MB Engine: Game Engine para a Construção de Jogos em HTML 5

Maik Basso[1], Carine Piovesan Lopes[1],

Fábio José Parreira[2], Sidnei Renato Silveira[2]

[1]Curso de Bacharelado em Sistemas de Informação – [2]Departamento de Tecnologia da Informação - Universidade Federal de Santa Maria (UFSM) – Centro de Educação Superior Norte RS (CESNORS) – Frederico Westphalen – RS – Brasil
maik@maikbasso.com.br, carinepiovesan@gmail.com,
fabiojparreira@gmail.com, sidneirenato.silveira@gmail.com

Resumo. *Este artigo apresenta o desenvolvimento de uma Game Engine (motor de jogos digitais) utilizando as tecnologias Javascript e HTML 5 com o intuito de facilitar a construção de jogos baseados em um contexto 2D (duas dimensões). A engine proposta neste artigo dispõe de recursos para a criação de interfaces, tais como elementos gráficos e sonoros, organizando o conteúdo e apresentando uma lógica simples para a criação de jogos digitais.*

Palavras chaves: *Jogos Digitais, Tecnologias Web, Game Engine.*

Abstract. *This paper presents the development of a Game Engine (digital game engine) using JavaScript and HTML 5 technologies in order to facilitate the construction of games based on a 2D context (two dimensions). The engine proposed in this paper has the resources to create interfaces such as graphics and sound elements, organizing content and featuring a simple logic to create digital games.*

Keywords: *Digital Games, Web Technologies, Game Engine.*

1. Introdução

Este artigo apresenta o desenvolvimento de um protótipo de *game engine*, com o objetivo de facilitar a construção de jogos digitais baseados em duas dimensões (2D), fornecendo uma lógica simples para a criação das interfaces e lógicas de jogo.

A motivação para a construção da "*MB Engine*" (nome proposto para a *engine*) surgiu a partir do momento em que um dos autores deste trabalho necessitou desenvolver um jogo educacional digital como trabalho de conclusão do curso (TGSI – Trabalho de Graduação em Sistemas de Informação) de Bacharelado em Sistemas de Informação na UFSM/Frederico Westphalen. A *MB Engine* está sendo construída com a utilização de tecnologias *web*, tais como HTML (*HyperText Markup Language*) na sua quinta versão – HTML 5, CSS (*Cascading Style Sheets*) versão 3 e a linguagem de programação *Javascript*. O conceito e a funcionalidade de cada uma destas tecnologias serão explorados no decorrer deste artigo.

Neste contexto, o presente artigo encontra-se dividido da seguinte forma: a seção 2 apresenta um breve referencial teórico envolvendo as áreas de estudo relacionadas ao desenvolvimento do protótipo de *engine*. A seção 3 apresenta alguns trabalhos relacionados, onde são descritos e comparados trabalhos que envolvem o desenvolvimento de motores de jogos. A seção 4 apresenta o processo de desenvolvimento da *MB Engine*, sendo analisadas as tecnologias e ferramentas usadas para a implantação do protótipo. Encerrando o artigo são apresentadas as considerações

finais, destacando os resultados obtidos até o presente momento, bem como as referências utilizadas.

2. Referencial Teórico

Esta seção apresenta um referencial teórico sobre as áreas envolvidas no desenvolvimento deste trabalho, destacando conceitos referentes a jogos digitais, a conceituação de *game engine* e sua aplicação no desenvolvimento de jogos digitais.

2.1. Jogos Digitais

Um jogo digital é uma atividade lúdica, formada por ações e decisões que resultam em uma condição final, que representa ganhar ou perder o jogo. Tais ações e decisões são limitadas por um conjunto de regras e por um universo que, no contexto dos jogos digitais, são regidos por um programa de computador. O universo contextualiza as ações e decisões do jogador, fornecendo a ambientação adequada à narrativa do jogo, enquanto as regras definem o que pode e o que não pode ser realizado, bem como as consequências das ações e decisões do jogador. Além disso, as regras fornecem desafios a fim de dificultar ou impedir o jogador de alcançar os objetivos estabelecidos (SCHUYTEMA, 2008).

