UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA - UESB
DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS - DCET
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

O USO DE ENGINES PARA O DESENVOLVIMENTO DE JOGOS ELETRÔNICOS

Esdras Rocha de Oliveira

VITÓRIA DA CONQUISTA
2013

ESDRAS ROCHA DE OLIVEIRA

O USO DE ENGINES PARA O DESENVOLVIMENTO DE JOGOS ELETRÔNICOS

Monografia de conclusão de curso apresentada à Universidade Estadual do Sudoeste da Bahia – UESB – como pré-requisito do título de Bacharel em Ciência da Computação.

Área de Concentração: Jogos de Computador.

Orientador: Prof. Dr. Roque Mendes Prado Trindade

VITÓRIA DA CONQUISTA

2013

ESDRAS ROCHA DE OLIVEIRA

O USO DE ENGINES PARA O DESENVOLVIMENTO DE JOGOS ELETRÔNICOS

Banca Examinadora:

_____

Prof. Dr. Roque Mendes Prado Trindade

_____

Prof.ª Dr.ª Alzira Ferreira da Silva

_____

Prof. Esp. Fabrício de Sousa Pinto

Dedico aos meus pais, José Marcos e Maria Lúcia, pelo apoio, força, incentivo, conselhos e compreensão.

AGRADECIMENTOS

Sou grato a Deus por mais esta conquista, e todas as vitórias que construíram o caminho até aqui.

Agradeço à minha mãe Maria Lúcia, por todo apoio, compreensão, carinho e amor que só ela soube me dar. Também ao meu pai José Marcos, que por muitas vezes me auxiliou a encontrar a saída de muitos problemas.

À minha irmã Elisama Rocha de Oliveira, pois sem ela eu não poderia estar aqui hoje.

Não menos importante, aos demais familiares que ao longo do tempo me foram como um suporte, de onde eu jamais cairia e sempre poderia contar.

Aos meus mestres do curso de Bacharelado em Ciência da Computação, que ao longo desses anos me incentivaram a buscar o conhecimento e me apontavam o caminho.

Tenho muito a agradecer à Prof.ª Cátia Mesquita, à Prof.ª Maísa Soares e à Celina Pereira que em momentos difíceis elas não se omitiram.

Ao meu orientador Prof. Dr. Roque Mendes, que sempre acreditou nas minhas idéias por mais malucas que fossem e me orientou neste trabalho.

À minha namorada Viviane Lima, que me apoiou e incentivou durante esta etapa muito importante da minha vida.

Ao meu amigo Henrique Júnior, que por muitas vezes me ajudou a superar os nada poucos obstáculos do curso de graduação.

“O futuro pertence àqueles que acreditam na beleza de seus sonhos.”

Elleanor Roosevelt

“Eu acredito demais na sorte. E tenho constatado que, quanto mais duro eu trabalho, mais sorte eu tenho.”

Thomas Jefferson

“O esforço de dois ou mais indivíduos trabalhando com unidade é maior do que a soma dos esforços dos indivíduos trabalhando individualmente”.

William B. Cornell

RESUMO

O desenvolvimento de jogos eletrônicos utilizando motores prontos permite facilitar a reutilização de software aumentando assim a produtividade, mantendo o foco diretamente para as regras, funcionalidades, características e fundamentos do produto – o jogo. Este trabalho teve como objetivo realizar um breve estudo do motor Game-Editor®, aplicado ao desenvolvimento de jogos educativos. Por fim, foi feito um estudo de caso criando uma fase do jogo “Em Busca da Fórmula Perdida”, utilizando alguns recursos do motor de jogo.

Palavras-chave: Motores de Jogos, Desenvolvimento de Jogos, Jogos na Educação e Computação Gráfica.

ABSTRACT

The development of electronic games using engines ready become so easy the reuse of software thereby increasing productivity, keeping the focus directly to the rules, features, characteristics and elements of the product - the game. This work had as objective to make a brief study of motor-Game Editor ®, applied to the development of educational games. Finally, we made a case study of creating a stage of the game "In Search of Lost Formula" using some features of the game engine.

Keywords: Engines Games, Game Development, Games in Education and Computer Graphics.

LISTA DE ABREVIATURAS E SIGLAS

2D – Duas dimensões

3D – Três dimensões.

ACM (*Association for Computing Machinery*) – Associação para Maquinaria de Computação.

API (A*pplication Programming Interface*) – Interfaces de Programação de Aplicativos.

ARB (*OpenGl Architecture Review Board*) – Conjunto de empresas que supervisionavam o OpenGL.

CPU (*Central Processing Unit*) – Unidade Central de Processamento.

IA Inteligência Artificial.

PC (*Personal Computer*) – Computador pessoal.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

O que segue nesta obra trata de jogos eletrônicos em um contexto de desenvolvimento, seu conteúdo contendo aborda um pouco de construção de jogos eletrônicos. É proposta a utilização de mecanismos que facilitem o desenvolvimento destes jogos mesmo para quem não seja especialista na área. Foi feito ainda um estudo de caso para mostrar o uso de um destes mecanismos aplicado à educação de crianças, no ensino específico da matemática.

O presente trabalho tem por objetivo destacar a utilização de motores durante a fase de desenvolvimento de jogos, visando à simplificação do processo criativo deste, que, pode ser visto como uma das mais valiosas ferramentas atuais de interação social, aumento criativo e intelectual. Falando especificamente sobre a Educação Infantil, segundo a Coordenação-Geral de Estudos e Pesquisas da Educação Fundamental (1997), nós temos um campo onde pequenos jogos eletrônicos têm uma grande importância na fase básica, pois os jogos estimulam o processo cognitivo, ou seja, auxiliam no desenvolvimento da percepção, atenção, raciocínio e imaginação da criança. Vale ressaltar que se o desenvolvimento do jogo estiver voltado para a área educativa, a aplicabilidade de seus resultados terá um maior nível, e com base nessa afirmação, desenvolveremos uma versão demonstrativa, através do uso de um motor de jogos, direcionada para o ensino das funções matemática básicas.

Buscou-se conceituar e definir engenharia de software, abordando padrões de projeto e metodologias de desenvolvimento, a fim de produzir o software com os padrões e recomendações da engenharia de software e formas ágeis de produção. Teremos ainda, conceitos de computação gráfica utilizado para funções do jogo como mudança de posição do personagem, entre outras ações do jogo. Apresentamos no apêndice A, a descrição e o documento completo do jogo, onde temos todas as características fundamentais do jogo que servirá para a produção do estudo de caso.

1.1. Objetivo

1.1.1. Objetivo Geral

Evidenciar como são os motores de jogos 2D na atualidade e como eles revelam-se mecanismo de agilidade na produção e desenvolvimento de jogos eletrônicos.

1.1.2. Objetivos Específicos

- ✓ Estudar o funcionamento geral dos motores de jogos.
- ✓ Identificar e analisar os principais motores de jogos do segmento 2D do mercado.
- ✓ Observar as principais características de cada um dos principais motores de jogos e escolher com base nessas características um motor que se adeque ao estudo de caso.
- ✓ Realizar estudo de caso com um dos motores de jogos estudados desenvolvendo o jogo: "Em Busca da Fórmula Perdida".

1.2. Contextualização e Motivação

Os jogos sempre despertaram interesse da humanidade durante toda a história, vários estilos de jogos despertaram a curiosidade em muitas vezes solucionar um desafio ou apenas divertir-se com grupos, segundo Alves (2003).

Minha vontade sempre foi fazer meu TCC com algo relacionado à jogos eletrônicos, e tive a oportunidade de realizar esse desejo com a ajuda do meu orientador. Sempre estive envolvido com jogos de consoles, de computadores, de tabuleiro entre outros, com isso tive a ideia de quando no fim do curso fazer algo assim.

Segundo Bueno (2010), os jogos podem ser usados como fator de integração social, como auxílio a exercitar o raciocínio, originador de estratégias, educador e como diversão apenas.

Os jogos de computador por sua vez tem tido um papel de catalisador onde propicia o acesso e a difusão facilitada dos mais modernos e diversos tipos de jogos criados na atualidade. Com isso o mercado de jogos tem sido um grande alvo de investimento e pesquisa. Por exemplo, em 2003 o mercado de jogos eletrônicos

na categoria de entretenimento digital movimentou cerca de U$ 22,4 bilhões somente com a comercialização de jogos, e se formos considerar o hardware e acessórios utilizados para esta finalidade a soma atinge U$ 55,5 bilhões, o que ultrapassa a indústria cinematográfica que no mesmo ano teve lucro de U$ 19 bilhões (ASSIS, 2003).

Em termos globais, segundo a empresa de pesquisa alemã *GfK Custom Research (GfK)*, o setor de games movimenta US$ 1 trilhão. Em 2012, no Brasil, a indústria de consoles vendeu um total de quase R$ 1 bilhão – o equivalente a um mercado como o russo, por exemplo – um crescimento expressivo de 43% em relação a 2011. Em software, o crescimento também foi grande: 72% a mais em faturamento, totalizando 242 milhões de Euros no mercado brasileiro, ou mais de R$ 600 milhões.

No curso de Ciência da Computação objetiva a formação teórica e prática em computação. A academia proporciona a expansão do conhecimento científico da computação e fornece fundamentos para solucionar os mais diversos problemas e a oportunidade de conhecer muitas disciplinas, nas quais obtemos conceitos que, neste caso, aplicam-se a jogos eletrônicos.

Ao longo deste trabalho iremos abordar alguns desses conceitos a fim de elucidar a forma que uma *Game Engine* trabalha, aplicar o conhecimento em um estudo de caso e abordar o uso de jogos na educação infantil como mecanismo auxiliar para o ensino e fixação do conhecimento.

1.3. Justificativa

Com base na crescente popularização da informatização, podemos observar o uso massivo de entretenimento digital, os jogos estão sendo usados dentro de casas, em aeroportos, em shopping e em todo lugar que exista tecnologia para usá-los.

Tamanha demanda nos impulsionou a estudar o jogo eletrônico como uma ferramenta de auxílio ao ensino e também, ao desenvolvimento de jogos cuja estratégia seja o auxílio à obtenção do conhecimento, de forma prazerosa e divertida como a maioria dos jogos é, ou seja, fazer com que a diversão possa ser mais proveitosa do que apenas um passatempo.

Utilizando o *Game-Editor*® pode-se obter mais agilidade e praticidade em desenvolver jogos. O uso desse motor no desenvolvimento nos impulsionou a então observar como se dá o desenvolvimento de jogos através de motores de jogos, nesse aspecto faremos um breve comentário a respeito do uso de motores na criação de jogos.

1.4. Metodologia

Com o objetivo principal de apresentar um breve estudo do motor de jogos Game-Editor®, foram pesquisadas algumas das principais motores de jogos que podem ser equiparadas a ela, em seguida observamos sua utilização. Também abordamos um pouco sobre jogos na educação, para tanto utilizamos na elaboração deste trabalho, como fontes de pesquisa, alguns meios de comunicação, dentre eles estão sites, livros, artigos e revistas.

Algumas ferramentas foram necessárias para construção de alguns objetos utilizados no estudo de caso aqui abordado, e elas foram: GIMP e Blender

O *GIMP*, que é um programa de código aberto voltado para criação e edição de imagens, e possibilita criação de desenho vetorial, o que para o propósito abordado era de grande importância dado o fato de que toda produção poderia ser alterada de acordo com o andamento do trabalho (GIMP, 2012).

O *Blender*, que é um programa de código aberto desenvolvido para modelagem, animação, texturização, composição, renderização, edição de vídeo e criação de aplicações interativas em 3D (Blender, 2012). O Blender foi utilizado como ferramenta de auxilio ao melhoramento das imagens utilizadas no jogo “Em Busca da Fórmula Perdida”

1.5. Estrutura do Documento

Este projeto monográfico se divide em seis capítulos. O capítulo 2 mostra a fundamentação teórica, sendo abordados os pontos mais importantes para o uso de desenvolvimento de jogos como ferramenta auxiliar ao processo de ensino-aprendizagem, obtendo assim uma melhor interação com conteúdo de forma mais dinâmica, segundo Alves (2003) e Bueno (2010). O capítulo 3 refere-se a evidenciar o uso de motores de jogos na atualidade como mecanismo de agilidade na produção e desenvolvimento de jogos comerciais. No capítulo 4 iremos mostrar a estrutura da

engine utilizada para desenvolver o estudo de caso deste trabalho e o trabalho propriamente dito. No capítulo 5 teremos a conclusão desta pesquisa e a proposta e descrição de alguns possíveis trabalhos futuros.

.

2. DESENVOLVIMENTO DE JOGOS ELETRÔNICOS

O desenvolvimento de jogos eletrônicos depende de conceitos predefinidos pela Engenharia de Software bem como suas diretrizes. A computação gráfica é a matéria fundamental na composição visual de um jogo eletrônico, pois, esta é a área que estuda transformação geométrica. No contexto da nossa proposta vemos também os jogos como ferramentas catalisadoras do aprendizado (Alves, 2003).

2.1. Engenharia de Software

Segundo Magalhães (2012), hoje temos várias metodologias para o desenvolvimento de um projeto de software, o autor considera alguns modelos que seriam os pais das metodologias. Ele divide os modelos de desenvolvimento de software em três grandes e principais, o desenvolvimento em Cascata, o desenvolvimento Iterativo e o desenvolvimento Incremental.

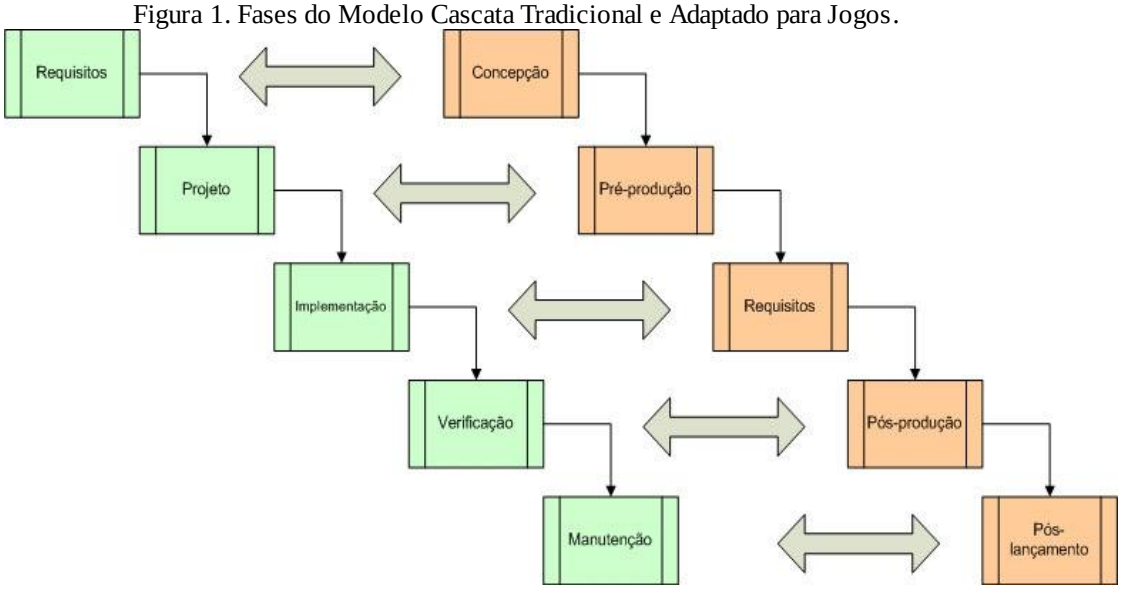
O desenvolvimento em Cascata é o mais tradicional dos três, por parecer mais simples e organizado, porém durante o desenvolvimento do projeto pode ocorrer inúmeras falhas decorrentes desse modelo, por isso vieram os métodos Iterativo e Incremental, com a ideia de substituir o modelo Cascata e acabar com as suas falhas. Magalhães (2012) ainda comenta que os modelos Iterativo e Incremental também têm as suas falhas.

