

Guilherme Adissy  
Rafael Barone

# **Estruturação de processo de criação de jogos independentes**

Brasil

2020



Guilherme Adissy  
Rafael Barone

## **Estruturação de processo de criação de jogos independentes**

Criação de uma estrutura bem definida centrada na criação de jogos digitais independentes e posterior aplicação da mesma no desenvolvimento de um jogo

Universidade de São Paulo

Escola Politécnica

Graduação

Orientador: Ricardo Nakamura

Brasil

2020

---

Guilherme Adissy

Rafael Barone

Estruturação de processo de criação de jogos independentes/ Guilherme Adissy

Rafael Barone. – Brasil, 2020-

66 p. : il. (algumas color.) ; 30 cm.

Orientador: Ricardo Nakamura

Trabalho de Conclusão de Curso – Universidade de São Paulo

Escola Politécnica

Graduação, 2020.

1. Jogos digitais. 2. Metodologia. I. Ricardo Nakamura. II. Universidade de São Paulo. III. Escola Politécnica. IV. Estruturação de processo de criação de jogos independentes

CDU 02:141:005.7

---

Guilherme Adissy  
Rafael Barone

## **Estruturação de processo de criação de jogos independentes**

Criação de uma estrutura bem definida centrada na criação de jogos digitais independentes e posterior aplicação da mesma no desenvolvimento de um jogo

Trabalho aprovado. Brasil, 7 de dezembro de 2020:

---

**Ricardo Nakamura**  
Orientador

Brasil  
2020



*It's dangerous to go alone! Take this.*





# Agradecimentos

A frase tornada célebre por Isaac Newton, "*Se eu vi mais longe, foi por estar sobre ombros de gigantes*" (em tradução livre), expressa muito do meu sentimento acerca do que foi conquistado ao longo dos últimos anos e, enfim, deste trabalho de conclusão de curso. Mas se Newton se refere a outros cientistas que vieram antes dele, nisso somos diferentes. Alguns dos gigantes a que aqui me refiro são, de fato, acadêmicos: os professores que me instruíram ao longo dos anos na Escola Politécnica, sobretudo o Professor Ricardo Nakamura, orientador deste trabalho. Mas os gigantes não se resumem a acadêmicos e incluem, acima de tudo, meus pais, cuja insistência em me prover com uma excelente educação sempre foi um gigantesco privilégio e, acima de tudo, me trouxe até aqui.

A esses gigantes, o meu muito obrigado.

*Rafael Barone*



# Resumo

O projeto visa estruturar um processo ou metodologia usado para o desenvolvimento de jogos eletrônicos independentes, esse processo deve auxiliar agilizando a parte de desenvolvimento do jogo e ao mesmo tempo mantendo ou aumentando a qualidade desses. Ao final, a metodologia estruturada será aplicada ao desenvolvimento de um jogo.

**Palavras-chaves:** Jogos-Eletrônicos, Jogos-Independentes



# Abstract

The project aims to structure a process or methodology used for the development of indie electronic games, this process should help speed up the development of the game and at the same time maintain or increase it's quality. At the end, the methodology will be applied to the development of a game.

**Key-words:** Electronic-Games, Independent-Games



# Lista de ilustrações

Figura 1 – Fluxo da metodologia desenvolvida . . . . .	32
Figura 2 – Frame do jogo Mega Man Battle Network . . . . .	35
Figura 3 – Frame do jogo Into The Breach . . . . .	36
Figura 4 – Primeira fase do protótipo testado . . . . .	37
Figura 5 – Segunda fase do protótipo testado . . . . .	38
Figura 6 – Terceira fase do protótipo testado . . . . .	38
Figura 7 – Terceira fase do protótipo testado . . . . .	39
Figura 8 – Reprodução do editor básico de fases construído . . . . .	43
Figura 9 – Diagrama simplificado dos estados de uma batalha . . . . .	45
Figura 10 – Ilustração do diálogo em sua forma simplificada . . . . .	46





# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>17</b>
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA</b>	<b>19</b>
2.1	Uma breve história sobre o mercado de video games	19
2.2	Jogos independentes	20
2.3	Problemas do segmento independente	20
<b>3</b>	<b>O JOGO DESENVOLVIDO</b>	<b>23</b>
3.1	Problemas abordados no trabalho	23
3.2	Tema	23
3.2.1	O tema escolhido	23
3.2.2	Por quê esse tema?	24
<b>4</b>	<b>METODOLOGIA DE DESENVOLVIMENTO</b>	<b>25</b>
4.1	Divisão em etapas	25
4.2	Core	25
4.3	Construção	27
4.4	Polimento	29
4.5	Ciclo de desenvolvimento	30
4.6	Resumo	32
<b>5</b>	<b>APLICAÇÃO DA METODOLOGIA</b>	<b>33</b>
5.1	Primeiro ciclo de desenvolvimento do Core	33
5.2	Segundo ciclo de desenvolvimento do Core	35
5.2.1	Core	35
5.2.1.1	Desenvolvimento	36
5.2.1.2	Testes	39
5.2.1.2.1	Primeiro teste	40
5.2.1.2.2	Segundo teste	40
5.2.1.2.3	Terceiro teste	41
5.2.1.3	Validação	41
5.2.2	Construção	41
5.2.2.1	Desenvolvimento	42
5.2.2.1.1	Gerador de <i>Boards</i>	42
5.2.2.1.2	Máquina de Estados	43
5.2.2.1.3	Sistema de Diálogos	44

5.2.2.1.4	Painel de Estados dos Personagens . . . . .	46
5.2.2.1.5	Habilidades . . . . .	47
5.2.2.1.6	Fábrica de Unidades . . . . .	48
5.2.2.1.7	A.I. . . . .	48
5.2.2.2	Testes . . . . .	49
5.2.2.3	Validação . . . . .	49
5.2.3	Polimento . . . . .	50
5.2.3.1	Desenvolvimento . . . . .	50
5.2.3.1.1	Enredo . . . . .	50
5.2.3.2	Finalização . . . . .	51
<b>6</b>	<b>CONCLUSÃO . . . . .</b>	<b>53</b>
<b>6.1</b>	<b>Sobre a aplicação da metodologia . . . . .</b>	<b>53</b>
<b>6.2</b>	<b>Metodologia revisada . . . . .</b>	<b>54</b>
<b>6.3</b>	<b>Próximos passos . . . . .</b>	<b>55</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>57</b>
	<b>APÊNDICES . . . . .</b>	<b>59</b>
	<b>APÊNDICE A – TEXTO DO TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO . . . . .</b>	<b>61</b>
	<b>APÊNDICE B – ROTEIRO DE ENTREVISTAS APÓS TESTE COM PROTÓTIPO . . . . .</b>	<b>63</b>
	<b>APÊNDICE C – TRANSCRIÇÃO DOS DIÁLOGOS DESENVOLVIDOS PARA O JOGO . . . . .</b>	<b>65</b>

# 1 Introdução

Criados na década de 1950, Videogames tornaram-se, a partir da década 1970, uma forma de expressão cultural de grande prevalência. Desde então, os jogos evoluíram de maneira marcante, bem como o desenvolvimento dos mesmos.

Na atualidade, uma das maneiras mais comuns de desenvolvimento de jogos trata dos chamados jogos independentes, usualmente jogos com conceitos inovadores desenvolvidos por pequenas equipes de programadores, designers, entre outros. De acordo com [Reichert \(2011\)](#), essa modalidade de desenvolvimento de jogos eletrônicos encontra diversos problemas ao longo de seu processo.

Apesar deles existirem há tanto tempo quanto outros segmentos do mercado, a sua popularização é mais tardia. Jogos independentes se tornaram mais rentáveis a partir de 2008, com a criação de serviços como XBox Live e Steam, que facilitaram a distribuição digital de jogos, permitindo que desenvolvedoras pudessem publicar seus jogos sem precisar de uma publicadora. Outro fator favorável foi a popularização de game engines como o Unity 3D. A partir desse período começaram a surgir muitas desenvolvedoras indie.

Apesar de ouvirmos falar bastante dos jogos que fizeram sucesso, a maioria deles não tem tanta sorte (especialmente quando falamos de jogos independentes). Devido a sua popularização tardia, os problemas desse segmento também surgiram depois e muitos deles ainda não foram resolvidos.

Considerando a popularidade dos jogos digitais e, sobretudo, dos jogos independentes, bem como os problemas envolvidos em seu desenvolvimento, este trabalho tem como objetivo estruturar uma metodologia para desenvolvimento de jogos eletrônicos independentes de maneira que agilize o desenvolvimento de jogos mantendo ou aumentando a qualidade desses, assim como atacando alguns dos problemas usualmente encontrados ao longo desse processo, dentre os quais se destacam problemas relacionados a definição de escopo do jogo e determinação de público alvo. Para se atingir esse objetivo, será definido um processo de projeto, partindo de referências da literatura sobre o processo de desenvolvimento de jogos em estúdios independentes. Na sequência, um jogo será desenvolvido seguindo esse processo, de modo a validá-lo. Os resultados serão avaliados criticamente, refinando assim o processo definido inicialmente.



## 2 Fundamentação Teórica

### 2.1 Uma breve história sobre o mercado de video games

Video games surgiram na década de 1950 e, naquela época, tinham um propósito de mostrar as capacidades de novas tecnologias. Assim, alguns acadêmicos também testavam formas rudimentares de inteligência artificial em jogos como xadrez e jogo da velha. Entre a década de 1950 e 1960, há relatos de jogos desenvolvidos por estudantes que tiveram acesso aos mainframes de suas universidades; um exemplo é o jogo Spacewar!, de acordo com [Graetz \(1981\)](#).

De acordo com [Video... \(2020\)](#) e [History... \(2020\)](#), na década de 1970 começaram a surgir os primeiros jogos comerciais. Entre eles pode-se citar Computer Space e Pong, que eram disponibilizados em bares e restaurantes. Nessa década também começaram a surgir os primeiros consoles caseiros de video games como The Magnavox Odyssey em 1972 e o famoso Atari 2600 em 1977. Em 1978 ocorre a criação de outro jogo icônico, o Space Invaders, que ajudou a popularizar os arcade games (máquina de jogo eletrônico instalada em estabelecimentos para entretenimento). Entre 1978 e 1982 ocorreu a chamada “Era de ouro dos arcades”, período em que estes se popularizaram (principalmente nos EUA), com vendas de arcade games chegando a 900 milhões de dólares em 1981, sendo que nessa época a indústria de jogos arcade já gerava uma receita anual de 3 bilhões de dólares nos EUA.

Em 1983 ocorre a crise dos jogos eletrônicos, quando o mercado de jogos sofreu uma recessão devido a fatores como a baixa qualidade de jogos, saturação do mercado e a ascensão do computador pessoal. Em 1985 a indústria foi revitalizada por empresas como a Nintendo com o lançamento do console Nintendo Entertainment System (NES). Em 1989 aconteceu a popularização dos consoles portáteis com o Game Boy da Nintendo, mesma época em que começaram a se popularizar jogos de PC (Personal Computer).

Os consoles mais conhecidos de hoje em dia surgiram a partir da década de 1990:, em 1994 foi o lançamento do Playstation da Sony e em 2001 o lançamento do XBox da Microsoft. Essas empresas (junto da Nintendo) estão no mercado de consoles até hoje, com lançamento de uma nova geração de consoles neste ano (2020). Esse mercado tem crescido e chegou a 47,9 bilhões de dólares em 2019.

Os jogos para celulares surgiram em 1997 quando a Nokia decidiu instalar Snake na sua linha de celulares. Porém, jogos só começaram a ser comercializados em celulares em 2000 através de operadoras e outros portais. Os jogos para celulares só começaram a se popularizar em 2008 com o lançamento da App Store da Apple para os sistemas iOS.

Desde então esse mercado vem crescendo a cada ano, chegando a 68,5 bilhões de dólares em 2019 (somando smartphones e tablets). Atualmente, é o segmento do mercado de jogos com a maior parcela (45%).

De acordo com [NEWZOO \(2019\)](#), em 2019 o mercado de jogos eletrônicos gerou 152,1 bilhões de dólares de receita e tem projeção de 196 bilhões de dólares para 2022. É um mercado muito grande e diversificado que cresce a cada ano.

## 2.2 Jogos independentes

Uma das segmentações do mercado de jogo são jogos independentes. São desenvolvidos por apenas uma pessoa ou uma equipe pequena, normalmente sem apoio financeiro e com pouca ou nenhuma verba disponível. Jogos desse tipo costumam se destacar por sua inovação, criatividade e experimentação artística.

Não existe uma definição exata do que é considerado indie, mas para fins desse trabalho definiremos indie como jogos feitos com equipes pequenas (até 20 pessoas) e com pouco ou nenhum financiamento.

Nos últimos anos o segmento de mercado de jogos indie vem crescendo bastante, foi estimado em 2016 que a receita desse segmento é de pelo menos 1 bilhão de dólares, com base em apenas jogos vendidos na plataforma Steam.

## 2.3 Problemas do segmento independente

De acordo com [Wawro \(2016\)](#), quando se trata de desenvolvedoras pequenas a continuação da empresa pode depender do sucesso de um jogo. Não é raro o caso em que uma empresa acaba falindo porque um jogo não vendeu o esperado.

A grande maioria dos problemas de jogos independentes se originam justamente do que os define: times pequenos com pouco dinheiro e, em alguns casos, pouca experiência. Alguns exemplos são:

- a) **Escopo:** devido as limitações, cada funcionalidade que entra no produto final deve ser escolhida cuidadosamente. Muitas features prolongam o desenvolvimento, adiando o lançamento e aumentando o custo do projeto, elevando o risco de seu cancelamento. Por outro lado poucas features podem afetar negativamente a jogabilidade e o conseqüentemente o sucesso do jogo.
- b) **Marketing:** algumas produtoras gastam milhões para divulgar certos jogos. Como competir com essa divulgação? Como competir com jogos com muito mais features e muito mais polidos que o seu?

