Mummy Rush

Graziella Bonizi
Grazi.bonizi@gmail.com
Projeto Integrador – Prof º Marcelo Hashimoto

Resumo

Este documento discorre acerca do desenvolvimento e justificativa de concepção do jogo Mummy Rush, um jogo multiplayer em que o objetivo é sobreviver a *waves* que surgem através da interface gráfica do jogo. Nas seções, serão abordados os conceitos gráficos, *client-server* e implementações e resultados obtidos aplicados ao Mummy Rush, além do enfoque mercadológico que visa justificar a disseminação dessa categoria de jogos independentes, e porque, principalmente no Brasil, há tantos estúdios independentes e pessoas desvinculadas a empresas entrando nesse mercado.

Palavras-chave: Indie Game. Independente. 2D. Jogo. Multiplayer.

Introdução

Os jogos independentes, como esperado, seguem na contramão do cenário de conhecimento comum, sem o envolvimento de grandes franquias de jogos e suas numerosas equipes de desenvolvimento, sem o mesmo nível de qualidade gráfica, jogabilidade e efeitos computacionais, e geralmente com histórias bem menos complexas.

Não faltam motivos, principalmente tecnológicos, para que a categoria de jogos independentes (majoritariamente com menores recursos em todas as esferas que englobam a concepção de um jogo) fique esquecida no tempo, mas a tendência atual é que esse mercado cresça e se torne cada vez mais rentável, apoiado no conceito retrô e apelando para o sentimento nostálgico facilmente explorado em velhos *gamers* que viveram a época dos consoles de 8, 16 e 32 bits, e nos mais jovens que se identificam com a simplicidade da *pixel art* [1].

1 Justificativa relevância do tema

Nunca antes o mercado brasileiro de jogos esteve tão aquecido. No último levantamento realizado, o Brasil movimenta R$ 1 bilhão ao ano [2], com um crescimento de 60% comparado ao mesmo período do ano anterior, faturando mais que Reino Unido, Alemanha e Espanha.

Ainda sem contar com um grande estúdio de desenvolvimento de jogos, o Brasil se alimenta de jogos 2D, quando o assunto entra no campo do desenvolvimento, que são bem recebidos em diversas plataformas e dispositivos, perpetuando esta categoria de jogos e conquistando uma parcela significativa do mercado [3].

2 O Jogo

Mummy Rush é um jogo *client-server*, *multiplayer*, em que o objetivo é sobreviver as waves de inimigos que surgem pelo mapa.

A jogabilidade é simples e direta, com gráficos que remetem aos antigos jogos dos consoles das eras de 8, 16 e 32 bits, usados hoje em dia pelos entusiastas de jogos 2D como uma fuga da realidade moderna em que o jogo, primeiro, é jogado visualmente, muitas vezes priorizando mais os gráficos 3D do que a própria experiência em jogabilidade e divertimento.

O ambiente do jogo se passa dentro de uma pirâmide, em que os jogadores são arqueólogos que tentam fugir após terem roubado a Orbe Sagrada do Faraó e, como consequência,

passam a enfrentar a fúria das múmias.

As personagens poderão utilizar de armas de fogo para se defender, podendo fazer upgrades conforme avançam no jogo.

Mummy Rush um jogo 2D com visão de perspectiva isométrica (vista de cima, de um ângulo que geralmente tem o valor de 45 graus), com controles de movimento via teclas AWSD e mira pelas teclas de seta. O sistema é baseado no modo cooperativo e os jogadores devem se ajudar para avançarem pelos níveis. Será essencial que os jogadores trabalhem em equipe para alcançarem níveis mais avançados e *waves* mais difíceis.

O estilo de jogo foi escolhido devido ao sucesso de jogos *indies* (independentes) semelhantes, por ser simples de implementar e por agradar, ao trazer o tema retrô nos dias de hoje.

