Code In Hand – Jogo de Baralho para Ensino de Programação a Estudantes de Computação

Leonardo Silveira Chagas, Diana F. Adamatti
Centro de Ciências Computacionais
Universidade Federal do Rio Grande
Rio Grande, Brasil
leoschagas@hotmail.com, dianaada@gmail.com

***Resumo*—Este trabalho apresenta um jogo sério que tem o intuito de ensinar conceitos de programação para alunos de cursos superiores na área das ciências computacionais. No contexto deste trabalho, é explicado o uso de jogos na educação, realizando uma revisão sistemática da literatura, bem como o processo de desenvolvimento do jogo, que foi devidamente documentado e seus resultados demonstrados. Para a coleta de dados, o jogo desenvolvido será testado com uma turma no segundo semestre de engenharia de computação da Universidade Federal do Rio Grande, onde os participantes responderão um questionário para dar o seu feedback sobre o jogo. Esta pesquisa também tem por objetivo, servir de contribuição para a área de desenvolvimento de jogos sérios.**

***Index Terms*—Jogos Sérios, Ensino de Programação, Jogos na Educação**

I. Introdução

As tecnologias estão avançando de uma maneira acelerada, sendo uma parte integral do nosso dia a dia, sendo encontradas no nosso lazer, como nos comunicamos, no nosso trabalho, etc. Entretanto, os métodos de ensino utilizados nos cursos superiores não estão evoluindo da mesma maneira, ainda utilizando o mesmo método tradicional de ensino, o que, para alunos de cursos das áreas das ciências computacionais, que possuem interesse na informática, pode causar a perda de interesse no curso.

Estudos realizados por [1] e [2] indicam que a didática dos professores e a falta de conhecimento base estão entre as razões que causam a maior insatisfação entre os alunos de ciências de computação da UFRGS (Universidade Federal do Rio Grande do Sul).

Essas pesquisas oferecem algumas alternativas para melhorar o curso, como o uso de disciplinas complementares para tentar cobrir a necessidade de conhecimento prévio dos alunos, ou uma melhoria na formação pedagógica dos professores para melhorar a didática dos mesmos. Uma das maneiras de enfrentar esses problemas, e que tem se popularizado nos últimos anos, é a utilização de jogos sérios para auxiliar no entendimento de conteúdos.

Jogos sérios são uma categoria de jogos onde o objetivo não é apenas divertir, mas sim, ensinar. Os jogos de computador fazem parte da realidade da maioria dos jovens hoje em dia. Dessa forma, unir o entretenimento promovido pelos jogos com metodologias de ensino, pode se tornar uma excelente ferramenta para melhorar o aprendizado nas instituições de ensino superior, não apenas em ciências computacionais, mas também em outras disciplinas, como pode ser observado pelos trabalhos de [3] e [4] que obtiveram resultados favoráveis com relação ao uso de jogos sérios na educação.

Por outro lado, o jogo desenvolvido no trabalho de [5] obteve resultados inconclusivos, com relação a melhoria dos participantes em entender programas de computador, por isso a metodologia para o desenvolvimento do jogo e para a realização de testes, é de extrema importância. Diversos jogos sérios têm sido desenvolvidos, voltados para a área de computação, como pode ser visto no trabalho de [6], que apresenta uma revisão de mais de 100 jogos sérios físicos e digitais, com o intuito de auxiliar na educação de diversas áreas de ciências computacionais.

Essa pesquisa também indica que, apesar de existirem diversos jogos sérios voltados para a área de computação, poucos deles possuem uma descrição sistemática do desenvolvimento do jogo, público-alvo ou estratégias de instrução. Esse fato dificulta os avanços que podem ser realizados nessa área, para melhorar a qualidade dos jogos sérios.

Como objetivo geral, este trabalho visa o desenvolvimento de um jogo sério, denominado *Code in Hand*, que possa auxiliar no aprendizado de conceitos básicos de programação, como variáveis e estruturas de repetição. Para tanto, tem-se os seguintes objetivos específicos:

- Uma breve análise de outros jogos sérios com propostas similares, buscando entender os pontos fortes e fracos de cada um;
- Desenvolver um jogo sério, com o intuito de auxiliar no ensinamento de conceitos básicos de programação;
- Documentar o processo de desenvolvimento do jogo, para contribuir com pesquisas futuras;
- Realizar experimentos com o jogo desenvolvido, para verificar se o mesmo atingiu os objetivos propostos, de ser uma ferramenta eficaz para o aprendizado de conceitos de programação.

II. Referencial Teórico

A. Jogos Sérios

Desde a infância, a maioria de nós é exposta a diversos tipos de jogos, como xadrez, amarelinha ou telefone sem fio. Nos últimos anos, foi aberto outro espaço, não somente para jogos físicos, mas para jogos digitais.

De acordo com [7], um jogo é uma competição entre adversários que agem sobre restrições, que são as regras do jogo, para ao final vencer. E possui os seguintes elementos: objetivos, regras e restrições, narrativa, interação, desafio, competição e conflito, recompensa e feedback.

Os jogos digitais tiveram seu ínicio em 1952 com o jogo OXO, que consistia de uma versão do clássico jogo da velha para o computador EDSAC. Desde então, diversas melhorias foram realizadas nesta área, transformando os jogos digitais em um grande setor desfrutado por mais de 3 bilhões de pessoas [8].