- c) **Design:** o fato de os desenvolvedores serem obrigados a escolher algumas poucas features para fazerem parte do jogo e de muitas vezes os jogos tentarem se destacar pela sua originalidade cria diversos problemas interessantes de design.
- d) **Processo:** como gerenciar um time pequeno a ser produtivo e engajado durante todo o desenvolvimento? Diferente de grandes desenvolvedoras, não é possível se dar o luxo de ficar constantemente mudando os membros da equipe, sendo que muitas vezes os membros do time estão vivendo de reservas de dinheiro, na esperança de que o jogo faça sucesso para terem um retorno.
- e) **Destaque no mercado:** como fazer o jogo se destacar em meio a tantos outros? Como provar que o seu jogo tem público, que ele faz sentido para uma certa audiência? Como provar que o seu jogo é melhor ou diferente de outros semelhantes?
- f) **Público:** como atingir o público desejado? Como garantir que esse público vai se interessar pelo jogo e ficará engajado ao jogá-lo?

É relevante ainda o fato de que, tendo pouco dinheiro para se manter, as empresas responsáveis pelo desenvolvimento de jogos independentes são impedidas de tomar ações comumente tomadas por grandes empresas do mercado de jogos, como adiar a data de lançamento de jogos (o que tem grandes implicações no marketing do jogo) se necessário, ou contratar mais pessoas para terminar o jogo a tempo.





## 3 O jogo desenvolvido

### 3.1 Problemas abordados no trabalho

Como esse trabalho visa a estruturação de um processo, nem todos os problemas listados na [seção 2.3](#) serão abordados. O intuito é que o processo ajude a diminuir o tempo de desenvolvimento, mas mantendo o jogo com certas características que acreditamos serem importantes para o seu sucesso.

Dessa forma, esse trabalho será focado no processo que será usado durante todo o desenvolvimento de um jogo. Procurando resolver problemas como: escopo, destaque no mercado.

Temas como marketing, praticamente não serão abordados neste trabalho, por fugirem muito do escopo. Enquanto temas como design não são o foco, eles aparecerão quando discutirmos diversos outros temas. Além disso, existem várias maneiras de se vender jogos, este trabalho será focado em jogos vendidos diretamente para o cliente final (B2C), ou seja, o jogo será vendido para as massas. Logo, deve ser atraente para um público amplo. Como o mercado é muito amplo, é possível encontrar diversas estratégias encontradas por desenvolvedores indie para vender seus jogos.

Por exemplo, de acordo com [Chalk \(2019\)](#), a Spiderweb Software (desenvolvedora indie fundada em 1994) encontrou o seu nicho, criaram um público fiel que gosta muito de seus jogos e apesar de não ser um público tão amplo, é o suficiente para manter a empresa.

Um outro exemplo, de acordo com [Klei... \(2020\)](#), é a Klei Entertainment (desenvolvedora indie fundada em 2005), que desenvolveu diversos jogos indie de sucesso que são bem diferentes entre si, como Don't Starve, Oxygen Not Included, Mark of the Ninja.

As estratégias abordadas aqui seguem uma linha de pensamento mais próxima do segundo exemplo (Klei Entertainment): desenvolver algo único que tenha uma ampla audiência.

### 3.2 Tema

#### 3.2.1 O tema escolhido

Como ao final do trabalho será desenvolvido um jogo, foi escolhido um tema para guiar e facilitar esse desenvolvimento. Assim, ele pode ser usado como um caminho a seguir.

Foi decidido que o tema escolhido será alguma causa social, ou seja, o jogo deve abordar de alguma forma (não necessariamente precisa ser tema central) alguma causa social.

Por familiaridade e facilidade de acesso a informações e pessoas que possam ajudar, será escolhida uma causa social como depressão, machismo ou homofobia. É importante nesse caso ter pessoas que dominam esse assunto ajudando de alguma forma no desenvolvimento, seja testando, discutindo ideias ou até fazendo parte do time. Ao abordar temas sensíveis como esses é importante validar as ideias que envolvem o tema como pessoas que dominam o assunto. Não fazer isso pode fazer com que a causa social que se tentava ajudar seja na verdade prejudicada.

### 3.2.2 Por quê esse tema?

Normalmente jogos que abordam temas com causas sociais são indie ou são de empresas que nasceram como indie e eventualmente tiveram sucesso e cresceram. Como o tema costuma ser abordado por empresas como as que estaremos estudando ao longo deste trabalho, faz sentido a escolha do tema. Além disso, é sempre bom ajudar causas como essa, para melhorar o ambiente em que vivemos, mesmo que o jogo seja só um pontapé inicial para que um pessoa se intere mais sobre algum tema e crie consciência sobre.

## 4 Metodologia de desenvolvimento

### 4.1 Divisão em etapas

Todo o jogo surge de uma ideia inicial, um conceito, etc. A partir daí, é preciso construir em cima disso, iterar e validar as ideias. Para isso, o desenvolvimento será quebrado em 3 etapas: core, construção, polimento.

### 4.2 Core

Nessa etapa inicial tudo ainda está muito cru, por isso o jogo ainda pode se moldar de diferentes formas, dependendo do que for ocorrendo durante os ciclos e das validações que forem feitas. O mais importante dessa etapa é validar a ideia principal do jogo: o core loop. De acordo com [Crafting... \(2020\)](#), o core loop é um conceito de game design que é basicamente o que o jogador irá fazer a maior parte do tempo. Ele pode ser quebrado em 3 partes: ação, recompensa e progresso.

- a) **Ação:** o jogador faz alguma ação. Por exemplo em um RPG isso pode significar matar um monstro.
- b) **Recompensa:** o jogador recebe uma recompensa por executar a ação. Seguindo exemplo anterior, ao matar o monstro o jogador pode receber ouro (uma moeda do próprio jogo)
- c) **Progresso:** o jogador usa a recompensa para fazer algum tipo de progresso. Seguindo a linha de pensamento anterior, ele pode usar o ouro para comprar equipamentos melhores para seu personagem.

Ao final disso, o ciclo se repete. Então, após comprar um equipamento melhor o jogador será capaz de enfrentar monstros mais fortes. Esses por sua vez irão dar mais ouro, o que permitirá que ele compre equipamentos ainda melhores e assim o ciclo continua.

Essa é uma explicação bem básica e simplificada do conceito, existem jogos que usam mais de um gameplay loop: um curto que ocorre durante as sessões que a pessoa está jogando (core loop) e um segundo maior que ocorre ao longo de múltiplas sessões. Esse segundo loop entra como uma ferramenta para fazer com que o jogador volte a jogar após a 1ª sessão. Nessa etapa iremos focar apenas no loop mais curto, que irá ocorrer múltiplas vezes durante uma sessão.

Iremos fazer isso por 2 motivos. O primeiro é devido às limitações de jogos indie que já foram citadas anteriormente, não se pode dar o luxo de focar em coisas grandes

ou em mais de uma coisa, é preciso garantir que essa parte que é a base do jogo esteja funcionando e validada, caso contrário o jogo nunca será terminado. O segundo motivo é que isso é o que o jogador fará a maior parte do tempo que ele está jogando, se essa parte central do jogo não for divertida e engajante, o jogador não irá voltar para uma sessão - talvez ele nem termine a primeira.

Nessa linha de focar em apenas uma coisa por vez, temos o exemplo de como a Klei Entertainment (desenvolvedora de títulos como *Don't Starve*, *Crypt of the Necrodancer*, *Oxygen not Included*, entre outros) desenvolve seus jogos. Eles tem uma linha de pensamento de design que foi explicada em um talk da GDC de 2016 (Game Developers Conference) chamado: *Be Spiky: A Decade of New Ideas* (Seja espetado: uma década de novas ideias - em tradução livre).

O conceito é bem simples e ele parte do pressuposto de que os desenvolvedores querem fazer algum jogo original, que se destaque do resto. Para que o jogo seja único e seja interessante é necessário focar em apenas uma experiência. O seu jogo deve ter como foco essa experiência de maneira que o resto pode (e dado o nosso escopo, até deve) ser medíocre. Seguindo até uma linha de pensamento em que adicionar mais features ao jogo, não irá torná-lo melhor necessariamente. Mas a pergunta que não quer calar é: como desenvolver um bom core loop?

Para fazer isso, será necessário definir quais vão ser as 3 partes do seu core loop: ação, recompensa e progresso. Se seguirmos a linha de pensamento citada anteriormente, o core loop sairá da experiência que queremos trazer para o jogador. A partir dela é possível extrair as 3 partes e iterar em cima disso, até que se consiga validar a ideia ou que o feedback recebido de um guia de para onde pivotar.

A primeira ideia, especialmente, não precisa ser muito polida. O mais importante é prototipar e validar essa ideia o quanto antes. Será explicado mais à frente neste trabalho como funcionará esse ciclo que envolve prototipação e validação.

É crucial que essa etapa seja curta. Se ela se alongar por diversos meses, isso pode custar caro (em questão financeira ou no na quantidade de features que serão entregues no produto final). Portanto, é muito importante prototipar rápido e tentar validar a ideia. Caso ela se mostre uma boa ideia, é possível seguir para a próxima etapa. Caso contrário, é preciso coletar o feedback de porquê essa ideia não é boa e reestruturá-la ou pivotar. Nesse caso, pivotar seria basicamente jogar fora o core loop que estávamos prototipando e fazer um novo.

Quando se é levado em consideração que toda vez que se pivota parte do trabalho é descartado (não é possível usar aquele protótipo ou parte dele para outras coisas) percebemos a importância de que esse ciclo de prototipação e validação seja rápida, caso contrário não será possível prosseguir com o desenvolvimento. Vale ressaltar que pivotar

não é algo ruim, pelo contrário, é muito melhor pivotar do que seguir com algo sem validação. Seguir com algo sem validação pode significar grandes problemas no futuro. Por exemplo, caso o core loop não tenha sido validado e ao final do projeto se percebe que ele não faz sentido ou que algo nele não é uma experiência agradável para o usuário, o que fazer? Pivotar no final do projeto, ou seja, jogar boa parte do trabalho feito fora? Lançar o jogo com problemas que atrapalham a experiência do jogador? Seguir com esses problemas irá custar no futuro tempo e/ou dinheiro, dois dos únicos escassos recursos que se têm.

A validação do core loop pode ser feita de diversas maneiras. No fundo, é importante saber se os usuários estão gostando do jogo, se eles se divertem jogando e se eles se sentem engajados enquanto jogam. Uma sugestão de como fazer isso é com 3 simples perguntas que podem ser feitas após o teste:

- a) Foi divertido jogar o protótipo?
- b) Caso o protótipo fosse mais longo, você teria jogado mais?
- c) Você teria interesse em jogar um jogo mais completo construído a partir desse protótipo?

Essas 3 perguntas buscam responder se o protótipo tem as características anteriormente citadas que um bom jogo tem. Caso a resposta dessas 3 perguntas sejam afirmativas, pode-se considerar o protótipo como validado. Importante ressaltar que essa não é a única maneira de validar, é apenas uma sugestão. Quanto se tiver um core loop validado, é possível seguir para a próxima etapa:

## 4.3 Construção

Essa é a etapa mais longa do desenvolvimento e também é a que pode mais facilmente se prolongar. Durante essa etapa serão desenvolvidas várias features para o jogo e qualquer pessoa que já trabalhou em um projeto de desenvolvimento de software sabe que sempre é possível colocar mais features no projeto. Por isso, é fundamental definir quais features devem entrar e quais devem ficar de fora. As features serão divididas em dois grupos:

- a) **Features cruciais:** são aquelas necessárias para que o usuário tenha uma boa experiência e que provavelmente ficaram de fora do protótipo inicial devido a sua simplicidade. Por exemplo: um menu, uma interface de usuário intuitiva, animações, etc
- b) **Features complementares:** são features que complementam a experiência do usuário. Que ajudam a mantê-lo engajado por mais tempo e que dão mais tempo de jogo. Por exemplo: mais fases, novos poderes para o personagem principal, história, etc.

É importante se notar que tudo que será construído a partir desse ponto deve ser feito em cima do protótipo que foi validado e tudo deve complementar o seu core loop. Caso algo fuja disso, é importante pensar se realmente faz sentido que isso seja parte do jogo.

Como citado anteriormente, essa etapa pode durar tanto quanto os desenvolvedores quiserem. Na maioria das vezes o que vai definir quanto tempo essa etapa vai durar é o tempo e/ou o dinheiro que a equipe tem, sempre lembrando que ainda existe uma última etapa depois dessa. Tendo uma lista de features priorizadas e com ciclos curtos de desenvolvimento, não deve ser muito problemático passar dessa etapa para o última depois que a maioria das features cruciais e uma parte das features complementares foram implementadas. E se lembrar de sempre validar as ideias e features que forem implementadas.

Como definir quais features complementares o jogo precisa? Enquanto na etapa anterior foi pensado muito na experiência do jogador em um única sessão e coisas para mantê-lo engajado por alguns minutos, talvez algumas horas, nessa etapa será pensado em como mantê-lo engajado por horas e dias (talvez até semanas, meses ou anos, o céu é o limite! Por céu, entenda-se dinheiro e tempo do projeto). Outra maneira de pensar nesse engajamento é como fazer para que o jogador jogue mais de uma sessão?

