UNIVERSIDADE DA BEIRA INTERIOR
Artes e Letras

Elevação da Narrativa Através de Mecânicas de Jogabilidade

João Luís Fernandes Caseiro

Dissertação para obtenção de Grau de Mestre em
Design e Desenvolvimento de Jogos Digitais
(2º ciclo de estudos)

Orientador: Prof. Doutor Frutuoso Gomes Mendes da Silva

Covilhã, Janeiro de 2019

Dedicatória

Quero dedicar esta dissertação a todas as pessoas que acreditam que os videojogos são meios eficazes não só para entreter, como também para educar, fazer refletir, e contar histórias num contexto mais pessoal, devido à sua natureza interativa.

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao professor doutor Frutuoso Gomes Mendes da Silva, que apesar do meus atrasos e absentismos no desenvolvimento da dissertação, esteve sempre prontamente disponível para me guiar e mostrar-me como melhorar os meus estudos e a construção do projeto.

Em seguida, queria agradecer a todos os professores do Mestrado em Design e Desenvolvimento de Jogos Digitais na Universidade da Beira Interior, que tiveram a coragem de dedicar tempo e esforço na fundação e preservação deste curso, sabendo os preconceitos que ainda perduram sobre os videojogos, mas que mesmo assim acreditam ser um meio digno de ser estudado e respeitado.

Também merecedor de muitos agradecimentos são os meus pais, que ao longo da minha licenciatura e mestrado, apostaram e acreditaram em mim, mesmo quando falhei de ir a encontro das suas expectativas, e sem os quais não seria possível ser a pessoa que sou hoje.

Sem esquecer, quero também agradecer a todos os meus colegas académicos, que proporcionaram-me fantásticas experiências ao longo dos anos em que fui um estudante universitário, tornando estes anos em algo inesquecível e que para sempre marcarão a minha vida.

Em último mas certamente não menos importante, queria agradecer à minha colega de projeto final e melhor amiga Mafalda Claro, que foi a pessoa que mais aturou o meu feitio chato e infantil, que foi a arquiteta do projeto final, que sem ela não ela possível estar formado, e que sempre foi um grande apoio nos momentos mais desmotivantes que um trabalho naturalmente traz. Este estudo e projeto não é só meu mas também dela.

Resumo

Graças a ser um meio relativamente jovem, os videojogos estão a encarar problemas em relação a como os seus elementos devem estar interconectados e como devem ser apresentados aos seus jogadores. É comum ver-se uma separação de valor entre a jogabilidade e a história dum videojogo, mesmo ao ponto de se acreditar que investir num ofusca o outro, não sendo possível ambos trabalharem em conjunto. Tal crença existe maioritariamente devido ao método mais comum de apresentar uma história num videojogo ser através de meios não-interativos, como por exemplo as *cutscenes*. Ao usarem-se métodos não-interativos num videojogo, tal é habitualmente visto como um menosprezo à natureza interativa do meio e que todo o seu potencial não está a ser utilizado, adicionalmente criando uma maior separação entre os dois elementos. Com base nisto, nesta dissertação, uma solução é procurada, primeiro olhando mais aprofundadamente para o problema, analisando as vantagens e desvantagens de usar narrações não-interativas, lendo discussões entre académicos que defendem cada lado entre a narratologia e a ludologiam e finalmente desenvolvendo um videojogo de demonstração com a intenção de apresentar métodos interativos capazes de contar uma história de uma forma a permitir que o jogador tenha uma melhor experiência narrativa.

Palavras-chave

Cutscenes, Narrativa, Lodologia, Interativo, *Flow*, Suspensão de Descrença

Abstract

Thanks to being a relatively young medium, videogames still face problems regarding how its element should be interconnected and presented to the players. It's common to see a separation of value between the gameplay and the story of a videogame, because it is believed that the investment in one overshadows the other, not being able to work together. Such belief exists mainly because the most common way to present a story in a videogame is through non-interactive ways, such as cutscenes. When using non-interactive methods in a videogame, it is often seen as a disregard to the interactive nature of the medium and that it is not being used to its full potential, further creating a divide between the two elements. And so, in this dissertation, a solution is searched, firstly by looking deeper into the problem, analyzing the advantages and disadvantages of using non-interactive narration, reading into the discussion between academics that defend each side of narratology and ludology, and then developing a videogame demo that intends to present interactive methods capable of telling a story in a way that allows the player to have a better narrative experience.

Keywords

Cutscenes, Narrative, Ludology, Interactive, Flow, Suspension of Disbelief

Conteúdo

Dedicatória iii

Agradecimentos v

Resumo vii

Abstract ix

Conteúdo xi

Lista de Figuras xiii

Lista de Tabelas xvii

Lista de Acrônimos xix

1 Introdução 1

1.1 Objetivos . . . 2
1.2 Metodologias . . . 2
1.2.1 Método de Estudo . . . 2
1.2.2 Limites de Estudo . . . 3
1.3 Estrutura da Dissertação . . . 4

2 Estado da Arte 5

2.1 Introdução . . . 5
2.1.1 Jogos . . . 5
2.1.2 Videojogos . . . 9
2.1.3 Narrativa . . . 11
2.2 Dificuldade de Ligação entre Narrativa e Jogabilidade . . . 14
2.2.1 Ludologia vs. Narratologia . . . 14
2.2.2 Jogabilidade vs. Narrativas explícitas . . . 16
2.3 Importância da Narrativa na Jogabilidade . . . 21
2.3.1 *Flow* e Suspensão de Descrença . . . 21
2.3.2 Limitação Parcial da Jogabilidade . . . 23
2.3.3 Emoção e Sentimento . . . 24
2.4 Conclusão . . . 27

3 Projeto 29

3.1 Pré-produção . . . 29
3.2 Ferramentas Utilizadas . . . 30
3.2.1 *Unity* . . . 30

3.2.2 *$P Point-Cloud Recognizer* 31
3.2.3 *Yarn Spinner* 32
3.3 Desenvolvimento do Protótipo 32
3.3.1 Estética 32
3.3.2 Movimento 35
3.3.3 Desenho 37
3.3.4 Diálogo 38
3.3.5 Design de Níveis 41
3.3.6 Puzzles e eventos 43
3.3.7 Interface 46
3.4 Resultado Final 48

4 Conclusão **53**
4.1 Dificuldades Encontradas 53
4.2 Passos Futuros 54

Bibliografia **55**

A Código desenvolvido **61**

B CD com materiais extra **83**
B.1 Conteúdo do CD 83

Lista de Figuras

2.1 Primeiros videojogos. 10
2.2 Cópia de *sprite* com coloração diferente devido às limitações computacionais. . . 11
2.3 Estrutura de três atos. 13
2.4 *Cutscene* no jogo *Pac-man*. 17
2.5 Interação entre duas personagens numa *cutscene*. 17
2.6 Eixos que representam os elementos que os três meios narrativos introduzem. . . 18
2.7 Gráfico representante da redução de interatividade quando *cutscenes* entram em cena. 18
2.8 Cena onde Lara se encontra junta ao veado após o matar. 20
2.9 Gráfico simples do conceito de *flow*. 22
2.10 Protagonista do jogo *Journey* a caminhar na neve. 24
2.11 Roda das Emoções de Plutchik. 26
2.12 Momento no final do prólogo do jogo *The Last of Us* (Naughty Dog, 2013). 27

3.1 Uso da mecânica de desenho para a criação de escadas. 30
3.2 Demonstração do *$P Point-Cloud Recognizer* no *website* oficial da ferramenta. (Wobbrock, 2012) 31
3.3 Interface do *Yarn*. 32
3.4 Ambiente de estilo livro pop-up no videojogo *Tengami*. 34
3.5 Mostra de três faixas distintas onde o jogador se consegue mover no videojogo *Little Big Planet*. (gamespy.com, 2008) 34
3.6 Camadas de adereços. 35
3.7 Sequência de *frames* no salto do Mario no videojogo *Super Mario Bros.* (Nintendo, 1985). 36
3.8 Sequência de *frames* no salto da personagem no Ink Tales. 36
3.9 Detetor de vértices numa fase inicial do projeto. 37
3.10 Reconhecimento de desenho e posterior substituição por um objeto. 39
3.11 Escolhas de falas no videojogo *Oxenfree* (engadget.com, 2016) 40
3.12 Variação do número de balões de fala no *Ink Tales*. 41
3.13 Concept art do nível demonstrativo do *Ink Tales*, desenhado por Mafalda Claro. . 41
3.14 Zona inicial do nível criado. 42
3.15 Zona final do nível criado. 42
3.16 Elemento de interface que indica a possibilidade de roubar o mel. 44
3.17 Estilo gráfico dos objetos que estão na posse da protagonista. 47
3.18 Diferença da perspetiva da câmara quando esta foca a personagem, e quando foca uma porção do nível. 48
3.19 Área inicial onde o jogador começa. 49

3.20 Entrada da aldeia e área de interação com o guarda. 49
3.21 Rio da aldeia e introdução ao *platforming*. 49
3.22 Roulotte do urso onde é possível adquirir o mel. 50
3.23 Praça principal da aldeia onde se situa o mercado e onde é possivel falar com múltiplos aldeões. 50
3.24 Moinho de vento no topo da colina onde é possível adquirir a farinha. 50
3.25 Casa dos sete cabritinhos onde ocorre o incidente que inicia a aventura do videojogo. 51
3.26 Entrada da torre do corvo onde é adquirido o poder de desenhar. 51
3.27 Plataformas trespassáveis e topo da torre do corvo. 51

A.1 Script encarregue de desativar a barreira que impede o lobo de atravessar a ponte quando o lobo oferece mel ao guarda. 61
A.2 Parte 1 do script que controla a orientação da câmara em todos os *frames*. . . . 61
A.3 Parte 2 do script que controla a orientação da câmara em todos os *frames*. . . . 62
A.4 Script que torna o corvo visível após o evento da fuga dos sete cabritinhos. Antes disso, o corvo encontra-se oculto. 62
A.5 Parte 1 do script que verifica a proximidade do jogador a um NPC e que faz a ligação ao sistema de diálogo. 63
A.6 Parte 1 do script que verifica a proximidade do jogador a um NPC e que faz a ligação ao sistema de diálogo. 63
A.7 Parte 1 do script que permite desenhar no espaço de jogo, que envia os dados para o sistema de reconhecimento de desenhos e que trata os dados recebidos desse mesmo sistema para imediatamente substituir por uma figura com dimensões correspondentes. 64
A.8 Parte 2 do script que permite desenhar no espaço de jogo, que envia os dados para o sistema de reconhecimento de desenhos e que trata os dados recebidos desse mesmo sistema para imediatamente substituir por uma figura com dimensões correspondentes. 65
A.9 Parte 3 do script que permite desenhar no espaço de jogo, que envia os dados para o sistema de reconhecimento de desenhos e que trata os dados recebidos desse mesmo sistema para imediatamente substituir por uma figura com dimensões correspondentes. 66
A.10 Script que ativa quando o jogador salta da torre do corvo, sinalizando o final da demonstração do videojogo através de um desvanecimento para branco, com uma frase de agradecimento de participação. 67
A.11 Script que inicia o evento final onde os cordeiros fojem de casa, os aldeões se deslocam de modo a bloquear a passagem do jogador para partes anteriores do nível, forçando-o a ir de encontro ao corvo, que também se torna visível a partir desse instante. 68

A.12 Script que permite a recolha da farinha e a remoção do objeto representante da farinha dentro da cena de jogo. 69

A.13 Script que permite a recolha do mel e a remoção do objeto representante do mel dentro da cena de jogo. 70

A.14 Script que permite ao jogador agarrar e arrastar desenhos que faça, e impede a viragem do sprite da personagem principal quando esta está a puxar algo na sua direção. 71

A.15 Parte 1 do script que está encarregue de tornar visível a interface que informa o jogador dos objetos que tem em posse, e que torna esses ícones translúcidos quando a personagem jogável se encontra em movimento. 72

A.16 Parte 2 do script que está encarregue de tornar visível a interface que informa o jogador dos objetos que tem em posse, e que torna esses ícones translúcidos quando a personagem jogável se encontra em movimento. 73

A.17 Parte 3 do script que está encarregue de tornar visível a interface que informa o jogador dos objetos que tem em posse, e que torna esses ícones translúcidos quando a personagem jogável se encontra em movimento. 74

A.18 Script que permite à câmara mudar de posição e desligar-se do jogador quando este entra a área correspondente ao mercado, e que faz a cªamara voltar ao seu posicionamento predefinido quando o jogador sai dessa área. 74

A.19 Script Que cria uma área de trigger que desativa a colisão duma plataforma correspondente, permitindo ao jogador que a trespasse de baixo para cima, mas que a torna sólida quando o jogador já se encontra acima dela, resultando numa plataforma trespassável. Permite também que o jogador passe de cima para baixo quando este carrega em simultâneo no botão correspondente de baixo e de salto. 75

A.20 Script gestor da interface ligada ao personagem jogável e que contém os balões de fala do jogador. Também torna possível o efeito de flutuação que cada balão de fala tem individualmente. 76

A.21 Parte 1 do script encarregue do movimento da personagem em todos os eixos, incluindo o afecto da gravidade, o deslize, a anulação de força de ascensão vertical quando o jogador bate em algum teto, e a deteção de uma plataforma trespassável nos pés da personagem. Também está encarregue de toda a animação que o sprite possui, estando ligado ao animator do Unity. 77

A.22 Parte 2 do script encarregue do movimento da personagem em todos os eixos, incluindo o afecto da gravidade, o deslize, a anulação de força de ascensão vertical quando o jogador bate em algum teto, e a deteção de uma plataforma trespassável nos pés da personagem. Também está encarregue de toda a animação que o sprite possui, estando ligado ao animator do Unity. 78

A.23 Parte 3 do script encarregue do movimento da personagem em todos os eixos, incluindo o afecto da gravidade, o deslize, a anulação de força de ascensão vertical quando o jogador bate em algum teto, e a deteção de uma plataforma trespassável nos pés da personagem. Também está encarregue de toda a animação que o sprite possui, estando ligado ao animator do Unity. 79
A.24 Parte 4 do script encarregue do movimento da personagem em todos os eixos, incluindo o afecto da gravidade, o deslize, a anulação de força de ascensão vertical quando o jogador bate em algum teto, e a deteção de uma plataforma trespassável nos pés da personagem. Também está encarregue de toda a animação que o sprite possui, estando ligado ao animator do Unity. 80
A.25 Parte 5 do script encarregue do movimento da personagem em todos os eixos, incluindo o afecto da gravidade, o deslize, a anulação de força de ascensão vertical quando o jogador bate em algum teto, e a deteção de uma plataforma trespassável nos pés da personagem. Também está encarregue de toda a animação que o sprite possui, estando ligado ao animator do Unity. 80
A.26 Script encarregue de levantar e baixar adereços do cenário conforma a proximidade do jogador, adicionalmente sendo possível controlar a velocidade de ascensão e descensão. 81
A.27 Script que faz as pás do moínho andarem à roda. 81
A.28 Script que faz a testura da água mover-se numa determinada direção, criando a ilusão de um rio no ambiente de jogo. 81

Lista de Tabelas

2.1 Elementos de definição de um jogo. 7

Lista de Acrônimos

UBI	Universidade da Beira Interior
NPC	Non-Playable Character (Personagem Não-Jogável)

Capítulo 1

Introdução

No início dos anos 50, os videojogos surgiram como um simples meio de demonstração tecnológica de uma forma lúdica. Começando por serem desenvolvidos em ambientes académicos numa altura onde a capacidade computacional era bastante reduzida comparada à de hoje em dia, todos esses jogos focavam-se somente nas mecânicas de jogabilidade, ignorando completamente a construção de um contexto narrativo.

Passadas duas décadas, os videojogos entraram no mercado, com o aparecimento de consolas e máquinas *arcade*, e jogos fabricados para estas começaram a apresentar algo que antes não existia no meio, uma narrativa. Mesmo que breve e limitada na altura, uma narrativa servia para motivar o utilizador a jogar, dando-lhe um propósito, i.e. uma recompensa, à superação de obstáculos que os jogos inerentemente têm. Um dos primeiros exemplos é o jogo de *arcade* *Donkey Kong* (Nintendo, 1981), onde é possível ver, ao iniciar, um macaco gigante a raptar a pretendente do protagonista controlado pelo jogador e a rir-se após o ter feito, o que pode ser interpretado não só como uma provocação à personagem jogável, como também ao jogador. Ao longo do jogo, a personagem que precisa de ser salva exclama por ajuda, motivando o utilizador a continuar a jogar até ao último nível, o qual, caso vença, mostra o macaco gigante a cair da torre onde mantinha a refém, enquanto o protagonista e a sua pretendente se reúnem novamente. Apesar de simples, a narrativa presente no jogo era algo único na altura, pois muitos outros videojogos da altura, como *Space Invaders* (Taito, 1978) e *Pac-Man* (Namco, 1980), punham o jogador num ambiente jogável, sem contexto narrativo em toda a sua duração.

Desde então, as narrativas presentes nos jogos foram-se aprimorando ao longo dos anos, parcialmente graças à rápida evolução das tecnologias de informação, as quais os videojogos usam como fundação. Esse progresso tecnológico fez com que vários desenvolvedores de jogos explorassem técnicas para a criação de narrativas mais complexas, e foi no cinema que eles maioritariamente se inspiraram, pois é a linguagem narrativa mais próxima dos jogos, devido a usarem técnicas audiovisuais para transmitir a grande maioria de informação ao espectador.

Consequentemente, começaram a surgir vários jogos como *Metal Gear Solid* (Konami, 1998) e *Uncharted* (Naughty Dog, 2007) que oferecem experiências bastante cinematográficas através de *cutscenes*, momentos automatizados onde o jogador não tem controlo sobre a personagem que normalmente opera, e que geralmente é usada como ferramenta de avanço narrativo. No

entanto, o uso desta técnica apresenta problemas, pois cria um distanciamento entre o progresso de narrativa e momentos de jogabilidade onde a história do jogo não avança. É com base neste problema que esta dissertação se desenvolveu, onde se procurou juntar jogabilidade e narrativa de uma forma mais uniforme, e mostrar como isso pode beneficiar a experiência do jogador.

1.1 Objetivos

Esta dissertação teve como objetivo principal o desenvolvimento de um jogo-projeto onde foram implementadas técnicas que favorecem o que é conhecido como harmonia ludonarrativa. Este termo é o oposto a dissonância ludonarrativa, um termo originalmente criado por Clint Hocking (Hocking, 2007), e define-se por um conflito entre a mensagem que a narrativa transmite, e a que a jogabilidade promove, criando um choque entre as duas que pode quebrar a imersão do jogador e afetar a experiência geral do jogo. Foi com base neste objetivo que se chegou à questão-chave da dissertação:

Como é que mecânicas de *gameplay* podem promover uma experiência narrativa?

Na tentativa de se responder a esta pergunta, foi feito um estudo inicial não só sobre a origem dos videojogos e da narrativa, como às suas naturezas antagónicas, o que faz com que a união entre ambas seja difícil. Ainda no estudo teórico, foram apresentadas razões pelas quais é importante a existência de uma narrativa num videojogo, como esta pode criar uma experiência única ao jogador, exclusiva a este meio.

Ao longo da dissertação foram também apresentados jogos presentes no mercado, alguns servindo de exemplo de dissonância ludonarrativa — sendo apresentadas as razões que o fazem ganhar essa característica — e outros de exemplo de harmonia ludonarrativa, aos quais serão apontados os elementos que no final se tornaram referências para o projeto prático.

1.2 Metodologias

1.2.1 Método de Estudo

Começando pelo estudo teórico, foram lidos vários textos académicos, tanto antigos — na busca de conhecimento sobre meios lúdicos e narrativos clássicos — como estudos mais recentes, não limitados ao meio escrito, expandindo-se a outros como o audiovisual, usados na pesquisa de

temas mais recentes, devido à relativa juventude dos videojogos e aos estudos sobre eles.

Na escolha de videojogos de exemplo, usou-se o conhecimento do autor sobre títulos com os quais o mesmo já está familiarizado e jogos analisados em outros estudos sobre os temas contidos nesta dissertação.

Na construção do jogo-projeto, foi usado um motor de desenvolvimento de jogos gratuito, e, para além do conteúdo desenvolvido pela equipa, foram utilizados vários *assets* [1], disponíveis na Internet e de uso livre em produtos comerciais.

1.2.2 Limites de Estudo

Com milhares de títulos lançados ao longo dos anos e dezenas de géneros de jogo existentes, mostrar-se-ia uma demanda impossível procurar em todos eles fortes exemplos de dissonância ou harmonia ludonarrativa. Por tais causas, achou-se necessário criar um limite de estudos de jogos-exemplo, sendo o primeiro ponto a idade que o jogo deve ter. Como a tecnologia evoluiu drasticamente desde o início do século e como os videojogos se aproveitam das novas tecnologias desenvolvidas, seria injusto comparar a apresentação narrativa entre um jogo dos anos 90 e um dos tempos de hoje. Assim sendo, serão só escolhidos jogos lançados até dez anos atrás à escrita desta dissertação.

O género de jogo é outro ponto a ter em atenção, pois as experiências que estes oferecem aos seus utilizadores pode diferenciar bastante se os seus géneros forem muito distintos. É impossível um videojogo de corridas frenéticas como *The Crew* (Ivory Tower, 2014) oferecer a mesma experiência que um jogo de gestão de uma quinta como *Stardew Valley* (ConcernedApe, 2016). De acordo com o jogo-projeto desenvolvido, serão analisados videojogos do género aventura, plataforma ou ambos.

Apesar de os videojogos já conseguirem conter narrativas complexas como um ponto forte do produto, há jogos ainda hoje que escolhem focar-se maioritariamente em mecânicas e deixam a narrativa em plano secundário. Nestes jogos, haver uma discrepância entre estas duas componentes não é tão reprovador pois não é o que os desenvolvedores destacam no produto como fator de venda. Só videojogos com um grande foco na narrativa é que serão avaliados nesta dissertação.

Como último ponto, é importante apontar que apesar de o mercado dos videojogos ser global, muitos títulos são exclusivos de certas regiões, alguns possuindo até barreiras linguísticas, sendo mais notória a diferença entre o mercado ocidental e oriental, e estando o autor numa área de

[1] Recursos já desenvolvidos para rápida aplicação em motores de desenvolvimento de jogos

mercado ocidental no globo, a exploração de videojogos está limitada a essa área.

Com base na informação apresentada, podem-se resumir os limites de estudo aos seguintes pontos:

- Lançado no máximo há dez anos atrás;
- Mesmo género de jogo (*platformer* e/ou aventura);
- Possuir uma intenção narrativa;
- Estar acessível nos mercados aos quais o autor tem acesso.

1.3 Estrutura da Dissertação

Para além deste primeiro capítulo que serviu como uma introdução ao tema e à pergunta chave à qual se pretende responder, podemos contar com mais 3 capítulos.

O segundo capítulo reflete sobre o os estudos académicos sobre quais o projeto se apoia, incluindo uma introdução contextual dos conceitos base de videojogo e narrativa.

O terceiro capítulo contém o relatório do desenvolvimento do projeto, as suas várias fases de evolução dos seus componentes, jogos-exemplo nos quais várias mecânicas do projeto se baseiam e as ferramentas de terceiros usadas.

O quarto e último capítulo trata-se da conclusão à qual se chegou após o desenvolvimento do projeto, como os estudos da dissertação foram implementados, e como isso acrescentou valor ao produto, na opinião do autor. Também aqui se fala sobre as dificuldades que surgiram ao longo da escrita da tese e desenvolvimento do jogo, assim como os passos futuros sobre a preparação do jogo-projeto para ser colocado no mercado dos videojogos.

Capítulo 2

Estado da Arte

2.1 Introdução

2.1.1 Jogos

Brincar é uma atividade fundamental no desenvolvimento de vários seres-vivos, podendo ser vista a ocorrer tanto entre crianças ou crias, como entre adultos, humanos ou animais. Apesar das brincadeiras de crianças e de adultos serem habitualmente diferentes, todas elas têm um propósito para além do prazer e bem-estar. Toda a brincadeira acaba por servir de treino a uma aptidão, seja ela física, mental ou social, podendo esta ser necessária posteriormente num ambiente real (Huizinga, 1949). Um caso notável são leões brincarem entre si quando se encontram em tempo de repouso, mantendo a sua aptidão física, para posteriormente estarem melhor preparados para caçar ou para quando é necessário defenderem as suas crias. O Homem, para além de antigamente também participar nestes costumes, criou uma ramificação de brincadeiras que hoje em dia são denominadas de jogos, que hoje em dia são uma parte indispensável de várias culturas humanas (Caillois, 2001).

