INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE - IFSUL, *CAMPUS* PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

ALISSON ALBERTI TRES

ESTUDO DE TÉCNICAS E PROCESSOS REFERENTES AO DESENVOLVIMENTO DE JOGOS ELETRÔNICOS MULTIPLATAFORMA

Lisandro Lemos Machado

PASSO FUNDO, 2015

ALISSON ALBERTI TRES

ESTUDO DE TÉCNICAS E PROCESSOS REFERENTES AO DESENVOLVIMENTO DE JOGOS ELETRÔNICOS MULTIPLATAFORMA

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-Rio-Grandense, *Campus* Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Lisandro Lemos Machado

PASSO FUNDO, 2015

ALISSON ALBERTI TRES

ESTUDO DE TÉCNICAS E PROCESSOS REFERENTES AO DESENVOLVIMENTO DE JOGOS ELETRÔNICOS MULTIPLATAFORMA

Trabalho de Conclusão de Curso aprovado em ____/____/____ como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

_____

Prof. Msc. Lisandro Lemos Machado

Orientador

_____

Prof. Msc. Élder Francisco Fontana Bernardi

Avaliador

_____

Prof. Esp. Carmen Vera Scorsatto

Avaliadora

_____

Prof. Dr. Alexandre Tagliari Lazzaretti

Coordenação do Curso

PASSO FUNDO, 2015

AGRADECIMENTOS

Primeiramente, agradeço aos meus pais, Vlademir José Tres e Cleuza Fátima Alberti, por todo carinho, dedicação, preocupação e cuidados proporcionados no decorrer de toda a minha vida.

Agradeço ao meu professor orientador, Lisandro Lemos Machado, por ter me acompanhado no decorrer desta trajetória, sempre compartilhando de sua sabedoria e aconselhamentos, assim como também agradecer pela sua infinita paciência e compreensão. Foi através de sua visão que este trabalho pode alcançar a forma em que se encontra.

Deixo meus agradecimentos à Gabriela Fritzen por ter contribuído ao trabalho realizando os desenhos conceituais para os personagens da Feiticeira e do Lorde das Trevas, sendo grato pela disposição e pelo empenho que você dedicou a eles.

Também agradeço a Giovani Elisio da Silva e Fernando Winckler Simor pela compreensão e oportunidades oferecidas, assim como também agradeço a todos os demais colegas da Parceria Sistemas pelo apoio e excelente convivência.

Por fim, exalto o meu agradecimento a todos os estimados colegas, professores e servidores do Instituto Federal Sul-Rio-Grandense Campus Passo Fundo, com os quais, em menor ou em maior grau, convivi no decorrer da realização desse curso, fazendo parte desta caminhada e protagonizando muitos dos melhores momentos de minha vida até então.

“Um game não é algo que possa ser feito por uma pessoa apenas. Bem, é possível que um criador de jogos realmente dedicado possa concluir um game por conta própria. Mas mesmo assim ele ainda precisaria de um jogador. Games crescem e amadurecem quando estes são criados, jogados e transmitidos várias e várias vezes.”

Shigesato Itoi, roteirista e designer responsável pelo homônimo game *Earthbound,* lançado originalmente para o Super Nintendo

RESUMO

Jogos eletrônicos não somente conseguem se destacar como sendo um dos meios de entretenimento mais populares existentes atualmente, mas também conseguem ser representantes da complexidade que pode ser alcançada pelos softwares contemporâneos, carregando consigo um potencial único para o desenvolvimento de aplicações em diversas áreas.

Em virtude de sua relevância, este trabalho propõe aprofundar-se nos fundamentos necessários para se realizar o desenvolvimento de jogos eletrônicos, assim como analisar e experimentar duas tecnologias de caráter distinto, as quais viabilizam esse tipo de desenvolvimento. Em particular, serão priorizados objetos de estudo que apresentem um caráter multiplataforma: tecnologias que possibilitam com praticidade disponibilizar um jogo eletrônico a um maior número de hardwares distintos e, consequentemente, maximizar o número de jogadores que um jogo eletrônico poderá alcançar.

A partir dos estudos realizados, protótipos criados com as tecnologias abordadas serão alvos de uma avaliação comparativa para determinar o quão aptas estas ferramentas são para o desenvolvimento de jogos multiplataforma.

Palavras-chave: jogos eletrônicos multiplataforma; design de games; Game Maker: Studio; desenvolvimento de games em HTML5.

ABSTRACT

Electronic games not only stands out as one of the most popular current means of entertainment. They can also represent the complexity that can be achieved by contemporary software, carrying within themselves a unique potential for developing applications in a variety of subjects.

In regard of its relevance, this paper proposes to study the necessary foundation for actualizing the development of electronic games, as well as to analyze and to experiment two distinct technologies that enable this kind of development. In particular, it will prioritize study objects that feature a multiplatform character: technologies that easily allow the distribution of an electronic game among a wider number of distinct hardware and, consequently, maximize the number of players that could be reached by an electronic game.

From the studies carried out, prototypes created with these technologies will be aimed at a comparative evaluation to determine how fit these tools are for multiplatform game development.

Key words: multiplatform games; game design; Game Maker: Studio; HTML5 game development.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface – Interface de Programação de Aplicativos
CLR – Common Language Runtime – Linguagem Comum em Tempo de Execução
CSS – Cascading Style Sheets – Folha de Estilo em Cascata
DDG – Documento de Design de Game
FPS – Frames per Second – Quadros por Segundo
HTML – HyperText Markup Language – Linguagem de Marcação de Hipertexto
IDE – Integrated Development Enviroments – Ambientes de Desenvolvimento Integrado
GML – Game Maker Language
JVM – Java Virtual Machine – Máquina Virtual do Java
MMO – Massively Multiplayer Online – Massivamente Multijogadores Online
RPG – Role-Playing Games – Jogos de Representação de Papéis
UML – Unified Modeling Language – Linguagem de Modelagem Unificada
W3C – World Wide Web Consortium – Consórcio da World Wide Web
WHATWG – Web Hypertext Application Technology Working Group – Grupo de Trabalho de Aplicação de Tecnologias de Hipertexto Web
YYC – YoYo Games Compiler – Compilador da YoYo Games

SUMÁRIO

1 INTRODUÇÃO

O século XX caracterizou-se por um inédito desenvolvimento tecnológico, uma evolução sem precedentes na história da humanidade. Aparelhos eletrônicos como rádios e televisores se consagraram em meio à cultura do homem contemporâneo. A próxima grande inovação que viria a surgir seria protagonizada por máquinas chamadas de computadores.

Através dos recursos oferecidos por eles, os computadores mostraram-se capazes de realizar cálculos complexos e de administrar e gerenciar informações. Com suas capacidades inovadoras, os computadores tiveram um contínuo processo de aprimoramento. Isso permitiu que suas funcionalidades expandissem progressivamente. Enquanto estes progrediam para se tornarem cada vez mais poderosos e invadir as casas dos cidadãos comuns, paralelamente ocorria o surgimento de outra tecnologia que também ganharia grande destaque: os jogos eletrônicos.

No cenário atual, os tais chamados *games*, tradução de “jogos” em inglês e termo comumente utilizado para se referir a jogos eletrônicos, representam um dos meios de entretenimento mais populares existentes. É difícil falar do surgimento destas aplicações sem mencionar a trajetória dos computadores, pois estes surgiram muito antes de haverem consoles de videogames e máquinas de fliperama. Conforme relatado por Novak,

> os primeiros passos do setor foram dados em departamentos de pesquisa de universidades, laboratórios, instalações militares e por fornecedores de produtos de defesa. Nas bases militares, games eletrônicos eram oferecidos aos recrutas para distraí-los dos rigores do treinamento básico. Enquanto isso, alguns estudantes, programadores, professores e pesquisadores de instituições acadêmicas e governamentais, insones e com excesso de trabalho, transformavam seus computadores mainframe em máquinas de jogos como uma maneira de relaxar de suas tarefas tradicionais de pesquisa [...]. Trabalhando de madrugada, esses pioneiros deram inicio ao que se tornaria uma das formas de entretenimento mais irresistíveis da História. (2010, p. 4).

Conforme ocorriam avanços no desenvolvimento da computação e da eletrônica, os jogos eletrônicos também trilhavam seu caminho junto a eles, até que eventualmente tornaram-se agentes de profundos impactos culturais, ao passo que

os computadores existentes nas bases militares e nas universidades seriam convertidos em aparelhos de jogos que marcariam presenças nos fliperamas e nas salas de estar.

Com o passar do tempo, o ritmo de tais evoluções tecnológicas continuou acelerando. Os frutos que esta jornada proporcionou estão à nossa disposição: computadores e dispositivos portáteis capazes de se comunicarem entre si, participantes de uma rede mundial na qual tudo o que acontece no mundo é registrado e compartilhado entre as pessoas. Eles não só revolucionaram a maneira de como o homem se comunica e trabalha com informações, mas também transformaram a maneira como ele se diverte. E os games são fortes representantes deste fenômeno. Contando com uma indústria cujos investimentos e lucros ultrapassam os de grandes produções hollywoodianas e com uma vasta variedade de propostas e personagens carismáticos, é cada vez mais evidente o potencial que estas aplicações interativas possuem. Potencial este que merece ser estudado e aprofundado, em virtude de sua grande relevância.

1.1 MOTIVAÇÃO

Percebendo como os jogos eletrônicos vêm conquistando um público alvo cada vez mais crescente e abrangente, observa-se o grande potencial que eles possuem. Atualmente, o número de pessoas que recorrentemente escolhem os games como forma de entretenimento é crescente, sendo possível executar esse tipo de aplicação em diferentes dispositivos, dentre eles consoles de vídeo game, os quais são dedicados para aplicações deste tipo, computadores e também em smartphones e tablets.

A recente revolução dos dispositivos móveis, que tem os tornado cada vez mais equivalentes a computadores completos que cabem na palma da mão, e das tecnologias de software estabelecem plataformas propícias ao desenvolvimento de jogos eletrônicos. A disponibilidade de recursos e interfaces para rodar aplicações, sendo um exemplo disso os navegadores de internet que estão se tornando capazes de executar aplicações cada vez mais robustas, possibilitam aos jogos eletrônicos crescerem tanto em complexidade como em destaque.

A questão da portabilidade, propiciada pelos dispositivos móveis, vem influenciando os paradigmas existentes no desenvolvimento de aplicações. No cenário atual, têm-se os recursos para que uma mesma ação possa ser realizada

através de inúmeros meios. Dependendo do objetivo de um determinado projeto, a portabilidade se mostra como sendo um aspecto de suma importância para que este possa vir a se tornar um sucesso. A conveniência de que em determinado momento se possa usar uma aplicação por meio de um smartphone e, em seguida, retomar ao seu uso em um computador é de um imensurável potencial, inclusive sendo algo que pode ser explorado pelo desenvolvimento de jogos eletrônicos.

Os games têm alcançando um número cada vez maior de usuários, apresentando vasta variedade em estilos e objetivos. A popularização tornou possível para muitos observar estes jogos como sendo mais do que meros meios de entretenimento, estimulando o pensamento de como estes poderiam trazer novas perspectivas para o desenvolvimento físico, psicológico e educacional do ser humano.

Certamente jamais houve momento mais promissor para engajar no estudo do desenvolvimento de jogos. É possível compreender o alcance dos jogos eletrônicos ao observar a Tabela 1, que contém as estimativas de vendas alcançadas por alguns dos mais populares consoles de videogame. Com a tal popularização que este meio vêm conquistando, o mercado encontra-se aquecido e receptivo a novidades. Assim, "o esforço se justifica pelo fato de o mercado nacional movimentar mais de R$ 6 bilhões em venda de consoles e games" (VARELLA; MARCEL, 2013).

Tabela 1: Estimativas de vendas totais de alguns dos mais populares consoles de videogame recentes

Console de Vídeo Game	Estimativa de unidades vendidas até 18/06/2015
Nintendo Wii	101.17 milhões de unidades
PlayStation 3 (PS3)	85.67 milhões de unidades
Xbox 360 (X360)	84.84 milhões de unidades
PlayStation 4 (PS4)	22.68 milhões de unidades
Xbox One (XOne)	12.63 milhões de unidades

Fonte: http://www.vgchartz.com/analysis/platform_totals/

Cada vez mais brasileiros estão se tornando consumidores de jogos, sejam eles para computadores, dispositivos móveis ou consoles de vídeo game, fortalecendo a presença dos games na cultura brasileira e construindo um promissor mercado consumidor. "O mercado de videogames no Brasil já fatura mais do que no Reino Unido, Alemanha e Espanha, afirmou [...] a consultoria GFK durante a prévia da feira de jogos Brasil Game Show (BGS) 2013" (PETRÓ; BRUNO, 2013). A nível mundial, Yung (2013) notifica que a indústria de videogames movimenta em torno de US$ 66 bilhões por ano, destacando que só a receita gerada a partir de jogos para dispositivos móveis gerava a expectativa de crescer cerca de 38% em 2013, totalizando US$ 8 bilhões. Um notável exemplo ocorreu durante o lançamento do game Grand Theft Auto V para o Playstation 3 e Xbox 360, o qual "vendeu 11,21 milhões de cópias em suas primeiras 24 horas, o que gerou 815,7 milhões de dólares de lucro. A marca de 1 bilhão foi ultrapassada após três dias" (CASTILHO, 2013).

A criação de um jogo eletrônico trata-se de um complexo processo que ocorre em diversas etapas e envolve a aplicação de diversos tipos de conhecimentos. Dada a variedade existente no mercado, possibilitar que esta aplicação seja capaz de rodar em diversos dispositivos é algo a ser pensado por possibilitar a expansão do público alvo que a aplicação poderá alcançar, embora isso também implique no aumento da complexidade do processo de desenvolvimento.

Nesse sentido, no decorrer deste trabalho será feita a contextualização de jogos eletrônicos e das tecnologias que viabilizam o desenvolvimento deste tipo de aplicativo em múltiplas plataformas. Os requisitos necessários ao desenvolver games também serão abordados para que, em seguida, possa-se experimentar a criação de um jogo. Isso ocorrerá tanto de forma conceitual, descrevendo o design que elabora um jogo eletrônico, quanto de forma prática, criando protótipos utilizando tecnologias capacitadas para este propósito. Os resultados obtidos a partir do desenvolvimento destes protótipos serão avaliados, comentando sobre as capacidades destas ferramentas e como elas impactaram neste processo.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Realizar um estudo e implementação das práticas e métodos relacionadas ao desenvolvimento de jogos, com enfoque em desenvolvimento multiplataforma.

1.2.2 Objetivos específicos

- Conceituar jogos eletrônicos, determinando seus principais aspectos e características;
- Pesquisar e estudar os processos e requisitos que envolvem o desenvolvimento de jogos eletrônicos;
- Realizar um estudo de metodologias e ferramentas de desenvolvimento de jogos eletrônicos, dando preferência a ferramentas que apresentem suporte a múltiplas plataformas;
- Aplicar os conhecimentos adquiridos na criação de protótipos de jogos eletrônicos multiplataforma utilizando diferentes ferramentas;
- Analisar o procedimento que envolveu o desenvolvimento dos protótipos e os resultados obtidos.

2 JOGOS ELETRÔNICOS MULTIPLATAFORMA

Passados alguns milhares de anos em evolução e já tendo dominado as habilidades necessárias para garantir sua sobrevivência, os seres humanos poderiam dispor de seu tempo para desenvolver suas capacidades sociais e culturais. Em meio a esse desenvolvimento surgiram as atividades as quais nós nos referimos como sendo os jogos.

Amaral e Paula (2007) remeteram a definição dada pelo historiador Johan Huizinga de que um jogo

> é uma atividade ou ocupação voluntária, exercida dentro de certos e determinados limites de tempo e espaço, seguindo regras livremente consentidas, mas absolutamente obrigatórias, dotado de um fim em si mesmo, acompanhado de um sentimento de tensão e alegria e de uma consciência de ser diferente da vida cotidiana. (p. 323).

Vianna (2013) também citou Huizinga ao atestar que “o ato de jogar está inserido nas mais diversas relações sociais, tais como a política, o trabalho, a poesia e mesmo a natureza” (p.15). Na medida em que se passou a buscar uma melhor compreensão do comportamento humano, tão logo se tomou consciência do quão relevante a prática de jogar é, a curiosidade dos estudiosos seria atiçada pelo desejo de avaliar o quão impactantes os jogos seriam para as pessoas. Ao realizar um estudo para avaliar a relevância dos jogos no processo de desenvolvimento humano durante a infância, Mattos e Faria (2011) destacaram que

> os jogos favorecem o domínio das habilidades de comunicação, nas suas várias formas, facilitando a auto-expressão. Encorajam o desenvolvimento intelectual por meio do exercício da atenção, e também pelo uso progressivo de processos mentais mais complexos, como comparação e discriminação; e pelo estimulo à imaginação. Todas as vontades e desejos das crianças são possíveis de serem realizados através do uso da imaginação, que a criança faz através do jogo. [...] Esses são alguns dos aliados na tarefa de desenvolver a criatividade e a independência das crianças. Segundo especialistas esses instrumentos dão possibilidades de formar pessoas independentes, capazes de recriar situações e não apenas repetir o que aprendem. (p. 3).

Sendo assim, podemos perceber como os jogos sempre exerceram um importante papel no decorrer dos vários anos da evolução humana, afinal, “ao longo

dos séculos, praticamente todas as civilizações conhecidas estiveram associadas a algum tipo de competição importante para a estruturação social da comunidade a qual pertenciam" (VIANNA et al., 2013, p.14). Muito antes do surgimento dos meios digitais, a humanidade desfrutaria de jogos através da utilização de cartas, dados, peças ou até mesmo de simplesmente papel e caneta. Muitos destes surgiram não somente para distrair, mas também para exercitar o raciocínio, sendo o caso de jogos de tabuleiro como o Chaturanga, originário da Índia no inicio do Século VI, cuja popularidade se disseminou pelo globo ao decorrer dos séculos, eventualmente originando o Xadrez como o conhecemos atualmente. Desde meados do século XIII, com a evolução das técnicas de impressão e com as revoluções culturais e artísticas, a criação e utilização de cartas disseminou por toda a Europa a prática de jogos que as utilizassem, colocando à prova a astúcia e a sorte de seus jogadores.

Diferentes tipos de cartas e diferentes tipos de regras eventualmente estabeleceriam uma grande variedade destes jogos, tais quais como o Poker ou o Truco, que até os dias atuais são amplamente apreciados e difundidos. A prática de atividades físicas também evoluiu para a composição de regras e objetivos, originando os tão diversos esportes que são praticados e adorados por incontáveis pessoas ao redor do mundo. Outro fascinante exemplo está no trabalho de Gary Gygax e Dave Arneson, que estabeleceram o padrão para os jogos de RPG em Dungeons & Dragons, criando um estilo de jogo complexo, mas simples de ser executado, voltado para aqueles cuja imaginação os possibilitem ir a mundos de fantasia e aventura.

Com o advento da computação protagonizado pelo Século XX, tão logo que os computadores passaram de poderosas calculadoras gigantescas para dispositivos menores que pudessem ser dispostos em uma sala de estar, com uma tela e apetrechos para se exibir e inserir dados com facilidade, os humanos logo estariam usando essas máquinas para manifestar o seu tão natural comportamento de jogar. Eis que surgem os jogos eletrônicos.

2.1 Jogos Eletrônicos

O elemento chave que difere um jogo eletrônico de um software qualquer está centrado no conceito de simulação, seja de um elemento real em particular ou de toda uma realidade própria, e na interação que o usuário possui com tal elemento simulado. Em sua essência, um jogo é um "sistema onde o jogador se engaja em um

conflito artificial, definido por regras, que resultam em um resultado quantificativo" (SALEN apud OLIVEIRA, 2012, p.10). De tal modo, jogos eletrônicos tendem a ser constituídos com base em regras e objetivos, os quais irão determinar como procederá a interação do usuário com o mesmo.

Aprofundando neste raciocínio, a caracterização de uma aplicação como sendo um jogo eletrônico é estabelecida através da experiência que essa interação com o seu universo, regras e objetivos são capazes de proporcionar ao jogador. Para Schuytema,

> As regras e o universo do game existem para proporcionar uma estrutura e um contexto para as ações de um jogador. As regras também existem para criar situações interessantes com o objetivo de desafiar e se contrapor ao jogador. As ações do jogador, suas decisões, escolhas e suas oportunidades, na verdade, sua jornada, tudo isso compõe a 'alma do game'. A riqueza do contexto, o desafio, a emoção e a diversão da jornada de um jogador, e não simplesmente a obtenção da condição final, é que determinam o sucesso do game. (2013, p.7).

São amplas as capacidades dos softwares contemporâneos. E tal caráter é perceptível ao observar a variedade de tipos de aplicações que são julgadas como sendo jogos eletrônicos. Existem games que exigem respostas rápidas e constantes de seus usuários para corresponder às ações frenéticas que se encontram em execução. Em contraponto, há games de comportamentos mais passivos, que não dispõem de tantas opções de interação mas que acarretam em um peso maior sob cada ação tomada pelo jogador.

É possível que existam games que impõem o cumprimento de tarefas específicas, assim como há também games que dispõem de ferramentas para que o jogador tenha liberdade de fazer o que bem quiser com o universo em seu escopo. Independentemente do tipo de jogabilidade ou do nível de complexidade que um game apresente, todos os jogos eletrônicos visam proporcionar ao jogador uma experiência satisfatória, e é a partir da qualidade desta experiência que se avalia o quão bom um game é.

Dependendo do objetivo de um determinado jogo e da maneira como suas regras são especificadas, este pode se enquadrar em um dentre os diversos gêneros de games existentes. Jogos do gênero de ação envolvem reações rápidas e julgamentos instantâneos por parte do jogador. Jogos de aventura são caracterizados por incluírem "exploração, coleta de itens, solução de quebra-

cabeças, orientação em labirintos e decodificação de mensagens [...], proporcionando tempo para o jogador refletir" (NOVAK, 2010, p.101). Em games que implementam as características de um RPG, "o jogador deve interpretar um personagem, além de evoluir o mesmo durante o jogo pelo acumulo de experiência adquirida através de batalhas, exploração e solução de problemas" (OLIVEIRA, 2012, p. 13). Jogos do gênero puzzle (termo inglês para quebra-cabeça, amplamente usado para definir games desta natureza) são especificamente voltados para desafiar a capacidade de raciocínio do jogador. Em jogos de estratégia, o jogador deve "elaborar estratégias/táticas baseando-se nas regras do jogo para ganhar do oponente ou conquistar um objetivo" (OLIVEIRA, 2012, p. 13). Os jogos de simulação, por sua vez, "tentam reproduzir sistemas, máquinas e experiências usando regras do mundo real" (NOVAK, 2010, p.106).

As particularidades da jogabilidade apresentada em um game também podem ser usadas como parâmetro para classificar os jogos. Os de ação que envolvem armas de fogo se consolidaram em seu próprio gênero, por exemplo. Já os games de simulação se dividem em diferentes sub-gêneros: simulação de veículos, simulação de esportes e simulação de processos do mundo real. Afirma-se também que é muito comum a convergência de aspectos de diversos gêneros diferentes em um único jogo.

Jogos eletrônicos atendem diversos tipos de motivações de seus jogadores. Algumas das motivações apontadas por Novak (2010) são a satisfação em competir com outros jogadores; os conhecimentos que podem ser adquiridos através de um game; a capacidade de domínio e conquista que o jogador poderá estabelecer dentro do universo do jogo; e o aspecto de escapismo das tensões e desafios existentes na vida real que os games são capazes de proporcionar. A autora também aponta para as várias áreas em que os games podem ser aplicados. Muito além do que apenas uma ferramenta de entretenimento, jogos eletrônicos têm se mostrado proeminentes para a criação de comunidades, espaços nos quais os usuários podem interagir entre si por intermédio da interface do game, e para fins educacionais, com o jogo eletrônico adotando um papel de ferramenta pedagógica. Tal aspecto educacional normalmente se manifesta em games que especificamente são criados com o objetivo de educar o jogador, embora diversos jogos que visam primariamente o entretenimento, em particular dos gêneros de estratégia e simulação, "permitem que os jogadores adquiram conhecimentos sobre objetos do

mundo real [...] enquanto jogam e aplicam conhecimentos adquiridos fora do game" (NOVAK, 2010, p. 76).

2.2 Aplicações multiplataforma

Aplicações de caráter multiplataforma podem ser caracterizadas conforme a definição apresentada pelo dicionário Michaelis (2009) como sendo um programa computacional que seja capaz de funcionar em várias plataformas ou equipamentos diferentes. Em outras palavras, uma vez que pelo menos grande parte do código-fonte de um programa tem a capacidade de ser compilado para rodar em mais de uma interface distinta, tal aplicação pode ser caracterizada como sendo multiplataforma.

No que se refere ao cenário dos jogos eletrônicos, o desenvolvimento tecnológico possibilitou uma ampla gama de aplicativos diferentes com a capacidade de rodar games. Essencialmente, jogos costumam ser executados em computadores pessoais, dispositivos portáteis como smartphones e tablets ou em aparelhos prioritários para execução de jogos (os chamados consoles de videogame). Destaca-se que existe uma grande variedade de sistemas para cada um desses itens: computadores podem apresentar sistemas operacionais Windows, Linux ou Macintosh; smartphones e tablets tendem a rodar com sistemas iOS ou Android; e existem diversos tipos de consoles de videogames disponíveis no mercado, sendo os mais proeminentes consoles atualmente: o Playstation 3 e o Xbox 360, com seus respectivos sucessores Playstation 4 e Xbox One também consolidando sua popularidade rapidamente. Dada essa vasta diversidade de dispositivos, qualquer jogo eletrônico que puder estar disponível em mais de uma das interfaces citadas, com a maioria de suas funcionalidades e recursos intactos, pode ser considerado como sendo um aplicativo multiplataforma.

No entanto, desenvolver para mais de uma plataforma não pode ser considerada como uma tarefa trivial, pois cada plataforma apresenta suas particularidades. Conforme Pedrosa et al. (2006), as diferenças que existem entre dispositivos, no que se refere à usabilidade, interfaces de apresentação, características de comunicação e capacidade de processamento, implicam na adaptação das interações e formas de apresentação do game para cada dispositivo no qual este será disponibilizado. Ainda que este cenário ofereça desafios, também

existem muitas possibilidades únicas que podem ser exploradas dentro de cada plataforma, pois:

> [...] um jogo específico pode oferecer aos usuários características interessantes da plataforma para o qual foi desenvolvido, como mobilidade, para o caso de jogos multiusuário móveis, ou uma interface gráfica rica e altamente interativa, para os jogos de computadores de mesa ou consoles de videogame. (PEDROSA et al, 2006, p.1).

O suporte multiplataforma pode ser alcançado de diferentes maneiras, dependendo do tipo de linguagem de programação sendo utilizada no desenvolvimento do jogo. Comentários acerca de especificidades de linguagens de programação serão mencionados no tópico a seguir.

2.3 Linguagens de Programação

Essencialmente, jogos eletrônicos não são diferentes de qualquer outra aplicação computacional. Todos os games são compostos por código escrito em uma determinada linguagem de programação que é convertido para poder ser executado em um determinado tipo de hardware.

Para que um hardware consiga realizar alguma ação é necessário que essa lhe seja descrita especificamente em linguagem de máquina[1]. Deitel P. e Deitel H. (2010) nos remetem à época em que surgiu a linguagem Assembly, que permitiu a escrita de comandos compreensíveis para humanos e que poderiam ser automaticamente convertidos em linguagem de máquina, tornando o trabalho de programação em algo muito mais prático de ser realizado. Ainda assim, muitos programadores sentiam que a programação em Assembly ainda era muito exaustiva. Assim, surgiriam as primeiras linguagens de programação de alto nível, que aprimoraram a maneira como os códigos são escritos de maneira a serem facilmente compreendidos pelas pessoas.

Desde então, uma infinidade de linguagens de programação surgiram, apresentando diferentes funcionalidades e metodologias. Escolher com qual linguagem se deseja trabalhar de fato não é algo trivial, pois existem muitos fatores

[1] A linguagem de máquina é composta por conjuntos de números descritos em binário, ou seja, por 0 e 1. Uma determinada sequência de conjuntos desses números irá corresponder a uma específica instrução que será realizada pelo processador do computador.

que envolvem o processo de desenvolvimento de uma aplicação computacional que deverão ser analisados pelo programador. Nas palavras de Schildt,

> “em uma análise final, se você quiser criar software de alto desempenho, use C++. Se precisar criar software altamente portátil, use Java ou C# [...] A questão aqui não é que linguagem é melhor por si só. Ao invés disso, a questão é que linguagem é a correta para o trabalho em questão” (2002, p. 5).

2.3.1 Suporte de uma linguagem de programação a múltiplas plataformas

Através do processo chamado de compilação, o código fonte escrito em uma determinada linguagem de programação, que possibilita “aos programadores escrever instruções que se parecem com o inglês cotidiano e contêm notações matemáticas comumente utilizadas” (DEITEL P. DEITEL H, 2010, p. 6), é convertido em linguagem de máquina.

Em diversas linguagens, como o C++ por exemplo, o processo de compilação é realizado de forma a gerar instruções que serão executadas diretamente pelo sistema operacional ou pelo hardware de uma máquina. As instruções, por sua vez, podem diferir entre os diferentes tipos de hardware existentes. Em contrapartida, existem linguagens que dispõem de uma aplicação denominada interpretador que irá executar as instruções compiladas e se encarregar de gerar os resultados. Isso significa que o código compilado não será lido pelo sistema operacional ou pelo hardware, mas sim por esta aplicação intermediária. Exemplos de linguagens interpretadas são o HTML, interpretado através dos navegadores de Internet, e o Java e C#, que utilizam as chamadas máquinas virtuais para executar as suas instruções: a JVM no caso do Java e a CLR para o C#. É nesse ponto em que se percebe como funciona o suporte a múltiplas plataformas de uma determinada linguagem.

Segundo um exemplo citado por Schildt (2002), para rodar um programa criado em C++ em diferentes tipos de sistema, o código precisará ser especificamente compilado para as instruções de cada um destes ambientes. Código escrito linguagens interpretadas como o Java ou C# é convertido para um conjunto de instruções padronizado de suas máquinas virtuais. Assim, essas linguagens conseguem facilmente serem multiplataforma, bastando apenas a

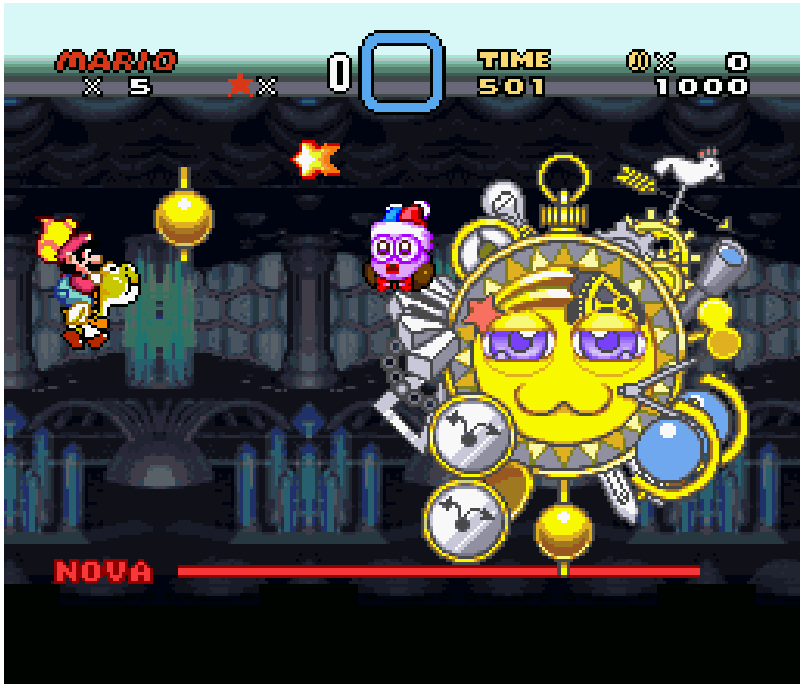
instalação da máquina virtual dentro do sistema para executar programas escritos nestas linguagens.

2.3.2 Recursos existentes em linguagens de programação para o desenvolvimento de jogos

Essencialmente, qualquer linguagem é passível de ser usada para a criação de jogos. Normalmente se recomenda a utilização de alguma linguagem de programação de alto nível[2] que faça uso de algum paradigma como a orientação a objeto, por exemplo, em decorrência do dinamismo e recursos que estas oferecem. Mesmo assim, nada impede que um jogo seja desenvolvido utilizando uma linguagem de nível mais baixo como o Assembly. O desenvolvedor japonês conhecido como *Carol* fez uso da programação em Assembly para realizar extensivas modificações no clássico jogo Super Mario World, originalmente lançado para o Super Nintendo em 1991, adicionando novas funções à sua jogabilidade. Estas em sua maioria são muito mais complexas do que as existentes no jogo original, como a adição de novos inimigos com comportamentos inéditos e diversas fases com funcionalidades únicas, conforme pode ser visto na Figura 1; e são caracterizadas por apresentarem ao jogador um considerável grau de desafio (o que justifica esse jogo ser popularmente conhecido pelo nome *Brutal Mario*).

[2] Linguagens de programação de alto nível determinam as instruções a serem realizadas por uma máquina utilizando uma sintaxe que é baste próxima da linguagem humana. Assim, elas possibilitam a criação de código-fonte que pode ser facilmente compreendido por seres humanos e descartam a necessidade de se conhecer a arquitetura do dispositivo específico com que se está trabalhando.

Figura 1: Tela do jogo Super Kitiku Mario, popularmente conhecido como Brutal Mario



Fonte: http://img1.wikia.nocookie.net/__cb20140706233459/brutalmario/images/0/0b/Nova.png

A questão é que tais linguagens de baixo nível não costumam ser utilizadas para a programação do game propriamente dito, mas sim para o desenvolvimento de bibliotecas ou ferramentas que serão utilizadas para o desenvolvimento de jogos, em virtude da proximidade ao hardware que estas possuem. Isso é evidente no trabalho da desenvolvedora *Vd-Dev*[3] em seu jogo *Ironfall Invasion* para o Nintendo 3DS. Sendo o seu objetivo obter o máximo de performance ao executar o jogo em sua plataforma alvo[4], eles optaram por conta própria construir cada ferramenta e recurso a ser utilizados por este jogo, programando-os diretamente para o processador do portátil, ou seja, através da programação em Assembly. O caso deles pode ser dito como sendo aparte, já que o comum é aproveitar os recursos e ferramentas mais robustas já existentes.

---

[3] http://www.vd-dev.com**/**

[4] Detalhes acerca das características do motor no qual o jogo Ironfall Invasion roda foram apresentados em um vídeo promocional divulgado pela sua desenvolvedora: http://youtu.be/M6sOmcrBCGg

Atualmente, as linguagens de alto nível dispõem de recursos que contemplam diversos dos requisitos básicos esperados de um bom jogo eletrônico como gráficos, música e a jogabilidade. C++ é um exemplo interessante por ser uma linguagem que oferece um bom equilíbrio entre programação de baixo nível e programação de alto nível. Sendo assim, essa linguagem já foi amplamente utilizada tanto para criar bibliotecas básicas para o tratamento de elementos gráficos ou sonoros, ferramentas de desenvolvimento e até mesmo games completos.