Como foi citado anteriormente, muitos jogos tem um segundo gameplay loop que ajuda a manter o jogador engajado por mais tempo. Para isso, o jogo pode ter algo que deixa o usuário empolgado para ganhar quando ele voltar na próxima sessão: uma mecânica nova que ele possa explorar, um trecho da história do jogo que ele ainda não descobriu, etc. E isso ainda pode ser agrupado de maneira a incentivar o jogador a voltar.

Por exemplo, nos diversos jogos da clássica franquia da Nintendo, Super Mario Bros, o core loop seria uma fase do jogo. Essas fases são agrupadas em mundos, de maneira que a cada mundo novo, as fases têm mecânicas e temáticas novas, além de uma certa progressão ao longo desses mundos: com a primeira fase sendo mais fácil mostrando como funciona a mecânica nova; as fases seguintes explorando cada vez mais essa mecânica nova e misturando ela com outras mecânicas antigas, tudo isso junto a um aumento de dificuldade; culminando na última fase mundo (um castelo), que tem um inimigo mais forte ao final (comumente conhecido como chefe). Ao final disso, ainda temos um elemento de história que ajuda nesse engajamento: a promessa de que “a princesa está em outro castelo”.

É possível ver como esse segundo gameplay loop complementa o core loop para manter o jogador engajado durante as sessões. A cada mundo novo há a introdução e desenvolvimento de uma nova mecânica no jogo, o que ajuda a tornar o jogo “novo” de certa maneira. Modificando um pouco a experiência do jogador durante o core loop (a fase), mas sem tirar a essência dessa experiência, apenas complementando ela. A progressão de

dificuldade dá sempre um desafio novo que o jogador deve superar. E ao final do mundo temos a promessa de que um novo mundo nos espera, com novos desafios e que talvez lá você consiga cumprir o seu objetivo final: resgatar a princesa.

Os exemplos citados de como aumentar o engajamento são mais sugestões e ideias de como fazê-lo, mas é possível encontrar diversos outros exemplos disso no mercado de jogos. Nessa etapa também é muito importante que se possa prototipar muito rápido (especialmente agora que já foi definido um formato e uma base do que se pretende fazer). Por exemplo, se for necessário fazer mais 50 fases para que se chegue um tamanho bom para o jogo, mas prototipar uma fase demora cerca de uma semana, fica inviável seguir com o projeto. Será gasto quase um ano, apenas para prototipar todas as fases (assumindo que todos os protótipos sejam validados na primeira tentativa, o que é muito improvável), esse período é muito longo para ser gasto nisso, além de que existem outras features que precisam ser implementadas. Nesse caso, o ideal seria que prototipar uma fase demorasse em torno de 1 a 2 horas.

Para endereçar esse problema, é preciso que sejam feitas ferramentas que permitam essa prototipação rápida. Iterar e testar em ciclos curtos ajuda muito na vazão do projeto e pode ser a diferença entre entregar algo com grande parte das features esperadas ou não conseguir nem ao menos terminar o produto.

Quanto se tiver um número considerável/desejável de features ou o tempo/dinheiro alocado para o projeto estiver chegando ao fim, é possível prosseguir para a última etapa.

## 4.4 Polimento

Essa é a última etapa de desenvolvimento e ao final dela o jogo deve ser lançado, ou ao menos devem começar a serem feitas as preparações para que isso ocorra. Durante essa etapa deve ser feito (caso ainda não exista) um planejamento de divulgação e publicação do jogo. Um fato muito importante dessa etapa é que durante ela, não deve ser implementada nenhuma feature nova.

Esse período deve ser focado em corrigir bugs e adereçar coisas que atrapalhem a experiência do usuário. Isso não quer dizer que essas coisas não devam ser atacadas na etapa anterior, apenas que ter um tempo dedicado a isso é importante. É importante ter uma lista com os bugs e problemas (se possível com sugestões de melhorias) do jogo para que ela seja priorizada. Essa lista deve vir principalmente dos testes feitos até então.

Essa lista será priorizada com base no ROI (Return Over Investment) de cada uma dessas tarefas, tarefas com um ROI maior devem ter maior prioridade. O ROI é algo completamente abstrato e servirá apenas como um guia. Essa etapa chega ao final quando nessa lista tiverem apenas coisas que o ROI não é tão alto. Nesse momento o jogo estará

pronto para o lançamento.

Um ponto de atenção aqui, é que caso exista uma tarefa nessa lista que exija um esforço muito grande e que seja considerada importante ser desenvolvida antes do lançamento do jogo, muito provavelmente houve alguma falha em etapas anteriores, falta de validação de algo, priorização errada de features, etc. E aqui deve-se ressaltar mais uma vez a importância de se fazer testes e validações constantes durante todo o desenvolvimento. Para que não ocorra uma situação como a citada agora, que causará um atraso no cronograma ou um produto pior, em ambos os casos, isso deve custar dinheiro que poderia ser economizado ou ganho.

## 4.5 Ciclo de desenvolvimento

Após estudar as principais dificuldades de desenvolvedoras independentes, pode-se perceber que, apesar de fazerem parte de contextos diferentes, essas dificuldades se assemelham muito ao de startups: problemas de escopo, dificuldades para entender os seus clientes, pouco dinheiro, pouco tempo, etc. Porém, existe uma diferença crucial entre essas duas empresas: startups são baseadas em resolver um problema, uma carência, uma deficiência do mercado. Ou, de acordo com [Ries \(2011\)](#): "O objetivo de uma startup é descobrir a coisa certa a criar – a coisa que os clientes querem e pela qual pagarão – o mais rápido possível."

Enquanto startups podem até ter dificuldade para definir qual é o problema que estão tentando resolver, esse problema existe. Uma desenvolvedora indie não supre uma necessidade do mercado, nem tenta resolver um problema que nenhum outro resolve. Por isso, é difícil definir o que seria um produto (no caso, um jogo) de sucesso antes de ele chegar no mercado. E sem essa definição, como saber se o jogo já é bom o suficiente para ser lançado?

Para isso, pode-se fazer um paralelo com startups e definir que o objetivo de uma desenvolvedora é entreter pessoas. E como medida de sucesso iremos tomar o quão divertido e engajante é um jogo. Com isso estamos presumindo que, se um jogo é divertido e engajante ele entretém as pessoas. Como medir se um jogo é divertido? Para medirmos se algo que foi feito é divertido, basta fazer com que pessoas testem o seu jogo e perguntá-las se elas se divertiram. Enquanto o engajamento pode ser facilmente medido com base no tempo gasto antes da pessoa para de jogar.

Obviamente isso são apenas guias para se ter uma noção de como está indo o desenvolvimento do jogo. Uma maneira de medir de alguma forma como está indo e de validar as ideias desenvolvidas. É muito importante fazer não só validações quantitativas (porcentagem das pessoas se divertiram), mas também qualitativas: tentar entender o que os usuários gostaram e também o que não gostaram. Assim é possível perceber com maior



precisão quais elementos estão funcionando e se a interação dos elementos do jogo está boa ou ruim.

Essa validação é importante porque ela faz parte do ciclo de desenvolvimento que será usado durante as duas primeiras etapas da produção do produto, como citado anteriormente. Esse ciclo de desenvolvimento é baseado no modelo de desenvolvimento de cliente descrito por Blank (2012). Esse ciclo iterativo será composto de 3 partes: hipótese, prototipação e validação.

- a) **Hipótese:** nessa etapa será elaborada uma hipótese. Por exemplo a hipótese pode ser o core loop.
- b) **Prototipação:** nessa etapa a hipótese elaborada anteriormente será prototipada. Por exemplo, será desenvolvido um joguinho bem simples (não precisa nem ao mesmo ter menus) apenas com o core loop proposto.
- c) **Validação:** aqui será validada a hipótese elaborada usando o protótipo desenvolvido. A melhor maneira de fazer isso é dando o protótipo para jogadores testarem e avaliar sua resposta ao mesmo.

Após a validação, podem ocorrer duas coisas: a hipótese se provou certa ou errada. Caso ela esteja errada, será necessário pivotar, ou seja, elaborar uma nova hipótese e recomençar o ciclo. É importante se notar que a cada iteração deve ser mais fácil elaborar a hipótese, de maneira que a cada vez se está mais perto de provar uma hipótese verdadeira. A maneira de fazer isso é coletando feedback daqueles que testaram o protótipo para entender melhor qual parte da hipótese foi o problema, o que não teve o resultado esperado e o que mudar para se obter o resultado desejado. No outro caso, se a hipótese se provou certa e a ideia foi validada, o próximo passo pode ocorrer de duas formas. Caso a etapa atual seja a de desenvolvimento do *Core*, avança-se para a etapa seguinte, isto é, a de Construção; caso a etapa atual seja a de Construção, avança-se para uma 4a e última etapa: execução. Nela, deve-se construir o que foi prototipado anteriormente, polir e transformar o protótipo em produto (no caso, em uma das features do jogo).

Essas 4 etapas estarão presentes ao longo de todo o projeto e a velocidade de iteração desse ciclo irá variar de caso a caso. No início, especialmente durante a etapa Core do desenvolvimento, é importante que esse ciclo seja curto, com protótipos com uma fidelidade menor, com o intuito de validar o mais rápido possível alguma hipótese (um core loop). Enquanto durante a etapa Construção, caso tenhamos alguma hipótese que foi baseada em diversos feedbacks que foram recebidos ao longo do desenvolvimento, podemos ter um protótipo com uma fidelidade maior (sempre tomando cuidado para não exagerar, lembrando que isso é apenas um protótipo). Vale ressaltar também que pode ser interessante que, na etapa de construção, seja feito um primeiro ciclo de desenvolvimento mais longo e elaborado, criando uma base para o jogo; em cima desta, nos ciclos de

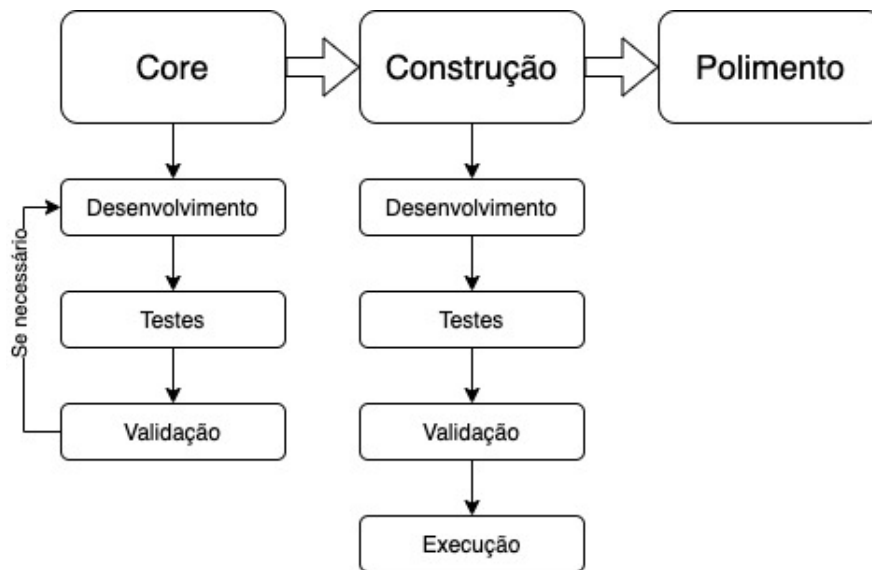
desenvolvimento subsequentes, é possível fazer pequenas alterações e melhorias e em seguida testá-las e validá-las de forma rápida.

Com um processo de desenvolvimento delimitado, pode-se partir para a aplicação do mesmo no processo de desenvolvimento de um jogo.

## 4.6 Resumo

Desta forma, é possível concluir que a metodologia estabelecida é dividida em três etapas: *Core*, Construção e Polimento. As primeiras duas etapas possuem ainda sub-etapas, isto é, etapas de desenvolvimento, testes e validação; no caso da primeira isso ocorre uma vez na definição do *core* do jogo e pode ser repetido caso se decida por uma mudança na temática do jogo ao final do processo de testes e validação; no caso da etapa de segunda, essas sub-etapas podem acontecer múltiplas vezes, relacionadas a desenvolvimento e teste de cada um dos elementos implementados. Um esquema da metodologia como um todo pode ser observado na [Figura 1](#)

Figura 1 – Fluxo da metodologia desenvolvida



Fonte: imagem própria

## 5 Aplicação da metodologia

Após delimitar e estruturar a metodologia de desenvolvimento, passou-se à fase seguinte deste trabalho, isto é, o desenvolvimento de um jogo utilizando o processo estruturado. Desta forma, começamos o desenvolvimento pela etapa de definição do *Core* do jogo.

### 5.1 Primeiro ciclo de desenvolvimento do Core

A primeira etapa, como delimitado anteriormente, consiste na determinação do core do jogo e isso foi feito seguindo exatamente os passos delimitados, isto é, hipótese e validação. A ideia inicial, tendo em mente o objetivo de desenvolver um jogo de causa social, foi desenvolver um projeto com o tema de relacionamentos abusivos. A ideia era criar um jogo que tratasse do assunto retratando-o como um problema colocando o jogador na pele de um abusadora, para que ficasse evidente para ele quão ruins as ações do personagem principal eram. Seguindo a metodologia, desenvolvemos o *core loop* do jogo, que, ambientado num cenário de RPG (*role playing game*) medieval, trataria da história de um herói que é ajudado por uma fada em suas batalhas; entretanto, ficaria claro para o jogador que esta ajuda causa danos à fada, o que faria com que a relação entre os dois personagens fosse caracterizada como um relacionamento abusivo.

