Universidade Estadual Paulista "Júlio de Mesquita Filho" – UNESP

Caio Vinícius Maldonado Faustino

Framework para jogos em HTML5

Bauru

2015

Universidade Estadual Paulista "Júlio de Mesquita Filho" – UNESP

Caio Vinícius Maldonado Faustino

Framework para jogos em HTML5

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação da Faculdade de Ciências da Universidade Estadual Paulista "Júlio de Mesquita Filho" – UNESP, Campus de Bauru.
Orientador: Dr. Wilson Massashiro Yonezawa

Bauru

2015

Dedico este trabalho aos meus queridos pais, Carlos e Isabel, que me apoiaram indubitavelmente por toda a minha jornada, me dando força, incentivo e imenso amor.

AGRADECIMENTOS

É por fim chegada a hora da despedida, mas que esta despedida não venha carregada do fardo do adeus e da saudade, mas sim da alegria e da gratidão que devem acompanhar todas as coisas boas da nossa vida, e esta jornada com certeza foi uma delas.

Nestes anos em que realizei o curso de Ciência da Computação, nesta grande faculdade e esta linda cidade que me acolheu, não poderia deixar de ser grato a tudo que me aconteceu. Sei que aos olhos da maioria não fui um aluno exemplar, e acabei prolongando minha estadia, mas acredito que tudo foi parte do aprendizado e do amadurecimento que deve ocorrer em cada um de nós.

Gostaria de agradecer acima de tudo aos meus pais, que sempre foram o fundamento de todo meu caráter e o apoio inquestionável durante toda a minha vida, me incentivando e encorajando mesmo quando eles próprios não se sentiam seguros.

Gostaria de agradecer aos meus professores, que tentaram sempre, dentro de suas crenças e suas limitações, nos tornar alunos melhores, não só por estudo acadêmico, mas também mostrando que a vida é feita de lutas e que devemos sempre buscar superar nossas dificuldades, pois todos são capazes e únicos.

E principalmente agradecer a todos os meus amigos, e todas as pessoas que conheci durante esses anos. Tenho certeza de que muitos levarei comigo para o resto da vida, e esse é um tesouro tão grande ou até maior do que todo conhecimento que adquiri na academia. Essas pessoas me ajudaram a crescer como indivíduo e a reconhecer a diversidade de culturas e personalidades da maneira mais rica e divertida possível.

Obrigado a todos por esses incríveis anos de minha vida.

Resumo

A relevância dos jogos na nossa sociedade, vem se tornando cada vez mais clara. À medida que novos estudos são realizados, novas aplicações surgem para tornar nosso cotidiano mais divertido e interessante, e as vantagens de seu uso podem ser vistas em áreas que antes não seriam consideradas.

Este estudo busca introduzir alguns aspectos da criação de jogos eletrônicos, visando a aplicação de conceitos já existentes em novas tecnologias. Com o resultado deste trabalho, a tarefa de criação de jogos pode parecer menos desafiadora, e isso possibilita que entusiastas e estudantes possam melhor compreender o campo estudado. Com um maior número de pessoas buscando criar jogos, há a tendência de elas irão se socializar e participar das comunidades existente em torno do desenvolvimento de jogos, compartilhando experiências e aprendendo uns com os outros, e isso resulta em melhores profissionais e melhores jogos para todos.

Palavras chaves: Jogos, *framework*, *web*, *HTML5*

Abstract

The relevance of games in our society has become increasingly clear in recent times. As new studies are made, new applications arise to make our daily lives more fun and interesting, and the advantages of its use can be seen in areas that would not be previously considered.

This study aims to introduce some aspects of the creation of electronic games, and looks to apply existing concepts using new technologies. With the result of this work, the task of creating games may seem less challenging, and that would allow for students and enthusiasts to better comprehend the field in which they study. With more people looking forward to making games, they tend to socialize and participate in the existing communities that revolve around game development, sharing experiences and learning from each other, which can only result in better professionals and better games for everyone.

Keywords: games, framework, web, HTML5

Lista de Figuras

Sumário

1 Introdução e contexto do trabalho

É cada vez mais clara a crescente valorização e utilização de jogos em diversas atividades em nossa sociedade. A grandiosidade do mercado de jogos de entretenimento já é reconhecida há algum tempo. A empresa de pesquisa *DFC Intelligence* (Forbes) estima que, o mercado mundial deverá chegar ao valor de 70,1 bilhões de dólares anuais até o ano de 2015. Mas, além disso, começa-se a ser notada também a utilização de jogos em áreas como da Educação, Medicina, Propaganda, e muitas outras.

Na maioria dos casos um jogo simples pode ser aplicado como complemento a outras atividades mais complexas. Em contextos como o de uma sala de aula ou um tratamento psicológico, os jogos podem ser utilizados para apresentar questões e analisar resultados de forma lúdica e descontraída. Porém, o processo de criação é muito complexo, envolvendo diversas áreas do conhecimento, o que costuma tornar o seu desenvolvimento inviável em um cenário onde a meta principal não seja o retorno financeiro.

Felizmente nos dias de hoje existe toda uma gama de ferramentas de auxílio que vem se expandindo, beneficiando o processo e tornado-o mais eficaz e rápido. Muitos dos desafios encontrados são comuns à criação de jogos eletrônicos e portanto, soluções conhecidas podem ser utilizadas para resolvê-los. Baseado nesta ideia, este trabalho propõe uma ferramenta para auxiliar o desenvolvimento do código inserido na criação de um jogo eletrônico para a plataforma *Web*.

1.1 Objetivo

1.1.1 Objetivo Geral

A criação de uma estrutura de código denominada *Framework*, servindo de base de código em projetos de jogos eletrônicos para a plataforma *Web*.

1.1.2 Objetivos Específicos

- Estudar as tecnologias de desenvolvimento para Web
- Fomentar o estudo e a criação de Jogos
- Desenvolver uma ferramenta para auxiliar outros desenvolvedores
- Criar um jogo simples como exemplo e prova de conceito da ferramenta.
- Incentivar o uso de novas tecnologias e múltiplos dispositivos

1.2 Justificativa

A estrutura criada permitirá que outros possam se beneficiar de um fluxo de controle já definido, podendo também ser estendido de acordo com as particularidades de cada jogo, incluindo métodos comumente utilizados, reduzindo o esforço e o tempo de desenvolvimento.

Com a redução no tempo, e consequentemente no custo, espera-se incentivar a criação de jogos, tanto recreativos, como os chamados jogos sérios, os quais buscam elaborar conceitos externos ao mesmo, com um objetivo além da diversão.

2 Conceitos e referências

Define-se o conceito de jogo, situando e dando base ao restante do estudo. Com um estudo sobre a história e a evolução da mídia e algumas definições sobre os tipos de jogos existentes, é possível ter uma visão mais imersa no contexto do desenvolvimento proposto.

2.1 Conceito de jogo e jogo digital

Não é comum haver necessidade de explicar a outra pessoa o que é um jogo. De certa forma todos já tiveram contato com algum tipo de jogo e possuem uma concepção estabelecida do que o mesmo significa. Mas, um estudo acadêmico possui o objetivo de definir e estudar tal conceito e ao buscar-se uma definição para o termo percebe-se que a interpretação individual pode ter suas variações.

Apesar de todo o estudo já realizado em relação aos jogos, não existe ainda uma definição formal e única que represente o que é um jogo em sua totalidade. *Jesse Schell* em seu livro “A Arte de *Game Design*, um Livro de Lentes” (2011) dedica um capítulo inteiro a este assunto, buscando combinar as definições de diferentes estudiosos e entender a sua essência.

Um dos processos de criação de um jogo chama-se *Game Design*. É onde se definem os objetivos, as regras, e como a junção dos elementos fará com que o jogo proporcione uma experiência única e divertida. Neste momento em particular é que surgem as discussões e as terminologias que buscam definir aquele jogo, e apesar de normalmente não haverem nomes para explicar um conceito, é muito importante que ele seja passado de forma clara, para que todos os envolvidos no projeto

estejam alinhados com o pensamento criativo do *Game Designer*. E isso sempre foi feito de maneira informal, sem regras e sem dicionário.

Mas buscar uma definição faz com que se pense sobre aquilo que está sendo abordado, e, portanto deve-se esclarecer algumas das possíveis definições de jogo. Uma frase simples onde se é pensada e afirmada por todos é "Jogo é algo que se joga". Isso já se faz algo distinto, pois são apenas os jogos que são jogados. Um brinquedo não pode ser considerado um jogo, já que com ele apenas se brinca, e da mesma forma um jogo não é considerado um simples brinquedo.

Brincar é algo que fazemos em busca de diversão, podemos brincar com um brinquedo ou com um amigo, não precisamos de um jogo para nos divertir. Já diversão é algo que envolve prazer, mas dificilmente poderia ser definida concretamente devido à natureza dos sentimentos humanos. Então, devemos voltar ao mérito do jogo, já que um jogo pode ser divertido, mas não o é, necessariamente.

Friedrich Schiller define que "Jogar é a despesa sem rumo de energia exuberante", ou seja, uma forma de libertarmos um excesso de energia existente em nós mesmos. Mas essa ideia de jogar sem um objetivo ou simplesmente "sem rumo" não se encaixa em nosso conceito em que os mesmos possuem objetivos e metas a serem concluídas.

Katie Salen e *Eric Zimmerman* (2012) definem que "Jogar é o movimento livre em uma estrutura mais rígida", porém essa frase torna-se demasiadamente ampla e engloba exemplos os quais podem ser claramente vistos como não sendo um jogo.

Em todas as definições abordadas, podemos inferir exemplos os quais a tornam falsa por ou incluírem casos que não são jogos, ou por deixarem de inserir outros que sejam. Mas, de certa forma, conseguimos encontrar características que, apesar de não serem aplicadas em todos os jogos, estão presentes em grande número em qualquer jogo. Consideremos que um jogo é algo:

- Que se participa por vontade própria;
- Que possui um objetivo;
- Que apresenta conflito;
- Que possui regras;
- Em que se pode ganhar ou perder;
- É interativo;
- Possui um desafio;
- Pode criar o seu próprio valor interno;

- Que engaja o jogador;
- Que apresenta um sistema fechado e formal.

Mas mesmo com todas as características citadas acima, ainda há a sensação de que deve-se existir algo além do que esteja faltando. Fazendo uma analogia com o mundo da programação, se uma função tem muitos parâmetros, provavelmente ainda lhe faltam alguns, o que significa que na verdade a sua estrutura e a forma de abordar o problema provavelmente não estão sendo tão eficientes.

Não existe portanto uma única definição do que é um jogo, da mesma forma que também não existem regras ou instruções definidas sobre como criar um jogo. O que existe hoje é o conhecimento e a experiência acumulados dos jogos e estudos que já foram feitos, e portanto estudar essa história nos permite entender um pouco melhor os elementos que compõem o desenvolvimento de um jogo.

2.2 História dos jogos

Ao estudar a evolução dos jogos através do tempo, é possível compreender os objetivos e as limitações de cada época, os instrumentos e as técnicas utilizados, e a motivação para criar novos métodos e novas formas de interação com o jogador.