Essa linguagem conta com diversas bibliotecas voltadas tanto para o desenvolvimento de jogos 2D quanto 3D. Em uma publicação no site Game From Scratch (2011), o autor menciona as bibliotecas SimpleDirect Media Layer (SDL)[5], Allegro[6] e Simple and Fast Media Library (SFML)[7], destacando como elas contemplam praticamente qualquer aspecto do desenvolvimento de um jogo 2D, sejam eles a jogabilidade, os gráficos, música ou conectividade via rede. Para o desenvolvimento de jogos tridimensionais com C++ pode-se fazer uso de DirectX[8] ou Open GL[9]. O Open GL em particular se destaca pelo seu vasto suporte a múltiplas plataformas e diversas linguagens de programação além de C++, ao passo que o DirectX é uma tecnologia específica para o desenvolvimento em plataformas e linguagens proprietárias da Microsoft.

Apesar de todo o potencial oferecido pelo C++, esta é uma linguagem bastante complexa para se trabalhar. Linguagens como Java e C# se apresentam como excelentes alternativas por apresentarem uma grande quantidade de bibliotecas padrões que muito facilitam o trabalho de programação. Isso permite alcançar o resultado esperado de maneira bastante prática e em poucas linhas de código, ao passo que em C++ é exigido muito mais esforço e trabalho manual para que se consiga concretizar este mesmo resultado (GAME FROM SCRATCH, 2011).

Java é uma das linguagens mais usadas utilizadas atualmente, podendo ser utilizada para o desenvolvimento de praticamente qualquer tipo de aplicação, estando jogos inclusos nisso. De maneira geral, Java não é uma das principais

---

[5] http://www.libsdl.org/

[6] http://liballeg.org/

[7] http://www.sfml-dev.org/

[8] http://msdn.microsoft.com/en-us/directx/aa937781

[9] http://www.opengl.org/

linguagens para o desenvolvimento de jogos, mas ainda assim dispõe de diversos recursos para tal, através de bibliotecas como LibGDX[10], Jogamp/jogl[11] e Lightweight Java Game Library (LWJGL)[12]. Enquanto C# também é capacitado para o desenvolvimento de qualquer tipo de aplicação, essa linguagem consegue se destacar para o desenvolvimento de jogos por ser utilizada no framework XNA[13], tecnologia de desenvolvimento de games para as plataformas da Microsoft (computadores com Windows, dispositivos móveis com Windows Phone e consoles de videogame da marca Xbox).

Indo além do Java e C#, há linguagens de script que podem ser executadas tanto de maneira independente como também dentro de outros programas e linguagens de programação. Phyton[14], Ruby[15] e Lua[16] são exemplos de linguagens de script, criadas visando proporcionar simplicidade, dinamismo, objetividade e portabilidade à prática de programação. Tal princípio vem possibilitando a popularização destas plataformas, principalmente para a criação de jogos. Lua em particular já vem sendo amplamente utilizada para desenvolver jogos através de sua integração com ferramentas já existentes. Há ainda a existência de bibliotecas para desenvolver jogos nestas linguagens, como o PyGame[17] para desenvolvimento com Python, por exemplo.

Também é importante fazer uma menção à linguagem HTML, que está emergindo como uma poderosa alternativa para a criação de jogos. O HTML é conhecido como sendo a tecnologia padrão usada para a criação de páginas da Web. Embora essa tecnologia tenha se estagnado por um longo período de tempo sem atualizações, as entidades mantenedoras da linguagem, W3C e WHATWG, têm trabalhado desde 2008 na quinta versão da linguagem, visando propor uma completa reformulação da linguagem para aumentar as suas capacitações. Tamanha é a distinção de suas versões anteriores e o potencial visado por ela que

---

[10] https://github.com/libgdx/libgdx/

[11] http://jogamp.org/

[12] http://lwjgl.org/

[13] http://msdn.microsoft.com/en-us/aa937791.aspx

[14] http://www.python.org.br/wiki

[15] https://www.ruby-lang.org/pt/

[16] http://www.lua.org/portugues.html

[17] http://www.pygame.org/wiki/about

se tornou frequente mencioná-la especificamente como HTML5. A nova versão prevê uma maior integração com as linguagens CSS e JavaScript e uma das grandes novidades presentes é o maior suporte nativo da linguagem a recursos gráficos e de multimídia. Estes aspectos abrem possibilidades para a criação e execução de ferramentas web mais poderosas, dentre as quais jogos eletrônicos são um dos tipos que mais têm se destacado.

Espalhados pela Internet já se encontram diversos exemplos de jogos desenvolvidos utilizando HTML e as perspectivas apontam para o contínuo aprofundamento do uso desta linguagem para o desenvolvimento de games. Através da Figura 2, pode-se observar um exemplo de um jogo implementado em HTML5, o qual possui seu código fonte escrito em Javascript e utiliza o marcador *Canvas* para gerar o jogo na tela do navegador de Internet.

Figura 2: Jogo desenvolvido em HTML5 rodando em um navegador de Internet através do marcador <canvas>

Fonte: Autor.

2.4 Trabalhos Correlatos

Considerando o quão proeminente os jogos eletrônicos têm se tornado, muitos acadêmicos foram motivados a construir trabalhos relacionados a games. Isso se mostra evidente em dois trabalhos recentes desenvolvidos por alunos do Instituto Federal Sul-Rio-Grandense Campus Passo Fundo. Thomaz Canali Xavier

publicou em 2011 como trabalho de conclusão a monografia *Estudo E Desenvolvimento De Jogos Para Internet Utilizando Unity 3D*. Já em 2012, Rômulo Reis De Oliveira defendeu a monografia *Desenvolvimento De Jogos Eletrônicos Online Em Tempo Real, Para Multiplos Jogadores E Multiplataformas*.

Ao observar o advento das redes sociais, Xavier (2011) ficou instigado pelo aspecto de entretenimento existente nelas, em razão das redes sociais possuírem suporte a integração e execução de jogos e aplicativos. Percebendo o potencial ali existente, propôs o desenvolvimento de um jogo demonstrativo o qual seria integrado a uma rede social.

> Os jogos para internet se tornaram muito conhecidos principalmente pelas redes sociais. Estas oferecem a oportunidade ideal para novos desenvolvedores que queiram mostrar seus trabalhos. A internet é uma ótima ferramenta para fazer a divulgação do profissional e do seu projeto. (XAVIER, 2011, p. 9)

Dado tal objetivo, Xavier não somente pesquisou o desenvolvimento de jogos, mas também realizou aprofundados estudos relacionados à estrutura e funcionamento do contexto em que seu jogo seria inserido: as redes sociais.

Oliveira (2012), por sua vez, notou os impactos que o advento de novas tecnologias, como os smartphones e tablets, têm causado no perfil de jogadores de games. A tecnologia atual permite que jogos sejam jogados em diversos lugares, de diferentes formas e inclusive em diversas plataformas, de tal modo que os jogadores esperam que seus games apresentem tais funcionalidades. Dessa forma, Oliveira destacou extensivamente no decorrer de sua monografia a questão de um jogo ser multiplataforma e as implicações que isso acarreta. Através de seu trabalho de pesquisa, "foi possível perceber que jogos MMO[18] e jogos multiplataformas são aspectos que estão em alta na indústria de jogos, entretanto, são poucos os títulos disponíveis que abordam esses dois aspectos" (OLIVEIRA, 2012, p.27). Isso o motivou a estabelecer como objetivo desenvolver um estudo de caso, de um jogo MMO multiplataforma, que adotaria os processos e tecnologias que foram estudados no decorrer de sua pesquisa.

---

[18] Tratam-se de games online que são jogados simultaneamente por uma grande quantidade de usuários, que compartilham de um mesmo mundo virtual e cujas ações são visíveis para todos os demais usuários.

Para alcançar os objetivos propostos em seus trabalhos, ambos fizeram uso de ferramentas de alto nível para auxiliar não somente no que tange a programação do jogo, mas sim em todo o conjunto de demais elementos que constituem um game e no gerenciamento de seus projetos. Notavelmente, as duas ferramentas pesquisadas por Xavier e Oliveira em seus trabalhos, Unity[19] (na época chamado de *Unity 3D*) e Construct 2[20] respectivamente, apresentam o suporte multiplaforma como um de seus principais atrativos. Ambas são capazes de compilar seus projetos de forma a torná-los passíveis de serem executados nas principais plataformas usadas para jogar atualmente. Durante seus respectivos trabalhos, ambos puderam constatar a importância do aspecto multiplataforma. Para Xavier um dos aspectos mais notáveis da Unity "é a sua vasta possibilidade de compilação. [...] É possível fazer jogos para consoles de videogames, dispositivos móveis, computadores pessoais e até mesmo para Internet" (2011, p. 38). Oliveira acredita na relevância do desenvolvimento multiplataforma "pois quando a produtora restringe o meio de acesso a esse jogo, a mesma perde a oportunidade de alcançar um número maior de pessoas, além disso, os jogadores querem ter acesso ao jogo de qualquer plataforma" (2012, p.7).

É interessante também observar as duas vertentes existentes em cada uma destes softwares. Unity 3D, como o nome já indica, é uma ferramenta voltada para o desenvolvimento de games com elementos tridimensionais, sendo que ela “oferece a tecnologia mais avançada em termos de renderização, iluminação, terrenos, partículas, física, áudio, programação e networking” (XAVIER, 2011, p. 33). A Construct 2, em contrapartida, é focada em “auxiliar no desenvolvimento de jogos eletrônicos com gráficos 2D para navegadores web, utilizando recursos de HTML5” (OLIVEIRA, 2013, p. 32). Destaca-se assim a distinção dos dois programas abordados em cada uma das monografias como sendo um exemplo da diversidade de ferramentas para o desenvolvimento de jogos existentes, algo que será melhor abordado neste trabalho posteriormente.

O aspecto onipresente da Internet está refletido nas ferramentas de interação social, as quais são um dos grandes atrativos existentes neste meio atualmente, e os trabalhos mencionados demonstram como os jogos eletrônicos constituem esse

---

[19] http://unity3d.com/pt

[20] https://www.scirra.com/construct2

panorama. Através destes trabalhos, é visível o grande escopo que os jogos eletrônicos apresentam atualmente. São muitas as áreas em que os games podem ser aplicados, trazendo consigo um grande leque de possibilidades.

3 DESENVOLVIMENTO DE JOGOS

A implementação de jogos eletrônicos agrega diversas tecnologias e conhecimentos. Para todo o bem sucedido processo de desenvolvimento de um game, é necessária a realização de todo um planejamento e documentação de que tipo de jogo deseja-se construir e quais serão as ações necessárias para alcançar este objetivo. Nota-se a existência de uma vasta variedade de tecnologias que poderão ser empregadas para construir um jogo. Muito além das tradicionais linguagens de programação disponíveis, podem-se encontrar várias alternativas de ferramentas com recursos capazes de contornar muitas das dificuldades existentes no processo de desenvolvimento de um jogo.

3.1 Etapas de Desenvolvimento

No principio, há o desejo. Uma ideia, deveras abstrata, que quando devidamente tratada é capaz de eclodir, tornar-se concreta, se materializando em algo maior. Dificilmente transformar um desejo em realidade costuma ser uma tarefa simples. Ao menos isso é verdadeiro se o desejo em questão for a criação de games. Afinal, já foi frisado como jogos eletrônicos são um tipo de software particularmente complexo de ser desenvolvido.

Para se lidar com a complexidade da criação de games, e poder transformar aquele desejo em um jogo completo e funcional, o seu processo de desenvolvimento costuma ser tratado através de um conjunto de diferentes etapas. Há um consenso de que todo game com ciclo de desenvolvimento bem sucedido irá conter um estágio inicial no qual é estabelecido o conceito do jogo, seguido do estágio que contempla a construção do jogo e, por fim, se o game conseguir ser finalizado, um ciclo de pós-produção para criar novos conteúdos ou avaliar a receptividade do game em questão (SCHUYTEMA, 2008).

Diferentes autores detalham esse processo de desenvolvimento através de sua distinção em uma série de etapas. Ao dissertar sobre o assunto, Novak (2011) distinguiu o período de desenvolvimento em oito etapas: conceito, pré-produção, protótipo, produção, alfa, beta, ouro e pós-produção.

Os primeiros detalhes de um jogo eletrônico são estabelecidos durante uma breve fase na qual é redigido um documento de conceito. Por meio da elaboração deste documento determina-se as características essenciais deste projeto, “sendo que cada ideia deve ser analisada nos aspectos de originalidade, inovação, público-alvo, plataforma e possibilidades de mercado.” (OLIVEIRA, 2012, p.16). Havendo um consenso entre todos os envolvidos neste processo, o game pode prosseguir para a subsequente etapa de Pré-produção. A proposta do jogo continuará sendo desenvolvida, agora através do chamado Documento de Design do Game (DDG), um documento muito mais extenso do que o seu antecessor, que contempla o máximo possível de elementos que constituem o jogo e que irá acompanhar os desenvolvedores durante toda a vida útil do desenvolvimento.

O modo de jogar idealizado durante a fase de pré-produção deve ser testado para saber se de fato é possível concretizar o que foi estabelecido durante o design e se tal conceito realmente poderá se tornar um game interessante. Assim, o trabalho de prototipagem possui grande importância durante esta etapa. “O protótipo acabado mostra sua visão, confirma que o seu programa de produção é realista e que você conseguirá traduzir sua ideia em realidade” (NOVAK, 2010, p.343). Com a aprovação dos protótipos, é permitido ao projeto adentrar no estágio de Produção, em que de fato ocorre toda a criação do game propriamente dito.

No momento em que o processo de desenvolvimento alcançar um ponto em que a jogabilidade e a interface de usuário estiverem funcionais o jogo estará em seu estágio *Alfa*, permitindo que o game possa ser jogado do começo ao fim, mesmo que os demais elementos ainda estejam incompletos. Quando todas as partes tiverem sido finalizadas e incorporadas, o jogo chegara em seu estágio *Beta*, em que “o objetivo é estabilizar o projeto e eliminar o maior número possível de defeitos antes que o produto comece a ser vendido” (NOVAK, 2010, p.348).

O momento em que o jogo se encaminha para ser disponibilizado para o público é denominado como sendo a fase Ouro, remetendo a uma expressão dita pelos desenvolvedores quando o jogo é lançado: ‘*the game went gold*’ (o jogo tornou-se ouro). Trata-se do momento em que o jogo é enviado para as fabricantes confeccionarem os discos que serão colocados no mercado ou quando estes são disponibilizados para download através de meios digitais.

Dependendo da natureza do projeto, a fase ouro pode significar o final do desenvolvimento de um jogo eletrônico. No entanto, cada vez mais se tem adotado a

prática de se realizar um período estendido de desenvolvimento de novos conteúdos ou continuar dando suporte a um jogo já lançado. Já há muito tempo no cenário dos jogos para computadores pessoais existiam os chamados pacotes de expansão que adicionam novas funcionalidades a um jogo, sendo os jogos das séries The Sims[21] e Civilization[22] conhecidos por essas práticas. O advento da Internet conseguiu estabelecer um cenário em que é muito prático se comercializar e transmitir atualizações para jogos já existentes, de modo que essa prática deixou de ser exclusiva dos jogos de computadores e passou a ser realizada também com jogos para consoles de videogame. Outra contribuição desse cenário é a prática da realização de atualizações com correções de bugs ou modificações nos atributos de um jogo (comum em jogos de luta, em que se tenta realizar um balanceamento entre os personagens, enfraquecendo personagens excessivamente poderosos e melhorando os demais). A realização de novas versões ou atualizações de um jogo constitui a fase de Pós-Produção, para a qual é indicado já haver alguma perspectiva desde o início do projeto.

3.2 Design de Games

A maioria dos jogos eletrônicos costumam começar com fase de idealização e prototipagem. Quando se deseja trabalhar em um projeto de grande escala e complexidade, não há nada mais indicado ao começar do que realizar um planejamento de longo prazo para ter bem definidos os objetivos deste projeto e quais os desafios que tais objetivos implicam. Sem este artifício, o projeto muito facilmente poderia sucumbir a qualquer dificuldade enfrentada ou obter resultados muito aquém de seu real potencial. Quando falamos do desenvolvimento de jogos eletrônicos, no qual as tarefas a serem realizadas são numerosas e desafiadoras, a prática do design de games tende a sempre ser dita como o ponto de partida. É através deste que se estabelece a concepção básica do que se trata esse jogo e qual o propósito de sua jogabilidade.

Uma característica inerente à computação é o forte impacto causado pelo surgimento de novas tecnologias. Estas sempre trazem consigo novas tendências e mudanças na maneira como os softwares são criados. Há pouco tempo o universo

---

[21] http://www.techtudo.com.br/noticias/noticia/2012/08/conheca-todas-expansoes-para-sims-3.html

[22] http://www.civilization.com/en/games/civilization-v-brave-new-world/

dos games foi afetado pelo surgimento dos smartphones e tablets, estabelecendo uma nova base consumidora com comportamentos bastante distintos daqueles que jogam em seus consoles ou computadores.

Aplicativos para dispositivos móveis não dispõem da mesma capacidade de processamento ou das interfaces de entrada (vulgarmente falando, os botões de um controle ou o conjunto teclado e mouse) existentes em consoles e computadores. Nesta situação podemos contemplar o esplendor desta atividade, na qual desenvolvedores devem conciliar as limitações técnicas, os anseios de seu público alvo e as aspirações dos próprios envolvidos no projeto.

O design de jogos pode ser visto praticamente como sendo uma ciência à parte. A maneira como o design é realizado varia muito para cada elemento presente em um jogo de um determinado gênero e a sua realização deve ser contínua, permanecendo em constante metamorfose durante todo o processo de produção de um jogo eletrônico. O ideal é que em meio ao projeto exista alguém capacitado para se responsabilizar pelo trabalho de design do jogo a ser criado, ou seja, deve haver um individuo que atue na função de designer de games.

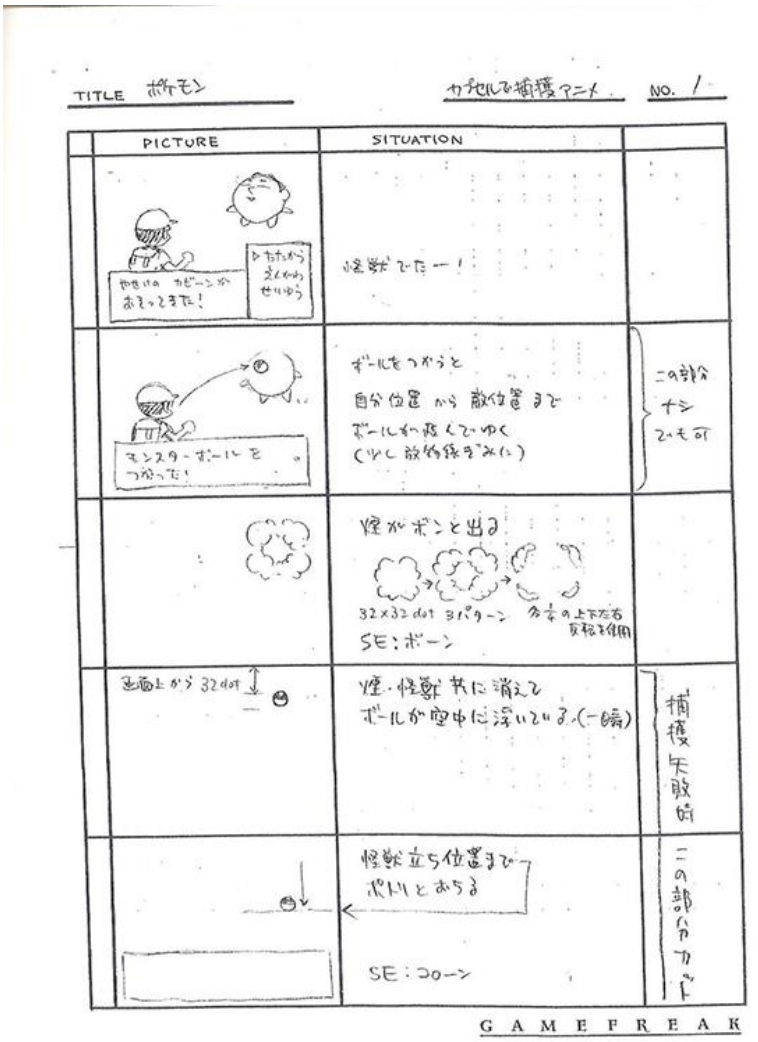
> Um designer de games [...] participa do processo de desenvolvimento da ideia até a implementação. A tarefa fundamental do designer é determinar o que um game – ou parte dele – deve fazer e, depois, documentar esse comportamento de modo que ele possa ser criado pela equipe de desenvolvimento. (SCHUYTEMA, 2013, p.18)

Novak (2010) comparou o trabalho de um designer de jogos ao de um engenheiro, por sua função realizar a adoção de soluções para que se possam projetar sistemas funcionais, sendo que, no caso de um jogo, estes sistemas seriam as suas interfaces e funcionalidades.

O produto final de seus esforços é o Documento de Design do Game, sendo a sua confecção a principal responsabilidade deste profissional. Schuytema (2013) destaca que os documentos de design podem existir de diferentes formas, dotados de estilos ou tipos de interpretação próprios, podendo ser documentos curtos ou com centenas de páginas e até mesmo podem fazer uso de recursos digitais como

Wikis[23]. Através da Figura 3 podemos compreender que há diversos artifícios que podem ser utilizados para a construção do DDG. Designers poderão utilizar recursos como texto descritivo, tabelas, quadros, gráficos e ilustrações para realizar a conceptualização do jogo e esquematizar o funcionamento do gameplay.

Figura 3: Página do documento de design dos jogos Pokémon Red Version e Green Version, detalhando o procedimento de captura de um Pokémon



カプセルで捕獲アニメ絵コンテ

ポケモンとの遭遇から、モンスターボールを使い、捕獲するまでの一連の動きが描かれた絵コンテ。モンスターボールでどのようにポケモンを捕獲するのかを詳細に描いている。

Fonte: http://www.betaarchive.co.uk/imageupload/1285977736.or.9687.jpg

[23] Wiki é um formato padronizado de página web voltado para disponibilizar coleções de documentos inter-relacionados, popularizado através da implementação da Wikipédia. Sua nomenclatura provém do idioma havaiano e significa rápido ou veloz.

A maneira como o design é tratado varia conforme a complexidade do jogo em questão. Ao se pensar em um jogo simples e pouco ambicioso, um programador pode simplesmente abrir a ferramenta de sua escolha e começar a programar imediatamente. Dentro da indústria dos games, no entanto, até o mais simplista dos games merece ter diversas páginas detalhando cada pequeno aspecto que o envolve. Não é enfadonho pensar desta forma, se novamente nos remetermos a ideia de que jogos eletrônicos são softwares bastante desafiadores de serem criados. Mesmo uma ideia aparentemente simples pode crescer exponencialmente em complexidade no momento em que você começar a conceber o seu código-fonte.

Tal conceito se torna ainda mais evidente no momento em que você passa a observar o desenvolvimento de seu jogo com um pouco mais de ambição. A interação direta que um jogador possui em um jogo eletrônico e do poder computacional das máquinas nas quais estes jogos estão rodando proporcionam um potencial de imersão que é inalcançável por qualquer outro meio de entretenimento. O jogador possui controle sobre o universo que existe dentro do game e as suas ações causam alterações neste universo que podem ser contempladas por ele em tempo real. Sem uma etapa de planejamento adequada, dificilmente este potencial poderia ser plenamente alcançado.

3.3 Elementos importantes em um jogo eletrônico

Não é cabível ao escopo deste trabalho se aprofundar nas muitas facetas que caracterizam um game ou nos inúmeros aspectos que contemplam o design de jogos, mas pode-se ao menos discorrer acerca de algumas das questões essenciais que devem consideradas ao se estabelecer quais serão os elementos básicos de um determinado game.

Isso porque decisões como: determinar o quão importante a presença de uma história é no escopo do jogo; especificar em qual gênero o jogo melhor se enquadraria; ou estabelecer uma equipe criar um jogo, avaliando quais seriam os tipos de profissionais disponíveis nela; causam grande impacto na maneira como será redigido o design do game e, consequentemente, como o seu desenvolvimento ocorrerá.

3.3.1 Definição das telas de um jogo eletrônico

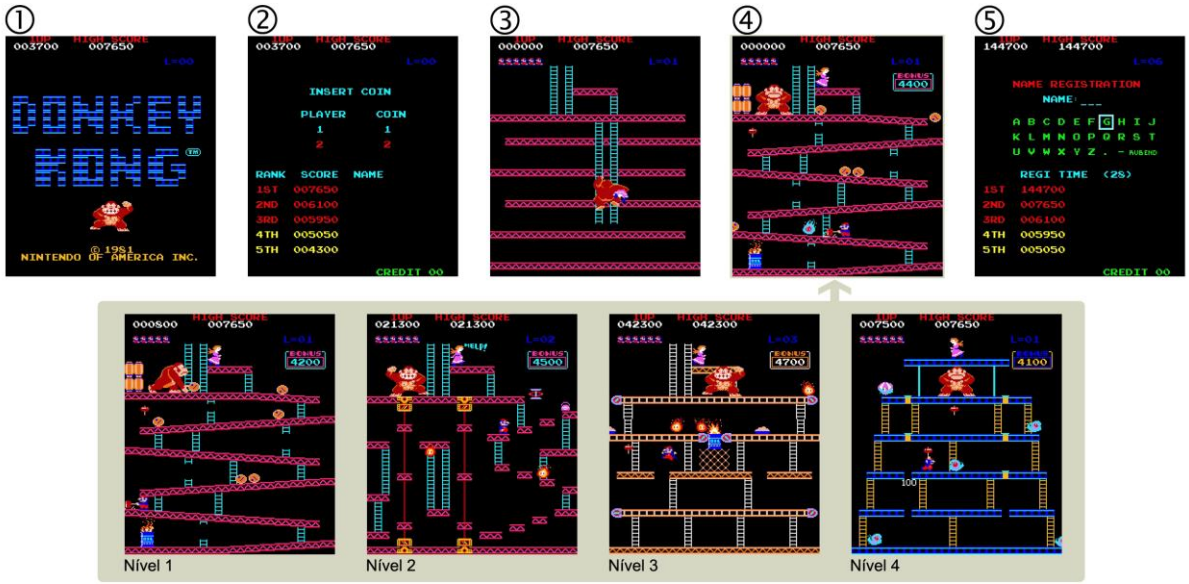
Parte da projeção de um jogo eletrônico envolve estabelecer os tipos de conteúdos que serão apresentados em um determinado momento de um jogo e qual a lógica que deverá ser aplicada nestes. Jogos eletrônicos são aplicações que poderão apresentar diferentes estados no decorrer de sua execução. A tela irá exibir o jogo executando comportamentos específicos dependendo do estado em que a aplicação se encontra.

No escopo geral da aplicação, um game é muito mais do que apenas o momento em que se está jogando. O design de um jogo eletrônico não contempla apenas o design das fases durante a execução da jogabilidade, mas todos os estados desta aplicação computacional. Todos os games precisam de uma tela inicial para mostrar o título do jogo e preparar o jogador para começar a jogar.

Enquanto em um momento o jogador está lutando contra inimigos, no momento seguinte este estará assistindo a uma animação referente ao enredo, em seguida estará configurando as roupas e acessórios do personagem e então decidirá retornar à tela inicial. Um dos objetivos do processo de design de games é determinar quais seriam estes estados e o fluxo de transição que ocorre entre eles.

Para melhor ilustrar este conceito, realizaremos uma análise utilizando como referência o jogo Donkey Kong, clássico dos fliperamas lançado em 1981 pela Nintendo. Observando todo o fluxo de execução deste game através da Figura 4, podemos identificar a ocorrência de pelo menos 5 telas com comportamentos distintos. A jogabilidade do jogo apresenta 4 níveis diferentes, o que pode ser considerado como sendo 4 telas diferentes, que somada às demais 4 telas estabelecem um jogo eletrônico que poderá apresentar 8 estados diferentes no decorrer de sua execução.

Figura 4: Lista de telas do jogo Donkey Kong durante a sua execução



Fonte: Autor.

1. A tela de título do jogo;
2. A tela em que são exibidas as opções para iniciar o jogo, assim como também as melhores pontuações salvas na memória;
3. A tela em que é executada uma animação na qual é introduzido o contexto do jogo e seu objetivo;
4. A tela em que ocorre a execução principal do jogo, com 4 níveis;
5. A tela em que são exibidos o resultado final de uma determinada partida;

3.3.2 Narrativa

Seja de maneira mais simplória, utilizando a narrativa para apresentar as regras do jogo e justificar a sua jogabilidade ou de maneira que o propósito do game em si seja o de contar uma história, a narrativa também é um aspecto que constitui o desenvolvimento de jogos eletrônicos.

Desde 1985, os games de plataforma 2D do Super Mario nunca foram focados em retratar uma narrativa, embora o conto de um herói que se aventura para salvar a princesa raptada sempre esteja presente como pretexto para as várias fases que irão desafiar o jogador, como pode ser visto na Figura 5.

Figura 5: Obrigado Mario! Mas a nossa princesa está em outro castelo e para resgatá-la você vai precisar passar pela próxima sequência de fases!

Fonte: http://i1.kym-cdn.com/photos/images/original/000/186/610/thankyou.png

Ter uma boa história para ser contada por detrás de um game é uma ótima maneira de se apresentar as regras e o mundo que permeia um determinado game, além de este ser um grande atrativo para muitos jogadores. Um exemplo do uso da narrativa em jogos está em Professor Layton[24], uma série de jogos de puzzle lançados para o Nintendo DS e Nintendo 3DS que contextualiza os seus quebra-cabeças através de uma rica narrativa apresentada através de texto narrativo e trechos em animação.

Um jogo que desejar apresentar uma narrativa, no entanto, terá aderido ao seu desenvolvimento uma etapa adicional para a sua construção, já que a elaboração de uma história de qualidade também pode ser dita como sendo algo desafiador.

Mesmo implicando em algo a mais a ser trabalhado durante o desenvolvimento, muitos desenvolvedores agregam histórias em seus games, pois cada vez mais os jogos eletrônicos vêm sendo considerados como efetivas

[24] http://professorlayton.nintendo.com/

ferramentas para contar histórias e proporcionar aos jogadores experiências reflexivas. O jogo de aventura desenvolvido pela *TellTale Games* inspirado no homônimo universo da série *The Walking Dead*[25] foi sucesso de público e crítica por sucessivamente ter retratado uma narrativa emocionalmente imersiva, na qual a jogabilidade permite aos jogadores adentrarem dentro deste universo e experienciar o impacto que as suas decisões causam a si mesmo e aos demais personagens envolvidos na trama.

3.3.3 Determinando o gênero do jogo

Ao contextualizar os jogos eletrônicos, foi mencionada a distinção em gêneros que os jogos eletrônicos possuem para catalogá-los com base em seu estilo de jogabilidade e objetivos. Algo surpreendente no mundo dos games é a extensa variedade de tipos que existem.

Analisando os extremos, de um lado há os jogos que unicamente e puramente servem para distrair o jogador por uma determinada quantidade de tempo, seja através de alguma atividade simplória ou o atribuindo desafios mentais. No outro extremo, temos jogos que apresentam narrativas complexas, mundos gigantescos e gameplay que proporciona ao jogador uma grande gama de ações executáveis dentro do game. E entre estas duas definições há quase infinitas possibilidades.

Uma das primeiras decisões a serem tomadas é estabelecer qual gênero melhor representa o tipo de jogo que se deseja criar. Isso porque alguns gêneros tendem a beneficiar certos tipos de experiências: jogos de ação são ótimos para entreter múltiplos jogadores simultaneamente por longos períodos de tempo, enquanto o gênero de quebra-cabeças, por sua vez, é particularmente excelente para criar jogos para pessoas que apenas querem se distrair por alguns instantes. Remetendo ao assunto tratado no item anterior, há gêneros como os jogos de aventura e os de RPG que beneficiam o propósito de se contar histórias.

Enquadrar um game dentro de um determinado gênero o beneficia por proporcionar a ele a aplicação de várias convenções já consagradas por outros jogos de natureza similar, mas na opinião da designer de games, escritora e

[25] http://www.telltalegames.com/walkingdead/season1/

educadora Tracy Fullerton (apud NOVAK, 2010) isso também pode acarretar na restrição do processo criativo, induzindo designers de jogos a meramente reproduzir as mesmas soluções já repetidamente aplicadas em tantos outros do mesmo gênero. Fullerton acredita que isso pode ser contornado através da mescla de diferentes gêneros de jogabilidade, combinando características de diferentes tipos pré-estabelecidos de jogos para incrementar a experiência original do game, podendo resultar em maneiras inovadoras de jogar e inclusive abrir a possibilidade para o surgimento de gêneros completamente novos.

Um jogo que nos remete a essa miscelânea de gêneros é Destiny[26], desenvolvido pelo estúdio Bungie e lançado para múltiplas plataformas em setembro de 2014. Este trata-se de um jogo de tiro em primeira pessoa massivamente multijogador online em que os jogadores exploram um mundo aberto e persistente e realizam a evolução de personagem característica de jogos de RPG. Através da combinação de gêneros, este consegue proporcionar uma experiência bastante singular se comparado com outros jogos de tiro disponíveis no mercado.

Outro exemplo de jogo que realiza essa intersecção de gêneros é Danganronpa: Trigger Happy Havoc[27], uma *visual novel*[28] que se utiliza de exploração em primeira pessoa e quebra-cabeças em momentos chave para transcorrer a sua narrativa, contando inclusive com a inusitada implementação da jogabilidade de jogos de tiro em primeira pessoa para, de maneira figurada, representar os momentos de discussão entre os personagens. Aqui, armas de fogo são substituídas por argumentos e evidências, os quais devem ser posicionados utilizando a mira na tela para serem, literalmente, disparados contra as falácias existentes nos argumentos de seus oponentes, conforme demonstrado na Figura 6. Propostas inusitadas como essa justamente costumam ser o atrativo de um game e o aspecto que mais poderá ser apreciado pelos jogadores.

