Faculdade de Educação Tecnológica do Estado do Rio de Janeiro

Graduação Tecnológica em Análise e Desenvolvimento de Sistemas

Vinícius Guerra Cardoso

TWOFOLD LAND: INTRODUZINDO COMPUTAÇÃO ATRAVÉS DE UM JOGO DIGITAL

Monografia

Rio de Janeiro
10 de Março de 2016

Vinícius Guerra Cardoso

TWOFOLD LAND: INTRODUZINDO COMPUTAÇÃO ATRAVÉS DE UM JOGO DIGITAL

Monografia apresentada ao curso de Graduação Tecnológica em Análise e Desenvolvimento de Sistemas da Faculdade de Educação Tecnológica do Estado do Rio de Janeiro como parte dos requisitos necessários à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Ricardo Portella de Aguiar

Rio de Janeiro
10 de Março de 2016

À comunidade dos jogos digitais, para que critiquem o que consomem tanto nos jogos quanto em outras mídias e acreditem que podem contribuir para o mundo com sua paixão.

Às mulheres e sua luta por respeito em sua representação nos jogos e outras mídias.

À minha família e minha companheira, Isabela. Ao nosso futuro, que construimos a cada momento.

Agradecimentos

Aos meus pais e minha companheira, que me apoiam na perseguição de meus sonhos, exposição de meus ideais e prática da minha moral, possibilitando-me o desenvolvimento desse projeto.

A todos os desenvolvedores dos jogos que preencheram minha vida com as experiências que me ajudaram tanto a fugir desse mundo quanto criticá-lo, me tornando integrante dele.

“Where we are today? We built it. This story - this ”legend“ - it’s ours. We can change the world - and with it, the future.”

(Big Boss)

Resumo

Twofold Land é o título do jogo digital desenvolvido no projeto em questão. Em vista da presente evasão na educação superior de Tecnologia da Informação, o objetivo principal do jogo é ensinar efetivamente conceitos básicos da computação. Inicialmente, o estudo apresenta as teorias que embasam o desenvolvimento do jogo: o Construtivismo de Jean Piaget e outros jogos educacionais. para validar o aspecto educacional, e o gênero de jogos Role Playing Game. Em apêndice, pode-se encontrar o projeto lógico contendo a documentação técnica de software. Em seguida, o universo do jogo, sua protagonista e as atividades desenvolvidas no jogo são descritas considerando as teorias apresentadas anteriormente. Enfim, colocam-se as escolhas de implementação e conclui-se sobre o desenvolvimento.

Abstract

Twofold Land is the title of the digital game developed in this project. Facing the current levels of withdrawal concerning Information Technology graduation courses, the main objective of the game is to effectively teach basic concepts used in computer science. Initially, the study presents the theories that underlie the game's development: Jean Piaget's Constructivism and other educational games, validating this game's educational aspect, then the Role Playing Game genre. The project's software documentation can be found within the appendix. Following, the game's universe, its protagonist and the activities she engage in are described considering the aforementioned theories. Finally, implementation choices are revealed along with conclusions about the project's development.

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
GURPS	Generic Universal Role Playing System
HP	Health Points
HUD	Heads Up Display
IDE	Integrated Development Environment
INEP	Instituto Nacional de Estudos e Pesquisas
PC	Personal Computer
RPG	Role Playing Game
UI	User Interface

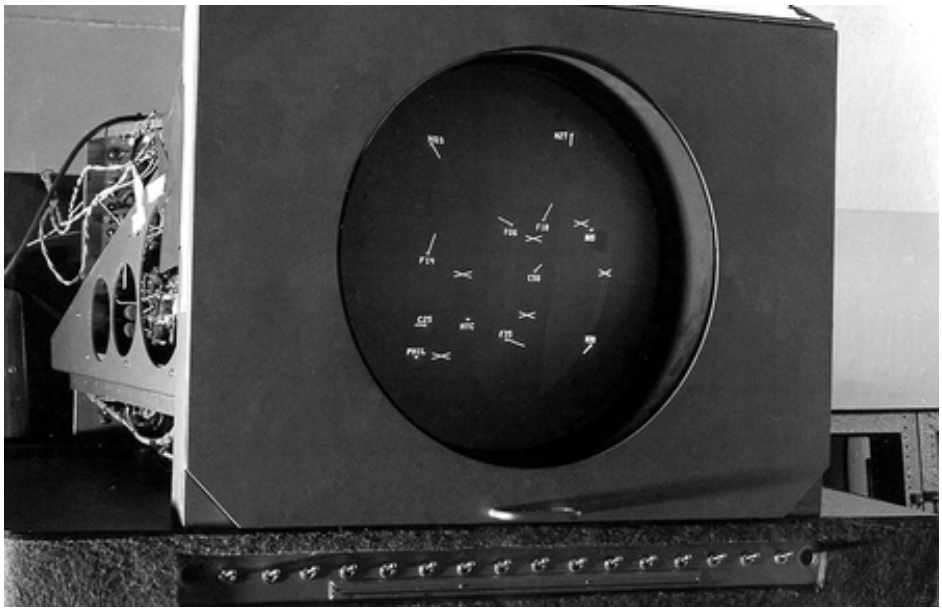
Sumário

1 Introdução

1.1 Contexto

O primeiro dispositivo considerado um videogame, de acordo com (COHEN, 2015), foi o *Cathode Ray Tube Amusement Device* — Dispositivo de Divertimento de Raios Catódicos, em tradução livre. Ele foi construído em 1947 com dispositivos analógicos e consistia no uso de botões para modificar a trajetória de projéteis e acertar seus alvos, inspirado em radares de guerra utilizados na Segunda Guerra Mundial.

Figura 1 – *Cathode Ray Tube Amusement Device*



A definição de videogame muda constantemente. já que inovações tecnológicas revolucionam a forma como interagimos com eles. Muitos não consideram o *Cathode Ray Tube Amusement Device* um videogame por não ser um dispositivo digital e ser tão diferente dos videogames de hoje.

Quanto aos controles, já foram usados dispositivos diversos. Desde as alavancas e botões mecânicos do *Magnavox Odyssey*, primeiro console pessoal, ao joystick com tablet embutido do console *Nintendo WiiU.* Sensores e câmeras de captura de movimentos e outros dispositivos também foram utilizados, assim como os HMDs (*Head Mounted Displays* ou Visor Montado na Cabeça, em tradução livre) de realidade virtual *Oculus Rift* e *Playstation VR.*

Os gráficos também sofreram mudanças consideráveis. Inicialmente os jogos eram compostos por *sprites,* que são basicamente imagens estáticas que podem ser trocadas em rápida sucessão para criar animações. Hoje, milhões de polígonos em

modelos 3D complexos animados com captura de movimento de atores reais são exibidos em resoluções que se equiparam às do cinema.

Figura 2 – *Metal Gear* (1987)

http://filmesegames.com.br/2014/metal-gear-solid-1998-entendendo-um-classico/

Figura 3 – *Metal Gear Solid V: The Phantom Pain* (2015)

As imagens acima são de jogos da série *Metal Gear*[1], criada por Hideo Ko-

[1] Série Metal Gear. Gênero: ação. Desenvolvida por Kojima Productions e Konami. Publicada pela Konami. Primeiro título lançado: Metal Gear, 1987, plataforma MSX2. Último título: Metal Gear Solid V: The Phantom Pain, 2015, multiplas plataformas.

jima, que revolucionou os jogos não só nos gráficos e jogabilidade, mas no conteúdo, trazendo seções cinematográficas e temas adultos. Essa série também é importante na definição do gênero de ação, que é um entre muitos tipos de jogos digitais. A diversidade de gêneros de videogames também é crescente, ampliando cada vez mais seu público.

O consumo de jogos digitais atualmente constitui uma parcela substancial do mercado de entretenimento. 1.775.489.000 jogadores movimentaram cerca de 81,5 bilhões de dólares no ano de 2014. Desse total, a América Latina conteve participação de 3,3 bilhões de dólares, de acordo com uma pesquisa do grupo Newzoo (2014).

No Brasil, esse mercado também é expressivo. De acordo com uma pesquisa citada por Digital (2015), "Quase a metade dos entrevistados admitiu gastar até R$ 150, em média, por mês, com jogos eletrônicos. [...] Em casos de datas especiais [...] os entrevistados admitem gastar um pouco mais: até R$ 200 em um dia.". Complementar a esses dados, foi observado por Mercado (2015) que "Mesmo em um contexto econômico de crise [...] o crescimento mínimo no setor [...] no Brasil poderá ficar em torno dos 2% ao longo de 2015".

As pesquisas acima destacam a abrangência e rentabilidade dos jogos digitais, mesmo em cenário de crise financeira. Entretanto, a educação tecnológica — necessária na manutenção do mercado e em sua inovação e evolução — segue caminho contrário. A taxa de evasão em períodos iniciais de cursos de computação é elevada. No ano de 2013, foram registrados 156.774 alunos matriculados em cursos na área da ciência computação no Brasil e apenas 20.262 — aproximadamente 12% — eram concluintes, de acordo com (INEP, 2013). Existem diversos motivos para a evasão, como incapacidade de pagamento da mensalidade em faculdades particulares ou dificuldades em frequentar o campus por questões de transporte. Contudo, um motivo que aflige aqueles em plena condição de frequentar a instituição é a dificuldade de se inserirem no contexto da computação. Disciplinas como algoritmos e arquitetura de computadores exigem grande capacidade de abstração, familiaridade com conceitos da computação e proficiência matemática e lógica.

Fazendo uso dos benefícios trazidos pelos videogames, esse trabalho almeja agir nessa contextualização inicial do estudante de computação, combatendo o quadro atual de evasão, como descrito nos objetivos a seguir.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral é contribuir para a educação tecnológica desenvolvendo um jogo digital que ensine efetivamente conceitos utilizados na computação através de uma experiência lúdica.

De acordo com o artigo de Maluf[2], essa experiência se dá em atividades planejadas pelo educador em que o estudante "desenvolve habilidades de forma natural e agradável, que gera um forte interesse em aprender e garante o prazer".

1.2.2 Objetivos Específicos

Os objetivos específicos desse trabalho são:

- Apresentar mecânicas de jogo referentes a conceitos da computação
- Fazer com que o jogador identifique as soluções adequadas para os problemas apresentados
- Permitir certo grau de liberdade na solução dos problemas, desenvolvendo o julgamento do jogador
- Apresentar problemas em camadas, fazendo com que o jogador conquiste reavalie cenários conhecidos e os compreenda gradualmente

Com isso, espera-se familiarizar o jogador com o jargão da computação, conceitos iniciais e ferramentas comuns na solução de problemas cotidianos da computação.

1.3 Justificativa

Foi notado pelo autor, através de experiências pessoais, que a exposição a jogos digitais diversos é capaz de ensino efetivo. Inicialmente, foi adquirida certa fluência na leitura do idioma inglês. Então, fazendo uso do vocabulário aprendido e informações básicas, temas relevantes da história e das ciências foram desenvolvidos com estudos posteriores.

A partir da motivação pessoal e a análise social apresentada na introdução, foi decidido que os jogos digitais são uma mídia abrangente e particularmente relevante no

[2] Fonte: http://www.profala.com/arteducesp178.htm. Autora: Angela Cristina Munhoz Maluf. Artigo sem data. Acessado em 23 de Novembro de 2015.

desenvolvimento de um projeto que favoreça a educação tecnológica. Além do aspecto pessoal e social, o projeto também é justificado cientificamente.

De acordo com (AURELIANO; TEDESCO, 2012), experiências anteriores com programação e matemática facilitam o aprendizado de certos conteúdos. A partir disso, visa-se utilizar o jogo para promover essa facilidade. O que é possível, já que Melo e Silva (2011) descrevem, a partir de relatos do uso de jogos digitais em sala de aula:

> sucesso na utilização de jogos digitais e de objetos de aprendizagem e vislumbram suas potencialidades para usos educacionais, comprovando a importância destes recursos para a aprendizagem dos alunos

Além do sucesso educacional mencionado, benefícios como aumento da eficiência no processamento neural, citados por (GRANIC; LOBEL; ENGELS, 2014), justificam o desenvolvimento desse projeto apresentando seus benefícios cognitivos.

Finalmente, de acordo com (AURELIANO; TEDESCO, 2012), uma das maiores dificuldades de alunos iniciantes é o uso de conceitos da computação na prática para a solução de problemas. Isso justifica o uso de um jogo digital pois ele simula a aplicação prática do conhecimento adquirido, agindo como solução desde a exposição de conceitos até o uso deles na solução de problemas.

1.4 Estrutura do Trabalho

No capítulo 2, encontram-se teorias que comprovam o valor educacional do jogo e projetos que as aplicam, além de informações sobre o gênero *Role Playing Game* ou RPG.

