tOOwer Defenders: Um Jogo para Auxiliar no Aprendizado de Programação Orientada a Objetos

Lucas dos Santos Martins[1], **Aline Vieira de Mello**[1]

[1]Universidade Federal do Pampa (Unipampa)
Alegrete – RS – Brasil

lgatts@gmail.com, alinemello@unipampa.edu.br

Resumo. *O aumento na capacidade de processamento dos computadores elevou a complexidade dos softwares, exigindo a adoção de novos paradigmas de desenvolvimento. A Programação Orientada a Objetos (POO) emergiu como uma solução para simplificar problemas complexos e promover a reutilização de código, porém seu aprendizado é desafiador. Neste contexto, apresentamos um jogo digital concebido para auxiliar no aprendizado da POO. O desenvolvimento baseou-se em princípios de Engenharia de Software e foi fundamentado em uma revisão da literatura sobre o tema. A validação qualitativa do jogo mostrou que ele foi eficaz em ensinar conceitos como instanciação e herança, mas ainda necessita de melhorias para transmitir o conceito de polimorfismo.*

Abstract. *The increase in computer processing power has led to more complex software, requiring the adoption of new development paradigms. Object-Oriented Programming (OOP) emerged as a solution to simplify complex problems and promote code reuse, but learning it is challenging. In this context, we present the digital game* ***tOOwer Defender****, designed to assist in learning OOP. The development was based on Software Engineering principles and was based on a literature review on the topic. The qualitative validation of the game showed that it was effective in teaching concepts such as instantiation and inheritance, but still needs improvements to convey the concept of polymorphism.*

1. Introdução

No final dos anos 1980, com a evolução da capacidade de processamento dos computadores, a complexidade dos softwares desenvolvidos também aumentou. Esse crescimento resultou em um aumento nos custos de produção de software, ultrapassando os custos de produção de hardware [de Souza Meirelles 1994]. Diante dessa mudança, tornou-se necessária uma evolução nas metodologias de desenvolvimento de software, que até então eram predominantemente procedurais [Van Emden 1997].

Em paralelo com a evolução do hardware, idealizava-se uma nova metodologia de programação, o paradigma orientado a objetos. A Orientação a Objetos (OO) permite modelar software de forma mais próxima da realidade, através da abstração de objetos do mundo real em objetos computacionais [Sebesta 2009]. Essa forma de modelagem auxilia na simplificação de sistemas complexos. Além disso, a OO facilita a reutilização de código, reduzindo o tempo de desenvolvimento. Devido ao aproveitamento de código por meio de abstração, herança e composição, a manutenção e evolução de sistemas se

DOI: 10.5753/sbie.2024.242551

tornam mais simples, uma vez que a modificação em uma classe pode afetar várias partes do sistema de uma só vez [Klump 2001].

Apesar da OO possuir diversas vantagens, estudantes de cursos de computação têm apresentado dificuldades em seu aprendizado. Os trabalhos de [Rais et al. 2011] e [Edirisinghe 2008] utilizaram questionários para constatar a dificuldade dos estudantes em compreender conceitos básicos de OO, como classes e objetos. Adicionalmente, uma análise das médias finais do componente curricular de Programação Orientada a Objetos (POO) dos cursos de Computação na instituição dos autores no período de 2020 a 2023 mostrou que existe uma taxa de reprovação de aproximadamente 34%.

Uma possível abordagem para contornar esse problema de aprendizado é a utilização de jogos como ferramenta de auxílio ao ensino. Existem alguns jogos digitais que foram desenvolvidos para o ensino de OO [Corral et al. 2014, Schmolitzky and Gottel 2014, Yan 2009, Karram 2021]. Entretanto, esses jogos possuem um caráter explicitamente educacional, muitas vezes deixando a desejar no quesito entretenimento. Além disso, alguns desses jogos ainda requerem codificação, o que pode diminuir o apelo lúdico e dificultar a experiência de aprendizado para alguns estudantes [Videnovik et al. 2023, Wang et al. 2022].

Estudos indicam que o engajamento é um fator crítico no aprendizado, pois aumenta a motivação e a retenção de conhecimento [Videnovik et al. 2023]. Neste contexto, jogos do gênero *Tower Defense* podem auxiliar no aprendizado de POO devido a sua capacidade de manter os jogadores engajados por longos períodos [Stardock 2023, Droid 2023]. Assim, apresentamos o jogo digital **tOOwer Defender** concebido para auxiliar na compreensão dos conceitos de OO. O desenvolvimento do jogo baseou-se em princípios de Engenharia de Software e foi fundamentado em uma revisão da literatura sobre o tema.

