UNIVERSIDADE FEDERAL FLUMINENSE

GABRIEL FERREIRA ALVES

CRIAÇÃO DE LAYOUTS PARA CONTROLES VIRTUAIS DE JOGOS USANDO DESIGN GENERATIVO

NITERÓI

2019

UNIVERSIDADE FEDERAL FLUMINENSE

GABRIEL FERREIRA ALVES

CRIAÇÃO DE LAYOUTS PARA CONTROLES VIRTUAIS DE JOGOS USANDO DESIGN GENERATIVO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Visual.

Orientador:
DANIELA GORSKI TREVISAN

Co-orientador:
ANSELMO ANTUNES MONTENEGRO

NITERÓI

2019

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

A474c Alves, Gabriel Ferreira
Criação de layouts para controles virtuais de jogos usando design generativo / Gabriel Ferreira Alves ; Daniela Gorski Trevisan, orientadora ; Anselmo Antunes Montenegro, coorientador. Niterói, 2019.
86 f. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2019.

DOI: http://dx.doi.org/10.22409/PGC.2019.m.12380757623

1. Jogo eletrônico. 2. Dispositivo de controle. 3. Algoritmo genético. 4. Aprendizado de máquina. 5. Produção intelectual. I. Trevisan, Daniela Gorski, orientadora. II. Montenegro, Anselmo Antunes, coorientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecária responsável: Fabiana Menezes Santos da Silva - CRB7/5274

GABRIEL FERREIRA ALVES

CRIAÇÃO DE LAYOUTS PARA CONTROLES VIRTUAIS DE JOGOS USANDO DESIGN GENERATIVO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Visual.

Aprovada em Abril de 2019.

BANCA EXAMINADORA

Profa. Dra. Daniela Gorski Trevisan - Orientadora, UFF

Prof. Dr. Anselmo Antunes Montenegro - Coorientador, UFF

Profa. Dra. Aline Marins Paes Carvalho, UFF

Prof. Dr. Esteban Walter Gonzalez Clua, UFF

Profa. Dra. Kate Cerqueira Revoredo, UFRJ

Prof. Dr. Alberto Barbosa Raposo, PUC-Rio

Niterói

2019

Agradecimentos

É uma alegria enorme poder estar aqui, desenvolver este trabalho e adquirir todo esse conhecimento. Nada disso seria possível sem o apoio de várias pessoas e instituições, por isso venho agradecê-las.

Primeiramente, agradeço a Deus. Sei que Ele sempre está comigo, tanto nos melhores quanto nos piores momentos. Sem Ele eu não sou nada.

Em seguida agradeço aos meus pais e meu irmão, minha base e minha estrutura, por terem acreditado nos meus sonhos e me apoiado em cada um deles.

À Marina Macedo, por todo amor, carinho, cuidado, pelos conselhos e pelo companheirismo. Por acreditar em mim da mesma forma que eu acredito nela. Eu te amo.

Aos meus orientadores, Daniela Trevisan e Anselmo Montenegro. Vocês acreditaram em mim e confiaram no meu trabalho. Sou eternamente grato por tudo isso e por todo o conhecimento transmitido.

Ao time D.E.S., Gabriel 'Kazumi' e Igor Carddoso, ao meu parceiro do Grimorium, Yuri Flagrare, ao Marcelo Moura, Lucas Abreu, à Letícia Moreira e Tatiane Gonçalves por acreditarem em mim, mesmo quando eu não era capaz disso e por sonharem alto comigo.

Aos game designers voluntários que participaram e opinaram sobre os resultados do trabalho.

Aos meus demais amigos espalhados pelo mundo e familiares que sempre acompanharam cada passo dado e torceram pelo meu sucesso.

Por fim, mas não menos importante, deixo meus agradecimentos à UFF, ao Instituto de Computação, à CAPES e ao ADDLabs. Foram vários conhecimentos adquiridos e oportunidades que só pude experimentar graças a vocês.

Sou grato a cada um porque sei que sozinhos vamos mais rápido, porém juntos vamos mais longe.

Resumo

Controles de videogames possuem um alto fator de influência nos jogadores, pois eles são responsáveis pela diversão e motivação de um jogo. A organização e disposição dos botões é um dos fatores relevantes ao desenvolver novos controles, já que estes são responsáveis por servir como entrada de ações dentro dos jogos. Tendo isso em vista, este trabalho apresenta a construção de um modelo de *design* generativo para apoiar *game designers* a encontrarem *layouts* diferentes e inovadores de controles virtuais para seus jogos, visto que um controle virtual pode ser implementado especialmente para um jogo específico, contendo apenas as ações necessárias para ele. O *design* generativo busca emular o processo evolutivo na área de *design*. Projetistas ou engenheiros inserem parâmetros de projeto em *softwares* de construção generativa e o *software* explora um espaço de projeto contendo um grande número de soluções possíveis, gerando rapidamente centenas ou até milhares de soluções. A partir daí, *designers* e engenheiros devem enfrentar o desafio de filtrar e selecionar os resultados para melhor atender às expectativas do usuário. Esta solução foi desenvolvida unindo algoritmos genéticos e técnicas de aprendizado de máquina como *Support Vector Machine* e K-means para criar *layouts* para o *designer* analisar. No estado atual do modelo, é necessário definir alguns parâmetros de entrada: posições dos botões, tipo de botões, número máximo de indivíduos (quantidade de *layouts*) que serão gerados pelo modelo e número de iterações que serão executadas. Ao fim da geração de indivíduos, realizado pelo algoritmo genético, uma técnica de classificação usando aprendizado de máquina chamada *Support Vector Machine* é aplicada para classificar os indivíduos entre válidos e inválidos, buscando facilitar a exploração do espaço de *design* por parte do projetista. Por fim, o K-means é responsável por agrupar *layouts* semelhantes em *clusters*. Os testes realizados buscaram medir a variabilidade dos resultados gerados pelo algoritmo, mostrando que várias soluções de diferentes controles e diferentes configurações podem ser desenvolvidas para jogos.

Palavras-chave: *Design* generativo, *gamepad*, controle virtual, algoritmos genéticos e aprendizado de máquina.

Abstract

Video game controllers have a high influence factor on players, as they are responsible for the fun and motivation of a game. The organization and arrangement of the buttons is one of the relevant factors when developing new controllers, since these are responsible for serving as input of actions within the games. With this in mind, this work presents the construction of a generative design model to support game designers to find different and innovative layouts of virtual controllers for their games, since a virtual controller can be implemented especially for a specific game, containing only the necessary actions for it. The generative design seeks to emulate the evolutionary process in the design area. Designers or engineers insert design parameters into generative building software, and the software explores a design space containing a large number of possible solutions, quickly generating hundreds or even thousands of solutions. From there, designers and engineers must face the challenge of filtering and selecting results to best meet user expectations. This solution was developed by linking genetic algorithms and machine learning techniques like Support Vector Machine and K-means to create layouts for the designer to analyze. In the current state of the model, it is necessary to define some input parameters: button positions, button type, maximum number of individuals (number of layouts) that will be generated by the model and number of iterations to be executed. At the end of the generation of individuals, performed by the genetic algorithm, a classification technique using machine learning called Support Vector Machine is applied to classify individuals between valid and invalid, seeking to facilitate the exploration of design space by the designer. Finally, K-means is responsible for grouping similar layouts in clusters. The tests performed sought to measure the variability of the results generated by the algorithm, showing that several solutions of different controllers and different configurations can be developed for games.

Keywords: Generative design, gamepad, virtual controller, genetic algorithm and machine learning.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ACM	:	*Association for Computing Machinery*;
CAD	:	*Computer Aided Design*;
D-Pad	:	*Directional Pad or Digital Pad*;
NES	:	*Nintendo Entertainment System*;
PSP	:	*PlayStation Portable*;
RPG	:	*Role-playing game*;
SNES	:	*Super Nintendo Entertainment System*;
SVM	:	*Support-vector machine*;
VR	:	*Virtual Reality*;

Sumário

Capítulo 1

Introdução

Desde o seu surgimento a indústria de jogos se mostrou promissora, ainda que com gráficos básicos, bidimensionais e pixelizados. Com o passar dos anos, o mercado de videogames evoluiu a ponto de se tornar a maior indústria de entretenimento [11], abordando temáticas mais complexas, implementando interações mais elaboradas e também adicionando elementos de outras indústrias do entretenimento, como cenas cinematográficas - conhecidas como *cutscenes* - e trilhas sonoras produzidas por grandes músicos, produtores e orquestras.

Mesmo com toda evolução, alguns aspectos ainda permanecem os mesmos. Um desses aspectos é o fato de que todo jogo permite que o jogador controle as ações de, pelo menos, um personagem por meio de uma interface e/ou dispositivo projetado com o objetivo de proporcionar a melhor experiência do usuário: os controles físicos de jogos, também conhecidos como *gamepads* ou *joysticks* [39].

Várias formas de interação usando dispositivos diferentes foram projetadas ao longo dos anos. É possível controlar as ações dos personagens dos jogos por simples movimentos dos dedos sobre botões do controle, por movimentos corporais, ou até mesmo tomar a visão de um personagem, imergindo em sua realidade por meio da Realidade Virtual [6]. Porém, ainda é fato que a maioria dos jogos mantém sua forma de interação ligada ao *joystick*. Grande parte dos consoles de videogame [29] [31] [38], mesmo com diferentes formatos, tamanhos e pesos, têm ao menos um controle físico para que o jogador possa executar as ações dentro dele.

Ainda com toda essa evolução, a indústria de jogos digitais continua a mudar. Atualmente, muitos jogos desenvolvidos são ferramentas não só de entretenimento, mas também esportivas, de simulação, educativas e afins. Algumas inovações foram propostas e

desenvolvidas em jogos digitais e, para que o usuário pudesse acompanhá-las, formas de interação com esses jogos foram revistas, replanejadas e remodeladas.

1.1 Os controles virtuais

Com o avanço da tecnologia e graças a sua dinamicidade, dispositivos *touchscreen* como *smartphones* e *tablets* também se tornaram controles de jogos: os *gamepads* virtuais. Eles são dinâmicos por possuírem componentes e dispositivos como conexão *bluetooth* e *wireless*, acelerômetros, giroscópios, câmeras e afins. Esses componentes podem contribuir para o desenvolvimento de diferentes formas de interação com jogos digitais [4], seja em jogos executados nos próprios dispositivos móveis, como os exibidos na Figura 1.1, ou em jogos executados em um outro dispositivo (computadores ou consoles) como mostra a Figura 1.2.

Figura 1.1: Jogos Pokémon e Sonic sendo executados em dispositivos móveis. **Fonte**: [17][42]

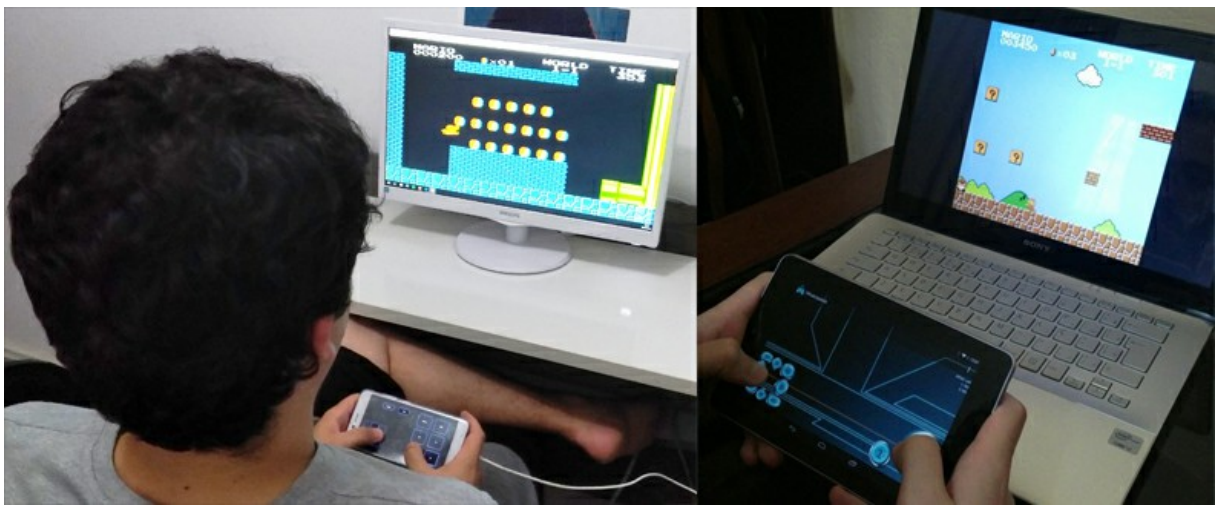

Figura 1.2: Exemplos de controles virtuais para serem usados em jogos executados em um computador. **Fonte**: [2] [40]

Como esses controles acessam os componentes dos dispositivos por meio de toques na tela e/ou por movimentação e rotação do dispositivo, os botões no controle virtual podem seguir as regras do jogo, não sendo necessário incluir vários botões que não serão usados durante a *gameplay*. Infelizmente isso não é possível em controles físicos, visto que, uma vez que o controle foi construído, todos os botões estarão sempre disponíveis para o usuário, ainda que não tenham nenhuma utilidade em determinados jogos.

Apesar do grande potencial que esses dispositivos têm ao controlar jogos, eles possuem uma grande desvantagem: a falta de *feedback* tátil. Devido a falta de botões físicos onde o jogador pode sentir o retorno das ações que ele executa, o número de vezes que ele precisa olhar para a tela do dispositivo aumenta, influenciando negativamente a sua experiência durante a *gameplay* [5].

Algumas soluções buscaram resolver ou suavizar a falta de *feedback* tátil por meio de componentes físicos [2][12][20][25], enquanto outras soluções foram desenvolvidas completamente como *softwares* [4][5][40]. As soluções que buscam tratar a falta de *feedback* tátil por meio de *hardware* são utilizadas em jogos que estão sendo executados no próprio dispositivo móvel, incluindo botões físicos que, ao serem pressionados, enviam informações por algum meio de comunicação para o jogo. Já o segundo tipo de controle, os controles virtuais, são implementados para que o jogador consiga controlar as ações em jogos que estão sendo executados em outros dispositivos como, por exemplo, um computador.

1.2 Objetivo e escopo do trabalho

O objetivo deste trabalho é desenvolver uma ferramenta que possa auxiliar *game designers* a encontrarem *layouts* diferentes e inovadores de controles virtuais para seus jogos, visto que um controle virtual pode ser implementado especialmente para um jogo específico, contendo apenas as ações necessárias para ele. Logo, não é objetivo deste trabalho fornecer *layouts* para controles físicos de videogames.

A indústria de jogos sempre busca inovar e tenta desenvolver uma melhor solução que torne possível atingir uma experiência satisfatória por parte de seus clientes. Cada jogo possui um número de ações e interações finitas que não são necessariamente as mesmas que em outros jogos, e podem variar entre diferentes classes de jogos, como jogos de ação e aventura; esportes e tiro; corrida e plataforma; e até dentro da mesma classe de jogos. Essas ações geralmente são associadas a um único botão no controle, portanto, haverá um número máximo de botões necessários para realizar todas as interações no jogo.

Com essas informações, levantamos a seguinte questão: como auxiliar a atividade criativa de *designers* para que seja possível desenvolver *layouts* diferentes e inovadores para controles virtuais de jogos? Para isso, foi definido um modelo de *design* generativo [7] onde o projetista deve informar ao sistema quais ações o jogador deverá ser capaz de executar dentro do jogo, a quantidade máxima de configurações que ele deseja que o modelo entregue como saída e quantas iterações o algoritmo executará antes de entregar esses *layouts*. Uma abordagem baseada em evolução genética foi usada para gerar uma variedade de configurações de *layouts*, abrindo o espaço de *design*.

Duas abordagens de aprendizado de máquina foram aplicadas: uma utilizando o algoritmo de classificação *Support Vector Machine* [43] para reduzir o espaço de busca de soluções a serem apresentados ao *game designer*, usando como ponto de partida as informações adicionadas a uma base de dados de treinamento, para que sejam sugeridos um conjunto de configurações de controles válidos para o jogo desejado; e outra, utilizando o algoritmo de clusterização K-means [21] para agrupar essas soluções dadas semelhanças entre elas, tornando possível medir a variabilidade do modelo e fazendo com que a visualização dos *layouts* por parte do *designer* seja mais intuitiva. Essas abordagens foram escolhidas porque o *design* generativo fornece uma grande gama de soluções, porém torna-se inviável para o profissional analisar todas elas.

Com base nos resultados gerados e filtrados, o *designer* poderá escolher qual ele acha mais adequado ao seu jogo e, se necessário, fazer ajustes e refinamentos em seu *design*. Uma solução baseada em *design* generativo pode ser usada como um ponto de partida robusto onde os resultados podem ser refinados por engenheiros de *software* e artistas [41]. Logo, é importante ressaltar que esta ferramenta é um *software* de suporte criativo ao *designer*, o verdadeiro responsável pelo desenvolvimento do controle de seu jogo.

Ao desenvolver as regras de criação de *layouts* no modelo de *design* generativo, usamos uma regra muito comum em controles tanto físicos quanto virtuais: o agrupamento de botões de mesma categoria e distanciamento dos que são de categorias diferentes, detalhado no Capítulo 4. Apesar dessa regra ser definida como um ponto de partida para se encontrar soluções viáveis, existe a hipótese de que ela poderia ser responsável pela restrição da criatividade do modelo. Para tratarmos isso, realizamos testes que utilizam e não utilizam a regra de agrupamento de botões para verificar como os resultados se comportam. Tais experimentos foram realizados porque o objetivo deste trabalho não é copiar padrões existentes em controles, mas sim sugerir outros padrões a partir deles.

Quanto à falta de *feedback* tátil mencionada anteriormente, foi desenvolvido em um

outro trabalho [2] soluções que suavizam a inexistência desse *feedback*. Apesar disso, sabemos que outros fatores influenciam fortemente a experiência do jogador, como usabilidade, capacidade de resposta, conforto e ergonomia [24], mas o foco deste trabalho está na busca de novos *layouts* que apresentem apenas as ações que serão executadas dentro de um jogo - como ocorria no Nintendo Game and Watch [26], por exemplo - e usabilidade baseados na experiência de jogadores.

1.3 Estrutura do trabalho

Este trabalho está dividido em seis capítulos. O primeiro contém a introdução, onde é apresentada a motivação, o objetivo deste trabalho e seu escopo.

No segundo capítulo é realizado um resumo da evolução dos videogames e seus controles ao passar dos anos.

O terceiro capítulo, por sua vez, apresenta outros trabalhos que se relacionam a esse, sendo feita uma revisão bibliográfica de artigos, dissertações, teses e livros das áreas que esse trabalho abrange.

O quarto capítulo aborda a metodologia e a implementação do trabalho. Nele são descritos os processos, as tecnologias e todas as questões que envolvem o projeto do modelo e seu desenvolvimento. Esse capítulo é dividido em três seções: a primeira descreve o modelo e seus parâmetros de entrada; a segunda detalha a definição e implementação do algoritmo evolutivo utilizado; por fim, a terceira é responsável por detalhar a técnica de aprendizado de máquina utilizada no modelo.

O quinto capítulo é responsável por apresentar e detalhar a execução dos experimentos e seus resultados, mostrando as definições de parâmetros, as saídas do modelo e as interpretações extraídas desses dados.

Por fim, o sexto capítulo traz as conclusões, trabalhos futuros, melhorias propostas e estudos que foram desenvolvidos em paralelo a este.

Capítulo 2

A evolução dos videogames e seus controles

Os jogos digitais e os videogames possuem uma história repleta de mudanças, evoluções e inovações. Embora existam informações de outros jogos de computador que foram datados anteriormente [18], o que foi considerado como 'o primeiro jogo digital para computador' foi o *Spacewar!*, programado por Steve Russel em Fevereiro de 1962 [19]. Cada jogador usava quatro dos oito interruptores do computador PDP-1 como controles de navegação, para que fosse possível girar a nave para a esquerda, para a direita, acelerar e atirar. Ganhava o jogo quem destruísse primeiro a nave do adversário.

Um dos problemas detalhados por Lu [23] sobre a experiência de jogar o *Spacewar!* era a altura elevada do monitor do computador PDP-1. Os jogadores não conseguiam executar as ações do jogo por muito tempo devido o desconforto gerado por olhar para a tela por um longo período. Além disso, a proximidade de um dos interruptores com o monitor favorecia quem o usava. Devido esses problemas, os desenvolvedores construíram duas pequenas caixas. Em cada uma delas havia uma chave de rotação e um botão para cada ação, criando assim o primeiro controle de videogame. A Figura 2.1 ilustra jogadores testando o jogo *Spacewar!*

Figura 2.1: Duas pessoas jogando o jogo *Spacewar!*. **Fonte**: [9]

Em 1967 o mundo presenciou a criação do primeiro console de videogame, conhecido como *The Brown Box* [33]. Ele era essencialmente uma caixa de madeira marrom retangular com dois controles conectados. Foi projetado para conectar-se a aparelhos de TV e disponibilizou aos jogadores um total de seis jogos - tênis, pingue-pongue, handebol, vôlei, perseguição e um jogo de tiro. Seu controle também era uma caixa marrom com três chaves de rotação e um botão. A Figura 2.2 mostra como era o videogame.

Figura 2.2: Console *The Brown Box*. **Fonte**: [27]

Em 1972, a Magnavox lançou o Magnavox Odyssey. Seu controle possuía duas chaves de rotação - uma para controlar a movimentação horizontal, outra para a movimentação vertical - e um botão para reiniciar o jogo. Três anos depois, a Magnavox decidiu melhorar o sistema Odyssey e começou a lançar novos modelos regularmente, como o Magnavox Odyssey 100, 200, 300, 400, 500, 2000, 3000 e 4000 [33]. A Figura 2.3 mostra todos os Magnavox Odyssey citados acima.

Figura 2.3: De cima para baixo, da esquerda para a direita: Magnavox Odyssey, Magnavox Odyssey 100, Magnavox Odyssey 200, Magnavox Odyssey 300, Magnavox Odyssey 400, Magnavox Odyssey 500, Magnavox Odyssey 2000, Magnavox Odyssey 3000 e Magnavox Odyssey 4000. **Fonte**: [33]

No entanto, os dispositivos Magnavox enfrentaram a concorrência do Atari 2600, onde o *joystick* possuía apenas um botão e um manche, este último podendo assumir oito direções diferentes. Este controle possuía um *design* simples que facilitava o primeiro contato de grande parte dos jogadores que não tiveram experiência prévia com outro console. Além disso, este *joystick* poderia ser usado em diversos jogos, sendo um dos primeiros a serem abrangentes e de natural manejo [30]. A Figura 2.4 ilustra o Atari 2600 e seu controle.

Figura 2.4: Atari 2600. **Fonte**: [3]

Enquanto isso, a Fairchild e a Coleco entraram no mercado de consoles de videogame.

A Coleco liberou os consoles *Telstar Ranger*, *Alpha, Colormatic* e *Combat*. Na mesma época, a *General Home Products* levou o *Wonder Wizard* à atenção dos jogadores. Seu controle era dotado de duas chaves de rotação e três interruptores - um para escolher jogos, outro para escolher a dificuldade e um último para ligar/desligar/resetar o videogame. A Figura 2.5 apresenta os consoles da Coleco e o Wonder Wizard.