Barros fala a respeito de um modelo voltado para jogos.

> *Processo Cascata para Jogos é uma versão adaptada do tradicional processo de desenvolvimento de sistemas denominado Cascata. Ele foi batizado com este nome em alusão a uma cascata, onde o fluxo de água tem um único sentido. Assim, analogamente, no modelo Cascata todas as atividades envolvidas com a produção de um sistema ocorrem de forma sequêncial* (BARROS, 2007).

Segundo Barros (2007) é possível observar na adaptação do processo em cascata para jogos uma relação direta entre as fases como pode ser observado na figura 1.

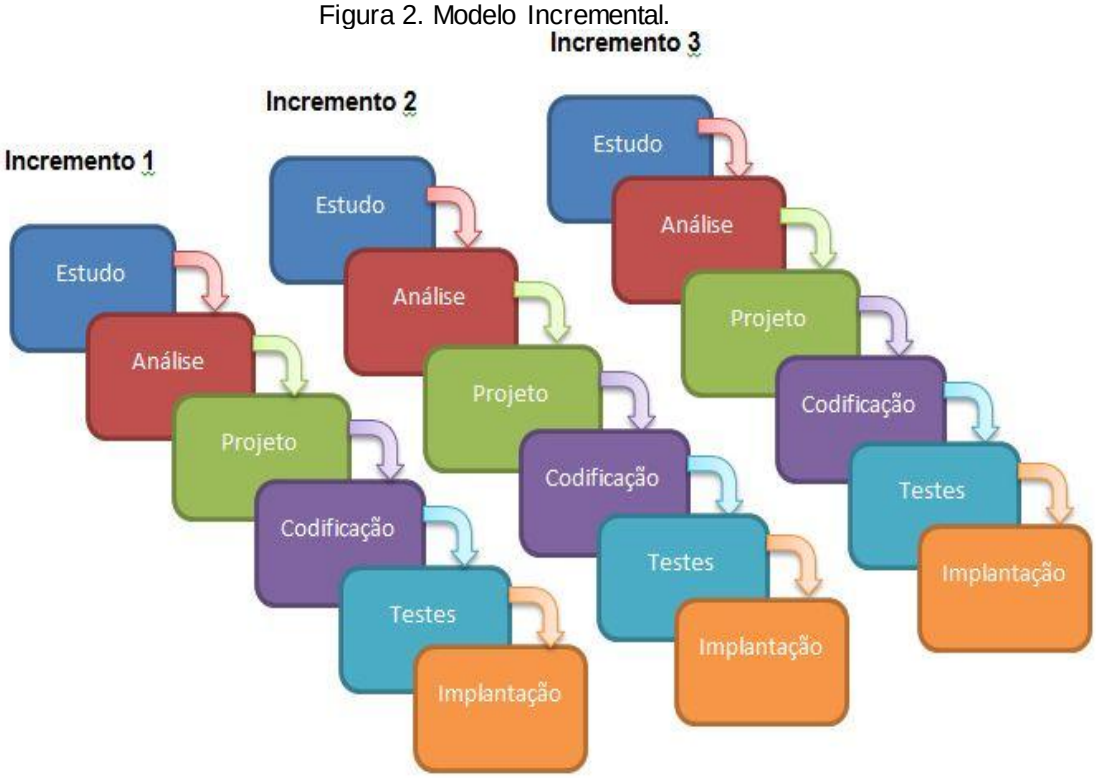
Figura 1. Fases do Modelo Cascata Tradicional e Adaptado para Jogos.



Fonte: BARROS, 2007

O desenvolvimento Incremental, figura 2, vem a ser uma estratégia de planejamento que se divide em estágios onde diferentes partes de um mesmo sistema são desenvolvidas em paralelo e após serem completadas são incorporadas ao todo.

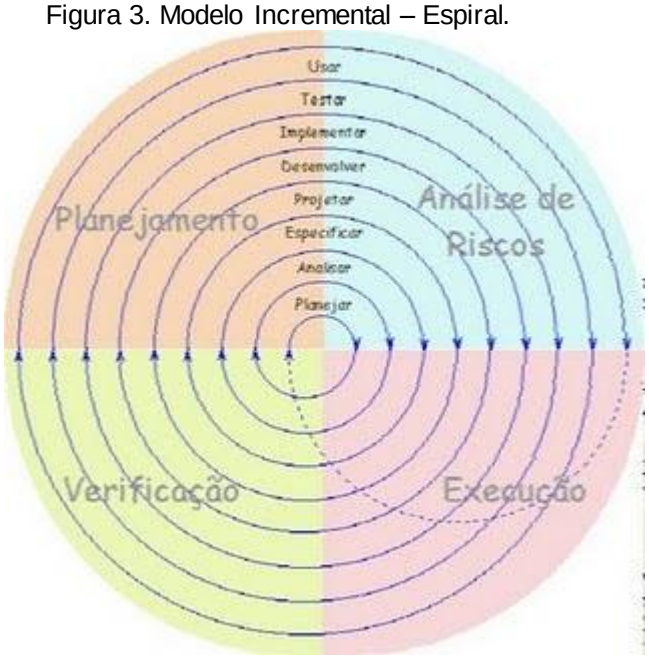
Figura 2. Modelo Incremental.



Fonte: BARROS, 2007

O desenvolvimento Iterativo, figura 3, por sua vez é a estratégia de planejamento de retrabalho onde divide o projeto em partes e elas vão sendo refeitas e em cada ciclo recebem melhorias, e estas se tornam novas versões do sistema completo (MAGALHÃES, 2012).

Figura 3. Modelo Incremental – Espiral.



Fonte: Magalhães, 2012

Existe também na Engenharia de Software a metodologia ágil, que é um conjunto de propostas para o desenvolvimento de software promovendo adaptação, fortalecimento do trabalho em equipe, auto-organização, entregas rápidas de alta qualidade e adoção de boas práticas, alinhando o desenvolvimento ágil às boas práticas de produção de software. Acima de tudo, a proposta do desenvolvimento ágil é aumentar a capacidade de criar e responder às mudanças, reconhecendo que está nas pessoas o elemento primário para guiar um projeto ao sucesso (MARCHESI et. al., 2002).

2.1.1. *Scrum*

O *Scrum* é um processo ágil que pode ser usado para gerenciar e controlar o desenvolvimento de softwares e produtos complexos[1], utilizando práticas interativas e incrementais, como pode ser visto na figura 4. Ele provê um *framework*

[1] As práticas do Scrum foram utilizadas na constituição do estudo de caso.

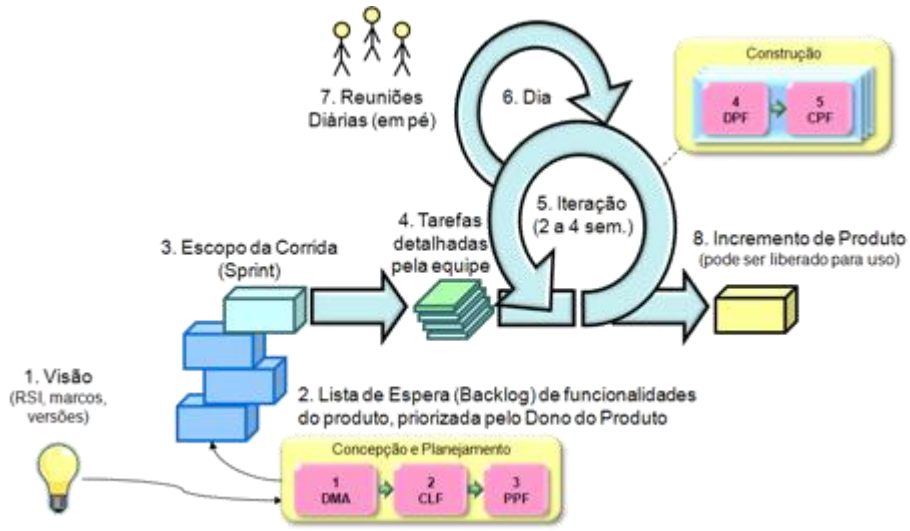
de gerenciamento de projetos que foca em ciclos curtos e completos de desenvolvimento (*Sprints*) (HIGHSMITH, 2002).

Alguns dos artefatos que são produzidos durante as fases do *Scrum*:

- **Product Backlog** - Um backlog é uma lista de itens priorizados a serem desenvolvidos para um software.
- **Sprint backlog** - O Sprint backlog é uma lista de itens selecionados do Product backlog e contém tarefas concretas que serão realizadas durante o próximo sprint para implementar tais itens selecionados.

O Scrum não é um processo prescritivo, não descreve o que deve ser feito a cada circunstância, sendo usado em trabalhos complexos, nos quais é impossível predizer todas as coisas que podem acontecer. A premissa é que vivemos em um mudo complexo, no qual não se pode planejar definitivamente o que será entregue, quando será entregue, com qual qualidade e custo (HIGHSMITH, 2002).

Figura 4. O Processo do Scrum.



Fonte: © 2012 HEPTAGON TI LTDA, 2013.

No Scrum, a equipe é responsável por desenvolver as funcionalidades, de forma auto gerenciável, auto organizável e multifuncional. A equipe também é responsável pelo sucesso de cada iteração e do projeto como um todo (PETRILLO, 2008).

2.2. Programação Orientada a Objetos

Para poder desenvolver um jogo se faz necessário criar diretivas de controle e de dinâmicas em um jogo, mas para isso é necessário manipular componentes de hardware, que são responsáveis por produzir o que lhe é ordenado pelo software.

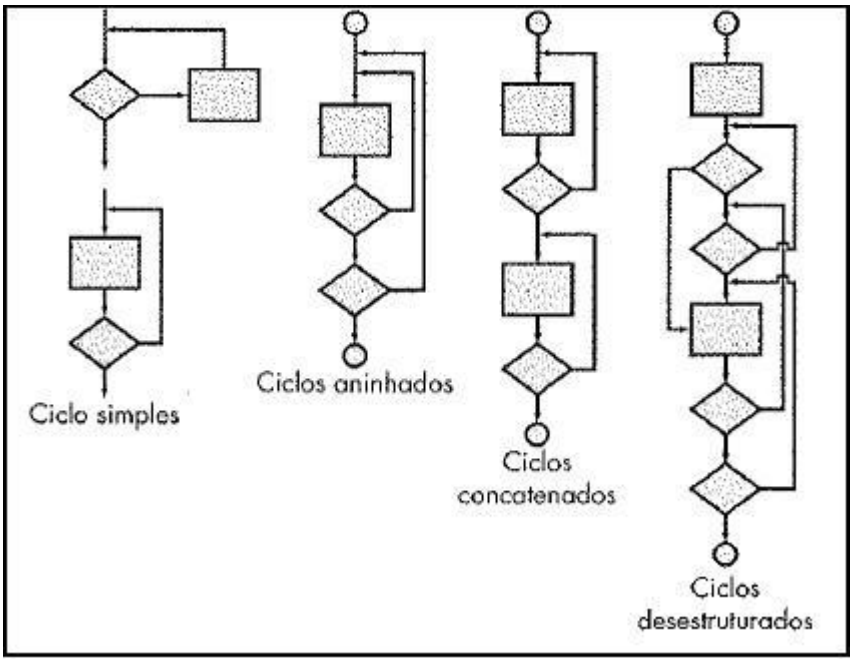
As diretivas são tomadas através de alguma linguagem de programação.

> *Uma linguagem de programação é um método padronizado para comunicar instruções para um computador. É um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador. Permite que um programador especifique precisamente sobre quais dados um computador vai atuar, como estes dados serão armazenados ou transmitidos e quais ações devem ser tomadas sob várias circunstâncias. Linguagens de programação podem ser usadas para expressar algoritmos com precisão* (Linguagem de Programação, 2012).

O projeto das linguagens de programação estruturada ou imperativa, figura 5, surgiu por volta da década de 1935 e foram assim projetadas por influência da arquitetura *von Neumann*, em um computador com essa arquitetura, tanto os dados quanto os programas são armazenados na mesma memória, e a CPU é separada da memória. Como resultado disso, a linguagem de programação tinha por obrigação modelar células de memória, obrigadas a agir de forma sequencial e de maneira iterativa de repetição, sendo a linguagem estruturada a mais eficiente nessa arquitetura (SEBESTA, 2000).

Mas, este tipo de linguagem não é adequada para os jogos atuais, já que o hardware atual permite um melhor processamento, melhor acesso à memória, placas gráficas dedicadas, paralelismo local e etc., com isso a linguagem de programação orientada ao objeto se adequou melhor ao ambiente moderno.

Figura 5. Programação Estruturada ou Imperativa.

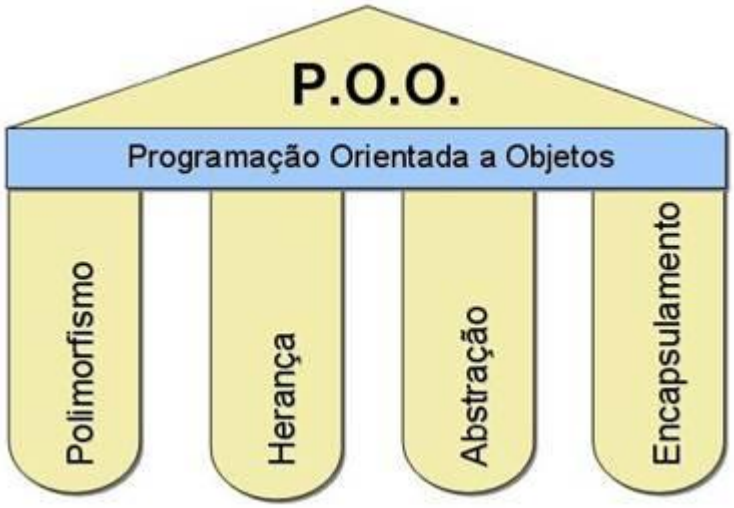


Fonte: Pressman, 2006

A orientação a objetos, figura 6, é um paradigma de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos.

Em alguns contextos, prefere-se usar modelagem orientada ao objeto, em vez de programação. De fato, o paradigma "orientação a objeto", tem bases conceituais e origem no campo de estudo da cognição, que influenciou a área de inteligência artificial e da linguística, no campo da abstração de conceitos do mundo real (SICA, 2006). Na qualidade de método de modelagem, é tida como a melhor estratégia para se eliminar o "gap semântico", dificuldade recorrente no processo de modelar o mundo real do domínio do problema em um conjunto de componentes de software que seja o mais fiel na sua representação deste domínio. Facilitaria a comunicação do profissional modelador e do usuário da área alvo, na medida em que a correlação da simbologia e conceitos abstratos do mundo real e da ferramenta de modelagem (conceitos, terminologia, símbolos, grafismo e estratégias) fosse a mais óbvia, natural e exata possível (SICA, 2006).

Figura 6. Programação Orientada a Objetos.



Fonte: Mourão, 2007.