2. Trabalhos Relacionados

Uma revisão sistemática foi realizada com o objetivo de identificar trabalhos focados no desenvolvimento e na validação de jogos aplicados ao ensino dos conceitos de OO. É importante destacar que trabalhos em que os estudantes desenvolvem jogos para aprender esses conceitos não foram incluídos nesta revisão. A seguir, o protocolo adotado para a revisão é descrito e os resultados obtidos são analisados.

2.1. Protocolo da Revisão

Para conduzir a revisão sistemática, o protocolo proposto por [Kitchenham et al. 2007] foi usado como base. O primeiro passo foi verificar a necessidade da revisão, o que envolveu a busca por outros trabalhos de revisão com o mesmo objetivo. Uma revisão realizada em 2016 com objetivos similares foi encontrada [Abbasi et al. 2017], mas ela incluía jogos desenvolvidos com a abordagem ”desenvolver para aprender”, o que não se alinha aos objetivos deste trabalho e não contemplava os últimos anos. Assim, constatamos a necessidade de uma nova revisão com as seguintes questões de pesquisa:

QP1. Quais jogos foram criados com o objetivo de ensinar OO?
QP2. Quais conceitos de OO são contemplados nos jogos propostos?
QP3. Alguma forma de validação dos jogos propostos foi realizada? Se sim, qual?

Para realizar a pesquisa, utilizamos três bases de dados digitais: Scopus, ACM Digital Library e IEEE Xplore. A seguinte *string* de busca foi elaborada e utilizada em todas as bases:

"teach*" OR "learn*")
AND ("oop" OR ("Object Oriented" AND "Programming" OR"Paradigm")))
AND "game")

Todos os artigos encontrados utilizando a *string* de busca foram considerados. Posteriormente, aplicamos os seguintes critérios de exclusão (CE):

CE1. O artigo é duplicado.
CE2. O artigo não aborda a utilização ou desenvolvimento de jogos para auxiliar o aprendizado de conceitos de OO.
CE3. O artigo não possui dados suficientes para responder às questões de pesquisa.

2.2. Resultados da Revisão

QP1. Quais jogos foram criados com o objetivo de ensinar OO? Foram encontrados treze jogos em doze artigos, pois um dos artigos utilizava dois jogos para o ensino de OO. O gênero de jogo utilizado variou bastante, incluindo dois jogos de RPG [Wong et al. 2014, Wong et al. 2017], dois jogos de Ação [Jiménez-Díaz et al. 2007, Zhang et al. 2013], três de *Puzzle* [Rais et al. 2011, Esper et al. 2014, Galgouranas and Xinogalos 2018], dois jogos de *Quiz* [Sharma et al. 2015, Ramirez-Rosales et al. 2016], um *Card Game* [de Oliveira et al. 2017], um de *Tower Defense* [Jordine et al. 2014] e até um de modelagem 3D [Djelil et al. 2015], em que o usuário precisa observar alguns conceitos de OO para criar os modelos. Esses jogos variam em gênero e metodologia, evidenciando diferentes estratégias de ensino que podem complementar o desenvolvimento do tOOwer Defenders.

QP2. Quais conceitos de OO são contemplados nos jogos propostos? Todos os jogos apresentam alguma forma de representar os conceitos de classe, objetos e instanciação. Cinco jogos trabalham com os conceitos de encapsulamento e herança [Rais et al. 2011, Wong et al. 2014, Sharma et al. 2015, Wong et al. 2017, de Oliveira et al. 2017]. Por fim, apenas quatro jogos abordam o conceito de polimorfismo [de Oliveira et al. 2017, Wong et al. 2014, Sharma et al. 2015, Wong et al. 2017]. Os conceitos aparecem de forma variada dentro dos jogos. Em alguns, por exemplo, estão na forma textual, como questionários dentro do próprio jogo. Em outros, como código que o jogador deve digitar para realizar uma ação dentro do jogo. E ainda, em alguns, os conceitos estão incorporados nos próprios elementos do jogo, como no jogo "Odyssey of Phoenix", onde a criação de uma receita representa classes e objetos, e o conceito de herança é explorado através de ingredientes compartilhados.

QP3. Alguma forma de validação dos jogos propostos foi realizada? Se sim, qual? Em relação às validações, três trabalhos não apresentaram nenhum tipo de validação [Jiménez-Díaz et al. 2007, Wong et al. 2014, Jordine et al. 2014]. Três trabalhos utilizaram questionários pré e pós-utilização dos jogos [Zhang et al. 2013, Sharma et al. 2015, Wong et al. 2017]. Esses questionários variaram desde testes sobre conceitos de OO até questionários qualitativos sobre a percepção dos

usuários em relação aos jogos. Quatro trabalhos [Rais et al. 2011, Djelil et al. 2015, Ramirez-Rosales et al. 2016, Galgouranas and Xinogalos 2018] utilizaram apenas questionários pós-utilização, que também variaram entre testes de conceitos e questões qualitativas. Apenas um trabalho [Esper et al. 2014] utilizou *logs* e gravações em vídeo das sessões de jogos para analisar o engajamento dos usuários.