Apesar da sua presença mundial, ou talvez por essa mesma razão, é difícil encontrar uma definição precisa do que se trata um jogo, com David Parlett (Parlett, 1999) a afirmar que é inútil tentar encontrar uma simples definição, devido à grande variedade de características dos vários jogos existentes mundialmente. Ainda assim, vários estudiosos tomaram como responsabilidade chegar a uma definição mais próxima possível do que acreditam ser verdadeiramente um jogo, e mesmo variando ligeiramente entre si, há múltiplos pontos que os autores concordam que delineia um jogo de qualquer outra atividade. Eric Zimmerman e Katie Salen analisam textos de outros autores na procura desta delineação, no seu livro *Rules of Play: Game Design Fundamentals* (Salen and Zimmerman, 2003). Juntando esses textos ao livro de Zimmerman e Salen, chegou-se a seis pontos delimitadores na definição de um jogo.

O ponto mais que mais facilmente se fala quando se discute a definição de um jogo é a existência de interatividade, pois o próprio verbo 'jogar' significar 'tomar parte numa atividade lúdica, por prazer ou recreio; brincar' [1]. Abt, Suits, Sutton-Smith e Zimmerman falam especificamente

[1] *Jogar* in Dicionário infopédia da Língua Portuguesa. Porto: Porto Editora, 2003-2018. [consult. 2018-

desta componente, mas também apontam que esta só por si é demasiado abrangente, pois qualquer atividade que origine uma espécie de resposta é, por definição, interativa, não sendo limitada a jogos. Crawford e Costikyan também mencionam a tomada de decisões como uma parte fundamental dos jogos, mas como uma decisão mental é seguida por uma ação, este ponto pode ser unido à interatividade.

Para uma atividade ser lúdica — e possivelmente um jogo — tem que resultar em prazer e desfrute do um ou vários participantes, e é impossível o surgimento de tal se não há a voluntariedade para participar nessas atividades. Dessa forma, Callois, Suits e Sutton-Smith consideram pertinente todo e qualquer participante de um jogo voluntariar-se para o fazer, pois ao ser-se forçada a participação, o jogo passa a poder ser considerado um trabalho ou uma tarefa ausente de prazer. Zimmerman aponta que apesar da voluntariedade ser um ponto válido na caracterização de um jogo, esta reflete sobre a atividade de jogar e não sobre o jogo em si.

O ponto que todos os autores excepto Costikyan mencionaram ser essencial na descrição de um jogo é a existência de regras que limitam o que um jogador pode e não pode fazer quando participa num jogo. Suit entra em maior detalhe sobre este ponto, especificando que as regras proíbem métodos mais eficientes de superar obstáculos, de maneira a criar condições que tornem o jogo mais interessante e justo.

Assim como as regras, a grande maioria dos autores considera objetivos como um factor determinante na caracterização de um jogo. Huizinga é o único autor que não faz qualquer menção sobre objetivos, mas tal pode ser devido a Huizinga estudar mais o *brincar* em vez do *jogo* (Caillois, 2001), sendo que o primeiro não possui um objetivo predefinido ao invés do segundo, e apesar de Callois e Crawford não apresentarem o objetivo como um dos pontos da definição de jogo, é possível ver a menção ao mesmo no texto de outros pontos presentes nas suas obras. Outro ponto que se pode considerar que tem ligação aos objetivos do jogo é o elemento de conflito, pois para um jogador atingir o objetivo dum jogo, é necessário que este enfrente conflitos, independentemente se forem regras ou obstáculos, no caso de jogos individuais e de cooperação, ou outros jogadores com objetivos antagónicos, em jogos competitivos.

Tanto temporalmente como espacialmente, os jogos são isolados da vida real, afirmam Huizinga, Callois, Crawford e Zimmerman. Isso não quer dizer que situam fora do domínio da realidade e que não seguem as leis do tempo e espaço, mas que em ambos são-lhe criados limites, também conhecidos como tempo de jogo, como os 90 minutos — mais possíveis prolongamentos — num jogo de futebol, e a área de jogo, como uma mesa na maioria dos jogos de cartas e de tabuleiro, onde fora desse intervalo e área o jogo deixa de existir.

Adaptando os elementos previamente apresentados e na tabela presente no livro de Salen e

11-12 13:29:09]. Disponível na Internet: https://www.infopedia.pt/dicionarios/lingua-portuguesa/jogar

Zimmerman (Salen and Zimmerman, 2003), pode chegar-se à seguinte tabela 2.1 onde se verifica o que cada autor defende como definição de jogo.

Tabela 2.1: Elementos de definição de um jogo.

Elementos de definição de um jogo	Interatividade	Voluntariedade	Regras	Objetivos	Isolamento
Parlett			■	■	
Abt	■		■	■	
Huizinga			■		■
Caillois		■	■	■	■
Suits	■	■	■	■	
Crawford			■	■	■
Costikyan	■			■	
Avedon & Sutton-Smith	■	■	■	■	
Salen & Zimmerman		■	■	■	■

É possível concluir que apesar de não haver uma definição absoluta do que constitui um jogo, os pontos prévios são importantes na distinção entre estes e qualquer outra atividade.

Fora da definição de jogo, Roger Callois (Caillois, 2001) apresentou um grupo de categorias que ajudam na diferenciação entre jogos, e a perceber o foco atrativo de cada um. Estas quatro categorias são: *Agôn* , *Alea* , *Mimicry* e *Ilinx* .

Agôn resume-se ao elemento competitivo presente na grande maioria dos jogos, através do confronto direto entre duas ou mais entidades independentes. Este confronto não se trata necessariamente de conflito físico, podendo ser demonstrado de inúmeras outras maneiras usando diversas perícias humanas como agilidade ou intelecto. Desportos são o exemplo que mais facilmente se insere nesta categoria, mas outros jogos que requeiram a adversidade entre entidades, sejam elas individuais ou coletivas, são considerados jogos *agôn* . O objetivo deste género de jogos é a demonstração de aptidões, sejam elas físicas ou mentais, para servir de comparação imediata com outros participantes, numa espécie de corrida para provar quem é o melhor num determinado aspeto humano.

Originada do latim e com o significado de ’aleatório; chance’, *alea* pretende categorizar todos os jogos que contém qualquer elemento fora do controlo dos jogadores que possa influenciar o resultado do jogo, também conhecida como sorte. Os jogos que mais facilmente se categorizam como *alea* são os jogos de casino, mas existem também outros jogos que contém o elemento de aleatoriedade — mesmo que em menor escala — como jogos que dão uso a dados ou um baralho, pois nestes é impossível ter controlo total sobre o movimento dos dados quando estes saem da mão do jogador, ou das cartas que são distribuídas no início e/ou durante o jogo. Jogos focados em *alea* são atrativos pelo facto que permitem que qualquer pessoa ganhe, independentemente se é a mais hábil ou não, pois a sorte pode ficar do lado de qualquer indivíduo a qualquer altura, e está fora do controlo de qualquer um, criando, totalmente ou parcialmente, um pé de igualdade entre concorrentes.

Graças ao isolamento natural que os jogos têm da vida real, alguns deles aos quais lhe é dado o nome de *mimicry* permitem que o jogador tome um papel fictício que de certa maneira afeta o seu comportamento ou maneira de pensar. De facto, o ponto focal deste género de jogos é a fuga ao quotidiano da realidade, permitindo a tomada de um papel extraordinário que, se tomado fora do jogo, poderia causar estranheza a espectadores. Jogos com elementos de *mimicry* permitem então que jogadores se tornem algo que normalmente não são, mas onde esse novo ser tem um papel funcional dentro do jogo. É possível verificar-se este elemento em jogos onde existe um contexto narrativo, como no jogo de tabuleiro *Dungeons & Dragons* (Gygax and Arneson, 1974), onde a cada participante é dado um papel de personagem, cada um delas com funcionalidades diferentes, e apesar de uma pessoa não deixar de ser quem é quando incorpora esse papel, a sua maneira de pensar ou agir pode mudar para melhor incorporar a personagem que lhe é atribuída, somente para o prazer que tal transformação traz ou para melhorar a chance de vitória, se a mudança de personalidade ou postura for oportuna no jogo.

Por último, *ilinx* trata-se da sensação de vertigem ou de adrenalina que alguns jogos originam, através da presença de atividades que quebram a estabilidade e segurança física. Apesar de jogos categorizados como *ilinx* poderem ser vistos como uma atividade de risco, a descarga de adrenalina permite que o jogador entre num estado de êxtase de tal maneira que atividades do quotidiano raramente proporcionam. Atividades que podem comprometer a segurança física, como desportos radicais, desportos de combate, corridas e brincadeiras estonteantes são exemplos desta categoria de jogos que visa experimentar com os limites do ser, pondo-os repetitivamente à prova.

Para além das quatro categorias de jogos propostas, Callois escreveu também sobre um espectro de dois extremos antagónicos onde qualquer jogo se insere mas nunca pertencendo totalmente a um lado ou ao outro (Caillois, 2001). De um lado encontra-se *paidia*, a liberdade de brincar instintivamente, sem objetivo para além da diversão e bem-estar, e à qual não são impingidas regras ou limites. No lado oposto a *paidia* , *ludus* abrange o conceito de regras e ordem, limites na liberdade que permitem que atividades sejam orientadas para um objetivo preciso.

É importante reforçar a ideia que nenhum jogo é totalmente *paidia* ou *ludus*, pois deixariam de ser um jogo se assim fosse. Se uma atividade for composta somente de elementos *paidia*, significa que nela não há objetivos nem regras, dois pontos elementares dos jogos, resumindo-se somente a uma brincadeira espontânea. No caso oposto, se uma atividade só conter regras e limites, isso proíbe a exploração de abordagens que os jogos obrigatoriamente precisam, limitando-se a ser um conjunto de ordens sem possibilidade de variação ou experimentação, outro elemento essencial à composição de um jogo. É possível então afirmar que todos os jogos possuem tanto *paidia* como *ludus*, cada um em quantidades diferentes. Facilmente é vista tal afirmação em vários jogos reconhecidos. No futebol, existe a regra de não usar as mãos ou

braços no transporte da bola para a baliza do adversário, mas, apesar de os pés serem os membros mais usados neste desporto, também há a liberdade de de usar o torso e cabeça para os jogadores terem mais liberdade no manejo da bola. Em jogos de tabuleiro como o monopólio, existe a regra da direção da travessia do tabuleiro, e que é necessário pagar quando um jogador se encontra na casa de outro. No entanto, nenhum jogador é obrigado a comprar terrenos sempre que a peça repousa num deles que esteja livre. Também não está indicado em nenhuma regra que os jogadores não podem, a qualquer altura, negociar trocas de propriedades, e alguns jogadores tomam essa liberdade para tornar o jogo mais interessante para eles. No final, um espetro tão vasto permite uma infinita variação de jogos que têm níveis diferentes de *paidia* e *ludus*, dificilmente sendo um igual ao outro.

2.1.2 Videojogos

Com o aparecimento do computador em ambientes académicos no final dos anos 50, professores e alunos começaram a desenvolver programas para testar as capacidades tecnológicas das máquinas, alguns destes acabando por serem versões digitais de jogos analógicos famosos, como o jogo-do-galo e ténis (Figura 2.1). Foi assim que nasceram os videojogos, jogos que usam a tecnologia computacional para criar uma representação audiovisual com qual os utilizadores podem interagir.

Ainda que inicialmente estes jogos tenham estado disponíveis somente a membros de instituições académicas e expostos em feiras tecnológicas, como a Exposição Nacional Canadense, onde o primeiro videojogo do *jogo do galo* foi inicialmente apresentado, não foram necessárias mais de duas décadas para os videojogos atingirem o mercado geral, através de consolas e máquinas de *arcade*. Desde então, a rápida evolução tecnológica permitiu a também rápida evolução dos videojogos, passando da limitação que os processadores de oito bits tinham (Figura 2.2) para computadores capazes de processar informação de forma tão rápida, que permite a construção de videojogos a um nível de detalhe mecânico e gráfico bastante semelhante à vida real.

O que diferencia os videojogos dos jogos analógicos — para além da interface digital — é o uso do computador para executar operações lógicas, que permite a existência de funcionalidades que seriam mais demoradas ou até impossíveis de replicar num ambiente analógico. Em *RPGs*[2] clássicos como *Dungeons & Dragons*, cálculos são realizados para a execução de múltiplas operações, cada uma ocupando meros segundos, mas que acumuladas revelam um tempo considerável aplicado nesta tarefa. Tarefa que um computador consegue correr em tempo impercetível, como é possível ver em inúmeros videojogos de *RPG*. Outro aspeto que dá uma vantagem aos videojogos sobre os jogos analógicos é a capacidade gráfica, pois ao situar-se num ambiente estritamente

[2] *Role Playing Game*

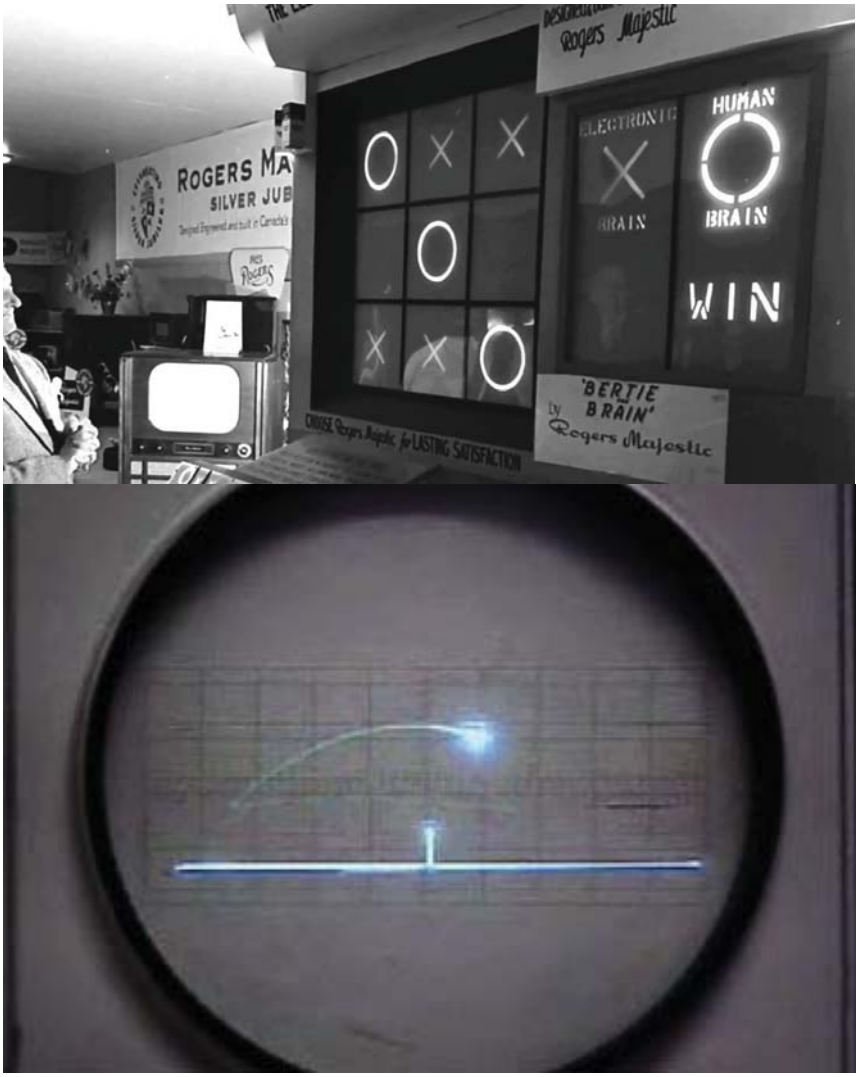

Figura 2.1: Primeiros videojogos.
(mobygames.com, 2017)(videogamehistory.wikia.com, 2012)

digital, é possível criar efeitos visuais que captam a atenção dos jogadores que é impossível replicar num jogo analógico.

Após da crise de 1983, onde a indústria de videojogos sofreu uma grande recessão económica devido à saturação de jogos, e que durante dois anos pôs em causa a viabilidade económica da indústria (Swink, 2009), deu-se um renascimento graças à entrada da *Nintendo Entertainment System* no mercado ocidental em 1985, com jogos icónicos como *Super Marios Bros.* (Nintendo, 1985) e *The Legend of Zelda* (Nintendo, 1986) que juntos venderam mais de 45 milhões de unidades. O sucesso da consola permitiu que o interesse do público ressurgisse, e daí em diante, os videojogos ganharam cada vez mais um lugar na vida quotidiana das pessoas, também por efeito da crescente adoção de computadores pessoais capazes de correr jogos e mais recentemente o mercado *mobile* que é de fácil acesso ao público.

O crescimento do sucesso dos videojogos é visível não só no seu impacto cultural, como também

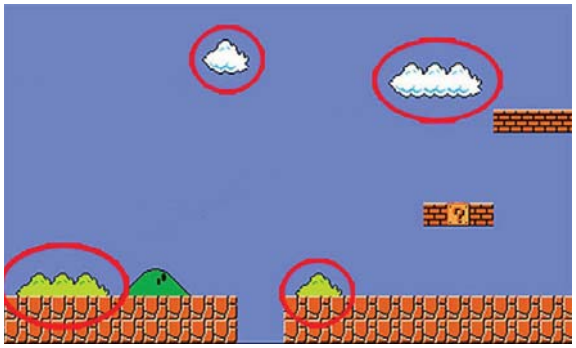

Figura 2.2: Cópia de *sprite* com coloração diferente devido às limitações computacionais. (facts.zone, 2018)

no seu rendimento económico. Em 2007, fez-se um estudo (Kortekaas, 2018) onde se registou uma receita global de 35 biliões de dólares na venda de videojogos, excluindo a venda de sistemas ou periféricos, e, em pouco mais de uma década, foi assinalado um crescimento de aproximadamente 300%, com um rendimento de 137,9 biliões de dólares em 2018, maior parte devido ao rápido crescimento do mercado *mobile* que, desde o surgimento do primeiro iPhone em 2007, já conquistou 51% do rendimento do mercado de videojogos. Está também previsto que estes valores continuem a aumentar, com a receita global da indústria a atingir os 180,1 biliões de dólares.

2.1.3 Narrativa

Assim como os jogos, as narrativas fazem parte da cultura humana desde os tempos antigos, chegando a estar presente mesmo na pré-história, através de pinturas e gravuras nas paredes de cavernas, onde uma sequência de imagens contavam histórias simples de rituais e hábitos. Um dos propósitos que as histórias sempre tiveram foi de transmitir conhecimentos sobre o mundo de uma geração para a seguinte, quer seja através de pintura, dança, fala ou outro meio, provando ser essencial para a sobrevivência e evolução de sociedades em tempos antigos. Mas a narrativa antiga não se limitava ao relato completamente fiel de eventos passados. Já no século XXI a.C., a *Epopeia de Gilgamesh* — a mais antiga obra literária registada — já contava com poetização de eventos, criando uma camada de ficção no que se acreditam ser acontecimentos reais. A ficção que já nesse tempo se mostrava presente foi tornando cada vez mais notável, com teatros na Grécia antiga compostos completamente por ficção, que na altura eram considerados eventos de honra a deuses e portanto de máxima importância. A tragedia fictícia apresentada nesses eventos marcou tanto a população do mundo ocidental, que as suas influências mantêm-se ainda hoje, graças ao que é conhecido como 'Cânone Ocidental', um conjunto de autores considerados exemplares, que — através das suas obras — moldaram a cultura artística ocidental (Bloom, 1994). Narrativas oriundas de texto e fala passaram então a ser adaptadas para vários meios artísticos, demonstrando a sua intertextualidade. Ao longo dos anos, meios mais modernos

como cinema, animação e videojogos continuaram essa adaptação, encontrando inspiração uns nos outros, acabando por criar obras que são uma amalgama de vários elementos de origens distintas.

Apesar da variedade de narrativas existentes e da diversidade de meios onde elas podem estar presentes, há elementos primordiais e uma estrutura definida que permitem que narrativas sejam reconhecidas de entre outros géneros de texto, independentemente da cultura ou período histórico. Aristóteles (Butcher, 1902) fala sobre seis elementos compositores de uma tragédia e a importância de cada um, não sendo, no entanto, obrigatório que todos estejam presentes, pois alguns meios artísticos possuem limites em alguns dos elementos que Aristóteles apresenta. Para além dos elementos de uma tragédia, Aristóteles apontou para uma estrutura que observou que todas as narrativas possuem, separando-a em três atos. Essa estrutura, em conjunto com os elementos, formaram o cânone narrativo que múltiplas obras e meios seguiram como regra, tornando-se um arquétipo da construção de narrativas.

- Fabula - Em primeiro lugar, Aristóteles fala sobre a fabula da tragédia ser o aspeto mais importante de entre todos os outros, pois uma tragédia ’é uma imitação, não do Homem, mas de uma ação e da vida’[3], e de tal forma uma narrativa não deve refletir sobre o carácter das suas personagens mas os eventos que as rodeiam, insistindo que é possível criar uma tragédia sem mostrar carácter, mas impossível de o fazer sem ação;

- Carácter - O Carácter de uma personagem permite que a audiência da obra perceba as suas características psicológicas e a consistência das suas ações, criando uma entidade credível, mesmo dentro de uma fabula. Para o espectador, o carácter das personagens deve ser apresentado resultante de ações, e não vice-versa, isto é, não se deve justificar ações através do carácter predefinido da personagem, pois a fabula deve permanecer como o elemento mais importante;

- Pensamento - O pensamento trata do efeito refletivo e emocional que as falas das personagens causam quando estas expões informações relevantes à fabula;

- Dicção - A dicção reflete sobre a maneira como as falas são entregues, de forma a poder distinguir um pergunta de uma afirmação ou de um comando, entre outras opções.

- Canto - Sabendo que o texto da obra de Aristóteles se incide sobre a atuação teatral, o canto retrata os coros presentes nas atuações do género, mas tal elemento também é aplicado em outros meios através de sons ambientais ou bandas sonoras.

[3]Tradução livre do autor. Texto original:”Tragedy is an imitation, not of men, but of an action and of life (...)”

- Espetáculo - Por último, o espetáculo aborda o elemento visual da narrativa, incluindo cenários, vestuário e, em alguns casos, efeitos visuais.

Devido à maneira como as tragédias teatrais eram construídas na sua altura, Aristóteles — para além de afirmar que uma narrativa era composta por um início, meio e fim — inseriu, entre atos, coros que tinham a função de expor aos espectadores a fabula até ai passada de uma forma sumariada, cantada com um tom que ajudasse a estabelecer um estado de espírito apropriado para os eventos da tragédia. Ao longo dos anos, e com alterações nas adaptações para outros meios, esta estrutura de três atos (Figura 2.3) foi moldada por Syd Field (Field, 2005), o qual transformou o início, meio e fim de Aristóteles na introdução, confronto e resolução, e onde antes se situavam os coros, passaram posteriormente a existir os pontos de *plot*, que servem de transição de um ato para o seguinte, apresentando um evento importante que muda a direção da narrativa de uma forma percetível e que de tal maneira capta a atenção do espectador.