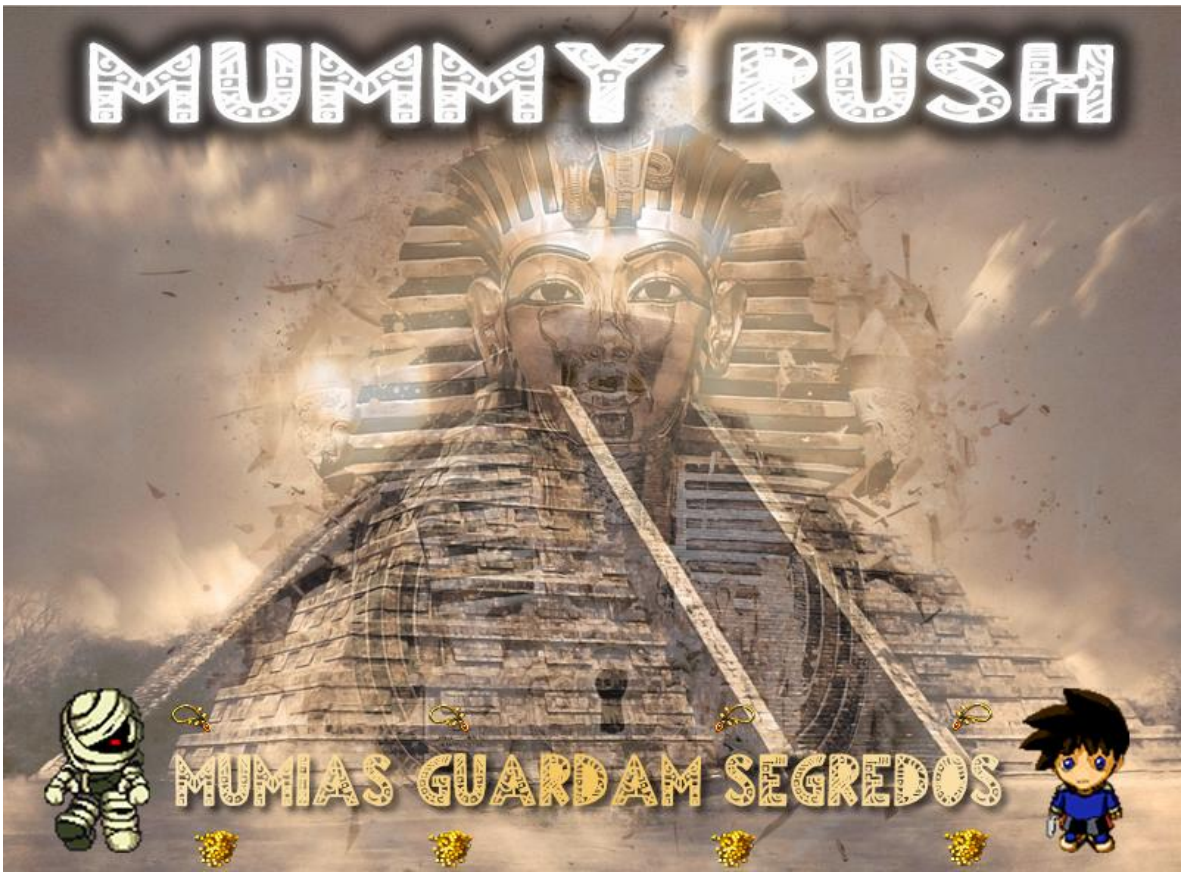

Figura 1 – Tela Inicial de Mummy Rush

O conceito visual par criação das personagens e elementos diversos envolvidos no jogo se baseia na *pixel art*, até hoje difundida no mundo *indie* dos jogos.

3 Mercado de jogos brasileiro

3.1 Brasil em números

O mercado brasileiro, tanto para desenvolvimento, quanto para consumo, está em alto crescimento, diferente de outros grandes mercados de games nos Estados Unidos, Europa e Japão, que mesmo maiores, não apresentam o mesmo desempenho no crescimento, podendo já no próximo ano, ser considerado a 3ª maior potência desse nicho de mercado.

O mercado de consumo é majoritariamente de importação, até mesmo de jogos brasileiros, pois muitos deles possuem uma *publisher* estrangeira responsável por sua distribuição.