Despendemos um tempo razoável na elaboração das ideias para esse jogo em termos de mecânicas e gameplay, sem, é claro, implementar nada, uma vez que ainda estávamos na fase de hipótese do ciclo de desenvolvimento. Nessa etapa, nosso objetivo era tentar garantir que o jogo poderia trazer algo de interessante para o jogador, porém com a ressalva de que, como trata-se de um tema extremamente delicado, o jogo não se tornasse ofensivo ou pouco sensível a pessoas que passaram por situações de relacionamento abusivo. Nesse momento ficou claro que, nesse caso, tratando-se de um jogo em que a narrativa era extremamente importante, principalmente para deixar claro que a relação entre os personagens era abusiva e que isso era algo ruim, era necessário desenvolver bem a narrativa, tanto quanto o *core loop*, bem como testá-la e validá-la. Esse desenvolvimento de narrativa não era algo que havia sido previsto quando a metodologia foi delineada da maneira que foi descrita no [Capítulo 4](#). Dessa forma, além de elaborar ideias de mecânicas, também foram elaboradas ideias de narrativa e enredo. Uma vez tendo feito isso, o passo seguinte era o de validação das hipóteses.

Vale ressaltar aqui que percebemos que era importante termos uma perspectiva de vítimas ou de pessoas próximas a vítimas de relacionamentos abusivos, no supracitado intuito de não tronar o jogo ofensivo ou pouco sensível. Essa ideia surgiu do exemplo do

jogo *Another Lost Phone: Laura's Story*; de acordo com Wilbur (2017), este jogo, assim como o nosso, trata de um relacionamento abusivo e, para acertar detalhes de temática e mensagem relacionados a esse tema, a equipe de desenvolvimento entrou em contato com profissionais experientes no tema. E foi algo análogo a isso que decidimos fazer. Para isso, uma das etapas do processo de validação que efetuamos foi entrar em contato com diferentes pessoas e grupos que dominam o assunto e, durante esse contato, um dos grupos posicionou-se contra a nossa ideia, o que acabaria fazendo com decidíssemos trocar o tema do jogo.

De acordo com as conversas que tivemos com esse grupo, chegamos a algumas conclusões que ocasionariam a troca de tema. A primeira delas trata da alta complexidade do tema, que tornaria bastante difícil desenvolver um jogo que fizesse um bom trabalho em retratar essa realidade, deixando clara toda a problemática compreendida por este tema. Também pesou o fato de o assunto ser extremamente sensível, capaz de gerar gatilhos para pessoas que jogassem o jogo, levantando até mesmo uma questão de quem seria o público alvo. Outra conclusão importante trata da questão de lugar de fala; como nenhum dos dois integrantes do grupo passou pela situação de um relacionamento abusivo, não é nosso lugar de fala tratar sobre o assunto e, para resolver isso, seria necessário um intenso contato com pessoas que tivessem esse lugar de fala, complicando bastante o desenvolvimento do projeto, ainda mais em tempos de pandemia, que dificultariam bastante o contato entre os integrantes do grupo e essas pessoas. Por último, o fato de termos um cronograma apertado (cerca de quatro meses) para a entrega do projeto torna todos esses fatores ainda mais complexos, aumentando muito a chance de que um ou mais dos problemas relatados acima se fizesse presente no nosso projeto.

Após deliberar extensamente sobre as questões expostas, decidimos trocar de tema, visando um com o qual tivéssemos mais liberdade para trabalhar sem ter que consultar outras pessoas e sem correr risco de ofender ou gerar gatilhos a ninguém. Dessa forma, decidimos que o tema a ser escolhido não necessitaria de um enfoque tão grande em causas sociais, de modo a simplificar o processo de criação do core do jogo.

Entretanto, ainda foi possível tirar algumas conclusões e aprendizado a partir dessa experiência. A primeira foi de que, em um jogo focado na narrativa, como era a ideia que tínhamos do nosso projeto, o desenvolvimento do enredo do jogo deve ser feito desde o início, junto ou até antes do gameplay, passando por processo de validação desde o princípio. Outro aprendizado foi que, caso a equipe de desenvolvimento esteja tratando de um tema que não domina, é necessário envolver pessoas que o dominem desde muito cedo no processo de desenvolvimento, deixando claro a importância das opiniões dessas pessoas no processo de desenvolvimento.

## 5.2 Segundo ciclo de desenvolvimento do Core

Uma vez decidido que o tema do jogo seria alterado, retornamos a fase de hipótese. Para não atrasar ainda mais o andamento do projeto, decidimos seguir na direção da segunda ideia que mais havíamos gostado na primeira vez em que havíamos passado por essa fase de hipótese: um jogo sobre um super herói jovem, no qual o jogador, além de comandar as ações do personagem em suas batalhas, tem que tomar decisões relacionadas ao dia a dia do personagem, que refletem em consequências durante as batalhas do herói. Por exemplo, o jogador pode ser confrontado sobre escolher entre ficar acordado até mais tarde estudando para uma prova, que pode lhe trazer benefícios futuros, mas fará com que ele tenha menos energia para a próxima luta, ou dormir cedo e lutar com a energia completa.

### 5.2.1 Core

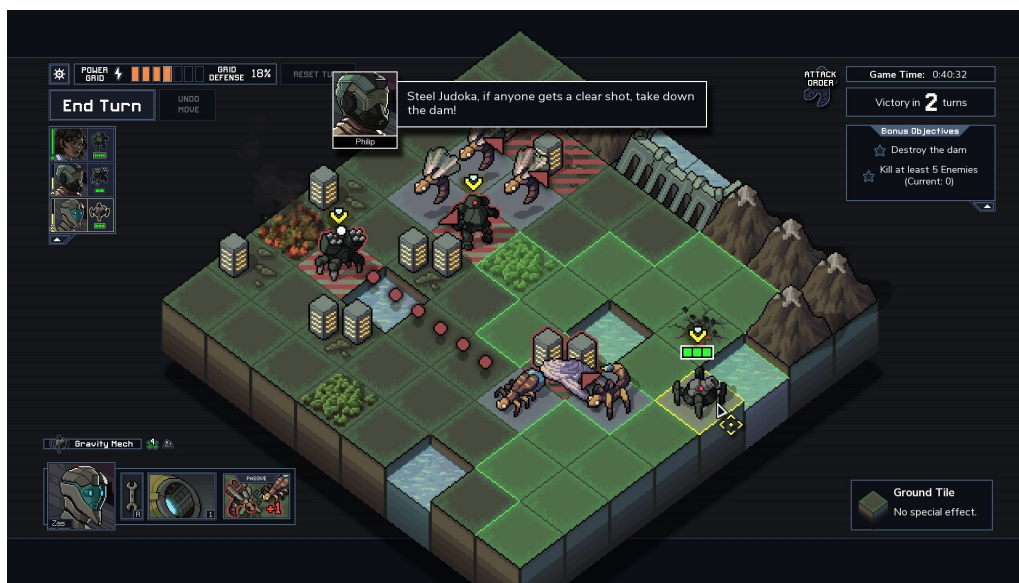
A mecânica mais importante no caso desse jogo, o core loop, seria a mecânica de batalhas. Decidiu-se seguir por um mecanismo de combate baseado em turnos, utilizando um tabuleiro retangular dividido em células como campo de batalha. Dentre as principais inspirações para este modelo, temos os jogos Mega Man Battle Network, produzido pela Capcom e lançado em 2001; e Into The Breach, produzido pela Subset Games e lançado em 2018. Frames dos jogos citados são apresentados na [Figura 2](#) e na [Figura 3](#).

Figura 2 – Frame do jogo Mega Man Battle Network



Fonte: [Lunning \(2019\)](#)

Figura 3 – Frame do jogo Into The Breach



Fonte: [Tran \(2018\)](#)




#### 5.2.1.1 Desenvolvimento

Uma vez definido o tema do jogo e sua mecânica mais importante, partiu-se para o próximo passo, isto é, o desenvolvimento da hipótese de core loop, que se resumiu basicamente em prototipá-lo da forma mais simples possível, como um protótipo em papel por exemplo, e testá-lo com possíveis jogadores. Antes de prototipar, foram feitas novas reuniões de brainstorming para, primeiramente, listar possíveis mecânicas a serem incorporadas nas batalhas, como tipos de ataques, esquemas de movimentação, tipos de inimigos, obstáculos para o jogador, power-ups, entre outros; e, em sequência, criar um grupo básico de fases a serem testadas com o jogador. O grupo de fases apresentado aos jogadores continha quatro fases, cada uma desenhada com o objetivo de testar algo específico. As fases foram prototipadas utilizando o Google Slides, de modo a permitir que fossem feitos testes remotamente, tendo em vista a situação atual relacionada à pandemia da COVID-19.

A primeira fase, reproduzida na [Figura 4](#), era a mais simples de todas e tinha como objetivo ensinar ao jogador a mecânica base do jogo e foi desenhada de modo a testar uma ideia que tivemos sobre a progressão do jogo. A ideia era que o jogador obtivesse poderes ao matar os inimigos, isto é, quando ele matasse um inimigo que possui um poder que ele não possui, ele adquiriria este poder.

Na [Figura 4](#), é possível visualizar o herói, no canto inferior esquerdo, bem como três vilões. O objetivo do herói e, por conseguinte, do jogador, é destruir todos os inimigos presentes no campo de batalha. Em vermelho, ao lado de cada personagem, é possível

Figura 4 – Primeira fase do protótipo testado

		 HP: 70 ATK: 20 MOV: 2	
			 HP: 50 ATK: 15 MOV: 1
 HP: 200 ATK: 25 MOV: 2			
		 HP: 70 ATK: 20 MOV: 2	







Fonte: imagem própria

visualizar seus health points (ou HP), isto é, sua vida; em verde, o dano promovido por ataques de curto alcance, que podem ser desferidos "em cruz" (isto é, para cima, para baixo e para os lados do personagem) nas células imediatamente adjacentes ao personagem; em azul, o dano promovido por ataques de longo alcance, que também podem ser desferidos "em cruz" em qualquer célula nas linhas que cruzam o personagem; e em amarelo, os dados de movimentação de cada personagem, isto é, quantas células o personagem podem avançar em cada turno. Aqui cabem dois esclarecimentos. O primeiro trata da ordem de ataque e movimento dos personagens: o jogador se movimenta e depois ataca, ao passo que os inimigos atacam e depois se movimentam; essa decisão foi tomada de modo a simplificar o modelo mental que o jogador deveria criar para montar sua estratégia de jogo; isto porque se os vilões pudessem se movimentar e aí sim atacar, ficaria difícil para o jogador escolher seus movimentos de modo a se esquivar dos ataques adversários. O segundo esclarecimento é que todos os personagens possuem apenas ataques de curto alcance, com exceção do inimigo mais à direita do mapa, que possui apenas ataque de longo alcance; assim, ao final dessa fase o herói adquire um novo tipo de ataque, o de longo alcance.

A segunda fase, representada na [Figura 5](#) e cujo objetivo, novamente, é destruir todos os oponentes, traz inimigos um pouco diferentes e a ideia é testar se o jogador fará uso do novo tipo de ataque adquirido. O número de inimigos, superior ao da última fase, torna bastante difícil para o jogador vencer esta fase sem fazer uso do novo ataque que possui. Para dificultar, a linha de inimigos menos à direita, que apresenta inimigos "voadores", tem movimentação maior que o herói, fazendo com que seja difícil para ele fugir dos mesmos, obrigando-o a matá-los o mais rápido possível.

A terceira fase, representada na [Figura 6](#), traz um novo objetivo para o jogador. Agora, além de matar todos os inimigos, ele deve se preocupar em proteger um outro personagem, representado pela imagem de um cão, imediatamente à direita do personagem


Figura 5 – Segunda fase do protótipo testado

			 HP: 15 ATK: 5 MOV: 3	
				 HP: 70 ATK: 20 MOV: 2
 HP: 150 ATK: 15 ATK: 25 MOV: 2			 HP: 15 ATK: 5 MOV: 3	
				 HP: 70 ATK: 20 MOV: 2
			 HP: 15 ATK: 5 MOV: 3	

Fonte: imagem própria

na figura. Uma observação a mais é que existe, à direita do cão, uma pedra, que inicialmente bloqueia os ataques de longo alcance da torre em direção ao cão. A ideia por trás desta fase é testar se essa mecânica de salvar um companheiro trazia um incentivo a mais para o jogador ou se fazia a fase ficar desinteressante para o mesmo. Vale ressaltar que é impossível passar dessa fase sem sacrificar alguns health points do herói para evitar que o cão seja atingido pelos ataques de longo alcance da torre. Essa fase foi testada de duas maneiras diferentes: na primeira, os "goblins"(os inimigos verdes que existem acima e abaixo da torre) tinham como objetivo atacar também o cão; na segunda, o objetivo deles era atacar o próprio herói. O objetivo destes testes diferentes era verificar qual seria mais difícil para o jogador.