O historiador holandês *Johan Huizinga*, em seu livro "*Homo Ludens"* (1938), argumenta que os jogos são uma condição primária para a criação da cultura humana. Ele vê o ato de jogar como ponto de partida para atividades humanas mais complexas, como a linguagem, as leis, a filosofia e a arte, e é a partir dessa premissa que voltamos às origens da humanidade para iniciar este breve estudo.

Indícios arqueológicos apontam que os primeiros instrumentos utilizados com a finalidade de entretenimento foram feitos a partir de ossos como o *Tálus*, constituindo o que podem ser considerados os primeiros jogos de Dados ou de Bugalha. Esses instrumentos também eram utilizados para fins divinatórios e de oráculo, formando uma relação entre o jogo e a crença de cada povo.

No Egito, em tumbas anteriores à primeira dinastia as quais datam algo próximo de 3500 a.C., encontrou-se o primeiro jogo de tabuleiro conhecido, o *Senet* conforme Figura 1. Com o tempo o jogo foi tomando uma erudição que refletia a religião egípcia da época, onde as peças representavam a alma e sua viagem pela vida após a morte.

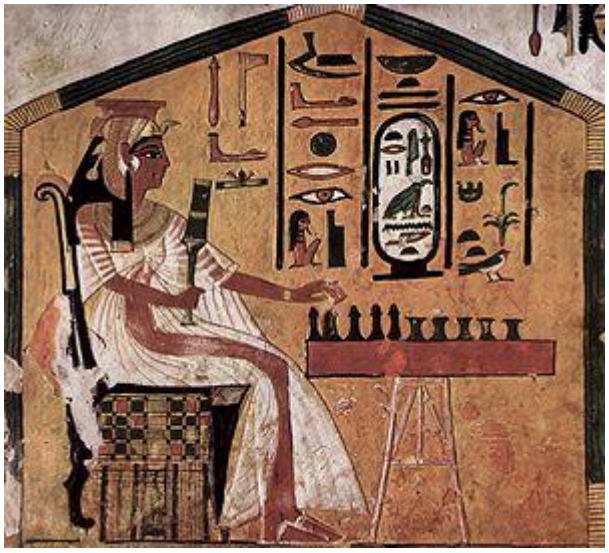

Figura 1 – pintura com a representação do jogo *Senet*.

Outros jogos de tabuleiro também podem ser encontrados em sites arqueológicos em grande parte das civilizações ao redor do mundo. Por exemplo, o *Ludus duodecim scriptorum*, ou "Jogo das Doze Marcas", no Império Romano, o que daria origem ao Tabula, e posteriormente o Gamão ou o Liubo, que foi muito popular na China durante a Dinastia *Han*, mas tempos mais tarde foi superado em popularidade pelo *Go*.

Com inúmeros outros exemplos pelo mundo, não apenas os jogos de tabuleiro, como também os dados e as cartas, são amplamente conhecidos e utilizados como instrumentos em diversos jogos, e novas formas de jogar são inventados até hoje.

2.2.1 Primeiros Eletrônicos

Buscando dar foco ao desenvolvimento tecnológico, sendo mais relevante a este estudo, serão analisados de maneira mais aprofundada, o surgimento e a evolução dos jogos em formato eletrônico.

Dependendo da definição e dos critérios utilizados, existem alguns candidatos ao título de primeiro jogo eletrônico, mas os exemplos mais notáveis foram criados com o objetivo de ilustrar as evoluções na teoria de Inteligência Artificial. O próprio *Alan Turing*, um dos pais da computação, escreveu em 1947 um algoritmo teórico para jogar Xadrez, o qual foi posteriormente desenvolvido por *Dietrich Prinz* em um dos primeiros computadores inventados, o *Mark I*.

Podemos, contudo, relacionar o início do real desenvolvimento de jogos com o lançamento dos computadores TX-0 e PDP-1 e a criação de uma cultura "*Hacker*" no MIT (*Massachusetts Institute of Technology*). Essas novas máquinas, mais compactas e potentes, expandiram as possibilidades e a criatividade dos alunos que se encontraram incentivados e motivados a explorar e modificar os dispositivos que lhes eram apresentados.

Encontramos assim, uma pequena coleção de jogos desenvolvidos pelos alunos, tanto no MIT como em algumas outras universidades. Alguns com o intuito de simular jogos, como jogo da velha (OXO), damas e *Nim*, o primeiro com *hardware* desenvolvido exclusivamente para essa finalidade (*Nimrod*). Mas todos eles eram restritos ao uso dos enormes computadores nos centros acadêmicos.

Apesar de todo o desenvolvimento computacional, um dos jogos com maior reconhecimento ao título de primeiro jogo eletrônico foi o "*Tennis for Two*". Criado pelo físico americano *William Higinbotham* em 1958, tinha como o objetivo principal o entretenimento dos visitantes do Laboratório Nacional de *Brookheaven*, e funcionava utilizando um computador analógico *Donner Model* 30 para calcular uma trajetória balística, que simulava uma bola de tênis sendo rebatida. Sua popularidade se deve ao fato de ter sido um dos primeiros jogos com uma representação gráfica, já que utilizava um osciloscópio como tela, e também por ter sido reconhecido alguns anos depois como avô dos vídeo games.

Outro ícone na história dos jogos é o "*Spacewar!",* considerado o primeiro jogo de tiro. Foi desenvolvido também no MIT pelos estudantes *Martin Graetz*, *Steve Russell* e *Wayne Wiitanen* em 1962. Nele duas naves com uma quantidade limitada de tiros e combustível devem destruir seu oponente em um espaço gravitacional em um campo estrelado. Sua popularidade é atribuída ao fato de ter sido feito para o PDP-1 e ter se espalhado pelas universidades que possuíam esse modelo, já que era utilizado pela empresa como teste operacional do computador e já o trazia salvo em sua memória principal. Além de sua relevância histórica, "*Spacewar!"* serviu de inspiração para que outros estudantes desenvolvessem novos jogos e formas de interação com a máquina, fomentando o início de uma indústria.

Figura 2 – fotografia *Dan Edwards* (a esquerda) e *Peter Samson* jogando

Uma versão emulada de "*Spacewar!"* pode ser encontrada no site "*Mass: werk*" e o código fonte pode ser visto no repositório *Github.* As 7522 linhas de código baixo nível, ilustram a dificuldade e a maneira como os primeiros jogos precisavam ser produzidos, totalmente dependentes do *hardware* em que iriam funcionar, era possível saber exatamente qual a capacidade computacional e a complexidade que o jogo poderia ter.

2.2.2 Comercialização

Não demorou muito para que se percebesse o potencial consumidor que havia em um jogo eletrônico interativo. Mas sem um modelo de negócios ou mercado estabelecido, alguns pioneiros se aventuraram em desenvolver jogos embarcados em um hardware próprio, praticamente a única alternativa devido às limitações tecnológicas e ao modelo de consumo da época.

Uma máquina com poder computacional suficiente para executar um jogo mais complexo e visual como o *Spacewar!* era também dona de um preço inacessível para a população. Mesmo após o lançamento de novos modelos de microcomputador como o PDP-11, era inimaginável um computador sendo utilizado na casa das pessoas. Entretanto, o mercado de máquinas no estilo *Arcade*, operadas por moedas, estava em ascensão com jogos eletromecânicos como o

Pinball, e esse era um mercado que tinha condições de investir em equipamentos um pouco mais caros se o retorno a longo prazo fosse possível.

Com esta ideia, *Nolan Bushnell* e *Ted Dabney* criaram em 1971 o *Computer Space*, uma réplica do *Spacewar!,* onde o jogador combatia duas naves controladas pelo computador. Mas percebendo que o uso de microcomputadores ainda era proibitivamente caro, eles conseguiram desenvolver o jogo utilizando uma máquina de estados feita com arquitetura TTL (Lógica Transistor-Transistor) para controlar o tubo CRT, e levaram a ideia até a *Nutting Associates*, uma empresa de *Arcade* da Califórnia.

Computer Space não alcançou o sucesso de vendas esperado, apesar de conseguir vender ao todo por volta de 1500 máquinas. Seu impacto negativo no mercado foi atribuído ao fato dele ter uma curva de aprendizado complexa e não atrair o público mediano.

2.2.3 Consoles

Enquanto isso em *Nashua* (*New Hampshire*, Estados Unidos), *Ralph H. Baer*, Engenheiro nascido na Alemanha, era apaixonado pela ideia de criar um jogo que pudesse ser jogado em uma TV. Enquanto trabalhava na *Sander Associates*, uma empresa de contrato militar, *Baer* viu uma oportunidade de negócio e entrou com um pedido de verba para seu supervisor, o qual aprovou a mesma e viabilizou o desenvolvimento do protótipo da *Brown Box* em 1967.

A caixa continha alguns jogos simples como *ping-pong* e *handball*, desenvolvidos utilizando tecnologia DTL (Lógica DiodoTransistor). Dois controles analógicos e nas versões posteriores uma arma de luz. Foi a primeira patente registrada de um vídeo game, pois havia sido desenvolvido já com o intuído de comercialização.

Baer buscou diversos fabricantes de televisores os quais estivessem interessados em vender o sistema como um adicional, e após grande resistência conseguiu um contrato com a *Magnavox*. O modelo ganhou um novo visual, e no lugar da sequência de chaves onde se escolhiam o jogo, foi utilizada uma entrada de cartucho, que, diferentemente dos futuros consoles, não continha informações sobre o jogo, e sim uma série que *jumpers* os quais informavam o sistema qual era o jogo escolhido.

E assim em 1972 nasceu o primeiro vídeo game doméstico, o *Magnavox Odissey*, ilustrado na Figura 3. Vendendo cerca de cem mil unidades no ano do lançamento, e por volta de trezentas e cinquenta mil até 1974, quando começaram a surgir os novos modelos. Além disso, a *Sander Associates* ganhou ao longo de vinte anos, mais de cem milhões de dólares em ações judiciais de patente contra outras empresas que se inspiraram no *Magnavox Odissey*, como o *Pong da Atari.*

Figura 3 – fotografia do console *Magnavox Odissey*

2.2.4 Arcade

Após o fracasso comercial do *Computer Space*, *Nolan Bushnell* e *Ted Dabney* decidiram iniciar sua própria empresa e fundaram a *Atari* em 1972. Buscando por novas ideias e uma forma de negócio mais sólida, contrataram o engenheiro recém formado *Al Alcorn*, para iniciar o desenvolvimento utilizando um conceito mais simples. Os donos na *Atari* solicitaram que ele criasse uma versão do jogo de tênis que *Nolan* havia visto em uma demonstração do *Magnavox Odissey* em Maio daquele ano.

Assim, o que era pra ser um protótipo, um exercício de conceito para *Alcorn* e seu conhecimento em lógica TTL, acabara se tornando o primeiro jogo da empresa, *Pong*, Figura 4. Depois de pronta, a máquina de *Arcade* ficou com um acabamento surpreendente, que *Nolan Bushnell* e *Ted Dabney* decidiram realizar testes de sua receptividade. Colocada em um bar local chamado *Andy Capp's Tavern*, o jogo obteve boa aceitação do público. Alguns dias após sua instalação, a máquina começou a apresentar defeitos, e *Alcorn* averiguando, chegou à conclusão de que o mecanismo de coleta de moedas não estava funcionando perfeitamente já que estava completamente cheio.