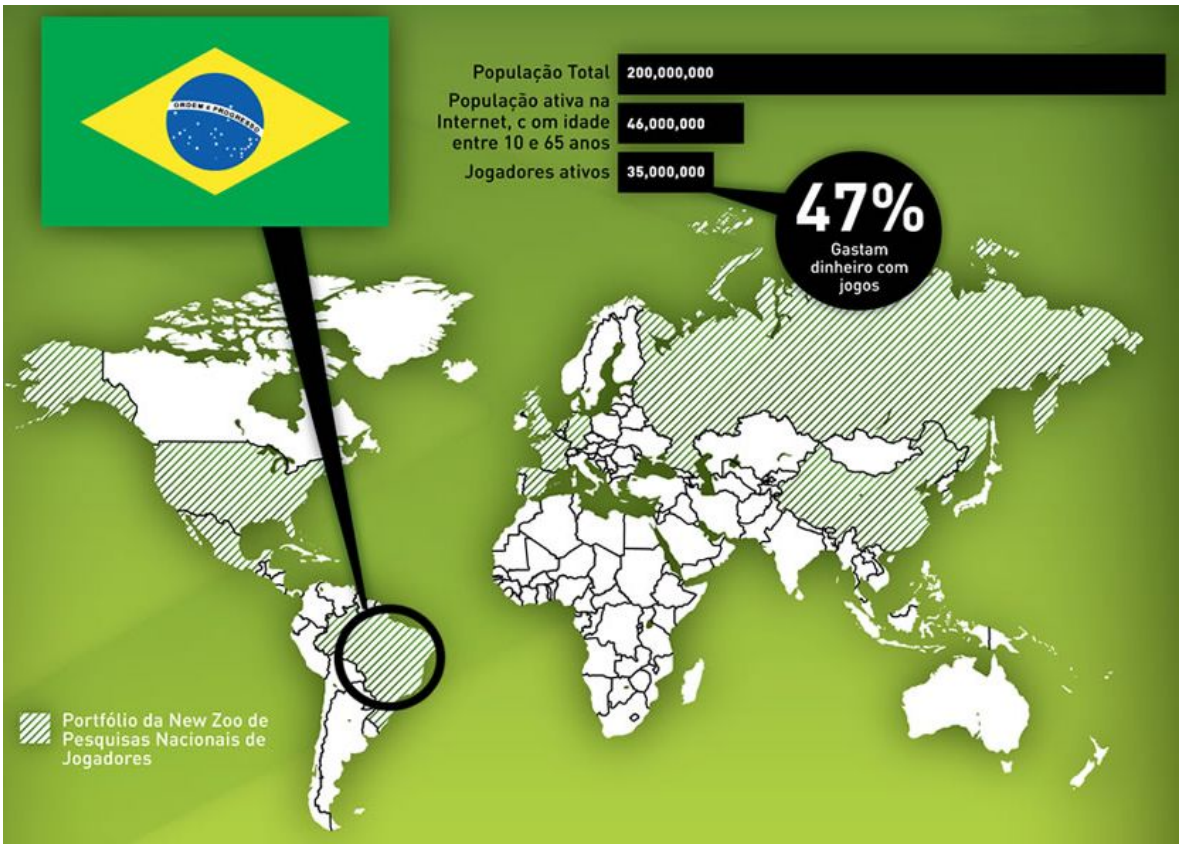

Figura 2 - Números do Mercado Brasileiro de jogos
Fonte – Joga Brasil [4]

3.2 Jogos Relacionados

Mesmo muito distante do *mainstream* das grandes superproduções de jogos e dos lucros milionários com vendas, é possível atacar uma boa parcela do aquecido mercado brasileiro com jogos simples [5] - no caso, a simplicidade principal exposta é principalmente, a gráfica.

O primeiro exemplo é o jogo "Roko Loko no Castelo do Ratozinger" [6], que vendeu mais de 400 mil cópias, tendo como pano-de-fundo uma história bem simples, onde um rockeiro tem o objetivo de salvar sua namorada de um castelo habitado por um Ratão – maquiando e trazendo para si o contexto de uma história bem famosa por todos *gamers* – e provando que uma boa ação de marketing muitas vezes é mais importante que a própria dificuldade em se desenvolver o jogo.

Figura 3 *– Roko Loko no Castelo do Ratozinger*

Outro exemplo é o também brasileiro “Oniken” [7], inspirado nos jogos de 8 bits que dominavam a década de 80. Onde a história se passa em um futuro pós-apocalíptico em que a raça humana havia sido exterminada, exceto pela base militar ONIKEN. No site do desenvolvedor, é possível baixar o jogo para os três principais sistemas operacionais, além de poder contribuir com uma quantia simbólica.