Esses avanços promoveram a realização de pesquisas, para unir os jogos digitais com outros setores, sendo um deles, o setor da educação. Dessa forma, os jogos sérios foram criados, buscando unir o entretenimento dos jogos digitais com estratégias de aprendizado, para suprir as necessidades de ensino dos jovens da atualidade.

B. Jogos Sérios na Educação

Os jogos sérios não são um conceito recente, eles tem sido utilizados para o aprendizado em diversas áreas. Um dos exemplos mais populares de um jogo sério digital é o jogo America's Army, lançado em 2002, ele é da categoria *first-person shooter* e foi criado para informar, educar e recrutar possíveis soldados para o exército americano. Outro exemplo popular de um jogo sério é a franquia de jogos Microsoft Flight Simulator, lançado incialmente em 1982, esses jogos possuem a intenção de simular realisticamente a pilotagem de diversos tipos de aeronaves. Na Figura 1 tem-se outro exemplo, do Intel IT Manager 3, que mostra como realizar diversas atividades gerenciais de forma segura, aprendendo com os erros e tendo um *feedback* instantâneo.

Fig. 1. INTEL IT Manager 3, mostrando algumas das vantagens do uso de jogos sérios na educação (fonte: Von Wangenheim (2013)).

Apesar da existência de jogos sérios a diversos anos, o seu uso nas instituições escolares apenas ganhou força nos últimos tempos. Diversas pesquisas tem sido realizadas, buscando integrar o uso de jogos sérios nas salas de aula, como pode ser visto no trabalho de [3], onde foi criado um jogo sério digital para ensinar conceitos de química, que se provou uma ferramenta efetiva com os participantes do experimento. Os autores explicam que "os resultados demonstraram que o jogo promoveu o aprendizado ativo, concentração e a utilização de tentativa e erro", eles também colocam que "o potencial completo de jogos educacionais ainda não foi descoberto".

[7] explica as vantagens do uso de jogos nas salas de aula, dizendo que eles oferecem um ambiente seguro estimulando a experimentação e a visualização de consequências. Outro mérito dos jogos sérios apresentado pelo texto, é o seu fornecimento de feedback instantâneo aos alunos, além do processo de aprendizagem se tornar mais divertido com a sua utilização.

Outra área da educação que recebe diversos estudos sobre jogos sérios, é o setor de ciência de computação. O trabalho de [6] realizou uma revisão sistemática, trazendo consigo mais de 100 jogos sérios, físicos e digitais, que buscam ensinar diferentes áreas da computação.

A pesquisa mostrou que os setores de engenharia de software e fundamentos de programação são as áreas mais populares para jogos sérios. Também foi observado que a maioria dos jogos encontrados são do gênero de simulação, buscando invocar situações da vida real, seguido por jogos de quebra cabeças e aventura.

Apesar desse grande número de jogos sérios desenvolvidos para a área de computação, os autores nos informam que a grande maioria dos jogos selecionados possuem pouca documentação explicando o processo de desenvolvimento, ou estratégias instrucionais empregadas. Os autores também nos informam que a maioria dos trabalhos realizou testes utilizando uma abordagem não experimental, o que dificulta a avaliação do impacto do jogo.

Estes quesitos dificultam o desenvolvimento de novos jogos sérios, pois não trazem consigo técnicas instrucionais ou elementos do game design, que são de extrema importância quando se planeja o desenvolvimento de um jogo sério.

C. Game Design Document

Quando se organiza o desenvolvimento de um novo jogo, a primeira etapa a ser feita é a pré-produção. Nesta etapa é realizado o planejamento do jogo, para definir o conceito principal, se existirá algum elemento multijogador, público alvo, etc. Para organizar essas ideias, na maioria das vezes é criado um *Game Design Document* (GDD).

Para tanto, pode-se utilizar diversas ferramentas. Aqui, adota-se o modelo *Unified Game Canvas*, que possui os seguintes elementos [9]:

- Game Concept: nome do jogo, objetivo, gênero;
- Game Player: o público alvo que pretende-se atingir;
- Game Play: o espaço do jogo, limitações, desafios, regras;
- Game Flow: aleatoriedade, testes de habilidade, incentivo;
- Game Core: mecânicas, efeitos;
- Game Interaction: controles, plataforma, configurações;
- Game Impact: emoção, diversão, aprendizado;
- Game Business: custo, canais de venda, material bônus.

Ainda na etapa de design, é desenvolvido um fluxograma representando as cenas do jogo, para auxiliar na produção da interface, como o exemplo da Figura 2.

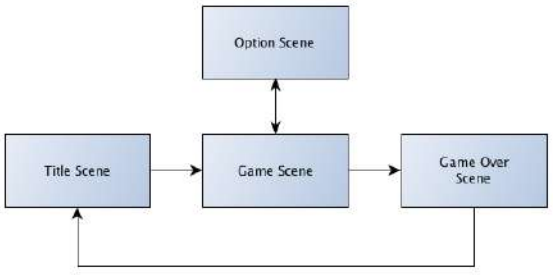

Fig. 2. Exemplo de um fluxograma de cenas simples. (fonte: https://www.raywenderlich.com/)

D. Engines para Jogos

Uma decisão de extrema importância a ser feita antes de iniciar o desenvolvimento de um jogo é escolher que *engine* de jogos será utilizada. As *engines* de jogos são camadas de software reutilizáveis que permitem a separação entre os conceitos de jogos e os recursos do jogo, como os níveis, a arte, etc [10].