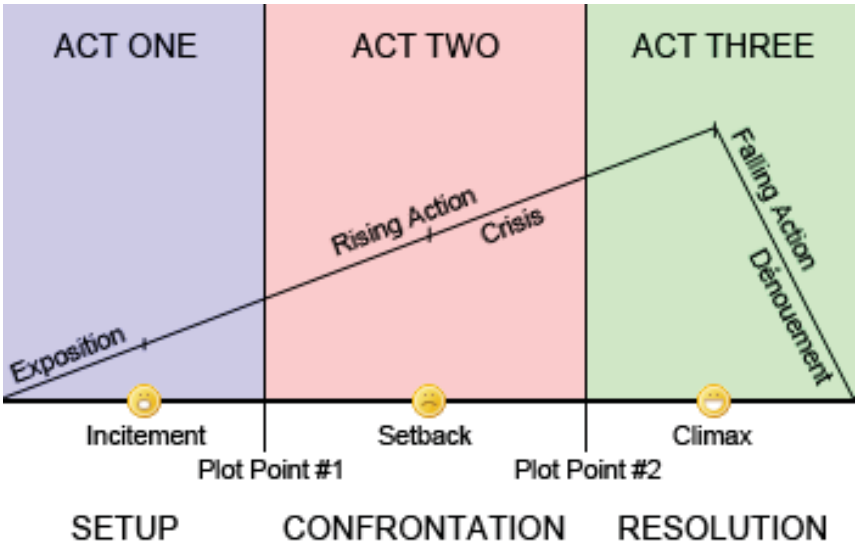

Figura 2.3: Estrutura de três atos.
(tvtropes.org, 2015)

A introdução é a fase inicial de uma narrativa e serve para contextualizar o espectador sobre a personagem principal, outras personagens secundárias com quem a principal tem relações, o mundo em que estas estão presentes e a premissa da história que se vai desenrolar. Este ato termina com a ocorrência do incidente motivador, o primeiro ponto de *plot* que empurra a personagem principal para fora da sua zona de conforto para a qual não é possível voltar, e que serve de impulso para que o protagonista procure a resolução do confronto apresentado neste momento.

O confronto é o maior ato numa obra narrativa, ocupando cerca de 50% da mesma — quando as restante ocupam 25% cada uma — onde se desenrola grande parte da ação, pois é neste ato que a personagem principal enfrenta um grande número de obstáculos, os quais não consegue facilmente superar sem primeiro necessitar de ficar mais forte ou mais consciente dos seus

limites e potenciais, mudando permanentemente quem é, geralmente para o melhor. No final deste ato encontra-se o segundo ponto de *plot*, onde o protagonista encontra as condições que acredita serem necessárias para o último confronto, e e antes que atinja o climax, a narrativa é transportada para o ato três, a resolução.

A resolução, o terceiro e último ato, é onde se situa o climax, o ponto de maior tensão e drama na narrativa e onde após isso há o declínio de ação, que acaba na resolução. Após todos os obstáculos presentes no ato II, que moldam a personagem para algo maior, o climax apresenta o último obstáculo que o protagonista precisa de superar para atingir a sua meta. Após esse momento, a pergunta da história é respondida, não obrigatoriamente de um modo em que o protagonista triunfa, podendo resultar num final desafortunado mas onde este aprende uma lição de valor que não aprenderia caso não embarcasse na aventura, torando-se em algo mais do que era antes.

2.2 Dificuldade de Ligação entre Narrativa e Jogabilidade

2.2.1 Ludologia vs. Narratologia

Com a inserção dos videojogos no mercado, textos académicos sobre os mesmos começaram a ser apresentados, mas graças à juventude dos videojogos como *medium* e produto, não existia ainda uma área de estudo especializada, o que fez com que surgisse a necessidade de os estudos originarem de outras áreas académicas já estabelecidas, como economia na construção de um sistema económico virtual ou arquitetura no design e desenvolvimento de níveis e estruturas. Mas mais notável foi o estudo narratológico dos videojogos, graças às semelhanças que estes possuíam com outros meios narrativos mais antigos, todos contando com personagens, ambientes, e eventos. Narratologistas começaram então na altura a ver os videojogos como meios propensos para a exposição de histórias, e que estes não deviam ser vistos somente como um produto tecnológico (Laurel, 1986). Graças a isso, técnicas normalmente usadas nas artes mais antigas foram propostas para serem aplicadas nos videojogos para lhes acrescentar valor como produto artístico (Laurel, 1993). No entanto, alguns académicos tinham uma opinião distinta sobre como os videojogos deviam ser estudados e avaliados. Estes autores acusavam os narratologistas de quererem colonizar os videojogos para as disciplinas artísticas (Aarseth, 1997) e defendiam que videojogos deviam ser avaliados estritamente pelo conjunto de regras e mecânicas que possuem, que isso é que lhes dá valor, e que a narrativa de um videojogojogo é um pequeno detalhe que — se modificado ou removido — não afeta o seu valor (Juul, 1999).

Opiniões tão opostas fez com que o autor e académico Gonzalo Frasca (Frasca, 1999) sentisse a

necessidade de criar uma disciplina de estudo que unificasse todos os conhecimentos adquiridos das outras áreas mas que também se tornasse independente das demais, escrevendo a seguinte proposta:

> O termo narratologia teve de ser inventado para unificar os trabalhos que os académicos de disciplinas diferentes estavam a fazer sobre narrativa. O estudo sobre jogos está numa situação semelhante: os tópicos têm estado a ser vastamente estudados através de disciplinas diferentes (por exemplo, psicologia, antropologia, economia e sociologia). No entanto, estes estudos são geralmente independentes, focando-se em pequenas características e sem olhar para os maiores padrões de conhecimento. Nós iremos propor o termo ludologia (oriundo de *ludus*, que em latim significa ”jogo”), para referir à ainda não-existente ”disciplina que estuda jogos e o ato de jogar”. Assim como a narratologia, a ludologia deve também ser independente do meio que apoia a atividade. [4]

Apesar de Frasca ter sugerido o uso do termo ludologia para criar uma união entre os dois grupos de académicos mencionados acima, o termo acabou por ser associado àqueles que eram contra a visão narratológica no estudo dos videojogos, mantendo ou até ampliando a discussão entre as duas comunidades (Ryan, 2001). O autor escreveu, quatro anos depois, um outro artigo de nome *Ludologists Love Stories, Too: Notes From A Debate That Never Took Place*, onde tenta corrigir vários mal-entendidos que acredita terem levado à discussão entre narratologistas e ludologistas, e que na realidade não encontra em nenhum estudo alguém de uma das disciplinas a considerar a perspetiva oposta como inválida.

No final, a discussão acabou por acalmar, com alguns autores a afirmarem que ambas as perspetivas são corretas, que os jogos conseguem oferecer tanto experiências narrativas como puramente lúdicas, dependendo de como são construídos (Kokonis, 2014), mas o efeito separador nunca chegou a desaparecer e acabou por se refletir noutras comunidades adeptas de videojogos, tanto nos jogadores como nos críticos.

[4]Tradução livre do autor. Texto original: The term narratology had to be invented to unify the works that scholars from different disciplines were doing about narrative. The research about games and play is in a similar situation: the topics have been broadly studied from different disciplines (for example, psychology, anthropology, economy and sociology). However, these studies are generally independent, focusing on small characteristics and without looking for bigger patterns of understanding. We will propose the term ludology (from ludus, the Latin word for ”game”), to refer to the yet non-existent ”discipline that studies game and play activities”. Just like narratology, ludology should also be independent from the medium that supports the activity.

2.2.2 Jogabilidade vs. Narrativas explícitas

Através da presença de elementos narrativos como personagens, ambientes, e eventos, os videojogos mostraram ter potencial de incorporar histórias num meio interativo. Para acrescentar valor aos seus produtos, os desenvolvedores procuraram inspiração nos seus antecessores narrativos, copiando as suas técnicas para a implementação de narrativas que cativassem mais os jogadores. De entre todas elas, o cinema foi aquela que mais foi estudada, devido a — assim como os videojogos — ser um meio audiovisual que toma proveito de um ecrã para a transmissão de informação.

No entanto, a adaptação de um meio cinematográfico para o lúdico mostrou não ser fácil nem direta. Tal deve-se às diferenças que os videojogos têm das artes narrativas tradicionais, denominadamente a interatividade. Onde nas artes tradicionais o espectador tem um papel exclusivamente passivo, sendo-lhe entregue a narrativa sem requisito de ação, nos videojogos tal já não é possível. Estes possuem o que o académico Espen J. Aarseth (Aarseth, 1997) chama de *Leitura Ergódica*, um género de leitura que requer um esforço não-trivial por parte do leitor para este poder avançar no texto. Num videojogo, este esforço traduz-se para jogabilidade, pois é através das ações e escolhas que um jogador progride na narrativa de um jogo. Porém, na tentativa da incorporação de uma narrativa trabalhada pelos desenvolvedores, surge um conflito sobre quem tem a autoria sobre os eventos de um jogo. Nas artes narrativas tradicionais, sendo exemplo livros e filmes, a história é previamente criada e trabalhada antes de ser apresentada ao público, e quando é apresentada, esta já encontra definida e inalterável. Um videojogo, no entanto, contém sempre algum grau de mutabilidade. A interatividade que estes promovem faz com que todas as ações que o jogador desempenhe resultem em eventos únicos, gerados exclusivamente por ele naquele momento. É essencialmente impossível dois jogadores replicarem exatamente as mesmas ações ao longo de todo um jogo, tornando-se cada um o autor das suas histórias dentro do meio. Após a análise de todos os pontos, é entendido que tanto os desenvolvedores como jogadores têm uma mão sobre a história que um jogo conta. Onde os jogadores conseguem criar narrativas implícitas únicas a cada um enquanto jogam, os desenvolvedores criam histórias explícitas através de meios sobre os quais os jogadores não têm controlo. Foi com base nessa limitação de autoridade do jogador que nasceu o método de narração que é atualmente o mais comum, as *cutscenes*.

Cutscenes são momentos onde o jogador não tem qualquer controlo sobre os eventos decorrentes, sendo estes temporariamente geridos pelo sistema de jogo de forma automática e inalterável. Esta tomada de posse pelo sistema permite que se fabriquem instâncias onde todas as ações ocorrem sempre da mesma maneira, tanto espacialmente como temporalmente, assim garantindo que certos acontecimentos num jogo são fiéis à visão dos seus desenvolvedores. Apesar de *cutscenes* já serem visíveis em jogos como o *Pac-Man* (Figura 2.4), estas só atingiram

um nível cinematográfico com a introdução de gráficos 3D em jogos.

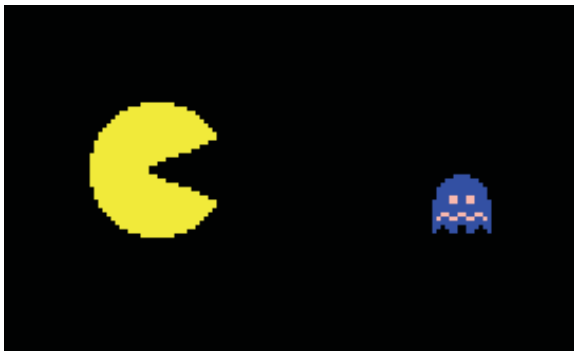

Figura 2.4: *Cutscene* no jogo *Pac-man*.
(joystickdivision.com, 2011)

Nesta evolução tecnológica, para além de se ter começado a usar polígonos na modelação de ambientes e personagens, um dos elementos que mais se transformou foi a câmara virtual. Onde nos jogos 2D a câmara tem uma orientação fixa, tomando maioritariamente uma perspetiva de *side-scroll* [5] ou *top-down* [6], nos jogos 3D esta permite que o ângulo que mostra a ação seja dinâmico, podendo-se mover em tempo real, quer seja automaticamente pelo sistema de jogo ou manualmente pelo jogador. Com essa liberdade que os jogos 3D trouxeram, tornou-se possível o uso de várias técnicas de filmagem, principalmente aquelas que requerem uma câmara dinâmica, e consequentemente, as *cutscenes* passaram a assemelhar-se bastante a cenas de filmes (Figura 2.5).

Figura 2.5: Interação entre duas personagens numa *cutscene*.
(duniagames.co.id, 2018)

Apesar de as *cutscenes* permitirem uma maior adição de elementos cinematográficos nos jogos, se não forem devidamente usadas, também podem trazer consigo múltiplas desvantagens para o meio que as acolhe. O exemplo mais notável — e já brevemente referido — é a obrigação do descarte completo da interatividade, o maior elemento diferencial entre os videojogos e outros meios narrativos. Terence Lee, designer de som e programador na equipa *Hitbox Team*, desenvolvedores do jogo *Dustforce* (Hitbox Team, 2012), escreveu no blog da equipa um artigo (Lee, 2013) sobre as várias formas de apresentar uma narrativa, e as desvantagens do uso de

[5] Termo inglês usado para vista lateral
[6] Termo inglês para vista aérea

alguns métodos cinematográficos na comunicação de histórias em jogos. O autor do artigo aponta que como os filmes adicionam o elemento de experiência sensorial na transmissão de ideias em relação aos livros, os videojogos acrescentam o elemento de interatividade aos demais (Figura 2.6), e se não se tirar proveito dos elementos diferenciais de cada meio na apresentação de uma história, a obra acaba por sair defetiva, pois o meio narrativo não está a ser usado em todo o seu potencial.

Figura 2.6: Eixos que representam os elementos que os três meios narrativos introduzem. (hitboxteam.com, 2013)

Lee também compara o uso de cinemáticas em videojogos com o uso de intertítulos nos filmes mudos. Onde nestes filmes os momentos de intertítulos reduzem a experiência sensorial a zero devido à imagem animada desaparecer enquanto o texto é exibido, no caso dos jogos as *cutscenes* reduzem a interatividade a zero, obrigando o jogador a passar de um ator a espetador (Figura 2.7).

Figura 2.7: Gráfico representante da redução de interatividade quando *cutscenes* entram em cena.

(hitboxteam.com, 2013)

Graças à facilidade de controlo de eventos que ocorrem dentro de *cutscenes*, e ao desejo de mostrar narrativas trabalhadas em detalhe, alguns desenvolvedores optam por expor a maioria, se não todas as partes importantes da história nestes momentos não-interativos. Tal decisão reflete-se na experiência dos jogadores, os quais podem sentir que todas as ações feitas nos

momentos jogáveis não têm impacto narrativo e que o jogo não confia neles para tomar decisões importantes autonomamente. De facto, os dois momentos de jogo podem-se encontrar tão desligados que a personagem presente nas *cutscenes* é essencialmente diferente da personagem que o jogador controla.

É erróneo achar-se que a motivação do jogador está sempre alinhada à da personagem que controla, pois pode variar de jogador para jogador e até variar dependente do estado de espírito com qual um jogador se encontra em momentos diferentes (Sylvester, 2013). Com os mundos fictícios que os videojogos oferecem, é normal que a curiosidade em algumas pessoas desperte, resultando numa vontade de explorar o ambiente e procurar os limites do que o jogo permite fazer, pois ao contrário da realidade, os videojogos não causam ameaças à vida do jogador na realização de certas ações. Esta simples discrepância entre a motivação do jogador e da personagem pode ser o suficiente para quebrar a suspensão de descrença (Coleridge, 2009), um estado de imersão em que uma pessoa entra quando aceita — por tempo limitado — uma obra fictícia como realidade. Essa quebra deve-se a múltiplos jogos colocarem a personagem jogável numa situação precária para narrativamente justificar a ação do jogador, mas se esse perigo não for sentido nos momentos jogáveis, toda a tensão criada nas *cutscenes* desvanece instantaneamente, expulsando o jogador do mundo fictício, pois o jogo acabou de perder a sua consistência narrativa. Quando o jogador deixa de estar investido na história do jogo, o distanciamento entre a narrativa e a jogabilidade acentua-se ainda mais, pois já que a ficção não foi capaz de entreter a mente do jogador, este vai procurar essa sensação fazendo o que achar mais interessante no momento, mesmo que essa atividade vá contra o rumo que a história está a tentar salientar. Nestes casos, é muito provável acontecer o que é chamado de *Desk Jumping* (Sylvester, 2013), a realização de uma ação pelo jogador que a personagem fictícia nunca faria no contexto narrativo em que se encontra, devido às suas motivações de ação serem bastante distintas, e daí resulta a criação de uma personagem diferente daquela presente nos momentos cinemáticos.

Um dos jogos que mais retrata isto é *Tomb Raider* (Crystal Dynamics, 2013), onde Lara Croft — a personagem principal e controlável — é apresentada como uma jovem inexperiente na carreira de arqueóloga, mas determinada mesmo assim. Mas antes que Lara pudesse encontrar aventuras, uma aventura encontrou a ela, e após um naufrágio, a protagonista encontrou-se sozinha numa ilha desconhecida. A personagem tenta procurar os seus amigos que partilhavam a expedição naval com ela, e entretanto encontra um arco que decide trazer com ela para a ajudar a sobreviver no ambiente hostil em que se encontra. Apesar da narrativa e as mecânicas do jogo até aqui estarem perfeitamente alinhadas, com a personagem a andar lentamente devido a ferimentos quando o jogador tentava fazê-la mexer-se, e ao automaticamente Lara olhar à volta com prudência, algo credível tendo em conta a situação em que se encontrava, é a partir daqui que se começa a notar múltiplas falhas nesta ligação. Em primeiro lugar, pouco após apanhar

o arco, Lara exclama para si mesma que necessita de arranjar comida se quer sobreviver. Em poucos instantes o jogador vê um veado que pode caçar, e após o animal ter sido acertado e o jogador se aproximar do corpo do animal, uma *cutscene* começa. Aqui é mostrado que o animal ainda está vivo e em sofrimento, e é nesse instante que Lara pede desculpa e aplica um último golpe para acabar com a dor do ser vivo (Figura 2.8). É estabelecido que a protagonista é alguém que valoriza a vida e que só mata algo se for estritamente necessário, mas as mecânicas de jogo contrariam a mensagem que a história acabou de divulgar, pois encontram-se outros animais na área — outros veados incluídos — que podem ser mortos, e desta vez não só não há efeito negativo psicológico para a personagem, como são dados pontos de experiência como recompensa por cada morte, que são usados para evoluir as habilidades da protagonista. Pouco tempo depois, Lara e os seus amigos agora reunidos são apanhados por mercenários armados que os ameaçam de morte. O grupo tenta eventualmente escapar e Lara esconde-se, mas rapidamente é vista por um membro dos mercenários e os dois acabam por entrar em conflito físico. No final da ação, a personagem principal tem uma arma de fogo nas mãos e é obrigada a disparar sobre o inimigo se não quer ser morta por ele. Após o gatilho ser pressionado e o mercenário morrer de forma grotesca, Lara mostra-se traumatizada pelo sucedido, chegando a ter reflexos de vómito, o que pode ser considerada uma reação comum dadas as circunstâncias, mas após esta morte humana, é permitido ao jogador matar qualquer outro homem, mais uma vez à troca de pontos de experiência para a melhoria da personagem, em vez de se ser penalizado por ir contra os princípios morais da personagem, e qualquer tentativa de furtividade não é recompensada, até se podendo considerar como penalização pois Lara acaba mais fraca do que se recorrer à violência.

Figura 2.8: Cena onde Lara se encontra junta ao veado após o matar.
(techsob.com, 2013)

Pode-se concluir que acabam por haver duas Lara Croft neste jogo. No lado narrativo, a protagonista é alguém que só mata se necessário para a sua sobrevivência, e que mostra medo da situação perigosa em que se encontra. Já a Lara dos momentos de jogabilidade não hesita em matar outros seres vivos para proveito próprio, mostrando uma perícia instantânea sobre artes que nunca praticou, como o arco e flecha.

Para além desta inconsistência afetar a caracterização de uma personagem, também afeta a

relação que o jogador tem com ela. Quando o jogo se encontra em momentos jogáveis, o jogador experiencia os eventos em primeira pessoa, pois incorpora-se na personagem, e é ele que reage ao que o jogo lhe apresenta, daí surgirem frases como "-Consegui resolver o puzzle." ou "-Finalmente derrotei este monstro!", mesmo que as ações em si sejam visualmente feitas por uma entidade dentro do jogo. Mas quando o jogo entra num momento não-interativo, e principalmente se a personagem agir de uma maneira muito distinta de como age nas mãos do jogador, faz com que a personagem deixe de parecer uma extensão do jogador naquele ambiente fictício, e passe a ser uma identidade completamente independente, mesmo que só por breves instantes. Desse sucedido resulta o que Lee (Lee, 2013) chama de 'dissonância de identidade', a dificuldade em o jogador perceber o seu papel no jogo, se está a ser incorporado na personagem ou se está somente a guiá-la, pois esta mostra ter uma personalidade estabelecia nas *cutscenes*, mas é permitido ao jogador fazer ações que vão contra essa personalidade quando este tem controlo sobre ela.

Foi com base nestes temas que Clint Hocking — atual diretor criativo na Ubisoft Montreal — escreveu no seu *blog* pessoal sobre o tema 'Dissonância Ludonarrativa' (Hocking, 2007), um termo cunhado pelo mesmo que serve como uma descrição generalizada da discrepância entre a componente narrativa e a componente lúdica presente em vários jogos. Nesse artigo, Hocking aponta para o jogo *Bioshock* (2K Boston, 2007) como um exemplo que sofre de dissonância ludonarrativa, e como isso impede o jogador de se conectar a ambos os elementos narrativo e lúdico ao mesmo tempo, pois estes encontram-se em conflito, obrigando-o ou a ignorar a parte narrativa e somente jogar o jogo pelo o que ele é ludicamente, ou investir-se na narrativa e ter que tolerar uma jogabilidade que não vai de encontro à história presente. Noutro artigo online, Brett Makedonski (Makedonski, 2012) aponta os jogos *Max Payne 3* (Rockstar Studios, 2012) e *Mass Effect* (BioWare, 2007) como outras vítimas desta divergência das componentes de jogos, podendo-se chegar à conclusão que dissonância ludonarrativa é algo que está presente em múltiplos videojogos conhecidos.

2.3 Importância da Narrativa na Jogabilidade

2.3.1 *Flow* e Suspensão de Descrença

Não obstante às desvantagens do uso de métodos não-interativos nos videojogos, é também importante apontar que existe uma utilidade — e em certos casos a necessidade — em usar tais técnicas para uma narrativa complexa ser bem retratada.

Todo o meio de entretenimento, seja ele através de atividade lúdica ou através da exposição a

uma narrativa, proporcionam uma sensação de imersão ao seu utilizador, e os videojogos conseguem atingir essa sensação de ambas as formas. Na dimensão lúdica, os videojogos permitem ao jogador entrar num estado de *flow* (Csikszentmihalyi, 2014), que se define como um período de tempo em que uma pessoa fica tão imersa numa atividade que esta se torna temporariamente o foco de toda a sua atenção. Ocorre uma perda de noção temporal e espacial, e toda a ação feita nesse estado parece natural, dando uma sensação de controlo e maestria, pois as ações decorrentes encontram-se num nível equilibrado entre a sua dificuldade e a habilidade de quem as faz, não sendo difíceis o suficiente para causar ansiedade mas também não sendo demasiado fáceis ao ponto de serem tediosas (Figura 2.9).

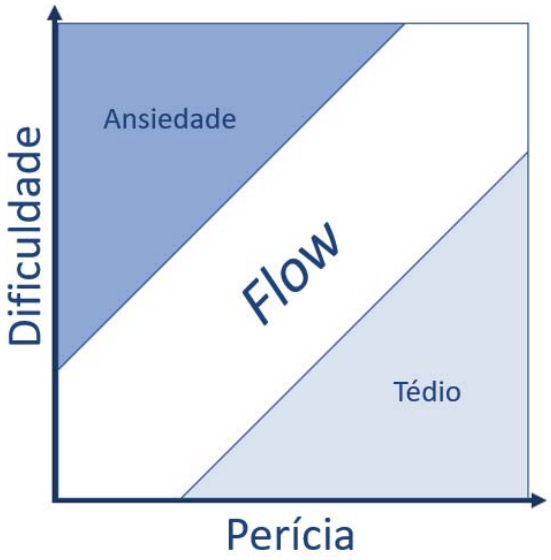

Figura 2.9: Gráfico simples do conceito de *flow*.

A segunda maneira de imergir um jogador num videojogo é através da suspensão de descrença (Coleridge, 2009), mas ao contrário do *flow*, a suspensão de descrença não requer que uma pessoa se imerja numa atividade, mas sim que se invista numa história de forma a interpretá-la — ao longo da sua duração — como real, ignorando elementos surreais em favor do entretenimento. Semelhante ao *flow*, há uma perda de noção do tempo e o espectador pode imergir-se ao ponto de reagir por instantes como se os acontecimentos da obra estivessem a ser sofridos por ele, notavelmente no caso de uso de meios visuais. É imprescindível apontar que os estados de *flow* e suspensão de descrença não são exclusivos, isto é, não é preciso abdicar de um pelo outro, sendo ambos possíveis de alcançar nos videojogos, oferecendo uma experiência única quando ambas a narrativa e jogabilidade são adequadamente implementadas.