2.3. Computação Gráfica

Computação Gráfica é a área da ciência da computação destinada ao estudo da geração, manipulação e interpretação de modelos e imagens de objetos utilizando o computador como ferramenta. Tais modelos integram várias outras disciplinas, como física, matemática, engenharia e arquitetura (TRAINA, 2003, p. 9).

No que diz respeito aos jogos, essa é a parte que os torna tão atrativos, as manipulações de modelos gráficos e as Transformações Geométricas como é o caso em jogos 2D, exemplo mostrado na figura 7, e as Sínteses de Imagens, que acontecem nos jogos 3D, como na figura 8, são as responsáveis pelas representações visuais a partir das especificações geométricas e visuais que as compõem (SILVA JÚNIOR, 2008).

Ao desenvolver um jogo algumas operações geométricas são utilizadas e suas definições matemáticas devem fazer parte do conhecimento, pois sem elas existe a possibilidade de não prevê algum evento por erro de algum cálculo ou noção para mensurar uma translação ou rotação por exemplo. Veremos na próxima seção algumas transformações geométricas.

Figura 7. Um Jogo 2D. MultiHero 0.94.

Fonte: MultiHero 2008.

Figura 8. Jogo 3D. Cryses 2.

Fonte: © 2013 Electronic Arts Inc..

2.3.1. Operações Geométricas

- **Operações Básicas 2D**

Formalmente, uma transformação de um conjunto não vazio U em um conjunto não vazio V é uma correspondência T que, a cada elemento x de U, associa um único elemento y = T(x) de V: denota-se F: U → V. O conjunto de elementos y para o qual existe um x tal que T(x) = y chama-se imagem de T. O conjunto U chama-se domínio e o conjunto V chama-se contradomínio de T (TRAINA, 2003, p. 50).

✓ **A translação**

Queremos realizar a translação de um objeto geométrico representados por um conjunto de pontos P_i pertencentes ao R^2. Para isso, adicionamos quantidades inteiras às suas coordenadas. Chamaremos estas quantidades inteiras de dx e dy. Assim, seja um ponto $P(x,y)$ sobre o qual será efetuada uma operação de translação e sejam P' as coordenadas do ponto após a translação. Podemos definir a função T como sendo $T(P) = T(x_p, y_p) = (x_p + dx, y_p + dy)$. Em forma matricial, temos que (TRAINA, 2003, p. 50):

P' = P + T, onde T = $\begin{bmatrix} dx \\ dy \end{bmatrix}$.

Não é difícil observar que, para transladar uma linha, basta transladar seus pontos limites.

✓ **Escala**

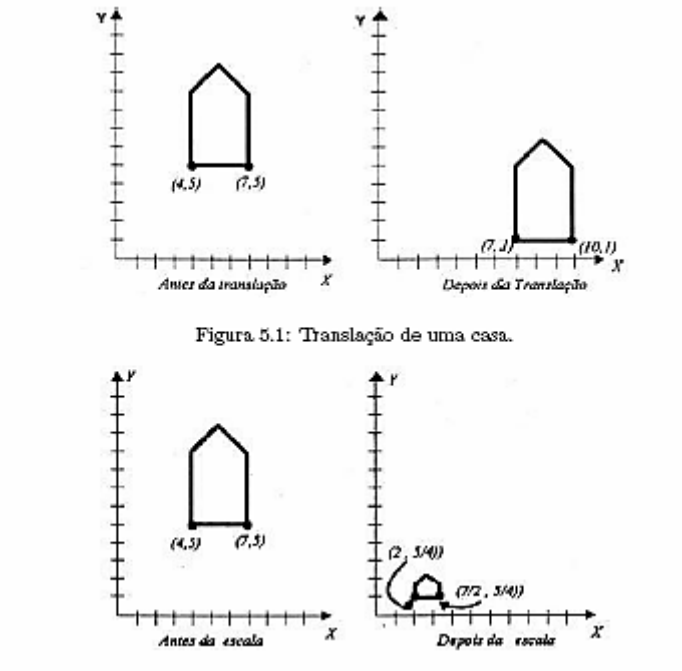
Agora queremos fazer um objeto parecer maior ou menor, ou seja, aumentar ou diminuir seu tamanho, em outras palavras, mudar sua escala, ver figura 9. Para isso, multiplicamos cada ponto P_i do objeto em questão por um fator de mudança de escala na horizontal (sx) e um fator de mudança de escala na vertical (sy). Assim, seja um ponto $P(x,y)$ sobre o qual será efetuada uma operação de translação e sejam P' as coordenadas do ponto após a translação. Podemos definir a função T como sendo $T(P) =$

$T(x_p, y_p) = (x_p * sx, y_p * \text{sy})$. Em forma matricial, temos que (TRAINA, 2003, p. 50):

$$P' = S\,P, onde\ S = \begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix}$$

Exemplo, figura 9:

Figura 9. Mudança de escala de uma casa. Como a escala é não uniforme, sua proporção é alterada.



Fonte: (Traina, 2003 pag. 50).

✓ **Rotação**

Por fim queremos rotacionar um objeto, figura 10, de certo ângulo θ com relação à origem. Assim, o ponto $T(x_p, y_p)$ é tal que:

$x_p = r * \cos\emptyset$;

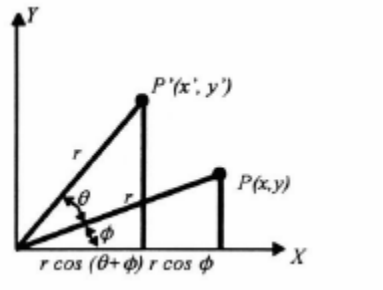
$$y_p = r * \sin\emptyset;$$

E o ponto P'(x_p', y_p') é:

$$x_p' = \text{r} * \cos(\emptyset + \theta) = \text{r} * \cos(\emptyset) * \cos(\theta) - \text{r} * \sin(\emptyset) * \sin(\theta)$$

$$y_p' = \text{r} * \sin(\emptyset + \theta) = \text{r} * \sin(\emptyset) * \cos(\theta) + \text{r} * \sin(\theta) * \cos(\emptyset)$$

Figura 10. Determinando a equação de rotação.



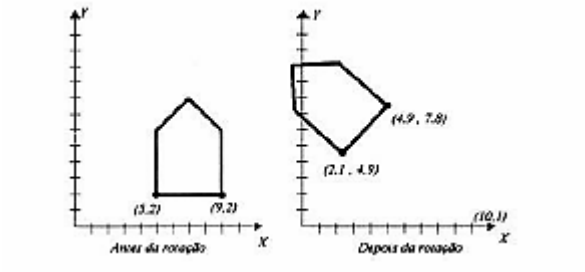
Fonte: (Traina, 2003 pag. 2).

Podemos, portanto definir a função T como sendo $T(P) = T(x_p, y_p) = (x_p * \cos\theta - y_p * \sin\theta, x_p * \sin\theta + y_p * \cos\theta)$. Em forma matricial, temos que:

$$P' = R\,P, onde\ R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Exemplo, figura 11:

Figura 11. Rotação de 45º de uma figura geométrica.



Fonte: (Traina, 2003 pag. 52).

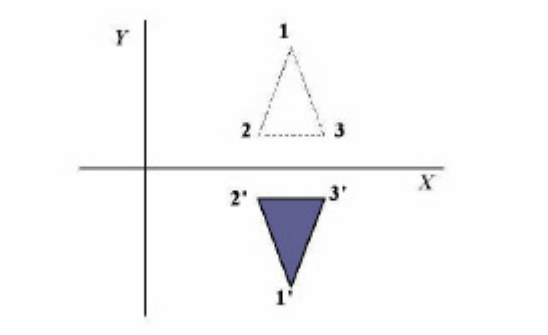
- **Espelhamento**

A transformação de reflexão, ou espelhamento, aplicada a um objeto, produz um objeto espelhado com relação a um dos eixos ou a ambos. A operação de espelhamento no $eixo\ x$, em coordenadas homogêneas, fica na forma: $T(xp, yp, 1) = (xp, -yp, 1)$. Em forma matricial, temos que (TRAINA, 2003, p. 55):

$$P' = MP, onde\ M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Exemplo, figura 12:

Figura 12. Reflexão de um objeto em torno do eixo X.



Fonte: (Traina, 2003 pag. 55).

Pode-se também adotar um eixo arbitrário A matriz de reflexão pode ser derivada pela composição de uma sequência de matrizes de reflexão e de rotação. Por exemplo, se a reflexão se der em volta da linha diagonal definida por $y = x$, a matriz de reflexão pode ser derivada da combinação das seguintes transformações:

1. rotação de *45°* na direção horária para fazer a linha $y = x$ coincidir com o $eixo\ x$.

2. reflexão em torno do $y = x$

3. rotação de *45°* na direção anti-horária para retomar a orientação original da linha $y = x$.

2.4. Jogos na Educação

Segundo Redação Mais Comunidade (2012), os jogos educativos estão sendo adotados cada vez mais por professores para tornar as aulas mais interessantes e divertidas.

Os jogos fornecem ao ser humano uma iteração suficiente para despertar muitas das funções cognitivas que possuímos, esta funções são responsáveis pelo que aprendemos quanto mais funções utilizarmos para aprender determinado assunto melhor fixado será o aprendizado, segundo Alves (2003) e Bueno (2010).

Logo abaixo temos um trecho do livro de Paulo Nunes de Almeida de 2003 que fala sobre educar ludicamente:

> Educar ludicamente tem um significado muito profundo e está presente em todos os seguimentos da vida. Por exemplo, uma criança que joga bolinha de gude ou brinca de boneca com seus companheiros não está simplesmente brincando e se divertindo; está desenvolvendo e operando inúmeras funções cognitivas e sociais; ocorre o mesmo com uma mãe que acaricia e se entretém com a criança, com um professor que se relaciona bem com seus alunos ou mesmo com um cientista que prepara prazerosamente sua tese ou teoria. Eles educam-se ludicamente, pois combinam e integram a mobilização das relações funcionais ao prazer de interiorizar o conhecimento e a expressão de felicidade que se manifesta na interação com os semelhantes (ALMEIDA, 2003. Cap 2-3, p. 33-68).

Com base em Almeida, 2003 podemos notar que a relação de aprendizado está diretamente entrelaçada ao fato de que quando agimos espontaneamente não há necessidade de forçar a aquisição do conhecimento, muito pelo contrário, é possível enquanto pratica ações que não estão relacionadas ao conhecimento ainda assim ele é obtido, pois o ser humano tem essa capacidade.

O especialista Francisco Osório de Carvalho Ramos, coordenador do curso de Jogos Digitais do Centro Universitário IESB, diz que, além de contribuir para a educação, os games também podem ajudar a integrar a criança na sociedade.

> Os jogos são muito importantes para a inclusão social. Por exemplo, estamos desenvolvendo um jogo em parceira com o curso de Psicologia da instituição para auxiliar na educação de crianças diagnosticadas com o autismo - uma alteração que afeta a capacidade de comunicação do indivíduo, de socialização (estabelecer relacionamentos) e de comportamento. Por meio de um aplicativo com cores e sons, conseguimos chamar a atenção da criança e ajudá-la em seu crescimento pedagógico (REDAÇÃO MAIS COMUNIDADE, 2012).

Com isso vemos que com os jogos realizando a tarefa de entreter, e ao mesmo tempo despertar inúmeras funções cognitivas, ao receber as informações vindas do jogo, elas serão absorvidas com mais facilidade, pois as pessoas estarão se divertindo com, por exemplo, o conteúdo de uma matéria. Tudo vai depender de como este conteúdo está disposto no jogo.

2.4.1. Proposta de um jogo educativo, com fundamentos matemáticos aplicados ao ensino fundamental

A Coordenação-geral de Estudos e Pesquisas da Educação Fundamental (1997), diz que as necessidades cotidianas fazem com que os alunos desenvolvam uma inteligência essencialmente prática, que permite reconhecer problemas, buscar e selecionar informações, tomar decisões e, portanto, desenvolver uma ampla capacidade para lidar com a atividade matemática. Quando essa capacidade é potencializada pela escola, a aprendizagem apresenta melhor resultado.

Os parâmetros curriculares nacionais, elaborados com a finalidade de regulamentar e parametrizar o ensino no país reconhecendo e elucidando o valor dos jogos como uma ferramenta de auxilio, aponta alguns recursos que os jogos possuem e que os tornam passiveis de utilização na educação de um modo geral.

Esses parâmetros curriculares falam que o jogo é um objeto sociocultural onde a Matemática está presente, e que também é uma atividade natural no desenvolvimento dos processos psicológicos básicos, esta atividade supõe um “fazer sem obrigação extrema e imposta”, porém faz-se necessário o uso de normas e controle.

Fizeram saber que no jogo, através da articulação entre o conhecido e o imaginado, desenvolve-se o autoconhecimento – até onde se pode chegar – e o conhecimento dos demais – o que pode ser esperado e em quais circunstâncias isso aconteceria.

Falam também que os jogos são as ações repetidas sistematicamente pelas crianças pequenas, porém possuem um sentido e estes são de forte significado, com isso eles possibilitam a compreensão, geram satisfação e formam hábitos que se estruturam num sistema. A repetição dessa mesma forma deve estar presente na atividade escolar, pois tem um importante papel, ele ajuda a criança a perceber regularidades.

Os jogos não proporcionam para as crianças apenas situações que se repetem, mas as ensina de modo natural, a lidar com símbolos e a pensar por analogia (jogos simbólicos), fazendo com que o significado das coisas passe a ser imaginados por elas. Ao criarem essas analogias, tornam-se produtoras de linguagens, desenvolvem convenções, capacitando-se para se submeterem a regras e fornecer explicações.

Ainda podemos ver nas referências do recurso aos jogos nos parâmetros curriculares, que as crianças passam a compreender e a utilizar convenções e regras que serão empregadas no processo de ensino e aprendizagem. Com isso elas têm favorecida sua integração num mundo social bastante complexo e proporciona as primeiras aproximações com futuras teorizações.

Já num estágio mais avançado, as crianças aprendem a lidar com situações mais complexas (jogos com regras) então começam a compreender que as regras podem ser combinações arbitrárias que os jogadores definem, elas percebem também que só podem jogar em função da jogada do outro ou da jogada anterior. Desse modo nota-se que os jogos com regras têm um aspecto importante, pois neles o fazer e o compreender constituem lados de uma moeda.

Os jogos de grupo, por sua vez, representam uma conquista cognitiva, emocional, moral e social para a criança e um estímulo para o desenvolvimento do seu raciocínio lógico.

E por fim, o elemento fundamental do jogo: o desafio que eles provocam no aluno, e desperta interesse e prazer. Neste ponto a Coordenação-Geral de Estudos e Pesquisas da Educação Fundamental, diz que é importante que os jogos façam parte da cultura escolar, cabendo ao professor analisar e avaliar a potencialidade educativa dos diferentes jogos e o aspecto que ele deseja trabalhar em sala de aula.

Nosso estudo de caso desenvolve um jogo que aborda alguns conceitos matemáticos segundo a Coordenação-geral de Estudos e Pesquisas da Educação Fundamental (1997), que competem ao primeiro ciclo, no que diz respeito às operações de números naturais utilizando os sinais convencionais de operações (+, -, x, :, =).