Figura 2.5: De cima para baixo, da esquerda para a direita: Telstar Ranger, Telstar Alpha, Telstar Combat, Telstar Colormatic e Wonder Wizard. **Fonte**: [10] [33]

No final dos anos 70, a Nintendo lançou o Color TV Game Series. No entanto, o console estava disponível apenas no Japão. Em 1979, a Mattel introduziu o console do Intellivision. A Coleco continuou abrindo caminho para o crescimento ao lançar os consoles Telstar Sportsman, Colortron, Marksman e Gemini. Em 1974, a Philips comprou a Magnavox e lançou o Philips Odyssey 2, o Philips Odyssey 2001 e o Philips Odyssey 2100. Apesar de tantos lançamentos, não houveram inovações significativas em seus controles. Veja esses consoles na Figura 2.6.

Figura 2.6: De cima para baixo, da esquerda para a direita: Color TV Game, Intellivision, Telstar Sportsman, Telstar Colortron, Telstar Marksman, Telstar Gemini, Odyssey 2, Odyssey 2001 e Odyssey 2100. **Fonte**: [33]

Nos anos 80, também conhecida como a Era de Ouro dos videogames, foi um período de inovação. A indústria fez um desvio dos jogos de pingue-pongue e deu os primeiros passos em direção à diversificação e desenvolvimento de jogos de aventura, RPG e luta.

No início da década, jogos como *Mario Bros.*, *The Legend of Zelda*, *Final Fantasy* e *Golden Axe* apareceram. Nessa época nascia o *Game and Watch*, uma linha de consoles portáteis que podem ser vistos na Figura 2.7. O primeiro console da linha possuía apenas dois botões, direita e esquerda. Porém, conforme os jogos foram evoluindo, a necessidade de mais botões se tornou real. Para solucionar o problema, engenheiros da Nintendo criaram uma alavanca digital em forma de cruz, operado pelo polegar, capaz de mover em quatro direções e endereçar oito valores possíveis. Esta inovação teve início nos consoles portáteis, avançou para os consoles de mesa e é utilizada até hoje. No entanto, a mudança mais notável foi a mudança de consoles dedicados com alguns jogos embutidos para sistemas de videogame baseados em cartucho.

Figura 2.7: Algums modelos do Nintendo Game and Watch. **Fonte**: [26]

Em 1983, a Sega lançou o SG-1000, mas obteve sucesso esporádico devido à introdução do console *Nintendo Entertainment System* (NES) pela Nintendo. Dois anos depois, em 1985, a Sega lançou o Sega *Master System*, que foi bem aceito pelos jogadores. O gamepad do NES possuía um *D-pad*, responsável por controlar as direções do personagem, e mais dois botões de ação. Seu *design* era muito bom para jogos de plataforma como o *Super Mario Bros.* por utilizar apenas direcionais, onde os botões horizontais do *D-Pad* eram os mais usados em conjunto com os botões de ação para pular e correr [30]. A Atari lançou um novo modelo de console, o Atari 5200. O Intellivision II e o ColecoVision também foram lançados nessa época. Os consoles são mostrados na Figura 2.8.

Figura 2.8: De cima para baixo, da esquerda para a direita: SG-1000, NES, Master System, Atari 5200, Intellivision II e ColecoVision. **Fonte**: [33]

No fim dos anos 80, a indústria dos videogames estava em um alto crescimento [30]. A Sega, ao criar o *Mega Drive*, apostou em um controle mais ergonômico, com um *D-Pad* e um botão de ação a mais que o NES. Algum tempo depois, lançou outro controle, este com 6 botões de ação. A Nintendo por sua vez lançou o Super Nintendo Entertainment System (SNES), com um controle mais ergonômico e com mais botões que o NES. O SNES continha, além dos direcionais, quatro botões de ação e introduziu o conceito de botões laterais, denominados L e R. Naquela época, a Atari lançou o modelo 7800 e a Nintendo também lançou uma nova linha de consoles portáteis, o Game Boy. Todos esses consoles são exibidos na Figura 2.9.

Figura 2.9: Da esquerda para a direita: Mega Drive, SNES, Atari 7800 e Game Boy. **Fonte**: [33]

No início dos anos 90, a SNK Neo Geo lançou Neo Geo AES. Ele era impressionante pelos gráficos que produzia naquela época, mas o preço alto fez com que o console não

alcançasse o esperado sucesso [33]. O ano de 1992 viu o lançamento do primeiro console de CD, o Philips CD-i. No ano seguinte, o Sega CD foi lançado. Em 1993, a Atari lançou o Atari Jaguar, o último console lançado pela empresa.

Anos depois, após uma parceria malsucedida entre a Sony e a Nintendo para que o Super Nintendo pudesse rodar jogos em formato de CD, a Sony lança o PlayStation. Ele possuiu o primeiro controle que sofreu modificações ergonômicas adicionando duas saliências inferiores para melhorar a empunhadura das mãos, que existem até hoje na maioria dos controles [30]. Além disso, adicionou mais dois botões laterais, contando com 4 botões nesta parte: L1, L2, R1 e R2. O *design* com dois analógicos do controle - nomeado como *DualShock* - permitia uma maior variedade de comandos.

A Sega lançou o console Genesis 2, seguido por Genesis 3 em 1997 e um novo console Saturno. Como resposta ao PlayStation, a Nintendo criou o Nintendo 64. O seu controle apresentou diversas inovações: ele possuía um *D-Pad*, seis botões de ação (visto que o SNES e o PlayStation possuiam apenas quatro), dois botões laterais e um manche analógico entre o *D-Pad* e os botões. Ele adicionou um gatilho na parte traseira do controle, conhecido como botão Z, que auxiliava consideravelmente em jogos de tiro. Também inseriu uma porta de entrada de expansões, onde uma das principais expansões utilizadas foi o *Rumble Pack*, que fazia o controle vibrar em determinadas situações.

Em 1998, a Sega lançou o Dreamcast. Este último forneceu suporte à Internet através de um modem embutido para jogar *on-line*, o que deu uma grande vantagem sobre a concorrência, além de adicionar um cartão de memória com uma tela de LCD no controle. Nessa época, a Nintendo lançou uma versão melhorada do Game Boy, o Game Boy Color. Os consoles e controles da década de 90 podem ser vistos na Figura 2.10.

Figura 2.10: Os consoles dos anos 90: Neo Geo AES, Philips CD-i, Sega CD, Atari Jaguar, Playstation, Sega Saturn, Genesis 2, Genesis 3, Nintendo 64, DreamCast e Game Boy Color. **Fonte**: [33]

O começo dos anos 2000 marcou a chegada do Playstation 2 e junto do console foi lançado o DualShock 2, uma versão melhorada do DualShock. Uma evolução realizada nesse controle foi que a maioria dos botões eram lidos pelos consoles como valores analógicos, indicando não só se estavam sendo apertados como também qual era a pressão exercida neles. A Nintendo então lançou um dispositivo que pudesse resistir à dura concorrência. Como resultado, o Nintendo GameCube chegou às lojas em 2001. Nesta geração, seu controle melhorou consideravelmente os aspectos ergonômicos, porém os botões laterais L e R se tornaram gatilhos com valores analógicos, sendo que o gatilho traseiro Z, presente no Nintendo 64, foi adicionado na lateral à frente do botão R. Na linha de portáteis, ela lançou o Game Boy Advance, que possuía retrocompatibilidade com os jogos do Game Boy e Game Boy Color. A Microsoft, por fim, lançou o console Xbox. O Xbox chegou com um disco rígido integrado que permitia aos jogadores salvar jogos. Os primeiros consoles dos anos 2000 podem ser vistos na Figura 2.11.

Figura 2.11: O PlayStation 2, o Nintendo Game Cube, o Game Boy Advance e o Xbox. **Fonte**: [33]

Em 2004 a Nintendo continuou investindo em portáteis lançando o Nintendo DS (*Dual Screen*). Sua maior inovação foi a presença de uma segunda tela *touchscreen*, onde ações podiam ser feitas pelo toque. Possuía uma cruz direcional, quatro botões de ação, e os botões *start*, *select*, L e R. A Sony, pronta para competir com a Nintendo, lançou o PlayStation *Portable*, popularizado como PSP. Ao contrário dos Dualshock que possuíam duas alavancas analógicas e dois botões L e R, o PSP possui apenas uma alavanca, um botão L e um botão R. Veja-os na Figura 2.12.

Figura 2.12: Nintendo DS e PlayStation Portable. **Fonte**: [1]

A geração seguinte de consoles apresentou a maior evolução na forma em que os usuários interagem com os aparelhos e seus jogos. Em 2005, a Microsoft lançou o Xbox 360. No ano seguinte, a Sony lançou o Playstation 3. Ambos os dispositivos possuem gráficos Full HD de 1080p. Os controles principais de ambos os consoles não mudaram muito, sofrendo apenas alguns ajustes como a substituição de fios para conectar o controle ao console por tecnologias *wireless*.

O Nintendo Wii iniciou uma nova tendência nos consoles. O seu controle, conhecido como Wii Remote, não se parecia com os controles dos demais videogames, mas com um controle de televisão. Sua principal característica era que, em conjunto com uma barra de sensores, o videogame conseguia capturar os movimentos realizados com o controle.

Após o sucesso introduzido nos consoles pela Nintendo com o Wii, a Sony lançou o *Playstation Move* e o *Playstation Eye*, que capturavam os movimentos realizados pelo

Move e alguns dos movimentos do usuário para executar as ações dentro do jogo. Enquanto isso, a Microsoft lançou o Kinect, um sensor capaz de capturar os movimentos do jogador. Ele utiliza uma câmera RGB, um sensor de profundidade, um microfone e um processador próprio para que, com esta nova forma de interação, os jogadores pudessem ter uma maior imersão, já que seus movimentos eram mapeados para os do personagem. Como são formas de interação mais simples, muitas pessoas começaram a se inserir no ramo e a se interessar por video *games*. Veja os videogames e suas ferramentas de inovação na Figura 2.13.

Figura 2.13: Os consoles que revolucionaram sua geração: PlayStation 3 (junto ao PlayStation Eye e PlayStation Move), Nintendo Wii e Xbox 360 (junto ao Kinect). **Fonte**: [33]

Os jogadores de console contemporâneos têm uma infinidade de dispositivos de primeira linha para escolher. Em 2011, a Nintendo disponibilizou o Nintendo 3DS, que usa uma tela com tecnologia que permite ver imagens em 3D sem óculos, enviando duas imagens com ângulos ligeiramente diferentes para cada olho, causando o efeito de profundidade e de tridimensionalidade para fora da tela. O console possui uma cruz direcional e uma alavanca que permite navegar, de forma confortável, para frente, trás, direita, esquerda e qualquer diagonal. Também inclui um acelerômetro e um giroscópio que permitem detectar movimentos do portátil. Em 2012, a Sony lançou o PS Vita, com duas alavancas analógicas, teclas direcionais, um conjunto de botões padrão do PlayStation, dois botões laterais (L e R), um botão PlayStation e botões *start* e *select*.

A Nintendo lançou o Wii U, que trazia consigo um novo controle principal, chamado de Wii U Gamepad, que mistura aspectos de *tablet* e *joystick*. O Gamepad possui uma tela LCD no centro, que pode funcionar tanto como uma tela auxiliar quanto como a tela principal do console, sem a necessidade do videogame estar conectado a uma televisão. Ele também possui compatibilidade com todos os acessórios do console da Nintendo da geração anterior. Os consoles são ilustrados na Figura 2.14.

Figura 2.14: Nintendo 3DS, PlayStation Vita e Nintendo Wii U. **Fonte**: [33]

A chegada do Playstation 4 em 2013 foi um sucesso. O dispositivo possui um controle refinado, bem como uma variedade de serviços conectados para jogos *online* e *streaming* de mídia. Juntamente do console veio o DualShock 4 como controle principal, que foi levemente redesenhado, introduziu um *touchpad* na parte superior da face do controle, melhorou os botões e substituiu os antigos e comuns '*select*' e '*start*' por '*options*' e '*share*' para se adequar às redes sociais.

Pouco depois, a Microsoft lançou o Xbox One, que se tornou o principal concorrente do PS4. Seu controle trouxe diversas melhorias em relação ao controle anterior: ele sofreu mudanças em seu design, adicionando baterias internas no controle. Os botões também passaram por modificações de posição e o *D-Pad* teve sua altura diminuída. Os analógicos e seu acabamento sofreram mudanças, trazendo melhorias no conforto para o jogador.

Em 2017 aconteceu o lançamento do Nintendo Switch, que foi um enorme sucesso de vendas. A proposta de ser um console de mesa e ao mesmo tempo um console portátil agradou fãs e novos entusiastas da indústria de *games*. Seus controles podem ser acoplados junto à tela *touchscreen*, caso o jogador prefira a abordagem portátil, porém eles podem ser facilmente removidos da tela e usados individualmente - um em cada mão - ou em conjunto, quando agrupados à uma base disponível juntamente do console. Os mais novos consoles da geração atual são mostrados na Figura 2.15.

Figura 2.15: PlayStation 4, Xbox One e Nintendo Switch. **Fonte**: [33]

Outras inovações também surgiram para atender as necessidades do público. A Sony vem investindo bastante em jogos para Realidade Virtual e lançou o PlayStation VR, um óculos de realidade virtual. A Nintendo, por sua vez, lançou o Nintendo Labo, que

aproveita da criatividade do jogador e do uso de papelão para desenvolver seus próprios controles adaptados ao contexto do jogo, como varas de pescar, piano e kit robô. Já a Microsoft resolveu inovar e criar um controle para que pessoas com necessidades físicas especiais pudessem ter uma melhor experiência de usuário durante a execução jogos. O Xbox Adaptive Controller é uma interface na qual é possível conectar *joysticks* e botões compatíveis para criar controles acessíveis sob medida para jogar no Xbox One e Windows 10. Outra inovação recente é o Google Stadia [16], o novo serviço de *streaming* da Google onde os jogos ficam armazenados nos servidores da empresa. Apenas será necessário um controle para se conectar direto aos servidores da empresa. Esse controle conta com a presença de um botão de compartilhamento que torna possível compartilhar uma partida ao vivo no YouTube, além de um botão para ativar o Google Assistente. As mais recentes inovações das três maiores empresas desenvolvedoras de consoles da atualidade podem ser vistos na Figura 2.16.

Figura 2.16: Inovações na área de *games*: PlayStation VR, Nintendo Labo, Xbox Adaptive Controller e Stadia Controller. **Fonte**: [16]

Capítulo 3

Trabalhos relacionados

Para ajudar os *designers* de jogos a desenvolver um ou mais *layouts* de controles virtuais ideais e inovadores para seus jogos, procuramos encontrar abordagens, técnicas e metodologias que auxiliem no processo de criação de ideias e suporte ao processo de *design*.

Depois de pesquisar em bibliotecas acadêmicas como o Google Acadêmico e a biblioteca ACM, não foram encontrados resultados sobre o uso de *design* gerativo aplicado na área de controles virtuais de jogos digitais, portanto, esta seção apresentará trabalhos e conhecimentos que serviram de base para o estudo e desenvolvimento da nossa solução.

3.1 Interfaces virtuais de controles para jogos

Com o surgimento dos dispositivos com a tecnologia *touchscreen* como *smartphones* e *tablets*, várias possibilidades surgiram para a indústria de jogos como um todo. Jogos são criados a todo momento, sejam eles para consoles, computadores ou dispositivos móveis e os *gamepads* virtuais, foco deste trabalho, possuem um grande potencial para controlá-los, visto que eles podem ser desenvolvidos exclusivamente para o contexto específico de um jogo.

Os *gamepads* virtuais também respeitam algumas regras e foram criados baseando-se em elementos organizacionais existentes em controles físicos, como os agrupamentos de botões direcionais, a possibilidade de incorporação de *d-pads*, a existência de botões de ação, entre outros.

Procurando soluções para suportar o processo de *design* de controles de jogos, foi encontrada uma proposta de usar um conjunto de técnicas de *Design Thinking*, como um dia na vida, mapa de empatia, persona, *workshop* de ideação e prototipagem de papel para

projetar um controle de jogo que agrade seus jogadores [2]. O *Design Thinking* é uma abordagem que busca a solução de problemas de forma coletiva e colaborativa, onde as pessoas - não apenas o consumidor final, mas todos os envolvidos na ideia - são colocadas no centro de desenvolvimento do produto.

Os protótipos desenvolvidos buscaram suavizar e até mesmo eliminar o problema da falta de *feedback* tátil em dispositivos *touchscreen*. Uma das soluções foi implementada e testada por diversos jogadores em dois jogos diferentes. Durante os testes, os autores puderam perceber que o desempenho dos jogadores ao utilizar a solução implementada foi melhor do que quando o controle virtual foi usado isoladamente.

Os resultados mostraram que as técnicas de *Design Thinking* foram úteis para informar e orientar os *designers* participantes na geração de ideias. Entretanto, apesar do trabalho se relacionar com este na forma em que buscam novas soluções que suportem o processo de *design*, nenhum suporte de *software* para o processo de *design* foi fornecido, visto que as soluções propostas buscavam resolver problemas relacionadas a falta de *feedback* tátil em controles virtuais para jogos. Os protótipos e a solução implementada podem ser vistos na Figura 3.1.

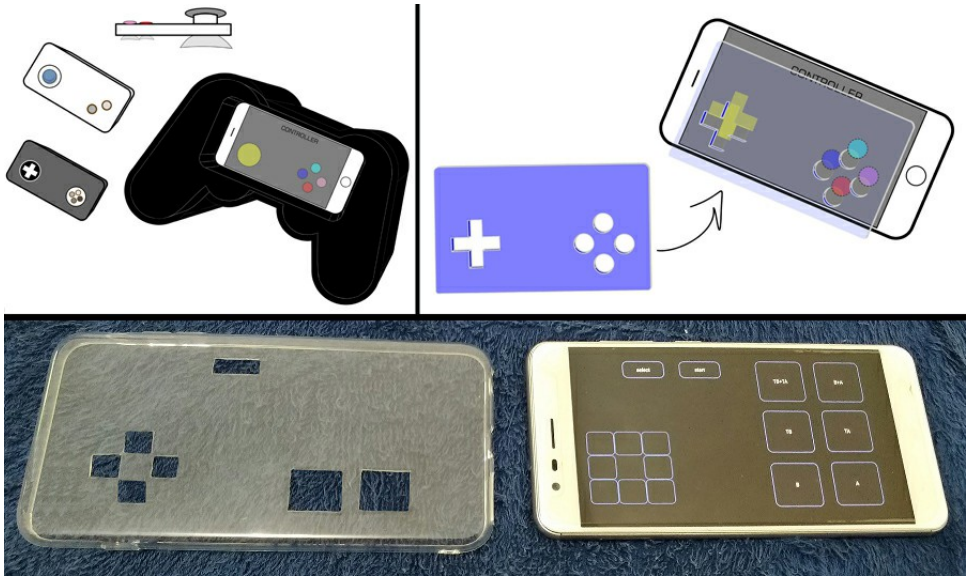

Figura 3.1: Acima, as soluções desenvolvidas ao fim do *workshop*. Abaixo, a segunda solução desenvolvida foi implementada. **Fonte**: [2]

No trabalho de Baldauf et al. [5], foi realizado um estudo de laboratório comparativo onde foram selecionados quatro *designs* de *gamepad* para *smartphones*. Cada um deles foi testado em dois jogos populares: o jogo *arcade* Pac-Man e o jogo de plataforma Super Mario Bros. Para a construção de protótipos de estudo funcional dos *gamepads* a serem

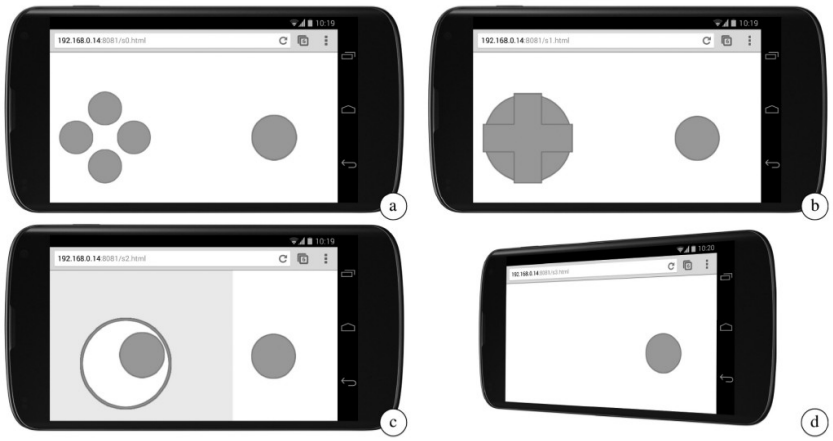

Figura 3.2: *Gamepad* virtual em um *smartphone*. **Fonte**: [5].

avaliados, foi utilizado o ATREUS, um *framework* de código aberto. Os componentes principais do ATREUS compreendem um aplicativo composto de um servidor e uma biblioteca *mobile*, capazes de trocar comandos de controle (por exemplo, se um botão foi pressionado ou liberado) através de uma conexão *websocket*.

Usando o *framework*, foram desenvolvidos os quatro controles: um contendo botões direcionais, um com um *d-pad* com 8 direções, um *joystick* virtual e um controle de inclinação, como mostra a Figura 3.2. Foi integrado à interface apenas um botão de ação de tamanho e posição iguais em cada *gamepad*.

Após a aplicação de testes com usuários, para o controle com botões direcionais, vários participantes relacionaram o *gamepad* com os respectivos controles físicos e mencionaram a falta negativa de *feedback* tátil. Os comentários feitos por usuários após o teste dos controle incluíram que os botões eram difíceis de atingir precisamente, que as direções opostas eram difíceis de pressionar e, portanto, os olhares para o dispositivo eram necessários. Para o controle de movimento, foram feitos comentários negativos, como "a posição neutra é difícil de encontrar". Os comentários para o jogo de plataforma foram mais positivos do que para o Pac-Man e foram feitas sugestões para que o controle de movimento fosse usado para simuladores de voo.

Baldauf et al. [5] desenvolveu quatro controles virtuais baseando-se nas configurações de controles já existentes. A construção de diferentes configurações de controles virtuais para jogos é a principal relação entre este trabalho e o dos autores, ainda que este trabalho tenha como objetivo auxiliar e dar suporte criativo ao *designer*, e não criar o controle em si.

Procurando por abordagens que usassem técnicas de inteligência artificial para apoiar

o *design* de controles de jogos, encontramos o trabalho de Torok et al [39] que lida com o *design* de controles de jogos para dispositivos *touchscreen*.

Eles concordam que as telas sensíveis ao toque não podem replicar a mesma precisão de um controle tradicional e a interface de controle projetada pelo *designer* do jogo não é necessariamente a melhor interface possível do ponto de vista ergonômico, podendo precisar de alguns ajustes para alcançar as configurações ideais. Para resolver esses problemas, sua solução introduz uma adaptação que deriva as preferências pessoais do usuário de uma série de eventos básicos, como pressionamentos de botões ou mudanças de jogabilidade internas, adaptando-se ao usuário e suas necessidades ergonômicas baseadas em abordagens de aprendizado de máquina.