Figura 4 – fotografia do *arcade* com o jogo *Pong*

Nolan tinha em mente licenciar a máquina para empresas de *Arcade* em *Chicago*, mas acreditou que poderia lucrar com fabricação própria. Conseguiu investimento no banco e iniciou a linha de montagem, que após um começo lento foi se aprimorando. *Pong* ao longo de sua existência vendeu por volta de 19 mil máquinas, fazendo com que outras empresas começassem a buscar também sua fatia no mercado.

2.2.5 Uma nova Indústria

Após o sucesso de *Pong* e seguindo a linha do *Odissey*, a *Atari* decidiu expandir suas atividades e criar também uma versão caseira do *Arcade*. Desenvolvido em 1974 com o intuito de encapsular a lógica do jogo em um único chip de circuito integrado, o projeto com codinome *Darlene* foi liderado por *Harold*

Lee, que permitiu que o produto obtivesse algumas vantagens tecnológicas, mas mantendo seu preço competitivo.

Tentaram contato com varejistas que tivessem interesse em vender o novo console, mas não obtiveram respostas positivas até conversar com o setor esportivo da *Sears*, uma grande loja de departamento na época, que já vendia o *Odissey*. Com um pedido de cento e cinquenta mil unidades, a *Atari* foi obrigada a abrir uma nova fábrica para atender a demanda e em seguida lançou as primeiras unidades com o nome o qual ficou conhecido também aqui no Brasil, Tele Jogo, mas que mais tarde voltou a ter a marca *Pong*.

Apenas alguns meses após seu lançamento, vários clones já apareciam no mercado, e sem uma patente do produto e nem um poder de montagem e distribuição que garantissem sua predominância, a *Atari* se viu obrigada a se estabelecer pela inovação. Enquanto a *Magnavox* começava uma ação judicial contra ela e outras empresas com clones do *Odissey*, que tinha sim uma patente datando o início do desenvolvimento em 1966.

Entretanto, no mercado dos "*coin-ops*" (*Arcades*, operados por moeda), durante os anos seguintes houve uma gradual tendência de substituir as máquinas eletro-mecânicas pelas eletrônicas. Alguns jogos foram lançados, inclusive pela própria *Atari*, como o *Breakout*, desenvolvido em 1976 por *Steve Wozniac* e *Steve Jobs*, futuros fundadores da *Apple*. Mas a popularidade do gênero só ficou evidente com o primeiro, e talvez o maior, sucesso de todos, *Space Invaders*. Inventado no Japão por *Tomohiro Nishikado* e distribuído inicialmente pela *Taito*, chegou aos Estados Unidos no mesmo ano de 1978 pela *Midway* e vendeu por volta de 300 mil máquinas no Japão e 60 mil nos Estados Unidos.

O sucesso de Space Invaders fez com que se iniciasse a chamada era de ouro dos Arcades. Após ele, outros grandes sucessos como *Pac Man* e *Donkey Kong* fizeram com que o mercado chegasse a um faturamento anual em moedas em torno de 8 bilhões de dólares em 1982.

O mercado de *Consoles* vinha também crescendo e aproveitando a popularidade dos jogos, gerando em torno de 3,8 bilhões de dólares no mesmo ano, muito próximo aos 4 bilhões gerados pela indústria de cinema por exemplo.

2.2.6 Crise e Ressurreição

Ao final de 1983, diversos fatores fizeram com que o mercado de jogos entrasse em crise na América. A grande demanda fez com que as empresas produzissem de maneira desleixada e acabaram saturando o mercado com jogos que as pessoas não queriam mais comprar. Isso levou à falência de inúmeras empresas do setor e muitos disseram que seria o fim da indústria.

Foi então que em 1985 veio o lançamento do *Famicon* pela *Nintendo*, ou como é conhecido na maior parte do mundo, o *NES* (*Nintendo Entertainment System*). Seu sistema de 8 bits era vendido com dois controles e até mesmo uma arma de luz, semelhante ao utilizado no *Odissey*.

Seu grande sucesso fez o mercado de *Consoles* crescer novamente, e estabeleceu alguns padrões de jogabilidade que são utilizados até hoje.

Neste cenário houveram diversas melhorias, novos sistemas com 16 *bits* a partir da década de 1990, e dês de então vemos grandes avanços no setor que podem ser facilmente acompanhados no mercado atual.

2.2.7 Jogos de PC

Mesmo com os lançamentos comerciais a partir da década de 70, os jogos para computador continuaram a ser desenvolvidos. Muitos jogos ainda eram criados nos mainframes das universidades, com experimentos interessantes e inovações tecnológicas. Jogos como o *Maze War* e o *Spasim* foram os percussores dos jogos de tiro em primeira pessoa, e outros como o *MUD* (*Multi-User Dungeon*) traziam o elemento de jogos em RPG para o mundo virtual.

Com o inicio do uso dos modems para acesso a redes de computadores nos anos 80, iniciou-se também o desenvolvimento de jogos *multiplayer*, ou seja, para vários jogadores. Essas novidades funcionavam independentes do mercado de jogos mais comum, mas com a popularização do uso de computadores pessoais como o *IBM PC* e o *Machintosh*, abria-se espaço para um pequeno mercado de jogos de computador.

Esse mercado se expandiu e se popularizou de forma mais evidente na década de 90 com o lançamento do jogos de tiro como *Doom* e *Quake*, que serão explicados mais a frente quando falarmos sobre a criação das *Engines*.

2.2.8 Portáteis

Um pequeno mercado de jogos portáteis começava em 1979, quando a *Milton Bradley Company* lançou o *Microvision*, um pequeno dispositivo com uma tela de *LCD*. Entretanto esse mercado só ganhou força quando a *Nintendo* resolveu apostar em jogos portáteis e lançou o *Game Boy* em 1989. Um dispositivo capaz de rodar diferentes jogos, semelhante a um *Console*, com pequenos cartuchos inseridos na parte traseira do aparelho. Dês de então várias melhorias foram feitas, e alguns concorrentes como o *PSP* (*Playstation Portátil*) surgiram no mercado.

Mas foi só recentemente, com a massiva popularização dos *Smartphones*, que os jogos portáteis cresceram de forma espetacular. Grande parte da população possui um dispositivo capaz de rodar jogos, e com uma tela sensível ao toque, às possibilidades de criação são enormes.

2.2.9 Web

Os jogos para *Web* sofrem diversas limitações e não era um mercado muito representativo.

Apesar de alguns exemplos notáveis ao longo da história como o jogo *Runescape*, que utilizava um *applet* em Java para funcionar no *browser*, o grande responsável pela popularização dos jogos online foi o *Flash*.

Em sites dedicados como o *Armor Games* e o *Kongregate*, inúmeros desenvolvedores independentes tentavam inovar ao criar simples jogos e animações em *Flash*, uma plataforma que permitia unificar o desenvolvimento em meio à guerra dos *Browsers* utilizando um *plugin* externo.

Com o surgimento das redes sociais como *Orkut* e *Facebook*, esses jogos tomaram uma proporção real de mercado utilizando a "viralidade" da rede para conseguir um enorme número de jogadores que poderia pagar micro-transações dentro do jogo.

2.3 Tipos de jogos

Como pôde ser visto ao longo da história, sempre houve uma busca por inovação que fez com que surgissem jogos de diferentes tipos. Quando o mercado de jogos eletrônicos se mostrou lucrativo e em crescimento constante, a

competitividade fez com que os produtores buscassem jogos que pudessem se diferenciar e atrair um público maior.

Com a ideia inicial de que as crianças eram o público alvo deste mercado, havia o consenso de que o poder aquisitivo vinha dos pais, e que eles, zelando pelo bem estar de seus filhos, seriam seletivos ao escolher apenas um "*video game*" para comprar. Sendo assim o foco da propaganda era convencer os pais de que determinado dispositivo representavam um melhor investimento. Um dos principais atrativos para isso eram os avanços tecnológicos apresentados.

Até os dias atuais as grandes produções buscam expandir os limites tecnológicos para trazer uma nova experiência ao usuário, e com isso alguns termos técnicos são utilizados para designar aspectos de um jogo. Assim temos uma das formas de classificar um jogo.

2D - Um jogo em que os gráficos são desenhados em duas dimensões, ou seja, computacionalmente é como se estivéssemos sobrepondo figuras em um plano.

3D - Um jogo em que os gráficos são calculados de maneira tridimensional, ou seja, existe um espaço virtual em que objetos são posicionados, e uma perspectiva deles é criada para ser exibida na tela.

8/16/32 *bits* - Uma das características na arquitetura do processador gráfico utilizado no console. Um avanço na arquitetura resultava em uma significativa melhora no poder de processamento e, portanto, na qualidade dos gráficos. Os jogos que funcionavam 8 *bits* são geralmente relacionados com a estética de *Pixel Art* característica da época.

Além dos avanços tecnológicos, outra maneira comum de catalogar os jogos, é criar categorias para os que já existem, e tentar agrupar aqueles que apresentem uma mecânica ou temática semelhantes em um mesmo gênero. Irei listar alguns dos mais comuns, porém esta é uma lista dinâmica, e um mesmo jogo pode estar associado a mais de uma categoria.

Shooter - Os jogos de tiro são um dos primeiros gêneros a surgir, com *Space War!* e os clássicos de *Arcade* como *Space Invaders*, muitos outros seguiram a mecânica de atirar em inimigos comandando um veículo ou personagem.

FPS - *First Person Shooter*, ou jogo de tiro em primeira pessoa. É um gênero que tem suas raízes ainda na era dos *Mainframes* das universidades, com jogos como o *Spasim*, que funcionava na rede *PLATO* por volta de 1974. O sucesso e

consolidação entretanto, só ocorreram no início dos anos 90 com a série de lançamentos da *Id Software*, *Wolfenstein* 3D, *Doom* e *Quake*.

Apesar de ter surgido com jogos de uma temática sangrenta, um ambiente futurista e monstros grotescos como inimigos, a mecânica principal de atirar com uma arma em primeira pessoa seguiu diversos rumos e inspirou jogos como de simulação de guerra, espionagem e até mesmo outras abordagens como de um quebra cabeças. Sua popularidade é vista atualmente em jogos de franquias como *Battlefield*, *Call of Duty*, *Counter-Strike*, entre outros.

Plataforma - São jogos tipicamente 2D, onde um personagem se utiliza de plataformas no cenário para progredir pela fase e atingir seus objetivos. O gênero surgiu ainda na época dos *Arcades*, e ficou amplamente reconhecida em jogos como *Super Mario* e *Mega Man*. Outro termo que costuma ser utilizado em conjunto com os jogos de plataforma é *Side Scroller*, que indica um jogo em que você progride avançando para os lados, geralmente da esquerda para a direita.

RPG - *Role Playing Game* é um termo utilizado em jogos, inclusive não digitais, onde você interpreta o papel de um personagem e busca agir de acordo com a sua conduta. É comum a herança dos RPGs clássicos de livro como *Dungeons & Dragons*, onde existe um sistema de níveis, pontos e habilidade que ditam a força e as funções do seu personagem. O maior atrativo desse gênero é a história imersiva e o sistema de combate. Jogos notáveis são *Zelda*, *Elder Scrolls*, e *Baldur`s Gate*.