Segundo Battaiola (2000), um jogo digital é composto, basicamente, por três partes: enredo, motor e interface interativa. O enredo define o tema, a trama, os objetivos do jogo e a sequência com a qual os acontecimentos surgem. O motor do jogo (*engine*) é o mecanismo que controla a reação do ambiente às ações e decisões do jogador, efetuando as alterações de estado neste ambiente. Por fim, a interface interativa permite a comunicação entre o jogador e o motor do jogo, fornecendo um caminho de entrada para as ações do jogador e um caminho de saída para as respostas audiovisuais referentes às mudanças do estado do ambiente.

Conforme Juul (2001), o fato de mundos fictícios existirem é a principal característica que distingue os jogos digitais dos não-digitais, ressaltando que a existência de mundos fictícios deve-se à existência de um mundo lúdico único onde o jogo se desenvolve. Nos jogos não-digitais acaba surgindo um mundo fictício, mas esse fica limitado ao imaginário de cada participante e não é compartilhado e delimitado como nos jogos digitais.

2.2. *Game Engine*

A *engine* ou motor de jogo é responsável por simular a parte física do mundo real dentro do ambiente do jogo. O nível de complexidade da *engine* define também o nível de realidade que a mesma proporciona ao usuário (BRITO, 2011).

Os motores podem ser considerados como bibliotecas de desenvolvimento, responsáveis pelo gerenciamento de diversos componentes do jogo tais como imagens, entrada de dados e outras funções (EBERLY, 2001).

O objetivo de um motor de jogos é agrupar funções fundamentais para o desenvolvimento de jogos, que podem se estender da interação com os periféricos de entrada até a renderização[5] dos cenários e personagens. Assim, várias aplicações podem ser desenvolvidas utilizando como base de código este componente central. Isto certamente reduz o tempo total de produção, à medida que concentra a equipe de trabalho em atividades de mais alto nível. Por mais genéricos que sejam, entretanto, os

[5] Renderizar é o ato de construir as imagens com utilização de contextos no *canvas* (MDN, 2015)

motores costumam ser projetados tendo em vista uma classe particular de jogos, como 2D ou 3D (OLIVEIRA, 2013).

3. Estado da Arte

Existem diversos estudos focados no desenvolvimento de *game engines* para auxiliar e acelerar o trabalho dos desenvolvedores de jogos. As *game engines* atuais estão cada vez mais completas e com recursos surpreendentes, incluindo aspectos físicos, visuais e sonoros, sem contar o suporte multiplataforma e responsividade, entre outros recursos.

Um exemplo de *game engine* de código aberto que trabalha com um contexto 2D é o *melonJS*. Esta *engine* possui um formato de mapa popular, permitindo projetar facilmente níveis usando um editor de mapas, proporcionando ao desenvolvedor mais tempo para implementar as características do jogo a ser desenvolvido. A construção de jogos nesta *engine* baseia-se na utilização recursos modernos e sem dependências de outras bibliotecas. Esta *engine* conta é compatível com diversos navegadores *web* do mercado, incluindo os de plataforma *mobile* (MELONJS.ORG, 2015).

Outro projeto de código aberto, disponível na Internet, é o *EaselJS* que, segundo a comunidade que mantém o projeto, é uma biblioteca de alta performance para o desenvolvimento de conteúdo 2D interativo em HTML5. Esta *engine* faz parte do pacote *CreateJS* e fornece uma lista de exibição rica em recursos que permitem manipular e animar gráficos. A *engine* também fornece um modelo interativo para manipulação do mouse e também para a utilização de telas de toque. Como a maioria das *engines*, esta *engine* não possui nenhuma dependência externa. Ela se destaca por ser uma *engine Javascript* que proporciona suporte não somente à criação de jogos, mas também anúncios publicitários, arte generativa e visualização de dados, entre outras possibilidades (CREATEJS.COM, 2015).