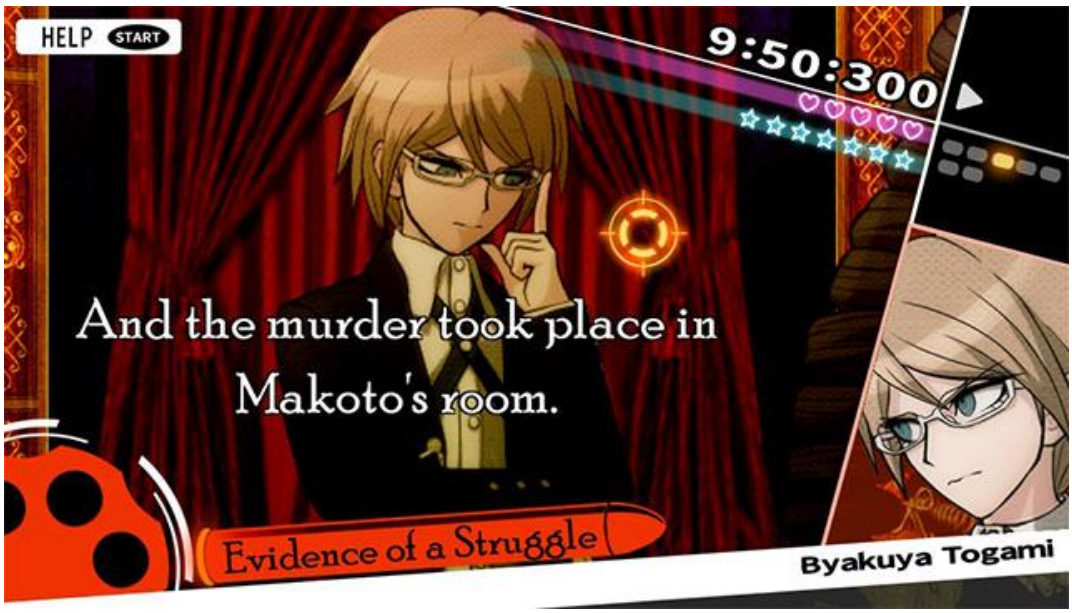
---

[26] http://www.destinythegame.com/pt

[27] http://nisamerica.com/games/danganronpa/index.php

[28] Visual Novel (literalmente romance visual em português), é um sub-gênero de jogo eletrônico especificamente centrado em contar histórias, sendo particularmente popular entre desenvolvedores e jogadores japoneses, que alia a literatura à interação e imersão que os games proporcionam ao jogador para retratar uma narrativa dinâmica na qual as ações do jogador definem os rumos da história.

Figura 6: Tela do jogo Danganronpa: Trigger Happy Havoc



Fonte: http://nisamerica.com/games/danganronpa/images/ss/system_04.jpg

3.3.4 Equipe de desenvolvimento

Até meados da década de 90, era comum que as equipes responsáveis pelo desenvolvimento de games fossem constituídas por pequenos grupos de pessoas, não sendo incomum haver uma única pessoa designada para cada tarefa, ou uma pessoa que se encarregasse de várias funções diferentes simultaneamente.

Foi apenas próximo da virada do século que a tecnologia alcançou o potencial e grau de complexidade que exigiu mais mão-de-obra para se construir um jogo. Pode-se dizer que, desde então, a demanda de pessoas para se construir games que utilizassem todo o potencial proporcionado pelo hardware atual cresceria exponencialmente.

Os jogadores ficaram encantados por poder controlar com precisão e fluidez personagens que visualmente beiravam o realismo através de vastos cenários, e logo se tornaram bastante exigentes quanto ao que gostariam de visualizar e experienciar em um jogo eletrônico. Para se construir as experiências complexas capazes atender as expectativas dos mais exigentes jogadores é necessário empregar e gerenciar um grande número de diferentes tipos de profissionais.

Conforme destacado por Oliveira (2012), a criação de jogos eletrônicos requer o emprego de programadores para escrever o código-fonte; designers gráficos para

tratar a parte visual do jogo; músicos ou sonoplastas para compor uma trilha sonora e efeitos sonoros; escritores para produzirem o roteiro, diálogos ou mesmo o texto descritivo que acompanha interface do jogo; e indivíduos designados especificamente para realizar testes à procura de falhas ou inconsistências na jogabilidade. Mesmo que a perspectiva de se trabalhar em um grande estúdio de jogos seja algo distante da realidade de um aspirante desenvolvedor de jogos brasileiro, as ferramentas que a tecnologia contemporânea fornece tornam o desejo de se criá-los uma realidade palpável.

Não é impossível para um único indivíduo encabeçar todas as diferentes partes que constituem um jogo de qualidade. Um popular exemplo disto é o jogo Cave Story[29], um jogo do gênero plataforma 2D focado em ação e exploração lançado originalmente em 2006, amplamente aclamado pela sua qualidade, o qual foi inteiramente desenvolvido por Daisuke "Pixel" Amaya, sozinho se encarregando de produzir o design, roteiro, programação, arte e áudio em seu tempo livre.

Ainda assim, o cenário ideal continua sendo a perspectiva de se haver pelo menos mais do que pessoa para se dividir as tarefas do desenvolvimento. Felizmente, graças à Internet, não é difícil localizar e se comunicar com pessoas que compartilhem dos mesmos interesses. Um caso curioso é o da visual novel Katawa Shoujo[30], que possibilita ao jogador socializar e, eventualmente, poder se envolver romanticamente com uma dentre cinco garotas portadoras de deficiências físicas. O gênesis do projeto ocorreu a partir de discussões casuais realizadas entre vários usuários das páginas sobre animé do infame fórum *4chan*[31] que eventualmente se mobilizaram e se converteram em uma equipe que conseguiu concretizar o desenvolvimento deste jogo.

O tal chamado cenário dos jogos *indie*, termo originado da abreviação da palavra *independente*, desenvolvidos por equipes compostas por algumas poucas pessoas (de maneira similar à como os jogos eram desenvolvidos até a década de 90) veem conquistando o mercado, se contrapondo às produções de grande porte.

---

[29] http://www.cavestory.org/

[30] http://www.katawa-shoujo.com/about.php

[31] Um dos mais populares fóruns de mensagens e imagens. Famoso pela sua moderação arbitrária e sistema de postagens anônimas, mas principalmente pelo comportamento anárquico de seus usuários. Recomenda-se total discrição a um indivíduo que ousar visitar este site.

Inclusive já há grupos de desenvolvedores independentes compostos por brasileiros como a Behold Studios[32], que já publicou jogos para Android, iOS e computadores, e a Swordtales, responsável pelo desenvolvimento do jogo Toren[33], exibido na Figura 7, para a plataforma Playstation 4 e computadores que recebeu apoio da Lei Rouanet de incentivo à Cultura[34].

Figura 7: Tela do jogo Toren, desenvolvido pelo estúdio brasileiro Swordtales

Fonte: http://toren-game.com/images/screenshots/image_09.jpg

3.4 *Engines* de Desenvolvimento de Jogos

Conforme explicado no capítulo anterior, todos os jogos eletrônicos são aplicações computacionais construídos a partir da escrita de código em uma determinada linguagem de programação. Dessa forma, pode-se afirmar que o programador é um indivíduo chave no processo de criação de um game. Isso porque, considerando o cenário em que um indivíduo solitário deseje desenvolver um jogo, mesmo que não se tenha o domínio das artes gráficas, não se saiba como compor músicas e nem mesmo se tenha a mínima noção sobre design de games, no fim das contas, jogos eletrônicos são software e para que se possa construir

[32] http://beholdstudios.com.br/

[33] http://toren-game.com/

[34] Lei nº 8.313 de 23 de dezembro de 1991, a qual estabelece medidas de incentivos fiscais para a aplicação de parte do imposto de renda em atividades culturais.

software o requisito mínimo sempre será ter algum conhecimento das intempéries da computação.

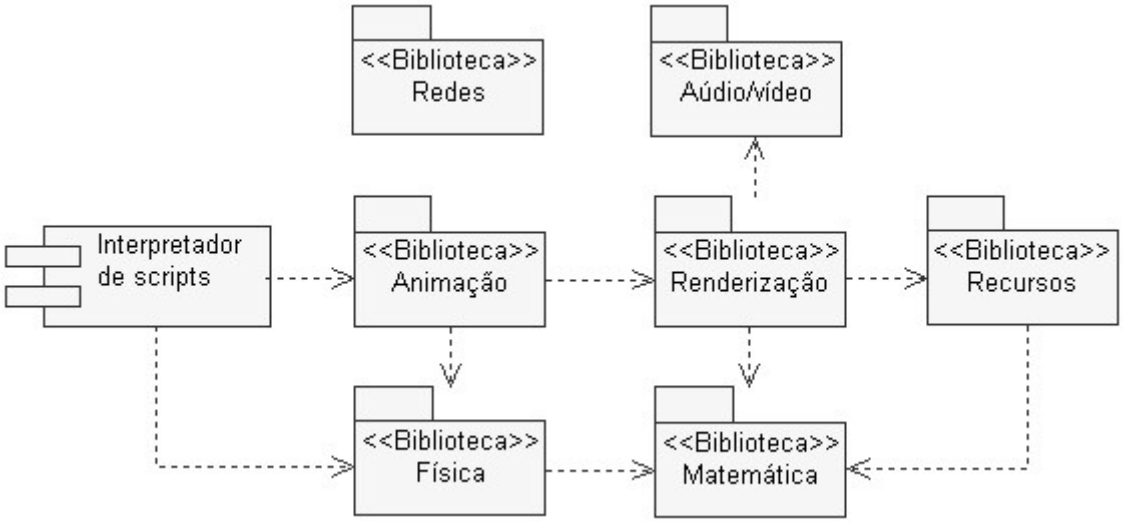
Sendo assim, devendo levar em conta os variados elementos que constituem um jogo eletrônico, vejamos alguns exemplos citados por Bassa et al. (2007): carregamento e detecção de colisão de objetos dentro do jogo; reprodução de áudio; e comunicação via rede. Para cada um destes é necessário que haja uma lógica de desenvolvimento e estruturação específicos para tornarem-se funcionais. É muito difícil e pouco provável que uma única equipe seja capaz de contemplar cada minucioso detalhe de todos os elementos que irão compor o jogo. O caminho mais lógico é fazer uso de ferramentas as quais já contemplam o tratamento de determinados recursos. O conjunto de tais recursos e ferramentas é denominado como *Engine de desenvolvimento*.

Segundo Novak,

> O termo ‘motor do game’ (em inglês, game engine) surgiu em meados da década de 1990 quando a id Software começou a licenciar o software básico utilizado nos games Doom e Quake, o que gerou um fluxo adicional de receita para a empresa. Isso permitiu que outros desenvolvedores projetassem seus próprios conteúdos e elementos gráficos utilizando o motor do game da id Software em vez de começar do zero. (2010, p.318)

A essência do uso de Engines para o desenvolvimento de jogos está no fato de estas poderem suprimir determinados requisitos para a sua implantação. Tratam-se de ferramentas “que podem ser consideradas como as de mais alto nível na cadeia do desenvolvimento, [...] abstraindo várias etapas do desenvolvimento através de editores e ferramentas gráficas” (FEIJÓ; PAGLIOSA; CLUA, 2006, p.128). Não seria necessário ao desenvolvedor, por exemplo, preocupar-se em garantir que sua aplicação consiga se comunicar para utilizar os recursos de hardware da máquina na qual ela estará rodando. Sendo assim, "os motores de jogos visam a suprir grande parte da demanda computacional, integrando diversas técnicas frequentemente necessárias nos jogos, simplificando muito a criação de um novo" (BESSA et al, 2007, p. 1). Isso é retratado através do diagrama na Figura 8, que ilustra os variados tipos de componentes que costumam estar encapsulados no núcleo de um motor para desenvolver jogos.

Figura 8: Diagrama de classes UML referente às diversas bibliotecas que podem estar presentes em uma engine de desenvolvimento de jogos



Fonte: FEIJÓ; PAGLIOSA; CLUA (2006, p. 129)

Todas as engines de desenvolvimento oferecem distintas ferramentas e recursos para trabalhar com os principais aspectos técnicos de um jogo. No entanto, cada engine de desenvolvimento existente atualmente é passível de oferecer não somente características únicas, mas também apresentar sua própria dinâmica de como ocorrerá o processo de desenvolvimento de um jogo. Uma determinada engine pode trabalhar em cima de uma específica linguagem de programação e funcionar de maneira a, estritamente, necessitar que o seu usuário possua conhecimentos de programação, ao passo que outra consiga abstrair tais conceitos de tal modo que o processo de construir um jogo possa se resumir a simplesmente selecionar e jogar elementos na tela.

Todas as engines apresentam pelo menos uma específica linguagem de programação primária a qual serve de base para o seu funcionamento e que pode ser utilizada para realizar a programação do jogo. jMonkeyEngine[35], por exemplo, trabalha especificamente com Java. Construct 2, Panda.js[36] e three.js[37]

[35] http://jmonkeyengine.org/

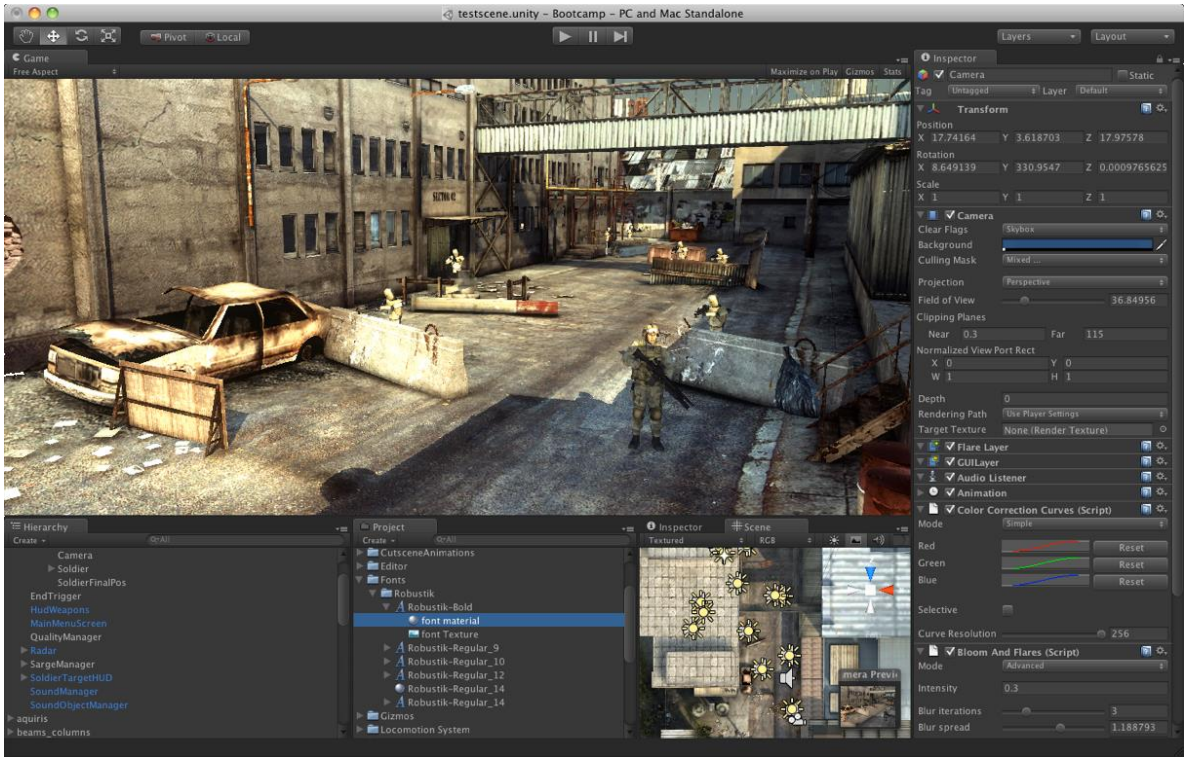
[36] http://www.pandajs.net/

[37] http://threejs.org/

implementam o seus jogos utilizando as tecnologias do HTML5. Corona[38] e LÖVE[39] utilizam Lua para codificar os games. Unity se destaca pelo uso integrado tanto de JavaScript quanto C#. Há também engines que vão além e apresentam sua própria linguagem de programação para a criação do jogo, sendo o caso do *Game Maker: Studio*[40] que utiliza sua própria linguagem de script: o *Game Maker Language* (GML).

Algumas engines como a Unity, a qual está apresentada na Figura 9, GameMaker ou Constuct 2, por exemplo, se destacam por apresentarem sua própria IDE para o desenvolvimento de jogos, ou seja, dispõem de um software com uma interface gráfica própria e ferramentas para o desenvolvimento de jogos empregando essa engine.

Figura 9: Interface da IDE presente no Unity



Fonte: BHOSLE (2013)

[38] http://www.coronalabs.com/products/corona-sdk/

[39] https://love2d.org/

[40] https://www.yoyogames.com/studio

As engines também diferem no que se diz respeito à como elas são distribuídas. Existem engines cujo uso é totalmente gratuito, podendo inclusive apresentar código-fonte aberto, o que possibilita a qualquer pessoa ver a programação escrita para construir essa ferramenta. É o caso das engines MonoGame[41], jMonkeyEngine e Blender Game Engine[42]. De maneira geral, a maioria das engines dispõem de uma versão gratuita, porém limitada, da ferramenta e de uma versão paga que oferece todos os recursos possíveis desta engine, sendo o caso do Unity, GameMaker e Construct2. A CryEngine[43] e a Unreal Developers Kit[44] podem ser utilizadas gratuitamente em sua completude para fins não-comerciais, caso contrário, é necessária a aquisição de uma licença de uso.

A CryEngine e a Unreal Engine em particular são interessantes por se tratarem de robustas ferramentas originalmente criadas visando o desenvolvimento de um específico jogo, mas que foram aprimoradas e disponibilizadas à venda para que outros desenvolvedores possam se beneficiar de seus recursos. Em contrapartida, é interessante mencionar também a existência de engines que são criadas para o uso exclusivo de uma determinada empresa de jogos, uma prática comum entre desenvolvedoras de grande porte, sendo o caso da *Frostbite*[45], desenvolvida para a criação de jogos pelos estúdios pertencentes à Electronic Arts, ou da *Luminous*[46], engine criada internamente pela Square-Enix para uso exclusivo no desenvolvimento de seus games.

Outro aspecto notável é que a maioria dessas engines oferecem suporte multiplataforma, ou seja, são capazes de compilar seus projetos em ao menos mais de um tipo de plataforma. Unity se destaca neste aspecto por poder exportar seus projetos para computadores, smartphones e até mesmo consoles de video game. Engines que utilizam HTML5 conseguem rodar seus jogos em praticamente qualquer

---

[41] http://www.monogame.net/

[42] http://www.blender.org/features/

[43] http://www.cryengine.com/community/downloads.php?view=detail&df_id=5310

[44] https://www.unrealengine.com/products/udk/

[45] http://www.frostbite.com/

[46] Uma demonstração técnica em vídeo detalhando as capacidades desta engine (edição de cenários, simulação de física e efeitos climáticos, iluminação, modelagem e animação 3D, efeitos visuais, etc.) e o seu uso no desenvolvimento do jogo Final Fantasy XV foi disponibilizada ao público: http://youtu.be/DDx7CRtVNFA

dispositivo com suporte à rede, graças ao aspecto ubíquo da própria linguagem. MonoGame, por sua vez, também é um exemplo interessante por proporcionar suporte a plataformas Linux, Mac OS e dispositivos móveis de projetos construídos utilizando C#, que originalmente somente seriam executáveis em plataformas da Microsoft.

4 DESENVOLVENDO UM JOGO ELETRÔNICO MULTIPLATAFORMA

Após muito ter dissertado sobre os aspectos teóricos e técnicos relacionados ao desenvolvimento de jogos eletrônicos, é chegado o momento de se aplicar os conceitos estudados na construção de algo concreto. Este capítulo será centrado na prática do desenvolvimento de um game com caráter multiplataforma.

4.1 Metodologia do desenvolvimento e análise

A metodologia a ser aplicada envolverá, primeiramente, o exercício da construção de um jogo eletrônico, isto é, a realização do design de um determinado game. Este design será utilizado como referência para o desenvolvimento de protótipos com capacidade multiplataforma, construídos com a finalidade de experienciar o processo de criação de um jogo eletrônico. Paralelo à confecção deste documento de design, serão feitos comentários referentes às decisões tomadas para a sua elaboração.

Para se obter um entendimento da ampla diversidade de ferramentas voltadas para o desenvolvimento de jogos e do impacto causado por essa escolha, serão escolhidas duas tecnologias com abordagens distintas para a criação de games. Serão abordados os recursos existentes nas ferramentas utilizadas e como estes foram aplicados no processo de desenvolvimento de um jogo eletrônico. Em particular, serão destacados aqueles que foram utilizados em benefício do desenvolvimento dos protótipos.

Uma vez que estes se encontrem funcionais, resta ainda realizar uma análise final do processo que envolveu a criação de ambos. Para tal, um determinado número de critérios de análise deverão ser elencados para auxiliar na reflexão sobre o que fora concretizado durante o desenvolvimento. Por meio destes parâmetros serão redigidos comentários sobre às capacidades das ferramentas utilizadas e dos percalços encontrados durante o seu uso.

4.2 Critérios de análise

Sendo que pretende-se fazer uma análise quanto ao processo de desenvolvimento do game dentro das determinadas ferramentas, é preciso estabelecer os parâmetros que serão usados para realizar essa reflexão. Sendo assim, foram elencados quatro critérios em particular, os quais foram julgados como sendo apropriados para determinar o foi positivo no desenvolvimento utilizado as ferramentas escolhidas e quais seriam os possíveis empecilhos atrelados ao seu uso.

O primeiro ponto de análise será centrado justamente nos *recursos existentes nas ferramentas* que genuinamente podem ser *utilizados em prol do desenvolvimento de um jogo eletrônico*. Trata-se de um ponto fundamental, que é avaliar se o que existe nestas ferramentas viabilizaria o seu uso para tal propósito.

Em sequência, pretende-se avaliar a facilidade de uso e de se programar jogos eletrônicos nas ferramentas escolhidas. Ou seja, determinar o quão prático é realizar o desenvolvimento de games por meio destas. Pretende-se aprofundar a análise das capacidades da tecnologia utilizada, avaliando se os recursos que elas oferecem contribuem para a construção do jogo.

O terceiro critério de análise é algo particularmente relevante ao escopo deste trabalho, *o suporte multiplataforma* presente nestas ferramentas. Um ponto crucial na escolha das tecnologias utilizadas neste trabalho é sua capacidade de permitir que o jogo criado possa ser jogado em múltiplas plataformas com o menos de trabalho possível. Serão apontadas em quais foi possível de fato testar o protótipo desenvolvido e avaliar aquilo que possibilitou a execução nesta plataforma.

Enfim, o último critério escolhido trata-se de algo sempre relevante no que se refere ao desenvolvimento de software, ganhando significado especial quando é relacionado a jogos eletrônicos: performance. A imersão é uma característica fundamental do ato de jogar e esta pode ser comprometida se ocorrerem falhas técnicas durante a sua execução. É necessário que haja o comprometimento dos desenvolvedores em garantir que o game consiga ser executado de maneira fluída em suas plataformas alvo. Mesmo nos pertencentes à séries de renome é comum haver um descaso com tal parâmetro de qualidade. O ano de 2014 foi um ano marcado pelo lançamento de jogos com performance aquém do esperado, com

Assassin's Creed Unity[47] e *Sonic Boom: Rise of Lyric*[48] sendo exemplos marcantes de jogos que foram disponibilizados ao público apresentando diversos problemas técnicos. Dessa forma, é importante que outra das tarefas fundamentais no desenvolvimento de jogos seja garantir eficiência na hora de executar o jogo. E isso deve ocorrer em todas as plataformas para as quais este é destinado.

4.3 Estabelecendo as ferramentas a serem utilizadas

No capítulo anterior, foram mencionados diversos tipos de tecnologias e ferramentas que podem ser utilizadas para realizar o desenvolvimento de jogos. Observando a grande variedade de opções disponíveis para realizar o trabalho de desenvolvimento de games, cada uma com sua própria abordagem, oferecendo diferentes funcionalidades e apresentando seus recursos e limitações. Uma das decisões mais importantes a ser tomada antes de começar a se criar um jogo eletrônico é estabelecer qual ferramenta será usada para tal propósito.

A escolha da tecnologia a ser utilizada no desenvolvimento irá determinar qual a experiência que o desenvolvedor terá para implementar o game que fora idealizado pelo processo de design. Essa escolha irá determinar as dificuldades e os desafios a serem enfrentados no desenvolvimento e irá estabelecer o potencial que poderá ser alcançado pelo jogo. Muitas poderiam ser as realidades do processo de criação de um jogo, dada a grande distinção existente entre o uso de uma tecnologia para outra.

Por isso, para melhor retratar essa variedade de ferramentas, discutir as divergências que existem entre elas e avaliar o impacto que o uso de uma determinada abordagem acarreta ao processo de criação de um game, iremos tomar como objeto de estudo duas tecnologias para a criação de jogos eletrônicos distintas. Para a implementação do jogo que fora estabelecido no capítulo anterior, serão utilizados duas tecnologias com recursos e funcionalidades diferentes entre si: *Game Maker: Studio* e HTML5.

---

[47] http://lmgtfy.com/?q=assassin%27s+creed+unity+glitches+slowdown

[48] http://lmgtfy.com/?q=sonic+boom+rise+of+lyric+framerate

4.3.1 Por que *Game Maker: Studio*?

Dentre as opções de ferramentas para o desenvolvimento de jogos, existem softwares que além de apresentarem recursos que auxiliam no processo de criação de um game, também se destacam por proporcionar recursos para que seus usuários possam criar jogos sem a necessidade de terem o conhecimento de programação, ou seja, sem que o usuário tenha conhecimento de como softwares de computador são criados. Através de sua interface gráfica, estas ferramentas abstraem a necessidade de escrever códigos e permitem que através de menus e ações como clicar e arrastar elementos na tela toda a lógica de funcionamento de um jogo eletrônico possa ser estabelecida. Softwares como o *Construct 2*, o *Clickteam Fusion*[49], os aplicativos da linha *RPG Maker*[50] e o *Game Maker: Studio* sempre se divulgam com grande enfoque nesta característica de poder criar games sem que se saiba programar.

Mas há um grande percalço nesta abordagem, na qual o desenvolvedor tenderá a estar limitado, de alguma maneira, por restrições impostas pela ferramenta com o propósito de facilitar o desenvolvimento. Ainda que a abstração da programação possa tornar o desenvolvimento de jogos algo atrativo para um maior número de pessoas, muitos ainda poderão preferir a liberdade e complexidade que pode ser alcançada através da escrita de variáveis e funções em código de programação. Assim, não é incomum que algumas destas ferramentas ofereçam uma interface para que os desenvolvedores escrevam o quanto de código eles quiserem e ainda se aproveitar as facilidades das ferramentas que abstraem a programação.

Dentre os exemplos citados anteriormente, o *Game Maker: Studio* é a ferramenta que mais se destaca em um dos parâmetros principais deste trabalho: o suporte multiplataforma. A quantidade de plataformas para as quais um projeto criado nesta ferramenta pode ser exportado, como pode ser observada na Tabela 2, pode ser dita como sendo invejável. Infelizmente, o acesso ao potencial multiplataforma da ferramenta fica restrito à aquisição de sua versão profissional mais a compra de licenças específicas para cada plataforma na qual se deseja compilar o projeto. No caso dos consoles da linha Playstation e do Xbox One, além

---

[49] http://www.clickteam.com/clickteam-fusion-2-5. Anteriormente conhecido como *Multimidia Fusion*.

[50] http://www.rpgmakerweb.com/

de comprar o módulo para compilação no *Game Maker: Studio*, também é necessário possuir uma Licença de Desenvolvedor que somente pode ser obtida diretamente com as fabricantes destes consoles (Sony e Microsoft, respectivamente).

Tabela 2 - Lista de Plataformas Suportadas pelo *Game Maker: Studio*

Plataforma	**Tipo de Licença**
Windows Desktop	*Standard* (Gratuita)
Windows 8 App	*Profissional* (US$149.99)
Mac OS X	Mac OS X Module (US$99.99)* *Requer já possuir a licença *Profissional*
Ubuntu	Ubuntu Linux Module (US$99.99)* *Requer já possuir a licença *Profissional*
HTML 5	HTML5 Module (US$199.99)* *Requer já possuir a licença *Profissional*
Android	Android Module (US$299.99)* *Requer já possuir a licença *Profissional*
iOS	iOS Module (US$299.99)* *Requer já possuir a licença *Profissional*
Windows 8 Phone	Windows Phone 8 Module (US$299.99)* *Requer já possuir a licença *Profissional*
Tizen[51]	Tizen Module (US$199.99)* *Requer já possuir a licença *Profissional*
Playstation 4	PlayStation® 4 Module (US$299.99)* *Requer já possuir a licença *Profissional* e possuir uma Licença de Desenvolvedor Playstation
Playstation Vita	PlayStation® Vita Module (US$299.99)* *Requer já possuir a licença *Profissional* e possuir uma Licença de Desenvolvedor Playstation

[51] Sistema operacional de código aberto para smartphones, tablets, smart Tvs, notebooks, veículos e diversos outros tipos de sistemas embarcados. Desenvolvido pela Linux Foundation em parceria com a Samsung, Panasonic e a Intel. https://www.tizen.org/

Playstation 3	PlayStation® 3 Module (US$299.99)* *Requer já possuir a licença *Profissional* e possuir uma Licença de Desenvolvedor Playstation
Xbox One	Xbox® One Module (US$299.99)* *Requer já possuir a licença *Profissional* e possuir uma Licença de Desenvolvedor Xbox

Fonte: http://www.yoyogames.com/studio

Apesar de suas capacidades multiplataforma envolverem custos, isso não desfavorece o *Game Maker: Studio* como objeto de estudo. Com sua primeira versão lançada em 1999 pelo cientista da computação Mark Overmars, o *Game Maker* viria a se tornar um conhecido aliado dos desenvolvedores de jogos em virtude de sua interface que preza pela rapidez e praticidade na criação de games. Seu trabalho perpetuou-se em consecutivas atualizações da ferramenta até meados de 2008, quando a deixou aos cuidados da recém-formada companhia *Yoyo Games*[52], a qual se dedicou ao aprimoramento e refinamento da proposta de desenvolvimento de jogos originalmente idealizada por Overmars.

Os anos de trabalho resultaram em uma robusta e acessível ferramenta que é considerada como referência ao se falar desta área, sendo competente e poderosa o suficiente para desenvolver projetos mais complexos e de maior escala. Este software recebe constantes melhorias e novas funcionalidades. Isso é evidenciado tanto pela quantidade de compiladores existentes como também em outras de suas funcionalidades adicionais. Um exemplo é o YoYo Compiler (YYC) o qual foi concebido para aprimorar o desempenho de funções presentes no desenvolvimento de jogos que requerem um uso mais intenso da CPU, como: processamento de Inteligência Artificial, iluminação em tempo real, simulação de física mais apurada, deformações geométricas em tempo real, detecção de colisão entre objetos e manipulação de dados (SHAUL, 2013, tradução nossa). Um outro recurso mais recente da ferramenta foi a criação do GameMaker: Player[53] - uma loja digital específica para projetos desenvolvidos nesta ferramenta, aos moldes das lojas de

[52] http://www.yoyogames.com/

[53] https://player.yoyogames.com/

aplicativos presentes em smartphones e de outras lojas de distribuição digital de games (como a *Steam*[54] ou a *Good Old Games*[55]).

4.3.2 Por que HTML5 e Javascript?

Conforme já fora comentado, com o contínuo aprimoramento do HTML no decorrer dos anos, a linguagem foi se tornando cada vez melhor estruturada e possibilitou a utilização conjunta de outras tecnologias como Javascript e CSS para a geração de páginas web com mais funcionalidades e melhor usabilidade. Seguindo as perspectivas de uma web mais centrada em recursos multimídia e adaptável para dispositivos móveis, as novidades do HTML5 objetivam torná-lo o mais capacitado e independente o possível, em contraponto a um passado no qual era necessário instalar aplicativos prioritários feitos por terceiros, como o Adobe Flash por exemplo, para possibilitar a existência de páginas mais dinâmicas e poder desfrutar de jogos eletrônicos através da Web. O processo de desenvolvimento do HTML5 está introduzindo novas perspectivas para o desenvolvimento web, de tal modo estabelecendo a sua grande relevância como uma plataforma de desenvolvimento para uma grande variedade de aplicações, dentre as quais podemos incluir os jogos eletrônicos.

Essa integração entre a estrutura HTML e o poder de aplicativos desenvolvidos com Javascript já pode ser amplamente apreciada através de sites como Codepen[56], que disponibilizam diversos protótipos de aplicações complexas que rodam diretamente no navegador de sua escolha. Outra mostra da capacidade desta linguagem está no seu uso no desenvolvimento de um emulador de jogos de fliperama, possibilitando aos visitantes do site *Internet Arcade*[57] experienciar games clássicos em seu navegador.

A sua relevância no atual cenário da tecnologia já justifica tomar o HTML5 como objeto de estudo. Mas o que torna atrativa a ideia de sua utilização como uma plataforma de desenvolvimento de jogos é a perspectiva de que qualquer desenvolvedor web possui à sua disposição uma ferramenta para produzir games,

---

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

[55] http:// www.gog.com/

[56] http://codepen.io/popular/

[57] https://archive.org/details/internetarcade

com capacidade suficiente para criar até mesmo games mais complexos, sem a necessidade da aquisição de licenças de uso e com um instantâneo suporte multiplataforma, graças ao seu caráter de linguagem interpretada através de softwares amplamente disponíveis em vários tipos de dispositivos.

Embora você não tenha a sua disposição todas as comodidades oferecidas por uma suíte completamente voltada para o desenvolvimento de jogos como o *Game Maker: Studio*, isso não necessariamente tornaria o desenvolvimento com HTML5 inviável. Apenas com o uso de Javascript já há matéria prima suficiente para se concretizar um jogo. Não obstante, existem diversas ferramentas adicionais para complementar o desenvolvimento com esta linguagem. Cada vez mais é proeminente o surgimento de engines de games construídas em HTML5 e Javascript, como também o uso da API de renderização gráfica em Javascript nativo: o WebGL[58]. Estas podem ser usadas para apoiar projetos mais complexos, enquanto mantém o fundamento de se construir jogos que rodem nos navegadores de diferentes dispositivos.

Mesmo com a existência de tais recursos e ainda que também existam engines para se desenvolver jogos para HTML5 (tendo alguns exemplos tendo citados no decorrer do trabalho, assim como o próprio *Game Maker: Studio* apresenta um módulo para compilar o projeto para HTML5), focaremos apenas nas funcionalidades padrões do Javascript para poder-se analisar a sua capacidade para o desenvolvimento de games.

4.4 Estabelecendo o design do jogo e definindo o protótipo

O jogo eletrônico estabelecido para os propósitos deste trabalho é um jogo de ação 2D, que almeja proporcionar ao jogador um entretenimento mais casual e momentâneo. Pode-se se dizer que o jogo em questão apresenta uma temática *arcade*, isto é, uma jogabilidade em que o jogador tentará manter uma partida pelo maior tempo possível almejando obter a maior quantidade de pontos. Esta é uma temática bastante próxima dos jogos de fliperama ou de games que costumam ser lançados para dispositivos móveis, inclusive sendo muito proeminente entre os desenvolvedores de jogos *indie*.

[58] https://www.khronos.org/webgl/

Para elaborar as mecânicas de jogo deste projeto foi preciso levar em consideração as implicações de seu aspecto principal: a portabilidade entre diferentes tipos de hardware. Assim, sua jogabilidade foi pensada de modo que pudesse ser facilmente executada tanto em computadores (através de mouse e teclado) como em dispositivos móveis (através de toques em sua tela). Esta também foi pensada de forma a ser favorável a partidas de curta duração, de forma a corresponder aos hábitos comuns de jogadores em dispositivos móveis.