3. Game Engines (Motores de Jogos)

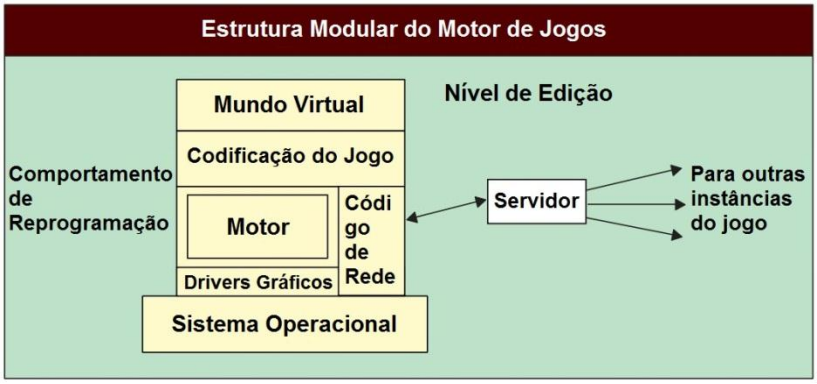
Para o auxilio e maior praticidade de desenvolvimento de jogos, foram criadas diretivas que abstraiam algumas etapas do desenvolvimento de jogos. Primeiro surgiram as Aplication Programming Interface (Interface de Programação de Aplicativos) - API que são bibliotecas adicionadas à linguagem de programação para reduzir a quantidade de funções as quais o programador tinha de fazer toda vez que começasse a desenvolver um jogo. Logo depois surgiram softwares que eram constituídos de mesas (considerado como o ambiente com ferramentas à mão) de desenvolvimento, onde a produção dos jogos é feita em alto nível, cabendo ao desenvolvedor utilizar pouco ou nenhum código para conclusão do jogo.

Os motores são bibliotecas de desenvolvimento, que dão suporte à linguagem e à API gráfica, na maioria das vezes tem a obrigação de organizar a estrutura do jogo, gerenciar imagens, processar entrada de dados e outras funções (GOMES; PAMPLONA, 2005). Os primeiros motores de jogos ficaram famosos com jogos como Doom e Quake que utilizaram programas com ferramentas que facilitaram e deram suporte ao processo de criação destes jogos (REIS, 2009).

Nos dias de hoje jogos são construídos modularmente, o motor do jogo refere-se ao conjunto de módulos de código de simulação que não especificam diretamente o comportamento do jogo (a lógica do jogo) ou o ambiente do jogo (dados de nível) (LEWIS; JACOBSON, 2005) (Tradução nossa).

O motor inclui módulos de manipulação de entrada, saída (renderização em 3D, desenho 2D, som) e física/dinâmica genéricos para os ambientes do jogo. A figura 13 mostra logo abaixo, mostra como é a estrutura global de um motor de jogos genérico (LEWIS; JACOBSON, 2005) (Tradução nossa).

Figura13. Estrutura Modular do Motor de jogos.



No nível superior são os ambientes ou cenários com os quais os jogadores interagem virtualmente. Eles vêm em uma grande diversidade de moldes e regras de interação, até mesmo diferentes tipos de simulação física. Muitos são de autoria de fãs de jogo, que usam principalmente um ambiente de desenvolvimento avançado que vem gratuitamente com o jogo. Muitas vezes eles incluem componentes de mundos existentes ou são construídos utilizando o conhecimento dos autores trocando informações livremente através da Web (LEWIS; JACOBSON, 2005) (Tradução nossa).

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 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 motores costumam ser projetados tendo em vista uma classe particular de jogos, como 2D ou 3D (ROCHA; BESSA; BEZERRA; MEDEIROS; OLIVEIRA; BANDEIRA, 2007).

3.1. Principais Motores de Jogos

- **Allegro**

Allegro é uma biblioteca livre de código fonte aberto para o desenvolvimento de vídeo games.

O objetivo principal é a independência de plataforma de operação. O mesmo código-fonte, sem tirar nem pôr um único caractere que seja, deve compilar e rodar em todas as plataformas suportadas. Um objetivo de curto prazo é a plataforma 64-bits.

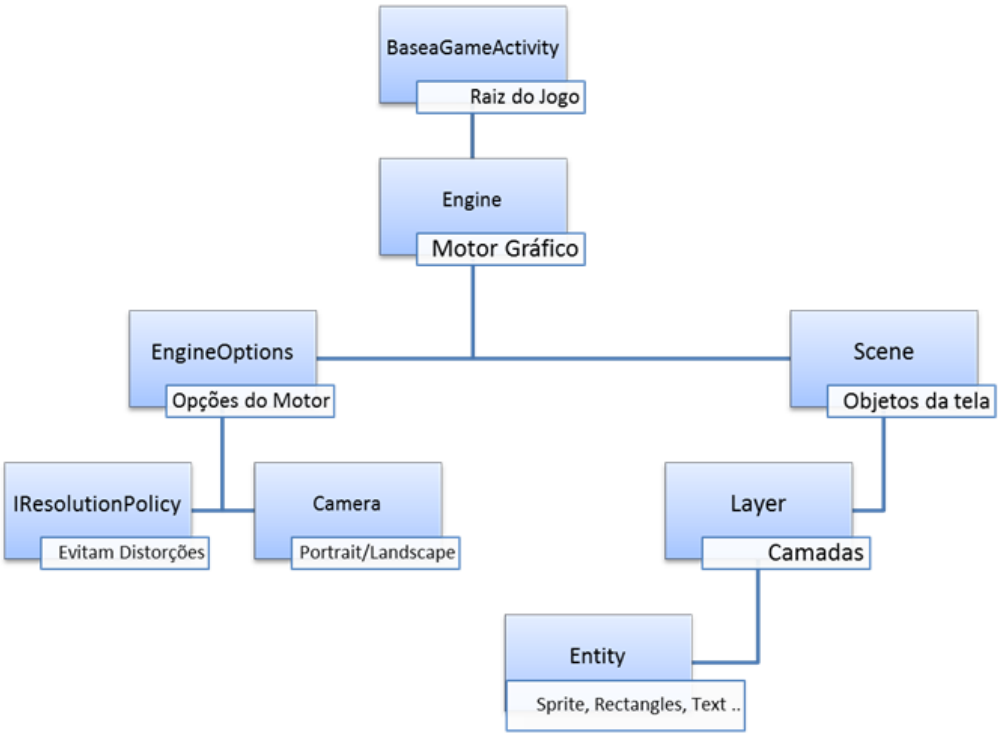
O seu principal uso é no escopo da programação de jogos. Atualmente ela possui uma grande comunidade, pois além de possuir diversos recursos nativamente (gráficos 2D, 3D com *OpenGL*, entrada de dados pelo teclado e mouse, RLE-Sprites, exibição de vídeos e controle de som) a API é bastante extensível fazendo que com existam diversos *addons* disponíveis.

A interface pública de acesso da biblioteca é escrita em C, porém há alguns extensores (não-oficiais) para outras linguagem de programação. Internamente seu código fonte é escrito em uma mistura de C, Assembly (drivers i386), C++ e *Objective-C*.

- **AndEngine**

A *AndEngine* 2D, figura 14, é um kit de desenvolvimento *Open Source* (Código Aberto) desenvolvida por Nicolas Gramlich, e tem o intuito de auxiliar principalmente iniciantes no desenvolvimento de jogos 2D para Android. Ela conta com suporte a detecção de colisões, implementa algumas propriedades físicas (gravidade, velocidade, elasticidade, etc.), auxilia no desenvolvimento de modos *multiplayer*, partículas e tem suporte a *multitouch*.

Figura 14. Arquitetura da AndEngine.



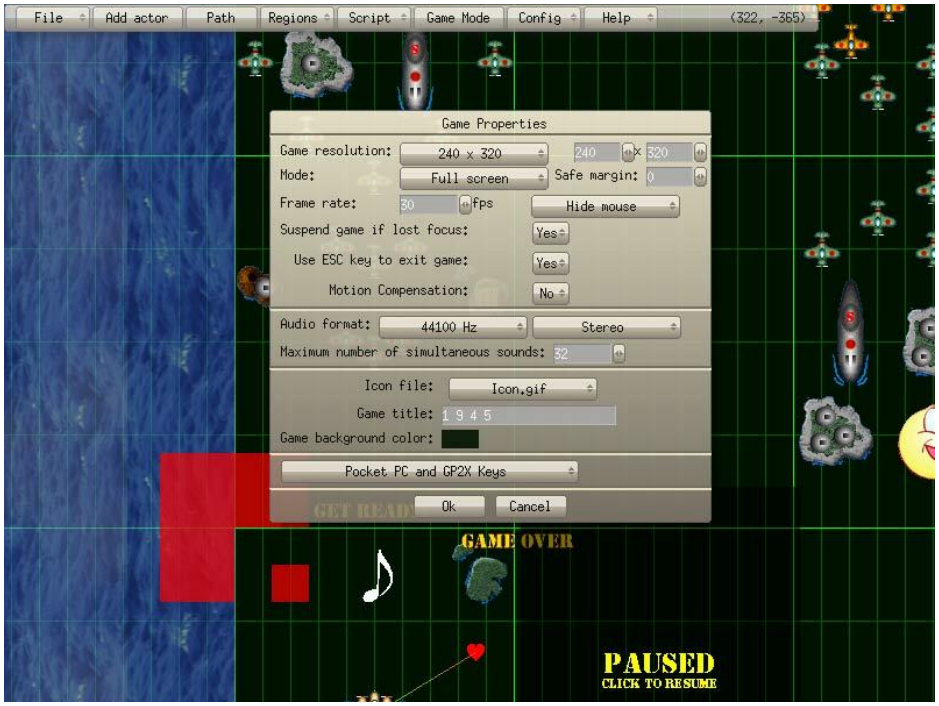
Fonte: ® AndEngine.

- **Game Editor**

Game Editor, figura 15, é um software de desenvolvimento de jogos 2D, que possui código aberto sob a licença GPL v3[2], visa simplificar a produção de jogos eletrônicos e é multiplataforma.

Configuração de propriedades do jogo, como resolução do jogo, qualidade do som no jogo, quantidade de frames por segundo e teclas especiais.

Figura 15. Configurações do Jogo.



Fonte: Game Editor, 2012.

- **GameMaker**

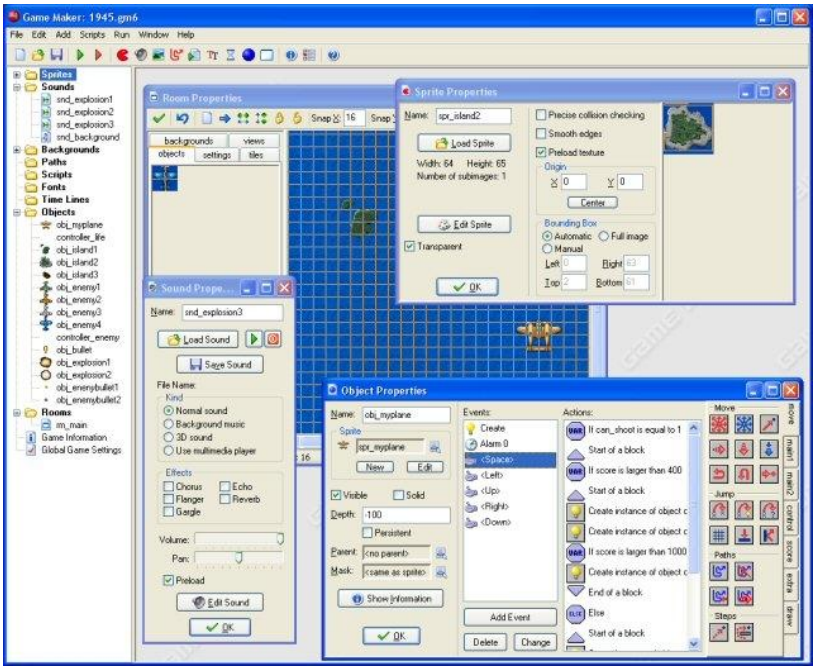
A GameMaker, figura 16, é um forte concorrente para o *Game-Editor*, porém apenas na plataforma Windows. Segundo seu fabricante, © 2007-2013 YoYo Games Ltd, ele é de baixo custo e de fácil aprendizado, e hoje já possui suporte para Mac®.

> Offering a drag-and-drop interface for rapid design and iteration of gameplay features, graphics and sound, as well as a fully-featured integrated development environment (IDE) and a flexible, built-in scripting language

[2] Segundo a © 2007 Free Software Foundation, esta licença determina como poderá ser usado o software que esteja protegido por ela.

(GML, GameMaker Language), GameMaker for Windows is the perfect low-cost tool for quick, easy games development for Microsoft Windows (YOYO GAMES, 2012).

Figura 16. Ambiente de desenvolvimento do GameMaker.



Fonte: GameMaker, 2012.

4. GAME-EDITOR® ESTUDO DE CASO

Para iniciar o desenvolvimento do estudo de caso, fez-se necessário decidir qual motor utilizar.

> A researcher's choice between the Quake and Unreal engines is similar to that between Coke and Pepsi. Both are more than adequate for their intended purpose yet each has its avid fans and detractors (LEWIS; JACOBSON, 2005).

De acordo com Lewis e Jacobson, (2005), citados logo acima, temos que avaliar qual seria o melhor motor para se trabalhar, para isso analisamos a proposta na qual se baseia na utilização de um motor de jogos para desenvolvimento.

O desenvolvimento de jogos tornou-se muito popular com a melhoria da interface dos motores de jogos, isso proporcionou a adesão de um público que não possui tanto conhecimento técnico para criar um jogo convencionalmente.

Outro fator importante para impulsionar essa melhoria na interface foi a complexidade existente em motores de jogo. Porque mesmo que o designer de jogos utilize uma ferramenta que abstraia a programação, ele ainda sentiria dificuldades caso fosse utilizado um motor cuja interface não fosse amigável e intuitiva.

Neste pensamento, foram avaliadas as seguintes questões, abstração da linguagem de programação, interface e também a licença para o desenvolvimento.

4.1. Comparação entre Game-Editor e Concorrentes

Na Sessão 3 deste trabalho descrevemos as principais concorrentes do Game-Editor®, elas têm características diferentes entre si, apesar de fazerem parte da mesma categoria e finalidade.

Os parâmetros observados para a análise comparativa e escolha da ferramenta a qual desenvolvemos o nosso estudo de caso foram:

- ✓ Proporciona maior agilidade no desenvolvimento de jogos;
- ✓ Facilidade de aprendizado;
- ✓ Necessidade de utilização de programação;
- ✓ Permite programação;
- ✓ Tipo de licença;
- ✓ Portabilidade entre plataformas;

- ✓ Quantidade de ferramentas prontas;
- ✓ Gênero.

A Allegro é uma API, baseada em C, *Open Source*, suporta e necessita de programação em C, C++, *Python* entre outras. Ela é apropriada para desenvolvimento de jogos 2D e 3D, por ser necessário saber programar previamente não foi considerada de fácil aprendizado e não é ágil no desenvolvimento, possui grande quantidade de funções prontas. Permite desenvolvimento para diferentes plataformas, porém o código não pode ser portado entre elas facilmente.

A *AndEngine* é feita em Java, é *Open Source* também, porém sua documentação é de difícil acesso, não há suporte, no entanto sua arquitetura é fácil de ser entendida e proporciona um bom gerenciamento para o jogo. Sua plataforma é *Android*™, feita para criação de jogos 2D, podendo criar jogos para dispositivos que suportam *Android*™. Ela é de fácil aprendizado, proporciona um desenvolvimento ágil, mas não permite programação.