3. Metodologia

Neste trabalho, foi utilizado um processo baseado no modelo cascata, devido ao conhecimento prévio dos requisitos e à pouca interação com os usuários finais durante o desenvolvimento. A Figura 1 mostra o processo adotado dividido nas fases de Engenharia de Requisitos, Modelagem, Prototipação, Implementação, Validação e Análise de Resultados, as quais são descritas a seguir.

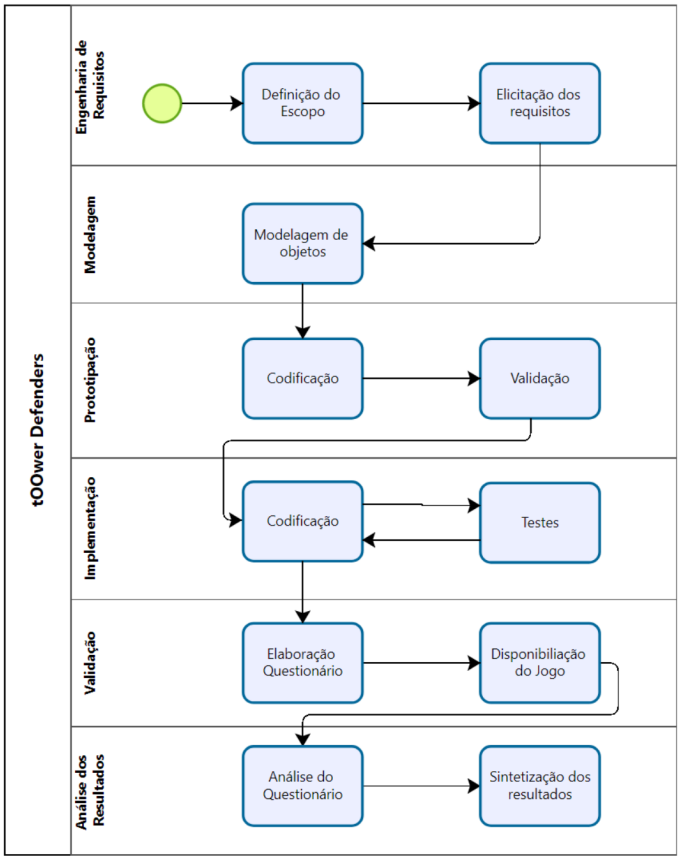

Figura 1. Processo adotado no desenvolvimento do jogo

3.1. Engenharia de Requisitos

Nesta fase foram definidos os requisitos do jogo a partir da revisão sistemática e entrevistas com professores de POO. O jogo foi planejado para incluir cinco fases distintas (Figura 2), cada uma com desafios progressivos para garantir o engajamento do jogador. Foram definidos cinco tipos de torres (Figura 3), cada uma representando um elemento (neutro, gelo, fogo, terra e vento), proporcionando variedade estratégica ao jogador.

Figura 2. Cinco fases do jogo

Figura 3. Cinco torres do jogo cada uma representando um elemento

Inicialmente, três tipos de inimigos foram planejados, cada um com variações baseadas nos elementos, totalizando quinze inimigos diferentes (Figura 4). Essa diversidade de inimigos foi projetada para desafiar os jogadores a adaptarem suas estratégias conforme avançam no jogo. Durante o desenvolvimento, foi identificada a necessidade de adicionar um inimigo final, proporcionando um clímax adequado ao final do jogo.

Figura 4. Três tipos de inimigos do jogo e sua variação por elemento

Ainda nessa fase, foi definida uma combinação de pixel art com arte 2D linear por serem agradáveis a um grande público. O pixel art, em particular, continua sendo popular devido ao seu apelo nostálgico e à capacidade de engajar a imaginação dos jogadores. Além disso, sua simplicidade e acessibilidade o tornam uma escolha

estética atraente tanto para desenvolvedores independentes quanto para consumidores [Studio 2022, Silva et al. 2013]. Destaca-se que todas as artes e músicas utilizadas no jogo possuem licença *Creative Commons*, permitindo o uso livre.

3.2. Modelagem

A fase de Modelagem envolveu a criação do diagrama de classes, o qual representa os objetos do jogo bem como as interações entre eles.

