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
Dezembro de 2015

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
Dezembro de 2015

Vinícius Guerra Cardoso
TWOFOLD LAND: INTRODUZINDO COMPUTAÇÃO ATRAVÉS DE UM JOGO DIGITAL/ Vinícius Guerra Cardoso. – Rio de Janeiro, Dezembro de 2015-
53 p. : il. (algumas color.) ; 30 cm.

Orientador: Ricardo Portella de Aguiar

Monografia – **FACULDADE DE EDUCAÇÃO TECNOLÓGICA DO ESTADO DO RIO DE JANEIRO**
GRADUAÇÃO TECNOLÓGICA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS , Dezembro de 2015.

IMPORTANTE: ESSE É APENAS UM TEXTO DE EXEMPLO DE FICHA CATALOGRÁFICA. VOCÊ DEVERÁ SOLICITAR UMA FICHA CATALOGRÁFICA PARA SEU TRABALHO NA BILBIOTECA DA SUA INSTITUIÇÃO (OU DEPARTAMENTO).

Vinícius Guerra Cardoso

TWOFOLD LAND: INTRODUZINDO COMPUTAÇÃO ATRAVÉS DE UM JOGO DIGITAL

IMPORTANTE: ESSE É APENAS UM TEXTO DE EXEMPLO DE FOLHA DE APROVAÇÃO. VOCÊ DEVERÁ SOLICITAR UMA FOLHA DE APROVAÇÃO PARA SEU TRABALHO NA SECRETARIA DO SEU CURSO (OU DEPARTAMENTO).

Trabalho aprovado. Rio de Janeiro, DATA DA APROVAÇÃO:

Ricardo Portella de Aguiar
Orientador

Professor
Convidado 1

Professor
Convidado 2

Rio de Janeiro
Dezembro de 2015

Dedicatória. . .

Agradecimentos

Epígrafe

Resumo

Abstract

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

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

Sumário

1 Introdução

1.1 Contexto

Jogos digitais abrangem públicos diversos, sendo distribuídos através de variados modelos de negócio. Seu consumo 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.

Esses dados 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. Em média, 32% dos alunos abandonam o curso, como publicado pelo Instituto Nacional de Estudos e Pesquisas, ou INEP, e citado por (SIMAS, 2012).

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.

1.2.2 Objetivos Específicos

Os objetivos específicos são promover familiaridade com conceitos da computação e, principalmente, que esses conceitos possam ser usados na prática para a solução de problemas, o que é uma das maiores dificuldades dos alunos iniciantes, descrita por (AURELIANO; TEDESCO, 2012).

1.3 Justificativa

De acordo com (AURELIANO; TEDESCO, 2012), experiências anteriores com programação e matemática facilitam o aprendizado de certos conteúdos. Jogos digitais são uma mídia adequada para isso, já que foi verificado por Melo e Silva (2011):

> [. . .] sucesso na utilização de jogos digitais [. . .] 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.

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 (UFRGS, 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 seguir, serão mencionados alguns jogos digitais e seu método.

2.2 Jogos Educacionais

A página https://code.org/learn 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.

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.

Tanto em RPGs tradicionais, no formato de jogos de tabuleiro, quanto em RPGs digitais, uma das formas de moldar o personagem é o sistema de níveis e atributos. O jogador adquire pontos de experiência conforme realiza tarefas. Esses pontos acumulam até que o jogador passe de nível. Ele então é recompensado com a possibilidade de investir em atributos definidos pelo conjunto de regras do jogo. A alocação desses pontos tem efeitos específicos no personagem, como a liberação de novas opções de interação ou novas habilidades. Esse processo é cíclico e a quantidade de experiência necessária para passar de nível costuma crescer cada vez mais, exigindo esforço sempre maior.

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

Figura 1 – Atributos de personagem em Dark Souls® 2

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 (UFRGS, 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

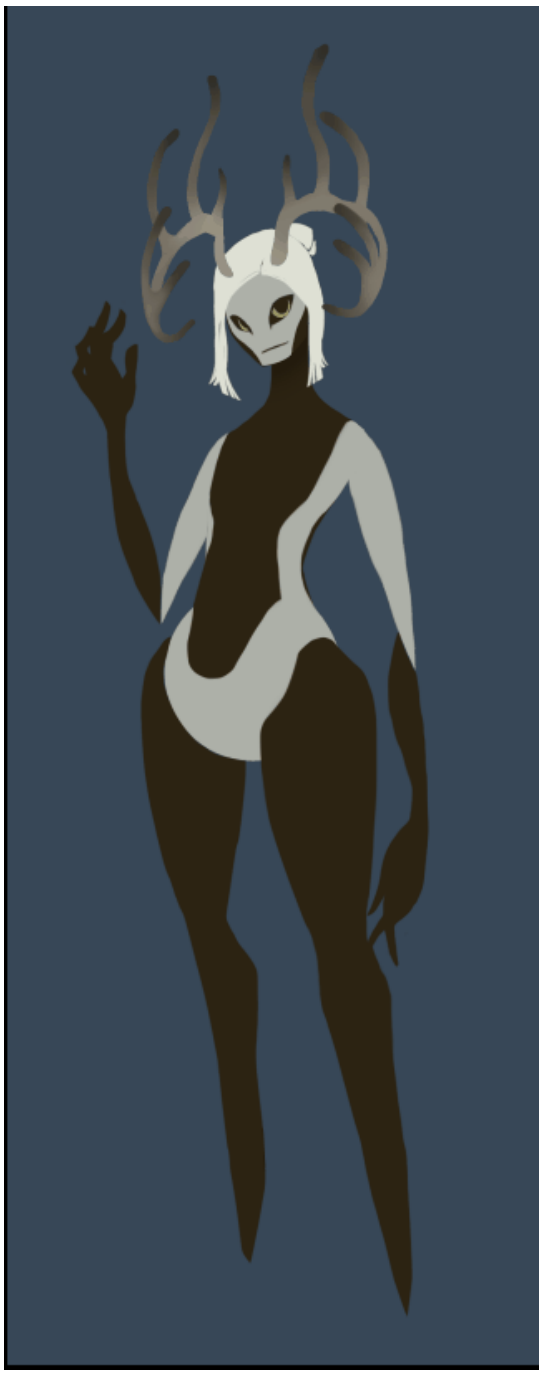
Em Twofold Land, o jogador assume o papel da protagonista, chamada Ricci. Ela é inspirada no Fauno — que é uma entidade mitológica romana — e carrega algumas de suas características. As mais marcantes são sua galhada, análoga aos chifres do Fauno e seu papel como entidade protetora e representativa da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

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.

