez. Eles argumentam que a dificuldade está no esforço necessário para se analisar o resultado de cada uma dessas possibilidades e escolher a melhor dentre elas [GB13].

Pérez, Calla, Valente, Montenegro e Clua propõem uma solução automática de balancear a dificuldade do jogo em tempo real, usando como parâmetro as ações que o jogador executa no jogo. O método proposto não tem preocupação em quantificar a dificuldade, apenas modifica o estado do jogo para que o jogador tenha a melhor experiência possível [PCV+15].

Kreveld, Löffler e Mutser estimam a dificuldade de diferentes jogos de puzzle usando as características que uma fase possui. Eles usaram uma função para estimar o valor, que pode ser consultado para verificar o impacto de cada característica na função. Para o treinamento da função, eles levaram em consideração apenas a opinião do jogador, sem verificar o porquê de ele achar difícil ou fácil [VKLM15].

Por último, Carvalho propõe uma técnica para geração procedimental para jogos do tipo *endless runner*. A técnica proposta tem como um dos passos a estimação da dificuldade dos elementos presentes no jogo, que também é o foco da pesquisa [dC12].

Esta pesquisa tem como inspiração a técnica desenvolvida por Carvalho [dC12], para adaptar e evoluir essa técnica usando um caso de uso diferente, que será um jogo do gênero de puzzle.

CAPÍTULO 4

Método

Neste capítulo é primeiramente descrito o procedimento utilizado na pesquisa para construir o classificador de fases, então será descrito o jogo que foi usado como caso de estudo, assim como suas regras e como ele funciona, posteriormente é discutido como a dificuldade de cada fase foi definida. Será discutido, também, quais as características da fase que usaremos para prever a dificuldade e o motivo que elas são importantes ou não para a predição. Por último será descrito o experimento, detalhando como foi o processo de coleta de dados e como a rede neural foi implementada.

4.1 Procedimento

O método proposto consiste em uma técnica automatizada usando uma rede neural em que, dado uma fase de um jogo como entrada, devolverá o valor estimado da dificuldade dessa fase. Os valores que a dificuldade podem assumir são fácil, médio e difícil, que servem como as classes das fases.

Usamos uma heurística para estimar a dificuldade e etiquetar as fases em fácil, médio e difícil. Essa estimativa usa dados coletados durante o jogo que representem o máximo possível a dificuldade que os jogadores tiveram durante a partida.

As características que serão usadas na rede neural devem ser descobertas e avaliadas.

É necessário que elas tenham um alto grau de relação com a dificuldade, mas não há forma certa de descobrir, por isso as escolhas são arbitrárias. Importante notar que a rede verifica a relação da característica, então mesmo que para alguma característica não haja uma relação com a dificuldade, ou essa relação seja fraca, o desempenho da rede ainda será satisfatório.

Assim que as características são definidas e todas as fases são etiquetadas usando a heurística definida, o conjunto de fases é dividido em dois subconjuntos, um conjunto de treino e um conjunto de testes. O conjunto de treino é então usado para treinar a rede neural e modelá-la, tornando-a um classificador de dificuldades de fases.

O conjunto de testes servirá posteriormente para a avaliação da precisão da rede modelada.

4.2 Caso de Estudo

O jogo estudado neste projeto se chama Geppetto Builder, e é um jogo infantil para crianças de cinco anos desenvolvido pela produtora *Manifesto Games Studio*. A proposta do jogo é desenvolver na criança capacidades intelectuais e motoras, como a lógica, a noção de espaço e coordenação motora.

O objetivo do jogo é a criança montar uma estrutura com blocos específicos, onde a estrutura é definida pela fase em que está jogando 4.1. O jogador é dado um total de 7 tipos diferentes de blocos em uma esteira para serem usados: um quadrado, um retângulo com o maior lado para baixo, um retângulo com o menor lado para baixo, um triângulo pequeno, uma meia esfera grande, um triângulo grande e uma meia esfera

pequena. Todos os blocos têm altura e largura múltiplos de 100 pixels

Cada fase é composta por uma matriz de espaços que podem ou não conter um bloco, cada espaço da matriz tem 100 pixels de altura e 100 pixels de largura. Um bloco também pode não estar alinhado com essa matriz, o espaço em pixels que ele está fora de alinhamento é chamado de *offset* e pode assumir os valores entre 0 e 99. O objetivo de uma fase é uma relação de cada espaço da matriz com um tipo de bloco e, para que a fase ser completada, todos esses espaços do objetivo precisam ter um bloco do tipo correto na posição. Na verificação, a posição do bloco é reduzida do *offset* que ele possui para alinhar com a matriz. O jogador tem uma margem de erro de 30 pixels para o bloco ser considerado no espaço.