A partir desta mentalidade, foi determinado que o game aplicaria uma jogabilidade que utiliza os cliques do mouse ou os toques em uma tela tátil como recurso para derrotar inimigos que estão presentes em seu cenário. O seu objetivo ficou centrado em se tentar eliminar o máximo possível de inimigos na tela antes que estes venham a zerar os pontos de vida do jogador. O jogador também possui um marcador para contabilizar pontos adquiridos a cada inimigo que é derrotado (Pontuação) e uma variável para controle de sua progressão no jogo (Nível). Outras ideias para elementos adicionais da jogabilidade surgiram no decorrer da construção do DDG, como o conceito de apresentar diferentes tipos de magias mais poderosas com restrições de uso e fases com inimigos gerados aleatoriamente e que apresentam fraquezas e resistências específicos a estas magias. Tais conceitos foram idealizados para que se pudesse atribuir ao jogo um grau de desafio mais elevado e, consequentemente, a perspectiva de um jogo mais interessante, engajante e divertido.

Para poder simplificar o fluxo de execução do jogo, toda a sua ação transcorrerá em um único cenário, ou em termos mais técnicos, em uma única tela da aplicação. Uma vez tendo atravessado as telas de menus e confirmado o início de uma nova partida, a aplicação deverá inicializar o fluxo de jogo e executar continuamente uma rotina para manter este fluxo em execução até que a partida seja finalizada. Este fluxo envolve inicializar os atributos básicos de vida e pontuação do jogador e determinar o primeiro conjunto de inimigos a ser enfrentado pela personagem principal do jogo. A partir deste momento, quando não houverem mais inimigos na tela e enquanto os pontos de vida do jogador não chegarem a zero, o jogo deverá continuar gerando novos níveis, apresentando novos conjuntos de inimigos para serem enfrentados.

Considerando o foco em criar uma experiência mais casual, sem requerer grande comprometimento por parte do jogador, este projeto não teve a perspectiva

de usar o meio dos jogos eletrônicos para contar uma narrativa complexa. Mas uma breve história está presente no escopo da aplicação, com o propósito de contextualizar o universo em que as ações do jogo ocorrem. Partindo de um cenário de fantasia medieval, foi estabelecido que o jogador estará no controle de uma habilidosa feiticeira, a qual foi escolhida para lutar contra o exército de monstros comandado pelo mestre das artes ocultas: *o Lorde das Trevas*.

Para dar validade a este exercício de criação de games, houve o esforço para idealizar o jogo "Desafio do Lorde das Trevas" em sua completude, estabelecendo todas as funcionalidades que iriam constituí-lo e possivelmente torná-lo um produto interessante. Mais do que apenas exercitar o design de games, porém, o DDG em questão também foi pensado de modo a ser aplicado para a criação dos protótipos a serem desenvolvidos com o *Game Maker: Studio* e em HTML5. Não há tempo útil e nem cabimento para se realizar a implementação de tudo que está descrito neste DDG, considerando que o trabalho objetiva o uso e análise destas ferramentas e não a construção de um jogo completo. Foi por essa razão que se optou pela criação de protótipos, os quais utilizariam apenas algumas poucas características descritas no DDG como referência de implementação ao se trabalhar com as tecnologias escolhidas.

Deste modo, é estabelecido que os protótipos construídos para a análise das ferramentas focarão na apresentação das diferentes telas da aplicação e no recurso básico de jogabilidade de atacar os inimigos através de cliques do mouse ou de toques na tela. Essa jogabilidade caracteriza um fluxo de jogo com o objetivo de conseguir chegar ao nível mais alto e à maior pontuação possível. No que se refere às responsabilidades na construção do jogo, foram estipuladas as tarefas de se realizar toda a programação dos protótipos e de desenhar os elementos gráficos que irão compor o jogo. No que se refere à sonoplastia do jogo, foram utilizadas músicas criadas por terceiros que estão disponíveis na Internet e efeitos sonoros sintetizados através do software *Bfxr*[59].

A descrição aprofundada das características do game em questão está presente na forma de seu Documento de Game Design completo, o qual pode ser visto no Apêndice 1 deste trabalho.

---

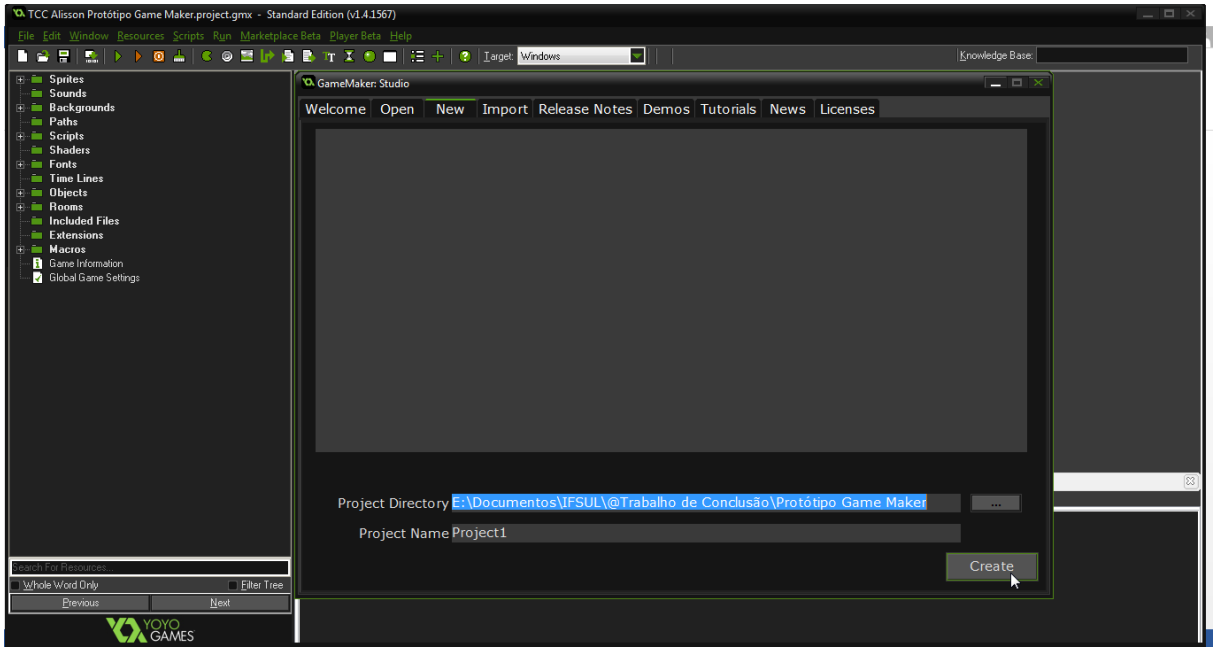
[59] http://www.bfxr.net/

4.5 Desenvolvimento de jogos utilizando *Game Maker: Studio*

O fundamento para se usar uma determinada engine para o desenvolvimento de jogos está nos benefícios que ela poderá oferecer aos desenvolvedores. É esperado que ela contemple requisitos como: permitir da melhor forma possível o encapsulamento de códigos que podem ser reutilizados para diversos projetos com alguma semelhança entre si; possibilitar a integração entre os recursos de arte com a programação; Tornar o desenvolvimento o mais independente possível de plataformas e tecnologias; Fazer com que a aplicação seja capaz de usar o máximo possível os recursos de hardware disponíveis; e permitir que o desenvolvimento possa ser gerenciado de maneira adequada (FEIJÓ; PAGLIOSA; CLUA, 2006).

É com base em princípios como estes que ferramentas como o *Game Maker: Studio* procuram maneiras de dispor aos seus usuários a melhor metodologia de desenvolvimento possível. É importante que a interface gráfica tenha a capacidade de ser convidativa para os usuários mais inexperientes e de proporcionar praticidade e produtividade durante a criação do projeto, afinal, este é um fator decisivo que fará um usuário dar preferência a esta ferramenta em detrimento de outras de natureza similar. Sendo assim, observemos a Figura 10, a qual retrata a interface do *Game Maker: Studio* ao se abrir o software e iniciar a criação de um novo projeto.

Figura 10: Interface gráfica do Game Maker: Studio, com uma janela aberta para se criar um novo projeto



Fonte: Autor.

Em meio ao escopo geral do aplicativo, o primeiro aspecto que deve ser ressaltado é a árvore de diretórios existente dentro de um projeto. A metodologia do *Game Maker: Studio* estabelece uma catalogação em grupos distintos para cada tipo de recurso que compõe um jogo eletrônico, atribuindo um diretório específico para cada um deles. Por meio desta estrutura, o desenvolvedor poderá inserir e gerenciar cada elemento que será utilizado no decorrer de seu jogo. Para auxiliar no gerenciamento do projeto, o usuário deverá atribuir um nome identificador único e exclusivo para cada elemento utilizado no jogo e poderá criar subdiretórios para os agrupar da maneira que julgar ser mais conveniente. Na versão *Standard* v1.4 da ferramenta, um recurso do projeto deverá se enquadrar em um dentre os seguintes treze grupos:

- **Sprites**: o termo inglês *sprite*, no contexto do desenvolvimento de jogos eletrônicos, refere-se a todo e qualquer elemento 2D que é utilizando dentro do jogo, ainda que seja bastante comum o uso deste termo para se referir especificamente às imagens dos personagens existentes dentro do jogo. Sendo assim, este diretório irá conter a maioria dos elementos gráficos que irão compor o projeto;
- **Sounds**: diretório voltado para a sonoplastia da aplicação. Todas as músicas de fundo e efeitos sonoros a serem empregados no game serão postos neste diretório;
- **Backgrounds**: diretório especificamente voltado para agrupar imagens que terão comportamento predominantemente estático no decorrer do jogo, como uma imagem de fundo que irá preencher a tela. Ao contrário dos *Sprites*, que são imagens ou conjuntos de imagens que serão utilizados para criar Objetos que apresentarão algum tipo de interação no fluxo do jogo, os backgrounds (como o próprio termo da língua inglesa já revela) são os elementos gráficos serão utilizados para definir os cenários das *Rooms* da aplicação. O *Game Maker: Studio* poderá estampar uma imagem inteira no fundo de uma *Room*,

ou poderá carregar e configurar uma imagem de *tileset*[60] para realizar manualmente a construção do cenário;

- **Paths**: trata-se de configurações para estabelecer trajetos pré-definidos que um Objeto poderá utilizar para se movimentar ou realizar uma animação durante a execução do game. Essencialmente, um elemento Path nada mais é do que um vetor que armazena um conjunto de posições horizontais (X) e verticais (Y) da tela do game e um determinado intervalo de tempo. Ao executar o game, a *engine* se encarregará de calcular as distâncias entre cada um destes pontos e realizar a movimentação do Objeto entre cada ponto deste trajeto;
- **Scripts**: grupo no qual pode-se concentrar as funções escritas utilizando a linguagem GML desta ferramenta. Ainda que o usuário não precise obrigatoriamente colocar todo o código-fonte escrito para o jogo neste grupo, os trechos de código-fonte inseridos aqui presentes poderão ser facilmente referenciados e executados a qualquer momento durante a execução do game;
- **Shaders**: *shaders* são um tipo especial de script voltado especificamente para manipulação dos elementos gráficos sendo renderizados na tela. No geral, este tipo de recurso é voltado para a programação de efeitos especiais;
- **Fonts**: diretório reservado para a importação de arquivos de fontes tipográficas, que poderão ser utilizadas pelas funções do *Game Maker: Studio* que desenham mensagens de texto na tela da aplicação. Um aspecto curioso da implementação de fontes pela ferramenta é que estas não são usadas em sua forma nativa, como vetores. Ao carregar um arquivo de fonte, o usuário determina o tamanho, estilo e conjunto de caracteres a ser utilizado. Com base nestas informações, o *Game Maker: Studio* cria um conjunto de imagens, passando a tratar cada caractere como se fosse um arquivo de imagem comum. É importante estar ciente desta característica, pois isso faz

[60] Tilesets são um conjunto de imagens agrupado em um único arquivo que ao ser carregado pode ser quebrado em várias partes de um tamanho fixo. Tais partes individuais podem ser aplicadas para construir a tela de um jogo, como se você estivesse montando um quebra-cabeça. Ao aplicar esta técnica no desenvolvimento de jogos é possível, por exemplo, construir e salvar os dados de uma fase em um game como uma matriz, tendo em cada posição é salva a referência para um pedaço específico do tileset.

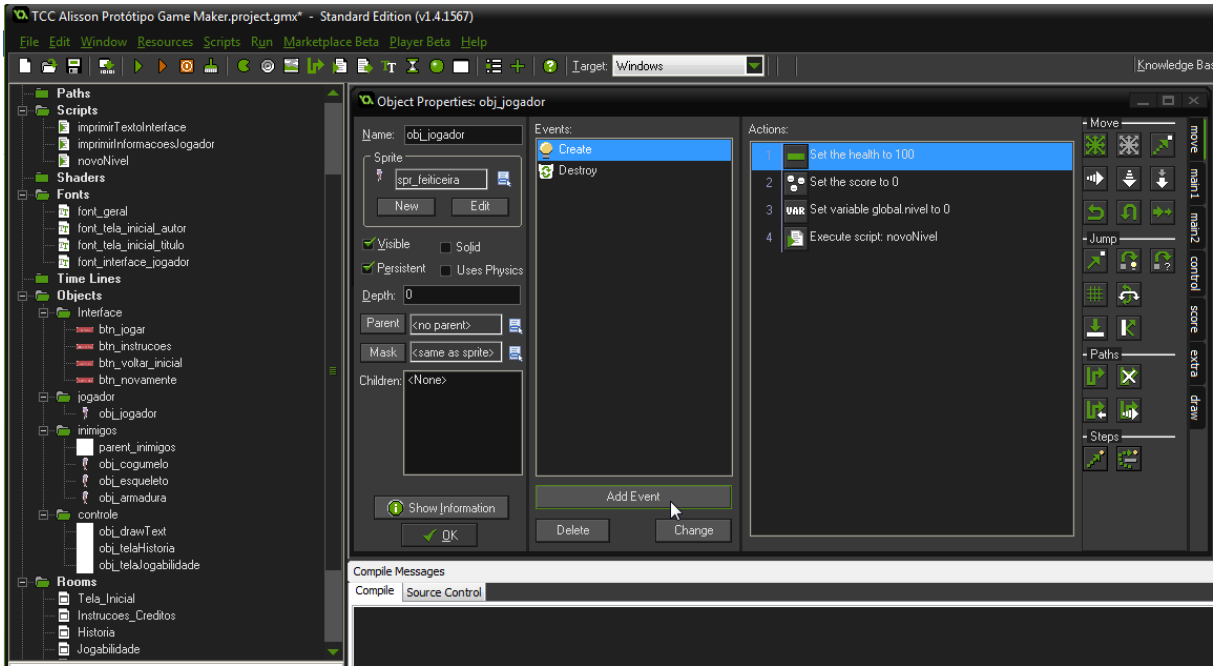
com que a fonte perca o seu caráter vetorial: a capacidade de poder manipular seu formato sem que ocorram distorções e modificar seu tamanho sem que ocorra perda na qualidade da imagem;

- **Time Lines**: possibilita estabelecer uma linha do tempo, um recurso do *Game Maker: Studio* no qual é possível especificar uma sequência de eventos e ações a serem executados durante um determinado momento do fluxo do jogo;
- **Objects**: o recurso fundamental de um projeto no *Game Maker: Studio*. A categoria *Objects* apresenta a real implementação dos elementos que irão dar vida ao seu game. Aqui, os recursos das categorias anteriores, como os *Sprites* ou os scripts, serão de fato utilizados para construir objetos dotados de comportamentos específicos. Toda a lógica de execução do jogo estará atrelada à existência e interação destes Objetos, sendo que deverão haver objetos para estabelecer o personagem que irá responder aos comandos do jogador ou os inimigos que irão desafia-lo ou ainda objetos que serão apenas comandos lógicos a serem executados em segundo plano durante o jogo;
- **Rooms**: nesta categoria estarão agrupadas todas as telas de um game. Há diferentes tipos de comportamentos que podem ser atribuídos às telas. Elas não somente correspondem aos mapas ou às fases nas quais o jogador irá se aventurar, mas sim a toda e qualquer interface pela qual o jogador irá interagir com o jogo, como uma tela inicial com botões para se iniciar o jogo ou realizar configurações, por exemplo;
- **Included files**: trata-se de qualquer tipo de arquivo adicional que, por uma razão ou outra, deverá ser distribuído junto com o arquivo executável do jogo.
- **Extensions**: tratam-se de arquivos de código-fonte externos que podem ser importados e utilizados dentro de seu projeto. Estes podem ser outros projetos do *Game Maker* (.gml), arquivos Javascript (.js) ou uma biblioteca de vínculo dinâmico do Windows (.dll);
- **Macros**: na programação para computadores é possível trabalhar com variáveis (um elemento que guarda algum tipo de valor manipulável por um breve momento durante a execução do código) e com constantes (valores que são fixos e estão presentes a qualquer momento na execução da

aplicação). *Macros* é o termo utilizado pelo *Game Maker: Studio* para se referir a todas as constantes presentes neste projeto.

Para cada um destes elementos há uma interface específica para a sua configuração. Toda vez que um determinado tipo de elemento for criado ou editado, uma nova janela dentro do *Game Maker: Studio* surgirá, disponibilizando todas as opções de configuração referentes à sua categoria, podendo ser visto em particular na Figura 11 a janela do editor de Objetos.

Figura 11: Interface gráfica do Game Maker: Studio, com a janela do editor de Objetos aberta



Fonte: Autor.

Essencialmente, a construção de um jogo utilizando o *Game Maker: Studio* ocorre através da criação de Objetos, os quais fazem uso de todos os demais recursos presentes no projeto e que serão programados para executar algum tipo de comportamento. A programação destes comportamentos poderá ocorrer através da interface gráfica do editor de Objetos ou pela criação de scripts utilizando os recursos da linguagem GML.

De tal modo o desenvolvedor poderá criar um Objeto para representar o personagem controlado pelo jogador (que terá um determinado *Sprite* atribuído a ele e será programado para executar alguma rotina quando uma tecla do teclado é

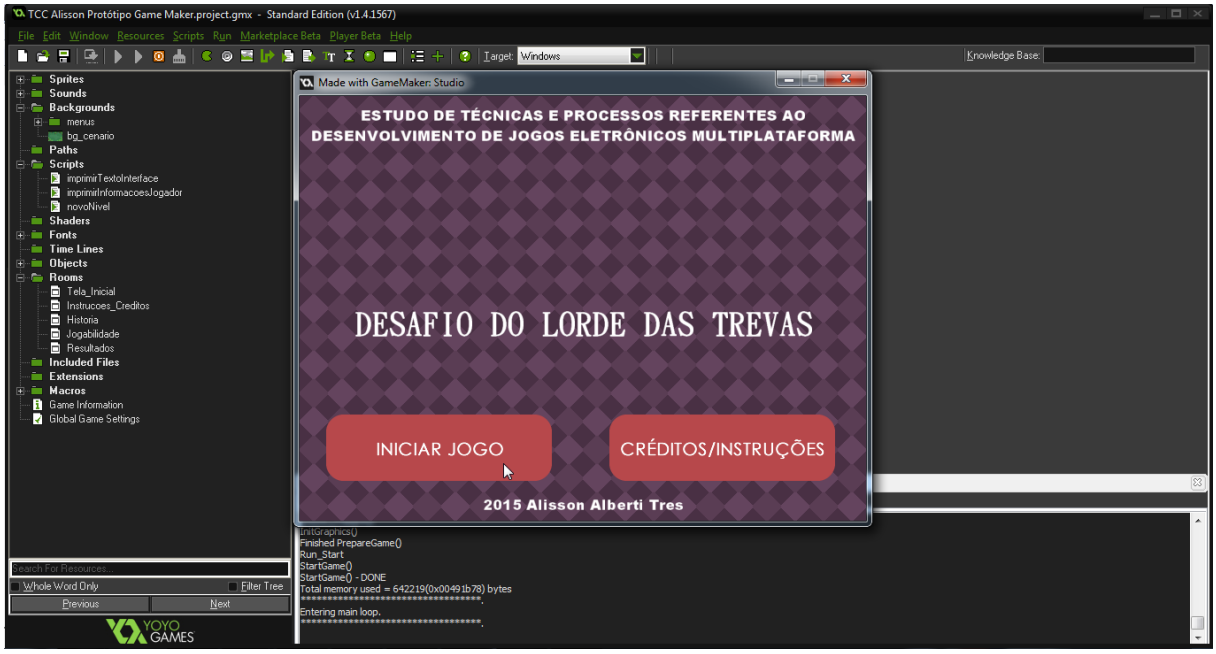
acionada ou quando ocorre uma colisão com um inimigo) e poderá criar Objetos ocultos (que não possuem um *Sprite* associado a ele e, portanto, não é fisicamente visível na tela, mas ainda assim está presente na execução do jogo) para realizar rotinas como automaticamente inserir novos inimigos na tela a cada 5 segundos, por exemplo. Por fim, através do editor de *Rooms*, os objetos serão posicionados em uma determinada tela e estarão aplicados no jogo.

O *Game Maker: Studio* pode ser dito como sendo uma tecnologia orientada à eventos. O funcionamento de uma aplicação criada com esta tecnologia envolve a execução de uma sequência de ações específicas quando um determinado tipo de evento acontece. A variedade de eventos com as quais podemos trabalhar é extensa: podemos realizar ações quando um objeto é criado; quando um objeto é destruído; quando um determinado botão do mouse é acionado; quando uma determinada tecla do teclado é pressionada; no momento em que a aplicação é iniciada; no momento em que a aplicação é finalizada; no momento em que uma determinada *Room* é iniciada; no decorrer de um determinado intervalo de tempo; etc.

A qualquer momento durante o desenvolvimento, é possível realizar a execução do projeto no estado em que se encontra para melhor acompanhar o seu andamento, fazer testes e identificar possíveis falhas. Uma vez finalizada a criação e todos os Objetos e a disposição dos mesmos nas Rooms do projeto, restaria apenas realizar o processo de compilação para gerar um arquivo executável do jogo. Para ambas estas ações o usuário poderá escolher uma dentre as plataformas alvo suportadas pelo *Game Maker: Studio* que estiverem disponíveis. Durante o processo, o painel de compilação exibirá detalhes de todo o procedimento sendo realizado, fazendo os devidos apontamentos caso alguma falha ocorra durante a compilação. Não havendo erros no projeto, a janela com o jogo em execução surgirá e o desenvolvedor poderá contemplar os resultados de seu trabalho.

Uma visão geral de como é o arquivo do projeto do protótipo desenvolvido no *Game Maker: Studio*, contendo configuração de objetos e código-fonte dos scripts, está disponível do Apêndice 2. Na Figura 12, pode-se ver o executável para Windows do protótipo gerado pelo *Game Maker: Studio*.

Figura 12: Protótipo construído no Game Maker: Studio sendo executado em um computador com Windows



Fonte: Autor.

4.6 Desenvolvimento de jogos utilizando HTML5

O HTML5 não somente trouxe uma atualização para que os padrões web possam aproveitar o máximo da tecnologia atual. Este também inaugurou novas perspectivas do que é possível realizar através de uma página na Internet. Dentre elas, destacamos que agora os softwares navegadores são capazes de rodar jogos eletrônicos de maneira independente.

O elemento chave responsável pelo crescente destaque do HTML5 como uma tecnologia para desenvolvimento de games é um novo recurso o qual foi introduzido com a recente reformulação da linguagem: o *Canvas*. *Canvas* em inglês significa tela, em referência às telas utilizadas por artistas para realizar pinturas. Assim como artistas usam sua tela para jogar cores e realizar desenhos, o elemento *Canvas* possibilita aos desenvolvedores web “delimitar uma área para criação dinâmica de imagens, como gráficos estáticos, jogos e gráficos dinâmicos e imagens em geral criadas com linguagem de programação dinâmica” (SILVA, 2011, p. 141).

A essência dos games em HTML5 é que pelo intermédio do recurso *Canvas* aliado à programação em Javascript é possível gerar e manipular elementos gráficos dentro destes softwares que são utilizados para acessar a Internet. Através da API

do *Canvas* os navegadores podem, em tempo real, realizar a renderização de diversos tipos de elementos gráficos, sendo capaz de: desenhar texto; configurar e desenhar formas geométricas como retângulos, círculos, linhas e outros tipos de formas através do uso de vetores; e desenhar imagens a partir da importação de arquivos presentes no servidor.

Uma vez conseguindo imprimir as imagens do seu jogo na tela do *Canvas*, você pode prosseguir com o trabalho de programação em Javascript para realizar a manipulação destas imagens, sendo possível gerar animações e reconhecer cliques do mouse ou o pressionar de uma determinada tecla do teclado. Consequentemente, isso torna o *Canvas* passível de ser programado para a exibir a movimentação do personagem em um jogo, estabelecer um sistema de colisão entre os objetos ali presentes e implementar a simulação dos elementos necessários para o funcionamento de sua jogabilidade. Ressalta-se, porém, que tais implementações não são tão simples de serem realizadas quanto possam parecer à primeira vista.

Naturalmente, o primeiro passo no desenvolvimento de uma aplicação para a web é abrir uma ferramenta de edição de texto apropriada para se trabalhar com programação. Há muitas opções interessantes para se escolher, como o *Netbeans*[61] ou o *Sublime Text*[62], que dispõem de recursos para auxiliar o programador a escrever o código-fonte e a gerenciar os arquivos presentes em seu projeto.

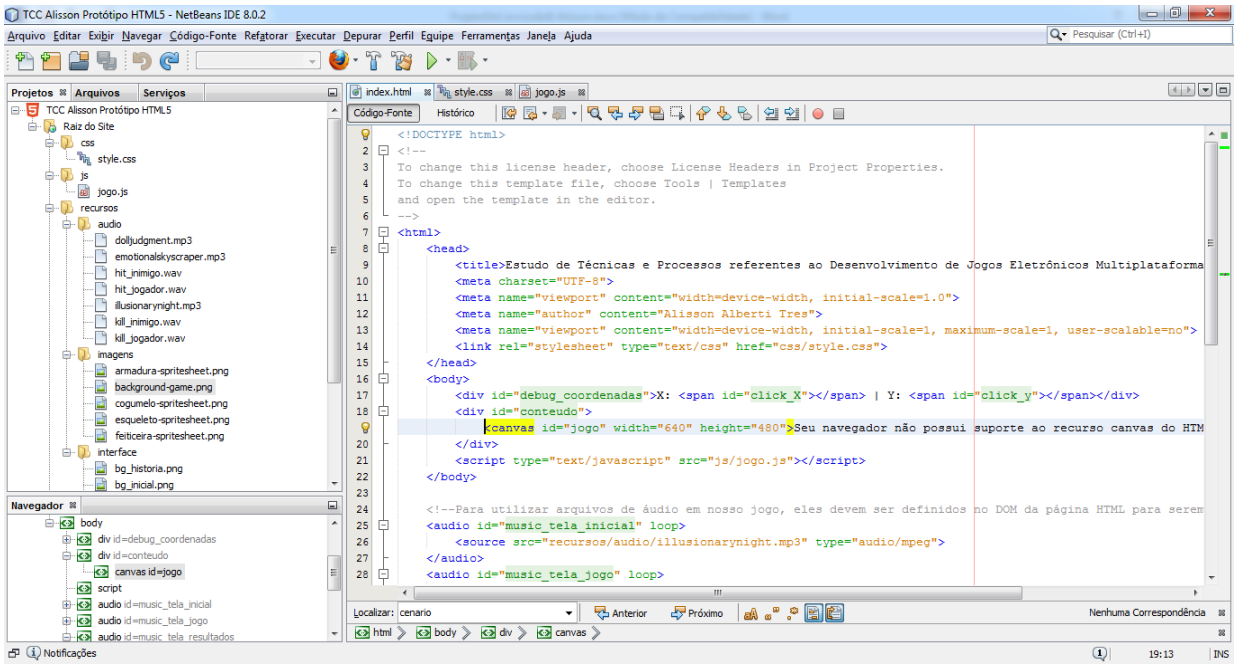
Como já é de costume no desenvolvimento para web, deve-se estabelecer para o projeto uma árvore de diretórios para organizar suas páginas HTML e todos os recursos que serão empregados no decorrer de sua execução. Você precisará ter pelo menos um arquivo HTML para ser a página inicial de sua aplicação e para dispor o elemento *Canvas*. Quando esta página é acessada, é preciso que ela esteja configurada para carregar os arquivos os scripts pertinentes à execução do jogo e os demais arquivos necessários, como um arquivo CSS que é sempre indicado para se estabelecer as configurações visuais de uma página web. Na Figura 13 podemos visualizar o software *Netbeans* exibindo a árvore de diretórios do protótipo em HTML5 e o código-fonte de seu arquivo *index.html*.

---

[61] https://netbeans.org

[62] http://www.sublimetext.com

Figura 13: Árvore de diretórios do protótipo em HTML5 e o código-fonte da página inicial



Fonte: Autor

É neste momento em que o desenvolvimento deixa de ser uma aplicação web comum e começa a explorar as peculiaridades do desenvolvimento de games. Para que possamos ter um jogo em HTML5, é necessário que existam uma série de rotinas em Javascript para estabelecer o fluxo do jogo, controlar os estados e as telas da aplicação e executar a lógica da jogabilidade. Tão logo a página web já tiver carregado, deve-se executar a função responsável pelo início deste fluxo.

Para que se possa gerar imagens na tela de um Canvas, é preciso determinar o contexto de sua superfície de desenho: se iremos trabalhar com elementos gráficos bidimencionais (2D) ou tridimencionais (3D). Isso é determinado ao se instanciar um objeto de contexto, uma variável a qual será utilizada para chamar as funções responsáveis pelo ato de imprimir tais elementos gráficos em um *Canvas*.

Uma peculiaridade relativa ao funcionamento do *Canvas* é que este não guarda nenhuma informação ou qualquer referência aos elementos gráficos nele contidos. O *Canvas* literalmente é apenas um vetor bitmap que simplesmente recebe e exibe imagens e elementos gráficos dentro de si. Por conta disto, não seria possível apagar ou modificar um elemento específico. Ao se inserir novos elementos em um Canvas, estes são simplesmente posicionados por cima de seja lá o que já

exista dentro dela. E a única forma de se retirar um recurso gráfico de uma canvas é limpado tudo que está nela presente.

Por mais que isso soe como algo que inviabiliza o uso da *Canvas* para se construir games ou mesmo qualquer tipo de aplicação interativa, a realidade é que existe um artifício bem simples para tornar a Canvas um palco para animações e interação com imagens. Para que a *Canvas* possa ser utilizada para o propósito de um jogo, é necessário que ela seja constantemente atualizada dentro de um intervalo de tempo. Isso pode ser facilmente através de variáveis, funções e objetos em Javascript. Se a Canvas não realiza nenhum controle dos elementos nela presentes, então cabe ao programador fazer o serviço de manualmente estabelecer e controlar todos os dados referentes ao posicionamento ou, no caso de um jogo eletrônico, dos comportamentos que são visualizados através das imagens presentes na Canvas.

Levando em consideração tudo isso, podemos pensar a lógica de execução de nosso protótipo da seguinte forma: Nosso código Javascript terá variáveis que guardam informações como o estado atual da aplicação ou as características do personagem controlado pelo jogador (quantidade de pontos de vida, nível atual, pontuação, imagem dos *sprites* que deverão aparecer na *Canvas*). Enquanto a aplicação estiver no estado de jogar, é preciso executar uma rotina para verificar se o jogador interagiu com a aplicação, verificar o estado em que cada objeto se encontra e chamar objeto de contexto para exibir os gráficos na tela do *Canvas*. Essa rotina permanecerá em execução enquanto a variável que guarda o valor dos pontos de vida do jogador não for menor ou igual a zero, evento que marcaria o fim do jogo.

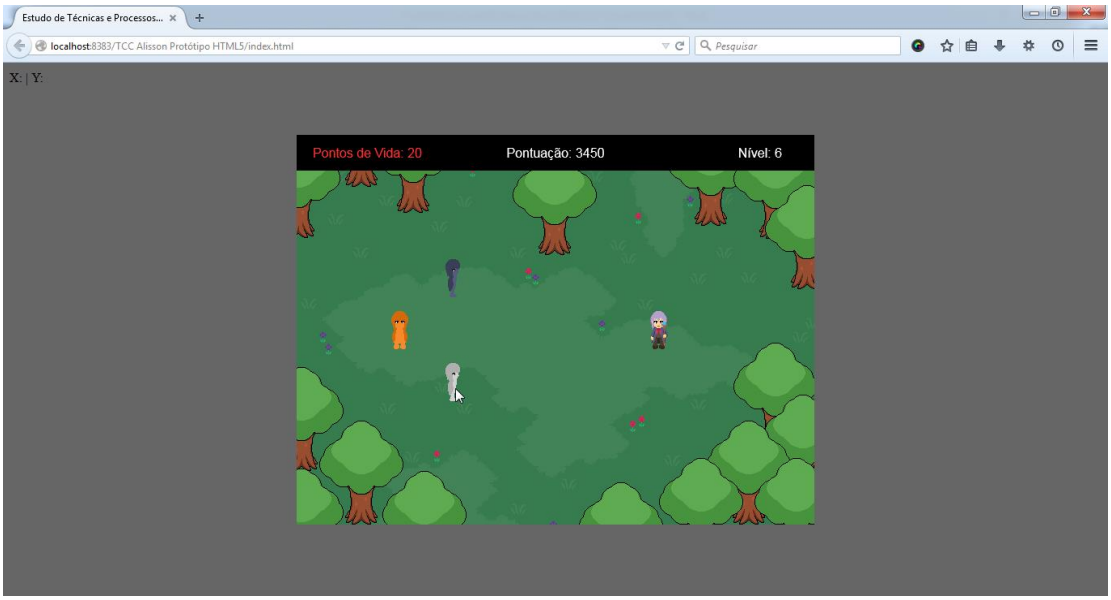
Nesse fluxo, deve haver uma cronologia para se renderizar os elementos gráficos na tela, pois os mais novos sempre irão sobrescrever em cima dos já existentes. Normalmente, o primeiro elemento a ser gerado será o cenário. Em seguida poderemos inserir o *sprite* do jogador. Os próximos elementos da lista seriam os inimigos, e neste momento seria necessário realizar algumas verificações. Quando o jogador clica ou toca em um inimigo, é preciso reduzir os pontos de vida dele. Se houver registro de um clique ou toque na *Canvas* dentro do intervalo que corresponde à posição deste inimigo na tela, realizaremos esta subtração. Em seguida, é preciso testar o estado dos inimigos, afinal, se neste momento seus pontos de vida chegarem a zero, este inimigo terá sido derrotado e não haverá mais

a necessidade de mostra-lo na tela. Por último, iremos exibir o texto que informa os pontos de vida, a pontuação e o nível do jogador. Se voltarmos ao capítulo 4.4, veremos que esta foi a descrição do algoritmo do fluxo do jogo que estabelecemos anteriormente, aplicado para trabalhar com a *Canvas* do HTML5.

Pode parecer muito, mas com a tecnologia que existe atualmente consegue-se tranquilamente executar essa rotina 30 ou 60 vezes em um único segundo, criando assim a tão desejada ilusão de movimento somada à interação com o usuário que caracteriza os jogos eletrônicos. Com um fluxo de jogo estabelecido e uma rotina para atualizar o posicionamento de cada elemento gráfico do jogo dentro de um intervalo de tempo, o programador poderá progredir construindo a lógica da jogabilidade, estabelecendo a interação do jogador com a aplicação e dos objetos presentes no game entre si.