No capítulo 3, o jogo será apresentado textualmente. A protagonista, controlada pelo jogador, será descrita, assim como sua relação com o mundo à sua volta. A partir disso, será feita a descrição das atividades que ela desenvolve e como o jogador a controla, o que se define como jogabilidade. Logo, a fase única e experimental presente no jogo será descrita, apresentando as atividades em sequência e as relacionando com os conteúdos educacionais em questão.

No capítulo 4, será exposta a modelagem de software de modo que as mecânicas do jogo possam ser entendidas tecnicamente.

No capítulo 5, as conclusões finais e propostas para continuação do desenvolvimento serão apresentadas.

2 Referenciais Teóricos

2.1 Construtivismo

O Inatismo, apresentado por Platão entre 427 e 347 a.C., foi uma das primeiras teorias relativas à cognição humana e ditava que o conhecimento é inato. Portanto, nem os dados externos, nem o formato de sua apresentação interferiam com o conhecimento de um indivíduo.

A antítese dessa teoria, posteriormente apresentada por Aristóteles entre 384 e 322 a.C., chama-se Empirismo. Nela, o conhecimento é disponibilizado pelo mundo exterior e absorvido pelos sentidos. Definia-se que a capacidade de aprender era congênita, eliminando a preocupação quanto à apresentação e didática.

O Construtivismo, proposto por Jean Piaget no século XX, sintetiza as teorias anteriores. Enquanto o Inatismo defende o conhecimento como inato e o Empirismo como externo, o Construtivismo apresenta o método de ensino como elo entre eles. O aprendizado, então, ocorre quando o indivíduo é estimulado a agir sobre o objeto de ensino. Assim, ele assimila os dados externos ao conhecimento prévio.

Para o entendimento dessa teoria, é importante a definição de algumas palavras-chave. De acordo com (LOUREIRO, 2009), e utilizando os grifos do autor para destaque das palavras-chave, a construção do conhecimento se dá através de um processo de **assimilação**, ou seja, inclusão de novos objetos a **esquemas** mentais — que são conjuntos de valores que facilitam a adaptação do sujeito ao ambiente, gerando comportamentos. Com a **acomodação** desse novo conhecimento de acordo com a realidade, ocorre a **equilibração** desses esquemas, causando a compreensão.

Dada essa definição de construção de conhecimento, é importante que o educador estruture suas ferramentas didáticas objetivando o “encaminhamento das etapas que desencadeiam e efetivam a construção do conhecimento” (NIEMANN; BRANDOLI, 2012, p.12).

Como citado na justificativa e verificado por Melo e Silva (2011), jogos digitais são uma ferramenta efetiva no desenvolvimento dessas etapas.

A jogabilidade de Twofold Land insere o jogador em um fluxo de aprendizado desenvolvido a partir da teoria em questão. A protagonista entra em contato com novas habilidades no mundo externo e elas são incluídas ao seu conjunto de habilidades disponíveis. Essa assimilação de novas habilidades é referente à inclusão de novos objetos nos esquemas mentais. A acomodação, entretanto, é desenvolvida pelo jogador,

já que novas habilidades trazem novo significado a objetos observados anteriormente, levando a uma revisão das informações adquiridas. A equilibração, então, é observada no decorrer do jogo, com o jogador adquirindo novas habilidades e dando novo significado a objetos então desconhecidos. O progresso do jogador é diretamente relacionado à compreensão das habilidades que ele adquire.

A observação de jogos educacionais diversos também ajudou a definir o jogo e é desenvolvida a seguir.

2.2 Jogos Educacionais

O site Code.org[1] contém uma série de jogos, sendo a maioria para públicos infantis. Existem também ferramentas de fácil entendimento para a criação de jogos com recursos pré-definidos. A maioria desses aplicativos foca no ensino rápido de conceitos simples e na utilização de algoritmos em linguagens de programação como *Javascript*.

Destacado na página mencionada acima, o jogo digital *CodeCombat*, é um dos mais complexos. Tem como plataforma o navegador e foi criado para a faixa etária a partir 9 anos. Seu *gameplay* consiste no controle de personagens em turnos através do uso de algoritmos, com fases sequenciais que focam em estruturas específicas da programação. São apresentadas várias linguagens de programação selecionáveis para uso durante o jogo, como Python, JavaScript e Lua.

Figura 4 – Tela do jogo de navegador CodeCombat

[1] Disponível em https://code.org/learn

Além de controlar o personagem durante o jogo, é possível customizá-lo com itens que mudam seus atributos no momento entre as fases. Esse é um elemento característico do gênero RPG, que será explorado a seguir.

2.3 RPGs

O entendimento do gênero é importante para que elementos da jogabilidade e do universo de Twofold Land sejam melhor reconhecidos. Assim, o jogo será familiar ao jogador, facilitando o aprendizado de suas mecânicas e, portanto, sua jogabilidade.

Role Playing Games ou RPGs são, em tradução livre, jogos de interpretação de papéis. Isso se dá pelo fato de que o jogador incorpora um personagem e tem poder sobre suas decisões e seu desenvolvimento, podendo moldá-lo até onde o jogo o permite. Inicialmente, esses jogos eram jogados em tabuleiro, sendo narrados por um mestre. Os jogadores, atuando como os personagens que criaram, acompanham a narrativa do mestre e tomam decisões que respeitam o livro de regras escolhido, como *Dungeons and Dragons* ou *Generic Universal Role Playing System*, conhecido como GURPS.

Figura 5 – Jogadores de RPG de mesa

http://www.dorkadia.com/2012/12/10/the-secret-sauce-part-ii/

Na era dos jogos digitais, os RPGs tomam diversas formas. Uma delas é a de JRPGs (*Japanese Role Playing Game* ou RPG japonês) como *Chrono Cross*[2]. Eles costumam contar com o estilo gráfico de animes ou animações japonesas, tramas dramáticas e bem desenvolvidas e sistemas de combate em turnos.

[2] Chrono Cross, lançado inicialmente em 1999 no Japão. Gênero: JRPG. Plataforma: Playstation. Desenvolvido e publicado por Squaresoft.

Figura 6 – Parte do cenário Arni Village, de Chrono Cross

MMORPGs (*Multiplayer Massive Online Role Playing Game* ou RPGs multijogadores massivos online) também são um formato comum, como em *World of Warcraft*[3], que já chegou a 12 milhões de jogadores de acordo com estudo publicado em (STATISTA, 2015).

Figura 7 – Parte do cenário Eversong Woods, de World of Warcraft

Outro gênero comum é o dos ARPGs (*Action Role Playing Games* ou RPGs de ação), que é uma categoria abrangente e contém tanto jogos com visão aérea e

[3] World of Warcraft, lançado em 2004. Gênero: MMORPG. Plataforma: PC. Desenvolvido e publicado por Blizzard Entertainment.

combate por habilidades, como *Diablo*[4], quanto jogos em terceira pessoa como a série *Dark Souls*[5] ou jogos em primeira pessoa como *Fallout 4*[6] ou *The Elder Scrolls V: Skyrim*[7].

Figura 8 – Visão do protagonista e a interface de usuário no jogo Diablo 3

Tanto em RPGs tradicionais, quanto em RPGs digitais, é comum o sistema de níveis de personagem. O jogador adquire pontos de experiência conforme progride pelo jogo. Esses pontos acumulam até que o jogador passe de nível. Ele então é recompensado com a possibilidade de investir em atributos e habilidades. Esse processo é cíclico e a quantidade de experiência necessária para passar de nível cresce, exigindo esforço sempre maior para evoluir.

Na imagem abaixo, pode-se observar a ficha de personagem do jogo digital *Dark Souls 2*.

[4] Série Diablo. Gênero: ARPG. Plataforma: PC. Desenvolvido e publicado por Blizzard Entertainment. Primeiro título: Diablo, 1996. Último título: Diablo III: Reaper of Souls, 2014.

[5] Série Dark Souls. Gênero: ARPG. Desenvolvida por FromSoftware. Publicada por Namco Bandai. Primeiro título lançado: Dark Souls, 2011, plataformas Playstation 3 e Xbox 360. Último título: Dark Souls 2: Scholar of The First Sin, 2015, plataformas Playstation 4, Xbox One e PC.

[6] Fallout 4, lançado em 2015. Gênero: ARPG. Plataforma: Xbox One, Playstation 4 e PC. Desenvolvido e publicado por Bethesda.

[7] The Elder Scrolls V: Skyrim, lançado inicialmente em 2011. Gênero: ARPG. Plataforma: Xbox 360, Playstation 3 e PC. Desenvolvido e publicado por Bethesda.

Figura 9 – Atributos de personagem em Dark Souls 2

Esse foco na caracterização do personagem leva a uma maior imersão do jogador na história, criando laços entre ele e o personagem que controla. Dessa forma, o jogador fica mais imerso no jogo e desenvolve sua espontaneidade de modo a preservar a vida de seu personagem e perseguir seus objetivos.

Isso facilita o desenvolvimento do fluxo de aprendizagem definido no Construtivismo. já que a construção do conhecimento depende da espontaneidade do sujeito, como descrito por (LOUREIRO, 2009), que agirá ativamente sobre seu personagem, assimilando constantemente os novos desafios e informações para que tenha sucesso.

Outro fator importante em RPGs é o contexto da estória, que ajuda a definir diversas características do jogo. O tipo de relacionamento que os personagens desenvolvem entre si, as dificuldades e os rivais encontrados, os itens e ambientes em questão, entre outros elementos, são encaixados no universo do jogo para que haja maior imersão possível. Normalmente, esse contexto é fantástico, apresentando magia ou elementos de ficção científica.

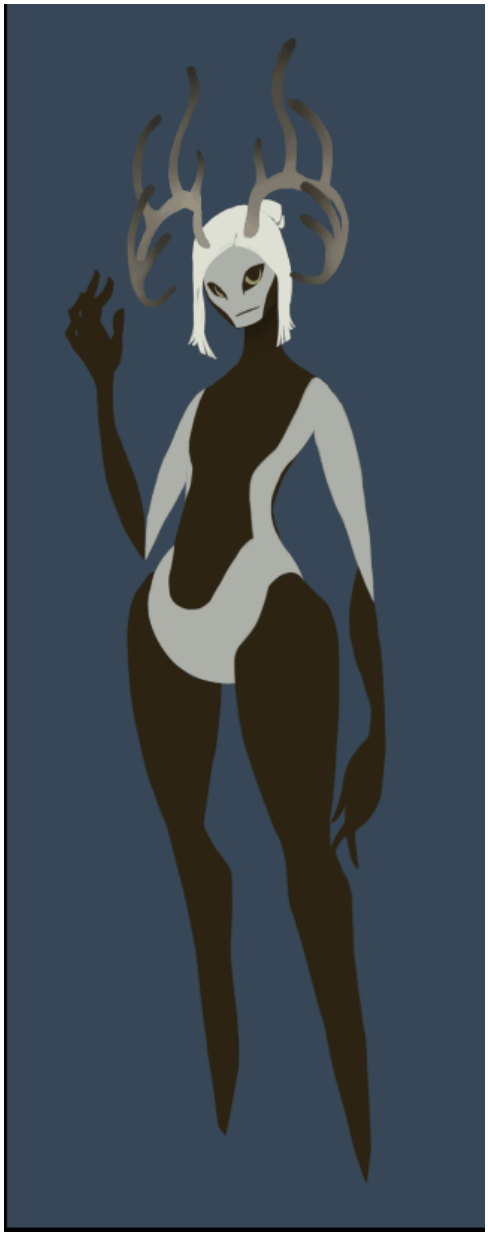
A partir do conhecimento dos conceitos educacionais Construtivistas e estilísticos de RPG, o jogo Twofold Land será apresentado a seguir, com foco na protagonista, sua jornada, sua relação com o mundo à sua volta e na jogabilidade.

3 Apresentação do Jogo

3.1 A Protagonista

Em Twofold Land, o jogador assume o papel da protagonista, chamada Ricci. Ela é inspirada no Fauno — que é uma entidade mitológica romana — e carrega algumas de suas características. As mais marcantes são sua galhada, análoga aos chifres do Fauno e seu papel como entidade protetora e representativa da natureza.

Figura 10 – Arte conceitual de Ricci, a protagonista



Elaborada pelo autor

O jogo embasa sua fantasia em preceitos da Computação, permitindo seu entendimento por analogias com conceitos conhecidos do universo dos RPGs. A forma principal de entender os conceitos que regem o mundo é pela observação da maneira como a protagonista interage com seu entorno.

A aquisição de habilidades por exemplo, remete a etapas da aprendizagem definidas pelo Construtivismo como descrito anteriormente. Ao entrar em contato com uma nova habilidade, que vem de um item coletado no cenário, ela é capaz de interagir com objetos conhecidos de formas diferentes.