Para que seja capaz de ela é capaz de interagir com o mundo ao entender informações sobre certos objetos e desencadear comportamentos neles, ela precisa assimilar conceitos que aprende ao absorver itens respectivos no jogo.

Ainda que haja interação com entidades do mundo, a protagonista não responde verbalmente a outros personagens. Eles se comunicam em linguagem humana, compreensível pelo jogador e de forma textual. A protagonista não responde, entretanto, pois sua comunicação com o mundo se dá através dos comandos computacionais que envia diretamente para outras entidades.

Figura 2 – Arte conceitual de Ricci, a protagonista



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. Na tela, encontram-se suas barras de *Health Points* (HP), que representa sua energia vital disponível, e *Stamina*, que representa da energia usada em interações.

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

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

3.2.1 Interação com Atores

Atores 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 caracterísiticas dos atores são nomeadas propriedades e os comportamentos, métodos. 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, construidos 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 ator, 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.

3.3 Fluxo da Fase

O jogo apresentará uma fase experimental. Nela, 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 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. Segue a descrição dessas atividades.

3.3.1 Aquisição de Interfaces

No início do jogo, a protagonista está em um ambiente trancado por tábuas de madeira e não possui nenhuma habilidade. A Interface IKinetic está disponível na sala e pode ser absorvida, permitindo interação com as tábuas.

3.3.2 Leitura do Codex

A leitura do Codex apresenta conceitos de orientação a objeto, com a apresentação de propriedades e métodos e a abstração de características comuns a vários

objetos.

Ao coletar a primeira Interface, o jogador é incentivado a ir para a definição dela, abrindo o Codex. Ao ler os métodos possíveis, o jogador é apresentado à possibilidade de puxar e empurrar objetos, o que pode ser usado nas tábuas que trancam a sala.

3.3.3 Utilização do Terminal

Ao clicar em uma das tábuas, é apresentado ao jogador que a Interface coletada constitui a tábua, logo, pode-se usar um dos métodos da Interface nela, tirando a tábua do lugar e, assim, abrindo passagem.

3.3.4 Criação de Spells

Na nova sala, existem alguns alçapões contendo Aura, uma porta com pedras bloqueando o caminho, um Compiler com um pouco de Aura perto e a Interface IUnlockable. Ao coletá-la, o jogador pode abrir os alçapões, mas eles fecham antes que ela possa puxar a Aura que está dentro. O jogador deve, então, criar um Spell e compilá-lo para abrir o alçapão e puxar a Aura rápidamente.

3.3.5 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 o ator desejado selecionado. Se o Spell criado estiver incorreto, ele pode ser descompilado, devolvendo a Aura gasta para que se possa criar outro.

3.3.6 Evolução de Interfaces

Com a Aura adquirida nos alçapões, o jogador pode evoluir a Interface IKinetic, ganhando força suficiente para empurrar as pedras que bloqueiam a passagem.

4 Implementação

4.1 Ambiente de Desenvolvimento

A implementação do jogo foi feita na *engine* — que é um ambiente para desenvolvimento de jogos — Unity® 5.2.1f1 através de *scripts* orientados a objeto em C# que usam sua API (*Application Programming Interface* ou interface de programação de aplicações). 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.*

A arte conceitual e as texturas para objetos 3D foram desenvolvidas no Adobe® Photoshop® Creative Cloud. Os objetos 3D foram desenvolvidos no Autodesk® 3ds Max® 2015.

5 Considerações Finais

5.1 Implementações Futuras

5.2 Conclusões

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.

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>.

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>.

SIMAS, A. *As graduações campeãs de desistência*. 2012. Disponível em: <http://www.gazetadopovo.com.br/educacao/vida-na-universidade/ufpr/as-graduacoes-campeas-de-desistencia-26khijqty1gurtas1veawhyz2>.

UFRGS, P. da E. *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>.