Figura 6 – Terceira fase do protótipo testado

				 HP: 40 ATK: 10 MOV: 3
 HP: 100 ATK: 10 ATK: 20 MOV: 2	 HP: 40	 HP: 10		 HP: 50 ATK: 10 MOV: 0
				 HP: 40 ATK: 10 MOV: 3

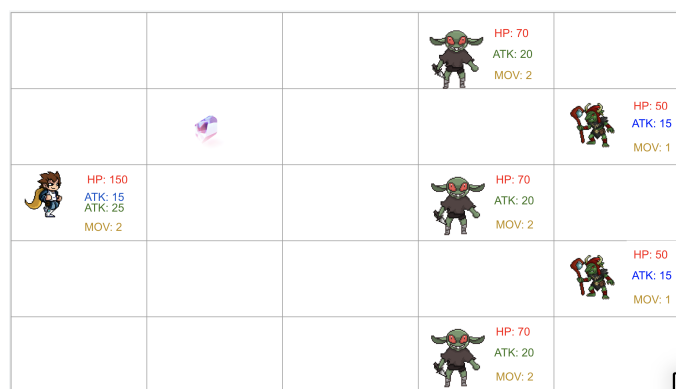
Fonte: imagem própria

A quarta e última fase do protótipo, representada na [Figura 7](#), traz uma nova mecânica a ser testada, a de power-ups. A ideia dos power-ups é que, em certos momentos do jogo, é oferecida ao jogador a possibilidade de coletar um item, nesse caso uma espécie



de cristal (na imagem abaixo, visível uma célula à direita e uma célula acima do herói), que lhe fornecerá algum benefício. A ideia desta fase era testar um balanceamento diferente das outras fases, no qual o herói, sem a ajuda dos power-ups, dificilmente conseguiria vencer a fase. O intuito disso é fazer com que a fase pareça mais difícil para o jogador, amplificando sua sensação de conquista ao conseguir passar da fase. Outro teste era relacionado aos tipos de power-ups: foram testados dois tipos diferentes, um que triplicava os valores de ataque do herói e outro que restaurava parte de seus health points. Aqui, vale ressaltar que o objetivo volta a ser o de destruir todos os oponentes presentes no campo de batalha.

Figura 7 – Terceira fase do protótipo testado



Fonte: imagem própria

Após prototipar essas fases fazendo uso do Google Slides, foi possível avançar para a fase seguinte do projeto, isto é, utilizar os protótipos de fato testando-os com voluntários.

### 5.2.1.2 Testes

A ideia de testar o jogo com voluntários tem como principal objetivo, de acordo com a metodologia estrutura no [Capítulo 4](#), validar a ideia de core, isto é, verificar se a experiência é divertida para o jogador. Além disso, existem alguns objetivos secundários, relacionados a validação de algumas mecânicas específicas, como, por exemplo, a progressão no jogo, objetivos em fases (como, no caso da fase ilustrada na [Figura 6](#), a necessidade de "salvar" um outro personagem) e mecânica de power ups (como ilustrado na [Figura 7](#)).

Antes de iniciar qualquer teste do protótipo, foi necessário o desenvolvimento de um Termo de Consentimento Livre e Esclarecido (TCLE), de acordo com a Resolução 466, de 12 de Dezembro de 2012. O texto do termo desenvolvido está reproduzido no [Apêndice A](#).

Uma vez tendo o TCLE preparado, foi possível iniciar os testes com voluntários. O roteiro simples do teste conta com os seguintes passos: leitura e consentimento com o TCLE via Google Forms; teste digital ou pessoal fazendo uso do google slides, no qual o

voluntário comanda o personagem principal e um dos estudantes responsáveis pelo trabalho comanda os inimigos; curta entrevista gravada com o participante. Foi escrito um roteiro contendo as perguntas a serem feitas ao voluntário durante essa entrevista, que se encontra reproduzido no [Apêndice B](#).

Foram realizados testes com três participantes, cujas principais conclusões de cada uma estão listadas nos itens abaixo.

#### 5.2.1.2.1 Primeiro teste

O primeiro teste apresentou problemas de balanceamento na terceira fase, isto é, se apresentou difícil demais para o jogador. Na fase em questão, apresentada na [Figura 6](#), o jogador deve defender um personagem secundário, representado na figura por um cachorro. A princípio, além da torre, mais à direita na imagem, os outros vilões também visavam atacar o cachorro e não o herói, sendo esta a causa da dificuldade excessiva. Este problema foi solucionado fazendo com que apenas a torre ataque o cachorro e os outros vilões ataquem o herói. Fora este problema, o teste correu bem e a participante considerou a experiência divertida; ela considerou inclusive que a fase do cachorro foi a mais interessante, justamente pela necessidade de salvar o personagem secundário, sendo esta uma característica "diferencial" da fase. Outra possível falha apontada pela jogadora foi a facilidade do primeiro nível, embora também tenha sido apontado que isto é compreensível, uma vez que se trata da fase que apresenta o jogo ao participante. Outro ponto de interesse apontado pela participante foi o da progressão atingida ao final da primeira fase, momento no qual o jogador consegue um poder novo. Quanto a mecânicas de movimentação e ataque, a participante destacou sentir que a movimentação de todos os personagens, não apenas do herói, era limitada demais e que uma modificação neste sistema poderia tornar o jogo mais interessante ainda; elencou também o fato de que um "ataque em diagonal" poderia ser bastante útil e interessante para o jogador.

#### 5.2.1.2.2 Segundo teste

O segundo teste transcorreu sem maiores problemas. Quanto ao objetivo principal do teste, o jogador considerou a experiência divertida, gostando principalmente da última fase pelo nível de dificuldade apresentado. Assim como a primeira participante, o jogador levantou pontos sobre a facilidade da primeira fase, mas também compreendendo que isso era motivado pela necessidade de apresentar as mecânicas do jogo. O jogador gostou bastante da mecânica de power-ups, especificando que se interessou pela diversidade (foram apresentados dois tipos de power-ups, como supracitado) e pela aleatoriedade dos pontos em que estes apareciam no mapa; entretanto, o jogador levantou o ponto de que a diferença de efeito dos power-ups pode ser um problema eventualmente para o jogador, dificultando seu planejamento durante a fase. Além disso o jogador sugeriu a possibilidade de um novo

tipo de dificuldade para o jogador, que não havia sido pensada, relacionada a limitação de tempo para o jogador durante a fase.

#### 5.2.1.2.3 Terceiro teste

O terceiro e último teste transcorreu sem maiores problemas. Assim como os outros dois jogadores, o terceiro participante gostou do jogo. Alguns pontos de interesse principais incluem a diversidade de tipos de ataque, bem como a quebra de expectativa proporcionada pela presença dos power-ups na última fase. Entretanto, o último jogador não gostou da terceira fase, na qual o jogador deve defender um personagem secundário; segundo ele, essa fase não necessitava de grandes decisões estratégicas, o sentimento do jogador era que o cachorro apenas o atrapalhava durante a fase. Uma sugestão do jogador para melhorar essa fase foi de acrescentar alguma recompensa ao jogador por salvar o personagem secundário. Além disso, algumas sugestões de mecânicas diferentes foram fornecidas pelo jogador: fases em que o objetivo é apenas sobreviver uma determinada quantidade de rodadas e fases em que o terreno apresenta limitações e obstáculos ao jogador.

#### 5.2.1.3 Validação

A partir dos testes foi possível concluir que o jogo atingia o seu objetivo, isto é, sendo divertido para os jogadores. Esta conclusão é extremamente importante por que, por si só, nos permite avançar para a etapa seguinte da criação do jogo, isto é, a etapa de construção, como especificado no [Capítulo 4](#). Além disso, foi possível também conseguir alguns insights sobre mecânicas que não havíamos pensado para o jogo, como a limitação de tempo mencionada na [subseção 5.2.1.2.2](#) e limitações e obstáculos impostos pelo terreno como mencionado na [subseção 5.2.1.2.3](#). Outra conclusão interessante é que algumas mecânicas ressoam de maneiras diferentes em públicos diferentes, como ilustram a [subseção 5.2.1.2.1](#) e a [subseção 5.2.1.2.3](#), uma vez que a voluntária do primeiro teste achou a fase em que o usuário deve defender um personagem interessante, ao passo que o voluntário do terceiro teste não achou essa fase divertida; entretanto, vale ressaltar que essa conclusão, enquanto interessante, não foi levada em consideração durante a fase de construção do jogo.

### 5.2.2 Construção

De acordo com a metodologia estabelecida no [Capítulo 4](#), após a etapa de determinação do core do jogo e devida validação, pode-se passar a fase de construção do jogo, que assim como a fase anterior é composta por fases de Desenvolvimento, Testes e Validação. Assim, após as conclusões tiradas na [subseção 5.2.1](#), pode-se passar a fase na qual é iniciado o desenvolvimento do jogo. Vale ressaltar que, como determinado no [Capítulo 4](#), o ideal é que este processo em etapas seja iterativo, entrando e saindo da

fase de Desenvolvimento conforme o resultado da etapa de Testes. Entretanto, devido a limitações de tempo impostas pelo calendário do projeto, foi realizado apenas um ciclo desses, no qual foi implementada uma versão bastante completa do jogo. Vale ressaltar que, como descreve a metodologia, essa implementação permite que a extensão da mesma, em outros ciclos de desenvolvimento (que serão listados como possíveis próximos passos deste projeto) seja bastante facilitada.

### 5.2.2.1 Desenvolvimento

Para facilitar o desenvolvimento do jogo, foi escolhida a *game engine* Unity. Essa escolha deve-se principalmente a dois motivos: os integrantes do projeto já tinham alguma experiência prévia com a ferramenta, em oposição a outras *game engines* célebres como Unreal e Godot; além disso, a Unity possui uma comunidade extremamente ativa e documentação bastante completa, de acordo com [Dealessandri \(2020\)](#).

Uma vez decidida a ferramenta a ser utilizada para o desenvolvimento, foi possível partir para o mesmo de fato. Muito do que foi desenvolvido utilizou como base o material desenvolvido por [Parham \(2015a\)](#), sob a MIT License. A primeira coisa a ser feita foi a obtenção de um conjunto de *assets* básico, contendo texturas e elementos de interface simples, que poderiam posteriormente ser substituídos com facilidade por outros mais complexos, no sentido de polir o jogo, durante a última etapa da fase seguinte a de Construção do jogo, isto é, a fase de Polimento do mesmo. A etapa de desenvolvimento do jogo pode ser explicada de forma seccionada, explicando as partes que foram implementadas e com qual objetivo.

#### 5.2.2.1.1 Gerador de *Boards*

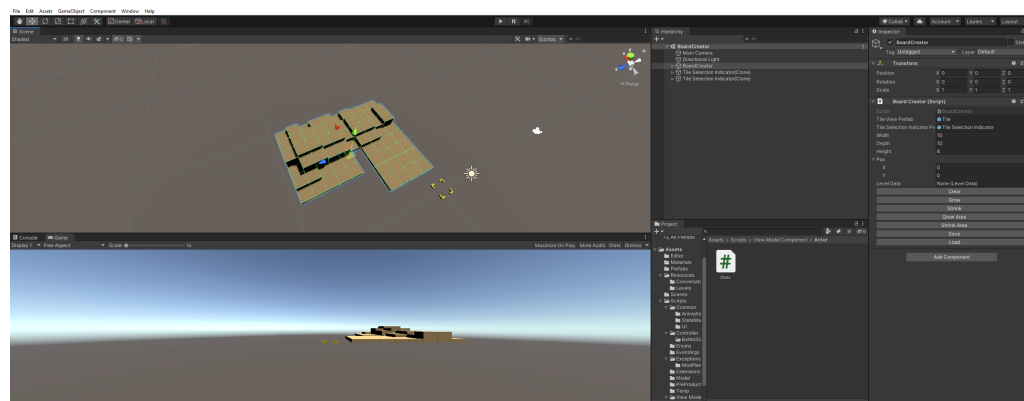
A partir dos *assets*, a etapa seguinte do desenvolvimento foi a construção de um gerador de *boards*. No contexto desse projeto, um *board* é uma fase do jogo, isto é, um tabuleiro repartido em quadrados onde o jogador e os inimigos serão posicionados, com objetivos, obstáculos, entre outros. O objetivo de criar este gerador é ter em mãos um editor visual de fases, podendo facilmente criar (de maneira manual ou automatizada - e randômica), editar e guardar fases para o jogo, que serão fornecidas ao jogador em alguma ordem particular. Para tal, foram criados *prefabs* (no contexto da engine e deste jogo, objetos a serem posicionados em cenas, que podem ser criados repetidamente com facilidade) representando células do tabuleiro, denominadas *tiles*, a serem utilizadas pelo gerador de fases para a criação das mesmas. É importante ressaltar que essas *tiles* explicitam posições do tabuleiro onde jogador e inimigos podem pisar; vale destacar também que o espaço de tela do jogo foi dividido usando *structs* customizadas, nomeadas de *points*, que representam os pontos na tela onde as *tiles* podem estar posicionadas. Vale também

ressaltar que essas estruturas são relacionadas, em cada fase, por uma estrutura de dados do tipo *Dictionary*.

Para criar o editor, foi utilizado o próprio *inspector* de cenas da Unity, aliado a um *script*, permitindo editar fases, criando e modificando *tiles* dentro da própria cena. No contexto da Unity, uma cena é uma parte do jogo (pode ser um nível, por exemplo, ou, nesse caso, o próprio editor de fases) contendo os objetos que dela participam.

Aliando os conceitos expostos, foi possível criar um editor básico de *boards*, no qual é possível criar e modificar fases com facilidade. Uma limitação neste momento é que só é possível alterar os espaços, sem editar outros aspectos como quantidade e posicionamento de inimigos, bem como posicionamento do heroi. Estas melhorias serão adicionadas posteriormente. O editor de fases construído pode ser observado na [Figura 8](#).