J-RPG - É uma variação do RPG criada para *designar* os jogos feitos no Japão e no oriente em geral. Alguns elementos como o estilo da história, dos gráficos e a natureza do combate são notavelmente diferentes dos RPGs criados na América, por isso alguns preferem fazer essa distinção. Esse prefixo para *designar* o estilo oriental também pode ser encontrado em outros gêneros. Alguns exemplos são *Final Fantasy*, *Chrono Trigger* e *Pokemon*.

MMORPG - *Massive Multiplayer Online RPG*, ou simplesmente MMO. É um termo que surgiu para designar jogos de RPG online com um grande número de jogadores simultâneos. O diferencial desse estilo é a interação social entre os jogadores, criando um mercado e uma cultura em torno do jogo. Grandes nomes são Última Online, *World of Warcraft* e *EVE Online*.

Luta - Os jogos de luta são um gênero onde dois oponentes escolhem personagens e lutam em uma mistura de reflexos e estratégia de golpes. Alguns são

apenas uma arena, enquanto outros apresentam uma progressão em fases com diversos inimigos. Esses jogos de lutas abertas também pode ser denominado *Hack and slash*, pelo estilo mais frenético das lutas. Entre os jogos mais famosos estão *Mortal Kombat*, *Street Fighter* e *God of War*.

Estratégia - Por ser um gênero com um público menor, alguns tipos de jogo bem diferentes acabam de encaixando nesta mesma categoria. Um deles é o estilo de estratégia por turnos, onde você gerencia suas unidades com calma e de maneira calculada. Um dos grandes sucessos desse gênero é o *Sid Meier's Civilization*

RTS - *Real Time Strategy* ou jogo de estratégia em tempo real é um estilo caracterizado pelo gerenciamento de recursos e evolução de uma base e um exército. Você comanda as unidades em tempo real e busca produzir um exército para destruir as bases inimigas. Os exemplos mais famosos desse gênero são *Starcraft*, *Warcraft* e *Age of Empires*.

MOBA - *Multiplayer Online Battle Arena* é um termo criado recentemente para tentar designar jogos no estilo de *Dota* e *League of Legends*. Estes consistem em dois times em um mapa fixo, com o objetivo de destruir a construção central da base inimiga, utilizando um personagem para cada jogador que lutam contra a equipe adversária.

2.4 Ferramentas de construção de jogos digitais

Como mostrado anteriormente, no início, os jogos digitais eram feitos por um grupo pequeno de pessoas, na maioria engenheiros e programadores que criavam toda a estrutura necessária. Mas é uma prática comum criar trechos de código que possam ser reutilizados, diminuindo o esforço necessário em futuros projetos. E assim surgiam aos poucos algumas ferramentas de desenvolvimento pessoais de cada desenvolvedor.

Com o amadurecimento da indústria, e o aumento no escopo dos jogos, um time de pessoas cada vez maior era necessário para entregar um produto de alta qualidade. *Designers*, ilustradores, escritores e compositores são alguns dos profissionais que contribuem para o desenvolvimento de um *game*. Para facilitar a inserção dos elementos criados por eles, e a fim de testar diretamente os resultados

obtidos, as empresas criavam ferramentas que auxiliassem na assimilação dos chamados *assets*.

Assim, cada empresa produzia sua coleção ferramentas para serem usadas internamente, como uma espécie de segredo industrial. A empresa que desenvolvesse as melhores ferramentas, poderia produzir os jogos com mais detalhes e um nível técnico maior em menos tempo, o que era visto como uma vantagem de mercado.

Porém no cenário independente, algumas pessoas tinham um desejo de expressar sua criatividade e fazer um jogo próprio. Mas como o nível técnico exigido para programar é alto, algumas outras ferramentas começaram a surgir na década de 80 para suprir essa necessidade. Apesar de não ter sido a primeira, a que ficou mais conhecida foi a série *RPG Maker*, criada pela *ASCII* (compania Japonesa, não confundir com o código de caractéres) em 1992, que permitia pessoas menos experientes a criarem um jogo de RPG em 2D para computador.

A ideia de licenciar comercialmente o núcleo de funcionamento de um jogo para outras empresas, surgiu após o grande sucesso dos primeiros jogos em FPS, permitindo com que as equipes pudessem crescer e se especializar. Os próximos lançamentos, como o *Quake III Arena* da Id Software e o *Unreal* da Epic Games, já foram feitos com o intuito de vender a sua “Engine” (termo criado nessa época que significa Motor) de funcionamento após o lançamento do jogo.

2.4.1 Elementos de um jogo digital

Após a criação das primeiras ferramentas, ficou mais clara uma separação entre os elementos que eram específicos de cada jogo, e os que serviam como estrutura e funcionalidade. Os códigos básicos ficavam a cargo das ferramentas, funções como movimento, colisão, renderização, cálculos de física, entre outros, são comuns a grande parte dos jogos.

Por outro lado os elementos específicos precisavam ser inseridos e manipulados para construir o cenário desejado. Eles entram como dados externos chamados de *assets*, que são salvos e associados aos objetos do jogo para criar o produto final desejado.

Alguns desses elementos são:

- Modelos em 3D

- Texturas
- Efeitos Sonoros
- Músicas
- Vozes
- *Scripts*
- Textos
- Mapas
- *Sprite sheets*

Esse tipo de estrutura é chamada de “modelo de dados”, pois o que define o jogo, suas mecânicas e seu visual, são os dados que são inseridos nas ferramentas para dar forma a ele. Esses elementos também são utilizados em outros modelos, porém não há uma separação tão clara do que seriam dados ou estrutura.

2.5 Framework

Segundo *Fayad* e *Schmidt* (1997), "Uma *framework* é uma aplicação reutilizável, semi-completa, que pode ser especializada para produzir aplicações customizadas". Diferente de outras técnicas de reutilização de código baseadas em classes e bibliotecas, uma *framework* é direcionada a um domínio de aplicação específico, e busca trazer soluções para problemas característicos à aquela área.

Os principais benefícios vêm da modularidade, reusabilidade, extensibilidade e inversão de controle presentes na estrutura. A modularidade ajuda a melhorar a qualidade do software ao localizar o impacto de mudanças de design e implementação, reduzindo o esforço que seria necessário para entender e manter um software existente. As interfaces estáveis melhoram a reusabilidade, definindo componentes genéricos que podem ser reutilizados para criar novas aplicações. Com foco no domínio específico, a contribuição de desenvolvedores experientes evita que problemas comuns precisem ser recriados e revalidados. Além de oferecer ganchos para métodos, que permitem expandir as funcionalidades das interfaces sem afetar sua estabilidade, isso faz com que o *framework* se mantenha extensível a novas características e funcionalidades.

Mas apesar das vantagens apresentadas, existem alguns pontos a serem considerados quando se pensa em utilizar uma *framework*. Desenvolver um software complexo já tem um nível de dificuldade bastante elevado, fazer com que além de

tudo ele seja modular, reutilizável e extensível, especialmente em um domínio de alta complexidade, é ainda mais difícil, e normalmente só consegue ser feito por *experts* na área. Porém os processos de arquitetura e engenharia de software já vêm sendo estudados, buscando tornar mais viável a criação de novas *frameworks*.

É necessário levar em conta o tempo requerido para aprender o funcionamento e as características de uma determinada *framework*. Caso esse aprendizado possa ser feito ao longo de alguns projetos, o desenvolvedor vai se familiarizando com o tempo e se torna mais produtivo. Porém se o uso é limitado a apenas um projeto em particular, muitas vezes esse custo de aprendizado faz com que sua utilização não seja uma vantagem.

Da mesma forma que um software amadurece com o tempo, a *framework* também o deve fazer, e as aplicações que a utilizam precisam se manter atualizadas com as alterações feitas. De certa forma, o desenvolvedor de aplicações e o usuário final acabam ficando dependentes do desenvolvedor da *framework*, pois a sua manutenção exige um profundo conhecimento dos seus componente e interrelacionamentos presentes em suas abstrações.

2.5.1 Engine

O termo *Engine* significa Motor em inglês, e começou a ser utilizado na década de 90 com a evolução dos jogos em 3D. Ele representa a base estrutural que faz o jogo funcionar, daí a analogia com o motor, que é responsável por dar funcionamento a diversos tipos de máquina.

Criar um jogo em 3D exigia um conhecimento computacional muito elevado, e o esforço necessário para desenvolver um software de renderização otimizado era muito alto. Disso surgiu a necessidade clara de reaproveitar o código e as ferramentas já existentes, e tornar um time responsável exclusivamente a isso, que poderia se especializar e buscar ultrapassar os limites tecnológicos de renderização, e outros aspectos da computação gráfica e dos fundamentos do desenvolvimento de jogos.

Não existe uma distinção estrita entre uma *framework* e uma engine, muitos termos e funcionalidades se misturam e se sobrepõe. Entretanto, profissionais da área como *Josh Petrie* apontam que em geral se espera que uma *engine* seja mais

completa e autônoma, capaz de realizar tarefas mais robustas por trás de uma abstração, da mesma forma que nos referimos a um motor de busca como o *Google*.

De um ponto de vista prático, é apenas um problema de terminologia, onde programas intitulado como *engines* podem utilizar outros intitulado de *frameworks* buscando funcionalidade, e vice versa. Quando pensamos em desenvolver um jogo, devemos buscar as ferramentas que nos tragam as funcionalidades que desejamos, da maneira mais rápida e fácil de ser utilizada.

2.5.2 Exemplos

Existem vários exemplos de *frameworks* e *engines* comerciais, aqui estão apenas algumas que são mais relevantes à história da indústria ou ao desenvolvimento deste estudo.

***id Tech* 1 -** Também conhecida como *Doom Engine*, foi a primeira *Engine* da *id Software* a ser reconhecida e licenciada para uso externo. Desenvolvida principalmente por *John Carmack* em 1993, inicialmente para computadores *NeXT*, foi então portada para *DOS* para o lançamento de *Doom* e posteriormente para consoles e outros sistemas. Feito em código C, usa um estilo peculiar para renderizar um mapa 3D a partir de um ambiente 2D, criando uma espécie de ambiente chamado de 2,5D, ou 2D e meio.

Após alguns anos seu código foi licenciado como código aberto e está disponível para estudo, juntamente com o jogo, no repositório *GitHub*.

***Unreal Engine* -** Desenvolvida pela *Epic Games* em 1998 juntamente com o jogo de mesmo nome, era concorrente direta da *id Tech* 2 e 3. Já com diversas características como renderização, detecção de colisão, inteligência artificial e código de rede, foi se desenvolvendo e tem sua versão mais recente com grande impacto na indústria até hoje.

Sua principal contribuição foi o uso de uma linguagem de script própria chamada *Unrealscript*, que permitia uma maior customização dos jogos e deu origem a uma comunidade de modificações para os jogos chamadas *mods*. Com elas era possível adaptar jogos existentes, adicionando ou modificando seus elementos, podendo inclusive resultar em jogos completamente novos.