2.3.2 Limitação Parcial da Jogabilidade

Uma forma que as mecânicas de jogo ajudam a mensagem narrativa a ser transmitida é através da limitação parcial da interatividade. Este método pode ser controverso tendo em conta que vai contra a fundação interativa que faz um videojogo, mas quando corretamente usadas, as limitações parciais ajudam a transparecer acontecimentos narrativos para o jogador quando este se encontra num momento jogável. As limitações parciais podem-se condensar em dois grupos: limites de exploração, e limites de ação.

Os limites de exploração são os mais comuns, presente em todos os videojogos para delimitar a área de jogo, mas mais predominantes em jogos onde o progresso é separado por níveis ou quando se trata de uma aventura linear. Nestes últimos casos, o jogador é impossibilitado de ir para um nível ou área anterior à que está, e é obrigado a seguir numa única direção. Esta limitação é melhor aplicada em jogos que querem transmitir uma sensação de urgência, pois permitir que o jogador explore livremente a área de jogo quebraria o contexto narrativo do jogo. O jogo *Prince of Persia: Sands of Time* (Ubisoft Montreal, 2003) é um exemplo onde o protagonista acidentalmente espalha uma espécie de vírus que transforma pessoas em monstros, e a mesma está rapidamente a alastrar-se, implicando que existe uma urgência em reverter o sucedido, sendo exposto ao jogador o que deve fazer e como o fazer para ter sucesso na sua demanda. Mecanicamente, o jogo usa portas que se abrem de um lado mas que se fecham após o jogador as passar, ou descidas impossível de voltar a subir para limitar a navegação do jogador e insistir a que este siga sempre em frente. *Inside* (Playdead, 2017) também usa o limite de exploração de uma forma inteligente. Para além de ser semelhante ao jogo *Prince of Persia: Sands of Time* no uso de plataformas às quais não se pode voltar após serem atravessadas, *Inside* apresenta-nos nos primeiros instantes uma personagem jogável que mostra uma linguagem corporal representativa de medo, sendo nos seguintes minutos mostrado a razão pelo qual, pois brevemente após o início e ao longo do jogo, várias entidades perseguem o protagonista e é o objetivo do jogador fugir a essas entidades, quer seja escondendo-se, fugindo ou contornando obstáculos resolvendo pequenos puzzles espaciais, mas eventualmente sempre avançando na mesma direção.

Limites de ação também estão presentes em vários títulos de renome, e define-se como a alteração de determinadas atividades quando certas condições são cumpridas. No jogo *Final Fantasy XV* (Square Enix, 2016), o jogador consegue usar as suas armas de combate de forma livre na maioria do mapa, mesmo quando inimigos não estão presentes, mas quando o protagonista entra numa cidade onde vários cidadãos residem, é-lhe impossibilitado qualquer ação de ataque, mesmo que este não fosse atingir alguém. É possível que esta decisão de design tenha sido tomada com a narrativa do jogo em mente, pois Noctis, o protagonista que o jogador controla, está fugitivo de forças imperiais que o perseguem, e o uso de armas num ambiente com muitos

espetadores seria algo que traria atenção indesejada. Outro bom exemplo é o jogo *Journey* (thatgamecompany, 2012), o qual tem como mecânica o uso de um recurso consumível que, por breves momentos, permite que a personagem voe. Este jogo que decorre maioritariamente no deserto, onde a personagem mostra estar confortável, mostra ao jogador que se deve aventurar até o topo de uma montanha. Perto do topo, um ambiente coberto por neve substitui as dunas de areia que até aí predominavam o mundo do jogo, e isso rapidamente afeta visivelmente a personagem, pois o recurso de voo começa lentamente a degenerar-se, deixando o jogador cada vez mais incapaz de usar esta habilidade, e para além disso, a personagem começa a ficar com neve presa ao seu robe, dando passos cada vez mais pesados que são sentidos pelo jogador graças à crescente lentidão de movimento, independentemente da intensidade da direção do analógico do comando (Figura 2.10).

Figura 2.10: Protagonista do jogo *Journey* a caminhar na neve.
(3djuegos.com, 2012)

Uma limitação parcial da jogabilidade não se limita a reduzir a interatividade de um videojogo, pois quando justificadamente e devidamente aplicada, esta ajuda a transparecer ao jogador os limites das suas personagens, não lhe sendo só mostrado mas também sentido através do controlo em tempo real, permitindo o jogador relacionar-se melhor com as impotências da personagem que controla, resultando num jogo narrativamente mais sólido e uma experiência mais credível.

2.3.3 Emoção e Sentimento

Meios de entretenimento, por natureza, têm como objetivo captar e manter a atenção dos seus espectadores, apresentando — de uma forma condensada — momentos extraordinários à vida habitual. Ao dedicar uma considerável atenção a certa atividade de entretenimento, o ser humano é propenso a investir-se sentimentalmente sobre a mesma, manifestando-se através de reações emocionais a acontecimentos importantes dentro desta. Os videojogos não são exceção, conseguindo obter e reter interesse através da diversificada interatividade que oferece.

Elevação da Narrativa Através de Mecânicas de Jogabilidade

Tendo como núcleo a ludicidade, os videojogos são capazes de facilmente originar emoções que os seus antecessores — jogos analógicos e, de certa forma, desportos — suscitam, tanto para os participantes, como para os seus espectadores. No entanto, estes meios interativos têm limites nas emoções que naturalmente despertam, pois apesar de ser possível provocar reações emotivas como vergonha, raiva ou tristeza, tal não é o objetivo focal dessas atividades, sendo esse a superação dos desafios que o jogo lúdico ou desportivo faculta e as emoções positivas que advêm dela. Todas as restantes são reações subjetivas e individuais que cada participante ou espectador tem sobre os eventos adversos ao objetivo da atividade, tomando como exemplo a possibilidade de se sentir tristeza ou raiva na derrota.

As obras de entretenimento de natureza narrativa, por sua vez, têm como um dos seus objetivos principais o despertar de um diverso leque de emoções, tanto positivas como negativas, para a criação de uma experiência emocional mais forte, diversificada e única (Figura 2.11). Através do uso de várias ferramentas que não estão presentes nas atividades lúdicas — como a comunicação do estado de espírito das personagens — as obras narrativas mostram-se superiores na criação de condições despertadoras de emoções. Essas reações podem revelar-se através da cópia de gesticulações físicas ou faciais visíveis, também conhecida como *mirror rule* (Zacks, 2015), ou através de uma resposta emocional adequada às condições presentes, sendo exemplo o sentimento de raiva quando uma injustiça provoca tristeza sobre alguém a quem se quer bem.

Esta ligação emocional é o que normalmente se denomina de empatia (Oatley, 1994), e tal não se limita a ocorrer sobre episódios presentes no mundo real, também podendo suceder numa relação fictícia quando um espectador se identifica em uma ou múltiplas personagens da obra, consequentemente imergindo-se na narrativa e imaginando-se fazer parte do seu mundo, resultando numa suspensão de descrença. Obras narrativas tomam então proveito dessa natureza humana, concebendo personagens relacionáveis através da criação de falhas comuns ao ser humano, facilitando a relação empática entre o ser real e fictício (Mar et al., 2009).

Possuindo vários elementos em comum a outros tipos de obras narrativas, os videojogos viram também a oportunidade de implementar histórias e personagens complexas que incentivassem o jogador a investir-se emocionalmente (Wardrip-Fruin, 2004), acrescentando valor ao produto, e, devido à evolução tecnológica, tornou-se cada vez mais fácil despertar reações emocionais no jogador, quer seja pela fidelidade gráfica que permite a criação de áreas, entidades e eventos mais percetíveis (Figura 2.12), ou no uso de elementos de apoio, como música dinâmica à ação do jogador, que complementem o estado de espírito que o jogo quer transmitir (Pynenburg, 2012).

Todavia, os videojogos possuem uma notável inconveniência na transmissão de informação da história em comparação aos outros meios narrativos, e isso deve-se à maneira como a audiência é fisicamente condicionada em cada um deles. Em obras audiovisuais como filmes e atuações

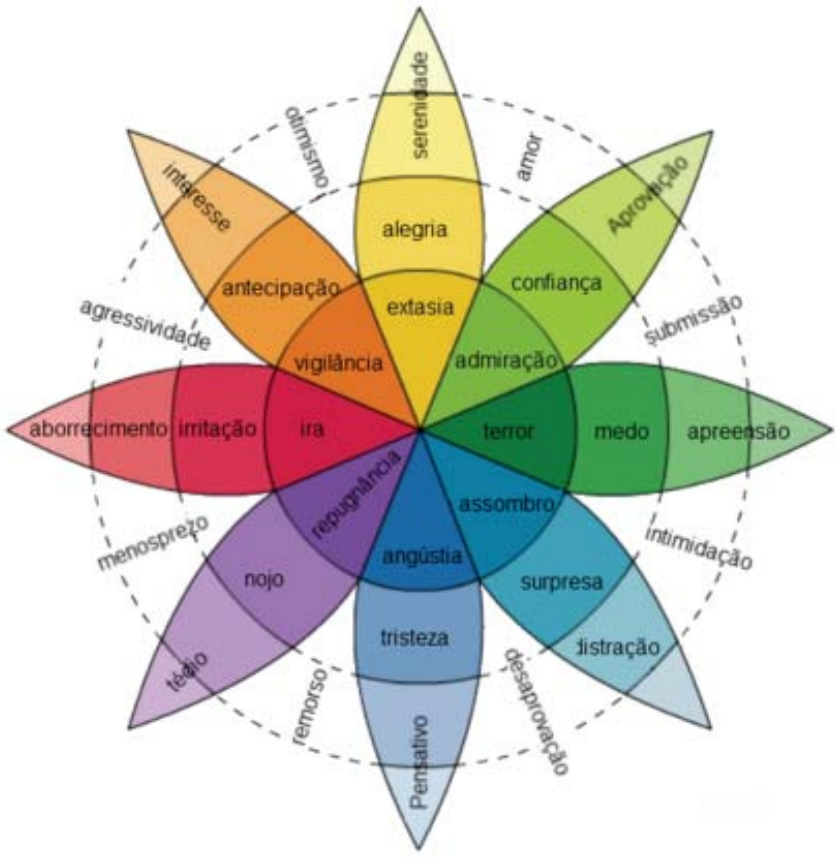

Figura 2.11: Roda das Emoções de Plutchik.
(pt.slideshare.net, 2012)

teatrais, os espectadores são geralmente sentados em lugares confortáveis e o ecrã ou palco das atividades respetivas ocupam uma grande maioria do campo de visão. Não é requerida qualquer ação física consciente e é também regulamentado que não se faça barulho durante o espetáculo, não só para não perturbar a atuação dos atores no caso de uma peça teatral, como para otimizar o foco tanto da própria pessoa como de outros. No caso de meios narrativos menos dinâmicos, como exemplo livros, estes são apreciados em ambientes sossegados e livre de muitas distrações, o que é muitas vezes necessário para se manter a atenção fundamental na leitura de texto.

Em qualquer um dos casos anteriormente falados, a passividade física auxilia o consumidor destas artes a perder noção do real e mergulhar sobre os acontecimentos da obra, e através da suspensão de descrença, a narrativa acaba por se tornar mais emocional e memorável. Os videojogos desvendam uma dificuldade na criação dessa passividade graças à sua natureza interativa, a qual pode estorvar a imersão na narrativa presente no jogo. Essa dificuldade é mais notável nos momentos onde o jogador tem mais liberdade de ação ou quando há conflito entre fontes de informação diferentes, e.g. uma fonte visual — como algo a acontecer no ambiente físico do jogo — e uma fonte textual — como um diálogo escrito entre personagens — pois à custa da atenção humana ser limitada, o pedido de atenção por parte de uma fonte obriga a restrição

Figura 2.12: Momento no final do prólogo do jogo *The Last of Us* (Naughty Dog, 2013). (jogosalvo.com.br, 2017)

na outra, sacrificando-se parte da informação que ambas querem transmitir ao mesmo tempo.

Nos momentos de maior exposição de narrativa, a limitação de ação do jogador — seja ela parcial ou total — pode ajudar na tomada de uma postura mais passiva. No entanto, os videojogos perdem parte da sua essência interativa quando agência do jogador é limitada. Por essa razão, é importante restringir o uso destes limites e usá-los só nos momentos necessários, tendo conhecimento das suas vantagens e desvantagens, para o jogador não sentir que a agência lhe está a ser retirada injustificadamente.

Assim sendo, a existência de narrativas explícitas, as quais requerem o uso de técnicas limitadoras de agência, são favoráveis para promover uma postura passiva por parte do jogador num meio altamente interativo, podendo assim os videojogos proporcionarem experiências emocionais que tanto atividades lúdicas como obras narrativas conseguem oferecer, tirando proveito dos pontos fortes de cada uma, usando as técnicas de cada uma nos momentos adequados.

2.4 Conclusão

Pode-se concluir que os videojogos, com a sua natureza interativa e a presença de vários elementos narrativos de meios audiovisuais, têm a capacidade de oferecer experiências narrativas tão ou mais complexas do que as se encontram em meios passivos como livros e cinema.

No entanto, é necessário saber-se quais são as condições ideais para usar elementos de cada lado, pois muitas características de cada um são antagónicas (Juul, 1999), não podendo ser utilizados ao mesmo tempo. Com a jogabilidade a poder ajudar na ligação entre jogador e

personagem e colocar o prévio num estado de *flow*, e a narrativa a imergir o jogador numa acreditada realidade alternativa graças à suspensão de descrença, é importante não deixar que a interatividade roube a atenção que a história requer, mas também não deixar que as limitações de agência nos momentos de exposição narrativa quebrem a experiência do jogador.

Tendo-se noção desses pontos, o videojogo-projeto procura encontrar e aplicar esse equilibrio entre a jogabilidade e a narrativa, tendo em mente a manutenção da agência sempre que possível, e mostrar como isso pode tornar a experiência narrativa do jogo mais rica.

Capítulo 3

Projeto

3.1 Pré-produção

O projeto desenvolvido teve a sua origem num trabalho teórico feito na disciplina 'Teoria e Metodologia do Design', lecionada pelo professor doutor Herlander Alves Elias no curso Design e Desenvolvimento de Jogos Digitais na Universidade da Beira Interior. Nessa disciplina, a aluna Mafalda Claro — um dos membros da dupla desenvolvedora deste projeto — criou uma proposta de design de videojogo, a qual apresenta como razão de conceção a falta de um bom uso de fábulas adaptadas para uma audiência mais adulta, que exibem personagens animais antropomórficos e temas morais que as fábulas naturalmente têm, mas trabalhando-as de forma a apresentá-las de uma maneira mais complexa e matura. Apesar da existência de alguns jogos com elementos naturais das fábulas, como um sistema de escolhas morais no jogo *Fable* (Lionhead Studios, 2004) ou animais antropomórficos como em *Armello* (League of Geeks, 2015), não é do conhecimento da equipa deste projeto a existência de um título popular que use todos os elementos essenciais de uma fábula em conjunto e direcionado para um público adulto. Com base nesta falha, viu-se uma oportunidade de explorar melhor esse mercado.

Ainda no mesmo semestre, a proposta inicialmente apresentada evoluiu, detalhando os elementos temáticos e mecânicos, e onde o nome foi mudado de *Prosopopoeia* para *Ink Tales*. Este novo nome reflete melhor a ligação às fábulas e à possibilidade de fazer escolhas que marcam o mundo do videojogo, como a tinta — *ink* em português — marca o papel, mantendo-se até ao presente como o nome do projeto desenvolvido nesta dissertação.

No semestre seguinte, as ideias exploradas nesta proposta começaram a ser postas em prática, através dum trabalho em equipa entre quatro alunos do mesmo curso, agora contando com ambos os membros da dupla desenvolvedora do projeto *Ink Tales* atual. A equipa de quatro membros desenvolveu nesse semestre um protótipo que sofreu alterações em vários aspetos, de modo a ir de encontro à capacidade dos vários alunos. De um *RPG* passou para um *platformer*/aventura em 3D mas com movimento limitado a 2 eixos, também conhecido como perspetiva 2.5D. Para além do movimento básico de correr e saltar, o videojogo contou com uma mecânica de combate a curta distância através do uso de uma arma semelhante a uma espada, e uma mecânica de desenho, onde o jogador tinha a liberdade de desenhar no ecrã sem restrições, e onde esse dese-

nho servia posteriormente de plataforma para o jogador ultrapassar vários obstáculos presentes no nível desenvolvido (Figura 3.1), tendo sido este último construído para a demonstração das mecânicas e visuais desenvolvidos ao longo desse semestre.

Figura 3.1: Uso da mecânica de desenho para a criação de escadas.

Foi com base no conjunto de projetos teóricos e práticos desenvolvidos ao longo desse ano letivo que o videojogo *Ink Tales* atual se baseou. Apesar de nenhuma mecânica ter sido diretamente transferida de um projeto para o outro, a orientação deste que acompanha a dissertação foi de encontro aos conhecimentos adquiridos no anterior. Também foram descartados alguns elementos que não iam ao encontro do tema do *Ink Tales* atual, como a remoção do combate, para favorecer a temática de fábulas onde é raro o confronto físico, e a troca dos modelos 3D a favor de *assets* 2D, para ir de encontro ao ambiente escolhido, um livro de *pop-up*, onde todo o cenário é composto por elementos que assemelham-se a papel.

3.2 Ferramentas Utilizadas

O desenvolvimento deste projeto contou com o uso de múltiplas ferramentas que não só aceleraram a construção do mesmo, como em alguns casos foram cruciais na implementação de algumas mecânicas que seriam impossíveis de desenvolver por limite de conhecimento ou de tempo, sendo as seguintes três as de maior mérito.

3.2.1 *Unity*

Como base de desenvolvimento do projeto, foi utilizado o motor de jogos *Unity*,(Unity Technologies, 2005-2018). Um motor de desenvolvimento de jogos é um programa com várias ferramen-

tas, como montagem de cenas, controladores de animações, um compilador de *scripts*[1], entre outros, que facilitam e aceleram o processo de desenvolvimento de projetos. Incorporado com o *Unity* também está opcionalmente o editor de código *Visual Studio* (Microsoft, 1997-2017), que também foi usado na construção de *scripts* aplicados no *Unity*, e que usa C# como linguagem de programação. O *Unity* também possui uma loja de *assets*, onde é possível descarregar ou comprar pacotes de utilidades de fácil aplicação em projetos dentro do programa, tendo sido alguns deles aplicados no *Ink Tales*, principalmente na área visual.

3.2.2 *$P Point-Cloud Recognizer*

O *$P Point-Cloud Recognizer* é um algoritmo de reconhecimento de traços escritos em ambientes digitais (Vatavu et al., 2012). Desenvolvido em *JavaScript* e *C#*, este algoritmo permite que algo desenhado numa área seja reconhecido como um símbolo ou carácter, através da existência duma base de dados com desenhos pré-feitos aos quais o desenho atual é comparado. No final é mostrado o nome do desenho da base de dados mais semelhante ao do utilizador (Figura 3.2).

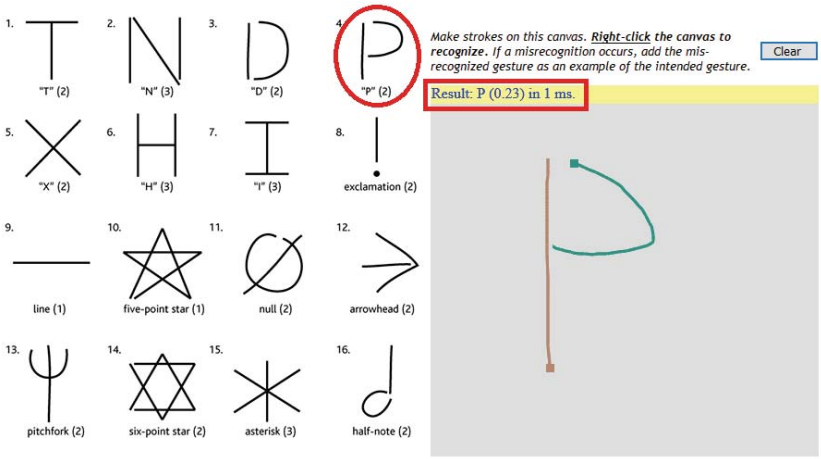

Figura 3.2: Demonstração do *$P Point-Cloud Recognizer* no *website* oficial da ferramenta. (Wobbrock, 2012)

Esta ferramenta foi posteriormente adaptada como um *asset* para o *Unity* pela equipa *The Viking Code*, tornando-se fácil a sua implementação no projeto para o uso na mecânica de desenho, sendo só necessárias algumas alterações da mesma para cumprir os requisitos procurados para este projeto.

[1] ficheiros que contêm informação e comandos na forma de uma linguagem de programação.

3.2.3 *Yarn Spinner*

Yarn Spinner (The Secret Lab, 2015) é um interpretador da linguagem *Yarn* (Infinite Ammo Inc., 2015) (Figura 3.3), que por sua vez foi baseada no *Twine* (Chris Klimas, 2009), sendo estes dois últimos interfaces que permitem a criação de árvores de diálogo. O *Yarn Spinner* interpreta então o conteúdo presente na interface *Yarn* para criar um sistema de controlo de diálogos através de condicionais, tanto dentro da interface *Yarn*, e. g. verificar se já se falou com uma personagem uma primeira vez, como as do *Unity*, como apresentar um diálogo diferente se o protagonista já fez uma certa ação controlada através do motor de jogos. Um sistema deste género permite assim criar um ambiente mais imersivo, devido a tornar mais notável a interligação entre as várias personagens do videojogo e o mundo em que elas vivem.

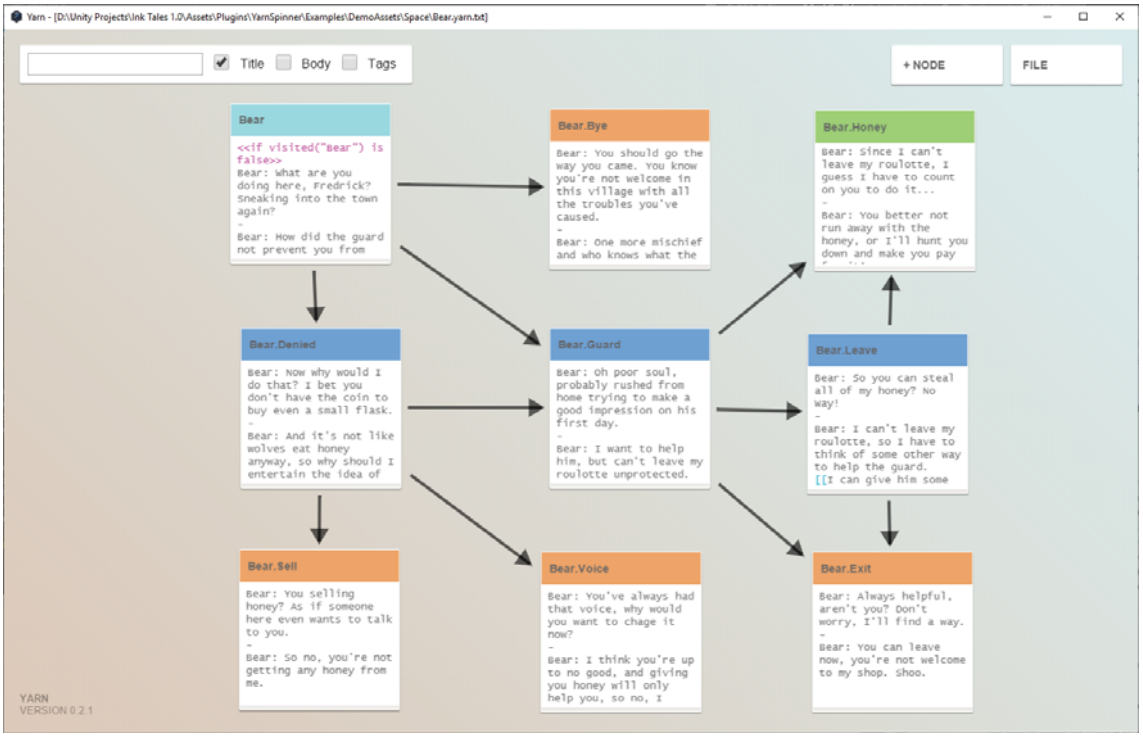

Figura 3.3: Interface do *Yarn*.

