Desenvolvimento e análise do impacto do jogo Cabinet na aprendizagem de Gestão de Sistemas e Redes

Carlos Antonio Bertoncelli Júnior

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação. No âmbito da dupla diplomação com a Universidade Tecnológica Federal do Paraná.

Trabalho orientado por:

Prof. Rui Lopes

Prof. Jorge Aikes Junior

Bragança

2017-2018

Desenvolvimento e análise do impacto do jogo Cabinet na aprendizagem de Gestão de Sistemas e Redes

Carlos Antonio Bertoncelli Júnior

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação. No âmbito da dupla diplomação com a Universidade Tecnológica Federal do Paraná.

Trabalho orientado por:

Prof. Rui Lopes

Prof. Jorge Aikes Junior

Bragança

2017-2018

Dedicatória

Dedico este trabalho aos meus pais Carlos Antonio Bertoncelli e Lisandra Aparecida de Souza Bertoncelli, que jamais deixaram de incentivar meus estudos. E que, sempre estiveram presentes e me apoiando em todas as etapas da minha vida.

Agradeço também Daisi Marsaro, que me deu suporte, conselhos e dicas ao longo do desenvolvimento de todo o trabalho, além de ter sido uma companheira excepcional em todos os momentos, ainda foi capaz de me ajudar com sua opinião em todas as esferas desta dissertação, assegurando a qualidade deste trabalho.

Agradecimentos

A realização desta dissertação de mestrado contou com importantes apoios e incentivos sem os quais não se teria tornado uma realidade e aos quais estarei eternamente grato.

Agradeço ao Professor Rui Pedro Lopes pela disponibilidade, atenção e o seu conhecimento na área que por meio de sua orientação puderam tornar possível o desenvolvimento desta dissertação.

Ao Professor Jorge Aikes Junior agradeço pela sua orientação, seu total apoio, disponibilidade, suas opiniões e críticas que serviram de colaboração ao longo do desenvolvimento desta dissertação.

Resumo

A utilização de jogos como atividades de ensino-aprendizagem tem o potencial de incrementar a motivação e, consequentemente, o envolvimento dos alunos na construção do seu conhecimento. Esta estratégia, designada por aprendizagem baseada em jogos (game-based learning) têm sido amplamente utilizada nos últimos anos em conjunto com metodologias de ensino ativas.

Neste trabalho descreve-se o desenvolvimento e aplicação de um jogo digital de estratégia para tablets, denominado Cabinet, criado com o objetivo de auxiliar na aprendizagem de sistemas e de redes, simulando um ambiente de gestão de data-centers. Este jogo, desenvolvido inicialmente para tabuleiro numa abordagem por turnos em competição entre dois jogadores, posiciona cada jogador como um gestor de data-center, devendo competir por recursos limitados e construir a melhor infraestrutura. O jogador que conseguir maior eficiência da alocação de recursos e número de serviços em execução é considerado o vencedor. Este jogo procura desenvolver no aluno diversos aspetos importantes que auxiliam no aprendizado e devem ser levados em consideração quando se trata da criação e gestão de um data-center e como suas escolhas definem como serão utilizados os recursos que têm ao seu dispor. Este jogo foi desenvolvido e utilizado no corrente ano, na unidade curricular de Gestão de Sistemas e de Redes de um curso de Engenharia Informática. O impacto do jogo foi avaliado por meio de análise quantitativa dos dados obtidos por dois processos: observação da interação jogador-jogador e jogador-jogo e entrevista semi-estruturada.

Palavras-chave: aprendizado baseado em jogos, gamificação, gamificação no ensino superior.

Abstract

The use of games as teaching-learning activities has the potential to increase motivation and, consequently, the involvement of students in the construction of their knowledge. This strategy, defined as game-based learning has been widely used in recent years in conjunction with active teaching methodologies.

This work describes the development and application of a digital strategy game for tablets, called Cabinet, created with the objective of assisting in the learning of systems and networks, simulating a data center management environment. This game was created initially to be a turn-based board game where two players play against each other, each player is a data center manager, being a competition for limited resources and a better infrastructure. The player who achieves greater resource allocation efficiency and number of running services is considered the winner. This game goal is to develop certain skills and important aspects in the student and teach what should be taken into account when it comes to creating and managing a data center and how the choices define the way that the resources will be used. This game was developed and used in the curricular unit of Management of Systems and Networks of Formation of Computer Systems. The impact of the game was evaluated through the qualitative analysis of the data obtained by the two processes: the player-player and player-game interactions and the semi-structured interview.

Keywords: game-based learning, gamification, gamification in higher education.

Conteúdo

Lista de Tabelas

Lista de Figuras

Siglas

API Application Programming Interface. 63

DNS Domain Name System. 56

EAD Ensino à Distância. 10

GPL General Public License. 50

GUI Graphical User Interface. 67

HUD Heads Up Display. 62

IA Inteligência Artificial. 70

IC Iniciação Científica. 14

LAN Local Area Network. 57

MMORPG Massive Multiplayer Online Role Playing Game. 17

RPC Remote Procedure Call. 38, 60–63

RPG Role-Playing Game. 23

TCC Trabalho de Conclusão de Curso. 14

UE Unreal Engine. 37, 38

Capítulo 1

Introdução

Desde 2010 a palavra Gamificação se tornou muito utilizada em diversos tipos de ambientes e atividades, sua utilização envolve conhecimento empírico sobre aspectos psicológicos dos indivíduos. O ato de Gamificar algo refere-se a usar elementos, estruturas e características de jogos em tarefas ou ações que não sejam jogos [1]. Para que um jogo possa cativar seu público deverá ter formas de recompensar seu jogador, algum tipo de reforço e um sistema de feedback focado em melhorar o envolvimento afim de tornar o jogo ainda mais atrativo [2].

Atividades de aprendizagem que permitam o envolvimento pessoal do indivíduo com o meio, baixa pressão e flexibilidade de aprendizagem garantem que se, dada a devida a liberdade de escolha, tornando a atividade mais atrativa para os indivíduos que a exercem, ainda, que em ambiente escolar se desenvolva uma competição saudável entre os indivíduos em busca do melhor desempenho[3].

Na área da educação e ensino, técnicas de Gamificação já foram usadas algumas vezes para melhorar a motivação e engajamento dos indivíduos. Apesar de ser uma área atualmente em desenvolvimento, que necessita de mais estudos para definir as melhores maneiras de gamificar certas atividades, já são conhecidas algumas dificuldades, por exemplo, a necessidade de abordagens distintas para gamificar atividades, trabalhos em grupo e outros elementos de interação em sala de aula [1]. Este ainda é um campo muito novo e que precisa ser estudado em diversos contextos pois suas aplicações podem ser bem

subjetivas. A objetividade de tornar o processo de ensino gamificado é fazer com que o estudante procure o conhecimento por si só dentro de um ambiente competitivo afim de melhorar seu desempenho e prepará-lo para o mercado de trabalho.

1.1 Enquadramento

No contexto atual, toda vez que um indivíduo age ou realiza uma tarefa na qual não o faz por obrigação, mas sim por vontade própria, acaba por ter mais motivação na tarefa e faz com que, a partir da geração de um hábito, isso se torne um lócus interno, proporcionando disciplina e autonomia para o individuo[4]. Olhando por outro ângulo, os indivíduos que seguem um modelo pré determinado demonstram inúmeros sentimentos negativos, pois sentem-se controlados por fatores externos, reduzindo a motivação.

No decorrer dos anos, o processo de aprendizagem foi constantemente adaptado à realidade dos indivíduos com o intuito de estimular a motivação e o engajamento. A utilização de metodologias de ensino ativas e mais participativas beneficiam os estudantes, fazendo com que a interação durante uma aula, por exemplo, não dependa somente do professor mas de forma que os alunos contribuam com o desenvolvimento geral[5]. A utilização de jogos em sala de aula tem-se mostrado uma maneira eficiente de proporcionar cenários desafiadores onde o aluno passa a lidar com a consequência de suas ações e também a ser recompensado por elas.

1.2 Objetivos

O objetivo geral desta dissertação é realizar o desenvolvimento da versão digital do jogo de tabuleiro denominado Cabinet para dispositivos móveis. Além disso, também foram realizados testes de aplicação em sala de aula com o objetivo de perceber o impacto no engajamento e motivação dos alunos. A partir do objetivo geral, foram propostos os seguintes objetivos específicos desta dissertação:

- Compreender e levantar todos os requisitos necessários para a criação da versão

digital do jogo Cabinet;

- Desenvolver o jogo Cabinet para dispositivos móveis;
- Aplicar o jogo em sala de aula;
- Analisar o impacto do jogo no aprendizado dos alunos por meio da observação da interação jogador-jogador e jogador-jogo;
- Avaliar o jogo por meio de entrevistas semi-estruturadas;

Capítulo 2

Contexto

Neste capitulo é realizada a descrição do tema deste trabalho, iniciando pelo detalhamento dos paradigmas pedagógicos educacionais e em seguida uma breve explicação sobre a metodologia de ensino tradicional e ativa.

2.1 Paradigmas Pedagógicos

O termo paradigma foi inicialmente utilizado com o intuito de designar um exemplo em relação a algo. Pode ser utilizado também para determinar as características dentro de um periodo histórico. Atualmente é utilizado pela maioria dos autores de forma a conceituar um processo ou padrão que possua um certo embasamento teórico, metodologia sólida e amplamente aceita [6].

No decorrer da história da humanidade, diversas formas de pensar e agir foram colocadas em prática com base nos estudos realizados por filósofos, matemáticos, fisicos e outros estudiosos. Essa evolução na forma de pensar e agir proporcionou à humanidade diversas transformações sociais, políticas e econômicas de maneira continua e dinâmica, mudando as crenças e valores acerca de diversos paradigmas [6].

A pedagogia pode ser definida como um conjunto de práticas e técnicas com o objetivo de tornar o método de ensino-aprendizagem mais eficiente para as pessoas envolvidas no processo. A área pedagógica possui uma abrangência que vai além da disposição dos

métodos de ensino em sala de aula, contempla também a disposição do ambiente escolar como um todo e busca encontrar as melhores metodologias de ensino para que o processo de aprendizado se torne mais eficiente e adequado para o maior número de estudantes possível [7].

Nas subseções seguintes estarão apresentadas e descritas os principais paradigmas pedagógicos e suas peculiaridades.

2.1.1 Prática Pedagógica Conservadora

O paradigma conservador é um dos paradigmas mais amplamente utilizado. Este paradigma vem sofrendo as influências de ideologias mais focadas na área cientifica e racional, tendo sofrido diversas influências do modelo proposto por Isaac Newton e René Descartes, de forma que o aprendizado deveria ocorrer de uma maneira linear e mais mecanicista focando ideologicamente na racionalidade e fatos, portanto não dando espaço para a subjetividade [8]. Desde a época de Galileu Galilei este modelo pedagógico vem sendo difundido com foco na disseminação de informações exatas, evitando o ensinamento de conhecimentos dúbios ou não comprovados no meio científico [9].

A necessidade de uma escolarização em massa após o período da revolução industrial e afim de realizar uma normalização dos ensinos e conteúdos lecionados fez com que a adoção do paradigma pedagógico conservador ocorresse. A busca por métodos de ensinos mais padronizados e conteúdos focados em determinadas áreas ocasionou no surgimento da teoria técnica do *curriculum* que visa estruturar um plano de aprendizagem graduativo e linear centrado nos alunos ou conteúdos. O *curriculum* detém ainda as matérias a serem lecionadas, a ordem em que devem ser apresentadas e de que forma será ensinado ao aluno, sendo considerado como uma forma de plano de estudos e tornando o professor um operário que deve realizar a execução desta "obra". A ideia de utilização deste *curriculum* é prover uma documentação que defina os procedimentos a serem realizados durante o processo de "construção" do aluno pelo professor, sendo que o segundo é considerado o protagonista do processo como um todo [9].

Além disso, o paradigma conservador busca mensurar a taxa de aprendizado fazendo uso de avaliações periódicas dentro do período letivo, de forma que apenas os alunos que aprenderam determinado conteúdo possam progredir em seu nível de formação. A progressão é garantida para os alunos que tiverem um resultado maior que a média definida pelo *curriculum* e caso o resultado seja menor do que o mínimo esperado, o aluno tem a obrigação de repetir o periodo letivo para poder realizar todas as etapas do processo novamente [8].

No Brasil, o paradigma pedagógico mais dominante é o conservador, entretanto este paradigma possui diversas vertentes, entre elas, se destacam as seguintes:

Tradicional: É um padrão que vem sendo seguido há milhares de anos. Uma das principais abordagens nesse processo é considerar o aluno como ouvinte, de forma que apenas se torne receptor do conhecimento mas não seja ativo no sentido de questionar aquilo que lhe ensinam e o relacionamento com o professor é verticalizado. Há uma hierarquia e o aluno deve responder a ela, sendo o professor o principal detentor do conhecimento em sala de aula. Quanto à metodologia de ensino empregada nessa abordagem, o conteúdo é apresentado pelo instrutor que usa um livro texto como base das lições que são repassadas aos alunos. Os conteúdos são apresentados de maneira linear e o mais coerentes possíveis [6]. O instrutor desta metodologia de ensino geralmente possui maestria do assunto a ser lecionado [10]. Todo o conteúdo é definido no inicio dos períodos letivos, as atividades possuem uma data definida para as entregas e realizações, de modo que os estudantes possuam metas a serem cumpridas. O grau de aprendizado de cada aluno é definido com base nas notas obtidas em avaliações realizadas no periodo letivo. A prática das matérias exibidas em sala é feita com exercicios repetitivos para melhorar a memorização dos conteúdos apresentados. Essa abordagem não leva em consideração o desempenho individual de cada aluno, apenas pressupõe que todos possuam as mesmas condições para que o aprendizado ocorra. Essa abordagem utiliza um processo que possui quatro ações sendo elas: escutar, ler, decorar e repetir [6].

Escolanovista: Difundiu-se em meados de 1930 no Brasil em algumas instituições de ensino, esta abordagem faz com que o aluno busque o conhecimento, tornando ele um

sujeito ativo no processo da própria aprendizagem. A abordagem escolanovista também leva em consideração a facilidade do indivíduo no processo de aprendizagem e coloca o professor em uma posição de facilitador do processo, tornando o meio mais democrático. Nesta abordagem o indivíduo tem que buscar o auto-conhecimento, sendo livre para se desenvolver no seu próprio ritmo e aprender colocando em prática tudo que estudou. Apesar da abordagem enfatizar a liberdade do individuo e tornar o processo menos verticalizado a mesma foi categorizada como conservadora por favorecer o aspecto individual ao invés do social [6];

Tecnicista: Esta abordagem possui diversas semelhanças com o método tradicional pois em ambos os casos o aluno é tratado como um ser passivo no processo de aprendizagem. Ele passa a ser treinado e moldado pelo professor que passa a possuir um papel de "engenheiro comportamental". Esta abordagem traz também uma semelhança em relação a escolanovista pois visa aprender e fixar os conteúdos através da prática de exercícios. Entretanto o aluno é tratado como um produto do início ao fim do processo de aprendizagem, portanto a ideia é preparar o produto (aluno) para o mercado de trabalho que será onde ocorrerá a venda da sua mão de obra [6].

Devido a diversas mudanças que ocorreram nas esferas sociais e tecnológicas dos últimos anos, esse paradigma pedagógico passou a ser considerado por muitos como um modelo ultrapassado de aprendizagem. Além de tornar todo o processo muito mecanizado ele não leva em consideração as limitações do individuo e apenas busca gerar a maior quantidade de mão de obra possível para atender o mercado [11].

2.1.2 Paradigma da Complexidade

O paradigma tradicional entrou em decadência a partir do século XX quando a visão científica de causa e efeito passou a ser ineficaz devido a diversas contradições da época[12]. Isso ocasionou como resultado uma quebra de alguns paradigmas sociais e demarcou a mudança entre o mundo moderno e contemporâneo.

Uma das principais diferenças que o paradigma emergente ou complexo possui é a visão

do individuo como um ser íntegro e complexo. Os paradigmas tradicionais tratavam o universo e suas diversas áreas com visões fragmentadas dos fatos, entretanto, o paradigma da complexidade tem como diferencial apresentar o estudo do todo e mostrar que a soma das partes é maior que o todo [8].

A mudança de visão fragmentada para a visão do todo teve início com a Biologia que sofreu grande influência de Einstein com a Teoria da Relatividade e que proporcionou uma visão íntegra do universo, suas energias, processos de mudança e acaba por defender a totalidade do mesmo. Outro precursor deste paradigma foi a Psicologia de Gestalt que dá foco nas interconexões dos fenômenos, analisando um objeto em determinado contexto e leva em consideração a função dele em relação ao todo [8].

Esta evolução de paradigma não só mudou diversos aspectos sociais na época mas também causou impactos na educação, pois essa noção de que o individuo faz parte do todo implicou que o homem participa da própria construção do conhecimento, com suas experiências empíricas, emoções e intuições. A partir desse paradigma as metodologias inter, pluri e transdiciplinar foram difundidas.

No paradigma pedagógico complexo a fragmentação dos conteúdos pelo professor pode ocorrer contanto que a contextualização seja levada em consideração e as devidas conexões sejam apresentadas. Porém, fica a cargo do estudante realizar a análise das devidas relações entre os conteúdos dispostos em sala [13].

Assim como no paradigma tradicional, o emergente possui também diversas abordagens, com suas devidas vantagens e desvantagens. As três principais abordagens apresentadas nesta seção são:

Sistêmica: Nesta abordagem é considerada a complexidade do indivíduo (estudante) e a coletividade do mesmo no meio social em que está inserido, suas habilidades e caracteristicas únicas. O professor busca potencializar o aprendizado de cada um dos alunos com trabalhos e atividades para que desenvolvam um senso social de suas individualidades que por consequência levará ao crescimento gradativo conforme as suas habilidades e limitações. A abordagem sistêmica tenta equilibrar os conteúdos práticos e teóricos lecionados em sala de aula com o intuito de relacionar os mesmos afim de tornar ainda

mais estimulante o processo de aprendizado pela visão do aluno [14];

Progressista: É uma abordagem com maior foco no liberalismo educacional, dá condições ao aluno de participar da aula de forma democrática, acrescentando seu conhecimento, sua opinião ou críticas. Entretanto, o professor é o líder responsável por mediar as interações e articulador do processo. Essa metodologia tem como principal ferramenta o diálogo entre os sujeitos que discutem coletivamente os referênciais, questionam os conteúdos apresentados e desta forma acabam por desenvolver um senso crítico [14];

Ensino com pesquisa: O objetivo desta abordagem é desenvolver uma formação crítica com a interação entre aluno-professor e fazer uso de recursos tecnológicos para a realização de pesquisas[14]. O aluno recebe um papel mais ativo nessa abordagem pois ele tem que se tornar um questionador, investigador e de forma autônoma buscar a resposta que deseja seja para compreender ou questionar o processo. Dentro da sala o professor busca desenvolver no aluno uma responsabilidade ética e garantir o desenvolvimento de sua cidadania.

Uma das bases desse modelo pedagógico é garantir que o professor não seja somente o responsável por ensinar o conteúdo proposto, mas que também seja o sujeito encarregado de despertar o senso crítico e questionador nos estudantes, preparando-os para a vida [14].

2.1.3 Paradigma da Sala de Aula Invertida

O paradigma da sala de aula invertida trata-se de um modelo relativamente novo que vêm sendo implantado no Brasil nos últimos anos. Consiste em tornar o professor uma fonte de apoio ao ensino e torna o aluno precursor do próprio método de aprendizagem [14].

No Brasil e em diversos países esse paradigma é difundido em forma de cursos de Ensino à Distância (EAD), pois o aluno não precisa necessariamente estar em um ambiente escolar para poder assistir às aulas. Geralmente as aulas são disponibilizadas de maneira *online* para que os alunos possam assistir e revisar os conteúdos quantas vezes forem necessárias.

Este paradigma torna o aluno protagonista do seu processo de aprendizagem, pois

além do mesmo ser responsável por assistir às aulas ainda há tarefas e atividades que devem ser entregues conforme os prazos determinados. As plataformas utilizadas para auxiliar o aluno costumam ser interativas e propicíam muitas vezes algum tipo de contato com os professores, monitores responsáveis e fóruns para discussões entre os alunos. O paradigma da sala de aula invertida resolve alguns problemas para estudantes do ensino superior, pois este método dá total liberdade quanto à flexibilidade dos horários de aula e não necessita que o aluno more na cidade da instituição de ensino [15].

Além do ensino à distância, o paradigma invertido também pode ser utilizado quando o aluno passa a buscar o conhecimento com uso de recursos de vídeo ou livros antes da aula expositiva. Este paradigma é suportado por possuir ambientes que são flexíveis para o ensino, o professor não é o único detentor do conhecimento, assumindo papel de facilitador para que o mesmo seja alcançado. Os conteúdos devem possuir uma certa linearidade para garantir a fácil assimilação dos mesmos e para o professor fica a responsabilidade de conseguir lidar com os alunos dentro e fora da sala de aula sempre incentivando reflexões e possuindo uma metodologia motivacional para os alunos [14].