A *GameMaker* seria o maior rival para o *Game-Editor*®, partilha do mesmo modelo de licença *Freeware*, possui ótimas ferramentas intuitivas para o usuário, é de fácil aprendizado, peca um pouco na falta de recursos, como Inteligência Artificial – IA e não suporta gráficos com resolução elevada, não é portável e é compatível apenas com *Windows* e *Mac*. Ela não necessita e não permite programação o que a torna um pouco limitada aos recursos disponíveis no próprio motor.

O *Game-Editor* é *Open Source,* multiplataforma, ou seja, o jogo criado com ele é portável para os sistemas operacionais de PC, tanto *Linux*, *Windows* e *Mac*. Ele não necessita de programação, mas permite programação em C, isso faz com que ele se torne mais versátil, pois caso alguma ferramenta do motor não seja satisfatória para o usuário, ou não exista, o mesmo poderia criar uma função programando. Possui uma boa curva de aprendizado, não necessita muito conhecimento para começar e permite um desenvolvimento ágil.

Feita estas comparações observamos quem a melhor ferramenta para utilizarmos no nosso estudo de caso foi o Game-Editor, veja abaixo a tabela 1 onde podemos observar a diferença entre os motores.

Tabela 1. Comparativo entre os motores de jogos

	Allegro	**AndAngine**	**GameMaker**	**Game-Editor**
Agilidade no desenvolvimento	Não	Sim	Sim	Sim
Facilidade	Não	Sim	Sim	Sim
Requer Programação	Sim	Não	Não	Não
Permite Programação	Sim	Não	Não	Sim
Tipo de Licença	Open Source	Open Source	Freeware	Open Source
Portabilidade	Não	Não	Não	Não
Ferramentas Prontas	Sim	Sim	Sim	Sim
Gênero	2D/3D	2D	2D	2D

Abaixo temos a tabela 2 elaborada no Simpósio Brasileiro de Games e Entretenimento Digital por Costa, Souza e Castanho (2011), em um procceding onde são comparadas as principais motores. Nesta comparação fica clara a motivação de usar o Game-Editor® como ferramenta para o estudo de caso, haja vista que suas características de facilidade uso no sentido de apesar de não necessitar o conhecimento de programação ele permite a programação e possui alto grau de flexibilidade alinhado com a quantidade de funcionalidades prontas. Vemos ainda que a mesma também é uma ferramenta gratuita e se adéqua às principais plataformas disponíveis no mercado atual.

Tabela 2. Comparativo das ferramentas para apoio ao desenvolvimento de jogos.

	Licença	Necessita conhecimento de programação	Permite programação	Número de funcionalidades prontas	Grau de flexibilidade	Gênero	Plataforma
Multimedia Fusion 2	paga ($131.49)	não	não	elevado	alto	todos	Windows
Game Maker	paga ($25.00)	não	sim	elevado	alto	todos	Windows
RPG Maker VX	paga ($59.99)	não	sim	elevado	alto	RPG	Windows
Game-Editor	gratuita	não	sim	médio	alto	todos	Windows, Linux, Mac OSX
GameSalad	paga ($99.00)	não	não	elevado	alto	todos	Mac OSX
JumpCraft	paga ($15)	não	sim	elevado	alto	todos	Windows
Kodu Game Lab	paga ($5.00)	não	não	elevado	médio	todos	Windows, XBox 360
MyGameBuilder	gratuita	não	não	baixo	baixo	aventura	Browser (Flash)
BYOND	gratuita	sim	sim	baixo	alto	todos	Windows, Linux, Mac OSX
Alice	gratuita	não	não	elevado	médio	todos	Windows
Venatio Creo	gratuita	não	não	médio	médio	Plataforma e *RPG*	Windows, Browser e XBox 360
<e-Adventure>	gratuita	não	não	elevado	médio	Aventura *Point-and-click*	Windows, Linux, Mac OSX

Fonte: (COSTA; SOUZA; CASTANHO, 2011).

4.2. Game-Editor®

Nesta etapa descrevemos o ambiente de desenvolvimento do Game-Editor e suas funcionalidades:

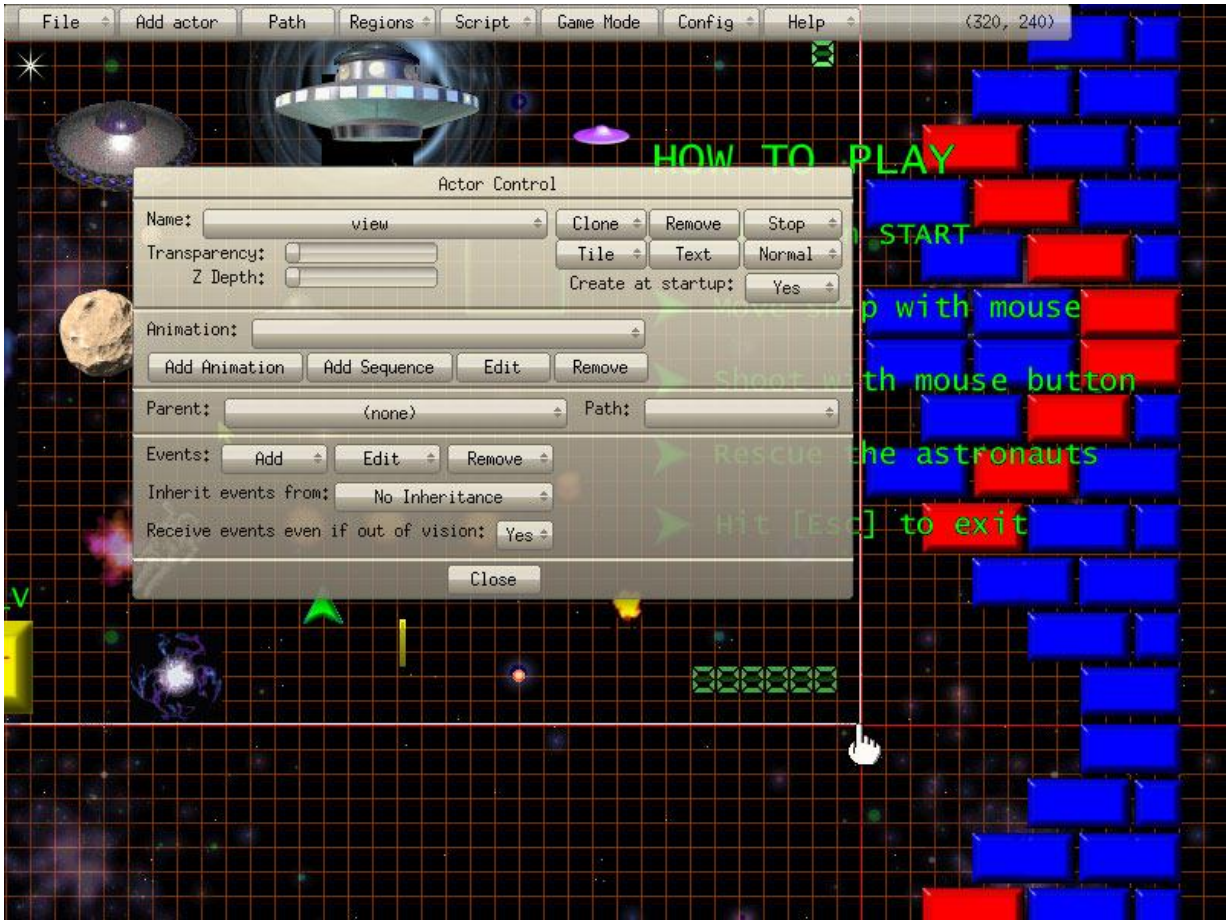
✓ **Controlador de Atores**

Controla os diferentes atores no jogo facilitando a interação com software. Ao desenvolver no Game Editor, você está no comando de tudo e de todos - seja naves espaciais, astronautas ou extraterrestres, todos elementos que compõem o jogo. Esta janela apresenta algumas funções que altera propriedades dos atores no jogo. Por exemplo a propriedade *Z Depth* que é responsável por controlar a coordenada Z dos atores, ou seja, a profundidade, apesar de tratar de um motor para jogos 2D, deve-se notar que existe sobreposição de objetos no ambiente o de

se passa o jogo, por exemplo uma nuvem a frente do sol, neste caso a nuvem teria em sua definição Z um valor que faria com que ela ficasse afrente do sol, na computação gráfica isso é conhecido como oclusão.

O controlador permite também controlar a transparência dos atores, onde um determinado ator poderia começar a ficar translucido até desaparecer a depender de alguma ação do jogo (colisão, timer). Ver figura 17. Podemos destacar nesta janela os botões para adicionar animações, eles são de grande importância para o jogo pois seus atores passam a ter o desenho feito para ele, por exemplo: a animação de um personagem ou a imagem de uma árvore do cenário.

Figura 17. Controlador de atores do Game Editor.



Fonte: Game Editor, 2012.

✓ **Eventos de Ativação**

Através desta ferramenta é possível detectar quando acontece algum evento, e qual deve ser a ação tomada para continuar o jogo. Ver figura 18. Os eventos de ativação é o que dá ação ao jogo, a cada avanço do jogador o desenvolvedor do jogo irá atribuir ações para os eventos. Como exemplo simples

podemos pensar em uma mina terrestre, quando o personagem passar por cima ou perto algo acontece (a mina explode – animação de uma explosão + som de explosão + duração do som + desaparecer com a bomba + ator perder pontos ou morre). Podemos pensar em algo acontecer em função do tempo, por exemplo acabou o tempo para conclusão da fase sem ter ocorrido a mesma então o jogador morre. Ou ainda ao encontrar com um inimigo mudar de música.

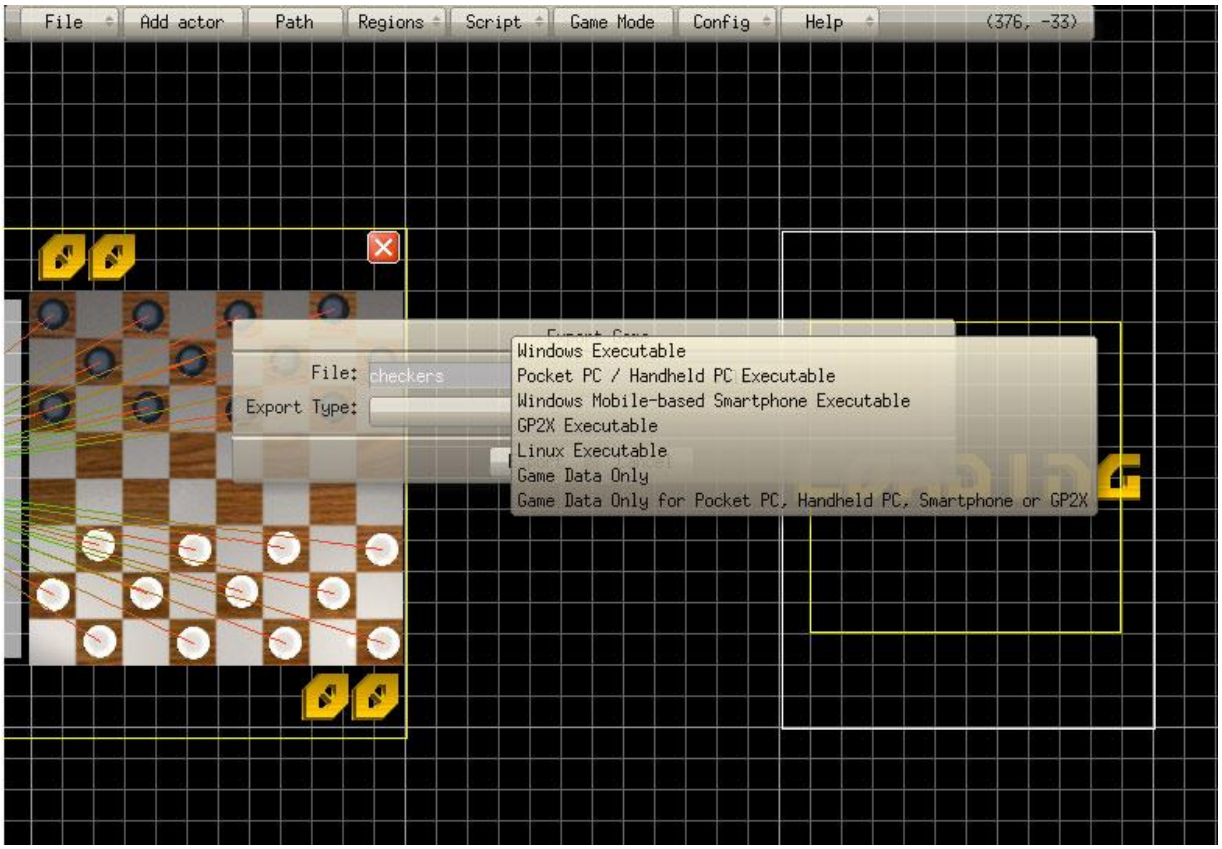
Figura 18. Eventos de Ativação.

Fonte: Game Editor, 2012.

✓ **Ferramenta de Exportação**

Configura atributos para poder utilizar o jogo em qualquer plataforma. Ver figura 19. Nesta ferramenta podemos alterar pra qual plataforma o jogo irá se destinar, definido a resolução de tela, os tipos de entradas (teclado, mouse, touch), versão de software do destino (Windows PC, Windows Mobile).

Figura 19. Ferramenta de Exportação.



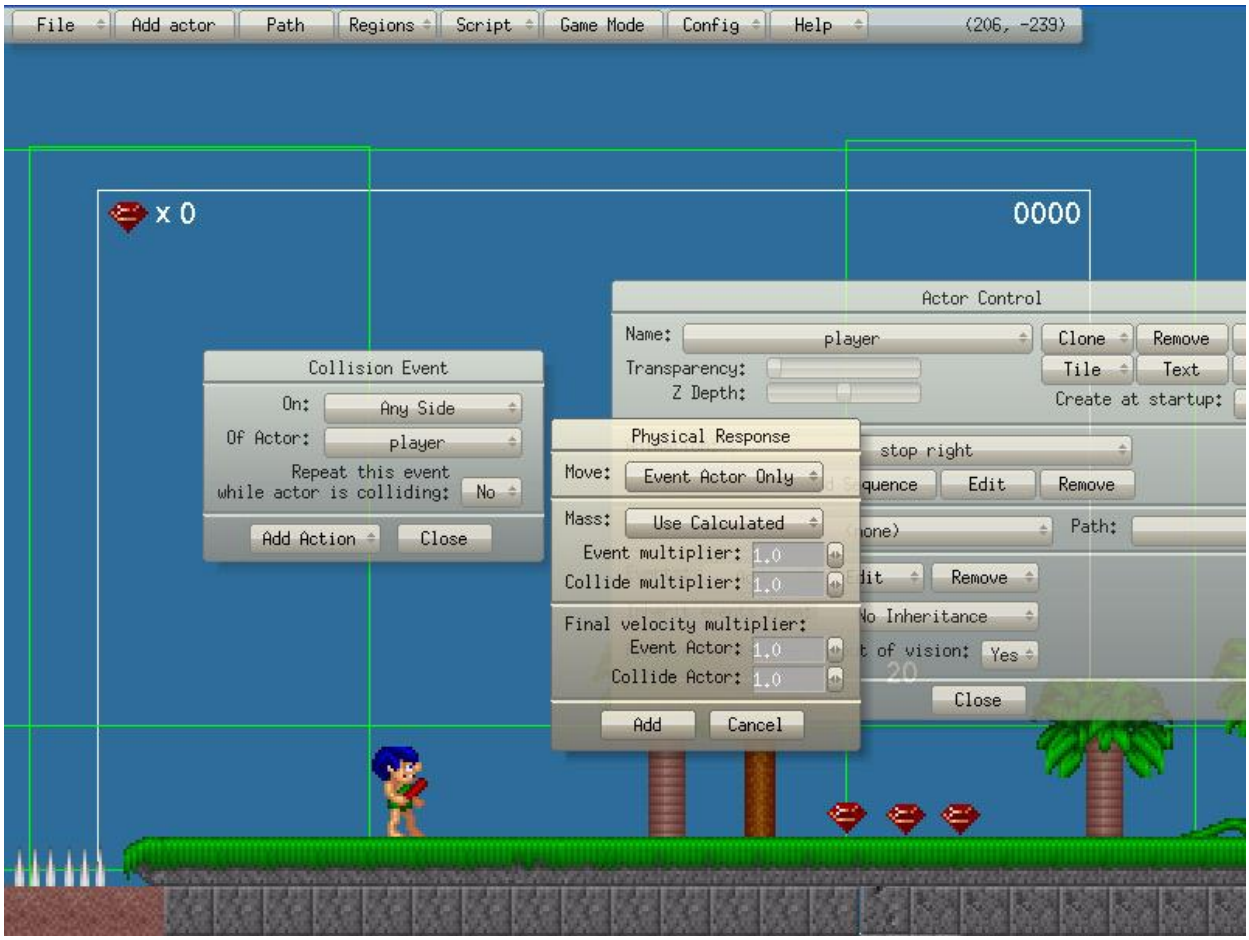
Fonte: Game Editor, 2012.