A principal desvantagem dessa abordagem é que a adaptação no *layout* do controle ocorre durante o tempo de interação do jogador e não durante o tempo de *design*. Às vezes, muitas mudanças no *layout* do controle podem confundir o jogador, porém isso pode ser usado pelo *designer* como uma forma de agregar diferentes níveis de dificuldade em seu jogo. Algumas das alterações que ocorrem no *layout* do *gamepad* podem ser vistas na Figura 3.3.

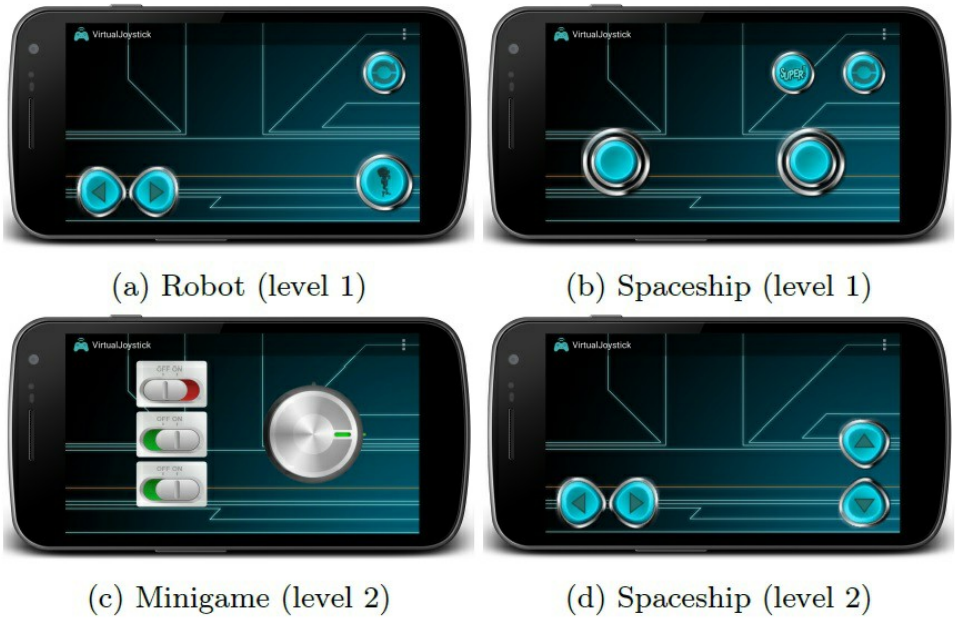

Figura 3.3: Alterações dos *layouts* do controle durante a *gameplay*. **Fonte**: [39].

Este trabalho possui semelhanças com o trabalho de Torok et al. [39], principalmente na busca de encontrar soluções para controles virtuais. Entretanto, seu trabalho realiza as soluções em tempo de interação, enquanto este as realiza em tempo de *design*.

3.2 *Design* generativo

O *design* generativo é uma abordagem promissora para resolver vários aspectos relacionados à interface do usuário [41]. Ele oferece novas oportunidades para problemas de concepção e criação em áreas como engenharia, arquitetura e *design*, possibilitando, por meio do uso de meta-heurísticas, determinar o layout dos componentes de interface, informando apenas uma quantidade limitada de parâmetros e produzindo novos e, às vezes, resultados inesperados [41]. Os pesquisadores argumentam que o que hoje é conhecido como *design* generativo surgiu no início dos anos 1970 com o trabalho de Frazer [14] e seu desenvolvimento passou por várias fases, sendo liderado por pesquisadores acadêmicos guiados pela teoria do *design* [22].

Troiano e Birtolo [41] discutem a aplicação de algoritmos genéticos para resolver dois problemas: a construção de *layouts* de menus hierárquicos e a seleção de paletas de cores com base em um conjunto de restrições. Ambos os problemas foram resolvidos usando Algoritmos Genéticos Simples [15]. Tais meta-heurísticas evolutivas foram escolhidas pelos autores devido à sua simplicidade, pela quantidade de soluções que podem ser geradas em um curto período e aos resultados que são capazes de proporcionar.

A realização dos experimentos em seu trabalho apresentaram resultados que comprovaram a capacidade de um algoritmo genético simples convergir para soluções com alta aptidão. A solução resultante pode ser usada, segundo os autores, como um ponto de partida robusto a ser refinado por engenheiros de *software* e artistas.

O uso de algoritmos genéticos para a construção de *layouts* é a principal relação entre este trabalho e o de Troiano e Birtolo [41]. Como foi dito pelos próprios autores, o uso de técnicas meta-heurísticas inspiradas na evolução natural são boas formas de gerarem uma grande quantidade de resultados rapidamente e que, posteriormente, poderão ser avaliadas por engenheiros e *designers*.

Em seu trabalho, Krish [22] propõe um modelo de projeto gerativo utilizando CAD, apropriado para problemas complexos de *design* multicritério. Este método baseia-se na construção de um projeto de genótipo dentro de um sistema CAD baseado em histórico e, em seguida, seus parâmetros sofrem variações aleatórias dentro de limites pré-definidos para gerar um conjunto de projetos distintos. Depois que os *designs* são gerados, eles são filtrados por vários envelopes de restrição, representando a viabilidade geométrica, manufaturabilidade, custo e outras restrições relacionadas ao desempenho, reduzindo o espaço de *design* em um espaço menor e viável, representado por um conjunto de *designs*

diferentes.

Apesar de não ser usado CAD na construção do modelo de *design* generativo deste trabalho, assim como no trabalho Krish [22], buscou-se filtrar os resultados que são dados como saída do *design* generativo a fim de reduzir o espaço de *design*, entregando apenas as saídas que podem trazer alguma contribuição para o trabalho do projetista.

Em sua tese, Reed [35] estuda algumas das aplicações potenciais do aprendizado de máquina no campo do design generativo. A autora analisa como o processo de design pode ser automatizado, uma vez que, com dados dados suficientes sobre o espaço de design, o aprendizado de máquina pode ser usado para encontrar a relação entre um projeto e suas propriedades. O estudo de caso escolhido para o trabalho é o design de cadeiras, onde foi feito usando técnicas de algoritmos evolutivos e árvores de decisão.

Os resultados, segundo a autora, foram bons, com muitos designs viáveis de cadeiras produzidos. A ideia de semelhança visual foi explorada usando esquemas para medir a diferença entre duas cadeiras, demonstrando que é possível usar dados simulados e aprendizado de máquina para tomar decisões de projeto em design generativo.

Por outro lado, muitas das tecnologias que se mascaram como *design* generativo, como otimização de topologia, otimização de treliça, tecnologias paramétricas ou similares, estão focadas em melhorar um projeto preexistente, não criando novas possibilidades de *design*, como no *design* generativo. A confusão surge porque seus parâmetros de entrada são semelhantes às entradas para muitas ferramentas de otimização. No entanto, o *design* generativo produz muitas soluções válidas em vez de uma versão otimizada de uma solução conhecida.

Algoritmos genéticos são métodos meta-heurísticos inspirados pela seleção natural [8] e são inquestionavelmente os mais dominantes na exploração de *design* computacional [22]. Eles manipulam várias gerações de uma população de projeto, onde todos seus elementos são avaliados por uma função de adequação - também conhecida como função *fitness* - que retorna um valor de aptidão, responsável por indicar o quão apto um indivíduo está para se deslocar para uma nova população e passar pelos processos de mudança genética - sendo geralmente operações de *crossover* e mutação.

A principal razão para escolhermos essa técnica está na variabilidade de resultados que são gerados entre si. Ao definir critérios de seleção alinhados a um determinado problema, uma execução do algoritmo pode levar a resultados que não foram apresentados em uma execução anterior, criando diversos resultados diferentes.

Este trabalho implementa um algoritmo genético como o núcleo da geração de resultados do *design* generativo, aplicando duas técnicas diferentes de mutação, e procura gerar uma grande variedade de resultados. Por outro lado, pode ser difícil para o *designer* de jogos analisar uma grande variedade de soluções. Tendo isso em mente, também propomos, em uma segunda etapa, usar uma abordagem de aprendizado de máquina para melhorar a filtragem das possíveis soluções.

Capítulo 4

O modelo de design generativo

O modelo proposto é constituído de três partes: a especificação de parâmetros, o algoritmo genético e o aprendizado de máquina. Na primeira parte, o usuário - um *game designer* - informa ao modelo os parâmetros essenciais para a criação dos *layouts* que serão dados como resultado. O modelo, em seu estado atual, recebe os seguintes parâmetros: a quantidade de botões, as ações que esses botões executam dentro do jogo, a quantidade máxima de *layouts* que deverá ser gerado pelo modelo e a quantidade de iterações que ele deverá executar.

A segunda parte é responsável por criar os *layouts* baseado nos parâmetros informados pelo usuário. Esses *layouts* serão definidos de forma aleatória no início da execução do modelo, passarão por operadores genéticos, serão avaliados por uma função de aptidão e alguns dos indivíduos serão selecionados para gerar a próxima população. O algoritmo irá executar essas operações n vezes, sendo n o número de iterações definido pelo *designer*. Ao fim das iterações, um conjunto de *layouts* estará definido.

Inúmeros indivíduos poderão ser apresentados nesse momento. Apesar da existência de uma função de aptidão que avalia cada *layout* em uma população, apenas a regra definida por ela não é capaz de julgar se um controle é válido ou não. Por isso, a terceira e última parte do modelo foi definida justamente para que fosse possível apresentar para o *designer* apenas aqueles que tivessem alguma utilidade para ele. Sendo assim, o algoritmo de aprendizado de máquina *Support Vector Machine* (SVM) [43] foi utilizado para, dado um conjunto de regras definidas por um *designer* e aprendidas pelo algoritmo, classificar os indivíduos entre válidos e inválidos, separando-os para que o usuário pudesse analisar uma menor gama de resultados. Após essa etapa, o algoritmo de clusterização K-means é responsável por procurar *layouts* semelhantes entre si e agrupá-los em *clusters*, simpli-

ficando a visualização por parte do *designer* e apresentando a ele conjuntos de soluções válidas.

O desenvolvimento da solução foi realizado utilizando a linguagem baseada em Java chamada '*Processing*' [34], altamente recomendada na comunidade de *design* generativo porque é um *software* gratuito, fácil de usar e acessível.

A Figura 4.1 ilustra o pipeline do modelo de *design* generativo proposto para solucionar o problema. As subseções a seguir detalham as definições feitas no modelo.

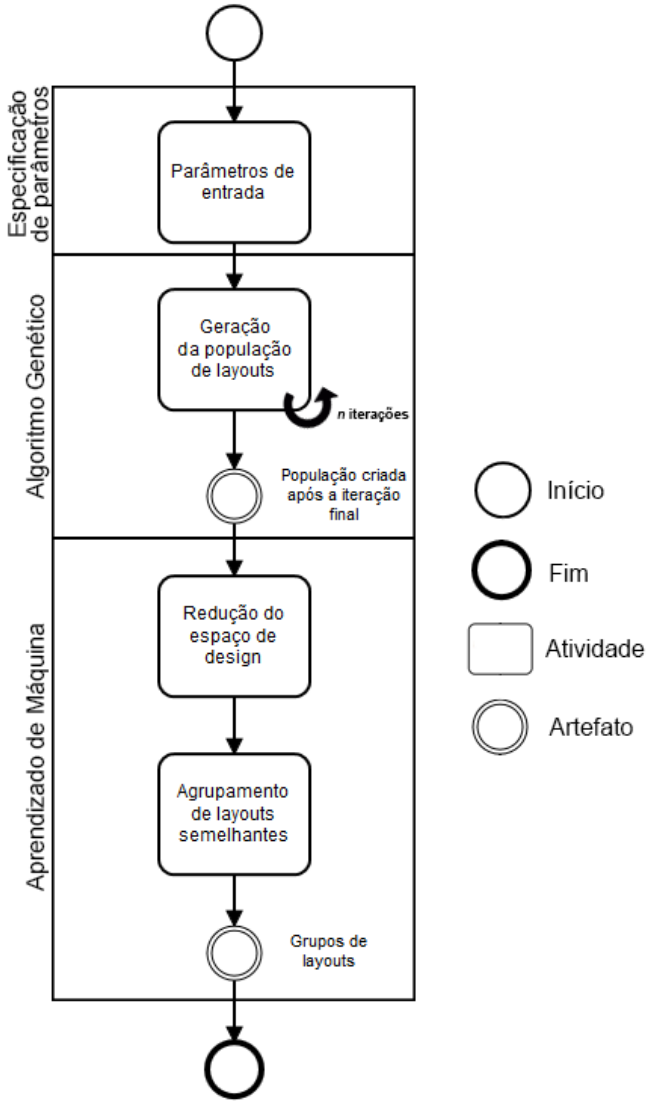

Figura 4.1: O pipeline do modelo de *design* generativo proposto. **Fonte**: O autor (2019).

4.1 Parâmetros de entrada

Para iniciar o desenvolvimento do modelo, foi necessário realizar um estudo prévio sobre os jogos, seus estilos e as maneiras de controlá-los. É dito por Rogers [36] que os jogos digitais se encaixam em um gênero previamente estabelecido, sendo que um jogo pode se encaixar em um ou mais gêneros, onde um é dominante. Podemos citar como exemplo o jogo *The Last of Us* [32] lançado para o PlayStation 3 e remasterizado para o PlayStation 4, classificado como um jogo de aventura contendo *puzzles* e características de um jogo de tiro.

Dito isso, foi feito um levantamento dos gêneros de jogos para que fosse possível conhecer toda ou grande parte das ações que podem ser executadas dentro dele. A Figura 4.2 contém um organograma que exibe os gêneros de jogos de acordo com Rogers [36].

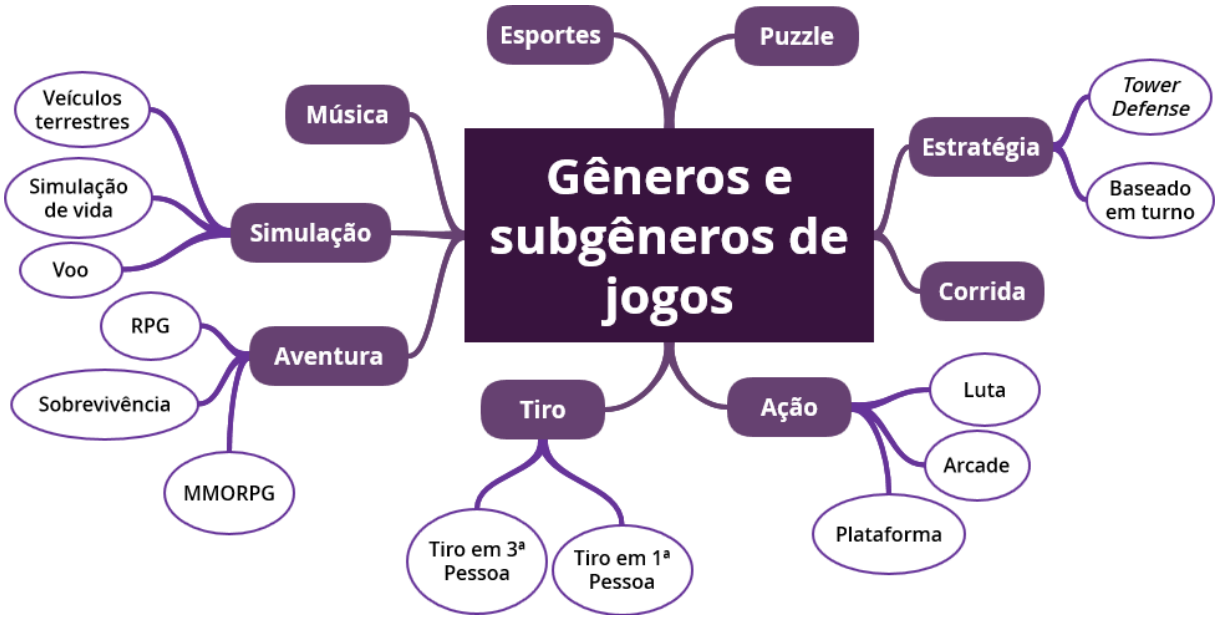

Figura 4.2: Gênero e subgêneros de jogos. **Fonte**: O autor (2019).

Diferentes gêneros de jogos requerem diferentes ações a serem executadas, o que leva a necessidade de existir diferentes botões em um controle para permitir que tais ações sejam executadas. Como foi listado na figura acima, um dos gêneros que um jogo pode ser classificado é "Ação", possuindo um subgênero "Plataforma". Neste subgênero, o jogador deve ser capaz de executar as ações "andar", "correr", "pular" e "agir", sendo essa última uma variável atrelada ao contexto definido na *gameplay* do jogo, como lançar um item, disparar um dispositivo ou executar outra ação. Outro gênero listado é "Tiro", que contém algumas atividades diferentes, como "mirar", "abaixar" e "recarregar munição", por exemplo.

Com o conhecimento sobre os gêneros do jogo, foi possível listar as ações que podem ser realizadas dentro deles. Essas ações - desde a 'andar' até pausar um jogo - são os primeiros parâmetros de entrada que o projetista deverá informar ao modelo.

Os últimos parâmetros especificados estão relacionados ao algoritmo genético: a quantidade máxima de *layouts* que serão gerados pelo modelo implica no tamanho da população do algoritmo genético. Ela é responsável por definir a quantidade máxima de *layouts* que o *designer* espera que o algoritmo gere ao fim das iterações. Já a quantidade de iterações consiste no número de etapas que o algoritmo executará antes de entregar os resultados.

Após a especificação dos parâmetros o algoritmo genético é iniciado, usando um processo análogo à seleção natural.

4.2 Algoritmo genético

Para compreensão desse tópico, é necessário conhecer termos como 'população', 'indivíduo', '*fitness*' e 'roleta'.

Uma população consiste em dois ou mais indivíduos, enquanto um indivíduo consiste em um conjunto de configurações de um controle. Por configuração de controle falamos de um *layout* e seus componentes, como o número de botões, as ações dos botões, suas classificações e suas posições (eixos X e Y) na tela.

Para o desenvolvimento do sistema, algumas estruturas de dados foram definidas. Na estrutura de dados associada a cada indivíduo, definimos três *arrays* chamados de cromossomos, sendo que cada um deles armazena informações essenciais para a criação de indivíduos:

- *chromosomeX*: contém as posições do eixo X de todos os botões solicitados pelo *designer*;
- *chromosomeY*: contém as posições do eixo Y dos botões;
- *classification*: contém a classificação desse botão (o papel que tal botão assume dentro de um controle, qual ação ele exerce).

As classificações mais básicas apresentadas na Tabela 4.1 são um conjunto fechado de opções encontradas no modelo.

Tabela 4.1: Classes de botões e ações básicas. **Fonte**: O autor (2019).

Classes	**Botões**			
Botões direcionais	para cima	para baixo	esquerda	direita
Botões de ação	pular	atirar	correr	defender
Botões extra	L1	L2	R1	R2
Botões de sistema	*start*	*select*	-	-

O tamanho dos *arrays* *chromosomeX*, *chromosomeY* e *classification* é definido no início do processo, quando o usuário informa o número de botões que o controle possuirá. Por exemplo, caso o usuário defina que o controle deverá possuir quatro botões direcionais, dois botões de sistema e dois botões de ação, cada um dos *arrays* assinalados acima terá oito posições, uma para cada botão.

Além dos três *arrays*, cada indivíduo também carrega três atributos:

- *fitness*: um valor numérico responsável por representar o quão apto um indivíduo é para seguir para uma nova população;
- *fitnessPercent*: o valor de aptidão (*fitness*) representado em porcentagem;
- *rouletteTrack*: um *array* de duas posições contendo o intervalo [início, fim] que o indivíduo detém sobre a roleta de seleção de indivíduos.

A Figura 4.3 ilustra sua organização.

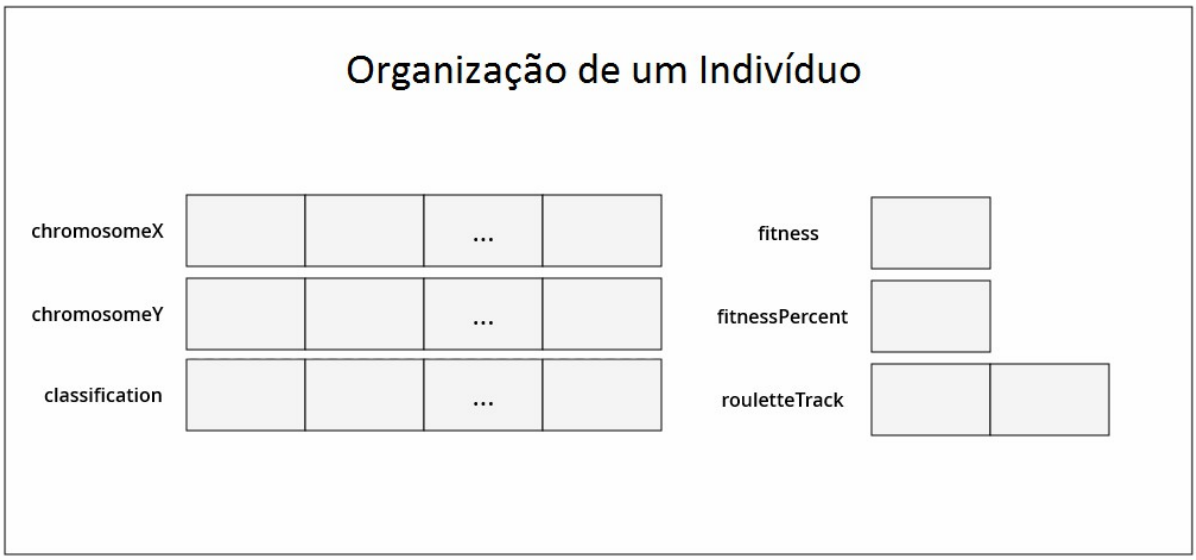

Figura 4.3: Representação da composição de um indivíduo dentro do modelo proposto. **Fonte**: O autor (2019).

Quando um indivíduo é criado, os cromossomos que contêm as posições X e Y dos botões são preenchidos com valores aleatórios, fazendo com que cada botão assuma uma posição qualquer dentro da área de controle. Já o cromossomo *classification* é preenchido com rótulos que representam as ações especificadas na Tabela 4.1.

O valor do atributo *fitness* é definido pela adequação que um indivíduo deve manter nas próximas gerações do algoritmo e é determinado pela seguinte equação:

Seja L um candidato a *layout*, ou seja, um indivíduo na população, com n botões. Sejam bx e by dois *arrays* que armazenam, respectivamente, as coordenadas x e y de cada botão i, $0 \leq i < n$. Além disso, vamos definir bt como uma matriz que identifica o tipo ou classe de um determinado botão i. A função objetivo que define o valor *fitness* de um layout L é dada por

$$f(bx, by, bt) = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} \left[\left(\frac{1 - \delta(bt_i, bt_j)}{l} dist(bx_i, by_i, bx_j, by_j) \right) + \delta(bt_i, bt_j) \left(\frac{1}{1 + dist(bx_i, by_i, bx_j, by_j)} \right) \right] \tag{4.1}$$

onde $\delta(bt_i, bt_j)$ é 1 se os botões i e j pertencerem à mesma classe e 0 se estiverem em classes diferentes. A distância euclidiana entre dois botões é dada por $dist(bx_i, by_i, bx_j, by_j)$; e $l = \sqrt{(w^2 + h^2)}$ é um fator de normalização, onde w e h são a largura e a altura do *layout*. Em resumo, o primeiro termo da soma é responsável por realizar um afastamento entre os botões de diferentes classes, enquanto o segundo termo tenta aproximar os botões na mesma classe.