Entre as interações permitidas pela assimilação de novas habilidades, pode-se destacar o destrancamento de baús. Eles possuem como chave uma sequência de dígitos binários que deve ser descoberta. Um número em base decimal é exibido e a conversão para binário deve ser feita.

Pela sua proximidade com a arquitetura computacional, Ricci entende como funciona a manipulação desses números, diferente dos outros personagens, fazendo com que os baús estejam intocados. Isso representa a discrepância entre linguagem de usuário final e linguagem de máquina e familiariza o jogador com ela.

Ainda que através de uma habilidade específica a protagonista seja capaz de compreender linguagem humana, ela não responde verbalmente, pois se comunica através dos comandos computacionais que envia diretamente para outras entidades. Sua personalidade é indefinida e é preenchida pelos aprendizados e experiências do jogador.

Com isso, foi arquitetada a maneira como o jogador desempenha as atividades de jogo, definido a seguir.

3.2 Jogabilidade

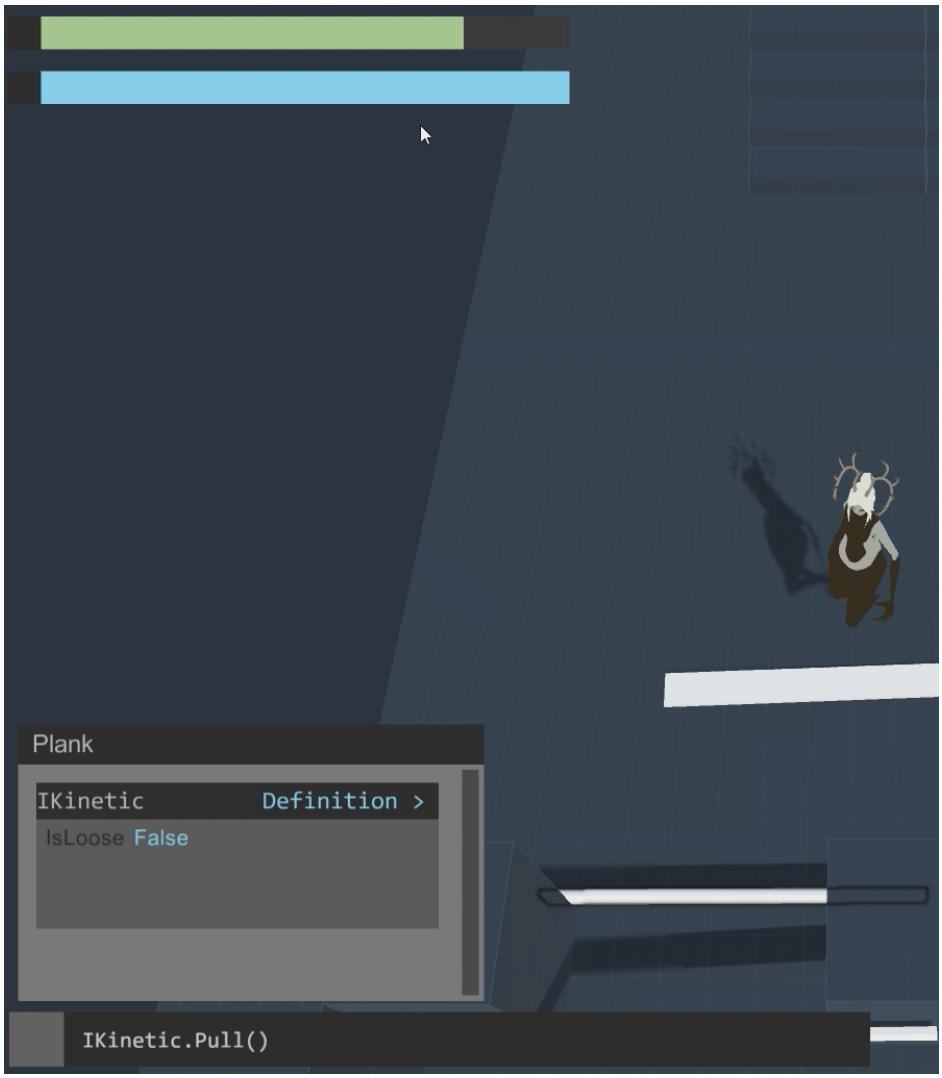
A visão do jogo é vertical, similar à da série Diablo, por exemplo. A movimentação da protagonista é realizada ao clicar no chão à sua volta.

Figura 11 – Ricci se movimentando em direção ao clique do jogador

Elaborada pelo autor

Na tela, encontram-se suas barras de *Health Points* (HP), que corresponde à sua energia vital disponível, e *Stamina*, que corresponde à energia necessária para interações.

Figura 12 – Barras de HP e Stamina na parte superiore Terminal e entidade selecionada na parte inferior

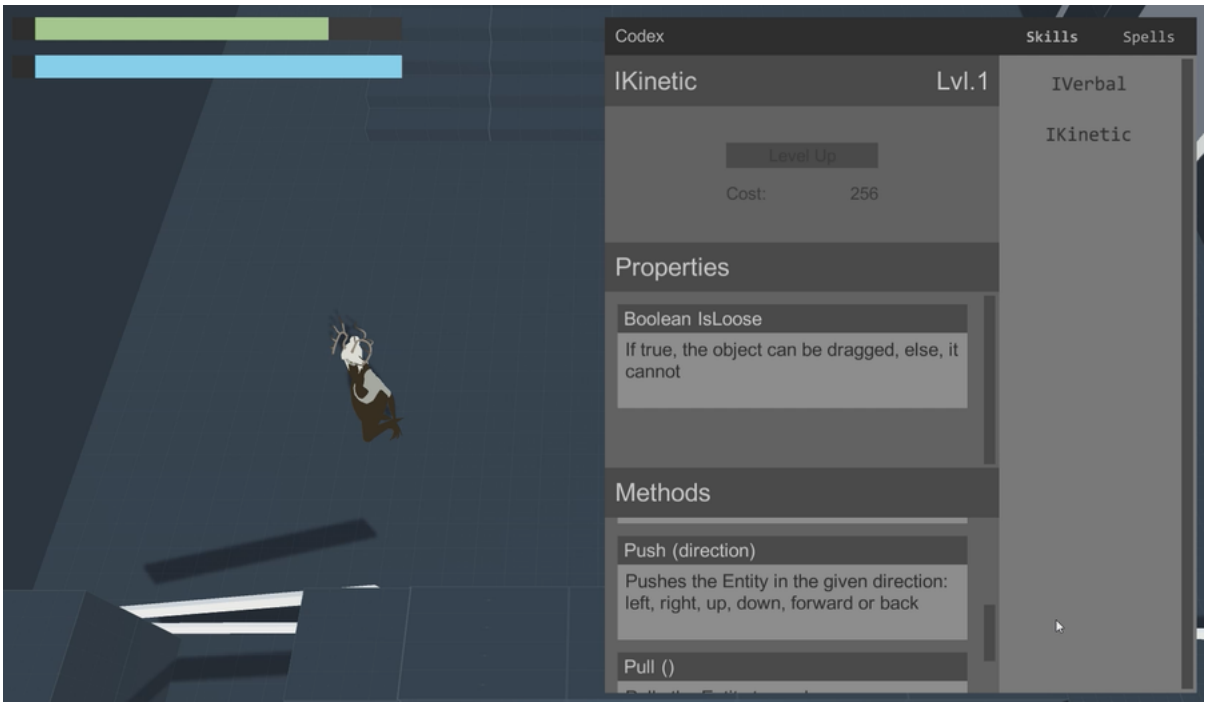


Existem dois fluxos principais de atividades no jogo, que serão descritos a seguir: o de interação com objetos do mundo — denominados entidades — e o de gerenciamento de habilidades, itens e algoritmos.

3.2.1 Interação com Entidades

Entidades têm suas características e comportamentos organizados em abstrações genéricas. Essas abstrações são interfaces — estruturas presentes em algumas linguagens de programação orientadas a objeto — e podem ser consultadas na seção Codex do jogo. As características das entidades são nomeadas propriedades e os comportamentos, métodos.

Figura 13 – Codex exibindo propriedades e métodos da habilidade IKinetic no canto direito



Os métodos constituem ações como abrir portas ou empurrar objetos, acessíveis através de comandos unitários enviados através do Terminal ou de algoritmos, que são sequências finitas de comandos pré-determinados, construídos na IDE.

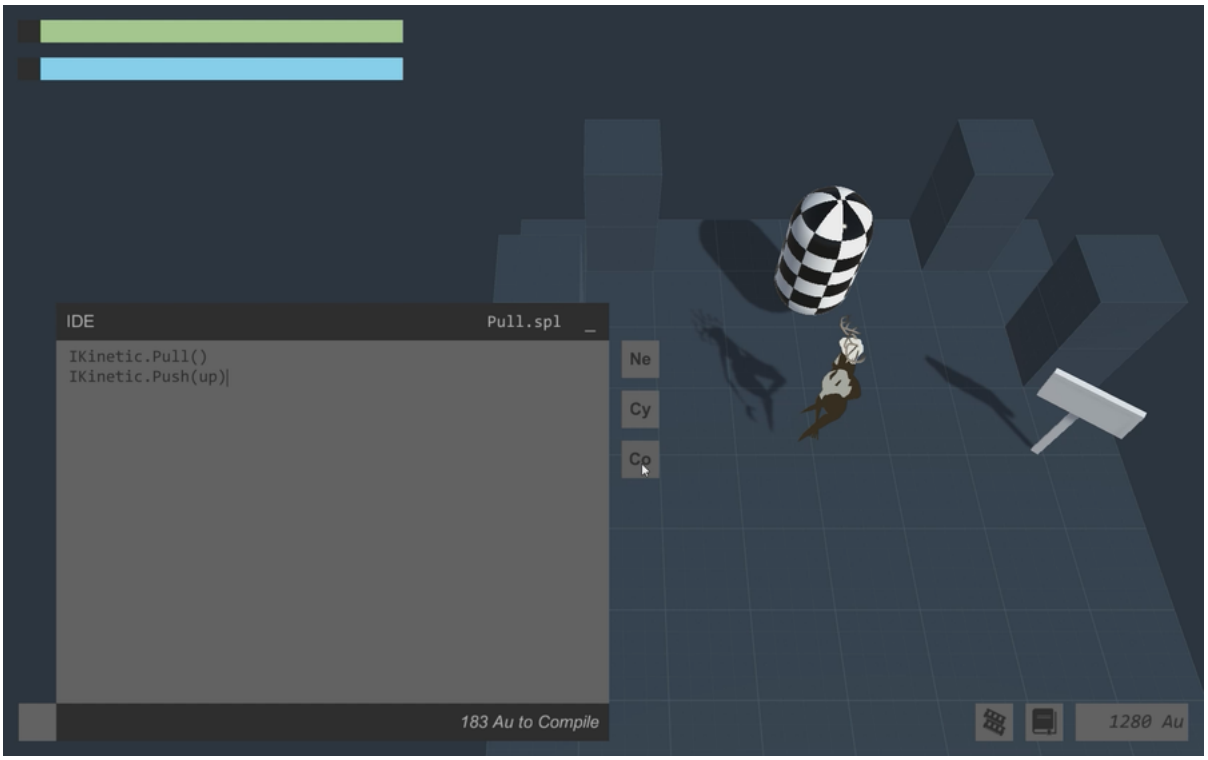
3.2.2 Gerenciamento de Personagem

O gerenciamento de personagem se dá basicamente no direcionamento do recurso adquirido no jogo, chamado Aura. Esse recurso pode ser aplicado em dois destinos.

Um deles é a evolução de *Skills*, ou habilidades, em português. Habilidades contém Interfaces respectivas e evoluções distintas e independentes, então cabe ao jogador decidir em quê investir. A evolução dessas habilidades deve ser motivada pelo tipo de interação preferida pelo jogador, já que causa mudanças específicas em cada entidade, tendo seu efeito percebido apenas através da experimentação.

Outro destino é a compilação de algoritmos. Esses algoritmos permitem a execução instantânea de sequências de comandos pré-definidos. Para criá-los, deve-se abrir a IDE, na qual o jogador pode escrever algoritmos chamando os métodos das interfaces conhecidas. Para que os algoritmos sejam utilizados, deve-se comprá-los usando a quantidade de Aura necessária, definida pelo custo de cada comando. Além disso, deve-se estar próximo do Compiler, estrutura estática que recupera as energias da protagonista e permite a compilação de algoritmos.