O jogo consiste até então de 53 fases, cada fase possui um conjunto de sombras com formatos de blocos que juntos formam uma estrutura. Para completar uma fase, a criança deve colocar os blocos correspondentes na sombra e completar a estrutura, sem haver nenhum bloco extra no jogo. Caso um bloco seja colocado em jogo erroneamente, ou caso o jogador desista do bloco, é possível jogá-lo para fora da tela, que o bloco será removido e não contará como estando em jogo.

O jogo possui uma física básica, com gravidade e colisão entre os objetos em jogo. O bloco só contará como objeto em jogo assim que ele for solto fora da esteira, o que significa que não há colisão quando o bloco é pego da esteira e ele só ocupará o espaço assim que for solto em jogo.

Figura 4.1 Exemplo de uma fase do jogo Geppetto Builder

4.3 Categorização das Fases

Para a estimação da dificuldade, é necessário encontrar uma heurística que represente o melhor possível esse valor. Algumas heurísticas foram selecionadas arbitrariamente para a execução do projeto, e então, após os resultados, foi selecionado a heurística que obteve o melhor resultado.

Para a escolha da heurística, os seguintes dados foram coletados durante os jogos:

- Número médio total de deslocamento de blocos;
- Porcentagem média de deslocamentos que resultaram no bloco estar em uma posição correta;

- O tempo médio total para completar a fase;
- O tempo médio da soma dos tempos em que o jogador usou para deslocar os blocos;
- O número médio de blocos que começaram a ser deslocados da esteira, mas que foram retornados sem serem colocados em jogo;
- O número médio de blocos que foram removidos de jogo.

Enquanto o número médio de blocos que foram removidos de jogo e o número médio de blocos retornados à esteira não possuem informação suficiente para estimar a dificuldade da fase por si só, eles podem ser usadas como uma combinação com outros dados para ter uma estimativa mais eficiente. Mas essa combinação não será trabalhada nesta pesquisa, usaremos apenas um valor para a heurística e por isso esses dois valores não serão usados.

Dentre os três outros dados restantes, foi levado em consideração a taxa de acerto do preditor usando cada uma das heurísticas. A porcentagem de deslocamentos de blocos corretos foi o dado que teve a maior taxa de acerto, e por isso foi o dado escolhido para fazer a classificação prévia das fases.

Todas as fases foram etiquetadas com três valores possíveis de acordo com a dificuldade estimada resultante da heurística, sendo esses valores fácil, médio e difícil. Para fazer a divisão, foi feito uma ordenação de todos as fases pelo valor da dificuldade estimada resultante, e, de acordo com a posição nessa ordenação ela seria classificada. 25% das fases com menor valor entraram na categoria de fácil, os 25% com maior valor entraram na categoria de difícil e os 50% restantes, que se encontraram no patamar intermediário, na categoria de médio.

4.4 Determinando as Heurísticas

Para coletar as informações durante o jogo, foi adicionado ao jogo um sistema de *analytics*, ou seja, uma maneira para descobrir, analisar e interpretar dados. Após completar uma fase, eventos são enviados para a plataforma Flurry com dados para poder calcular as heurísticas, se uma fase não é completada por qualquer motivo, os eventos não são enviados.

O tempo levado para completar a fase é enviado em um evento assim que a fase é completada. Para calcular a média, o tempo de cada evento é somado e dividido pelo número de eventos.

Para calcular o número médio total de deslocamento de blocos, um evento é enviado para a plataforma toda vez que um bloco é colocado em jogo da esteira, ou um bloco já em jogo é deslocado para mais de cinco pixels. O evento contêm informações como o bloco deslocado, se o bloco foi colocado em uma posição correta e o tempo levado para fazer o deslocamento. O valor médio é computado somando todas as ocorrências do evento e dividindo pelo número de eventos de fases completadas.

Para calcular a porcentagem média de deslocamentos que resultaram no bloco estar em uma posição correta, o mesmo evento do deslocamento de blocos é usado, mas o número de eventos que resultaram em uma posição correta é somado e dividido pelo número de eventos total.

O tempo médio em que o jogador passou deslocando blocos é computado usando o tempo no evento de deslocamento de blocos somados e divididos pelo número de eventos de fases completadas.

O número médio de blocos removidos do jogo é enviado em um evento no fim da

fase, contendo um valor resultante do número de blocos contidos na solução subtraído do número total de blocos colocados em jogo. Para cada bloco que foi retornado à esteira, um evento é enviado ao fim da fase, assim, somando todos os eventos e dividindo pelo número de eventos de fases completadas é possível calcular o número médio de blocos retornados à esteira.