O código-fonte do protótipo desenvolvido em HTML5, com comentários para explicar o fluxo da aplicação e as funções utilizadas no decorrer do código, está disponível no Apêndice 3. Através da Figura 14 está ilustrada a execução do protótipo em HTML5 rodando no navegador Firefox.

Figura 14: Protótipo em HTML5 rodando em um computador com Windows através do navegador Firefox



Fonte: Autor.

5 RESULTADOS

Após estudar e realizar o uso da engine *Game Maker: Studio* e das tecnologias do HTML5, a maior de todas as constatações a serem feitas é de que as duas ferramentas podem gerar bons resultados, embora os processos para chegar a estes sejam diferentes.

Ambas as tecnologias foram capazes de produzir os protótipos conforme foram especificados ao final da Seção 4.4. De fato, ao se colocar cada protótipo lado a lado e executá-los simultaneamente, pouco se notam diferenças no funcionamento de cada um. Mas por detrás da aparente execução similar, muito distintos são as funcionalidades que foram utilizados em cada ferramenta para se construir a lógica presentes nestes protótipos de games. O uso de cada tecnologia trouxe seus próprios benefícios e também os seus desafios na hora de realizar a construção destes protótipos.

Partindo dos pontos de análise definidos no Capítulo 4, aplicando-os sob os protótipos e sob o processo de desenvolvimento que os construíram, pretende-se nos próximos capítulos realizar uma avaliação das tecnologias escolhidas para praticar o desenvolvimento de jogos eletrônicos.

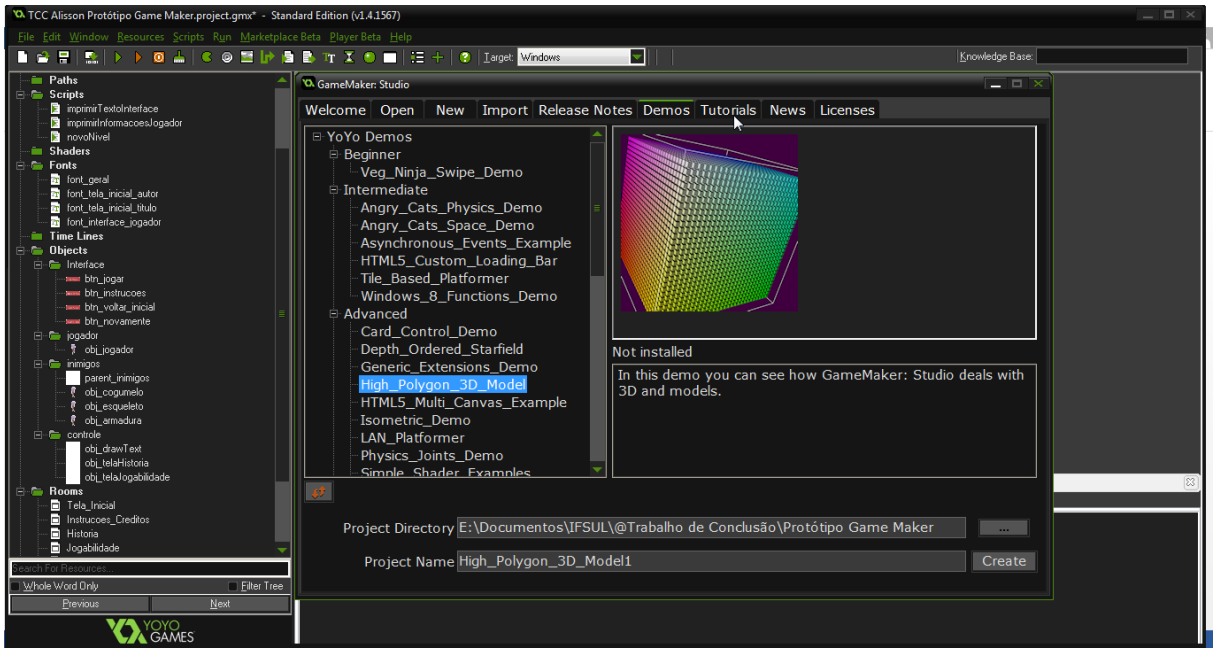
5.1 Visão geral das ferramentas utilizadas

Ao comentarem acerca do uso da ferramenta *Game Maker: Studio*, desenvolvedores comumente elogiam a facilidade e rapidez para se realizar o desenvolvimento através de seu uso, seja para realizar atividades de prototipagem ou para de fato criar um game completo. Isso é algo que pude testar durante o uso da ferramenta e trata-se de uma opinião da qual também passo a compartilhar.

O mais impressionante na ferramenta não são as suas capacidades técnicas ou amplo suporte multiplataforma, mas o próprio compromisso deste em ser a mais amigável o possível, principalmente para usuários inexperientes que recentemente começaram a se dedicar ao desenvolvimento de games. Isso é evidenciado não somente através de suas funcionalidades, mas também da ampla variedade de

manuais didáticos, demonstrações e tutoriais embutidos diretamente na ferramenta, vide a Figura 15.

Figura 15: Aba de demonstrações e tutoriais embutidos no Game Maker: Studio



Fonte: Autor.

Uma característica relativa ao uso de engines é que, invariavelmente, você terá que se submeter ao aprendizado do uso específico de seus recursos. Analisando toda a trajetória percorrida pela ferramenta, desde as primeiras versões criadas por Mark Overmars seguidas pelos constantes aprimoramentos por parte da YoYo Games, o *Game Maker: Studio* consegue inspirar confiança em seus usuários, justificando a escolha desta ferramenta para o trabalho de se criar jogos e validando o empenho para se adequar ao seu modelo de criação de jogos e investir em suas funcionalidades.

Por outro lado, olhar para os rumos que tem sido tomados no desenvolvimento do HTML5 e da linguagem Javascript é nada menos do que fascinante. Trata-se de um reflexo do quanto a tecnologia da Internet se desenvolveu nos últimos anos. Não apenas na perspectiva do puro avanço tecnológico, mas do impacto na cultura e na maneira como utilizamos a informática. Os Navegadores de Internet já serviram apenas para carregar conteúdos de texto por intermédio do HTML presente em um site. Hoje, eles podem ser o palco para a

execução de qualquer tipo de aplicação que se possa imaginar. A perspectiva é que cada vez mais aplicações venham a se convergir para utilizar as tecnologias do padrão web.

Harris (2013) destaca que as vantagens oferecidas pelo HTML5 para o desenvolvimento de jogos são: a ampla disponibilidade e gratuidade da ferramenta; sua maior acessibilidade para iniciantes em programação, se comparada com linguagens como o C++; o vasto público-alvo que um jogo desenvolvido em HTLM5 pode ser alcançado em virtude do fato deste não precisar de plug-ins adicionais, funcionando apenas com as capacidades já existentes nos navegadores de Internet modernos; e a facilidade de criação de jogos para dispositivos móveis. O HTML5 possui a capacidade de viabilizar o desenvolvimento de games para todos, permitindo que o estudo de game design e a criação de jogos eletrônicos possa ser praticado por uma maior parcela de pessoas.

Também é válido que a questão multiplataforma também seja levada em consideração dentro desta reflexão. Ao se dedicar ao engenhoso projeto que é o desenvolvimento de um jogo, não há sentimento mais natural do que desejar que seu game seja jogado pelo maior número possível de pessoas. Pensar em desenvolvimento multiplataforma é um dos caminhos que permite viabilizar esse desejo, e ambos o *Game Maker: Studio* e o HTML5 demonstram que também seguem tal perspectiva ao fornecerem recursos para minimizar a complexidade de se executar um aplicação em diferentes tipos de hardware.

5.2 Análise das ferramentas com base nos protótipos

Conforme fora discutido ao se explicar as etapas que envolvem o desenvolvimento de games, a criação de protótipos é uma atividade importante durante as etapas iniciais de desenvolvimento, por permitir testar a aplicabilidade e efetividade daquilo que fora estabelecido no Documento de Design do Game.

Através do uso de alguns dos conceitos propostos no DDG do jogo “Desafio do Lorde das Trevas” como ponto de referência, os protótipos de um jogo eletrônico foram efetivamente criados utilizando o *Game Maker: Studio* e o HTML5. A partir desta experiência e daquilo que fora concretizado nestes protótipos, as duas ferramentas terão uma avaliação de suas capacidades para desenvolver jogos descritas nos próximos capítulos, remetendo aos critérios de análise definidos durante a especificação da metodologia.

5.2.1 Recursos disponíveis nas ferramentas em prol do desenvolvimento de jogos eletrônicos

Considerando o tipo de software que é, o *Game Maker: Studio* como um todo é um grande conjunto de recursos para serem utilizados em prol do desenvolvimento de jogos. Seja através das opções oferecidas para manipular cada tipo de elemento presente no projeto, da maneira como ele faz o gerenciamento dos arquivos e da lógica de execução do jogo ou até da própria apresentação de sua IDE, cada detalhe da ferramenta foi pensado para que desenvolvedores pudessem maximizar a sua produtividade.

Para cada tipo de recurso existente no projeto é oferecido pela ferramenta editores com várias opções de configuração. Através do editor de *Sprites*, foi possível desenhar Gráficos 2D dentro da própria ferramenta, além de haver a possibilidade de exportar ou importar elementos gráficos e definir uma sequência destas imagens para montar uma animação. Já no editor de Objetos, através de sua funcionalidade de clicar e arrastar ações em uma ordem específica, montar algoritmos para realizar determinadas ações na ocorrência de um evento em específico é algo muito prático de ser feito.

A variedade de eventos com as quais pode-se trabalhar é extensa, o que possibilitou: iniciar rotinas específicas quando uma *Room* é iniciada ou quando um objeto é criado; reconhecer os cliques do mouse para transitar entre os menus da aplicação e também estabelecer a jogabilidade para causar danos aos inimigos quando estes são clicados; e somar pontos à pontuação total do jogador ao ocorrer a eliminação de um inimigo. Outro tipo importante de evento utilizado foram os *Alarms* (alarmes em inglês), uma funcionalidade presente no *Game Maker: Studio* voltada para a execução de uma determinada rotina após um intervalo de tempo, o qual foi utilizado para estabelecer o comportamento dos inimigos reduzirem os pontos de vida do jogador de tempos em tempos.

Mesmo com tais opções, aprender e fazer uso da programação utilizando GML é algo recomendável. A linguagem GML é capaz de realizar todas as operações presentes na interface do editor de objetos e ainda possibilita fazer uso de uma quantidade maior de funcionalidades através de sua API. Na construção do protótipo, scripts foram utilizados para configurar estilos de fonte e gerar de forma

dinâmica e em tempo real textos para títulos, instruções e dados do jogador. Um outro script foi escrito para realizar a rotina de configurar um novo nível durante o jogo, utilizando funções matemáticas para escolher aleatoriamente os tipos de inimigos que em seguida terão os valores das variáveis de seus atributos configurados para serem instanciados na tela.

Noções relacionadas à Orientação à Objetos também podem ser aplicadas aos Objetos do *Game Maker: Studio*, em particular a relação de herança entre objetos. Tal artifício foi usado para configurar os inimigos do jogo. Há três tipos de inimigos com comportamentos similares mas também com características únicas. Um objeto genérico com os comportamentos básicos de um inimigo foi criado para ser utilizado como referência por todos os objetos de inimigos. Isso permite a todos possuírem as mesmas características básicas, enquanto permite a possibilidade de cada um ter características particulares.

As funcionalidades existentes dentro da ferramenta ainda conseguem ir além daquilo que acabara de ser citado e que fora aplicado para criar o protótipo. Sistemas de detecção de colisões mais aperfeiçoados; manipulação gráfica através de *Shaders*; requisições para servidores externos; funções da linguagem GML para realizar cálculos matemáticos, manipular vetores, trabalhar com estruturas de dados, ou realizar testes no funcionamento da aplicação; configuração de servidores para controle de versionamento do projeto; integração com o Facebook ou o Steam são apenas alguns exemplos interessantes da extensão das capacidades desta *engine*.

O potencial do uso das tecnologias HTML5 e Javascript para o desenvolvimento de jogos, por sua vez, também é genuíno. O Javascript que é executado através dos navegadores em nada fica para trás se comparado com outras das linguagens de programação mais fortes do mercado.

A essência dos jogos em HTML5 está nos vastos recursos presentes na API do *Canvas*, os quais podem ser utilizados através da chamada do método *getContext()* ao se estabelecer se estamos trabalhando com gráficos 2D ou 3D. Sendo o foco o projeto um jogo predominantemente 2D, a API nos oferece uma série de métodos para gerar gráficos dinamicamente dentro do *canvas*. Há funções para renderizar formas geométricas específicas como retângulos sem preenchimento com a função *strokeRect()* ou com preenchimento utilizando *fillRect()*. Combinando o *fillRect()* com o atributo do contexto 2D *fillStyle* foi possível estampar por toda a *canvas* uma imagem de fundo de 64 pixels de largura por 64 pixels de

altura. Outro grupo de funções muito utilizado no decorrer do protótipo foram os métodos para configurar e renderizar texto na tela do *canvas*, utilizando os atributos *fillStyle, font, textAlign e textBaseline* para, respectivamente, configurar cor, tipografia, posicionamento horizontal e posicionamento vertical de um conteúdo de texto a ser desenhado na tela com o método *fillText()*.

A construção do restante da lógica envolveu combinar a API do *canvas* com os demais artifícios da linguagem Javascript. Através dos objeto padrão *Image*, presente na própria API do Javascript, é possível carregar os elementos gráficos do jogo para serem renderizados na *canvas* durante o seu fluxo de execução utilizando a função *drawImage()*. O objeto *Image* também conta com diversos métodos para manipular as imagens carregadas, como a capacidade do próprio *drawImage()* de determinar um tamanho e uma área específicos de uma imagem para exibir na *canvas*. Através do marcador *Audio*, também introduzido no HTML5, o Javascript é capaz de tratar a sonoplastia do game, carregando uma música de fundo em paralelo e executando efeitos sonoros em determinados momentos da lógica do jogo. Outro recurso presente no Javascript foram os métodos *addEventListener(), removeEventListener(), setInterval()* e *clearInterval()*. O *addEventListener()* permite atribuir a algum elemento presente em uma página da Internet uma rotina que deverá ser executada continuamente dentro de um determinado intervalo de tempo, até que o método *removeEventListener()* seja invocado para interromper este ciclo. *setInterval()* funciona de maneira similar, mas executando uma determinada função continuamente num intervalo de tempo, terminando apenas quando *clearInterval()* é chamado. Foi graças a estes recursos que foi possível implementar alguns dos comportamentos fundamentais para que pudéssemos ter um jogo em HTML5: a rotina para atualizar os gráficos presentes na *canvas* constantemente; a rotina para monitorar se ocorreram cliques ou toques na área da *canvas* e recuperar os pontos X e Y onde essa interação ocorreu; e a rotina para que os inimigos realizem ataques ao jogador, subtraindo seus pontos de vida.

Todos os elementos que compõem o jogo foram determinados no código-fonte através de variáveis globais, vetores e Objetos do Javascript. A lógica do jogo pôde ser fragmentada em diversas funções, o que não somente auxilia na organização do código como também permite a reutilização de rotinas que são executadas com frequência, atribuindo mais dinamismo à execução do jogo. Assim,

novamente remetemos à aplicação dos conceitos da Orientação à Objeto, que auxiliam na construção de códigos-fontes mais organizados e eficientes.

O processo de desenvolvimento de aplicações utilizando *Game Maker: Studio* e HTML5 se distinguem muito um do outro, com ambos apresentando suas características únicas e maneiras diferentes de implementar certas funcionalidades em comum. Mas a questão principal é que ambos possuem os artifícios necessários para que jogos eletrônicos possam ser desenvolvidos, evidenciado no fato destas tecnologias distintas terem conseguido implementar um mesmo tipo de lógica e terem gerado protótipos que são praticamente idênticos em execução.

5.2.2 Facilidade de uso e programação

É de se esperar que uma ferramenta chamada *Game Maker: Studio* seja uma suíte que proporcione aquilo que é necessário para realizar a criação de jogos eletrônicos com praticidade. E de fato, a recíproca é verdadeira: a IDE e os recursos presentes no *Game Maker: Studio* genuinamente oferecem uma experiência soberba para realizar o que é essencialmente uma tarefa bastante complexa.

Ao divulgar o *Game Maker: Studio*, a YoYo Games sempre fez questão de exaltar a sua rica interface gráfica e suas opções para a criação de jogos sem a necessidade de se ter conhecimentos de programação. A interface de criação de Objetos é um recurso da ferramenta que abstrai a necessidade de escrever código-fonte, possibilitando a programação do comportamento dos objetos através do manuseio de botões que representam características específicas. Estas poderão ser configuradas e posicionadas em uma sequência lógica de funcionamento. Nota-se aqui a presença de vários artifícios essenciais da programação (testes lógicos, loops, manipulação de valores em variáveis). Apenas com o clicar e arrastar destes botões é possível determinar todo o funcionamento que um dado objeto irá ter dentro do game.

Mesmo com os recursos do editor de objetos, a ferramenta obviamente ainda permite utilizar-se da programação para criar funcionalidades mais complexas. Através da linguagem de script *Game Maker Language* (GML) o usuário da ferramenta dispõe de todos os artifícios essenciais para se realizar o trabalho de programação, permitindo o encapsulamento através do uso de classes e funções. Sua sintaxe é bastante próxima de outras linguagens de script como o Lua ou

mesmo o próprio Javascript, fazendo dela uma linguagem de programação poderosa, mas de fácil compreensão e praticidade de uso.

Como já mencionado no item anterior, a disposição organizacional de um projeto no *Game Maker: Studio* é bem agradável de se trabalhar com. Sua interface gráfica é intuitiva e suas ferramentas proporcionam boa eficiência ao se trabalhar com os vários tipos de elementos que compõem um jogo eletrônico. A ferramenta ainda conta internamente com uma extensa documentação, detalhando seus recursos e suas funcionalidades, e ainda conta com uma integração à sua base de dados online para consulta de informações de uso. O único porém existente seria o software e a sua documentação se encontrarem até então disponíveis apenas em inglês. Possuir certa familiaridade com o idioma é recomendável para que melhor se possa compreender e tirar proveito dos recursos mais avançados da ferramenta. Ainda assim, crê-se que a barreira do idioma não seja forte o bastante para comprometer a usabilidade da ferramenta.

Evidentemente, sem que se tenha uma *engine* pronta para cuidar dos aspectos essenciais da execução de um game, o desenvolvedor terá de se encarregar de criar por conta própria os métodos para estabelecer um fluxo de jogo funcional. Somente então haveria cabimento para poder começar a elaborar a lógica do game em si. Este foi o cenário enfrentado ao se realizar o desenvolvimento do protótipo em HTML5, usando apenas os artifícios básicos do Javascript sem nenhum framework adicional. A principal consequência desta escolha foi a maior demora para se realizar a produção do protótipo. O tempo gasto para desenvolver o protótipo em HTML5 foi notavelmente maior, sendo que a maior parte deste tempo foi voltada a criar o método que atualiza a *Canvas* continuamente. Ao começar o desenvolvimento do protótipo no *Game Maker: Studio*, o processo iniciou diretamente focado na lógica do jogo em si, beneficiando-se ainda dos recursos de sua interface gráfica.

Sem o uso de uma engine ou mesmo um framework para o desenvolvimento de games, indubitavelmente se estará aumentando a carga de trabalho e a complexidade do projeto. É válido que, antes de se começar o desenvolvimento de um jogo, seja feita uma reflexão quanto ao seus requisitos e se o uso de alguma engine ou framework em particular não seria benéfica para o processo.

Mesmo não dispondo de facilidades na mesma intensidade que as presentes no *Game Maker: Studio*, realizar o desenvolvimento de um jogo em HTML5 usando

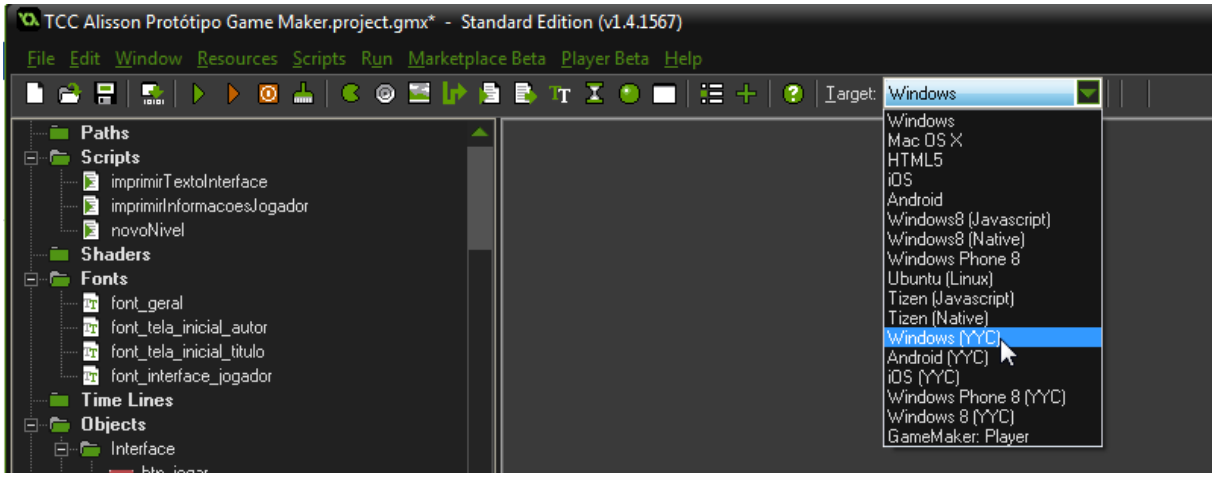
uma IDE já consegue ser suficientemente para auxiliar a gerenciar os arquivos do projeto e a escrever o código-fonte, contribuindo consideravelmente a produtividade. E com a relevância do HTML5 no cenário atual da tecnologia e com a proeminência do *Game Maker: Studio* entre as *engines* de desenvolvimento de jogos, ambos dispõem de uma quantidade vasta de artigos e materiais didáticos para que se possa aprender sobre seu uso na criação de jogos. A quantidade de tais materiais na língua portuguesa, inclusive, é bem razoável. Claro que a maior parte da documentação de uso e alguns dos mais completos tutoriais estão disponíveis apenas em línguas estrangeiras, uma realidade que é característica da área da informática como um todo.

5.2.3 Capacidade e eficácia multiplataforma

O principal fator que fez ambos *Game Maker: Studio* e HTML5 serem escolhidos dentre tantas tecnologias para o desenvolvimento de jogos foram a variedade de plataformas que estes conseguem suportar e a maneira como tal feito é alcançado.

Ao contextualizar o *Game Maker: Studio*, muito foram exaltadas suas capacidades multiplataforma. Sua proposta é a de entregar um executável dentro de uma determinada plataforma através da simplória ação de escolher a plataforma desejada dentro de uma lista e clicando em um botão, o que podemos ver através da Figura 16.

Figura 16: Escolha da plataforma para qual deseja-se compilar um projeto do Game Maker: Studio



Fonte: Autor.

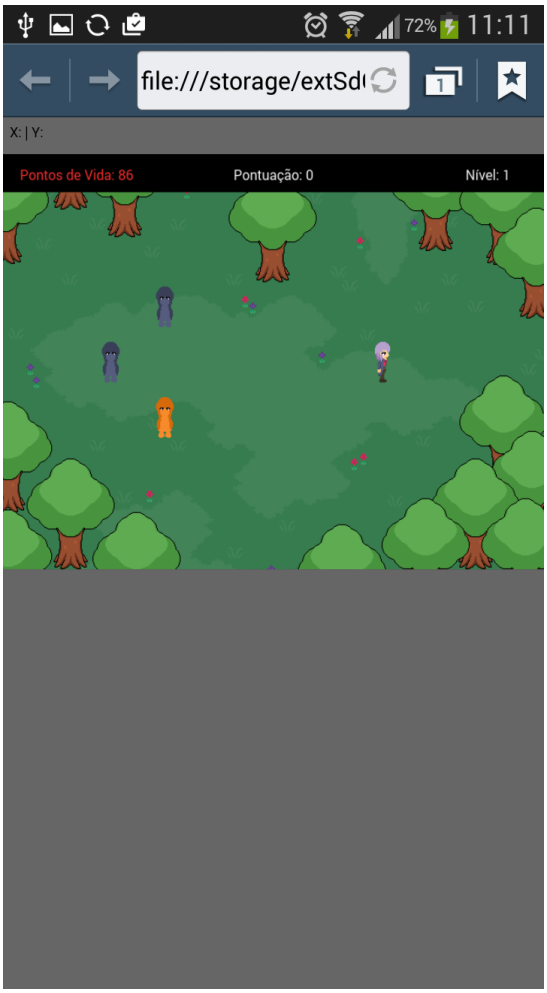
Infelizmente, há um contraponto no uso do *Game Maker: Studio* quando o desenvolvimento é pensado para múltiplas plataformas. Ainda que o *Game Maker: Studio* possua uma atraente variedade de plataformas para as quais ele é capaz de rodar seus jogos, há uma forte barreira monetária impedindo o acesso do desenvolvedor a essas plataformas.

Por mais que se desejasse poder ver o protótipo criado com *Game Maker: Studio* sendo executado em vários tipos de plataformas, a criação de protótipos multiplataforma nesta ferramenta acabou por ficar apenas na teoria, pois não foi possível executar o protótipo em outro sistema além de um computador com Windows. O suporte multiplataforma da ferramenta é evidente, mas para poder explorar este potencial, o desenvolvedor precisará pegar sua carteira e investir na aquisição de licenças para poder utilizar o compilador de um determinado tipo de plataforma. Considerando a facilidade e a eficácia do processo para gerar o executável para Windows, podemos apenas supor que a compilação para as demais ferramentas também ocorra de maneira efetiva.

Este foi um aspecto que acabou por exaltar a escolha do HTML5 como objeto de estudo paralelo. Beneficiando-se do aspecto de linguagem interpretada do HTML5, o segundo protótipo estava prontamente apto para rodar nos navegadores de Internet de diversas plataformas (na Figura 17 podemos observar o protótipo sendo executado em um dispositivo Android). Mas é preciso destacar que jogos

desenvolvidos em HTML5 precisam de um cuidado especial na parte de testes, por questões de incompatibilidades entre os diferentes navegadores.

Figura 17: Protótipo em HTML5 rodando no navegador de Internet de um dispositivo Android



Fonte: Autor.

A W3C estabelece quais são as capacidades do HMTL5. Mas como estas serão executadas na prática fica a cargo dos desenvolvedores de cada navegador de Internet. Como consequência, é comum que determinadas funções ou características do HTML5 ou Javascript funcionem de maneiras diferentes (ou até mesmo sequer existam) em cada navegador. Assim, para que uma aplicação em HTML5 consiga ser efetivamente multiplataforma, o desenvolvedor deverá estar atento a tais particularidades e aplicar soluções para garantir a execução da lógica em diversos navegadores de Internet.

5.2.4 Performance

Para o jogador comum, sem muitos conhecimentos da parte técnica do funcionamento de aplicações computacionais, um jogo eletrônico com boa performance é um jogo eletrônico que: consegue manter uma taxa de quadros por segundo (FPS[63]) estável, sem oscilações ou quedas frequentes; que é competente em executar sua lógica e a renderização gráfica dentro de um escopo de requisitos de hardware sem apresentar travamentos ou lentidão; e que é capaz de executar o fluxo de jogo sem a ocorrência de bugs, falhas na lógica do jogo ou problemas na hora de exibir os gráficos.

Com a variedade de tipos de hardware no mercado, quando um jogo eletrônico é elaborado, é necessário estabelecer uma plataforma alvo e trabalhar com os requisitos e limitações técnicas que esta pode apresentar. Não será um problema se o seu jogo não conseguir rodar em um hardware de 10 anos atrás. Mas será um problema se o seu jogo tiver sido especificado para rodar em um determinado console de vídeo game ou um hardware específico e apresentar lentidão ou mesmo quando alguma rotina que pudesse ser considerada como simples acaba necessitando uma quantidade absurda de poder computacional para ser executada.

Se para rodar jogos complexos com gráficos detalhados é necessário possuir um hardware robusto, por dedução podemos imaginar que para desenvolver tais jogos é preciso utilizar hardware tão ou mais poderoso quanto. Neste sentido, ambos *Game Maker: Studio* e HTML5 são destaques por se tratarem de tecnologias bem modestas no que se refere a requisitos de hardware. A YoYo Games aponta que para rodar o *Game Maker: Studio* os requisitos mínimos seriam de 512 MB de memória RAM, uma placa de vídeo com 128MB e no mínimo possuir suporte ao DirectX9, mas também estabelece que o ideal para se trabalhar com a ferramenta seria em um computador com pelo menos 4GB de memória RAM. Já para o HTML5, se o seu computador consegue rodar a versão mais recente de qualquer um dos

[63] A taxa de quadros por segundo (em inglês, Frames per Second ou FPS) refere-se à quantidade de imagens em sequência presentes em 1 segundo. Essa quantidade impacta na fluídez presente na ilusão de movimento que presenciamos ao ver vídeos. Enquanto na indústria cinematográfica o padrão estipulado é de 24 FPS (ou seja, em um segundo de vídeo há a transição entre 24 imagens estáticas), jogos eletrônicos costumam operar entre 30 FPS ou 60 FPS.

navegadores de Internet atuais e consegue abrir um editor de texto então você já atende aos requisitos para começar a desenvolver aplicações para a web.

O parâmetro de performance mais evidente em um jogo eletrônico é a medida da taxa de quadros por segundo. Se ao executar um jogo em um determinado hardware e este apresentar uma taxa de quadros baixa ou até mesmo instável, isso significa que ou o hardware em questão não tem capacidade suficiente para processar todas as rotinas do jogo e atualizar a tela com frequência ou que este jogo não foi otimizado apropriadamente. Um aliado dos jogadores mais detalhistas é o software Fraps[64], um software para realizar gravações da tela do computador mas que também é amplamente com conhecido pelo seu medidor e analisador de FPS. Os protótipos foram desenvolvidos e testados em um computador com 4GB de memória RAM, com um processador da 1ª Geração do Intel Core i5, placa de vídeo Nvidia GeForce 640m, rodando no sistema operacional Windows 7 64-bit. Trata-se de um hardware que já conta com 5 anos de uso, mas que ainda consegue rodar a maioria das aplicações com competência e não se distancia muito dos hardwares amplamente disponíveis no mercado atualmente.

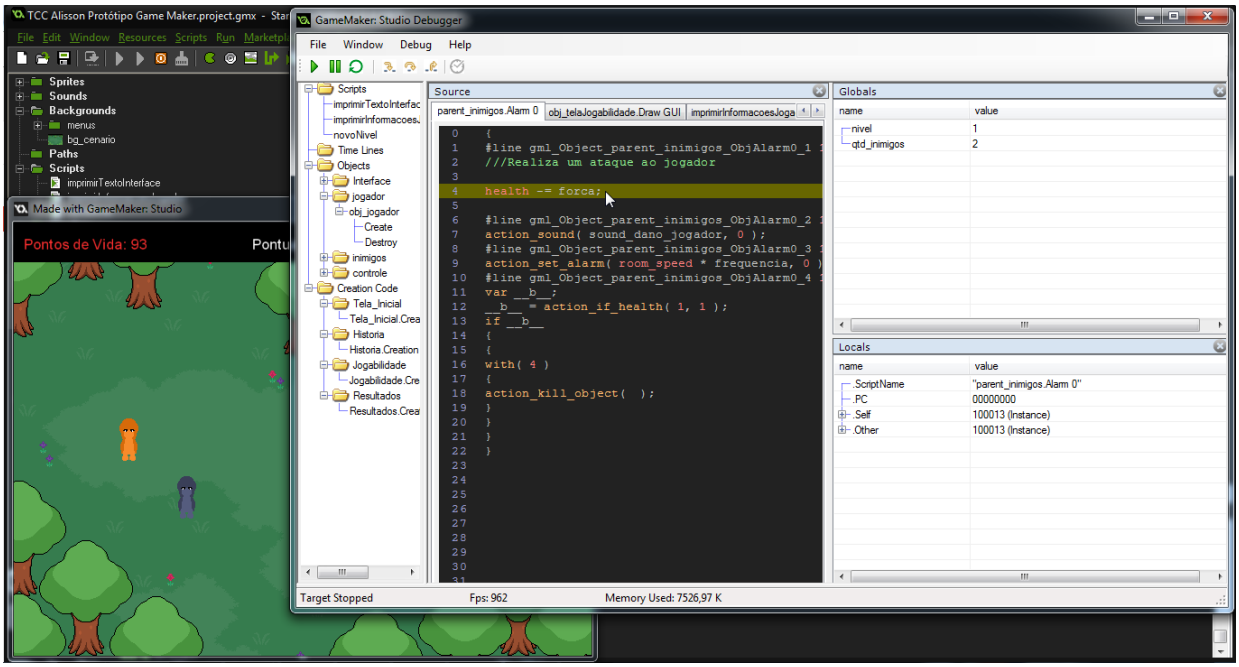
Ambos os protótipos mostraram-se eficientes em suas execuções, conseguindo executar o jogo com taxas de frequência estáveis de tanto 30 FPS quanto 60 FPS. Levando em consideração o quão relativamente simples estes protótipos são isso talvez não seja nada surpreendente, mas é confortante observar que ambas as tecnologias apresentam um mínimo de competência para produzir e executar simples aplicações.

É perceptível que a Yoyo Games teve um cuidado especial no que se refere ao tratamento da performance dos games criados com seu software. Um jogo desenvolvido na ferramenta poderá ser executado em um modo de testes no qual o desenvolvedor consegue controlar a taxa de FPS, o consumo de memória da aplicação e quais são as rotinas que estão em execução no momento. Na Figura 18 podemos ver o protótipo construído na ferramenta sendo executado neste modo.

---

[64] http://www.fraps.com/

Figura 18: Protótipo construído no Game Maker: Studio rodando em modo de debug



Fonte: Autor.

No que se refere à performance em jogos produzidos em HTML5, há alguns pontos a serem levados em consideração. Garantir uma boa performance em game construído com esta tecnologia irá exigir mais atenção da parte do desenvolvedor. Cada pequena instrução digitada pelo programador e cada arquivo que precisa ser solicitado no servidor irá ter o seu impacto na aplicação final.

A performance de um jogo em HTML5 é muito dependente da qualidade da Internet e da carga de processamento do Navegador para carregar os elementos do jogo como as imagens e as músicas. Ainda que os hardwares e navegadores de Internet atuais consigam rodar jogos complexos e seja possível conseguir acesso a conexões de rede de alta velocidade, é recomendável que jogos em HTML5 sejam projetados para serem curtos e para não necessitarem carregar arquivos pesados. A web funciona através de um modelo cliente/servidor, no qual o seu computador se comunica com uma outra máquina solicitando algum conteúdo que ela contém. Esta comunicação é influenciada pela capacidade desta máquina servidora em suprir todas as solicitações que ela recebe e também pela qualidade da conexão que você está estabelecendo com o servidor. Cada recurso a ser utilizado em seu jogo em

HTML5 precisará ser solicitado ao servidor através deste protocolo de comunicação que de muitas maneiras podem ocorrer falhas.