Definido um valor *fitness* para o indivíduo, o atributo *rouletteTrack* é determinado calculando a proporção que aquele valor tem entre a soma dos valores de *fitness* de toda a população.

A roleta é uma técnica de seleção de indivíduos. Por meio dela, todos indivíduos têm atrelados a eles uma faixa da roleta, sendo esta faixa definida com base em seu valor de aptidão: indivíduos com valores de aptidão mais altos terão faixas maiores na roleta, possuindo maiores chances de serem selecionados para fazer parte da nova população. Embora indivíduos com menores valores de aptidão tenham intervalos menores da roleta, eles ainda têm a chance de serem selecionados para participar da nova população, causando diversidade entre os indivíduos.

Os intervalos da roleta para cada indivíduo são determinados usando o valor *fitness*-

Percent da seguinte forma:

- deve-se ordenar a população em ordem crescente usando os valores de percentual *fitness* como referência;
- depois disso, cada indivíduo é ligado a uma faixa de roleta. O primeiro indivíduo (aquele com menor valor *fitnessPercent*) recebe o intervalo inicial, que varia de zero até o valor que corresponde ao seu *fitnessPercent*. A faixa da roleta do próximo indivíduo se iniciará do valor *fitnessPercent* do indivíduo anterior até a soma desse valor com sua própria porcentagem;
- o passo anterior se repetirá até que 100% das faixas da roleta sejam distribuídas.

Com valores de *fitness* e *fitnessPercent* calculados e tendo definido os intervalos da roleta que cada indivíduo possui, a execução dos operadores genéticos é acionada. Pelo número de iterações especificadas pelo *designer*, as seguintes etapas serão executadas:

- uma nova população é definida (uma iteração é executada);
- para cada indivíduo na população, um valor entre 0 e 100% será sorteado. O indivíduo que for responsável por aquele valor em sua faixa da roleta será selecionado para se submeter às próximas operações;
- operadores de mutação podem ser acionados para modificar o indivíduo;
- o indivíduo selecionado e possivelmente modificado será adicionado em uma nova população, sendo que os indivíduos antigos serão removidos.

As operações de mutação e *crossover* são importantes para fornecer a evolução dos indivíduos. Neste trabalho, não é utilizado nenhum operador *crossover*. Em vez disso, usamos dois operadores de mutação: *mutation1* e *mutation2*, cada um deles tendo um valor de 0 a 1 de probabilidade de ser acionado pelo algoritmo. Definimos a probabilidade 0,5 de uma mutação tipo 1 ocorrer, enquanto a probabilidade de uma mutação tipo 2 ocorrer é 0,8.

A primeira mutação é responsável por alterar a posição de qualquer botão dentro do *layout* gerado. Essa operação calcula um vetor de deslocamento em uma direção aleatória e move aquele botão até uma distância máxima de 100 pixels a partir de sua posição original.

Enquanto isso, a segunda mutação é responsável por alterar a posição de um ou mais botões com outros botões no mesmo *layout*. Por exemplo, por meio dessa mutação, o botão direcional 'para cima' pode mudar de posição com o botão de ação 'pular'. No modelo, definimos regras para que ocorressem mutações entre diferentes botões de diferentes classes de acordo com a quantidade de botões de cada configuração:

- *layouts* com um botão não executam a *mutation2*;
- *layouts* com dois ou três botões trocam apenas dois botões entre si (cada um de uma classe diferente);
- *layouts* com mais de três botões trocam quatro botões entre si (dois botões de uma classe com dois botões de outra classe).

A Figura 4.4 ilustra, respectivamente, a *mutation1* e a *mutation2*

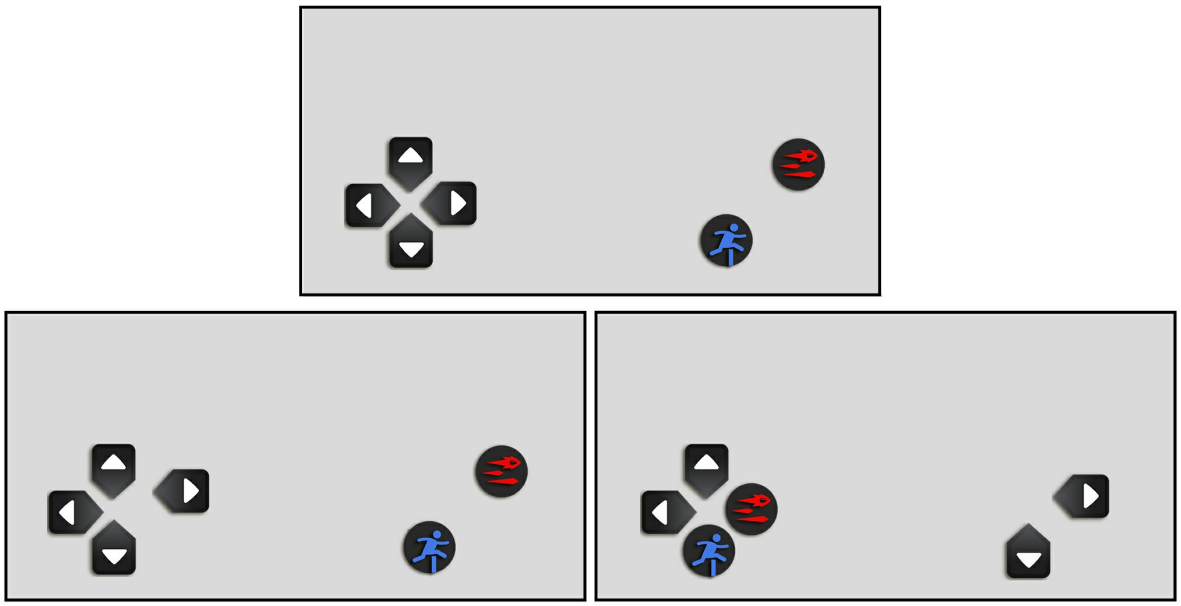

Figura 4.4: Exemplo de layout padrão seguido das mudanças ocorridas após o uso dos operadores *mutation1* e *mutation2*, respectivamente. **Fonte**: O autor (2019).

4.2.1 Filtragens e correções

Por fim, algumas filtragens e correções são realizadas. A primeira consiste na verificação e correção de colisão entre os botões para que um botão não se sobreponha a outro dentro de um *layout*. Uma função que verifica a colisão entre os botões é acionada para comparar

se a área de dois botões não se intersectam em algum momento. Caso isso ocorra, um dos botões é movido em alguma direção aleatória até que eles não colidam mais.

Outra verificação trata as posições dos pares de botões para cima/para baixo e esquerda/direita, como mostra a Figura 4.5. Para que um controle seja passível de uso por um jogador, a disposição dos botões na área do controle deve respeitar uma semântica como, por exemplo, o fato da ação de movimentar um personagem para a esquerda ser inconscientemente realizado ao se clicar no botão mais a esquerda na região dos botões direcionais. Baseado nesta regra, o primeiro *layout* não deverá ser gerado pelo modelo.

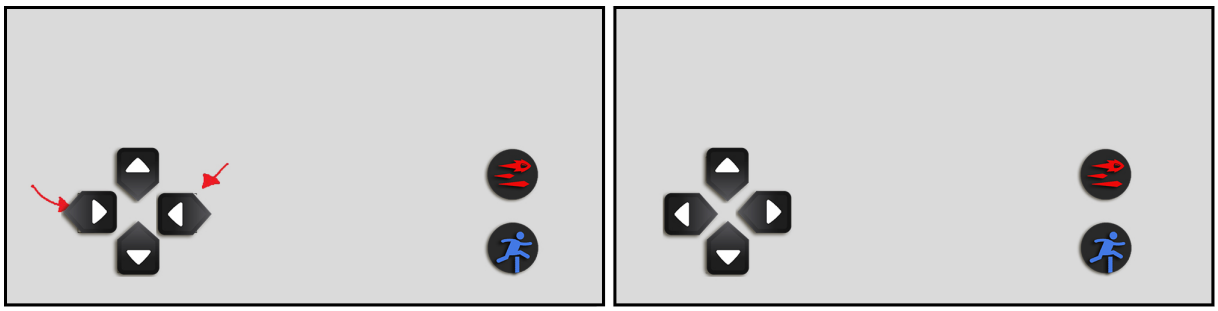

Figura 4.5: Variação inválida de uma configuração de controles apresentado na figura à esquerda, enquanto a figura à direita apresenta sua versão corrigida. **Fonte**: O autor (2019).

No final das n iterações especificadas, a população final será definida. Essa população contará com a quantidade de indivíduos que o *designer* definiu como um dos parâmetros de inicialização do modelo. Caso seja determinado um número muito grande de indivíduos a serem gerados, muitos resultados serão confeccionados e, consequentemente, o *designer* terá que analisar todos eles. Visto que analisar uma grande quantidade de indivíduos demanda tempo e esforço por parte do profissional, buscamos uma solução que pudesse analisar e determinar quais indivíduos realmente merecem a atenção dele, assim, otimizando seu trabalho e entregando possíveis candidatos a serem *layouts* finais para o controle. Sendo assim, a subseção a seguir apresenta a determinação da classificação de indivíduos usando o técnicas de aprendizado de máquina.

4.3 Aprendizado de máquina, a classificação de indivíduos e a variabilidade entre os resultados

As possibilidades de resultados gerados pelo algoritmo genético são amplas. Milhares de resultados podem ser gerados em um curto tempo, porém isso nem sempre é interessante para o *designer*.

Após a apresentação dos resultados gerados pelo algoritmo evolutivo, foi percebido que alguns casos não eram interessantes o suficiente para serem mostrados para o usuário. Um dos casos é especificado na Figura 4.6, onde os botões direcionais não respeitam nenhuma lógica organizacional presente em controles de jogos e não apresentam uma disposição intuitiva para interação, o que pode gerar grande esforço cognitivo por parte do jogador ao tentar executar ações em um jogo. Sendo assim, não há a necessidade de apresentar essa configuração para o *designer*, pois certamente ela será descartada.

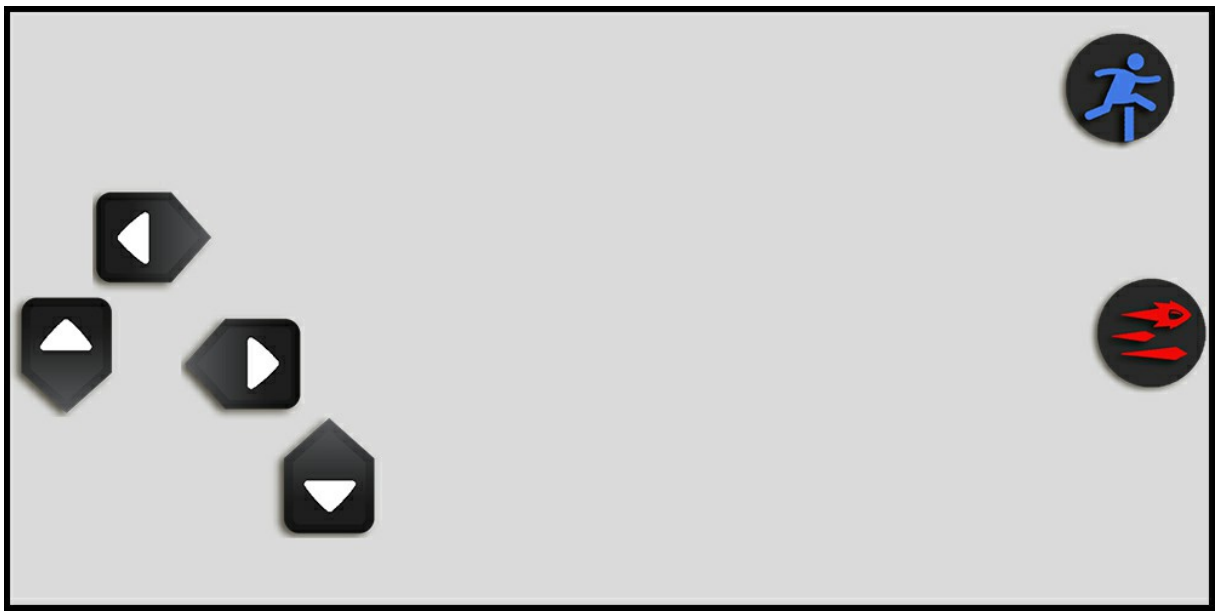

Figura 4.6: Exemplo de configuração inválida gerada pelo modelo de *design* generativo. **Fonte**: O autor (2019).

Ainda que exista uma função objetivo no algoritmo genético responsável por avaliar um indivíduo, apenas o valor resultante dessa função não é suficiente para determinar se um indivíduo é um bom *layout* de controle. Isso pode ser afirmado pois a função de objetivo desenvolvida apenas avalia as distâncias entre classes de botões, mas não leva em consideração a semântica que os botões em questão carregam com eles. Algumas regras de correção e filtragem também foram aplicadas para corrigir outros casos, como detalhado na subseção 4.2.1, porém não é desejado que muitas regras sejam associadas na etapa do algoritmo genético neste modelo, pois isso resultaria em uma limitação criativa do modelo, impedindo que algumas soluções fossem criadas.

Dito isso, a busca por uma solução que aplicasse rótulos aos resultados fornecidos pela solução genética era necessária. Sendo assim, o algoritmo de classificação *Support Vector Machine* (SVM) proposta por Vapnik [43] foi escolhido, onde, por meio de amostras de treinamento rotuladas, é capaz de gerar uma superfície de separação de elementos. O

SVM geralmente precisa de um grande número de vetores de suporte para gerar uma superfície de separação, a fim de tratar bem de um problema complicado de separação não linear. Isto inevitavelmente aumenta a carga computacional do SVM na classificação de novas amostras, uma vez que é dispendioso computacionalmente calcular a função de decisão com muitos parâmetros diferentes de zero [45].

Essa técnica de classificação foi escolhida após alguns testes com outras técnicas e algoritmos, como árvore de decisão, apresentada no trabalho de Reed [35] e *matching* de imagens. Ao usar o SVM, foi possível alcançar resultados esperados e, por isso, manteve-se a utilização dessa técnica.

4.3.1 Treinamento da base

Para o modelo, duas classificações são feitas: indivíduos válidos e indivíduos inválidos. Os primeiros são aqueles que podem ser escolhidos pelo *designer* como candidatos a configurações de controle de jogos, dada a posição e agrupamento dos botões na área do controle, enquanto a segunda classificação é formada por *layouts* que não representam configurações que podem ser usadas pelo jogador, o usuário final.

Após análises de várias configurações de controles de jogos dadas como saída pelo algoritmo genético, alguns critérios foram observados e determinados como essenciais para a criação de um *layout* válido ou inválido. São eles:

- O botão 'para cima' deve ser o botão direcional que está mais ao topo em relação aos outros botões direcionais, assim como botão 'para baixo' deve ser o mais abaixo, o 'direita' deve ser o mais à direita e o 'esquerda' deve ser o mais à esquerda. Consequentemente, caso haja um alinhamento vertical entre botões 'direita' e 'esquerda' (um estar posicionado acima do outro) ou um alinhamento horizontal entre botões 'para cima' e 'para baixo' (um estar ao lado do outro), este *layout* será inválido;
- Caso a maioria dos botões estejam concentrados na área central do controle, este *layout* será inválido;
- Caso os algum botão esteja localizado na extremidade do controle, tocando a borda da área do controle, este *layout* será inválido;
- Caso hajam botões que tocam suas bordas entre si, este *layout* será inválido.

Como dito anteriormente, é necessário que exista uma base de dados de treinamento

para que seja possível avaliar outras bases. Para a construção das bases de treinamento, foram usadas populações contendo três, seis e nove botões geradas pelo algoritmo genético. Essas quantidades de botões foram determinadas para que fosse possível observar e analisar o comportamento criativo do modelo para controles com uma menor e uma maior quantidade de botões associada a ele.

Como a base de dados se baseia nas posições de todos os botões na área do controle, foi necessário classificar conjuntos que contivessem diferentes quantidades de botões para que, quando uma configuração fosse definida pelo *designer*, o classificador relativo àquela quantidade de botões seja ativado. Em resumo, uma configuração que possua seis botões na área do controle deverá ser avaliada pelo classificador que foi treinado com seis botões.

Cada base de dados foi rotulada manualmente por um *game designer*, sendo elas arquivos CSV com as posições x e y de cada botão presente na tela e, ao final do conjunto de posições, foi adicionada a qualidade daquele conjunto, representada pelo rótulo 'válido' ou 'inválido'. Feito isso, os arquivos CSV são dados como entrada para o classificador onde, por meio de uma validação cruzada, foi calculada a acurácia da predição do algoritmo.

Um dos cuidados necessários a se tomar diz respeito a base balanceada. É importante que exista uma quantidade balanceada de indivíduos rotulados com ambas as classificações pois, caso hajam mais configurações válidas que inválidas ou vice-versa, o classificador tenderá a ficar enviesado.

Após a conclusão de uma geração do algoritmo genético, o conjunto de vetores com suas posições é avaliado por uma base de dados treinada relativa a quantidade de botões presentes na área do controle. Tendo classificado um a um, o modelo exibe os resultados para o *designer*, separando-os em pastas diferentes, relativas às classificações que o SVM determinou.

4.3.2 Agrupamento de *layouts* semelhantes

Ainda que os resultados sejam classificados em válido e inválido, para o *designer*, é interessante que haja uma grande variabilidade de resultados gerados pelo modelo. A variabilidade implica na diversidade de resultados disponibilizados para ele, a fim de que seja possível testar diversas configurações válidas e encontrar aquela ou aquelas que mais se adéquam à necessidade do projetista ao criar um controle ideal para seu jogo.

Dito isto, buscou-se encontrar uma solução tornasse possível agrupar *layouts* semelhantes para tornar possível medir a variabilidade dos resultados válidos gerados. O

algoritmo de clusterização K-means [21] foi escolhido por ser um algoritmo que particiona dados em *clusters* baseando-se em fatores similares que esses dados compartilham. Neste trabalho, essa técnica é responsável por agrupar *layouts* com características semelhantes em um mesmo *cluster*.

O algoritmo de clusterização K-means é um método comumente usado para particionar automaticamente um conjunto de dados em k grupos. A partir de um grupo, ele prossegue selecionando k centros de agrupamentos iniciais e, em seguida, os refinando iterativamente [44]. O objetivo do agrupamento de dados é descobrir o agrupamento natural de um conjunto de padrões, pontos ou objetos para que, então, seja possível fazer análises e absorver conhecimentos intrínsecos aos dados [21]. A definição mais tradicional dessa técnica não é capaz de determinar o número ideal de *clusters*, sendo necessário informar previamente o número de agrupamentos que se deseja separar os dados.

Para definir o número de *clusters* e encontrar uma quantidade de agrupamentos que pudesse ser ideal para a medição da variabilidade dos resultados, nós utilizamos uma extensão do K-means que utiliza o *Bayesian Information Criterion* (BIC) [13]. Ele baseia-se, em parte, na função de verossimilhança e pode ser usado para comparar modelos com diferentes parametrizações, com diferentes números de componentes ou ambos.

Como entrada para o algoritmo, é fornecido um arquivo CSV contendo as posições nos eixos x e y de cada botão concatenadas com as distâncias entre cada um dos botões. Após definir o número de *clusters*, o algoritmo é responsável por separar os *layouts* em seus respectivos *clusters*. No modelo, o número de agrupamentos indica a quantidade de *layouts* diferentes que foram gerados, o que torna possível calcular a variabilidade das soluções geradas, já que, para o *designer*, não é interessante que sejam entregadas várias soluções parecidas entre si, mas sim, soluções diferentes.

Capítulo 5

Experimentos e resultados

Os experimentos foram realizados buscando descobrir se existe alguma ligação entre o número de iterações e a quantidade de indivíduos válidos gerados pelo modelo e, além disso, medir a variabilidade dos resultados. É importante que o modelo forneça uma gama de resultados úteis para o *designer* e que os *layouts* não sejam todos semelhantes entre si.

Primeiramente, foi necessário definir para qual gênero de jogos iremos gerar as configurações dos controles. Partimos do pressuposto que o *game designer* deseja construir controles para jogos do gênero plataforma, onde ações como 'pular', 'correr', 'atirar', 'defender' e 'mover em quatro direções' devem ser possíveis. Sendo assim, um dos controles deve possuir apenas três botões: dois direcionais (esquerda e direita) e um de ação (pular). Outra configuração de controle deve possuir seis botões, sendo eles quatro botões direcionais (para cima, para baixo, direita e esquerda) e dois botões de ação (atirar e pular) e, por fim, um último *layout* de controle deve possuir nove botões: os quatro direcionais citados anteriormente, outros quatro botões de ação (pular, atirar, defender e correr) e um botão de sistema (*start*).

Após o estabelecimento desses parâmetros, foram definidos o número de iterações que o modelo deveria executar até definir a população final e o número de indivíduos que devem ser apresentados nessa população após as n iterações. Foram definidos 10 tipos de iterações diferentes: 1000, 2000, 3000, até 10.000 iterações. Já a quantidade de indivíduos foi estabelecida como 1000, 2000 e 3000 indivíduos para cada configuração de controle com três, seis e nove botões. Foram obtidas um total de 90 populações, como mostra a Tabela 5.1.

Tabela 5.1: Definição dos parâmetros para geração das populações para os experimentos

Quantidade de botões em um layout	3			6			9		
Quantidade de indivíduos em uma população	1000	2000	3000	1000	2000	3000	1000	2000	3000
Quantidade de iterações para gerar cada população	1000	1000	1000	1000	1000	1000	1000	1000	1000
	2000	2000	2000	2000	2000	2000	2000	2000	2000
	3000	3000	3000	3000	3000	3000	3000	3000	3000
	4000	4000	4000	4000	4000	4000	4000	4000	4000
	5000	5000	5000	5000	5000	5000	5000	5000	5000
	6000	6000	6000	6000	6000	6000	6000	6000	6000
	7000	7000	7000	7000	7000	7000	7000	7000	7000
	8000	8000	8000	8000	8000	8000	8000	8000	8000
	9000	9000	9000	9000	9000	9000	9000	9000	9000
	10000	10000	10000	10000	10000	10000	10000	10000	10000

Após criar todos as configurações, cada população final foi submetida a avaliação do classificador SVM que, segundo uma base de dados de treinamento, classificou cada *layout* entre 'válido' ou 'inválido'. Nesse caso, foram utilizadas bases classificadoras específicas para três, seis e nove botões.

Cada botão presente em um controle carrega consigo uma semântica que dá sentido à sua presença no *layout*. Assim sendo, a classificação de indivíduos de diferentes populações foi realizada levando em consideração a posição dos botões na área do controle, os agrupamentos entre eles e as distâncias entre os botões de uma mesma classe - botões direcionais, de ação ou de sistema. Esse conhecimento pertencente ao *designer* é aprendido pelo classificador SVM durante a fase de treinamento.