3.3 Desenvolvimento do Protótipo

3.3.1 Estética

O ambiente de um videojogo é um dos componentes mais cruciais para a experiência que este proporciona, pois apoia-se sobre os vários sentidos do ser humano para criar um mundo credível, mesmo que fictício. É importante esclarecer que o ambiente não se trata somente dos gráficos que um videojogo tem, é também a coerência dos elementos que formam o mundo, sejam eles

mecânicos, visuais ou auditivos. Quando um videojogo falha na coerência do seu ambiente, o jogador é relembrado que não faz parte do mundo em que acreditava estar presente e que é somente um agente externo. Uma série de videojogos exemplar disso é o *Grand Theft Auto* (GTA) (Rockstar North, 2013), que promovem ter um enorme mundo *sandbox*[2] realista, mas onde na maioria dos títulos — excluíndo o GTA II e V — existem garagens chamadas *Pay 'n' Spray*, onde o jogador consegue reduzir ou eliminar qualquer nível de procura pela polícia, quando a única coisa que essas garagens fazem é pintar o carro de uma cor diferente, algo que no mundo real seria impensável de funcionar. Esta quebra das regras realistas que o GTA promove faz com que a suspensão de descrença do jogador se perca, pois tais eventos obrigam o jogador a aceitar regras que narrativamente não fazem sentido dadas as regras desse mundo.

Com isto em mente, o ambiente do *Ink Tales* procurou ir sempre de encontro às regras impostas pela narrativa, de forma a criar um ambiente coerente e capaz de provocar e manter a imersão do jogador. Olhando para o contexto narrativo do projeto — as fábulas — um livro foi um dos primeiros ambientes nos quais a equipa imaginou que o videojogo decorresse, pois é o meio físico onde mais vezes se encontra este género narrativo. O uso de um livro obrigaria a utilização de gráficos 2D para o ambiente se manter coerente, graças a imagens em livros não possuírem uma verdadeira profundidade. Mas escolher implementar uma estética 2D provou também não ser a solução, pois foi notado que o género de *platformers* — sendo o género do *Ink Tales* — é geralmente construído em 2D, podendo-se considerar um mercado saturado com base na procura de videojogos na *Steam*[3] com as *tags* '*platformer*' e '2D', que resulta na apresentação de mais de mil títulos, ao invés de '3D' no lugar do '2D', onde só aparecem dez. Foi neste instante que se reparou que o uso de um livro como ambiente do *Ink Tales* não requereria obrigatoriamente que o seu estilo gráfico fosse em 2D. Isso deve-se à existência dos livros pop-up, que, graças à forma como usam um eixo adicional, permitem que o *Ink Tales* possa usar um estilo 3D sem enfraquecer a coesão entre a narrativa e o ambiente.

O uso desta perspetiva não é no entanto algo novo, existindo já no mercado alguns videojogos que trabalham com ela. Dois títulos que chamaram mais a atenção foram o *Stick It to the Man* (Zoink Games, 2013) e *Tengami* (Nyamyam, 2014). O primeiro título apresenta um ambiente onde o cenário é visivelmente feito de cartão e as personagem contam com uma finura que assemelha folhas de papel. Já o segundo exemplo — *Tengami* — trabalha mesmo com um ambiente que simula um livro de pop-up (Figura 3.4), ao ponto da criadora ter criado montagens físicas de papel do cenário e só posteriormente as implementou no videojogo, para garantir que resultaria num cenário possível de existir no mundo real (Schneidereit, 2015).

Foi com base nos exemplos acima apresentados e no uso de livros de pop-up físicos que começou a construção do ambiente deste projeto. Semelhante ao *Stick it to the Man*, o movimento no *Ink*

[2]Género de videojogos que promove uma grande liberdade de ação e exploração
[3]Plataforma de distribuição digital de videojogos

Figura 3.4: Ambiente de estilo livro pop-up no videojogo *Tengami*.

Tales limita-se maioritariamente a dois eixos relativos à câmara: o eixo X — esquerda e direita — e o eixo Y —cima e baixo — só ocasionalmente aproveitando o eixo Z — frente e trás — para a mudança de faixa de movimento, semelhante ao videojogo *Little Big Planet* (Media Molecule, 2008) (Figura 3.5).

Figura 3.5: Mostra de três faixas distintas onde o jogador se consegue mover no videojogo *Little Big Planet*. (gamespy.com, 2008)

Desta forma — e como a câmara virtual mantém sempre a mesma orientação — a construção do cenário está focada no eixo em que a personagem se move, sendo que a área no eixo Z é mais limitada. Tal não quer dizer que a sensação de profundidade não é importante na construção do ambiente do *Ink Tales*, antes pelo contrário. Para o ambiente do videojogo parecer devidamente um livro de pop-up, foi necessária uma maior atenção à povoação do espaço do cenário onde o jogador não se desloca, criando várias camadas de adereços (Figura 3.6).

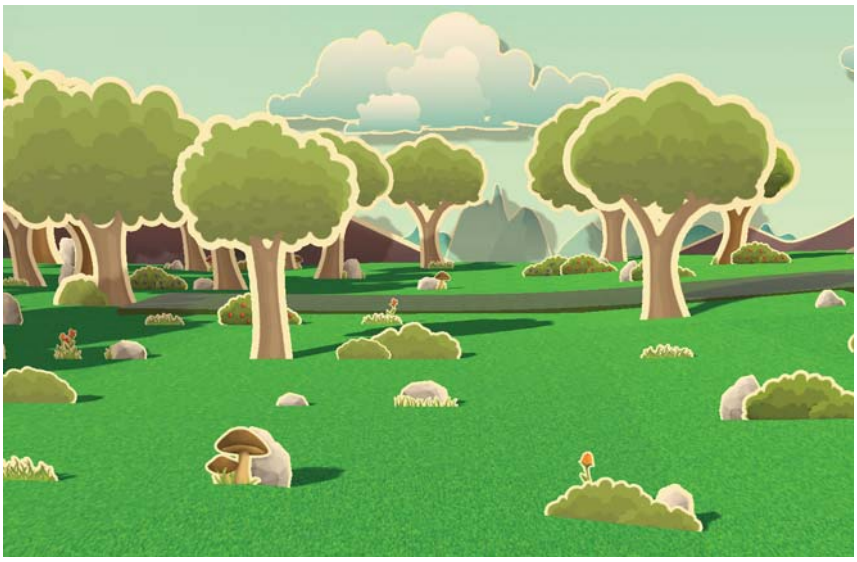

Figura 3.6: Camadas de adereços.

3.3.2 Movimento

O controlo físico de uma personagem, apesar de ser um elemento presente na maioria dos videojogos e às vezes considerado pelos jogadores como algo básico, é na realidade algo complexo ao qual desenvolvedores devem prestar muita atenção no desenvolvimento dum videojogo, pois é a ação mais realizada quando o movimento físico está presente, e caso esse movimento não seja agradável de ver e praticar, toda a restante experiência do videojogo pode ser manchada, decidindo o sucesso ou falhanço de um título, principalmente quando se trata de um *platformer*. Para a implementação de movimento no *Ink Tales*, inicialmente tentou-se criar algo o semelhante ao mundo real, onde o movimento horizontal é subjugado à inércia, não terminando abruptamente quando a direção é largada pelo jogador, e onde o movimento vertical simula uma parábola, em que o tempo de ascensão é idêntico ao de descensão. Apesar do movimento horizontal desde logo ter dado uma sensação de fluidez, o vertical mostrou-se demasiado leve, semelhante ao que se vê acontecer na lua. Na procura de uma solução para esse problema, foi feita uma pesquisa adicional que levou ao encontro de uma palestra (Pittman, 2016) e do livro *Game Feel* (Swink, 2009) e ambos apresentam uma prática que múltiplos videojogos aplicam no movimento vertical para parecerem mais responsivos e fluídos. Esta prática trata-se do aumento da gravidade na descensão, que apesar de não ir de encontro à realidade, títulos reconhecidos pelo seu controlo intuitivo a implementam na sua jogabilidade, com o *Super Mario Bros.* (Nintendo, 1985) a ser o videojogo mais aclamado neste departamento. Dentro deste exemplo é possível verificar que a descensão do *Mario* é mais rápida que a ascensão (Figura 3.7), onde num conjunto de 28 *frames*[4], a personagem só começa o seu movimento descendente a partir da décima nona *frame*, cerca de a 68% do tempo da animação.

[4]Sequência de imagens estáticas, que, ao serem mostradas sucessivamente a alta velocidade, criam a ilusão de movimento.

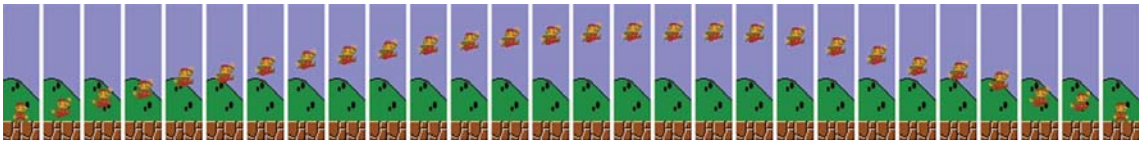

Figura 3.7: Sequência de *frames* no salto do Mario no videojogo *Super Mario Bros.* (Nintendo, 1985).
(BoardToBitsGames, 2017)

Outro elemento do salto presente em muitos jogos é a possibilidade de interromper a ascensão da personagem, mais habitual no género *plataformers*, o que permite controlar a altura que um salto alcança, dependente do tempo de pressão no botão correspondente. Quando o botão de saltar é mantido durante toda a ascensão da personagem, esta efetua o maior salto que lhe é possível, mas se a qualquer instante durante a subida o botão correspondente é largado, a força ascendente é reduzida ou até anulada, permitindo uma queda antecipada.

Com base nesses conhecimentos adquiridos, e para o movimento da personagem principal do projeto ser o mais natural possível, os elementos anteriormente apresentados foram emulados no projeto *Ink Tales* (Figura 3.8), resultando num salto onde a descensão começa no décimo quarto frame no total de vinte e dois, cerca de 63% e provando à equipa serem uma melhor solução relativamente à mais realista.

Figura 3.8: Sequência de *frames* no salto da personagem no Ink Tales.

Uma última mecânica de movimento desenvolvida adicionalmente para o *Ink Tales* foi um sistema de deslizamento de colinas. Esta mecânica ativar-se-ia caso a personagem do jogador se apoiasse sobre uma superfície com uma inclinação de 50º ou superior, e serviria para limitar a passagem de certas áreas, requerendo abordagens adicionais para a travessia de níveis. Durante o deslizamento, é impossível ao jogador controlar o movimento horizontal da personagem — mesmo quando salta — até esta atingir um solo que não ative mais deslizamento, de modo a simular a aceleração que um corpo tem nestas situações, e também para impedir que o jogador tente atravessar áreas que são desenhadas para limitar essa deslocação. Apesar do nível demonstrativo do projeto *Ink Tales* prévio ao atual ter sido desenhado com uma inclinação sobre a qual esta mecânica podia ser observada, o nível da demonstração do projeto atual não possui nenhuma colina que permita mostrar esta mecânica em ação, devido à temática narrativa não o permitir. Mesmo assim, é possível ver os blocos de código responsáveis pela execução desta mecânica nas figuras em anexo A.24.

3.3.3 Desenho

De acordo com a narrativa de *Ink Tales*, o mundo das fábulas está a desvanecer e um dos objetivos da personagem principal é restaurar esse mundo. Para tal, é-lhe dado uma ferramenta que mecanicamente permite ao jogador desenhar livremente na área de jogo, e que, caso o desenho seja reconhecido como uma figura anteriormente desbloqueada, e. g. um quadrado, esse desenho é prontamente substituído por um objeto com uma forma correspondente, e com qual a personagem consegue interagir.

Inicialmente, foi proposta uma maneira de reconhecer um desenho como uma figura através dos vértices que o desenho possuiria. Para reconhecer o número de vértices que o desenho teria, desenvolveu-se desde raiz um detetor de orientação de aresta, e uma margem de tolerância com uma forma triangular que, caso trespassada, incrementava uma unidade à variável que conta o número de vértices do desenho, e após isso voltava a calcular a linha de orientação da aresta e tolerância repetidamente até o jogador acabar de desenhar (Figura 3.9).

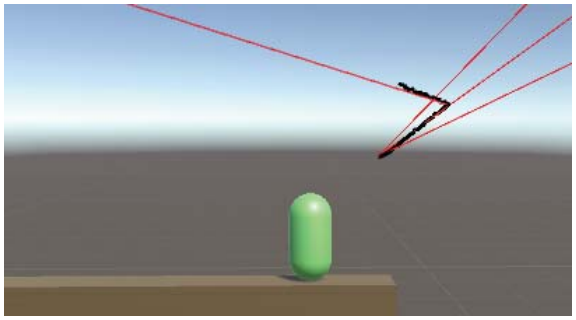

Figura 3.9: Detetor de vértices numa fase inicial do projeto.

Nesta fase de desenvolvimento do algoritmo, apesar de teoricamente não se notarem falhas, houve uma grande dificuldade de implementação dessas ideias, principalmente no cálculo de orientação das arestas posteriores à primeira, e foi por volta dessa altura que foi encontrado o algoritmo *$P Point-Cloud Recognizer*, também este capaz de identificar desenhos, como já descrito no sub-capítulo 3.2. Com esta ferramenta, foi possível — de uma forma mais fácil e robusta — criar um sistema de reconhecimento de desenhos, desta vez com o auxilio de uma base de dados, que foi preenchida pela equipa do *Ink Tales* com os seus próprios desenhos de figuras, para serem posteriormente usados como comparação a qualquer desenho feito pelo jogador. Após uma figura da base de dados ser reconhecida como a mais semelhante à desenhada pelo jogador, o programa devolve o nome dado ao desenho pré-feito quando foi inserido na base de dados. Para a construção da base de dados, o *$P Point-Cloud Recognizer* conta com uma interface composta somente por uma simples caixa de texto e um botão. É através dessa interface que é possível associar um nome a um desenho, e esse nome pode depois ser processado por outros *scripts* para correr eventos ou verificar condicionais, como é o que acontece dentro do *Ink Tales*.

Todo o sistema de reconhecimento de desenho tem como objetivo a criação de objetos com os quais a personagem jogável pode interagir, incluindo agarrar e arrastar, e após um desenho ser substituído por uma figura, esta torna-se suscetível à gravidade, mas antes é calculado o tamanho do desenho para ser substituído por um objeto com dimensões correspondentes, de forma a fazer com que a transformação pareça o mais autêntico possível, ajudando na manutenção da imersão do jogador 3.10.

O videojogo conta de momento com três figuras que podem ser reconhecidas: um traço horizontal, um quadrado e um triângulo retângulo, sendo que este último têm duas variantes, onde o ângulo de noventa graus pode-se encontrar no canto inferior esquerdo ou no direito, conforme o desenho é feito. No cálculo das dimensões das figuras, tanto nos triângulos como no quadrado, são obtidas as coordenadas dos pontos de maior e menor valor no eixo horizontal, é medida a distância entre os dois, e o mesmo repete-se no eixo vertical. Já no traço horizontal, a única dimensão medida é a do eixo X, pois para esta não se assemelhar a um retângulo quando tem um comprimento elevado, é necessário que lhe seja dada uma altura fixa no eixo Y.

Por último, é necessário ter em consideração que cada pessoa desenha de forma única, foi necessário criar múltiplas variações para que todos os desenhos dos jogadores sejam devidamente identificados. Uma pessoa geralmente começa o desenho de uma figura geométrica a partir de um vértice e qualquer direção, portanto, calculando o número de vértices de cada figura e duplicando esse valor para cada direção que é possível começar o desenho, concluiu-se que o quadrado contará com oito variações e o triângulo com seis. O traço horizontal foi um caso especial, pois apesar de ser possível desenhar uma simples linha da esquerda para a direita ou vice-versa, também foi considerada a possibilidade de desenhar um retângulo com uma altura muito limitada, pois o objeto resultante desse desenho tem ele próprio uma certa altura, mesmo que estreita, acabando o traço por ter dez variações, oito derivadas do retângulo, mais as duas da linha. Independentemente da figura, a base de dados foi criada tendo em mente a possibilidade de ligeira inclinação nos desenhos, pois é irreal esperar que todos os jogadores desenhem todas as figuras de forma perfeita, e sendo o objetivo do sistema a compatibilidade com o maior número de pessoas possíveis, a base de dados foi construída por um grupo composto de dez pessoas, dos dezasseis aos cinquenta e sete anos, com dez exemplos para cada variação de cada figura, acabando a base de dados por ter no final duzentos e quarenta exemplos.

3.3.4 Diálogo

Para além da existência de animais antropomórficos e morais no final do conto, as fábulas também contam geralmente com diálogos entre as suas personagens, que, em conjunto com as suas ações, apresentam ao leitor as suas personalidades. Sendo o *Ink Tales* um videojogo

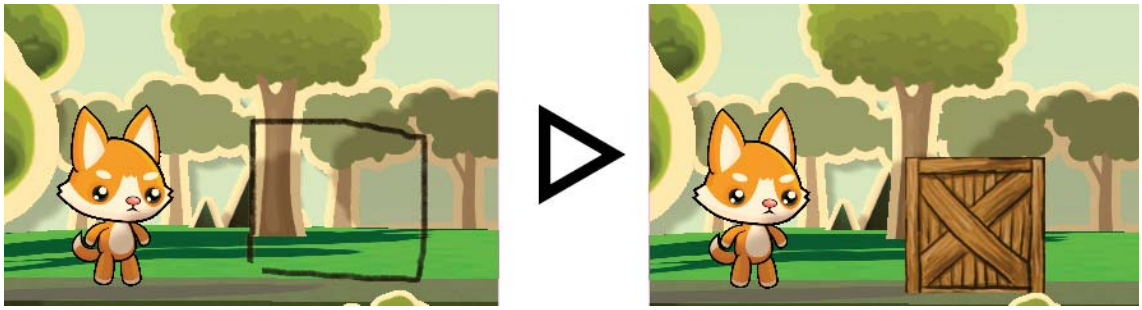

Figura 3.10: Reconhecimento de desenho e posterior substituição por um objeto.

bastante focado no seu conteúdo narrativo, e por conter em si múltiplas fábulas, o diálogo foi desde o início uma das características chave que este videojogo tencionou ter.

Tendo em mente o que Lee falou no seu blog (Lee, 2013), a equipa desenvolvedora do *Ink Tales* quis que fosse possível o jogador nunca perder a agência, mantendo sempre ativo este elemento que diferencia os videojogos dos restantes meios artísticos. Neste projeto, a manutenção de agência infiltrou-se no sistema de diálogo, pois onde na grande maioria dos videojogos em que existem diálogos entre personagens o jogador perde o controlo físico quando está no meio de uma conversa, no *Ink Tales* esta agência mantém-se, podendo o jogador andar e saltar enquanto conversa e até fugir da mesma, à qual os *NPCs* reagem de acordo com a estranheza da ação, resultando em diálogos diferentes na próxima vez que são abordados, de modo a criar um mundo mais entrelaçado e credível. A agência, no entanto, não se manifestou somente sobre o controlo físico da personagem controlável, também sendo visível no diálogo em si, pois é dado ao jogador a possibilidade de escolher, entre várias frases curtas, aquela que servirá de resposta ao que está a ser falado pelos *NPCs* presentes na conversa. Essas escolhas que o jogador toma, por sua vez, influenciam porções grandes ou pequenas da narrativa do videojogo, denotando ao jogador que todas as escolhas importam e que todas elas trabalham em conjunto para a criação de uma resolução moral.

No momento de conceção desta mecânica, já era do conhecimento da equipa títulos que têm sistemas de diálogo semelhantes ao pretendido implementar neste projeto, destacando-se o *Oxenfree* (Night School Studio, 2016) e *Night in the Woods* (Infinite Fall, 2017), ambos videojogos onde o diálogo é a mecânica principal que permite ao jogador escolher o que dizer entre um número de opções predefinidas e onde o diálogo é o que faz avançar a narrativa do videojogo. Ao estudar como o *Night in the Woods* implementou o seu sistema de diálogo, foi descoberta uma ferramenta *open source*[5] — de nome *Yarn Spinner* — que foi desenvolvida especificamente para o videojogo, que é capaz de interpretar árvores de diálogo presentes na interface *Yarn* e de apresentá-las como uma sequência de texto no *Unity*. Ao se perceber as funcionalidades que esta ferramenta dispõe, como já anteriormente falado no capítulo 3.2, a equipa do *Ink Tales* viu-a como um *asset* essencial para a implementação rápida de um sistema robusto e complexo

[5] *Software* de livre acesso para estudos, modificações, e distribuições sem restrições de direitos de autor

o suficiente para possibilitar a construção de árvores de diálogo capazes de receber e enviar dados de variáveis de e para o *Unity*. Apesar das suas funcionalidades, o *Yarn Spinner* não dispõe de uma interface predefinida no *Unity* para mostrar os diálogos oriundos do *Yarn*. Para tal, foi necessário construir uma de raiz no *Unity*, e daí — através de um *script* — puxar frase a frase do *Yarn* através do *Yarn Spinner*. Na ideação de como o diálogo iria aparecer no *Unity*, a maior influência foi o videojogo *Oxenfree*. Neste título, as falas das personagens normalmente aparecem no fundo do ecrã, como é geral nos videojogos, para não obstruir acontecimentos que estejam a decorrer no centro do ecrã, onde o foco de ação normalmente se encontra. No entanto, quando são dadas escolhas de falas ao jogador, estas aparecem como balões de fala diretamente por cima da personagem jogável (Figura 3.11). Tal escolha de design foi provavelmente implementada para o jogador não ter de desviar o olhar de onde as personagens se encontram, pois assim como no *Ink Tales*, a personagem jogável no *Oxenfree* é controlável em todos os instantes, nunca sendo necessário parar para falar. Esta colocação dos balões de fala por cima da personagem jogável permite que o jogador consiga ver facilmente o que pode dizer, e adicionalmente ver onde se situa fisicamente, sem necessitar de olhar para outros pontos no ecrã. Assim sendo, o *Ink Tales* conta com uma apresentação de diálogos bastante semelhante à do *Oxenfree*, pois essa apresentou uma solução ideal para o problema do distanciamento entre os momentos de diálogo e os de ação.

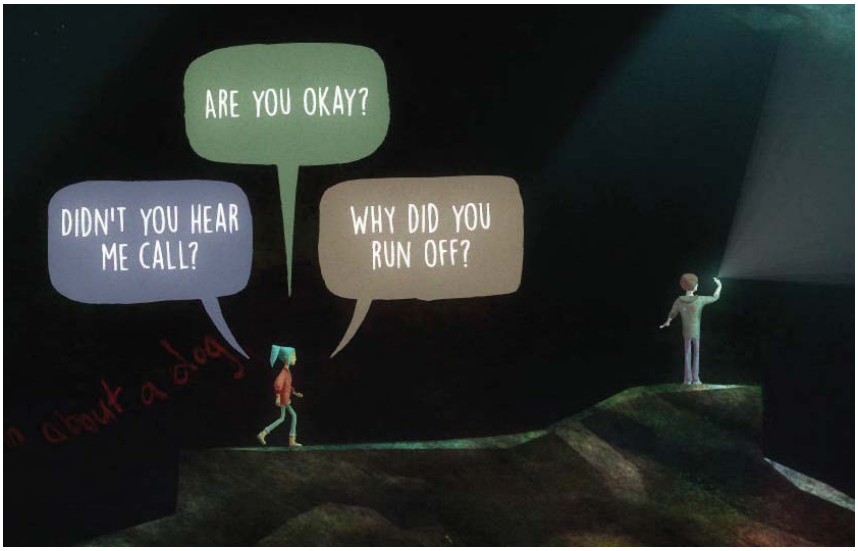

Figura 3.11: Escolhas de falas no videojogo *Oxenfree* (engadget.com, 2016)

O sistema de balões de fala foi desenhado para interpretar os dados enviados pelo *Yarn* antes de mostrar a interpretação no *Unity*. Tendo em conta que nem todos os diálogos contam sempre com três opções de resposta — o maior número de opções possíveis — foi necessário programar o sistema de fala de modo a só fazer aparecer o número de balões igual às opções de fala que o *Yarn* disponibiliza (Figura 3.12).