Figura 8 – Reprodução do editor básico de fases construído



Fonte: imagem própria

#### 5.2.2.1.2 Máquina de Estados

A etapa seguinte foi a construção de uma máquina de estados, responsável por controlar a lógica do jogo. Um padrão de arquitetura de máquina de estados bastante usual, de acordo com [Parham \(2015a\)](#), é o uso de uma simples flag registrando qual o estado em que o jogo se encontra. Entretanto, este modelo encontra problemas de escalabilidade, com a complexidade crescendo muito conforme o número de estados aumenta. A solução para este problema é registrar o estado em um objeto, este contendo seus próprios atributos e métodos; isto é, cada possível estado do jogo representa uma classe diferente, que herda de uma classe básica contendo métodos e atributos comuns a todas as classes filhas. As classes filhas da classe base, por sua vez, podem também ter classes filhas, como é o caso do estado de batalha (*BattleState*), que contém "sub-estados" para as diferentes etapas da batalha, como inicialização e movimentação. Vale ressaltar que a criação e destruição de objetos referentes ao estado é controlada por uma outra classe, que representa a máquina

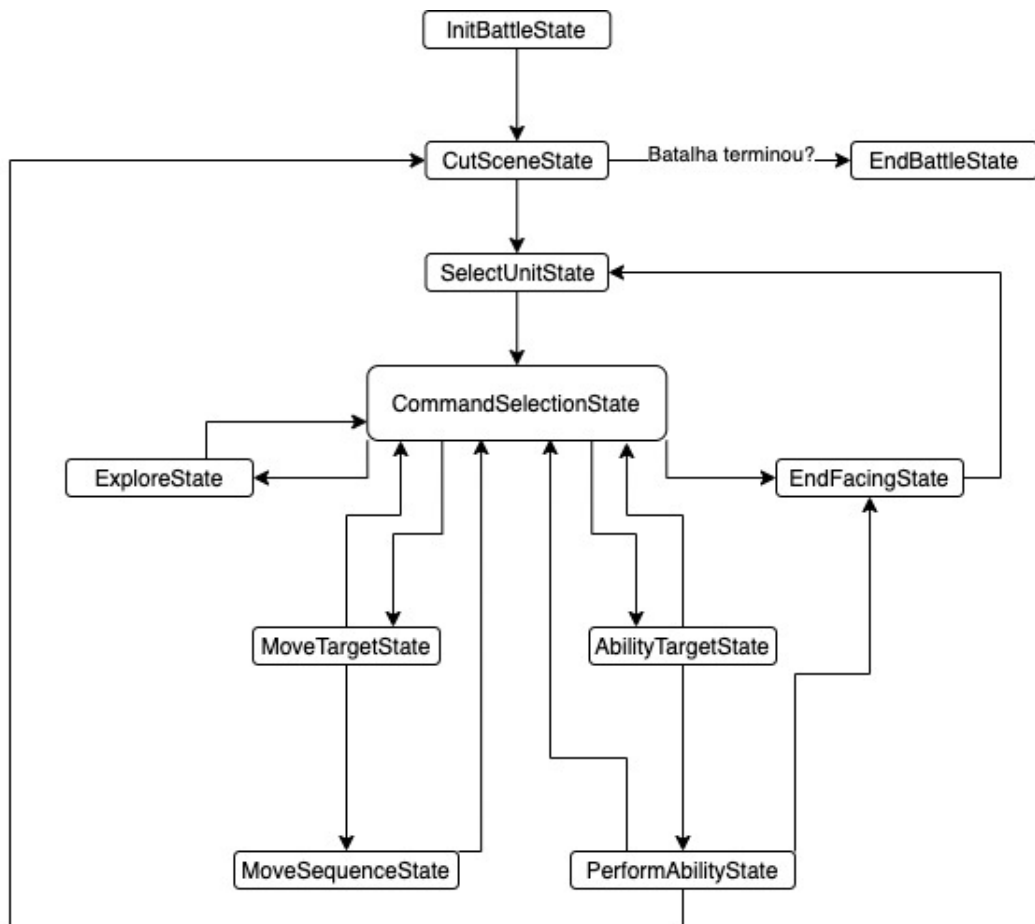
de estados (*StateMachine*); que por sua vez também possui classes filhas referentes a estados do jogo, como no caso, também, da batalha, que possui uma classe específica para controlar os estados (*BattleController*).

Para exemplificar o funcionamento da máquina de estados, está ilustrado, de maneira simplificada, na [Figura 9](#) o fluxo que ocorre durante uma batalha. O primeiro estado, *InitBattleState*, é responsável por inicializar a batalha, isto é, criar tabuleiro, posicionar os personagens, entre outras atribuições iniciais. Uma vez que isto é feito, a máquina transiciona para o estado *CutSceneState*, responsável por apresentar uma cutscene, que pode ser de início ou de final da batalha. Por enquanto, trataremos do caso de cutscene inicial, isto é, ao final da mesma, o estado transiciona para *SelectUnitState*, no qual o jogo seleciona, de acordo com uma ordem predeterminada, qual personagem jogará (o herói ou algum dos inimigos). Após isso, a máquina evolui para o *CommandSelectionState*, no qual o jogador (ou a máquina, de modo automático) escolherá que ação deseja fazer para o personagem: explorar (*ExploreState*), se movimentar (*MoveTargetState* e o subsequente *MoveSequenceState*), efetuar um ataque (*AbilityTargetState* e o subsequente *PerformAbilityState* - existem alguns estados intermediários relativos a tipo de ataque que foram omitidos aqui por simplicidade) ou encerrar seu turno (*EndFacingState*). Caso o jogador decida explorar, ele entra em uma situação em que pode mover a câmera pelo cenário, de modo a visualizá-lo melhor; em seguida, o jogador pode retornar ao estado de seleção de comando. Caso o jogador decida se mover, o *MoveTargetState* é o estado no qual ele seleciona, dentro de algumas opções, para qual *tile* deseja mover seu personagem, tendo como estado seguinte aquele em que a movimentação de fato ocorre, *MoveSequenceState*, retornando enfim para o estado de seleção de comando. Caso decida atacar, ele escolhe qual inimigo deseja atacar em *AbilityTargetState* e, em seguida, no estado *PerformAbilityState* o ataque de fato acontece e, enfim, a máquina pode retornar para a seleção de comando se o jogador ainda puder se mexer ou entrar no estado de encerrar o turno, caso o jogador já tenha se mexido; é possível também que o jogo evolua para o estado de cutscene novamente, caso alguma condição de vitória tenha sido atingida após o ataque tenha sido feito; neste caso, após a cutscene a batalha é encerrada, com o estado *EndBattleState*. O último estado para o qual o jogo pode evoluir a partir da seleção de comando é o *EndFacingBattle*, responsável por encerrar o turno e retornar à seleção de qual personagem jogará.

### 5.2.2.1.3 Sistema de Diálogos

Após implementar a máquina de estados, relevante para quaisquer situações do jogo, partiu-se para o desenvolvimento do sistema de diálogos do jogo, relevante para a mecânica de escolhas citada no início da [seção 5.2](#). Recapitulando, a ideia é que, entre as batalhas, o jogador deva fazer escolhas relacionadas à vida pessoal do herói, que terão reflexos durante a batalha; estas escolhas serão feitas via diálogo com outros personagens.

Figura 9 – Diagrama simplificado dos estados de uma batalha



Fonte: imagem própria

Decidiu-se seguir um processo de desenvolvimento baseado em criar uma implementação básica de diálogos, a princípio sem escolhas para o jogador, apenas apresentando falas do personagem principal e de outros que com ele dialogariam; para, em seguida, melhorar esse sistema, tornando as falas do personagem principal customizáveis, isto é, permitindo ao jogador escolher, dentro de um conjunto de opções, quais falas o personagem diria.

A implementação do sistema de diálogos se baseia em quatro classes principais:

- SpeakerData***: classe responsável por guardar as falas de um personagem dentro de um diálogo, bem como a posição em que essas falas aparecerão na tela.
- ConversationData***: classe responsável por agrupar objetos da classe *SpeakerData* relacionados a um mesmo diálogo dentro do jogo; isto é, todo diálogo dentro do jogo é implementado por meio de um objeto da classe *ConversationData*, que guarda as falas que pertencem àquele diálogo.

- c) **ConversationPanel**: classe responsável pela visualização das falas na tela do jogo.
- d) **ConversationController**: classe responsável por controlar o processo de um diálogo, mostrando os elementos de UI necessários com as falas correspondentes.

A implementação inicial resultou em um sistema simples, como ilustrado na [Figura 10](#). O próximo passo é alterar essas classes, de modo a permitir que o usuário escolha as falas a serem ditas pelo seu personagem. Essa etapa do desenvolvimento foi postergada para a etapa de Polimento do jogo, conforme a metodologia explicada no [Capítulo 4](#).

Figura 10 – Ilustração do diálogo em sua forma simplificada



Fonte: [Parham \(2015b\)](#)

#### 5.2.2.1.4 Painel de Estados dos Personagens

Outro ponto interessante do desenvolvimento do jogo foi a criação dos painéis de estado dos personagens envolvidos nas batalhas; isto é, os elementos de UI responsáveis por mostrar para o jogador as informações relevantes sobre os personagens envolvidos na batalha (tanto o herói como seus inimigos), bem como um avatar sinalizando a qual personagem pertencem aqueles valores.

Para a lógica por trás desse elemento, foi criada uma nova classe denominada *StatPanelController*, responsável por mostrar e esconder os painéis na interface do jogo. Essa classe é, por sua vez, acionada em diferentes estados do jogo, quando se faz necessário mostrar esses painéis retratando informações do herói ou de seus inimigos.



### 5.2.2.1.5 Habilidades

Um elemento extremamente relevante do jogo diz respeito às habilidades do personagem, isto é, o que ele pode fazer: movimentação e ataques. O funcionamento desses elementos está atrelado a três pontos básicos.

O primeiro ponto é o menu de habilidades, isto é, o elemento de UI responsável por mostrar ao usuário quais ações ele pode realizar, bem como permitir que ele selecione uma delas. O funcionamento desta parte é relativamente simples e é controlado pela classe *AbilityMenuPanelController*, referenciada em alguns dos estados de batalha.

Em seguida, outro ponto importante diz respeito ao alcance das ações do jogador, isto é, até onde podem se estender suas ações. No caso de uma movimentação, isto diz respeito a até onde no tabuleiro o jogador pode se mover partindo da posição em que seu personagem se encontra antes do movimento; e no caso do ataque, diz respeito a até onde seus ataques alcançam, a partir da posição em que ele se encontra. Este comportamento é um pouco mais complexo que o anterior, visto que ações diferentes possuem alcances calculados de maneira diferente. Estão inclusos no jogo os seguintes cinco tipos de cálculo de alcance diferentes:

- a) **Alcance constante:** é um alcance em linha reta horizontal e vertical, que não varia em nenhum momento do jogo.
- b) **Alcance próprio:** algumas ações de personagem podem ter como objetivo ele próprio, e assim o alcance se estende somente ao espaço no tabuleiro em que ele se encontra.
- c) **Alcance infinito:** é um alcance que engloba todas os espaços do tabuleiro.
- d) **Alcance em cone:** um alcance que depende da orientação para a qual o personagem está virado, englobando um cone naquela direção.
- e) **Alcance em linha:** semelhante ao alcance constante, porém se estendendo da posição do jogador até os extremos do tabuleiro.

Esses alcances são utilizados para calcular e mostrar ao jogador quais espaços do tabuleiro ele pode atingir com suas ações, sejam elas movimentações ou ataques.

O terceiro e último ponto relevante diz respeito às áreas de efeito das habilidades. Algumas habilidades como ataques possuem, além de um alcance, uma área de efeito, isto é, quando o jogador seleciona um espaço no tabuleiro para executar uma ação, pode acontecer de os espaços adjacentes àquele espaço sofrerem algum efeito da ação. Essas áreas de efeito possuem relação com o alcance explicado anteriormente e podem ser de três tipos:

- a) **Área unitária:** dentro do alcance do jogador, apenas um dos espaços do tabuleiro (o que ele selecionar) será afetado por aquela ação.

- b) **Área específica:** diz respeito a ações que influenciam, além do espaço selecionado, os espaços imediatamente adjacentes a ele. Vale ressaltar que esse tipo de ação pode acabar até influenciando espaços fora do alcance do jogador, caso ele selecione, por exemplo, um extremo de seu alcance.
- c) **Área total:** diz respeito a ações que influenciam todo o seu alcance, isto é, a área de efeito é exatamente igual ao alcance.

Essas áreas tem duas funções: mostrar ao jogador quais espaços do tabuleiro serão afetados por uma ação e, além disso, calcular se um ou mais personagens serão atingidos por uma determinada ação e, em caso positivo, quais personagens o serão.

#### 5.2.2.1.6 Fábrica de Unidades

Outro ponto interessante da implementação diz respeito à criação dos personagens dentro de uma cena do jogo. Ao inicializar uma fase, é necessário criar objetos para o personagem principal e seus inimigos; para automatizar esse processo foi criada uma classe cuja única responsabilidade é criar personagens de acordo com inputs fornecidos, a classe *UnitFactory*.

Durante a inicialização da fase, métodos dessa classe são chamados de modo a criar e posicionar personagens no tabuleiro, com os corretos valores de ataque, movimentação, entre outros.

#### 5.2.2.1.7 A.I.

Após codificar diversos comportamentos necessários como estados de batalha, elementos de UI, criação automática de personagens, entre outros, o único elemento faltante à uma fase plenamente funcional, na qual um jogador pode controlar o herói contra os inimigos é uma inteligência artificial para os inimigos. Nesse ponto da implementação, o jogo poderia ser jogado por duas pessoas, uma controlando o herói e a outra os inimigos por exemplo, uma vez que todos os sistemas necessários a essa batalha estão funcionando.

Após analisar algumas opções de sistemas para inteligência artificial, como Minimax, descrito em [Minimax \(2020\)](#), e *Monte Carlo tree search*, descrito em [Monte... \(2020\)](#), decidiu-se por uma implementação mais simples de A.I., inspirada pelo sistema Gambit da série de jogos Final Fantasy, como ilustrado por [Fabro \(2019\)](#).