3.3. Prototipação

Nesta fase, foi desenvolvida uma versão inicial do jogo em Unity, que continha as mecânicas básicas (instanciação, ataque da torre, herança e wave de inimigos) e apenas um tipo de inimigo e uma fase, para validar as ideias e conceitos com professores especialistas na área. A prototipação ajudou a identificar problemas e pontos de melhoria antes da implementação final. Na Figura 5 é possível ver a diferença entre o protótipo usado para validação de conceitos e o jogo final.

Figura 5. Diferentes versões do jogo tOOwer Defenders: o protótipo (à esquerda), a primeira versão (à direita), e a versão final (abaixo).

3.4. Implementação

Para a implementação final do jogo, que pode ser acessado no repositório, diferentes *engines* foram utilizadas ao longo do projeto. Inicialmente, foi escolhida a *Unity* devido a sua popularidade e ampla gama de recursos. No entanto, o progresso no desenvolvimento tornou-se lento. Decidiu-se então migrar para o libGDX, uma *engine* baseada em Java. Apesar de facilitar o desenvolvimento inicial, o libGDX revelou-se insuficiente em termos de recursos básicos de uma *engine* de jogo. Finalmente, a escolha recaiu sobre a Godot Engine, uma engine de código aberto que utiliza GDscript, uma linguagem com sintaxe semelhante ao Python. O fluxo de trabalho com Godot mostrou-se muito mais produtivo, permitindo um desenvolvimento mais ágil e eficiente.

3.4.1. Elementos Didáticos

Dentro do jogo, vários elementos foram introduzidos com o objetivo de ensinar os conceitos de POO. O primeiro deles é o conceito de classe e instâncias. Na primeira fase, o jogador é apresentado a uma classe de cristal neutro (cinza), acessível no menu lateral do jogo (Figura 6). Ao clicar nesse menu, o jogador tem a opção de instanciar essa classe, criando uma entidade dentro do jogo. Para isso, é necessário posicionar a instância em uma das bases (Figura 6) que tenha afinidade com o elemento da classe (mesma cor). Por exemplo, uma torre do elemento neutro (cor cinza) pode ser posicionado somente em bases de elemento neutro (cor cinza). Essa mecânica foi projetada para ilustrar o conceito de tipo e contrato de função, onde uma função só aceita argumentos do mesmo tipo.

Figura 6. Menu lateral com a classe de cristal neutra

Na segunda fase, é introduzido o conceito de herança. No início dessa fase, o jogador recebe a segunda classe de cristal, a de gelo (cor azul). Quando ele clica na classe no menu lateral, a opção de herança se torna disponível. Ao clicar nela, um novo menu é aberto (Figura 7), onde ele pode selecionar fazer a herança entre as classes que possui. Ao fazer isso, ele pode optar por sobrescrever ou não o elemento do cristal. Caso opte por sobrescrever, o cristal mantém seu elemento original, já que o cristal selecionado no menu lateral é a classe filha no processo de herança. Se desativar a sobrescrita, a classe usará o elemento da classe mãe (classe que é escolhida no *dropdown*).

Ao finalizar a herança, tanto a classe quanto as instâncias terão um contorno da cor da classe mãe (Figura 8), indicando a herança. Quando se instancia uma classe com herança, o usuário pode escolher tanto bases do elemento da própria classe quanto do elemento da classe mãe (Figura 8). Isso demonstra como uma subclasse pode herdar atributos e métodos de sua superclasse, reforçando a reutilização de código e a extensão de funcionalidades.

Na terceira fase, o conceito de polimorfismo é demonstrado com cada cristal possuindo um ataque específico. Por exemplo, a classe neutra tem um ataque visualmente

Figura 7. Menu de herança. Imagem da esquerda mostra a opção de sobrescrever ligada mantendo assim o elemento da classe. A imagem central a opção de sobrescrever está desligada (retângulo vermelho) fazendo o elemento mudar para gelo (elemento da classe mãe). Por fim a imagem da direita mostra o *drop-down* das classes mães disponíveis para herança.

Figura 8. Mecanismo de herança. A imagem da esquerda mostra o contorno de herança na classe (menu lateral) e na instância (cristal seguindo o mouse). A imagem à direita superior mostra a instanciação da classe neutra e base do mesmo elemento. Já a imagem à direita inferior apresenta a instanciação da classe neutra com herança de gelo em uma base de gelo.

maior e com dano dobrado, já a classe de gelo pode atingir até três inimigos ao mesmo tempo (Figura 9). Quando o usuário instancia uma classe com herança na base do mesmo elemento da classe filha, ele pode escolher usar o ataque da classe mãe ou da classe filha (Figura 9). No entanto, se a instância for criada na base do elemento da classe mãe, ele só poderá usar o ataque da classe mãe. Isso ilustra como objetos de subclasses podem ser tratados como objetos de sua superclasse (polimorfismo).