Para o treinamento da base usada para classificar *layouts* de três botões, foram rotulados 301 indivíduos manualmente, onde 150 deles foram classificados como 'válidos' e 151 como 'inválidos'. O treinamento usando o SVM resultou em uma classificação correta de 259 indivíduos, 86.05% de toda a população. A matriz de confusão do treinamento dessa base pode ser vista na Tabela 5.2.

Para a base relativa a seis botões, 442 indivíduos foram classificados manualmente,

sendo 221 classificados como 'válidos' e 221 como 'inválidos'. O treinamento classificou 86.88% dos indivíduos corretamente. Sua matriz de confusão pode ser vista na Tabela 5.3.

Por fim, uma população de 306 indivíduos foi classificada manualmente para construir a base responsável por classificar configurações de controles com nove botões. Na classificação manual realizada pelo *designer*, 131 indivíduos foram classificados como 'válidos' e 175 como 'inválidos'. Como apresentado na matriz de confusão, exibida na Tabela 5.4, 273 indivíduos foram classificados corretamente, sendo 89.21% das configurações.

Tabela 5.2: Matriz de confusão relativa ao treinamento da base responsável por classificar configurações de controles com três botões. **Fonte**: O autor (2019).

n = 301	**Classificado pelo SVM como INVÁLIDO**	**Classificado pelo SVM como VÁLIDO**
Classificado pelo *designer* como INVÁLIDO	119	32
Classificado pelo *designer* como VÁLIDO	10	140

Tabela 5.3: Matriz de confusão relativa ao treinamento da base responsável por classificar configurações de controles com seis botões. **Fonte**: O autor (2019).

n = 442	**Classificado pelo SVM como INVÁLIDO**	**Classificado pelo SVM como VÁLIDO**
Classificado pelo *designer* como INVÁLIDO	178	43
Classificado pelo *designer* como VÁLIDO	15	206

Tabela 5.4: Matriz de confusão relativa ao treinamento da base responsável por classificar configurações de controles com nove botões. **Fonte**: O autor (2019).

n = 306	**Classificado pelo SVM como INVÁLIDO**	**Classificado pelo SVM como VÁLIDO**
Classificado pelo *designer* como INVÁLIDO	154	21
Classificado pelo *designer* como VÁLIDO	12	119

Tendo os três classificadores, cada população resultante da execução do modelo de *design* generativo foi submetida a avaliação do classificador SVM respectivo ao seu *layout*. Após a avaliação, os indivíduos classificados como 'válidos' são separados dos classificados como 'inválidos'.

Ao analisar os *layouts* classificados como 'válidos' pelo classificador SVM, foi possível perceber que esses resultados seguem as regras definidas pelo *designer* ao classificar a base de dados de treinamento, assim como os resultados classificados como inválidos também o seguem. As Figuras 5.1, 5.2 e 5.3 apresentam alguns dos resultados classificados como 'válidos' pelo classificador, enquanto as Figuras 5.4, 5.5 e 5.6 ilustram alguns desses resultados inválidos.

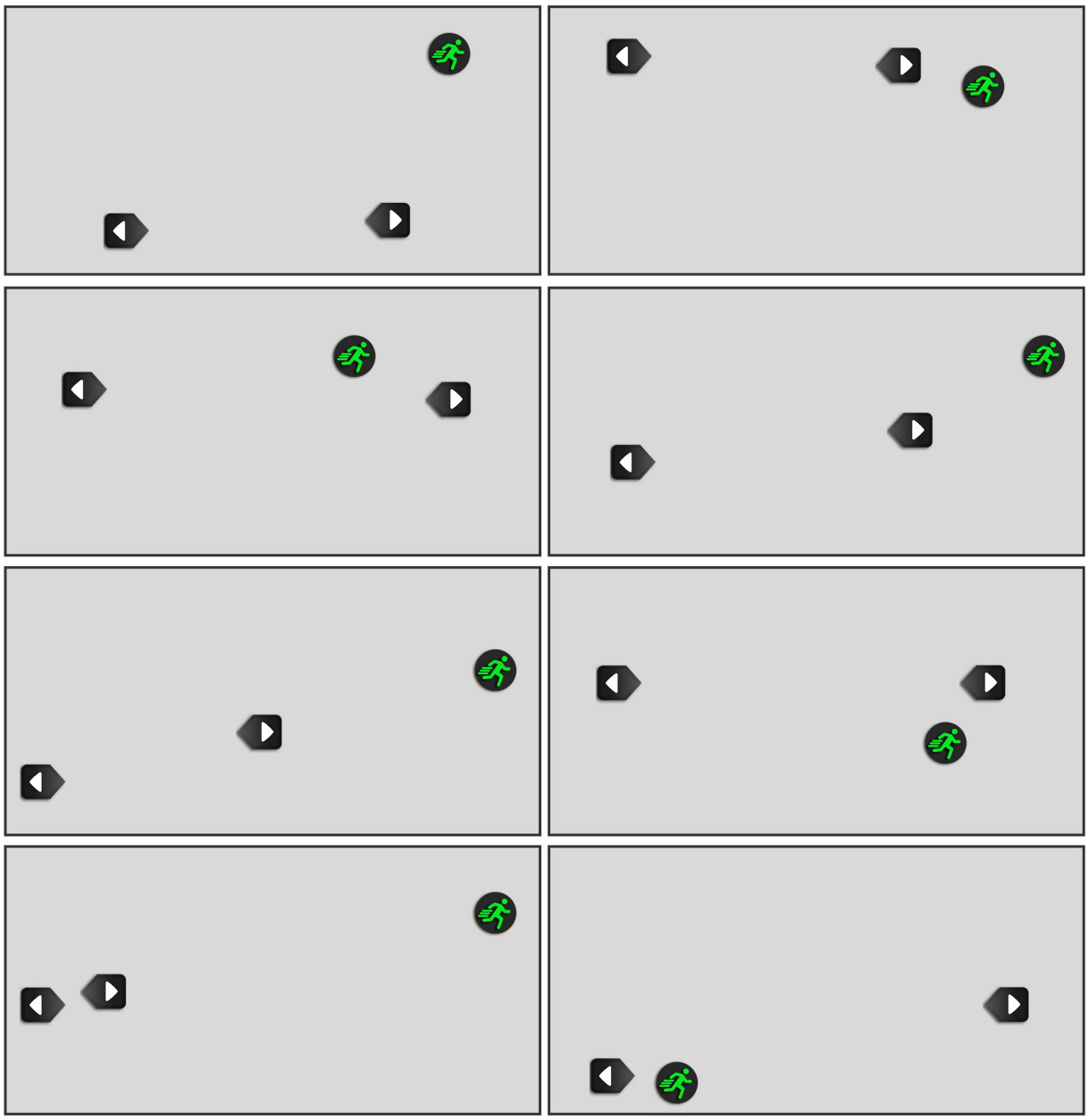

Figura 5.1: Exemplos de *layouts* com três botões classificados como 'válidos' pelo classificador. **Fonte**: O autor (2019).

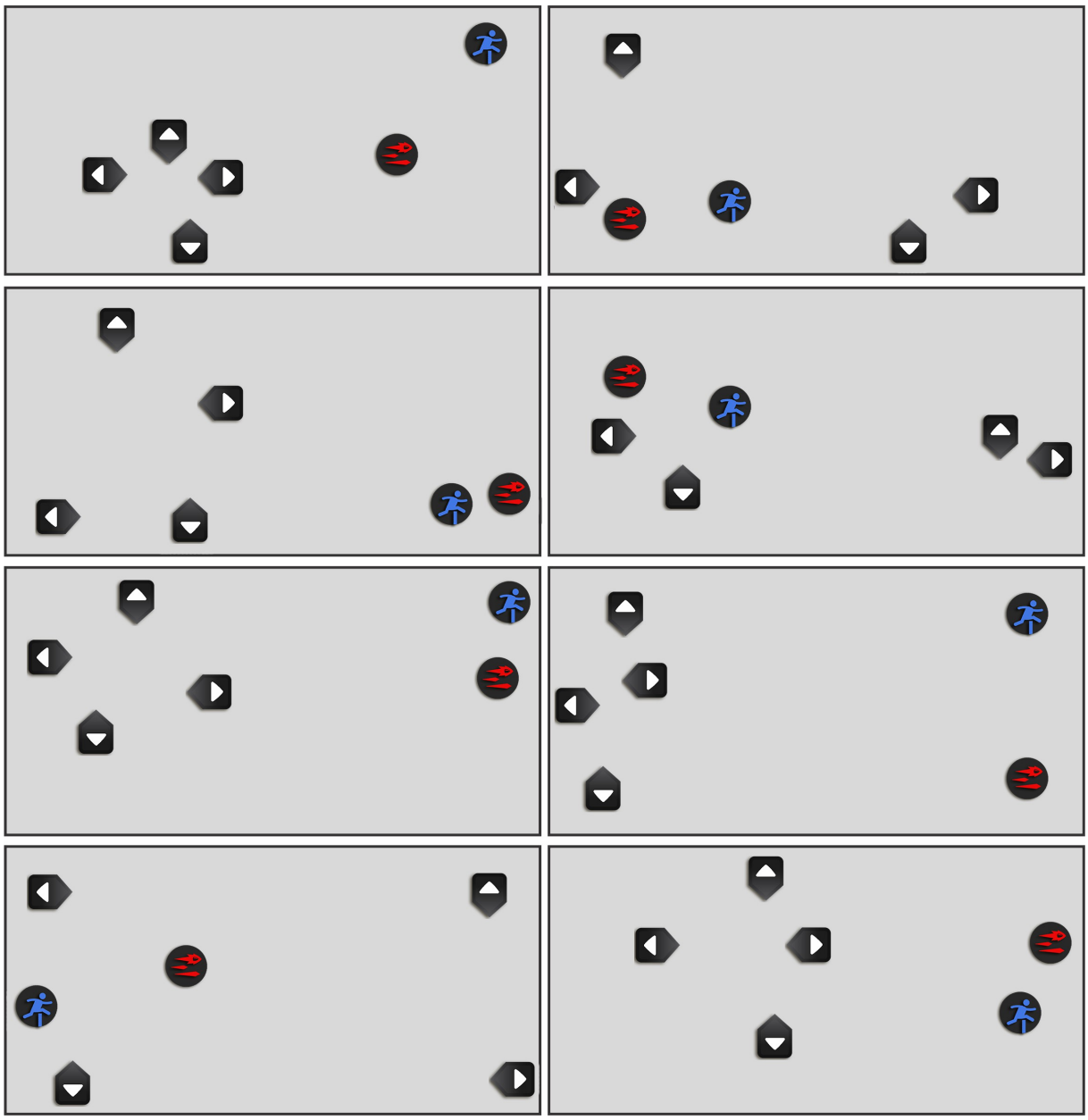

Figura 5.2: Exemplos de *layouts* com seis botões classificados como 'válidos' pelo classificador. **Fonte**: O autor (2019).

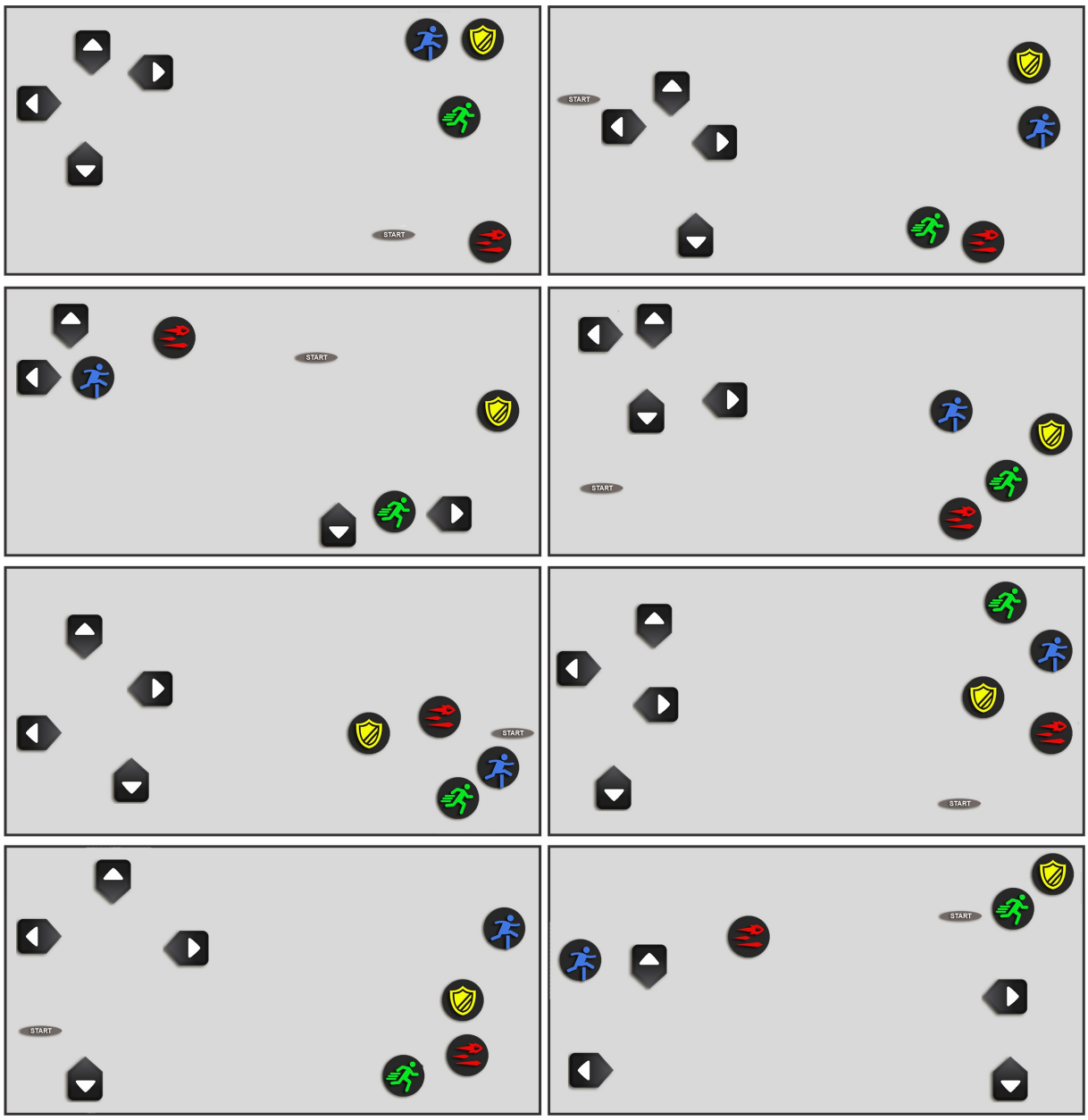

Figura 5.3: Exemplos de *layouts* com nove botões classificados como 'válidos' pelo classificador. **Fonte**: O autor (2019).

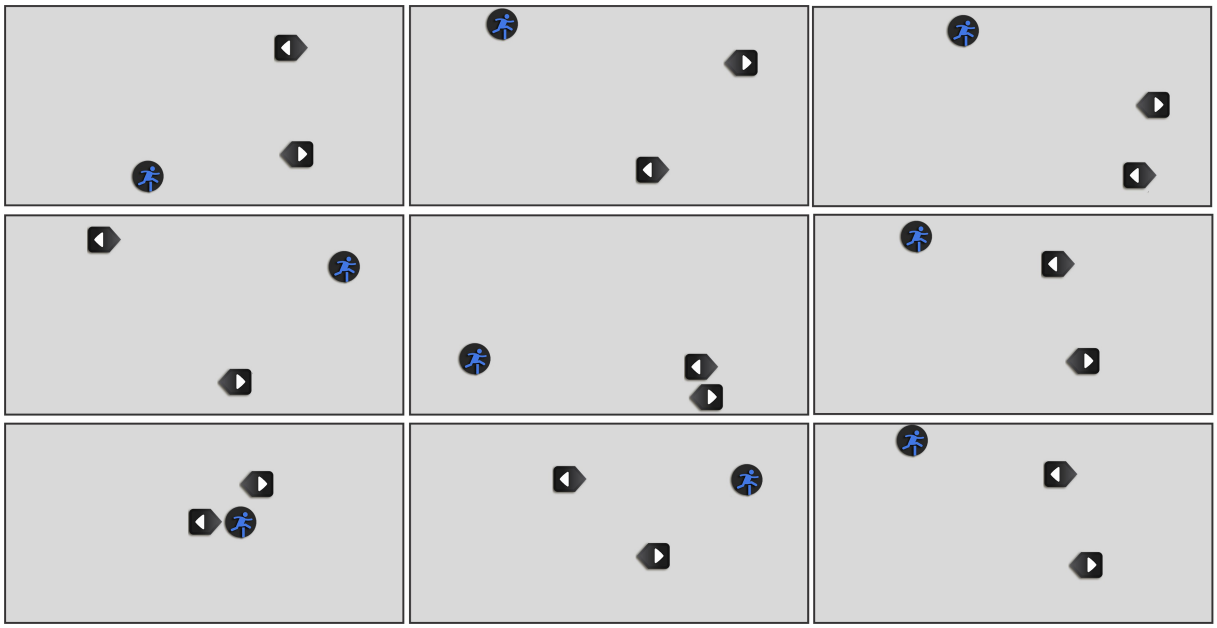

Figura 5.4: Alguns *layouts* com três botões classificados como 'inválidos'. **Fonte**: O autor (2019).

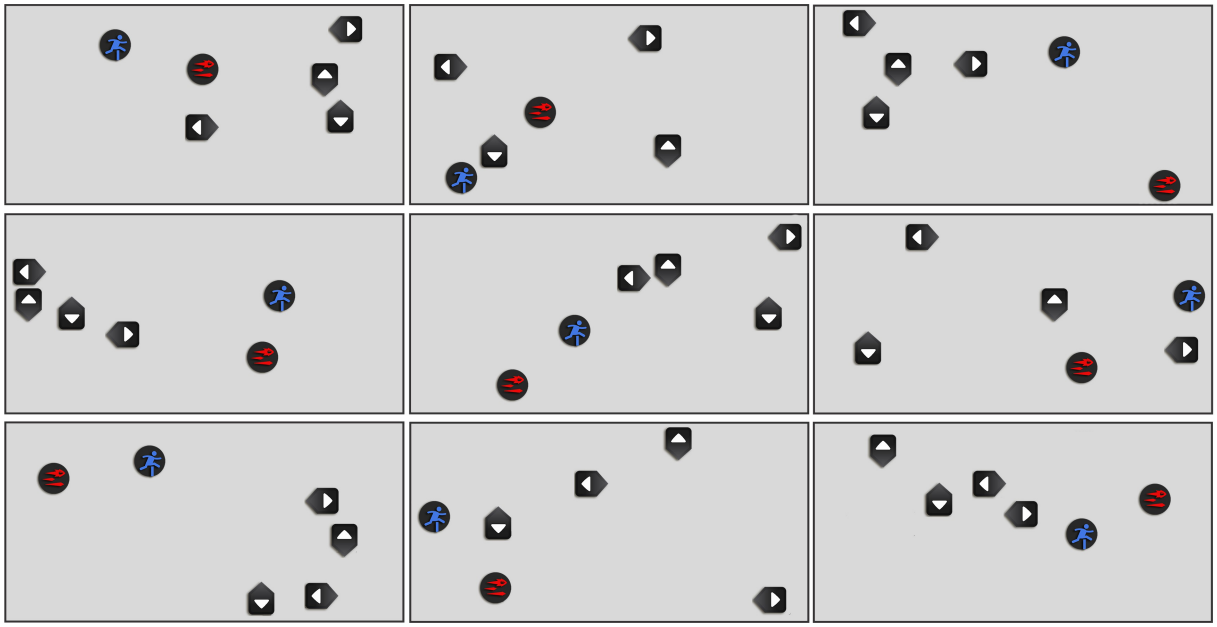

Figura 5.5: Alguns *layouts* com seis botões classificados como 'inválidos'. **Fonte**: O autor (2019).

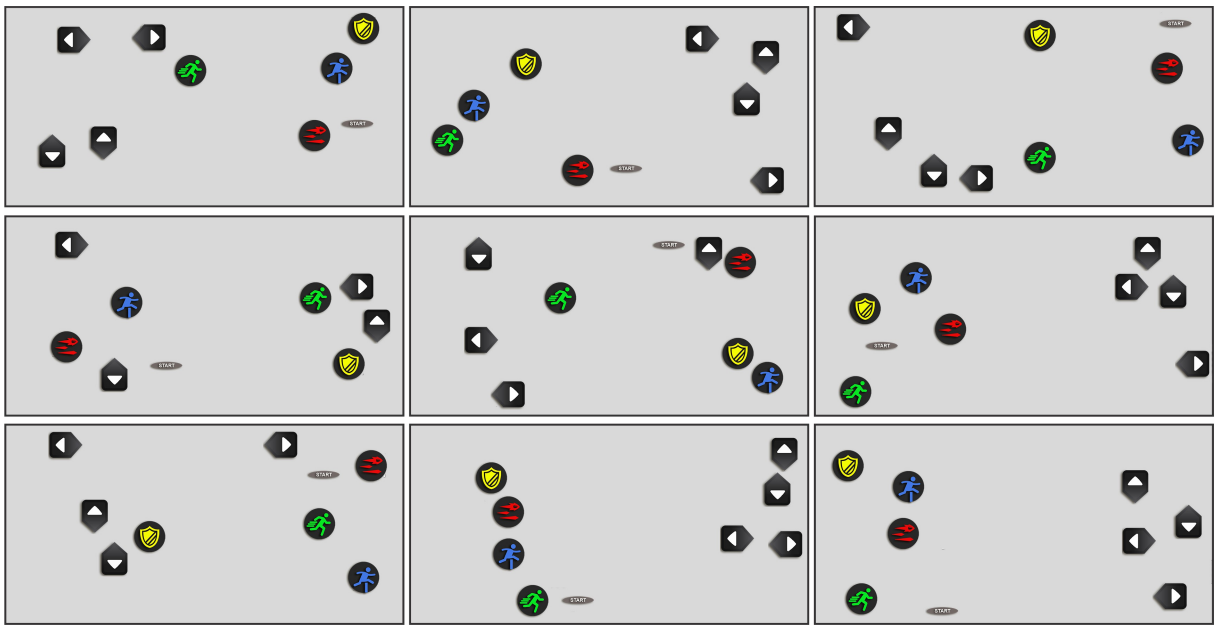

Figura 5.6: Alguns *layouts* com nove botões classificados como 'inválidos'. **Fonte**: O autor (2019).

5.1 Análise de variabilidade dos *layouts* gerados e classificados

Para encontrar o número ideal de *clusters* e, por fim, separar os *layouts* em agrupamentos semelhantes, foi utilizado o algortimo de clusterização K-means. As Tabelas 5.5, 5.6 e 5.7 apresentam, para cada população, a quantidade de indivíduos válidos determinados pelo classificador SVM, seguido da quantidade de *clusters* que os indivíduos classificados como válidos foram agrupados pelo K-means.

Tabela 5.5: Quantidade de indivíduos válidos e *clusters* em populações com 1000, 2000 e 3000 indivíduos - *layouts* contendo três botões. **Fonte**: O autor (2019).