Certas desenvolvedoras de jogos possuem suas próprias *engines*, porém ainda existem algumas *engines* disponibilizadas para o público. Algumas das principais *engines* utilizadas no mercado são:

- **Unity**[1]: anunciada na conferência de desenvolvedores mundial da Apple em 2005, ela se tornou uma das primeiras grandes *engines* utilizadas por desenvolvedores indies[2]. A Unity é uma ferramenta muito versátil, permitindo o desenvolvimento de jogos tanto em 3D, quanto em 2D, facilita o processo de importar jogos para multiplas plataformas, como computador, dispositivos móveis ou consoles e aceita scripts escritos com UnityScript (linguagem própria da engine com sintaxe similar a JavaScript) ou C#. Entretanto, C# é a linguagem recomendada, já que o suporte para UnityScript foi descontinuado. A *engine* também possui grande suporte da comunidade, com diversos tutoriais focados no desenvolvimento com a Unity e uma loja de recursos, onde é possível baixar aquivos de música, arte, ou scripts já feitos, de graça ou pagando o preço decidido pelo autor do pacote. A Unity está disponível de graça para uso pessoal ou para empresas que possuem um lucro anual menor que US$100 000, ou pagando uma taxa mensal ou anual para ter acesso a Unity Pro que também disponibiliza certas ferramentas extras, como o GPU profiler, que auxilia com a otimização do jogo.
- **Unreal Engine**[3]: essa *engine* teve seu início em 1998, desenvolvida pela Epic Games para a criação do jogo Unreal. Entretanto, ela somente foi liberada para o público em 2014 com a versão Unreal Engine 4. A Unreal Engine é mais complexa de aprender, se comparada com a Unity, porém ela ainda possui suas vantagens como a sua capacidade de renderizar gráficos de alta fidelidade, sem ocorrer grandes prejuízos na performance do jogo, o que a torna uma favorita entre desenvolvedores AAA[4]. A linguagem de programação suportada pela Unreal Engine é o C++, além de possuir programação visual com a utilização de nodos, chamada de Blueprint Visual Scripting, permitindo que pessoas com pouco conhecimento de C++ ou de programação em geral, possam desenvolver mecânicas simples. A Unreal Engine está disponibilizada completamente de graça para uso pessoal, mas caso o desenvolver venda um jogo produzido com a *engine*, uma taxa de 5% dos lucros do produto deverá ser pago para a Epic Games, apenas se o jogo gerou um rendimento maior do que 1 milhão em dólares americanos. Diferente da Unity, a Unreal Engine disponibiliza o seu código fonte para quem tem acesso a engine.
- **Godot**[5]: é uma *engine* mais recente, tendo o seu lançamento inicial em 2014, porém ela já tem se tornado a favorita por certos desenvolvedores indie. Por ser uma *engine* mais recente e por não possuir um grande suporte financeiro, ainda possui problemas em sua performance com jogos 3D, o que as outras *engines* já tem com questão resolvida. Porém Godot ainda possui grandes vantagens se comparado com seus concorrentes, sendo uma delas o preço da *engine*, que é completamente de graça, até mesmo para jogos que geraram um grande rendimento. Similar a Unreal Engine, o Godot também disponibiliza o seu código de maneira *open source*, com menos restrições do que a Unreal Engine, além de possuir uma base de código menor, a tornando uma das *engines* mais fáceis de contribuir código. Godot também possui maior suporte para diferentes linguagens de programação, podendo receber programas escritos em GDScript, linguagen similar a Python própria da engine, C++ e C#.

As principais características destas engines, são comparadas na Tabela I.

E. Trabalhos Relacionados

Para o desenvolvimento do jogo proposto neste trabalho, foi realizada uma busca por trabalhos que englobassem o desenvolvimento de jogos e ensino de ciência de computação, com o intuito de observar as estratégias de ensino aplicadas nos jogos citados, seus pontos fortes e fracos e as tecnologias utilizadas. Foram selecionados os seguintes trabalhos:

- Program Wars: desenvolvido por [5], é um jogo de cartas que busca ensinar conceitos de programação e segurança cibernética. Ele pode ser jogado entre dois jogadores ou um jogador contra o computador. O objetivo do jogo é acumular pontos utilizando cartas de instrução, repetição

[1]https://unity.com/
[2]Refere-se a desenvolvedores que produzem jogos com baixo orçamento
[3]https://www.unrealengine.com
[4]Classificação para jogos que possuem grande escopo e orçamento
[5]https://godotengine.org/

TABLE I
COMPARAÇÃO ENTRE AS ENGINES SELECIONADAS

Engine	Tipos de Jogos	Pontos Fortes	Preço
Unity	2D e 3D	Importação para um grande número de plataformas e facilidade de uso	De graça para versão padrão com assinatura mensal ou anual para versão pro
Unreal Engine	3D	performance boa para gráficos de alta fidelidade e linguagem visual excelente para fazer protótipos rápidos	De graça com taxa de 5% dos lucros para jogos ge renderam mais de 1 milhão de dólares americanos
Godot	2D	A engine é completamente de graça e seu código está disponível de maneira open source	De graça

e decisão, enquanto ataca o oponente com cartas de *malware* e *hack* e se protege de ataques do oponente com cartas de segurança cibernética. A análise realizada pelos autores, revelou que mais da metade dos participantes tiveram um declínio na compreensão de um programa após jogar o jogo, que pode ter sido causado pelo número limitado de questões no questionário e o uso de múltiplas linguagens de programação nos exemplos. Por outro lado, a maioria dos participantes conseguiu relacionar as cartas do jogo com conceitos de programação. Como melhorias para o jogo, foi discutido a adição de novas cartas para aumentar a complexidade do *gameplay* e das interações entre jogadores, e uma expansão do sistema de pontuação para adicionar novas maneiras de se obter pontos.