4.5 Características da fase

Para determinar a dificuldade de uma fase do jogo Geppetto Builder, usamos a seguinte lista de características presentes em uma fase:

- Número de blocos presentes na solução: quanto maior o número de blocos, maior deve ser a dificuldade de colocá-los juntos, pois um movimento errado pode derrubar toda a estrutura montada.
- Número de blocos diferentes: supomos que quanto mais opções possíveis, mais confuso será a fase. Alguns blocos são parecidos, o que aumenta ainda mais a confusão.
- Altura da estrutura: quanto mais alta a estrutura for, mais instável se torna e mais fácil de derrubá-la.
- Largura da fase: embora não seja tão marcante quanto a altura, colocar um bloco ao lado de outro precisa da interação entre os blocos e a colisão pode derrubar a estrutura.
- Alinhamento dos blocos: o alinhamento dos blocos tem relação com a estabilidade da estrutura.
- Área interna não ocupada: é área não ocupada de um bloco no espaço de 100

pixels quadrados, mas o espaço tem que ter todos os espaços adjacentes ocupados. Essa área não ocupada tem influência na estabilidade da estrutura.

Embora essas características possam ser importantes para a estimação da dificuldade, na lista de fases não há fases suficientes com valores não nulos de *offset* ou área interna não ocupada para que a predição seja confiável. Por esse motivo o alinhamento de blocos e a área interna não ocupada não serão usados como características para a predição de dificuldade nessa pesquisa.

4.6 Experimento

4.6.1 Coleta dos Dados

A coleta dos dados foi feita com 8 crianças com idades na faixa de 4 a 6 anos, enquanto esperavam para ser atendidas em um hospital, e cada criança jogou uma média de 30 fases. Antes de jogar, apenas foi dito como o jogo funciona e depois nenhuma outra informação enquanto elas jogavam

A escolha das fases foi feita usando a plataforma Playfab, que devolve uma fase quando o jogo faz uma requisição de uma fase. O sistema de escolhas de fases implementado no Playfab é aleatória, mas sem repetição, ou seja, quando uma fase é jogada, ela não pode ser repetida até que todas as 53 já tenham sido jogadas. O objetivo disso é para que todas as fases tenham uma quantidade semelhante de informações.

4.6.2 Construção do Classificador

Para a construção do decisor, foi usado o programa Knime, que oferece diversas ferramentas para auxiliar na análise de dados.

A previsão foi feita usando o algoritmo RProp para rede neural *multi-Layer Feed-forward*.O conjunto de levels foi particionado em aproximadamente 80% para o treinamento do decisor e aproximadamente 20% para o teste, usando amostras estratificadas de acordo com a dificuldade das fases, correspondentes a 42 fases para treinamento e 11 para teste.

O diagrama do decisor por rede neural é mostrado na figura 4.2, e cada bloco tem a seguinte função:

- *File Reader*: é responsável por ler o arquivo com extensão csv que contém todas as características de cada fase e sua heurística de dificuldade e retornar na saída um conjunto de todas as fases lidas.
- *Normalizer*: tem a função de normalizar cada uma das características do conjunto de fases, usando uma estratégia mínimo-máximo, onde o valor mínimo dentre todas as características assume o valor 0 e o valor máximo assume o valor 1, e todos os outros valores são mapeados nesse intervalo proporcionalmente.
- *Partitioning*: bloco responsável por dividir o conjunto de fases em conjunto de testes e conjunto de treinamento.
- *RProp MLP Learner*: recebe o conjunto de treinamento como entrada e usa o algoritmo RProp *multi-layer perceptron* para treinar uma rede neural. A saída do bloco é a rede treinada.
- *Multi-LayerPerceptron predictor*: usa a rede treinada recebida como entrada para classificar o conjunto de testes recebido como segunda entrada.

- *Scorer*: bloco que mostra a matriz de confusão e a porcentagem de fases no conjunto de teste que foram classificados corretamente.