	1000 indivíduos		2000 indivíduos		3000 indivíduos	
iterações	válidos	*clusters*	válidos	*clusters*	válidos	*clusters*
1000	333	11	738	14	1049	19
2000	198	7	638	17	828	20
3000	373	14	832	15	770	17
4000	332	7	438	13	1212	15
5000	415	13	759	17	1159	24
6000	348	11	525	16	1158	17
7000	338	14	551	14	1153	22
8000	384	11	493	12	1145	22
9000	339	10	813	15	1196	10
10.000	297	10	542	14	895	20

Tabela 5.6: Quantidade de indivíduos válidos e *clusters* em populações com 1000, 2000 e 3000 indivíduos - *layouts* contendo seis botões. **Fonte**: O autor (2019).

	1000 indivíduos		2000 indivíduos		3000 indivíduos	
iterações	válidos	*clusters*	válidos	*clusters*	válidos	*clusters*
1000	180	7	534	15	849	14
2000	344	11	379	10	835	20
3000	148	4	256	9	364	10
4000	364	11	529	13	833	14
5000	110	7	536	15	808	18
6000	268	10	323	9	803	15
7000	248	9	721	17	772	18
8000	257	9	595	13	1026	21
9000	134	7	547	13	469	14
10.000	128	7	528	12	469	13

Tabela 5.7: Quantidade de indivíduos válidos e *clusters* em populações com 1000, 2000 e 3000 indivíduos - *layouts* contendo nove botões. **Fonte**: O autor (2019).

	1000 indivíduos		2000 indivíduos		3000 indivíduos	
iterações	válidos	*clusters*	válidos	*clusters*	válidos	*clusters*
1000	58	3	82	4	199	5
2000	48	3	107	4	152	4
3000	41	3	345	6	212	5
4000	54	3	104	7	800	12
5000	74	4	169	4	605	11
6000	88	4	739	13	174	9
7000	296	7	393	10	599	9
8000	70	5	111	5	599	9
9000	51	5	620	9	160	6
10.000	349	6	125	3	1007	13

Foi observado que tanto o número de *clusters* quanto a quantidade de indivíduos válidos oscilam quando o número de iterações varia para cada população. Isso pode ser explicado em parte pelo fato da função objetivo não ser capaz de capturar todos os aspectos de um bom *layout*, logo, esse resultado justifica o uso de aprendizado de máquina para ajudar o projetista a identificar um bom *design*. A natureza estocástica da seleção de indivíduos pode propagar, embora com menor probabilidade, indivíduos com baixo valor *fitness* para as gerações subsequentes, que podem então ser classificados como válidos pelo SVM, já que a técnica não garante 100% de acurácia na classificação de indivíduos.

Para compreender melhor a relação entre a quantidade de indivíduos 'válidos' e a quantidade de *clusters* possíveis, foi calculada a razão entre o valor de *clusters* e o de indivíduos 'válidos'.

Suponhamos que um conjunto de 100 indivíduos válidos foi organizado em 10 *clusters*, enquanto um conjunto de 200 indivíduos válidos também foi organizado em 10 *clusters*. Desta forma, o primeiro conjunto obteve uma razão de 0.10 *clusters*/indivíduos válidos, enquanto o segundo conjunto obteve uma razão inferior, 0.05 *clusters*/indivíduos válidos. Se aumentarmos o número de indivíduos em uma população, mas a variabilidade deles for igual ou menor que populações com uma quantidade inferior de indivíduos, esses resultados não acrescentarão novas informações para o *designer*. Como afirmamos anteriormente, temos interesse em apresentar uma gama de resultados que sejam distintos entre si para o usuário final. As Figuras 5.7, 5.8 e 5.9 ilustram graficamente razões calculadas em cada

população criada pelo modelo.

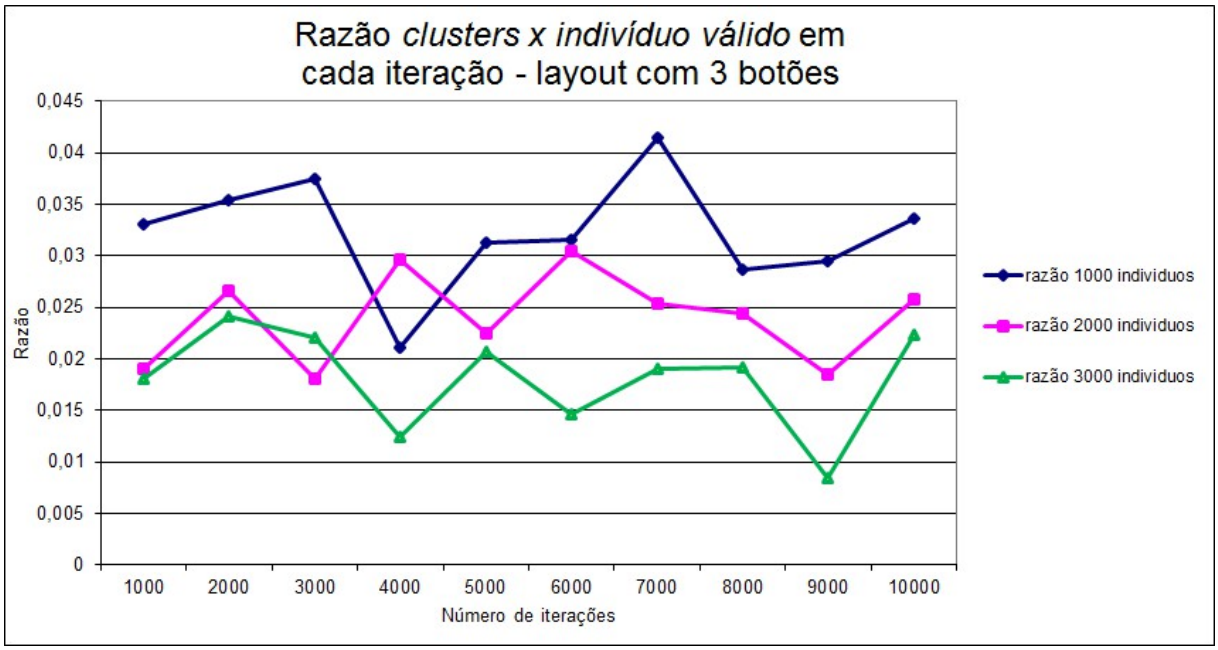

Figura 5.7: Representação gráfica do valor da razão entre o número de *clusters* e a quantidade de indivíduos classificados como 'válidos' em uma população de 1000, 2000 e 3000 indivíduos - *layouts* contendo três botões. **Fonte**: O autor (2019).

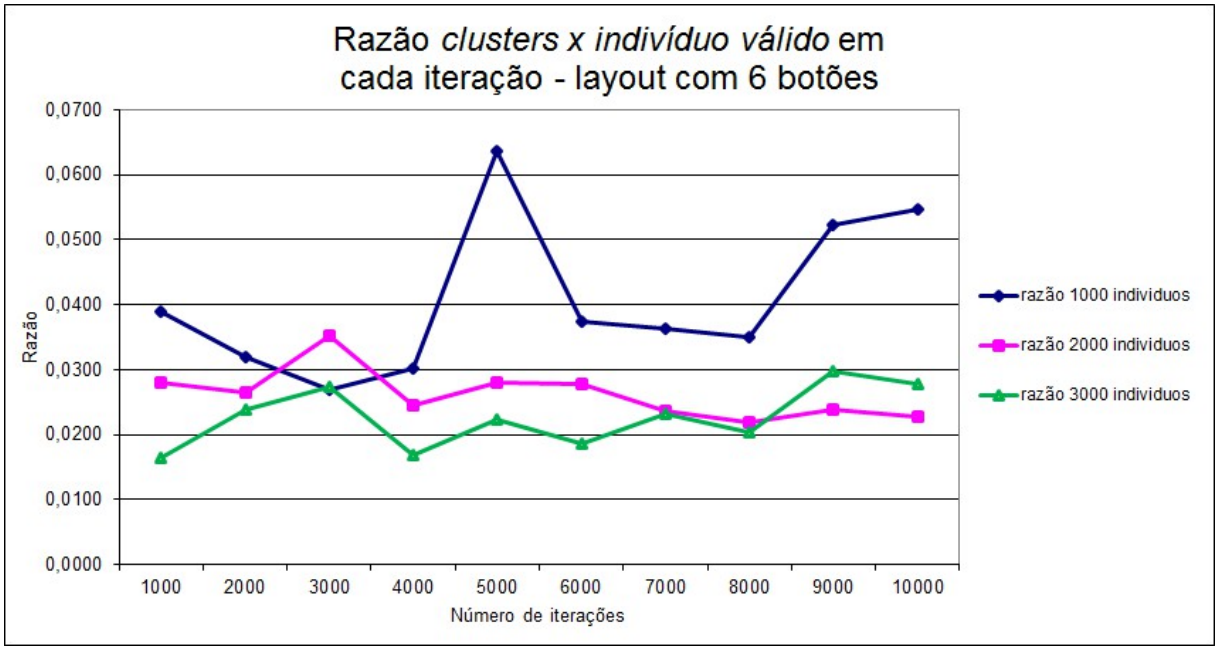

Figura 5.8: Representação gráfica do valor da razão entre o número de *clusters* e a quantidade de indivíduos classificados como 'válidos' em uma população de 1000, 2000 e 3000 indivíduos - *layouts* contendo seis botões. **Fonte**: O autor (2019).

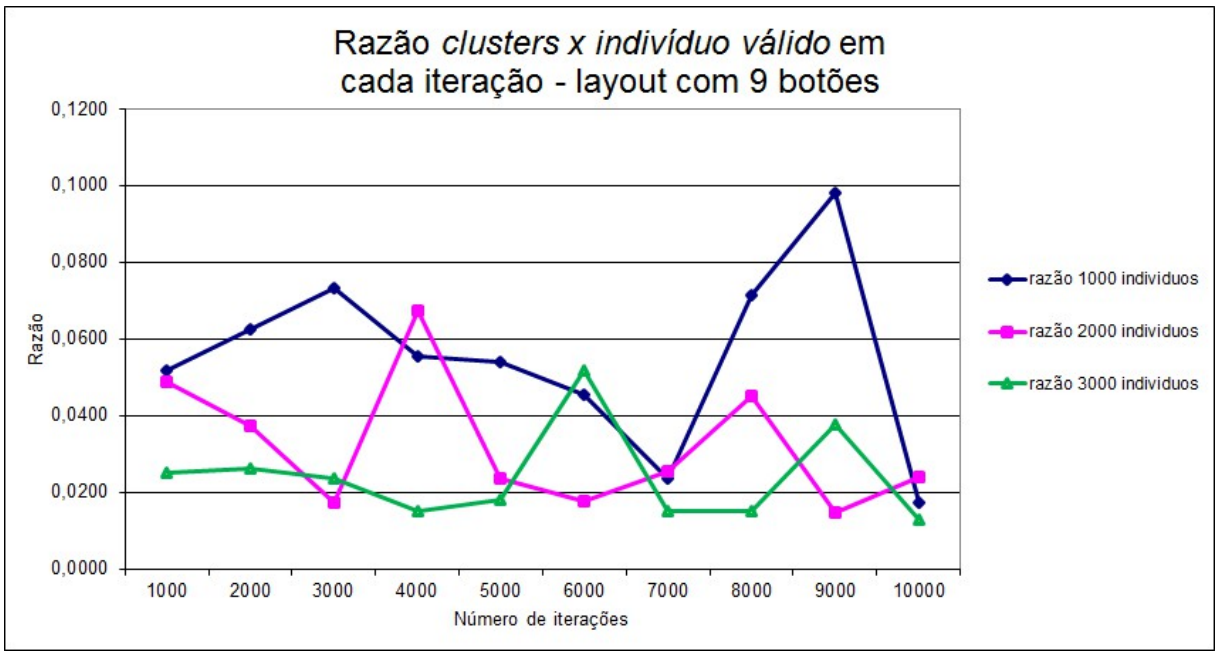

Figura 5.9: Representação gráfica do valor da razão entre o número de *clusters* e a quantidade de indivíduos classificados como 'válidos' em uma população de 1000, 2000 e 3000 indivíduos - *layouts* contendo nove botões. **Fonte**: O autor (2019).

Analisando os gráficos, podemos perceber que, em grande parte das vezes, a razão de *clusters* por indivíduos válidos é maior na população de 1000 indivíduos. Isso acontece em todas as três configurações contendo diferentes quantidades de botões, exceto nas seguintes iterações:

- iteração 4000 no *layout* com três botões;
- iteração 3000 no *layout* com seis botões;
- iterações 4000, 6000, 7000 e 10.000 no *layout* com nove botões.

Como esse comportamento ocorreu em poucos casos, muito provavelmente causado pela randomicidade do algoritmo genético ao determinar indivíduos, percebemos que uma população com 1000 indivíduos é suficiente para gerar conjuntos de indivíduos válidos com uma maior variabilidade.

Apesar dessa constatação, existe a hipótese de que a função objetivo do algoritmo genético poderia ser responsável pela restrição da criatividade do modelo, já que ela tende a aproximar botões de mesma classe e afastar botões de classes diferentes. Estamos cientes de que ela pode criar um viés no processo de validação, pois ele orienta a criação de soluções candidatas. Para testar essa hipótese, resolvemos executar um outro teste: para cada

configuração de controle (três, seis e nove botões), foram geradas populações com 10.000 indivíduos. Cada uma delas passou pela mesma quantidade de iterações estabelecidas no teste anterior, porém os indivíduos dessas populações não foram avaliadas por uma função objetivo. As probabilidades de 0.5 da *mutation1* e 0.8 da *mutation2* ocorrerem para cada indivíduo se mantiveram. Ao fim de cada geração, a classificação pelo SVM e a clusterização foram executadas e as razões de variabilidade foram calculadas. A Tabela 5.8 exibe as quantidades de indivíduos válidos e *clusters* nas populações com 10.000 indivíduos para três, seis e nove botões, enquanto as Figuras 5.10, 5.11 e 5.12 apresentam os gráficos apresentados anteriormente, porém desta vez contendo o valor da razão dessas populações adicionadas a eles.

Tabela 5.8: Quantidade de indivíduos válidos e *clusters* em populações com 10.000 indivíduos que não foram avaliadas pela função objetivo - *layouts* com três, seis e nove botões.

	Três botões		Seis botões		Nove botões	
iterações	válidos	*clusters*	válidos	*clusters*	válidos	*clusters*
1000	2387	34	1899	30	1063	23
2000	3104	41	1821	30	1299	21
3000	2512	34	1980	29	1093	21
4000	2150	33	2171	31	1934	31
5000	2084	32	2075	31	1357	20
6000	1827	32	1591	23	1417	24
7000	2647	38	2074	32	1468	22
8000	2186	36	1678	26	1264	20
9000	2658	39	1944	25	1602	25
10.000	2286	34	1972	30	1212	24

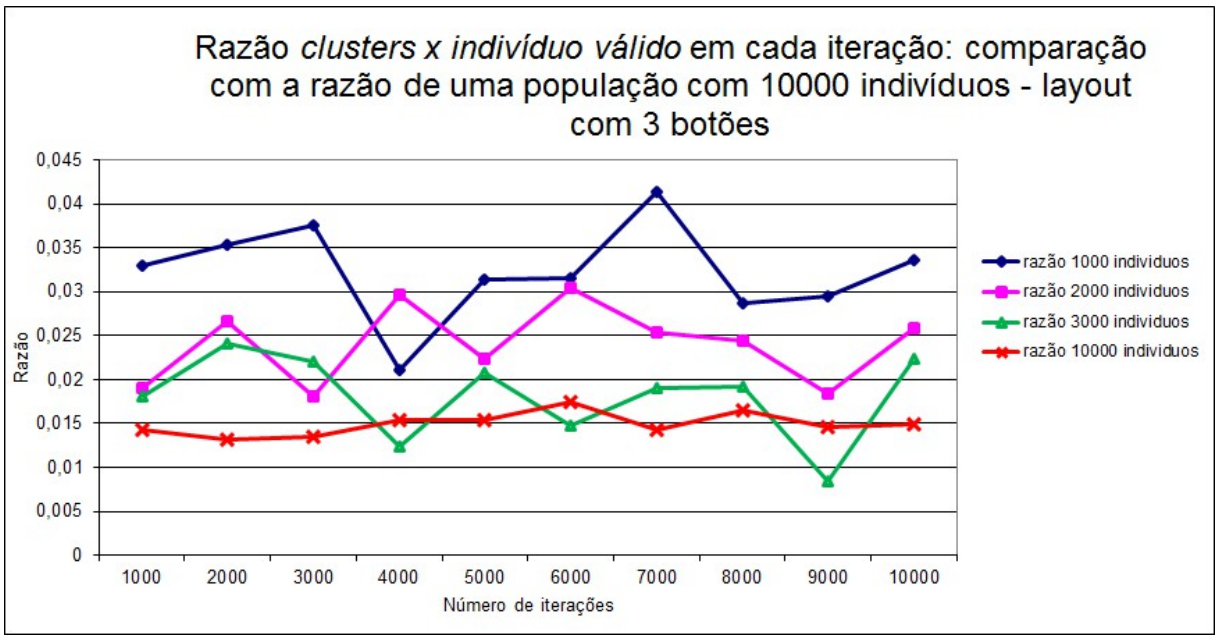

Figura 5.10: Razões de *clusters* por indivíduos válidos contendo os dados de populações com 10.000 indivíduos que não foram avaliadas por uma função objetivo - *layouts* com três botões. **Fonte**: O autor (2019).

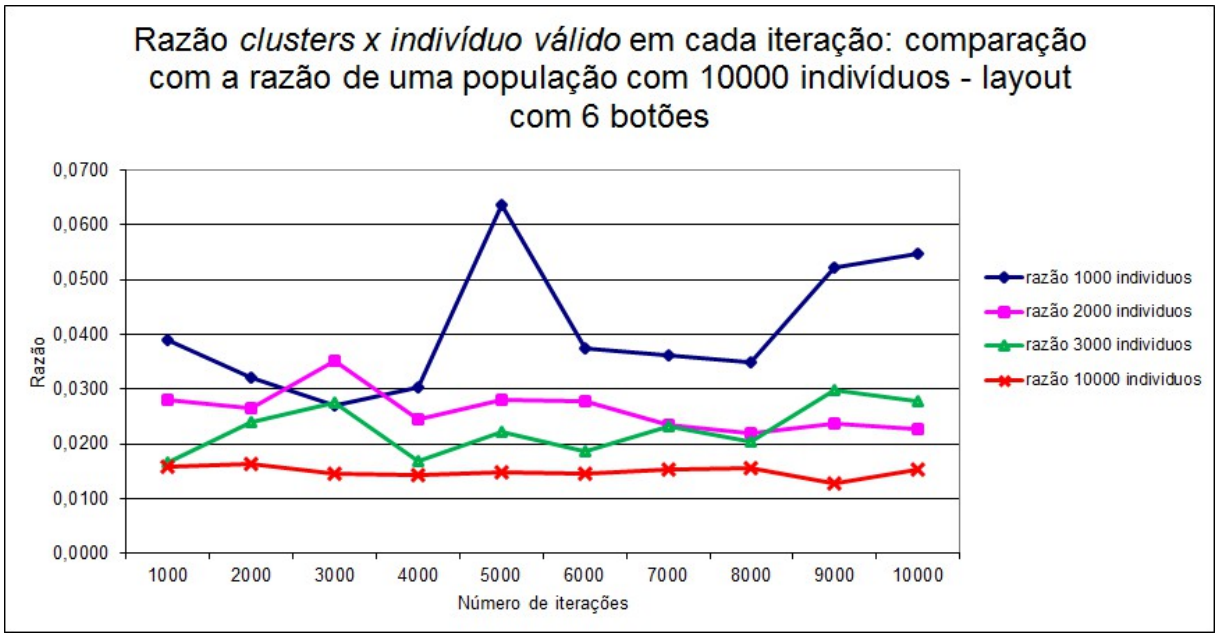

Figura 5.11: Razões de *clusters* por indivíduos válidos contendo os dados de populações com 10.000 indivíduos que não foram avaliadas por uma função objetivo - *layouts* com seis botões. **Fonte**: O autor (2019).

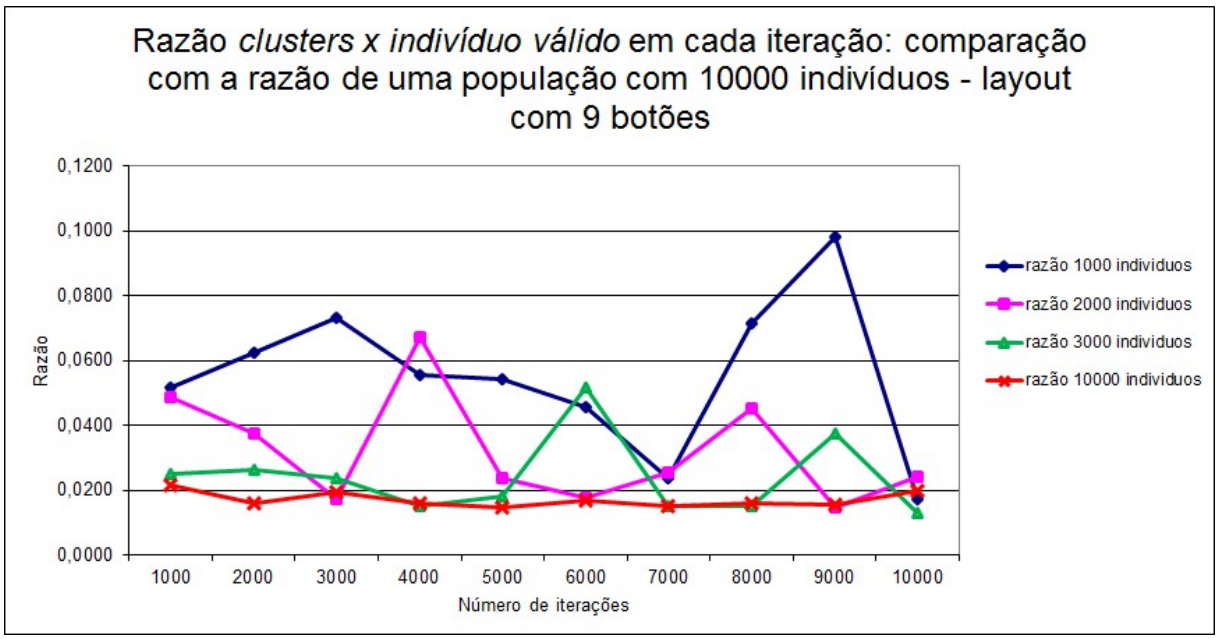

Figura 5.12: Razões de *clusters* por indivíduos válidos contendo os dados de populações com 10.000 indivíduos que não foram avaliadas por uma função objetivo - *layouts* com nove botões. **Fonte**: O autor (2019).

Como pode ser visto, a razão da população que não foi avaliada pela função objetivo foi, na maioria das vezes, menor que das outras populações. Isso mostra que a função objetivo usada em nosso modelo é razoável, não limitando a criatividade do modelo. Apesar de criarmos essa função objetivo, nada impede que outras funções de aptidão sejam construídas e usadas para avaliar indivíduos de uma população. Como dito anteriormente, procuramos restringir o mínimo possível a criatividade do modelo para que uma grande quantidade de resultados fossem criados sem que todos fossem idênticos entre si.