Sem um controle adequado, pode-se ter problemas de sincronia na execução dos métodos do contexto da Canvas. Este comportamento não é favorável para a execução de jogos, já que a lógica de um game não deveria estar em execução sem que os objetos que compõem o jogo ou uma fase tenham sido carregados e estejam prontos para uso. Durante o desenvolvimento do protótipo em HTML5, ocorreram situações em que a lógica do jogo já estava sendo executada mesmo sem todos os elementos gráficos estarem carregados, gerando erros na execução do Javascript e ocasionando bugs durante a jogabilidade. Em uma outra situação, essa dessincronia fez com que as imagens de fundo, o primeiro elemento na ordem cronológica de renderização do *Canvas*, sofresse um atraso que faria o fundo sobrescrever os demais conteúdos gráficos da tela.

Para poder contornar o problema, foi criada uma rotina para carregar cada um dos elementos gráficos e sonoros do game, estabelecendo o início do fluxo do jogo apenas quando todos estes elementos estivessem presentes na memória. Observando os benefícios desta abordagem, pode-se afirmar que esta é uma prática totalmente recomendável ao se realizar o desenvolvimento de jogos em HTML5 e Javascript.

5.3 Resumo e análise dos resultados

Nas seções anteriores houve o foco em analisar o desenvolvimento dos protótipos utilizando *Game Maker: Studio* e HTML5, dentro das perspectivas de: o que cada uma oferece para desenvolvermos jogos eletrônicos; o quão prático é esse processo utilizando tais tecnologias; a disposição de plataformas nas quais elas possibilitam a execução do jogo; e como é o tratamento da performance de execução dos jogos produzidos nelas.

Ao se colocar estas duas tecnologias em avaliação o objetivo não foi determinar objetivamente qual é superior, afinal, é inviável se comparar dessa forma uma linguagem de programação com um software completo. Esta avaliação, e consequente comparação feita entre as duas, é focada na experiência de desenvolvimento de ambas. Daquilo que cada tecnologia é capaz de fazer e como podemos desenvolver games com elas. Um reflexo da variedade de tecnologias diferentes para se desenvolver jogos eletrônicos que foram mencionadas no

decorrer do trabalho. Uma síntese de todos os apontamentos da avalição destas duas ferramentas dentro dos critérios de análise pode ser vista na Tabela 3.

Tabela 3: Resumo comparativo da análise do Game Maker: Studio e do HTML5

Game Maker: Studio	HTML5
Recursos disponíveis nas ferramentas para criar games	
• Suíte completa para desenvolvimento de jogos • Facilidade de gerenciamento e manipulação de arquivos • Interface gráfica produtiva • Construção da lógica do jogo através da interface gráfica ou dos métodos da linguagem GML	• Visão do jogo através do recurso Canvas • Recursos presentes no HTML5 contemplam os requisitos para o funcionamento do jogo. Exemplos: ➢ drawImage() do objeto Canvas ➢ objeto Image do Javascript ➢ <audio> do HTML5 ➢ addEventListener() do Javascript ➢ setInterval() do Javascript
Facilidade de uso e programação	
• Extensa documentação (em inglês) inclusa na ferramenta • Interface gráfica permite gerenciar todos os componentes utilizados no jogo • Abstrai a necessidade de se conhecer programação para produzir o jogo	• Tecnologia amplamente utilizada e difundida • Requer mais trabalho para contemplar os requisitos básicos da execução do jogo
Capacidade e eficácia multiplataforma	
• Amplo suporte a plataformas diferentes, mas exige a aquisição individual do compilador para cada plataforma	• Nativamente e instantaneamente multiplaforma • Requer atenção para incompatibilidades entre navegadores de Internet

Performance	
• Protótipo roda à 30 e à 60 FPS • Não requer hardware robusto para desenvolver e jogar • *Engine* se encarrega de gerenciar carregamento dos recursos utilizados durante o jogo • Oferece opções para otimização e debug	• Protótipo roda à 30 e à 60 FPS • Não requer hardware robusto para desenvolver e jogar • Requer controle manual para gerenciamento dos recursos carregados e utilizados pelo jogo • Performance relativa à qualidade da conexão de Internet e capacidade do Servidor

Fonte: Autor.

Os recursos presentes no *Game Maker: Studio* são funcionais e a sua metodologia para gerenciar e construir o projeto contribuem significativamente para o processo de desenvolvimento. Em sua versão gratuita a ferramenta consegue abranger todos os requisitos necessários para se criar jogos robustos, com qualidade e facilidade. Mesmo assim, ainda é necessário pagar para utilizar todo o potencial da ferramenta, havendo limitações na disposição de plataformas nas quais se pode executar o jogo e inclusive no escopo que pode ser alcançado por um game desenvolvido na versão gratuita[65].

Sem o uso de ferramentas já existentes, o trabalho para se desenvolver jogos se torna muito mais complexo. Isso pôde ser constatado durante o desenvolvimento utilizando *Canvas* do HTML5 e Javascript. Além da lógica do game, foi preciso elaborar uma estrutura de execução contínua para reproduzir na tela o estado atual da aplicação, controlar manualmente todos os recursos do jogo e garantir boa performance ao tentar executar tudo isso simultaneamente. De certa forma pode-se afirmar que uma pequena *engine* de jogos em HTML5 foi desenvolvida durante esse processo. Nada disso serviu de empecilho para construir o protótipo em questão com sucesso. Mas devemos levar em conta que esse protótipo representa um jogo relativamente simples e que tentar desenvolver um game mais ambicioso nestas circunstâncias certamente acabaria sendo problemático.

[65] O vídeo de apresentação do YoYo Compiler demonstra o limite que um jogo produzido na versão gratuita do *Game Maker: Studio* consegue alcançar, em comparação com as capacidades do YYC presente apenas na versão paga da ferramenta: https://youtu.be/BPJ4hCfTZEM

A experiência de se desenvolver jogos em HTML5 reflete a complexidade deste tipo de software e evidenciou o trabalho que deve ser feito para se estabelecer a estrutura básica de um game. Isso inclusive faz com que apreciemos o trabalho feito pela YoYo Games no *Game Maker: Studio*. Afinal, inicialmente eles também precisaram passar pelo mesmo tipo de experiência demonstrado durante o desenvolvimento em HTML5 para tornar o *Game Maker* compatível com novas plataformas.

6 CONSIDERAÇÕES FINAIS

Em meio a todos os levantamentos feitos na trajetória deste trabalho, creio que uma das maiores afirmações que possam ser feitas é que “vivemos em uma época estimulante para ser um designer de games, porque existem muitos caminhos técnicos e criativos para o progresso, a inovação e a originalidade criativa” (SWAIN apud NOVAK, 2011, p.138). A tecnologia chegou a um ponto em que a capacidade dos hardwares amplamente disponíveis já não mais restringem o que é possível se fazer em jogo eletrônico, tanto quanto costumava ocorrer nos videogames das décadas passadas.

Os games são um fenômeno cultural. Já há muito tempo os jogos eletrônicos são influentes na vida das pessoas que os jogam. Muitos personagens originados em games como Super Mario, Megaman, Pac-Man e Sonic the Hedgehog (vistos da esquerda para a direita na Figura 19) transcenderam os seus meios e deixaram suas marcas na cultura do ser humano. Jogos eletrônicos fazem parte da vida cotidiana das pessoas e a variedade de estilos de games e plataformas para se jogar games apenas aumenta o número de adeptos deste meio e a influência geral que estes ocasionam. E todos estes fatores estabelecem uma perspectiva de que talvez cada vez mais pessoas também desejem ser não apenas jogadores, mas também criadores de jogos eletrônicos.

Figura 19: Tela do jogo Super Smash Bros. for Wii U com um quarteto de icônicos personagens dos games

Fonte: http://i.imgur.com/fn59s2L.jpg

As tecnologias utilizadas na construção de protótipos no decorrer deste trabalho são bastante distintas entre si, o que ocasionou trajetórias de desenvolvimento muito diferentes. Mas ainda assim, o produto final obtido através de o uso de cada uma delas são essencialmente idênticos. Com ambas as tecnologias foi possível criar protótipos baseados na proposta do DDG e ambos os protótipos conseguem executar todas as suas funções. Cada tecnologia traz consigo vantagens e empecilhos e mesmo que os processos sejam diferentes, ambas as tecnologias apresentam os recursos necessários e foram capazes de realizar a mesma tarefa.

A facilidade de criação proporcionada por ferramentas como o *Game Maker: Studio* ou a ubiquidade presente no HTML5 possibilitam que a criação de jogos eletrônicos se torne cada vez mais acessível e convidativa. A existência de um tipo de ferramenta não desmerece a existência de outros tipos. Há várias ferramentas que realizam uma mesma tarefa, mas as particularidades de cada uma possibilitam atender as necessidades específicas de cada usuário. Vamos nos lembrar do exemplo do estúdio *Vd-Dev* citado no início do trabalho que, mesmo dentre a variedade de tecnologias à disposição deles, resolveram escolher trabalhar com

Assembly. Mesmo que seja mais prático trabalhar com uma linguagem mais alto-nível para auxiliar no desenvolvimento do jogo deles, foi justamente o Assembly que veio a atender às suas necessidades e produzir os resultados que eles desejavam.

O que se pode constatar comparando a experiência de desenvolvimento com *Game Maker: Studio* e a experiência de desenvolvimento em HTML5 é que embora trabalhar apenas utilizando Javascript não chegue a ser considerado algo inviável, os benefícios de se trabalhar com a estrutura e funcionalidades de uma engine de desenvolvimento são muito marcantes. Colocando o *Game Maker: Studio* e o desenvolvimento em HTML5 usando Javascript em confronto um com o outro, o *Game Maker: Studio* acaba prevalecendo por conta de seu conjunto da obra. Ainda que o Javascript seja livre, amplamente utilizado e instantaneamente multiplaforma, essas qualidades por si só não conseguem confrontar a robustez dos recursos presentes em uma ferramenta criada especificamente para o desenvolvimento de games.

A experiência da construção do protótipo em HTML5 utilizando apenas Javascript foi válida como uma prova de conceito das capacidades desta tecnologia. Mas uma vez que o objetivo final de um projeto venha a ser criar um jogo completo, é contraindicado se limitar apenas ao básico do Javascript. É recomendável fazer uso de alguma tecnologia que consiga encobrir os requisitos técnicos para se rodar um jogo, de modo que o desenvolvimento possa ser focado inteiramente na lógica que irá compor o game e no funcionamento de sua jogabilidade. Enquanto o próprio *Game Maker: Studio* apresenta um módulo pago para compilar seus projetos para HTML5, existe uma variedade cada vez mais crescente de frameworks gratuitos e livres para auxiliar na criação de jogos em HTML5, destacando os frameworks *Quintus Engine*, *Phaser* e Panda.js como recomendações interessantes para serem estudadas.

Mesmo com tais constatações, gostaria de exaltar que desenvolver o protótipo em HTML5 utilizando apenas Javascript foi uma experiência fascinante por ter proporcionado um melhor entendimento de como funciona a rotina de execução de um jogo eletrônico, das nuances do processamento de dados e do processo de renderização gráfica que precisam ocorrer a todo momento durante a sua execução. O conhecimento agregado através deste experimento evidencia como o desenvolvimento desse tipo de software pode ser enriquecedor o aprendizado da programação e do funcionamento de aplicativos computacionais.

No geral, a relevância acadêmica de jogos e da prática de se jogar já é observada há bastante tempo, com diversas pesquisas e publicações que buscam analisar o que é um jogo e quais poderiam ser as suas aplicações. Em sua dissertação, Paul G. Brewster (1956 apud GEORGES, 1975, tradução nossa) atestou que mesmo com a ampla diversidade de opiniões referentes ao que se constitui a ideia de jogos há um consenso geral entre todos que estudaram o assunto de que esta certamente é uma área de grande importância e potencial. Ele justificou isso enumerando benefícios como a possibilidade de aprender mais sobre a história e os relacionamentos dos seres humanos; a perspectiva de se observar como crianças se comportam em ambientes com regras rígidas estabelecidas pelos mais velhos ou formuladas por elas mesmas; a oportunidade de se aprender sobre outras formas de expressão, citando como exemplo a música e a dança; e entender como o homem dispõe de seu tempo.

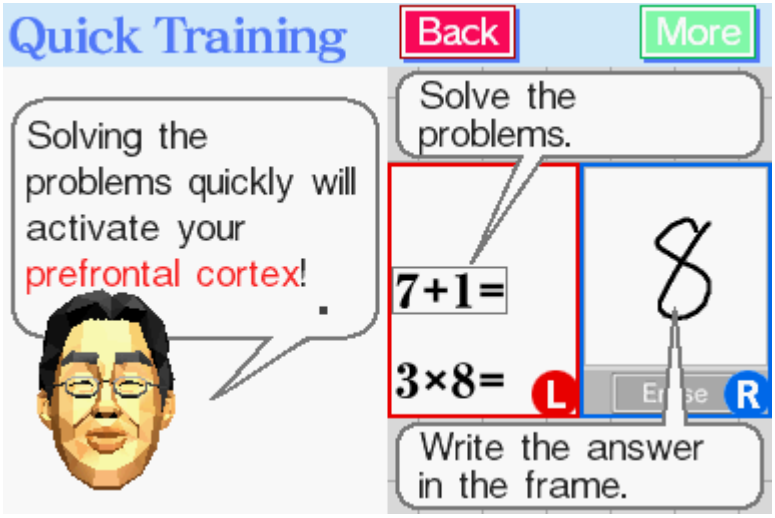
Dessa forma, evidenciamos mais uma fantástica característica referente aos jogos eletrônicos que é a sua ampla variedade de objetos de estudo. Há muitos aspectos que podem ser trabalhados, como toda a teoria que envolve o design de games, os artifícios técnicos relacionados à criação de jogos ou mesmo e as maneiras como os games podem ser aplicados e o impacto que eles causam nas pessoas.

A variedade de tecnologias existentes nesta área foi extensivamente frisada durante a trajetória do trabalho, com várias menções breves do que são apenas algumas dentre todas as opções existentes. Este trabalho objetivou realizar um estudo geral dos diversos artifícios referentes à criação de jogos eletrônicos multiplataforma, os quais possuem uma grande quantidade de conhecimentos agregados, de modo que cada tópico que fora abordado no decorrer desta monografia é passível de se ter trabalhos inteiros dedicados unicamente a eles.

No que se refere a possíveis tópicos de pesquisa, em particular gostaria de destacar o potencial existente no emprego de jogos na área da educação. É bem provável que a primeira coisa que lhe venha a mente sejam os games educacionais de caráter elementar e infantilizado. Saiba que jogos com caráter educacional não precisam ficar presos a este estereótipo e que o aprendizado através de jogos eletrônicos pode estar presente até mesmo em estilos de games mais tradicionais. Os jogos da série *Brain Age*, visto na Figura 20, lançados para a plataforma Nintendo DS objetivam o desenvolvimento do raciocínio através de atividades

matemáticas, lógicas e linguísticas, sempre acompanhados pela orientação do carismático Doutor Kawashima, por intermédio de uma interface simples e neutra que torna o jogo agradável para tanto jogadores mais novos quanto pessoas da terceira idade. Os jogos de estratégia da série Civilization também são um exemplo interessante por não estar enquadrado dentro da ideia de um jogo educacional, apesar de apresentarem uma dinâmica de jogabilidade na qual o jogo transmite conhecimentos de História e Sociologia, enquanto o jogador aplica os saberes que ele já conhece e desenvolve o raciocínio lógico ao elaborar sua estratégia para a vitória enquanto gerencia os seus recursos e lida com as adversidades presentes em seu cenário.

Figura 20: Tela do jogo Brain Age - Train Your Brain in Minutes a Day!



Fonte: http://migre.me/qiyET

No decorrer deste trabalho, almejei retratar muitos dos processos e conhecimentos relacionados aos jogos eletrônicos, e minha expectativa é que este trabalho possa servir de inspiração para outros que também compartilham da paixão por este meio e que acreditam no potencial que este carrega consigo. E que, preferencialmente, este possa ser a gênese para outros projetos focados em design e programação de jogos para múltiplas plataformas, de forma que se possa explorar as particularidades que cada tipo de hardware ou sistema apresentam e poder disponibilizar jogos que poderão ser contemplados pelo maior número possível de jogadores em suas plataformas de escolha.

REFERÊNCIAS

AMARAL, Sílvia Cristina Franco; PAULA, Gustavo Nogueira de. A nova forma de pensar o jogo, seus valores e suas possibilidades. *Pensar a Prática*, Goiânia, v. 10, n. 2, jul./dez. 2007. Disponível em: <www.revistas.ufg.br/index.php/fef/article/download/1098/1676>. Acesso em 06 Out. 2014.

BESSA, Aline et al. O Desenvolvimento de um Motor Multiplataforma para Jogos 3D. 2007. Disponível em: <http://www.inf.ufrgs.br/~cebbezerra/publications/bessa2007dmm.pdf>. Acesso em: 11 jul. 2013.

BHOSLE, Rohit. Unity 3D Game Engine Multi Platform. 2013. Disponível em: <http://www.rohitbhosle.com/portfolio/unity3d/>. Acesso em: 2 dez. 2013.

CASTILHO, Ivan Nikolai Barkow. Grand Theft Auto V quebra sete recordes mundiais com suas vendas. 2013. Disponível em <http://www.ps3brasil.com/noticia.php?id=31297>. Acesso em 5 nov. 2013.

DEITEL, Paul; DEITEL, Harvey. Java: como programar. 8. ed. São Paulo: Pearson Prentice Hall. 2010.

GAME FROM SCRATCH. I want to be a game developer... now what? 2011. Disponível em: <http://www.gamefromscratch.com/post/2011/08/04/I-want-to-be-a-game-developer.aspx>. Acesso em 18 jun. 2013.

GEORGES, Robert A. The Study of Games by Elliott M. Avedon; Brian Sutton-Smith. Western Folklore, v.34, p.155-158, 1975. Disponível em <http://www.jstor.org/stable/1499097>. Acesso em 3 Set. 2014.

HARRIS, Andy. HTML5 Game Development For Dummies. Hoboken, New Jersey: John Wiley & Sons, Inc. 2013

MATTOS, Regiane Cristina Ferreira; FARIA, Moacir Alves de. Jogo e Aprendizagem. *Revista Eletrônica Saberes da Educação*, São Roque, v. 2, n. 1, jan./dez. 2011. Disponível em: <http://www.facsaoroque.br/novo/publicacoes/pdf/v2-n1-2011/Regiane.pdf>. Acesso em 06 Out. 2014.

MICHAELIS. Significado de "multiplataforma". Disponível em: <http://michaelis.uol.com.br/moderno/portugues/index.php?lingua=portugues-portugues&palavra=multiplataforma>. Acesso em: 11 jul. 2013.

NOVAK, Jeanie. Desenvolvimento de Games. São Paulo: Cengage Learning, 2010.

OLIVEIRA, Rômulo Reis de. Desenvolvimento de Jogos Eletrônicos Online em Tempo Real, para Múltiplos Jogadores E Multiplataformas. 2012. Trabalho de Conclusão de Curso (Curso Tecnólogo em Sistemas para Internet) – Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense, Campus Passo

Fundo, Passo Fundo, 2012. Disponível em: <http://inf.passofundo.ifsul.edu.br/graduacao/monografias-defendidas/2012-2/RomuloReisDeOliveira.pdf> Acesso em: 12 mai. 2013.

PEDROSA, Davi et al. Serviços de Adaptacão de Jogabilidade para Jogos Multiplataforma Multiusuário. 2006. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/sbgames/2006/019.pdf>. Acesso em: 11 jul. 2013.

PETRÓ, Gustavo; ARAUJO, Bruno. Mercado de games no Brasil já fatura mais do que no Reino Unido, diz GFK. 2013. Disponível em <http://g1.globo.com/tecnologia/games/brasil-game-show/2013/noticia/2013/10/mercado-de-games-no-brasil-ja-fatura-mais-do-que-no-reino-unido-diz-gfk.html>. Acesso em 5 nov. 2013.

SHAUL, Brandy. GameMaker: Studio introduces YoYo Compiler and cross-platform Shader support. 2013. Disponível em <http://www.adweek.com/socialtimes/gamemaker-studio-introduces-yoyo-compiler-and-cross-platform-shader-support/542353>. Acesso em 19 mai. 2015.

SCHILDT, Herbert. C++: guia para iniciantes. Rio de Janeiro: Editora Ciência Moderna Ltda., 2002.

SCHUYTEMA, Paul. Design de games: uma abordagem prática. São Paulo: Cengage Learning, 2013.

SILVA, Maurício Samy. HTML5 A linguagem de marcação que revolucionou a web. São Paulo: Novatec Editora, 2011.

VARELLA, João; MARCEL, Diego. Pátria dos games. 2013. Disponível em <http://www.istoedinheiro.com.br/noticias/120016_PATRIA+DOS+GAMES>. Acesso em 5 nov. 2013.

VIANNA, Ysmar. et al. Gamification, Inc. Como reinventar empresas a partir de jogos. – 1. ed. – Rio de Janeiro: MJV Press, 2013.

XAVIER, Thomaz Canali. Estudo e Desenvolvimento de Jogos para Internet Utilizando Unity 3D. 2011. Trabalho de Conclusão de Curso (Curso Tecnólogo em Sistemas para Internet) – Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense, Campus Passo Fundo, Passo Fundo, 2011. Disponível em: <http://inf.passofundo.ifsul.edu.br/graduacao/monografias-defendidas/2011/ThomazXavier.pdf > Acesso em: 12 mai. 2013.

YUNG, Rodrigo. Indústria de games móveis deve movimentar US$ 8 bilhões em 2013. 2013. Disponível em <http://codigofonte.uol.com.br/noticias/industria-de-games-moveis-deve-movimentar-us-8-bilhoes-em-2013>. Acesso em 5 nov. 2013.

APÊNDICES

APÊNDICE 1 – Documento de Design do Game “Desafio do Lorde das Trevas”

DOCUMENTO DE GAME DESIGN

Nome do Projeto: The Dark Lord’s Challenge / Desafio do Lorde das Trevas

Autor: Alisson Alberti Tres

Plataformas alvo: Computadores pessoais, smartphones e tablets

Descrição: Jogo de ação, de caráter casual e arcade, com o objetivo de alcançar a maior pontuação possível antes de ser derrotado

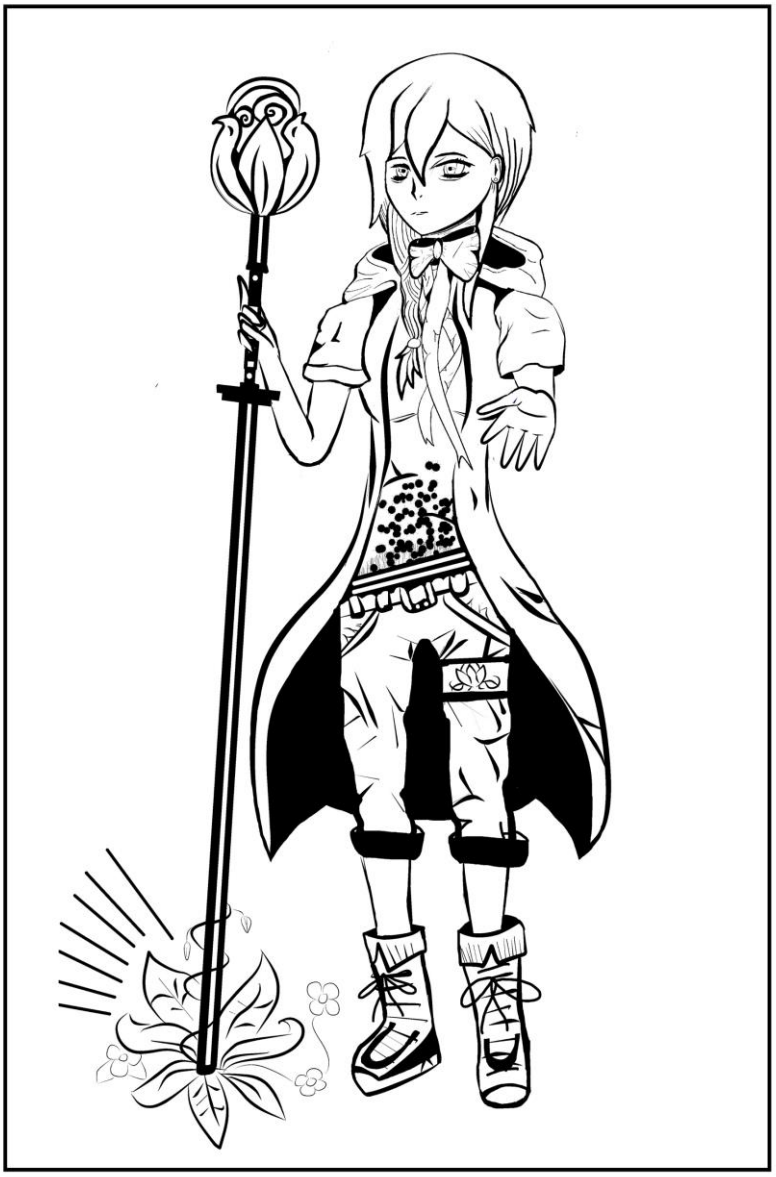
Índice

Premissa do Jogo:

História:

Um reino distante vive tempos de crise, estando sob ameaça do inescrupuloso e sádico Lorde das Trevas. Carregado de um grande orgulho, o Lorde desafia o monarca com o propósito de colocar à prova as habilidades dos súditos deste reino e a destreza de seu exército de criaturas horrendas. Eis que o monarca do reino aceita o desafio e atribui à sua mais talentosa Feiticeira a tarefa de derrotar o maior número possível de elementos desta tropa maligna.

Personagens:



A Feiticeira

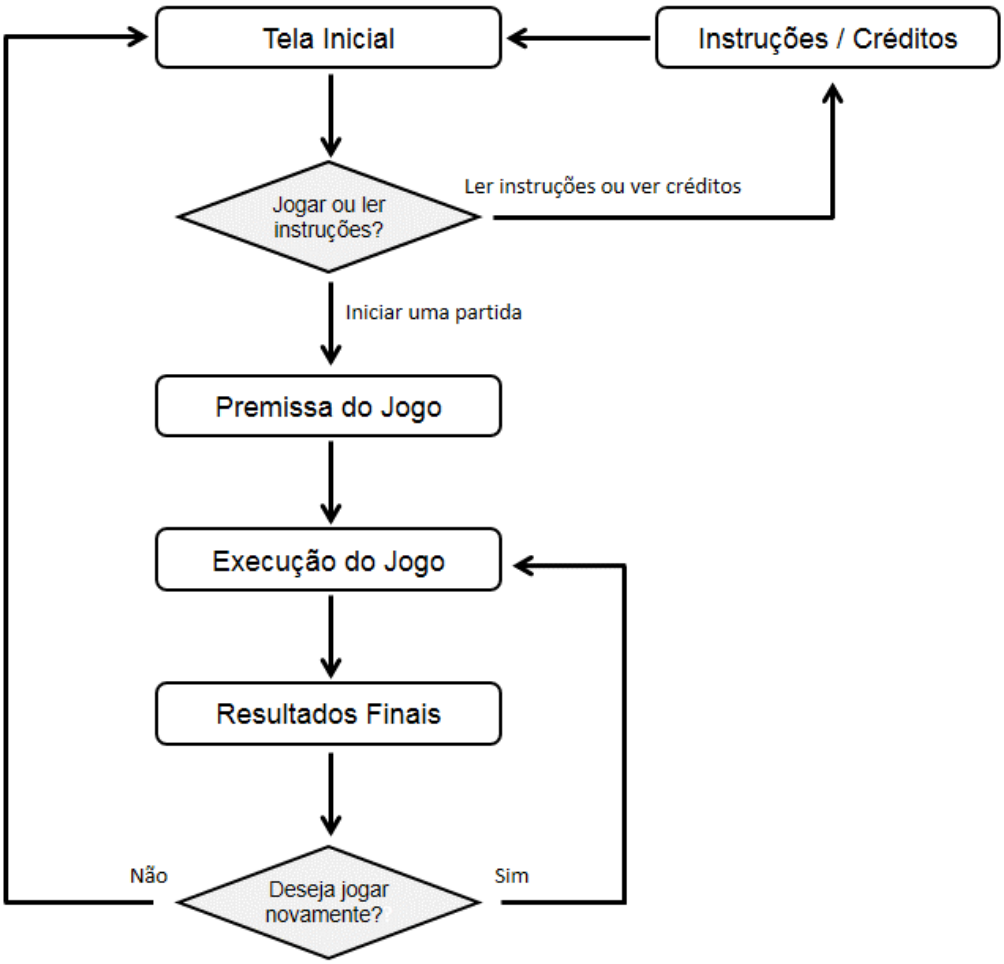
Especialista no domínio e manipulação das forças da natureza, a Feiticeira é adorada por todos no reino e é respeitada pelos mais nobres membros da hierarquia real. A Feiticeira é vista como uma encarnação da bondade e um alicerce para a esperança em mundo de miséria e escuridão. A Feiticeira é uma mulher de poucas palavras, priorizando expressar-se através de suas ações. Por vezes as altas expectativas provindas das pessoas ao seu redor a deixam nervosa, mas a Feiticeira jamais hesita quando há alguém em apuros clamando por auxílio. Suas habilidades também incluem o domínio da alquimia e exercício da fitoterapia.

O Lorde das Trevas

Autoridade máxima no domínio das artes ocultas, trata-se de uma entidade dotada de uma personalidade sádica, egocêntrica e intimidadora. Não há mensuração para o seus poderes, capazes de manipular os elementos. O Lorde das Trevas é culturalmente visto como a fonte de toda a maldade e agonia que existem no mundo. Há quem afirme que o Lorde das Trevas deseja dominar o mundo e escravizar a humanidade, mas suas ações pouco refletem tal objetivo. Não se sabe ao certo quais são os seus verdadeiros anseios, se comportando sempre de forma misteriosa. Sua posição perante o monarca do reino propõe nada menos do que um mera competição amigável.

Fluxo e Telas da Aplicação:

Fluxograma de execução e descrição de telas da aplicação:



Tela de Inicial

Lista o título da monografia a qual este jogo faz parte e o título do jogo em si. Apresenta botões para iniciar uma partida ou visualizar as instruções e informações sobre o jogo.

Instruções / Créditos

Exibe uma explicação do funcionamento e objetivos deste jogo, assim como também a listagem dos autores responsáveis pelos recursos (programação, gráficos e áudio) utilizados em sua construção.

Premissa do Jogo

Tela que antecede o início da partida, introduzindo o jogador ao contexto do universo em que as ações deste jogo ocorrem. Isto é feito através da exibição de um parágrafo apresentando a breve narrativa criada para este o jogo.

Execução do Jogo

Tela a qual apresenta o momento da aplicação em que o jogo propriamente dito ocorre. Dentre todas as telas existentes, é o único momento em que a lógica da jogabilidade é executada.

A natureza do funcionamento deste jogo é feita de uma partida inteira seja executada nesta tela. Aqui são carregados todos os elementos relacionados à interface do jogo e será executada toda a lógica do jogo.

A progressão dos níveis ocorre de modo que não há a necessidade de alternar para um outro tipo de tela. Todas as ações ocorrem durante a execução de um mesmo ciclo, que é interrompido no momento em que se alcança a condição de derrota.

Resultados Finais

Ao finalizar uma partida, esta tela será executada. Nela serão apresentadas ao jogador a pontuação final alcançada no decorrer da partida e as opções para ou começar uma nova partida ou retornar para a tela inicial da aplicação.

Jogabilidade:

Objetivo:

O jogador deverá alcançar o nível mais alto que conseguir, derrotando os conjuntos de inimigos presentes em cada nível, até que os pontos de vida da feiticeira cheguem a zero.

Para esta tarefa, o jogador conta com um número limitado de recursos: ataques diretos, que podem ser utilizados a qualquer momento mas não possuem muita força, e magias, que apesar de serem poderosas, custam uma determinada quantidade de pontos de magia e possuem um intervalo de tempo entre cada uso.

O jogador deverá desenvolver a melhor maneira de gerenciar seus recursos para conseguir derrotar o maior número possível de inimigos.

Controles:

Dispositivos de entrada utilizados: Mouse (em computadores) ou Tela tátil (em dispositivos que possuem suporte a esta tecnologia, como smartphones e tablets).

Durante as telas iniciais de menus e seleções:

Clique com o botão esquerdo do mouse ou *toque na tela tátil*: realiza a tomada de decisão de qual a próxima tela da aplicação a ser exibida.

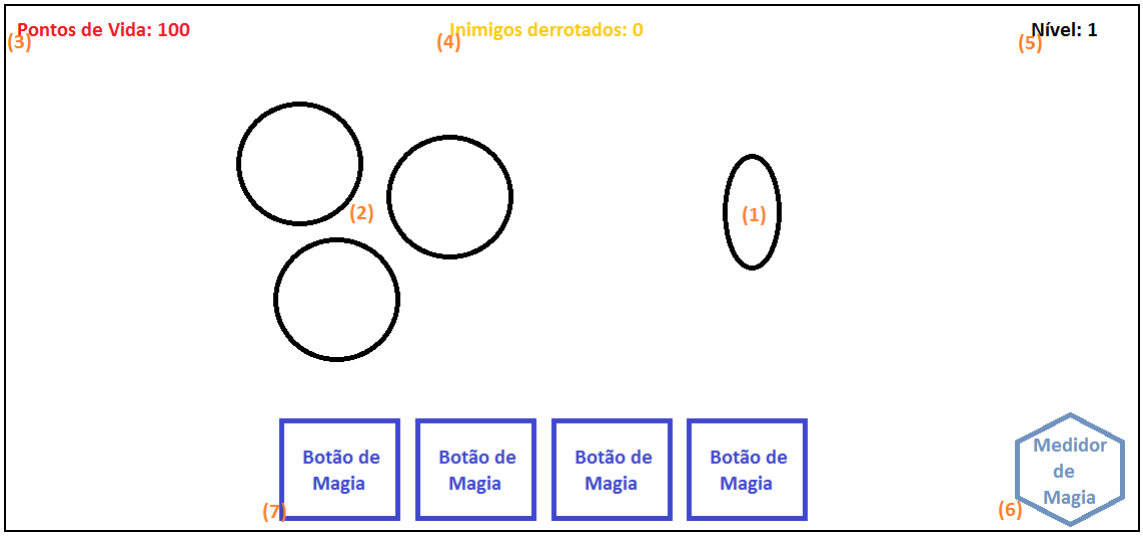
Durante a tela de jogo:

Clique com o botão esquerdo do mouse ou *toque na tela tátil*: realiza uma ação, dependendo do elemento da interface com o qual foi interagido.

Quando há interação com os inimigos na tela, é realizado o ataque direto que subtrai um determinado valor dos pontos de vida do inimigo em questão.

Quando há interação com um ícone de magia, esta fica selecionada para quando ocorrer o próxima interação com um inimigo, ocorrer a execução deste ataque especial contra o inimigo escolhido. Esta ação poderá causar uma grande quantidade de dano ou uma quantidade nula de dano dependendo das fraquezas e resistências que um inimigo possui.