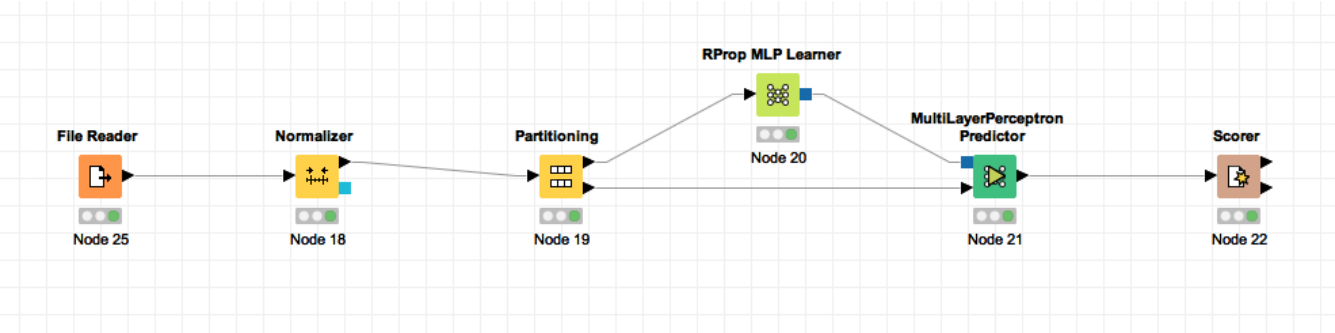

Figura 4.2 Modelo da rede neural

CAPÍTULO 5

Resultados e Análise

Este capítulo apresenta primeiramente os resultados obtidos no experimento e em seguida uma análise e discussão destes resultados.

5.1 Resultados

Para verificar a qualidade da nossa solução, analisamos a porcentagem de predições em que a rede neural resultou em uma dificuldade equivalente à dificuldade estimada. Também analisamos a matriz de confusão resultante da previsão.

Para avaliar a corretude da técnica proposta, analisamos as fases pertencentes ao conjunto de testes e a dificuldade estimada para ela pela rede neural.

O teste proposto resultou em uma taxa de 72.727% de precisão na predição das dificuldades das fases testadas, onde 8 das fases foram corretamente classificadas e 3 foram incorretamente classificadas.

A matriz de confusão é demonstrada na tabela 5.1. As classes de dificuldade são representadas por inteiros sendo 0 a representação de fácil, 1 a representação de médio e 2 a representação de difícil. Cada linha da matriz representa a quantidade total de fases em cada classe de dificuldade (fácil, médio ou difícil), enquanto cada coluna demonstra

Tabela 5.1 Matriz de confusão gerada pela rede neural.

		Classe Prevista		
	dificuldade	0	1	2
	0	3	0	0
Classe Real	1	2	2	1
	2	0	0	3

o número de fases previstas para cada classe de dificuldade.

5.2 Análise dos Resultados

A partir da taxa de acerto alcançada, podemos concluir que nossa heurística, apesar de ser uma estimativa da dificuldade, nos trás uma aproximação confiável da dificuldade das fases. Demonstrando uma relação direta com as características da fase que o preditor é capaz de identificar. Então, usando a heurística proposta, é possível classificar as fases com uma taxa de confiabilidade de 72.727%.

Esse resultado mostra que é possível estender o método proposto por Carvalho [dC12] para jogos de outros gêneros, onde dificuldade deve ser medida de formas diferentes.

Também se pode notar que é possível construir um classificador para fases do jogo puzzle com uma boa taxa de acerto, usando heurísticas para que não haja intrusões desnecessárias durante o jogo.

CAPÍTULO 6

Conclusões

Nesta pesquisa foi mostrado a importância de saber o valor da dificuldade que uma fase possui. Este valor tem diversas utilidades, como ordenação de uma lista de fases ou avaliação de uma fase gerada procedimentalmente. O valor da dificuldade também é essencial para aplicar técnicas que mantenham o jogador no estado de fluxo.

Para descobrir uma estimação da dificuldade, foi proposto uma heurística coletada durante o jogo de diversas crianças. O objetivo dessa heurística é não ser intrusiva, como seria caso perguntasse a dificuldade após cada fase, podendo ser acoplada a um jogo real.

Estendendo o método de Carvalho [dC12], mostramos que é possível aplicar esse método para outros jogos de outros gêneros. Também ressaltamos que o método proposto por Carvalho é apenas um passo para um sistema de geração procedimental para jogos, ou seja, é possível adaptar o sistema usando o classificador proposto nesta pesquisa.

A heurística escolhida foi o percentual de blocos que as crianças colocam erroneamente, pois, dentre todas as possibilidades discutidas, foi a que possuiu maior porcentagem de acerto no preditor quando o experimento era executado.

No experimento, usando a heurística proposta, foi possível obter uma boa taxa de acerto na estimativa da dificuldade. Isso implica que as características presentes na fase tem uma relação com a heurística proposta. Se é possível usar a heurística como uma

medida da dificuldade, então também é possível prever qual a estimativa da dificuldade que uma fase possui.

A taxa de acerto pode talvez ser melhorada se usarmos uma heurística mais elaborada, como por exemplo um conjunto com pesos de várias heurísticas, sendo uma melhor estimativa da dificuldade que o jogador teve na fase. Mas essa pesquisa será vista em trabalhos futuros.