5.2 Agrupamentos dos *layouts*

Além de apenas medir a variabilidade das configurações geradas pelo modelo, utilizamos a quantidade de *clusters* para organizar os *layouts* classificados como válidos pelo SVM. Desta forma, os *layouts* semelhantes entre si ficam todos em um mesmo diretório, facilitando a visualização por parte do *designer*.

As Figuras abaixo apresentam exemplos de *layouts* agrupados em determinados *clusters* (figuras menores), seguidos de um '*layout* ideal' proposto por um *designer* após analisar esses resultados.

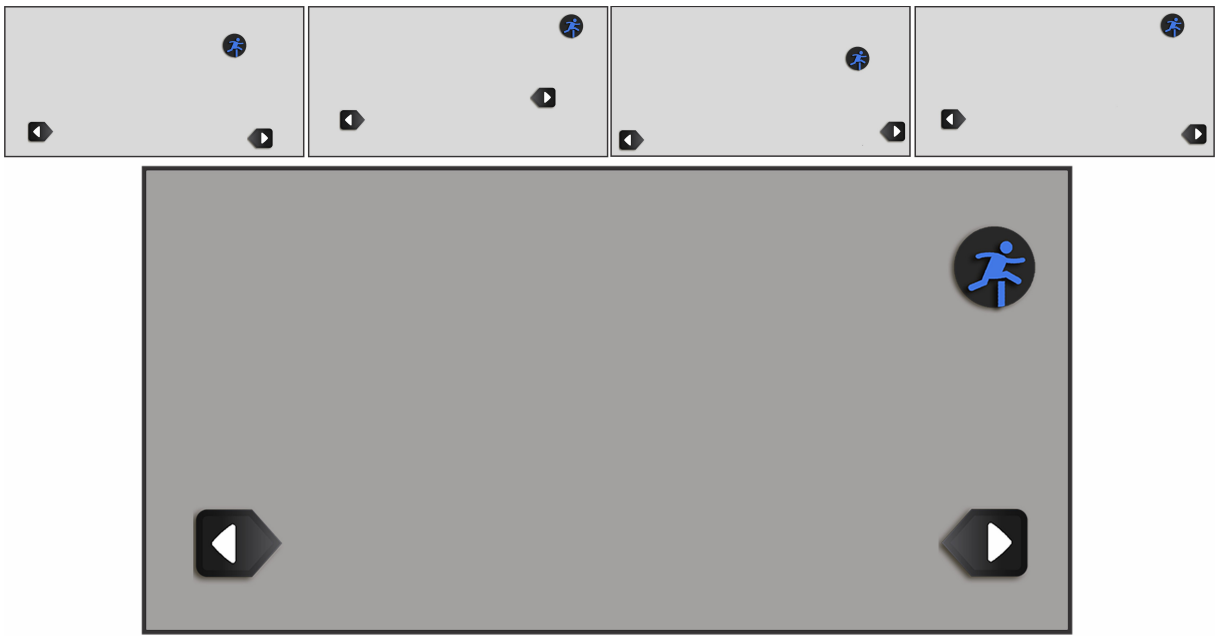

Figura 5.13: Quatro *layouts* gerados pelo modelo contendo três botões agrupados em um mesmo *cluster* (tamanho da população: 1000; iteração: 3000; número do *cluster*: 8), seguido de um *layout* desenvolvido por um *designer* após analisá-los. **Fonte**: O autor (2019).

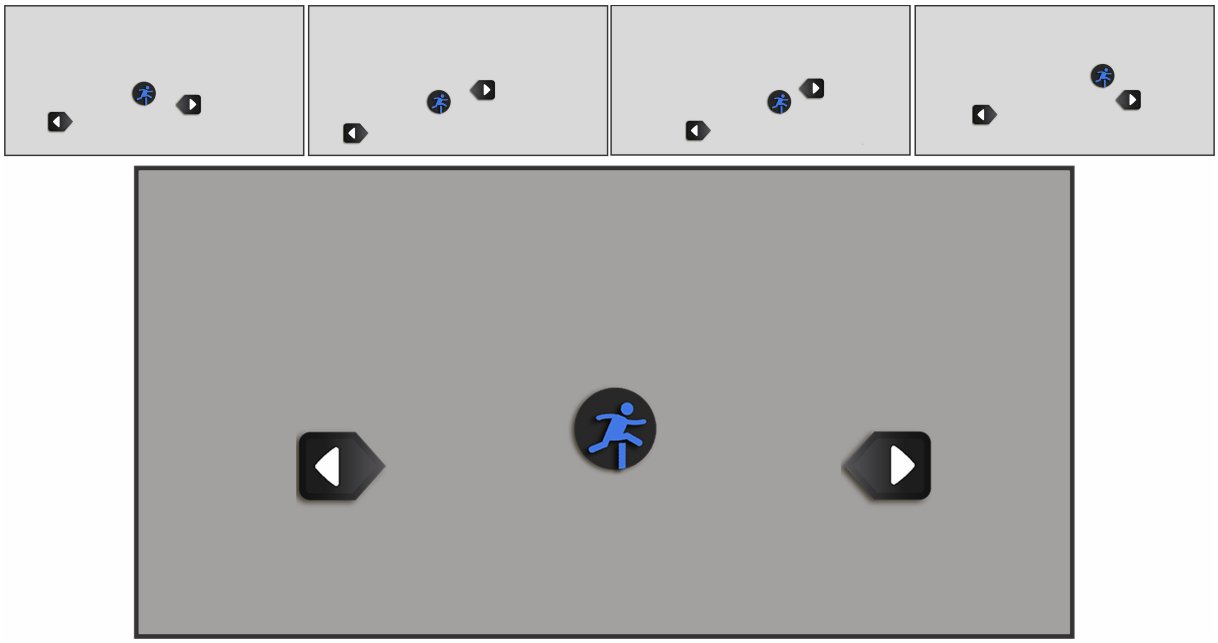

Figura 5.14: Quatro *layouts* gerados pelo modelo contendo três botões agrupados em um mesmo *cluster* (tamanho da população: 2000; iteração: 6000; número do *cluster*: 1), seguido de um *layout* desenvolvido por um *designer* após analisá-los. **Fonte**: O autor (2019).

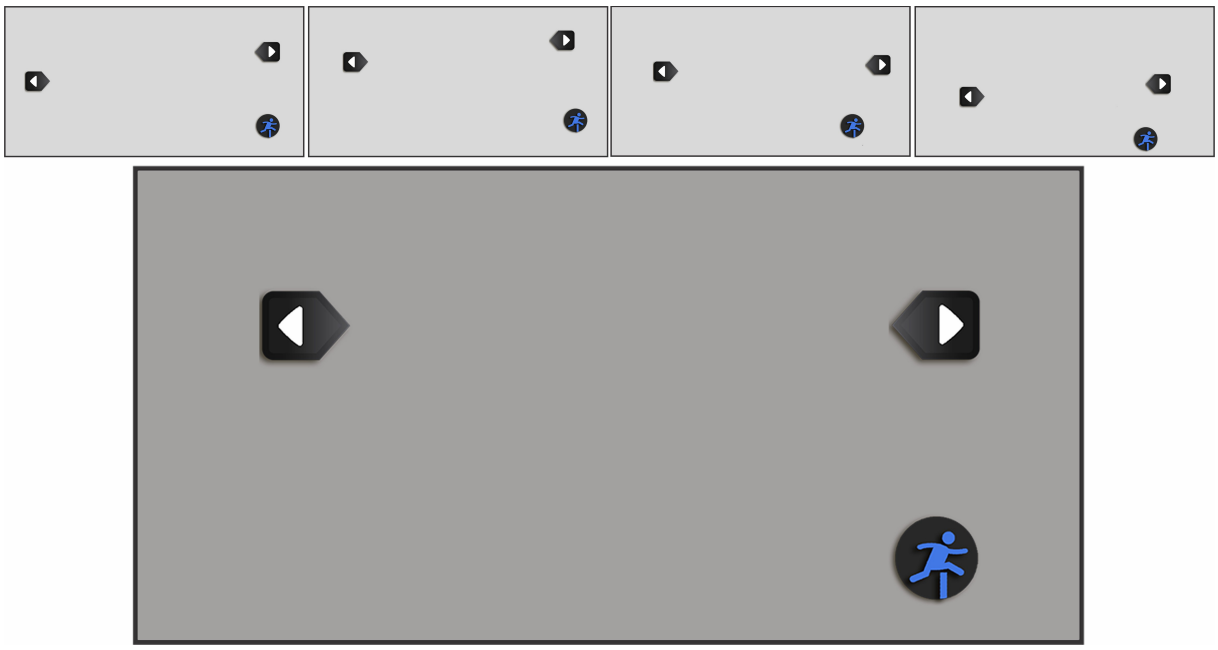

Figura 5.15: Quatro *layouts* gerados pelo modelo contendo três botões agrupados em um mesmo *cluster* (tamanho da população: 3000; iteração: 4000; número do *cluster*: 4), seguido de um *layout* desenvolvido por um *designer* após analisá-los. **Fonte**: O autor (2019).

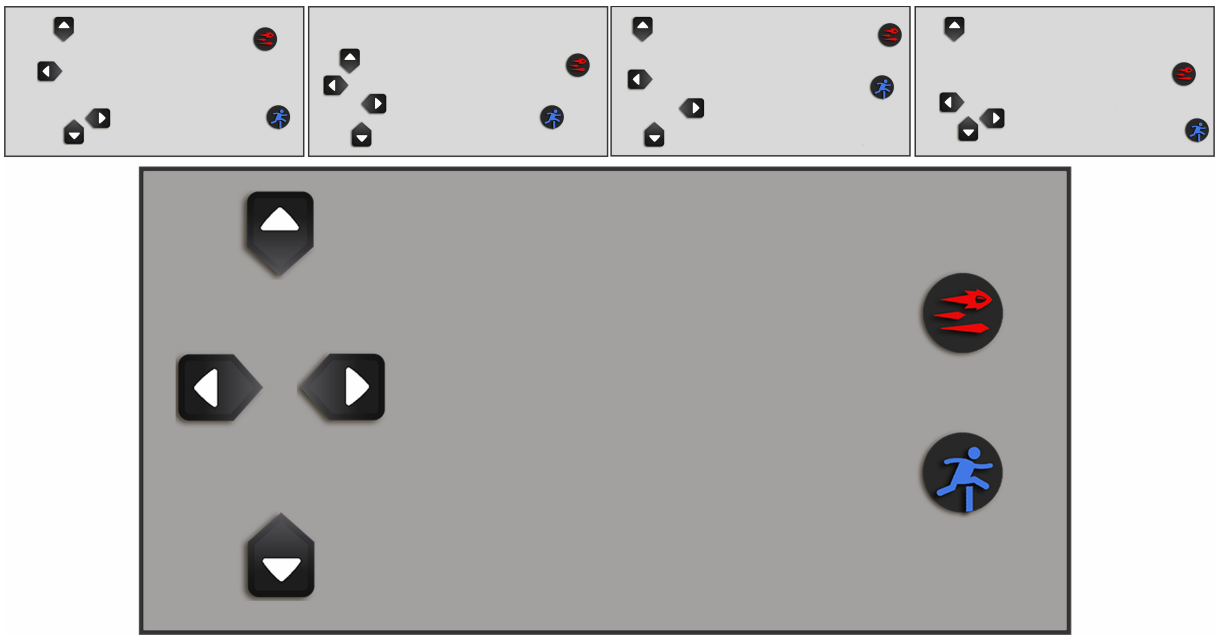

Figura 5.16: Quatro *layouts* gerados pelo modelo contendo seis botões agrupados em um mesmo *cluster* (tamanho da população: 1000; iteração: 4000; número do *cluster*: 1), seguido de um *layout* desenvolvido por um *designer* após analisá-los. **Fonte**: O autor (2019).

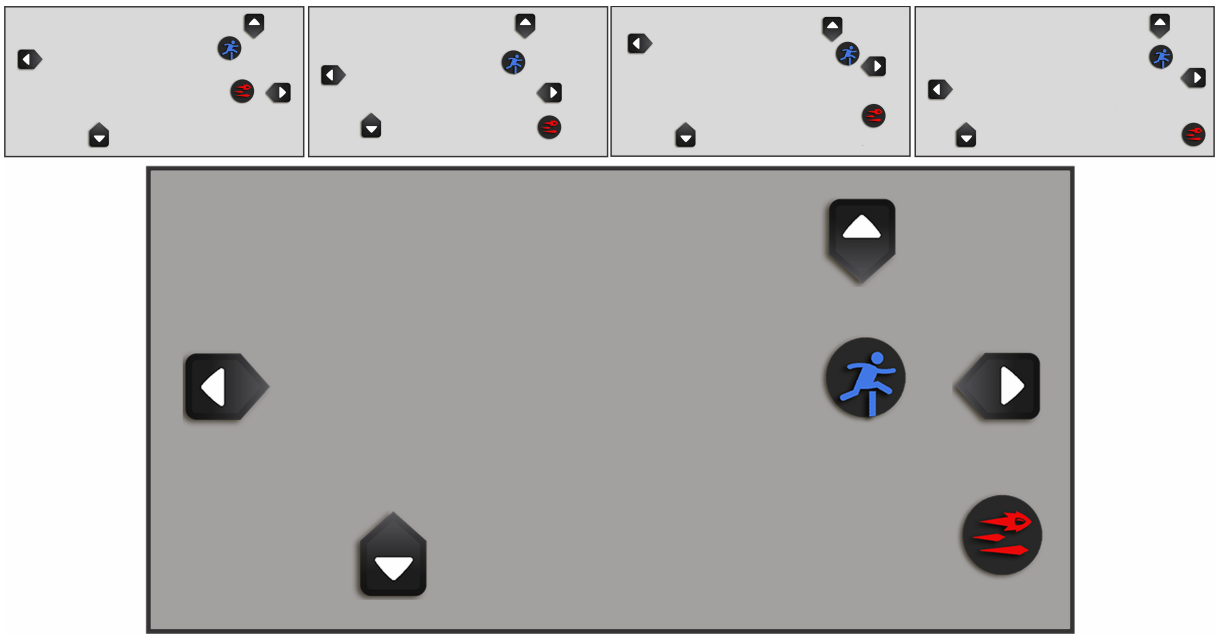

Figura 5.17: Quatro *layouts* gerados pelo modelo contendo seis botões agrupados em um mesmo *cluster* (tamanho da população: 2000; iteração: 2000; número do *cluster*: 6), seguido de um *layout* desenvolvido por um *designer* após analisá-los. **Fonte**: O autor (2019).

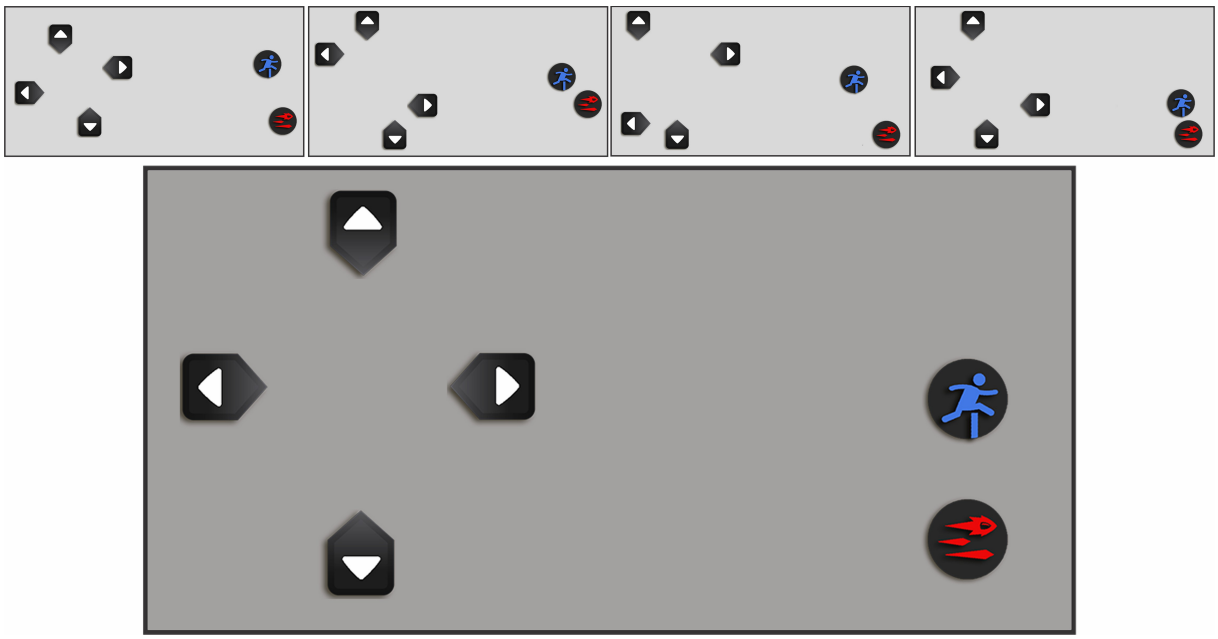

Figura 5.18: Quatro *layouts* gerados pelo modelo contendo seis botões agrupados em um mesmo *cluster* (tamanho da população: 3000; iteração: 5000; número do *cluster*: 9), seguido de um *layout* desenvolvido por um *designer* após analisá-los. **Fonte**: O autor (2019).

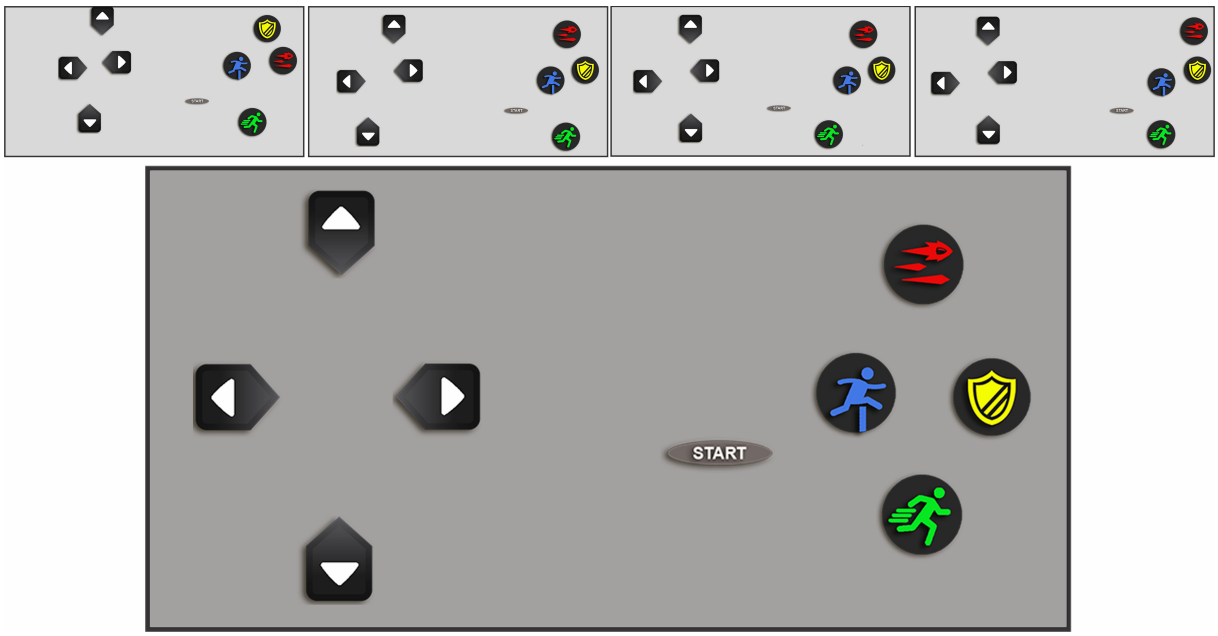

Figura 5.19: Quatro *layouts* gerados pelo modelo contendo nove botões agrupados em um mesmo *cluster* (tamanho da população: 1000; iteração: 3000; número do *cluster*: 3), seguido de um *layout* desenvolvido por um *designer* após analisá-los. **Fonte**: O autor (2019).

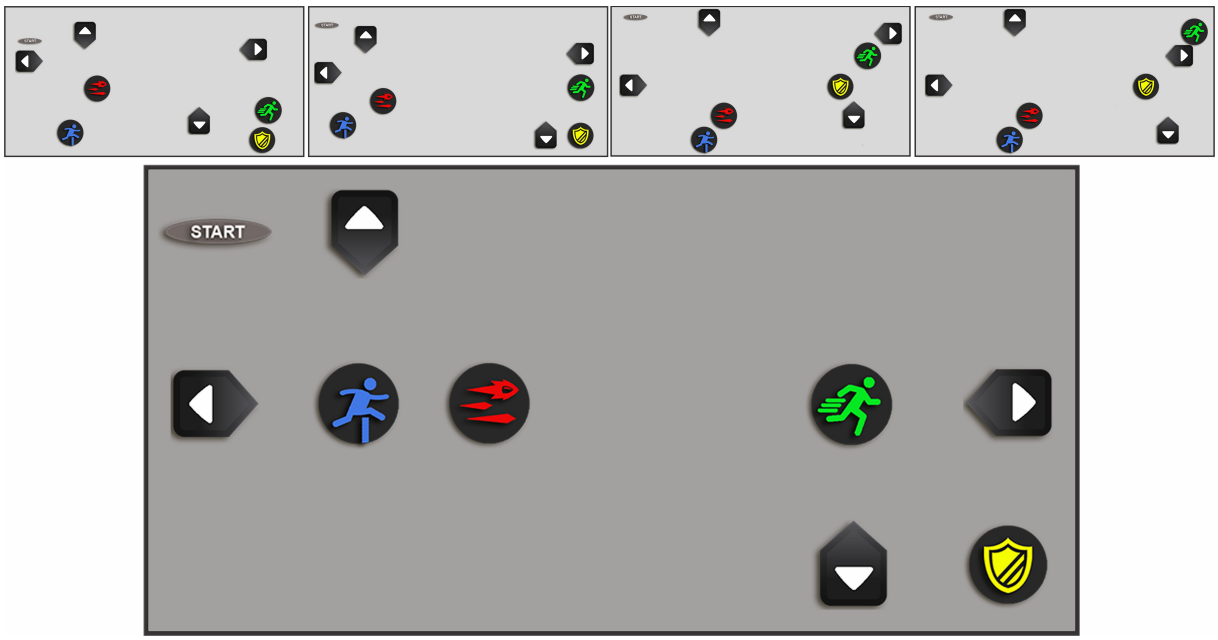

Figura 5.20: Quatro *layouts* gerados pelo modelo contendo nove botões agrupados em um mesmo *cluster* (tamanho da população: 2000; iteração: 3000; número do *cluster*: 3), seguido de um *layout* desenvolvido por um *designer* após analisá-los. **Fonte**: O autor (2019).

Figura 5.21: Quatro *layouts* gerados pelo modelo contendo nove botões agrupados em um mesmo *cluster* (tamanho da população: 3000; iteração: 10.000; número do *cluster*: 8), seguido de um *layout* desenvolvido por um *designer* após analisá-los. **Fonte**: O autor (2019).