Figura 4 – *O jogo brasileiro Oniken*

4 Protótipo

Segue abaixo uma imagem do protótipo do jogo:

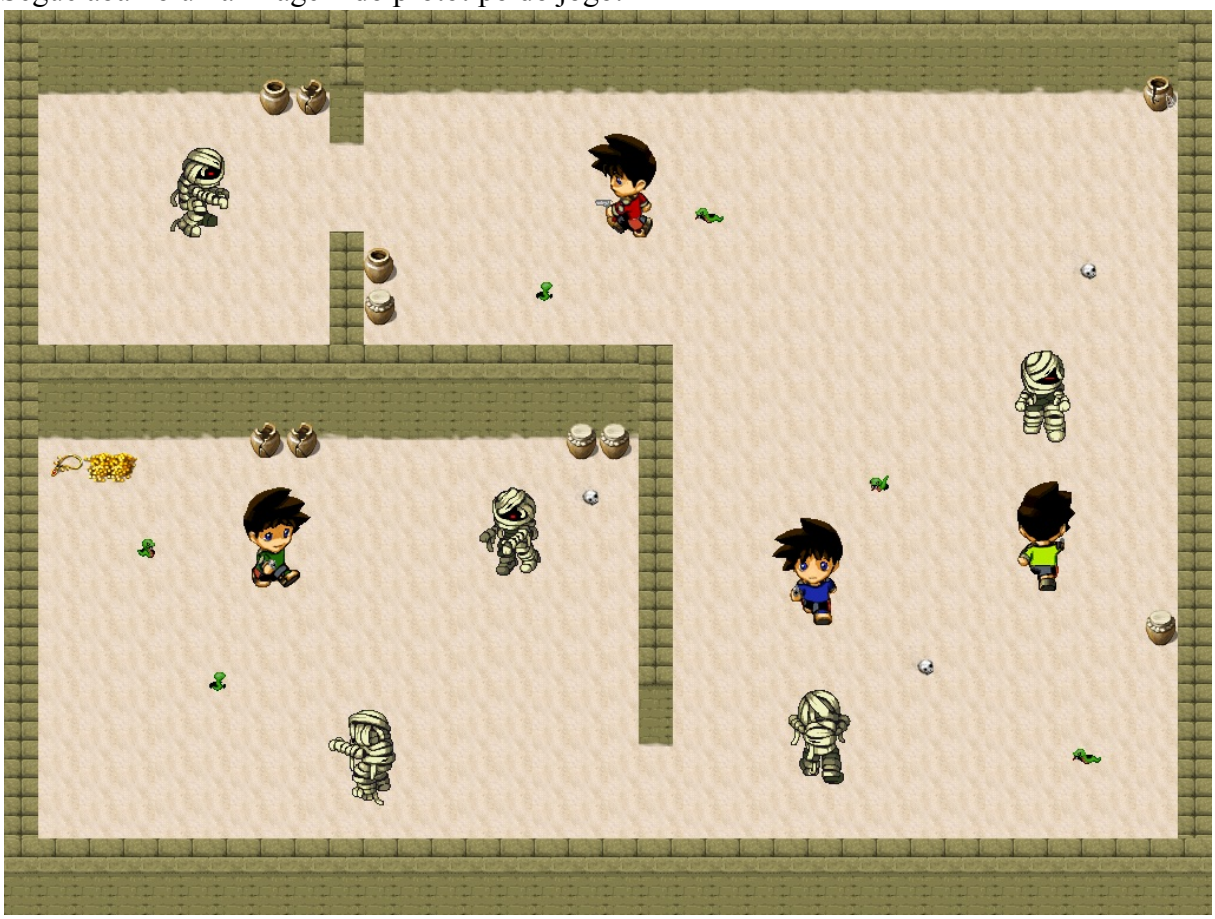

Figura 5 – *Protótipo do Mummy Rush*

O jogo tem visão de perspectiva isométrica, colaborativo com 4 jogadores, 2 tipos de monstros (cobras e múmias) e munição infinita. Para que um jogador seja eliminado basta que colida com um monstro (independente do tipo) e para que elimine um monstro basta que este seja atingido por uma quantidade pré-estipulada de tiros, variando do tipo de monstro.

O jogo é ganho quando todos os monstros são eliminados, o jogo é perdido quando todos os jogadores são eliminados.

Numa próxima etapa poderiam ser implementadas outras fases, com outros mapas e maior número e resistência de monstros.

5 Arquitetura Prevista

Desenvolvimento em C, com ambiente Xubuntu 32 bits e utilização do Allegro para interface gráfica. Utiliza estrutura cliente - servidor com comunicação via sockets permitindo múltiplos clientes. Os arquivos foram separados da seguinte maneira:

5.1 Listas

Possui as definições das *structs* de Jogador, Tiro e Monstro e as listas ligadas de cada um.