É graças à troca de informações entre o *Yarn* e o *Unity* — através do *Yarn Spinner* — que é possível

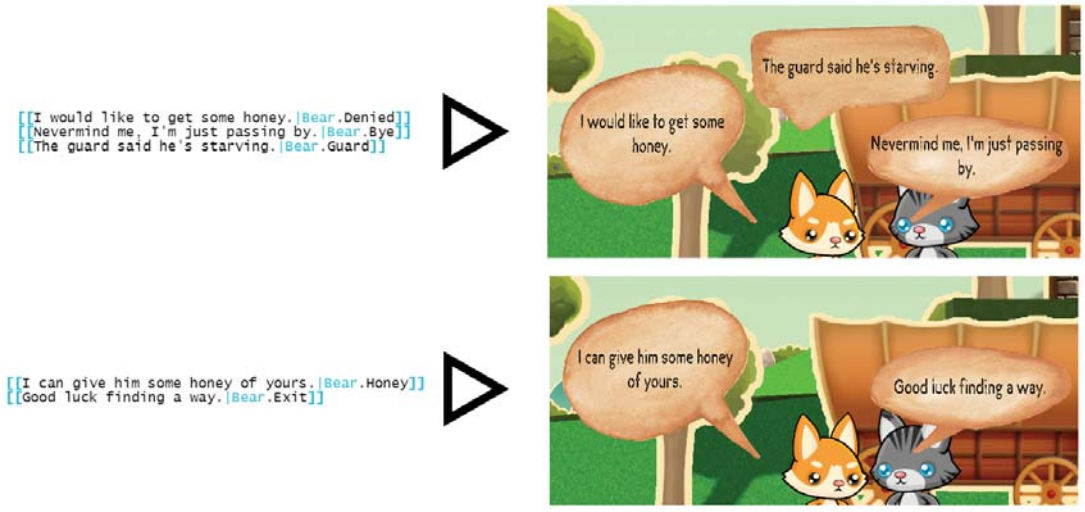

Figura 3.12: Variação do número de balões de fala no *Ink Tales*.

ter este sistema capaz de mostrar diálogos distintos conforme várias condições no mundo do jogo, assim como permite moldar esse mesmo mundo conforme as escolhas feitas nas interações verbais com os vários *NPCs*, tudo isto para no final criar um mundo mais interligado entre os seus elementos, dando a ideia de algo mais real, promovendo a suspensão de descrença e a imersão do jogador.

3.3.5 Design de Níveis

A demonstração do projeto conta com um nível que corresponde a uma zona inicial da história completa do *Ink Tales*, tratando-se duma aldeia perto de onde o lobo — o personagem principal — mora. Por ser uma zona inicial e rural, este nível pretende transmitir uma sensação de serenidade e segurança, com muitos elementos da natureza e nenhum perigo, contando também com a ajuda de sons e música para criar esse efeito. Essa ideia inicial resultou no *concept art* visível na figura 3.13.

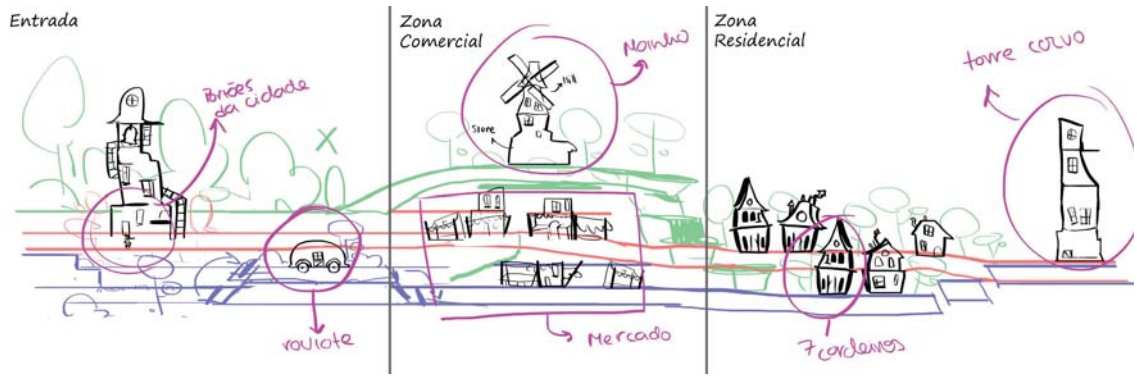

Figura 3.13: Concept art do nível demonstrativo do *Ink Tales*, desenhado por Mafalda Claro.

Como foi apresentado no sub-capítulo 3.3.1, todo o ambiente do jogo assemelha-se a um livro *pop-up*, incluindo este nível de demonstração (Figura **??**). Para acentuar a ideia deste tipo de ambiente, vários adereços no cenário erguem-se quando entram em cena, simulando a elevação de imagens quando se abrem páginas em livros *pop-up*. Para ajudar o jogador a guiar-se no ambiente virtual, o *Ink Tales* empresta uma conhecida técnica de design chamada *color coding*, que se trata do uso de colorações em objetos que sobressaem do resto do ambiente para chamar a atenção do jogador, muitas vezes criando um simbolismo da cor que toma um significado específico para ele saber instintivamente do que se trata. No *Ink Tales*, esta técnica foi aplicada na área onde o jogador pode andar, separando-a do resto do cenário, como é visível na figura 3.14 e 3.15.

Figura 3.14: Zona inicial do nível criado.

Figura 3.15: Zona final do nível criado.

Numa perspetiva técnica, o nível é composto por três planos distintos que formam os caminhos pelos quais o jogador caminha, incluindo cinco conectores localizados em vários pontos ao longo do nível que possibilitam a deslocação entre planos, permitindo ao jogador explorar e abordar os desafios do nível em qualquer ordem desejada. Tematicamente, o nível é constituído por três áreas: a entrada, a zona comercial e a zona residencial (Figura 3.13), e onde cada uma delas tem o objetivo de introduzir cada mecânica do jogo, através de *puzzles*, os quais são falados em detalhe no sub-capítulo 3.3.6.

O *design* de níveis não se limita à construção de um cenário, também é preciso povoá-lo. Para

ser fiel à ideia de uma aldeia, os *NPCs* presentes no nível são de número reduzido, estando a maioria deles concentrados na zona comercial, devido à feira que decorre nesse instante. Para além de existirem para criar um ambiente vivo e dinâmico, estes *NPCs* têm o propósito de expor ao jogador a relação que o lobo tem com o resto das personagens, através de diálogos entre elas e o jogador. Pode-se considerar que existem na aldeia quatro variedades de *NPCs*, cada uma delas com funcionalidades diferentes. O primeiro *NPC* que aparece em cena é um guarda que não deixa o lobo entrar na aldeia. Esta é a primeira interação em diálogo que o jogador pode ter e onde lhe é apresentado o status social que o lobo tem naquela zona através das justificações que o guarda dá para não o deixar passar. A segunda variedade de *NPCs* são os aldeões. Este conjunto de personagens, todos localizados no centro da aldeia têm o único propósito de relembrar ao lobo porque é que ele não é bem-vindo na aldeia. Estes não tentam expulsar fisicamente o lobo mas avisam-no constantemente que estão de olho nele e ele devia ir embora. A terceira variedade de *NPCs* são os cabritinhos. Estas personagens são o objetivo de o lobo encontrar-se na aldeia, pois de acordo com a fábula d'*O Lobo e os Sete Cabritinhos*, o lobo tenta entrar na casa das crias enquanto a mãe delas se encontra no mercado, mas este não consegue enganar os cabritinhos devido ao seu pêlo escuro e à sua voz grossa, opostos à mãe cabra. Para corrigir isso, o lobo arranja farinha, com a qual cobre os pés, e come mel para suavizar a voz, conseguindo assim iludir com sucesso as crianças e invadir a sua casa. No *Ink Tales*, a história varia ligeiramente para ser possível adaptar a um videojogo, mas o objetivo inicial de ir de encontro aos cabritinhos e posteriormente dos objetos é mantido. O último *NPC* chave presente no nível é o corvo, que será o animal responsável por dar ao lobo a ferramenta mágica que permite ao jogador usar a mecânica de desenho. Na realidade, o corvo é uma representação animal do narrador que acompanha a história do jogo, e o qual aparece esporadicamente ao lobo para o auxiliar em momentos-chave da história, tomando um papel semelhante ao de um mentor para o personagem principal.

Todo o design do nível foi tido em mente com o objetivo de o jogador conseguir experimentar — mesmo que por breves momentos — todas as características e mecânicas-chave do *Ink Tales*: movimento, exploração, diálogo, recolha de objetos e desenho. Para além disso, este nível insere o jogador num instante da narrativa onde é possível perceber as regras do mundo, a situação da personagem que controla, e despertar curiosidade sobre como a história se desenrolará após o final da demonstração.

3.3.6 Puzzles e eventos

O puzzle principal contido nesta demonstração do projeto é composto por vários desafios pequenos que usam as várias mecânicas do jogo, estando todas relacionadas com o objetivo principal do nível que é recriar parte da fábula *O lobo e os sete cabritinhos*. Os episódios da fábula

replicados são a ida do lobo à casa dos cabritinhos e a recolha de mel e farinha para estes lhe abrirem a porta.

Apesar de a ida à casa dos cabritinhos ser o que motiva o lobo a ir à procura da farinha e do mel, o jogador pode em qualquer instante recolher estes objetos sem precisar de ir previamente à casa das crias ovinas, quer seja por acaso ou por já conhecer a fábula aqui presente, mas, quer seja antes ou depois de o lobo saber o que tem de coletar, os objetos não aparecem destacados no ambiente, e não há qualquer efeito especial que indique onde estes objetos se situam ou como se obtém.

Para saber como obter os objetos, é necessário falar e ter atenção ao que os *NPCs* dizem, pois todos eles apresentam pistas sobre como o lobo pode encontrar o que procura. Mesmo que seja possível apanhar os objetos em qualquer ordem, o mais fácil de descobrir e o que aparece primeiro na fábula original é o mel, e no *Ink Tales*, este encontra-se na *roulotte* do urso. Para o apanhar, existem dois caminhos que o jogador pode tomar. O primeiro é tentando convencer o urso a dar o mel caso o jogador tenha falado previamente com o guarda e tenha descoberto que ele está esfomeado. Outra forma de apanhar o mel é roubando-o diretamente da *roulotte*, colocando-se o lobo na lateral do veículo onde o urso não o vê, e isto torna visível um elemento da interface que indica que é possível roubar o mel.

Figura 3.16: Elemento de interface que indica a possibilidade de roubar o mel.

A farinha prova ser mais difícil de encontrar, pois encontra-se nos sacos ao pé do moinho no topo da aldeia, um lugar mais desviado do caminho principal. Assim como o mel, não há nenhum efeito visual que destaque a farinha do resto do ambiente, e como elemento adicional de

dificuldade, não aparece nenhum elemento de interface quando o jogador se situa no espaço certo. Para o jogador saber que consegue apanhar a farinha nos sacos ao pé do moinho, pode fazê-lo através da dedução própria, associando o moinho à fabricação de farinha e os sacos ao recipiente onde normalmente se armazena farinha, mas outro método mais amigável e que faz uso duma das mecânicas base do *Ink Tales* e através do diálogo com várias personagens no nível. O guarda no início do nível, após afirmar que não pode deixar o lobo passar, exclama que não teve a oportunidade de tomar o pequeno-almoço antes de entrar no posto de vigia, e que está cheio de fome ao ponto de comer qualquer coisa. Um aldeão também aponta que o guarda tem o hábito de ir a correr para o posto de trabalho com uma fatia de pão com mel na boca. Estas informações ajudam o jogador a perceber a ligação entre o guarda e o mel, e também faz com que sejam desbloqueadas falas adicionais com o urso da *roulotte* que permitem ao lobo pedir o mel para oferecer ao guarda. Com o mel já no inventário, ao conversar com o guarda, é apresentada a opção de oferecer algum mel para lhe saciar a fome. Após parte do mel ser oferecida, o guarda devaneia como gosta de pão com mel, e que precisa de ir buscar os seus sacos de farinha ao pé do moinho que mandou fazer, para poder fazer o seu pão. Após sabida esta informação já é clara a localização da farinha para o jogador.

Em qualquer instante, o jogador por ir à casa dos cabritinhos, os quais expõem falas diferentes conforme os objetos coletados pelo lobo, e dão indicações indiretas do que falta fazer para a história ir de acordo com a fábula. Se o jogador se dirigir à casa sem nenhum objeto ou só com a farinha, os cabritinhos exclamam que a voz de quem fala não é da mãe deles e não abrem a porta. Se o lobo bate à porta só com o mel, os cabritinhos afirmam que conseguem ver os pés escuros de lobo por debaixo da porta, opostos aos pés brancos da mãe e reafirmam que não abrem a porta. Mas se o lobo se dirigir à casa dos cabritinhos quando já tem os dois objetos, a porta já se encontra aberta e nesse momento são vistos vários cabritinhos fisicamente deformados e a fugir em direção oposta ao resto da aldeia.

Com o barulho dos cabritinhos a fugir mas sem verem nada, os aldeões ocupam o caminho entre a casa das crias e o resto da aldeia, acreditando que o lobo fez algo mal e impedem-no de voltar para trás, forçando-o a ir em direção à torre do corvo. A torre que antes se encontrava fechada e sem ninguém lá dentro, conta agora com o corvo à porta e este oferece ao lobo uma caneta mágica capaz de criar objetos a partir de desenhos, para este conseguir fugir da ira dos aldeões.

O lobo consegue a partir desse momento criar caixas que usa para trepar a torre do corvo. Ligado ao design do nível, existem nesta torre um tipo de plataformas presentes em inúmeros *platformers*, que podem ser trespassadas ao serem subidas mas que se tornam sólidas quando o jogador já se encontra no seu topo. Em alguns videojogos é também possível saltar para baixo destas plataformas no mesmo plano que se sobem, somente precisando da combinação dos botões de saltar e da direção para baixo, algo também possível de fazer no *Ink Tales*. No

final, o lobo acaba por fugir ao saltar do topo da torre do corvo para o bosque próximo, e assim são terminados os eventos e puzzles presentes neste projeto.

3.3.7 Interface

Com o objetivo de querer manter o jogador imerso na narrativa do videojogo, a interface foi desde o início pensada e desenhada de modo a ser o menos intrusiva possível, tomando um aspeto minimalista mas ainda assim fácil de perceber. Um videojogo que influenciou bastante a tomada deste estilo gráfico foi o *Life is Strange* (Dontnod Entertainment, 2015), que semelhante ao *Ink Tales*, tem um grande foco na apresentação duma narrativa e nas relações entre as personagens, e onde elementos da interface só aparecem quando é necessário transmitir informação pertinente ao momento, notando-se que numa grande parte do videojogo não são visíveis nenhuns elemento de interface no ecrã.

Uma das maneiras que a interface é usada no *Ink Tales* é na exibição dos objetos que o jogador possui, tanto para relembrar ações que já foram feitas, assim como para servir de pista do que fazer a seguir, caso o objetivo esteja relacionado com esses mesmos objetos. Graficamente, os objetos em inventário aparecerão no canto superior esquerdo do ecrã, com um estilo semelhante a um esboço de uma só cor, assim como no *Life is Strange* (Figura 3.17), de maneira a não roubar muito a atenção do jogador e possivelmente quebrar o *flow* do momento. Devido aos objetos serem usados brevemente após serem coletados, os seus ícones mantêm-se sempre visíveis no canto do ecrã, mas tomam um tom translúcido enquanto o jogador se move pelo mapa, de modo a não interferir tanto enquanto este toma ações alheias aos objetos, mas deixando a possibilidade de serem vistos a qualquer instante caso o jogador decida olhar para o canto em que eles se encontram. Após serem apanhados, os ícones só voltam a tornar-se opacos quando o jogador pára por uns segundos, de modo a podê-lo ajudar na reflexão sobre o que tem que fazer dado os itens que têm em posse.

O outro objetivo da existência da interface no *Ink Tales* — e mais frequente de observar — é a indicação de ações possíveis de iniciar. Este elemento aparece por cima da personagem principal sempre que existe uma ação ou diálogo que pode ser iniciado nesse momento. O topo da personagem foi escolhido como lugar para este elemento de interface com o objetivo de tornar mais fácil o jogador ver quando pode interagir com algo sem precisar de deslocar a visão para pontos afastados de onde a sua personagem se encontra. Graficamente, essa notificação é apresentada através da palavra que define da ação, e. g. 'conversar', e o tipo de letra com que as palavras são exibidas tem um estilo semelhante aos traços aplicados nos desenhos dos objetos em inventário, resultando numa interface coerente e minimalista.

Figura 3.17: Estilo gráfico dos objetos que estão na posse da protagonista.

A grande maioria das interfaces nos videojogos estão unidas ao ecrã de jogo, ocupando normalmente as suas bordas e mantendo-se estáticos no seu lugar, mas existem também interfaces que tomam proveito do espaço de jogo para criar efeitos que fogem à norma. No *Ink Tales*, tanto os balões de fala como o indicador de ação estão dentro da área do jogo, onde o cenário e as personagens se encontram, graças à construção de um *canvas*[6] ligado à personagem principal. Esta conexão com uma personagem dentro do ambiente de jogo permite dar aos balões de fala um visual semelhante ao resto do cenário, parecendo também eles elementos de um livro *pop-up*, os quais têm um efeito flutuante enquanto pairam por cima do lobo, criando um ambiente mais vivo e interligado, promovendo assim a suspensão de descrença do jogador.

Apesar de não fazer parte da interface em si, mas por mérito de ser a lente pela qual todo o jogo é visto, a câmara também é um elemento essencial do projeto *Ink Tales*. Esta câmara foi programada de modo a poder trabalhar múltiplos ângulos e distâncias na apresentação da interface e da ação do jogo. Apesar do cenário e as personagens manterem a sua orientação independentemente da posição da câmara, para transparecer a aparência de um mundo feito de papel, o *canvas* que acompanha o protagonista foi programado para estar sempre orientado na direção da câmara, independentemente de onde esta se encontre, para não dificultar a seleção de opções de fala, assim como a leitura das mesmas e do indicador de ação (Anexo A.20.). Para além de seguir a personagem principal com um movimento suave, a câmara do *Ink Tales* foi programada para conseguir desligar-se desta e colocar-se em qualquer ponto fixo no espaço (Anexo A.3.) quando certas condições são atendidas. Esta perspetiva alternativa permite apresentar áreas de um ângulo diferente, onde é mais benéfico focar o cenário em vez das personagens, sendo exemplo uma área graficamente mais trabalhada e onde não existem muitas ações para além da sua travessia, ou no caso de o jogador encontrar-se num puzzle onde é benéfico, ou até necessário, ver uma maior área do que o que a câmara normalmente oferece no seu modo predefinido.

[6] Área onde são colocados elementos de interface.

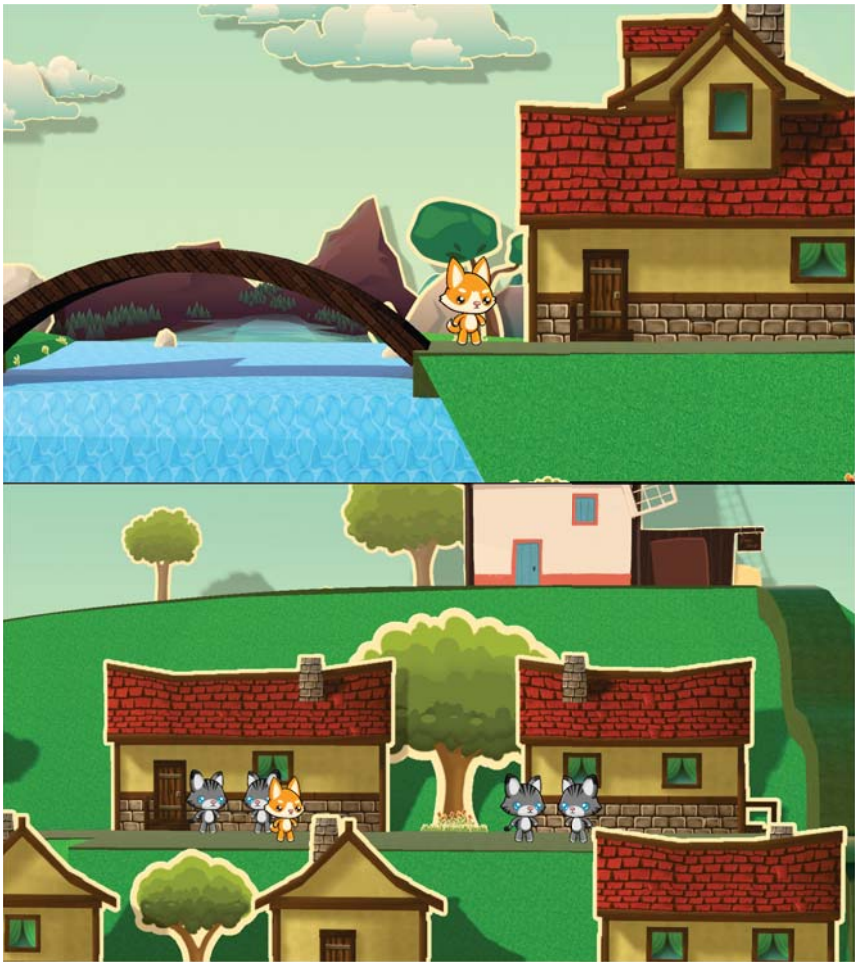

Figura 3.18: Diferença da perspetiva da câmara quando esta foca a personagem, e quando foca uma porção do nível.

3.4 Resultado Final

No final, o projeto *Ink Tales* pôde contar com uma demonstração que, apesar de só conter uma área, permitiu mostrar todas as mecânicas base desenvolvidas, que serão aplicadas em todo o projeto, onde foi possível montar uma área que transparecesse claramente a ideia de um mundo dentro de um livro *pop-up*, com uma interface simples mas capaz de cumprir o seu papel de guiar o jogador. Com mais de meia dúzia de *NPCs*, uma área com vários caminhos para explorar, múltiplos puzzles para resolver, e tudo isto com o objetivo de criar uma adaptação mais interativa de uma fábula, o estado deste projeto reflete — numa pequena fatia — a experiência que o jogador pode esperar ter no videojogo *Ink Tales* quando este estiver completo.

Figura 3.19: Área inicial onde o jogador começa.

Figura 3.20: Entrada da aldeia e área de interação com o guarda.

Figura 3.21: Rio da aldeia e introdução ao *platforming*.

Figura 3.22: Roulotte do urso onde é possível adquirir o mel.

Figura 3.23: Praça principal da aldeia onde se situa o mercado e onde é possivel falar com múltiplos aldeões.

Figura 3.24: Moinho de vento no topo da colina onde é possível adquirir a farinha.

Figura 3.25: Casa dos sete cabritinhos onde ocorre o incidente que inicia a aventura do videojogo.

Figura 3.26: Entrada da torre do corvo onde é adquirido o poder de desenhar.

Figura 3.27: Plataformas trespassáveis e topo da torre do corvo.

Capítulo 4

Conclusão

Com esta dissertação, pôde-se chegar à conclusão que os videojogos requerem uma diferente abordagem na apresentação de histórias graças à sua natureza distinta dos outros meios narrativos. Devido a serem um meio interativo, os videojogos permitem que os jogadores tomem um papel bastante ativo nos eventos duma narrativa, conseguindo estes imergirem-se na obra de uma maneira única devido a estados como o *flow* que requerem que o utilizador se encontre em ação, oposto ao estado passivo que todos os outros meios artísticos obrigam os utilizadores a tomar. Para tirar-se proveito do elemento de interatividade único que os videojogos possuem, é importante saber as condições em que não se justifica o sacrifício da agência, mas também saber quando a remoção parcial ou total da mesma acaba por oferecer uma melhor experiência ao jogador. As *cutscenes*, por exemplo provaram, através dos estudos feitos, que são um melhor método na mostra de detalhes nas personagens ou cenário do que momentos jogáveis. Ao encontrar este equilíbrio, tornou-se possível o desenvolvimento dum projeto que reflete estas aprendizagens teóricas de modo a testar como é que elas conseguem acrescentar valor comercial e artístico a um videojogo.