Apesar deste paradigma apresentar muitas vantagens, inicialmente existe um certo periodo de adaptação até que o aluno consiga se policiar e assistir as aulas e realizar tarefas dentro dos prazos. Este paradigma se assemelha muito com o emergente pois coloca o aluno em uma posição ativa no processo de aprendizagem, ou seja, para conseguir conhecimento dependerá quase que exclusivamente da sua motivação e empenho [15].

2.2 Metodologias de Ensino-Aprendizagem

Metodologia é definida como um conjunto de métodos ou práticas para alcançar um determinado objetivo em uma determinada área. Também pode ser considerada uma forma de determinar com métodos cientificos uma relação intelectual, ética e social de se realizar um determinado plano atendendo os critérios e de maneira eficaz [16].

As metodologias de ensino tem como objetivo buscar métodos que sejam eficientes e motivadores que servem como base para o ensino do aluno. É uma das muitas áreas

que pertencem a pedagogia e geralmente são desenvolvidas por pedagogos que buscam encontrar os melhores métodos e técnicas no contexto de ensino em que se encontram [11].

Com o passar dos anos diversas metodologias de ensino foram utilizadas e melhoradas com o objetivo de extrair o melhor desempenho possível. Existem diversas metodologias a serem utilizadas na educação, porém a utilização delas deve ser feita de forma que o contexto e ambiente sejam levados em consideração[11].

O desenvolvimento de cada metodologia deve levar em consideração as necessidades do estudante, bem como suas limitações e qualidades. As diferentes estratégias de ensino podem levar a resultados bons ou ruins, dependendo do perfil do estudante e as características do mesmo, que podem ser exibidas no decorrer do processo de aprendizagem. Isso ainda levando em consideração que a singularidade de cada individuo pode afetar o progresso no decorrer da aprendizagem, além de que o ambiente social que o mesmo se encontra pode influênciar de maneira direta [17].

A qualidade de um bom modelo pedagógico não depende somente das técnicas a serem utilizadas, mas sim do ambiente e da cultura do povo local, pois isso influencia diretamente no nivel de aprendizado do público alvo. Com a evolução tecnológica é possível observar uma tendência de mudanças em todas as áreas da sociedade, inclusive nas metodologias de ensino pois o fácil acesso a informações permite o desenvolvimento do conhecimento [6].

A globalização exige maior reflexão por parte do professor a respeito dos processos da sociedade, sendo eles, educacionais, econômicos, financeiros, políticos e sociais. Essa mudança exige que os docentes possuam uma mente aberta e flexibilidade nas metodologias para se adaptar as mudanças que estão ocorrendo [18].

2.2.1 Metodologias Tradicionais

Aprendizado baseado em sujeitos trata-se de um dos modelos mais tradicionais de ensino, pois faz uso de livros texto, materiais de apoio e instrutores. O conteúdo é apresentado pelo instrutor que usa um livro texto como base das lições que são repassadas aos alunos.

Os conteúdos são apresentados de maneira linear e o mais coerentes possíveis. O instrutor desta metodologia de ensino geralmente possui maestria do assunto a ser lecionado [10].

O professor que utiliza esta metodologia costuma apresentar os conteúdos utilizando geralmente como material de apoio, o quadro, slides ou livros. Durante a explicação ou apresentação dos conteúdos os alunos podem realizar perguntas pertinentes ao assunto [10].

Todo o conteúdo é definido no início dos periodos letivos, as atividades possuem uma data definida para as entregas e realizações, de modo que os estudantes possuam metas a serem cumpridas. O grau de aprendizado de cada aluno é definido com base nas notas obtidas em avaliações realizadas no periodo letivo.

Esta metodologia de ensino foi difundida durante o século XVIII em escolas públicas francesas a partir da difusão do iluminismo tendo como base a interação aluno-professor. Em muitas escolas e universidades brasileiras este método ainda é o mais utilizado, apesar de não ser o mais adequado atualmente [4].

2.2.2 Metodologia Ativa de Ensino

A partir do momento que o foco é a aprendizagem, o papel do professor em sala de aula deixa de ser o indivíduo que expõe o conteúdo, mas passa a ser o responsável por instigar o aluno a buscar o conhecimento [16].

Deve-se preparar os estudantes para atender ao mercado de trabalho, as competências profissionais atualmente vêm exigindo um senso crítico e perfil pró-ativo[16]. Para que estas qualidades possam ser despertadas no estudante é necessário que as práticas pedagógicas em sala de aula propiciem a discussões, pesquisas, superação de desafios e a resolução de problemas [16].

As metodologias ativas tem por objetivo principal colocar o aluno como responsável pelo desenvolvimento de seu próprio conhecimento, despertar o interesse e fazer com que o aluno encontre a determinação necessária para se desenvolver. A sensação de liberdade e controle do próprio aprendizado aumentam a curiosidade, motivação intrínseca,

criatividade, desempenho escolar e a satisfação [4].

Nas metodologias tradicionais de ensino as práticas de obediência e submissão são a base da aprendizagem, aquilo que o professor ensina é o correto e não se deve argumentar. Entretanto, na metodologia de ensino ativa o próprio aluno deve buscar entender o conteúdo, o professor serve apenas como um suporte no processo.

A metodologia ativa pode ser aplicada de diversas formas, os principais processos de metodologia ativa citados por [4] são:

Estudo de caso: É caracterizado por levantar um problema do mundo real e suas características de forma que o aluno deva buscar uma solução adequada com as informações que ele possui sobre o problema, definindo as técnicas e métodos que devam ser empregados para a solução;

O processo do incidente: É caracterizado por um trabalho em grupo, que segue um padrão parecido com o estudo de caso, porém os alunos recebem a informação do problema de forma resumida e com base nisso eles podem fazer perguntas para obter mais informações acerca do problema. Este processo tem foco no trabalho em equipe e na análise e questionamento dos problemas propostos. Ao final da avaliação do problema cada equipe deve apresentar os resultados encontrados para toda a sala;

O método de projetos: É caracterizado pela associação do processo de ensino, pesquisa e extensão. Normalmente esse processo é utilizado para desenvolver o conhecimento técnico acerca de um determinado assunto, analisar as propriedades e aproximar o máximo possivel os conteúdos de sala com aquilo que são na realidade;

A pesquisa científica: É caracterizado pelo desenvolvimento de trabalhos científicos como projetos de Iniciação Científica (IC), Trabalho de Conclusão de Curso (TCC) ou até mesmo como colaboradores em projetos dos professores. Esse foco no desenvolvimento de trabalhos científicos permite aprimorar o senso de elaboração de ideias do aluno e dá a capacidade de melhorar ainda mais suas habilidades intelectuais. Desenvolve também a observação, descrição, análise, argumentação e a síntese do aluno pelo fato de que essas são etapas cruciais para a elaboração de qualquer trabalho de cunho científico.

2.3 Sumário

Neste capítulo foram apresentados e descritos alguns dos principais paradigmas pedagógicos de ensino, uma breve explicação sobre as metodologias de ensino tradicional e ativa. No próximo capítulo será apresentado alguns dos conceitos primordiais acerca da Gamificação.

Capítulo 3

Gamificação

Neste capítulo estão descritas as definições de Gamificação segundo a visão de alguns autores e o que está envolvido no processo para Gamificar uma tarefa ou atividade.

Por definição a palavra Gamificação consiste no ato de inserir elementos de jogos dentro de uma determinada atividade afim de motivar as pessoas a atingirem determinados objetivos. Os elementos a serem inseridos podem ser desafios, recompensas, pontos e pode-se adicionar regras para definir as diretrizes da atividade. Um dos principais objetivos em gamificar uma determinada atividade é aumentar os níveis de engajamento para estimular a motivação de maneira intrínseca e extrínseca. A Gamificação pode fazer uso também de ambientes simulados e lúdicos com o intuito de tornar a atividade muito mais atrativa [19].

Jogos sempre tiveram um papel muito importante na vida das pessoas, a milhares de anos elas já buscavam formas de se divertir. Descobertas recentes sobre artefatos da idade antiga mostram que os humanos da época jogavam um tipo alternativo de gamão [20].

A área de jogos está em constante crescimento e muito vem sendo estudado a respeito dos benefícios que eles proporcionam aos jogadores. Com base nisso diversas empresas começaram a tentar gamificar suas atividades. As empresas buscam gamificar suas operações para que a motivação e o engajamento passem a se tornar parte do dia-a-dia, melhorando a produção e o trabalho em equipe de seus funcionários [20].

Segundo [20] diversas empresas de desenvolvimento de *software* têm tentado realizar a

Gamificação de algumas tarefas e atividades para melhorar o rendimento e a eficácia das equipes, principalmente quanto ao desenvolvimento das documentações dos projetos que demandam maior nível de detalhamento e tempo.

Vários processos de uma empresa podem ser gamificados com o objetivo de engajar, socializar, motivar, ensinar e fidelizar os colaboradores e clientes[21].

Diversos estudos acerca da Gamificação na educação vêm sido realizados para tornar o processo de ensino mais eficiente e motivante [22]. A motivação no ambiente escolar é um dos principais fatores que garante o resultado desejado, apesar de ser um desafio a ser alcançado. A utilização de elementos de jogos na educação proporcionam suporte ao aprendizado em diferentes atividades e contextos [1] [23].

Nas seções seguintes estarão descritos alguns dos elementos mais pertinentes em jogos, e que costumam ser os mais utilizados nos processos de Gamificação.

3.1 Tipos de Jogadores

Assim como existem jogos de diversas categorias, para os mais variados gostos, há também jogadores com diversos tipos de perfis. Nesta seção serão apresentados os tipos de jogadores e suas peculiaridades.

Com o passar do tempo alguns estudos a respeito dos principais perfis de jogadores foram sendo realizados. Esses estudos tiveram como base o comportamento dos jogadores no ambiente em que estavam inseridos, quais tipos de ações eles tomavam e como era a forma de interação dos mesmos com outros jogadores [21]. Com base nessas diretrizes, foram definidos quatro perfis distintos para os jogadores, cada qual com sua peculiaridade e características, abaixo seguem descritas:

Socializadores: Os jogadores desta categoria jogam com o intuito de socializar com outros jogadores, realizar trabalhos em equipe e fazer amizades. Esse perfil de jogador é encontrado facilmente em jogos que dependem da interação social: Massive Multiplayer Online Role Playing Game (MMORPG), Mahjong, Poker, entre outros [21]. O perfil desse tipo de jogadores não se preocupa necessariamente com ganhar ou perder, mas sim com

manter ou criar novas relações sociais durante o jogo;

Exploradores: Os jogadores desta categoria buscam extrair o máximo que o jogo pode proporcionar. Tentam encontrar fases secretas, completar todas as missões e tarefas dos jogos. No geral, eles tendem a focar na experiência que o jogo todo pode proporcionar. Esse tipo de jogador é caracterizado também por estudar o ambiente e a jogabilidade buscando o aperfeiçoando das suas ações no decorrer do jogo [19];

Conquistadores: Os jogadores desta categoria possuem como característica a busca constante pela vitória. [2] citam que é bastante arriscado fazer um *Game Design* focando apenas neste tipo de usuário, pois a motivação dele depende quase que exclusivamente da vitória, assim que o usuário perde ele deixa de ter interesse pelo jogo;

Predadores: Os jogadores desta categoria têm como principal motivador ganhar de seus adversários, porém ganhar não é o suficiente, a vitória segundo a percepção deste tipo de jogador deve ser "exibida" para os perdedores [2].

Na próxima seção deste trabalho serão apresentados alguns conceitos relativos ao *Flow* responsáveis por manter o engajamento e a atração do usuário pela atividade.

3.2 A Teoria do Flow

Nesta seção será descrito o que é a Teoria do *Flow*, como ela influencia no engajamento do indivíduo e os princípios que tornam essa teoria muito utilizada em diversas áreas, inclusive na Gamificação de atividades ou processos.

Inicialmente, os estudos iniciaram pela busca em tentar entender o que torna uma pessoa feliz, como esse processo acontece e também o que é necessário para manter uma pessoa neste estado [24]. Notou-se que, quando algumas pessoas realizavam uma tarefa na qual gostavam elas acabavam atingindo um estado de felicidade, não por serem recompensadas de alguma forma, mas por simplesmente realizarem uma tarefa na qual a recompensa é própria execução da mesma.

O conceito de *Flow*, consiste em realizar uma atividade com um foco tão grande que as coisas acontecendo ao redor do indivíduo deixam de ser importantes bem como a noção

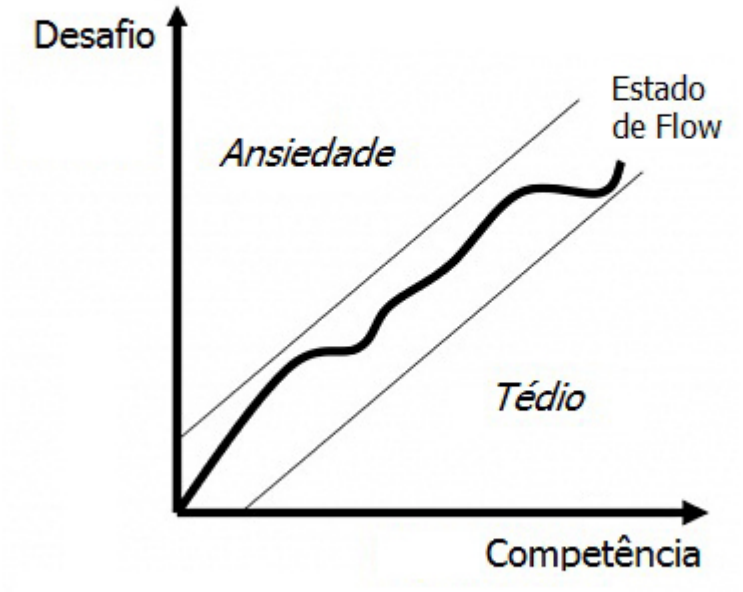

Figura 3.1: A teoria do Flow. Fonte: http://www.rtsconsultoria.com.br

do tempo, que se acaba por perder. Tudo isso devido à sensação de prazer e felicidade envolvidas no processo [25]. Conforme observado na Figura 3.1, o estado de *Flow* acaba por ficar entre a ansiedade e o tédio, de forma que o nível de desafio exija um determinado nível de habilidade para garantir que o indivíduo não fique ansioso ou entediado.

O termo *Flow* passou a ser muito utilizado como um termo técnico para a motivação intrínseca. Este termo foi proveniente da descrição das pessoas que atingiam o estado de felicidade e excitação, que descreviam o processo como um tipo de "fluxo" [25].

A teoria do *Flow* está atrelada diretamente à dificuldade imposta ao jogador perante os desafios que ele deverá enfrentar [26]. Conforme apresentado na Figura 3.1, quanto maior a variação da linha na região demarcada maior é a oscilação da dificuldade proporcionada ao jogador. Há diversas discussões sobre o que é melhor, definir um fluxo constante ou com variações no decorrer do jogo, segundo o ponto de vista do autor um fluxo que varia

tende a manter o jogador mais engajado [26]. O aumento de dificuldade seguido por uma recompensa que torna as coisas um pouco mais fáceis acaba por se tornar um ciclo repetitivo até o fim do jogo na maioria dos casos.

Muitos *Game Designers* levam em consideração a teoria do *Flow*, porém ao alterar os níveis de dificuldade e definir as recompensas os impactos disso devem ser analisados e balanceados para evitar o descontentamento do jogador.

A próxima seção trata de detalhar e definir o que são alguns dos Objetivos Extrínsecos presentes em jogos e atividades Gamificadas.

3.3 Objetivos Extrínsecos

Os objetivos Extrínsecos presentes nos jogos tratam de fazer com que o jogador aprenda a realizar uma determinada função ou atividade sem que ele se dê conta de que as ações realizadas durante o jogo lhe deram esse conhecimento.

Um jogo em que o objetivo do jogador é auxiliar o personagem a realizar a separação de uma maçã entre os outros personagens [19]. Ao conseguir concluir o jogo de maneira correta o jogador aprendeu de forma extrínseca a realizar operações utilizando frações.

Diversos objetivos Extrínsecos podem ser definidos em um jogo, os objetivos devem ser balanceados para evitar a sobrecarga do jogador [4]. Caso haja objetivos extrínsecos de maneira exagerada em um contexto, o jogador pode passar a se sentir manipulado e isso acaba por sabotar a motivação e engajamento que o jogo deveria prover.

3.4 Motivação Intrínseca e Extrínseca

Quando se trata de jogos, um dos fatores que prendem um jogador é a motivação. A motivação gerada a partir dos jogos é separada em duas categorias: intrínseca e extrínseca [27].

A motivação intrínseca é definida como o envolvimento do sujeito com o jogo pelo desejo próprio, pelo jogo ser de alguma forma desafiador, prazeroso ou intrigante. Está

diretamente relacionado com as emoções que o jogo é capaz de despertar na pessoa e por possuir uma ligação tão próxima com o jogador este é considerado o principal motivador quando o foco é "prender" a atenção e garantir o engajamento [21] .

Motivadores extrínsecos em jogos costumam ser algum tipo de compensação dentro do jogo por realizar determinado objetivo. Os motivadores extrínsecos mais comuns utilizados em jogos são os pontos, recompensas, medalhas ou posição em um *ranking* [27]. Geralmente os *Game Designers* acabam por colocar muitos motivadores extrínsecos nos jogos quando na verdade deve haver um equilíbrio entre os motivadores intrínsecos e extrínsecos. Quando a motivação extrínseca é saturada o indivíduo passa a se sentir manipulado no processo e aquilo que deveria ser um incentivo à realização de determinada atividade passa a tornar-se um motivo para não a realizar [4] .

Muitos confundem o ato de Gamificar uma atividade com a simples inserção de motivadores extrínsecos sem analisar o contexto da atividade. Não basta adicionar pontos e um placar esperando que o desempenho de um funcionário ou aluno melhore. O objetivo a ser alcançado deve ser estudado para saber que as emoções são necessárias para alcançar a motivação intrínseca de forma a garantir o engajamento pessoal na atividade gamificada [27].

3.5 Octalysis Framework

Esta seção tem como foco apresentar e detalhar as características do *Framework* Octalysis que será utilizado na etapa de *Game Design* do presente trabalho.

O Octalysis é um framework criado para auxíliar as pessoas na compreensão das características e impactos causados na implementação de certos atributos envolvidos no processo de Gamificação [27].

Na Figura 3.2 é possível observar que o *Framework* é representado por um octógono, onde cada vértice do mesmo descreve um atributo que deve ser levado em consideração ao desenvolver um projeto Gamificado.

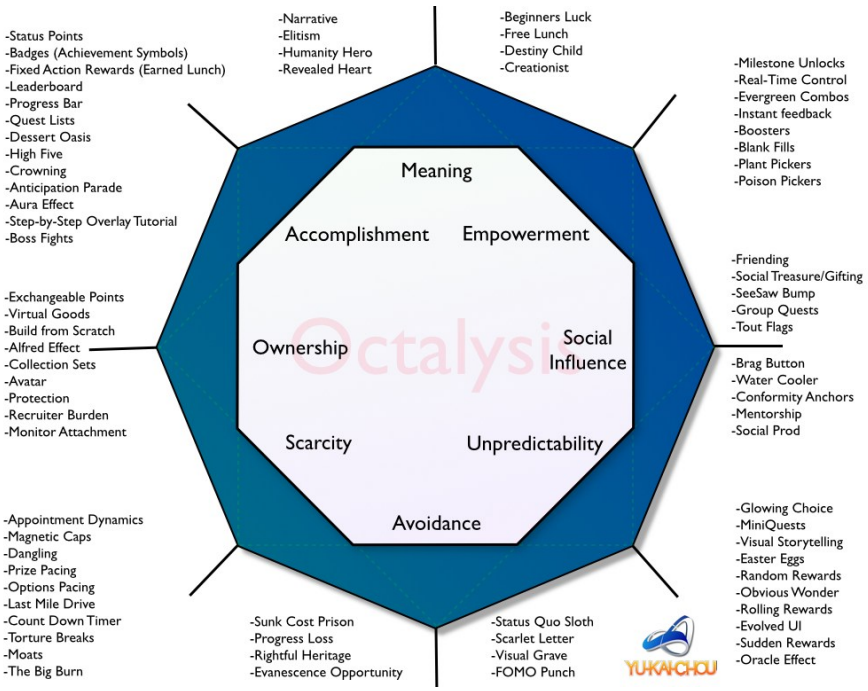

Figura 3.2: Octalysis Framework. Fonte: http://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/

Os oito núcleos do processo de Gamificação estão diretamente ligados com as motivações dos jogadores [27]. O criador da *Framework* enfatiza que toda ação dentro do jogo deve possuir características dos oito núcleos (Figura 3.2), caso contrário essa ação não irá gerar o resultado esperado pelo *Game Designer*.

Nas subseções seguintes estarão citadas e especificadas cada um dos oito núcleos para descrever os motivadores intrínsecos e extrínsecos envolvidos no processo de Gamificação [27].

3.5.1 O Significado

Neste núcleo do *Framework* o objetivo é tornar o jogador um personagem imprescindível na narrativa do jogo, fazer com que o mesmo se sinta o "escolhido" para realizar determinada tarefa.