***Source* -** Desenvolvido pela *Valve*, é a sucessora da *GoldSrc*, uma das inúmeras *engines* que surgiram a partir da liberação do código da *Quake Engine* (*id*

Tech 2). Apesar de ter grande parte do código totalmente reescrito, é um bom exemplo do legado que as engines da id Software deixaram para a indústria de jogos.

Mesmo tendo sido desenvolvida com o foco em jogos de FPS, hoje já abrange estilos de quebra-cabeça, MMO, MOBA, e outros. Seus desenvolvedores fazem modificações incrementais e constantemente escrevem artigos sobre seus estudos e avanços tecnológicos, que podem ser encontrados no site da *Valve Software*.

***Unity* -** É uma engine que vem se tornando muito popular por ter uma licença básica grátis, incentivando desenvolvedores independentes a testarem e criarem seu jogos para uma diversificada gama de plataformas, com um investimento inicial baixo. Esse fator, e o grande crescimento do mercado de jogos *mobile* que aconteceu nos últimos anos, fizeram com que ela crescesse em popularidade e em nível técnico, trazendo maiores possibilidades aos desenvolvedores independentes e estudantes da área.

XNA - Uma *framework* lançada pela *Microsoft* em 2006, baseada em outra *framework* da empresa chamada .NET (*dot net*). Foi desenvolvida com o intuito de ajudar desenvolvedores independentes a criarem jogos para algumas plataformas, incluindo *Windows* e *Xbox* 360, especificamente a loja de jogos independentes *Xbox Live Indie Games*. Seu foco principal era o desenvolvimento de jogos leves, que pudessesm rodar nas plataformas com suporte ao *runtime* da *framwork* com pouca ou nenhuma modificação no código. Foi descontinuada após mudanças nas plataformas da empresa, sua última versão foi estável é de 2011.

***Crafty* -** *Framework* de jogos em *HTML*5 criada por *Louis Stowasser* em código aberto,é um exemplo de uma das ferramentas que surgiram para incentivar o desenvolvimento nessa nova tecnologia web, que vem sendo aprimorada e recebida de forma positiva pela comunidade.

CreateJS - É um conjunto de bibliotecas para facilitar o desenvolvimento de jogos em *HTML*5, *EaselJS* cria animações utilizando *spritesheets*, *TweenJS* cria movimentações elásticas comuns, *SoundJS* para a reprodução de audio, e *PreloadJS* para o carregamento assíncrono de *assets*.

3 Projeto da *Framework – Start Game* Bauru

A *framework* desenvolvida neste trabalho, com o nome de *Start Game* Bauru, é uma ferramenta de pequeno porte, que busca trazer funcionalidades básicas do desenvolvimento de um jogo digital, para a plataforma *Web*, utilizando principalmente a *tag* canvas do *HTML*5.

Esta plataforma foi escolhida pois a partir de 2009 e 2010 houve uma grande atenção por parte da indústria e da mídia sobre o desenvolvimento da versão 5 do *HTML*, que traz novas funcionalidades como semântica, acessibilidade e reprodução nativa de mídia. Uma grande mudança era prevista no ambiente de desenvolvimento web, e os *Browsers* estavam buscando se atualizar à medida que as especificações da nova versão eram decididas na *W3C*. No final de 2014 foi dado o status de "Recomendado" para a versão da linguagem, que significa que esta já está estável e não deve sofrer mais grandes modificações.

Juntamente com algumas novidades na linguagem *CSS*, que também vem sendo atualizada para sua versão 3, vemos grandes melhorias acontecendo ao redor da linguagem *Javascript*. O motor de interpretação V8 produzido pela *Google*, traz já um significativo ganho de performance para seu *browser*, *Chrome*, e vem sendo utilizado inclusive em aplicações de servidores com a plataforma *NodeJS*.

O crescimento da comunidade de desenvolvedores *Javascript*, faz com que o número de bibliotecas e outras ferramentas aumente cada vez mais, o que chama a atenção das empresas que buscam estarem atentas para manter sua competitividade no mercado. A *Oracle*, por exemplo, anuncia que a uma das melhorias da versão 8 do *Java*, será seu interpretador de Javascritpt, um concorrente ao V8 da *Google*, escrito em linguagem *C*++.

Todas essas novidades, e o fato de que as linguagens *Web* buscam atender o maior número possível de dispositivos, despertam também a curiosidade dos desenvolvedores de jogos, que vêm na plataforma uma nova oportunidade de criar produtos com maior alcance. Existem diversas barreiras ao criar um jogo sem um hardware definido, mas é justamente esse tipo de problema que as *frameworks* ou *engines* ajudam a solucionar.

3.1 Requisitos

Como vimos anteriormente, o crescimento e popularização das *engines* começou com a mudança para o ambiente 3D, que exigia um conhecimento técnico mais aprofundado e um maior esforço de desenvolvimento base. Apesar de já existirem grandes avanços que exploram a renderização 3D na *Web*, utilizando *WebGL*, as grandes *engines* comerciais ainda estão explorando suas possibilidades e limitações.

Neste estudo vamos explorar apenas algumas funcionalidades básicas de um jogo em 2D e soluções para algumas características do desenvolvimento *Web* explicadas a seguir:

3.1.1 Requisição Assíncrona

Na plataforma *Web*, uma página é requisitada ao servidor de origem e então transferida e carregada no browser do cliente. Buscando melhorar a experiência do usuário, as melhores práticas de desenvolvimento tentam fazer com que o tempo de carregamento da página seja o menor possível. Porém no campo dos jogos eletrônicos, os *assests* necessários incluem arquivos considerados grandes para o carregamento usual de uma página, como imagens e arquivos de áudio.

Assim uma forma de evitar uma espera indeterminada é carregar apenas o necessário para a exibição da página, e começar o carregamento de forma assíncrona. Utilizando a tecnologia *AJAX*, podemos avisar o usuário do processo de carregamento e solicitar arquivos através de requisições *HTTP*.

3.1.2 *Game Loop*

A estrutura central de um jogo do ponto de vista da programação é o ciclo de jogo, ou game loop. Enquanto a maioria dos programas aguarda uma entrada do usuário para responder, os jogos devem continuar funcionando independente de uma ação do usuário, podendo essa falta de resposta ser inclusive parte da mecânica do jogo.

Um *loop* de jogo consiste basicamente dos cálculos necessários em um movimento ou jogada, e em seguida o código responsável por apresentar os resultados. Dada a natureza gráfica dos jogos eletrônicos, a renderização dos elementos está diretamente ligada ao ciclo de processamento e ao desempenho do

jogo, onde se entende que quanto maior o número de ciclos por segundo, mais precisos serão os cálculos e mais suaves as animações.

3.1.3 Abstração de Eventos do usuário

Por definição todo jogo deve ter algum tipo de interação com o jogador, que pode ser realizada através de diferentes instrumentos. Idealmente uma camada de abstração permite que o jogo possa responder de forma genérica a diferentes formas de entrada.

A plataforma *Web* por sua natureza aberta permite que inúmeras formas de navegação possam ser utilizadas, dês de o controle de uma *SmartTV*, um console ou até mesmo dispositivos e sensores biométricos. Entretanto para manter o escopo do projeto viável iremos nos concentrar nos métodos de entrada comuns a computadores e dispositivos móveis, como mouse, teclado e telas sensíveis ao toque.

3.1.4 Rotinas Gráficas

Um conhecimento moderado de computação gráfica é necessário para exibir os objetos de um jogo na tela de maneira correta, e isso demandaria um tempo de estudo e desenvolvimento consideráveis. O uso de rotinas ou bibliotecas prontas melhora o desempenho e reutilização de código, permitindo que seja possível dar mais tempo e atenção a outros problemas mais específicos ao jogo desenvolvido.

3.1.5 Estrutura de Objetos

Ao criarmos elementos em um jogo, é comum perceber que muitas funcionalidades são comuns, e que a maioria deles precisa de uma estrutura de dados básica, para seu funcionamento e interação com os demais elementos. Assim uma estrutura comum a objetos de um jogo pode ser criada para os padronizar e organizar.

3.1.6 Estrutura de Cenas

À medida que os jogos começam a se tornar cada vez maiores e mais complexos, cresce também a dificuldade em lidar computacionalmente com a quantidade de dados necessários para o seu funcionamento. Com um espaço de

memória limitado, é necessário organizar os elementos que serão carregados em determinado momento, enquanto outros que não sejam necessários possam permanecer em memória secundária.

Existe uma técnica chamada de gerenciamento de cenas, que consiste em separar as telas do jogo e suas estruturas de forma a as encapsular, para que possam ser gerenciadas individualmente. É comum definir uma cena pela sua imagem de fundo, já que esta é um dos maiores objetos que se costuma ter em determinado instante. As cenas podem ser carregadas e descarregadas de acordo com a necessidade, e consistem de telas como o *menu*, *ranking*, fases, etc...

3.1.7 Colisão

Um problema comum em jogos é a simulação de características físicas, como gravidade e colisão. Para efeito de jogabilidade não é necessário que essa simulação seja exata, pelo contrário, muitas vezes para dar maior fluidez à jogabilidade, conceitos físicos são alterados para dar característica ao universo fictício do jogo.

Apesar das regras físicas serem maleáveis de um jogo para outro, a detecção de colisão é um elemento presente em toda simulação. Com ela somos capazes de saber quando um objeto entrou em contato com outro objeto, ou com os limites o ambiente. É com ela que sabemos quando um tiro atingiu um inimigo, ou quando o jogador encostou em um espinho por exemplo, e a partir daí efetuar as mudanças necessárias no estado do jogo.

3.2 *Design* da Arquitetura (modular – diagrama de blocos)

O objetivo fundamental da arquitetura deste projeto, é mesclar os conceitos aprendidos sobre o desenvolvimento *Web* e de Jogos, e unir os dois de forma familiar para pessoas com conhecimento em um desses campos. As tecnologias individuais a cada campo podem ser identificadas e distintas das que possam representar conceitos novos para o desenvolvedor que busca utilizar esta *framework*.

Em um primeiro momento temos o carregamento da página *HTML*, apenas com seu conteúdo e estilo em *CSS* para que o carregamento inicial seja rápido e

prenda o usuário em um ambiente personalizado. Em seguida é carregado o *Script* inicial da *framework* que inicia a requisição do código completo para iniciar o jogo.