Figura 14 – Algoritmo descrito na IDE e pronto para ser compilado e usado como Spell



3.3 Cenário

O jogo apresentará um cenário experimental. Nele, cada atividade desenvolvida pelo jogador terá base em um conceito da computação, como apresentado anteriormente na seção de Jogabilidade. Esses conceitos incluem algoritmos, orientação a objeto, cálculo binário, arquitetura de computadores, entre outros.

O desenvolvimento de um cenário com percursos marcantes, boa caracterização e colocação de objetos é importante para que o jogador se localize. Além disso, um ambiente que seja natural auxilia na imersão, facilitando o foco do jogador.

O discorrer dessa fase terá dois momentos: no primeiro, o jogador será apresentado a problemas isolados que podem ser resolvidos através da experimentação das mecânicas de jogo. Dada a apresentação dessas mecânicas, chega o segundo momento, no qual jogador será exposto a problemas que exigem o uso desses conceitos de forma complementar.

Esse cenário representa o local onde Ricci se materializa no mundo humano. A trama contida no escopo desse projeto diz respeito à sua saída desse local e seu aprendizado sobre si mesma e sua relação com o mundo à sua volta. Abaixo estão as principais atividades para o desenvolvimento desse aprendizado.

3.3.1 Aquisição de Interfaces

Nesse momento, a protagonista nasce. Ela surge em um templo que foi construído com informações e desafios que a permitem aprender sobre suas habilidades e como ela funciona. No ambiente em que está, a única coisa que pode ser feita é a aquisição da Skill IVerbal. Com a aquisição dessa Skill, ela é capaz de ler um objeto que a explica sobre Skills e seu uso.

A seguir, é disponibilizada uma nova Skill, a IKinetic, e mais um objeto surge, falando sobre a leitura de propriedades e o uso de métodos e sugere que o Codex seja lido.

3.3.2 Leitura do Codex e uso de Métodos

O Codex apresenta conceitos de orientação a objeto. Entidades têm características que são apresentadas através de propriedades e comportamentos que podem ser desencadeados através do uso de métodos das Skills conhecidas.

A seguir, o caminho está bloqueado por um abismo e uma ponte está suspensa por duas tábuas. Ao ler sobre o IKinetic no Codex, o jogador pode perceber que se pode usar o método Pull para puxar essas tábuas e abrir caminho.

3.3.3 Criação de Spells

Ao cruzar a ponte, o jogador desce por um caminho até um ponto com um Compiler próximo. Nesse ponto, a câmera foca em um labirinto abaixo. Após ver o labirinto, o jogador continua seguindo, tirando pedras de seu caminho e colocando plataformas para que passe com o uso dos métodos de IKinetic.

Ao chegar no labirinto visto anteriormente, pode-se subir em uma plataforma e empurrá-la em uma direção para prosseguir, mas a plataforma se desfaz e a personagem volta se o comando não for dado rápido o suficiente e no momento correto. Aí é quando o jogador deve subir, analisar o labirinto e compilar um Spell que faça com que a plataforma siga a rota correta.

3.3.4 Alocação e Utilização de Spells

Ao criar o Spell, o jogador deve abrir a janela Storage e alocar o Spell criado no endereço de memória desejado ao clicar no espaço respectivo e no Spell criado. Para usar o Spell, deve-se digitar seu endereço de memória no Terminal, tendo a entidade desejada selecionada.

Os fatos acima e outros são explicados ao jogador em uma placa próxima do Compiler.

3.3.5 Evolução de Skills

Após cruzar o labirinto, o jogador adquire Aura, a Skill IUnlockable e se depara com uma pedra em seu caminho. Como a pedra é maior que as anteriores, uma placa sugere que ele evolua sua IKinetic, tendo mais força para movê-la e seguir em frente.

4 Implementação

Para o desenvolvimento de jogos, usam-se ambientes de desenvolvimento próprios chamados *engines*. Esses ambientes visam o uso de elementos gráficos e a interação com eles através de *scripts*, que são códigos criados pelo desenvolvedor a partir da API (*Application Programming Interface* ou interface de programação de aplicações) da *engine*.

A implementação do jogo foi feita na *engine* Unity 5.2.1f1 através de *scripts* orientados a objeto em C#. A IDE (*Integrated Development Environment* ou ambiente de desenvolvimento integrado) escolhida para edição dos *scripts* e *debug* é o Microsoft Visual Studio Enterprise 2015 com o *plugin Visual Studio Tools for Unity 2015.* Essas ferramentas apresentam auxílio na compleção e correção de código e teste em tempo real com avaliação de valores de variáveis.

Para versionamento, foi utilizado o programa *SourceTree,* que é um cliente de Git. O projeto está hospedado no GitHub[1].

A arte conceitual e as texturas para objetos 3D foram desenvolvidas no Adobe Photoshop Creative Cloud. Os objetos 3D foram modelados e animados no Autodesk 3ds Max 2015. Seus materiais foram atribuídos na Unity. Além do autor do trabalho, o jogo foi desenvolvido por outro integrante[2], que providenciou modelos 3D a partir da arte conceitual do autor.

O padrão de projeto utilizado foi o *Scrum*, padrão de projeto ágil que foca na entrega iterativa de pacotes de valor a curto prazo. Através dele, foi possível trabalhar em equipe de forma organizada a fazer entregas constantes, permitindo avaliação do progresso e da adequação do software a seus objetivos.

O jogo foi implementado em inglês e português brasileiro, o que pode ser mudado ao executar o jogo com o parâmetro -language com valores en para inglês ou pt-Br para português.

A resolução escolhida na execução do aplicativo é nativa ao monitor, verificada automaticamente pela *engine* e aplicada ao iniciar.

Embora sua mudança não seja automática como a mencionada acima, a localização é outra funcionalidade presente, permitindo maior acessibilidade ao jogo. Atualmente, a localização traz as linguagens inglês e português brasileiro, mas implementações futuras permitem a inclusão de quaisquer outras linguagens desejadas.

[1] Projeto hospedado em https://github.com/viniciusguerra/twofoldland

[2] Isabela Marcondes Tostes, cursa graduação tecnológica em Design 3D e Animação Digital na Universidade Veiga de Almeida. Contato através de isabelatostes@outlook.com.

Uma das características mais importantes da implementação foi o desenvolvimento com base na função Reflection da linguagem C#. Isso permite que o código fonte seja acessado diretamente durante a execução do jogo, tornando fidedignas as chamadas de método e visualização de propriedades. A sintaxe no jogo é a de C# sem o uso de ponto e vírgula para separar sentenças, tornando similar à linguagem Lua.

O código fonte ser acessado diretamente em uma arquitetura desacoplada permite que outras linguagens de programação sejam utilizadas e ensinadas em implementações futuras.

A seguir, serão apresentadas as considerações sobre o desenvolvimento do projeto e resultados adquiridos.

5 Considerações Finais

No decorrer do projeto, foi percebido que a metodologia de desenvolvimento foi adequada, embora o tempo tenha sido insuficiente para a criação de mais cenários de jogo. Grande parte do tempo de desenvolvimento foi dedicado à fundamentação teórica da parte educacional, fazendo com que o cenário de jogo fosse reduzido à apresentação da jogabilidade.

Considerando, então, a equipe de dois integrantes, como mencionado anteriormente, o período de desenvolvimento devia ser, preferencialmente, superior a um ano. O projeto se adequaria melhor aos 6 meses disponíveis se feito por uma equipe maior ou se o jogo em questão contemplasse menos mecânicas e uma arquitetura mais simples.

O objetivo principal foi alcançado, como verificado por testes com jogadores de diversos níveis de conhecimento. A experiência lúdica do jogo é capaz de familiarizar o jogador com os conceitos apresentados, já que têm sua utilidade prática no jogo. Espera-se que trabalhos similares sejam desenvolvidos em outros gêneros de jogos para aumentar a abrangência dessa técnica de ensino.

Trabalhos futuros consistirão na continuação do desenvolvimento do jogo, incluindo novos cenários, habilidades, mecânicas de jogo e possível implementação de outras linguagens de programação. Depois disso, a distribuição do jogo através da plataforma *Steam Greenlight.* Após o desenvolvimento e distribuição do jogo, devem ser feitos testes em sala de aula para avaliação da sua eficácia educacional, comparando grupos que usaram o jogo com grupos que não usaram. Resultados devem ser estudados em um artigo futuro.

Finalmente, o presente trabalho contribui para a literatura com um jogo digital que fundamenta sua jogabilidade no Construtivismo e ensina efetivamente uma série de ideias que facilitam o estudo posterior de Tecnologia da Informação.

Referências

AURELIANO, V. C. O.; TEDESCO, P. C. de A. R. Ensino-aprendizagem de Programação para Iniciantes: uma Revisão Sistemática da Literatura focada no SBIE e WIE. In: *23º Simpósio Brasileiro de Informática na Educação*. [S.l.: s.n.], 2012.

COHEN, D. *Cathode-Ray Tube Amusement Device – The First Electronic Game*. 2015. Disponível em: <http://classicgames.about.com/od/classicvideogames101/p/CathodeDevice.htm>.

DIGITAL, O. *Metade dos jogadores brasileiros gasta até R$ 150 por mês em games*. 2015. Disponível em: <http://olhardigital.uol.com.br/noticia/metade-dos-jogadores-brasileiros-gasta-ate-r-150-por-mes-em-games/51523>.

GRANIC, I.; LOBEL, A.; ENGELS, R. C. M. E. The Benefits Of Playing Video Games. *American Psychologist*, Washington, DC, v. 69, n. 1, p. 66 – 78, Janeiro 2014. Disponível em: <https://www.apa.org/pubs/journals/releases/amp-a0034857.pdf>.

INEP. *Sinopses Estatísticas da Educação Superior – Graduação 2013*. 2013. Disponível em: <http://portal.inep.gov.br/superior-censosuperior-sinopse>.

LOUREIRO, A. M. *A Teoria de Jean Piaget e a Realidade Escolar*. 2009. Disponível em: <http://www.ufrgs.br/psicoeduc/wiki/A_Teoria_de_Jean_Piaget_e_a_Realidade_Escolar>.

MELO, D. M. B. de; SILVA, K. C. da. JOGOS DIGITAIS E OBJETOS DE APREDIZAGEM NO ENSINO DA MATEMÁTICA. In: *III Encontro Regional de Educação Matemática*. [s.n.], 2011. Disponível em: <http://www.pucrs.br/famat/viali/tic_literatura/artigos/objetos/CC_Melo_e_Silva.pdf>.

MERCADO, C. *Mercado de games e atacado online apontam crescimento mesmo em um cenário de crise financeira*. 2015. Disponível em: <http://www.conexaomercado.com.br/wp/index.php/2015/06/mercado-de-games-e-atacado-online-apontam-crescimento-mesmo-em-um-cenario-de-crise-fina>.

NEWZOO. *Top 100 Countries Represent 99.8% of $81.5Bn Global Games Market*. 2014. Disponível em: <http://www.newzoo.com/insights/top-100-countries-represent-99-6-81-5bn-global-games-market/>.

NIEMANN, F. de A.; BRANDOLI, F. Jean Piaget: um aporte teórico para o construtivismo e suas contribuições para o processo de ensino e aprendizagem da Língua Portuguesa e da Matemática. In: UNIVERSIDADE DA CAXIAS DO SUL. *IX Seminário ANPED Sul*. 2012. p. 1 – 14. Disponível em: <http://www.ucs.br/etc/conferencias/index.php/anpedsul/9anpedsul/paper/viewFile/770/71>.

STATISTA. *Number of World of Warcraft subscribers from 1st quarter 2005 to 2nd quarter 2015 (in millions)*. 2015. Disponível em: <http://www.statista.com/statistics/276601/number-of-world-of-warcraft-subscribers-by-quarter/>.