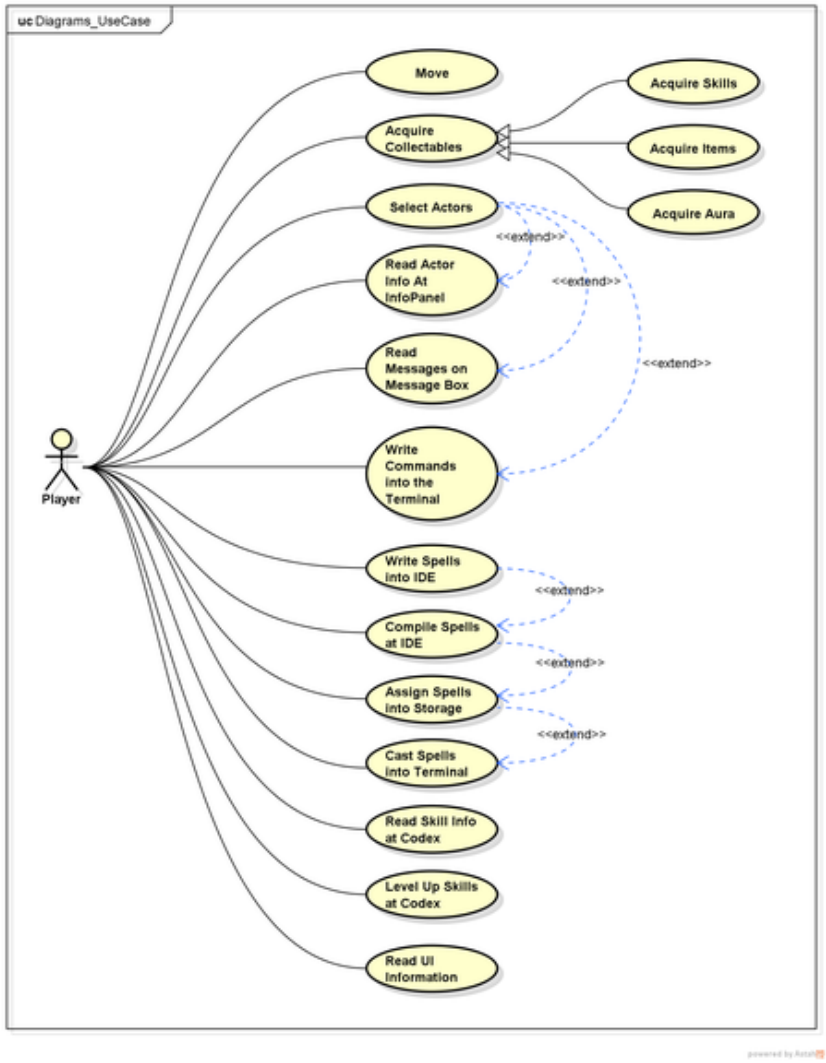
Apêndices

APÊNDICE A – Projeto Lógico

A.1 Casos de Uso

A.1.1 Diagrama de Casos de Uso

Figura 4 – 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

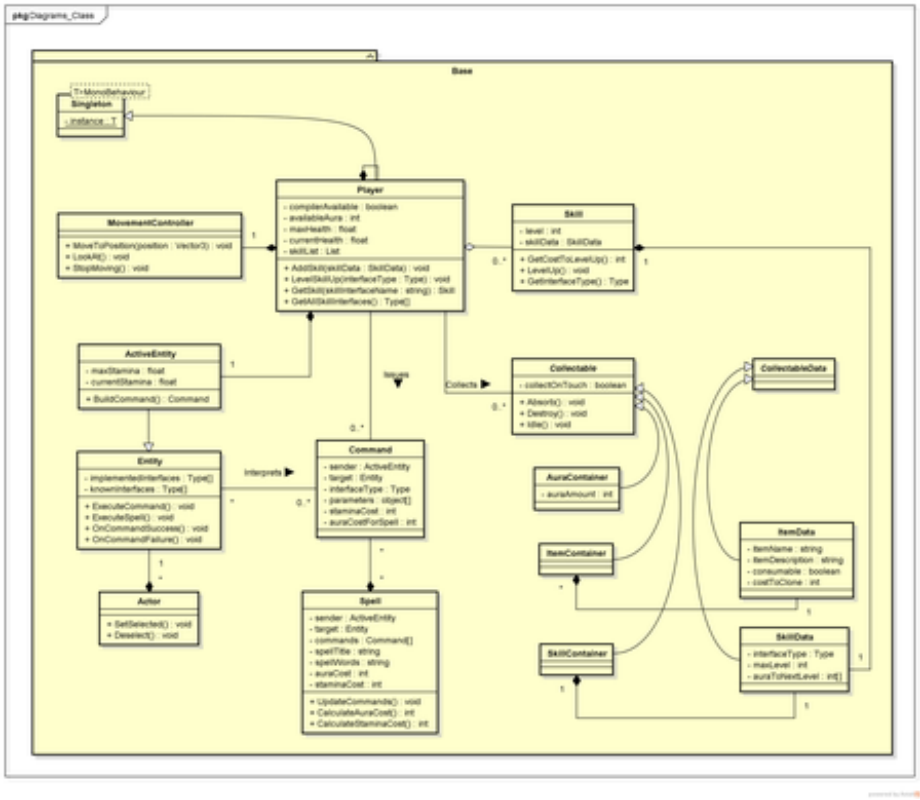
A.2.2 Especificação das Classes Base

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.

Figura 5 – Diagrama de Classes das classes base



Elaborado pelo autor

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.

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.

Figura 6 – O modelo 3D da protagonista no jogo

Elaborada pelo autor

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

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.