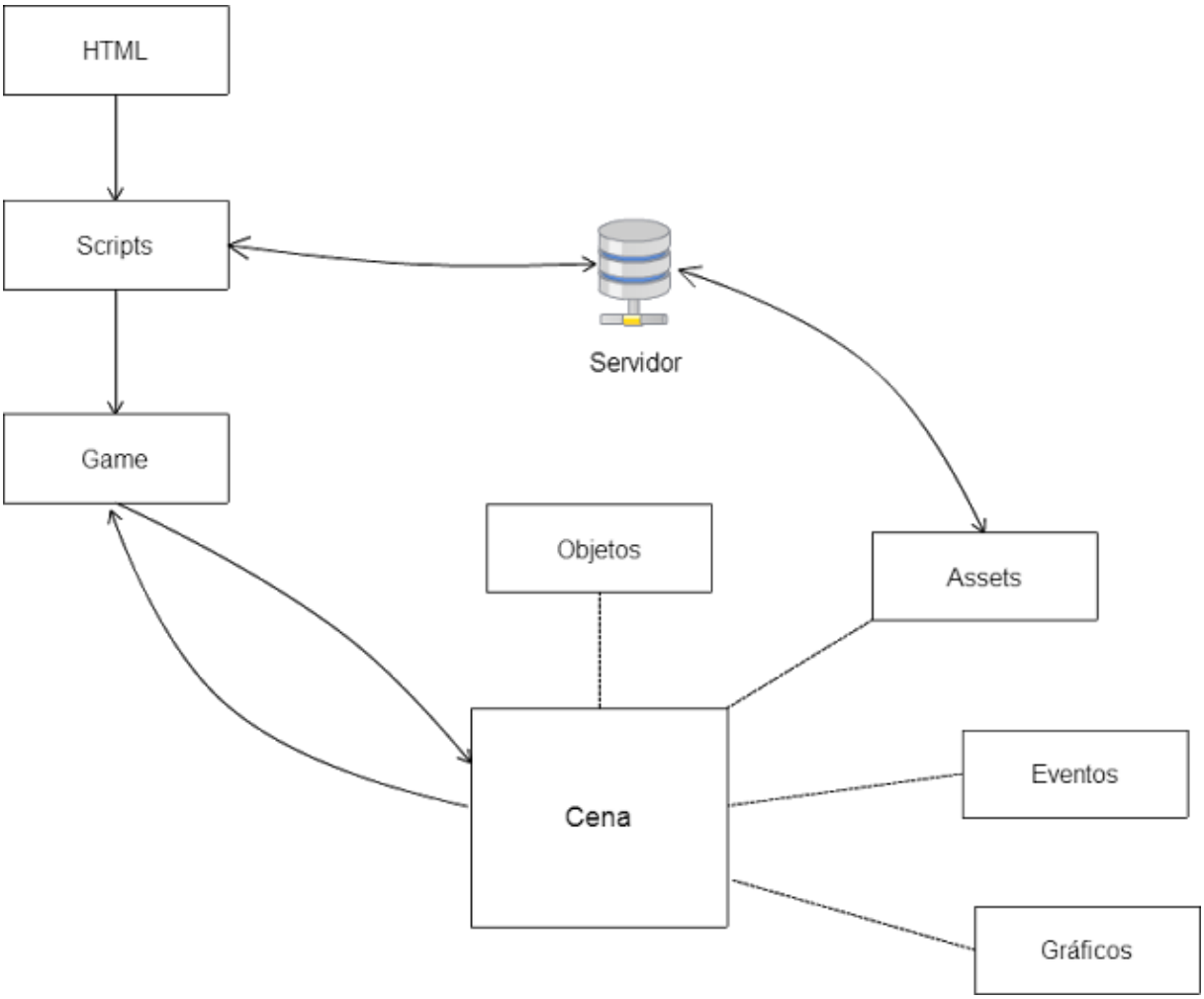

Figura 5 – fluxograma das macros

Após o carregamento da estrutura, o gerenciador de cenas do jogo inicia o processo de carregar os *Assets* necessários para aquela tela. Isso permite distribuir a carga de requisições de forma que ela só aconteça quando for necessário, ou que, em futuras verões, o carregamento possa ocorrer em segundo plano enquanto uma cena está em funcionamento.

O ciclo de funcionamento de uma cena é onde ocorre a maior parte do processamento do jogo, podendo realizar chamadas a bibliotecas e métodos externos que auxiliem no funcionamento desejado para uma aplicação específica. É nela que se utilizam os objetos do jogo, eventos do usuário e rotinas gráficas que o compõem.

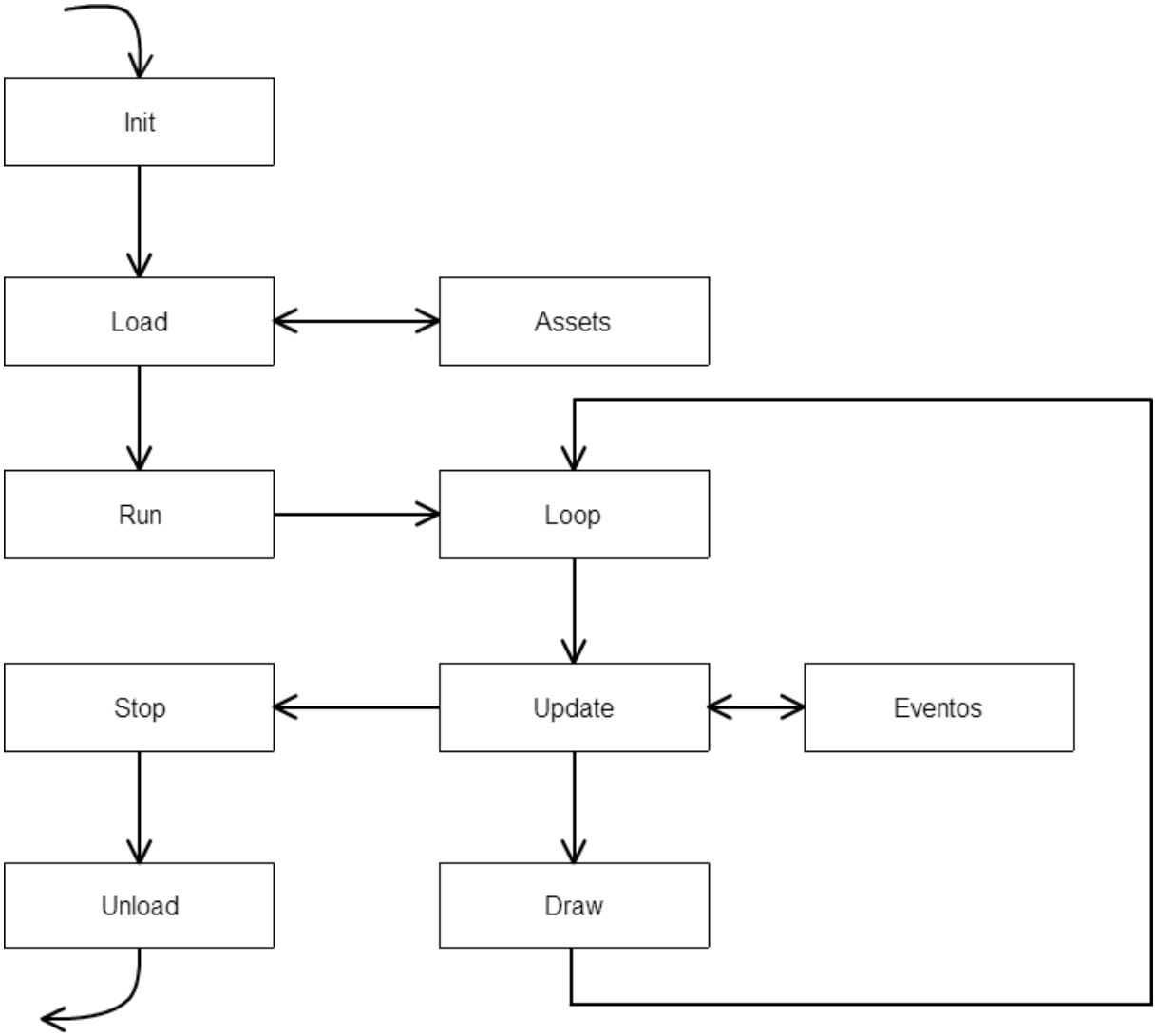

Figura 6 – fluxograma das cenas

3.3 *Framework – Start Game* Bauru

O ponto de início de qualquer página na web é o arquivo *HTML* que será enviado pelo servidor após a requisição da *URL* feita pelo browser e está descrito no Quadro 1.

```
<!DOCTYPE HTML>
<html>
	<head>
		<meta charset="utf-8">
		<meta http-equiv="content-language" content="pt-br">
		<meta name="author" content="Caio Faustino">
		<meta name="reply-to" content="caiofaustino@gmail.com">
		<title>Game Framework - Caio Faustino</title>

		<link rel="stylesheet" type="text/css" href="css/utils/normalize.css" />
		<link rel="stylesheet" type="text/css" href="css/utils/utils.css" />
		<link rel="stylesheet" type="text/css" href="css/style.css" />
		<script src="js/lib/jquery-2.0.3.min.js"></script>
		<script src="js/lib/modernizr.custom.36529.js"></script>
	</head>
	<body>
		<div id="container">
			<canvas id="gameCanvas">Seu Browser não suporta Canvas.</canvas>
		</div>

	<script src="js/base/main.js"></script>
	  </body>
</html>
```

Quadro 1

Neste arquivo podemos observar a estrutura mínima necessária para o uso da *framework*, o estilo que pode ser personalizado, o elemento canvas, e o carregamnto dos scripts, das bibliotecas e do carregador da *framework*. Outros elementos podem ser inseridos na página dês de que haja um espaço bem definido para o canvas.

No *script* principal temos apenas três funções, uma chamada na biblioteca *jQuery* que é feita após o termino do carregamento da página *HTML*, e chama o carregamento dos *scripts* da *framework*.

```
$(document).ready(function()
{
        LoadFramework();
});
```

Quadro 2

Esta função exibida no Quadro 2 carrega um arquivo *JSON* que contém os *scripts* da *framework*, juntamente com o restante do código desenvolvido para a aplicação, e os chama de forma assíncrona e sequencial como pode ser visto na função do Quadro 3.

```
$.getJSON("js/json/scripts.json", function(json){
        Modernizr.load({
                load: json.scripts,
                callback: function (url, result, key) {
                        console.log(url, result, key);
                        if(key == json.scripts.length-1)
                        {
                                LaunchGame();
                        }
                }
        });
});
```

Quadro 3

Após isso o jogo é iniciado, criamos um *buffer* para desenhar os gráficos antes de serem exibidos, evitando um efeito de que os objetos “pisquem” ao serem redesenhados, e o gerenciador de cenas busca a primeira cena descrita em um *array* de *strings* como pode ser visto no Quadro 4.

```
this.Init = function() {
    if (_canvas && _canvas.getContext){
        canvas = _canvas.getContext('2d');
        _buffer = document.createElement('canvas');
        _buffer.width = _canvas.width;
        _buffer.height = _canvas.height;
        buffer = _buffer.getContext('2d');
        self.Load();
    }
};

this.Load = function() {
    eventos = new Eventos();
    self.ProximaTela();
};
```

Quadro 4

Enquanto na estrutura de uma cena, ocorre o ciclo principal de um jogo, com o carregamento dos *assets* necessários e o loop de jogo, que é feito a partir da função *Window.requestAnimationFrame*(), que é chamada pelo *browser* especificamente para realizar animações, buscando manter uma atualização em torno de 60 vezes por segundo, e é interrompida quando a janela perde foco.

```
this.Loop = function() {
    var currentTime = Date.now();
    alphaTime = currentTime - lastTime;
    lastTime = currentTime;

    self.Update();
    self.Draw();
    //https://developer.mozilla.org/en-
US/docs/Web/API/window/requestAnimationFrame
    self.gameLoop = requestAnimationFrame(self.Loop.bind(self), _canvas);
};
```

Quadro 5

É nesta estrutura do Quadro 5, dentro das funções de *Update* e *Draw*, que o desenvolvedor vai montar a maior parte do seu código, específico a cada jogo, com as regras de movimentação, pontos, animações e assim por diante. Sendo assim ainda seria ideal encapsular melhor o código e “esconder” as rotinas básicas, mas devido à natureza de herança por protótipo do *Javascript*, um estudo mais aprofundado nessa questão seria necessário para atingir esse objetivo.