Figura 9. Tipos de ataques. A imagem mais a esquerda demonstração o ataque classe neutra, a imagem central o ataque da classe de gelo e por fim a imagem da direta o cristal neutro utilizando o ataque de gelo devido a herança.

3.5. Validação

Para realizar a validação do jogo, os participantes foram convidados a jogar o jogo **tOOwer Defenders** até alcançar pelo menos a terceira fase e depois responderem um questionário de avaliação que se encontrava na tela inicial do jogo.

O questionário inicia com um termo de consentimento livre e esclarecido em que a pesquisa é apresentada e o participante deve concordar com o uso dos dados fornecidos de forma anônima para este trabalho. Somente após o aceite, as demais perguntas são apresentadas. Três perguntas são de múltipla escolha, uma é aberta e as demais foram respondidas utilizando uma escala *Likert* de cinco pontos, variando de "discordo totalmente" a "concordo totalmente".

O questionário foi dividido em quatro seções, conforme apresentado na Tabela 1. A primeira seção visa conhecer o participante, incluindo perguntas sobre o curso que está cursando e a frequência com que joga videogames. A segunda seção busca entender o conhecimento do participante sobre conceitos de POO. A terceira seção é dedicada à avaliação do jogo pelo participante, abrangendo aspectos de entretenimento e a eficácia do jogo na compreensão dos conceitos de POO. E, por fim, há um espaço aberto para criticas, elogios e sugestões.

3.6. Análise de Resultados

A fase final de Análise de Resultados envolveu a interpretação dos dados coletados para avaliar o sucesso do jogo tanto em termos de entretenimento quanto de educação.

4. Resultados da Avaliação

4.1. Perfil

O jogo **tOOwer Defenders** foi avaliado por nove pessoas. Quanto ao curso de graduação (Q1), cinco participantes cursam ou cursaram Engenharia de Software, dois participantes cursaram Sistemas de Informação, um participante cursou Ciência da Computação e um participante é estudante da educação básica. Portanto, o perfil dos participantes é quase totalmente composto por estudantes de cursos da área de computação.

Tabela 1. Questões da validação

Id	Questão	Tipo
	Perfil	
1	Qual seu curso?	Múltipla Escolha
2	Em qual período do curso você está?	Múltipla Escolha
3	Eu jogo videogames regularmente.	Likert
	Conhecimentos de POO	
4	Eu entendo os conceitos básicos de Programação Orientada a Objetos (POO).	Likert
5	Eu já desenvolvi projetos que utilizam POO.	Likert
6	Eu me sinto confiante ao aplicar POO em meus projetos de programação.	Likert
	Avaliação do Jogo	
7	Quanto tempo mais ou menos você jogou?	Múltipla Escolha
8	A jogabilidade é intuitiva e fácil de aprender.	Likert
9	O nível de dificuldade é adequado.	Likert
10	Eu achei o jogo divertido e envolvente.	Likert
11	O jogo me ajudou a entender como criar instâncias de classes em POO.	Likert
12	O jogo facilitou minha compreensão sobre como funciona a herança entre classes em POO.	Likert
13	O jogo me ajudou a entender como aplicar polimorfismo em POO.	Likert
	Espaço aberto	
14	Espaço aberto para elogios, críticas, comentários e sugestões.	Aberta

Em relação do período do curso (Q2), cinco participantes já são formados. Três são estudantes do 5º, 6º e 10º período e um não é estudante de curso de graduação. Portanto, a maioria dos participantes ou já está formado ou está em períodos avançados do curso, indicando um nível avançado de conhecimento. Isso sugere que suas avaliações são mais críticas, entretanto, ao considerar a eficácia do jogo em auxiliar a compreensão dos conceitos, pode haver um viés devido ao conhecimento prévio desses participantes.

Quanto a jogar videogame regularmente (Q3), sete participantes indicaram que jogam videogames regularmente e dois responderam de forma neutra. Isso pode influenciar positivamente a receptividade e a interação com o jogo desenvolvido, pois esses participantes provavelmente têm experiência prévia com mecânicas de jogo e interfaces similares. Além disso, podem fornecer *feedback* valioso sobre a jogabilidade e a eficácia do jogo em termos de engajamento e diversão.

4.2. Conhecimentos de POO

Como esperado pelas respostas anteriores, seis participantes indicaram que entendem bem os conceitos básicos de POO (Q5), o que sugere que a maioria tem uma boa compreensão teórica desses conceitos, dois participantes indicaram que não entendem os conceitos básicos de POO e um participante indicou que tem uma compreensão moderada dos conceitos.