Graças a um maior foco na manutenção da agência do jogador — deixando este ter sempre controlo completo sobre a sua personagem e como ela responde a interações — foi possível no *Ink Tales* estimular estados psicológicos como o *flow* e a suspensão de descrença, que por sua vez promovem a imersão do jogador no videojogo e acima de tudo, na sua narrativa. Ao serem implementadas decisões de design que requerem que o jogador tenha uma postura mais ativa — como a escolha de diálogos — a narrativa deste videojogo tem uma maior conexão com o jogador, pois é-lhe permitido expressar-se através da sua personagem e ver como o mundo fictício reage, resultando numa experiência narrativa mais pessoal e impactante.

4.1 Dificuldades Encontradas

Tanto no desenvolvimento da dissertação como no projeto, o autor deparou-se com vários obstáculos, os quais apesar de terem sido maioritariamente superados, requereram um grande período de tempo para serem resolvidos.

Na parte teórica, foram estudados tópicos desconhecidos ao autor, principalmente no ramo da psicologia, para se perceber os estados psicológicos que o ser humano experiencia quando se imerge tanto em obras fictícias como em atividades, e porque as procura consumir. Assim como a psicologia, o estudo da narrativa e como esta se insere nos videojogos foi uma matéria não familiar para o autor, sobre a qual tiveram de ser feitas pesquisas aprofundadas sobre a sua origem, a estrutura em que normalmente esta se apresenta e como varia conforme o meio em que é apresentada. Devido à falta de experiência do autor na escrita de documentos de carácter académico, muitas vezes notou-se uma dificuldade em transparecer ideias para a forma escrita.

No desenvolvimento do projeto, apesar da sua programação estar mais dentro do ramo de formação do autor, também foram encontradas várias dificuldades não esperadas. Como já foi apresentado no sub-capítulo 3.3.3, o reconhecedor de desenho desenvolvido de raiz pelo autor acabou por não ter sido implementado, pois ainda que a sequência de lógica no seu desenvolvimento aparentasse estar correta, a implementação da mesma mostrou-se difícil ao ponto de eventualmente ter sido substituída por um *asset* capaz de fazer tudo que a ferramenta do autor tinha o objetivo de fazer. Ainda no projeto mas já desligado da programação, o design do nível foi mais complicado do que o esperado, maioritariamente devido à falta de experiência nessa área. Foi só com a ajuda do outro membro da equipa, através de um esboço da ideia que o nível seria (Figura 3.13), que o autor conseguiu construí-lo dentro do motor de videojogos. Por último, outro grande obstáculo foi a união entre elementos 2D e 3D no ambiente do jogo. Ainda que graficamente não tenha havido problemas nesta área, mecanicamente estas duas classes de objetos não conseguem naturalmente interagir no *Unity*, principalmente no âmbito das colisões, trespassando sempre um pelo outro. Adicionalmente, componentes que só são compatíveis com objetos 2D, como um animador de *sprites*, tiveram de ser trabalhados para poderem ser implementados num objeto 3D, havendo uma insuficiência de documentação sobre o tópico devido à raridade da tentativa de junção de elementos 2D e 3D.

4.2 Passos Futuros

Sendo que o projeto já conta com todas as mecânicas base em funcionamento, e a história do videojogo já tem um plano de eventos-chave escritos, o próximo passo da equipa é continuar a desenvolver mais áreas que continuem a história do lobo no mundo das fábulas. Ao mesmo tempo, serão procuradas mais maneiras de poder instalar as teorias da ligação entre as mecânicas e a narrativa estudadas e testar com utilizadores se essas implementações são notadas ou se mostram não ter efeito. Olhando também para outros videojogos, e como estes são recebidos pelo público e críticos, buscar-se-ão inspirações para o desenvolvimento de elementos ainda não presentes.

Bibliografia

2K Boston (2007). Bioshock. [Windows PC, XBox 360, PlayStation 3, Mac OS, iOS, PlayStation 4, XBox One]. Califórnia, Estados Unidos da América: 2K Games.

3djuegos.com (2012). Análisis de journey. https://www.3djuegos.com/juegos/analisis/9032/0/journey/. Última vez acedida a 16/09/2018.

Aarseth, E. (1997). *Cybertext: Perspectives on Ergodic Literature*. ISBN: 0801855780. The John Hopkins University Press.

BioWare (2007). Mass effect. [Windows PC, PlayStation 3, XBox 360]. Califórnia, Estados Unidos da América: Eletronic Arts.

Bloom, H. (1994). *The Western Canon*. ISBN: 0151957479. Harcourt Brace & Company.

BoardToBitsGames (2017). Better jumping in unity with four lines of code. https://www.youtube.com/watch?v=7KiK0Aqtmzc. Última vez acedida a 21/12/2018.

Butcher, S. (1902). *The Poetics of Aristotle*. The McMillan Company.

Caillois, R. (2001). *Man, Play, and Games*. University of Illinois Press, Ilinóis, Estados Unidos da America. Obra original publicada em 1958.

Coleridge, S. T. (2009). *Biographia Literaria*. The Floating Press, Auckland, Nova Zelândia. Obra original publicada em 1817.

ConcernedApe (2016). Stardew valley. [Windows PC, Mac OS, Linux, PlayStation 4, XBox One, Nintendo Switch, PlayStation Vita, iOS]. Londres, Inglaterra: Chucklefish.

Crystal Dynamics (2013). Tomb raider. [Windows PC, Mac OS, Linux, PlayStation 3, XBox 360, XBox One, PlayStation 4, Nvidia Shield]. Tóquio, Japão: Square Enix.

Csikszentmihalyi, M. (2014). *Flow and the Foundations of Positive Psychology: The Collected Works of Mihaly Csikszentmihalyi*. Springer Science+Business Media, Dordrecht, Holanda.

Dontnod Entertainment (2015). Life is strange. [Android, iOS, Linux, Windows PC, Mac OS, PlayStation 3, PlayStation 4, XBox 360, XBox One]. Tóquio, Japão: Square Enix.

duniagames.co.id (2018). Inilah 7 pasangan dalam game yang

bisa bikin kamu baper! https://duniagames.co.id/news/9644-inilah-7-pasangan-dalam-game-yang-bisa-bikin-kamu-baper?page=2. Última vez acedida a 18/10/2018.

engadget.com (2016). Paranormal thriller 'oxenfree' hits ps4 in may. https://www.engadget.com/2016/04/27/oxenfree-ps4-new-game-plus-interview/. Última vez acedida a 01/01/2019.

facts.zone (2018). The bushes and clouds in super mario bros are the same, just colored differently. https://facts.zone/archives/4000. Última vez acedida a 15/10/2018.

Field, S. (2005). *Screenplay: The Foundation of Screenwriting*. ISBN: 0385339038. Random House.

Frasca, G. (1999). Ludology meets narratology: Similitude and diferences between (video)games and narrative. *Parnasso*, 3.

gamespy.com (2008). Littlebigplanet walkthrough & strategy guide. http://ps3.gamespy.com/playstation-3/media-molecule-ps3-game-/guide/page_3.html. Última vez acedida a 05/01/2019.

Gygax, G. and Arneson, D. (1974). Dungeons & Dragons. Washington, Estados Unidos da América: Wizards of the Coast.

Hitbox Team (2012). Dustforce. [Windows PC, Mac OS, Linux, PlayStation 3, PlayStation Vita, XBox 360]. Texas, Estados Unidos da América: Digerati Distribution.

hitboxteam.com (2013). Designing game narrative. https://hitboxteam.com/designing-game-narrative. Última vez acedida a 09/09/2018.

Hocking, C. (2007). Ludonarrative dissonance in bioshock. http://www.clicknothing.typepad.com/click_nothing/2007/10/ludonarrative-d.html, Última vez acedido em 07/10/2018.

Huizinga, J. (1949). *Homo Ludens: A Study of the Play-element in Culture*. ISBN: 0710005784. Routledge & Kegan Paul Ltd.

Infinite Fall (2017). Night in the woods. [Windows PC, Mac OS, Linux, XBox One, PlayStation 4, Nintendo Switch, iOS, Android]. Michigan, Estados Unidos da América: Finji.

Ivory Tower (2014). The crew. [Windows PC, PlayStation 4, XBox 360, XBox One]. Montreuil, França: Ubisoft.

jogosalvo.com.br (2017). 10 melhores momentos de the last of us. https://jogosalvo.com.br/10-melhores-momentos-de-the-last-of-us/. Última vez acedida a 26/11/2018.

joystickdivision.com (2011). Five things we learned from pac-man. https://web.archive.org/web/20170810113530/http://www.joystickdivision.com/2011/01/five_things_we_learned_from_pa.php. Última vez acedida a 08/09/2018.

Juul, J. (1999). A clash between game and narrative. Master's thesis, Universidade de Copenhaga.

Kokonis, M. (2014). Intermediality between games and fiction: The "ludology vs. narratology" debate in computer game studies: A response to gonzalo frasca. *Acta Universitatis Sapientiae, Film and Media Studies*, 9:171–188.

Konami (1998). Metal gear solid. [PlayStation]. Tóquio, Japão: Konami.

Kortekaas, K. (2018). 2018 global games market report. Technical report, Newzoo.

Laurel, B. (1986). *Toward the Design of a Computer-based Interactive Fantasy System*. PhD thesis, Ohio State University.

Laurel, B. (1993). *Computers as Theatre*. ISBN: 9780321918628. Addison-Wesley.

League of Geeks (2015). Armello. [Windows PC, Mac OS, Linux, PlayStation 4, Box One, iOS, Nintendo Switch, Android]. Victória, Austrália: League of Geeks.

Lee, T. (2013). Designing game narrative. http://www.hitboxteam.com/designing-game-narrative, Última vez acedido em 07/10/2018.

Lionhead Studios (2004). Fable. [XBox, Windows PC, Mac OS, XBox 360]. Washington, Estados Unidos da América: Microsoft Studios.

Makedonski, B. (2012). Ludonarrative dissonance: The roadblock to realism. https://www.destructoid.com/ludonarrative-dissonance-the-roadblock-to-realism-235197.phtml, Última vez acedido em 08/10/2018.

Mar, R. A., Oatley, K., and Peterson, J. B. (2009). Exploring the link between reading fiction and empathy: Ruling out individual diferences and examining outcomes. *Communications*, 34(4):pp. 407–428.

Media Molecule (2008). Little big planet. [PlayStation 3]. Califórnia, Estados Unidos da América:

Sony Interactive Entertainment.

mobygames.com (2017). Bertie the brain promo art. https://www.mobygames.com/game/arcade/bertie-the-brain/promo/promoImageId,204889/. Última vez acedida a 15/10/2018.

Namco (1980). Pac-man. [Arcade]. Tóquio, Japão: Nintendo.

Naughty Dog (2007). Uncharted: Drake's fortune. [PlayStation 3]. Califórnia, Estados Unidos da América: Sony Interactive Entertainment.

Naughty Dog (2013). The last of us. [PlayStation 3]. Califórnia, Estados Unidos da América: Sony Interactive Entertainment.

Night School Studio (2016). Oxenfree. [Windows PC, Mac OS, XBox One, PlayStation 4, Nintendo Switch, Linux, iOS, Android]. Califórnia, Estados Unidos da América: Night School Studio.

Nintendo (1981). Donkey kong. [Arcade]. Quioto, Japão: Nintendo.

Nintendo (1985). Super Mario Bros. [Nintendo Entertainment System]. Tóquio, Japão: Nintendo.

Nintendo (1986). The legend of zelda. [Famicom Disk System, Nintendo Entertainment System]. Tóquio, Japão: Nintendo.

Nyamyam (2014). Tengami. [iOS, Wii U, Windows PC, Mac OS]. Londres, Inglaterra: Nyamyam.

Oatley, K. (1994). A taxonomy of the emotions of literary response and a theory of identification in fictional narrative. *Poetics*, 23(1-2):53–74.

Parlett, D. (1999). *The Oxford History of Board Games*. 0192129988. Oxford University Press.

Pittman, J. K. (2016). Building a better jump. In *Game Developers Conference*.

Playdead (2017). Inside. [XBox One, Windows PC, PlayStation 4, Nintendo Switch, iOS]. Copenhaga, Dinamarca: Playdead.

pt.slideshare.net (2012). Como usar o monitoramento de mídias sociais em campanhas políticas. https://pt.slideshare.net/scup/workshop-scup-como-usar-o-monitoramento-de-mdias-sociais-em-campanhas-polticas. Última vez acedida a 21/11/2018.

Pynenburg, T. (2012). Games worth a thousand words: Critical approaches and ludonarrative harmony in interactive narratives. Master's thesis, University of New Hampshire.

Rockstar North (1997-2013). Grand theft auto. [Android, Dreamcast, iOS, Mac OS, Windows PC, MS-DOS, Nintendo DS, PlayStation, PlayStation 2, PlayStation 3, PlayStation 4, Windows Phone, XBox, XBox 360, XBox One]. Nova Iorque, Estados Unidos da América: Rockstar Games.

Rockstar Studios (2012). Max payne 3. [PlayStaion 3, XBox 360, Windows PC, Mac OS]. Nova Iorque, Estados Unidos da América: Rockstar Games.

Ryan, M.-L. (2001). Beyond myth and metaphor - the case of narrative in digital media. *Computer Games & Digital Textualities*, 1.

Salen, K. and Zimmerman, E. (2003). *Rules of Play: Game Design Fundamentals*. ISBN: 9780262240451. The MIT Press.

Schneidereit, J. (2015). Tengami: The art of a folding world. In *Game Developers Conference*. `https://www.youtube.com/watch?v=DvEvtD6KJ7k`.

Square Enix (2016). Final fantasy xv. [XBox One, Windows PC, PlayStation 4]. Tóquio, Japão: Square Enix.

Swink, S. (2009). *Game Feel: A Game Designer's Guide to Virtual Sensation*. ISBN: 9780123743282. Morgan Kaufmann Publishers.

Sylvester, T. (2013). *Designing Games: A Guide to Engineering Experiences*. O'Reilly Media, Sebastopol, Califórnia.

Taito (1978). Space invaders. [Arcade]. Tóquio, Japão: Taito.

techsob.com (2013). Why realism in video games is sought after. `http://techsob.com/why-realism-in-video-games-is-sought-after/`. Última vez acedida a 15/09/2018.

thatgamecompany (2012). Journey. [PlayStation 3, PlayStation 4, Windows PC]. Califórnia, Estados Unidos da América: Sony Interactive Entertainment.

tvtropes.org (2015). Three act structure. `https://tvtropes.org/pmwiki/pmwiki.php/Main/ThreeActStructure`. Última vez acedida a 20/10/2018.

Ubisoft Montreal (2003). Prince of persia: The sands of time. [PlayStation 2, GameCube, XBox,

Windows PC, PlayStation 3]. Montreuil, França: Ubisoft.

Vatavu, R.-D., Anthony, L., and Wobbrock, J. O. (2012). Gestures as point clouds: A $p recognizer for user interface prototypes. In *ICMI '12*, pages 273–280, Santa Monica, California.

videogamehistory.wikia.com (2012). Tennis for two. `http://videogamehistory.wikia.com/wiki/File:Tennis_for_two.jpg`. Última vez acedida a 15/10/2018.

Wardrip-Fruin, N. (2004). *First Person: New Media as Story, Performance, and Game*. ISBN: 0262232324. The MIT Press.

Wobbrock, J. (2012). $p point-cloud recognizer. `https://depts.washington.edu/madlab/proj/dollar/pdollar.html`, Última vez acedido em 31/12/2018.

Zacks, J. M. (2015). *Flicker: Your Brain on Movies*. Oxford University Press, 198 Avenida Madison, Nova Iorque, NY 10016.

Zoink Games (2013). Stick it to the man. [Wii U, Windows PC, Mac OS, XBox One, iOS, PlayStation Vita, PlayStation 3, PlayStation 4, Nintendo Switch]. Liverpool, Inglaterra: Ripstone.

Apêndice A

Código desenvolvido

```
public class BarrierController : MonoBehaviour {

    [SerializeField]
    private BoxCollider boxCollider;

    [YarnCommand("disable")]
    public void BarrierState(string info) {
        bool state = (info == "true");
        boxCollider.enabled = state;
    }
}
```

Figura A.1: Script encarregue de desativar a barreira que impede o lobo de atravessar a ponte quando o lobo oferece mel ao guarda.

```
public class CameraController : MonoBehaviour {

    [SerializeField]
    private Transform target;
    [SerializeField][Range(0f,1f)]
    private float DefaultSmoothSpeed;
    [SerializeField]
    private Vector3 defaultOffset;
    [HideInInspector]
    public Vector3 offset;
    [HideInInspector]
    public float smoothSpeed;
    private Vector3 velocity = Vector3.zero;
    private bool isLockedOnTarget;

    private void Start() {
        DefaultCameraPosition(0.125f);
    }

    private void FixedUpdate() {
        Vector3 desiredPosition;
        if (isLockedOnTarget == true) {
            desiredPosition = target.position + offset;
        } else {
            desiredPosition = offset;
        }
        Vector3 smoothedPosition = Vector3.SmoothDamp
            (transform.position, desiredPosition, ref velocity, smoothSpeed);
        transform.position = smoothedPosition;
    }
```

Figura A.2: Parte 1 do script que controla a orientação da câmara em todos os *frames*.

```
    public void ChangeCameraPosition
        (Vector3 newOffset, float newSmoothSpeed, bool isLockingOnTarget = true) {
        isLockedOnTarget = isLockingOnTarget;
        offset = newOffset;
        smoothSpeed = newSmoothSpeed;
    }

    public void DefaultCameraPosition(float newSmoothSpeed) {
        offset = defaultOffset;
        smoothSpeed = newSmoothSpeed;
        isLockedOnTarget = true;
    }
}
```

Figura A.3: Parte 2 do script que controla a orientação da câmara em todos os *frames*.

```
public class DisableCrow : MonoBehaviour
{
    [SerializeField]
    GameObject crow;

    private void OnTriggerEnter(Collider other) {
        if (other.tag=="Player") {
            crow.SetActive(false);
        }
    }
}
```

Figura A.4: Script que torna o corvo visível após o evento da fuga dos sete cabritinhos. Antes disso, o corvo encontra-se oculto.

```
public class DialogueController : MonoBehaviour {

    [SerializeField]
    Text actionPrompt;
    [SerializeField]
    DialogueRunner dialogueRunner;
    [SerializeField]
    DialogueUI dialogueUI;

    void Start () {
        actionPrompt.enabled = false;
    }

    private void OnTriggerStay(Collider other) {
        if(other.transform.tag == "NPC") {
            if (other.GetComponent<NPC>() && dialogueRunner.isDialogueRunning == false) {
                actionPrompt.text = "Talk";
                actionPrompt.enabled = true;
            }
            if (Input.GetMouseButtonDown(0) && dialogueRunner.isDialogueRunning == false) {
                actionPrompt.enabled = false;
                dialogueRunner.StartDialogue(other.GetComponent<NPC>().talkToNode);
            }
        } else if (other.transform.tag == "GoatsHouse") {
            if (other.GetComponent<NPC>() && dialogueRunner.isDialogueRunning == false) {
                actionPrompt.text = "Knock";
                actionPrompt.enabled = true;
            }
            if (Input.GetMouseButtonDown(0) && dialogueRunner.isDialogueRunning == false) {
                actionPrompt.enabled = false;
                dialogueRunner.StartDialogue(other.GetComponent<NPC>().talkToNode);
            }
        }
    }
```

Figura A.5: Parte 1 do script que verifica a proximidade do jogador a um NPC e que faz a ligação ao sistema de diálogo.

```
    private void OnTriggerExit(Collider other) {
        if (other.GetComponent<NPC>() && dialogueRunner.isDialogueRunning == true) {
            dialogueRunner.Stop();
            dialogueUI.DialogueInterrupted();
        } else if (other.GetComponent<NPC>()) {
            actionPrompt.enabled = false;
        }
    }
}
```

Figura A.6: Parte 1 do script que verifica a proximidade do jogador a um NPC e que faz a ligação ao sistema de diálogo.

```
public class DrawingController : MonoBehaviour {

    [SerializeField]
    private Transform gestureOnScreenPrefab;
    private List<Gesture> gestureSet = new List<Gesture>();
    private List<Point> points = new List<Point>();
    private Vector3 cursorPosition = Vector3.zero;
    private List<Transform> shapesOnScreen;
    public Transform player;
    public Camera LineRendererCamera;
    private Vector3 prevCursorPosition;
    private LineRenderer lineRenderer;
    private Material lineMat;
    private Transform prefab;
    public Transform squarePrefab;
    public Transform lTrianglePrefab;
    public Transform rTrianglePrefab;
    public Transform linePrefab;
    private string message;
    private bool isDrawn;
    private string newGestureName = "";
    private Result gestureResult;
    bool canDraw = false;

    void Start () {
        shapesOnScreen = new List<Transform>();
        prevCursorPosition = new Vector3(-999, -999, -999);
        TextAsset[] gesturesXml = Resources.LoadAll<TextAsset>("GestureSet/");
        foreach (TextAsset gestureXml in gesturesXml) {
            gestureSet.Add(GestureIO.ReadGestureFromXML(gestureXml.text));
        }
    }
```

Figura A.7: Parte 1 do script que permite desenhar no espaço de jogo, que envia os dados para o sistema de reconhecimento de desenhos e que trata os dados recebidos desse mesmo sistema para imediatamente substituir por uma figura com dimensões correspondentes.

```
void Update () {
    if (canDraw) {
        if (Input.GetMouseButtonDown(1)) {
            Transform gesture = Instantiate(gestureOnScreenPrefab, transform.position,
                transform.rotation);
            lineRenderer = gesture.GetComponent<LineRenderer>();
        }

        if (Input.GetMouseButton(1)) {
            cursorPosition = new Vector3(Input.mousePosition.x, Input.mousePosition.y,
                player.position.z - LineRendererCamera.transform.position.z);
            if (cursorPosition != prevCursorPosition) {
                points.Add(new Point(cursorPosition.x, cursorPosition.y, 0));
                prevCursorPosition = cursorPosition;
                lineRenderer.positionCount = points.Count;
                lineRenderer.SetPosition(points.Count - 1,
                    LineRendererCamera.ScreenToWorldPoint(
                        new Vector3(cursorPosition.x, cursorPosition.y, cursorPosition.z)));
            }
        }

        if (Input.GetMouseButtonUp(1)) {
            isDrawn = true;
            if (points.Count > 1) {
                Gesture candidate = new Gesture(points.ToArray());
                gestureResult = PointCloudRecognizer.Classify(candidate,
                    gestureSet.ToArray());

                message = gestureResult.GestureClass + " " + gestureResult.Score;

                if (gestureResult.Score >= 0.85) {
                    if (gestureResult.GestureClass == "Square") {
                        prefab = squarePrefab;
                    } else if (gestureResult.GestureClass == "lTriangle") {
                        prefab = lTrianglePrefab;
                    } else if (gestureResult.GestureClass == "rTriangle") {
                        prefab = rTrianglePrefab;
                    } else if (gestureResult.GestureClass == "Line") {
                        prefab = linePrefab;
                    } else {
                        return;
                    }
```