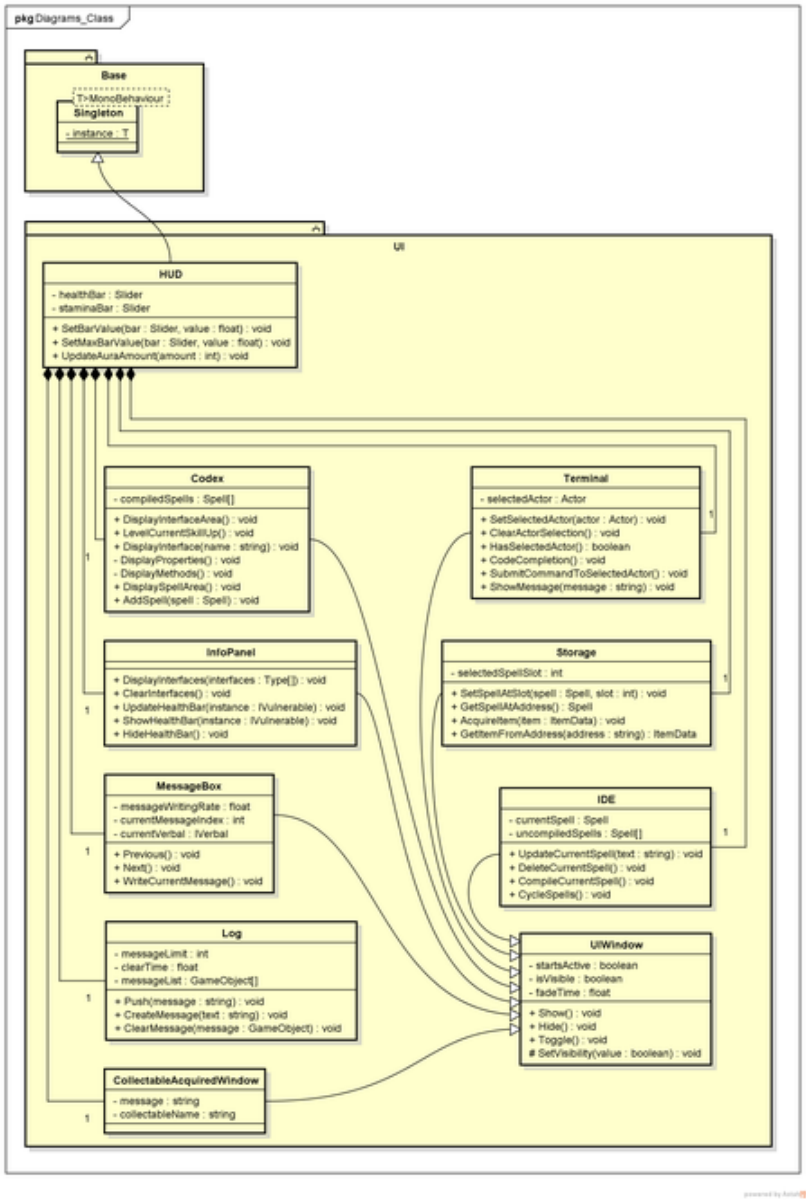
A.2.4.2 UIWindow

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

A.2.4.3 Terminal

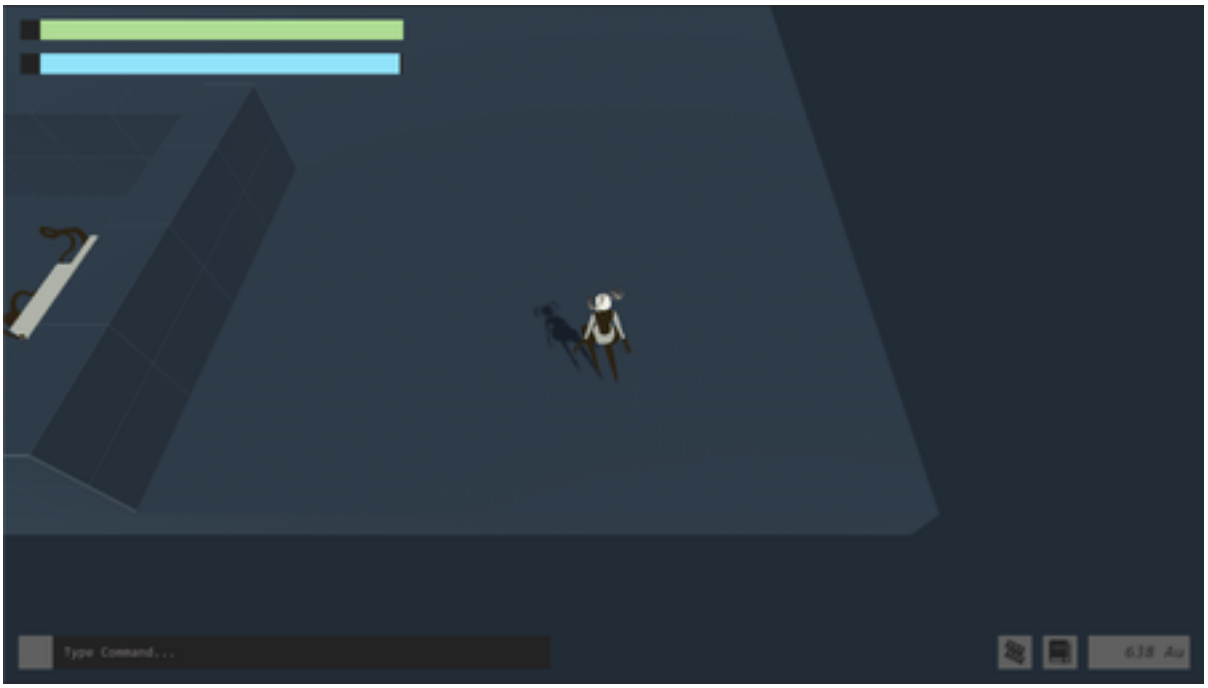
Herda de UIWindow. É o elemento principal de interação, através do qual se usam itens e chamam métodos em Actors a partir das Skills disponíveis.Tem função

Figura 7 – Diagrama de Classes das classes de Interface de Usuário (UI)



Elaborado pelo autor

Figura 8 – Captura de tela do HUD



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 9 – 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.

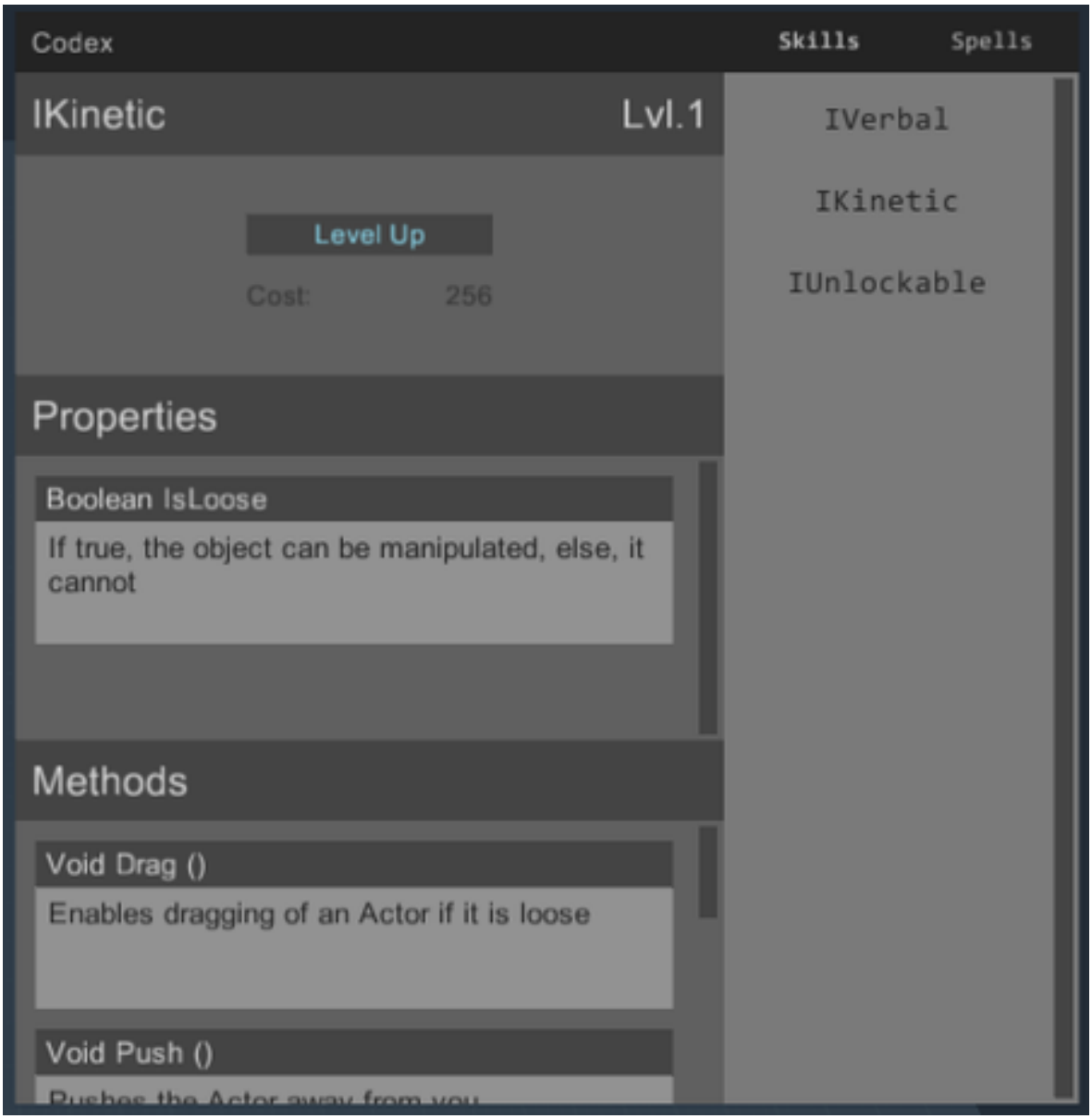
Na seção de Skills, ao selecionar uma Skill, é mostrado o botão de evolução se ela for disponível além das propriedades e dos métodos da Interface respectiva com suas descrições de uso.

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.

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

Figura 10 – Codex na seção de Skills com a Skill IKinetic selecionada



Codex na seção de Spells e selecionando um Spell compilado.