A sensação de fazer parte da narrativa do jogo garante que o jogador se empenhe para buscar o desfecho da narrativa e alcançar seus objetivos [27]. Garante por exemplo,

que o jogador busque realizar seus objetivos para auxiliar a equipe ao invés de beneficiar somente a si mesmo.

3.5.2 Realizações

O grupo de realizações na Gamificação é formado por tudo aquilo que o indivíduo é capaz de obter jogando, sejam elas: missões, tarefas, pontuação, conquistas, medalhas, troféus e outros tipos de recompensas pelas ações ou tarefas realizadas.

Pontos são definidos como sendo um dos principais motivadores extrínsecos, fazendo parte da Gamificação *White Hat* (apresentada em detalhes na subseção 3.5.10) e é considerado uma das bases de realizações na Octalysis [27].

Para que um processo de Gamificação seja motivador é necessário recompensar o usuário pelas ações realizadas de forma que ele continue buscando concluir mais objetivos ao longo do tempo [2] . Também faz parte desta categoria prover atividades e missões desafiadoras e intrigantes para engajar o indivíduo e fazer com que ele foque em seu objetivo almejando algum tipo de recompensa.

Este núcleo é considerado como uma das etapas mais fáceis de serem implementadas em uma atividade gamificada, porém deve receber atenção especial para que as recompensas não sejam muito fáceis e ao mesmo tempo não devem ser impossíveis de serem alcançadas [2] [27] [19].

3.5.3 Empoderamento

É definido no *Framework* Octalysis pelo senso de controle que o jogador possui sobre suas próprias ações. Por exemplo, um Role-Playing Game (RPG) possui como diferencial que as ações a serem realizadas dependem do jogador. O rumo que seu personagem segue é determinado por aquilo que o jogador deseja realizar. Toda ação realizada pode ou não levar a um mesmo fim, no caso de um RPG as ações levam a consequências que podem alterar o final de uma missão ou tarefa. Esse senso de controle sobre o jogo aumenta a motivação intrínseca do jogador.

Este núcleo contempla também as atividades que retornam o *feedback* instântaneo ao jogador. Características do jogo que permitam ao jogador realizar certas ações de diversas formas diferentes. Uma analogia pode ser feita atrelando este núcleo também a criatividade, o Lego por exemplo, permite o empoderamento de seus jogadores, pois quem define o que será feito a partir das peças é o jogador e portanto o *Game Designer* não precisa adicionar muito conteúdo ao jogo quando na verdade a dinâmica do mesmo garante que apenas a criatividade seja o limitante dos resultados gerados [27].

3.5.4 Senso de Propriedade

Neste núcleo ficam as características que envolvem o senso de propriedade com aquilo que o jogador adquiriu ou conquistou no decorrer do jogo. As personalizações de avatar, compras ou *drop* de equipamentos fazem surgir um certo vínculo entre o jogo e o jogador [27].

A sensação de possuir algo dentro do jogo como se fosse um bem material garante uma sensação de controle sobre o mesmo e por consequência, o jogador acaba buscando melhorar seu item, personagem ou status dentro do jogo. Quanto mais tempo é gasto para melhorar um personagem ou processo, maior será o sentimento de posse e por consequência o jogador sentirá maior vínculo com o jogo [2].

3.5.5 Influência Social

Este núcleo contempla todos os atributos sociais do jogo, que vão desde a comunicação, o trabalho em equipe e até a competição entre jogadores. Um dos principais motivadores sociais é a competição entre os jogadores, seja com base em um *ranking* ou até mesmo para a realização de determinadas atividades, trata-se de um forte motivador intrínseco [24] [28] [4] [22] [19] [27].

A ajuda a outros jogadores também faz parte da esfera de Influência Social e acaba por fortalecer os laços interpessoais e melhora os trabalhos em equipe.

Até mesmo quando um jogador obtém alguma habilidade, item, derrota um boss,

realiza uma tarefa considerada difícil ou alcança determinada posição em um *ranking*, em um cenário competitivo isso acaba se tornando motivo para que outros jogadores se empenhem com o objetivo de superar seus adversários.

3.5.6 Escassez

A escassez é definida no *Framework* Octalysis como um atributo que faz os jogadores buscarem tudo aquilo que for raro e único dentro do jogo, seja um item ou uma missão secreta. Pelo fato de não ser fácil conseguir concluir um objetivo ou determinado item isso acaba gerando um senso de competição entre os jogadores, seja por ser o único a conseguir determinado objetivo ou com o intuito de superar outros jogadores que já conseguiram realizar a tarefa [27].

Esse núcelo também controla, por exemplo, o quanto um usuário possui de liberdade dentro do jogo, não poder avançar para uma determinada região do mapa sem completar certa missão, ou ter de esperar um certo tempo até que possa concluir uma tarefa [27].

3.5.7 Imprevisibilidade

Neste núcleo da Octalysis *Framework* é levado em consideração todos os elementos imprevisíveis dentro do jogo, que tornam ele o mais dinâmico possível [27].

Elementos de imprevisibilidade, quando se trata das recompensas, são imprescindíveis para uma dedicação maior ao realizar as tarefas ou missões propostas [21]. Isso garante uma maior relevância das atividades no ponto de vista do jogador, de forma que uma mesma missão possa dar recompensas distintas para dois jogadores diferentes, dependendo simplesmente do acaso.

A imprevisibilidade vai muito além de acrescentar elementos randômicos atrelados à missões, tarefas e conquistas [27]. A imprevisibilidade deve ser considerada um elemento necessário, sempre que possível deve ser utilizado junto às narrativas com o intuito de manter o jogador motivado e engajado [27] [19].

3.5.8 Perda e Evitação

O sentimento de perda para o jogador é uma das piores sensações. Elas podem ser perda de tempo, perda de itens, perda das conquistas, derrota em uma batalha ou perca de progresso no geral, acabam por incentivar uma motivação mais intensa afim de evitar a perca [27].

A sensação de que ao perder uma missão fará com que o jogador perca o tempo e empenho investidos em seu personagem até o momento atual faz com que o engajamento atinja um de seus ápices afim de alcançar o objetivo.

Objetivos que possuam determinados prazos à serem cumpridos também são capazes de gerar a motivação para evitar a perda de coisas que foram realizadas, ou que seriam obtidas.

3.5.9 Left Brain vs Right Brain

Os núcleos do *Framework* Octalysis são separados pelos chamados motivadores Intrínsecos (*Right Brain*) e Extrínsecos (*Left Brain*) [27]. Ambos elementos motivadores foram discutidos na subseção 3.4.

Na Figura 3.3 pode-se notar que os núcleos pertencentes a categoria *Left Brain* são aqueles considerados motivadores Extrínsecos, o jogador se sente motivado pois quer obter algo, independente de ser objetivo ou um item por exemplo.

Entretanto, analisando o *Right Brain* logo pode-se notar que os núcleos como Empoderamento, Influência Social e Imprevisibilidade estão ligados diretamente com a motivação Intrínseca. Pois a utilização de sentimentos como poder, criatividade, surpresa e a socialização com outros jogadores é de certa forma recompensador por si só [27].

Para atividades que exijam um maior tempo de dedicação do usuário, [27] recomenda que o *Game Design* seja focado nos motivadores intrínsecos, pois os extrínsecos acabam por prejudicar de maneira direta o rendimento geral da atividade.

O foco em motivadores Intrínsecos garante que a atividade seja mais divertida e engajadora, permitindo que os usuários demorem muito mais tempo para perderem a motivação

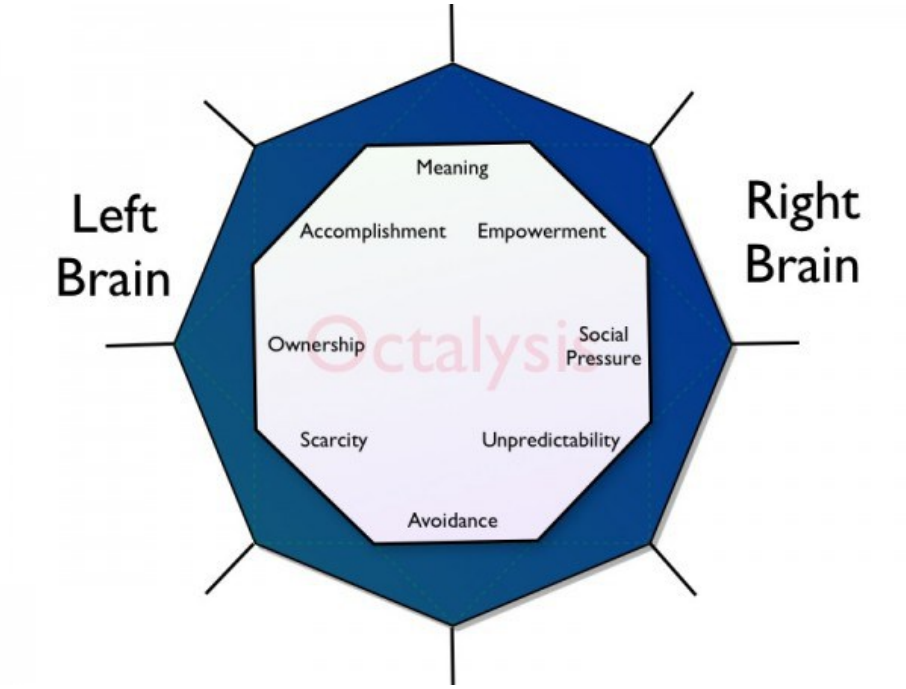

Figura 3.3: Left Brain vs Right Brain. Fonte: http://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/

pelo jogo.

3.5.10 White Hat vs Black Hat Gamification

Além das divisões realizadas de maneira vertical no modelo do *Framework* Octalysis, foram realizados também separações horizontais entre os oito núcleos do octógono, separados como *Black Hat Gamification* e *White Hat Gamification* com o intuito de separar as motivações extrínsecas e intrínsecas [27].

A *White Hat Gamification* contempla todas as motivações consideradas positivas para o jogador e envolve os núcleos do octógono superiores que permitem ao jogador expressar suas habilidades e que de alguma maneira são capazes de proporcionar um propósito à pessoa que está jogando.

Enquanto que jogos focados em *Black Hat Gamification* utilizam motivações negativas para que os jogadores cumpram seus objetivos, as atividades que se baseiam em evitar a perca e a imprevisibilidade garantem que o jogador não saiba o que está por vir. Neste

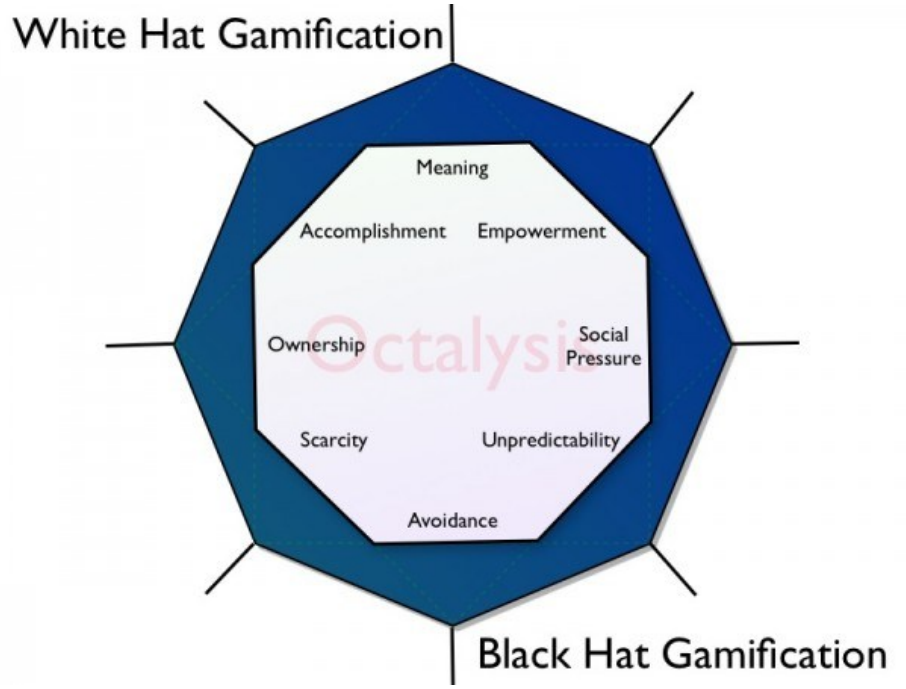

Figura 3.4: White Hat vs Black Hat. Fonte: http://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/

caso, o jogador passa a ser controlado pelo jogo, de forma que ele passe a lutar para conseguir coisas que são realmente difíceis e sempre que o mesmo tente desistir uma sensação de vazio e derrota é capaz de o deixar abatido [27].

Outros autores também citam algumas das vantagens e desvantagens de se usar motivadores positivos (*White Hat*) e negativos (*Black Hat*) [2] [19]. Deve ser enfatizado que apesar de os motivadores serem negativos, sua utilização não precisa necessariamente estar atrelada à algo ruim, existem diversos exemplos de motivadores negativos utilizados em processos de Gamificação para auxiliar em dietas ou acabar com certos tipos de dependências.

Em outras palavras, quando se busca um engajamento de curto prazo a melhor opção de Gamificação é focar nos elementos Black Hat e seus motivadores extrínsecos. Caso o objetivo seja visar uma motivação de longo prazo e fazer com que as pessoas se sintam bem ao executarem determinada tarefa, deve-se aplicar elementos White Hat à atividade Gamificada [27].

3.5.11 Sensação

Até o momento, Gamificação foi somente associada aos fatores psicológicos que a envolvem. Entretanto, o sucesso de uma atividade Gamificada depende também muitas vezes da sensação física que os jogos passam aos sentidos. Apesar de [27] não incluir a sensação como parte do Octalysis *Framework* por ele se encarregar apenas dos fenômenos psicológicos que envolvam a Gamificação.

Tanto os fatores psicológicos como os físicos precisam estar balanceados nas atividades à serem desenvolvidas, pois sem o *feedback* físico por mais que haja motivadores psicológicos suficientes, caso as percepções captadas pelos nossos sentidos não seja agradável ou responsiva a perca de interesse se torna iminente. O mesmo acontece caso o jogo utilize apenas fatores físicos, sem os motivadores psicológicos em pouco tempo o interesse pela atividade é perdido [27].

3.6 Sumário

Com base nas informações dispostas neste capítulo, foi possível entender o funcionamento mais detalhado da *Framework Octalysis*, os tipos de jogadores, os motivadores que envolvem um processo de gamificação e uma breve expicação sobre a teoria do *Flow*. Na tabela 3.1 encontra-se um detalhamento mais resumido sobre as oito bases da Framework apresentada.

Tabela 3.1: Tipos de motivadores da Octalysis *Framework*

Atributo	**Tipo**	**Tempo**
Significado	White Hat	Longo Prazo
Realizações	White Hat	Médio-Longo Prazo
Empoderamento	White Hat	Médio-Longo Prazo
Senso de Propriedade	White/Black Hat	Médio Prazo
Pressão Social	White/Black Hat	Médio Prazo
Escassez	Black Hat	Curto Prazo
Imprevisibilidade	Black Hat	Curto Prazo
Perda e Evitação	Black Hat	Curto Prazo

Como pode ser observado, os elementos da Framework que utilizam motivadores intrínsecos (*White-Hat*) tendem a atuar a longo prazo, enquanto que os motivadores extrínsecos (*Black-Hat*) agem a curto prazo mas podem fazer com que os usuários abandonem o jogo ou atividade assim que perderem o interesse.

No capítulo seguinte será apresentado e detalhado algumas das principais etapas do processo de Game Design segundo a visão de alguns dos principais autores da área.

Capítulo 4

Processo de Game Design

Neste capítulo será definido alguns dos principais aspectos quando se trata de *Game Design*, e o que os *Designers* levam em consideração ao desenvolver um jogo.

Game Design pode ser definido como o ato de decidir como o jogo será [26]. Porém, a função de um *Game Designer* é analisar e definir como será cada aspecto do jogo, sua história, mecânicas e a jogabilidade.

Os jogos costumam ser criados e desenvolvidos em equipes, que trabalham em conjunto para desenvolver conforme a visão e objetivos definidos pelos *Game Designers*. Apesar do jogo se tornar um bem de consumo para o jogador, devido ao processo que envolve o jogo é impossível prever como será a utilização e aceitação do mesmo [29]. Diferente de um filme que possui uma história linear e o público alvo pode somente observar enquanto o mesmo acontece, no jogo as pessoas se tornam o personagem principal e ao permitir uma certa liberdade ao mesmo torna as reações quase que imprevisíveis. Cada jogador pode ter uma perspectiva diferente quanto ao jogo e cabe ao *Game Designer* alinhar isso afim de proporcionar uma experiência inesquecível [29].

Todo *Game Designer* trabalha com o foco em 3 pilares, mecânicas, dinâmicas e estética [26] . Esses pilares serão descritos nas seções seguintes.

4.1 Mecânicas e Dinâmicas

As mecânicas tratam-se dos conjuntos de regras e condições para alcançar o objetivo desejado. Também são responsáveis por definir como será o tipo de jogo, a visão do mesmo, como será a jogabilidade de uma maneira geral [26]. A mecânica de jogo é definida como uma etapa que consiste de sete elementos principais responsáveis por determinar como será o decorrer do jogo, sendo elas: missões/desafios, *ranking*, medalhas/troféus, níveis, pontos, ambientação/integração, ciclos de engajamento.

As dinâmicas consistem em como ocorrerá a interação dos jogadores com a mecânica do jogo, como serão os controles do mesmo sobre o jogo, quais as formas de feedback do usuário.

Nas próximas subseções estará descrito algumas características imprescindíveis das mecânicas e dinâmicas de jogos.

4.1.1 Missões e Desafios

As missões e desafios que envolvem as mecânicas do jogo podem ou não estar diretamente atreladas ao roteiro da estória produzida. O objetivo de utilizar missões é trazer o jogador ao centro da história e fazer com que ele se sinta uma peça importante para contribuir com o todo, além de recompensá-lo pelos resultados na conclusão das missões e puni-lo caso não consiga cumprir as tarefas em tempo hábil ou conforme o esperado [26].

Os desafios e missões criados dentro dos jogos devem ser estudados quanto ao seu nível de dificuldade pois caso o jogador tente executar uma tarefa que seja praticamente impossível repetitivamente pode fazer com que o jogador saia do estado de *Flow* e se sinta frustrado.

4.1.2 Ranking

O sistema de *ranking* ou *leaderboard* é considerado uma mecânica de jogo utilizado para estimular a competitividade dentro do ambiente. O seu objetivo é expor certos tipos de pontos dos jogadores de maneira decrescente.

Um sistema de *ranking* garante que o jogador queira alcançar o topo das colocações e se manter como um dos líderes nessa determinada categoria de pontuação [2]. Simplesmente pelo fato do jogador poder ver a pontuação daqueles que estão abaixo dele já serve como um motivador para evitar que seja ultrapassado e procure ultrapassar aqueles que estão acima dele.

Os tipos mais comuns de *ranking* são os de níveis, experiência, pontuação por fase ou dinheiro. Além disso existem algumas formas de dispor esses dados aos jogadores. O *Game Designer* pode optar por exibir apenas um determinado número de jogadores de cada categoria (*leaderboard*) ou exibir todos os jogadores e suas determinadas pontuações (*ranking*).

4.1.3 Medalhas e Troféus

Quando se trata de recompensas, é muito importante definir o que o jogador pode ou não receber, desde itens, poderes, alguma conquista especial ou troféu. O importante nesta categoria das mecânicas é nunca compensar demais o jogador ao ponto em que obter as recompensas seja muito fácil ou que seja impossível [2].

Uma grande característica das medalhas e troféus é o fator expositivo que elas trazem consigo, ao tornar as medalhas e troféus de um jogador visíveis de alguma forma para outros jogadores isso acaba por gerar um certo nível de competitividade.

4.1.4 Níveis

Os níveis dentro de um jogo determinam o progresso que o jogador teve durante sua experiência, podem caracterizar o quão forte é seu personagem e servir como um meio de gerar competitividade entre os jogadores [2].

O aumento de nível nos jogos costuma ser não linear pois desta forma há o aumento gradativo da dificuldade e com isso os demais elementos do jogo devem acompanhar o crescimento do jogador ao longo do tempo. Na maioria dos jogos quando se sobe de nível o poder ou força do jogador tende a aumentar e por consequência os elementos do jogo

devem adaptar-se afim de manter o jogador dentro do estado de *Flow* [2].

Níveis costumam ser apresentados ao jogador de maneira numérica ou através de barras de progressão, deixando explícito o progresso do jogador.

4.1.5 Ambientação

Tratando-se da ambientação, o *Game Designer* deve planejar os elementos do jogo para que o jogador se sinta realmente como o personagem principal e que, para alcançar o desfecho da história, os personagens ou ambiente dependam das ações tomadas pelo jogador [21].

Para o desenvolvimento de uma boa ambientação o enredo deve ser levado em consideração. Estudos acerca do mesmo devem ser feitos para que o ambiente consiga atrair o jogador. A ambientação define o tempo que o jogador ficará engajado ao jogo, o seu comprometimento com o mesmo e também o propósito que o *Game Designer* deseja passar ao jogador. Por isso, ao realizar a Gamificação esse elemento é muito importante e deve ser levado em consideração [21].