O projeto *Phaser* por sua vez, é um dos projetos de *game engine* mais bem conceituados e completos do mercado atual. Além de ser baseado em *software* livre, é mantido por uma comunidade que disponibiliza atualizações constantes. O projeto suporta todos os recursos necessários para a criação de jogos incluindo animações, pré-carregamento de arquivos, sons, métodos de entrada entre outros. Além disso, a equipe que mantém o projeto *Phaser* disponibiliza alguns produtos e serviços, tais como livros e cursos de apoio ao aprendizado de desenvolvimento de *games*. Outro ponto relevante atualmente é a compatibilidade com a maioria dos navegadores *web* do mercado (PHOTONSTORM, 2015).

Observando os recursos disponibilizados pelas *engines* apresentadas neste artigo, destaca-se que a *MB Engine* é um projeto que tende a se estender conforme as necessidades que vão surgindo, visando contemplar diferentes aspectos que já estão em funcionamento em outras *engines*. Mesmo tendo um número de recursos reduzido, a *MB Engine* se destaca pelo fato de ser uma *game engine* que dispõe de estrutura multi-canvas e pelo fato de ser uma *engine responsiva, e*ssas duas propriedades não foram identificadas nas *engines* analisadas. Serão descritas na próxima sessão deste artigo as duas propriedades existentes na *MB Engine*.

4. Solução Implementada

A *MB Engine* foi desenvolvida com o intuito de auxiliar no desenvolvimento de jogos digitais baseados em um contexto 2D (bi-dimensional). O projeto foi desenvolvido com a utilização de tecnologias *web* tais como HTML5, CSS3 e *Javascript*. A presente *engine* dispõe de recursos ainda considerados básicos para a construção de jogos, porém, funciona de forma simples e intuitiva, auxiliando na construção das estruturas do jogo e também na sua lógica.

O processo de desenvolvimento da *engine* está sendo realizado de forma paralela ao desenvolvimento do protótipo de um jogo educacional digital, que está sendo desenvolvido como TGSI na UFSM/Frederico Westphalen (BASSO et. al., 2015). Desta forma, sempre que um novo recurso é necessário no protótipo do TGSI, este recurso é adicionado à *MB Engine*.

4.1. Tecnologias Empregadas

Como mencionado anteriormente, a *game engine* está sendo desenvolvida utilizando tecnologias *web* de última geração. O HTML5 contém diversos recursos essenciais que possibilitam a construção de jogos digitais. O *canvas,* um dos recursos do HTML5, considerado por Meyer (2011) um dos mais poderosos da linguagem, também é aplicado na *game engine*.

O *canvas* possibilita a construção de imagens por meio de um contexto de duas dimensões (*2D*). Estas imagens, denominadas *frames*, podem ser modificadas dinamicamente sendo controladas por tempo ou eventos ocorridos no elemento. O elemento *canvas* permite que *scripts*[6] modifiquem a tela *bitmap*[7] formando animações. O fator determinante é a resolução, ou seja, as medidas de altura e largura que definem o tamanho do *bitmap* do *canvas*. A utilização do elemento ainda é altamente recomendada para a renderização de gráficos do jogo, ou outras imagens visuais em tempo real (MEYER, 2011; SHANKAR, 2012).

O HTML5 auxilia na definição da estrutura do jogo ou projeto. Sendo assim, tem-se a necessidade de utilização de outras duas linguagens fundamentais em projetos *web*, o CSS para compor os estilos dos elementos do jogo e a linguagem *Javascript*, para a produção dos *scripts* que comporão a lógica de jogo, controle de tempo e espaço, validação das respostas, entre outros aspectos. É importante ressaltar que, como o elemento *canvas* não permite a utilização direta de folhas de estilos em cascata (CSS) em seus elementos internos, esta será feita por meio da utilização de *Javascript* (SHANKAR, 2012).