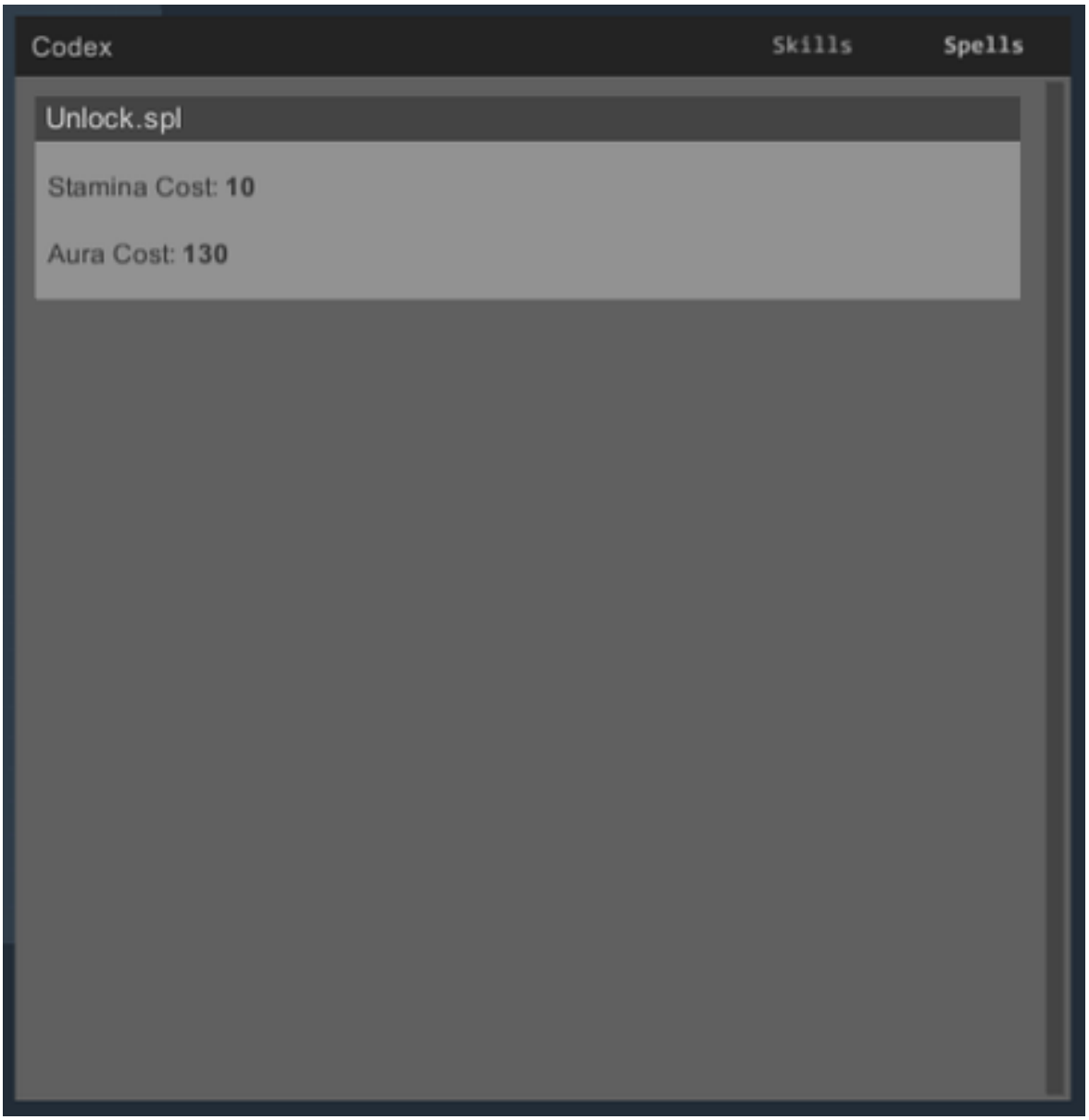
A.2.4.6 IDE

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.