4.1.6 Engajamento

Os níveis engajamento irão depender de muitas circunstâncias dentro do jogo, elementos que costumam influenciar diretamente nisso é a dificuldade, as mecânicas de jogo, os efeitos visuais e sonoros utilizados.

Desenvolver o jogo pensando em como o jogador irá se sentir e reagir a determinados elementos ou eventos é uma das tarefas mais difíceis do *Game Designer*. Além disso a jogabilidade e suas mecânicas são fatores importantes quando se trata de alcançar um estado de engajamento na maior parte dos jogadores [2].

Existem diferentes abordagens para garantir o engajamento dos jogadores e também ferramentas como a *Octalysis Framework* que definem os tipos de elementos e características de cada categoria. Apesar de existirem diversos tipos de abordagem, cada uma conta com suas vantagens e desvantagens que ao longo do tempo podem definir o engajamento

ou não dependendo do perfil do jogador [27].

4.1.7 Pontos

Segundo [2] existem diversas formas de se utilizar a mecânica das pontuações dentro de uma atividade Gamificada, as mais comuns são:

Pontos de Experiência: São responsáveis por definir o *rank* ou nível de um jogador, para que ele conheça sua evolução ou progresso ao longo do jogo. Tudo que um jogador fizer dentro do jogo costuma dar à ele pontos de experiência, sendo recompensado por todo objetivo ou tarefa concluída. Existem jogos os quais periodicamente resetam a quantidade de experiência dos seus jogadores com o objetivo de aumentar a competitividade;

Pontos de Habilidade: São utilizados para desempenhar determinadas atividades dentro do jogo. Em alguns jogos pontos de habilidade são obtidos ao realizar atividades que exijam esses pontos, sendo assim, uma forma de "treinar" o personagem. No entanto, existem outros que ao subir de nível atingindo a experiência necessária o jogador recebe uma determinada quantidade de pontos que poderão ser distribuídos na Habilidade que o jogador desejar;

Pontos de Karma: São utilizados normalmente em jogos clássicos com o intuito de recompensar o jogador por suas boas ações, que vão desde o acesso frequente ao jogo a até uma boa conduta social (no caso de jogos *online*). Estes pontos podem ser distribuídos de um jogador para outro como uma forma de honrar o outro jogador;

Pontos de Reputação: São utilizados em praticamente todos os jogos que envolvam algum tipo de atividade social. É um dos sistemas de pontuação mais complexos do ponto de vista de um *Game Designer* pois o mesmo deve prover mecanismos de incentivo para a obtenção desta reputação e métodos de punição ou desencorajamento das atitudes que geram uma má reputação;

Pontos resgatáveis: São responsáveis por garantir um certo comércio virtual dentro do jogo, com esses pontos é possível realizar trocas por itens, bônus e outros tipos de vantagens dentro do jogo. Por gerar um tipo de comércio pode ser que esses pontos

venham a sofrer inflações ou desvalorizações conforme a oferta e procura dos jogadores [2].

A mecânica de um jogo é uma das áreas mais sensíveis quando se trata de *Game Design* pois as demais etapas serão construídas com base nela. Portanto, sem uma mecânica bem definida o processo de desenvolvimento como um todo acaba se tornando comprometido.

A próxima seção deste capítulo trata diretamente dos elementos visuais e sonoros, sua importância no processo de Game Design em conjunto com as categorias apresentadas até o momento.

4.2 Estética

A estética consiste nas sensações e emoções que o jogo é capaz de proporcionar ao jogador através das interações entre mecânicas e dinâmicas [2].

Na categoria de *Game Design* quando se trata de estética, refere-se a tudo aquilo que é capaz de despertar emoções através dos sentidos. Geralmente os autores atribuem a essa categoria a criação de belos gráficos, utilizando tecnologias de última geração capazes de envolver os jogadores. Efeitos sonoros criados com a intenção de ambientar o jogador, também fazem parte desta categoria pois ela é responsável por acionar certas emoções quando necessário, seja de alerta, felicidade ou tristeza.

No próximo capítulo deste trabalho estará descrito os materiais e métodos que foram necessários para desenvolver a realização do mesmo.

4.3 Sumário

Neste capítulo foram discutidos algumas definições e aspectos que compoem a área de Game Design. No próximo capítulo serão descritos os materiais e métodos utilizados para o desenvolvimento do Cabinet.

Capítulo 5

Tecnologias, Materiais e Métodos

O foco deste capítulo será a especificação dos materiais necessários e os métodos a serem utilizados no decorrer do desenvolvimento do presente trabalho.

5.1 Materiais

Nesta seção e suas subseções estarão descritos os materiais que serão utilizados para o desenvolvimento. De uma maneira geral, serão explicados os diferenciais de cada ferramenta evidenciando a escolha de cada uma delas.

5.1.1 Unreal Engine

Toda a base e desenvolvimento da aplicação tratada neste trabalho será feita utilizando a Unreal Engine (UE)[1], que se trata de um motor gráfico multiplataforma e possui ferramentas para auxiliar no desenvolvimento de jogos, cinemáticas e simuladores.

A UE possui um motor físico capaz de realizar os mais variados cálculos como: colisões, atritos, gravidade, acelerações, entre outros.

Além disso ainda conta com um editor de materiais embutido, capaz de controlar diversos aspectos, como reflexão, mapas normais, lightmaps e também a utilização de

[1] www.unrealengine.com

shaders específicos para a definição de superfícies.

Uma das principais vantagens que a UE proporciona é a possiblidade de portar o jogo para as principais plataformas disponíveis atualmente, com desempenho bem favorável e sem que seja necessário a modificação e adaptação da estrutura do jogo.

A engine proporciona também aos seus usuários um *Framework* para o desenvolvimento de jogos *multiplayer*, inclusive com o suporte nativo a algumas plataformas como a Steam[2]. Isso permite aos desenvolvedores a economia de tempo na etapa de desenvolvimento. Por padrão a UE também já vem preparada com mecanismos para a utilização de servidores dedicados, bastando que toda a estrutura do jogo esteja bem definida e seus Remote Procedure Call (RPC) estejam seguindo o padrão estabelecido, a sincronização de eventos e atores ocorre de maneira transparente para o usuário final.

Outro grande diferencial ao utilizar a UE é a versatilidade no quesito programação, pois a mesma permite a construção dos códigos na linguagem C++ que é uma linguagem consolidada e robusta, muito conhecida mundialmente devido a empregabilidade da mesma em diversos setores e também há uma opção alternativa ao código mas que também pode ser utilizada para a programação que é através da utilização de blueprints, que são a utilização de métodos e funções de forma visual e extremamente focada na praticidade do usuário e pessoas que não estão habituadas a programação.

5.1.2 Linguagem C++

A linguagem C++ começou a ser desenvolvida inicialmente em 1979 com o nome de *C with Classes* e tinha como objetivo seguir uma sintaxe o mais parecida possível com C, onde a diferença principal seria em relação a adoção do paradigma de programação orientado a objetos, que estava começando a ser visto como tendência na época.

C with classes foi concebido inicialmente com a intenção de ser uma biblioteca modular para C, capaz de entregar todas as vantagens do paradigma orientado a objetos. Ao longo do seu desenvolvimento o criador notou que uma biblioteca não seria o suficiente para

[2]http://store.steampowered.com/

englobar todos os aspectos que o novo paradigma traria a linguagem. Foi a partir disso, que então se sucedeu o desenvolvimento da nova linguagem com uma gramática que segue os mesmos princípios da linguagem C.

Assim como a linguagem C, a linguagem C++ é fortemente tipada. Possui sobrecarga de operadores e funções, necessita de um compilador próprio. No quesito orientação a objetos, a linguagem é muito completa e permite utilizar todas as principais características do paradigma. Isso inclui herança múltipla, polimorfismo estático, polimorfismo dinâmico e encapsulamento.

Um diferencial muito importante que a linguagem C++ trouxe consigo foi o tratamento de exceções, que permite ao desenvolvedor recuperar ou lidar com a execução de seu programa caso o estado de execução não ocorra como o esperado.

5.1.3 Blueprint

A linguagem visual blueprint é considerada pela sua desenvolvedora como uma linguagem de *scripting* visual. Essa linguagem possui algumas peculiaridades em relação a outras linguagens visuais, sendo executada dentro de uma máquina virtual que atua sobre a Unreal Engine e traduzidas para linguagem C++ conforme o fluxo e a disposição dos nós dentro da blueprint.

Apesar da linguagem blueprint não ser uma linguagem compilada ela dispõe de algumas vantagens em relação ao C++ puro, pois a máquina virtual responsável pela sua execução é capaz de interpretar o código e dar um feedback instantâneo ao programador. Isso permite que os dados de entrada em uma função sejam os esperados conforme definidos no corpo da função e, além disso, evita erros básicos que tornam o desenvolvimento geral da aplicação mais complicado. Todos os tipos de dados dentro do editor possuem uma cor específica em seus nós, facilitando a identificação dos *inputs* e *outputs* [30].

Todo arquivo em blueprint possui a extensão “uasset” que é um formato proprietário da empresa Epic Games (desenvolvedora da Unreal Engine). Além disso, cada arquivo dessa extensão é considerada uma classe para a linguagem blueprint, no momento da

criação de uma blueprint é necessário escolher qual é a classe base. Com base nisso, a Unreal Engine determina a estrutura da nova classe. Toda classe inicia com ao menos duas funções, um construtor que pode ser do tipo “Event BeginPlay” ou “Event OnConstruct” e uma função do tipo timer “Event OnTick”. Conforme pode ser observado na Figura 5.1.

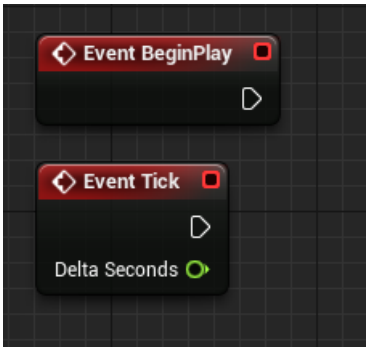

Figura 5.1: Funções base das classes feitas em blueprint. Fonte: autoria própria.

Assim como em outras linguagens, as instruções dentro dos construtores ocorrem apenas no momento de instanciação da classe. Em contrapartida a execução das instruções contidas dentro da função “Event OnTick” ocorrem a cada quadro de execução da cena.

A linguagem possui todos os tipos de dados primitivos e, além disso, a funcionalidade para criar novas estruturas de dados automaticamente disponibilizando todos os *handles* necessários para a estrutura de forma dinâmica dentro de cada classe da linguagem blueprint. Na Figura 5.2 foi criada uma função com o nome de “MyFirstFunction” onde ela espera como entrada todos os tipos de dados primitivos, o primeiro nó simbolizado pela cor branca é o nó do tipo *Exec* responsável pela execução das instruções conforme a ordem de suas conexões dentro da blueprint.

Outra característica dessa linguagem é a integração e controle de modelos, ambientes, sons e animações de forma dinâmica com seu projeto. Existem nós específicos para controlar rotação, escalonamento, translação de objetos, modelos e animações. Além disso a linguagem blueprint conta também com o controle avançado de sons 2D e 3D permitindo alterar as percepções sonoras de profundidade com volumes e características dinâmicas.

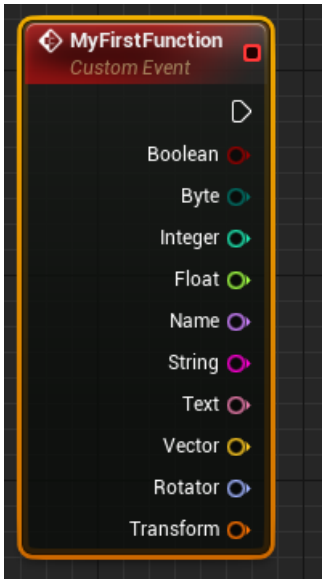

Figura 5.2: Função que recebe como parâmetro todas os tipos primitivos. Fonte: autoria própria.

O paradigma orientado a eventos em blueprint

Uma característica muito importante que esta linguagem introduz é a forma como lida com eventos e como os eventos são capazes de disparar a execução de instruções ou funções dentro da classe blueprint.

A linguagem visual blueprint já vem preparada para lidar com eventos do tipo: "On-Touched", "OnClicked", "OnPressed", "OnHover", "OnOverlap", ‘ "OnCollisionBegin", "OnCollisionEnd" e muitos outros que passam a se tornar específicos com base na herança da classe. Estes eventos pré-definidos quando disparados executam as ações definidas pelo programador,tornando possível, por exemplo, saber com que dedo o evento de toque na tela foi executado, ou que ator realizou a colisão com o objeto ao qual a classe blueprint pertence.

Além dos eventos pré-definidos é possivel criar novos eventos que serão disparados com base em condições definidas pelo programador. Esta característica é denominada *event dispatcher* e tem por objetivo criar eventos customizados ou realizar a comunicação entre diferentes classes de blueprint sem que seja necessário cast de classes ou passagem de objetos por parâmetros [30].

Na Figura 5.3 está a demonstração de como é realizado o bind de um evento "MyEventDispatcher" no construtor da classe "Actor1", para definir qual será a função a ser executada qual o evento for chamado.

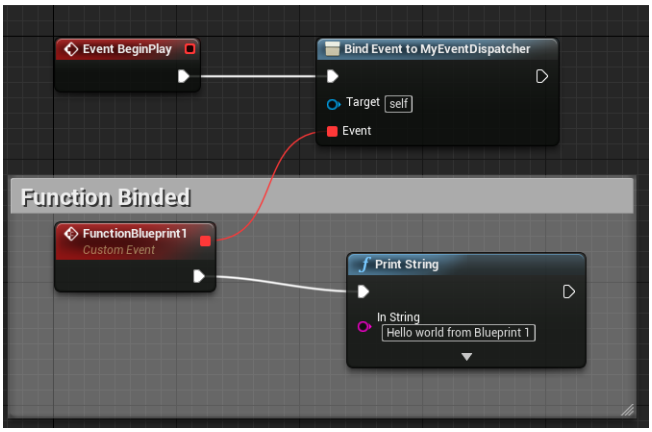

Figura 5.3: Realização do bind de evento no construtor do blueprint. Fonte: autoria própria.

A chamada do evento a partir de outra classe pode ser visto por meio da Figura 5.4, para que o mesmo seja executado apenas é necessário uma referência ao objeto e então a realização da chamada pelo nó "Call" seguido do nome definido no momento da criação do *EventDispatcher*

Figura 5.4: Realização do bind de evento no construtor do blueprint. Fonte: autoria própria.

O paradigma orientado a objetos em blueprint e a integração com a linguagem C++

Dentro da linguagem visual blueprint o conceito de orientação a objetos é usado amplamente e tem as seguintes características: herança, polimorfismo, sobrecargas e virtualização de métodos. Uma das suas limitações quanto a orientação a objetos é quando se faz necessário a utilização de heranças múltiplas, pois o mesmo não possui suporte.

A linguagem blueprint foi toda concebida sobre a linguagem C++ e sua desenvolvedora fez questão de manter não só a comunicação entre as duas linguagens como também permitir que classes blueprint herdem classes em C++. Isso permite uma gama gigantesca de possibilidades no decorrer do desenvolvimento.

É possível garantir por meio de pequenos comandos, por exemplo, que uma função em C++ seja exposta para blueprints ou que uma classe em blueprint que possui herança de uma classe C++ faça sobrecarga ou realize polimorfismo. Tudo isso garante que durante a fase de desenvolvimento os programadores possam disponibilizar interfaces ou classes intermediárias por meio de blueprints para que membros da equipe não especializados em programação possam realizar alterações e a prototipagem de modelos, animações e cenários.

Abaixo segue um exemplo de como ceder visibilidade de uma variável pertencente a uma classe feita em C++ para as classes do tipo blueprint dentro de um projeto da Unreal Engine.

```
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category=Animations)
class UPaperFlipbook* RunningAnimation;
```

Como pode ser observado por meio das linhas de código acima o UPROPERTY é a chamada padrão para definir a visibilidade de variáveis e métodos dentro do arquivo de cabeçalho das classes C++, por meio dela é possível definir o tipo de controle que a classe em blueprint terá, podendo ser: “EditInstanceOnly”, “EditDefaultsOnly”, “VisibleAnywhere”, “VisibleInstanceOnly”, “VisibleDefaultsOnly”, “BlueprintReadOnly”, “BlueprintReadWrite” e “AssetRegistrySearchable”.

Exemplos utilizando a linguagem

Para simplesmente escrever uma string na tela a linguagem blueprint possui um nó chamado “Print” que permite definir o texto a ser exibido, por quanto tempo o mesmo será exibido e a cor do mesmo.

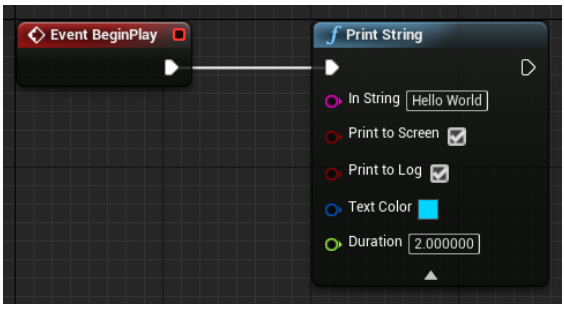

Figura 5.5: Simples execução de um nó print com o texto “Hello World” exibido por 2 segundos. Fonte: autoria própria.

Na Figura 5.6 é possível observar como são realizadas operações de multiplicação e divisão na linguagem blueprint. A operação de divisão não se diferencia em nada além do nó de operação, o mesmo vale para as operações de adição e subtração.

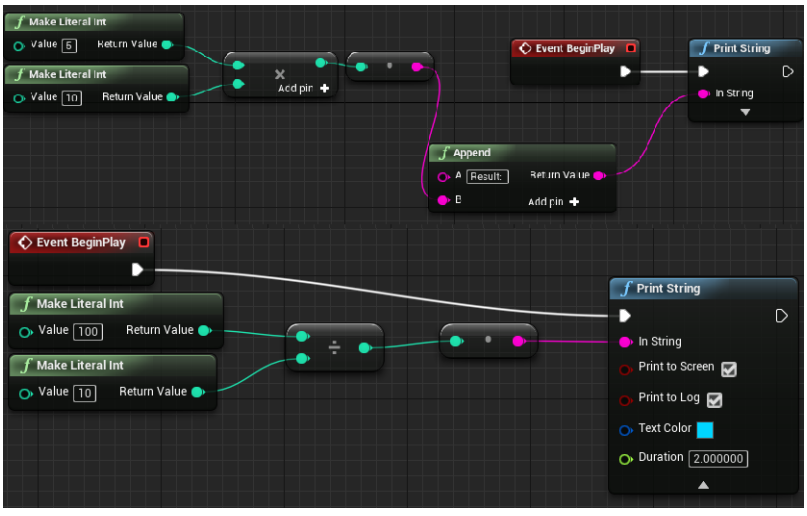

Figura 5.6: Criação de duas variáveis inteiras locais e a simples realização de uma multiplicação (topo da imagem) e a realização de uma divisão (inferior da imagem) . Fonte: autoria própria.

Na linguagem blueprint as operações que lidam com condicionais como switch e if

(denominado branch em blueprint) possuem nós bem intuitivos e auxiliam bastante os iniciantes. Pode-se observar na Figura 5.7 que também é possível realizar comparações com valores primitivos. O nó branch que é considerado uma equivalência ao if-else possui duas saídas, uma true e outra false das quais o fluxo é definido com base no alcance da condição estipulada como entrada do nó.

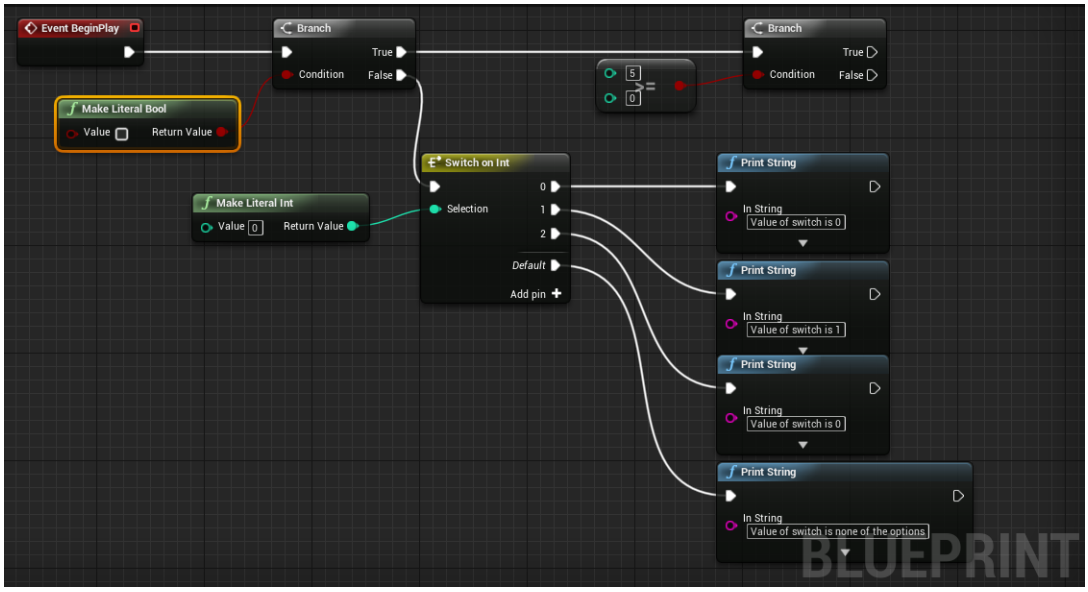

Figura 5.7: Utilização de nós condicionais em blueprint. Fonte: autoria própria.