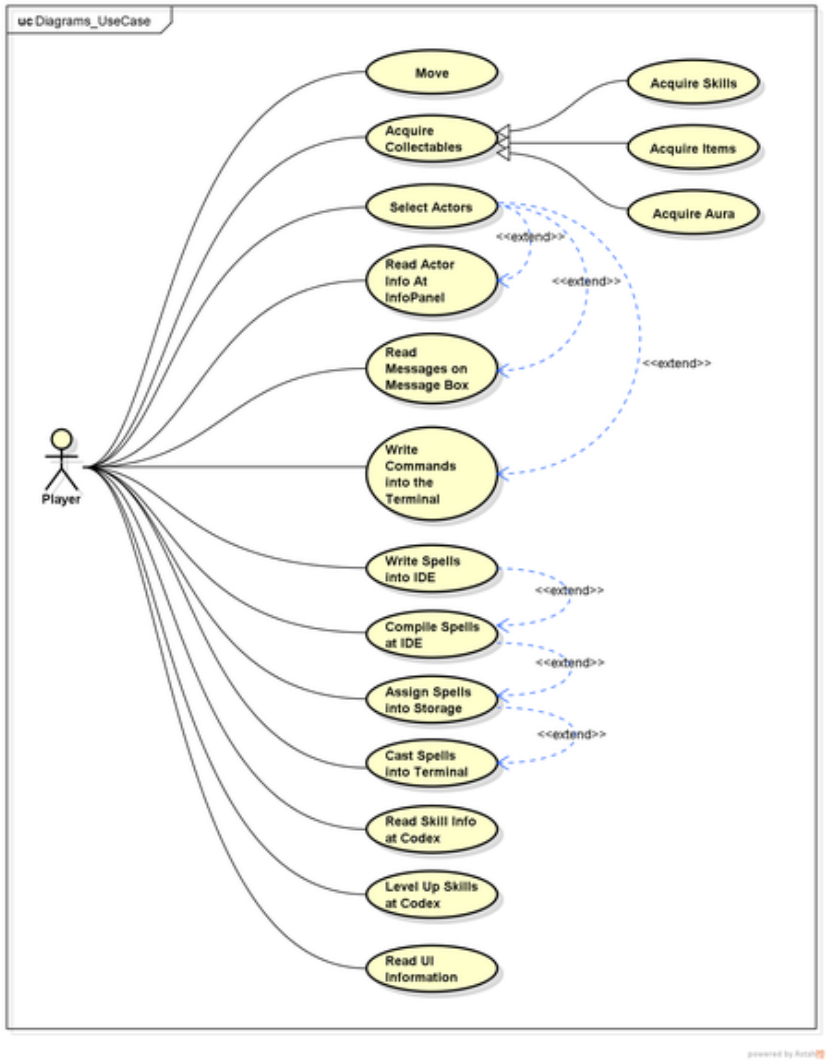
Apêndices

APÊNDICE A – Projeto Lógico

A.1 Casos de Uso

A.1.1 Diagrama de Casos de Uso

Figura 15 – Diagrama de Casos de Uso



Elaborado pelo autor

A.1.2 Especificação de Casos de Uso

Tabela 1 – UC01 - Select Actors

Descrição	Um Actor é selecionado para interação pelo jogador
Atores	Player
Pré-Condições	Que o Actor esteja dentro da distância máxima de seleção
Fluxo Principal	1. O jogador clica em um Actor 2. O Actor é selecionado
Fluxos Alternativos	Nenhum
Fluxos de Exceção	Nenhum
Pós-Condições	O Actor fica selecionado
Extensões	UC08, UC09, UC13

Elaborada pelo autor

Tabela 2 – UC02 - Move

Descrição	O jogador se desloca para o local selecionado
Atores	Player
Pré-Condições	Que a superfície clicada seja acessível e o caminho esteja liberado
Fluxo Principal	1. O jogador clica em uma superfície acessível 2. O jogador se deslocará automáticamente pelo caminho mais curto
Fluxos Alternativos	1. O jogador clica em uma superfície acessível 2. Por ter um obstáculo no caminho, o jogador se desloca até o obstáculo e para
Fluxos de Exceção	Nenhum
Pós-Condições	O jogador estará em uma posição diferente da inicial
Extensões	Nenhuma

Elaborada pelo autor

Tabela 3 – UC03- Acquire Collectables

Descrição	O jogador adquire recursos coletáveis
Atores	Player
Pré-Condições	Que o recurso esteja acessível para contato físico ou dentro da distância máxima de seleção
Fluxo Principal	1. O jogador usa UC02 para se locomover ao local do recurso 2. O jogador absorve o recurso em questão
Fluxos Alternativos	1. O jogador clica no recurso 2. O jogador absorve o recurso em questão
Fluxos de Exceção	Nenhum
Pós-Condições	O jogador terá um novo recurso
Extensões	UC14, UC15, UC16

Elaborada pelo autor

Tabela 4 – UC04 - Write Spells into IDE

Descrição	O jogador cria Spells através da IDE
Atores	Player
Pré-Condições	A IDE está aberta e um Spell está selecionado
Fluxo Principal	1. O jogador seleciona a caixa de texto da IDE 2. O jogador digita métodos das Skills conhecidas
Fluxos Alternativos	Nenhum
Fluxos de Exceção	1. O jogador clica no botão no canto superior direito da janela 2. O Spell não compilado é deletado
Pós-Condições	O jogador terá um Spell não compilado salvo na IDE
Extensões	UC05

Elaborada pelo autor

Tabela 5 – UC05- Compile Spells at IDE

Descrição	O jogador compila Spells através da IDE
Atores	Player
Pré-Condições	O jogador está próximo de um Compiler, está com a IDE aberta e o Spell em exibição e tem Aura suficiente
Fluxo Principal	O jogador clica no botão Co à direita da IDE
Fluxos Alternativos	Nenhum
Fluxos de Exceção	Nenhum
Pós-Condições	O jogador terá um Spell compilado salvo no Codex na seção Spells
Extensões	UC06

Elaborada pelo autor

Tabela 6 – UC06 - Assign Spells into Storage

Descrição	O jogador aloca um Spell compilado para uso
Atores	Player
Pré-Condições	O jogador tem pelo menos um Spell compilado disponível na seção Spells do Codex e a janela Storage aberta
Fluxo Principal	1. O jogador clica em um espaço de Spell 2. O Codex abre mostrando os Spells compilados disponíveis ao lado do Storage 3. O jogador clica em um Spell compilado
Fluxos Alternativos	Nenhum
Fluxos de Exceção	1. O jogador clica em um espaço de Spell 2. O Codex abre mostrando os Spells compilados disponíveis ao lado do Storage 3. O jogador clica no espaço de Spell novamente 4. O Codex fecha e a seleção é cancelada 5. Modificações no espaço de seleção são descartadas
Pós-Condições	O Spell compilado selecionado fica disponível em um espaço de memória, possibilitando sua chamada pelo Terminal
Extensões	UC07

Elaborada pelo autor

Tabela 7 – UC07 - Cast Spells into Terminal

Descrição	O Actor selecionado interpreta um Spell enviado pelo jogador
Atores	Player
Pré-Condições	Um Actor está selecionado, o Terminal está selecionado e um Spell está alocado
Fluxo Principal	1. O jogador digita o endereço de memória do Spell desejado 2. O Actor interpreta os métodos do Spell
Fluxos Alternativos	Nenhum
Fluxos de Exceção	Nenhum
Pós-Condições	O Actor interpreta instantâneamente os métodos do Spell
Extensões	Nenhuma

Elaborada pelo autor

Tabela 8 – UC08 - Write Commands into Terminal

Descrição	O jogador desencadeia comportamentos no Actor selecionado
Atores	Player
Pré-Condições	Um Actor está selecionado, o jogador possui pelo menos uma Skill com um método disponível, o Terminal está selecionado
Fluxo Principal	1. O jogador insere o método de uma Skill disponível 2. O Actor interpreta o método como definido na implementação de sua Entity
Fluxos Alternativos	Nenhum
Fluxos de Exceção	1. O jogador insere o método de uma Skill disponível 2. O jogador não possui Stamina suficiente 3. O método não é executado
Pós-Condições	Um comportamento é desencadeado no Actor selecionado
Extensões	Nenhuma

Elaborada pelo autor

Tabela 9 – UC09 - Read Actor Info at InfoPanel

Descrição	O jogador lê valores de propriedades do Actor selecionado no InfoPanel
Atores	Player
Pré-Condições	Um Actor está selecionado
Fluxo Principal	As informações são exibidas no InfoPanel
Fluxos Alternativos	Nenhum
Fluxos de Exceção	Nenhum
Pós-Condições	O jogador acompanha os valores das propriedades em tempo real
Extensões	Nenhuma

Elaborada pelo autor

Tabela 10 – UC10 - Read Skill Info at Codex

Descrição	O jogador lê propriedades e métodos de Interfaces contidas em Skills disponíveis
Atores	Player
Pré-Condições	O jogador contém pelo menos um Skill e o Codex está aberto na seção Skills
Fluxo Principal	1. O jogador seleciona uma Skill na lista de Skills disponíveis 2. O jogador lê sobre propriedades, métodos e suas descrições
Fluxos Alternativos	Nenhum
Fluxos de Exceção	Nenhum
Pós-Condições	O jogador descobre possibilidades de análise e interação com Actors através das propriedades e métodos de suas Skills
Extensões	Nenhuma

Elaborada pelo autor

Tabela 11 – UC11 - Level Up Skills at Codex

Descrição	O jogador evolui Skills através do Codex
Atores	Player
Pré-Condições	O Codex está aberto, pelo menos uma Skill com possibilidade de evolução está disponível e selecionada, o jogador possui Aura suficiente
Fluxo Principal	1. O jogador clica no botão de evolução 2. A quantia de Aura é gasta 3. A Skill evolui
Fluxos Alternativos	Nenhum
Fluxos de Exceção	Nenhum
Pós-Condições	As chamadas de método serão interpretadas diferentemente pelos Actors
Extensões	Nenhuma

Elaborada pelo autor

Tabela 12 – UC12 - Read UI Information

Descrição	O jogador lê informações sobre o uso de certos elementos da Interface
Atores	Player
Pré-Condições	A janela desejada está aberta
Fluxo Principal	1. O jogador clica no título da janela 2. O jogador lê informações sobre o uso daquela janela 3. O jogador pressiona Esc para continuar
Fluxos Alternativos	Nenhum
Fluxos de Exceção	Nenhum
Pós-Condições	O jogador terá informações sobre o uso da janela selecionada
Extensões	Nenhuma

Elaborada pelo autor

Tabela 13 – UC13 - Read Messages on MessageBox

Descrição	O jogador lê mensagens dadas pelo Actor selecionado
Atores	Player
Pré-Condições	O Actor em questão implementa a Interface IVerbal
Fluxo Principal	1. O jogador seleciona o Actor em questão 2. Uma janela é exibida com mensagens daquele Actor 3. O jogador pode passar para as próximas mensagens e voltar para as anteriores 4. Um comportamento pode ser desencadeado no final das mensagens
Fluxos Alternativos	1. O Actor em questão ativa a janela de mensagens a partir de proximidade ou outra condição 2. O jogador pode passar para as próximas mensagens e voltar para as anteriores 3. Um comportamento pode ser desencadeado no final das mensagens
Fluxos de Exceção	Nenhum
Pós-Condições	O jogador terá recebido mensagens de um Actor
Extensões	Nenhuma

Elaborada pelo autor

Tabela 14 – UC14 - Acquire Skills

Descrição	O jogador adquire uma nova Skill
Atores	Player
Pré-Condições	UC03
Fluxo Principal	1. O jogador adquire uma nova Skill 2. Uma janela mostra o nome da nova Skill e sugere ir para sua definição no Codex
Fluxos Alternativos	Nenhum
Fluxos de Exceção	Nenhum
Pós-Condições	O jogador possui uma nova Skill
Extensões	Nenhuma

Elaborada pelo autor

Tabela 15 – UC15 - Acquire Items

Descrição	O jogador adquire um novo Item
Atores	Player
Pré-Condições	UC03
Fluxo Principal	1. O jogador adquire um novo Item 2. Uma janela mostra o nome do Item
Fluxos Alternativos	Nenhum
Fluxos de Exceção	Nenhum
Pós-Condições	O jogador possui um novo Item
Extensões	Nenhuma

Elaborada pelo autor

Tabela 16 – UC16 - Acquire Aura

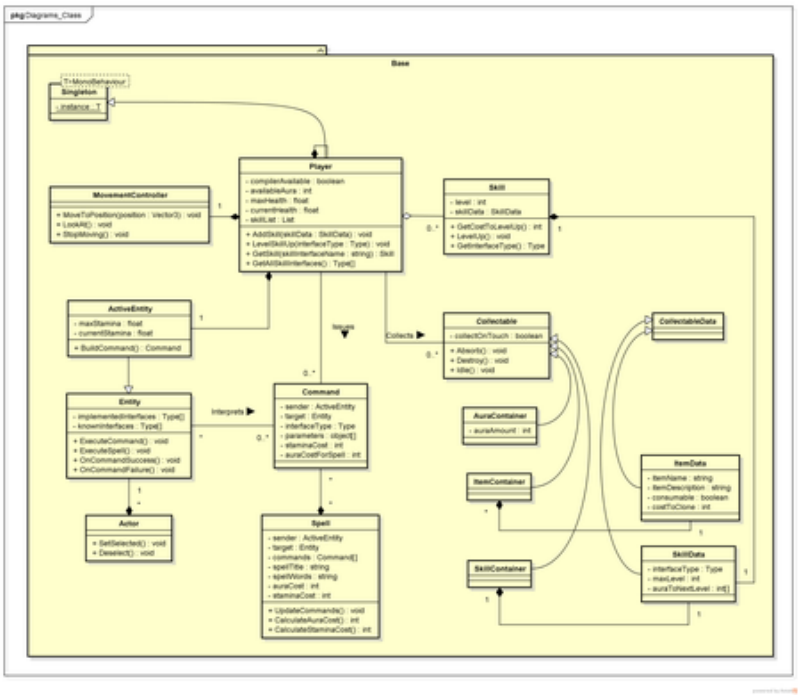
Descrição	O jogador adquire uma quantia de Aura
Atores	Player
Pré-Condições	UC03
Fluxo Principal	Uma quantia em Aura é somada ao contador no canto inferior direito da tela e disponibilizada para uso
Fluxos Alternativos	Nenhum
Fluxos de Exceção	Nenhum
Pós-Condições	Uma nova quantia de Aura é disponibilizada para uso
Extensões	Nenhuma

Elaborada pelo autor

A.2 Classes

A.2.1 Diagrama de Classes Base

Figura 16 – Diagrama de Classes das classes base



Elaborado pelo autor

A.2.2 Especificação das Classes Base

A.2.2.1 Singleton

Implementa o *design pattern* chamado Singleton. Ele permite acesso direto a uma instância única através de um atributo estático somente leitura, não precisando procurar pelo objeto, já que pode ser encontrado no tipo.