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



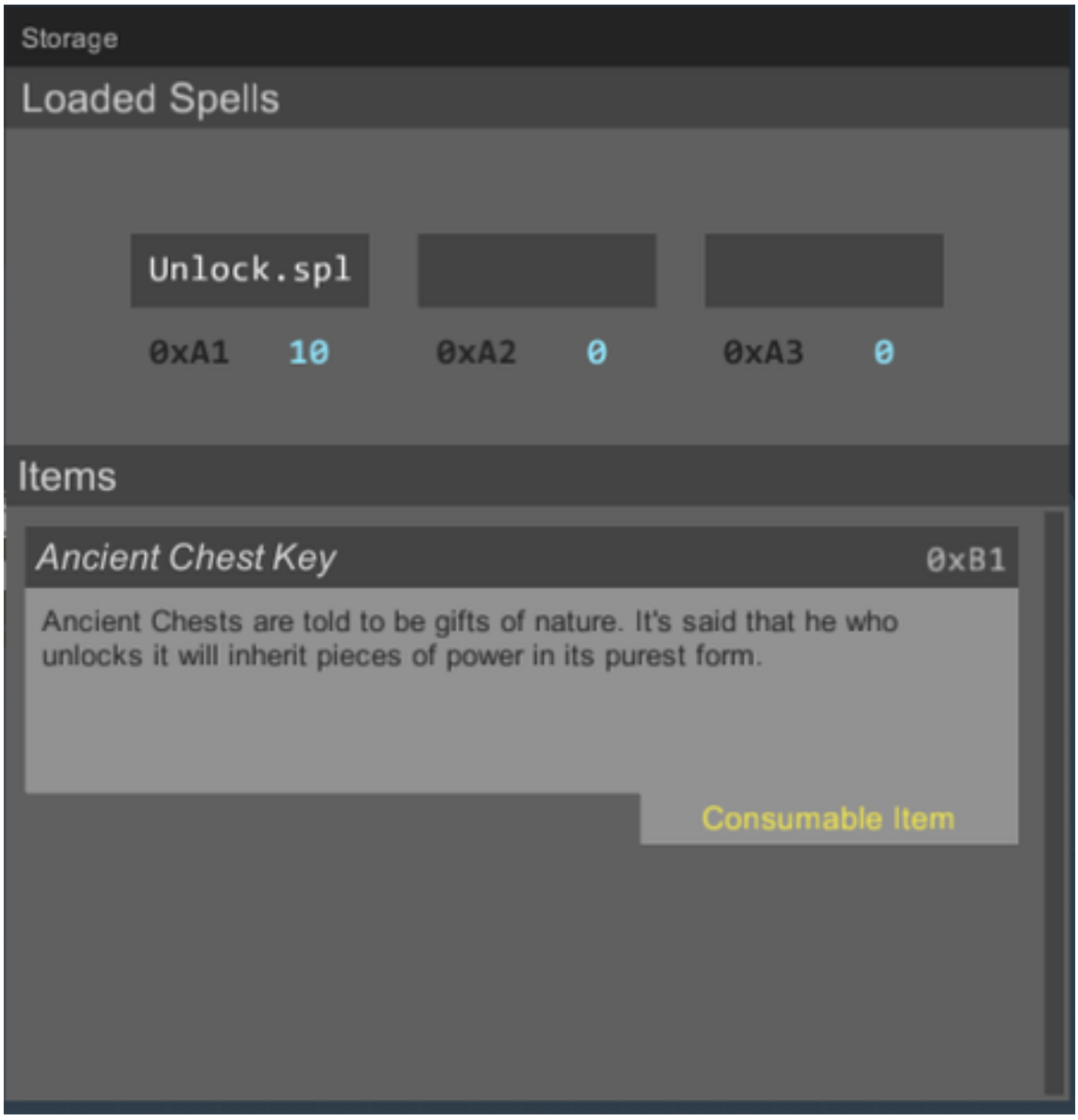
A.2.4.8 MessageBox

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.

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 12 – Storage com um Spell alocado e um item em exibição



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 13 – IDE com um Spell não compilado em edição

Figura 14 – InfoPanel com exibindo a Interface IUnlockable e o valor de uma de suas propriedades

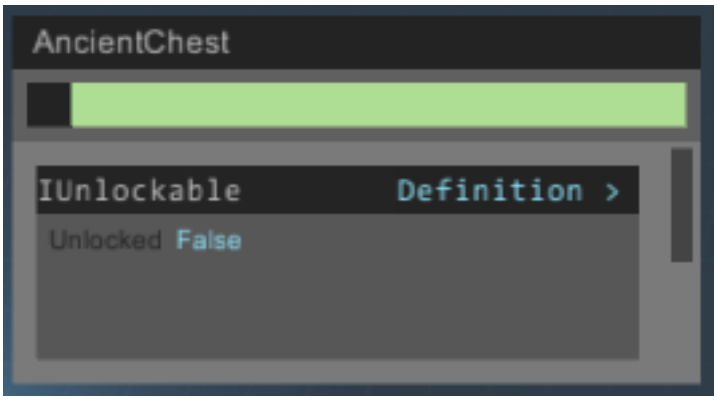


Figura 15 – MessageBox exibindo mensagens de um Actor específico

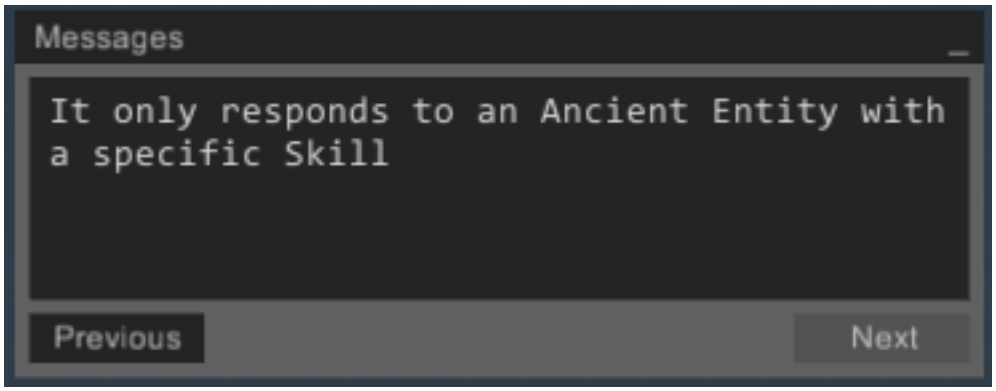


Figura 16 – CollectableAcquiredWindow sendo mostrada ao coletar a Skill IVulnerable

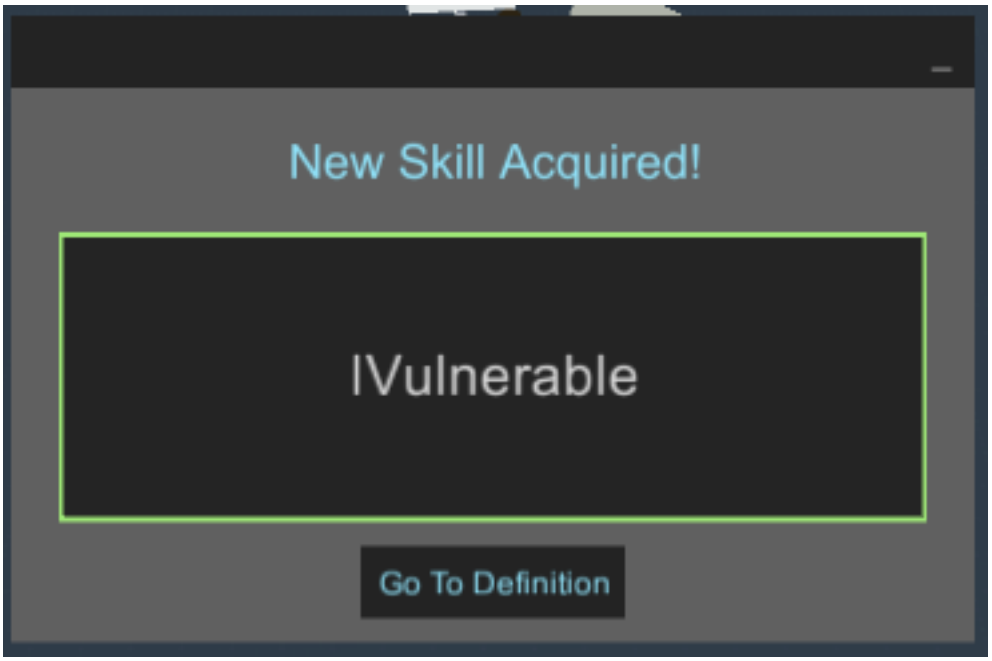
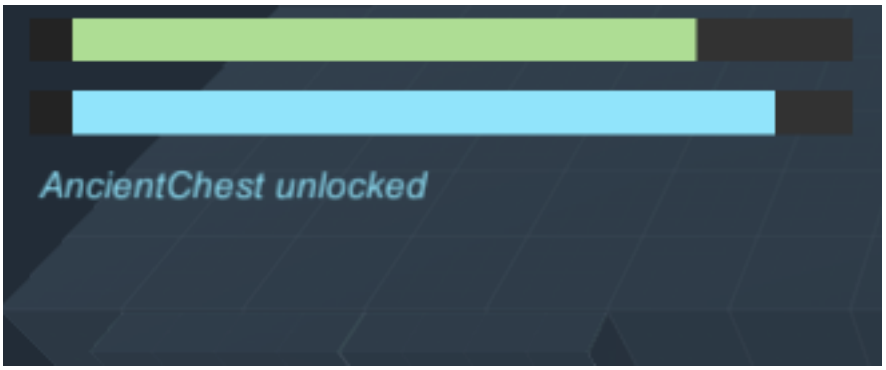