✓ **Física**

Manipula a física do jogo, detecção de colisão, e propriedade dos atores, como velocidade, peso, gravidade do jogo. Ver figura 20. Com esta janela podemos criar e transformar as propriedades dos atores para que alcancem o maior nível de realismo possível. Para isso necessitamos apenas de um pouco de conhecimento de algumas propriedades. O ator ele tem sua superfície de contato, e a colisão, por exemplo, é detectada quando existe a intersecção entre as superfícies de contato de dois ou mais atores. Tendo em vista essas informações podemos definir qual a massa de nossos atores, qual a intensidade da repulsão em caso de colisão, podemos definir a existência de gravidade e a sua intensidade.

No menu *Physical Response* definimos qual a resposta física de algum tipo de contato, pois, naturalmente, haveria certa diferença entre o contato de uma pedra com o chão em comparação com uma bola de borracha com o mesmo chão. Neste momento definiríamos qual seria a resposta de cada um desses objetos ao colidirem.

Figura 20. Física.



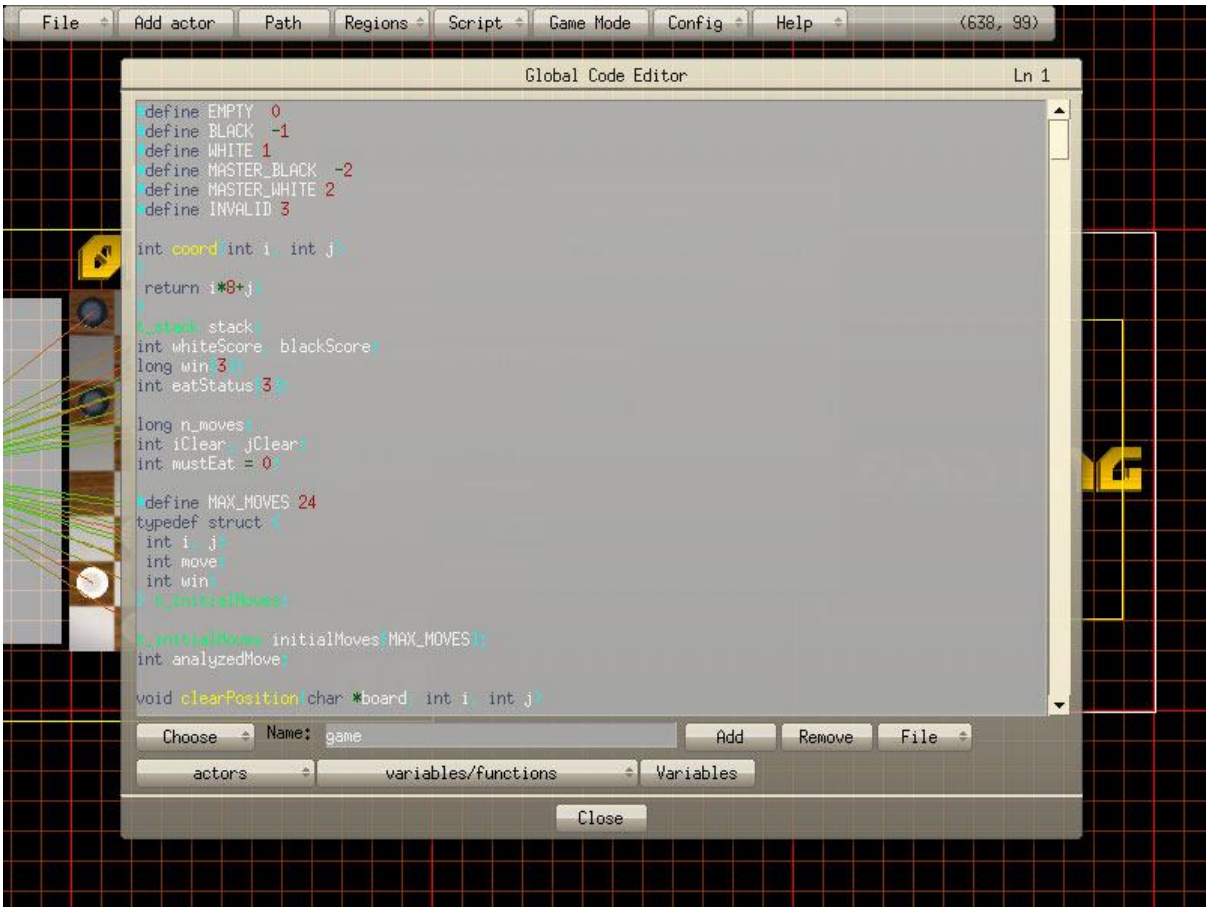
Fonte: Game Editor, 2012.

✓ **Janela para Criação de Script**

Permite que o usuário crie comportamentos e funções personalizadas para seu jogo. Ver figura 21. Particularmente, considero essa a janela mais poderosa do Game-Editor®, pois nela pode ser feita muitas coisas diferentes. Esta janela necessita um pouco de conhecimento em C, então caso o desenvolvedor quisesse ele poderia criar uma estrutura para guardar informações e/ou configurações de jogadores, ele poderia criar novas funções para o jogo, ou até mesmo propriedades físicas mais elaboradas. Nesta já nela o desenvolvedor pode inclusive utilizar de um programa em C, criado anteriormente para atender suas necessidades, não necessitando de um retrabalho.

Para os mais avançados em programação teriam a oportunidade de manipular variáveis e criar macros para facilidade de organização. Otimizar funções e reutilizar o que você escreveu em outros jogos.

Figura 21. Script.



Fonte: Game Editor, 2012.

✓ **Mapas**

Possibilita visualização completa dos mapas criados para o jogo, com facilidade de edição. Ver figura 22. Os mapas podem ser construídos com essa possibilidade. Pense em um cenário, depois pense em como colocar cada coisa nesse cenário, se observar bem seu cenário teria um chão, esse chão não seria construído em uma imagem única isso daria muito trabalho para manter e organizar. Com o Game-Editor® existe a possibilidade de utilizar partes (“pequenas” imagens) para construir um todo. Essas partes são chamadas de *bricks* (tijolos), pois assim como a construção de uma parede geralmente é feita com tijolos no cenário é a mesma coisa. A ferramenta permite utilizar uma imagem que representa uma parte de chão, por exemplo, e repeti-la a fim de construir todo o chão do seu cenário. Com isso para construir qualquer chão no seu cenário independente das formas diversas que ele estiver você precisará apenas de um tijolo para fazê-lo, com isso aumenta a versatilidade e a agilidade de produção de jogos.

Figura 22. Mapas.

Fonte: Game Editor, 2012.

✓ **Paths**

Como a figura abaixo mostra desenhar um caminho para seus atores é simples é possível controlar as ações de cada componente do jogo, desenhando caminhos para cada um. Ver figura 23.

Os *Paths* são formas de criar animações restritas dentro do seu jogo, não necessitando de uma animação prévia como é o mais comum nesse tipo de jogos. Esses *Paths* são caminhos que algum ator dentro do seu jogo irá seguir, eles são muito úteis quando você quer uma ação que pode ser ou não repetitiva, mas é arbitraria dentro do jogo, ou seja, ao definir que algum ator irá seguir aquele caminho ele o fará. Por exemplo, isso é muito comum nos jogos árcade, quando um personagem pega uma moeda (o mais comum) ela faz um pequeno movimento vertical e desaparece, esse movimento pode ser feito facilmente com um caminho.

Os paths são importantes para jogos com obstáculos, por exemplo, jogos que possuem plataformas que sobem e descem e/ou coisas que realizam

movimentos similares a pêndulos. Eles ficam bem simples de serem feitos, no Game-Editor® basta clicar no botão “Path”, clicar em adicionar, escolher a quantidade de pontos (quanto mais pontos mais suaves serão as curvas do caminho) que seu caminho terá e posicioná-lo como quiser.

Figura 23. Caminhos do Game Editor.

Fonte: Game Editor, 2012.

4.3. Implementação do Jogo em Busca da Fórmula Perdida

O jogo Em Busca da Fórmula Perdida [3]usado para elucidar algumas diretivas do motor Game-Editor® contou, neste trabalho, com a produção da fase um. O jogo é inteiramente educacional, pois ele faz o jogador controlar o personagem a fim de resolver enigmas matemáticos, que fazem parte da solução da problemática do jogo, focados na história, enfrentando desafios e ultrapassando obstáculos que estimulam o raciocínio lógico. Voltado para o publico infantil na faixa de 6 a 8 anos. Além disso, ele exige do jogador reação e coordenação entre a visão e as mãos em

[3] O documento de Game Design se encontra no Apêndice A.

situações de pressão, assim, fazendo com que o jogador tenha a sobrevivência como principal objetivo logo que cada desafio é temporizado.

Em Busca da Fórmula Perdida foi desenvolvido para a plataforma PC, o sistema operacional escolhido foi o Windows 7 de 32 bits, porém o jogo pode ser portado para as demais plataformas operacionais suportadas pela engine. Para o desenvolvimento do jogo tivemos que partir do Game Design, o qual embora simples foi mais que suficiente para o pequeno jogo que criamos de acordo com estudo de Schuytema (2008).

Neste jogo buscamos criar uma motivação para o jogador continuar buscando, na fase um do jogo o jogador descobre que a fórmula perdida possui 32 partes. Essas partes seriam distribuídas por cavernas no planeta terra, deixei em aberto se seriam a mesma quantidade de cavernas para a quantidade de partes da fórmula ou se poderia haver mais de uma parte da fórmula em uma mesma caverna ou não haver em parte alguma em alguma caverna. O jogo foi idealizado para ser evolutivo, ou seja, o nível de dificuldade ir aumentando conforme o jogador passasse de fase.

Inicialmente (fase um) existem apenas questões de adição, posteriormente elas ficariam mais difíceis antes de aparecer a próxima operação fundamental.

As regras do jogo seriam o tempo de cada fase que não poderia ser estendido, o tempo de resposta de cada questão, a quantidade limite de vidas. O jogador iniciaria com todas as vidas e o tempo da fase iria diminuindo e caso errasse ou estourasse o tempo de resposta de uma questão ele perderia uma vida. Ao acertar as questões ele acumula pontos.

Para o fim do jogo, ao fim da ultima fase, esperava fazer uma cena de uma formula matemática simples, para o nível do jogador. A princípio as fórmulas seriam coletadas pelo jogador, então no fim da fase ele teria 31 fórmulas e estaria obtendo a de número 32, logo elas se juntariam no ar, com alguns efeitos divertidos, e em seguida mostraria a formula se juntando ao núcleo da terra para salvar o planeta e então seria o fim do jogo.

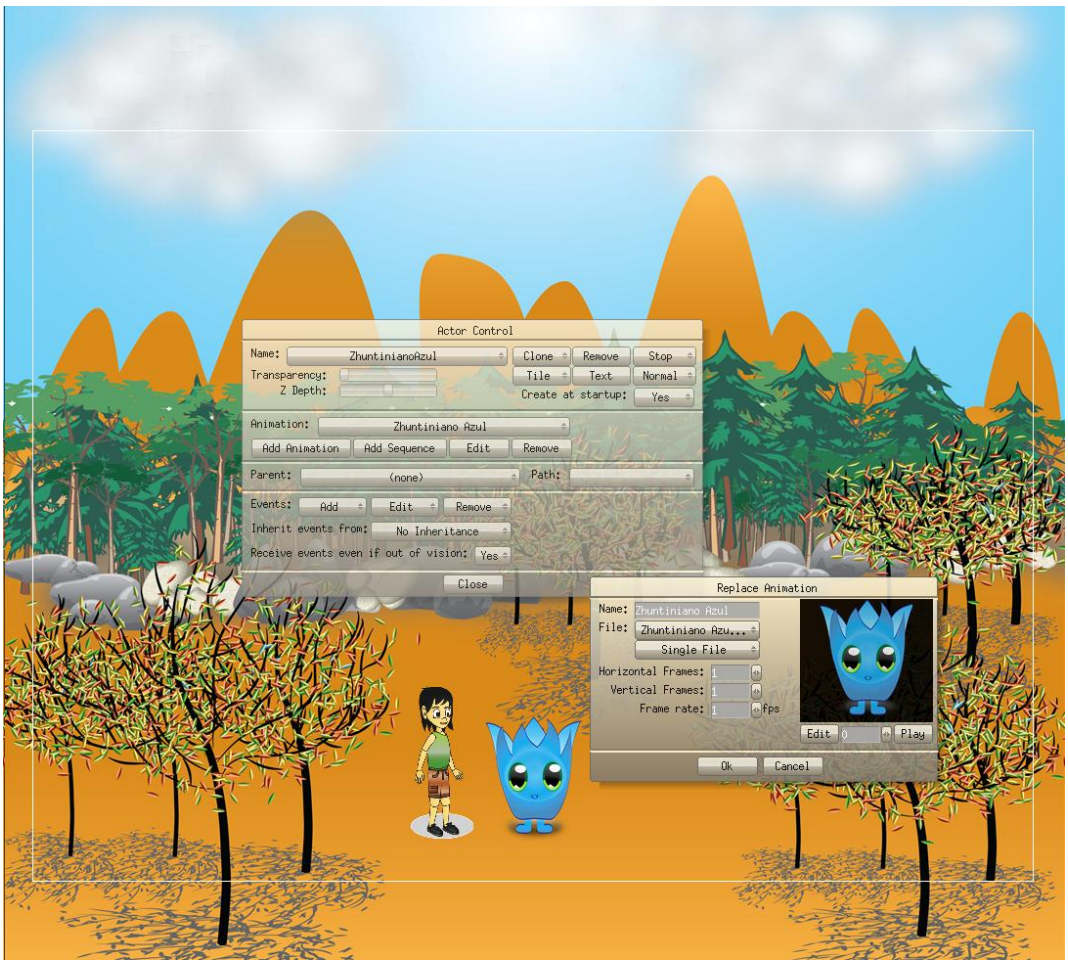
Apesar de ser um jogo acadêmico, todo o jogo possui design próprio com todos os recursos criados especialmente para ele, os personagens e ambientes foram totalmente idealizados com o propósito de serem utilizados neste trabalho.