Os laços de repetição que a linguagem blueprint possui são basicamente o WhileLoop e ForEachLoop, que são idênticos em comportamento quando comparados com linguagens como C++. Na Figura 5.8 é realizada a operação de 100 repetições no laço while, onde a cada repetição é realizado o incremento da variável Int do tipo inteiro e posteriormente adicionado a um *Array* de inteiros, ao final da execução o laço ForEach entrará em ação mostrando na tela cada um dos valores inseridos dentro do *Array*.

As funcionalidades e a abordagem que a linguagem tem facilita o aprendizado e proporciona ainda uma forma de se manter uma equipe de programadores mais experientes e amadores trabalhando juntos por meio das interfaces entre a linguagem blueprint e a C++.

A linguagem de programação visual blueprint possui características que facilitam

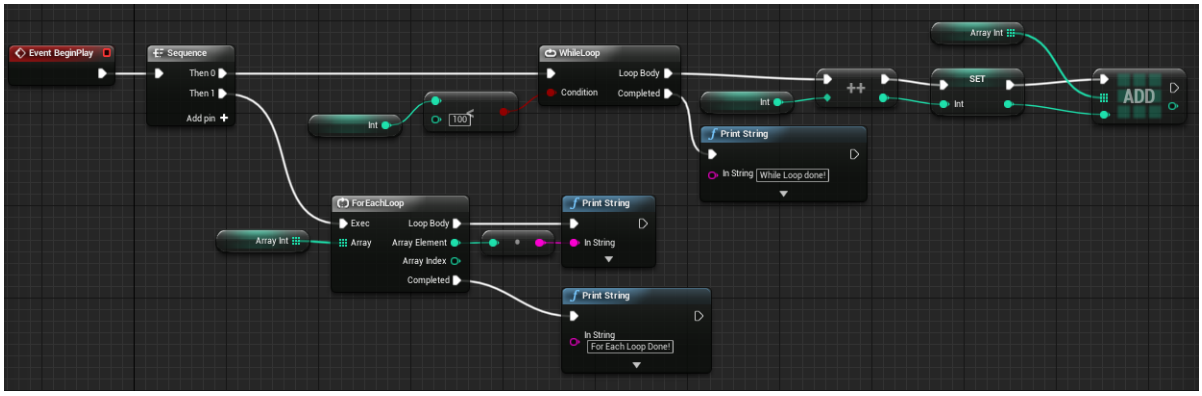

Figura 5.8: Utilização de nós condicionais em blueprint. Fonte: autoria própria.

muito a programação. Entretanto, quando se trata de algo mais específico que exige uma melhor performance é necessário programar em uma linguagem de mais baixo nível, no caso da Unreal Engine, a solução seria programar em C++.

Recomenda-se que a linguagem seja utilizada para a prototipagem e criação de elementos que não sejam muito custosos computacionalmente, pois além de agilizar o desenvolvimento o blueprint permite ter um feedback visual da execução dos nós e se necessário também é possível realizar o *debugging* de maneira dinâmica enquanto a aplicação está sendo executada.

5.1.4 Behavior Tree

As behaviour trees tratam-se de estruturas hierárquicas no formato de árvores, que partem de um nó raiz até aos nós folha, que são responsáveis pela execução de ações e retornam valor de verdadeiro ou falso, dependendo no sucesso de suas execuções.

Olhando de maneira superficial para as Behaviour Trees é possível notar que as mesmas se assemelham muito com as FSM (Finite State Machines), entretanto possuem diversos mecanismos e diferenciais nos controles dos nós para evitar bugs e tornar mais fácil a leitura e estruturação dos estados da árvore.

As behaviour trees contam com alguns tipos de nós, sendo eles:

- Raiz: Nó inicial responsável pela execução da árvore de comportamentos;

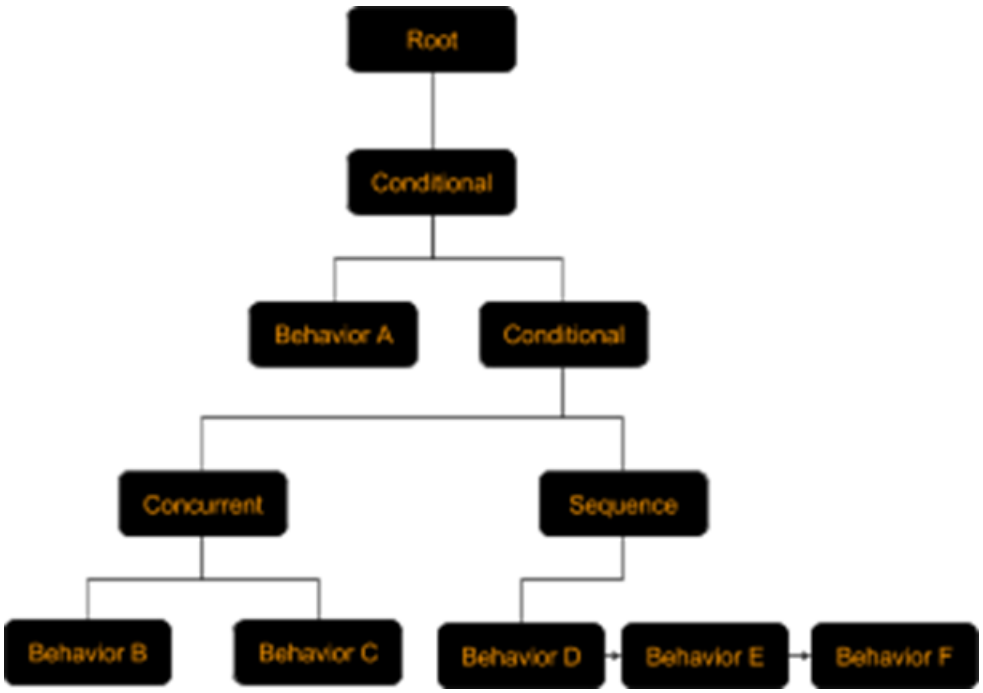

Figura 5.9: Exemplo de estrutura utilizada em Behavior Tree. Fonte: gamasutra.com.

- Control Flow: Nó responsável por controlar como será realizada a execução das tarefas de seus nós filhos. Este nó pode ser dos seguintes tipos:
  - Composite ou Selector: Nó que se comporta de maneira semelhante a porta lógica OR onde a execução com sucesso de ao menos um dos seus nós filho é o suficiente para determinar que o estado do nó do tipo Composite seja true;
  - Decorator ou Sequence: Nó que é tratado como uma forma de porta lógica AND pelo fato de que seu valor somente será true se todos os seus nós filhos forem executados com sucesso.
- Folha: Nó que realiza uma ação e retorna o valor de true (realizou a ação) ou false (não realizou a ação) ao seu nó pai.

Uma característica que permite a reutilização de nós já programados é o fato das Behaviour Trees possuírem a persistência dos dados obtidos pela árvore permitindo, por exemplo, que um nó folha ao executar a chamada de outra behaviour tree passe esses

valores (condicionais, estados) para a árvore que será invocada, o que poderá fazer com que o agente se comporte de maneira diferenciada com base nas ações passadas.

Algumas bibliotecas e engines que implementam as Behaviour Trees permitem ao desenvolvedor explorar o potencial desse tipo de estruturas, de forma que árvores encadeadas possam ser executadas de maneira paralela para um mesmo agente, fazendo com que uma IA dentro do jogo busque eliminar o jogador e também monitore e reaja ao ambiente para buscar o melhor ponto para cobertura ou ângulo para mira enquanto troca tiros ou golpes com o jogador. No geral, as implementações de behaviour tree tendem a seguir o modelo padrão para controle do fluxo de execução, que atribui ao menos um dos seguintes estados para cada nó:

- Success;
- Failure;
- Running.

Esse tipo de estrutura permite que os nós pai retenham o resultado da execução de seus nós filhos e que com base nisso outras ações possam ser realizadas, como por exemplo, quando ao realizar um disparo se não houver mais munição disponível o nó pai receberá "Failure" como valor e então o nó adjacente superior na hierarquia se encarrega de executar uma tarefa para buscar mais munição.

O processamento paralelo trouxe consigo algumas vantagens e permitiu também que as behaviour trees fossem utilizadas em conjunto, de forma que houvesse a troca de informações entre agentes pré-determinados, como por exemplo, ao avistar o jogador a IA começa a disparar contra o mesmo e também invoca e comunica-se com as behaviour trees dos outros agentes disponíveis na área, fazendo com que suas ações sejam coordenadas e que possam ser realizadas de maneira cooperativa, seja contra ou com o jogador.

Behavior trees na Unreal Engine

Em relação ao modelo conceitual das Behaviour Trees, a desenvolvedora da Unreal Engine realizou algumas alterações e melhoramentos, criando a sua própria versão da Behaviour Tree com maior praticidade, legibilidade e eficiência na execução das tarefas. Um dos principais diferenciais entre o modelo conceitual e a implementação da Unreal Engine foi a utilização de Decorators para fazer o controle da execução dos nós folha, fazendo assim com que não haja um retorno da execução realizada pelo próprio nó.

Outro grande diferencial das adaptações realizadas pela equipe da Epic Games foi a mudança de como era realizado a execução de comportamentos paralelos. Seguindo a definição conceitual da Behaviour Tree um Composite Parallel Node é utilizado para lidar com comportamentos concorrentes de seus nós filhos, certas regras podem ser definidas e quando forem atingidas tasks podem ser executadas. Para suprir a função do Composite Parallel Node foram criados 3 tipos de nós, que são os seguintes:

- Simple Parallel Nodes: Semelhante ao Composite porém ele limita que hajam apenas dois nós filhos, garantindo a facilidade na utilização e na hora de realizar o debugging;
- Services: Nó especial que realiza callbacks a cada X segundos realizando atualizações periódicas dos estados dos nós;
- Decorator: Semelhante ao services porém o único objetivo deste tipo de nó é verificar quando os nós filhos são finalizados prematuramente.

5.1.5 Octalysis Framework

Conforme descrito na seção 3.5, a Octalysis Framework[3] é uma ferramenta utilizada para o Game Design voltado ao engajamento do jogador. Este Framework será utilizado para definir os elementos de Game Design a serem utilizados neste trabalho.

[3]http://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/

Esta ferramenta auxilia no equilíbrio de certos elementos de Game Design evitando que o foco seja somente motivadores intrínsecos ou extrínsecos, o balanceamento desses tipos de elementos é crucial para garantir que o jogador sinta-se bem ao jogar e mantenha-se o maior tempo possível entretido.

5.2 Blender

Blender[4] é uma suíte de ferramentas 3D gratuíta e de código aberto, que possui licença de utilização General Public License (GPL). As ferramentas incluem a criador e editor de modelos 3D, criador e editor de armações, animador, ferramenta para simulação, renderizador de cenas e modelos criados [31].

O blender é uma ferramenta multi plataforma feita utilizando OpenGL com o objetivo de manter uma experiência consistente e compatibilidade nas principais plataformas do mercado. Além disso, o software possui compatibilidade com os formatos mais utilizados da área de modelagem e design gráfico, permitindo a importação e exportação de modelos, animações e figuras para auxiliar no processo de criação e desenvolvimento [31].

Essa suíte se sobressai por possuir todas as principais ferramentas necessárias para lidar com modelos 3D e seus diversos aspectos. A interface amigável e intuitiva não só facilita o processo de modelagem e desenvolvimento como também aumenta a eficiência do processo garantindo que o tempo desempenhado nessa etapa seja majoritariamente focado no processo artístico, construção dos modelos, criação dos materiais e texturas.

Além do blender conter uma ferramenta de modelagem 3D, ainda conta também com plugins e ferramentas para auxiliar no processo de modelagem, como por exemplo, ferramentas de retopologia de modelos 3D capazes de reduzir a quantidade de polígonos de uma estrutura modelada, sem que ocorra a perca da característica da superficie dos objetos, garantindo assim um melhor desempenho sem que haja uma perca efetiva de qualidade.

[4]https://www.blender.org

5.3 Hardware

O hardware utilizado para o desenvolvimento da aplicação com a Unreal Engine trata-se de um computador portátil pessoal da marca Msi, que possui 16 giga bytes de memória ram, processador intel i7 7700HQ de 2.8Ghz e uma placa de vídeo GTX 1050 com memória gddr5 de 4 giga bytes. Além disso foi utilizado um disco de estado sólido com 240 giga Bytes de armazenamento e um disco rígido de backup com 1 tera byte de armazenamento.

Nesta seção foi comentado as ferramentas e linguagens a serem utilizadas no decorrer deste trabalho, na próxima seção será descrito como se dará o desenvolvimento do mesmo.

5.4 Métodos

Inicialmente para o desenvolvimento deste trabalho será necessário realizar a etapa de *Game Design* para definir quais elementos do jogo original de tabuleiro estará presente na versão digital que será desenvolvida e como a interação ocorrerá entre o usuário e o jogo, por meio das mecânicas.

Ao final da primeira etapa do levantamento de informações acerca do game design e mecânicas, será realizada então a definição das tarefas a serem realizadas para o desenvolvimento juntamente de um prazo médio para a realização de cada uma delas.

Após o desenvolvimento de cada uma das tarefas serão realizados testes práticos para assegurar o funcionamento da tarefa e verificar se o resultado é o esperado dentro do escopo definido previamente.

A metodologia escolhida para realizar a análise durante o período em que houve a aplicação do Cabinet segue um princípio de observação das interações entre jogador-jogador e jogador-jogo, além de também ter sido realizado uma entrevista semi-estruturada com o objetivo de coletar *feedback* dos alunos acerca de sugestões ou problemas e dificuldades encontrados.

A análise da aplicação por observação ocorreu por meio de gravações de vídeo das partidas realizadas e também pela interação entre os jogadores enquanto foi realizada.

Além disso também foi aplicado um questionário com 13 questões, com o objetivo de ter uma base qualitativa para auxiliar na avaliação do jogo. O questionário foi aplicado tanto para os alunos que jogaram a versão analógica quanto para os que jogaram a versão digital.

O questionário aplicado continha questões como a classificação do objetivo do jogo, classificação da diversidade e área de aplicação do mesmo. Havia também questões para avaliação do cenário, visual gráfico, tempo de jogo, quantidade de partidas até a primeira vitória, avaliação das regras e instruções.

Após a etapa de aplicação foi realizada uma avaliação dos aspectos do jogo utilizando a Octalysis *Framework* para definir quais são os motivadores intrínsecos e extrínsecos provenientes dos elementos e mecânicas presentes no Cabinet, que servem como base para mensurar os efeitos que cada uma delas possui sobre os alunos e verificar se é possível correlacionar os dados obtidos por meio da avaliação por observação com a perspectiva que o framework disponibiliza acerca dos elementos do jogo.

5.5 Sumário

Neste capítulo foram descritos as ferramentas e hardware necessários para o desenvolvimento deste trabalho, e também os métodos que englobam o levantamento das mecânicas de jogo, requisitos e o desenvolvimento do mesmo, assim como as metodologias escolhidas para a aplicação do jogo em sala de aula e sua avaliação. O próximo capítulo tratará de descrever todo o processo de desenvolvimento da plataforma.

Capítulo 6

Desenvolvimento/Implementação

Neste capítulo será apresentada a implementação do jogo, assim como será descrito todos os problemas enfrentados e as soluções desenvolvidas com o objetivo de alcançar o resultado final para que fosse possível a aplicação e teste do jogo em sala de aula.

6.1 Regras do Jogo

No início de cada partida é necessário decidir que jogador inicia o jogo, sendo decidido de maneira arbitrária entre os jogadores.

O jogo consiste basicamente em 2 fases: a fase de coleta dos recursos e a fase de compra/instalação de equipamentos. Na primeira fase, os jogadores escolhem os recursos que desejam coletar na rodada, sendo que um recurso já selecionado não pode voltar a ser escolhido até o final da rodada atual.

Cada jogador possui três trabalhadores, de forma que cada um deles possa realizar a coleta de um único recurso. Ao escolher o recurso desejado, o jogador recebe instantaneamente o mesmo. Ao final da etapa de coleta, os jogadores entram na fase de compra, onde escolherão como irão gastar os recursos coletados, podendo ser por meio da aquisição de novos racks, aquisição de novos servidores ou serviços.

Como pode ser observado na Figura 6.2, cada rack possui espaço físico para receber somente dois servidores e, até ao final da partida, podem ser adquiridos somente três

Figura 6.1: Tabuleiro do jogo Cabinet. Fonte: Rui Pedro Lopes.

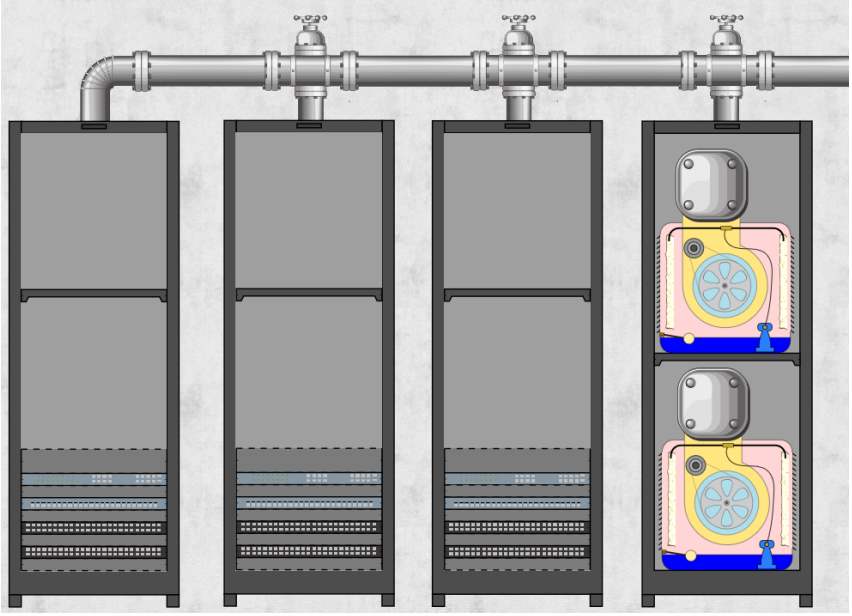

Figura 6.2: Racks para servidores. Fonte: Rui Pedro Lopes.

racks.

Após a realização de todas as ações desejadas pelos jogadores na etapa de compra, é iniciada uma nova rodada, onde os jogadores poderão novamente realizar a coleta de recursos. O jogo termina assim que forem realizadas 8 rodadas completas entre os jogadores e o vencedor é definido com base na pontuação obtida ao longo do jogo.

O cálculo da pontuação ao final de cada partida é realizado levando em consideração

a utilização do espaço disponível nos racks adquiridos e também com base na utilização dos servidores adquiridos, por meio da contagem de serviços instalados em cada um deles.

Para cada espaço ocupado em um rack o jogador recebe dois pontos, enquanto que para cada espaço vazio o jogador perde três pontos. Cada serviço de rede ou aplicação empresarial instalado num servidor vale um ponto. Entretanto, após instalado o quarto serviço de rede, cada serviço de rede sobressalente passa a valer dois pontos. Após a instalação da quinta aplicação empresarial, cada uma das aplicações empresariais sobressalentes passam a valer dois pontos.

6.2 Os recursos

Uma noção importante que o Cabinet passa ao aluno no decorrer do jogo é o conceito de recursos. Assim como em um ambiente real, durante as partidas um fator decisivo para a vitória é saber como e onde os utilizar.

Para auxiliar no aprendizado do aluno foram escolhidos os recursos que seriam utilizados no decorrer do jogo. Abaixo segue a listagem de todos os recursos que podem ser obtidos no Cabinet e suas características:

-  Fonte de alimentação: recurso necessário para alimentar energeticamente os componentes de hardware dos servidores;
-  Componentes de Hardware: recurso que representa peças de hardware, essêncial para o funcionamento dos servidores;
-  Interface: recurso que corresponde a periféricos utilizados como meio de interface, que têm como principal objetivo permitir a interação entre ser humano e computador;
-  Recurso de Virtualização: recurso indispensável para habilitar a virtualização

em servidores, permitindo assim a instalação de mais sistemas operacionais e a melhor utilização do hardware disponível;

-  Sistema operacional: recurso utilizado para a ativação dos sistemas operacionais nos servidores adquiridos e que serve como base para a instalação de serviços de rede e aplicação empresarial;
-  Serviço de rede: recurso que representa serviços de rede como: servidor Domain Name System (DNS), servidor web, servidor de e-mail e outros. Este recurso pode ser instalado nos servidores desde que o mesmo possua um sistema operacional disponível para receber o serviço;
-  Aplicação empresarial: esse recurso representa todo tipo de aplicação que possa se fazer necessária em um ambiente empresarial. Este recurso pode ser instalado nos servidores desde que o mesmo possua um sistema operacional disponível para receber o serviço.