Sobre o desenvolvimento de projetos que utilizam POO (Q6), seis participantes indicaram que já desenvolveram projetos, o que indica experiência prática com esses con-

ceitos. Isso sugere que o *feedback* deles sobre o jogo pode ser baseado em um conhecimento já consolidado. Dois participantes indicaram que nunca desenvolveram projetos utilizando POO e um tem uma experiência prática moderada.

4.3. Avaliação do Jogo

Em relação ao tempo de jogo (Q7), cinco participantes jogaram por mais de 15 minutos, três jogaram entre 5 e 15 minutos e 1 participante jogou até 5 minutos. Isto demonstra que a maioria dos participantes se engajaram na tarefa de avaliar.

A Figura 10 mostra as respostas dos participantes para as questões Q8 a Q13, que buscam obter a percepção do usuário quanto a jogabilidade, facilidade de uso, diversão e envolvimento do jogo, além de auxilio no entendimento dos conceitos de POO.

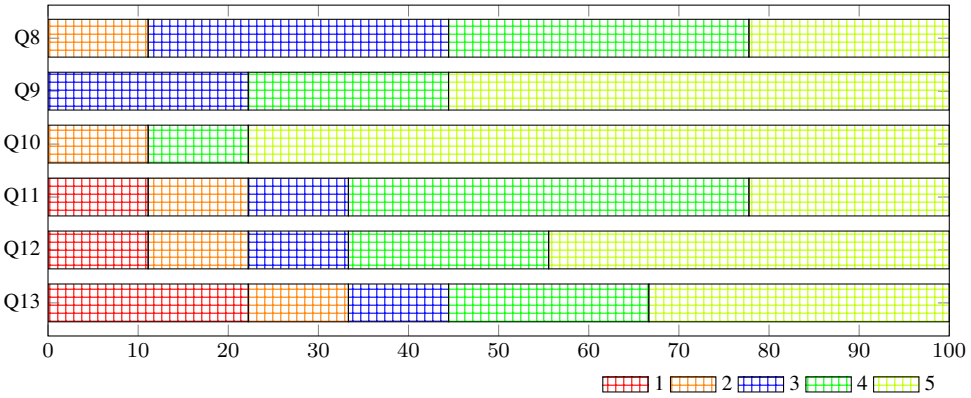

Figura 10. Respostas dos participantes para as questões Q8 a Q13

Cinco participantes consideraram que a **jogabilidade é intuitiva e fácil de aprender** (Q8). Isso sugere que o jogo está bem projetado em termos de usabilidade para a maioria dos jogadores. No entanto, três participantes deram uma resposta neutra, indicando que há espaço para melhorias, e um participante discordou, sugerindo que o jogador pode ter encontrado dificuldades significativas para aprender a jogar.

Quanto à **dificuldade do jogo** (Q9), sete participantes concordaram que o nível de dificuldade é adequado e dois foram neutros, demonstrando que no geral o jogo possui um bom nível de dificuldade. Ao analisar os *feedbacks* fornecidos na questão aberta (Q14), percebemos que existe a possibilidade de deixar o jogo mais desafiador.

Sete participantes acharam o **jogo divertido e envolvente** (Q10), dando a nota máxima (5) e um participante atribuiu a nota 4. Isso sugere que o jogo foi bem-sucedido em termos de entretenimento, conseguindo capturar e manter o interesse dos jogadores. Por outro lado, um participante atribuiu a nota 2, possivelmente devido à falta de familiaridade ou preferência pelo gênero do jogo.

Sobre o jogo ajudar a entender como **criar instâncias de classes** (Q11) em POO, seis participantes indicaram que o jogo os ajudou. Isso sugere que o jogo é eficaz em ensinar este conceito básico de POO para a maioria dos jogadores. Um participante atribui uma resposta neutra, o que pode indicar que, embora ele tenha encontrado algum valor educacional, o impacto do jogo poderia ser mais forte. E dois participantes indicaram que o jogo não os ajudou significativamente (notas 1 e 2). Isso destaca a necessidade de

aprimorar as partes do jogo que tratam da criação de instâncias de classes, possivelmente fornecendo mais exemplos práticos ou instruções mais claras.

Sobre o conceito de **herança** (Q12), seis participantes indicaram que o jogo facilitou sua compreensão sobre como funciona a herança entre classes em POO, sugerindo que o jogo é eficaz em ensinar este conceito para a maioria dos jogadores. Um participante forneceu uma resposta neutra e dois participantes indicaram que o jogo não facilitou significativamente sua compreensão sobre herança (notas 1 e 2).