- Super Mario Logic: o trabalho de [11], apresenta um jogo baseado na famosa franquia Super Mario, onde o objetivo é chegar até o fim da fase utilizando blocos de comandos que são executados com a utilização de estruturas de programação. O jogo foi testado com quarenta e três estudantes que possuem disciplinas associadas à lógica de programação em sua grade curricular, obtendo resultados bem positivos com os participantes. Para trabalhos futuros, destacou-se a implementação de um sistema de *ranking* para promover o desafio entre os estudantes, e desenvolver um sistema de tempo inicial para cada desafio, para fazer com que o aluno analise o desafio e pense em como solucionar o problema.
- Baralho das Variáveis: neste trabalho de [12], é discutido um jogo sério de cartas, onde o objetivo é organizar as cartas de variáveis e operadores, para no final obter o resultado desejado que pode ser uma string ou um número. Os testes foram realizados com uma turma de Sistemas de Informação da Universidade Federal do Pará, onde foi aplicado um questionário para comparar o aprendizado dos alunos que jogaram o jogo com os que não jogaram. Observou-se que os participantes que jogaram o jogo, obtiveram uma média maior de acertos no questionário. Entre os trabalhos futuros, foram destacados o desenvolvimento de novas fases, melhoria dos níveis de dificuldade e melhoria das animações e recursos audiovisuais.

Analisando os jogos selecionados, é possível observar que a complexidade do jogo e a dificuldade dos objetivos, são considerados pontos importantes para certos participantes. Por esse motivo, o jogo desenvolvido neste trabalho incluirá desafios em até três níveis de dificuldade: fácil, médio e difícil. Dessa forma será possível agradar os jogadores que podem se sentir frustrados quando não conseguem concluir um objetivo, e também os jogadores que apreciam um desafio maior para testar suas habilidades.

Outro ponto importante que pode ser levantado, se encontra no trabalho de [5] que obteve menos sucesso do que os outros trabalhos, apontou o baixo número de questões e a utilização de múltiplas linguagens de programação nos exemplos do questionário como as principais causas para seus resultados. Com isso vale lembrar a importância de um bom questionário, com uma quantidade suficiente de questões e o uso de apenas uma linguagem de programação ou pseudocódigo para exemplificar as perguntas.

III. METODOLOGIA

Para realizar a modelagem do jogo será utilizado o GDD *Unified Game canvas*. Como não se planeja vender o jogo, o elemento Game Business foi desconsiderado na elaboração do canvas. Um fluxograma de cenas também será projetado, para auxiliar na elaboração da interface de usuário.

Comparando as *engines* apresentadas no referencial teórico, foi decidido utilizar a Unity para o desenvolvimento desse jogo. Essa decisão ocorreu em conta das ferramentas disponibilizadas pela Unity para a construção tanto de jogos 2D quanto 3D, por permitir a importação para um grande número de plataformas e por possuir uma loja de recursos com uma grande quantidade de conteúdo para acelerar o processo de desenvolvimento do jogo.

O layout das cartas será criado utilizando o GIMP, software de manipulação de imagens *open source*. O seguinte modelo foi utilizado para a criação das cartas do jogo (conforme modelo em Figura 3).

Para a obtenção de resultados, o jogo será testado com uma turma no segundo semestre de engenharia de computação da Universidade Federal do Rio Grande, onde os alunos tentarão resolver alguns dos desafios definidos no jogo. Para realizar a coleta de dados, os participantes também deverão responder um pequeno questionário, para servir como avaliação sobre a qualidade do jogo.

IV. DESENVOLVIMENTO DO JOGO

Como apresentado na metodologia, o primeiro passo a ser tomado no desenvolvimento do jogo é o planejamento do design utilizando o modelo *Unified Game Canvas*. O canvas criado pode ser visualizado na Figura 4.

No *Unified Game Canvas* desenvolvido, importante ressaltar que o público alvo do jogo são pessoas que estão começando

Fig. 3. Modelo utilizado para a criação do layout das cartas (fonte: https://www.thegamecrafter.com/)

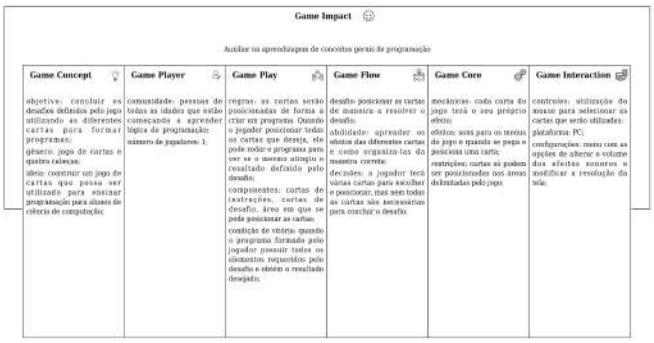

Fig. 4. *Unified Game Canvas* projetado para o jogo

a aprender programação em qualquer faixa etária. O jogo será desenvolvido apenas para PC, com o *gameplay* focado na utilização de mouse e teclado. Destaca-se também o objetivo do jogador no jogo, em criar programas utilizando as diferentes cartas e seus efeitos, onde a condição de vitória ocorre quando o programa desenvolvido, possui todos os componentes requeridos pelo desafio e o mesmo obtém o resultado desejado.