Figura 17 – Log mostrando uma mensagem de feedback



A.3 Atividades

A.3.1 AD01 - Seleção de Ator

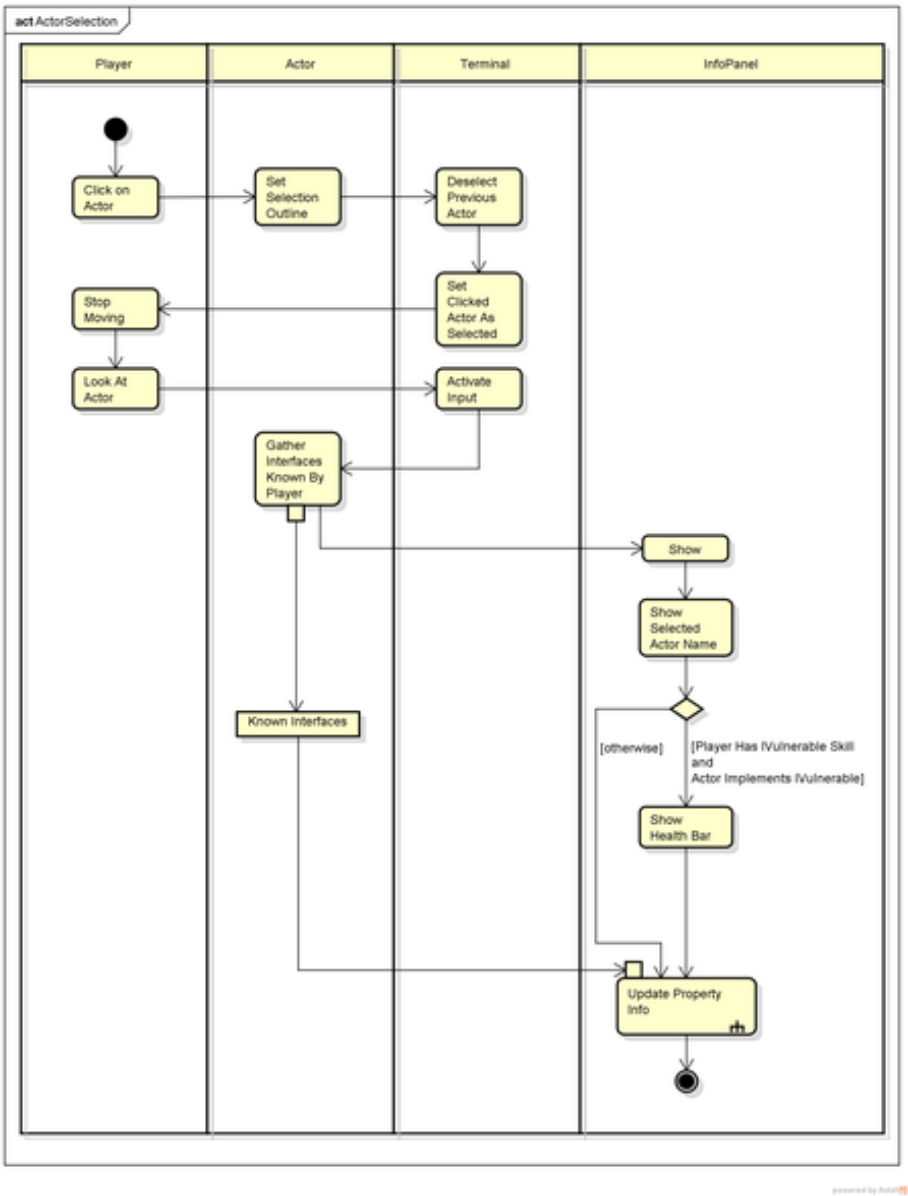
A.3.2 AD02 - Atualização do InfoPanel

A.3.3 AD03 - Entrada de comandos no Terminal

A.3.4 AD04 - Construção de Commands

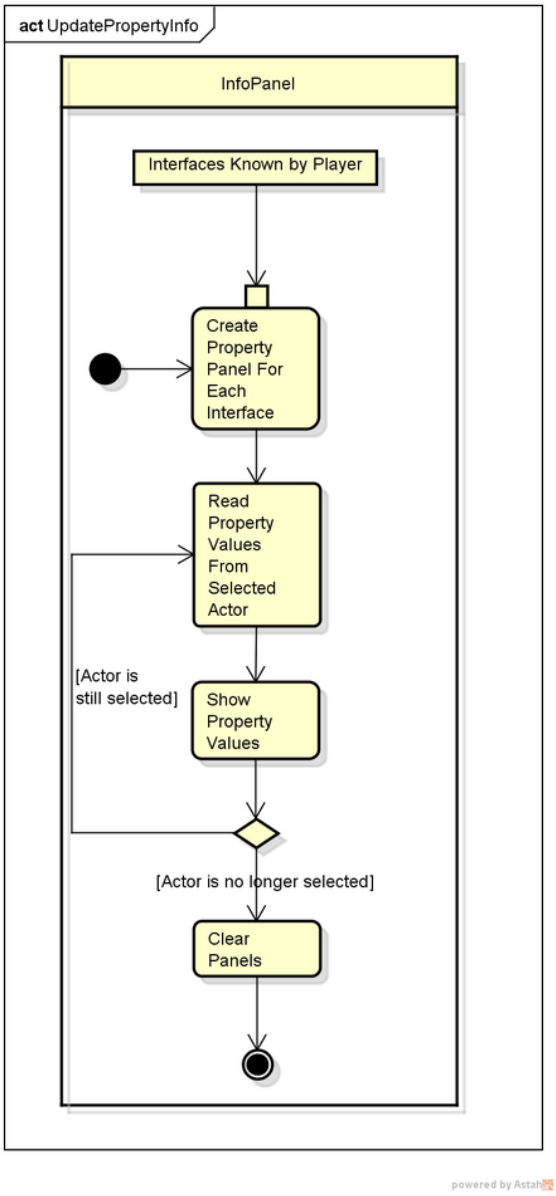
A.3.5 AD05 - Submissão de Commands

Figura 18 – Diagrama de Atividade para seleção de Actors



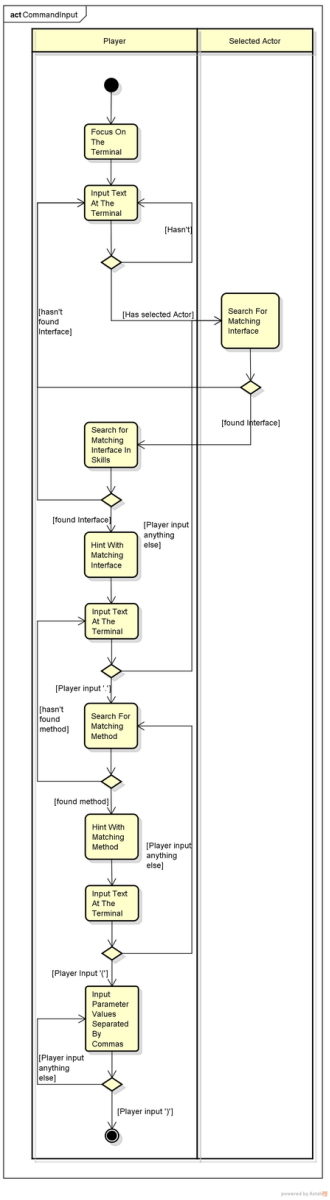
Elaborado pelo autor