Cinco participantes concordaram que o jogo os ajudou a entender como aplicar **polimorfismo** em POO (Q13). Isso sugere que o jogo é eficaz em ensinar o conceito de polimorfismo para a maioria dos jogadores. No entanto, um participante ficou neutro e três participantes discordaram, destacando a necessidade de aprimorar as partes do jogo que tratam do polimorfismo.

4.4. Espaço aberto

Sete participantes escreveram no espaço aberto para elogios, criticas, comentários e sugestão de melhorias. Baseado nas respostas, foi possível extrair alguns pontos de melhoria: **aprimorar tutoriais**, incluindo mais detalhes e interações para explicar os conceitos de POO; **ajustar dificuldade**, tornando os desafios mais equilibrados; **adicionar *feedbacks***, melhorando a compreensão dos jogadores; **expandir conteúdo**, adicionando mais fases e elementos ao jogo para aumentar o engajamento dos jogadores.

5. Considerações Finais

Este trabalho apresenta como produto o jogo **tOOwer Defenders**, jogo do gênero *Tower Defense* que foi desenvolvido utilizando processos de engenharia de software, desde sua concepção com levantamento de requisitos até a escrita do código fonte. O jogo possui código aberto e pode ser acessado no repositório.

A avaliação do jogo indicou que ele foi bem recebido pelos participantes em termos de entretenimento e diversão. A maioria dos participantes achou o jogo divertido e envolvente, destacando seu sucesso como uma ferramenta de entretenimento.

No que diz respeito ao auxílio na compreensão dos conceitos de POO, observou-se que a criação de instâncias de classes e o conceito de herança foram bem compreendida pela maioria dos participantes. No entanto, polimorfismo foi o conceito que recebeu as menores avaliações, indicando a necessidade clara de aprimoramento nas seções do jogo que tratam desse conceito. Adicionalmente, foi observado que os participantes com menor conhecimento prévio sobre os conceitos de POO tendem a dar notas mais baixas, o que sugere que o jogo poderá ser mais eficaz quando utilizado como ferramenta complementar, com o professor introduzindo os conceitos e utilizando o jogo para reforçar o aprendizado.

Para trabalhos futuros, recomenda-se aprimorar os tutoriais, ajustar a dificuldade e fornecer *feedbacks* mais claros. Além disso, sugere-se conduzir um estudo com um número maior de participantes em um ambiente de sala de aula. Isso permitiria uma análise quantitativa mais robusta, possibilitando a avaliação do nível de conhecimento dos participantes antes e depois do uso do jogo.

Referências

[Abbasi et al. 2017] Abbasi, S., Kazi, H., and Khowaja, K. (2017). A systematic review of learning object oriented programming through serious games and programming approaches. In *2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pages 1–6.

[Corral et al. 2014] Corral, J. M. R., Balcells, A. C., Estévez, A. M., Moreno, G. J., and Ramos, M. J. F. (2014). A game-based approach to the teaching of object-oriented programming languages. *Computers & Education*, 73:83–92.

[de Oliveira et al. 2017] de Oliveira, E. D., Camargo, M. C., Barbosa, C. R. S., Brancher, J. D., de Oliveira Barros, V. T., de S. Campos, V. V., and de Barros, R. M. (2017). The power of the game as a mediator tool paradigm of object oriented teaching-learning process. In *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE.

[de Souza Meirelles 1994] de Souza Meirelles, F. (1994). Evolução da microinformática: ciclos, cenários e tendências. *Revista de Administração de Empresas*, 34(3):62–80.

[Djelil et al. 2015] Djelil, F., Albouy-Kissi, B., Albouy-Kissi, A., Sanchez, E., and Lavest, J.-M. (2015). Towards a 3d virtual game for learning object-oriented programming fundamentals and c++ language. In *Proceedings of the 7th International Conference on Computer Supported Education - Volume 2*, CSEDU 2015, pages 289–294, Portugal. SCITEPRESS - Science and Technology Publications, Lda.

[Droid 2023] Droid, G. (2023). Understanding the history and evolution of tower defense games – tower defense games – main video game genres. Accessed: 2024-06-15.

[Edirisinghe 2008] Edirisinghe, E. M. N. S. (2008). Teaching students to identify common programming errors using a game. In *Proceedings of the 9th ACM SIGITE Conference on Information Technology Education*, SIGITE '08, pages 95–98, New York, NY, USA. ACM.

[Esper et al. 2014] Esper, S., Wood, S. R., Foster, S. R., Lerner, S., and Griwold, W. G. (2014). Codespells: How to design quests to teach java concepts. *J. Comput. Sci. Coll.*, 29(4):114–122.

[Galgouranas and Xinogalos 2018] Galgouranas, S. and Xinogalos, S. (2018). jAVANTGARDE: A cross-platform serious game for an introduction to programming with java. *Simulation & Gaming*, 49(6):751–767.