A. Desenvolvimento das cenas do jogo

Em seguida ocorreu o início do planejamento das cenas do jogo. O fluxograma da Figura 5 representa o fluxo de cenas, desde o início do jogo até o jogador começar algum desafio.

O jogo consistirá de quatro cenas, incluindo o menu principal onde o jogo começa, com botões para iniciar o jogo, entrar no menu de opções e sair do jogo. O menu de opções, com *sliders* para alterar o volume dos efeitos sonoros e da música, um botão de *dropdown* para alterar a resolução e um botão para voltar para o menu principal. Clicando no botão de iniciar do menu principal irá direcionar o jogador para a cena de seleção de desafio, permitindo a escolha de um desafio para tentar resolver de dificuldade fácil, moderada ou difícil e um

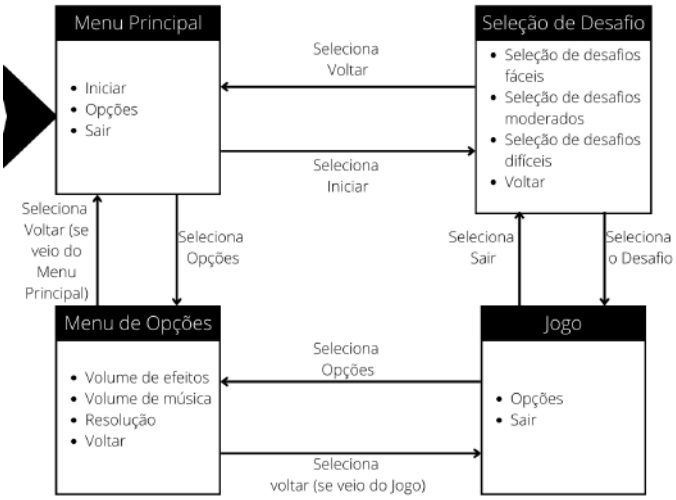

Fig. 5. Fluxo de cenas do jogo desenvolvido

botão para voltar para o menu principal. Por fim, quando o jogador seleciona o desafio ele será colocado na cena do jogo, onde ele pode tentar resolver o problema ou voltar para a cena de seleção de desafio clicando no botão sair.

B. Desenvolvimento do gameplay do jogo

Com as cenas finalizadas, deu-se início ao desenvolvimento das funcionalidades relacionadas ao *gameplay* do jogo. O primeiro passo foi definir as estruturas de código que serão utilizadas. Como o jogo tem o objetivo de ensinar apenas conceitos básicos de programação, as seguintes estruturas de código serão utilizadas nos desafios:

- variáveis;
- operações matemáticas básicas (soma, subtração, multiplicação, divisão, resto);
- *print*;
- *if* e *else*;
- operações de comparação (igual a que, maior que, maior ou igual que, menor que, menor ou igual que).

Após, foram criadas cartas baseadas nessas estruturas de código, utilizando o modelo demonstrado na metodologia (Figura 6).

Cada carta possui a estrutura que ela representa no centro, e abaixo uma breve descrição do que ela faz. Também foram criadas cartas de desafio, que possuem a estrutura apresentada na Figura 7.

O topo da carta mostra o nível de dificuldade do desafio, e o restante explica o programa que o desafio espera. Cada Desafio possui certas cartas, que são distribuídas para o jogador na cena de jogo. Na Figura 8 tem-se uma representação da tela do usuário, quando o mesmo inicia um desafio.

Na cena de jogo, estão os botões para voltar para o menu principal, menu de opções e rodar o programa, com a carta de desafio ao lado deles. No meio está a área que o jogador pode soltar as cartas, para formar o programa, e abaixo estão as cartas disponibilizadas pelo desafio. Vale notar que o jogador

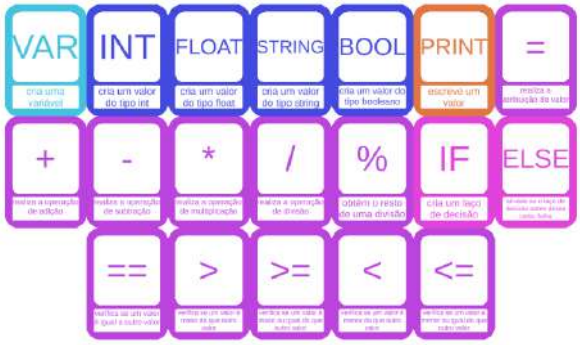

Fig. 6. Cartas criadas para o jogo

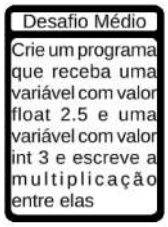

Fig. 7. Carta de desafio desenvolvida para o jogo

sempre recebe mais cartas do que é necessário para completar o desafio. Essa decisão foi feita para aumentar a dificuldade dos desafios e incentivar o usuário à pensar em todas as opções que ele possui para completar o seu programa.