Figura 19 – Diagrama de Atividades para atualização de propriedades do Actor selecionado



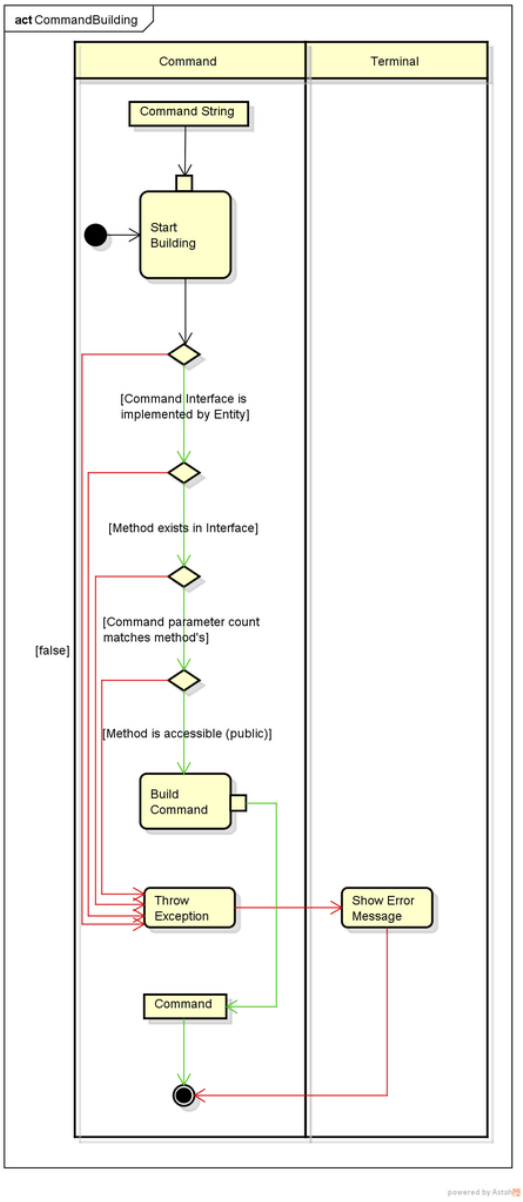
Elaborado pelo autor

Figura 20 – Diagrama de Atividade para entrada de comandos no Terminal pelo jogador



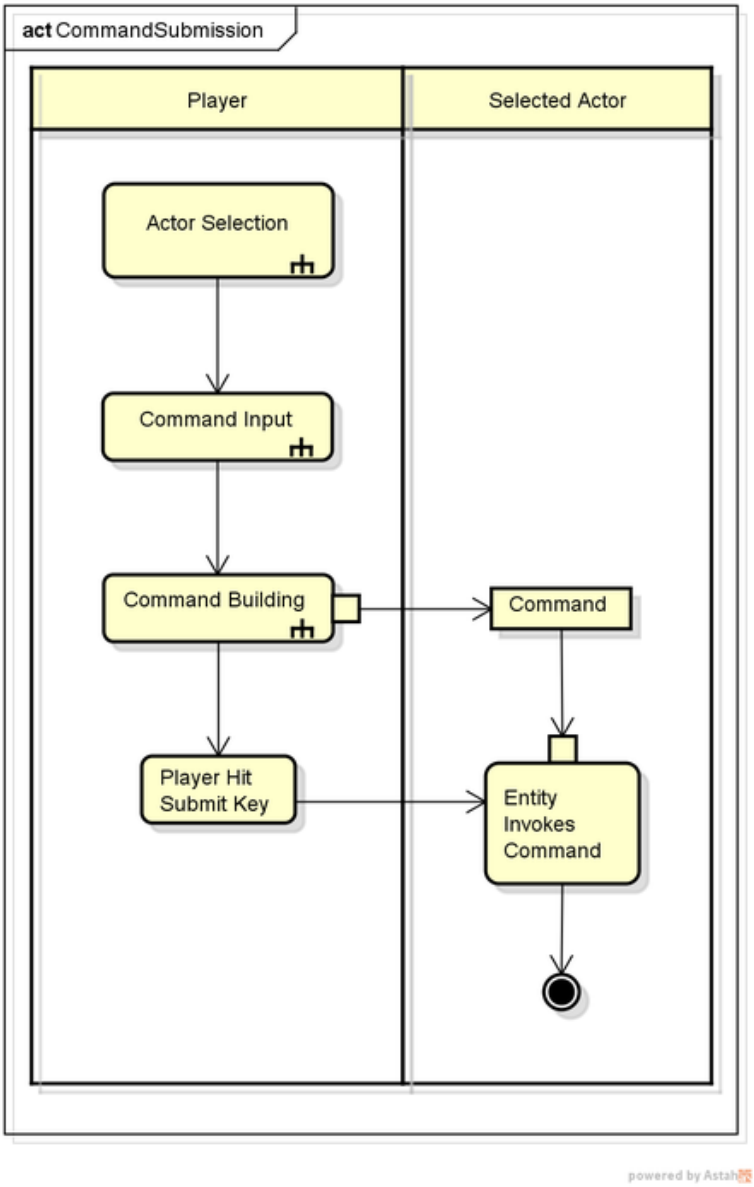
Elaborado pelo autor

Figura 21 – Diagrama de Atividade para construção de Commands



Elaborado pelo autor

Figura 22 – Diagrama de Atividade para submissão de Commands para o Actor selecionado



Elaborado pelo autor