5.2 Cliente

Executado nas máquinas dos usuários. Responsável pela exibição da interface gráfica, conexão e comunicação com servidor e captação de eventos de tecla acionados pelo usuário. Tem o papel de solicitar ao servidor a inclusão, movimento e disparo do jogador e atualizar a tela com as solicitações do servidor.

O conceito básico de funcionamento do Cliente é resumido no algoritmo a seguir:

1. Inicializa os componentes do Allegro;
2. Conecta ao servidor;
3. Recebe requisições do servidor na seguinte ordem:
   - 3.1. Recebe o Id e a posição inicial de cada jogador, inclui o jogador na lista ligada e inicializa seu bitmap;
   - 3.2. Recebe as informações de cada monstro, inclui o monstro na lista ligada e inicializa seu bitmap;
   - 3.3. Recebe qual o jogador do usuário;
   - 3.4. Recebe solicitação de início de jogo;
4. Exibe o mapa;
5. Inicializa a thread de tratamento de eventos do Allegro:
   - 5.1. Capta as teclas (AWDS) e verifica se houve movimento:
     - 5.1.1. Se a posição após o movimento for válida, solicita ao servidor essa nova posição para o jogador.
   - 5.2. Capta as teclas (setas) e verifica se houve disparo:
     - 5.2.1. Se sim, envia a solicitação de disparo ao servidor.
   - 5.3. Atualiza a tela com as posições de cada célula ativa nas listas de jogadores, tiros e monstros.
6. Inicializa a thread de leitura de mensagens do servidor:
   - 6.1. Capta o tipo de mensagem - se é referente ao jogo, jogador, tiro ou monstro:
   - 6.2. Se for sobre o jogador, verifica o tipo de ação:
     - 6.2.1. Se for movimento, atualiza a posição do jogador solicitado;
     - 6.2.2. Se for remoção, desativa o personagem
   - 6.3. Se for sobre o tiro, verifica o tipo de ação:
     - 6.3.1. Se for inclusão, inclui o tiro na lista ligada e inicializa seu bitmap;
     - 6.3.2. Se for movimento, atualiza a posição do tiro para a requisitada;

6.3.3. Se for remoção, desativa o tiro;
6.4. Se for sobre o monstro, verifica o tipo de ação:
6.4.1. Se for movimento, atualiza a posição do monstro para a requisitada;
6.4.2. Se for remoção, desativa o monstro;
6.5. Se for sobre o jogo, verifica o tipo de ação:
6.5.1. Se for jogo perdido, exibe a mensagem "Game Over" e encerra a thread;
6.5.2. Se for jogo ganho, exibe a tela de jogo ganho, o ranking de cada jogador e encerra a thread;
7. Une as threads;
8. Desconecta;
9. Finaliza os componentes;
10. Encerra.

5.3 Servidor

Responsável pelo gerenciamento de conexões, recebimento de requisições e envio de informações aos clientes, além do gerenciamento do jogo no geral.

O conceito básico de funcionamento do Servidor é resumido no algoritmo a seguir:

1. Inicializa os componentes do socket;
2. Inicializa o Allegro para utilizar as verificações de colisão posteriormente;
3. A cada conexão de cliente, inclui o jogador na lista ligada com sua posição inicial;
4. Inclui um número pré estipulado de monstros na lista ligada, com a posição inicial, o tipo, a "vida" inicial e o jogador que o monstro irá "perseguir".
5. Após o número estipulado de clientes ter se conectado:
5.1. Envia as informações dos jogadores para cada um dos clientes;
5.2. Envia as informações dos monstros para cada um dos clientes;
5.3. Envia para cada cliente qual o jogador correspondente;
5.4. Envia para cada cliente o sinal para iniciar o jogo.
6. Inicializa, para cada cliente, uma thread com o seguinte escopo:
7. Recebe mensagens do cliente e verifica o tipo:
7.1. Se for Jogador, verifica o tipo de ação:
7.1.1. Se for movimento, atualiza a posição da célula de jogador e envia a informação de movimento de jogador a todos os clientes;
7.2. Se for Tiro, verifica o tipo de ação:
7.2.1. Se for disparo, inclui o tiro na posição solicitada e envia a informação de inclusão de tiro a todos os clientes;
8. Inicializa a thread de tiros:
8.1. Para cada tiro ativo na lista:
8.1.1. Move para a próxima posição (MUV considerando o ângulo do disparo);
8.1.2. Se houver colisão com paredes, desativa o tiro e replica a informação de remoção de tiro a todos os clientes;
8.1.3. Se houver colisão com monstro:
8.1.3.1. Desativa o tiro;
8.1.3.2. Diminui a "vida" do monstro. Verifica se a vida atingiu zero:
8.1.3.2.1. Se sim, desativa o monstro e aumenta o ranking do jogador que disparou o tiro. Envia a requisição de remoção de monstro a todos os clientes. Verifica se o jogo foi ganho, isto é, se todos os monstros foram desativados:
8.1.3.2.1.1. Se sim, envia a informação de jogo ganho e o

ranking de cada jogador a todos os clientes. Encerra a thread.