É importante para objetos únicos como gerenciadores ou a classe do jogador.

A.2.2.2 Player

É a classe do personagem controlado pelo jogador. Herda de Singleton e possui uma MovingEntity que o permite se locomover e interpretar Commands de outras Entities.

Gerencia as Skills colectadas, permitindo fazer uso delas para interagir com objetos do mundo através de Commands.

Implementa IVulnerable, o que significa que pode tomar dano de armadilhas e ataques de seus inimigos.

Figura 17 – O modelo 3D da protagonista no jogo

Elaborada pelo autor

A.2.2.3 Command

Objeto pelo qual é desenvolvida a interação entre Entities. Definido pelo tipo de Interface na qual será procurada um método e passados certos parâmetros.

Guarda a ActiveEntity que o emitiu para gasto da quantia de Stamina caso seja bem sucedido. Guarda também a Entity de alvo para que o método definido seja invocado nela.

A.2.2.4 Entity

É qualquer objeto do mundo que recebe e interpreta Commands. Implementa uma série de Interfaces dando possibilidades de interação com o jogador e outras Entities.

A.2.2.5 ActiveEntity

Herda de Entity e além de receber Commands, é capaz de enviá-los. Por isso, tem o recurso Stamina. Um comando só pode ser emitido se a ActiveEntity tiver Stamina suficiente. Esse recurso se regenera continuamente após uma espera determinada depois da emissão de um Command.

A.2.2.6 Movement Controller

Componente que permite a movimentação de personagens. Utiliza uma malha de navegação que leva em consideração os objetos estáticos da cena para definir as áreas navegáveis. Através do uso dessa malha, é possível calcular a trajetória dado um destino.

A.2.2.7 Actor

É qualquer objeto que pode ser selecionado pelo jogador e contém uma Entity à qual Commands podem ser enviados.

A.2.2.8 Spell

Responsável pelos algoritmos, essa classe passa por dois momentos. Antes da compilação, novos Commands podem ser inseridos, mudando seu custo de compilação, e seu nome pode ser mudado.

Depois de compilado, o Spell é guardado no Codex e pode ser alocado no Storage, onde recebe um endereço de memória que pode ser usado para chamá-lo em Entities através do Terminal.

A.2.2.9 Skill

Skill é uma habilidade definida por um SkillData. É o que dita quais interações são possíveis com as Entities do mundo e é o recurso mais valioso a ser encontrado por abrir novas possibilidades de interação e permitir a progressão do jogador.

Pode ou não ter progressão, tendo efeitos variados nos objetos do mundo. A progressão é feita através do Codex e custa uma quantia específica de Aura.

A.2.2.10 Collectable

É qualquer objeto que possa ser coletado, seja por contato físico ou através de um clique.

A.2.2.11 AuraContainer

Herda de Collectable e é o objeto físico que ao ser coletado dá Aura para o jogador.

Aura é o recurso finito que o jogador gasta na compilação de algoritmos, uso de itens e evolução de Skills.

A.2.2.12 ItemContainer

Herda de Collectable e é o objeto físico que ao ser coletado dá um novo Item para o jogador. Esse item é, então, guardado no Storage.

Se o item for consumível, ele pode ser usado uma só vez e não gasta recursos do jogador. Se não for consumível, ele pode ser usado quantas vezes for preciso desde que o jogador tenha Aura suficiente para clonar cada instância desejada.

A.2.2.13 SkillContainer

Herda de Collectable e é o objeto físico que ao ser coletado dá uma nova Skill para o jogador, aparecendo entre as Skills no Codex.

A.2.2.14 CollectableData

Complementar aos Collectables, que são responsáveis pela parte física dos itens, CollectableDatas contém os dados dos objetos coletados e são efetivos para definição e organização em tempo de desenvolvimento.

A.2.2.15 ItemData

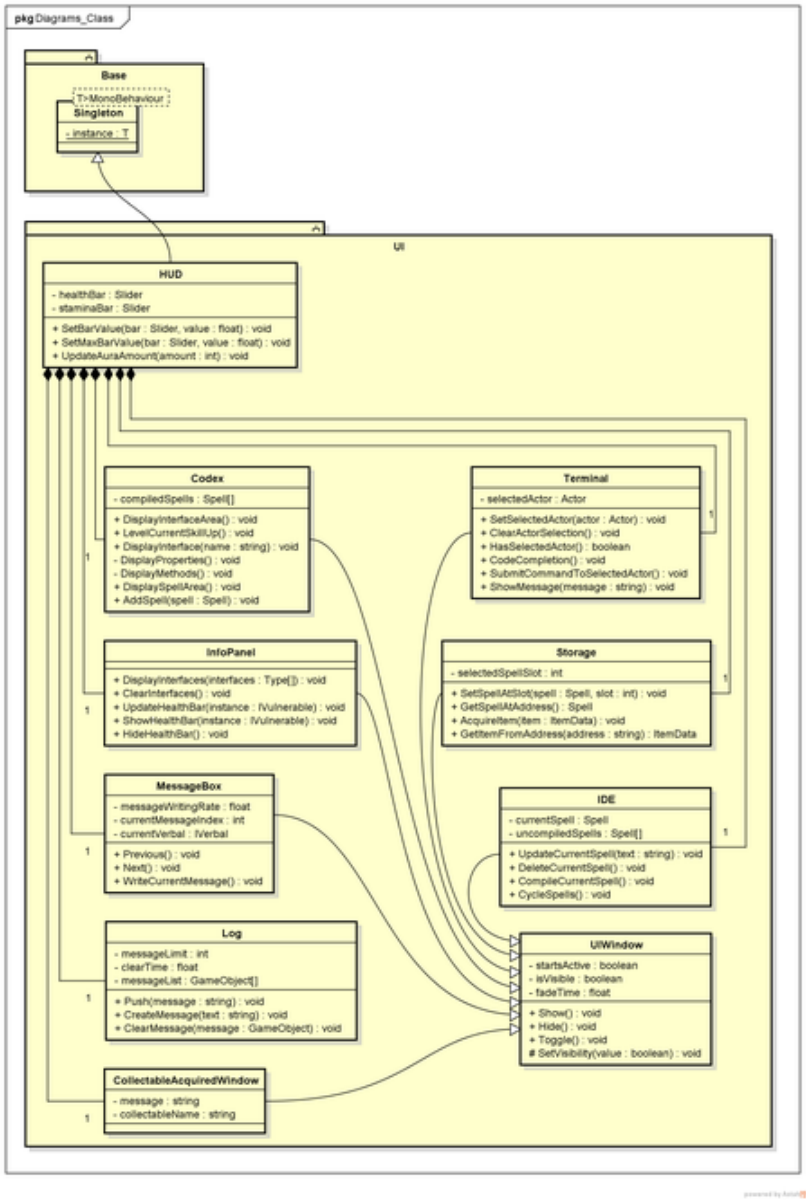
Dados de ítens coletáveis. Definido pelo nome do item, sua descrição, mostrada no Storage, se é consumível ou não e, se não, o custo em Aura para uso.

A.2.2.16 SkillData

Dados de Skills. Definido pelo tipo da Interface a exibir as propriedades e métodos, o nível máximo acessível e a quantidade de Aura necessária para evoluir cada nível.

A.2.3 Diagrama de Classes de UI

Figura 18 – Diagrama de Classes das classes de Interface de Usuário (UI)



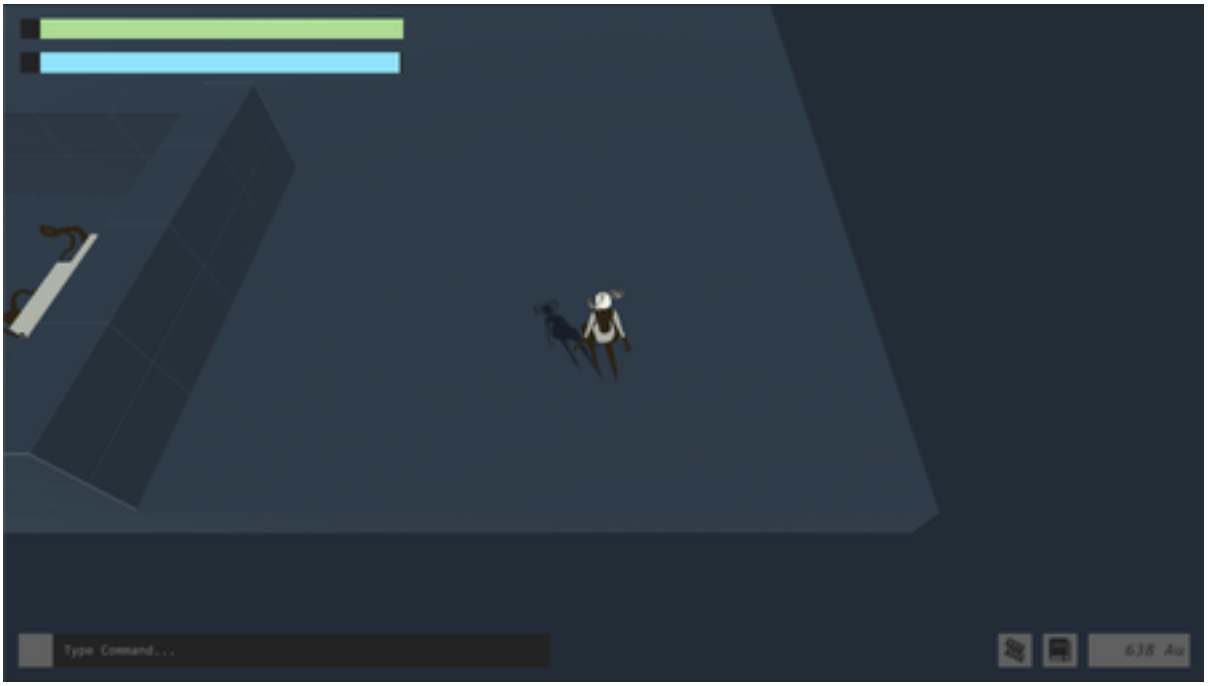
Elaborado pelo autor

A.2.4 Especificação de Classes de UI

A.2.4.1 HUD

Herdando de Singleton, o Heads Up Display, ou HUD, contém todos os elementos de interface de usuário. As barras de HP e Stamina e o contador de Aura são gerenciados por ele.

Figura 19 – Captura de tela do HUD



A.2.4.2 UIWindow

UIWindow é uma classe base que define janelas que podem ser abertas e fechadas.

A.2.4.3 Terminal

Herda de UIWindow. É o elemento principal de interação, através do qual se usam itens e chamam métodos em Actors a partir das Skills disponíveis.Tem função de permitir e auxiliar o uso de comandos, dando sugestões de conforme o jogador digita. Ao cometer erros de sintaxe, o Terminal mostra mensagens.

Também guarda o Actor selecionado pelo jogador, podendo ser consultado por outras classes através do HUD.