O *Javascript* é uma linguagem de programação leve, interpretada pelos *browsers* (navegadores *web*). Por meio desta linguagem podem ser manipulados quaisquer elementos de um documento HTML, definindo estilos, mudando suas propriedades, definindo ações, entre outros aspectos (FLANAGAN, 2004).

4.2. Estrutura e Funcionamento da *Engine*

A *engine* desenvolvida apresenta uma estrutura simples, composta de métodos, funções e eventos implementados em *Javascript*. Basicamente um jogo construído com a *MB Engine* possui quatro arquivos: o primeiro arquivo é o "index.html", ele é o arquivo HTML onde são feitas as chamadas de todos os arquivos *Javascript* de forma assíncrona, ou seja, os arquivos são carregados em paralelo para acelerar o processo de *onload* da página.

Os outros três arquivos são arquivos *Javascript* organizados da seguinte maneira: o primeiro contém a função "*function Arquivos(){}*". Neste arquivo devem-se adicionar vetores contendo os *links*/referências aos arquivos multimídia que a *engine* deve carregar para a construção do jogo. O segundo arquivo contém a lógica do jogo, tal

[6] *Script* é um código composto por uma sequência de passos durante a execução de um programa (MSDN, 2015a)

[7] Um bitmap é uma matriz de bits que especifica a cor de cada pixel em uma matriz retangular de pixels. O número de bits dedicados a um pixel individual determina o número de cores que podem ser atribuídos a esse pixel (MSDN, 2015b)

como sua estrutura. Neste arquivo são criadas as interfaces do jogo seguindo o pressuposto de que cada função seja considerada uma interface. Dentro destas funções principais deve-se ter dois métodos obrigatórios que são o "renderizar();" e o "destruir();", sendo que o primeiro é responsável por montar e apresentar a interface ao usuário e, o segundo, por remover a interface da tela. A tela principal de um jogo desenvolvido com a *MB Engine* deve ser nomeada de "Principal();". A partir desta tela principal todas as outras telas são chamadas, construindo-se a lógica do jogo.

O último arquivo é o da *MB Engine* que, por sua vez, funciona da seguinte forma: quando o evento "*onload*" da página HTML for disparado, a *MB Engine* cria um novo contexto de jogo, chamando a tela que faz o carregamento de todos os arquivos multimídia contidos no *script* descrito acima. Posteriormente, a tela *Principal* é criada e o método *Renderizar* é disparado. Após isso o jogo se dá por iniciado. A Figura 1 apresenta o funcionamento da *MB Engine*.

A *MB Engine* também possui funções matemáticas baseadas no cálculo da regra de três para que seja possível aplicar o conceito de responsividade, adaptando os jogos aos diversos tamanhos de *displays* disponíveis no mercado. O conceito de responsividade está ligado ao ato de adaptar o *layout* a qualquer dispositivo, tela e resolução, com objetivo de garantir a boa experiência do usuário, possibilitando navegação e leitura confortáveis sem comprometer o conteúdo (SILVA, 2014).

Outra parte muito importante da estrutura da *engine* é a disponibilização de esqueletos de elementos gráficos e sonoros para o jogo. Por exemplo, pode-se criar um retângulo na tela instanciando um objeto contido na classe "var retangulo = new ElementosGraficos().retangulo;"; então seu método renderizar pode ser chamado "retangulo.renderizar();". A mesma lógica de utilização da *engine* pode ser aplicada aos sons do jogo.