8.1.4. Se não houver colisão, envia a informação de movimento de tiro para todos os clientes.

9. Inicializa a thread de monstros:
   9.1. Para cada monstro na lista:
      9.1.1. Move para a próxima posição segundo o algoritmo A* [8] e verifica se houve colisão com jogador:
         9.1.1.1. Se sim, desativa o jogador e envia a informação de remoção de jogador a todos os clientes. Verifica se o jogo foi perdido, isto é, se todos os jogadores foram desativados:
            9.1.1.1.1. Se sim, envia a informação de jogo perdido a todos os clientes. Encerra a thread.
         9.1.1.2. Se não houver colisão, envia a informação de movimento de monstro a todos os clientes.
10. Une as threads;
11. Finaliza os componentes;
12. Encerra.

5.4 Interface

Responsável pelos métodos e atributos básicos do Allegro.

5.5 Colisão

Responsável por ler uma imagem do mapa e transformá-la numa matriz de 0's e 1's, sendo 0 uma coordenada livre e 1 uma coordenada bloqueada. Utilizado tanto pelo Cliente quanto pelo Servidor para verificação de colisão com paredes.

5.6 Util

Possui especificação de *enums* e métodos comuns tanto ao cliente como ao servidor, além do protocolo de serialização de informações como Jogador e Tiro.

6 Previsto X Realizado

- Estrutura cliente - servidor com múltiplos clientes: Ok
- Paralelismo: Ok
- Arte: Ok
- Listas ligadas: Parcial
- Protocolo de comunicação entre o cliente servidor: Ok
- Controle de jogador (inclusão, movimento, remoção, ranking): Parcial
- Controle de tiro (inclusão, movimento, remoção): Ok
- Controle de monstro (inclusão, movimento, remoção): Pendente
- Controle de jogo ganho: Pendente
- Controle de jogo perdido: Pendente
- Refinamento de alocação/liberação de memória: Pendente
- Otimização do código: Pendente

7 Considerações Finais

Jogos que fomentam o conceito 2D são ainda hoje muito bem aceitos pelos *gamers* das

mais diversas faixas etárias, e consequentemente são capazes de angariar bons lucros para os participantes desse tipo de projeto, que geralmente se reúnem em pequenos grupos e até mesmo individualmente para desenvolvê-los.

Aplicar os conceitos de *client-server* utilizando-se a linguagem de programação C, desenvolver programaticamente a dinâmica de tiros, colisões e movimentos, e integração da interface gráfica utilizando a biblioteca gráfica Allegro foram responsáveis por uma pesquisa maciça em diferentes campos da tecnologia e etapas da concepção de um jogo, contribuindo com o desenvolvimento pessoal e profissional do aluno.

8 Referências

1. Adele Goldberg and Robert Flegal, "ACM president's letter: Pixel Art", *Communications of the ACM*, Vol. 25, Issue 12, Dec. 1982.
2. UNG. **Mercado de Games**. Disponível em http://www.ung.br/zf/noticia/3377-mercado-de-games-cresce-no-brasil-e-gera-empregos-na-area
3. BRASIL GAMER. **Como anda o mercado de games no Brasil**. Disponível em http://www.brasilgamer.com.br/articles/2013-09-13-como-anda-o-mercado-de-games-aqui-no-brasil
4. JOGA BRASIL. Disponível em http://www.jogabrasil.com.br/
5. GMBR. **Ganhando dinheiro com jogos.** Disponível em http://gmbr.forumeiros.com/t10998-sim-e-possivel-ganhar-dinheiro-com-jogos-2d-veja-como
6. MARCIO BARALDI. **Roko Loko no catelo do Ratozinger**. Disponível em http://www.marciobaraldi.com.br/baraldi2/jogo-do-roko.html
7. JOYMASHER. **Oniken, the game**. Disponível em http://joymasher.com/oniken/
8. STANFORD. Introductuon to A*. Disponível em http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html.