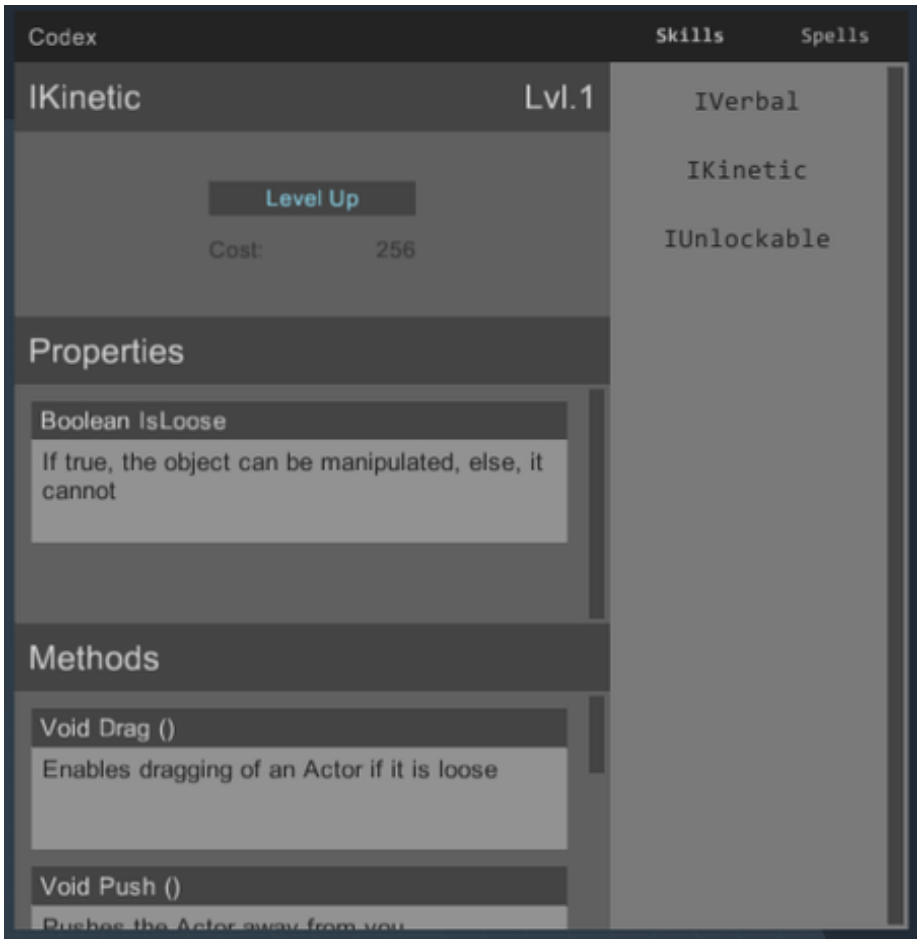
Figura 20 – Terminal aguardando comando do jogador

A.2.4.4 Codex

Herda de UIWindow. É o ponto de referência para todas as Skills adquiridas e Spells compilados.

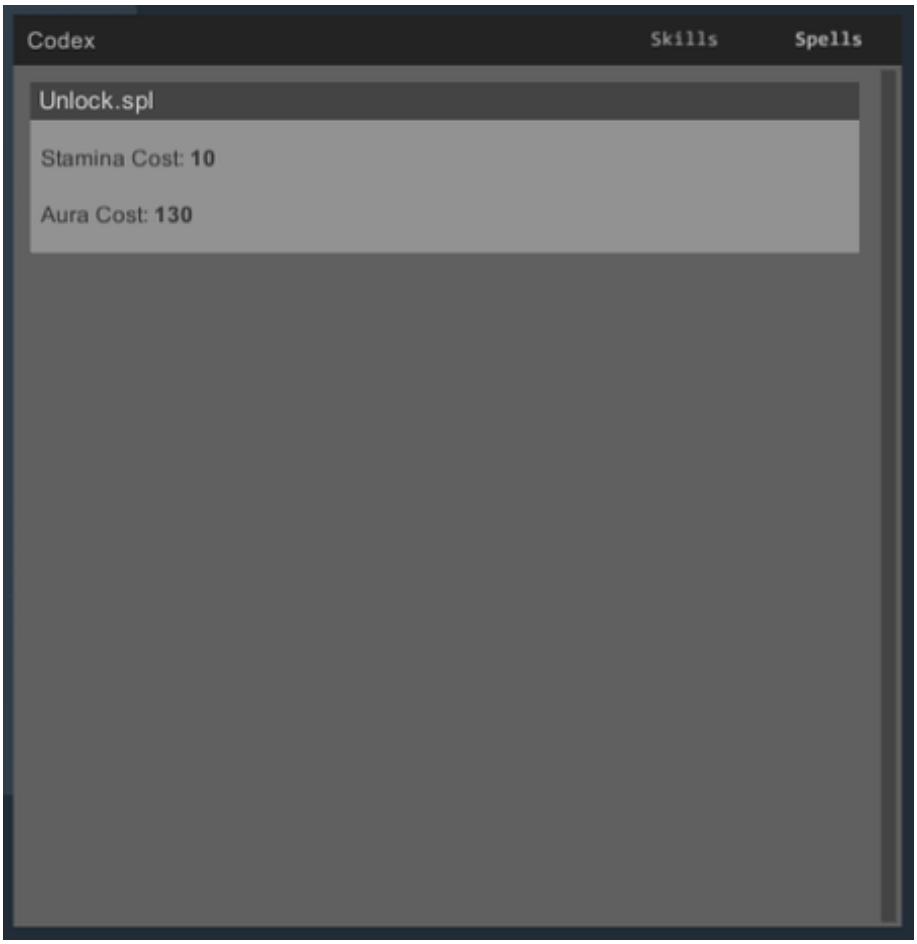
Na seção de Skills, ao selecionar uma Skill, é mostrado o botão de evolução se ela for disponível além das propriedades e dos métodos da Interface respectiva com suas descrições de uso.

Figura 21 – Codex na seção de Skills com a Skill IKinetic selecionada



Na seção de Spells, é possível ver os Spells compilados identificados por nome, a quantia de Stamina necessária para usá-los e a quantia em Aura que custaram.

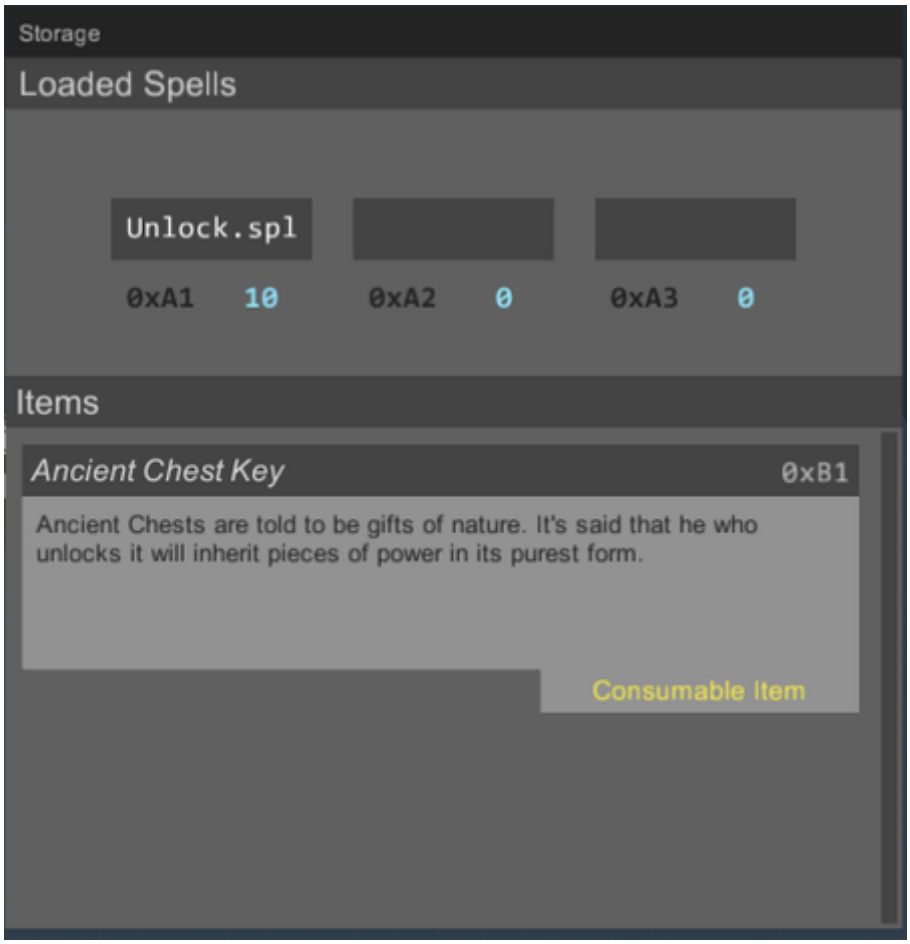
Figura 22 – Codex na seção de Spells com um Spell compilado em exibição



A.2.4.5 Storage

Herda de UIWindow. Guarda os itens coletados, exibindo suas propriedades, e permite a alocação de Spells ao clicar em um dos espaços disponíveis, abrindo o Codex na seção de Spells e selecionando um Spell compilado.

Figura 23 – Storage com um Spell alocado e um item em exibição



A.2.4.6 IDE

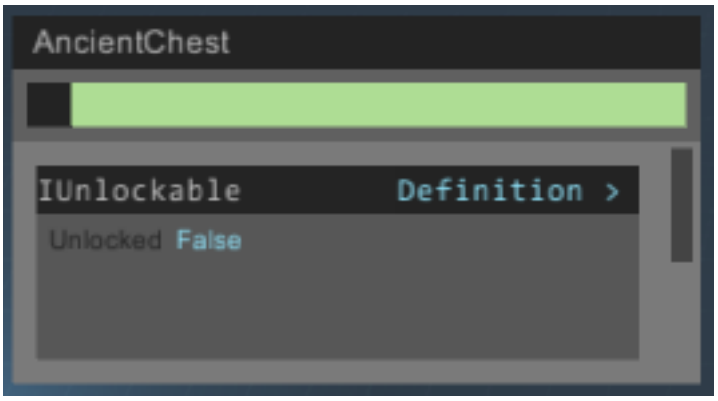
Herda de UIWindow. Permite a criação de Spells não compilados, a edição de seus comandos e seu nome e a compilação do Spell selecionado se um Compiler estiver próximo do jogador.

Figura 24 – IDE com um Spell não compilado em edição

A.2.4.7 InfoPanel

Herda de UIWindow. Permite a visualização dos valores de certas propriedades de Interfaces conhecidas pelo jogador e implementadas pelo Actor selecionado. Para cada Interface exibida há um botão que leva para a definição dela no Codex.

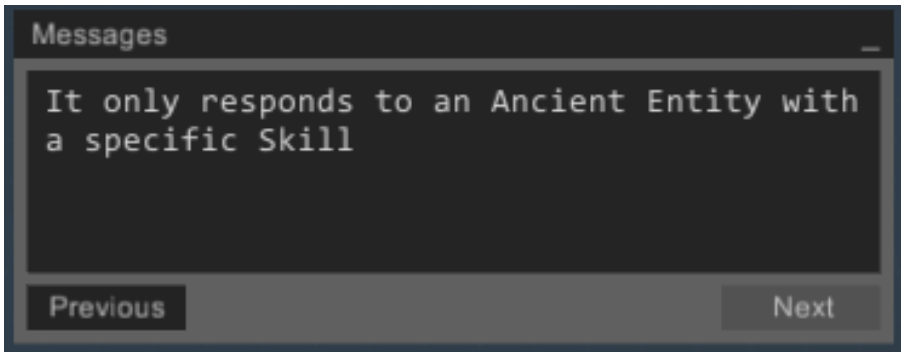
Figura 25 – InfoPanel com exibindo a Interface IUnlockable e o valor de uma de suas propriedades



A.2.4.8 MessageBox

Herda de UIWindow. Exibe mensagens de Actors se o jogador conhecer a Interface IVerbal. Alguns Actors mostram mensagens ao ser selecionados, alguns mostram de acordo com outros critérios, como distância.

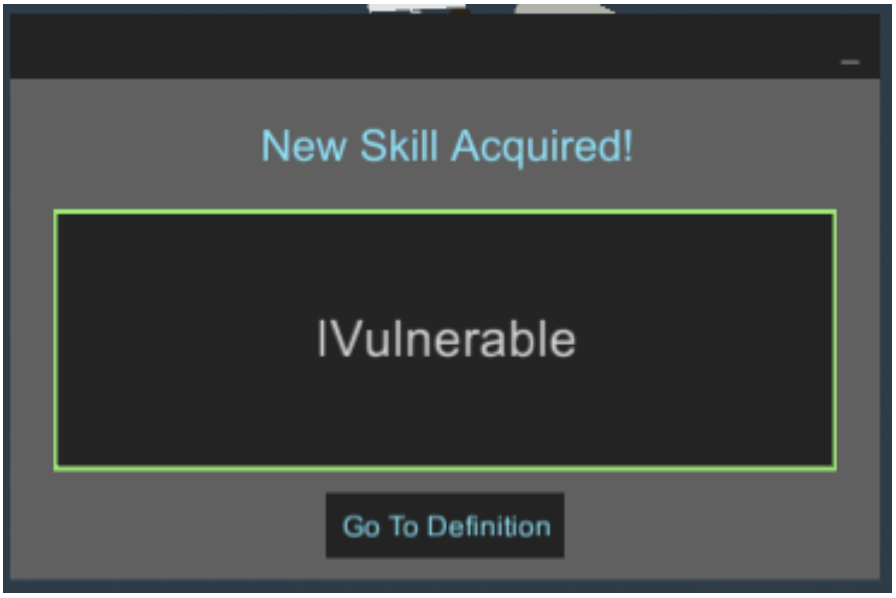
Figura 26 – MessageBox exibindo mensagens de um Actor específico



A.2.4.9 CollectableAcquiredWindow

Herda de UIWindow. É uma janela de feedback exibida quando o jogador coleta uma Skill ou um Item. Ao coletar uma Skill, mostra um botão que permite a abertura do Codex na definição da Skill adquirida.