Ao final dos testes, buscamos opiniões de outros *designers* sobre os '*layouts* ideais', construídos após a análise das saídas do modelo. Cinco *game designers* (GD1, GD2, GD3, GD4 e GD5), quatro do sexo masculino e um do sexo feminino, com idade entre 18 e 30 anos tiveram contato com os 9 *layouts* candidatos. Os *designers* possuem entre um e cinco anos de experiência em *design* de jogos.

Os *layouts* foram apresentados a eles em um *smartphone* ASUS Zenfone 3 Zoom com tela de 5.5 polegadas. Foi questionada a opinião deles sobre aqueles *layouts*, pedindo que levassem em consideração a disposição e agrupamento dos botões na área do controle virtual. Alguns comentários realizados durante as análises dos *game designers* encontram-se abaixo:

GD1: *"Esse (Figura 5.19) é o 'padrão', na minha opinião. Meu único problema é o espaçamento na vertical. Apertar os botões de cima e de baixo parece meio desconfortável. Pela mesma razão, não gostei desse (Figura 5.21)."*

"Entre os de três botões, esse (Figura 5.13) é o que me agrada mais. Uma única mudança que eu faria é jogar o botão de pulo mais pro inferior da tela. É mais ergonômico jogar quando os botões ficam mais próximos dos dedões, e quando a gente segura o celular 'de lado' eles tendem a ficar mais para baixo."

"Esse (Figura 5.20) eu achei interessantíssimo! Mesclar os locais dos botões de ação com os de movimento é criativo e eu acho que daria muito certo pra um jogo tipo o League of Legends, por exemplo. Os botões no canto inferior seriam usados com mais frequência e os outros seriam usados como um 'especial'."

GD2: *"Fica mais fácil alcançar todos os direcionais desta forma (Figura 5.20) do que todos em um canto só."*

"Três botões com o pulo no meio (Figura 5.14) não é tão confortável porque é mais difícil de alcançar ele, preciso fazer muito esforço pra alcançar o botão do meio."

"Para usuários iniciantes, as setas agrupadas (Figuras 5.16, 5.18, 5.19 e 5.21) podem ser mais interessantes, mas acho mais legais os que separam as setas (Figuras 5.17 e 5.20)."

"Esse (Figura 5.18) não é muito confortável para mim e provavelmente não será para outras pessoas que tenham dedos pequenos."

GD3: *"Gostei dos controles, porém tenho um pouco de desconforto ao tentar alcançar o botão 'para cima' nesses controles (Figuras 5.16, 5.18, 5.19 e 5.21). Acredito que seja melhor colocar eles um pouco mais para baixo."*

GD4: *"Gostei desse (Figura 5.14). Já consigo imaginar um jogo para esse controle."*

"O botão de pular próximo do botão 'para frente' (Figura 5.17) não me parece bom. Acho que não conseguiria apertar os dois botões ao mesmo tempo."

"Esse é clássico, gosto dele (Figura 5.16). Eu realmente prefiro esses controles 'padrão'."

"Esse (Figura 5.19) também me parece bom, apesar de que eu giraria todos esses botões de ação no sentido anti-horário. O botão de pulo mais abaixo, correr na direita, atirar na esquerda e defender em cima me parece mais certo."

GD5: *"Acho que os que funcionam melhor são os em que os botões estão nos cantos inferiores. Nossos dedões são curtos pra alcançar os botões que estão lá em cima."*

"Eu tive um pequeno problema com os que mesclam a setinha e as ações (Figuras 5.17, 5.20). Se eu precisar pular e apertar o 'pra frente' ao mesmo tempo fica meio complicado se eles estiverem do mesmo lado."

"Acho que o start deveria estar no meio do controle, acho mais intuitivo."

Em suma, os *layouts* que separavam os direcionais em dois grupos dividiram opiniões.

A posição do botão de ação em configurações com três botões foi relatada como muito distante, dificultando a sua combinação com outras ações. Também foram feitos comentários sobre o alcance do dedo do usuário e altura dos botões, além de indicações de aproximações entre botões de ação.

Capítulo 6

Conclusão

Neste trabalho, abordamos o problema de auxiliar um *designer* no processo de criação de diferentes *layouts* de *gamepad* usando técnicas de *design* generativo. A principal contribuição deste trabalho é um *pipeline* de *design* generativo para *design* de *gamepad* baseado em técnicas de algoritmos evolutivos e aprendizado de máquina. O algoritmo evolutivo é responsável por criar um grande número de configurações de *gamepads* baseado nas entradas fornecidas pelo usuário: número máximo de configurações a serem geradas, quantidade de iterações que deverão ser executadas para gerar essas configurações, quantidade e tipo dos botões que deverão estrar presentes em sua organização. Após a sua execução, o modelo fornece como saída as configurações solicitadas pelo usuário. Como geralmente este é um número grande, torna-se inviável para o *designer* analisar todos. Como certamente alguns desses serão descartados por ele, visto que os controles de videogame respeitam algumas regras organizacionais, aplicamos a técnica de aprendizado de máquina *Support Vector Machine* para classificar indivíduos como 'válidos' ou 'inválidos'. Além disso, propomos uma maneira de medir a variabilidade das soluções candidatas usando uma versão estendida do algoritmo K-means, que pode determinar automaticamente o número de *clusters* no grupo de indivíduos.

Avaliamos os resultados do método proposto para executar um conjunto de testes com diferentes configurações de controles (quantidade de botões e seus tipos), tamanhos de populações (a quantidade de soluções produzidas) e o número de iterações. Os experimentos mostraram que, em média, o número de *layouts* classificados como válidos pelo classificador que utiliza de aprendizado de máquina aumenta com o tamanho da população, porém esta nem sempre é uma verdade quando observamos o comportamento do número de *clusters*.

Ao calcular a razão de *clusters* por indivíduos válidos em cada população, percebe-

mos que, para cada configuração de controle, as razões correspondentes às gerações de populações com 1000 indivíduos são maiores que as de 2000 e 3000 indivíduos, mostrando que uma população com essa quantidade de elementos é suficiente para representar uma grande variabilidade de resultados.

Baseado na separação dos *layouts* em *clusters*, foram usadas saídas do modelo como inspiração para a criação de algumas configurações de controles virtuais e apresentados a alguns *designers* com experiência em construção de jogos. Em resumo, grande parte dos projetistas ficaram surpresos com as configurações construídas e houveram comentários sobre o alcance do dedo do usuário e altura dos botões.

6.1 Limitações e trabalhos futuros

Algumas alterações nos operadores do algoritmo genético são propostas como trabalhos futuros: a primeira consiste na implementação do operador de crossover, fazendo com que *layouts* diferentes troquem comportamentos entre si. Uma outra alteração proposta consiste na aplicação das mutações apenas em indivíduos com um baixo valor *fitness*, fazendo com que isso possa alterar seu valor de aptidão.

Outro trabalho futuro consiste na análise das configurações rotuladas como inválidas pelo classificador sendo que o *designer* as rotulou como válidas (e vice versa) na etapa de treinamento da base de classificação, visto nas matrizes de confusão. É possível que existam padrões entre os *layouts* que o classificador erroneamente categoriza como pertencentes a outra classe e que podem ser corrigidos.

Sobre a avaliação de configurações de controles entre válido e inválido, é interessante que existam gradações de avaliação para diferentes análises. Uma melhoria futura consiste na adição de uma categoria 'intermediário', para que controles possam ser melhor categorizados e organizados.

Quanto à análise de variabilidade de *layouts*, a técnica utilizada pode considerar alguns *layouts* de controles em um mesmo cluster, como os apresentados na Figura 6.1. O algoritmo K-means não interpreta a semântica presente nos controles de videogame, uma vez que ele analisa apenas as posições dos botões dentro do espaço de *layout* do controle. Neste caso, os *layouts* colocados no mesmo *cluster* são muito semelhantes quando visto o posicionamento de botões dentro da área do *gamepad*, mas no contexto da experiência do jogador eles são muito diferentes, o que pode gerar um alto esforço cognitivo por parte do usuário ao tentar executar ações dentro de um jogo. Como trabalho futuro, pretendemos

estudar outras técnicas de avaliação e categorização, como abordagens de correspondência de grafos [28] ou técnicas de visão computacional, como subtração de imagens [37] para melhorar a forma como medimos a variabilidade.

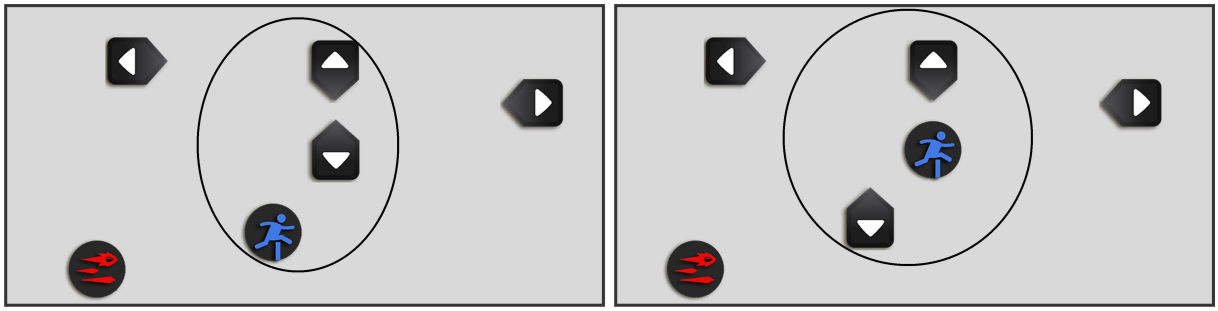

Figura 6.1: Indivíduos inseridos no mesmo *cluster* quando seus botões são colocados em posições semelhantes. No entanto, esses *layouts* não são semelhantes porque os botões alterados têm semânticas diferentes. **Fonte**: O autor (2019).

Uma modificação na forma como o modelo generativo funciona também é proposta: tornar a ferramenta interativa e incremental, de forma que o *designer* possa fazer parte do processo de decisão do modelo, onde soluções são geradas pelo *software*, escolhidas pelo *designer* e, então, novas soluções sejam geradas baseadas nas opiniões fornecidas pelo mesmo. Essa solução irá explorar um novo conjunto de parâmetros de entrada, como tamanho, forma e cor dos botões na área do controle.

Além disso, considerando o grande número de soluções geradas pelo modelo proposto, uma ferramenta de visualização pode ser proposta para ajudar e orientar os *designers* de jogos a navegar em todo o espaço de *design* e escolher os *layouts* que melhor atendem as suas necessidades.

Por fim, não foi feita nenhuma avaliação sobre o modelo de *design* generativo em si, apresentando-o para *designers* para que pudessem usá-lo. Um trabalho futuro visa introduzir o modelo no âmbito de trabalho de *designers* e analisar se a ferramenta foi útil para eles.

6.2 Publicação

Um trabalho foi desenvolvido buscando suportar o processo de *design* de controles de jogos por meio de um conjunto de técnicas de *Design Thinking* e resultou em uma publicação no *ICEC 2018* (*International Conference on Entertainment Computing*).

Referências

[1] Adrenaline. Psp e nintendo 3ds. *Disponível em: (https://adrenaline.uol.com.br/2011/04/08/7661/psp-vende-mais-que-o-3ds-no-japao/). Acessado em 21 de Março de 2019; 19h28min.* (2019).

[2] Alves, G. F.; Souza, E. V.; Trevisan, D. G.; Montenegro, A. A.; de Castro Salgado, L. C.; Clua, E. W. G. Applying design thinking for prototyping a game controller. In *International Conference on Entertainment Computing* (2018), Springer, pp. 16–27.

[3] Atari. Atari 2600. *Disponível em: (https://www.voxodyssey.com/atari-2600/). Acessado em 21 de Março de 2019; 19h24min.* (2019).

[4] Baldauf, M.; Adegeye, F.; Alt, F.; Harms, J. Your browser is the controller: advanced web-based smartphone remote controls for public screens. In *Proceedings of the 5th ACM International Symposium on Pervasive Displays* (2016), ACM, pp. 175–181.

[5] Baldauf, M.; Fröhlich, P.; Adegeye, F.; Suette, S. Investigating on-screen gamepad designs for smartphone-controlled video games. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 12*, 1s (2015), 22.

[6] Bastug, E.; Bennis, M.; Médard, M.; Debbah, M. Toward interconnected virtual reality: Opportunities, challenges, and enablers. *IEEE Communications Magazine 55*, 6 (2017), 110–117.

[7] Bentley, P. *Evolutionary design by computers*. Morgan Kaufmann, 1999.

[8] Bjørlykhaug, E.; Egeland, O. Mechanical design optimization of a 6dof serial manipulator using genetic algorithm. *IEEE Access 6* (2018), 59087–59095.

[9] Center, L. Innovative lives: the pioneers of spacewar! *Available in: (http://invention.si.edu/about/events/innovative-lives-pioneers-spacewar). Accessed on January 31, 2019; 10:35 pm.* (2018).

[10] Consoles, R. G. Wonder wizard. *Disponível em: (https://www.retrogamingconsoles.com/consoles/wonder-wizard/). Acessado em 07 de Fevereiro, 2019; 15h50min.* (2019).

[11] D'Argenio, A. M. Statistically, video games are now the most popular and profitable form of entertainment. *Available in: (https://www.gamecrate.com/statistically-video-games-are-now-most-popular-and-profitable-form-entertainment/20087). Accessed on December 17, 2018; 2:05 pm.* (2018).

[12] DesignBoom. Grayhaus brick joystick. *Disponível em: (https://www.designboom.com/readers/grayhaus-brick/). Acessado em 27 de Abril de 2018 às 13h17min.* (2018).

[13] Fraley, C.; Raftery, A. E. How many clusters? which clustering method? answers via model-based cluster analysis. *The computer journal 41*, 8 (1998), 578–588.

[14] Frazer, J. Creative design and the generative evolutionary paradigm. In *Creative evolutionary systems*. Elsevier, 2002, pp. 253–274.

[15] Goldberg, D. E.; Holland, J. H. Genetic algorithms and machine learning. *Machine learning 3*, 2 (1988), 95–99.

[16] Google. Google stadia. *Disponível em: (https://www.stadia.dev/). Acessado em 20 de Março de 2019; 14h16.* (2019).

[17] Greenbot. 20 classic games you can play on your android phone. *Disponível em (https://www.greenbot.com/article/2987637/20-classic-games-you-can-play-on-your-android-phone.html). Acessado em 16 de Março de 2019 às 14h44min.* (2019).

[18] Hope, C. Game history. *Disponível em: (https://www.computerhope.com/history/game.htm). Acessado em 07 de Fevereiro, 2019; 15h20min.* (2019).

[19] Hope, C. When was the first computer game released? *Disponível em: (https://www.computerhope.com/issues/ch001007.htm). Acessado em 07 de Fevereiro, 2019; 15h00min.* (2019).

[20] Ipega. Pg-9021. *Disponível em (http://www.ipega.hk/). Acessado em 27 de Abril de 2018 às 13h17min.* (2018).

[21] Jain, A. K. Data clustering: 50 years beyond k-means. *Pattern recognition letters 31*, 8 (2010), 651–666.

[22] Krish, S. A practical generative design method. *Computer-Aided Design 43*, 1 (2011), 88–100.

[23] Lu, W. Evolution of video game controllers: How simple switches lead to the development of the joystick and the directional pad by.

[24] Merdenyan, B.; Petrie, H. User reviews of gamepad controllers: A source of user requirements and user experience. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play* (2015), ACM, pp. 643–648.

[25] Moga. Moga hero power. *Disponível em (http://www.mogaanywhere.com/controllers/ heropower). Acessado em 27 de Abril de 2018 às 13h17min.* (2018).

[26] Nintendo. Nintendo game and watch. *Disponível em: (https://auction.catawiki.com/kavels/7938301-nintendo-game-watch-super-mario-bros). Acessado em 21 de Março de 2019; 19h26min.* (2019).

[27] of American History, T. N. M. The brown box, 1967–68. *Disponível em: (http://americanhistory.si.edu/collections/search/object/nmah_1301997). Acessado em 08 de Fevereiro, 2019; 16h59min.* (1967).

[28] Oliveira, A.; Tessarolli, G.; Ghiotto, G.; Pinto, B.; Campello, F.; Marques, M.; Oliveira, C.; Rodrigues, I.; Kalinowski, M.; Souza, U., et al. An efficient similarity-based approach for comparing xml documents. *Information Systems 78* (2018), 40–57.

[29] One, M. X. Xbox design lab. *Disponível em: (https://controllers.xbox.com/en-us). Acessado em 4 de Janeiro de 2019; 10h20.* (2019).

[30] Pelegrino, P. M. *Um Controle Virtual Dinâmico para Jogos Adaptado ao Gameplay e ao Usuário.* Tese de Doutorado, Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brasil, Setembro 2016.

[31] PlayStation, S. Dualshock®4 wireless controller. *Disponível em: (https://www.playstation.com/en-us/explore/accessories/gaming-controllers/dualshock-4/). Acessado em 4 de Janeiro de 2019, 10h16.* (2019).

[32] PlayStation®. The last of us™. *Available in: (https://www.playstation.com/en-us/games/the-last-of-us-remastered-ps4/). Accessed on December 30, 2018; 5:15 pm.* (2018).

[33] RealGear. Video game consoles evolution (1967-2017) – 50 years of glory. *Disponível em: (https://www.realgear.net/news/hardware/video-game-consoles-evolution-1967-2017-50-years-glory/). Acessado em 08 de Fevereiro, 2019; 17h44min.* (2017).

[34] Reas, C.; Fry, B. Processing. *Available in: (https://processing.org/). Accessed on December 21, 2018; 10:42 am.* (2018).

[35] Reed, K. Machine learning applications in generative design.

[36] Rogers, S. *Level Up! The guide to great video game design.* Wiley, 2014.

[37] Russ, J. C. *The image processing handbook.* CRC press, 2016.

[38] Switch™, N. Nintendo switch™ joy-con™. *Disponível em: (https://www.nintendo.com/switch/choose-your-joy-con-color/). Acessado em 4 de Janeiro de 2019; 10h18.* (2019).

[39] Torok, L.; Pelegrino, M.; Trevisan, D.; Montenegro, A.; Clua, E. Designing game controllers in a mobile device. In *Design, User Experience, and Usability: Designing Pleasurable Experiences DUXU* (Cham, 2017), A. Marcus and W. Wang, Eds., Springer International Publishing, pp. 456–468.

[40] Torok, L.; Pelegrino, M.; Trevisan, D.; Montenegro, A.; Clua, E. Smart controller: Introducing a dynamic interface adapted to the gameplay. *Entertainment Computing 27* (2018), 32–46.

[41] Troiano, L.; Birtolo, C. Genetic algorithms supporting generative design of user interfaces: Examples. *Information Sciences 259* (2014), 433–451.

[42] Twitgoo. 10 best gba (game boy advance) emulators for android. *Disponível em (https://twitgoo.com/best-gba-emulators-android/). Acessado em 16 de Março de 2019 às 14h43min.* (2019).

[43] Vapnik, V. *The nature of statistical learning theory.* Springer science & business media, 2013.

[44] Wagstaff, K.; Cardie, C.; Rogers, S.; Schrödl, S., et al. Constrained k-means clustering with background knowledge. In *Icml* (2001), vol. 1, pp. 577–584.

[45] Zhan, Y.; Shen, D. Design efficient support vector machine for fast classification. *Pattern Recognition 38*, 1 (2005), 157–161.

APÊNDICE A – TERMO DE CONSENTIMENTO E LIVRE ESCLARECIMENTO

UNIVERSIDADE FEDERAL DE FLUMINENSE
INSTITUTO DE COMPUTAÇÃO

TERMO DE CONSENTIMENTO E LIVRE ESCLARECIMENTO

Avaliação de *layouts* de controles virtuais de videogames

Prezado(a),

Você está sendo convidado(a) a participar de um estudo que tem por objetivo coletar opiniões de *game designers* sobre a organização de diferentes *layouts* de controles virtuais contendo diferentes quantidades de botões. Esses *layouts* foram criados pensando em controles virtuais que são executados em dispositivos *mobile* com a tecnologia *touchscreen*, como *smartphones* e *tablets*.

Este procedimento não apresenta riscos uma vez que nenhum tipo de intervenção será necessária. A qualquer momento o usuário poderá expor sua opinião sobre o *layout* que está analisando. Com isso, todas as opiniões servirão apenas para que possamos compreender o ponto de vista de um *designer* sobre a organização de controle. O usuário poderá, a qualquer momento, desistir do teste.

A equipe envolvida no estudo é composta por um estudante de mestrado (Gabriel Alves) e os professores responsáveis (Dra. Daniela G. Trevisan e Dr. Anselmo A. Montenegro).

Solicitamos sua autorização para o uso dos dados obtidos durante o experimento e pela transcrição do áudio, com o objetivo de produzir artigos técnicos e científicos, sempre garantindo seu anonimato.

Agradecemos sua participação e colaboração.

Responsável para contato:
Gabriel Ferreira Alves (gabrielferreiraalves@id.uff.br)
Telefone: (21) 96514-3586
Rua Doutor Paulo Alves, 110, Niterói, RJ, Brasil

TERMO DE CONSENTIMENTO

Declaro que fui informado sobre todos os procedimentos da pesquisa, que recebi de forma clara e objetiva todas as explicações pertinentes ao projeto e da garantia de anonimato dos meus dados. Eu compreendo que neste estudo, as observações dos experimentos e procedimentos de tratamento serão feitas comigo.

Declaro que fui informado que posso me retirar do estudo a qualquer momento.

Nome por extenso _____.

Assinatura _____ Niterói, ____/____/____.

CONSENTIMENTO PARA GRAVAÇÕES DE ÁUDIO

Eu, _____, permito que o grupo de pesquisadores relacionados abaixo obtenha gravação da minha voz para fins de pesquisa cientifica e educacional.

Eu concordo que o material e informações obtidas relacionadas à minha pessoa possam ser publicados em aulas, congressos, palestras ou periódicos científicos desde que seja sempre garantido meu anonimato de forma a não permitir a minha identificação.

As gravações ficarão sob a propriedade do grupo de pesquisadores pertinentes ao estudo e sob a guarda dos mesmos.

Nome ou responsável legal: _____

Assinatura: _____

Equipe de pesquisadores: Dra. Daniela Gorski Trevisan (Professora), Dr. Anselmo Antunes Montenegro (Professor) e Gabriel Ferreira Alves (Aluno de Mestrado).

Data:_____

APÊNDICE B – QUESTIONÁRIO DE PERFIL DO PARTICIPANTE

QUESTIONÁRIO DE PERFIL DO PARTICIPANTE

*Obrigatório

Digite seu nome *

Sua resposta

Qual a sua faixa etária? *

- ◯ Entre 18 e 30 anos
- ◯ Entre 31 e 40 anos
- ◯ Entre 41 e 50 anos
- ◯ Entre 51 e 60 anos
- ◯ Acima de 60 anos

Qual o seu sexo? *

- ◯ Masculino
- ◯ Feminino
- ◯ Prefiro não dizer
- ◯ Outro:

Há quanto tempo você trabalha com jogos? *

- ◯ Há menos de um ano
- ◯ Entre um e dois anos
- ◯ Entre dois e três anos
- ◯ Entre três e quatro anos
- ◯ Entre quatro e cinco anos
- ◯ Entre cinco e dez anos
- ◯ Há mais de dez anos

ENVIAR

Nunca envie senhas pelo Formulários Google.