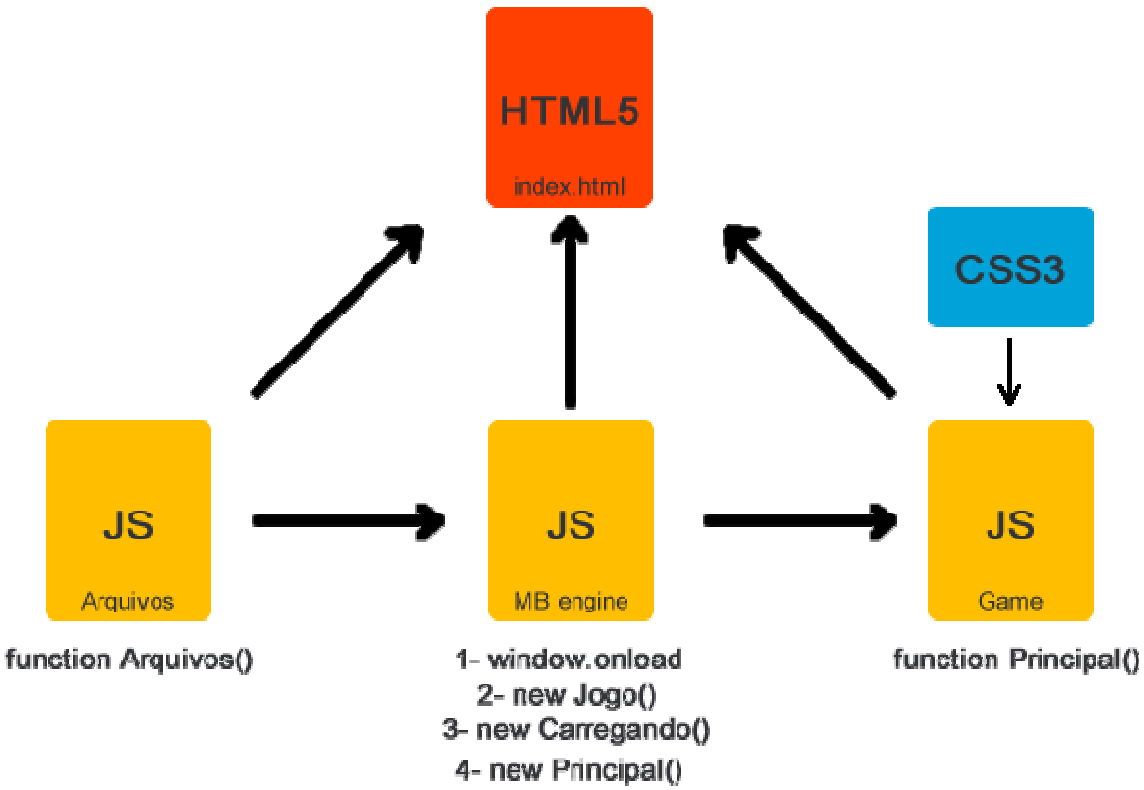

Figura 1. Funcionamento da *MB Engine*. Fonte: dos autores.

Neste contexto, o importante a ressaltar é que a *MB Engine* é *multi-canvas*, ou seja, a cada elemento gráfico criado, um novo *canvas* é criado, dando liberdade e independência na construção das interfaces, sendo que os elementos da interface podem ser manipulados separadamente.

5. Considerações Finais

O desenvolvimento da *engine* está ocorrendo de forma paralela ao desenvolvimento de um jogo educacional digital (BASSO et. al., 2015), no qual as funcionalidades da *engine* estão sendo colocadas em prática, o que possibilita que aprimoramentos sejam feitos constantemente. A Figura 2 apresenta a tela inicial do jogo *Aprendendo com o Zag* (BASSO et. al., 2015), desenvolvido a partir da *MB Engine*.

Figura 2. Tela Inicial do Jogo *Aprendendo com o Zag* Fonte: dos autores.