Para formar o programa o jogador deve arrastar as cartas e soltá-las nos espaços designados. Toda vez que uma nova carta é posicionada, novos espaços são criados, permitindo que o jogador posicione cartas de maneira horizontal para continuar na mesma linha, ou de maneira vertical para criar uma linha nova. Quando o usuário quiser, ele poderá apertar o botão de rodar o código, fazendo com que o jogo passe por todas as cartas posicionadas, escrevendo o conteúdo delas em um arquivo de texto que representa o código do usuário. Após o jogo ler todas as cartas, o arquivo de texto passará para a etapa de análise de código, que será explicada mais adiante.

Dentro da cena de jogo também é possível ampliar o

Fig. 8. Cena de jogo ao iniciar um desafio

tamanho das cartas, para ajudar o jogador, caso ele tenha dificuldades em enxergar o conteúdo das cartas. Para fazer isso, basta posicionar o mouse em cima de uma carta da sua mão para ampliar ela ou clicar na carta de desafio para ampliá-la (Figura 9).

Fig. 9. Exemplo de uma carta jogável sendo ampliada

Uma decisão importante que foi preciso tomar no começo do desenvolvimento do jogo, foi entre deixar o jogador digitar os valores que ele quiser em certas cartas ou disponibilizar essa cartas com valores pré definidos. Pretendeu-se seguir com a primeira opção, pois ela simula o ambiente de programação com mais acurácia e dispõe mais oportunidades para o usuário aprender com os erros (Figura 10). As cartas em que o jogador pode digitar o valor são cartas de variável, *int*, *float*, *bool* e *string*.

Fig. 10. Exemplo da interface de digitação de uma carta de variável

Os desafios foram separados em três níveis de dificuldade: fácil, médio e difícil. As dificuldades foram classificadas da seguinte maneira:

- fácil: programas de uma a três linhas, focado apenas no uso de variáveis e *print*, poucas cartas;
- médio: programas maiores, utilização de mais operações matemáticas;
- difícil: programas grandes, foco na utilização de blocos *if* e *else*.

São três desafios para cada dificuldade e o jogador no começo só tem acesso ao primeiro desafio de cada uma, podendo desbloquear os próximos ao completar o desafio anterior. Ao posicionar o mouse em cima do botão de uma desafio, a carta daquele desafio aparece na direita dos botões, permitindo que o usuário leia o que o problema pede, antes de iniciá-lo. Ao concluir o problema o usuário é levado ao menu principal e o botão que leva ao desafio concluído ganha uma coloração amarela, para indicar a vitória.

C. Análise de código

Para que o objetivo do jogo seja alcançado, percebeu-se a necessidade de automatizar o processo de avaliação de código do usuário, visto que nem sempre será possível que tenha uma pessoa com o conhecimento para fazer a análise manual. Para isso a análise automática dos códigos foi divida em três partes:

- análise sintática;
- análise estrutural;
- análise de saída.

1) Análise sintática: Essa análise verifica se o código do usuário é um código válido. Para que essa análise funcione, foi criada uma gramática de linguagem de programação simples utilizando o ANTLR4, gerador de interpretadores poderoso, muito utilizado para a construção de linguagens, ferramentas e *frameworks*.

Caso aconteça algum erro durante o processo de compilação, o jogador irá receber uma mensagem indicando a linha onde o erro aconteceu e a descrição do erro.

2) Análise estrutural: A análise estrutural busca por certas linhas de código, que são vistas como necessárias para o programa. Cada desafio é dado algumas linhas de código que são essenciais para sua conclusão. O jogo lê o arquivo de texto do programa, linha por linha, e compara com as linhas essenciais do desafio, ignorando espaços, acentos e transformando todas as letras para minúsculas.

Após ele verifica se as linhas essenciais do código estão na ordem correta. Nesta etapa o jogador pode receber um erro sobre a estrutura de seu código ou a ordem da estrutura.

3) Análise de saída: A análise de saída observa o *output* do código do usuário e compara com uma lista de valores que podem ser aceitos como resposta para o desafio. Quando o jogo encontra uma carta de *print* no programa criado pelo usuário, ele escreve os argumentos passados para o *print* em um arquivo chamado *output.txt*. Os conteúdos desse arquivo, são então comparadas com um *array* de valores *string* que são considerados como respostas válidas para o problema.

Caso aconteça um erro nessa etapa, o jogador receberá uma mensagem indicando a resposta esperada do desafio, e a saída recebida de seu programa.

V. Testes com o Jogo Code in Hand

Os testes do jogo foram realizados em duas turmas diferentes da Universidade Federal de Rio Grande, uma com alunos do curso de Sistemas de Informação e outra com alunos de Engenharia da Computação. Para ambas as turmas, os testes ocuparam uma hora no início da aula de Algoritmos e Estruturas de Dados I, disciplina do primeiro ano dos referidos cursos, totalizando 23 alunos voluntários.