6.3 Multiplayer

O desenvolvimento do Cabinet foi iniciado após a definição dos elementos de game design no qual o jogo iria herdar de sua versão de tabuleiro. Como grande parte dos elementos foram mantidos e já possuindo um game design consolidado, foi possível então focar no desenvolvimento dos elementos gráficos, sonoros e mecânicos do jogo.

Uma das primeiras etapas do desenvolvimento foi definir como seriam realizadas as interações entre jogador-jogador. A princípio a ideia era fazer com que toda a partida acontecesse com somente um dispositivo, de forma a que os jogadores realizassem as jogadas de maneira intercalada. Entretanto, pensou-se em utilizar as funcionalidades que a Unreal Engine proporciona, para não só permitir que cada aluno pudesse jogar em seu dispositivo, mas também em rede local e por meio da internet.

A Unreal Engine possui a estrutura necessária para lidar com partidas multijogador pois permite ao desenvolvedor definir como serão tratados os atores dentro do jogo e que propriedades ou aspectos do jogo serão replicados entre os jogadores, com o objetivo de manter uma sincronização dos dados e a integridade ao longo da partida.

Inicialmente cogitou-se permitir somente partidas em Local Area Network (LAN), entretanto, após realizar testes em redes *wireless* foi notado que em alguns destes testes houve bloqueio por parte das definições da rede, não permitindo que dispositivos conseguissem interagir entre si. Para solucionar esse problema, foi realizado estudos na documentação da Unreal Engine para encontrar formas de criar uma instância dedicada, de forma que os jogadores possam se conectar a um servidor externo sem que haja a necessidade da interação direta entre ambos os dispositivos.

Como cada partida do Cabinet é jogada entre apenas dois jogadores, após o desenvolvimento do servidor dedicado foi necessário achar um meio de manter e controlar diversas instâncias de servidores, para garantir que os jogadores conseguissem criar e se juntar a partidas a partir de qualquer lugar do mundo. Para que isso fosse possível, foi desenvolvido uma aplicação em Java com o objetivo de controlar todas as sessões abertas, ou mesmo permitir que novas instâncias sejam criadas. Essa aplicação em Java realiza a comunicação com os clientes por meio de conexões síncronas feitas por meio de sockets. Essa aplicação em java recebe conexões dos clientes sempre que for necessário criar uma nova sessão, buscar por uma sessão já existente e também para encerrar sessões ativas.

A aplicação Java é responsável por manter a lista de sessões ativas e informações relativas a cada uma delas, além disso, sempre que o cliente faz uma requisição para a aplicação, a mesma realiza a operação e retorna a confirmação da mesma, para que por fim a conexão seja encerrada.

Na Figura 6.3 pode-se observar o fluxo de operações realizadas durante uma requisição feita pelo cliente a aplicação Java responsável pelo gerenciamento das sessões. A operação em questão trata-se da criação de uma nova sessão, onde o cliente se conecta ao gerenciador de sessões e em seguida solicita a criação de uma nova sessão, nesse momento o gerenciador executa uma nova sessão e atribui um número aleatório (definido com base em regras)

a sua porta de conexão e, em seguida, envia ao cliente o número de sessão gerado. A partir disso, o cliente conecta-se a sessão criada e aguarda para que o segundo jogador se conecte.

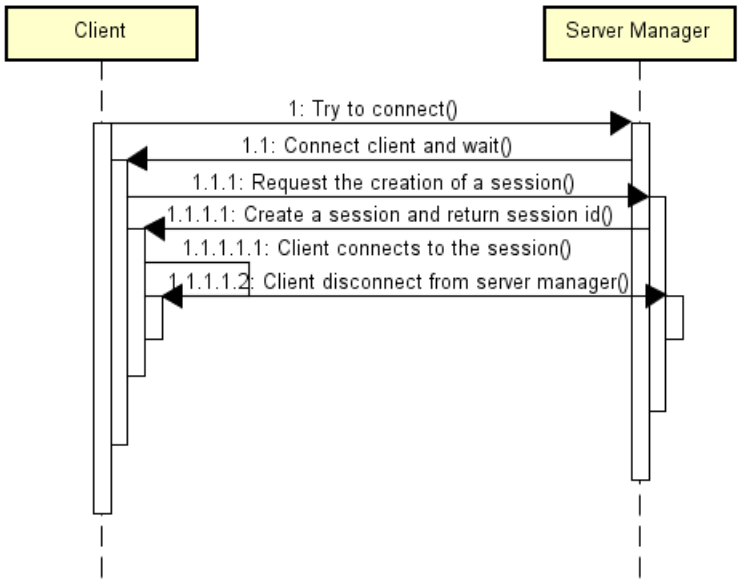

Figura 6.3: Diagrama de sequência representando a solicitação da criação de uma nova sessão. Fonte: autoria própria.

A operação de busca por uma sessão ativa segue o fluxo exibido na Figura 6.4, na qual o cliente se conecta à aplicação Java responsável pelo gerenciamento das sessões e solicita o id de uma sessão ativa. O servidor então percorre sua lista tentando encontrar uma sessão em que há vagas disponíveis para o cliente. Assim que encontra uma sessão disponível o gerenciador retorna um pacote com o id da sessão na qual o cliente se deve conectar e então o cliente encerra a conexão com o gerenciador de sessões.

Além das duas operações já descritas sobre o gerenciador de servidores, há uma terceira operação responsável por encerrar sessões ativas. A solicitação dessa operação pode ser realizada tanto pelo cliente ao encerrar o jogo quanto pela própria instância do servidor, quando o mesmo notar que não há jogadores conectados a ela por um período igual ou maior a 120 segundos. Essa operação garante que não hajam recursos desperdiçados com instâncias que não possuem jogador algum, assim como essa operação garante também

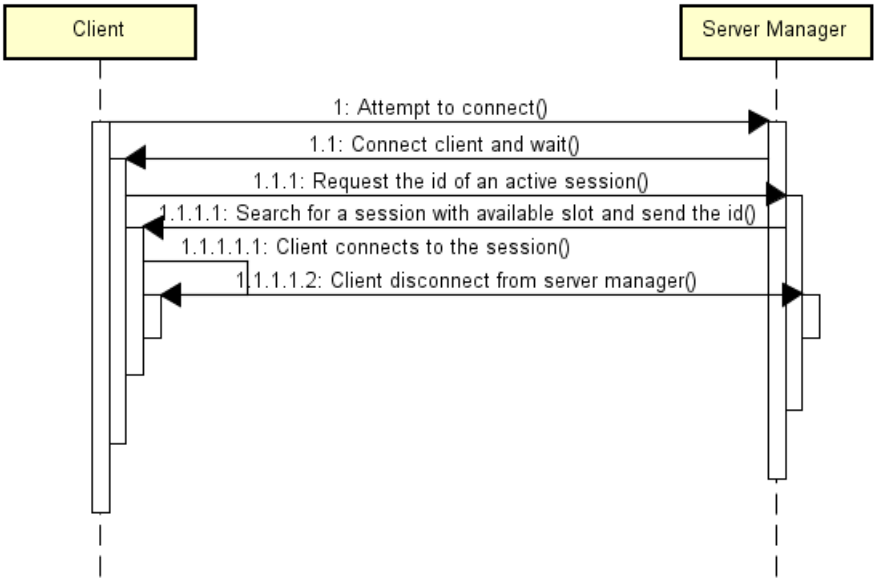

Figura 6.4: Diagrama de sequência representando a solicitação da criação de uma nova sessão. Fonte: autoria própria.

que a operação de busca por sessões ativas seja íntegra, não redirecionando o jogador para um sessão em que não há jogadores.

Toda a comunicação que ocorre entre o cliente e o gerenciador de sessões se dá por meio de um host dinâmico para facilitar caso seja necessário realocar a aplicação central para outra máquina. A resolução deste host dinâmico e também toda a comunicação via socket no cliente foi feito por meio da classe "TCPSocketActor" na linguagem C++.

6.4 Replicação e controle

Antes de começar efetivamente o desenvolvimento das mecânicas e outros elementos relativos a jogabilidade foi necessário entender como que a Unreal Engine lida com as informações, como é feita a replicação das variáveis e quem detém o poder sobre os objetos e os agentes.

A estrutura de toda cena criada na Unreal Engine segue um padrão, todo jogador

conectado a uma partida possui seu próprio *PlayerController*, na qual é uma classe responsável por lidar com diversos tipos de comandos, retém o controle sobre as interfaces utilizadas para comunicação do dispositivo com o jogo em si.

Cada classe *PlayerController* possui uma referência a classe *PlayerState*. A classe *PlayerState* é responsável por armazenar informações do jogador ao longo da partida, entretanto, quem detêm a autoridade sobre o *PlayerState* é a classe denominada *GameState*.

O *GameState* é responsável por manter e gerenciar as informações durante aquela determinada partida, também se encarrega de manter uma lista *PlayerArray* na qual contém todas as instâncias criadas e ativas da classe *PlayerState* ao longo da partida. Os jogadores possuem somente uma cópia do *GameState* e sempre que alguma informação é alterada na instância contida no servidor, os jogadores passam a receber atualização das mudanças. Esse processo é tratado pela Unreal Engine por replicação.

Inicialmente houve muitos problemas relacionados a replicação de características e variáveis de atores e objetos em cena. Isso ocorreu devido ao fato de que, na Unreal Engine a replicação só ocorre quando o dono do objeto foi quem realizou a alteração. Além disso, é necessário que a variável ou componente tenha sido definida como replicável, caso contrário todos os jogadores que possuam acesso a essa variável ou componente não receberão as alterações.

Sempre que um cliente precisa executar uma ação em uma partida multijogador é necessário definir a abrangência dessa ação, se por exemplo, essa ação pode alterar algo no cenário ou alguma informação que pode se tornar relevante para o desfecho do jogo é necessário que todos os jogadores estejam sincronizados. Sempre que um cliente realiza uma ação desse tipo, ele deve repassar a informação para que seja executado no servidor que em seguida se encarregará de sincronizar todas as instâncias. Essa execução no servidor é chamada de RPC.

Todo método ou evento criado na Unreal Engine pode ser definido como um RPC, entretanto, para que seja tratado como tal ele precisa ser chamado a partir de uma classe ou objeto o qual seja pertencente ao *PlayerController* responsável por executar a ação caso contrário é necessário criar um método intermediário que delegará a ação ao servidor. Um

exemplo disso são os objetos do cenário, no qual o servidor é dono. O jogador não pode executar uma RPC que foi declarada para aquele objeto diretamente pois o mesmo não possui autoridade sobre o objeto, portanto, é necessário que o *PlayerController* possua um método que passe como referência a instância do objeto com o qual pretende interagir, chamando uma função que será do tipo "*Executes on Server*" delegando a interação para que o servidor realize as operações necessárias sobre o objeto no qual ele possui autoridade. Tudo isso para garantir que todas as instâncias dos clientes estejam sincronizadas entre si, garantindo que tudo na partida saia como o esperado e de quebra também evita a utilização de trapaças nas partidas, tendo em vista que toda ação será validada pelo servidor e não pelo cliente.

Todo jogo desenvolvido utilizando a Unreal Engine possui uma estrutura básica, a qual deve ser seguida para garantir o funcionamento básico do jogo. Essa estrutura pode ser observada na Figura 6.5, onde é possível notar o papel fundamental das classes GameMode, GameState, PlayerController, Pawn, HUD, Input, AIController, PlayerCameraManager.

6.5 Mecânicas

As mecânicas foram um dos primeiros elementos a serem implementados, e de certa forma um dos maiores desafios no desenvolvimento do Cabinet. Isso se dá pelo fato de não haver muitos jogos de tabuleiro com multiplayer desenvolvidos na Unreal Engine, além de também ser uma engine na qual não possuo experiência.

No início da partida do Cabinet é necessário definir quem será o jogador que terá o direito de realizar a primeira jogada. Durante o desenvolvimento da versão digital foi definido que isso seria escolhido com base no primeiro jogador a se conectar na sessão, desta forma, o último jogador a se conectar será o segundo a realizar a jogada no primeiro turno.

Sempre que um jogador realiza sua jogada na etapa de coleta de recursos é necessário comunicar ao servidor que aquele jogador realizou sua jogada, impedir que ele realize mais jogadas e passar essa permissão para o adversário. Após cada jogador realizar as 3

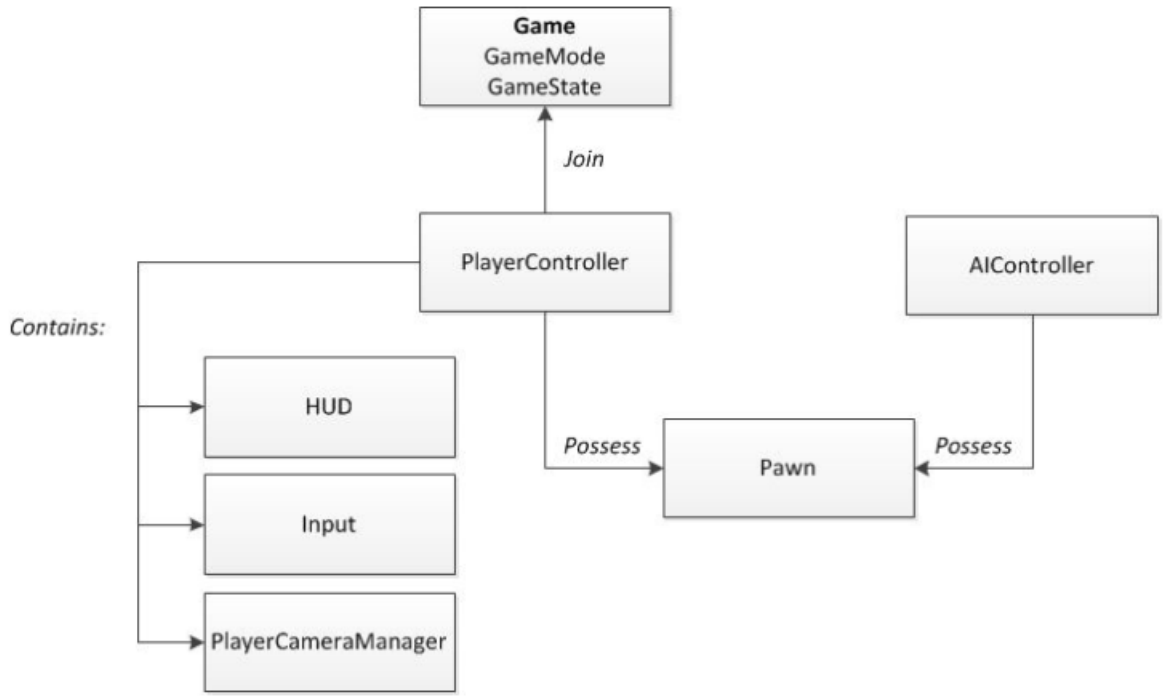

Figura 6.5: Diagrama representativo da estrutura básica das classes na Unreal Engine. Fonte: www.blaenkdenum.com/notes/unreal-engine/#user-interface.

jogadas a que tem direito, o *round* atual é encerrado.

Ao encerrar o *round* é necessário revogar as permissões de realizar jogadas de ambos os jogadores e, então, o servidor faz o uso de RPC dos clientes com o objetivo de deslocar as câmeras de cada client para seu data center, ao mesmo tempo que, habilita na tela do jogador o *widget* que lhe permite realizar as operações de compra de racks e servidores. Ainda durante a fase de compra o aluno pode instalar serviços nos servidores por meio de uma operação de arrastar e soltar, onde um recurso contido no Heads Up Display (HUD) pode ser arrastado sobre um servidor já instalado com o objetivo de ativar o serviço naquele servidor.

Logo ao iniciar a etapa de compra do primeiro *round* é obrigatório ao jogador adquirir um *rack*. Ao adquirir um *rack* o jogador tem a opção de colocar em dois locais, que serão representados com uma silhueta verde do modelo do rack. Conforme referido em 6.1, cada rack pode receber até três servidores. Sempre que o jogador solicite a compra de um

servidor no jogo, é executado um RPC no servidor que verificará se ele possui os recursos necessários e o espaço disponível para a aquisição. Caso o resultado dessa verificação seja verdadeiro é exibido ao jogador as opções de locais disponíveis para a alocação do *hardware* adquirido. Com isso o jogador pode definir em que local irá colocar o hardware pressionando sobre a silhueta do servidor exibida nos locais disponíveis.

Após encerrar todas as compras e instalações desejadas o jogador pode definir que está pronto por meio de um *checkbox* no canto de sua tela, executando um RPC no servidor que incrementa uma variável responsável por controlar os jogadores que estão prontos para o próximo *round*. Sempre que essa variável atingir o valor 2 o servidor fará chamadas RPC nos clientes que estão conectados para que suas câmeras se voltem ao tabuleiro, em seguida dará permissão de jogada ao jogador que recolheu o recurso de interface no último *round*. Caso esse recurso não tenha sido coletado na partida anterior, o primeiro jogador a realizar a jogada será o mesmo jogador que iniciou no *round* anterior.

Quando chega o final dos 8 *rounds*, conforme descrito em 6.1, ambos os jogadores possuem seus direitos de interação com o jogo removidos para que o servidor possa realizar a etapa de contabilização dos pontos. Após a contabilização dos pontos o servidor define quem foi o vencedor da partida e executa um RPC em cada cliente, responsável por exibir a tela final da partida contendo a pontuação obtida e se o jogador foi o vencedor.

Na seção seguinte será explicado com maior nível de detalhamento como foi feita a implementação das regras, mecânicas que foram descritas nesta seção.

6.6 Programação

Inicialmente toda a programação do jogo foi feita em Blueprints devido a facilidade da criação e busca por sessões multijogador. A Unreal Engine disponibiliza para utilização a comunicação com API de algumas das principais plataformas de jogos, que podem ser utilizadas para vinculação de pontuação e conquistas do jogador, além disso, permite também um gerenciamento das sessões multijogador. Os testes iniciais do desenvolvimento foram realizados utilizando a plataforma Windows e serviço de plataforma padrão *Null* no

qual estabelece que todas as seções estariam registradas somente no âmbito de rede local. Ao serem realizados testes com as plataformas de dispositivos móveis, pôde-se constatar que apesar de ser possível criar sessões utilizando a plataforma padrão como *Null*, por questões de segurança dos sistemas operacionais, a funcionalidade de *broadcast* das sessões não estava funcionando, consequentemente, a busca por sessões nunca retornava resultado algum. O problema relacionado com a visibilidade das sessões foi um dos motivadores que levaram ao desenvolvimento do servidor dedicado e o gerenciador de sessões para o Cabinet, com o objetivo de facilitar a partida entre os jogadores sem que houvesse preocupação quanto a possíveis problemas de comunicação direta entre dispositivos em uma mesma rede.

A maior parte do jogo foi desenvolvida utilizando Blueprints pelo fato de que toda a lógica de programação pôde ser implementada de maneira mais rápida e as alterações que se fizeram necessárias puderam ser realizadas e testadas em questão de segundos. Apesar da utilização do Blueprint ser mais prática do que a linguagem C++, perde-se em desempenho. A diferença torna-se ainda mais notável quando operações que exijam um maior desempenho são requisitadas. Isso deve-se ao fato de que uma máquina virtual é acoplada à construção final do jogo. No entanto, é possível recuperar cerca de 90% da performance perdida por meio da nativização dos arquivos Blueprint. A nativização é uma funcionalidade relativamente nova na Unreal Engine, que permite aos desenvolvedores converter suas classes criadas em Blueprint para código nativo C++ no momento em que é construído o pacote final do jogo. Isso faz com que seja possível aliar a facilidade de prototipação dos Blueprints ao desempenho do C++, acelerando o processo de desenvolvimento e também reduzindo a possibilidade de erros na programação, levando em consideração que ao desenvolver em Blueprint todas as entradas e saídas de dados, assim como o fluxo das operações são verificados pela máquina virtual enquanto os testes estão sendo realizados, isso permite garantir que tudo funcione conforme o esperado, ou na pior das hipóteses, alerte ao desenvolvedor caso algo não saia como o esperado, por meio de mensagens intuitivas e claras.

Pelo fato da Unreal Engine se tratar de um motor gráfico multiplataforma, o desenvolvimento utilizando ela permite que o deploy seja realizado para as principais plataformas do mercado, como: android, iOS, Windows, Linux, Mac, Xbox one, Playstation 4 e Nintendo Switch.

Devido a essa característica foi possível focar os esforços no desenvolvimento de uma solução única capaz de atender a todas as plataformas com o mesmo resultado final, sem a necessidade de adaptação de código ou recursos gráficos ou mecânicas.

6.7 Modelagem

Para o desenvolvimento da versão digital do Cabinet foi necessária a criação de modelo para os servidores já existentes na versão física. Todos os modelos criados foram baseados em alguns aspectos, as figuras do tabuleiro original, e também em aspectos reais de servidores do tipo. Além disso, foi necessário manter um nível de detalhamento baixo tendo em vista o desempenho dos dispositivos móveis. Foram criados modelos com baixa quantidade de polígonos.

Figura 6.6: Servidores disponíveis no Cabinet. Fonte: autoria própria.