Este trabalho usa uma heurística simples para medir a dificuldade, mas em trabalhos futuros há a possibilidade de desenvolver uma heurística que seja mais precisa na quantificação da dificuldade.

Referências Bibliográficas

[ALN11] Maria-Virginia Aponte, Guillaume Levieux, and Stephane Natkin. Measuring the level of difficulty in single player video games. *Entertainment Computing*, 2(4):205–213, 2011.

[AS10] Daniel Ashlock and Justin Schonfeld. Evolution for automatic assessment of the difficulty of sokoban boards. *IEEE Congress on Evolutionary Computation (CEC)*, 2010.

[Che07] Jenova Chen. Flow in games (and everything else). *Communications of the ACM*, 50(4):31–34, 2007.

[Csi13] Mihaly Csikszentmihalyi. *Flow: The psychology of happiness*. Random House, 2013.

[dC12] Leonardo Vieira de Carvalho. Bonnymetrics - ferramenta para jogos infinitos. Master's thesis, Universidade Federal de Pernambuco, 2012.

[dMdM14] Rubem José Vasconcelos de Medeiros and Tacio Filipe Vasconcelos de Medeiros. Procedural level balancing in runner games. In *Computer Games and Digital Entertainment (SBGAMES), 2014 Brazilian Symposium on*, pages 109–114. IEEE, 2014.

[GB13] Matej Guid and Ivan Bratko. Search-based estimation of problem difficulty for humans. In *International Conference on Artificial Intelligence in Education*, pages 860–863. Springer, 2013.

[HNB12] Guy Hawkins, Keith Nesbitt, and Scott Brown. Dynamic difficulty balancing for cautious players and risk takers. *International Journal of Computer Games Technology*, 2012:3, 2012.

[Hun05] Robin Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 429–433. ACM, 2005.

[HVPLP16] Hendrik Horn, Vanessa Volz, Diego Pérez-Liébana, and Mike Preuss. Mcts/ea hybrid gvgai players and game difficulty estimation. *IEEE Conference on Computational Intelligence and Games (CIG)*, 2016.

[IDKP+07] Wijnand IJsselsteijn, Yvonne De Kort, Karolien Poels, Audrius Jurgelionis, and Francesco Bellotti. Characterising and measuring user experiences in digital games. In *International conference on advances in computer entertainment technology*, volume 2, page 27, 2007.

[JP10] Petr Jarušek and Radek Pelánek. Difficulty rating of sokoban puzzle. In *Proc. of the Fifth Starting AI Researchers' Symposium (STAIRS 2010)*, pages 140–150, 2010.

[KL12] Alex Kuang and T Lextrait. Dynamic difficulty adjustment. *Worchester Polytechnic Institute Final Report*, 2012.

[MG09] Olana Missura and Thomas Gärtner. Player modeling for intelligent difficulty adjustment. In *Discovery Science*, volume 5808, pages 197–211. Springer, 2009.

[MK07] Timo Mantere and Janne Koljonen. Solving, rating and generating sudoku puzzles with ga. *IEEE Congress on Evolutionary Computation*, 2007.

[NC14] Jeanne Nakamura and Mihaly Csikszentmihalyi. The concept of flow. In *Flow and the foundations of positive psychology*, pages 239–263. Springer, 2014.

[PCV+15] Lizeth Joseline Fuentes Pérez, Luciano Arnaldo Romero Calla, Luis Valente, Anselmo Antunes Montenegro, and Esteban Walter Gonzalez Clua. Dynamic game difficulty balancing in real time using evolutionary fuzzy cognitive maps. *14th Brazilian Symposium on Computer Games and Digital Entertainment*, 2015.

[Pel11] Radek Pelánek. Difficulty rating of sudoku puzzles by a computational model. In *FLAIRS Conference*, 2011.

[SPD11] Nathan Sorenson, Philippe Pasquier, and Steve DiPaola. A generic approach to challenge modeling for the procedural creation of video game levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):229–244, 2011.

[TBI08] Tim J.W. Tijs, Dirk Brokken, and Wijnand A. IJsselsteijn. Dynamic game balancing by recognizing affect. *Fun and Games*, pages 88–93, 2008.

[TP11] Joshua Taylor and Ian Parberry. Procedural generation of sokoban levels. In *Proceedings of the International North American Conference on Intelligent Games and Simulation*, pages 5–12, 2011.

[VKLM15] Marc Van Kreveld, Maarten Löffler, and Paul Mutser. Automated puzzle difficulty estimation. In *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*, pages 415–422. IEEE, 2015.