Para tanto utilizamos algumas ferramentas auxiliares citadas na Sessão 1 (Blender, GIMP).

Pelo fato de o jogo ser simples e não necessitar de agentes e/ou oponentes, não houve a necessidade de criar I.A., apenas um sorteio de desafios[4], para o jogador poder resolver.

Abaixo as figuras 24, 25 e 26 mostram parte da produção do jogo Em Busca da Fórmula Perdida.

Figura 24. Produção da Abertura.

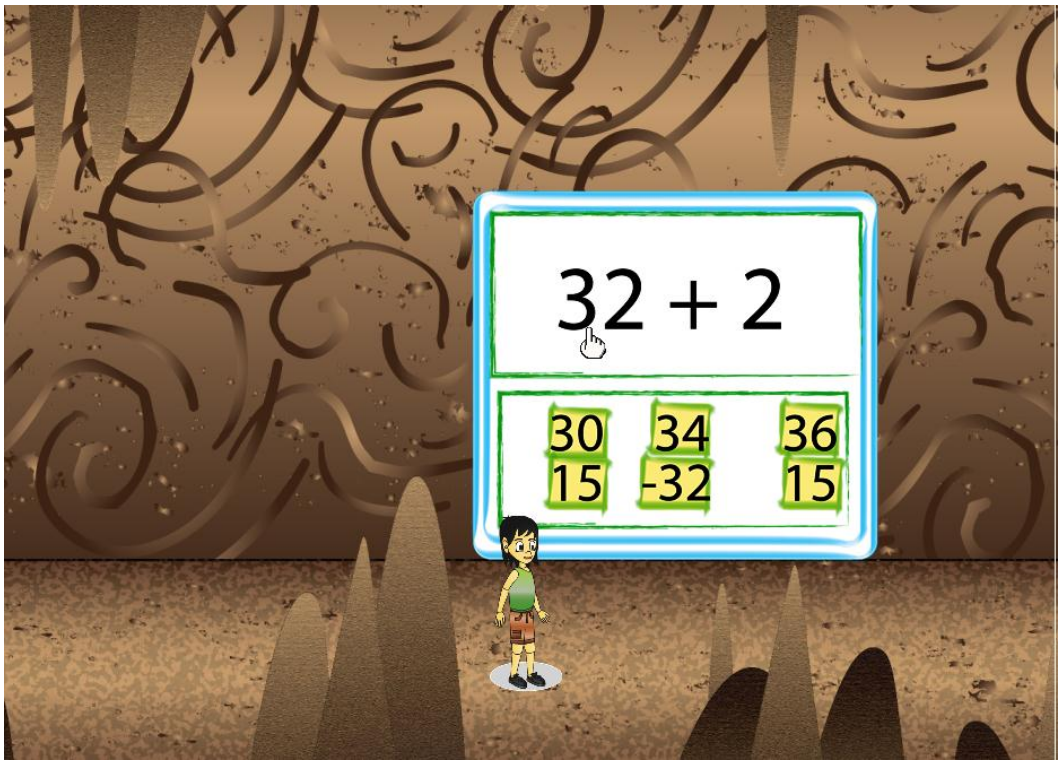


A figura 24 mostra a construção do cenário, juntamente com seus elementos: árvores, pedras, montanhas, sol, céu e nuvens. Este cenário faz parte do primeiro diálogo do jogo, nesta cena acontece a iteração entre o Marcos (personagem principal) e o Azula (Zhuntiniano).

[4] Questões matemáticas elaboradas com base nas operações com números naturais e utilização de sinais convencionais (+, -, x, :, =), descrita em “Conteúdos Conceituais e Procedimentais” intitulada de Conteúdos de Matemática para o primeiro ciclo e contida no documento “Parâmetros Curriculares Nacionais”. Normatização essa, elaborada pela Coordenação-geral de Estudos e Pesquisas da Educação Fundamental (1997).

A ideia é espalhar o conteúdo para as demais fases. Esse diálogo não é muito extenso, conta pouca história, e já deixa o jogador jogar. Quando no modo de jogo existe a possibilidade do jogador pular a introdução. Ainda na questão do diálogo, vale ressaltar que a faixa etária dos jogadores aponta que caso o jogo "demore" de começar a possibilidade da criança perder o interesse é grande.

Figura 25. Posiciona Objetos na Caverna.



A figura 25 mostra um pouco do cenário dentro da caverna e um esboço do desafio. A caverna ela é como será visto na primeira fase, essa é uma caverna no planeta Zhunte, é uma típica caverna Zhunte com estalactites no teto e parte do chão e marcas nas paredes. O Marcos é mostrado pequeno justamente para passar a impressão de a caverna ser gigantesca.

O desafio na primeira fase tem apenas duas opções de resposta. Ao aprofundar mais na questão da faixa etária do publico alvo, foi observado que nesta idade as crianças demoram um pouco para poder associar os símbolos, e também ainda não conseguem fazer cálculos com números com mais de uma casa decimal.

Para as demais fases deverá ser observado esses detalhes para que o jogo não deixe de agradar ou se torne frustrante para as crianças.

Em toda essa produção utilizamos o *GIMP* para criar as imagens e texturizá-las, juntamente com o *Blender*.

Figura 26. Topografia de Em Busca da Fórmula Perdida.

Essa topografia foi vetorizada no *GIMP*, utilizando as ferramentas de texto e alguns efeitos. A mesma técnica foi utilizada nas demais formas textuais como os botões, por exemplo.

Primeiro definimos que iriamos utilizar o *Scrum Solo* que é uma versão para programadores solo. Este desmembramento do Scrum nos permite utilizar o *sprint*, o *backlog* de produto e o *backlog* de *sprint*.

O *backlog* de produto ficou dessa forma:

- ✓ Desenvolver documento game design
  - Personagens;
  - Enredo;
  - Contextualização;
  - Características;
  - Quadro de questões;
  - Itens.
- ✓ Produzir os componentes gráficos
  - Avatar personagem;
  - Topografia;
  - Sons;
  - Cenários;
  - Roteiro;
  - *Dashboard*;
  - *Storyboard*;
  - Lógica;
  - Textos.

- ✓ Montagem
  - Movimento personagens;
  - Temporizadores;
  - Colisões;
  - Mudança de música;
  - Pontuação;
  - Contador de vidas;
  - Funções;
  - Banco de questões;

Para o desenvolvimento utilizou-se três iterações, estas foram suficientes para cobrir todo o processo de construção.

1. No primeiro *sprint* foi elaborado:
   a. O documento game design, este sendo de fundamental importância para o rumo do jogo, pois serviu como norte para as demais iterações. Nele continha a personalidade do jogo, enredo, contextualização, descrição do mundo e dos personagens, fluxo do jogo, descrição dos itens do jogo e das demais características fundamentais à sua constituição.
2. No segundo *sprint* construímos todos os artefatos utilizados pelo jogo. Nesta fase houve a produção:
   a. Dos personagens;
   b. Dos itens do jogo;
   c. Do mundo;
   d. Dos cenários;
   e. Dos componentes informativos (Score[5], Timer[6], Life[7] e etc.);
   f. Dos elementos sonoros;
   g. Da topografia;
   h. Dos componentes gráfico/auditivo/textual que compõem a estrutura lógica do jogo.

[5] Score: contagem, neste caso a contagem de pontos obtidos pelo jogador durante a partida.
[6] Timer: contador de tempo decorrido, no contexto, diz respeito ao tempo passado no jogo. Particularmente no jogo “Em Busca da Fórmula Perdida”, o timer é utilizado de forma decrescente a fim de pressionar o jogador aumentando o desafio e exigindo uma reação mais rápida.
[7] Life: vida. Aqui tratamos vida como a quantidade de erros que permitem o jogador a continuar jogando, ou seja, iniciamos o jogo com certa quantidade (número inteiro positivo) de vida e ela é decrescida até chegar a zero, então o jogo para.

3. Por fim o terceiro *sprint* a composição de todos os artefatos no âmbito do jogo, ou seja, foi o momento em que foram atribuídas:
    a. Aos componentes;
    b. As suas funções e características lógicas para o perfeito funcionamento do jogo;
    c. As animações do personagem principal;
    d. Dos desafios;
    e. As funções lógicas de jogo;
    f. Atribuímos às teclas do teclado funções que permitem ao jogador interagir com o seu personagem no jogo;
    g. Foram criados os contadores de pontos, de vida e de tempo.

Após o ultimo *sprint* o jogo estava pronto. Alguns ajustes foram necessários. Os testes foram feitos pelo desenvolvedor.

A figura 27 mostra a tela inicial do jogo, ela contém os botões “Jogar” e “Sobre” e Topografia do jogo. Ao clicar em “Jogar” inicia-se o jogo. Ao clicar em “Sobre” abre uma janela com informações sobre o jogo e como jogar.

Figura 27. Tela inical do jogo Em Busaca da Fórmula Perdida.

5. CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi expor o processo de criação de jogos com motores de jogos, bem como o desenvolvimento de um estudo de caso baseado na metodologia do *Game-Editor*®.

A elaboração de jogos com um motor mostrou-se na prática, ao ponto que, com conhecimentos básicos a respeito de lógica de jogos e operações de transformações de objetos geométricos, qualquer interessado poderia criar um jogo.

O estudo de caso deu-se com base na ótima aplicabilidade de jogos menos complexos na educação básica, já que permitem uma aprendizagem dinâmica ao despertar inúmeras funções cognitivas.

Com base no estudo realizado, verificou-se que a utilização de um motor de jogos torna seu desenvolvimento mais fácil, pois possui uma boa quantidade de ferramentas prontas de modo que não se faz necessário o conhecimento de programação para desenvolver um jogo.

Podemos afirmar também que o *Game-Editor*® mostrou-se uma ferramenta intuitiva, com uma interface mais amigável que a dos concorrentes. Seus recursos são um tanto limitados, mas atende ao seu propósito, que é a criação de jogos com pouca complexidade, com boa interface e de maneira direta, sem a necessidade de conhecimento profundo de programação. Vimos também que se faz necessária à utilização de técnicas adicionais, como por exemplo, alguns fundamentos de computação gráfica para desenvolvimento de jogos, pois sem seus conceitos o processo de produção seria, mesmo com o *Game-Editor*®, menos ágil.

O uso da metodologia ágil proporcionou ao desenvolvimento**,** em todas as suas fases**,** uma visão clara dos objetivos a serem atingidos, bem como as técnicas de planejamento. Pois os *spints* visavam cumprir o objetivo visando sempre a conclusão do produto de modo que cada *sprint* atendia a uma parte do *backlog* do produto.

Concluímos com este trabalho que modelar um jogo com um motor proporciona agilidade, facilidade e abstração das etapas que não dizem respeito ao contexto puro da história do jogo em si. O *Game-Editor*® foi a melhor escolha para este trabalho**,** pois apresentou melhores indicadores de praticidade, sua interface clara manteve sempre o objetivo de facilitar o desenvolvimento sem a necessidade de configurações extensas, como no caso dos concorrentes**.**

5.1. Sugestões para trabalhos futuros

Como sugestão de trabalhos futuros, tenho como proposta o prosseguimento do desenvolvimento do projeto “Em Busca da Fórmula Perdida”, no intuito de criar novos desafios, onde o ensino seria melhor aplicado.

Desenvolver o jogo especificamente para smartphone e tabletes, alcançando também seus usuários.

Utilizando o *Game-Editor*® pode-se obter o jogo em 2D, mas proponho ainda a utilização de outro motor para uma nova versão do jogo em 3D, o que seria ainda mais interessante para o usuário.

Outra proposta é o desenvolvimento de um novo jogo educativo com base na Game Editor, que traga como o objetivo um maior dinamismo na aprendizagem de novas línguas, como Libras por exemplo. O novo software traria fases de assimilação do vocabulário da nova língua com imagens flutuantes, e também com os seus sons.

6. REFERÊNCIAS

© 2012 HEPTAGON TI LTDA (Brasil). **Scrum**. Disponível em: <http://www.heptagon.com.br/scrum>. Acesso em: 05 mar. 2013.

ALMEIDA, Paulo Nunes de. EDUCAÇAO LUDICA: TECNICAS E JOGOS PEDAGOGICOS. In: ALMEIDA, Paulo Nunes de. **Educação Lúdica:** Técnicas e Jogos Pedagógicos. 9. ed. São Paulo: Edições Loyola, 2003. Cap. 2-3, p. 33-68.

ALVES, Álvaro Marcel Palomo. **A história dos jogos e a constituição da cultura lúdica**: The history of games and the constitution of play culture. **Linhas**: Udesc, Florianópolis, v. 4, n. 1, 1 jan. 2003. Semestral.

ASSIS, D., MATIAS, A**. Game supera cinema como opção de entretenimento em 2003**. Artigo da Folha de São Paulo. Disponível em: http://www1.folha.uol.com.br/folha/ilustrada/ult90u40114.shtml. 30 ago. 2012.

BARROS, Raphael Lima Belém de. **Análise de Metodologias de Desenvolvimento de Software aplicadas ao Desenvolvimento de Jogos Eletrônicos.** 2007. 79 f. Dissertação (Graduação) - Curso de Ciência da Computação, Departamento de Centro De Informática, Universidade Federal De Pernambuco, Recife, 2007. Cap. 5. Disponível em: <http://www.cin.ufpe.br/~tg/2007-1/rlbb.pdf>. Acesso em: 29 ago. 2012.

BLENDER. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2012. Disponível em: <http://pt.wikipedia.org/w/index.php?title=Blender&oldid=35367842>. Acesso em: 29 ago. 2012.

BUENO, Elizangela. **JOGOS E BRINCADEIRAS NA EDUCAÇÃO INFANTIL:** ensinando de forma lúdica. 2010. 43 f. Trabalho de Conclusão de Curso (Graduação) - Curso de Pedagogia, Universidade Estadual de Londrina, Londrina, 2010.

COORDENAÇÃO-GERAL DE ESTUDOS E PESQUISAS DA EDUCAÇÃO FUNDAMENTAL (Brasil). Departamento de Política da Educação Fundamental e Secretaria de Educação Fundamental. **PARÂMETROS CURRICULARES NACIONAIS:** Matemática : Ensino de primeira à quarta série. Brasília: MEC/SEF,

1997. 142 p. Disponível em: <http://portal.mec.gov.br/seb/arquivos/pdf/livro03.pdf>. Acesso em: 22 jan. 2013.

COSTA, Igor Augusto de Faria; SOUZA, Alessandra Silva de; CASTANHO, Carla Denise. Gameka: Uma ferramenta de desenvolvimento de jogos para não programadores. In: SIMPÓSIO BRASILEIRO DE GAMES E ENTRETENIMENTO DIGITAL, 10., 2011, Salvador. **Gameka: Uma ferramenta de desenvolvimento de jogos para não programadores.** Salvador: Sbc - Proceedings Of Sbgames, 2011. v. 1, p. 1 - 4. Disponível em: <http://sbgames.org/sbgames2011/proceedings/sbgames/papers/comp/short/17-92309_2.pdf>. Acesso em: 17 mar. 2013.