Interface do Jogo:



1. A Feiticeira, o avatar do jogador dentro do universo do jogo;
2. Grupo de inimigos a serem enfrentados no nível atual;
3. Listagem da quantidade de pontos de vida que o jogador possui no momento;
4. Listagem da quantidade de inimigos derrotados / pontuação do jogador até o momento;
5. Listagem do nível em que o jogador se encontra no momento;
6. Listagem da quantidade de pontos de magia que o jogador possui no momento.
7. Opções de magias que o jogador dispõe para atacar os inimigos presentes no momento.

Progressão do Jogo:

O jogo é composto por diversos níveis, cada um contendo um conjunto específico de um até três inimigos atribuídos/escolhidos aleatoriamente. Ao iniciar uma partida o jogo irá gerar o primeiro nível, composto pelo primeiro conjunto de inimigos.

Um nível é considerado como concluído no momento em que os pontos de vida de todos os inimigos do conjunto presente neste nível sejam zerados. Cada inimigo derrotado acresce um determinado valor na pontuação final do jogador. Logo em seguida é iniciado próximo nível, o que significa que o jogo deverá gerar o próximo conjunto de inimigos a ser enfrentado.

Não há um final propriamente dito para ser alcançado no jogo, sendo que este ciclo se repete até que os pontos de vida do jogador sejam zerados. Quando isto ocorre, a parte funcional do jogo é dada como encerrada e ocorre a passagem para tela na qual a pontuação final do jogador é calculada (pontos adquiridos por cada inimigo derrotado multiplicado pelo nível alcançado pelo jogador) e exibida na tela.

Junto com a pontuação final também é oferecido ao jogador a opção de imediatamente iniciar uma nova partida ou retornar para a tela inicial da aplicação.

Resultado final:

Ao término de uma partida ocorre o cálculo da pontuação alcançada pelo jogador, resultante da quantidade de inimigos derrotados e da quantidade de níveis concluídos pelo jogador.

Lista de Objetos do Jogo:

Abaixo está descrita a listagem de objetos que irão compor o jogo, com seus respectivos atributos básicos.

Atributos do jogador:

Personagem jogável: A Feiticeira	
Força Inicial: 1	Pontos de vida: 100
Variável para armazenar Pontuação	
Variável para controlar Nível atual	
Pontuação final = Pontuação x Nível	

Lista de inimigos do jogo:

Número identificador: 1	
Nome: Cogumelo Ambulante	
Força: 3	Pontos de vida: 10
Frequência de ataques: A cada 6 segundos	
Fraqueza: Fogo	
Resistente a: Gelo	
Pontos adquiridos ao ser derrotado: 100	

Número identificador: 2	
Nome: Guerreiro Esqueleto	
Força: 5	Pontos de vida: 15
Frequência de ataques: A cada 4 segundos	
Fraqueza:Gelo	
Resistente a: Fogo	
Pontos adquiridos ao ser derrotado: 250	

Número identificador: 3	
Nome: Armadura Assombrada	
Força: 7	Pontos de vida: 15
Frequência de ataques: A cada 5 segundos	
Fraqueza: Nenhuma	
Resistente a: Todos os tipos de magias	
Pontos adquiridos ao ser derrotado: 500	

Lista de Créditos e Recursos utilizados no Jogo:

Abaixo segue a listagem dos indivíduos que possuem algum envolvimento, seja ele direto ou indireto, no processo de desenvolvimento deste jogo:

Design de Jogo, Programação e Gráficos 2D:
Alisson Alberti Tres

Supervisão de Projeto:
Lisandro Lemos Machado

Arte Conceitual de Personagens:
Gabriela Fritzen

Efeitos sonoros:
Alisson Alberti Tres – Utilizando o software de sintetização de efeitos sonoros *Bfxr*

Músicas de background:
Jun'ya Ōta (ZUN)
Músicas originalmente compostas para os jogos da série *Touhou Project*, desenvolvido pela *Team Shanghai Alice*. Utilizadas neste projeto para fins educacionais e não-comerciais.

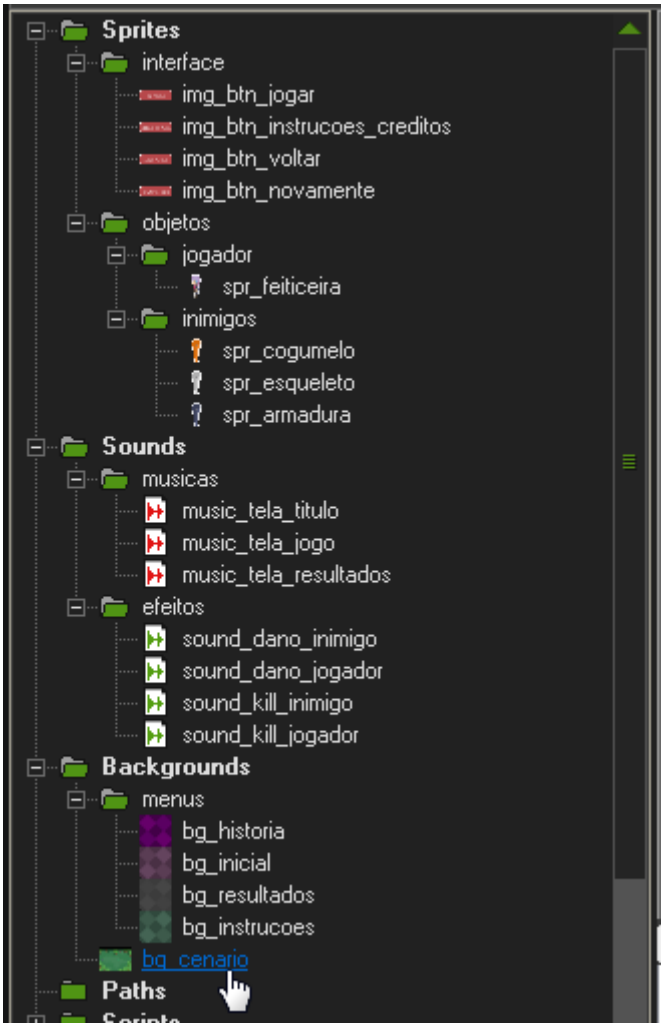
- Tela Inicial: *Illusionary Night ~ Ghostly Eyes (*do game *Touhou Eiyashou ~ Imperishable Night)*
- Execução do Jogo: *Doll Judgment ~ The Girl who Played with People's Shapes (*do game *Touhou Youyoumu ~ Perfect Cherry Blossom)*
- Resultados Finais*: Emotional Skyscraper ~ Cosmic Mind (*do game *Touhou Seirensen ~ Undefined Fantastic Object)*

APÊNDICE 2 – Descrição do projeto do protótipo no Game Maker: Studio

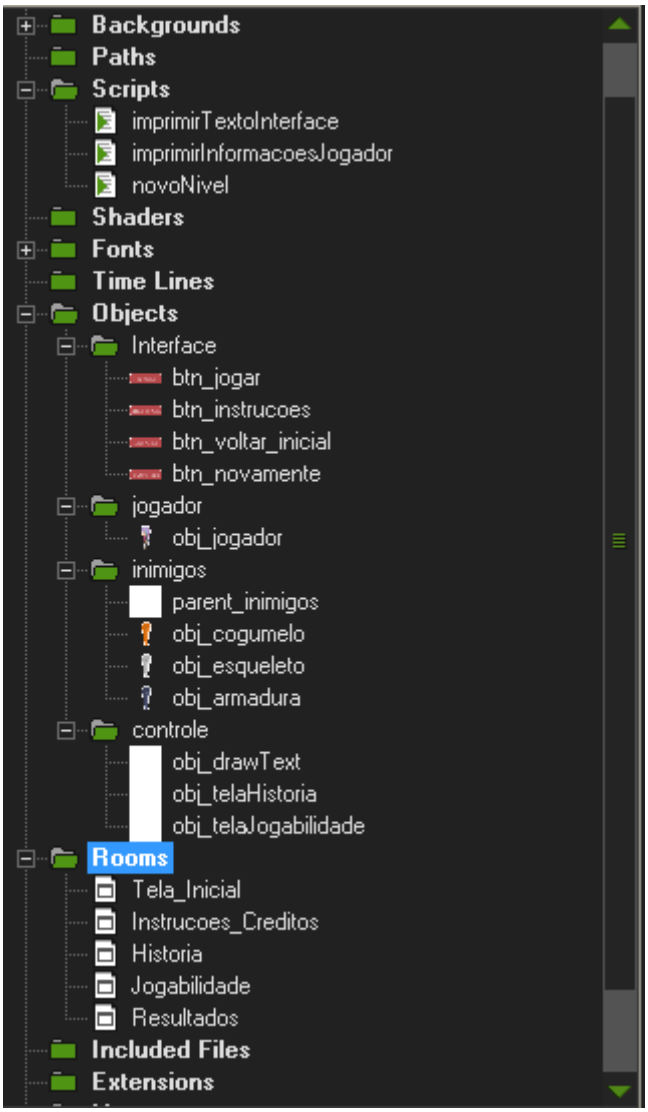
Por meio deste apêndice, pretende-se mostrar um pouco da interface do projeto criado no *Game Maker: Studio* e mostrar as configurações realizadas em alguns de seus objetos mais pontuais.

O projeto do protótipo no Game Maker: Studio está disponível no **Github** neste endereço:
https://github.com/alisson-tres/TCC-Prototipo-de-jogo-eletronico-no-Game-Maker-Studio

Através da árvore de diretórios do *Game Maker: Studio*, podemos observar todos os tipos de arquivos que foram carregados para dentro do projeto:



Na árvore de diretórios também encontramos todos os elementos internos que são utilizados pela ferramenta e que foram criados dentro dela:



Três scripts feitos com a linguagem GML foram definidos individualmente dentro do grupo de *Scripts*.

imprimirTextoInterface

```
1
2 ////////////////////////////////////////////////////////////////////
3 if(room == 0) {
4 //Imprime na tela textos relacionados à Tela Inicial da Aplicação
5     draw_set_colour(c_white);
6     /* draw_set_colour() estabelece a cor a ser usada em um objeto a ser renderizado */
7     /* o Game Maker:Studio possui um determinado conjunto de constantes para cores */
8
9     draw_set_font(font_geral);
```

```
10    /* draw_set_font() estabelece a fonte a ser usada para imprimir texto na tela */
11    /* a fonte padrão é Arial 12, mas pode-se carregar uma das fontes que você estabeleceu
12    na categoria Fonts */
13
14    draw_set_halign(fa_center);
15    draw_set_valign(fa_top);
16    /* draw_set_halign() e draw_set_valign()determinam o alinhamento horizontal e vertical
do texto a ser exibido na tela */
17
18    draw_text(room_width/2,10,"ESTUDO DE TÉCNICAS E PROCESSOS
REFERENTES AO#DESENVOLVIMENTO DE JOGOS ELETRÔNICOS
MULTIPLATAFORMA");
19    /* Digitando o caractere # você pode fazer quebras de linha no texto a ser exibido na tela
*/
20
21    draw_set_font(font_tela_inicial_titulo);
22    draw_text(room_width/2,room_height/2,"DESAFIO DO LORDE DAS TREVAS");
23
24    draw_set_font(font_tela_inicial_autor);
25    draw_text(room_width/2,450,"2015 Alisson Alberti Tres");
26
27 }
28
29 ////////////////////////////////////////////////////////////////////
30 if(room == 1) {
31 //Imprime na tela textos relacionados às Instruções e Créditos do jogo
32
33    draw_set_colour(c_white);
34    draw_set_font(font_geral);
35
36    draw_text(room_width/2,25,"COMO JOGAR:");
37
38    draw_text(room_width/2,50,"> Cliques com o botão esquerdo do mouse");
39    draw_text(room_width/2,75,"> Toques na tela tátil");
40
41    draw_text_ext(room_width/2,190,"Clique ou toque nos inimigos para ataca-los e causar-
lhes dano.#Você ganhará pontos por cada inimigo derrotado.#Derrote o máximo possível de
inimigos antes que os seus pontos de vida acabem!",25,620);
42    /* Estabelece uma "caixa de texto" com quebra de linha automática, possuindo um
determinado espaçamento entre linhas, dentro de uma determinada largura */
43 }
44
45 ////////////////////////////////////////////////////////////////////
46 if(room == 2) {
47 //Imprime na tela textos relacionados à Tela de História do jogo
48
49    draw_set_colour(c_white);
50
51    draw_set_font(font_geral);
52
```

```
53     draw_set_halign(fa_center);
54
55     draw_text_ext(room_width/2,10,"Um reino distante vive tempos de crise, estando sob ameaça do inescrupuloso e sádico Lorde das Trevas. Carregado de um grande orgulho, o Lorde desafia o monarca com o propósito de colocar à prova as habilidades dos súditos deste reino e a destreza de seu exército de criaturas horrendas. Eis que o monarca do reino aceita o desafio e atribui à sua mais talentosa Feiticeira a tarefa de derrotar o maior número possível de elementos desta tropa maligna.",25,620);
56     /* Estabelece uma "caixa de texto" com quebra de linha automática, possuindo um determinado espaçamento entre linhas, dentro de uma determinada largura */
57
58     draw_text(room_width/2,440,"CLIQUE NA TELA PARA COMEÇAR!");
59 }
60
61 if(room == 4) {
62 //Imprime na tela textos relacionados à Tela de Resultados do jogo
63
64     draw_set_colour(make_colour_rgb(255, 77, 77));
65     draw_set_font(font_geral);
66     draw_set_halign(fa_center);
67     draw_text(room_width/2,room_height/2 - 60,"GAME OVER!");
68
69     draw_set_colour(c_white);
70     draw_text(room_width/2,room_height/2 - 30,"Pontos adquiridos: " + string(score));
71
72     draw_text(room_width/2,room_height/2,"Nível alcançado: " + string(global.nivel));
73
74     draw_set_colour(make_colour_rgb(207, 207, 83));
75     draw_text(room_width/2,room_height/2 + 30,"Sua pontuação final: " + string(score * global.nivel) );
76 }
```

O script ***imprimirTextoInterface*** renderiza os textos que aparecem nas Rooms de menus antes e após a execução da jogabilidade.

imprimirInformacoesJogador

```
1 //Durante a execução de uma partida, este script imprime na tela texto relativo aos dados do jogador: seus pontos de vida, sua pontuação e seu nível atual
2
3 draw_set_font(font_interface_jogador);
4
5 ///Draw Pontos de Vida
6 draw_set_colour(make_colour_rgb(255, 48, 48));
7
8 draw_set_halign(fa_left);
9 draw_set_valign(fa_top);
10
11 draw_text(10,15,"Pontos de Vida: " + string(health));
```

```
12 /* Ao concatenar strings com outros tipos de variáveis, deve-se converte-las em strings. Caso contrário, ocorrerão erros na execução do jogo, pois o Game Maker não permite esse tipo de concatenação entre variáveis de tipos diferentes. */
13
14 ///Draw Pontuação
15 draw_set_colour(c_white);
16
17 draw_set_halign(fa_center);
18
19 draw_text(room_width/2,15,"Pontuação: " + string(score));
20
21 ///Draw Nível
22 draw_set_colour(c_white);
23
24 draw_set_halign(fa_right);
25
26 draw_text(600,15,"Nível: " + string(global.nivel));
```

O script ***imprimirInformacoesJogador*** é usado para renderizar os dados do jogador durante a execução de uma partida no jogo.

novoNível

```
1 global.nivel++;
2 global.qtd_inimigos = 3;
3
4 for(i = 0; i < 3; i++) {
5    //Neste protótipo, cada inimigo possui uma posição fixa na tela do jogo
6    switch(i) {
7       case 0:
8          posX = 96;
9          posY = 208;
10          break;
11       case 1:
12          posX = 160;
13          posY = 144;
14          break;
15       case 2:
16          posX = 160;
17          posY = 272;
18          break;
19    }
20
21    /* Escolhe aleatoriamente um número de 1 a 3 para determinar a escolha de um tipo de inimigo */
22    var inimigo_tipo = floor(random(3)) + 1;
23
24    /* Cria-se uma instância de um determinado tipo de inimigo na tela do jogo */
25    switch(inimigo_tipo) {
```

```
26        case 1:
27            inimigos[i] = instance_create(posX,posY,obj_cogumelo);
28            break;
29        case 2:
30            inimigos[i] = instance_create(posX,posY,obj_esqueleto);
31            break;
32        case 3:
33            inimigos[i] = instance_create(posX,posY,obj_armadura);
34            break;
35    }
36 }
```

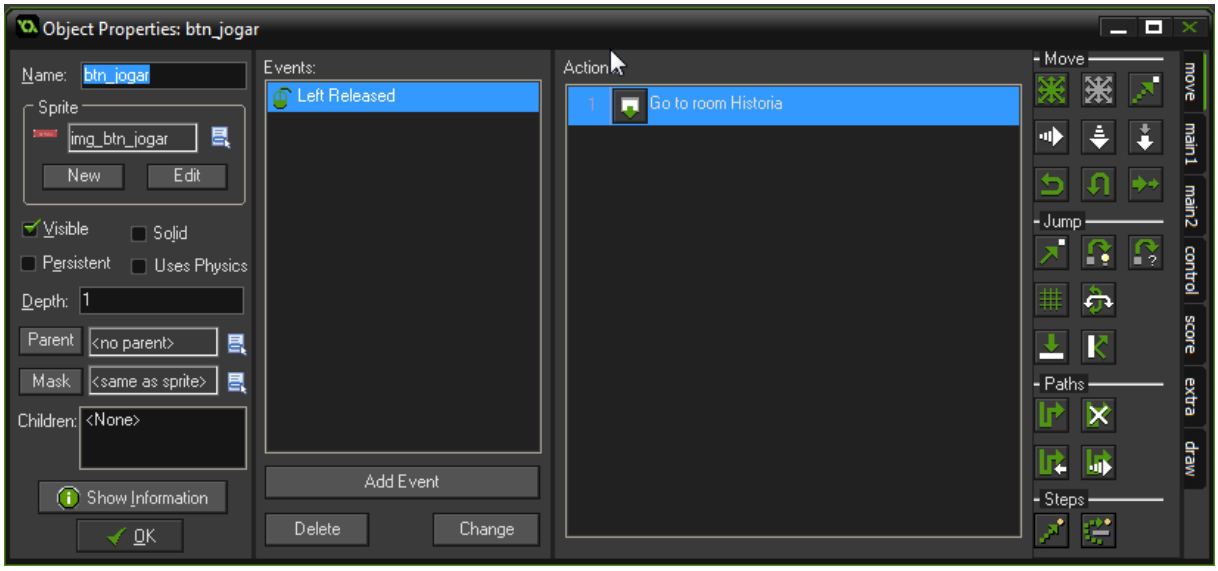
O script ***novoNivel*** contém a rotina que gera um novo conjunto de inimigos para serem enfrentados pelo jogador.

Creation code da *room* Tela_Inicial

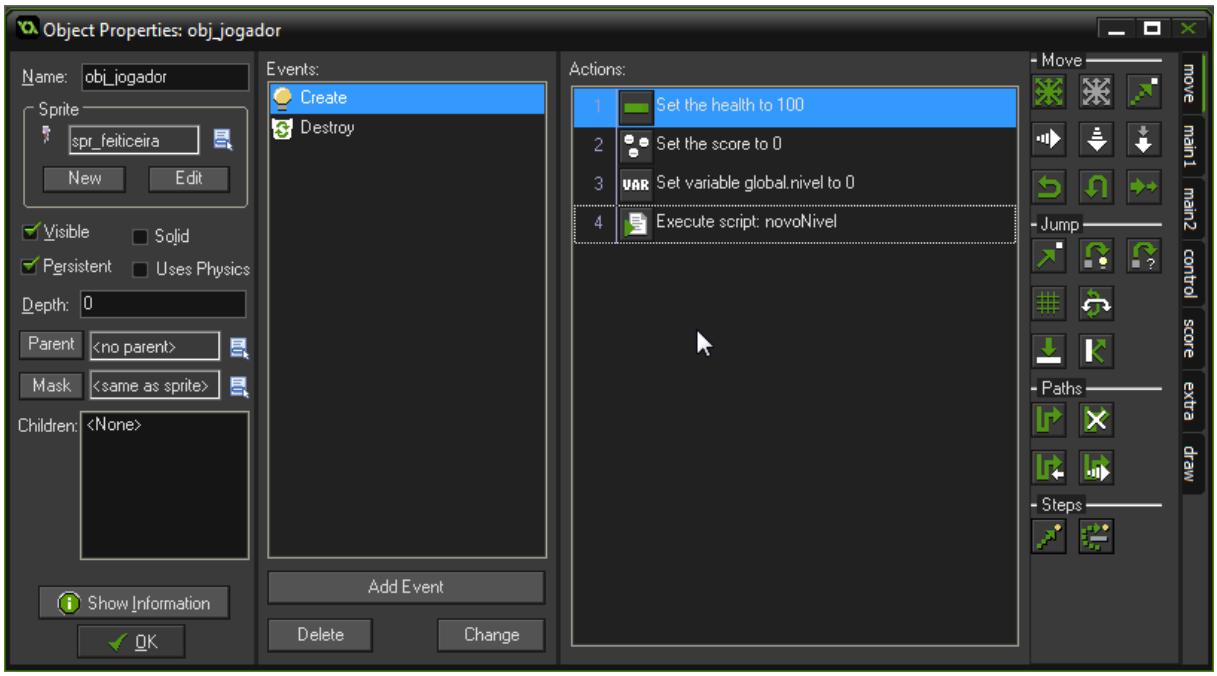
```
1 //Verifica se a música das telas iniciais está em execução...
2 if !audio_is_playing(music_tela_titulo) {
3     //Interrompe a execução de todos os outos audios em execução no momento
4     audio_stop_all();
5     //Chamada do método para executar uma música em loop
6     audio_play_sound(music_tela_titulo, 10, true);
7 }
```

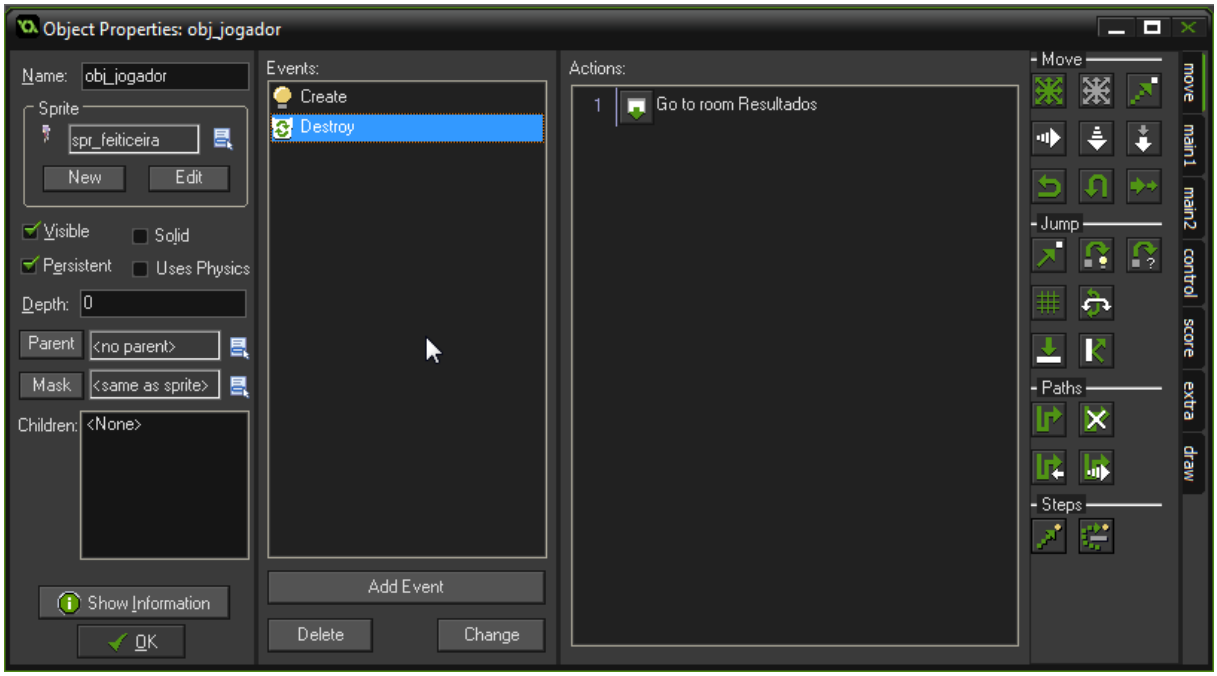
Scripts também podem ser configurados para serem executados automaticamente ao início de uma determinada *Room*. Isso foi aplicado no jogo para executar as músicas de fundo. Acima, temos como o exemplo a rotina que é executada toda vez que a *room* Tela_Inicial é carregada.

Os objetos da aplicação foram subdivididos em três categorias: *interface*, *jogador*, *inimigos* e *controle*.

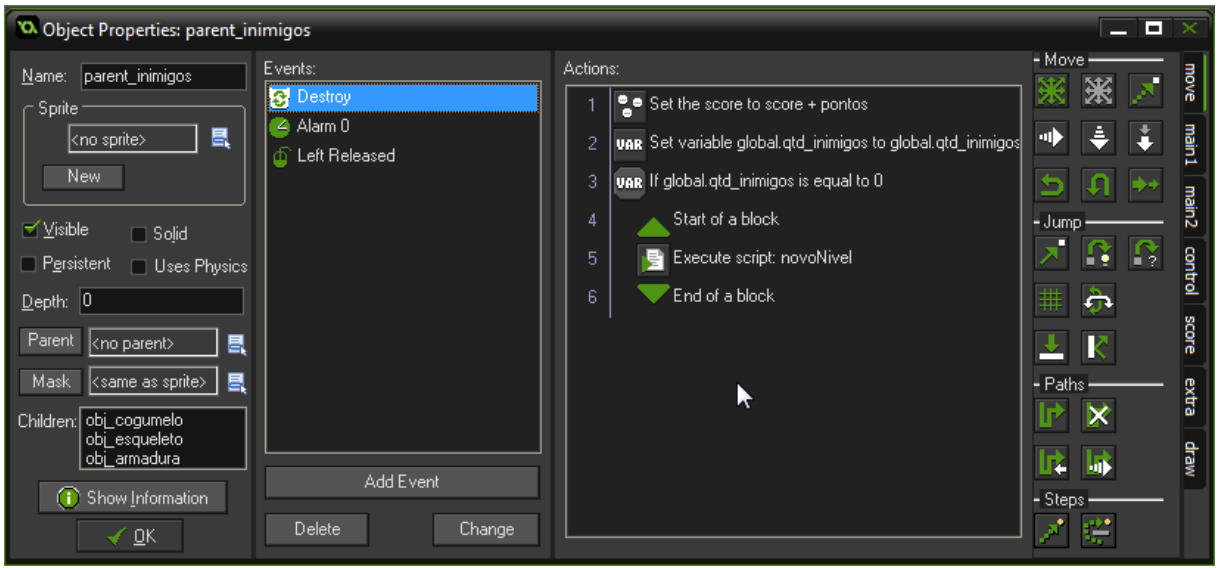


O subgrupo *Interface* contém os objetos de todos os botões da interface do jogo. Cada um foi configurado para realizar a transição para uma outra Room, ou seja, alterar a tela em execução na aplicação.



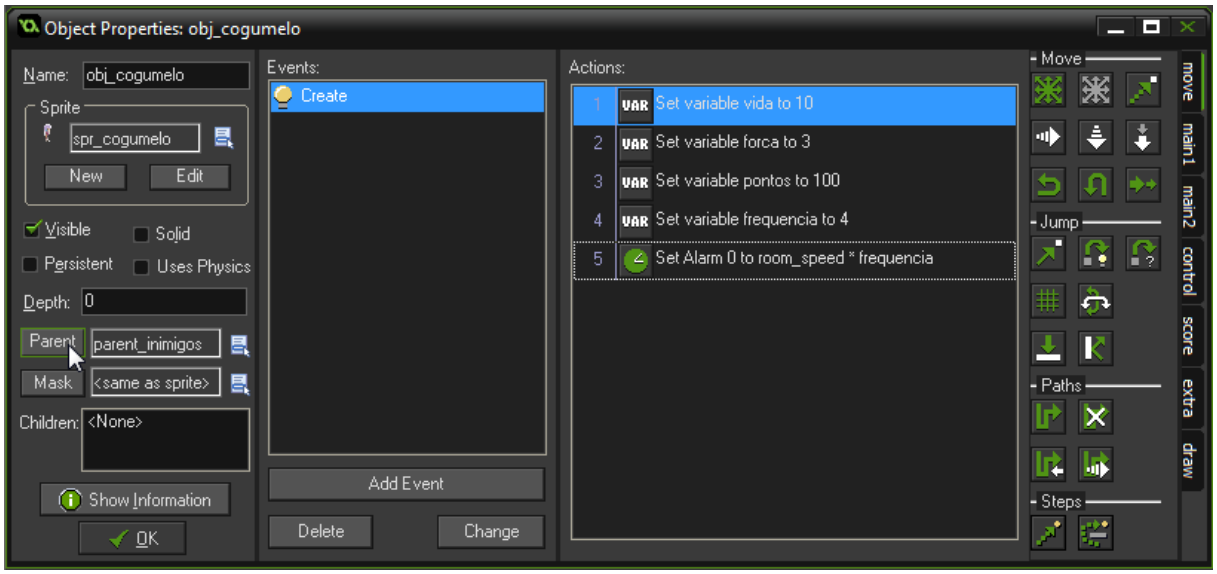


O subgrupo Jogador contém um objeto que não somente mostra a Feiticeira na tela, mas também foi feito para realizar o controle da inicialização dos atributos do jogador (pontos de vida, pontuação do jogador, nível atual) e o controle de quando a partida deve ser encerrada (que ocorre no momento em que este objeto for destruído).

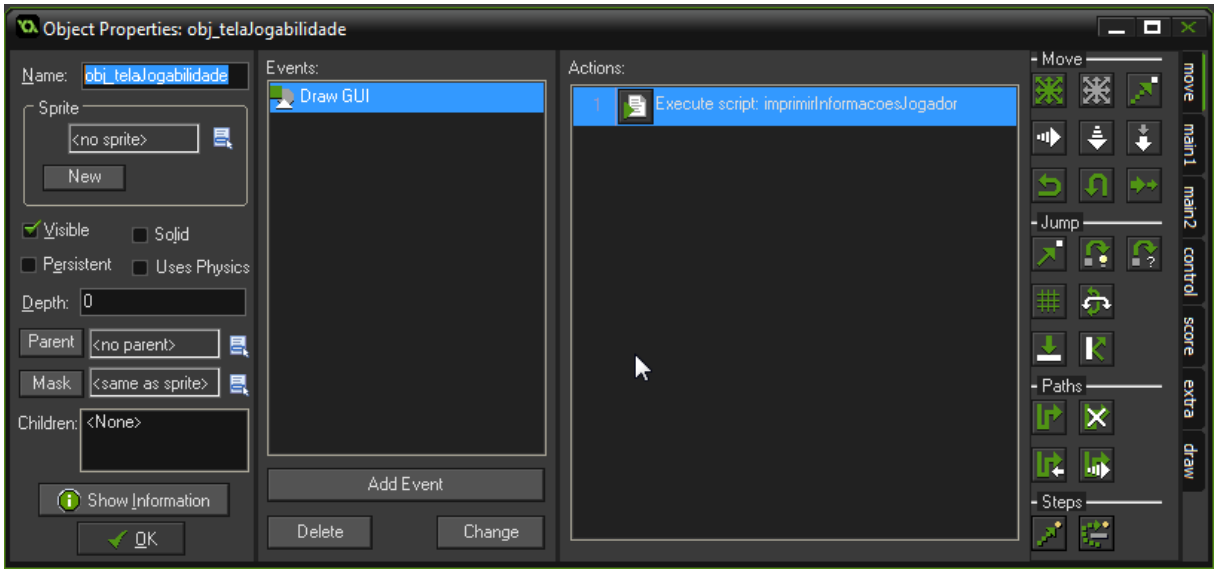


Todos os objetos de inimigos a serem enfrentados pelo jogador. Embora cada inimigos tenha seus próprias características, todos possuem alguns comportamentos em comum: quando são clicados, é subtraído um determinado valor de seus próprios pontos de vida; através de um Alarm, eles atacam o jogador de tempos em tempos; e quando destruídos, um determinado valor é adicionado à pontuação do jogador.

Aplicando o conceito de herança da Orientação à Objeto, o *Game Maker: Studio* possibilita criar uma classe Pai, a qual pode ser atribuída a outros objetos. No caso, o objeto chamado *parent_inimigos* agrega todos estes comportamentos comuns.

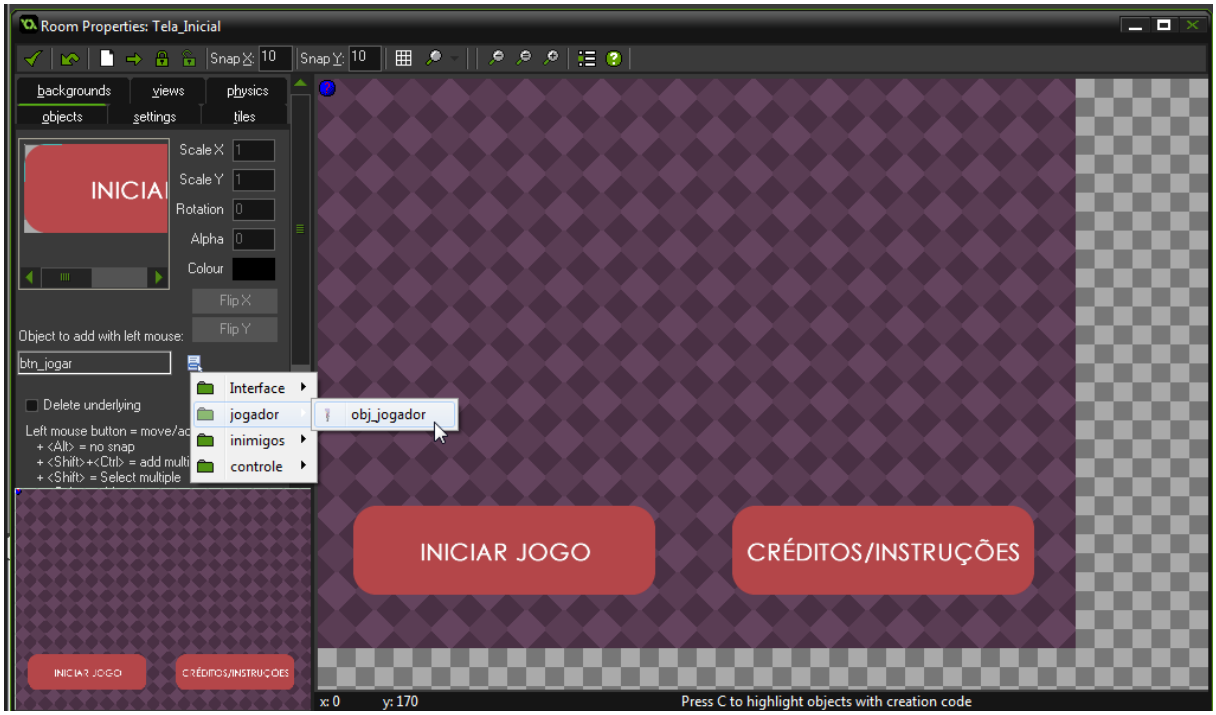


Cada inimigo propriamente dito, por sua vez, terá sua rotina de inicialização particular. No campo ***Parent***, é determinado que este objeto herda todos os eventos definidos para o *parent_inimigos*.