A modelagem dos trabalhadores foi realizada de forma que a distinção entre o trabalhador de cada jogador é realizada com base na cor do capacete do mesmo, sendo que, o trabalhador de capacete vermelho pertence ao jogador 1 e o trabalhador de capacete amarelo pertence ao jogador 2.

Além de elementos primordiais envolvendo as mecânicas do jogo, também adicionado

Figura 6.7: Trabalhadores no Cabinet. Fonte: autoria própria.

detalhes que caracterizam esses tipos de ambientes. Além dos racks para os servidores, foi adicionado também um rack com switch contendo uma tela e um teclado que tratam-se da interface a qual os trabalhadores utilizam para a instalação dos serviços nos servidores.

Figura 6.8: Trabalhador realizando a instalação dos serviços nos servidores. Fonte: autoria própria.

Na Figura 6.8 é possível observar a visão do data-center de um dos jogadores, no qual o mesmo possui um rack com 3 servidores instalados e o seu respectivo trabalhador está a realizar operações com o objetivo de instalar os serviços que o jogador definiu.

Outro fator importante que impactou agressivamente na performance em dispositivos móveis foi o tipo de materiais utilizados em cada objeto presente no cenário. Todos os materiais foram feitos utilizando a textura criada no software Blender em conjunto com os mapas de luz e oclusão de ambiente, para garantir que todas as luzes que incidem sobre os objetos em questão não sejam refletidas ou destorcidas pelos seus devidos materiais, proporcionando assim um efeito de luz estático sobre os objetos e poupando processamento gráfico.

6.8 Graphical User Interface

Toda a parte do design, user interface e user experience foi desenvolvida levando em consideração padrões da área de design. Essas escolhas vão desde, posicionamento dos elementos gráficos na tela, tons de cor e efeitos visuais.

A paleta de cores da interface gráfica do usuário foi escolhida utilizando tons de cor azul. Essa escolha foi baseada em estudos que identificam o apelo de cada cor e definem a cor azul como uma cor neutra, que passa confiança e segurança ao usuário. Além disso, a cor azul tem um grande diferencial em relação à outras cores, ela é facilmente distinguível e doenças como daltonismo raramente afetam a distinção dessa cor [32] . A cor azul se tornou um padrão, não só pela sua aceitação dos usuários mas também por ter sido amplamente adotada em plataformas mundialmente conhecidas como: Facebook, Twitter, LinkedIn e PayPal.

Além da questão de cores, o posicionamento dos elementos na tela foi escolhido com o objetivo de dar ao usuário conforto e propiciar um bom uso da interface. Informações relevantes e necessárias foram dispostas sempre no lado esquerdo e no centro da dela, pois conforme as convenções de usabilidade, os usuários tendem a analisar sempre os elementos gráficos da esquerda para a direita, fazendo uso desse conceito a Graphical User Interface

Figura 6.9: Interface gráfica do cabinet durante uma partida. Fonte: autoria própria.

(GUI) da Figura 6.9 referente a etapa de compra do jogo foi construída.

Como pode ser observado na Figura 6.9, ela foi construída exibindo o nome dos servidores a esquerda próximo ao botão de informação, com o objetivo de instigar o aluno a clicar e tomar conhecimento a respeito daquele tipo de servidor, enquanto que no meio há uma figura ilustrativa que representa o servidor e exibe os tipos de serviços que o servidor é capaz de executar, a direita desta, há os requerimentos necessários para realizar a compra do determinado servidor, próximo ao botão de compra.

Todos os elementos pertencentes a GUI têm seus posicionamentos ajustados de acordo com o tamanho da tela na qual Cabinet está sendo executado, além disso ocorre também o escalonamento destes elementos quando sua execução acontece em dispositivos com proporções diferentes, tudo isso para garantir que a experiência seja o mais constante possível independente da plataforma.

6.9 Inteligência Artificial

Durante o desenvolvimento das mecânicas e o sistema multijogador, viu-se a necessidade de trazer também uma alternativa aos alunos caso não conseguissem se conectar aos colegas ou não tivessem algum colega disponível com quem competir. Conforme descrito em 5.1.4, a Unreal Engine conta com um conjunto de ferramentas capazes de lidar com a estrutura padrão de Behavior Trees.

Com base nas mecânicas, foi construída a estrutura de comportamento do agente. Pelo fato do jogo ser em turnos tornou-se relativamente mais prático de realizar a definição das ações a serem tomadas pelo agente.

Devido a divisão do jogo na etapa de coleta e na etapa de compra foi possível segmentar a árvore de comportamentos em duas ramificações. O nó principal da árvore é responsável por definir em qual etapa do round o agente está, desta forma é possível definir qual será o nó seguinte que será executado por meio de verificações em nós do tipo *decorator* que podem ser utilizados para verificar condições ou estados de variáveis do agente. Caso o agente esteja na etapa de coleta, então ele faz uma varredura do ambiente para determinar se é o turno dele. Caso contrário para a execução da árvore pelo período de um segundo. Se for o turno do agente ele irá analisar os recursos que ainda não foram coletados e estão disponíveis, além de levar em consideração os recursos que ele necessita para a etapa de compra existe também um fator de aleatoriedade envolvido fazendo com que as jogadas do agente não se tornem previsíveis.

Na etapa de compra de hardware e instalação de serviços o agente entra em uma ramificação diferente da vertente de coleta, nesta ramificação é realizado uma verificação para assegurar que o agente possua espaço (racks) para acondicionar os servidores adquiridos, caso o mesmo não possua racks então ele adiciona ao menos um e então passa a tarefa seguinte, na qual ele verificará se tem os requisitos necessários para adquirir algum tipo de servidor. A escolha do servidor que será adquirido não depende só dos requerimentos de compra mas o agente também leva em consideração os serviços que ele tem disponível para instalar e com base nisso ele tenta escolher o servidor que melhor se encaixa. Caso não

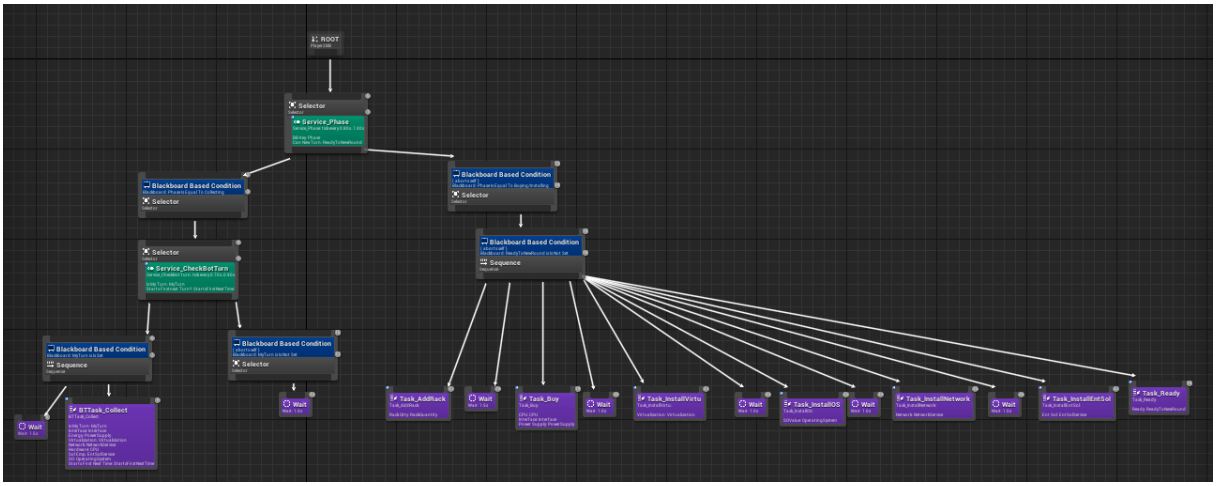

Figura 6.10: Diagrama de sequência representando a solicitação da criação de uma nova sessão. Fonte: autoria própria.

haja serviços para tomar como base no momento de sua escolha a Inteligência Artificial (IA) irá então usar um fator de aleatoriedade para auxílio na tomada de decisão.

Após a etapa de compra dos servidores, entra em execução a etapa de instalação de virtualizações, na qual o agente buscará verificar se existem servidores que pertençam a ele que suportem virtualização e ainda não foram ativados. Em seguida, se houve recursos para ativação ela é então realizada, caso contrário o recurso de virtualização é marcado como necessário para que no próximo round o agente tente priorizar a coleta do recurso.

A tarefa seguinte é a instalação de sistemas operacionais caso o agente possua recursos o suficiente e também hajam servidores dos quais lhe pertença e que ainda não possuem sistemas operacionais instalados ele irá realizar a instalação desses recursos, caso não possua o recurso para isso o agente então marca como prioridade para que no próximo *round* esse recurso passe a ser coletado. O mesmo comportamento ocorre para os serviços de rede e de solução empresarial, entretanto, a prioridade de instalação é maior para servidores que tem suporte a uma maior quantidade de serviços, fazendo assim com que a IA receba pontos sempre que possível por estar utilizando a capacidade plena dos servidores que possui.

6.10 Sumário

Neste capítulo foi explicado conceitos e aspectos importantes que foram levados em consideração no decorrer do desenvolvimento deste trabalho, no próximo capítulo será descrito e discutido como foram realizados as aplicações e avaliação do jogo Cabinet em sala de aula e a análise realizada com a utilização do octalysis framework.

Capítulo 7

Testes e Avaliação

Neste capítulo foi discutido sobre as aplicações do Cabinet realizadas em sala de aula e a avaliação do jogo realizada por meio da observação entre a interação jogo-jogador e jogador-jogador e com a utilização do Octalysis framework.

7.1 Aplicação em sala de aula

Ao longo da aplicação, 23 alunos jogaram a versão digital do jogo Cabinet e 10 alunos jogaram a versão analógica. As aplicações foram realizadas em duas turmas das disciplinas de Gestão de Sistemas e de Redes, do curso de Engenharia Informática do Instituto Politécnico de Bragança.

Antes de iniciar sua primeira partida, os alunos que jogaram a versão digital do Cabinet foram orientados a seguir um breve tutorial contido dentro do jogo, que explica as mecânicas envolvidas ao longo dos turnos e algumas dicas que podem ser fundamentais para auxiliar na vitória. Já os alunos que jogaram a versão analógica do Cabinet, contaram com um livro de regras para servir de base e orientação ao longo da partida.

Durante a aplicação do jogo Cabinet, um dos fatores mais analisados foi a interação entre os alunos. Por se tratar de um jogo que coloca ambos os alunos em um cenário competitivo, onde cada um tem que buscar construir o melhor data-center com uma

pré-determinada quantidade de turnos e os recursos que são coletados ao longo dos mesmos, o jogo foi capaz de causar euforia e comoção dos jogadores em busca da vitória. Mesmo aqueles que acabavam por perder, buscavam jogar novamente contra os mesmos adversários com o objetivo de os superar.

Foi notável a melhora de desempenho dos alunos ao longo das partidas, pois no início cada jogador ainda não está habituado com as mecânicas e uso dos recursos. Entretanto, assim que eles passam a perceber a importância de cada tipo de recurso, a competitividade entre os alunos aumenta. Isso é notável no momento em que eles passam a coletar os recursos não somente para uso próprio, mas muitas vezes para evitar que seu adversário obtenha vantagem ou mesmo o monopólio de algum recurso de hardware ou software que pode ser primordial para garantir a vitória.

Figura 7.1: Jogador comemorando sua primeira vitória no Cabinet. Fonte: autoria própria.

Durante a aplicação do Cabinet ficou evidente o quanto a aula ficou dinâmica e os alunos se envolveram no processo. Além disso, pôde-se obter diversas informações relevantes por meio de gravações das partidas realizadas que servirão para o melhoramento do jogo.

As gravações das partidas permitiram realizar a análise da frequência de cliques na tela, assim como o modo em que a interação ocorreu com a interface do jogo, e com base nisso, foi possível melhorar alguns aspectos, afim de deixar mais intuitivo algumas ações ao longo da partida e melhorando a interação jogador-jogo. Além disso, notou-se que nas primeiras partidas de cada aluno, o mesmo levava em torno de 3 a 4 rounds para compreender como os recursos são utilizados para a compra e instalação dos serviços em servidores. Só a partir daí que suas ações passaram a ser tomadas com base nas necessidades e buscando atingir sua condição de vitória.

Questionário

No final da aplicação os alunos foram convidados a responder ao questionário que serve de base para avaliação de algumas características presentes no Cabinet. Com os resultados obtidos foi possível tirar diversas conclusões acerca do jogo.

A Figura 7.2 representa o resultado obtido no questionário sobre a questão referente as mecânicas de jogo, onde, em uma escala de 1 a 5, o aluno deveria classificar o que achou das mecânicas, considerando 1 para não muito diferente do que eles já conheciam, e 5 sendo considerado uma experiência única e inovadora. Como é possível observar na Figura 7.2, a maioria dos alunos considerou ao menos a experiência como "diferente da maioria", podemos atribuir este resultado ao estilo que o Cabinet possui, pois a mistura entre tabuleiro e simulação por turnos dá ao aluno umas perspectiva diferente acerca dos elementos são envolvidos durante a partida.

Em um jogo por turnos um dos elementos mais importantes são as regras, pois elas que definem o que pode ou não ocorrer durante as partidas. As regras além de limitarem o comportamento, também podem impor desafios aos jogadores, tornando a jogabilidade mais cativante. Para mensurar o que os jogadores acharam das regras do Cabinet, foi colocado no questionário uma questão relativa a pontuação que eles dariam para as regras em uma escala de 1 a 10, considerando que 1 seriam regras "pobres" enquanto que 10 seriam regras "excelentes" e mais do que apropriadas para o estilo de jogo (Figura 7.3). Com isso foi possível constatar que a grande maioria dos jogadores classificou as regras

Figura 7.2: Resultados obtidos no questionário acerca das mecânicas de jogo. Fonte: autoria própria.

do jogo com a nota 7, representando algo entre “apropriado” e “excelente”.

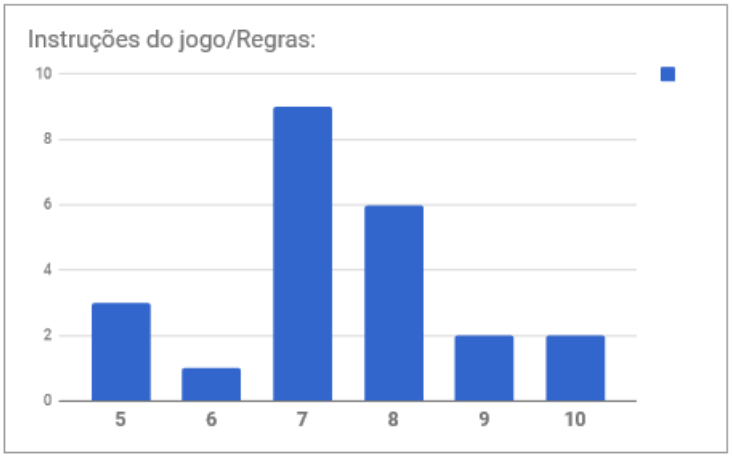

Figura 7.3: Resultados obtidos no questionário acerca das regras de jogo. Fonte: autoria própria.

Outro fator importante e decisivo para o jogo, e que deve ser levado em consideração tanto na etapa de game design quanto ao longo do desenvolvimento, é o tempo de duração para cada partida. Um tempo muito longo de partida pode não ser apropriado para um jogo a ser aplicado em sala de aula, além disso, há também o risco do jogo se tornar maçante e repetitivo. Enquanto que, por outro lado, um jogo com partidas de curta duração pode não usar completamente o potêncial de motivação e envolvimento que os jogos são

capazes de proporcionar. É necessário encontrar um meio termo, com a intenção de englobar as vantagens de jogos de curta duração e de longa duração, buscando minimizar suas desvantagens. Em média, durante a aplicação realizada as partidas do Cabinet duraram 15 minutos.

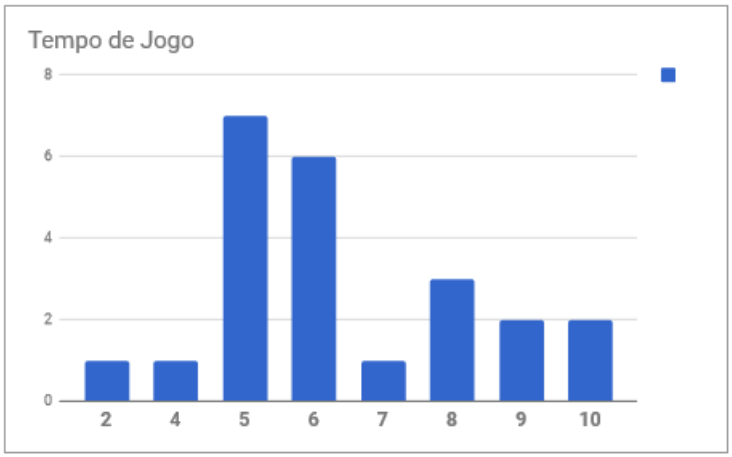

Figura 7.4: Resultados obtidos no questionário acerca da duração das partidas. Fonte: autoria própria.

Na Figura 7.4 é possível observar o resultado obtido no questionário acerca do tempo de jogo no Cabinet, em uma escala de 1 a 10, onde 1 significa "muito curto" e 10 significa "muito longo", os resultados estão majoritariamente distribuídos ao centro. O gráfico dá a entender que o tempo foi classificado entre apropriado e muito longo. Por meio da análise por observação da interação jogador-jogo pudemos notar que os alunos que optaram por iniciar uma partida sem seguir o tutorial inicial recomendado no início da aplicação, foram os que mais levaram tempo até concluir suas partidas, pois além de não entender completamente como o jogo funcionava, acabavam por não conseguir decidir com clareza as melhores ações a serem tomadas.

Outro fator interessante notado na análise da interação entre jogador-jogo foi que os jogadores que realizaram o tutorial no início da aplicação, em sua maioria, foram os que tiveram a vitória com o menor número de partidas jogadas. Por meio da Figura 7.5 pode-se ter uma noção sobre a quantidade de partidas que a amostra de alunos necessitou até conseguir sua primeira vitória.

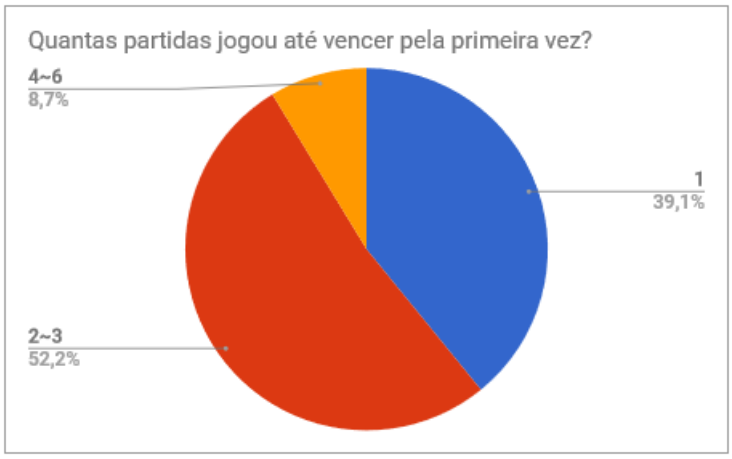

Figura 7.5: Resultados obtidos no questionário acerca da quantidade de partidas até conseguir a vitória. Fonte: autoria própria.

Um característica importante que os jogos no geral devem possuir é o fato de deixar claro quais são os objetivos acerca do jogo, o que o jogador deve fazer para alcançar a condição de vitória. Muitas vezes isso pode ser feito de maneira sutil e indireta, enquanto que existem abordagens mais incisivas que apontam ao jogador a direção da vitória. O Cabinet assume um papel diferente nesse sentido, pois o jogador pode aprender as condições básicas por meio do tutorial, ou mesmo aprender na prática no decorrer das partidas. Observando a Figura 7.6 podemos notar que a maioria dos alunos avaliaram esta característica como "excepcional" ou "incrível".

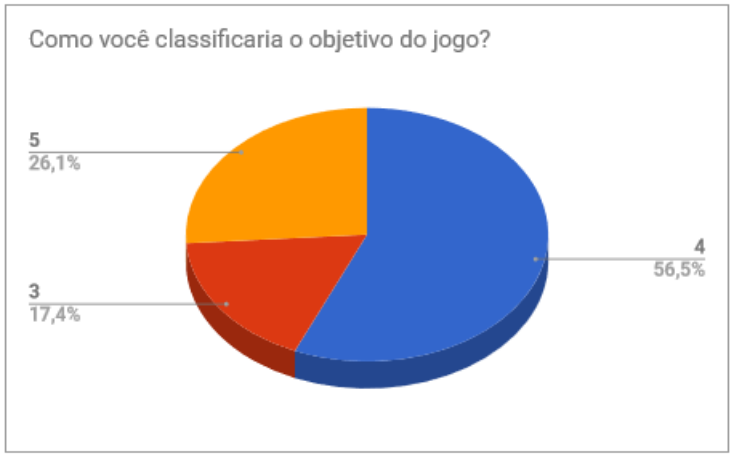

Figura 7.6: Resultados obtidos no questionário acerca da avaliação dos objetivos do Cabinet. Fonte: autoria própria.