Esse sistema é baseado em uma lista de ações possíveis para o personagem controlado pela inteligência artificial, ordenadas com base em prioridade de execução. Entretanto, além da prioridade, cada ação possui também uma condição de realização. Assim, para escolher qual(is) ação(ões) uma unidade inimiga irá realizar, a inteligência artificial passa por essa lista item a item, realizando as ações cujas condições sejam satisfeitas até que, por limitação do jogo, o personagem não possa mais realizar ações.

Decidiu-se por esse sistema simplificado por duas principais razões. A primeira é que ele é, de fato, mais simples de implementar do que sistemas mais complexos. A segunda é que uma inteligência artificial "menos inteligente" pode proporcionar ao jogador momentos de satisfação, quando ele sente que levou a melhor sobre a inteligência.

#### 5.2.2.2 Testes

Devido à limitação de tempo do projeto, decidiu-se fazer um único grande ciclo de desenvolvimento, composto pelo desenvolvimento de todas os elementos citados em [subseção 5.2.2](#); ao finalizar a implementação, pode-se passar à fase de testes das características implementadas. Entretanto, em decorrência das práticas de isolamento social causadas pela pandemia da COVID-19, foram feitos apenas testes internos do jogo.

Esses testes internos apontaram alguns pontos de melhoria, que poderiam ser atacados na etapa de Polimento ou ficarem listados como próximos passos no desenvolvimento deste jogo, caso fosse decidido que a implementação desses pontos fugiria ao escopo deste trabalho. Dois pontos bastante relevantes dizem respeito à narrativa e enredo do jogo que, nesse ponto do desenvolvimento, ainda estavam bastante crus. Foi decidido que esses pontos seriam atacados ainda neste trabalho, durante a etapa de polimento. Foi possível também encontrar alguns bugs, que também seriam tratados na fase seguinte da metodologia.

Outros pontos que necessitam melhorias dizem respeito a elementos que, no princípio do desenvolvimento (na etapa de *core*) haviam sido listados como elementos do jogo, mas haviam sido deixados de lado durante a fase de construção para efeitos de simplificação do projeto. Estes incluem características como sistema de decisões do jogador nos diálogos; e progressão no jogo, com o personagem controlado pelo jogador obtendo novos poderes na medida em que o jogador progride ao longo do jogo.

#### 5.2.2.3 Validação

O mais importante passo após a construção do jogo é a etapa de validação após os testes. Isto é, decidir se o jogo atende as características delimitadas e testadas na etapa de *core*, bem como avaliar se o jogo continua tão divertido quanto o protótipo montado e testado naquela etapa.

Ao finalizar a etapa de construção, foi possível avaliar que o jogo atende as principais características predefinidas, sendo um jogo de combate baseado em turno bastante semelhante ao protótipo montado. De acordo ainda com a opinião dos alunos envolvidos nesse projeto, a versão do jogo obtida após o processo de construção consegue ser bastante divertida.

### 5.2.3 Polimento

Após desenvolver e testar uma versão básica do jogo, foi possível alinhar alguns elementos do jogo que precisariam de polimento para atingir uma versão final do jogo. Esses itens a serem melhorados incluem elementos tanto previstos durante o processo de desenvolvimento da etapa de Construção, uma vez que foram feitas escolhas de modo a simplificar esse trabalho, permitindo que o jogo básico ficasse pronto para ser testado e validado o mais rápido possível; os itens incluem também elementos a serem revisados depois do processo de teste e validação da versão inicial do jogo; por último, incluem elementos da área artística do jogo, como trilha sonora e roteiro.

#### 5.2.3.1 Desenvolvimento

O desenvolvimento, no contexto da etapa de polimento, foi dividido em algumas partes a serem melhoradas.

##### 5.2.3.1.1 Enredo

Um dos quesitos que careciam de mais elaboração era o enredo do jogo. A ideia, após a decisão de mudar de tema, era desenvolver um jogo sobre um super herói jovem, que deve, além de proteger a cidade em que vive, tomar decisões acerca de sua vida pessoal, sendo beneficiado ou penalizado por essas decisões durante os combates. Porém, em nenhum momento haviam sido determinadas características mais detalhadas desse herói, como nome, origem, motivação, entre outros. Por conseguinte, um dos passos da etapa de polimento foi desenvolver uma história mais elaborada para a personagem. A ideia é que essa história servisse como pano de fundo para o jogo desenvolvido, entretanto, sem que fosse completamente implementada no software desenvolvido no contexto do trabalho de conclusão de curso, uma vez que seria demasiadamente elaborado e trabalhoso, fugindo ao escopo deste trabalho. Foi decidido então que seria escrita uma história base, algo nas linhas de uma sinopse, e que essa base seria vagamente referenciada no software desenvolvido ao final do projeto. Uma elaboração completa do roteiro fica listada como um dos possíveis próximos passos no desenvolvimento do software.

Desta forma, desenvolveu-se um enredo básico tratando de uma super heroína chamada de Aprendiz. No contexto do jogo, a Aprendiz é uma jovem heroína que tem uma impressionante capacidade de adquirir habilidades apenas tocando em pessoas que as possuem. Quando vê alguém executando uma manobra de skate, por exemplo, tocando no braço dessa pessoa por alguns segundos ela consegue imediatamente adquirir a habilidade de fazer essa manobra. O que já parecia um poder útil passa a se tornar necessário quando sua cidade começa a ser atacada misteriosamente por criaturas sobrenaturais e a jovem percebe que, ao tocar nas mesmas, mesmo após mortas, ela adquire as habilidades mágicas dessas criaturas. Agora ela deve utilizar as habilidades que já tem para conseguir novas e

descobrir quem está por trás dos ataques a sua cidade. E tudo isso deve ser conciliado com o dia a dia da vida normal da jovem Manuela, que felizmente até hoje conseguiu esconder seus poderes de todos ao seu redor. Além disso, o criador dos seres sobrenaturais é um vilão (que se revelaria em alguma das fases iniciais do jogo) chamado Professor. Na verdade, ele é justamente um dos professores de Manuela em seu colégio, a única pessoa a perceber os poderes da jovem e seu objetivo ao criar os seres e enviá-los para atacar a cidade é justamente prepará-la para ser uma super-heroína na cidade. Porém, algo que surge como boa intenção acaba se tornando exagerado e verdadeiramente perigoso para a cidade.

### 5.2.3.2 Finalização

A etapa de finalização do jogo em questão incluiu principalmente quesitos artísticos. Ela se resumiu a melhorias na interface, como inclusão de *assets* para representar os personagens envolvidos, efeitos sonoros de movimentação e ataque, bem como uma trilha sonora para o jogo. Também está incluída na finalização do jogo a resolução de *bugs* que surgiram ao longo da etapa de construção. Ao final desse processo, a última etapa realizada foi a exportação de um jogo para o executável, que pode ser facilmente distribuído para quem quiser jogar o jogo.



## 6 Conclusão

### 6.1 Sobre a aplicação da metodologia

De acordo com o que foi desenvolvido ao longo do [Capítulo 5](#), foi possível verificar que a aplicação da metodologia de desenvolvimento especifica no [Capítulo 4](#) tem pontos fortes, mas que, por questões que não haviam sido previstas quando da especificação inicial da metodologia, houver problemas no processo. Estes problemas nos levaram a elaborar uma versão revisada da metodologia, que será descrita mais adiante.

Dentre os pontos fortes, um de grande destaque está relacionado ao que foi relatado na [seção 5.1](#) e na [seção 5.2](#). Nessas seções, explicou-se que para iniciar o desenvolvimento foi decidido que o core do jogo estaria relacionado ao assunto de relacionamentos abusivos, porém, ao seguir a metodologia e tentar validar esta opção, chegou-se a conclusão de que este tema tinha uma série de problemas de resolução extremamente complexa, o que promoveu a decisão de trocar para um tema mais simples. Esse episódio ilustra que a metodologia permitiu uma decisão de mudança ainda bem no começo do desenvolvimento, isto é, não foi necessário "perder tempo" com prototipação e desenvolvimento da primeira ideia de tema para perceber que seria necessária uma mudança.

A validação do core também se mostrou bastante importante no desenvolvimento da segunda ideia de jogo (que foi a desenvolvida), como ilustrado na [subseção 5.2.1.2](#), que discorre sobre os testes feitos com o protótipo do jogo. Essa etapa de testes proporcionou algumas ideias que não haviam sido discutidas inicialmente sobre possíveis mecânicas para o jogo, trazendo *inputs* bastante interessantes para o processo de desenvolvimento do jogo.

Outro ponto especificado pela metodologia e que se provou bastante interessante foi o de desenvolver, logo no início da etapa de construção, ferramentas que ajudassem a encurtar o processo de desenvolvimento subsequente. No caso da nossa aplicação da metodologia, o exemplo que se encaixa neste ponto foi o desenvolvimento do gerador e editor de fases, que permitiu encurtar o tempo de desenvolvimento das fases, individualmente. Se tivéssemos mais tempo hábil, isso provavelmente teria se provado ainda mais interessante, uma vez que teria facilitado iterações de desenvolvimento, teste e validação de diferentes fases para o jogo.

Por outro lado, houve problemas na aplicação da metodologia; principalmente no que foi relatado na [seção 5.1](#) e na [seção 5.2](#). O primeiro, mais intrinsecamente ligado à metodologia, tem a ver com o foco do jogo. No caso do desenvolvimento de um jogo focado na narrativa, o desenvolvimento dessa narrativa deve vir até antes do desenvolvimento do core de *gameplay*; no caso do jogo sobre relacionamento abusivo, antes de validar a

primeira ideia de narrativa, partiu-se para o desenvolvimento de ideias de mecânica de jogo, o que se provou infrutífero. Entretanto, esse problema oferece um importantíssimo ponto a ser revisado na metodologia: a inclusão de elementos como a narrativa nas decisões tomadas na primeira etapa do desenvolvimento. O segundo problema, mais genérico ao desenvolvimento de *software* de maneira geral, é sobre o desenvolvimento de temas com o qual a equipe de desenvolvimento não tem tanto conhecimento e a necessidade, em tais casos, de envolver no projeto pessoas que tenham esse conhecimento desde o princípio; no caso do primeiro tema escolhido, nenhum dos participantes desse projeto passou em algum momento de sua vida pela dificuldade de estar em um relacionamento abusivo e, portanto, torna-se necessário o envolvimento de conhecedores do assunto desde os primórdios do desenvolvimento.

Assim, foi possível atestar que a metodologia definida no [Capítulo 4](#) tem alguns pontos de melhoria a serem atacados, mas que consegue, mesmo assim, atingir seus objetivos de melhorar o processo de desenvolvimento de jogos independentes.

## 6.2 Metodologia revisada

De acordo com o que foi estabelecido na seção anterior, faz sentido descrever uma metodologia revisada ao final do projeto. Resumidamente, a metodologia engloba as seguintes etapas:

- a) **Core:** essa etapa inclui a determinação dos elementos principais do jogo. Isto diz respeito ao *core loop*, como havia sido descrito em [Capítulo 4](#), mas, de acordo com os aprendizados supracitados, deve englobar também outros quesitos importantes a depender do tipo de jogo desenvolvido. Em um jogo cuja narrativa é importante, ela também deve ser determinada aqui, por exemplo. Essa etapa inclui, em seu ciclo de desenvolvimento, sub-etapas de hipótese, testes e validação;
- b) **Construção:** desenvolvimento do jogo, propriamente dito. De acordo com as premissas estabelecidas na etapa anterior, deve ser construído o jogo. Idealmente, os ciclos de desenvolvimento envolvidos nessa etapa devem ser ordenados de modo que nos primeiros ciclos sejam desenvolvidas ferramentas que ajudem a tornar os ciclos subsequentes mais curtos. Estes ciclos de desenvolvimento devem incluir etapas de hipótese, testes, validação e execução;
- c) **Polimento:** uma vez que o jogo esteja desenvolvido, devem ser feitas alterações pequenas, sem implementação de novas features, apenas corrigindo bugs e problemas de usabilidade, bem como preparando o jogo para o lançamento.



## 6.3 Próximos passos

No caso deste projeto, os próximos passos podem ser divididos em duas partes. A primeira delas seria inclusão de novos ciclos de desenvolvimento característicos do processo de construção, no qual novos elementos e mecânicas seriam adicionadas ao jogo. Algumas delas, de acordo com a pretensão inicial do projeto, deveriam estar prontas agora; entretanto, infelizmente não conseguimos encaixar o desenvolvimento das mesmas no calendário estabelecido. Elas incluem: elaboração de um contingente maior de fases; implementação de sistema de escolhas nos diálogos, que mudariam características das batalhas de acordo com opções do jogador (alguns exemplos de diálogos com essa funcionalidade chegaram a ser escritos e estão reproduzidos no [Apêndice C](#)). Outros possíveis próximos passos discutidos incluem, principalmente, extensões que aumentariam o engajamento dos jogadores por meio do aumento de tempo de jogo, como por exemplo implementação de novos modos de jogo, como um modo *multiplayer*; elaboração de um roteiro mais detalhado, com começo, meio e fim, para a história do jogo; inclusão de objetivos opcionais nas fases e colecionáveis espalhados pelo jogo.

Além disso, outros possíveis próximos passos dizem respeito à etapa de polimento e estão mais relacionadas com o lançamento do jogo em si, algo que não faz sentido no escopo desse projeto, uma vez que o intuito não era desenvolver um jogo a ser de fato lançado no mercado. Esses próximos passos incluiriam campanha de marketing, o que inclui elaboração de material de divulgação como *teasers*, *trailers*, artes de divulgação e o lançamento, de fato, do jogo em lojas eletrônicas como a Steam ou a Playstation Store.