Figura A.8: Parte 2 do script que permite desenhar no espaço de jogo, que envia os dados para o sistema de reconhecimento de desenhos e que trata os dados recebidos desse mesmo sistema para imediatamente substituir por uma figura com dimensões correspondentes.

```
                    Transform pCopy = Instantiate(prefab,
                        new Vector3(lineRenderer.bounds.center.x,
                        lineRenderer.bounds.center.y, player.position.z),
                        Quaternion.identity);
                    pCopy.transform.localScale *= Mathf.Clamp(Mathf.Max(
                        lineRenderer.bounds.extents.x, lineRenderer.bounds.extents.y),
                        float.MinValue, 3.0f);
                    if (gestureResult.GestureClass == "Line") {
                        pCopy.transform.localScale = new Vector3(
                            pCopy.transform.localScale.x, 0.25f, 1);
                    } else {
                        pCopy.transform.localScale = new Vector3(
                            pCopy.transform.localScale.x, pCopy.transform.localScale.y, 1);
                    }
                    if (shapesOnScreen.Count >= 1) {
                        Destroy(shapesOnScreen[0].gameObject);
                        shapesOnScreen.RemoveAt(0);
                    }
                    shapesOnScreen.Add(pCopy);
                }
            }

            isDrawn = false;
            points.Clear();
            lineRenderer.positionCount = 0;
            Destroy(lineRenderer.gameObject);
        }
    }
}

[YarnCommand("CanDraw")]
public void CanDraw() {
    canDraw = true;
}
}
```

Figura A.9: Parte 3 do script que permite desenhar no espaço de jogo, que envia os dados para o sistema de reconhecimento de desenhos e que trata os dados recebidos desse mesmo sistema para imediatamente substituir por uma figura com dimensões correspondentes.

```
public class FinishTrigger : MonoBehaviour
{
    public CameraController cameraController;
    [SerializeField]
    Image image;
    [SerializeField]
    Text text;
    [SerializeField]
    GameObject panel;

    private void Start() {
        image.color = new Color(1, 1, 1, 0);
        image.color = new Color(0, 0, 0, 0);
        panel.SetActive(false);
    }

    private void OnTriggerEnter(Collider other) {
        if (other.tag == "Player") {
            cameraController.ChangeCameraPosition(
                cameraController.gameObject.transform.position, 0, false);
            panel.SetActive(true);
            StartCoroutine(FadeFinish());
        }
    }

    IEnumerator FadeFinish() {
        for (float i = 0; i < 1; i += Time.deltaTime) {
            image.color = new Color(1, 1, 1, i);
            text.color = new Color(0, 0, 0, i);
            yield return null;
        }
    }
}
```

Figura A.10: Script que ativa quando o jogador salta da torre do corvo, sinalizando o final da demonstração do videojogo através de um desvanecimento para branco, com uma frase de agradecimento de participação.

```
public class FinalEventTrigger : MonoBehaviour {

    [SerializeField]
    GameObject goats, crow, villager1, villager2, villager3, villager4,
        finalBarrier1, finalBarrier2;
    [SerializeField]
    SpriteRenderer sprite2, sprite4;
    BoxCollider boxCollider;
    bool run = false;
    private Vector3 velocity = Vector3.zero;

    private void Start() {
        //crow.SetActive(false);
        boxCollider = GetComponent<BoxCollider>();
    }

    [YarnCommand("FinalEvent")]
    public void FinalEvent() {
        boxCollider.enabled = false;
        crow.SetActive(true);
        villager1.transform.position = new Vector3(115.5f, -2, 1.25f);
        villager2.transform.position = new Vector3(115, -2, 0.75f);
        sprite2.flipX = false;
        villager3.transform.position = new Vector3(115, 2.1f, 4.75f);
        villager4.transform.position = new Vector3(114.5f, 2.1f, 5.3f);
        sprite4.flipX = false;
        finalBarrier1.SetActive(true);
        finalBarrier2.SetActive(true);
        run = true;
    }

    private void Update() {
        if (run) {
            goats.transform.position = Vector3.SmoothDamp(goats.transform.position,
                new Vector3(190, 0.5f, 1.174f), ref velocity, 0.95f);
        }
    }
}
```

Figura A.11: Script que inicia o evento final onde os cordeiros fojem de casa, os aldeões se deslocam de modo a bloquear a passagem do jogador para partes anteriores do nível, forçando-o a ir de encontro ao corvo, que também se torna visível a partir desse instante.

```
public class CatchFlour : MonoBehaviour {

    [SerializeField]
    Text actionPrompt;
    [SerializeField]
    Image image;
    [SerializeField]
    InventoryController inventoryController;

    public ExampleVariableStorage variableStorage;

    private void Start() {
        image.color = new Color(1, 1, 1, 0);
    }

    private void OnTriggerStay(Collider other) {
        if(other.tag == "Player" && Input.GetMouseButtonDown(0)) {
            variableStorage.SetValue("$hasFlour", new Yarn.Value(true));
            gameObject.SetActive(false);
            inventoryController.CatchFlour();
        }
    }

    private void OnTriggerExit(Collider other) {
        if (other.tag == "Player") {
            actionPrompt.enabled = false;
        }
    }
}
```

Figura A.12: Script que permite a recolha da farinha e a remoção do objeto representante da farinha dentro da cena de jogo.

```
public class CatchHoney : MonoBehaviour {

    [SerializeField]
    Text actionPrompt;
    [SerializeField]
    Image image;
    [SerializeField]
    CharacterController cc;
    [SerializeField]
    InventoryController inventoryController;
    public ExampleVariableStorage variableStorage;

    private void Start() {
        image.color = new Color(1, 1, 1, 0);
    }

    private void OnTriggerStay(Collider other) {
        if(other.tag == "Player") {
            actionPrompt.text = "Steal";
            actionPrompt.enabled = true;
            if (Input.GetMouseButtonDown(0)) {
                variableStorage.SetValue("$hasHoney", new Yarn.Value(true));
                gameObject.SetActive(false);
                actionPrompt.text = "";
                inventoryController.CatchHoney();
            }
        }
    }

    private void OnTriggerExit(Collider other) {
        if (other.tag == "Player") {
            actionPrompt.enabled = false;
        }
    }
}
```

Figura A.13: Script que permite a recolha do mel e a remoção do objeto representante do mel dentro da cena de jogo.

```
public class InteractionController : MonoBehaviour {

    public float pushForce = 2.0f;
    private RaycastHit rHit;

    [HideInInspector] public bool isGrabbing;
    [HideInInspector] public Ray ray;
    private Transform objectGrabbed;

    private void Start() {
        ray = new Ray(transform.position, transform.right);
    }

    private void Update() {
        ray = new Ray(transform.position, ray.direction);
        Debug.DrawRay(ray.origin, ray.direction, Color.magenta);

        if(Physics.Raycast(ray, out rHit, 1f)) {
            if (Input.GetKeyDown(KeyCode.E) && rHit.transform.tag == "Grabbable") {
                isGrabbing = !isGrabbing;
            }

            if (isGrabbing && rHit.transform.tag == "Grabbable") {
                objectGrabbed = rHit.transform;
                rHit.transform.parent = transform;
            } else if (!isGrabbing && rHit.transform.tag == "Grabbable") {
                rHit.transform.parent = null;
            }
        } else if(isGrabbing) {
            objectGrabbed.parent = null;
            isGrabbing = false;
        }
    }
}
```

Figura A.14: Script que permite ao jogador agarrar e arrastar desenhos que faça, e impede a viragem do sprite da personagem principal quando esta está a puxar algo na sua direção.

```
public class InventoryController : MonoBehaviour
{
    bool isFadedIn = false, isFadedOut = false;
    [SerializeField]
    Image honeyImage, flourImage, item1Image, item2Image;
    [SerializeField]
    CharacterController characterController;

    private void Update() {

        if (item1Image.sprite != null && characterController.velocity !=
            new Vector3(0,0,0) && item1Image.color.a > 0.65f) {
            StopCoroutine("FadeInItem1");
            StartCoroutine("FadeOutItem1");
        } else if (item1Image.sprite != null && characterController.velocity ==
            new Vector3(0, 0, 0) && item1Image.color.a < 0.85f) {
            StopCoroutine("FadeOutItem1");
            StartCoroutine("FadeInItem1");
        }

        if (item2Image.sprite != null && characterController.velocity !=
            new Vector3(0, 0, 0) && item2Image.color.a > 0.65f) {
            StopCoroutine("FadeInItem2");
            StartCoroutine("FadeOutItem2");
        } else if (item2Image.sprite != null && characterController.velocity ==
            new Vector3(0, 0, 0) && item2Image.color.a < 0.85f) {
            StopCoroutine("FadeOutItem2");
            StartCoroutine("FadeInItem2");
        }
    }

    public void CatchHoney() {
        StartCoroutine(CatchHoneyCoroutine());
    }

    public void CatchFlour() {
        StartCoroutine(CatchFlourCoroutine());
    }
```

Figura A.15: Parte 1 do script que está encarregue de tornar visível a interface que informa o jogador dos objetos que tem em posse, e que torna esses ícones translúcidos quando a personagem jogável se encontra em movimento.

```
IEnumerator CatchHoneyCoroutine() {
    if (item1Image.sprite == null) {
        item1Image.sprite = honeyImage.sprite;
        for (float i = 0; i < 1; i += Time.deltaTime) {
            item1Image.color = new Color(1, 1, 1, i);
            yield return null;
        }
    } else {
        item2Image.sprite = honeyImage.sprite;
        for (float i = 0; i < 1; i += Time.deltaTime) {
            item2Image.color = new Color(1, 1, 1, i);
            yield return null;
        }
    }
}

IEnumerator CatchFlourCoroutine() {
    if (item1Image.sprite == null) {
        item1Image.sprite = flourImage.sprite;
        for (float i = 0; i < 1; i += Time.deltaTime) {
            item1Image.color = new Color(1, 1, 1, i);
            yield return null;
        }
    } else {
        item2Image.sprite = flourImage.sprite;
        for (float i = 0; i < 1; i += Time.deltaTime) {
            item2Image.color = new Color(1, 1, 1, i);
            yield return null;
        }
    }
}

IEnumerator FadeOutItem1() {
    for (float i = item1Image.color.a; i > 0.3f; i -= Time.deltaTime) {
        item1Image.color = new Color(1, 1, 1, i);
        yield return null;
    }
}
```

Figura A.16: Parte 2 do script que está encarregue de tornar visível a interface que informa o jogador dos objetos que tem em posse, e que torna esses ícones translúcidos quando a personagem jogável se encontra em movimento.

```
    IEnumerator FadeInItem1() {
        yield return new WaitForSeconds(1);
        for (float i = item1Image.color.a; i < 1; i += Time.deltaTime) {
            item1Image.color = new Color(1, 1, 1, i);
            yield return null;
        }
    }

    IEnumerator FadeOutItem2() {
        for (float i = item2Image.color.a; i > 0.3f; i -= Time.deltaTime) {
            item2Image.color = new Color(1, 1, 1, i);
            yield return null;
        }
    }

    IEnumerator FadeInItem2() {
        yield return new WaitForSeconds(1);
        for (float i = item2Image.color.a; i < 1; i += Time.deltaTime) {
            item2Image.color = new Color(1, 1, 1, i);
            yield return null;
        }
    }
}
```

Figura A.17: Parte 3 do script que está encarregue de tornar visível a interface que informa o jogador dos objetos que tem em posse, e que torna esses ícones translúcidos quando a personagem jogável se encontra em movimento.

```
public class MarketCameraTrigger : MonoBehaviour {

    public CameraController cameraController;

    private void OnTriggerEnter(Collider other) {
        if (other.tag == "Player") {
            cameraController.ChangeCameraPosition(new Vector3(102, 5, -24), 0.25f, false);
        }
    }

    private void OnTriggerExit(Collider other) {
        if (other.tag == "Player") {
            cameraController.DefaultCameraPosition(0.25f);
        }
    }
}
```

Figura A.18: Script que permite à câmara mudar de posição e desligar-se do jogador quando este entra a área correspondente ao mercado, e que faz a cªamara voltar ao seu posicionamento predefinido quando o jogador sai dessa área.

```
public class OneWayPlatformer : MonoBehaviour {

    public Collider parentCollider;
    public CharacterController playerCharacterController;

    private void Start() {

        if (parentCollider == null) {
            Debug.LogError(transform.parent.name + " is missing the parent collider");
        }

        if (playerCharacterController == null) {
            Debug.LogError(transform.parent.name + " is missing the player collider");
        }

        BoxCollider box = GetComponent<BoxCollider>();
        box.size = new Vector3(transform.localScale.x+0.5f, 25, transform.localScale.z);
        box.transform.Translate(0,-0.5f,0);
    }

    private void OnTriggerStay(Collider other) {
        if (other.transform.tag == "Player") {
            Physics.IgnoreCollision(playerCharacterController, parentCollider);
        }

    }

    private void OnTriggerExit(Collider other) {
        if (other.transform.tag == "Player") {
            Physics.IgnoreCollision(playerCharacterController, parentCollider, false);
        }
    }
}
```

Figura A.19: Script Que cria uma área de trigger que desativa a colisão duma plataforma correspondente, permitindo ao jogador que a trespasse de baixo para cima, mas que a torna sólida quando o jogador já se encontra acima dela, resultando numa plataforma trespassável. Permite também que o jogador passe de cima para baixo quando este carrega em simultâneo no botão correspondente de baixo e de salto.

```
public class PlayerCanvasController : MonoBehaviour {

    public Transform targetCamera;
    public Transform player;
    private Vector3 velocity = Vector3.zero;
    private Vector3 velocityBubbles = Vector3.zero;
    [SerializeField]
    private float smoothSpeed;
    [SerializeField]
    private Transform[] speechBubble;
    [SerializeField] private float speed = 1;
    [SerializeField] private float bubbleHeight = 1;
    private float[] timeCounter = {0, 0, 0};

    private void Start() {
        for (int i = 0; i < speechBubble.Length; i++) {
            timeCounter[i] = Random.Range(0f, 7f);
        }
    }

    void Update() {
        transform.LookAt(2 * transform.position - targetCamera.position);
        Vector3 smoothedPosition = Vector3.SmoothDamp(
            transform.position, player.position, ref velocity, smoothSpeed);
        transform.position = smoothedPosition;
        for (int i = 0; i < speechBubble.Length; i++) {
            timeCounter[i] += speed * Time.deltaTime;
            float y = Mathf.Sin(timeCounter[i]) * bubbleHeight;
            speechBubble[i].localPosition += new Vector3(0, y, 0);
        }
    }
}
```

Figura A.20: Script gestor da interface ligada ao personagem jogável e que contém os balões de fala do jogador. Também torna possível o efeito de flutuação que cada balão de fala tem individualmente.

```
public class PlayerController : MonoBehaviour {

    CharacterController characterController;
    InteractionController interactionController;
    SpriteRenderer spriteRenderer;
    Animator animator;
    private Vector3 movement;
    private float horizontalMovement;
    private float verticalMovement;
    private float depthMovement;
    private float directionFacing = 1;
    public bool canMoveZ = false;
    private bool isChangingLanes = false;
    private Vector3 targetLane;
    public float runSpeed = 7f;
    public float turnSpeed = 0.15f;
    private Ray Zplus;
    private Ray Zminus;
    private RaycastHit ZplusHit;
    private RaycastHit ZminusHit;
    private Vector3 velocity = Vector3.zero;
    public float jumpForce = 2.275f;
    public float gravity = -0.1f;
    private string platformTag;
    public float slidingSpeed = 6;
    public float MinSlideAngle = 45;
    private float standingAngle;
    private Ray ray;
    private RaycastHit rHit;
    private Vector3 rCross;
    public CameraController cameraController;

    private void Start() {
        characterController = GetComponent<CharacterController>();
        interactionController = GetComponent<InteractionController>();
        spriteRenderer = GetComponentInChildren<SpriteRenderer>();
        animator = GetComponentInChildren<Animator>();
        targetLane = transform.position;
    }
```

Figura A.21: Parte 1 do script encarregue do movimento da personagem em todos os eixos, incluindo o afecto da gravidade, o deslize, a anulação de força de ascensão vertical quando o jogador bate em algum teto, e a deteção de uma plataforma trespassável nos pés da personagem. Também está encarregue de toda a animação que o sprite possui, estando ligado ao animator do Unity.

```
private void Update() {
    Run();

    Jump();

    SlideDown();
}

private void FixedUpdate() {

    Gravity();
    characterController.Move(new Vector3(horizontalMovement * Time.deltaTime,
        verticalMovement, depthMovement * Time.deltaTime));
}

void Run() {
    horizontalMovement = Input.GetAxis("Horizontal") * runSpeed;

    if (Input.GetAxisRaw("Horizontal") == 0) {
        horizontalMovement = horizontalMovement / 2;
        animator.SetBool("isRunning", false);
    }

    if (interactionController.isGrabbing) {
        horizontalMovement = horizontalMovement * 0.50f;
    }

    if((Input.GetAxisRaw("Horizontal") != 0 || Input.GetAxisRaw("Vertical") != 0) &&
        !interactionController.isGrabbing){
        directionFacing = Input.GetAxisRaw("Horizontal");
        animator.SetBool("isRunning", true);
    }

    if (directionFacing == 1f) {
        interactionController.ray.direction = interactionController.transform.right;
        spriteRenderer.flipX = false;
    } else if (directionFacing == -1f) {
        interactionController.ray.direction = -interactionController.transform.right;
        spriteRenderer.flipX = true;
    }
```

Figura A.22: Parte 2 do script encarregue do movimento da personagem em todos os eixos, incluindo o afecto da gravidade, o deslize, a anulação de força de ascensão vertical quando o jogador bate em algum teto, e a deteção de uma plataforma trespassável nos pés da personagem. Também está encarregue de toda a animação que o sprite possui, estando ligado ao animator do Unity.

```
        Zplus = new Ray(transform.position + new Vector3(0, -1, 0.2f), Vector3.forward);
        Zminus = new Ray(transform.position + new Vector3(0, -1, -0.2f), Vector3.back);

        if (Physics.Raycast(Zplus, out ZplusHit, 2)) {
            if (ZplusHit.transform.tag == "ZArea" && Input.GetAxisRaw("Vertical") == 1 &&
                !isChangingLanes) {
                isChangingLanes = true;
                targetLane = ZplusHit.point;
            }
        }
        if (Physics.Raycast(Zminus, out ZminusHit, 2)) {
            if (ZminusHit.transform.tag == "ZArea" && Input.GetAxisRaw("Vertical") == -1 &&
                !isChangingLanes) {
                isChangingLanes = true;
                targetLane = ZminusHit.point;
            }
        }

        if (Mathf.Abs(transform.position.z - targetLane.z) > 0.1f) {
            transform.position = Vector3.SmoothDamp(transform.position, new Vector3(
                transform.position.x, transform.position.y, targetLane.z), ref velocity, 0.2f);
        } else {
            isChangingLanes = false;
        }
    }


    void Jump() {
        animator.SetFloat("yVelocity", verticalMovement);
        if (Input.GetButtonDown("Jump") && characterController.isGrounded &&
            !(Input.GetAxisRaw("Vertical") == -1 && platformTag == "1WayPlatform") &&
            !interactionController.isGrabbing) {
            animator.SetTrigger("hasJumped");
            verticalMovement = jumpForce;
        }

        if (Input.GetButtonUp("Jump") && characterController.velocity.y > 0) {
            verticalMovement = verticalMovement / 2;
        }
    }
```

Figura A.23: Parte 3 do script encarregue do movimento da personagem em todos os eixos, incluindo o afecto da gravidade, o deslize, a anulação de força de ascensão vertical quando o jogador bate em algum teto, e a deteção de uma plataforma trespassável nos pés da personagem. Também está encarregue de toda a animação que o sprite possui, estando ligado ao animator do Unity.

```
    void Gravity() {
        verticalMovement += gravity;

        if (characterController.velocity.y < 0 && !characterController.isGrounded) {
            verticalMovement = verticalMovement * 1.05f;
        }

        if (characterController.isGrounded)
        {
            verticalMovement = Mathf.Clamp(verticalMovement, gravity * 10, Mathf.Infinity);
            animator.SetBool("isGrounded", true);
        } else {
            verticalMovement = Mathf.Clamp(verticalMovement, Physics.gravity.y, Mathf.Infinity);
            animator.SetBool("isGrounded", false);
        }
    }


    void SlideDown() {
        if ((standingAngle < -MinSlideAngle || standingAngle > MinSlideAngle)) {
            horizontalMovement = (-Mathf.Abs(-(0.02f * Mathf.Abs(
                standingAngle + MinSlideAngle) - 0.9f) + 0.9f) + 0.9f) * slidingSpeed;
        }
    }

    private void OnControllerColliderHit(ControllerColliderHit hit) {

        if (characterController.isGrounded) {
            platformTag = hit.transform.tag;
            ray = new Ray(hit.point + Vector3.up, Vector3.down);
            Physics.Raycast(ray, out rHit);
            Debug.DrawRay(hit.point, ray.direction, Color.red);
            rCross = Vector3.Cross(Vector3.up, rHit.normal);
            if (rCross.z < 0) {
                standingAngle = Vector3.Angle(rHit.normal, Vector3.up);
            } else {
                standingAngle = -(Vector3.Angle(rHit.normal, Vector3.up));
            }
        }
```

Figura A.24: Parte 4 do script encarregue do movimento da personagem em todos os eixos, incluindo o afecto da gravidade, o deslize, a anulação de força de ascensão vertical quando o jogador bate em algum teto, e a deteção de uma plataforma trespassável nos pés da personagem. Também está encarregue de toda a animação que o sprite possui, estando ligado ao animator do Unity.

```
        if((characterController.collisionFlags & CollisionFlags.Above) != 0 &&
            verticalMovement > 0) {
            verticalMovement = -0.02f;
        }

        if (Input.GetAxisRaw("Vertical")==-1 && Input.GetButton("Jump") &&
            characterController.isGrounded && hit.transform.tag == "1WayPlatform") {
            Physics.IgnoreCollision(characterController, hit.collider);
        }
    }
}
```

Figura A.25: Parte 5 do script encarregue do movimento da personagem em todos os eixos, incluindo o afecto da gravidade, o deslize, a anulação de força de ascensão vertical quando o jogador bate em algum teto, e a deteção de uma plataforma trespassável nos pés da personagem. Também está encarregue de toda a animação que o sprite possui, estando ligado ao animator do Unity.

```
public class RotationController : MonoBehaviour {

    GameObject player;
    public float speed = 3;
    public float triggeDistance = 12;

    private Vector3 liedDown;
    private Vector3 StandingUp;

    private void Start() {
        player = GameObject.FindGameObjectWithTag("Player");
        liedDown = new Vector3(85, transform.eulerAngles.y, transform.eulerAngles.z);
        StandingUp = new Vector3(0, transform.eulerAngles.y, transform.eulerAngles.z);
        transform.position = new Vector3(transform.position.x, transform.position.y + 0.001f,
            transform.position.z);
    }

    void Update () {
        if (Mathf.Abs(transform.position.x - player.transform.position.x) -
            (Mathf.Abs(transform.position.z - player.transform.position.z)*0.4) <
            (triggeDistance) ) {
            Vector3 destAngle = Vector3.Slerp(transform.eulerAngles, StandingUp,
                speed * Time.deltaTime);
            transform.eulerAngles = destAngle;
        } else {
            Vector3 destAngle = Vector3.Slerp(transform.eulerAngles, liedDown,
                speed * Time.deltaTime);
            transform.eulerAngles = destAngle;
        }
    }

}
```

Figura A.26: Script encarregue de levantar e baixar adereços do cenário conforma a proximidade do jogador, adicionalmente sendo possível controlar a velocidade de ascensão e descensão.

```
public class Rotate : MonoBehaviour {

    public float speed;
    Renderer center;

    private void Start() {
        center = GetComponentInChildren<Renderer>();
    }

    void Update () {
        transform.RotateAround(center.bounds.center, new Vector3(0, 0, 1), speed);
    }
}
```

Figura A.27: Script que faz as pás do moínho andarem à roda.

```
public class TextureScroll : MonoBehaviour {

    public float scrollX = 0.5f;
    public float scrollY = 0.5f;

    // Update is called once per frame
    void Update () {
        float offsetX = Time.time * scrollX;
        float offsetY = Time.time * scrollY;
        GetComponent<Renderer>().material.mainTextureOffset = new Vector2(offsetX, offsetY);
    }
}
```

Figura A.28: Script que faz a testura da água mover-se numa determinada direção, criando a ilusão de um rio no ambiente de jogo.

Apêndice B

CD com materiais extra

B.1 Conteúdo do CD

O CD que vai em conjunto com esta dissertação conta com:

- Ficheiro PDF com cópia digital da dissertação;
- Executável da demonstração do projeto;
- Asset open-source base responsável pelo sitema de diálogo;
- Asset open-source base responsável pelo reconhecimento de desenhos.