Uma das preocupações ao longo do desenvolvimento do Cabinet, foi acerca da quantidade de opções possíveis de serem realizadas ao longo dos *rounds*, e pra isso foi também criada uma questão com o objetivo de fazer o aluno avaliar esse quesito.

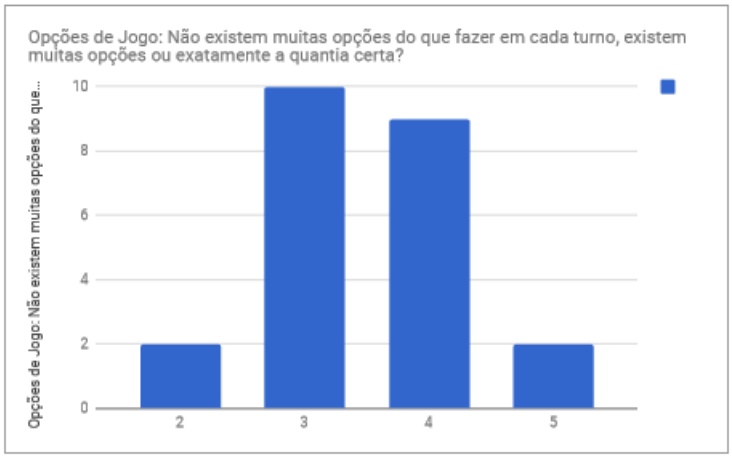

Figura 7.7: Resultados obtidos no questionário acerca da avaliação das opções de ação do Cabinet. Fonte: autoria própria.

Na Figura 7.7 fica evidente de que a quantidade de opções disponíveis, segundo o ponto de vista dos jogadores, é suficiente tendendo a muitas. Atribuímos esse resultado obtido às diversas possibilidades de combinações e versatilidade disponível para o usuário, que pode optar por realizar a instalação dos tipos de servidor que quiser, seguindo sua própria abordagem afim de garantir a maior quantidade de pontos possível.

Em comparação entre a versão física e digital a equipe notou que houve maior duração nas partidas do jogo físico. Atribuímos a essa diferença o fato de que todos os alunos que jogaram a versão física leram o manual de regras durante as primeiras partidas. Outra diferença notada entre as versões por meio da observação é que mesmo o jogo físico exigindo maior interação entre os alunos, durante a aplicação da versão digital foi capaz de notar mais facilmente as emoções e o espírito competitivo durante as partidas, se comparadas a versão analógica. Entretanto para que fosse possível mensurar a diferença real entre o jogo analógico e sua versão digital seria necessário um assunto mais aprofundado acerca desse assunto, tendo como foco aspectos que não foram levados em consideração durante a aplicação aqui citada.

7.2 Análise utilizando Octalysis Framework

Nesta seção será discutido e analisado alguns dos elementos e seus aspectos presentes no jogo digital Cabinet seguindo como base as diretrizes estabelecidas no Octalysis Framework.

Como é possível observar na Figura 7.8, somente duas das oito principais características do Octalysis não puderam ser preenchidas, pela ausência de elementos que busquem colocar o aluno em um estado de protagonismo, e também pela falta de elementos que se encaixem na categoria de avoidance.

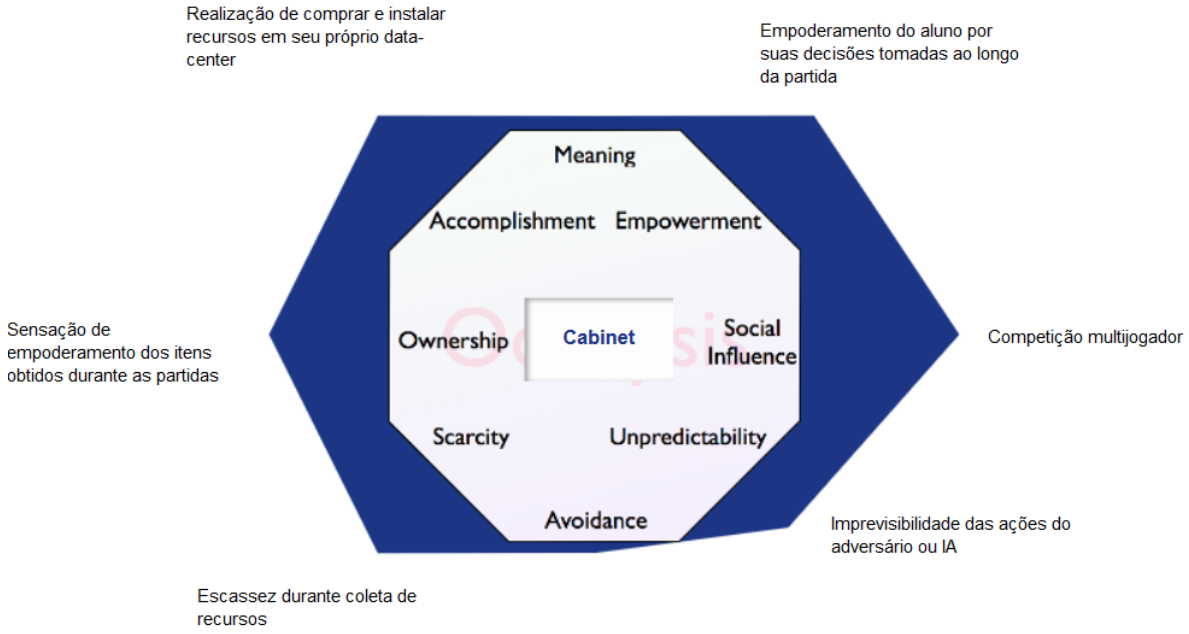

Figura 7.8: Gráfico do Cabinet gerado utilizando o Octalysis Framework. Fonte: autoria própria.

Pelo gráfico gerado (Figura 7.8) nota-se uma homogeneidade entre os elementos do lado esquerdo e direito do octógono, caracterizando um certo balanceamento entre motivadores intrínsecos e extrínsecos, sendo que há um apelo levemente maior para os motivadores extrínsecos, levando em consideração o grande foco que o jogo possui nos elementos adquiridos ao longo da partida, que acabam por definir e ser um dos fatores que garantem

a vitória.

O aspecto de realizações do Cabinet foca bastante nas jogadas e *hardware*/*software* comprados e instalados pelo usuário, essas realizações permitem que o jogador veja seu progresso ao longo da partida com base na quantidade de elementos instalados em seu data-center. O sentimento de posse é em partes englobado por essas mesmas características, na qual o aluno passa a perceber que aqueles objetos adquiridos são dele durante a partida e por meio da utilização deles em comparação com os itens obtidos pelo adversário é que se definirá o resultado final da partida.

Quando se trata de escassez no Cabinet, ela está fortemente ligada a influencia social quando a partida é realizada contra outro jogador ou ligada a fatores de imprevisibilidade quando a partida ocorre contra a inteligência artificial. No caso da influencia social isso se dá pelo fato de os jogadores começarem a realizar jogadas não somente para obter os recursos necessários, mas também para evitar que seu oponente consiga o recurso que ele precisa, o mesmo ocorre no caso da inteligência artificial, que tentará buscar os requisitos para as compras e instalações ao mesmo tempo em que também tenta evitar que o jogador consiga monopolizar recursos essenciais como fontes de energia e sistemas operacionais.

Empoderamento é uma característica muito forte em jogos que pretendem focar em motivadores intrínsecos, no Cabinet ela está presente por meio das escolhas do jogador que acabam por definir o desfecho das partidas. As escolhas vão desde qual o hardware mais apropriado para se comprar, qual seriam os melhores recursos a serem coletados e quando é o momento propício para expandir a capacidade de servidores.

7.3 Problemas encontrados

Durante as aplicações do Cabinet em sala de aula alguns problemas foram encontrados pelos alunos. Um dos problemas encontrados foi durante a aquisição de um rack adicional, caso o aluno efetuasse um click duplo sobre a posição na qual ele deseja adicionar o rack, o evento adicionaria a quantidade de racks do aluno o valor de 2 unidades. Caso isso acontecesse na inserção do primeiro rack impediria a continuação do jogo pelo fato

da interface sumir, não permitindo que o aluno finalizasse sua jogada. A solução para esse problema foi a implementação de uma condicional que garante que o evento só será chamado 1 vez em um determinado periodo de tempo, evitando assim que uma sequência de clicks desencadeie múltiplas chamadas do método de ativação e permitindo validar a ação seguinte.

Outro problema que ocorreu durante a aplicação, apesar de não ser um problema grave ao ponto de impedir a continuação da partida, foi o fato de que quando os turnos se encerravam rapidamente, os trabalhadores que ainda tinham tarefas a serem realizadas acabavam por empilhar as novas ações em suas listas de afazeres e, caso isso acontecesse em uma caixa contendo recursos, o mesmo executava as animações e o cenário ficava com as caixas abertas devido a ação decorrente do turno anterior, mesmo que a caixa possuísse ainda recursos a serem coletados. Este problema foi solucionado por meio da implementação de métodos que verifiquem a existência de ações pendentes na lista de cada trabalhador antes de avançar para um novo turno. Para garantir que não houvessem discrepâncias visuais sempre que um novo turno se inicia, as listas de ações de cada trabalhador são esvaziadas com o objetivo de interromper qualquer coleta futura de recurso que pertencia a um turno passado e, com isso, foi necessário fazer com que a escolha do recurso a ser coletado

O maior problema encontrado durante a aplicação foi a falta de memória na execução do servidor gerenciador de sessões. Ao alcançar exatamente 4 sessões ativas (8 jogadores) o servidor gerenciador não possuía mais memória para que suas *threads* pudessem operar e lidar com as requisições de novas conexões, negando assim toda e qualquer nova conexão. A princípio, durante a aplicação não foi possível constatar com plena certeza de que o problema era esse e portanto foi necessário utilizar a criação de partidas em rede local. Devido a configurações como *firewall* e *softwares* como antivirus alguns computadores não conseguiram criar sessões visíveis ou encontrar sessões disponíveis. Após a aplicação foi possível constatar que a solução para o problema do servidor gerenciador de sessões foi o aumento da quantidade de memória disponível para a máquina virtual do Java.

Durante a aplicação e a construção da Figura 7.8 foi constatado a ausência de elementos

que foquem na categoria de *meaning* e *avoidance* definidas pelo octalysis framework. Apesar de os aspectos e mecânicas do Cabinet proporcionarem elementos para 6 das 8 categorias, foi notado que o sentimento de perca estava indiretamente ligado a exposição social do indivíduo. Quanto a categoria de *meaning*, a ausência de um enredo dificulta por partes o desenvolvimento desta categoria, entretanto, uma solução proposta seria a introdução de missões e uma pequena história envolvendo cada uma dessas missões, onde seria possível fazer com que o jogador alcançasse uma certa pontuação ou a instalação de uma certa quantidade de determinado hardware ou software em uma partida. Isso adicionaria um propósito específico ao jogo, protagonizando o aluno como o indivíduo responsável por solucionar determinado problema ou demanda.

7.4 Sumário

No próximo capítulo serão descritas as conclusões tiradas durante o desenvolvimento e aplicação deste trabalho, as dificuldades encontradas e também os trabalhos futuros.

Capítulo 8

Conclusões e Trabalhos futuros

Neste capítulo são discutidas as conclusões tiradas sobre o desenvolvimento e aplicação do jogo Cabinet em sala de aula, as metodologias aplicadas, as análises realizadas e também os trabalhos a serem realizados no futuro com o objetivo de acrescentar e também melhorar elementos que o Cabinet possui.

Jogos digitais são jogados pela grande maioria dos jovens, apesar de que esses jogos não costumam ser focados essencialmente na educação, entretanto, a maior parte dos jogos consegue despertar a motivação no seu público alvo e também acabam por contribuir com a aprendizagem de maneira indireta. Jogos vêm sendo utilizados frequentemente em práticas pedagógicas ativas de ensino como uma ferramenta de auxílio ao aprendizado principalmente pelo aspecto dinâmico que eles possuem, enredo e mecânicas que envolvem o jogador e o coloca como protagonista e responsável pelo desfecho do jogo, motivando-o e prendendo sua atenção.

Trazer novas ferramentas para auxiliar no ensino na sala de aula com o objetivo de aumentar a motivação dos alunos e despertar também a busca pelo conhecimento nos dias de hoje tem-se tornado difícil. Muitas vezes os professores têm de encontrar formas criativas para cativar atenção do aluno, disperso por algum dispositivo eletrônico. O Cabinet, em sua versão digital, procura utilizar os recursos tecnológicos que estão a disposição e que antes poderiam ser tratadas como distrações, para que agora se tornem verdadeiras ferramentas de auxílio no ensino.

Ao longo do desenvolvimento do Cabinet houve inúmeras incertezas sobre a maneira mais adequada de realizar as adaptações necessárias entre a versão física e a versão digital. Grande parte do desenvolvimento da versão digital foi feita levando em consideração aspectos que caracterizam o Cabinet. O ambiente e os objetos contidos nele, em sua maioria, sofreram forte influência dos objetos reais.

Uma análise qualitativa realizada por meio das observações e entrevistas sugere que o entendimento dos elementos presentes no jogo e a velocidade das interações entre jogo-jogador e jogador-jogador podem variar com base no estilo de cada jogador e na sua facilidade de aprendizado e adaptação ao ambiente do jogo.

Além do propósito de ajudar a aprender, o Cabinet ainda faz com que os alunos interajam entre si e se divirtam realizando algo que pertence ao escopo da disciplina e tornará a experiência de estudo deles mais agradável. Cabinet não se trata apenas de um jogo, mas de uma maneira diferenciada e dinâmica de ensinar elementos que pertençam a disciplina de Gestão de Sistemas e de Redes.

A área da educação está em constante evolução e cada vez mais tenta buscar a inovação nas práticas de ensino, seja utilizando novas tecnologias ou por meio de melhoramentos em técnicas já consolidadas. Apesar das inúmeras adversidades e fatores que muitas vezes acabam por distrair ou desmotivar os alunos, novas metodologias e técnicas de engajamento proporcionam maneiras diferenciadas e dinâmicas de fazer com que o aluno participe em sala de aula e sinta-se motivado a aprender.

O aprendizado baseado em jogos é uma área que ainda não foi completamente explorada mas já apresenta um grande potencial. Com maiores estudos sobre, será possível encontrar melhores maneiras de se otimizar o processo de ensino por meio dos jogos. O Cabinet possui um grande potencial a ser explorado, mostrou ser uma boa ferramenta no ensino de aspectos gerenciais de um data-center, entretanto, por meio de novas mecânicas o seu escopo pode ser ampliado, podendo ser usado para ensino de arquiteturas de redes ou sistemas operacionais. Isso é um pequeno exemplo de áreas as quais o Cabinet poderia focar, entretanto, a mesma premissa é válida também para a criação de outros jogos que podem vir a se tornar ótimas ferramentas de auxílio ao ensino.

Bibliografia

[1] C. Dichev e D. Dicheva, *Gamifying education: what is known, what is believed and what remains uncertain: a critical review*, 1. International Journal of Educational Technology in Higher Education, 2017, vol. 14, p. 9.

[2] G. Zichermann e C. Cunningham, *Gamification By Design*. 2011, p. 208. arXiv: `arXiv:1011.1669v3`.

[3] A. C. T. A. Machado, S. É. Rufini, A. G. Maciel e J. A. Bzuneck, "Estilos motivacionais de professores: preferência por controle ou por autonomia", *Psicologia: Ciência e Profissão*, vol. 32, n.º 1, pp. 188–201, 2012.

[4] N. A. N. Berbel, "As metodologias ativas e a promoção da autonomia de estudantes", *Semina: Ciências Sociais e Humanas*, vol. 32, n.º 1, pp. 25–40, 2011.

[5] R. Campbell, J. Okey, I. J. Quitadamo, M. J. Kurtz, G. Paul, R. and Nosich e B. S. Rathburn, "Curriculum & Leadership Journal _ Skills for the 21st Century_ teaching higher-order thinking", *CBE - Life Sciences Education*, vol. 6, pp. 1–15, 1993.

[6] M. A. Behrens e A. R. R. Lopes, "Ação docente no ensino superior : reflexões sobre paradigmas educacionais inovadores na prática pedagógica", pp. 1–16, 2014.

[7] M. Amélia e S. Franco, "Práticas pedagógicas de ensinar-aprender : por entre resistências e resignações", pp. 601–614, 2015.

[8] M. A. Behrens e A. L. T. Oliari, "A evolução dos paradigmas na educação: do pensamento Científico tradicional a complexidade", *Dialogo educacional*, vol. 7, n.º 22, www.portaleducacao.com.br, 2007.

[9] A. Correia e P. Dias, "A evolução dos paradigmas educacionais à luz das teorias curriculares", *Revista Portuguesa de Educação*, vol. 11, n.º 1, pp. 113–122, 1998.

[10] M. Bhardwaj e S. Sharma, "Methodology in Engineering Education", *2013 IEEE International Conference in MOOC, Innovation and Technology in Education (MITE)*, pp. 185–189, 2013.

[11] M. Huang, D. Malicky e S. Lord, "Choosing an optimal pedagogy: A design approach", *Proceedings - Frontiers in Education Conference, FIE*, pp. 1–6, 2006.

[12] M. C. Moraes, "O paradigma educacional emergente", *Século XXI*, 2004.

[13] R. Yus, *Educação integral: uma educação holística para o século XXI. proposta para o currículo escolar*, Porto Alegre, 2002.

[14] P. F. Martins e S. T. Filipak, "a Sala De Aula Invertida Como the Flipped Classroom As a Convergent Methodology To the Paradigm of Complexity", pp. 118–135, 2016.

[15] E. Triantafyllou e O. Timcenko, "Introducing a flipped classroom for a statistics course: A case study", *2014 25th EAEEIE Annual Conference (EAEEIE)*, pp. 5–8, 2014. URL: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6879373`.

[16] T. S. Borges e G. Alencar, "Metodologias ativas na promoção da formação crítica do estudante: o uso das metodologias ativas como recurso didático na formação crítica do estudante do ensino superior", *Cairu em Revista*, vol. 3, n.º 4, pp. 119–143, 2014.

[17] R. L. Nagem, D. D. O. Carvalhaes e J. Dias, "Uma proposta de metodologia de ensino com analogias", *Revista portuguesa de educação*, vol. 14, n.º 1, pp. 197–213, 2001.

[18] M. W. L. and Amélia e M. A. Behrens, "Paradigmas educacionais e o ensino com a utilização de mídias Educational paradigms and education with the use of media", pp. 245–270, 2010.

[19] R. I. Busarello, *Gamification Principios e Estrategias*. 2016.

[20] F. S. de Castro, "A gamification framework as a collaboration motivator for software development teams", 2016.

[21] Y. Vianna, M. Vianna, B. Medina e S. Tanaka, *Gamification, Inc.: Como reinventar empresas a partir de jogos*. 2013, p. 116.

[22] A. Fallis, "Gamificação Nas Práticas Pedagógicas: Um Desafio Para a Formação De Professores Em Tempos De Cibercultura", *Journal of Chemical Information and Modeling*, vol. 53, n.º 9, pp. 1689–1699, 2013.

[23] R. Pedro Lopes, "Gamification As a Learning Tool", *International Journal of Developmental and Educational Psychology. Revista INFAD de Psicología.*, vol. 2, n.º 1, p. 565, 2016, ISSN: 0214-9877. DOI: 10.17060/ijodaep.2014.n1.v2.473. URL: http://www.infad.eu/RevistaINFAD/OJS/index.php/IJODAEP/article/view/473.

[24] L. M. F. V. R. U. C. R. B. T. Vanzin, *Gamificação*. 2014, p. 302.

[25] M. Csikzentmihalyi, "Flow: the psychology of optimal experience", *USA: Harper Perennial Modern Classics edition*, 1990.

[26] J. Schell, *The art of game design*. 2008, vol. 1, p. 489.

[27] Y.-K. Chou, *Actionable Gamification - Beyond Points, Badges and Leaderboards*. 2014.

[28] A. C. T. Klock, M. F. de Carvalho, B. E. da Rosa e I. Gasparini, "Análise das técnicas de Gamificação em Ambientes Virtuais de Aprendizagem", *RENOTE - Revista Novas Tecnologias na Educação*, vol. 12, n.º 2, pp. 1–10, 2014. URL: http://seer.ufrgs.br/index.php/renote/article/view/53496.

[29] R. Hunicke, M. LeBlanc e R. Zubek, "MDA: A Formal Approach to Game Design and Game Research", *Workshop on Challenges in Game AI*, pp. 1–4, 2004.

[30] Epic Games, *Blueprints Visual Scripting*. URL: `https://docs.unrealengine.com/en-us/Engine/Blueprints`.

[31] P. Bhawar, N. Ayer e S. Sahasrabudhe, "Methodology to create optimized 3D models using blender for android devices", *Proceedings - 2013 IEEE 5th International Conference on Technology for Education, T4E 2013*, pp. 139–142, 2013. DOI: `10.1109/T4E.2013.41`.

[32] Hawlitschek, Florian and Jansen, Lars Erik and Lux, Ewa and Teubner, Timm and Weinhardt, Christof, "Colors and trust: The influence of user interface design on trust and reciprocity", *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2016.