[Jiménez-Díaz et al. 2007] Jiménez-Díaz, G., Gómez-Albarrán, M., and González-Calero, P. A. (2007). Pass the ball: Game-based learning of software design. In *Entertainment Computing – ICEC 2007*, pages 49–54. Springer Berlin Heidelberg.

[Jordine et al. 2014] Jordine, T., Liang, Y., and Ihler, E. (2014). A mobile-device based serious gaming approach for teaching and learning java programming. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. IEEE.

[Karram 2021] Karram, O. (2021). The role of computer games in teaching object-oriented programming in high schools - code combat as a game approach. *WSEAS Transactions on Advances in Engineering Education*, 18:37–46.

[Kitchenham et al. 2007] Kitchenham, B., Charters, S., Budgen, D., Brereton, P., Turner, M., Linkman, S., Jørgensen, M., Mendes, E., and Visaggio, G. (2007). Guidelines for

performing systematic literature reviews in software engineering. *Empirical Software Engineering*, 12(2):131–164.

[Klump 2001] Klump, R. (2001). Understanding object-oriented programming concepts. In *2001 Power Engineering Society Summer Meeting. Conference Proceedings (Cat. No.01CH37262)*, volume 2, pages 1070–1074 vol.2.

[Rais et al. 2011] Rais, A. E., Sulaiman, S., and Syed-Mohamad, S. M. (2011). Game-based approach and its feasibility to support the learning of object-oriented concepts and programming. In *2011 Malaysian Conference in Software Engineering*. IEEE.

[Ramirez-Rosales et al. 2016] Ramirez-Rosales, S., Vazquez-Reyes, S., Villa-Cisneros, J. L., and de Leon-Sigg, M. (2016). A serious game to promote object oriented programming and software engineering basic concepts learning. In *2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT)*. IEEE.

[Schmolitzky and Gottel 2014] Schmolitzky, A. W. and Gottel, T. (2014). Guess my object: An 'objects first' game on objects' behavior and implementation with bluej. In *Proceedings of the 2014 Conference on Innovation 38; Technology in Computer Science Education*, ITiCSE '14, pages 219–224, New York, NY, USA. ACM.

[Sebesta 2009] Sebesta, R. W. (2009). *Concepts of Programming Languages (9th Edition)*. Pearson.

[Sharma et al. 2015] Sharma, S., Stigall, J., and Rajeev, S. (2015). Game-theme based instructional module for teaching object oriented programming. In *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE.

[Silva et al. 2013] Silva, M. A. G., Montenegro, A., Clua, E., Vasconcelos, C., and Lage, M. (2013). Real time pixel art remasterization on gpus. In *2013 XXVI Conference on Graphics, Patterns and Images*, pages 274–281.

[Stardock 2023] Stardock (2023). Siege of centauri dev journal: What makes a good tower defense game? Accessed: 2024-06-15.

[Studio 2022] Studio, R. (2022). Why pixel art games are still so popular. Accessed: 2023-05-15.

[Van Emden 1997] Van Emden, M. H. (1997). Object-oriented programming as the end of history in programming languages. In *1997 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, PACRIM. 10 Years Networking the Pacific Rim, 1987-1997*, volume 2, pages 981–984 vol.2.

[Videnovik et al. 2023] Videnovik, M., Vold, T., Kiønig, L., Madevska Bogdanova, A., and Trajkovik, V. (2023). Game-based learning in computer science education: a scoping literature review. *International Journal of STEM Education*, 10(1):54.

[Wang et al. 2022] Wang, L.-H., Chen, B., and Hwang, G.-J. (2022). Effects of digital game-based stem education on students' learning achievement: a meta-analysis. *International Journal of STEM Education*, 9(26).

[Wong et al. 2017] Wong, Y. S., Hayati, I. M., Yatim, M., and Hoe, T. W. (2017). A propriety game based learning mobile game to learn object-oriented programming — odyssey of phoenix. In *2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. IEEE.

[Wong et al. 2014] Wong, Y. S., Yatim, M. H. M., and Tan, W. H. (2014). Use computer game to learn object-oriented programming in computer science courses. In *2014 IEEE Global Engineering Education Conference (EDUCON)*. IEEE.

[Yan 2009] Yan, L. (2009). Teaching object-oriented programming with games. In *Proceedings of the 6th International Conference on Information Technology: New Generations*, pages 969–974.

[Zhang et al. 2013] Zhang, J., Caldwell, E. R., and Smith, E. (2013). Learning the concept of java inheritance in a game. In *Proceedings of CGAMES2013USA*. IEEE.