GAME EDITOR (Estados Unidos). **Game Editor:** The Cross Plataform Game Creator. Disponível em: <http://game-editor.com/Game_Editor:About>. Acesso em: 30 ago. 2012.

GIMP. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2012. Disponível em: <http://pt.wikipedia.org/w/index.php?title=GIMP&oldid=35598278>. Acesso em: 29 ago. 2012.

GOMES, P. C. R.; PAMPLONA, V. F. **M3ge: um motor de jogos 3D para dispositivos móveis com suporte a mobile 3D graphics**. FURB/BCC. 2005.

HIGHSMITH, J. **Agile Software Development Ecosystems.** 1ª ed. USA: Addison Wesley, 202.

LEWIS, Michael; JACOBSON, Jeffrey.GAME ENGINES: IN SCIENTIFIC RESEARCH. **Communications Of The Acm**, Pittsburgh, v. 45, n. 1, p.27-31, 01 jan 2005. Semanal. Disponível em: <http://usl.sis.pitt.edu/ulab/pubs/p27-lewis.pdf>. Acesso em: 30 ago. 2012.

LILGE, Cassio et al. A indústria dos games no Brasil. **Nas Páginas do Dia: O mundo pelos seus olhos,** Pelotas, 13 jun. 2012. p. 01-03. Disponível em: <http://letras.ufpel.edu.br/naspaginasdodia/reportagens/index.php?cod=37>. Acesso em: 30 ago. 2012.

LINGUAGEM DE PROGRAMAÇÃO. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2012. Disponível em:

<http://pt.wikipedia.org/w/index.php?title=Linguagem_de_programa%C3%A7%C3%A3o&oldid=35585920>. Acesso em: 29 ago. 2012.

MAGALHÃES, Guilherme. **Os Modelos de Desenvolvimento de Software.** Disponível em: <http://protocoloti.blogspot.com.br/2012/03/os-modelos-de-desenvolvimento-de.html>. Acesso em: 30 ago. 2012.

MARCHESI, M. et al. **Extreme Programming Perspectives.** [S.1]: Addison Wesley, 2002.

MOURÃO, Rodrigo Carreiro. Orientação a Objetos no Delphi for PHP: Como aplicar conceitos de POO em aplicações PHP? **Clube Delphi 93**, São Paulo, n., p.1-2, 2007.

PETRILLO, Fábio Dos Santos. **Práticas ágeis no processo de desenvolvimento de jogos eletrônicos:** Best practices in process development of electronic games. 2008. 168 f. Dissertação (Mestrado) - Curso de Programa de Pós-graduação em Computação, Departamento de Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2008. Cap. 3. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/22809/000738163.pdf?sequence=1>. Acesso em: 30 ago. 2012.

PRESSMAN, R. S. **Engenharia de Software**. 6. ed. Rio de Janeiro: McGraw-Hill, 2006, 720p.

REDAÇÃO MAIS COMUNIDADE (Brasil). **Especialistas avaliam a importância dos jogos na educação.** Disponível em: <http://www.maiscomunidade.com/conteudo/2008-05-19/brasilia/141383/ESPECIALISTAS-AVALIAM-A-IMPORTANCIA-DOS-JOGOS-NA-EDUCACAO.pnhtml>. Acesso em: 29 ago. 2012.

REIS, F. V. DOS. **Jogo da cabanagem: projeto e implementação**. Instituto de Tecnologia. Universidade Federal do Pará. Belém – Pará. 2009.

ROCHA, Rodrigo; BESSA, Aline; BEZERRA, Carlos E. B.; MEDEIROS, Ivan; OLIVEIRA, Caio; BANDEIRA, H.; **O Desenvolvimento de um Motor Multiplataforma para Jogos 3D** in: VI Simpósio Brasileiro de Jogos para

Computador e Entretenimento Digital - Trilha Computação, 2007, São Leopoldo, RS, Brasil.

SCHUYTEMA, Paul. **Design de Games: Uma abordagem prática.** 1ª Edição. São Paulo: Cengage Learning. 2008. 447p.

SEBESTA, R. W. **Conceitos de Linguagem de Programação.** *Bookman, 2000.*

SICA, Carlos. **PHP Orientado a Objetos – Fale a Linguagem da Internet**. 1ª ed. Rio de Janeiro: Ciência Moderna, 2006.

SILVA JÚNIOR, David Dantas Da. **O USO DE DESENVOLVIMENTO DE JOGOS NO PROCESSO DE ENSINO-APRENDIZAGEM DO CURSO DE CIÊNCIA DA COMPUTAÇÃO.** 2008. 84 f. Dissertação (Graduação) - Curso de Bacharelado Em Ciência Da Computação, Departamento de Ciências Exatas, Universidade Estadual Do Sudoeste Da Bahia, Vitória da Conquista, 2008. Cap. 2.

TRAINA, Agma J. M.; OLIVEIRA, Maria C. F. de. **Apostila de Computação Gráfica.** São Paulo: 2003.

YOYO GAMES (Estados Unidos). **YoYo Games Revolutionizes Developer Services With Unprecedented Monetization Options In GameMaker: Studio.** Disponível em: <http://www.yoyogames.com/gamemaker/windows/>. Acesso em: 30 ago. 2012.

APÊNDICE A – DOCUMENTO GAME DESIGN

Game Design – Em Busca da Fórmula Perdida

Conceito

Em Busca da Fórmula Perdida se passa no planeta *Zhunte*, situado na galáxia Cosenus vizinha à Via Láctea, este planeta é habitado pelo povo denominado *Zhuntinianos*, este povo é aficionado por matemática tudo que envolve sua vida cotidiana é constituído puramente de matemática. Um dia um sábio, habitante do planeta *Zhunte* descobre que em pouco tempo o meteoro Integral irá colidir com um planeta da galáxia vizinha, a Terra. Ele sabe que existe um jeito de alterar a trajetória do meteoro Integral, a "Fórmula perdida" é uma lenda antiga no planeta *Zhunte*, a lenda conta que inserindo a "Fórmula perdida" na equação do universo é possível alterar qualquer coisa no universo. A tal fórmula está perdida na Terra e os *Zhuntinianos* não suportam a atmosfera terrena, então escolheram dois jovens da Terra para irem a uma jornada perigosa e cheia de misteriosos enigmas para encontrarem a "Fórmula perdida" e salvar a Terra da destruição eminente.

Maria e Marcos são duas crianças que adoram desafios, e eles são os escolhidos para buscar a "Fórmula perdida". Maria é a típica menina sapeca, tem onze anos, cabelo ruivo sardas nas bochechas e é a melhor amiga de Marcos, garoto tímido de nove anos, branco e rico. Eles sempre estudaram juntos e em um dia como qualquer outro eles são abduzidos pelos Zhuntinianos e recebem a importante missão de salvar a Terra, porém antes de buscarem a "Fórmula perdida" na Terra precisam mostrar que são capazes de completar tal desafio.

É nesse ponto que nosso jogo começa. Assim que Maria e Marcos entendem a história e aceitam a missão, Marcos é teleportado para uma caverna em Zhunte onde tem o objetivo de encontrar um antigo pergaminho que indicará onde está escondida a "Fórmula perdida". O primeiro desafio se dá em uma caverna que possui inúmeros enigmas matemáticos como chaves, eles servem como preparação

para chegar até o desafio final que ao ser solucionado revela a localização do antigo pergaminho.

Em *Busca da Fórmula Perdida* é um jogo que proporciona para o jogador a sensação de imersão na aventura, a cada desafio conquistado o jogador avança para um novo desafio, cada desafio tem um nível, cada nível proporciona certo avanço.

O personagem anda o tempo todo dentro da caverna, e é teleportado pelos *Zhuntinianos* quando consegue completar determinada etapa. Marcos conta apenas com seu próprio raciocínio para resolver os enigmas.

Gênero

O gênero de *Em Busca da Fórmula Perdida* é educativo e tem como base a plataforma PC.

Público alvo

Em *Busca da Fórmula Perdida* é voltado para o público infantil que esteja cursando o primeiro ciclo definido pelo Ministério da Educação.

Audiência

Não haverá audiência.

Descrição

O *Em Busca da Fórmula Perdida* começa e então são mostradas opções de iniciar o jogo, alterar configurações de som e sair do jogo.

Escolhendo iniciar jogo uma tela para entrada do nome do jogador é aberta, o jogador é obrigado a preenchê-la caso queira continuar jogando.

Em seguida o jogador começa a jornada como Marcos, único personagem a ser usado nesta fase, e então é iniciado um contador de tempo para acabar a fase e onde cada obstáculo é solucionado com a resolução de um problema de aritmética fundamental que também possui um tempo limite para ser respondido. Caso o jogador erre a resposta do problema ou o tempo de responder chegue a zero ele

perde uma vida e volta para um desafio anterior, caso não haja mais vidas para este jogador o jogo acaba.

Quando o jogo acaba o jogador vê sua pontuação, o *highscore* e pode reiniciar ou sair do jogo.

Informação de marketing

Não haverá marketing para este jogo!

Custo

A produção será criada com componentes com licença aberta.

Tempo de desenvolvimento

O tempo para desenvolvimento é de dois meses.

Especificações Técnicas

Em Busca da Fórmula Perdida tem como principal plataforma *Windows*.

Requerimentos de hardware mínimo: Celeron 1.0 GHz, vídeo 64MB, 1GB RAM, 10 GB HD, Monitor, teclado e mouse.

Requerimentos de hardware recomendado: Celeron 2.0 GHz, vídeo 256MB, 2GB RAM, 30GB HD, Monitor, teclado e mouse.

Especificações do Jogo

Em *Busca da Fórmula Perdida* em sua versão *Beta* possui apenas uma fase, que podem ser jogadas em dois níveis de dificuldade diferentes, nível fácil e nível médio. No fácil o jogador inicia com 5 vidas, no médio inicia com 3 vidas.

A pontuação é baseada no nível de dificuldade, na quantidade de vidas usadas para concluir a fase, na pontuação ganha por concluir a fase e no tempo que foi usado para terminar a fase.

Ao terminar o jogo é exibido um *highscore* com os dez melhores resultados obtidos.

O jogador pode a qualquer momento do jogo configurar o som do jogo.

O jogo é *singleplayer*.

A visão do jogador é de terceira pessoa, com visão de topo na maior parte do jogo e visão lateral em alguns momentos.

Os personagens principais são: Maria e Marcos (apenas Marcos nesta fase).

Os demais personagens são: O sábio *Zhuntiniano*, e os demais habitantes de *Zhunte*.

Status	**Mensagem**
Inicio do jogo.	Esteja atento ao tempo da fase!
Faltando 15 segundos para fim do tempo da fase.	Rápido o seu tempo está acabando!
Faltando 3 segundos para fim do tempo da resolver um problema.	Mais rápido!
Errou o problema ou o tempo dele acabou.	☹
Fim do tempo da fase.	☹
Respondeu correto em 1 segundo.	☺
Terminou a fase em menos de 50% do tempo.	☺

Dispositivos de Entrada

Serão usadas as letras do teclado para entrada do nome do jogador e as setas para movimentação do personagem, o mouse será usado para resolução dos problemas.

Design Gráfico e Arte

- Composto pelo planeta Zhunte.

  O Planeta é composto por terra alaranjada, e vegetação parte floresta com árvores comuns, de tonalidade verde, tronco marrom e outra parte árvores colorida.

Figura 28. Planeta Zhunte.

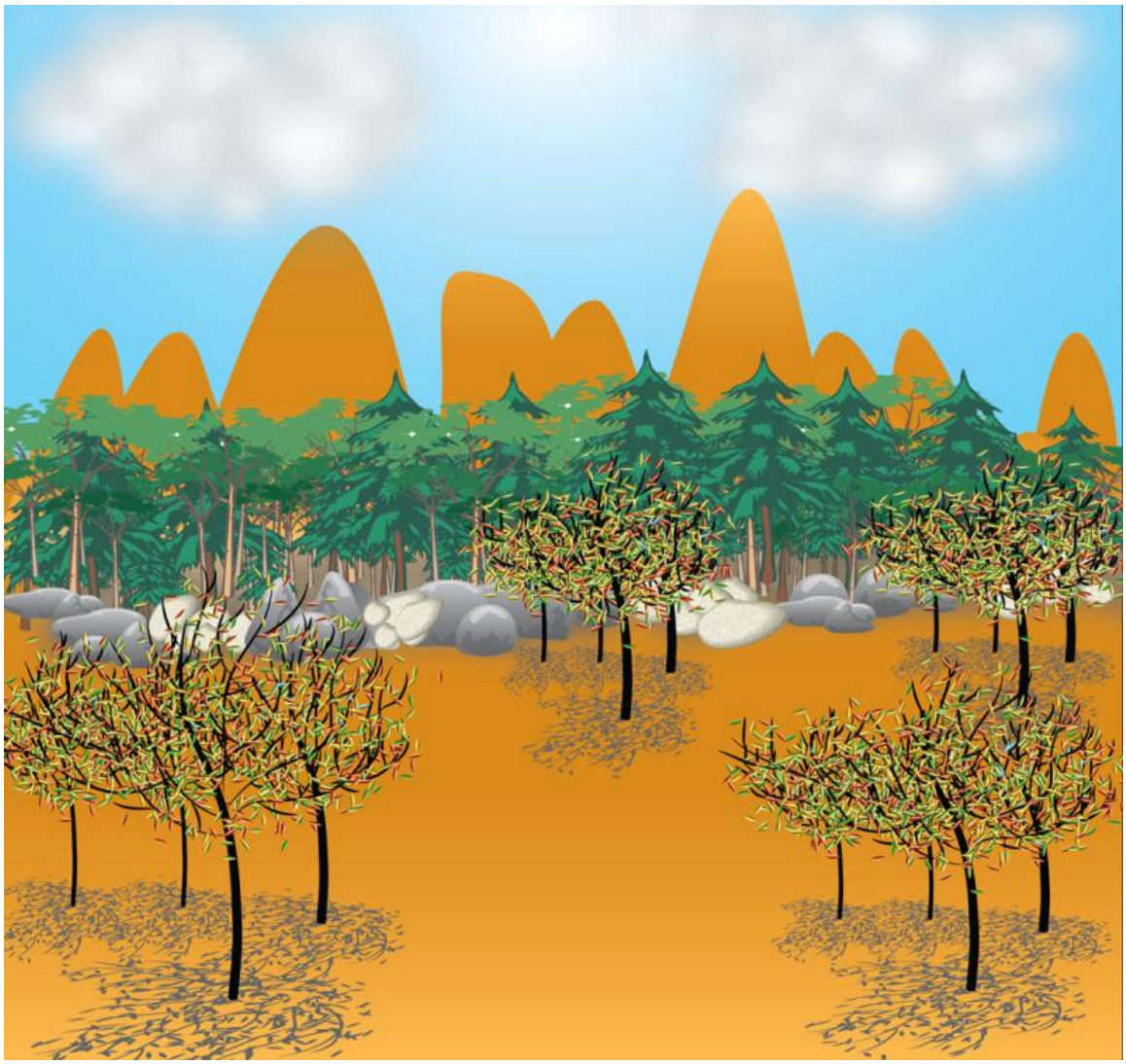


Figura 29. Caverna Zhunte.

A caverna onde se passa a maior parte do jogo.

Marcos.

Figura 30. Personagem Marcos.

Sonorização

Teremos os sons: Música de menu, Música contínua do jogo, Som de fim de jogo e o Som de resposta certa/errada.