Os objetos do subgrupo *Controle* se diferenciam dos demais por serem objetos que não são visíveis (note que a caixa ***Visible*** está desmarcada). Isso significa que estes objetos não possuem um sprite para representa-los na tela. Mas isso não significa que eles não estejam presentes na tela. Objetos desta natureza podem ser usados para criar rotinas que ficam rodando em paralelo à execução do jogo. No caso deste

protótipo, foi criado o objeto de controle *obj_drawText* para fazer a chamada do script de renderizar texto na tela, o objeto *obj_telaHistoria* para identificar iniciar o fluxo do jogo quando ocorrer um clique na tela e o objeto *obj_telaJogabilidade* para executar o script que renderiza os dados do jogador na tela.



Com os objetos finalizados, utilizamos o editor de Rooms para que os estes possam plenamente fazer parte da execução do jogo.

APÊNDICE 3 – Código-fonte do protótipo em HTML5

O código-fonte do protótipo em HTML5 está disponível no **Github** neste endereço:

https://github.com/alisson-tres/TCC-Prototipo-de-jogo-eletronico-em-HTML5

TCC Alisson Protótipo HTML5\index.html

```
1 <!DOCTYPE html>
 2 <html>
 3   <head>
 4     <title>Estudo de Técnicas e Processos referentes ao Desenvolvimento de Jogos Eletrônicos Multiplataforma</title>
 5     <meta charset="UTF-8">
 6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
 7     <meta name="author" content="Alisson Alberti Tres">
 8     <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">
 9     <link rel="stylesheet" type="text/css" href="css/style.css">
10   </head>
11   <body>
12     <div id="debug_coordenadas">X: <span id="click_X"></span> | Y: <span id="click_y"></span></div>
13     <div id="conteudo">
14       <canvas id="jogo" width="640" height="480">Seu navegador não possui suporte ao recurso canvas do HTML5. Sinto muito. =,(</canvas>
15     </div>
16     <script type="text/javascript" src="js/jogo.js"></script>
17   </body>
18
19   <!--Para utilizar arquivos de áudio em nosso jogo, eles devem ser definidos no DOM da página HTML para serem carregados na página-->
20   <audio id="music_tela_inicial" loop>
21     <source src="recursos/audio/illusionarynight.mp3" type="audio/mpeg">
22   </audio>
23   <audio id="music_tela_jogo" loop>
24     <source src="recursos/audio/dolljudgment.mp3" type="audio/mpeg">
25   </audio>
26   <audio id="music_tela_resultados">
27     <source src="recursos/audio/emotionalskyscraper.mp3" type="audio/mpeg">
28     <br>Desculpa, mas o seu navegador não possui suporte às músicas de fundo do jogo... =,(
29   </audio>
30   <audio id="sfx_hit_jogador">
31     <source src="recursos/audio/hit_jogador.wav" type="audio/wav">
32   </audio>
33   <audio id="sfx_hit_inimigo">
```

```
34        <source src="recursos/audio/hit_inimigo.wav" type="audio/wav">
35    </audio>
36    <audio id="sfx_kill_jogador">
37        <source src="recursos/audio/kill_jogador.wav" type="audio/wav">
38    </audio>
39    <audio id="sfx_kill_inimigo">
40        <source src="recursos/audio/kill_inimigo.wav" type="audio/wav">
41        Lamento, mas o seu navegador não possui suporte aos efeitos sonoros do jogo... =,(
42    </audio>
43 </html>
```

TCC Alisson Protótipo HTML5\css\style.css

```
1 /*
2    Folha de estilo para a página na qual é executado o jogo em HTML5
3    Autor     : Alisson A. Tres
4 */
5
6 body { background-color: #666666 }
7
8 #debug_coordenadas { position: absolute; }
9
10 #conteudo { width: 100%; height: 100%; }
11
12 canvas { position: absolute; top:0; bottom: 0; left: 0; right: 0; margin:auto; background-color: #000; }
```

TCC Alisson Protótipo HTML5\js\jogo.js

```
1 /*
2  * Código-fonte do protótipo em HTML5 do jogo 'Desafio do Lorde das Trevas'
3  * Autor     : Alisson Alberti Tres
4  */
5
6 // Variáveis Globais
7 var tela = 0; /* Controla qual a tela na qual a aplicação se encontra */
8 var jogoCanvas; /* Contém a referência ao elemento Canvas presente na página */
9 var ctx; /* objeto de desenho 2D para imprimir elementos dentro do Canvas */
10 var carregado = 0;
11 var rotinaCarregarRecursos;
12
13 var canvas_x = ""; /* Armazena a posição X de um clique/toque na Canvas */
14 var canvas_y = ""; /* Armazena a posição Y de um clique/toque na Canvas */
15
16 var fps = 30; /* Determina a quantidade de quadros por segundo na qual o jogo estará rodando */
17
18 // Vetores e Objetos
19 var jogador; /* Guarda os dados básicos do jogador durante uma partida */
20 var inimigos = []; /* Vetor responsável por guardar os objetos dos inimigos presentes na tela */
```

```
21 var sprites = {}; /* Objeto responsável por guardar objetos Image correspondentes aos sprites utilizados durante a partida */
22 var interface = {}; /* Objeto responsável por guardar objetos Image correspondentes aos botões da interface */
23 var background = {};  /* Objeto responsável por guardar objetos Image correspondentes às imagens de fundo no decorrer do jogo */
24 var music = {};  /* Objeto responsável por guardar referência às músicas de fundo */
25 var sounds = {};  /* Objeto responsável por guardar referência aos efeitos sonoros */
26
27 /*--------------------------------------------------------------------------
28  * Assim que a página web carregar, realiza a chamada do método que inicia o fluxo do jogo
29  */
30 window.addEventListener("load",function() {
31     /* Estabelece referência ao elemento Canvas no index */
32     jogoCanvas = document.getElementById("jogo");
33      /* Inicializa um objeto de desenho 2D para imprimir elementos dentro do Canvas */
34     ctx = jogoCanvas.getContext("2d");
35
36     /* Atribuí ao elemento Canvas no index um evento de espera para execução de uma função que captura a posição X e Y de um clique do mouse no Canvas */
37     jogoCanvas.addEventListener("mousedown", function(event) {
38         canvas_x = event.pageX - jogoCanvas.offsetLeft;
39         canvas_y = event.pageY - jogoCanvas.offsetTop;
40 /*pageX e pageY englobam toda a área visível do site no navegador. É preciso subtrair os excessos da área externa (offset) para se obter a real posição de um clique dentro do escopo do Canvas.*/
41
42         document.getElementById("click_X").innerHTML = canvas_x;
43         document.getElementById("click_y").innerHTML = canvas_y;
44     });
45
46     /* Atribuí ao elemento Canvas no index um evento de espera para execução de uma função que captura a posição X e Y de um toque de tela tátil no Canvas */
47     jogoCanvas.addEventListener("touchend", function(event) {
48         event.preventDefault();
49         /*primeiramente, cancelamos a execução padrão do método 'touchstart', que é o monitoramento contínuo de toques ou movimentos na tela tátil
50          Por conta deste comportamento, as informações da interação com uma tela tátil são armazenadas em um vetor. Assim, podemos recuperar a posição X e Y de um breve toque através da primeira posição deste vetor. */
51         canvas_x = event.targetTouches[0].pageX - jogoCanvas.offsetLeft;
52         canvas_y = event.targetTouches[0].pageY - jogoCanvas.offsetTop;
53         /*Novamente fazemos o tratamento para recuperar a posição de X e Y apenas do escopo da Canvas*/
54
55         document.getElementById("click_X").innerHTML = canvas_x;
56         document.getElementById("click_y").innerHTML = canvas_y;
57     });
58
```

```
59    /*
60     * Para saber quando que todos os recursos utilizados no jogo foram carregados
61     * está sendo utilizado uma variável chamada 'carregado' que controla quantos arquivos
já carregaram
62     * e uma função executada automaticamente por meio de um Interval que verifica se o
valor de 'carregado' já alcançou a quantidade de total de arquivos do jogo
63     */
64    //Imagens de Fundo
65    background["inicial"] = new Image();
66    background.inicial.onload = function() { carregado++; };
67    background.inicial.src = "recursos/interface/bg_inicial.png";
68    background["instrucoes"] = new Image();
69    background.instrucoes.onload = function() { carregado++; };
70    background.instrucoes.src = "recursos/interface/bg_instrucoes.png";
71    background["historia"] = new Image();
72    background.historia.onload = function() { carregado++; };
73    background.historia.src = "recursos/interface/bg_historia.png";
74    background["resultados"] = new Image();
75    background.resultados.onload = function() { carregado++; };
76    background.resultados.src = "recursos/interface/bg_resultados.png";
77    background["cenario"] = new Image();
78    background.cenario.onload = function() { carregado++; };
79    background.cenario.src = "recursos/imagens/background-game.png";
80    //Botões da Interface
81    interface["btn_jogar"] = new Image();
82    interface.btn_jogar.onload = function() { carregado++; };
83    interface.btn_jogar.src = "recursos/interface/btn_jogar.png";
84    interface["btn_voltar"] = new Image();
85    interface.btn_voltar.onload = function() { carregado++; };
86    interface.btn_voltar.src = "recursos/interface/btn_voltar.png";
87    interface["btn_instrucoes"] = new Image();
88    interface.btn_instrucoes.onload = function() { carregado++; };
89    interface.btn_instrucoes.src = "recursos/interface/btn_instrucoes_creditos.png";
90    interface["btn_novamente"] = new Image();
91    interface.btn_novamente.onload = function() { carregado++; };
92    interface.btn_novamente.src = "recursos/interface/btn_novamente.png";
93    //Sprites dos personagens
94    sprites["sprite_feiticeira"] = new Image();
95    sprites.sprite_feiticeira.onload = function() { carregado++; };
96    sprites.sprite_feiticeira.src = "recursos/imagens/feiticeira-spritesheet.png";
97    sprites["sprite_cogumelo"] = new Image();
98    sprites.sprite_cogumelo.onload = function() { carregado++; };
99    sprites.sprite_cogumelo.src = "recursos/imagens/cogumelo-spritesheet.png";
100    sprites["sprite_esqueleto"] = new Image();
101    sprites.sprite_esqueleto.onload = function() { carregado++; };
102    sprites.sprite_esqueleto.src = "recursos/imagens/esqueleto-spritesheet.png";
103    sprites["sprite_armadura"] = new Image();
104    sprites.sprite_armadura.onload = function() { carregado++; };
105    sprites.sprite_armadura.src = "recursos/imagens/armadura-spritesheet.png";
106    //Músicas e Efeitos Sonoros
```

```
107     music.music_tela_inicial = document.querySelector("#music_tela_inicial");
108     music.music_tela_inicial.load();
109     music.music_tela_inicial.addEventListener("canplaythrough", function() {
carregado++; }, false);
110     music.music_tela_jogo = document.querySelector("#music_tela_jogo");
111     music.music_tela_jogo.load();
112     music.music_tela_jogo.addEventListener("canplaythrough", function() { carregado++;
}, false);
113     music.music_tela_jogo.volume = 0.5;
114     music.music_tela_resultados = document.querySelector("#music_tela_resultados");
115     music.music_tela_resultados.load();
116     music.music_tela_resultados.addEventListener("canplaythrough", function() {
carregado++; }, false);
117     sounds.hit_jogador = document.querySelector("#sfx_hit_jogador");
118     sounds.hit_jogador.load();
119     sounds.hit_jogador.addEventListener("canplaythrough", function() { carregado++; },
false);
120     sounds.hit_inimigo = document.querySelector("#sfx_hit_inimigo");
121     sounds.hit_inimigo.load();
122     sounds.hit_inimigo.addEventListener("canplaythrough", function() { carregado++; },
false);
123     sounds.kill_jogador = document.querySelector("#sfx_kill_jogador");
124     sounds.kill_jogador.load();
125     sounds.kill_jogador.addEventListener("canplaythrough", function() { carregado++; },
false);
126     sounds.kill_inimigo = document.querySelector("#sfx_kill_inimigo");
127     sounds.kill_inimigo.load();
128     sounds.kill_inimigo.addEventListener("canplaythrough", function() { carregado++; },
false);
129
130     rotinaCarregarRecursos = setInterval(iniciarJogo, 500);
131 });
132
133 /* Função para iniciar o jogo quando todos os recursos tiverem sido carregados */
134 function iniciarJogo() {
135     if(carregado === 20) {
136         clearInterval(rotinaCarregarRecursos); //encerra a execução deste Interval
137         main();
138     }
139 }
140
141 /*----------------------------------------------------------------------------
142  * Função responsável pelo loop principal do jogo
143  */
144 function main() {
145     // Executa um método para limpar a tela do Canvas
146     ctx.clearRect(0, 0, ctx.canvas.width, ctx.canvas.height);
147     //Carrega uma função correspondente ao estado atual da aplicação
148     switch (tela) {
149         case 0:
```

```
150             telaInicial();
151             break;
152         case 1:
153             telaInstrucoes();
154             break;
155         case 2:
156             telaHistoria();
157             break;
158         case 3:
159             comecarJogo();
160             break;
161         case 4:
162             telaResultado();
163             break;
164         default:
165             break;
166     }
167 }
168
169 /* ---------------------------------------------------------------------
170  * Função responsável por carregar a tela inicial do jogo
171  */
172 function telaInicial() {
173
174     //Interrompe a execução da música de GAME OVER
175     music.music_tela_resultados.pause();
176     music.music_tela_resultados.currentTime = 0;
177
178     //Renderiza o fundo da tela
179     var padrãoFundo = ctx.createPattern( background.inicial, "repeat" );
180     ctx.fillStyle = padrãoFundo;
181     ctx.fillRect(0, 0, 640, 480);
182
183     //Renderiza os títulos da tela inicial
184     ctx.fillStyle = "#FFFFFF";
185     ctx.font = "16px Arial Black, Gadget, sans-serif";
186     ctx.textAlign = "center";
187     ctx.textBaseline = "top";
188     ctx.fillText("ESTUDO DE TÉCNICAS E PROCESSOS REFERENTES
AO",jogoCanvas.width/2,10);
189     ctx.fillText("DESENVOLVIMENTO DE JOGOS ELETRÔNICOS
MULTIPLATAFORMA",jogoCanvas.width/2,26);
190         /* Por padrão, o método fillText() não é capaz de realizar quebras de linha, ou seja, a
string é sempre exibida no Canvas em uma única linha. Ele também não reconhece comandos
especiais de strings, como o '/n' para quebra de linha */
191
192     ctx.font = "24px Arial Black, Gadget, sans-serif";
193     ctx.textAlign = "center";
194     ctx.fillText("DESAFIO DO LORDE DAS
TREVAS",jogoCanvas.width/2,jogoCanvas.height/2);
```

```
195
196     ctx.font = "14px Arial Black, Gadget, sans-serif";
197     ctx.textAlign = "center";
198     ctx.fillText("2015 Alisson Alberti Tres",jogoCanvas.width/2,460);
199
200     //Renderiza os botões da interface
201     ctx.drawImage(interface.btn_jogar, 40, 370);
202     ctx.drawImage(interface.btn_instrucoes, 345, 370);
203
204     //Estabelece a evento para quando um dos botões na tela inicial forem clicados
205     jogoCanvas.addEventListener("mousedown", menu_telaInicial);
206
207     //Toca música da tela inicial
208     music.music_tela_inicial.play();
209 }
210
211 function menu_telaInicial() {
212     if ((canvas_x >= 40 && canvas_x < 295) && (canvas_y >= 370 && canvas_y < 445))
{
213         // Ocorreu um clique/toque no botão de Iniciar Jogo
214         tela = 2;
215         jogoCanvas.removeEventListener("mousedown", menu_telaInicial);
216         main();
217     }
218     if ((canvas_x >= 345 && canvas_x < 600) && (canvas_y >= 370 && canvas_y <
445)) {
219         // Ocorreu um clique/toque no botão de Instruções/Créditos
220         tela = 1;
221         jogoCanvas.removeEventListener("mousedown", menu_telaInicial);
222         main();
223     }
224 }
225
226 /* ---------------------------------------------------------------------
227  * Função responsável por carregar a tela de instruções/créditos do jogo
228  */
229 function telaInstrucoes() {
230
231     //Renderiza o fundo da tela
232     var padraoFundo = ctx.createPattern( background.instrucoes, "repeat" );
233     ctx.fillStyle = padraoFundo;
234     ctx.fillRect(0, 0, 640, 480);
235
236     ctx.fillStyle = "#FFFFFF";
237     ctx.font = "18px Arial Black, Gadget, sans-serif";
238     ctx.textAlign = "center";
239     ctx.fillText("Como jogar:",jogoCanvas.width/2,25);
240     ctx.fillText("> Cliques com o botão esquerdo do mouse",jogoCanvas.width/2,50);
241     ctx.fillText("> Toques na tela tátil",jogoCanvas.width/2,75);
242
```

```
243     ctx.textAlign = "center";
244     ctx.fillText("Clique ou toque nos inimigos para ataca-los e causar-lhes
dano.",jogoCanvas.width/2,195);
245     ctx.fillText("Você ganhará pontos por cada inimigo
derrotado.",jogoCanvas.width/2,220);
246     ctx.fillText("Derrote o máximo possível de inimigos antes que os seus
pontos",jogoCanvas.width/2,246);
247     ctx.fillText("de vida acabem!",jogoCanvas.width/2,272);
248
249     //Renderiza o botão para voltar à tela inicial
250     ctx.drawImage(interface.btn_voltar, 345, 370);
251
252     //Estabelece a evento para quando o botão de voltar for clicado
253     jogoCanvas.addEventListener("mousedown", menu_telaInstrucoes);
254 }
255
256 function menu_telaInstrucoes() {
257     if ((canvas_x >= 345 && canvas_x < 600) && (canvas_y >= 370 && canvas_y <
445)) {
258         // Ocorreu um clique/toque no botão de voltar à Tela Inicial
259         tela = 0;
260         jogoCanvas.removeEventListener("mousedown", menu_telaInstrucoes);
261         main();
262     }
263 }
264
265 /* ---------------------------------------------------------------------
266  * Função responsável por carregar a tela em que é contada a história do jogo
267  */
268 function telaHistoria() {
269
270     //Interrompe a execução da música de fundo da tela anterior
271     music.music_tela_inicial.pause();
272     /*O controlador de áudio possui apenas dois estados: em execução(play) e pausado. É
preciso definir manualmente a posição da música para o início. Assim, quando a Tela Inicial
for carregada a música irá iniciar desde o começo.*/
273     music.music_tela_inicial.currentTime = 0;
274
275     //Renderiza o fundo da tela
276     var padraoFundo = ctx.createPattern( background.historia, "repeat" );
277     ctx.fillStyle = padraoFundo;
278     ctx.fillRect(0, 0, 640, 480);
279
280     ctx.fillStyle = "#FFFFFF";
281     ctx.font = "18px Arial Black, Gadget, sans-serif";
282     ctx.textAlign = "center";
283     ctx.fillText("Um reino distante vive tempos de crise, estando sob ameaça
do",jogoCanvas.width/2,15);
284     ctx.fillText("inescrupuloso e sádico Lorde das Trevas. Carregado de um grande
",jogoCanvas.width/2,35);
```

```
285     ctx.fillText("orgulho, o Lorde desafia o monarca com o propósito de colocar à
",jogoCanvas.width/2,55);
286     ctx.fillText("prova as habilidades dos súditos deste reino e a destreza de seu
",jogoCanvas.width/2,75);
287     ctx.fillText("exército de criaturas horrendas. Eis que o monarca do reino aceita o
",jogoCanvas.width/2,95);
288     ctx.fillText("desafio e atribui à sua mais talentosa Feiticeira a tarefa de derrotar
",jogoCanvas.width/2,115);
289     ctx.fillText("o maior número possível de elementos desta tropa
maligna.",jogoCanvas.width/2,135);
290
291     ctx.font = "18px Arial Black, Gadget, sans-serif";
292     ctx.textAlign = "center";
293     ctx.fillText("CLIQUE / TOQUE NA TELA PARA
COMEÇAR!",jogoCanvas.width/2,420);
294
295     //Estabelece a evento para quando o botão de voltar for clicado
296     jogoCanvas.addEventListener("mousedown", toqueNaTelaParaComecar);
297 }
298
299 function toqueNaTelaParaComecar() {
300     // Ocorreu um clique/toque na tela... VAMOS COMEÇAR A JOGAR!
301     tela = 3;
302     jogoCanvas.removeEventListener("mousedown", toqueNaTelaParaComecar);
303     main();
304 }
305
306 /* ---------------------------------------------------------------------
307  * Função responsável por carregar a tela de resultados
308  */
309 function telaResultado() {
310
311     //Interrompe a execução da música de fundo da tela anterior
312     music.music_tela_jogo.pause();
313     music.music_tela_jogo.currentTime = 0;
314     //Toca música de GAME OVER
315     music.music_tela_resultados.play();
316
317     //Renderiza o fundo da tela
318     var padraoFundo = ctx.createPattern( background.resultados, "repeat" );
319     ctx.fillStyle = padraoFundo;
320     ctx.fillRect(0, 0, 640, 480);
321
322     //Calcula a pontuação final do jogador na partida: pontos por inimigos derrotados
MULTIPLICADO pelo nível alcançado
323     var pontuacaoFinal = jogador.pontuacao * jogador.nivel;
324
325     ctx.fillStyle = "#FF4D4D";
326     ctx.font = "24px Arial Black, Gadget, sans-serif";
327     ctx.textAlign = "center";
```

```
328     ctx.fillText("GAME OVER!",jogoCanvas.width/2,jogoCanvas.height/2 - 60);
329
330     ctx.fillStyle = "#FFFFFF";
331     ctx.font = "20px Arial Black, Gadget, sans-serif";
332     ctx.textAlign = "center";
333     ctx.fillText("Pontos adquiridos: " + jogador.pontuacao,
334         jogoCanvas.width/2,jogoCanvas.height/2 - 20);
335     ctx.font = "20px Arial Black, Gadget, sans-serif";
336     ctx.textAlign = "center";
337     ctx.fillText("Nível alcançado: " + jogador.nivel,
338         jogoCanvas.width/2,jogoCanvas.height/2 + 20);
339
340     ctx.fillStyle = "#CFCF53";
341     ctx.font = "24px Arial Black, Gadget, sans-serif";
342     ctx.textAlign = "center";
343     ctx.fillText("Sua pontuação final: " + pontuacaoFinal,
344         jogoCanvas.width/2,jogoCanvas.height/2 + 55);
345
346     ctx.drawImage(interface.btn_novamente, 40, 370);
347
348     ctx.drawImage(interface.btn_voltar, 345, 370);
349
350     //Estabelece um evento para quando um dos botões for clicado
351     jogoCanvas.addEventListener("mousedown", menu_telaResultados);
352 }
353
354 function menu_telaResultados() {
355     if ((canvas_x >= 40 && canvas_x < 295) && (canvas_y >= 370 && canvas_y < 445))
{
356         // Ocorreu um clique/toque no botão de Jogar Novamente!
357         tela = 3;
358         jogoCanvas.removeEventListener("mousedown", menu_telaResultados);
359         main();
360     }
361     if ((canvas_x >= 345 && canvas_x < 600) && (canvas_y >= 370 && canvas_y <
445)) {
362         // Ocorreu um clique/toque no botão de voltar à Tela Inicial
363         tela = 0;
364         jogoCanvas.removeEventListener("mousedown", menu_telaResultados);
365         main();
366     }
367 }
368
369 /* ---------------------------------------------------------------------
370  * Função responsável por carregar a lógica de execução do jogo
371  */
372 function comecarJogo() {
373
374     //Interrompe a execução da música de GAME OVER
375     music.music_tela_resultados.pause();
```

```
376     music.music_tela_resultados.currentTime = 0;
377
378     //Instancia um Objeto para o jogador
379     jogador = {
380         vida       : 100,  // pontos de vida do jogador
381         pontuacao  : 0,    // contador de pontos do jogador
382         nivel      : 0,    // controla o nível em que o jogador se encontra no momento
383         imgSrc     : null, // objeto Image que contém os sprites da Feiticeira
384         animFrame  : 0,    // atributo auxiliar para o controle das animações da Feiticeira
385         timerJogo  : null  // guarda referência ao Interval que executa a lógica do jogo
386     };
387
388     //Instancia um modelo de Objeto para os inimigos a serem enfrentados
389     var inimigo = {
390         vida       : 0,    // pontos de vida deste tipo de Inimigo
391         forca      : 0,    // o quanto de dano este causa ao jogador
392         pontos     : 0,    // quantidade de pontos atribuída ao jogador quando este inimigo é
derrotado
393         frequencia : 0,    // determina o intervalo de tempo em milissegundos que este
inimigo realiza um ataque
394         imgSrc     : null, // objeto Image que contém os sprites deste Inimigo
395         animFrame  : 0,    // atributo auxiliar para o controle das animações deste Inimigo
396         posX       : 0,    // posição X deste inimigo na tela
397         posY       : 0,    // posição Y deste inimigo na tela
398         timerAtaque : null, // guarda referência ao Interval que realiza ataques ao jogador
399         ativo      : false // se diferente de false, indica que este inimigo ainda não foi
derrotado
400     };
401
402     // Vamos instanciar neste vetor três objetos para gerenciar os inimigos presentes em
cada nível
403     inimigos[0] = Object.create(inimigo);
404     inimigos[0].posX = 96;
405     inimigos[0].posY = 208;
406
407     inimigos[1] = Object.create(inimigo);
408     inimigos[1].posX = 160;
409     inimigos[1].posY = 144;
410
411     inimigos[2] = Object.create(inimigo);
412     inimigos[2].posX = 160;
413     inimigos[2].posY = 272;
414
415     // Atribuo à personagem da Feiticeira seus respectivos sprites
416     jogador.imgSrc = sprites.sprite_feiticeira;
417
418     //Inicio a música de fundo que toca durante a partida
419     music.music_tela_jogo.play();
420
421     //Inicia o fluxo de jogo: estabelece a execução contínua do método atualizar tela
```

```
422     jogador.timerJogo = setInterval(atualizarTela, 1000 / fps);
423 }
424
425 function novoNivel() {
426     // Incrementa o nível atual do jogador
427     jogador.nivel++;
428
429     // Gera um conjunto de inimigos para este nível
430     for (i = 0; i < 3; i++) {
431         //Estabelece um número aleatório entre 1 e 3 para determinar o tipo de inimigo
432         var inimigo_tipo = Math.floor(Math.random() * 3) + 1;
433         switch (inimigo_tipo) {
434             case 1:
435                 //Inimigo em questão será um 'Cogumelo Ambulante'
436                 inimigos[i].vida = 10;
437                 inimigos[i].forca = 3;
438                 inimigos[i].pontos = 100;
439                 inimigos[i].frequencia = 4000;
440                 inimigos[i].imgSrc = sprites.sprite_cogumelo;
441                 inimigos[i].ativo = true;
442                 break;
443             case 2:
444                 //Inimigo em questão será um 'Guerreiro Esqueleto'
445                 inimigos[i].vida = 15;
446                 inimigos[i].forca = 5;
447                 inimigos[i].pontos = 250;
448                 inimigos[i].frequencia = 6000;
449                 inimigos[i].imgSrc = sprites.sprite_esqueleto;
450                 inimigos[i].ativo = true;
451                 break;
452             case 3:
453                 //Inimigo em questão será uma 'Armadura Assombrada'
454                 inimigos[i].vida = 20;
455                 inimigos[i].forca = 4;
456                 inimigos[i].pontos = 500;
457                 inimigos[i].frequencia = 5000;
458                 inimigos[i].imgSrc = sprites.sprite_armadura;
459                 inimigos[i].ativo = true;
460                 break;
461         }
462
463     }
464
465     // Posiciona o nosso pelotão de inimigos para atacar a feiticeira em determinado
intervalo de tempo!
466     inimigos[0].timerAtaque = setInterval(function () {
467         jogador.vida = jogador.vida - inimigos[0].forca; // realiza um ataque ao jogador. A
quantia subtraída do jogador é determinada pelo atributo 'ataque' deste inimigo.
468         sounds.hit_jogador.currentTime = 0;
469         sounds.hit_jogador.play(); // executa o efeito sonoro de ataque ao jogador
```

```
470     }, inimigos[0].frequencia);
471     inimigos[1].timerAtaque = setInterval(function () {
472         jogador.vida = jogador.vida - inimigos[1].forca; // realiza um ataque ao jogador. A quantia subtraída do jogador é determinada pelo atributo 'ataque' deste inimigo.
473         sounds.hit_jogador.currentTime = 0;
474         sounds.hit_jogador.play(); // executa o efeito sonoro de ataque ao jogador
475     }, inimigos[1].frequencia);
476     inimigos[2].timerAtaque = setInterval(function () {
477         jogador.vida = jogador.vida - inimigos[2].forca; // realiza um ataque ao jogador. A quantia subtraída do jogador é determinada pelo atributo 'ataque' deste inimigo.
478         sounds.hit_jogador.currentTime = 0;
479         sounds.hit_jogador.play(); // executa o efeito sonoro de ataque ao jogador
480     }, inimigos[2].frequencia);
481 }
482
483 /*
484  * Realiza todos os cálculos e testes condicionais que determinam o fluxo do jogo, antes de renderizar a tela.
485  */
486 function atualizarTela() {
487     // Executa um método para limpar a tela do Canvas
488     ctx.clearRect(0, 0, ctx.canvas.width, ctx.canvas.height);
489     // Se os pontos de vida do jogador chegarem a zero, finaliza as rotinas de execução do jogo e carrega a tela de resultados
490     if(jogador.vida <= 0) {
491         clearInterval(jogador.timerJogo);
492         clearInterval(inimigos[0].timerAtaque);
493         clearInterval(inimigos[1].timerAtaque);
494         clearInterval(inimigos[2].timerAtaque);
495         tela = 4;
496         sounds.kill_jogador.currentTime = 0;
497         sounds.kill_jogador.play(); // executa o efeito sonoro que representa a derrota do jogador
498         main();
499     } else {
500         //Se o jogador ainda estiver ativo, atualiza o frame de animação
501         jogador.animFrame++;
502         //Para este protótipo, foi estipulado que a Feiticeira e os inimigos possuem animações com 4 sprites atualizados a cada dois frames. Totalizando assim um ciclo de animação com 8 frames.
503         if(jogador.animFrame > 3) {
504             //retorna para o frame inicial da animação
505             jogador.animFrame = 0;
506         }
507         //Se existe registro de um clique ou toque na tela, verifica se ele ocorreu em um dos inimigos na tela
508         //Antes disto, é necessário checar se os inimigos estão ativos...
509         for(i = 0; i < 3; i++) {
510             if(inimigos[i].ativo) {
511                 //Se tiver sido registrado um toque na tela
```

```
512             if(canvas_x !== null && canvas_y !== null) {
513                 if ((canvas_x >= inimigos[i].posX && canvas_x < inimigos[i].posX + 64)
&& (canvas_y >= inimigos[i].posY && canvas_y < inimigos[i].posY + 64)) {
514                     //Ocorreu um toque no Inimigo!
515                     inimigos[i].vida -= 1;
516                     sounds.hit_inimigo.currentTime = 0;
517                     sounds.hit_inimigo.play(); // executa o efeito sonoro de ataque ao inimigo
518                 }
519             }
520             //Se a vida do inimigo tiver chegado a zero...
521             if(inimigos[i].vida <= 0) {
522                 //encerra a execução da função de atacar o jogador
523                 clearInterval(inimigos[i].timerAtaque);
524                 //altera o seu estado para inativo
525                 inimigos[i].ativo = false;
526                 //incrementa a pontuação do jogador
527                 jogador.pontuacao += inimigos[i].pontos;
528                 // executa o efeito sonoro de inimigo derrotado!
529                 sounds.kill_inimigo.currentTime = 0;
530                 sounds.kill_inimigo.play();
531             //Se o inimigo ainda estiver ativo, atualiza o frame de animação do inimigo
532             } else {
533                 inimigos[i].animFrame++;
534                 if(inimigos[i].animFrame > 3) {
535                     inimigos[i].animFrame = 0;
536                 }
537             }
538         }
539     }
540     //Limpa variáveis de captura de tela
541     canvas_x = null;
542     canvas_y = null;
543     document.getElementById("click_X").innerHTML = "";
544     document.getElementById("click_y").innerHTML = "";
545 
546     //Se todos os inimigos estiverem inativos, significa que o nível foi concluído e o
jogo deve gerar o próximo nível
547     if(!inimigos[0].ativo && !inimigos[1].ativo && !inimigos[2].ativo) {
548         novoNivel();
549     }
550 
551     //Após todos os cálculos e testes serem feitos, Executa a renderização dos elementos
gráficos na tela...
552     renderizarTelaJogo();
553 }
554 }
555 
556 /*
557  * Função dedicada a gerar na tela todos os elementos gráficos presentes durante a
execução de uma partida
```

```
558  */
559 function renderizarTelaJogo() {
560     //Renderiza o cenário da tela
561     ctx.drawImage(background.cenario, 0, 0);
562 
563     //Renderiza os pontos de vida do jogador
564     ctx.fillStyle = "#FF3030";
565     ctx.font = "16px Arial, Helvetica, sans-serif";
566     ctx.textAlign = "left";
567     ctx.textBaseline = "top";
568     ctx.fillText("Pontos de Vida: " + jogador.vida,20,15);
569 
570     //Renderiza a pontuação do jogador
571     ctx.fillStyle = "#FFFFFF";
572     ctx.font = "16px Arial, Helvetica, sans-serif";
573     ctx.textAlign = "center";
574     ctx.textBaseline = "top";
575     ctx.fillText("Pontuação: " + jogador.pontuacao,jogoCanvas.width/2,15);
576 
577     //Renderiza o nível do jogador
578     ctx.font = "16px Arial, Helvetica, sans-serif";
579     ctx.textAlign = "right";
580     ctx.textBaseline = "top";
581     ctx.fillText("Nível: " + jogador.nivel,600,15);
582 
583     //Renderiza a Feiticeira
584     ctx.drawImage(jogador.imgSrc, 64 * jogador.animFrame, 0, 64, 64, 416, 208, 64, 64);
585 
586     //Se o Inimigo 1 estiver ativo, renderiza ele
587     if(inimigos[0].ativo) {
588         ctx.drawImage(inimigos[0].imgSrc, 64 * inimigos[0].animFrame, 0, 64,
64,inimigos[0].posX, inimigos[0].posY, 64, 64);
589     }
590 
591     //Se o Inimigo 2 estiver ativo, renderiza ele
592     if(inimigos[1].ativo) {
593         ctx.drawImage(inimigos[1].imgSrc, 64 * inimigos[1].animFrame, 0, 64,
64,inimigos[1].posX, inimigos[1].posY, 64, 64);
594     }
595 
596     //Se o Inimigo 3 estiver ativo, renderiza ele
597     if(inimigos[2].ativo) {
598         ctx.drawImage(inimigos[2].imgSrc, 64 * inimigos[2].animFrame, 0, 64,
64,inimigos[2].posX, inimigos[2].posY, 64, 64);
599     }
600 
601 }
602 
603 /* ---------------------------------------------------------------------
604  Fim do arquivo. A WINNER IS YOU! */
```