Ao final desse período de uma hora, os estudantes responderam um questionário para avaliarem o jogo. O Questionário foi baseado no modelo SUS (System Usability Scale) e possui as seguintes perguntas:

1) Digite o seu email.(Texto)
2) Qual é o seu curso? (Texto)
3) Essa é a sua primeira vez cursando a disciplina de Algoritmos e Estruturas de Dados I? (Sim/Não)
4) Eu acho que gostaria de jogar esse jogo com frequência. (Escala Likert 1-5)
5) Eu acho que o jogo tem um bom nível de complexidade. (Escala Likert 1-5)
   Por que? (Texto - Opcional)
6) Eu achei o jogo fácil de jogar. (Escala Likert 1-5)
   Por que? (Texto - Opcional)
7) Eu acho que conseguiria jogar o jogo sem a ajuda de uma pessoa com conhecimentos técnicos. (Escala Likert 1-5)
   Por que? (Texto - Opcional)
8) Eu acho que as várias funções do jogo estão muito bem integradas. (Escala Likert 1-5)
   Por que? (Texto - Opcional)
9) Eu acho que o jogo não apresenta muita inconsistência.(Escala Likert 1-5)
   Por que? (Texto - Opcional)
10) Eu imagino que as pessoas aprenderão como jogar esse jogo rapidamente. (Escala Likert 1-5)
    Por que? (Texto - Opcional)
11) Eu achei que o jogo não é muito atrapalhado de jogar.(Escala Likert 1-5)
    Por que? (Texto - Opcional)
12) Eu me senti confiante ao jogar o jogo. (Escala Likert 1-5)
    Por que? (Texto - Opcional)
13) Eu não precisei aprender muitas coisas novas antes de conseguir jogar o jogo. (Escala Likert 1-5)
14) Dê suas sugestões em como o jogo pode ser melhorado. (Texto - Opcional)
15) faça o upload do arquivo data.txt que se encontra na pasta Assets/Resources do jogo. (Arquivo)

A. Resultados obtidos

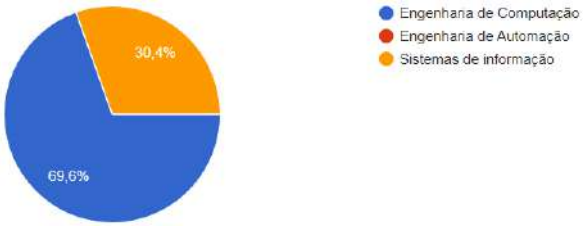

Fig. 11. Respostas da questão “Qual é o seu curso?”

A Figura 11 mostra a distribuição de alunos para cada turma de testes, onde temos 7 alunos de sistemas de informação e 16 para engenharia de computação.

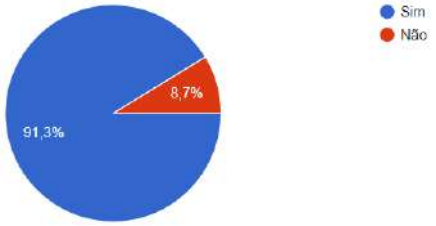

Fig. 12. Respostas da questão "Essa é a sua primeira vez cursando a disciplina de Algoritmos e Estruturas de Dados I?"

Pela Figura 12 é possível ter uma ideia do nível de experiência dos alunos. A maioria está cursando a disciplina de Algoritmos e Estruturas de Dados I pela primeira vez, enquanto apenas 2 alunos já cursaram a disciplina anteriormente, podendo ter um conhecimento maior de programação do que o restante dos estudantes.

O restante das questões buscam avaliar o jogo, de acordo com o que os alunos pensaram durante o período de uma hora que eles jogaram.

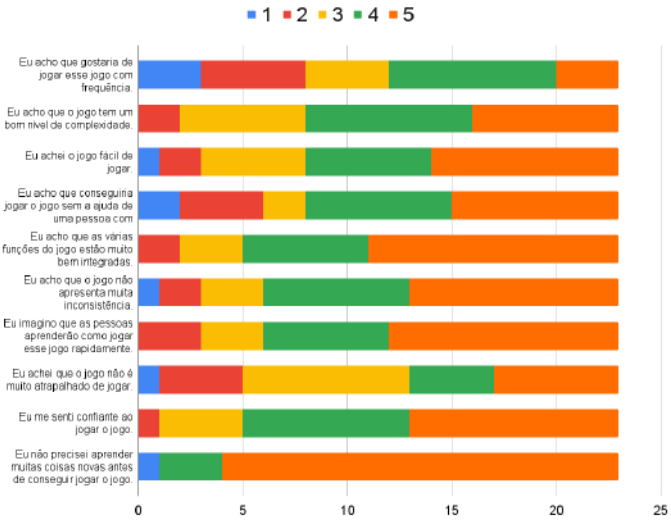

Fig. 13. Respostas das perguntas que avaliam a qualidade do jogo

Observando as respostas obtidas na Figura 13 pode-se perceber, que os alunos receberam o jogo de uma maneira bem positiva. Vale destacar que, como os alunos já tiveram experiência com programação, eles se sentiram confiantes para resolver os problemas exigidos pelo jogo. Entretanto em diversas questões, foi levantada a queixa que pessoas com pouco ou nenhum conhecimento em programação, terão muitas dificuldades para entender e jogar o jogo. Para amenizar esse problema, foi planejado a criação de dicas nos desafios, para introduzir os conceitos de programação e melhorar o tutorial, para que ele apresente explicações sobre o funcionamento do jogo de forma mais aprofundada.