Como complemento ao ciclo da cena, temos uma biblioteca que auxilia no gerenciamento de eventos, como toque, *click* e o teclado. Esta biblioteca exibida no Quadro 6 não está completa e precisaria ser revista para incluir novas práticas de tratamento de eventos de toque pelos *browsers*. Mas ainda é funcional e sua interface pode ser reaproveitada em novas versões, não havendo necessidade de mudança no código das aplicações.

```
var mouse = {x:1, y:1, poll:false, target: null, isDown:false};

this.MouseDown = function(e) {
    posCanvas = findPos(_canvas);
    if (isTouch) {
        addEventListener("touchend", self.MouseUp);
        _canvas.addEventListener("touchmove", self.MouseMove);
    }
    else {
        addEventListener("mouseup", self.MouseUp);
        _canvas.addEventListener("mousemove", self.MouseMove);
    }
    mouse.isDown = true;
    calcMouse(e);
    self.buscaTarget();
};
```

Quadro 6

Podemos ver um trecho onde os eventos de mouse e de toque são tratados para um objeto abstrato do mouse. Uma melhor abordagem deveria permitir que ambos os eventos pudessem ocorrer em um mesmo cenário, mas essa é uma melhoria que não pode ser implementada nesta versão da *framework*.

E por fim um pouco do código utilizado na biblioteca gráfica que auxilia o desenvolvedor nas tarefas de desenho, que costumam ser as mais trabalhosas por utilizarem alguns conceitos matemáticos com vetores e matrizes.

```
function desenhaRotacionado(image, x, y, angle, context) {
    // salva contexto
    context.save();

    // move para onde queremos desenhar
    context.translate(x, y);

    // rotaciona
    context.rotate(-angle);

    // desenha pelo meio da imagem
    context.drawImage(image, -(image.width/2), -(image.height/2));

    // retorna para o contexto inicial
    context.restore();
}

function desenhaEscala(obj, mult) {
    buffer.drawImage(obj.img, 0, 0, obj.img.width, obj.img.height, obj.x, obj.y,
obj.img.width*mult, obj.img.height*mult);
}
```

Quadro 7

No Quadro 7 vemos as rotinas de desenhar um objeto rotacionado ou em escala na tela, algo não muito complexas que exige um entendimento maior das funcionalidades da *API* de desenho do canvas. Esta biblioteca pode ser aprimorada tanto em novas versões como pelo próprio desenvolvedor que busque acrescentar rotinas que sejam relevantes ao seu uso particular.

3.3.1 Bibliotecas utilizadas

Algumas bibliotecas de uso comum em projetos para *Web* já estão inclusas na *framework* e podem ser utilizadas nas aplicações criadas. Elas são:

JQuery - É uma biblioteca de uso geral, amplamente utilizada em projetos comerciais e já a muito tempo consolidada, com grande confiabilidade. Tem como objetivo otimizar o código necessário para buscar e modificar elementos *HTML* através da estrutura *DOM*, além de eventos, animações, requisições *Ajax* e muito mais.

Modernizr - Uma pequena biblioteca que detecta a disponibilidade da implementação nativa de novos recursos *Web*, principalmente de *HTML*5 e *CSS*3. Permite também o carregamento externo de *scripts* que podem implementar o uso de recursos que não estejam presentes no browser utilizado.

4 Exemplo de game criado com o *framework Start Game* Bauru

Um jogo simples com uma mecânica de tiro foi desenvolvido para exemplificar e comprovar a utilização da framework e está descrito neste capítulo.

4.1 Roteiro do game

Este é um jogo extremamente simples, com o único intuito de exemplificar a utilização da *framework Start Game* Bauru para o desenvolvimento de jogos.

O jogo é do gênero *Shooter*, um dos primeiros a serem criados, e consiste em uma nave que anda pelo espaço e precisa atirar nos inimigos que estiverem vindo em sua direção para não ser destruída. O objetivo é permanecer vivo o maior tempo possível e fazer o maior número de pontos, que podem ser obtidos apenas por se manter vivo, e por destruir naves inimigas.

A jogabilidade e as regras estão simplificadas ao máximo, apenas para que se possa ver o jogo em funcionamento, mas com mais tempo seria possível adicionar novas mecânicas e novos desafios para o jogador. Esse tipo de modificação fica a cargo do *Game Designer*, que decide quais as melhores maneiras de engajar o usuário, e passa para o desenvolvedor da aplicação as alterações a serem feitas.

4.2 Elementos do game

Os elementos centrais do jogo são a Nave controlada pelo jogador, os tiros que ela dispara, os inimigos a serem enfrentados e a imagem de fundo que dá ambientação. Além dos scripts necessários para o funcionamento do jogo.

A nave consiste em um objeto de jogo que contém um *Drawable*, ou seja, uma imagem que pode ser desenhada na tela. Seu controle é feito através do mouse ou de uma tela sensível ao toque, ao pressionar a tela a nave se move para a esquerda ou para a direita com velocidade constante, e ao final do movimento ela dispara um tiro em linha reta.

Os inimigos também são objetos de jogo contendo um *Drawable*, que surgem no topo da tela e descem em velocidade constante até atingirem a nave do jogador, ou o fundo da tela. Eles são destruídos ao serem atingidos por um tiro disparado pelo jogador.

O tiro é um mais um elemento que pode ser desenhado, ele é criado quando o jogador finaliza o movimento da nave, e segue em linha reta para cima com velocidade constante até atingir um inimigo ou sair da tela de jogo.

A imagem de fundo representa o espaço, que juntamente com os desenhos escolhidos para as naves dá um tema de guerra no espaço, em homenagem ao jogo *Spacewar!* que inspirou o início da criação dos jogos eletrônicos.

4.5 Imagem do jogo

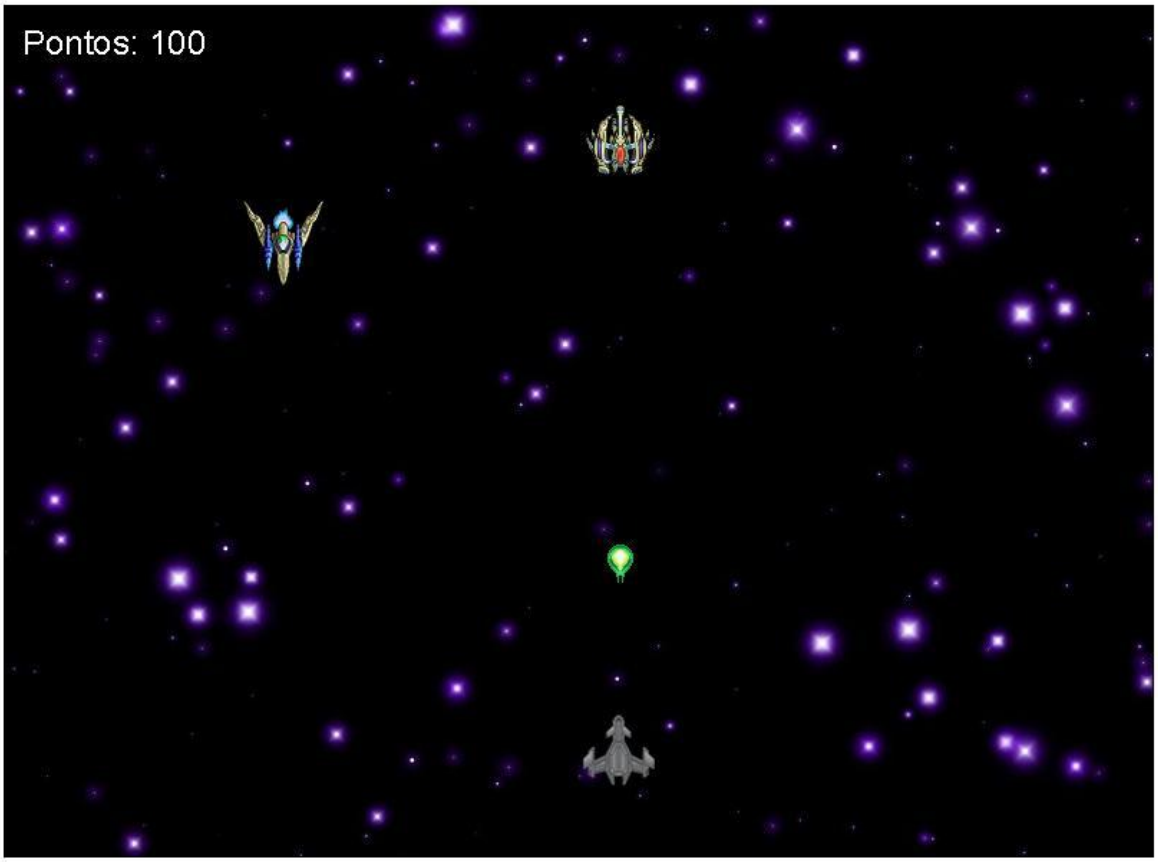

Figura 7 – tela do jogo

5. Considerações finais

O desafio de desenvolver um jogo é muitas vezes subestimado por aqueles que nunca tentaram de fato o fazer. Entretanto, ao estudar o assunto percebe-se que a gama de desafios e técnicas utilizadas abrange muito mais do que uma pessoa seria capaz de dominar durante sua vida, principalmente pela diversidade de áreas e competências que fazem parte dessa área. Por esse motivo percebe-se que cada vez mais os profissionais se especializam em tarefas bem específicas, em times com até centenas de pessoas trabalhando para desenvolver um jogo comercial de grande porte.

Seria de certa forma inocente acreditar que este trabalho pudesse oferecer soluções completas e definitivas para o seu propósito, algo praticamente impossível dada à natureza progressiva da criação de jogos e da plataforma escolhida. Mas o aprendizado vem com a busca e a prática, e os estudos realizados foram de suma importância para aprimorar conhecimentos na área e podem servir de orientação ou base para novos estudos.

Em uma nova abordagem, já seria possível rever alguns conceitos e buscar aprimorar a ferramenta, de forma a torná-la mais completa e otimizada. Mas mesmo em seu atual estado, já podemos compreender o funcionamento básico de um jogo e realizar tarefas mais simples, incentivando a curiosidade e abrindo possibilidade para que novos estudos sejam feitos. Este é o objetivo principal, aprender cada vez mais e contribuir com o crescimento e a valorização do cenário de desenvolvimento de jogos.

6. Referências

AMAZONAS, Daniel Souza. **Desenvolvimento de Jogos 3D em Java com a Utilização do Motor Gráfico Irrlicht.** 2007. 61 f. TCC (Graduação) - Curso de Ciência da Computação, Faculdade Lourenço Filho Ciência da Computação, Fortaleza, 2007

BAER, Ralph. **Genesis: How the Home Video Games Industry Began.** 2014. Disponível em: <http://www.ralphbaer.com/how_video_games.htm>. Acesso em: 11 fev. 2015.