Algumas dificuldades estão presentes no projeto. Uma delas é a falta de suporte a algumas funcionalidades do HTML5 por parte de alguns dispositivos móveis, o que impossibilita a execução da *engine* em dispositivos móveis com versão de *Android* anterior a 5.0. Por outro lado, a *engine* apresenta um conceito ágil e flexível, proporcionando liberdade ao desenvolvedor, para construir sua estória de jogo de forma simples e intuitiva, em um curto prazo de tempo.

O próximo passo a ser realizado referente à implementação da *engine* envolve a análise e implementação das recomendações de acessibilidade, para que a mesma fique disponível à utilização para o maior número possível de usuários independente de suas limitações e habilidades.

Referências

BATTAIOLA, A. L. (2000). Jogos por computador: Histórico, relevância tecnológica e mercadológica, tendências e técnicas de implementação. Anais do XIX Jornada de Atualização em Informática. Curitiba: SBC 2000.

BASSO, M.; KLISZCZ, S.; PARREIRA, F. J.; SILVEIRA, S. R. (2015). Desenvolvimento de um Jogo Educacional Digital para Auxílio à Alfabetização

utilizando Redes Neurais. UFSM: Frederico Westphalen, 2015. TGSI – Trabalho de Graduação em Sistemas de Informação.

BRITO, (2011). Blender 3D: jogos e animações interativas. São Paulo: Novatec.

CREATEJS.COM (2015). CreateJS. Disponível em: <http://createjs.com>. Acesso sem setembro de 2015.

EBERLY, D. H. (2001). 3D game engine design: a practical approach to real-time computer graphics. São Francisco: Morgan Kaufmann.

FLANAGAN, David. JavaScript: o guia definitivo. 4. ed. Porto Alegre: Bookman, 2004.

JUUL, J. (2001). Half-Real: Video Games between Real Rules and Fictional Worlds. 1º ed. Cambridge: The MIT Press.

MDN Mozilla Developer Network (2015). Utilização Básica do Canvas. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/Guide/HTML/Canvas_tutorial/Utilizacao_basica>. Acesso em setembro de 2015.

MELONJS.ORG. (2015) melonJS: a lightweight HTML5 game engine. Disponível em: <http://melonjs.org>. Acesso em setembro de 2015.

MEYER, J. (2011). O Guia Essencial do HTML5: Usando jogos para aprender HML5 e Javascript. Rio de Janeiro: Ciência Moderna.

MSDN Microsoft Developer Network (2015a). JavaScript: princípios básicos. Disponível em: <https://msdn.microsoft.com/pt-br/library/6974wx4d%28v=vs.94%29.aspx>. Acesso em setembro de 2015.

MSDN Microsoft Developer Network (2015b). Tipos de Bitmaps. Disponível em: <https://msdn.microsoft.com/pt-br/library/at62haz6%28v=VS.110%29.aspx>. Acesso em setembro de 2015.

OLIVEIRA, E. R. (2013). O Uso de Engines para o Desenvolvimento de Jogos Eletrônicos. Monografia de conclusão de curso apresentada à Universidade Estadual do Sudoeste da Bahia – UESB. Vitória da Conquista, 2013.

PHOTONSTORM (2015). Phaser: desktop and mobile HTML5 game framework. Disponível em: <http://phaser.io/>. Acesso em setembro de 2015.

SCHUYTEMA, P. (2008). Design de games: uma abordagem prática. São Paulo: Cengage Learning.

SHANKAR, A. R. (2012). Pro HTML5 Games (Expert's Voice in Web Development). E-book: Apress, Disponível em: <http://www.apress.com/9781430247104>. Acesso em 17 de maio de 2015.

SILVA, Arthur De Almeida Pereira Da; Design Responsivo: Técnicas, Frameworks e Ferramentas. Universidade Federal Do Rio De Janeiro. Centro de Ciências Exatas e Tecnologias. Escola de informática Aplicada. Rio de Janeiro, 2014. Disponível em: <http://bsi.uniriotec.br/tcc/201412Almeida.pdf>. Acesso em 10 setembro de 2015.