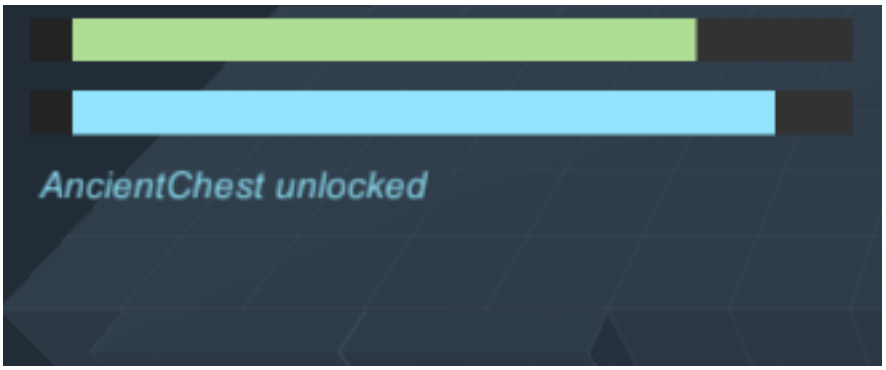
Figura 27 – CollectableAcquiredWindow sendo mostrada ao coletar a Skill IVulnerable



A.2.4.10 Log

Única classe de UI que não herda de UIWindow. É uma área da tela, abaixo das barras de energia, que exibe mensagens de feedback que somem com o tempo.

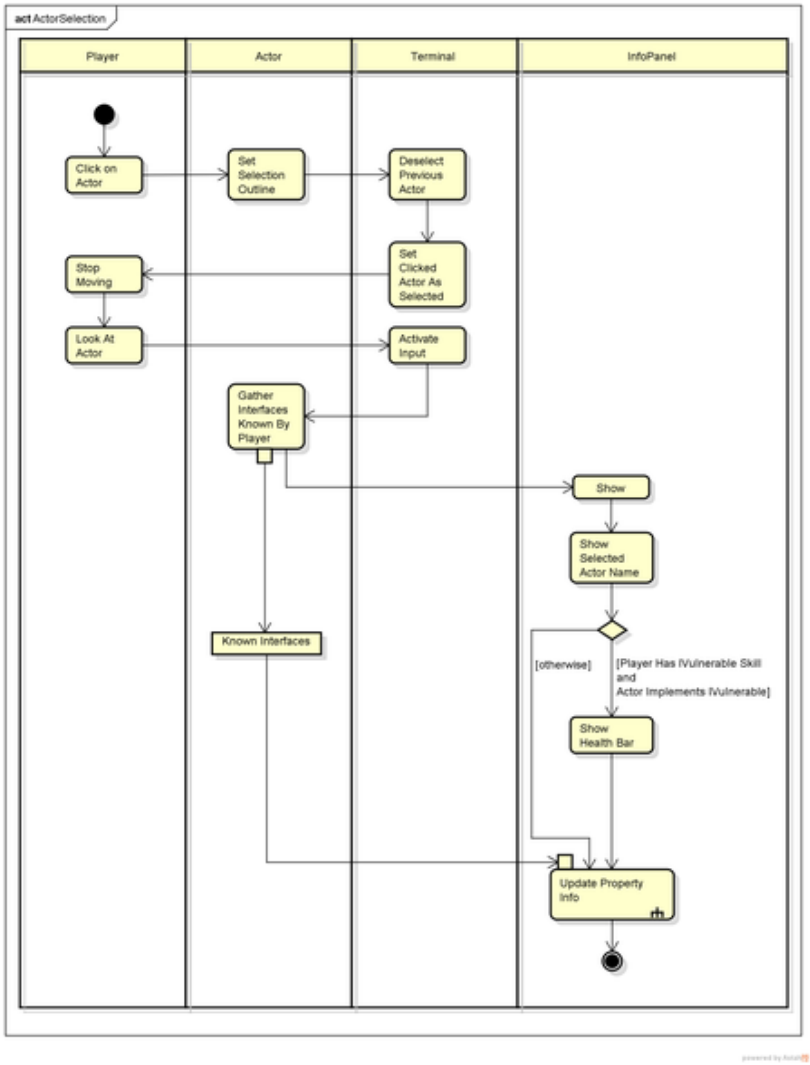
Figura 28 – Log mostrando uma mensagem de feedback



A.3 Atividades

A.3.1 AD01 - Seleção de Ator

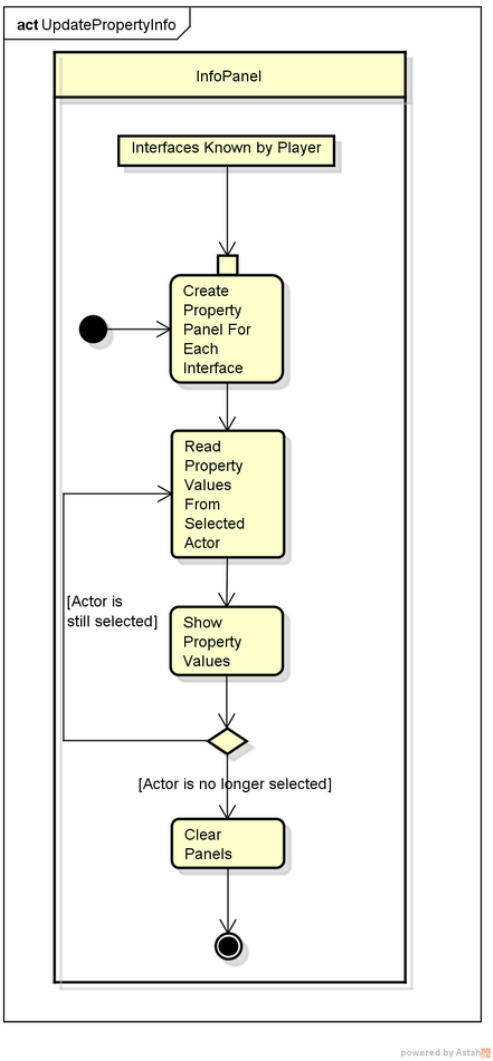
Figura 29 – Diagrama de Atividade para seleção de Actors



Elaborado pelo autor

A.3.2 AD02 - Atualização do InfoPanel

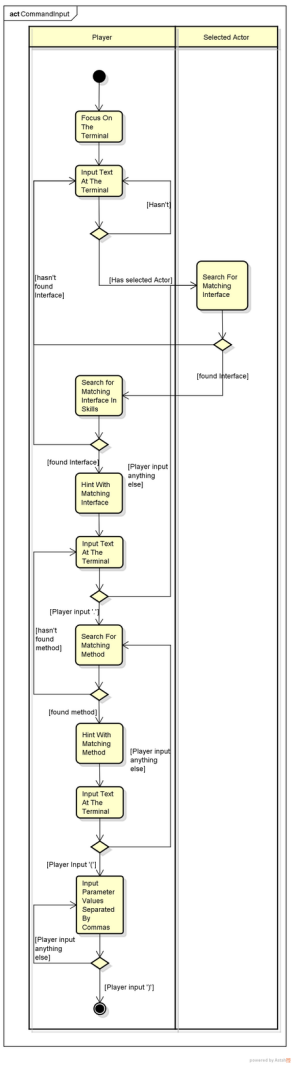
Figura 30 – Diagrama de Atividades para atualização de propriedades do Actor selecionado



Elaborado pelo autor

A.3.3 AD03 - Entrada de comandos no Terminal

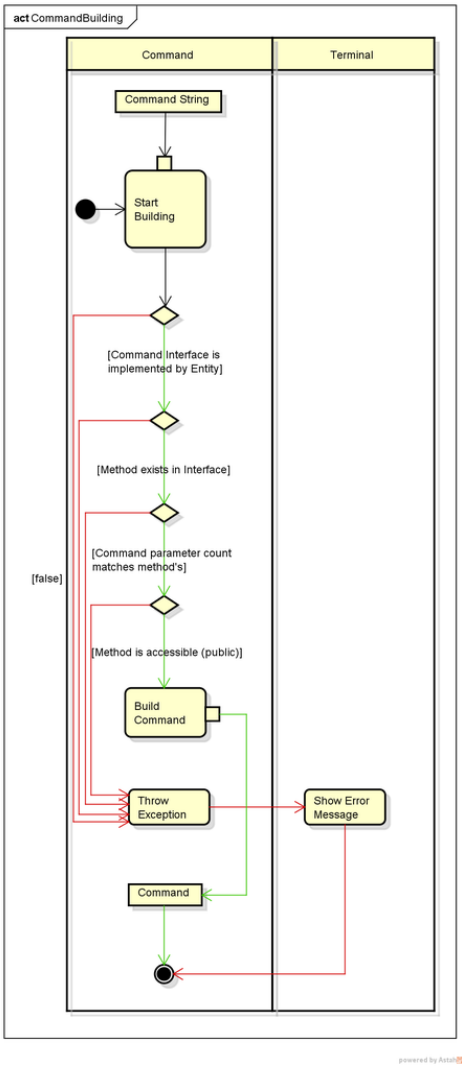
Figura 31 – Diagrama de Atividade para entrada de comandos no Terminal pelo jogador



Elaborado pelo autor

A.3.4 AD04 - Construção de Commands

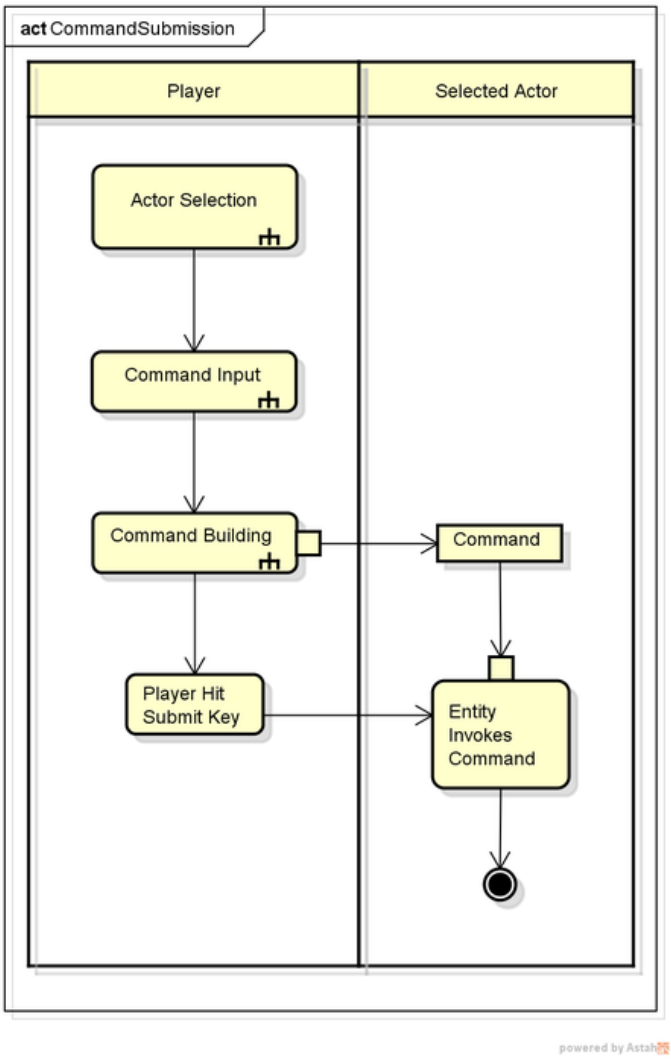
Figura 32 – Diagrama de Atividade para construção de Commands



Elaborado pelo autor

A.3.5 AD05 - Submissão de Commands

Figura 33 – Diagrama de Atividade para submissão de Commands para o Actor selecionado



Elaborado pelo autor