_____. **Videogames: In the Beginning**. Springfield: Paperback, 2005.
BARBOSA, Laura Monte Serrat. **Projeto de trabalho: uma forma de atuação psicopedagógica.** 2.ed. Curitiba: L. M. S, 1998.

BISHOP, Lars et al. Designing a PC Game Engine. **Computer Graphics In Entertainment,** Usa, p.2-9, jan. 1998. Bimestral.

BOTELHO, Luiz. **Jogos educacionais aplicados ao e-learning**. Disponível em: <http://www.elearningbrasil.com.br/news/artigos/artigo_48.asp.> Acesso em: 11 jan 2015.

BRIGHT, Peter. **HTML5 specification finalized, squabbling over specs continues.** 2014. Disponível em: <http://arstechnica.com/information-technology/2014/10/html5-specification-finalized-squabbling-over-who-writes-the-specs-continues/>. Acesso em: 01 jan. 2015.

CACHE, Jonnie. Disponível em: <https://gist.github.com/JonnieCache/4258114> Acesso em: 7 fev. 2015.

CARMACK, John. **John Carmack's Blog.** 2004. Textos diversos. Disponível em: <http://armadilloaerospace.com/n.x/johnc/recent updates/archive?news_id=290>. Acesso em: 1 fev. 2015.

COX, Stu et al. **What is Modernizr?** 2015. Disponível em: <http://modernizr.com/docs/>. Acesso em: 13 fev. 2015.

D. C. Schmidt et al, **Frameworks: Why They Are Important and How to Apply them E ectively**. ACM Queue magazine, Volume 2, Number 5, 2004.
DOOM. Id-Software. Disponível em: <https://github.com/id-Software/DOOM> Acesso em: 7 fev. 2015.

Drucker, P.F. (1993). **Post-Capitalist Society**. New York: HarperCollins.
FAYAD, Mohamed E.; SCHMIDT, Douglas C.. Object-Oriented Application Frameworks. **Communications Of The Acm,** Usa, v. 40, n. 10, p.32-38, out. 1997. Mensal.

_____. **Object-oriented Application frameworks. Communications of the ACM**, Vol. 40, 10 p., 1997. Disponível em: <http://dl.acm.org/citation.cfm?id=262798>. Acesso em: 1 fev. 2015

FOUNDATION, Jquery (Ed.). **What is jQuery?** 2015. Disponível em: <http://jquery.com/>. Acesso em: 13 fev. 2015.
GREGORY, Jason. **Game Engine Architecture.** Wellesley, Massachusetts, Usa: A K Peters, 2009.

HALLOCK, Austin. **Why HTML5 will succeed for gaming. Gamasutra Blogs**. Jul. 2012. Disponível em: <http://www.gamasutra.com/blogs/AustinHallock/20120715/174150/> Acesso em: 14, fev. 2015.

HARMAN, Stace (Comp.). **HISTORY OF FIRST:** PERSON SHOOTERS. Disponível em: <http://uk-microsites.ign.com/history-of-first-person-shooters/>. Acesso em: 8 fev. 2015.

HICKSON, Ian. **HTML 5.** San Francisco: Google, 2014. 863 p. Disponível em: <https://html.spec.whatwg.org/print.pdf>. Acesso em: 5 jan. 2015.

HISTORY, The National Museum Of American. **The Brown Box, 1967–68.** 2001. Disponível em: <http://americanhistory.si.edu/collections/search/object/nmah_1301997>. Acesso em: 3 jan. 2015.

HUIZINGA, Johan. **Homo Ludens**: O Jogo Como Elemento Da Cultura. 5. ed. São Paulo: Perspectiva, 1999.

JS, Create (Org.). **A JavaScript library that makes working with the HTML5 Canvas element easy.** Disponível em: <http://createjs.com/EaselJS>. Acesso em: 9 fev. 2015.

KIRSTEN TASHEV (Usa). Computer History Museum (Org.). Spacewar! 2014. Disponível em: <http://pdp-1.computerhistory.org/pdp-1/?f=theme&s=4&ss=3>. Acesso em: 8 jan. 2015.

LEWIS, Michael; JACOBSON, Jeffrey. GAME ENGINES IN SCIENTIFIC RESEARCH. **Communications Of The Acm,** Usa, v. 45, p.27-31, jan. 2002. Mensal.

MACHADO, Marcos Eduardo; MACHADO, Paulo Maurício (Comp.). Divindades egípcias de pewter em lindo jogo de xadrez temático Leia mais: http://www.materiaincognita.com.br/divindades-egipcias-de-pewter-em-lindo-jogo-de-xadrez-tematico/#ixzz3TzsLrl3M. 2014. Disponível em: <http://2.bp.blogspot.com/-Nw5tvIVfng4/T1bKSZbqjKI/AAAAAAAAAf0/62Ltm6y_U0M/s1600/Queen+Nefertari+playing+senet.jpg>. Acesso em: 05 jan. 2015.

MAKAR, Jobe. **MacromediaFlash MX Game Design Demystified: The Official Guide to Creating Games with Flash**. Berkeley, CA: Peachpit Press, 2002.
MARINI, Joe. **Native Apps in HTML5?** San Francisco: Google, 2013. 29 slides, color.

MANUAL PDP-1. Disponível em: <http://research.microsoft.com/en-us/um/people/gbell/digital/PDP%201%20Manual%201961.pdf> Acesso em: 10 fev. 2015.

MARINO-NACHISON, David. **Ralph H. Baer, a father of video gaming, dies at 92.** 2014. Disponível em: <http://www.washingtonpost.com/national/health-science/ralph-h-baer-a-father-of-video-gaming-dies-at-92/2014/12/07/a24c8964-7e6e-11e4-8882-03cf08410beb_story.html>. Acesso em: 2 jan. 2015.

MASS: WERK. **Spacewar.** Disponível em: <http://www.masswerk.at/spacewar/> Acesso em: 10 fev. 2015.

MORATORI, Patrick Barbosa. **POR QUE UTILIZAR JOGOS EDUCATIVOS NO PROCESSO DE ENSINO APRENDIZAGEM?** 2003. 33 f. Dissertação (Mestrado) - Curso de Informática Aplicada à Educação, Universidade Federaldo Rio de Janeiro, Rio de Janeiro, 2003.

NAYAK, MALATHI, **A look at the $66 billion video-games industry**. Reuters, Jun. 2013. Disponível em: <http://in.reuters.com/article/2013/06/10/gameshow-e-idINDEE9590DW20130610>. Acesso em: 14, fev. 2015.

PUBLICATIONS. **Valve Corporation**. Disponível em: <http://www.valvesoftware.com/company/publications.html> Acesso em: 10 fev. 2015.

RAGNARSSON, Ólafur Andri. **Importance of Design Patterns and Frameworks for Software Development**. Tölvumál Magazine: 2007. Disponível em: <http://www.olafurandri.com/papers/Importance%20of%20Design%20Patterns%20and%20Frameworks%20for%20Software%20Development.pdf> Acesso em: 15 fev. 2015.

RALPH Baer and Bill Harrison testing the Brown Box [1969]. 2011. (159 min.), VHS, son., P&B. Original de 1969. Disponível em: <https://www.youtube.com/watch?v=scaLx1l_j5w>. Acesso em: 15 jan. 2015.

RIEHLE, Dirk. **Framework Design:** A Role Modeling Approach. 1999. 229 f. Tese (Doutorado) - Curso de Technical Sciences, Swiss Federal Institute Of Technology Zurich, Hamburg, 2000.

RISHE, Patrick. **Trends in the Multi-Billion Dollar Video Game Industry: Q/A with Gaming Champ Fatal1ty**. Forbes**,** USA, p.1-3, 23 nov. 2011. Mensal. Disponível em: <http://www.forbes.com/sites/prishe/2011/12/23/trends-in-the-multi-billion-dollar-video-game-industry-qa-with-gaming-champ-fatal1ty/>. Acesso em: xx mes. 2014.

SALEN, Katie; ZIMMERMAN, Eric. **Fundamentos Do Design De Jogos.** São Paulo: Blucher, 2012. Volume 1.

SCHELL, Jesse. **A Arte de Game Design:** O Livro Oficial. São Paulo: Elsevier, 2011.

SCHMIDT, Douglas C. et al, **Frameworks: Why They Are Important and How to Apply**
Them Effectively. ACM Queue magazine. Volume 2, Number 5: 2004. Disponível em: <http://www.dre.vanderbilt.edu/~schmidt/PDF/Queue-04.pdf> Acesso em: 10 fev. 2015.

SHEA, Cam. **AL ALCORN INTERVIEW:** The creator of Pong on the birth of Atari, holographic gaming and being paid to not show up to work.. 2008. Disponível em: <http://www.ign.com/articles/2008/03/11/al-alcorn-interview>. Acesso em: 8 fev. 2015.

THE ATARI MUSEUM. **Pong**: **for your home tv**. Disponível em: <http://www.atarimuseum.com/videogames/dedicated/homepong.html>. Acesso em: 10 fev. 2015.

USA, TEXAS. DON FUSSELL. (Comp.). **CS 378: Computer Game Technology.** Austin: University Of Texas At Austin, 2012. 20 slides, color.
USA. Creative Commons. Wikipedia (Comp.). Magnavox Odyssey. 2011. Disponível em: <http://en.wikipedia.org/wiki/Magnavox_Odyssey#mediaviewer/File:Magnavox-Odyssey-Console-Set.jpg>. Acesso em: 05 jan. 2015.

USA. Creative Commons. Wikipedia (Comp.). Pong. 2011. Disponível em: <http://pt.wikipedia.org/wiki/Pong#mediaviewer/File:PongVideoGameCabinet.jpg>. Acesso em: 03 jan. 2015.

Valente, J.A. (1993a). **Diferentes Usos do Computador na Educação**. Em J.A. Valente (Org.), Computadores e Conhecimento: repensando a educação (pp.1-23). Campinas, SP: Gráfica da UNICAMP.

Valente, J.A. (1993b). **Por Quê o Computador na Educação**. Em J.A. Valente (Org.), Computadores e Conhecimento: repensando a educação (pp. 24-44). Campinas, SP: Gráfica da UNICAMP.

WHAT is a game framework versus a game engine? 2012. Disponível em: <http://gamedev.stackexchange.com/questions/31772/what-is-a-game-framework-versus-a-game-engine>. Acesso em: 8 fev. 2015.

WHAT'S the difference between an “engine” and a “framework”? 2011. Disponível em: <http://stackoverflow.com/questions/5068992/whats-the-difference-between-an-engine-and-a-framework>. Acesso em: 16 fev. 2015.

Wenger, E. (1987) **Artificial Intelligence and Tutoring System: Computational and Cognitive Approaches to the Communication of Knowledge**. Califórnia: Morgan Kaufmann Publishers.

WINTER, David. **Atari PONG: The Home Systems**. 2013. Disponível em: <http://www.pong-story.com/atpong2.htm>. Acesso em: 12 fev. 2015.