# Referências

BLANK, S. G. *Do sonho à realização em 4 passos*. 1. ed. São Paulo: Editora Évora Ltda., 2012. Citado na página 31.

CHALK, A. *Spiderweb Software's latest crappy-looking RPG is now live*. 2019. Internet. Disponível em: <<https://www.pcgamer.com/spiderweb-softwares-latest-crappy-looking-rpg-is-now-live/>>. Acesso em: 05 dez. 2020. Citado na página 23.

CRAFTING a Strong Core Loop. In: THE Free to Play Game Design Bible. mobilefreetoplay, 2020. Disponível em: <<https://mobilefreetoplay.com/bible/crafting-strong-core-loop/>>. Acesso em: 02. dez 2020. Citado na página 25.

DEALESSANDRI, M. *What is the best game engine: is Unity right for you?* 2020. Internet. Disponível em: <<https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-unity-the-right-game-engine-for-you>>. Acesso em: 23 nov. 2020. Citado na página 42.

FABRO, F. *Final Fantasy XII's Gambit System was like programming for beginners*. 2019. Internet. Disponível em: <<https://dev.to/fihra/final-fantasy-xii-s-gambit-system-was-like-programming-for-beginners-7d1>>. Acesso em: 06 dez. 2020. Citado na página 48.

GRAETZ, J. M. *The origin of Spacewar*. 1981. Internet. Disponível em: <<https://www.wheels.org/spacewar/creative/SpacewarOrigin.html>>. Acesso em: 02 dez. 2020. Citado na página 19.

HISTORY of videogames. In: WIKIPÉDIA, a Enciclopédia Livre. Wikimedia, 2020. Disponível em: <[https://en.wikipedia.org/wiki/History\\_of\\_video\\_games](https://en.wikipedia.org/wiki/History_of_video_games)>. Acesso em: 02. dez 2020. Citado na página 19.

KLEI Entertainment. In: WIKIPÉDIA, a Enciclopédia Livre. Wikimedia, 2020. Disponível em: <[https://en.wikipedia.org/wiki/Klei\\_Entertainment](https://en.wikipedia.org/wiki/Klei_Entertainment)>. Acesso em: 05. dez 2020. Citado na página 23.

LUNNING, J. *Capcom Gave Up On 'Mega Man Battle Network,' But Thomas Moon Kang Never Did*. 2019. Internet. Disponível em: <<https://www.newsweek.com/mega-man-battle-network-kickstarter-one-step-eden-1428425>>. Acesso em: 17 nov. 2020. Citado na página 35.

MINIMAX. In: WIKIPÉDIA, a Enciclopédia Livre. Wikimedia, 2020. Disponível em: <<https://en.wikipedia.org/wiki/Minimax>>. Acesso em: 06. dez 2020. Citado na página 48.

MONTE Carlo tree search. In: WIKIPÉDIA, a Enciclopédia Livre. Wikimedia, 2020. Disponível em: <[https://en.wikipedia.org/wiki/Monte\\_Carlo\\_tree\\_search](https://en.wikipedia.org/wiki/Monte_Carlo_tree_search)>. Acesso em: 06. dez 2020. Citado na página 48.

- NEWZOO. *Global games market report*. 2019. Internet. Disponível em: <[http://resources.newzoo.com/hubfs/Reports/Newzoo\\_2019\\_Global\\_Games\\_Market\\_Report\\_Press\\_Copy\\_v2.pdf](http://resources.newzoo.com/hubfs/Reports/Newzoo_2019_Global_Games_Market_Report_Press_Copy_v2.pdf)>. Acesso em: 02 dez. 2020. Citado na página 20.
- PARHAM, J. *The Liquid Fire*. 2015. Internet. Disponível em: <<http://theliquidfire.com/>>. Acesso em: 23 nov. 2020. Citado 2 vezes nas páginas 42 e 43.
- PARHAM, J. *Tactics RPG Conversations*. 2015. Internet. Disponível em: <<http://theliquidfire.com/2015/06/29/tactics-rpg-conversations/>>. Acesso em: 26 nov. 2020. Citado na página 46.
- REICHERT, K. *Top 5 Problems Faced By Indie Game Developers*. 2011. Internet. Disponível em: <[https://www.gamasutra.com/blogs/KateReichert/20121101/180767/Top\\_5\\_Problems\\_Faced\\_By\\_Indie\\_Game\\_Developers.php](https://www.gamasutra.com/blogs/KateReichert/20121101/180767/Top_5_Problems_Faced_By_Indie_Game_Developers.php)>. Acesso em: 06 dez. 2020. Citado na página 17.
- RIES, E. *A Startup Enxuta*. 1. ed. Rio de Janeiro: GMT Editores Ltda., 2011. Citado na página 30.
- TRAN, E. *Into The Breach Review: A Mechanized Masterpiece*. 2018. Internet. Disponível em: <<https://www.gamespot.com/reviews/into-the-breach-review-a-mechanized-masterpiece/1900-6416865/>>. Acesso em: 17 nov. 2020. Citado na página 36.
- VIDEO game industry. In: WIKIPÉDIA, a Enciclopédia Livre. Wikimedia, 2020. Disponível em: <[https://en.wikipedia.org/wiki/Video\\_game\\_industry](https://en.wikipedia.org/wiki/Video_game_industry)>. Acesso em: 02 dez 2020. Citado na página 19.
- WAWRO, A. *Devs share real talk about surviving the latest 'indiepocalypse'*. 2016. Internet. Disponível em: <[https://www.gamasutra.com/view/news/268134/Devs\\_share\\_real\\_talk\\_about\\_surviving\\_the\\_latest\\_indiepocalypse.php](https://www.gamasutra.com/view/news/268134/Devs_share_real_talk_about_surviving_the_latest_indiepocalypse.php)>. Acesso em: 02 dez. 2020. Citado na página 20.
- WILBUR, B. *Getting the message right in Another Lost Phone*. 2017. Internet. Disponível em: <[https://www.gamasutra.com/view/news/306729/Getting\\_the\\_message\\_right\\_in\\_Another\\_Lost\\_Phone.php](https://www.gamasutra.com/view/news/306729/Getting_the_message_right_in_Another_Lost_Phone.php)>. Acesso em: 06 dez. 2020. Citado na página 34.

# Apêndices



# APÊNDICE A – Texto do Termo de Consentimento Livre e Esclarecido

Prezado(a) Senhor(a),

Esta pesquisa está relacionada ao desenvolvimento de um jogo digital e está sendo desenvolvida pelos alunos Guilherme Adissy e Rafael Barone do curso de Engenharia de Computação da Escola Politécnica da Universidade de São Paulo (EPUSP), sob orientação do Professor Ricardo Nakamura.

O objetivo do estudo é testar o protótipo de jogo desenvolvido pelos alunos que será implementado pelos mesmos futuramente como parte do trabalho de conclusão de curso. Vale ressaltar que não está sendo feito nenhum teste relativo à capacidade dos envolvidos, o teste é relacionado exclusivamente ao protótipo.

Solicitamos a sua colaboração para participar de um teste com o protótipo bem como de uma entrevista gravada ao final do teste, e também a sua autorização para apresentar os resultados deste estudo durante a apresentação do nosso trabalho. Seu nome e imagem só serão utilizados para fins de estudo e demonstração do projeto.

Esclarecemos que sua participação no estudo é voluntária e, portanto, o(a) senhor(a) não é obrigado a fornecer informações e/ou colaborar com as atividades detalhadas pelos alunos caso não queira. Caso decida não participar do estudo, ou resolva, a qualquer momento, desistir da participação, não sofrerá nenhum dano ou consequência jurídica. Os alunos estarão a sua disposição a qualquer momento para esclarecer quaisquer dúvidas que possam vir a surgir.

Considerando que fui informado(a) dos objetivos e da motivação do estudo proposto, de como será a minha participação e dos procedimentos envolvidos no mesmo, declaro meu consentimento livre e esclarecido em participar da pesquisa, declarando também estar de acordo com a divulgação e utilização dos dados obtidos nos moldes supracitados.





## APÊNDICE B – Roteiro de entrevistas após teste com protótipo

O roteiro estruturado consiste de duas partes: as perguntas a serem feitas para o jogador (apresentando entre parentêses o objetivo da pergunta, quando aplicável) e os pontos de atenção, que não são mencionados para o entrevistado, servindo apenas para guiar a análise da experiência do jogador e da entrevista. Dito isso, as perguntas eram:

- a) Qual(s) emoção vc sentiu jogando? (Investigar melhor o sentimento)
- b) Qual foi a sua fase predileta? Por quê? (Entender sobre o level design)
- c) Teve alguma fase que não gostou? Se sim, quais?
- d) Se tivessem mais fases, você jogaria?
- e) Se a pessoa não tiver chegado ao final do jogo: o que lhe fez desistir de jogar?
- f) O que mais gostou?
- g) O que não gostou?
- h) O que poderíamos melhorar?
- i) O que você sentiu falta?
- j) Existe alguma informação que seria importante mostrar o tempo todo para o jogador?

Além disso, os principais pontos de atenção eram:

- a) Quais informações o jogador perguntou?
- b) Quantas vezes a pessoa tentou cada fase
- c) Se o jogador desistiu de alguma fase
- d) Ramp-up das mecânicas do jogo



## APÊNDICE C – Transcrição dos diálogos desenvolvidos para o jogo

- a) Manuela tem uma prova de biologia amanhã, deseja ficar até tarde estudando para a prova? Opções:
- Sim (penalização de movimentação - cansaço; bonificação de ataque - aprendeu mais sobre biologia e sabe onde são os pontos que causam mais dor). Mensagem: 'Parabéns! Manuela estudou e foi super bem na prova, agora ela sabe um pouco mais de biologia e entende melhor os corpos dos inimigos, conseguindo causar mais dano neles. Porém, por ter ficado até tarde acordada, Manuela está cansada e sua movimentação está um pouco mais lenta.
  - Não (possível penalização - tristeza pq foi mal na prova). Mensagem: 'Infelizmente Manuela foi mal na prova e ficou bem chateada com isso, assim ela está menos concentrada e seus ataques um pouco menos efetivos. Mas tudo bem, ela precisa estar descansada para proteger a cidade!'
- b) Hoje é aniversário de uma amiga de manuela, num boliche, deseja ir? Opções:
- Sim (bonificação de ataque a distância - melhorou mira jogando boliche). Mensagem: "A festa foi demais! E de quebra, Manuela treinou sua pontaria e seus ataques a distância estão ainda mais efetivos!'
  - Não (bonificação de movimentação - descansou melhor). Mensagem: 'Com grandes poderes, vêm grandes responsabilidades. Foi uma boa ficar descansando para estar preparada pra próxima batalha!'
- c) Uma garota popular do colégio está aborrecendo Manuela, deseja confrontá-la? Opções:
- Sim (penalização, Manuela teve que ficar em detenção no colégio até mais tarde e está cansada - mas descarregar a raiva foi bom e ela tem uma bonificação de ataque). Mensagem: 'Manuela discutiu com a garota e parece que por enquanto ela vai parar de incomodar, o que deixou Manuela muito feliz, fazendo com que seu ataque esteja ainda mais efetivo. Mas um monitor do colégio viu a discussão e colocou as duas em detenção, Manuela teve que ficar até mais tarde no colégio e está cansada, o que a deixa mais lenta'
  - Não (Sem bonificação nem penalização). Mensagem: 'É, não vale a pena ficar brigando e se desgastando por bobeira. . . mas aquela garota é bem irritante.'
- d) Manuela tem uma aula de educação física opcional hoje, deseja participar? Opções:

- Sim (bonificação de esquiva, no próximo combate ao ser atacada, existe uma chance de Manuela conseguir esquivar e não sofrer dano). Mensagem: 'A aula foi ótima e ajudou a melhorar o preparo físico de Manuela! Agora, quando ela for atacada, talvez ela até consiga se esquivar dos seus inimigos!'
  - Não (Sem bonificação nem penalização). Mensagem: 'A aula até deve ter sido legal, mas provavelmente só ia deixar Manuela cansada sem ajudar em nada na luta contra as criaturas'
- e) A mãe de Manuela perguntou se ela poderia ir ao mercado fazer algumas compras, deseja aceitar? Opções:
- Sim (Manuela compra um energético e tem bonificação de movimentação, mas perde acurácia em seus ataques). Mensagem: 'Manuela foi ao mercado e, além do que sua mãe pediu, comprou uma latinha de energético! Agora ela está até tremendo de tanta energia, vai estar ainda mais rápida... mas talvez tenha um pouco de dificuldade pra mirar em seus ataques'
  - Não (A mãe de Manuela fica chateada com a filha e briga com ela, Manuela fica chateada e os ataques mais fracos). Mensagem: 'Não há tempo para ir ao mercado quando se está lutando contra o crime! Mas a mãe de Manuela ficou chateada e brigou com ela, danificando a concentração de Manuela e tornando seus ataques menos efetivos'
- f) Hoje tem uma festa na casa de um amigo de Manuela e ela quer muito ir, mas seu pai disse que ela já tem ido em muitas festas ultimamente... deseja insistir? Opções:
- Sim (O pai de manuela deixa ela ir, ela vai e fica até tarde, então está bem cansada no outro dia e sofre penalização de movimentação). Mensagem: 'O pai finalmente deixou! Manuela foi à festa e se divertiu demais, mas estará um pouco cansada e mais lenta no dia seguinte'
  - Não (Ela desistiu de ir na festa e ficou chateada com isso, seus ataques estarão mais fracos). Mensagem: 'Melhor não ir mesmo, se Manuela insistisse, seu pai provavelmente ficaria bravo. Mas Manuela ficou chateada de perder a festa e sua concentração e seu ataque estarão um pouco piores amanhã'