Como o Code In Hand é um jogo de cartas, ele não será para o agrado de todos. Mesmo assim, uma boa parte dos alunos responderam que jogariam o jogo com uma certa frequência. No geral, a complexidade do jogo e o funcionamento de seus componentes foram considerados bons, entretanto, os estudantes acreditam que o jogo é meio atrapalhado de se jogar. Espera-se que as melhorias no tutorial, consigam resolver esse problema.

Na área de sugestões, alguns alunos disseram que gostariam que fossem adicionados desafios mais complexos, para aumentar a dificuldade para quem entende de programação. Também foi sugerido a adição de dicas para os desafios, para auxiliar as pessoas com pouco ou nenhum conhecimento de programação. Por fim, outras sugestões que apareceram diversas vezes foram a melhoria da interface, para torná-la mais intuitiva, e polir as mecânicas do jogo.

A análise dos arquivos data.txt será realizada como trabalho futuro, para ter tempo de realizar uma mineração de dados extensa. É esperado que a partir desses arquivos seja possível obter quais questões os alunos tiveram mais dificuldades, os erros mais comuns obtidos nos testes, qual desafio foi o menos concluído, etc.

VI. Conclusão

Este trabalho teve como objetivo principal o desenvolvimento de um jogo de cartas sério chamado Code In Hand, para servir como uma maneira divertida no auxílio do ensino de programação. Ao longo do texto foi apresentada a fundamentação teórica, desde o que são jogos sérios e sua história, o uso de *GDDs* e *game engines* para o desenvolvimento de jogos, o ensino de programação desde a infância, jogos sérios com a temática similar ao tema proposto neste trabalho e como ocorreu o desenvolvimento do jogo Code In Hand.

Para avaliar a qualidade do jogo, o mesmo foi testado com duas turmas da Universidade Federal de Rio Grande. Uma com 7 alunos do curso de Sistemas de Informação e outra com 16 alunos de Engenharia de Computação, cada uma com uma hora para jogar o jogo, para após responder um questionário.

Analisando as respostas obtidas, percebeu-se que o jogo foi bem recebido pelos alunos. Observando o *feedback* passado pelos participantes dos experimentos, foi possível vislumbrar os seguintes trabalhos futuros:

- Melhorar o tutorial para explicar mais ao fundo as mecânicas do jogo;
- Adicionar dicas aos desafios para introduzir conceitos de programação, para ajudar os jogadores que não possuem esse conhecimento;
- Melhorar a interface do jogo;
- Polir as mecânicas do jogo, como a troca e renomeação de cartas sem precisar remover a carta da mesa, pressionar enter para confirmar o valor da carta, correção de *bugs*;
- Realizar a mineração de dados sobre os arquivos data.txt para obter mais informações interessantes sobre os testes.

Cabe discutir o escopo que o jogo possa ter, em relação a diversos níveis de conhecimento de programação. Os participantes dos experimentos já tiveram meses de experiência com programação, e assim tiveram uma certa facilidade ao jogar o jogo. Entretanto, pessoas que não possuem esse conhecimento, iram encontrar muito mais dificuldades para jogar o jogo.

Referências

[1] NETO, V. D. A. Uma análise sobre reprovação no curso de Ciência da Computação na UFRGS sob a ótica dos alunos. Porto Alegre: Universidade Federal do Rio Grande do Sul, 2021.

[2] RODRIGUES, F. S. Estudo sobre a evasão no curso de Ciência da Computação da UFRGS. Porto Alegre: Universidade Federal do Rio Grande do Sul, 2013.

[3] RASTEGARPOUR H., . M. P. The effect of card games and computer games on learning of chemistry concepts. Procedia - Social and Behavioral Sciences, v. 31, p. 597– 601, 2012.

[4] CASAROTTO G. BERNARDI, A. Z. C. e. R. D. M. R. I. Logirunner: um jogo de tabuleiro como ferramenta para o auxílio do ensino e aprendizagem de algoritmos e lógica de programação. Renote, Porto Alegre, v. 16, 2018.

[5] ANVIK, V. C. J.; RIEHL, J. Program wars: A card game for learning programming and cybersecurity concepts. In: . Minneapolis, MN, USA. ACM, New York, NY, USA: In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE'19), 2019.

[6] BATTISTELLA PAULO GRESSE VON WANGENHEIM, C. Games for teaching computing in higher education – a systematic review. IEEE Technology and Engineering Education (ITEE) Journal, v. 9, p. 8–30, 2016.

[7] WANGENHEIM, C. G. von Wangenheim e Aldo von. Ensinando Computação com Jogos. São Paulo: Bookess, 2013.

[8] CLEMENT, J. Number of video gamers worldwide in 2021, by region. Statista, 2022. Disponível em: ¡https://www.statista.com/¿.

[9] SARINHO, V. T. Uma proposta de game design canvas unificado. SBC – Proceedings of SBGames, 2017.

[10] ANDRADE, A. Game engines: a survey. EAI Endorsed Transactions on Game-Based Learning, v. 2, p. 150615, 11 2015.

[11] PANEGALLI, F. S. Super Mario Logic: um jogo sério para lógica de programação. Santa Maria: Universidade Federal de Santa Maria, 2016.

[12] FRANÇA ROBERTO COSTA NUNES, R. C. d. S. Elvis Lopes de. Jogo baralho das variáveis uma proposta de utilização de jogos para disciplina de Programação. Marabá: Universidade Federal do Pará, 2012. Disponível em ¡http://repositorio.unifesspa.edu.br/handle/123456789/192¿.